



Universidad de Granada

Departamento de Teoría de la Señal,
Telemática y Telecomunicaciones

Integración de archivos y herramientas radioastronómicas en la arquitectura del Observatorio Virtual

Juan de Dios Santander Vela
Departamento de Astronomía Extragaláctica,
Instituto de Astrofísica de Andalucía-CSIC



Tesis Doctoral



Departamento de Teoría de la Señal,
Telemática y Telecomunicaciones
Universidad de Granada

Departamento de Astronomía Extragaláctica
Instituto de Astrofísica de Andalucía - CSIC

**Integración de archivos y herramientas
radioastronómicas en la arquitectura del
Observatorio Virtual**

Memoria presentada por:

Juan de Dios Santander Vela

para optar al grado de

Doctor por la Universidad de Granada

Dirigida por:

Lourdes Verdes-Montenegro Atalaya (IAA-CSIC)
Enrique Solano Márquez (LAEX-CAB/INTA-CSIC)

Granada, 17 de Abril de 2009

Como directores de la tesis titulada *Integración de archivos y herramientas radioastronómicas en la arquitectura del Observatorio Virtual*, presentada por **D. Juan de Dios Santander Vela**,

Dña. Lourdes Verdes-Montenegro Atalaya, Doctora en Ciencias Físicas y Científico Titular del Departamento de Astronomía Extragaláctica del Instituto de Astrofísica de Andalucía (CSIC), y **D. Enrique Solano Márquez**, Doctor en Ciencias Matemáticas del Laboratorio de Astrofísica Estelar y Exoplanetas del Centro de Astrobiología (LAEX-CAB/INTA-CSIC).

DECLARAN:

Que la presente memoria, titulada *Integración de archivos y herramientas radioastronómicas en la arquitectura del Observatorio Virtual* ha sido realizada por **D. Juan de Dios Santander Vela** bajo su dirección en el Instituto de Astrofísica de Andalucía (CSIC). Esta memoria constituye la tesis que **D. Juan de Dios Santander Vela** presenta para optar al grado de **Doctor por la Universidad de Granada**.

Granada, a 17 de Abril de 2009

Fdo:
Lourdes Verdes-Montenegro Atalaya

Fdo:
Enrique Solano Márquez

Juan de Dios Santander Vela, autor de la tesis *Integración de archivos y herramientas radioastronómicas en la arquitectura del Observatorio Virtual*, autoriza a que un ejemplar de la misma quede ubicada en la Biblioteca de la Escuela Superior de Ingeniería Informática de Granada.

Fdo.: Juan de Dios Santander Vela
Granada, a 17 de Abril de 2009

A mi abuela, que nunca creyó
vivir para ver este momento.

Y a quienes siempre han
estado a mi lado, incluso
cuando menos me lo merecía:
sé que siempre, de una forma
u otra, podré contar con
vosotros pase lo que pase.

Desde que orbitaron los primeros satélites, hacía unos cincuenta años, billones y cuatrillones de impulsos de información habían estado llegando del espacio, para ser almacenados para el día en que pudieran contribuir al avance del conocimiento. Sólo una minúscula fracción de esa materia prima sería tratada; pero no había manera de decir qué observación podría desear consultar algún científico, dentro de diez, o de cincuenta, o de cien años. [. . .] Formaban parte del auténtico tesoro de la Humanidad, más valioso que todo el oro encerrado inútilmente en los sótanos de los bancos.

Arthur C. Clarke (1917-2008),
2001: *Una Odisea Espacial* (1968).

Índice general

Índice de figuras	v
Índice de cuadros	ix
Índice de listados	xi
Agradecimientos	xiii
Resumen	xix
I Introduction: Astronomy and the Virtual Observatory	1
1 Introduction	3
1.1 Technical development of astronomy	3
1.2 Astronomy data today	5
1.3 Astronomical archives: benefits and problems	6
1.4 Thesis aim	10
1.5 Thesis context	11
2 The Virtual Observatory	13
2.1 The VO: solving astronomical archival issues	13
2.2 VO architecture and philosophy	15
2.3 VO data formats	17
2.4 VO data access protocols	21
2.5 VO data models	21
2.6 VO applications	23
2.7 VO inter-application messaging	23
2.8 VO resource registry	26
2.9 The International Virtual Observatory Alliance	27
2.10 The VO from the point of view of different actors	29
2.11 Precedents to the VO	41
2.12 Comparable activities in other disciplines	42
2.13 Status of the VO	45

II	Radio astronomical archives in the VO	47
3	Introduction	49
4	Data modelling in the VO	51
4.1	Elements of a data model	52
4.2	Semantics, UCDs, UTypes and IVOA vocabularies	53
4.3	Role of data models in the VO	54
4.4	Data modelling diagrams	55
4.5	Existing IVOA data models	55
4.6	Other astronomical data modelling efforts	61
4.7	Conclusions	62
5	Radio Astronomical Data Model for Single-dish radio telescopes	65
5.1	Basic requirements of astronomical archives	65
5.2	RADAMS requirements and overview	67
5.3	Observation and ObsData	69
5.4	Characterisation	69
5.5	Target/Field	70
5.6	Provenance	71
5.7	Policy	71
5.8	Curation	72
5.9	Packaging	72
5.10	Conclusions	73
6	RADAMS: Characterising radio astronomical observations	75
6.1	ObsData and Characterisation	76
6.2	Target	94
6.3	Conclusions	98
7	RADAMS: Curation, Packaging and Policy	101
7.1	Curation	101
7.2	Policy	104
7.3	Packaging	106
7.4	Conclusions	112
8	Data Provenance	115
8.1	Provenance in e-Science	116
8.2	Provenance in astronomy and astrophysics	119
8.3	Properties of an IVOA Data Provenance proposal	122
8.4	Conclusions	123
9	RADAMS: Data provenance	125
9.1	Instrumental provenance	125
9.2	Environmental provenance	132

9.3	Processing provenance	134
9.4	Conclusions	136
III Bringing legacy tools to the VO		137
10	Legacy astronomical packages and the VO	139
10.1	VO-enabling applications	140
10.2	Inter-application messaging in the VO	144
10.3	SAMP: the Simple Application Messaging Protocol	145
10.4	Implementing SAMP into an existing application	148
10.5	Benefits of a SAMP-based API	150
10.6	Conclusions	153
11	MOVOIR: MOdular VO InteRface	157
11.1	SAMP Messages and MTypes	158
11.2	Standard SAMP message types (MTypes)	160
11.3	Creating alternative response patterns	163
11.4	Describing MType parameters	168
11.5	Providing default values and settings to modules	173
11.6	MOVOIR modules to implement	173
11.7	Salient features of the MOVOIR	176
11.8	Main issues	177
11.9	Conclusions	179
IV Thesis applications		181
12	Implementations of RADAMS-based radio astronomical archives	183
12.1	The Robledo DSS-63 archive	183
12.2	The IRAM 30m archive	192
12.3	Conclusions	211
13	Enhancing massa to support VO datasets	213
13.1	VO Downloader	214
13.2	VO Data Converter	218
13.3	ConeSearcher	221
13.4	AppleScript XML-RPC SAMP tester	223
13.5	Bringing massa into the VO	224
13.6	Conclusions	229
Conclusions and future work		231
	Future work	232
Conclusiones y trabajo futuro		235

Trabajo futuro	236
V Appendices	239
A IVOA structure	241
A.1 Working Groups	241
A.2 Interest Groups	243
A.3 Steering bodies and committees	243
A.4 Standard definition process	244
B Introduction to XML	247
B.1 Data markup	247
B.2 XML markup	248
B.3 XML validation	249
B.4 XML semantics	251
B.5 The XML Schema language	252
C VOTable format definition	253
C.1 VOTable DTD	253
C.2 VOTable XML Schema Definition	253
C.3 VOTable structure	256
D VO protocols	273
D.1 Simple ConeSearch	273
D.2 Simple Image Access Protocol	274
D.3 Simple Spectral Access Protocol	278
Bibliografía	283

Índice de figuras

1.1	Atmospheric opaqueness versus wavelength.	5
1.2	Evolution of the ESO data holdings	8
1.3	Evolution of telescope areas and CCD pixel resolution	9
2.1	Virtual Observatory architectural overview	16
2.2	VOTable example	22
2.3	IVOA member organisations	29
2.4	IVOA organisational building blocks	30
2.5	User view of the VO	32
2.6	Screenshot of the Aladin Sky Atlas	33
2.7	Screenshot of TOPCAT	34
2.8	Sequence diagram: User VO interaction	35
2.9	Application view of the VO	39
4.1	The Observation Data Model	56
4.2	Characterisation Data Model per axis	57
4.3	Accuracy class and CharacterisationAxis	58
4.4	Space and Time Coordinates data model overview	60
4.5	High-level overview of the Spectral DM	61
5.1	Data modelling layers for different kinds of VO archives	66
5.2	RADAMS general class organisation	68
6.1	ObsData class data model	77
6.2	Spatial axis frame metadata	78
6.3	Temporal axis metadata	82
6.4	Spectral axis metadata	88
6.5	Observable axis metadata	89
6.6	Target data model	97
7.1	Curation data model	102
7.2	Policy data model	105
7.3	Role determination algorithm	107
7.4	VOPack structure	111

7.5	VOPack schema listing	113
8.1	Taxonomy of data provenance techniques in e-Science	116
8.2	FITS headers showing AIPS History	121
9.1	Provenance.Instrument data model	126
9.2	Provenance.AmbientConditions data model	132
9.3	Provenance.Processing data model	134
10.1	Modules for monolithic a message-based VO-enabled applications	143
10.2	Life-cycle of a SAMP hub	146
10.3	Life-cycle of a SAMP client	147
10.4	Simplified event flow of a SAMP-enabled application	149
11.1	Format for describing MTypes.	160
11.2	<code>table.load.votable</code> MType description	161
11.3	<code>table.load.fits</code> MType description	161
11.4	<code>table.highlight.row</code> MType description	162
11.5	<code>table.select.rowList</code> MType description	162
11.6	<code>image.load.fits</code> MType description	163
11.7	<code>coord.pointAt.sky</code> MType description	163
11.8	<code>spectrum.load.ssa-generic</code> MType description	164
11.9	<code>movoir.describe.mtype</code> MType description	169
11.10	Using <code>movoir.describe.mtype</code> on the <code>table.load.votable</code> method.	170
11.11	Verbose description of the <code>table.load.fits</code> method.	171
11.12	Verbose description of the <code>table.load.fits</code> method.	172
11.13	<code>movoir.configuration.set</code> MType description	174
11.14	<code>movoir.configuration.get</code> MType description	175
12.1	DSS-63 70-meter antenna	184
12.2	High level, layered architecture for the Robledo Archive	189
12.3	Implementation of the data model for the DSS-63 archive	191
12.4	Prototype search form for the DSS-63 archive.	193
12.5	IRAM 30-meter antenna.	194
12.6	High level, layered architecture for the IRAM 30m archive	196
12.7	Implementation of the data model for the IRAM 30m archive	197
12.8	TAPAS home screen	203
12.9	TAPAS search form	204
12.10	TAPAS search results	205
12.11	TAPAS project detail page	206
12.12	TAPAS observation scans	207
12.13	TAPAS scan detail for an on-off calibration	208
12.14	TAPAS scan detail for an OTF map	209
12.15	TAPAS project sources	210
12.16	TAPAS policy	211

13.1	VO Downloader: <code>initSamp()</code>	216
13.2	VO Downloader receiving a <code>VOTable</code>	217
13.3	VO Downloader in the JSAMP Hub Monitor	217
13.4	Handler of the <code>movoir.describe.mtype</code> message.	219
13.5	Listing of <code>movoirVOT2FITSconverter.py</code>	220
13.6	Listing of <code>movoirAMIGAcone.py</code>	221
13.7	Aladin showing the <code>MOVOIR.ConeSearch</code> module	222
13.8	Example AppleScript-based SAMP application	224
13.9	<code>AS_SAMP</code> AppleScript client	225
13.10	Screen capture of <code>massa</code>	226
13.11	<code>massa</code> registered with the JSAMP Hub	227
13.12	<code>massa</code> 's data selection window	229
A.1	IVOA Document Process	245
C.1	Elements of the <code><VOTABLE></code> tag	263
C.2	Attributes of the <code><COOSYS></code> tag	265
C.3	Elements of the <code><RESOURCE></code> tag	266
C.4	Elements of the <code><TABLE></code> tag	267
C.5	Elements of the <code><DATA></code> tag	268
C.6	Elements of the <code><TABLEDATA></code> tag	268
C.7	Elements of the <code><STREAM></code> tag	269

Índice de cuadros

2.1	List of VO data discovery applications	24
2.2	List of VO data handling and manipulation applications	24
2.3	List of VO spectral analysis and SED fitting applications	25
2.4	List of VO development resources	25
2.5	List of other VO applications and portals	26
6.1	AxisFrame.Spatial metadata	80
6.2	Coverage.Spatial.Location metadata	81
6.3	Coverage.Spatial.Bounds metadata	81
6.4	Coverage.Spatial.Support metadata	81
6.5	Coverage.Spatial.Sensitivity metadata	82
6.6	Coverage.Spatial.Resolution metadata	82
6.7	Accuracy.Spatial metadata	83
6.8	AxisFrame.Temporal metadata	85
6.9	Coverage.Temporal.Location metadata	86
6.10	Coverage.Temporal.Bounds metadata	86
6.11	Coverage.Temporal.Support metadata	87
6.12	Coverage.Temporal.Resolution metadata	87
6.13	Accuracy.Temporal metadata	88
6.14	AxisFrame.Spectral metadata	90
6.15	Coverage.Spectral.Location metadata	91
6.16	Coverage.Spectral.Bounds metadata	91
6.17	Coverage.Spectral.Support metadata	91
6.18	Coverage.Spectral.Sensitivity metadata	92
6.19	Coverage.Spectral.Resolution metadata	92
6.20	SamplingPrecision.Spectral metadata	92
6.21	Accuracy.Spectral metadata	93
6.22	AxisFrame.Observable metadata	95
6.23	Coverage.Observable.Location metadata	96
6.24	Coverage.Observable.Bounds metadata	96
6.25	Coverage.Observable.Support metadata	96
6.26	Coverage.Observable.Resolution metadata	97
6.27	SamplingPrecision.Observable metadata	97

6.28 Accuracy.Observable metadata	98
7.1 Policy metadata	106
7.2 Policy related Users metadata	108
7.3 Policy related Project metadata	108
7.4 Policy related DataID metadata	109
7.5 Valid packUnit attribute values	112
8.1 Data provenance management systems' properties	120
9.1 Provenance instrument metadata	127
9.2 Instrument location metadata	128
9.3 Antenna configuration metadata	128
9.4 Feed configuration metadata	129
9.5 Beam configuration metadata	129
9.6 Receiver metadata	130
9.7 Spectrum metadata	130
9.8 Velocity metadata	131
9.9 AmbientConditions metadata	133
9.10 Opacity metadata	133
9.11 Processing Step	135
9.12 Calibration metadata	136
10.1 JSAMP message latency tests	152
11.1 SAMP data types	159
12.1 DSS-63 properties, versus other antennas	184
12.2 Antenna, receiver and spectrometer properties of DSS-63	185
12.3 Observing and switching modes of the IRAM 30m	195
C.1 Valid encodingType attributes	265
C.2 Valid encodingType attributes	271
C.3 Valid dataType attributes	272

Índice de listados

C.1	VOTable DTD	254
C.4	VOTable XML Schema	257

Agradecimientos

Terminar y entregar una tesis doctoral no es cosa sencilla. Se pasa a veces mucha angustia, porque uno llega a pensar que su trabajo no merece la pena, o que jamás lo acabará. Y aunque se trate de un esfuerzo eminentemente personal, la ayuda de otras personas es crucial, y a todos ellos quiero ofrecer mi agradecimiento más sincero.

A mis padres los primeros, por su apoyo incondicional, que me ha facilitado todo, especialmente porque ellos siempre fomentaron mi curiosidad en el sentido del término latín *curiositas*, tal y como lo recoge el Diccionario de Autoridades: *Deseo, gusto, apetencia de ver, saber y averiguar las cosas, cómo son, suceden, o han pasado*.

No deja de parecerme sintomático que, desde 1992, la primera acepción del DRAE sea *Deseo de saber o averiguar alguien lo que no le concierne*. Sin *curiositas* no hay ciencia posible.

A continuación tengo que agradecerle a Lourdes, directora de esta tesis, su gran valentía: valentía a la hora de contratar a una persona de cierta edad, con un perfil técnico-comercial, sobre cuya capacidad de integración en un grupo de investigación tenía razonables dudas; valentía a la hora de permitir que una persona que, en principio, iba a hacer un trabajo determinado, realizase a su vez una tesis doctoral bajo su supervisión; y valentía por tener siempre miras altas para sí, para el proyecto, y todos nosotros, incluso en los momentos en que podía no ser políticamente correcto. Pero por encima de eso tengo que agradecer, además, su amistad. Y también es mérito suyo que amistad y dirección de tesis nunca hayan estado en conflicto. El rigor y calidad de este texto se debe a ella en gran medida, mientras que los errores son míos sin discusión.

A mi co-director de la tesis, Enrique Solano, tengo que agradecerle que hace años se diera cuenta de que el Observatorio Virtual representaba el futuro de la astrofísica, y que se lanzase a crear el Observatorio Virtual Español (SVO, Spanish Virtual Observatory), sin cuyo apoyo me habría sido imposible completar mi formación. Además él ha confiado en mí para ser representante del SVO en diferentes órganos, y ha apoyado siempre mi trabajo desde su puesto como miembro del ejecutivo de IVOA. Y sin lugar a dudas, tengo que agradecerle sus valiosas aportaciones para completar esta tesis.

A José Francisco Gómez, el ser co-supervisor de mi trabajo de investigación tutelada, durante el cuál desarrollé la versión preliminar del RADAMS. El rigor del RADAMS se debe en buena medida a sus aportaciones.

This work would have not been possible without the work and input of many people from the different IVOA working groups: thanks to Thomas Boch, François Bonnarel, Mireille Louys (double thanks!), Alberto Micol, Pedro Osuna, and Mark Taylor, among many others.

A todo el grupo AMIGA, por ser uno de los grupos más simpáticos y acogedores en los que jamás me haya integrado. Y es que tengo cosas que agradecer a todos y cada uno de ellos:

- A Gilles Bergond, su sentido del humor, y siempre estar dispuesto para explicar lo que se le pregunte, incluso cuando yo no era más que un recién llegado.
- A Dani Espada, sus consejos sobre la tesis, su gran capacidad para visualizar y contar todo el proyecto AMIGA, y su perenne buen ánimo; he intentado inspirarme en esas dos capacidades.
- A Víctor Espigares, sus extensos conocimientos del NCS del IRAM, de bases de datos, y su talento informático, sin el cual no existiría TAPAS. Y cómo no agradecerle su ayuda con la preparación de la que a la postre fue mi exitosa entrevista de postdoc.
- A Emilio García, su guía sobre la literatura de modelos de datos de IVOA y el primer gran empujón, pero especialmente el permitirme relajarme con una de las cosas con las que más disfruto, la divulgación científica, en ese magnífico programa suyo que es *A Través del Universo*.
- A Stéphane Leon, sus invitaciones a café con churros en las que acabábamos hablando de trabajo, y también de lo que no es trabajo. Y por confiar en que un ingeniero podía convertirse en observador radioastronómico, proporcionándome uno de los momentos más felices de mi vida.
- A Ute Lisenfeld, su hospitalidad en múltiples ocasiones; y tengo que agradecerle también a su marido su paciencia en esas mismas ocasiones, y las sonrisas de sus niñas, tímidas al principio, pero con ganas de divertirse a costa de los visitantes al final.
- A Vicent Martínez, ser siempre una fuente constante de ánimo, una enciclopedia musical, que además además es capaz ponerse a cantar o tocar lo que le echen.
- A Breezy Ocaña, el no ser jamás una *galaxia aislada*, sino toda un ejemplo de *galaxia en interacción*, disparando la formación de estrellas anímicas. . . Y no hay que olvidar su capacidad de enseñarme todo tipo de bailes.

- A José Enrique Ruiz, la oportunidad tan singular de continuar una amistad de hace 18 años como si el tiempo no hubiera pasado, y sin que eso le haya impedido ser constructivamente crítico con mi trabajo
- A Pepe Sabater, su curiosidad por todo lo que tiene que ver con técnicas relacionadas con la astrofísica, y sus aportaciones al desarrollo futuro de todos los becarios, de las que también me he beneficiado.
- A Simon Verley, pese a no haber podido pasar mucho tiempo con él, todo el trabajo que ha realizado para su tesis, que hace que por comparación cualquier otra parezca trivial, y que le profese una enorme admiración.
- a Susana Sánchez, reciente incorporación a la rama e-Científica del grupo, siempre hay que agradecerle su perenne sonrisa, y también que velara por que no me molestaran con el teléfono en los últimos momentos de la tesis. ¡Viva el Palo!
- a Ana Rejón, ultimísima incorporación formal del grupo (aunque se había incorporado antes de corazón), le agradezco el cariño que le pone siempre a todo lo que hace, y a la relación con los demás miembros del equipo. Y no hay que olvidar sus conocimientos de psicología, que me han ayudado a superar los momentos de máxima tensión durante la escritura de la tesis. Danke schön!

Y cómo no agradecer al insigne Jack Sulentic sus reflexiones sobre la astrofísica en general y su relación con la vida usando como medio sus degustaciones de vinos. Long live to Giacomo Imperatore!

El resto de compañeros del IAA, becarios, post-docs, y personal científico, también han contribuido a hacer de mi estancia un paso de lo más agradable. Comencemos por los futboleros: Diego Bermejo, Daniel Cabrera, *Maligno* Cantero, *Charly* Carrasco, Darío Díaz, René Duffard, Javier Gorosabel, Carlos Gracia, Jonatan Hernández, Jorge Iglesias, José Luis Jaramillo, Martin *Mates* Jelinek, David Martín, Pablo Mellado, Daniel Reverte, Miguel Ángel Sánchez, *Wanchope* Suárez, Antonio de Ugarte, y algunos más con los que no he podido pasar tanto tiempo.

También ha habido cantidad de compañeros de risas y alegrías: Marcos Aparicio, Begoña Ascaso, A.J. Cuesta, Antonio García, Maya García, Gabriella Gilli, Inma González, Marta González, Omaira González, Luc Jamet, Yolanda Jiménez, Carolinha Kehrig, Francisco López, Vicent Martínez, Mar Roca, Cristina Rodríguez, Pepe Sabater, Walter Sabolo, Meme Sánchez, Charo Sanz (enhorabuena, mamaita). Cuando ha habido momentos no tan alegres, vosotros sabéis quiénes han estado ahí.

E interesantísimas conversaciones de café o sobremesa, con Iván Agudo, Víctor Aldaya, Emilio Alfaro, Pedro Amado, Carlos Barceló, Miguel Cerviño, Víctor Costa, Rafael Garrido, José Luis Jaramillo, Isabel Márquez, Jaime Perea,

Enrique Pérez, Pepe Vílchez, y muchos más. Mención especial merece Paco Navarro: ¡se le saluda, caballero!

Quisiera destacar a todos los compañeros que han dedicado parte de su escaso tiempo libre a realizar labores de representación del colectivo de estudiantes predoctorales: Pepe Sabater, Daniel Espada, Geli Carballo, Marcos Aparicio (enhorabuena, papaito), Antonio García, Meme Sánchez, y Marta González (espero no dejarme a nadie). Su trabajo para que no se menoscabe nuestra labor de investigación y nuestro aporte a la actividad científica del centro como colectivo es fundamental. Las Sesiones de Ciencia, Cine y Debate (CCD) del IAA son un invento vuestro del que disfruta todo el centro, y que han sido posibles gracias a Daniel Guirado, Mar Roca, Alberto Molino y Fabio Zandanel, que coordinan o han coordinado dichas actividades.

Y hablando del centro, aprovecho para mandarle un beso a María de los Ángeles Cortés, sin la cual creo sinceramente que el IAA no funcionaría ni la mitad de bien.

Una de las actividades que más me ha servido para relajarme, aunque no la he podido disfrutar todo lo que habría querido, es bailar tango. Tengo que agradecer a William y Carina sus excelentes clases; a la gente que tuvo la iniciativa de sacar adelante las milongas de jueves y domingos, todas las oportunidades de baile; y a todos mis compañeros su paciencia y consejos. Un abrazo a Aleida, Ana (todas vosotras), Breezy, Carina, Natalia, Silvia y muchas más.

He agradecido antes a Emilio García el poder participar en *A Través del Universo*. Pero el disfrute no habría sido el mismo sin Pablo Santos, Ana Tamayo, Felipe Astrologuito, el Capitán Kirk, Chewie, el Reportero Urbanita, ni el Astromático y Blanquita. Y aunque Silbia López no quiera considerarse parte del equipo, le mando un beso porque ella también es muy importante. Y a Ana Rejón y Nieves Fiestas también las cuento aquí, porque también han tenido sus *apariciones estelares*.

No me quiero olvidar de mi vida pre-científica: sin lo que aprendí mientras estaba trabajando para BK Brokers, IMPURSA, Trevenque Sistemas de Información y Nadales Libros, tampoco estaría escribiendo estos agradecimientos, siendo a estas dos últimas compañías a quienes más debo, por diferentes razones. Mi agradecimiento a Juan Ramón Olmos de Trevenque Sistemas de Información, y a Francisco Martínez de Zócalo Libros, así como a mis compañeros de Trevenque Jairo Bolívar, Alejandro Morales, José Antonio Vacas, y Antonio Díaz. Y también a alguien que tuvo una corta estancia, pero me abrió los ojos al camino de la investigación: Manuel Díez-Minguito. Compañeros menos directos, pero también memorables, fueron Rafael Maroto, Paco Martínez Liñán (qué conversaciones de sobremesa), e incluso Francisco Varo (del que aprendí mucho, incluyendo lo que no pretendió enseñarme, aunque fue incluso más útil). And my gratitude to Mark Cameron and John Weisberg, two people I enjoyed working with as we shared the same passion for detail, and for enjoying ourselves the few times we were able to meet together.

También quiero contar en esta parte a Alfonso Tejedor y Carlos Burges por prestarme una *Memoria de Acceso Aleatorio*, una voz en Internet, a veces válvula de escape, y origen de la difusión en *podcast* de *A Través del Universo*.

Una parte más formal de estos agradecimientos: tengo que reconocer el soporte del Plan Nacional de Astronomía y Astrofísica del Ministerio de Ciencia e Innovación, ya que directamente a través de sus proyectos AYA2002-03338, AYA2005-07516-C02-01 y AYA2008-06181-C02-01 he disfrutado de financiación para realizar esta tesis, e indirectamente por la financiación a la Red Temática del Observatorio Virtual Español (Spanish Virtual Observatory, AYA2008-02156, AYA2005-04286). También agradezco al CSIC la concesión de una beca del programa I3P durante 2006, y han colaborado en mi formación los proyectos con financiación europea EuroVO-DCA (RI031675), VOTech (011892-DS), y EuroVO-AIDA (RI2121104). ¡Y cómo no agradecer a la Organización Europea de Investigación Astronómica en el Hemisferio Sur (ESO) que haya valorado positivamente este trabajo, tanto como para contratarme!

Quisiera terminar dando las gracias a mis amigos de toda la vida, a los que no he podido ver tanto como quisiera por dedicarle tiempo a esta tesis: Ángel, Fermín, Hortensia, Ilu, Isi, JR, Lola (qué niña más linda es Elsa), Rafa, Raquel, *Ruly*. . . Lo único que me apena es que terminar la tesis no me va a dar mucho más tiempo para estar con vosotros. . . pero sí espero que algo más, sobre todo si decidís visitarme en aquél lugar del mundo donde finalmente acabe. Y le mando un beso fortísimo a mi hermano y mis sobrinos Alba y Álex, que no sé si llegarán a poder recibirlo.

Un último mensaje: si al lector de estos agradecimientos le parece que me lo pasé demasiado bien escribiendo esta tesis, sólo una cosa tengo que decirle: los cuatro años que he pasado en este centro han pasado volando, y mi trabajo no habría sido ni la mitad de bueno si no me lo hubiera pasado así de bien. Y si de alguien me olvidé, sentiré tanta vergüenza cuando me lo diga que espero que se pueda considerar castigo suficiente.

¡Gracias!

Resumen

El nacimiento de la astrofísica se produce cuando se pasa de la medición de los movimientos periódicos de los cuerpos celestes a la interrogación de luz por medio de la espectroscopía. Una forma más poética de decirlo sería afirmar que la astrofísica es la ciencia del análisis extremadamente cuidadoso de la luz de los cuerpos celestes.

Desde hace tiempo, ese análisis se realiza de forma digital, pero sin que exista una uniformidad entre los datos proporcionados por cada tipo de observatorio, y ni siquiera entre observatorios del mismo tipo.

Dado que la tendencia actual en la astrofísica nos dirige hacia el análisis multifrecuencia de los objetos celestes (utilizando observatorios que barren el espectro electromagnético desde las ondas de radio hasta los rayos gamma, pasando por el infrarrojo, la luz visible, los rayos ultravioleta y los rayos-X), pero cada una de esas bandas de información se obtiene con instrumentos y observatorios diferentes (por ejemplo, es imposible observar rayos-X o rayos gamma si no es desde un telescopio espacial), se hace necesario uniformar la forma de expresar información científica dentro del mundo de la astrofísica.

Además, y tal y como expresa la cita de Arthur C. Clarke que da entrada a esta tesis, es posible encontrar información que no se esperaba en los datos guardados. Sin embargo, dado que la capacidad de generación de información de los detectores astronómicos viene también dominada por la Ley de Moore, el incremento de la cantidad de información guardada es exponencial, por lo que de nuevo se hace necesario establecer un cambio en la forma de tratar los datos astrofísicos.

Necesitamos, pues, una infraestructura que permita el acceso distribuido y uniforme (tanto en protocolos de acceso, como en la descripción de la información) a los datos que pueda necesitar el astrónomo, pero también que proporcione servicios de análisis remoto que minimicen al máximo la necesidad de transferir cantidades ingentes de información entre el archivo y la estación de trabajo.

Esa infraestructura, basada en tecnología de servicios web, tecnología grid, y en la descripción de datos mediante modelos de datos basados en XML, se conoce como Observatorio Virtual, y viene desarrollándose desde 2001, y fue validada en 2002 con el desarrollo del Astrophysical Virtual Observatory (AVO), una aplicación que era capaz de mostrar imágenes que se

obtenían a partir de diferentes archivos, y de dibujar sobre esas imágenes las localizaciones de medidas y observaciones tomadas por otros observatorios.

Desde nuestro grupo de investigación se lidera el proyecto AMIGA (Análisis del Medio Interestelar de Galaxias Aisladas), que pretende realizar una caracterización estadística multifrecuencia de un conjunto de más de mil galaxias seleccionadas por un estricto criterio de aislamiento, lo que garantiza que se han visto libres de interacciones con galaxias de tamaño comparable durante el último millar de millones de años. Debido a que las propiedades que nos interesan son las del hidrógeno neutro y gases en estado molecular como H_2 o CO, las longitudes de onda de radio son de particular interés para nosotros.

El desarrollo del Observatorio Virtual, sin embargo, ha venido dominado por la zona de luz visible, que es en la que contamos con mayor experiencia, pero también en la que existía un mayor número de archivos ya disponibles.

El propósito de esta tesis, por tanto, es la de proporcionar un marco en el cual se puedan crear archivos radioastronómicos, y se puedan integrar en el Observatorio Virtual. Veremos que es necesario ampliar los modelos actualmente existentes dentro del Observatorio Virtual para poder acomodar los datos radioastronómicos, y aprovecharemos para proporcionar modelos de datos de observaciones más genéricos que los existentes.

Además, es necesario poder integrar las actuales herramientas de análisis radioastronómicas con el Observatorio Virtual. En esta tesis, desarrollamos un mecanismo para la incorporación al Observatorio Virtual tanto de aplicaciones para las que se dispone de código fuente como para aquellas que no pueden manipularse.

Dicho mecanismo de compatibilidad con el Observatorio Virtual vuelve a utilizar técnicas básicas de computación remota como XML-RPC para establecer un sistema de mensajería entre diferentes módulos basado en un modelo de publicación/subscripción, tanto de proceso de datos como de acceso al Observatorio Virtual. Se reduce así al mínimo la intervención en las aplicaciones, que sólo deben incorporar un pequeño módulo de mensajería, dependiendo del resto de módulos para el descubrimiento y manipulación de datos del Observatorio Virtual.

Como efecto secundario de esta modularidad, y la existencia de los mecanismos de publicación/subscripción, se crea un mecanismo para la creación de módulos de funcionalidad dinámicamente descubribles, y que permite la extensión de cualquier aplicación que soporte la suscripción a una serie de mensajes ya establecidos.

Por último, procedemos a validar el desarrollo de la tesis mediante el desarrollo e integración en el Observatorio Virtual de dos archivos radioastronómicos, para los radiotelescopios DSS-63 de 70m, ubicado en Robledo de Chavela (Madrid), e IRAM 30m de Pico Veleta, en Sierra Nevada (Granada), y la integración en el observatorio virtual de una herramienta para espectroscopía, *massa* (MAdrid Simple Spectral Analysis).

Part I

Introduction: Astronomy and the Virtual Observatory

Chapter 1

Introduction

1.1 Technical development of astronomy

Astronomy has always been a technology enabled science. The sky was well known to the ancient cultures who found it a source for their calendaring systems, which allowed them to predict floods, find the best time of the year for seeding, and even reinforce the status of religious ministers due to their connection to the universe. But even that early astronomy demanded accurate instruments for timekeeping, angle measurement, and building alignment to mark specific parts of the year thanks to the motion of the sun in the sky throughout the year.

It was not until the times of Galileo, the first historically acknowledged user of a telescope, that astronomy suffered another impulse. The discovery of the Galilean moons revolving around a celestial body other than the Earth put an end to the Ptolemaic illusion that everything in the sky revolved around the Earth, and helped establishing the Copernican paradigm shift, where our planet was no longer the centre of the known universe. That shift renewed the interest in astronomy, and thanks to it the orbits of planets were determined, larger and better telescopes were built to find fainter objects, and more planets and moons were found.

The next leap in astronomy was the invention of spectroscopy, together with the recognition of *fingerprints* of elements in the spectra, which for the first time allowed us to know what August Comte had thought impossible to learn: what was the constitution of *heavenly bodies*¹ [1]. Not only that, but spectroscopy can tell us what the physical conditions inside a remote part of

¹Comte wrote: *The mathematical thermology created by Fourier may tempt us to hope that, as he has estimated the temperature of the space in which we move, we may in time ascertain the mean temperature of the heavenly bodies: but I regard this order of facts as for ever excluded from our recognition. We can never learn their internal constitution, nor, in regard to some of them, how heat is absorbed by their atmosphere. We may therefore define Astronomy as the science by which we discover the laws of the geometrical and mechanical phenomena presented by the heavenly bodies.*

the universe are like. This development was so important that even astronomy changed its name, becoming astrophysics. We must also remember that for most celestial bodies our only source of information is the light² they emit, absorb, eclipse or otherwise affect. It is the careful treatment of such light, together with the always improving knowledge of the physical processes affecting it what allows us to recover from that electromagnetic radiation large amounts of information from extremely distant and dim objects.

Learning how to permanently register light was first achieved in 1826 by Nicéphore Niépce, and a more repeatable and faster process was announced by Louis Daguerre in 1839, who took himself the first photograph of the Moon during that very year [2, 3]. Just in 1843 the process is applied for the first time to the spectrum of the sun by J.W. Dapper, leading to the discovery of new lines in ultraviolet wavelengths, long before Bunsen and Kirchhoff showed that those lines were due to absorption by several atomic species. Photography made astrophysics a truly experimental science, as spectra could be now recorded and compared between observations.

Another breakthrough in astronomy came hand in hand with a new technology: the discovery of extraterrestrial radio signals by Karl Jansky in 1933, for which he tried to fix a position that seemed to be coincidental with the centre of our own galaxy [4, 5]. This opened a new window for astrophysics, as the radio sky seemed at first very different, almost unrelated to the visible sky, and many different objects were discovered, such as pulsars, quasars, galactic jets, et cetera. One of the most relevant new insights for cosmology was the discovery of the Cosmic Microwave Background radiation by Penzias and Wilson [6, 7], which backed our current *Big Bang* model of our Universe.

Then came the new windows opened when the space career started, allowing humankind, for the first time, to have observatories outside of the atmosphere, which is opaque for radiation other than radio and visible light—see figure 1.1—. We were rewarded with the discovery of strong X-ray sources marking active galaxy nuclei, supernovae, and the incredibly bright and distant Gamma-Ray Bursts. By being free of the atmosphere, the Hubble Space Telescope has provided us with the deepest view of our Universe, thanks to the repeated, accumulated exposure of the Ultra Deep Field [8].

Of course, our current observatories, both ground-based and space-borne, have only been possible after Charge Couple Devices (CCDs) started to replace photographic plates. The sensibility of CCDs (measured as their quantum efficiency, or percentage of times a photon incidence produces a measurable change in the sensing element) is much superior to that of photographic plates, allowing the detection of fainter objects. Besides, several other capabilities, specially direct electronic output and linearity in their response,

²In this thesis, we will use *light* to refer to all kinds of detectable electromagnetic radiation, from the radio band to the gamma rays, and when referring specifically to visible electromagnetic radiation, which our eyes can perceive, we will always use the *visible* adjective.

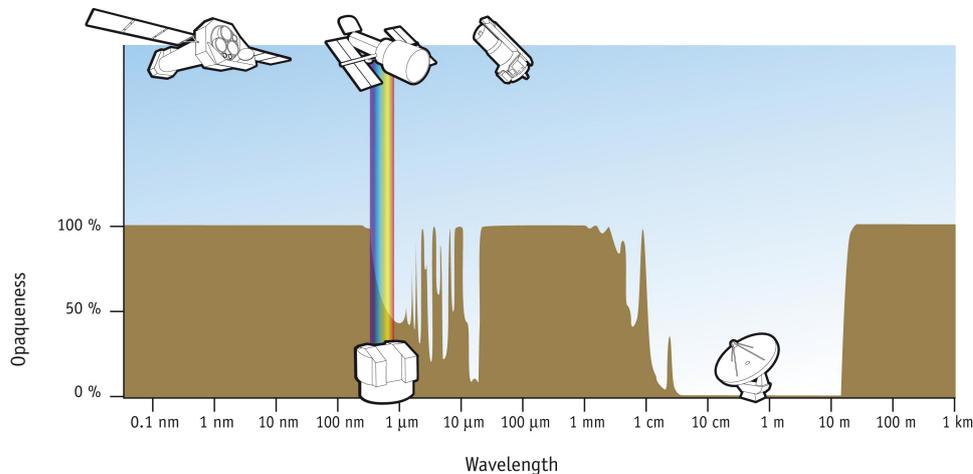


Figure 1.1: Atmospheric opacity as a function of wavelength. Earth's atmosphere is opaque to most electromagnetic radiation, except for the visible window, which reaches from the near-infrared to the near (soft) ultraviolet radiation. There is also a radio window for electromagnetic radiation with wavelengths between 1 mm and several tens of meters, and it is not totally opaque for radiation above the 1/10 of a millimetre. The main responsible for atmospheric absorption of sub-mm radiation is water vapour, and dry regions such as those in the South Pole or in the Atacama Desert can be used for observations in the sub-mm range. This image was created for NASA by STScI under Contract NAS5-26555 and for ESA by the Hubble European Space Agency Information Centre, and is in the public domain.

make them much more desirable than film for scientific, and particularly astronomical purposes.

1.2 Astronomy data today

Contemporary astronomy is built around the idea of digitised observations. Everything is quantised, digitised, and processed using digital computers, something made easier by the digital nature of CCDs.

This digital nature makes it more natural to mix datasets from different observatories, giving birth to what is called multiwavelength astronomy. By combining the light received from as many instruments as possible, we can learn an increasing number of properties of distant objects. For instance, we can try to fit spectral models to the actually recorded spectrum, being that fitting more reliable when considering the most bands.

Building that multiwavelength picture is, nevertheless, not trivial. Astronomers need to perform their own observations of the objects of their

interest with many different observatories and instruments, something very costly in terms of both time (observation proposals have to be written, and if approved then the actual observation has to be performed, processed, and analysed) and effort (spent in learning the different packages needed to reduce astronomical data from different observatories); or instead, astronomers can rely on observations previously performed by other astronomers, and stored in the archive of observatories of their interest.

In any case, very few observatories have archives, and those which have them provide datasets with very different science requirements. Some observatories provide raw data, which have to be combined with calibration data for the astronomer to perform the data reduction, while others provide already reduced data products with very little information on how that reduction was performed, which were the observing conditions, and so on. Each different archive is accessed through its own access portal, has different access policies, different data browsing mechanisms, and data are finally delivered in different formats. And there is the additional issue of finding out which archives exist which carry data of our interest.

An interesting exception to the lack of archives are space-borne observatories. These facilities are so expensive, essentially due to the launching costs, that collected data have to be made available to the community after typically one year of proprietary period in order to maximise the scientific return from their observations. The cost of the archive is a small fraction of the operational cost for the mission, and all space satellites provide some sort of access to their archives. The kind of data products provided by each mission, however, is not standardised, either.

A third source of data for the astronomer are large sky surveys, which have started to take place in the last few years, in which data are collected by dedicated wide-field telescopes, with reduction pipelines working for several spectral bands. The pipelines perform digital processing on the images, and determinate tens or hundreds of properties for selected objects. Such is the case of recent surveys, such as the Sloan Digital Sky Survey (SDSS) [9, 10], such as the Two Micron All Sky Survey (2MASS) [11], but also for the digitised version of the Second Palomar Observatory Sky Survey (POSS-II, D-POSS), or even the older, National Geographic Society-Palomar Observatory Sky Survey (NGS-POSSdigitised).

Not all surveys are optical, and there are radio surveys such as the the FIRST [12] and NVSS [13] surveys, performed with the Very Large Array radio interferometer (VLA), or the ALFALFA [14, 15], being performed with the Arecibo radio telescope.

1.3 Astronomical archives: benefits and problems

As pointed in the section before, self-performed observations are but one of the ways for astronomers to collect data relevant to their research, while data archives, be them originated from the systematic storage of observational data from each telescope and instrument, from broad sky surveys, or from space-borne missions, conform nowadays the main resource of astronomical data.

Archives provide several benefits for astronomers:

Efficiency in resource usage If the observation an astronomer wishes to perform has already been made, there is no need to go through the full process of writing observation proposals or to wait for the allocated time. The data can be downloaded by many different users, serving many different purposes, some perhaps never considered at the time of the observation.

Time domain exploration Some astronomical objects have properties varying in time. Comparing observations taken in different moments allows to study periodic phenomena (variable stars, asteroid rotation, pulsars, et cetera), or transient phenomena (novae and supernovae, Gamma-Ray Bursts, et cetera)

Statistical inference Most astronomical processes (specially those regarding extra-galactic astronomy) have time scales much larger than our civilisation life-span, and our only way to explore them is to take into account as many objects of the same type as possible. This includes defining object types, something which can be simplified by data mining techniques, which in turn require large datasets to explore.

Non-exclusive access Public archives allow astronomers from countries with limited research resources to access high-quality data, and produce top-level science.

However, in their current incarnation, archives pose several problems:

Ever increasing datasets There is no way to know when a particular observation will be useful, so all observations must be stored, and together with them all the ancillary data (weather conditions, seeing, opacity, telescope orientation, instrumental calibration observations, et cetera) needed to reduce the raw observation data in order to get scientific data products. That means every year archives have to manage more and more data. An example can be seen in figure 1.2.

Ever increasing dataset size Telescopes are built with 25 to 50 years life-spans, but instruments are changed, improved, and added, so that the

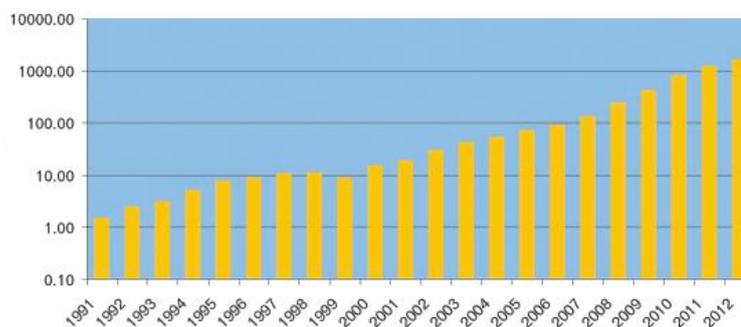


Figure 1.2: Evolution of the amount of data stored in the ESO archive since 1991 to 2003 (in Terabytes in logarithmic scale), together with predictions till 2012. Reproduced from [17].

same telescope can provide more and more resolution and sensitivity over the years, as CCDs tend to follow Moore’s Law [16], and double every two years —see figure 1.3—. That makes each dataset to be archived larger for each new facility or instrument.

Ever improving data reduction techniques The actual data being collected are in the form of voltages, or detection counts, that have to be converted in physical magnitudes such as temperature, emitted energy, mass, et cetera, which can be derived thanks to the knowledge of all the physical processes affecting the measurement, but also to a very careful measurement of observational effects. When the technical knowledge of those effects improves, many archives provide a new version of the derived data, contributing both to the data increase, but also to the need to document how each version has been obtained.

Non-uniform, non-centralised access Nowadays, most instrumental archives are accessible via internet, but each different telescope —or even each different instrument— has a different web portal, with different query parameters, which are difficult to access in an automated way. Besides, there is no way for an astronomer —much less so for a software system— to learn when a particular archive, or data set within that archive, has been released, as many observations are held for a certain *proprietary period* during which the original observer has exclusive access to it.

Thus, astronomers in the era of archives face the following problems:

- finding out and retrieving already available data from existing archives, remaining aware of additional datasets which might be released at any time;
- combining datasets from different archives in a sensible way;

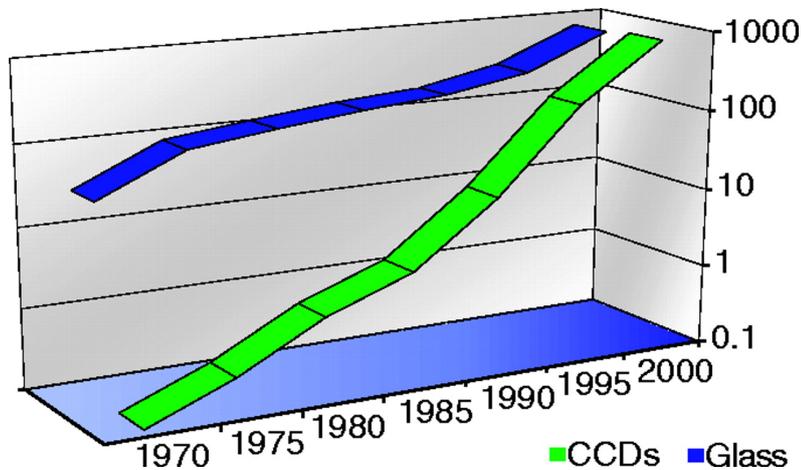


Figure 1.3: Telescope collecting surface area (labelled Glass) and CCD pixel resolution in logarithmic scale versus linear time since 1970 (arbitrary units). It can be seen that CCDs, governed by Moore's Law, have a much steeper increase rate, doubling roughly every 2 years, whereas telescope area doubles every 25 years. Reproduced from [18].

- managing huge datasets simultaneously in order to derive statistical properties;
- performing analysis of large datasets on a distributed system; this problem is compounded by the bottleneck of network bandwidth, which not increasing at the same rate as the astronomical data set size.

A suitable solution for those issues, then, should provide:

- Mechanisms for finding each and every data repository available in each moment must exist.
- Complete descriptions of each data repository, so that those not containing data of interest for the astronomer—as specified by several selection criteria, such as wavelength or resolution— can be filtered out.
- Unified data description and format, so that data coming from heterogeneous sources can be easily combined, and operated with.
- Minimum data transfer between users of archived data and the data; data processing must be moved preferentially to the server, and only processed data should be sent back to the astronomer.

Such a system would provide each astronomer in the world with a virtual observing facility able to picture the sky in all the wavelengths at the same

time, without the need for the astronomer to manually discover each of the datasets which will conform that picture, or to perform conversions on data coming from different instruments. That system exists, and is called the Virtual Observatory.

Discussions on the Virtual Observatory concept were started with the *Virtual Observatories of the future* [19] and *Mining the sky* [20] conferences, held in 2000 at Caltech and Garching bei München, respectively. The north american National Virtual Observatory project started in 2001 with a 10 million dollar funding, and 17 centres involved. Finally, Jim Gray and Alex Szalay summarised and generalised the concept in their 2001 article *The World-Wide Telescope* [18] in *Science*, and the community started prototyping the system in 2002.

However, many different *virtual observatories* can be built following the previous definition. In order to have a single Virtual Observatory where all datasets and tools are interoperable, standards need to be set and adhered to. Thus a standards sanctioning body is needed, and that is the role of the International Virtual Observatory Alliance (IVOA).

The VO is still in development, but nearing what is called the *operations stage*, where astronomers are regularly using VO tools for their research. However, in this thesis will see that there are several problems with the current incarnation of the VO, particularly in the scope of radio astronomy, and multidimensional data access.

In Spain, the national VO initiative is the Spanish Virtual Observatory (SVO), which joined the IVOA in 2004, and which is focused in promoting science with the VO in Spanish astronomy, and in providing specific tools for performing that science.

We wish to emphasise that this is the first technical thesis developed within the SVO framework.

1.4 Thesis aim

This thesis is devoted to the study of:

The VO infrastructure Which are the components of the VO, and which are the interfaces to them, with special emphasis on missing or underdeveloped blocks for radio astronomical data. This is the scope of chapter 2 of Part I.

Modelling radio astronomical data For radio astronomical data to be properly described within the VO data models are needed. Part II is devoted to the RADAMS, the data model developed for radio astronomical observations.

Bringing legacy tools to the VO As there are many man-years of experience invested in many already existing astronomical tools, we will study how to incorporate those tools into the VO ecosystem. We have developed a Modular VO Interface for Radioastronomy (MOVOIR) which provides both a GUI for accessing the VO, and tools for adapting legacy tools to use VO protocols. Part III is devoted to it.

Applied work We have used the RADAMS data model as the basis for the astronomical archives of the IRAM 30m and DSS-63 70m radio telescopes, and the MOVOIR as the basis for bringing the *massa* and *madcuba* legacy applications into the VO. We show our results in Part IV is devoted to the archives which have been built using the RADAMS.

As a result of this study, complete VO-compliant radio astronomy model has been created, two astronomical archives have been implemented, and a software tool has been developed for allowing legacy radio astronomical tools access the VO.

1.5 Thesis context

This thesis work has been developed and written within an astrophysical research project (AMIGA³, Analysis of the interstellar Medium of Isolated GALaxies) whose objective is studying a sample of isolated galaxies, with more than 1000 members. A special emphasis is given on radio observations in the centimetre, millimetre, and sub-millimetre wavelength ranges because the (sub)mm spectral band delivers fundamental information to learn about physicochemical processes in the interstellar medium (ISM). It is relevant to note that astronomy in the (sub)mm range is suffering a strong technological advance, with new astronomical facilities, such as the Sub-Millimetre Array⁴ (SMA), and the well-advanced construction of the Atacama Large Millimetre Array⁵ (ALMA).

As public data access in the radio wavelength is limited, we had resolved to contribute in the building of radio data archives, working together with the IRAM to provide an archive for the IRAM 30m antenna. In parallel, we also undertook the development of the scientific archive for spectroscopic observations of the DSS-63 70m antenna at Robledo de Chavela, part of NASA's Deeps Space Network.

Since our group does intensive analysis of 3D data at all wavelengths (in fact the current trend in spectroscopy), we also decided to collaborate in bringing existing software packages to solve the mentioned tasks, in order

³<http://amiga.iaa.csic.es/>

⁴<http://sma1.sma.hawaii.edu/>

⁵<http://almaobservatory.org/>

to make our research work more efficient. We have collaborated with Jesús Martín-Pintado's group at the Molecular and Infrared Astrophysics Department (DAMIR) of the Institute of Matter Structure, developers of the MASSA (MAdrid Single Spectrum Analysis) and madcuba (MADrid CUBe Analysis) tools⁶ for the Heterodyne Instrument for the Far Infrared⁷ (HIFI) of the soon to be launched Herschel spacecraft⁸, in order to make both tools compatible with the VO.

All the problems we need to solve are very similar to those of the astronomical community at large (save the emphasis in the radio band), namely:

- easy-to-use data look-up tools, in order to get multi-wavelength data for every object, to be retrieved from online archives;
- data combination tools, taking into account different data formats, coordinate systems, file metadata;
- physical parameter extraction tools: each different parameter must include different physics, and needs its own interface.

The community had already started to provide an information-technology-based solution: the Virtual Observatory. All the work performed for this thesis has been built within that framework, and in collaboration with the Spanish Virtual Observatory initiative.

⁶Project wiki: <http://damir.iem.csic.es/mediawiki-1.12.0/index.php/Portada>

⁷<http://www.sron.nl/divisions/lea/hifi/>

⁸<http://herschel.esac.esa.int/>

Chapter 2

The Virtual Observatory

virtual

adjective

- almost or nearly as described, but not completely or according to strict definition: *the virtual absence of border controls.*
- Computing not physically existing as such but made by software to appear to do so: *a virtual computer.*

observatory

noun

- a room or building housing an astronomical telescope or other scientific equipment for the study of natural phenomena.
- a position or building affording an extensive view.

The New Oxford American Dictionary, 2nd Edition

2.1 The Virtual Observatory: solving astronomical archival issues

In the previous chapter the current status of multi-wavelength, archive based astronomy was laid as well as the problems arising when dealing with the increasing number of data archives available to the astronomical community, and with the increasing sizes for each data unit. Additionally, these data units must be combined in order to obtain a multiwavelength view of our universe.

As the archives are already distributed across the world, the solution must be network-enabled, and must be as modular as possible, so that different data providers and astronomical tool developers can work independently, and

rely on common interfaces. A Service Oriented Architecture (SOA), where data providers create web-services, and tool developers use web-services interfaces to query them fits that description, and allows reuse of already existing and deployed technology.

It must be noted that astronomical data reduction for large instruments is a highly specialised task, and that data reduction techniques are improved when knowledge about the underlying processes (astrophysical and observational) improves. This specialisation, and the long term variability of reduction techniques, makes unfeasible the centralisation of astrophysical archives.

In any case, those services must be oriented to astrophysics, and responses must include metadata describing the peculiarities of each data set. Lastly, in order for applications to find out both existing and new services and data sets, a common service registry is needed for VO applications to find out suitable services and data sets.

Jim Gray and Alexander Szalay, in “The World-Wide Telescope” [18], were among the first to outline such a system, which is called the Virtual Observatory (VO). In that paper, they analysed the already mentioned exponential trends of instrumental data output and archived data holdings increase, and noticed the not equally growing gain in astrophysical insight as an unmistakable sign that astronomers were not being able to cope with the new data-driven situation, and needed new tools to get the most of the extremely large datasets now available to them.

An example of a widely used, very large astronomical dataset is the already mentioned SDSS. In its latest release (DR7), the SDSS consists of more than 15 TB of image data, plus more than 25 TB of ancillary data, and 18 TB of catalogue data. There is not enough bandwidth at the SDSS or at the different research institutions to transfer all the data to all researches which would like to use the SDSS. And in the future, surveying telescope such as the Large Synoptic Surveying Telescope (LSST) will produce and process around 7 TB of raw data per night.

Exploiting this ever-increasing amount of data is only feasible by scientific-case guided data selection, together with automated data mining techniques. But for that to be performed in a fully automated way, data archives and data analysis tools must become interoperable.

For achieving the interoperability we will need, as stated by F. Genova in her “Interoperability” article [21]:

- common data formats;
- common data access protocols; and
- common data models for the same type of observations, as independent as possible of the generation of the dataset, so that data from very different observatories and instruments can be combined.

In addition, as VO services are distributed across the globe, and can be deployed anywhere, anytime, one or several services registries are needed, so that

But for those common formats, protocols and data model to become truly compatible an standardisation body is needed. In the VO, that body is the International Virtual Observatory Alliance.

In the next sections we will see how the VO achieves archive interoperability, which mechanisms provides for minimising local data processing as much as possible, and what is the high level organisation of the IVOA.

2.2 VO architecture and philosophy

In Gray and Szalay's paper [18], the stated philosophy for the VO is that of an e-infrastructure¹ which makes all astronomical data in the world available for astronomers as if they were in their own desktop, without the limitations of desktop computing.

For many large datasets the user should not deal with the data directly, as the data transport time for many modern datasets is not negligible, and the data producer's infrastructure can be better suited for remote processing. In time, remote processing has to become commonplace, as it will become the only solution to let users ask questions to datasets much larger than typical workstation can manipulate, avoiding network bottlenecks at the same time.

In any case, sufficient metadata must be provided so that astronomers do not need to download data to see if they can be useful or interesting, and perform instead a suitable selection of datasets based on metadata. In particular, data quality assessment through metadata inspection and evaluation allows astronomers not to retrieve, for instance, low resolution datasets if they need precise astrometric measurements, while other astronomers interested on obtaining upper limits of an object's emissions might find them useful.

How is that vision actually implemented? Figure 2.1 portraits a simplified, all-encompassing vision of the Virtual Observatory. In that figure we see the Virtual Observatory from the lowest level supporting implementation (network cabling, routers, storage media, et cetera; what is usually known as *the iron*), to base internet protocols, and grid computing middleware, to VO services and protocols implemented on top of that infrastructure, and applications using both services and protocols to present the user with a unified interface.

It is legitimate to ask how far is the Virtual Observatory today from being completely transparent to the user. The answer to that is that there are several

¹The *e* in e-infrastructure does not stand for *electronic*, but for *enhanced*, as in *e-Science*, or *enhanced science*. The enhancement is produced by means of the massive use of networked resources, such as data grids, computation grids, distributed storage, et cetera, which conform the *e-infrastructure*.

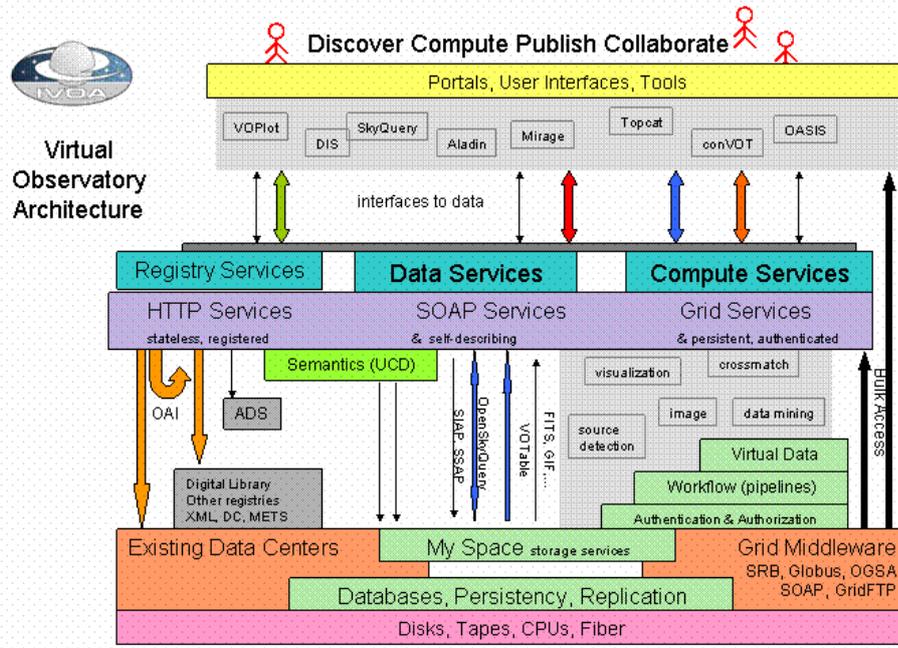


Figure 2.1: Architecture of the Virtual Observatory, seen from the low level implementation (bottom) to the user (top). Users perform high-level activities, such as computations, data discovery, data mining, and even publishing into the VO by means of applications, scripting tools, or web portals. Applications communicate with the VO by means of IVOA approved protocols to access the service Registry, Data Services to retrieve astronomical images, spectra or, in the near future, complex table access to astronomical databases. Computing Services are needed for those computations which are costly to perform locally due to bandwidth or processor requirements. Registries communicate with each other via the Open Archive Initiative (OAI) harvesting protocols, to ensure that registry changes propagate from each registry to the rest. Registries are queried through SOAP-based protocols, to ensure compatibility with other OAI registries, while the remaining VO protocols use simple HTTP GET (REST-ful) interfaces. From the IVOA Virtual Observatory Architecture Overview diagram, <http://www.ivoa.net/Documents/Notes/IVOArch/IVOArch-20040615.html>.

factors which make the Virtual Observatory still a separate environment for astronomical research:

- VO applications and portal are still unknown to many astronomical users, including some data providers. The different VO groups are making an effort in the dissemination of the VO concept, both for astronomical users and data providers. Many different workshops and schools are being promoted in order to reach users and providers.
- Many interesting datasets are still not available via the VO. The NVO and the Euro-VO projects, are developing tools to make data publishing easier for small research groups. However, they demand a high level of computing expertise in the domain of networking, web services technologies, XML, and of the inner workings of the VO. Besides, a commitment to maintain the archive operational in the long term means the research group has to keep storage and network resources with a high level of availability. This makes it difficult for those small groups to become data publishers if they do not deploy a complete archive.
- Many different useful tools for astronomers predate the Internet era, with many legacy tools unable to access Virtual Observatory datasets.

2.3 VO data formats

One of the three key aspects of interoperability, as we have seen, is data formats. If applications do not know how to operate with the files containing the data relevant to them it does not matter if data was compliant with a given data model, or if it was accessible from a common access protocol.

The FITS data format

It was radio astronomy, in particular radio interferometry, which started with the need for a common data format. Interferometric observations provide astronomical images by means of the inverse Fourier Transform of a sparsely sampled 2D Fourier expansion. As reconstruction algorithms needed expensive equipment and long processing times to provide the images, and later additional cleaning algorithms had to be run, it made sense to create a common data format which allowed for the interchange of scientific grade astronomical image (and visibilities) products, so that data could be moved to powerful enough computers. That format is the Flexible Image Transport System (FITS), created in the late seventies, and finally published in 1981 [22].

The main benefit from the FITS file format was the decoupling of actual instrument data from data about the instrument and observation setup (metadata). Data resided in image or table extensions, while metadata was

expressed in the form of ASCII headers, such as TELESCOP for specifying an observatory, or INSTRUME for specifying a particular instrument within that observatory.

However, the FITS file format, over the years, developed its own share of problems:

- Only a small core vocabulary is defined. Additional headers can be used in non-standard ways. The IAU Working Group in charge of the file format does not mandate particular keywords, or proposes best practices for FITS archival.
- Multiplicity of *de facto* per instrument and per package FITS standards: for instance, the AIPS++ radio interferometry reduction program uses a convention called FITS-IDI [23] (FITS Interferometry Data Interchange), while AIPS, its ancestor, uses the UVFITS convention. On the single dish side, IRAM uses the IMB-FITS format [24], while others use the SDFITS [25] (Single Dish FITS) convention. That means that observation metadata, such as calibration curves, tipping measurements (skydips), et cetera, could be included with the file as additional tables or not, depending on observatory, and some times depending on the instrument.
- FITS is not an appropriate file format for archival purposes. There is a flat header for all extensions, and it is physically joined to the corresponding metadata. In order to index a FITS archive, additional layers have to be applied.
- FITS files cannot be streamed on the fly from an instrument: given the fact that FITS is block oriented (due to its origin as an image *transport* format using computer tapes), it is very difficult to generate a valid FITS file from a continuous stream of data. At most, different FITS files can be created for different runs of an observation (per scan, or per integration), and then written down as a FITS file. But a truncated FITS file is very difficult to recover without manual tweaking, or to be interpreted automatically, apart from the ASCII Header part. Conversely, FITS files cannot be read sequentially, either, and a FITS file needs to be completely read (apart from the Header), in order to interpret its data.
- There is no way to combine data from different instruments, as units and scales are specified in a human readable, but not computer understandable form.

The latest *Definition of the Flexible Image Transport System (FITS)* document [26] still includes phrases referring to some features of the FITS file format as legacy, but used by the earlier packages, while new instruments

use additional mechanisms not compatible with those used by the oldest tools. It is not uncommon (or unheard of) needing to manually change FITS files, or file generation parameters, to make FITS files read from an archive, or observed from other instruments, compatible with particular tools.

With all of its shortcomings, the fact that FITS files allow for the storage of images, spectra, and other tables, with additional metadata describing those data products, has been enormously beneficial for the astronomical community, as it has made data distribution and tools development considerably easier.

Besides, the main capabilities of the FITS file format, which make it the most successful data format in astronomy to date, and is still in wide use today (to the point of being an integral part of the VO), are the following:

Flexibility At the same time FITS weakness and strength, the format flexibility² has allowed it to respond to the changing needs in astronomy, transporting data both from ground-based and space-born optical telescopes' images, radio telescope single dish spectra, maps or On-The-Fly observations, radio interferometer visibility, imaging, and data cubes, optical Fabry-Perot interferometer data cubes, Integral Field Units 3D spectroscopy, et cetera. The array of different telescopes and instruments using the FITS file format comprises the entire observational community.

Large availability of support libraries From the beginning, astronomy software developers could rely on `fitsio`³, a FITS read and write library with access to the full array of capabilities of the FITS data format. That library was written in FORTRAN, but soon a `cfitsio` library was released for C/C++ development. For Java, support is provided by the `nom.tam.fits`⁴ library, and for Python a NumPy-compatible `PyFITS`⁵ library exists.

There are a large number of archives providing FITS files, and for the VO to be successful, and have those archives provide VO compatibility, the VO must accommodate FITS files.

²FITS flexibility is based on the ability of FITS files of containing an arbitrary number of multidimensional arrays, each one with its own FITS headers for describing its content. However, it is not possible to specify whether different arrays are related in any way. For instance, if one of the arrays corresponds to a reduced spectrum, and another one corresponds to sky spectrum which has been subtracted, the only way to know it would be by manual inspection of the FITS file.

³<http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html>

⁴<http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/java/v0.9/>

⁵http://www.stsci.edu/resources/software_hardware/pyfits

The VOTable

Any new data format wishing to achieve the same diffusion as FITS needs to keep FITS' main strengths —flexibility, and availability of libraries and tools—, while addressing most of its shortcomings.

The VOTable is the main VO data format for the interchange both of tabular data, and of metadata related to any kind of astronomical data, and has the following properties:

XML data format The VOTable is an XML-based data format, and as such, it earns a wide availability of data writing and consumption libraries. XML documents can be queried through the XQuery language, or transformed in different kind of documents with XML Style-sheet Transformations (XSLTs). Plus, XML documents and processors are inherently internet-ready, providing mechanisms for linking with datasets located anywhere in the internet. Prior to the VOTable, different attempts at combining XML with FITS files had been made [27, 28], and other XML-based based file formats [29, 30] had been devised to replace FITS files.

Namespace support Stemming from its XML origin, VOTables can embed terms from different namespaces, thus allowing further flexibility and extensibility, without losing the origin and semantics of the extension.

Data and metadata separation VOTables are much more verbose than FITS files —something common to all XML-based data formats—, and typical data sizes are bigger. However, the linking mechanism allows XML-metadata to be processed and queried without having to download the actual astronomical data. What is more, particular FITS sections can be referenced from any VOTable, so that different VOTables can point to the same table of a FITS file, or the same VOTable can point to different tables of the same FITS file.

Astronomical and astrophysical semantics VOTables have astronomy specific tags, such as coordinate system definitions, but `FIELD` elements can optionally provide `units` attributes, and for the clarification of the general kind of information in each field, a `ucd` attribute —short for Universal Content Descriptor, UCD— can be used. We will see that the UCDs provide metadata with semantics which come from IVOA defined data models, and that additional `utype` attributes can be used to further specify a particular field within a particular data model.

Support for FITS file linking The VOTable can hold data by itself, but the linking mechanisms are typically used for providing internet access to FITS files.

Figure 2.2 shows an example VOTable obtained from the AMIGA VO catalog, with the metadata for the description of a four column table, with three rows returned.

The complete definition of the VOTable, in terms of its XML Document Type Definition (DTD), and an XML Schema, can be found in appendix C.

2.4 VO data access protocols

The common file format helps in the data interchange between different systems, and in letting different applications share data. However, in order to retrieve those data from the different archives existing in the VO, common data access protocols are needed.

Then main astronomical data products produced as astronomical observation facilities are the reduced image, and the reduced 1D spectra. Working with those images and spectra, different properties of objects in the universe, such as temperature, distance, diameter, or more directly, received flux, Point Spread Function (PSF), et cetera, can be derived, and positional catalogues created.

For those kinds of data the initial VO protocols to be created were the Simple Image Access Protocol [31] (SIAP) and the Simple Spectral Access Protocol [32, 33] (SSAP). For catalogues the Simple Cone Search (SCS, or ConeSearch) protocol [34] was devised, with the idea of retrieving table rows from positional catalogues centred around a certain position, and from a certain angular distance from the centre (hence the Cone in the protocol name).

Apart from data-product driven access protocols, an additional data access protocol is needed for querying the VO Registry. That protocol is based on the Open Archives Initiative

You can find a brief description of each protocol interface in appendix D for quick reference.

2.5 VO data models

With a common data format and a common data access protocol astronomers would be able to access different archives using automated tools, and the data could be sent to different applications.

However, astrophysical datasets are very complex in nature, and with the flexibility of the VOTable or the FITS file formats many different files can be constructed which contain the same astrophysical information, but in a incompatible form.

A data model is a description of the set of entities needed for information storage in a particular field, and specifies both the data being stored, and the relationships between them.

```

<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE version="1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.ivoa.net/xml/VOTable/VOTable/v1.1">
  <COOSYS ID="J2000" equinox="J2000" epoch="J2000" system="eq_FK5"/>
  <RESOURCE name="AMIGA Fundamental Physics Parameters">
    <DESCRIPTION>AMIGA Search byparameters results</DESCRIPTION>
    <INFO name="VOTable temporary file (24 hours cache expired)"
value="http://amiga.iaa.csic.es:8080/XML-CACHE/AMIGA_VOTable.09-01-09.1231514155.xml"/>
    <INFO name="QUERY_STATUS" value="OK"/>
    <INFO name="Objects Found" value="3"/>
    <TABLE name="Search Results">
      <FIELD name="CIG Number" ID="col1" datatype="char" ucd="meta.id; meta.main" arraysize="8*">
        <DESCRIPTION>CIG Number in KIG catalogue</DESCRIPTION>
        <LINK>http://nedwww.ipac.caltech.edu/cgi-bin/nph-ex_refcode?refcode=1973AISA0...8...3K</LINK>
      </FIELD>
      <FIELD name="RA J2000" ID="col2" unit="deg" datatype="float" precision="5" width="9" ref="J2000"
ucd="pos.eq.ra; meta.main">
        <DESCRIPTION>Right Ascension J2000 (Leon and Verdes-Montenegro. 2003)</DESCRIPTION>
        <LINK>http://amiga.iaa.csic.es:8080/FCKeditor/UserFiles/File/cig_ref2.ps</LINK>
      </FIELD>
      <FIELD name="DEC J2000" ID="col3" unit="deg" datatype="float" precision="4" width="7" ref="J2000"
ucd="pos.eq.dec; meta.main">
        <DESCRIPTION>Declination J2000 (Leon and Verdes-Montenegro. 2003)</DESCRIPTION>
        <LINK>http://amiga.iaa.csic.es:8080/FCKeditor/UserFiles/File/cig_ref2.ps</LINK>
      </FIELD>
      <FIELD name="Vr" ID="col4" unit="km/s" datatype="int" ucd="src.veloc.hc; meta.main" width="5">
        <DESCRIPTION>Recession Velocity (Verdes-Montenegro et al. 2005)</DESCRIPTION>
        <LINK>http://amiga.iaa.csic.es:8080/FCKeditor/UserFiles/File/aa2280-04.pdf</LINK>
      </FIELD>
      <GROUP Name="Name">
        <FIELDRef ref="col1"/>
      </GROUP>
      <GROUP Name="Coords">
        <FIELDRef ref="col2"/>
        <FIELDRef ref="col3"/>
      </GROUP>
      <GROUP Name="Velocity">
        <FIELDRef ref="col4"/>
      </GROUP>
      <DATA>
      <TABLEDATA>
        <TR>
          <TD>CIG 523</TD>
          <TD>184.17545</TD>
          <TD>69.4629</TD>
          <TD>-5</TD>
        </TR>
        <TR>
          <TD>CIG 663</TD>
          <TD>227.29241</TD>
          <TD>67.2143</TD>
          <TD>-247</TD>
        </TR>
        <TR>
          <TD>CIG 802</TD>
          <TD>260.03112</TD>
          <TD>57.9093</TD>
          <TD>-292</TD>
        </TR>
      </TABLEDATA>
      </DATA>
    </TABLE>
  </RESOURCE>
</VOTABLE>

```

Figure 2.2: Example of VOTable obtained from the AMIGA catalogue VO interface, <http://amiga.iaa.csic.es/DATABASE/>. We can see that after the initial XML declaration, and opening the <VOTABLE> tag, the <COOSYS> tag is used to specify the coordinate system equinox and epoch, and then the data is described in the <RESOURCE>/<TABLE> tag. The <FIELD> tag is used for every column in the table, the <GROUP> tag is used to group together related fields, and then the <DATA>/<TABLEDATA> tag contains the table rows between <TR> tags, and with columns separated by the <TD> tag.

For the VO, there are several kinds of data models which can be built:

A general data model for astronomical observations such a data model would be centred around the idea of astronomical observation and data reduction setup.

Data model for individual data products these data data models would exist for the different data products created with the different astronomical instruments: images, spectra, data cubes (collections of images or spectra), photometric data, et cetera.

In order to identify that a particular datum, appearing anywhere in a VO dataset, has a precise astronomical meaning the Unified Content Descriptors (UCDs) were created. They conform a controlled vocabulary whose precise meaning is set by the IVOA, and allow, for instance, to identify if a table column corresponds to flux in a particular radio band, or provides an astronomical coordinate, and so on.

We will see in Part II that the data modelling effort is still ongoing within the VO, and one of the main contributions of this thesis is a complete observation data model for radio astronomy, with some parts usable for non radio astronomical observations. More detail in UCDs will be offered there.

2.6 VO applications

VO data formats, access protocols, and data models are the infrastructure on which the rest of the VO relies. We call *VO application* to any software package of any kind which makes uses of existing VO services to perform the visualisation of data in VO format, queries of VO services, or even computations on existing datasets either in VO format, or retrieved from the VO.

However, having VO users in mind, it is best to classify applications depending on whether they allow users and/or developers to create new packages, or if they are intended to be used for scientific analysis. And in this latter case, the kind of scientific use they support.

We have provided a series of tables with a non-exhaustive list of different classes of VO applications: table 2.1 compiles VO applications for data discovery; table 2.2 combines applications for data manipulation and handling; VO applications specific for spectral analysis are shown in table 2.3; table 2.4 shows tools, libraries, and reference portals to the VO for application developers; finally, difficult to classify VO resources can be found in table 2.5.

2.7 VO inter-application messaging

The data access protocols mentioned in section 2.4 need the deployment of a full-featured web server. However, in a local machine, with several VO

Application	Description
Aladin	Interactive federated sky atlas.
NVO Datascope	Portal for positional queries to all NVO registered services.
Octet	CVO Registry Observation CaTalog Exploration Tool.
VODesktop	A resource-centered desktop client for VO: includes VOExplorer, Query and Task Runner, Astroscope, VOSpace Browser, Astro Runtime.

Table 2.1: List of VO data discovery tools and applications, from the IVOA, NVO and EuroVO TC applications and tools' lists.

Application	Description
Atasmaker	Grid software for bulk image resampling.
Mirage	Multi-dimensional visualisation of data from VOTable source files.
Montage	Science-grade custom mosaics from a portal.
NOAO VOTool	A visual VOTable authoring and editing tool.
NOAO WCSFixer	Automatic WCS correction for uploaded images.
Remote Visualisation System (RVS)	Distributed software for visualisation and analysis of remotely located astronomical images with VO support.
TOPCAT	Viewer and editor for tabular information. Based on the STILTS tool set.
Treeview	Hierarchical data format viewer with XML and VOTable support.
VisIVO	A VO-compatible visualisation tool for large datasets.
VOPlot	Tool for visualizing astronomical data from VOTable sources.
VOFilter	XML filter for OpenOffice Calc to Read/Write VOTable Files.
VOTable2XHTML	XSLT Style-sheet for exporting VOTable files to HTML.
WESIX	Web Enabled Source Identification with X-matching, portal for image upload, source extraction, and cross correlation with selected survey catalogues.

Table 2.2: List of VO data handling and manipulation applications, from the IVOA, NVO and EuroVO TC applications and tools' lists.

Application	Description
Euro3D	Spectral analysis tool for Euro3D formatted Integral Field Units (IFUs) datasets, by Igor Chilingarian.
Specview	Visualisation and analysis tool for 1-D astronomical spectrograms.
SPLAT	Spectral Analysis Tool from Starlink.
VOSA	A web-based tool developed by the SVO for automatic analysis of Spectral Energy distributions from online spectra.
VOSED	A web-based tool developed by the SVO for building Spectral Energy distributions from online spectra.
Yafit	Yet Another Fitting tool for fitting curves to data points, which can be used to fit model spectra to observed photo metric data points, by Mark Taylor.
VOSpec	A Tool to Handle VO-SSAP compliant spectra.

Table 2.3: List of VO applications for spectral analysis and spectral energy distribution (SED) fitting, from the IVOA, NVO and EuroVO TC applications and tools' lists.

Developer Tool	Description
AR Command line	Python wrapper for the AstroRuntime VO middleware.
CDS Developer's Corner	Web site for developers of the Centre de Données de Strasbourg, where references to CDS web-services and CDS Java software can be found, including unit handling.
JSAMP	JSAMP is a Java implementation of the SAMP messaging protocol written in Java by Mark Taylor.
PHP VO Client Library	PHP interface classes to ConeSearch, SIAP, SkyNode, and VORegistry services.
Python VO Client Library	Python interface classes to ConeSearch, SIAP, SkyNode, and VORegistry services.
Saada	Auto-configurable database generator and VO Service publisher for medium to small sized datasets, directly from FITS files.
SAMPy	SAMPy is a Python implementation of the SAMP messaging protocol, with additions to allow for SAMPy applications to communicate with remote computer, and not just locally. By Luigi Paioro.
SAVOT	Simple Access to VOTable, SAVOT is a Java-based library to parse VOTable documents, written by André Schaaff from the CDS.
STILTS	Command-line tools for arbitrarily large table manipulation and format conversion, including FITS and VOTable formats. By Mark Taylor.
VO-CLI	Command-line Tools for the VO, to be used with IRAF.

Table 2.4: List of VO developer utilities, libraries, and resources, from the IVOA, NVO and EuroVO TC applications and tools' lists.

Application	Description
GC Theoretical Models	Prototype tool for comparing globular cluster (GC) simulations with observed color-magnitude diagrams.
NOAO NVO Portal	VO portal providing different views for NOAO-hosted data.
Pegasus	Workflow Management on the Grid.
STC Metadata	Space-Time Coordinate metadata for the VO.
VO Services	A growing selection of VO services in production.
WESIX	<i>Web Enabled Source Identification with X-matching</i> , portal for image upload, source extraction, and cross correlation with selected survey catalogs.

Table 2.5: Other VO applications and portals not presented yet, from the IVOA, NVO and EuroVO TC applications and tools' lists.

applications running at the same time, and with all of them understanding VOTables and FITS files, a simple messaging protocol between applications would allow them to notify, or be notified, that some data is available, and if the messaging protocol supported it, several applications in communication could be showing different but interacting views of the same data.

Such a messaging protocol was prototyped and was initially known as the Platform for Astronomical Tool InterConnection (PLASTIC), and implemented in many different VO applications, such as TOPCAT, or the Aladin Sky Atlas. The shortcomings of the PLASTIC protocol gave birth to the Simple Application Messaging Protocol (SAMP), now in Proposed Recommendation stage.

The SAMP messaging protocol is described in section 10.3.

2.8 VO resource registry

The only way VO clients can remain aware of existing and newly incorporated services is by means of a registration point where public data services are announced.

In the VO, there is no centralised, preferential registry. Instead, many registries exist, but with the ability to harvest entries from other registries, so that there can exist specialised registries on one hand, and shared registries entries on the other.

VO registry and harvesting infrastructure has been built around the Open Archives Initiative⁶ (OAI) standard resource registries, so that an already in place infrastructure could be leveraged. The OAI has developed a Protocol for Metadata Harvesting (OAI-PMH) [35], that allows OAI registries to harvest other OAI registries they might know about (with an entry in their own

⁶<http://www.openarchives.org/>

registry, for instance), so that VO resources can be entered in one registry, and they should propagate to any other OAI-PMH compliant registries.

That allows VO tools to point to a single harvesting registry, and rest assured that VO servers will be found in any of those registries.

Registered resources (OAI-PMH records) provide at least the Dublin Core metadata set [36], which specifies resource types, publishers, curators, et cetera. Additionally, service-specific resource profiles exist for the different data access service types, and include astronomical data for service such as the sky region covered by the observation—footprint—, wavelength, instruments providing the data, et cetera.

In particular, the VOResource [37] XML Schema has been developed as the acceptable OAI-PMH record formats for generic VO registries, and the VODataService schema [38] is in development to further standardise data service entries.

Several VO registries provide web pages to query the registries apart from the usual OAI-PMH interface, such as the NVO Carnivore⁷, or the ESA-VO registry⁸.

2.9 Standardising the VO: the International Virtual Observatory Alliance (IVOA)

As previously stated, for VO common data formats, interfaces, and data models to become truly standard requires a standards enforcing authority, and within the VO the International Virtual Observatory Alliance (IVOA) is the organisation in charge of developing and sanctioning interoperability standards.

The International Virtual Observatory Alliance was created in 2002, after the first national VO associations realised that they needed to keep working together, and that an international standards-setting organisation was needed to sanction Virtual Observatory standards, steer the development of new ones, and foster the spread of the VO to most data providers.

The IVOA founders were the US National Virtual Observatory (NVO), the European (AVO) and supporting centres—Centre de Données astronomiques de Strasbourg (CDS), AstroGrid, German Astrophysical Virtual Observatory (GAVO)—, the Russian Virtual Observatory (RVO), VO-India, e-Astronomy Australia (eAA; later Australian Virtual Observatory, Aus-VO) and the Canadian Virtual Observatory (CVO), which in 2002 met at the ESO Headquarters in Garching, and agreed in forming the International Virtual Observatory Alliance, following the statement drafted by Peter Quinn, Robert Hanisch, and Andy Lawrence [39]. The minutes of that meeting can be found in [40],

⁷<http://nvo.caltech.edu:8080/carnivore/>

⁸<http://esavo.esa.int/registry/>

and the conference proceedings compiled in the book *Toward an International Virtual Observatory* [41].

Following that initial meeting, several services were created, and Aladin was provided with a VO interface, giving birth to the first Astrophysical Virtual Observatory (AVO) prototype [42].

The inspiration for the IVOA organisation is the World Wide Web Consortium⁹ (W3C), in the sense of providing the steering effort for standards to be agreed upon within different Working Groups (WGs), so that there is an official forum in which such standards are proposed, discussed, and finally approved. The main difference with the W3C is that the latter is *pay per play*—W3C participants pay in order to influence the standardisation process—, while the IVOA is open to any interested party with expertise in a given field.

Presently, the IVOA is formed by 16 VO projects from all over the world¹⁰, from Armenia, Australia, Canada, China, Europe, France, Germany, Hungary, India, Italy, Japan, Korea, Russia, Spain, the United Kingdom, and the United States (see figure 2.3). Membership is open to additional national and international projects¹¹.

The Spanish participation in IVOA is performed through the Spanish Virtual Observatory¹² (SVO) [43]. The seed for the SVO were the efforts of the Laboratorio de Astrofísica Espacial y Física Fundamental (Laboratory for Spatial Astrophysics and Fundamental Physics, LAEFF¹³) to build a VO-compliant archive from the IUE Newly Extracted Spectra (INES) archive¹⁴. This thesis is part of the IAA contribution to the SVO.

As mentioned earlier, the IVOA is a standards-sanctioning organisation and a steering force for the development of interoperability protocols, applications, and best practices regarding the Virtual Observatory. As seen on previous sections, there are different development areas within the VO, and there exist several Working Groups focused in each particular area. There are also Interest Groups which do not focus in actual development, but instead serve as a discussion forum for issues not directly pertaining to the Virtual Observatory, but which might eventually end up forming part of it.

The main IVOA organisational building blocks are Working Groups, Interest Groups, and Coordination and steering committees. Their detailed discussion can be found in Appendix A. Figure 2.4 shows graphically the high-level organisation of the IVOA.

⁹<http://w3c.org/>

¹⁰<http://ivoa.net/pub/members/>

¹¹ VO projects wishing to join the IVOA must follow the IVOA Guidelines for Participation <http://ivoa.net/Documents/latest/IVOAParticipation.html>.

¹²<http://svo.laeff.inta.es/>

¹³LAEFF is part of the National Institute for Aerospace Technologies, INTA

¹⁴In fact, the VO INES archive was one of the first spectroscopic archives to be available to the VO.



Figure 2.3: The 16 IVOA member organisations, as of August 2008. The Spanish Virtual Observatory (SVO) joined the IVOA in 2004. Several European countries have their own national VO initiatives (France's VO-France, Germany's GAVO, Italy's VObs.it, Spain's SVO, UK's AstroGrid), which also form part of the European Community funded Euro-VO.

IVOA Working Groups are not completely equal in scope. Most of them have to do with particular parts of the IVOA, as seen in figures 2.1 and 2.9.

2.10 The VO from the point of view of different actors

Which parts of the Virtual Observatory do users interact with? What does a particular astronomer need to know?

In the following subsections we will offer several portraits of the VO as seen from the points of view of different actors: the astronomical user, the application developer, the data provider, and the data service developer.

The VO from the user's point of view

For a non-technical user (one who wishes to use the VO via standard tools, and does not wish to access directly the VO infrastructure), the VO can be seen just as a set of portals (applications or web sites) which access all of

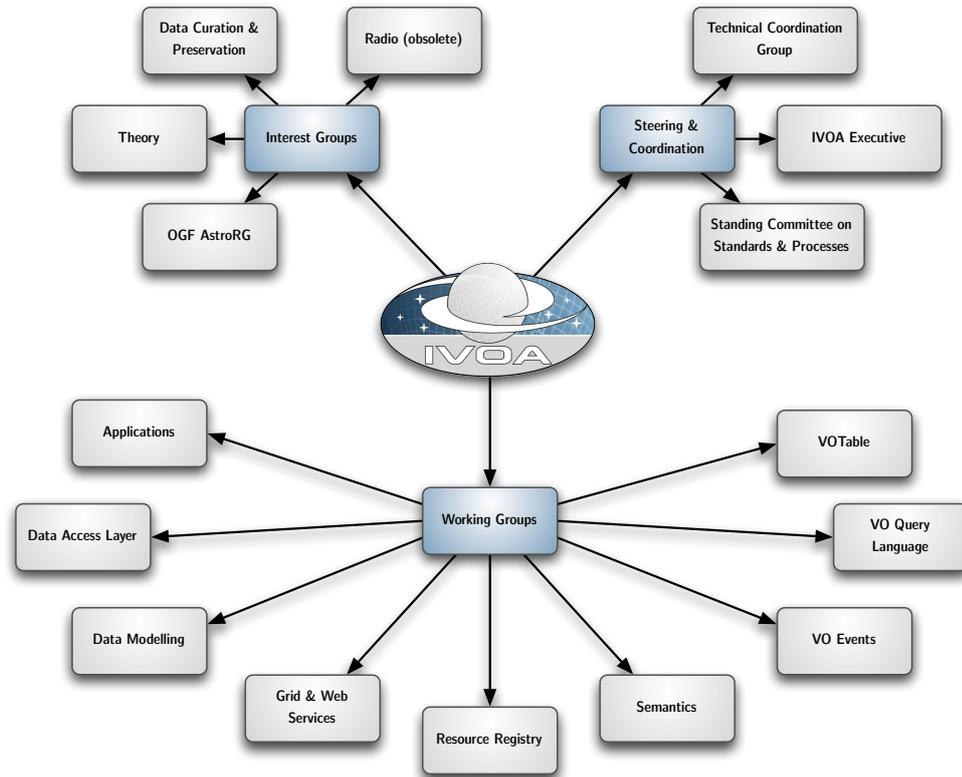


Figure 2.4: IVOA organisational building blocks: Working Groups, Interest Groups, and Coordination boards. The IVOA activity for standards development is performed in the Working Groups, while Interest Groups gather together IVOA users with common interests outside the Virtual Observatory (such as the Open Grid Forum, or the Data Curation and Preservation interest groups), or outside existing Working Groups (such as the Radio astronomy or or Theory interest groups). The Steering & Coordination bodies provide the main direction for IVOA activities, technical coordination across the existing Working Groups (i.e., Data Modelling and Data Access Layer in the development of interoperable protocols and data set access tools), and take care of possible changes to existing standards.

the services and data available in the Virtual Observatory. Some of them might also retrieve additional data from non-VO-compliant archives, and even mix them with local datasets. Users would launch different VO applications depending of the tasks they might wish to perform (Aladin Sky Atlas¹⁵ for catalogue and image queries; TOPCAT¹⁶ for catalogue and table manipulation; VOSpec¹⁷, SpecView¹⁸ or SPLAT-VO¹⁹ for spectra manipulation; VODesktop²⁰ for Registry queries, remote task execution, and access to the VOspace; or web portals such as the NVO Datascope²¹ for all-encompassing browser-based archive searches). See figure 2.5 for a conceptual diagram of the VO for astronomers.

This can be better explained by means of an example. Imagine Alice is an astronomer who has already been introduced to the VO, and that she is searching for catalogues, images, and spectra on σ Orionis, a star-forming region in the Orion belt.

She would fire up a tool such as the Aladin Sky Atlas, and use the search box to either specify the coordinates of the zone she is interested in, or would just write `sigma orionis`, and let Aladin query a name solver service, such as Sesame, to come up with the coordinates.

After having set the coordinates, Alice could set an angular radius for the search of interesting datasets, setting the separation for objects in the vicinity, or could just use the default radius. Aladin would ask different archives for both images and catalogues, laying catalogue data with coordinate information on top of the retrieved images, which can be composed from several bands.

She could, then, select catalogue data and send it to a table manipulation and plotting application, such as TOPCAT, to explore properties of the region. Figure 2.6 shows what Alice's screen could look like while working with Aladin; figure 2.7, instead, shows her computer's screen while using TOPCAT.

Another way to visualise the interaction of users with the VO is by means of a sequence diagram. Figure 2.8 shows how users do not interact with VO services, they just interact with applications to provide object names, validate coordinates, and provide additional search criteria. VO applications are responsible of communicating with the different VO services (NameSolver, Registry, and data Services) on behalf of the user.

For this user, the VO is just a set of software packages which can retrieve information from VO-enabled archives, and perform operations on such data, and on local data. The main differences with the way she used to work are:

¹⁵<http://aladin.u-strasbg.fr/>

¹⁶<http://www.star.bris.ac.uk/~mbt/topcat/>

¹⁷<http://esavo.esa.int/vospec/>

¹⁸http://www.stsci.edu/resources/software_hardware/specview

¹⁹<http://www.astrogrid.org/wiki/Install/Downloads>

²⁰<http://www.astrogrid.org/wiki/Install/Downloads>

²¹<http://heasarc.gsfc.nasa.gov/vo/>

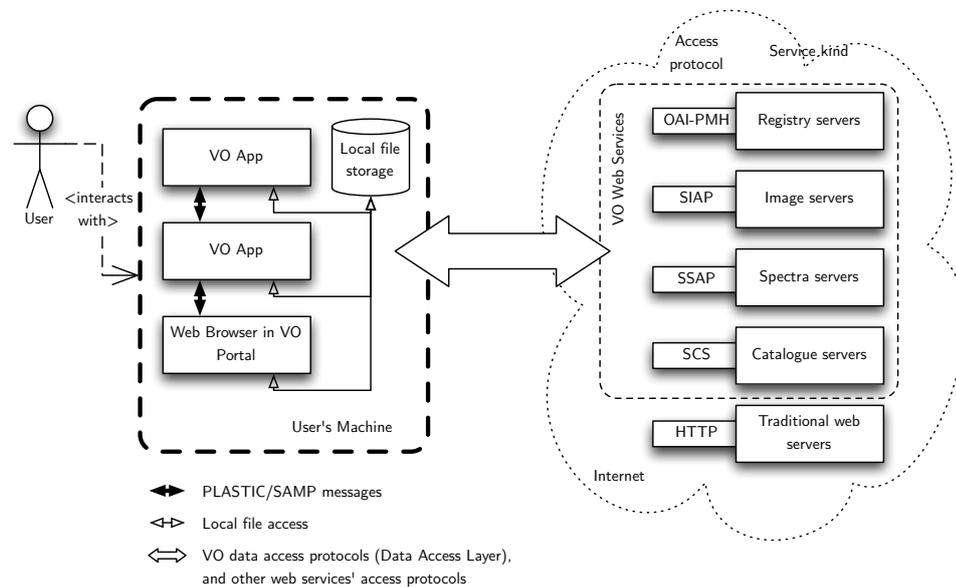


Figure 2.5: High level VO architecture from the user's point of view. A user interacts with the VO via a variety of VO-aware, locally run applications, and accesses either locally archived files, or remote files, either via VO protocols or using a web browser to access a VO-enabled web portal. Users must only be aware of the different VO applications —VO app in the figure— and/or VO portals of their interest, and that there exist interoperable image, spectra, and catalogue servers, transparently accessed from their toolset. They must also be aware that some VO applications can send messages and data between them, sending for instance data for analysis to some applications, and sending the received results to other applications for plotting. All the VO systems in the Internet —dotted cloud— are indispensable for the operation of the VO, but are completely transparent to the user.

- She uses spatial indexing on the data. Spatial selection goes first, archive, instrument, and wavelength selection are optionally performed in the next stage.
- She does not have to know how many different archives might be of interest to her: within the VO, all archives containing data in the spatial zone of her interest will be queried. She can decide which data to finally download by looking at the archive metadata (to see if the wavelength range is appropriate or not, or who is responsible for the data quality), or at the particular dataset metadata (to learn specifics such as exposure time, observing configuration, and data quality flags).
- She can combine the strengths of different tools by means of the VO

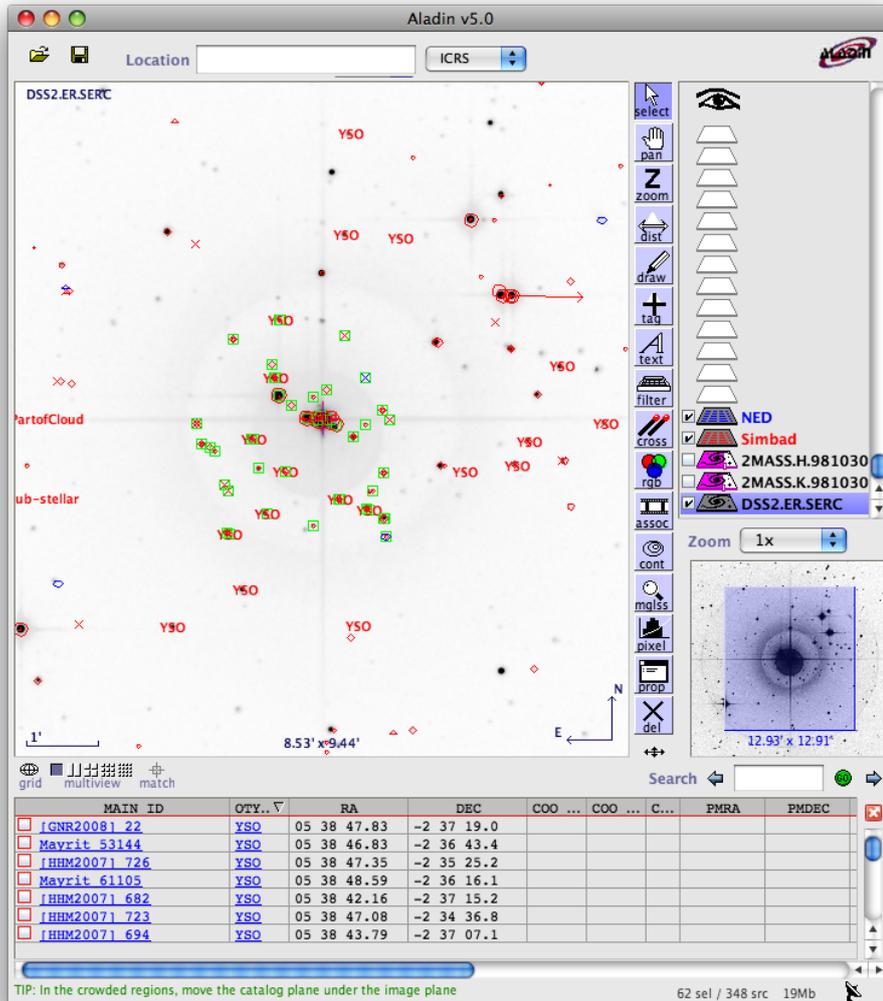


Figure 2.6: Screenshot of the Aladin Sky Atlas: after having entered the coordinates for the σ Orionis region in the search box on top of the window, Aladin has queried the Simbad and NED catalogues, together with the DSS2 server, and has overlaid catalogue objects on top of the DSS2 image. Besides, the astronomer has retrieved H-band and K-band images from the 2MASS survey, and has composed the three images by setting image transparencies. Besides, some data points of the central region have been selected, and are shown in the lower part of the window, sorted by the OTYPE (object type) column.

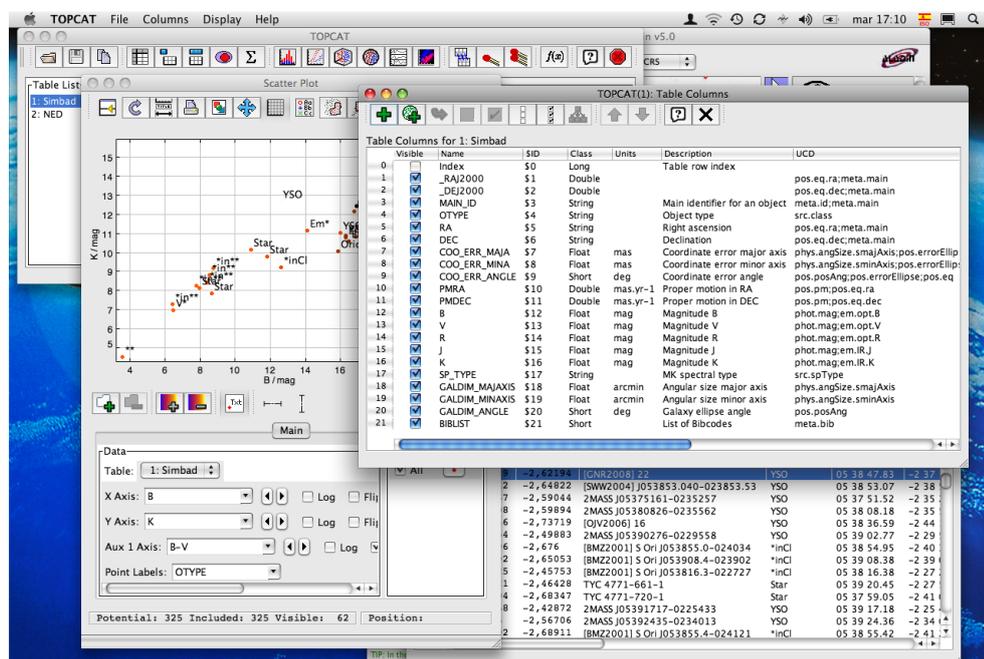


Figure 2.7: Screenshot of TOPCAT after having received two catalogues from Aladin via the PLASTIC VO messaging protocol. The windows shown, from left to right, and from top to bottom, are the main TOPCAT window, showing the two tables received from Aladin, which can be seen popping out to the right; a scatter plot of the B magnitude against the K magnitude, using the $B - V$ color index for data point coloring, and with $OTYPE$ as object label; the table column metadata, showing data types, units, description, and the UCD for of each column; and the actual table data.

messaging protocols.

- She can explore catalogues with higher confidence, because data columns are fully tagged with standard descriptors (UCDs, Unified Content Descriptors), which allow automated identification of those datasets.

In other words, she has transparent access to many archives not just from a single tool, but from any tool which is able to query the VO. In *marketing* terms, any tool sporting a *VO-compatible* badge.

However, if she wants to exploit the data she has just collected, for instance in order to perform cross-matching between sources, or to create different color diagrams, she needs to know how to find which data columns correspond to different astronomical or astrophysical concepts. In other words, she is concerned with data semantics.

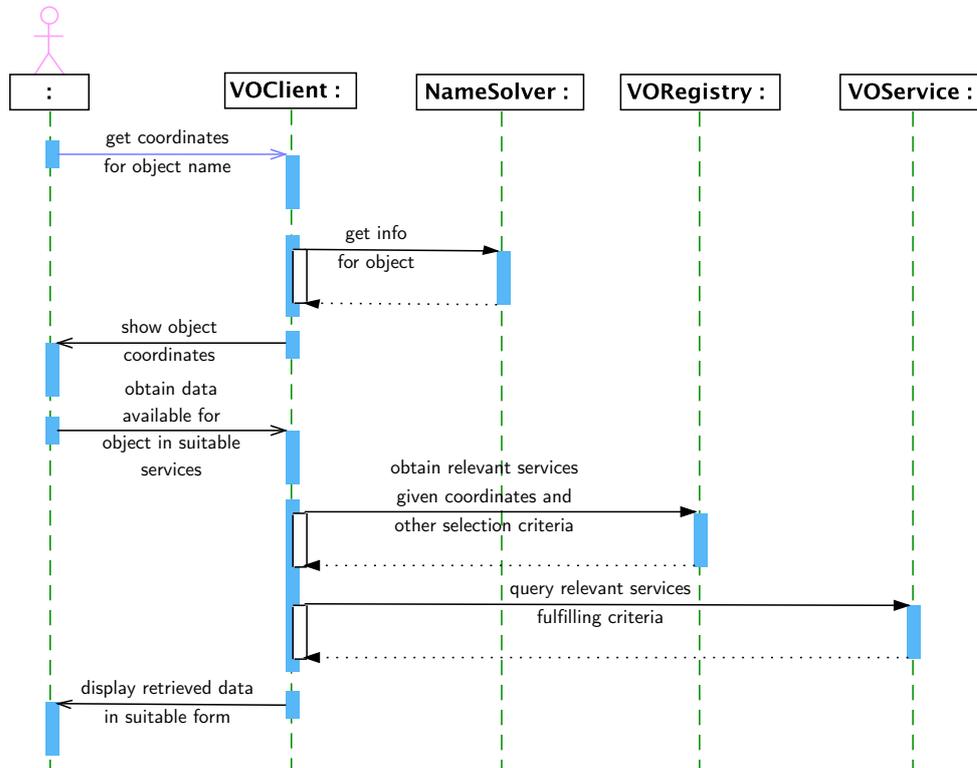


Figure 2.8: UML sequence diagram of the user interaction with the VO. Users will typically use a VO client application to get information on particular objects of their interest. The user will first provide an object name, and the VO client will use an astronomical NameSolver to return celestial coordinates. With the coordinates, and some additional criteria, which might come from the application (a spectral analysis application will ask for spectral services; an image processing or manipulation package will ask for image services), or from additional criteria (wavelength, data quality, for instance) set by the user. Once the services are found, the application will query them (here only one such service query is shown), and the retrieved data will finally be provided to the user in a suitable form. This sequence diagram shows how user interaction with the VO is restricted to the use of VO application(s), while the orchestration of VO queries corresponds to the application.

The VO from the application developer's point of view

VO Developers view of the VO arises from what the VO offers for their applications. A VO-enabled application, be it brand new or upgraded, can be of two types: data centric or workflow centric. Some data centric applications can be scripted to be part of a workflow.

Data centric applications tend to be either data reduction applications or high level analysis applications which work on individual datasets at a time. For those applications, the VO is used for:

Finding out interesting datasets That usually means finding archives with data in the desired wavelength range for certain positions of the sky.

Retrieving candidate datasets Candidate datasets (images, spectra, data cubes) are downloaded one by one, or as a package, to be individually analysed in the local computer.

Local processing One by one processing of the different datasets in order to obtain scientific grade data products, such as chemical abundance, region temperatures, ionisation states, stellar population age, et cetera. Anything that is obtained from the treatment of the light and the knowledge of the physical process at hand.

In this view of the VO, only a few benefits are added over the traditional astronomer's workflow, such as providing a single programmatic interface for all application to access all archives. That is, the VO provides a uniform data and metadata access, together with uniform data and metadata format to applications. Additionally, tools which implement VO messaging protocols allow the astronomer to use specialised tools for each task, while maintaining always the benefits of having each application connected to the VO.

On the other hand, workflow centric applications are applications which work with large datasets in large batches. For instance, from object types plus magnitudes in the 5 SDSS bands, photometric redshifts can be established²². Or data from different catalogues at different wavelengths can be cross-matched in order to find objects with particular properties which cannot be found through the original data of each individual dataset²³.

²²In fact, such processing is performed by the SDSS pipeline itself.

²³If we have a catalogue of photometric magnitude measurements in n bands per object, and combine it with m additional photometric measurements in different bands, much more precise properties of those objects can be derived. As photometric redshift estimations make use of differences of magnitudes in as many bands as possible, their reliability depends on the number of pairs of magnitudes which can be built, $\binom{n}{2}$. But if we add m additional measurements, $\binom{n+m}{2}$ colours can be formed, and the relative increase in colours which can be formed is $\frac{(n+m)(n+m-1)}{n(n-1)}$, which is higher than the increase of just adding m colours.

For those applications, the VO provides many more benefits: the access to archives is uniform, allowing simpler scripts, based around web services toolkits, to access all needed archives. Datasets are retrieved in a common format, and ready to analyse with XML tools.

Additionally, the VO community (in particular, the UK AstroGrid team²⁴) has developed what is called the Common Execution Architecture (CEA) [44], a web-services wrapper for remote execution of tasks which can be called either with simple parameters (for instance, generating a synthetic spectrum from Kurucz models for stars with a given surface temperature, surface gravitational acceleration, and stellar type), or tasks which perform transformations on data in VOTable form (such as format transformations, or image generation from tabular data). As those tasks are exposed through a web services wrapper, users can get virtual data (data which was generated on the fly), or re-processed data, with the same kind of queries used for data retrieval.

In any case, the protocols that a VO application should implement are:

Object name queries Many VO queries are spatially related, but for many sources there is a known source name, whose position is already well established. In those cases where the user wishes to query for other datasets near the source, being able to obtain the most recent coordinates, in different epochs, for a given object name, is one of the most important services in the VO. There are several services providing this target name to coordinates resolution, namely Sesame, the NASA Extragalactic Database (NED), and Simbad. However, Sesame is able to query both NED and Simbad, and is the primary service for name solving.

An application querying Sesame must be able to parse not only the coordinates being provided, but also the different object aliases returned by the service, such as conventional names, coincidental sources at different wavelengths, morphology, redshift, and distance²⁵.

²⁴<http://www.astrogrid.org/>

²⁵ For instance, we can query Sesame about *M31*, and we will learn that it is also known as *Andromeda Nebula*, *Andromeda Galaxy*, or simply *Andromeda*. In specific bands, it corresponds, for instance, with several strong infrared sources from the IRAS catalogue: *IRAS F00400+4059* and *IRAS 00400+4059*. Sesame also returns object type information, and simply from querying Sesame we can learn that *M31* is an active galaxy of *LINER* type, with a morphology type of *Sb* (a spiral galaxy without bars, with packed arms), and at a redshift (*z*) of -0.001004 , meaning that *M31* is, in fact, blueshifted, getting nearer to the Milky Way at a rate of approximately 300km/s.

The query which delivers all of this information, in XML format, is:

<http://cdsweb.u-strasbg.fr/axis/services/Sesame?method=sesame&resultType=xi&name=M31>

See the CDS Developer's corner, <http://cdsweb.u-strasbg.fr/cdsws.gml>, for details on this and other services.

Registry queries Independently of the kind of VO services to be accessed, all applications must be able to query the VO Registry (or registries) so that they can look for specific services.

Data access queries Once we have coordinates to make our query (or queries), and the application (or the user) has selected entries in the registry that can provide useful datasets, the data access protocols relevant to the application (SCS, SIAP, SSAP, and in the future the TAP) have to be queried, and the returned data (in VOTABLE format) parsed. Appendix D is devoted to describe the interface to those services, and provides links to their complete specification.

CEA services If we want our application to take advantage of several remote VO services, such as those providing data transformation capabilities, mosaicing²⁶, signal processing, or many others running under the CEA, we must implement CEA services calls. The detailed description and specification of the CEA can be found in [44].

Local messaging services Optionally, the VO interoperability protocols for local data interchange and collaboration must be implemented. This allows a more modular development of VO tools, letting other applications to pre-process the data, and perform analysis in different ones.

In addition, as VO particular data formats are XML-based, XML manipulation toolkits are needed in order to build and interpret VO data, together with FITS processing libraries for applications which must have access to the actual data.

From the list above, it can be seen that the VO application interface for programmers (VO API) consists, then, of several more or less independent blocks, as shown in figure 2.9.

Let us imagine Bruce is an astronomical software developer. He has been creating astronomical software packages before the VO concept, and each time a new interesting archive appeared, he had to modify his application to perform the following tasks, every single time, without being able to reuse any code:

Service query A new service query form, button, or other kind of user interactivity had to be added.

Data access protocol The query had to be implemented by encoding the service URL, service kind, and query parameters. Additionally, if the archive returns data in a particular format, a parser has to be implemented.

²⁶Mosaicing is the non-trivial operation of overlaying several astronomical images, with embedded coordinates and resolution information, in order to provide a composition of the original images with a common resolution.

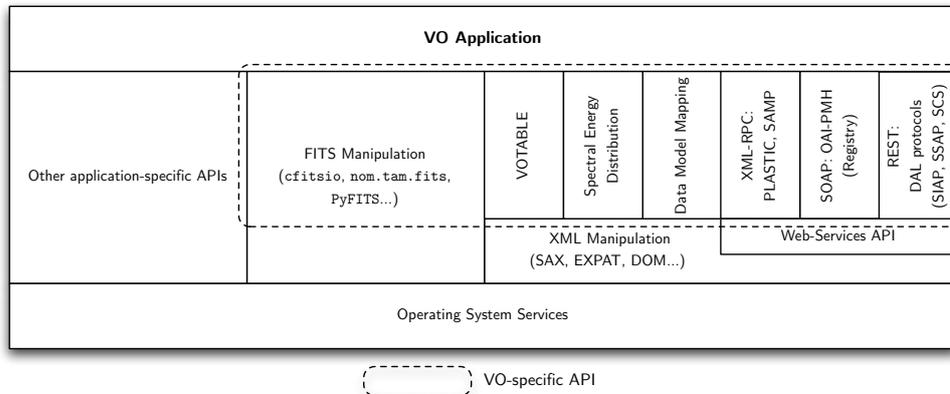


Figure 2.9: The VO as seen from an application (developer) point of view. Applications trying to make use of the VO use web services technologies, such as REST queries, SOAP queries, or XML-RPC messages. Web-services calls require XML manipulation toolsets, which are also used to create, manipulate, and extract information from VOTables, the Spectral Energy Distribution XML representation, or different data model mappings. Finally, VOTABLEs can point to FITS files, and applications which manipulate scientific data must be able not only to read data from VOTables, but directly from FITS files. The VO specific part can be found in the dashed rounded rectangle, while the rest corresponds to either general APIs, or application-specific APIs.

Data translation The response data comes usually in a way that it is specific for the archive/service being queried, and must be translated into the data model of the querying application.

However, all steps above are common for all VO services providing the same kind of data (catalogues, images, spectra), with the addition of the Registry query step needed to find out all services providing that kind of data, which again has to be implemented only once.

It must be noticed that once the data are in the internal data model of the application, it does not matter which was the origin of a particular data set (except for documenting its provenance), and that is the way VO data can interoperate with non-VO data. An example of application mixing VO with non-VO data is Aladin: originally Aladin queried different non-VO services, and all VO-services were added as just one additional data source for Aladin.

The VO from the service developer's point of view

For Bruce, then, the VO is technically rather simple (compared with the archive access and interface operations already in place in the archive): the services have to provide access to existing datasets through a web-services

interface (most of the time, REST-ful services are used for the VO, except for VO Registry services, which use SOAP queries).

However, the hardest part for VO-services developers is the interfacing between the actual data being stored, and the VO data formats, which also include VO semantics. This is achieved by means of a systematic mapping between stored data and data models.

In particular a VO service needs to provide:

VO Service Endpoint(s) A service provider must provide one or more VO services (delivering images, spectra, or catalogue data). Each different service must provide an URL for accessing it, the endpoint. This URL is used to build VO queries by adding parameters to it. Different services, in the sense that the data they provide corresponds to a different scientific data product, need to provide different endpoints.

DAL Parser and Query Mechanism The service endpoint parameters have to be parsed, converted into a query for the internal database (or any other data access and filtering mechanism), and results obtained. In short, the service must comply with the interfaces described in appendix D.

VOTable generation mechanism Once the query has returned results, these must be returned into the VOTable format. This includes not only the building of the main tags, which can be considered a wrapper for a tabular result, but also the assignment of semantic attributes such as ucds and utypes from data model mappings.

Data Model Mapper The VOTable generation mechanism, and the DAL Parser, need a mapping between the data stored in the database and the data model for the kind of observation being performed. This mapping is used for the filling of the already mentioned semantic attributes.

CEA wrapper A service which provides computing services in the VO environment has to comply with the CEA. A computing service which was not built for the VO can integrate into the CEA infrastructure by encapsulating it into a CEA intermediate service (wrapper).

Other task to be eventually performed by the VO Service developer is the endpoint registration. Even when the VO Service can be used as long as its endpoint is known by any other tool (be it a simple command-line script, or a complete application), new services can only be discovered by tools if they are registered in a VO Registry.

2.11 Precedents to the VO

Due to the purely scientific, non-commercial nature of astrophysical data, public access to them has always been the norm. With the connection of most educational and research institutions to the Internet, the first primitive online archives were born. One of the first was that of the International Ultraviolet Explorer (IUE), which was made available to the public in 1985. As the World Wide Web did not exist yet at that time, astronomers had to login to remote servers from which the data were downloaded through the File Transfer Protocol (FTP).

Apart from the data from observations themselves, articles and other publications on astronomical objects were also of great value, and during the *Astronomy from large databases* conference [45], held in 1987, a proposal was made for the design and operation of a system to compile all astronomical publications, the SAO/NASA Astrophysics Data System (ADS). A prototype was built in 1990, with the system being released to the public in early 1993 [46]. The system was made possible thanks to the collaboration of the publishing journals.

Simultaneously, systems like the Simbad Astronomical Database²⁷ [47] (an online database hosted by the Centre de Données Astronomiques de Strasbourg, CDS), or the NASA/IPAC Extragalactic Database²⁸ [48] (NED) contained data with diverse information on observed astrophysical objects, which were compiled, maintained and published online by the respective large scale astronomical data centres.

As all of these initiative had grown independently, they were developed with different considerations in mind, and were not coordinated. However, the possibility of joining publications available at the ADS, and the Simbad/NED databases, allowed astronomers to have access to publications on the objects of their interest, or to access more data in those databases from the corresponding literature. The joint ADS/NED/Simbad effort was called URANIA [49], and is one of the first joint initiatives for data service interoperability in astronomy. The main product of that agreement is the *bibcode*²⁹, a unique identifier of publications which encodes details on the publication date, publication kind, and/or journal, and author³⁰.

Outside the URANIA project, the INES³¹ (IUE Newly Extracted Spectra) [50] archive, holding reprocessed IUE observations, represents the first instance of interoperability between two completely different astrophysical

²⁷<http://simbad.u-strasbg.fr/simbad/>

²⁸<http://nedwww.ipac.caltech.edu/>

²⁹http://doc.adsabs.harvard.edu/abs_doc/help_pages/bibcodes.html

³⁰When astronomers upload data related to a particular publication to the NED or Simbad services, or that data is gathered from the publication, the bibcode allows the cross-identification of sources and publications.

³¹<http://sdc.laeff.inta.es/ines/>

services: one holding spectral information (INES), the other keeping bibliographical references (ADS). It was possible to access the information available in ADS for a particular object from the INES archive, and from the ADS the INES spectra used for a particular article could be accessed. Other service linked to the ADS through bibcodes is the VizieR³² service for Astronomical Catalogues [51].

Another important milestone in the development of online, interoperable archives was the NASA multi-mission archive initiative, which tried to provide a common archival infrastructure across different space-borne missions. Three portals were finally created: MAST³³ (Multi-mission Archive at Space Telescope) [52], mainly for optical-ultraviolet observations³⁴; IRSA³⁵ (InfraRed Science Archive) [53], for the infrared; and HEASARC³⁶ (High Energy Astrophysics Science Archive Research Center) [54] for high-energy (X-rays, gamma rays) astrophysical observations. These portals allowed for the first time a unified view of data from widely different provenance under the same query interface. MAST also provided some applications for data visualisation across mission, such as the COPLOT³⁷ tool. However, data products from each mission were, still, quite different.

Finally, regarding the use of applications in order to combine and manipulate data from different services, one of the first examples prior to the VO development is the Aladin Sky Atlas. Initially, Aladin was only able to access CDS-based catalogues and images, which were accessed through custom built, Aladin-specific protocols. However, Aladin helped in making developers and astronomers realise that any other system which shared the data access and data description protocols would allow either to improve Aladin, by allowing it to access more datasets, or more easily recreate Aladin-like functionality, without the need of being a data and service provider.

2.12 Comparable activities in other disciplines

We will conclude our introduction of the VO with a selection of other e-science activities which are similar in scope to the Virtual Observatory, but belong to different scientific fields, or have a different technology base: the data storage support of data grids, Geographical Information Systems, and the Bioinformatics Harvester.

³²<http://webviz.u-strasbg.fr/viz-bin/VizieR>

³³<http://archive.stsci.edu/>

³⁴VLA FIRST images are also available through MAST.

³⁵<http://irsa.ipac.caltech.edu/>

³⁶<http://heasarc.gsfc.nasa.gov/>

³⁷http://archive.stsci.edu/mast_coplot.html

Data grids

In e-Science talks, grid technologies tend to come up first and foremost. It is very usual for national e-Science initiatives in European countries to be represented by the National Grid Initiatives, as if e-Science could only be performed via grid technologies. It is true that grid middleware provides tools to implement much of VO core functionality, but at the cost of a greater complexity, and greater demands on client systems.

The definition of grid computing by Ian Foster as *a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities* [55] is just one part of the Virtual Observatory.

But computation is just a fraction of the VO. In fact, the VO can be more closely identified with a *data grid* [56]: an extension of the grid protocols in order to create a standard for distributed data storage, data identification, and metadata management which can integrate archives from different disciplines.

In the VO, the *data grid* exists via the data access protocols, and the standardisation of metadata. A version of the VO can exist implemented on top of grid protocols, but web-services technology was chosen instead in order to minimise the cost of entry for participant institutions/research groups: web-services technology is easier to implement and deploy, both for servers and specially for clients, and it was very important to be able to have a productive VO that could engage the community.

In the future, however, the VO can move more towards a grid technology foundation, when the computing needs overcome the data needs, and the VO metadata standards are complete. In the meantime, VO protocols are as technology agnostic as possible. For instance, the IVOA is working in a distributed file system for VO tools, reminiscent of grid data storage pools, with VO specific metadata and implementation independent (it can use WebDAV, NFS, GridFTP, or any other supporting protocol), called VOspace [57].

Geographical Information Systems

There are many similarities between the VO and Geographical Information Systems (GIS):

Coordinate system based Both for GIS and the VO, coordinates are one of the main indices on data. Finding data nearby a particular coordinate is also a common feature, and the retrieval of nearby candidates cannot be based on floating point differences, as it would become computationally costly. The solution for fast operation of coordinate-based data retrieval is the hierarchical partitioning of the search space using, for instance, techniques such as the Hierarchical Triangular Mesh [58] or the Quad Tree Cube [59].

Object to coordinates resolution Also common to both problem domains is the need to obtain coordinates for a particular object whose only known property is the name. In astronomy such needs are typically handled by the Simbad or NED services, while different GIS systems will use different local or remote databases for obtaining the coordinates.

Use of metadata GIS objects are primarily graphs³⁸ with geographical coordinates attached to both connections and nodes. But those curves represent roads, buildings, water and gas distribution pipes, agricultural plantations, et cetera, and each object kind has its own set of metadata, in the same way VO data uses metadata to establish object kind, and information kind.

The main difference between the VO and GIS systems is that for the VO there is a strong standardisation force mandating which data formats to use, and there is usually no restriction in data usage³⁹, and interoperation and networking is a key concept in system design, while it is not so for GIS.

However, some GIS systems such as Google Earth⁴⁰ have started to show the potential of having a common geographical description format combined with the distributed dataset creation and distribution: GIS data, both images and object catalogues with metadata can be updated on the data server, and all clients access updated data from then on.

It is remarkable in this context that Google Earth has evolved to include Google Sky, which demonstrates the similar nature of GIS and VO systems when the distributed, interoperable data access is taken into account.

Bioinformatic Harvester

The Bioinformatic Harvester⁴¹ (BioH) is a bioinformatic meta search engine at KIT Karlsruhe Institute of Technology for genes and protein-associated information. It queries data from more than 35 bioinformatic sites, which allows for searches in the genome and proteins of several animals of the most important sequenced animals —among them humans, rats, or the drosophila (fruit fly)— or on the complete collection.

The similarities with the VO lie in the distributed nature of the integrated systems, but the similarities end there:

- the BioH does not use a unified, standardised protocol for harvesting the data available in the different resources; instead, it uses a custom built harvesting engine for each collection.

³⁸In the mathematical sense: a set of directed connections between sets of nodes.

³⁹After all, astrophysical data tend not to be of immediate value for entrepreneurial use.

⁴⁰<http://earth.google.com/>

⁴¹<http://harvester.fzk.de/harvester/>

- there is only one entry point to the Harvester, and there is no public service access entry point for external applications.

We can see the BioH as bioinformatic equivalent of the state of affairs in astronomy prior to the VO: similar to the joint ADS and Vizier systems.

2.13 Status of the VO

As we have said, the VO initiatives were conceived around 2000, and first prototypes built around 2002. Even when some parts of the VO are still being built—being the IVOA Table Access Protocol (TAP) one of the salient examples of *in progress* infrastructure—, and some of them being subject to changes (the Image and Spectral Access protocols are being reviewed), many of the tools are mature enough as to produce scientific results beyond what was possible prior to the VO.

In particular, the first paper on VO-enabled science was published by Paolo Padovani in 2004 [60], and reported on the discovery of faint, obscured quasars with Virtual Observatory tools. However, more than half of all VO-enabled scientific papers⁴² have been published since 2007 and onwards, while more than 90 percent of papers were published after 2006, which indicates a shift toward the maturity of the VO infrastructure and tools.

This can be easily correlated with the development of IVOA standards: the first IVOA Recommendation of the VO Table dates from October 2003⁴³; the first agreed working draft for the Simple Image Access protocol was published in 2004⁴⁴; the first ConeSearch services were available soon after that, with the ConeSearch specification being published in 2006; the Simple Spectral Access protocol, finally, was only proposed as a recommendation in 2007⁴⁵, and prior to that only prototype spectral services, reusing the Image Access protocol, existed.

It is interesting to mention that one of the national VO initiatives pushing the most towards the development of VO scientific tools is the SVO, as more than half of VO papers have been published by SVO-related groups.

In the IVOA InterOp meeting of Baltimore (October, 2008), the former IVOA Chair, Dave de Young, proclaimed the VO had entered the operational status, where all infrastructure is mostly put into place, and science is routinely possible with existing tools.

In this thesis, we will try to help in bringing radio astronomical archives and tools to that operational stage.

⁴²<http://www.euro-vo.org/pub/fc/papers.html>

⁴³<http://www.ivoa.net/Documents/PR/VOTable/VOTable-20031017.html>

⁴⁴<http://www.ivoa.net/Documents/cover/SIA-20040524.html>

⁴⁵<http://www.ivoa.net/Documents/cover/SSA-20070604.html>

Part II

Radio astronomical archives in the VO

Chapter 3

Introduction

Due to the more abstract nature of radio astronomical observations, there are less astronomers and observatories providing radio astronomical data when compared to those working on the visible part of the spectrum. However, as multi-wavelength studies become more and more commonplace in astrophysics, and are essential for studies such as the one being carried out within the AMIGA group, the capability of having access to radio astronomical data becomes of the utmost importance.

There are few radio astronomical observatories with proper archives (most of them provide just logbooks, or observation registries), so providing VO-compatible radio astronomical archives is a valuable contribution to the community both because of the availability of the archive itself, and the bonus possibility of using VO-tools to access it.

For existing archives, the data model of the archive itself does not usually match the data model of the VO, mainly due to the difference in scope: archive data models reflect the data origin, and are built to answer the queries of engineers, commissioning scientists and scientists; VO data models, on the other hand, are built to describe observations so that the description is enough for astronomers, or computer queries, to assess the usefulness of a particular piece of data.

For a new archive, however, the internal archive data model can be built by mirroring VO data models, while adding non-VO information so that all needs can be covered.

In this part, we will describe the RADAMS, a data model for single-dish radio astronomical archives reflecting existing IVOA data models, but which at the same time provides definitions and proposals for modelling additional observational aspects. We will use this data model for the building of the IRAM 30m and DSS-63 archives, which will be shown in Part [IV](#).

In addition, some parts of the RADAMS will be used in the development of the MOVOIR, a modular VO interface and API to the VO, which will be described in Part [III](#).

We will start by introducing which are the existing IVOA data models in chapter 4, and then we will evaluate parts of those models to create the RADAMS (Radio Astronomical DATA Model for Single-dish observations) in chapter 5. Chapter 6 explains how the RADAMS can be used to characterise astronomical observations, and the following chapters explore in detail which are the missing parts in IVOA data models which are needed for the RADAMS: Curation, Packaging and Policy are specified in chapter 7, while data provenance in e-science is reviewed in chapter 8, and the lessons learned applied to RADAMS' Provenance in chapter 9.

Chapter 4

Data modelling in the VO

The interoperability of tools and archives in the VO is only achievable by standardising the way datasets are accessed, and how the scientific data they contain is described. The first part is solved by means of standard access protocols, while the latter needs the development of suitable data models.

A data model can be defined as a complete description of the set of entities needed for information storage in a particular field, which specifies both the data being stored, and the relationships between them.

In this way, it can also be seen as the framework in which questions about the data (and metadata) can be posed: only questions which can be answered regarding the information AND relationships encoded in the data model can be answered within the VO framework. And considered this way, a uniform, interoperable observation data model can be mapped into a uniform set of questions which can be answered within the VO.

Two main classes of data models are typically considered:

Domain data models These are high-level descriptions of the entities to be taken into account in order to fully implement a data model. Particular attributes of the data entities involved are not set, except for those needed for establishing the relationship between entities.

Implementation model Implementation-dependent description of the particular entities and attributes to be actually stored.

VO data models need to be a mixture of the model types above: for the data models to be used across observations with different instruments, and with different scientific objectives in mind, a high-level description is needed. In addition, VO data models affect how data from observatories' archives are exposed through VO services, as the way such data are stored by the archive might differ from the IVOA standards. However, there are particular attributes that need a precise description in order for the model to be useful,

and to be able to implement it, so VO data models need to fix the kind of metadata they support.

Within the VO, data models apply not only directly to the scientific data, but to the metadata describing them. As the way to structure information depends on the application domain, VO data models describe astronomical datasets in a way that is as instrument independent as possible, to ensure that the same description can be used for data with different origins. Users must also be able to query those data models to be able to find datasets which comply with certain properties.

4.1 Elements of a data model

When defining a VO data model, we have to specify:

Entities Being the data model building blocks, they group related attributes within a data model. They can be mapped to Classes in Object-Oriented Programming (OOP), or Elements in XML.

Fields They are the actual data elements of the model. They map to Attributes in OOP, and they can be mapped to Attributes or to Elements without children in XML.

Relationships The different entities and fields have hierarchical or relational relationships: an observation project has projected observations, and all entities which share a common project ID are related, for instance. For the data model to be uniquely defined those relationships must be made explicit.

Cardinalities The number of object instances allowed as part of a relationship. It is specified as a range of valid number of entities. For instance, an observation can be related to any number of data files, but it needs to be related at least with one, meaning that the cardinality of the Observation to Data files relationship would be specified as 1..*. For objects which can appear any number of times, including none, the cardinality is expressed as 0..*. Objects which are optional, but if present can appear just once have their cardinality expressed as 0..1, and so on.

Data types For computers to be able to correctly interpret a data stream a Data type needs to be specified. For instance, object IDs could be Integers, but they are normally textual, so String data must be used. We could consider the restrictions which can be defined for complex data types in XML as part of the data typing.

Units No physical quantity can be specified without providing its units. Physical-data related Fields need Units to be specified, or Units have to

be a fixed property of certain Fields, but they either need to exist as an implicit attribute of a particular field, or to have their own dedicated Field.

Semantics As observation metadata are related to real-world elements and quantities, VO data models also imply data semantics —i.e., what is exactly meant in the real world by a particular field— to avoid ambiguities. Most of VO semantics are provided via Unified Content Descriptors (UCDs) and UTypes.

4.2 Semantics, UCDS, UTypes and IVOA vocabularies

UCDs are a controlled vocabulary¹, under the supervision of the IVOA Semantics WG, which provides a list of *atoms* which can be used to identify fields as corresponding to specific astronomical quantities. For instance, a field containing the Right Ascension can be identified by the UCD atom `pos.eq.ra`, while a photometric flux² in the *V* band³ can be identified by juxtaposing the two UCD atoms `phot.flux; em.opt.v`. This provides both a unified vocabulary to identify any astrophysical quantity, and an automatic knowledge discovery tool for fields with arbitrary relationships. In fact, UCDS were born out of a joint CDS/ESO data mining effort [61].

However, UCDS can only provide *data kind* information, but not relationship information. In a sense, they are a kind of specialised *unit*, complementary —orthogonal— to physical units: in the same way that quantities with the same physical units can be very different in nature (i.e., both the decay time for an isotope and the oscillation period for a pendulum are both measured in seconds, but do not have any other physical connection), fields with identical UCDS can also be related to different real-world phenomena. In order to allow such deeper relationships to be expressed, and disambiguate metadata fields UTypes were born.

UTypes are created from a hierarchical data model by enumerating the different parents a particular field has in that hierarchy. For instance, a field containing the Right Ascension in equatorial coordinates for where an instrument was pointed to corresponds to the spatial coverage characterisation, in particular to the Location property, and thus it would sport a UType of `characterisation.coverage.spatial.location`, the UCD would be `pos.eq.ra`, and its units could be any angular unit.

But even with the help of units, UCDS and UTypes, sometimes it can be difficult to tag a particular piece of data with meaningful semantics, specially

¹<http://www.ivoa.net/Documents/latest/UCDlist.html>

²Photometric flux is the integrated flux received within a particular band; in practice, it is the integral of the emitted flux weighed by the corresponding filter response.

³The *V* band is defined by a filter with central wavelength around 540 to 550 nm, and with Full Width Half-Maximum of 80 to 90 nm.

for data which does not have a direct place in a VO data model. For that we can borrow techniques from the Semantic Web (an effort for providing web documents with semantics, so that, for instance, a table of camera prices can be tagged so that software tools can identify in it prices, if possible belonging to digital cameras, even to particular brands), and provide one or more standardised astronomical vocabularies. The IVOA Semantics WG⁴ has started recreating controlled vocabularies such as UCDs in Semantic Web form, and even the IAU thesaurus has been recreated in that way [62]. We will show how we are using them in the IRAM 30m archive in order to provide semantics to data related to antenna engineering terms.

4.3 Role of data models in the VO

We can identify in the VO four different phases, and we can see that in all of them data models play a central role:

Discovery Datasets available in the VO have to be discoverable for them to appear automatically in VO tools. The VO Registry holds information regarding existing datasets so that they can be easily discovered. For this phase to be standardised, we need: data models for Resource meta-data (ResDM), where a Resource is either a data provider, an authority, or a data service; a data model for Space-Time Coordinates (STC), so that coordinate-based, region-based or time-based searches can be performed; and a data model for dataset Characterisation (CharDM), so that searches on physical or instrumental properties are possible. In addition, the UCD and IVOA thesaurus (IVOAT) are relevant in this phase.

Evaluation Datasets have to be evaluated in order to assess their applicability to the kind of analysis we might wish to perform; for instance, in order to do image mosaicing we need a certain coordinate overlap, and in order to do image stacking we need an almost complete overlap, and comparable resolutions. The main data model involved in this phase is the CharDM.

Data Access There is an implicit data model in the IVOA data access protocols, the Data Access Layer (DAL), which is centred on targets (coordinates with tolerances/search radii), and uses several properties from the CharDM, such as the Coverage in several axes, and a streamlined form of the STC.

Transformation When creating a new dataset, or transforming an existing one, a new CharDM instance needs to be created, one that characterises

⁴<http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/IvoaSemantics>

the union of the participating datasets. If the transformed data set is a spectrum, the Spectral data model (SpecDM) is needed both for obtaining the complete description of the original data and describing the transformed product. There is no existing data model yet for images or for more complex data within the VO. In addition, in order to trace the origin of the transformed image we would need to use a Provenance data model, that apart from being an integral part of the Observation data model (ObsDM), it should be built in a stand-alone form so that it can be applied to newly generated, non-observational data.

4.4 Data modelling diagrams

For specifying the aforementioned elements several notations exist, being the Unified Modelling Language⁵ (UML) by Rumbaugh, Jacobson, and Booch [63] one of the most widely used. Some of the IVOA data models have been specified in terms of UML diagrams, and will be used when available.

However, UML tools tend to be too onerous for data modelling when the data model is not going to be used to generate code for implementing that data model in memory, and we have resorted to simplified entity-relationship diagrams [64], with specification of the cardinalities when they are not made explicit in the text, and relational attributes included in the diagrams.

4.5 Existing IVOA data models

We have talked about different data models in use within the VO. In this section we will review the data models most relevant to the development of observation related archive data models, and see what their implementation level is.

Observation

The Observation Data Model (ObsDM) [65] was started as an effort to create a common framework in which all kinds of astronomical observations could be described. In that regard, it can be thought of as a Domain model, but with a strong focus on the ability to perform queries on the stored observation metadata.

In words of its authors, each ObsDM instance *describes a single dataset which may be [either] a dataset corresponding to an observation of the sky, [or] a dataset derived from many observations, [but] with the stipulation that the dataset is intended to be analysed independently of other datasets, and contains all the primary data needed for such analysis.*

⁵<http://www.uml.org/>

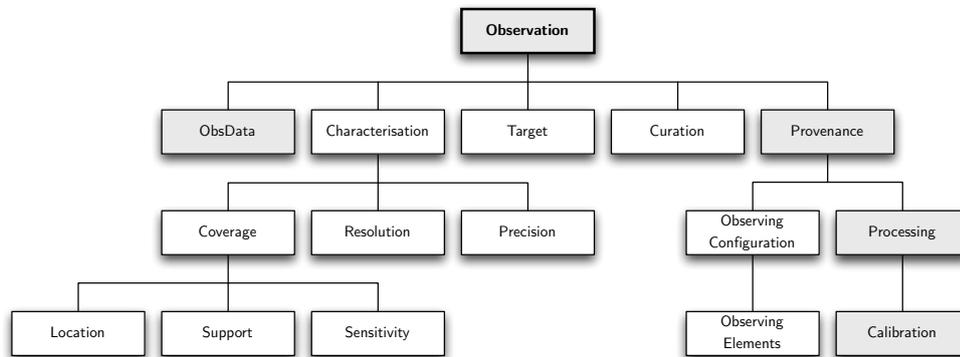


Figure 4.1: The Observation Data Model. Diagram recreated from the *General Model for Observation* figure in the Data Model for Observation working draft [65] from 2005. The shadowed rectangles show data and metadata items which can change in case of a reanalysis of the observational data, highlighting the essential difference between Provenance due to observing configuration setup and data processing. The hierarchical association does not show object cardinality, i.e., the number of times instances of an object can be related to its father object.

The ObsDM tries to provide, in a first approximation, all metadata needed for data selection and retrieval, while being extensible to the more specific case of all the metadata needed by data analysis applications.

Figure 4.1 shows the general model for observation as contained in the first draft of the ObsDM working draft.

First, that figure shows that the main constituents of the ObsDM are the dataset Characterisation (where in parameter space can the observation be found), Target (what is known about the target of our observation), Curation (who is responsible for getting this observation into the archive, or publishing it into the VO), and Provenance (what has been done to the set of photons which correspond to this observation).

The figure also illustrates the main difference between two different classes of Provenance: the metadata and data that would change in case of a reprocessing of observed data are shadowed in grey. Neither the raw data themselves nor their characterisation would change, while the data provenance not having to do with the observational setup would have to reflect those changes.

Another issue that can be derived from that figure is that the Characterisation is one of the most observation technique independent sub-data models of the Observation, and as such the work on the Observation data model was delayed until the Characterisation data model was finished.

We must note that even when the STC [66], and CharDM [67], have

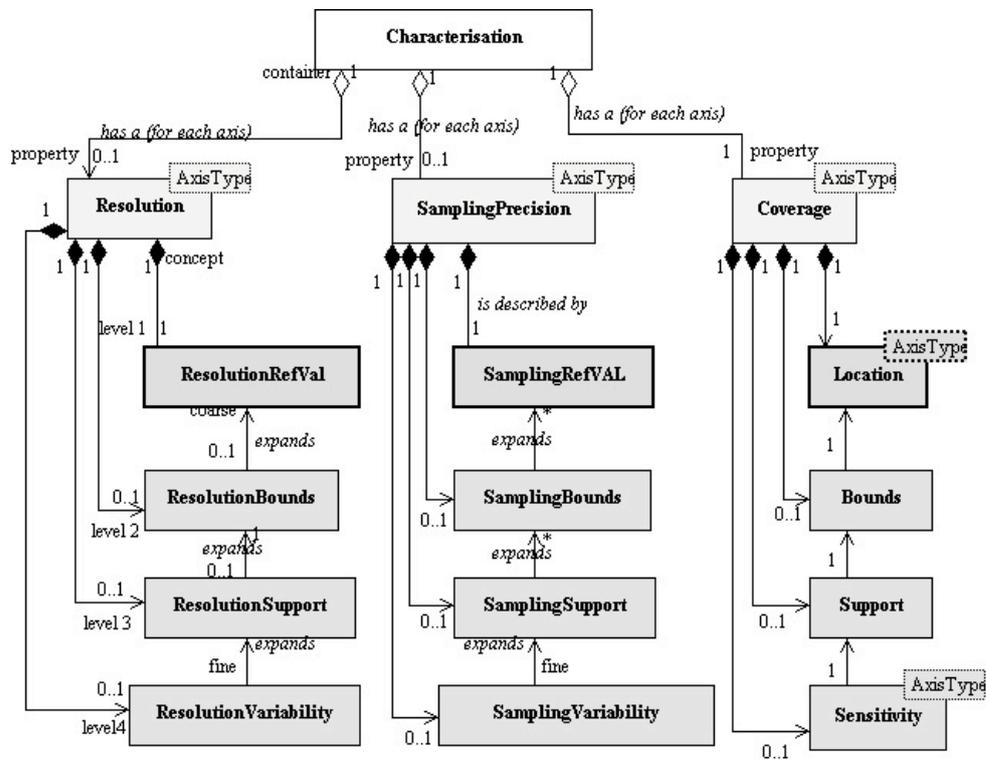


Figure 4.2: Characterisation Data Model, described per axis, as a UML class diagram. Reproduced from the CharDM recommendation [67].

reached the status of Recommendation, the ObsDM [65] is still an early working draft whose main role has been providing momentum to the STC and Characterisation efforts.

Characterisation

As previously said, the Characterisation data model (CharDM) deals with the question *Where in parameter space is this observation?*, where the parameter space is comprised of the observation world coordinates (WCS) axis, the time axis, the spectral axis (and its complementary velocity axis), and the observable axis. Additional axes could be taken into account, as the structure for each axis is practically identical.

The CharDM (*Data Model for Astronomical DataSet Characterisation* [67]) is currently an IVOA Recommendation (since March 2008), and it covers all the sub-classes established for the CharDM in the ObsDM, but a few more. Figure 4.2 shows the current structure of the CharDM per specified axis.

We can see that a Characterisation instance, composed of several CharacterisationAxis instances, each of them with a Coverage class, and optional

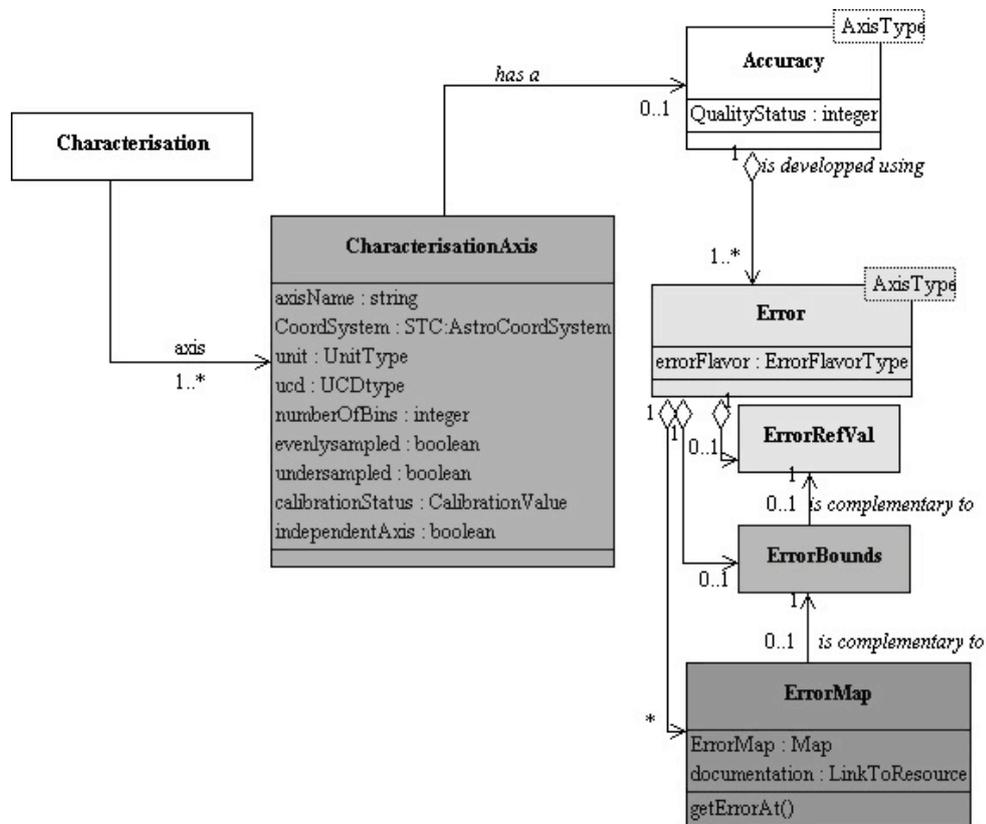


Figure 4.3: Each Characterisation instance must contain one or more CharacterisationAxis instances, to which an optional Accuracy instance can be attached. Reproduced from the CharDM recommendation [67].

SamplingPrecision and Resolution classes. The Coverage class must include a Location specification, and if they exist, both the SamplingPrecision and Resolution classes must provide reference values (RefVal). Both Location and Reference values can be optionally further specified by Bounds and Support, and a final specification in the form of Sensitivity for Coverage, and of Variability for SamplingPrecision and Resolution.

It can also be noticed that the current CharDM has evolved with respect to the CharDM outlined with the ObsDM draft: there is an additional Bounds class related to Coverage in each of the axes to be characterised.

In addition, and not shown in figure 4.2, an Accuracy class has been added which specifies both data quality and errors. Figure 4.3 shows the Accuracy class. Each CharacterisationAxis instance can optionally contain an Accuracy instance, which is further defined by an Error class which, in the same spirit of the remaining Characterisation classes, has a RefVal, Bounds, and instead of Variability an ErrorMap.

We can see that the CharDM is an integral part of any dataset description to be made, and is as observation independent as possible. Indeed, the CharDM is also used within the VODataResource service description in order to characterise whole archives, or particular subsets of interest in the Registry.

Space and Time Coordinates

The Space and Time Coordinates data model [66] (STC) was created in order to have a systematic way of specifying different coordinate systems within the VO.

While initially the supported coordinate systems for VOTables and data access protocols have been Equatorial, with the possibility to further specify an epoch, and the IAU has gone a step further with the approval of the International Coordinate Reference System (ICRS), equatorial coordinates are not the natural system for several astronomical disciplines, such as Solar System science, or Galactic studies.

The STC is both a data model (it specifies quantities and relationships), but also incorporates two ways of expressing serialisations of data model instances: STC-X, an XML-based STC serialisation, and STC-S, which is string-based and more compact and human readable, for use in data models where an XML payload cannot be delivered, or ease of writing is desirable.

Within the VO there are three places where the STC can be used: data access queries, Characterisation of observation Coverage in both the Spatial and Temporal axes, and the Object to Position resolution. However, for most of the existing VO services, either J2000 or ICRS equatorial coordinates are assumed.

The complexity of the STC stems from the very different coordinate systems used in astronomy, and the aim to be an all-encompassing effort. A glimpse of its complexity can be seen in figure 4.4, which shows the most basic STC entities⁶.

Even when the STC is an IVOA Recommendation, is subject to analysis for better embedding in other data models (see the Spectrum case in the following section), and tools for creating/analysing STC entities are still to be developed.

Spectrum

The Spectrum data model (SpecDM) is different from the data models above in that it describes a particular data product, and not just a generic observation or an observation set.

Figure 4.5 shows the high-level overview of the Spectrum data model. Compared with the ObsDM, —see figure 4.1— and the CharDM —see figure 4.2—, it can be seen that the Spectral data model is based on the ObsDM,

⁶And on the STC Recommendation length, which runs at 109 pages.

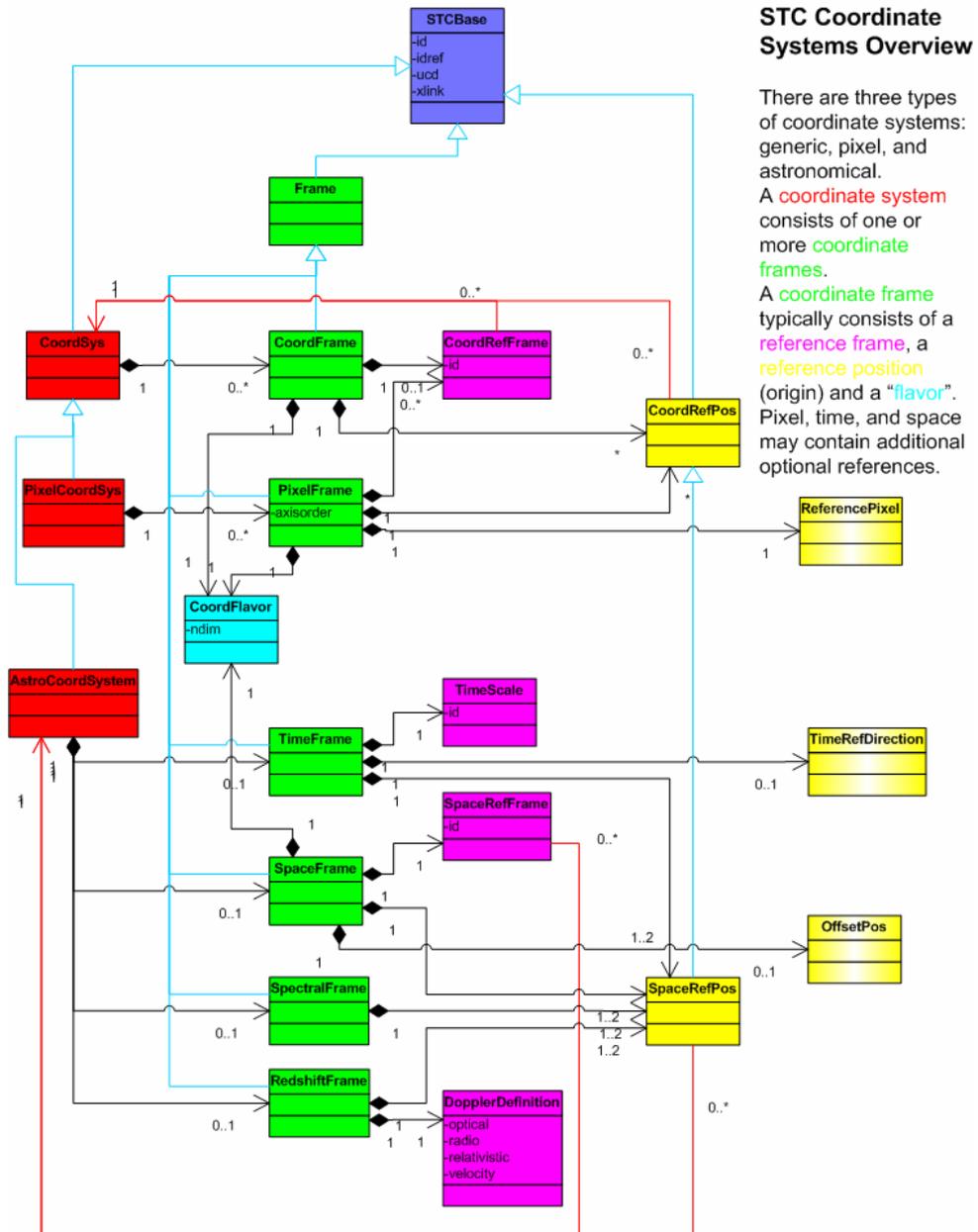


Figure 4.4: An overview of the Space and Time Coordinates data model. Reproduced from [66].

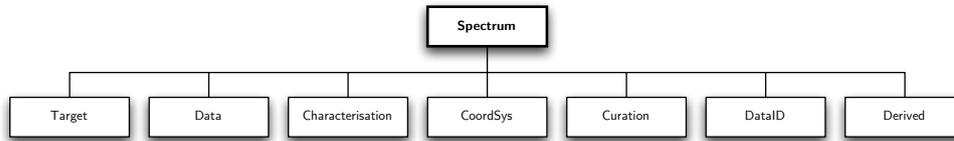


Figure 4.5: High-level overview of the Spectral data model. The root class is the Spectrum class, which can be compared to the Observation class in the ObsDM.

without specifying a Provenance class; it specifies a CoordSys class separate from Target, and also a DataID (a simplified data curation for a particular entity); and a Derived class for holding information which does not belong to the Spectrum, but can be derived from it under several assumptions, such as the Signal-to-Noise Ratio (SNR), Redshift and Amplitude variability.

Due to the already mentioned complexity of the STC, the SpecDM uses a simplified version of STC-S entities to describe spatial regions (circles and polygons for the SpecDM) and temporal coordinates.

There are many different techniques to obtain spectra, and each of them produces a different data product. The most important difference concerns on whether the spectrum is obtained by a diffraction or refraction method, which provides a sampling on wavelength; or if the spectrum is obtained via a Fourier transform method (from the Fourier transform of an auto-correlation signal, for instance), which provides a sampling on frequency. As frequency and wavelength for electromagnetic radiation are linked by a non-linear equation ($\lambda = cv^{-1}$), it is very important for astronomical data processing applications to treat each case differently⁷.

The SpecDM has reached IVOA Recommendation status, but its Characterisation class is still not fully compatible with the last version of the CharDM, as sanctioned as IVOA Recommendation. Debate is still open on how to use it for time series, instead of the time series provisions included in the STC data model. Finally, the Spectrum class can only identify one single spectrum. Work has yet to start on the Spectral Associations data model, in order to describe, for instance, On-The-Fly spectra, 3D IFU spectra, or any other multidimensional datasets where one of the dimensions is frequency, wavelength or velocity.

4.6 Other astronomical data modelling efforts

Apart from the IVOA data models mentioned above, we have reviewed some other data models relevant to astronomical observations' data modelling, or

⁷To the point that the SpecDM recommends using different Spectrum entities for spectra which are available both with a frequency and a wavelength scale.

data modelling for radio astronomical archives.

The only IVOA Note on radio astronomy data prior to the publication of the RADAMS was the *Data model for Raw Radio Telescope* by Lamb and Power [68].

Many aspects of our work —specially controlled vocabularies, and the extensibility to interferometry— were initially based upon this model.

Other models reviewed include the data model used for publication of the Australian Telescope Compact Array (ATCA) [69] archive⁸, the scientific archive domain model of the National Optical Astronomy Observatory (NOAO) [70] archives⁹, and the ALMA Science Data Model¹⁰ [71].

A special mention is made of the MPEG-7 and MPEG-21 overview documents [72, 73], available in web-page form at <http://www.chiariglione.org/>, used as the basis for our Policy and Curation sub-models.

4.7 Conclusions

Within the VO, data to be delivered must be described in the most complete possible way so that automated selection and manipulation tools can rely on that description in order to manipulate, select, and ultimately provide *understanding* of the described data.

Both the data themselves and the metadata need to conform to a common data model, in order to make interoperability between different systems possible. Those data models are governed by the IVOA DM WG.

The only observation-related data model already in Recommendation stage is the SpecDM. The other data models having reached Recommendation status are the STC data model, and the CharDM.

The SpecDM shows that it is possible to use the ObsDM as a template to create an observation-specific data model, and both the CharDM and the STC have shown their modularity in order to be embedded in other data models. However, STC entities are usually simplified before being properly used.

In order to create a complete VO-compliant data model which can hold all kinds of radio observations from different single-dish observatories we will provide, then:

Compatibility with existing and/or proposed data models Any VO-compliant data model should make use of existing and recommended IVOA data models, and should try to fit themselves within the data model most related to the kind of data to be described.

⁸<http://atoa.atnf.csiro.au/>

⁹<http://archive.noao.edu/nsa/>

¹⁰<http://aramis.obspm.fr/~alma/ASDM/ASDMEntities/>

Use of existing data models in comparable e-Science disciplines The Virtual Observatory is one example of an e-Science based discipline: the main success of the VO comes from the federation of disperse datasets, and their aggregation through network access protocols. As such, it would be wise to make use of the existing e-Science middleware and conventions for non astronomy-specific parts of the system.

Support for spectral associations The SpecDM provides support for spectral measurements, but there is no way to relate spectra taken from On-The-Fly observations, or spectra extracted from a radio data cube, or maps of continuum measurements. This support is added to the RADAMS by combining the CharDM and Packaging.

Specification of missing classes The ObsDM proposed classes for dealing with data access Policies, data Provenance, and data Packaging, while the CharDM proposed a Sensitivity class. A data model for radio astronomical observations would need to provide definitions for such classes.

Support for radio astronomical observing modes There are widely different observing modes in radio astronomical, single dish, observatories. A VO data model for them would have to support them.

Extensibility for radio interferometric observations Most of the data model details can be generalised for radio interferometric observations, and where possible the way to extend the data model in order to support interferometric observations is made explicit.

We will address these requirements in the following chapters.

Chapter 5

RADAMS: a Radio Astronomical Data Model for Single-dish radio telescopes

In the past chapters we have been making emphasis on how the VO works by providing common data access protocols, common data formats for information interchange, and a common data model for expressing, within the constraints of the data model and data access protocols of the VO, the description of observational data.

In particular, in the previous chapter we have shown the general framework proposed by the IVOA for describing whole observations, characterising the parameter space occupied by them, and using a specific format/data model for indicating space and time restrictions. We have also seen that the Observation data model is not complete, while the CharDM and STC are mature enough to have been granted the IVOA Recommendation status. However, they have never been applied to radio astronomical archives, and have never been used as the basis for a radio astronomical archive.

In this chapter we will try to answer the question: *How to architecture a VO-compatible radio astronomical archive?*, and for that we will use the outlined ObsDM as a foundation in order to create a complete data model for radio astronomical observations which can be used both for the development of single-dish radio astronomical archives, enhancing the existing IVOA Observation data model at the same time.

5.1 Basic requirements of astronomical archives

We will start stating the obvious: a radio astronomical online archive has to provide a way for astronomers to find and retrieve radio astronomical observations. The most simple incarnation of such an archive would include

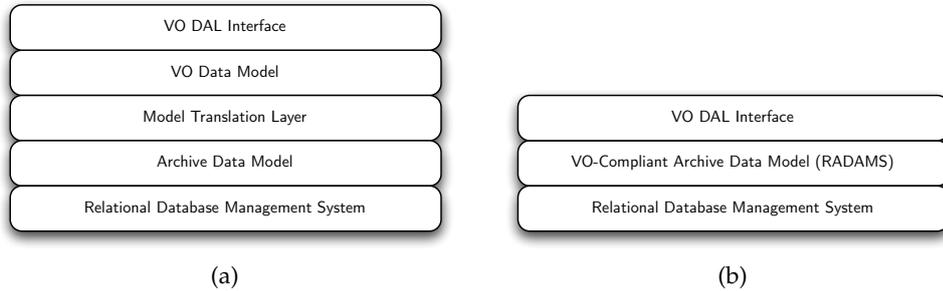


Figure 5.1: Data modelling layers for different kinds of VO archives. Sub-figure (a) shows the layered structure of a VO archive built on an existing, while (b) shows the simplicity of an archive built from scratch for VO compatibility.

a complete list of all observation files, together with a system for retrieving the actual file containing the observational data set.

With such a simple archive system only people aware of what was observed and contained in each file would be able to use it. Thus, the most basic additional requirement all astronomers would need is coordinate searching, or the ability to identify files with positions in the celestial sphere.

The next requirement in order of importance is the identification of the band of the electromagnetic spectrum being scanned by the instrument, as different bands provide diverse information on the physical processes taking place in the observed region. This is the first place where the specifics of radio astronomy begin to emerge.

These pieces of information do not actually belong to the science data themselves, and are metadata for the observation. In particular, the two examples above can be identified to the lowest levels of detail in the CharDM for the spatial and spectral axes (Location and Bounds in both axes).

This small example tells us two things: first, it shows that the most important metadata needed for information filtering are already part of the proposed IVOA data models; and second, that such data models can be used as a blueprint for astronomical archives

A welcome side-effect of directly using VO data models as the basis for astronomical archives is that their use simplifies the archival development, and the integration of fully described archives in the VO.

To illustrate this, we will compare the development layers and complexity needed to build VO compatibility both on existing archival systems, and by building an archive from scratch. Figure 5.1 shows both kinds of archives side by side.

In the case of an archive with an already existing infrastructure —subfigure 5.1a—, VO data models have to be mapped on top of a translation layer in charge of creating VO entities from the existing archive data model. But

if the archive is going to be built from scratch, the Data Access Layer (DAL) interfaces¹ can be built on top of the VO entities directly provided by the archive infrastructure—in the case of subfigure 5.1b, the RADAMS—.

It should be noticed that for building VO archives on top of an existing, non-VO archive, the translation layer has to provide some metadata which do not only need translation, but in many cases those metadata have to be extracted or even calculated from either the FITS headers, the actual FITS data tables/images, or even the observation logs².

5.2 RADAMS requirements and overview

Once established that a VO-based data model can indeed be used as the basic architecture for an astronomical archive, we can define which will be the requirements of such a data model for radio astronomical observations, our RADAMS.

Based on the Observation and Characterisation DMs The RADAMS is a data model for the metadata regarding a radio astronomical single-dish observation, and as such is based on the Observation data model. In order to complete the RADAMS, extensions are provided within the framework of the ObsDM and the CharDM.

Separation of data and metadata In a RADAMS based archive, data will be stored in the form of FITS files or VOTables, whereas all metadata will be stored in database form complying with the RADAMS. For already existing archives, this condition is relaxed, but a query mechanism for all RADAMS metadata should be available.

Single-feed observations Metadata and observations are considered always to be referring a single feed. If a telescope or instrument is able to provide several feeds simultaneously, each one will be stored separately, and will refer to the same observing proposal, but will have a separate existence.

Radio astronomical data products The RADAMS will be able to support data of the following kinds:

- Single-pointing flux measurements;
- Single-pointing radio astronomical spectra, or spectra associations;

¹The DAL data model and interface will be briefly described in relation with the different data products to be delivered.

²Possibly, even a combination of all of them. Actual examples will be shown in the chapter devoted to radio astronomical characterisation with the RADAMS.

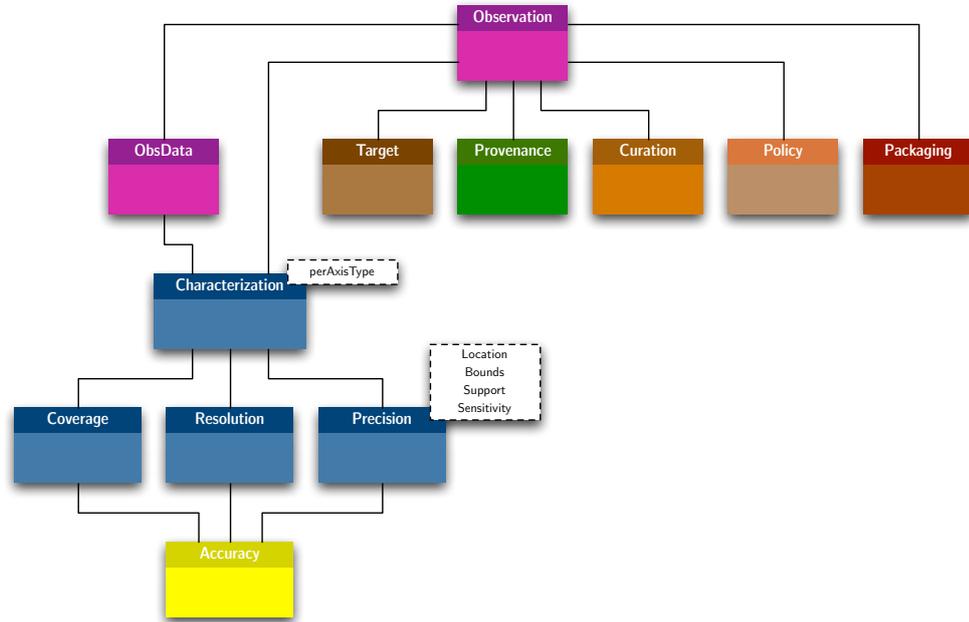


Figure 5.2: RADAMS general class organisation. Different colours correspond to different sub-models, and will be kept in the next chapters.

- On-the-fly on-off, spectra data, or continuum flux observations, in the form of images and data cubes;

The data product definition must be made compatible with existing IVOA DAL protocols: images (SIA protocol), sets of spectra (SSA protocol), and coordinates-based data tables.

From those requirements, we have built the RADAMS data model, whose high-level structure is shown in figure 5.2. We can see that we have closely followed the ObsDM, and the CharDM, and that in order to use the whole ObsDM we will need to provide definitions for some of the classes.

Those classes and sub-models defining the RADAMS will be, then:

Observation Root class for the data model. It works as a hub to which both actual observational data (ObsData) and the remaining metadata are linked together.

ObsData Describes how to access the actual data being described by all the RADAMS' classes.

Target Describes the target of the observation, providing as much information as available for already known targets.

Characterisation Corresponds to the CharDM classes.

Provenance Links all the information regarding the origin of the observation, both from an administrative, technical and scientific point of view.

Curation Describes who is responsible for maintaining a particular observation, a set of observations, or all of the archive.

Policy Is used to specify the access rights for different systems and persons accessing the archive.

Packaging Describes the way an observation, or a set of observations, are actually delivered when the archive is queried.

We will describe these classes in the following sections, and we will leave their exact implementation details to specific chapters.

5.3 Observation and ObsData

The Observation class is the root class of the data model. It describes an arbitrarily large dataset that can be derived directly from an observation, or from several observations. Generally, we will consider an Observation as the set of data recorded with the same instrumental setting, and with the same associated target during a continued period of time.

On the other hand, the ObsData class is a proxy class representing the actual data, and holding a reference that the archive interface can use to retrieve or provide science-ready data files.

The Observation class, then, is used to link together the different aspects of the observation, from the originating proposal (Curation) to the reduced data (ObsData), and taking into account also the data Characterisation, the data access Policy, etc.

However, a direct link is also preserved between the ObsData and the Characterisation, so that if data is repackaged the mapping to the Characterisation metadata can be preserved.

The Observation class, then, only needs a unique identifier for each unique observation, what brings to the table what is the minimum observation (in practice, ObsData) to be collected by the RADAMS. We will answer that in the next chapter.

5.4 Characterisation

The Characterisation class provides the quantitative and qualitative description of the observational data in multiple axes. This information is stored conforming to the most current CharDM IVOA Recommendation [67].

As seen in the previous chapter, we can consider that any astronomical measurement occupies a position in a multi-dimensional space, defined by several axes. Initially, this axes are just four (but more can be added): spatial (coordinates), temporal, spectral, and a fourth axis corresponding to the observed physical quantity (e.g., measured flux, or polarisation, in the case of radio observations). Characterisation gives us a description of the data in different attributes and levels of detail inside this multi-dimensional space, including physical units and scales.

The Characterisation class for each individual axis can be divided in the following subclasses:

Coverage Specifies which part of the multi-dimensional space has been encompassed by the measurement. That is, when was the observation made and for how long, which field was covered, which bands were studied, what was the range of observed flux, and so on. Each of those questions belongs to a different axis.

Resolution Specifies data resolution in each of the axes. Resolution is independent of the sampling, as it is a property of the instrument/observing process.

Precision If the data are sampled in any axis (e.g., for spectral data in the frequency domain, data are sampled by means of filter banks, or by FFT from an autocorrelation signal), this class will include information on the sampling precision. This is different from resolution: resolution is a property of the instrument, due to uncertainties on the energy distribution of the source, because of convolution of the source's and the instrument's profiles.

Accuracy Specifies the minimum error rate achievable for each axis, both from systematic and random errors.

From this group we can see that for Coverage, Resolution and Precision information can be defined *a priori* for a whole observing program (in the Characterisation of a Registry entry, for instance), but they can be different for each actual observation dataset, and those will be the ones stored by the RADAMS.

In contrast, Accuracy information needs to be evaluated *a posteriori*: the lower bounds of accuracy can be known from instrument calibration and knowledge of the observation process, but for some axes assessing the accuracy actually achieved needs access to historical information.

5.5 Target/Field

As we have mentioned several times, astronomers study most of the time objects they already know, while other times they probe a certain part of the

sky to see if there is something new, or even survey a whole region in order to later exhaustively study that region and all objects they can find in it.

This class allows the RADAMS to include information about observed targets. Such target can be mapped to known astronomical objects, to detected sources, to a given field, et cetera. Having this entity in a separate class from the Observation one allows to perform queries by Target, and the ability to link external target information using VO services (such as Sesame, Aladin, the SkyNode interface, et cetera).

In order to be linked with ObsData and Observation classes, the unique identifier for the observation is stored, which allows for queries on given targets to return observations, but also for observation sets to retrieve Target information.

5.6 Provenance

This class groups all the information describing the way an astronomical dataset was originated. It has to include the instrumental setting for that particular observation, coordinates for the telescope, weather and atmospheric conditions, and all the information about data pre- and post-processing, when such data were created by processing already existing datasets³.

Again, the link with ObsData is the unique observation identifier, but for some of the information, which are recorded automatically by the engineering data collection systems, time-stamps will need to be used to bound the relevant information (for instance, focus and pointing calibration observations).

5.7 Policy

In astronomy, observation data are the property of the investigator having suggested and planned the observation, for a period which in some cases depends completely on the observer, which might decide never to release to the public the observations he made, and in other cases depend on policies implemented by the organisation providing the observing facility. Typical cases are 12 to 24 months of proprietary data from the moment the observation was performed to the moment the observation is made available to the general community.

Some observatories have policies where different access rights are available depending on the country or the organisation a particular investigator belongs to, due to different agreements between the hosting organisation and data requesting parties.

³This is already the case for reduced data, which makes extensive use of calibration information, background models and measurements, et cetera.

This class allows both for the tagging of the observation data, and for establishing the data access rights for datasets depending on the Policy information for the dataset itself, and for the user.

Examples of widely different data access policies can be found in archives such as the European Southern Observatory Archive⁴ (ESO). A more general view on scientific data access policies can be found on the Committee on Data for Science and Technology (CODATA) of the International Council for Science (ICSU) on their Scientific Data Policy Statements website⁵.

In order to accommodate all these different kinds of data access policies, we have chosen a role-based approach, where different permissions are allowed not to each different user, but regarding the role that user has against a particular piece of information. This mechanism will be discussed in detail in chapter 7.2.

5.8 Curation

To be considered part of the VO community, all VO resources have to be included in a Registry. The Curation class encompasses all the metadata needed for well-formed archive and data VO registry. In addition, we also integrate an ObservingProgram subclass, proposed by Anita Richards in the ObsDM document [65], which acts as an intermediate class that allows grouping together different observations with a common goal, such as a any kind of survey, or follow-up observing program, for instance.

Curation information is generated first for the whole archive, and later on different observations can be associated (via a Packaging different from the default for the archive) with a different curator, so that they can either be properly registered as a different resource, or at least downloaded, packaged data, can refer to the curator for that particular collection.

5.9 Packaging

Archive queries result in different datasets, and a particular dataset for a given Observation can contain several measurements. The Packaging class specifies what is being delivered by the archive as a result for a given query, and the organisation of data packages different VOTables link to. We will provide an initial development of this class, suitable for the needs of the RADAMS. We will also propose a VO general packaging system, the VOPack.

⁴The archive of the ESO, for instance, makes header information (including pointing) available immediately after observation, together with calibration data. Actual science data are only released after a proprietary period (which can be extended under demand), together with observation proposal abstracts. Their policy can be found at <http://archive.eso.org/cms/eso-data-access-policy>

⁵http://www.codata.org/data_access/policies.html

The principles for the VOPack are providing an off-line mechanism for assessing the main properties applicable to the whole package of datasets: complete Characterisation of Coverage, Resolution, and Precision in all relevant axes, together with common Curation, Policy, Targets and Data Provenance.

5.10 Conclusions

In this chapter we have shown that a complete version of the ObsDM can be used as a blueprint for building astronomical archives, making easier the task of building VO services publishing the archive assets.

In a high-level overview, the needs for radio astronomy are quite similar to the needs of astronomy in general: data need to be accessed by coordinates, targets and then spectral and temporal filters applied. We have also shown that the differences for radio astronomical observations lie in the Characterisation part, specially in Spectral coverage, and also in spatial resolution; and in the Provenance class, which deals with the details on how was the observation observed.

With those specifics in mind, we have built the RADAMS, a Radio Astronomical DATA Model for Single-dish observations, which will be used both as a valid blueprint for building a radio astronomical archive from scratch, by virtue of its implementation of the ObsDM and the CharDM, and the specification of yet to be defined by the IVOA data models within the ObsDM.

In particular, the data models receiving the most attention from the RADAMS are:

Characterisation Need to characterise radio astronomical observations from their particular properties.

Provenance The origin of the observation in radio astronomy is fundamentally different from that of optical observations, as the optical path⁶ and detectors are completely different in nature.

Packaging This model, in principle, has no radio astronomical specifics, but it has been developed as a way to provide different kinds of associated observations together. In a sense, is a complement to the SpecDM (and the ObsDM) lack of a way to declare that several observations are not to be independently considered, but that instead form a coherent unit for scientific analysis.

⁶Even for non-optical (or non-visible) electromagnetic radiation the path followed by photons inside a photon collection instrument is called the optical path. The means for making photons of different wavelengths follow a particular optical path is, as it can be imagined, dependent of the wavelength range.

Target Again, this model should not be, in principle, specific to radio astronomy. It has been defined in a way as general as possible as to be used for the SpecDM, the ObsDM, and also be able to host information from services such as Simbad, VizieR, and NED, dealing with compilations of information for named/catalogued sources.

As a result, we will provide specific guidance for the serialisation of radio astronomical observations in the next chapter, while we will provide additional details on the Curation, Packaging and Policy data models on chapter 7, leaving the subject of data provenance to chapters 8 and 9.

Chapter 6

RADAMS: Characterising radio astronomical observations

In the previous chapter we analysed how an astronomical archive could be built using the basis of the ObsDM, if the missing data models suggested by the ObsDM document were to be implemented. We also illustrated how VO compatibility could be added to archives built on such a basis much more easily.

After that, we showed the high level overview of the RADAMS, which was at first glance almost indistinguishable from the ObsDM, something which is indeed a feature of the RADAMS. Finally, we provided a first overview of the definition of each particular class and sub-models, and the way they mesh together.

Once we have introduced the RADAMS, we will start studying it in detail in this and the following chapters. In particular, this chapter will be devoted to how the RADAMS stores the Observation and ObsData information, and how the CharDM data model is filled for the different observation modes in radio telescopes.

In addition, in those chapters devoted to the detailed exploration of the RADAMS, we will select appropriate FITS Keywords and UCDS for FITS and VOTable serialisations of observational data. FITS Keywords will be selected first from the official FITS mandatory headers [74]. If no mandatory keyword exists, they will be chosen from the Multi-Beam FITS Raw Data Format [24], and lastly from the NRAO GBT FITS data format [75]¹. Where `assign` appears, it means that a suitable FITS keyword has yet to be selected for that particular database attribute.

UCDS will be selected from the UCD1+ vocabulary [76], using the recommended juxtaposition technique in order to clarify the meaning of any given term. In some cases, there are no existing UCD atoms, and no UCD

¹Sometimes those keywords cannot appear in the main header of a FITS file, but instead in a FITS extension table, but that will not be specified.

atom combination, properly describing the attribute. In those cases, we will propose corresponding UCD unique atoms, for stand-alone or combined use.

We also want to acknowledge the initial effort by Lamb and Power in creating a draft for an IVOA Note [68] in which radio astronomical data were initially modelled. Many of the controlled vocabularies proposed in this and the following chapters have used that document as a source.

6.1 ObsData and Characterisation

Following the IRAM MultiBeam FITS definition document [24], based on the ALMA Software Glossary [77], we can define several degrees of observational data in radio astronomy:

Dump This is the smallest interval of time for which a set of correlated data can be accumulated and output from the backend stage.

Integration Set of dumps, all identical in configuration, which are accumulated and form the basic recorded unit.

Sub-scan Set of integrations performed while the antennas complete an elemental pattern across the source: e.g., an azimuth displacement on a pointing source, a focus displacement within a focus calibration, et cetera.

Scan Set of sub-scans with a common goal, such as a pointing scan, a focus scan or an atmospheric amplitude calibration scan, among others.

Hence, the minimum useful unit to be recorded in the archive is the sub-scan, which can be appropriately described, but the minimum scientifically significant unit is the scan, so depending on the kind of granularity desired one could base the archive upon one or the other. Higher order units—such as multiple scans on the same source, for enhancing the signal-to-noise ration, or scans to different sources belonging to the same observing program—should be derived from Project metadata.

To link together these scans with the rest of the data model we will use the `ObsData` class (which could also be referred to as `RawData`) as the root class for the RADAMS. From `ObsData` instances, we will form a tree of data that will describe a particular scan, both by its properties in the Spatial, Temporal, Spectral, and Observable axes, and its respective Accuracy.

We show a high level overview of the `ObsData` class and its relationships with `AxisFrame`, `Coverage`, `Resolution`, and `Accuracy` classes in figure 6.1. Compare it with figures 4.2 and 4.3.

ObsData Instances of this class (also to be referred as `RawData`) will contain the final data. As discussed previously, we will keep either whole scans to determine the scientific data, or processed data (such as spectra).

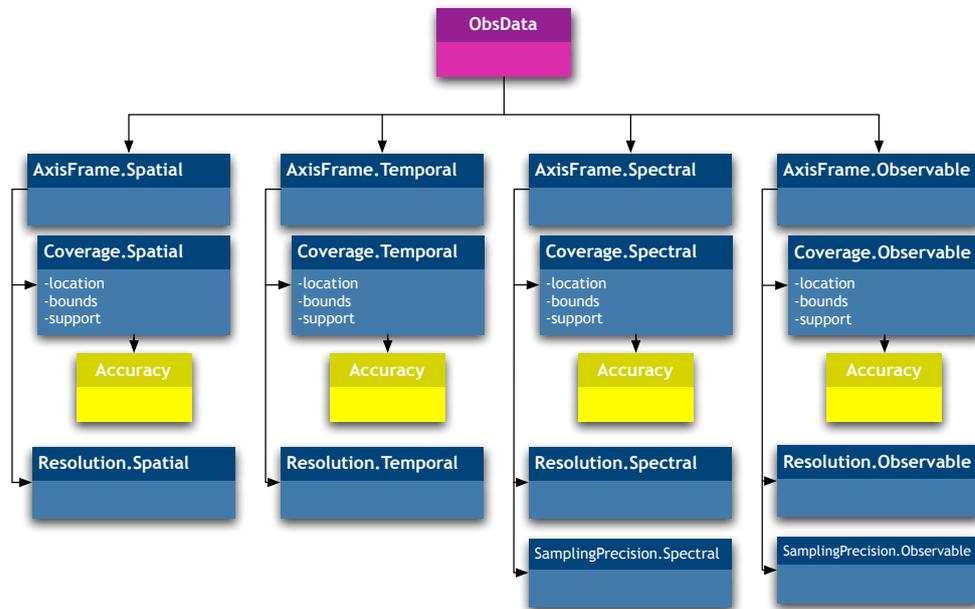


Figure 6.1: ObsData class data model; the different axes for the Characterisation part of the data model are shown.

AxisFrame Instances of this class will contain metadata that describe properties of the axis, such as units, calibration state, et cetera.

Coverage We will describe the position of the archived data in the parameter space: where and when was the telescope pointed, and what wavelength range was observed. Several subclasses exist:

Location This subclass of each Coverage axis describes the characteristic value for each of them. For instance, in the spatial axis Coverage.Spatial.Location would be set to the central point of the observed field, and for the temporal axis Coverage.Temporal.Location would hold either the middle or the start time of the scan.

Bounds Maximum and minimum values of the axis; for instance, Coverage.Spectral.Bounds would give us the maximum and minimum frequencies of the spectrum, and Coverage.Temporal.Bounds would give us the starting and ending time of the observation.

Support Set of parameters in that axis where we have valid observational data. For instance, in the temporal axis Coverage.Temporal.Support could be a set of intervals when data were gathered, excluding the pauses for reissuing scans.

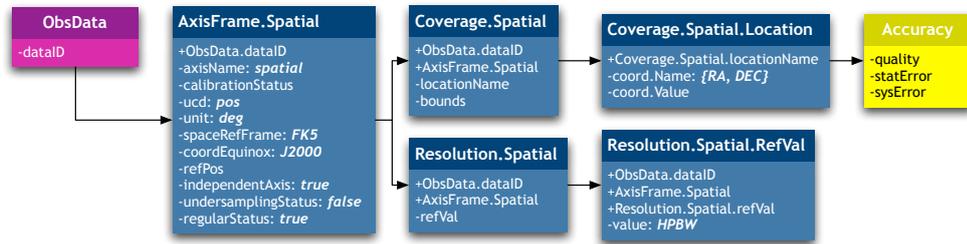


Figure 6.2: Spatial axis frame and coverage metadata.

Sensitivity Sensitivity is an additional refinement to Support, where for the sections of the axis with Support for the observation a response function of the instrument in the given axis is provided. This is especially useful for cases with a large number of small interruptions in the data, there is need for data resampling, filter profiles have to be accounted for, and so on.

We have to remember that these subclasses form a hierarchy by which Location is mandatory, and all further refinements optional, but for a refinement to appear all levels above it must appear to: for instance, we can provide just Location, but if Support appears, Bounds needs to appear as well.

Resolution Instances of this class are used to describe resolution in each axis. For instance, Coverage.Spatial.Resolution would be defined in single-dish observations by the Half-Power Beam Width (HPBW).

SamplingPrecision For a sampled axis—such as the Spectral axis, in the case of spectroscopic data—an instance of this class will hold sampling precision, described as a pixel scale.

Accuracy All of the aforementioned classes should have an accompanying class in order to describe errors and data quality for each axis. In the case of archives without data mining capabilities, we only have to provide Accuracy instances related to the Coverage classes.

AxisFrame.Spatial and Coverage.Spatial

In this subsection, we describe the classes that configure the description of the spatial axis (AxisFrame.Spatial), and the characterisation of the coverage in such axis (Coverage.Spatial). Figure 6.2 shows the classes and their relationships.

AxisFrame.Spatial This instance describes the properties of the Spatial axis, such as calibration status, units (normally, degrees), reference frame, epoch, spatial sampling type and sampling status, et cetera.

For the DSS-63 antenna, the spatial axis is never sampled (because we are not storing maps at this stage), and each scan only needs two pairs of coordinates (plus pointing accuracy) to describe the antenna-pointing pattern.

As the coordinate space is in this case bi-dimensional, and properties for one of the coordinates do not have to be equal for the other, we should either choose between a vector approach, using two dimensional arrays of coordinates, or splitting the Spatial classes in different subclasses. We choose the first approach, and thus all AxisFrame.Spatial attributes will have a space-separated array of values (just two values separated with an space, for a bi-dimensional coordinate space).

If the archive were to be exploited in the third dimension, an additional distance and/or redshift coordinate should be entered. As we have seen, extension in the spatial axis is straightforward. A second option would add an additional axis to the Characterisation data model.

Coverage.Spatial.Location For the DSS-63 antenna, the stored values for this class correspond to the antenna pointing coordinates. If a Target is linked, Spatial.Location information could be the same, unless the target is not directly associated, but found by cross-correlation with the Targets database.

Coverage.Spatial.Bounds Bounds for spatial data would be the maximum and minimum for spatial coordinates (normally, right ascension and declination) when observing the source. For single point spectroscopic data, this class is not needed, as the antenna does not describe any path across the source.

Coverage.Spatial.Support Typically, the stored values for this class should be equal to those of Coverage.Spatial.Bounds, except when invalid data within Coverage.Spatial.Bounds might exist, or in order to describe an elaborate scanning path on the source. This class would describe such a path.

Coverage.Spatial.Sensitivity In the case of the DSS-63 antenna, Spatial.Sensitivity would be defined as the beam pattern of the antenna, in principle in 1D taking symmetries into account, but 2D could be considered, as it would be the case for a synthesised beam in interferometry. Another possibility is the combination of beam pattern with receiver efficiency at different elevations.

Table 6.1: AxisFrame.Spatial metadata.

Attribute	FITS Keyword	UCD	Description
axisName	assign	meta.id; meta.main	Axis name.
calibrationStatus	assign	obs.calib; meta.code	Calibration status from a controlled vocabulary: uncalibrated, calibrated, relative ^a , normalized ^b .
ucd	assign	meta.ucd; meta.main	Main UCD for the axis.
unit	assign	meta.unit; meta.main	Main units for the axis.
refPos	assign	meta.ref; meta.id	Identification of the origin position within the spaceRef-Frame from a controlled vocabulary; See Space-Time Coordinate Data Model [78].
spaceRefFrame	WCSNAME or RADESYS	pos.frame; meta.id	Identification of the reference system from a controlled vocabulary; see Space-Time Coordinate Data Model [78]: FK4, FK5, ELLIPTIC, et cetera.
coordEquinox	assign	pos; time.equinox	Equinox (only if needed).
epoch	assign	pos; time.epoch	Epoch (only if needed).
independentAxis	assign	pos; obs.param; meta.code	Boolean flag indicating whether the axis is independent of the rest or not.
undersamplingStatus	assign	pos; obs.param; meta.code	Boolean flag indicating whether the data are sampled in this axis or not.
regularStatus	assign	pos; obs.param; meta.code	Boolean flag used in case of sampled data, indicating whether sampling is regular or not.

^arelative refers to calibrated data, except for an additive or multiplicative constant.

^bnormalized refers to dimensionless quantities, such as those resulting from the division between two commensurable datasets.

Table 6.2: Coverage.Spatial.Location metadata.

Attribute	FITS Keyword	UCD	Description
coord.name	assign	pos; meta.name	Name of the coordinate whose value goes in coord.value.
coord.value	assign	pos; meta.number	Numeric value for the coordinate coord.name.

Table 6.3: Coverage.Spatial.Bounds metadata.

Attribute	FITS Keyword	UCD	Description
coord.name	assign	pos; meta.name	Name of the coordinate whose value goes in coord.value.
coord.maxValue	assign	pos; meta.number; stat.max	Minimum numeric value for the coordinate coord.name.
coord.minValue	assign	pos; meta.number; stat.min	Maximum numeric value for the coordinate coord.name.

Table 6.4: Coverage.Spatial.Support metadata.

Attribute	FITS Keyword	UCD	Description
code	assign	pos; meta.name	Code for the interval where we will be defining support; an array of [coord.code, startValue, endValue] tuples can be used to define spatial support.
startValue	assign	pos; meta.number	2D start value.
endValue	assign	pos; meta.number	2D end value.

Table 6.5: Coverage.Spatial.Sensitivity metadata^a.

Attribute	FITS Keyword	UCD	Description
theta[n]	assign	pos.posAng	Theta angle for the n th beam pattern normalised response.
phi[n]	assign	pos.posAng	Phi angle for the n th beam pattern normalised response.
response[n]	assign	arith.factor	N th normalised beam pattern response.

^aSymmetrical beam patterns could be defined just in one dimension, with pairs of [theta, response] values.

Table 6.6: Coverage.Spatial.Resolution metadata.

Attribute	FITS Keyword	UCD	Description
referenceValue	assign	pos.angResolution	Resolution reference value.

AxisFrame.Temporal and Coverage.Temporal

In this subsection, we describe the classes that configure the description of the temporal axis (`AxisFrame.Temporal`), and the characterisation of the coverage in such axis (`Coverage.Temporal`). Figure 6.3 shows the classes and their relationships.

AxisFrame.Temporal Describes the properties of the time axis, such as whether the axis is calibrated or not (`calibrationStatus`), units, reference system (`refPos`), time-scale (`timescale`), whether the axis is sampled or not (`undersamplingStatus`), and if sampled, whether sampling is regular or not (`samplingStatus`). In the case of spectroscopic data, time is not sampled.

We will use the `independentAxis` attribute to signal axis dependency as true for continuum observations.

Coverage.Temporal Collects all the temporal characterisation of the observational data.

Coverage.Temporal.Location Location instances hold the most representative value for each axis. For the temporal axis, it holds the time of observation, either the starting time or the time at the middle of the observation.

Table 6.7: Accuracy.Spatial metadata.

Attribute	FITS Keyword	UCD	Description
quality	assign	pos; meta.code.qual	Quality code for spatial coordinates; invalid data are flagged with a quality code of 1.
sysError	assign	pos; stat.error.sys	Systematic error for spatial coordinates.
sysErrorHigh	assign	pos; stat.error.sys; stat.max	Maximum systematic error for spatial coordinates.
sysErrorLow	assign	pos; stat.error.sys; stat.min	Minimum systematic error for spatial coordinates.
statError	assign	pos; stat.error	Statistical error for spatial coordinates.
statErrorHigh	assign	pos; stat.error; stat.max	Maximum statistical error for spatial coordinates.
statErrorLow	assign	pos; stat.error; stat.min	Minimum statistical error for spatial coordinates.

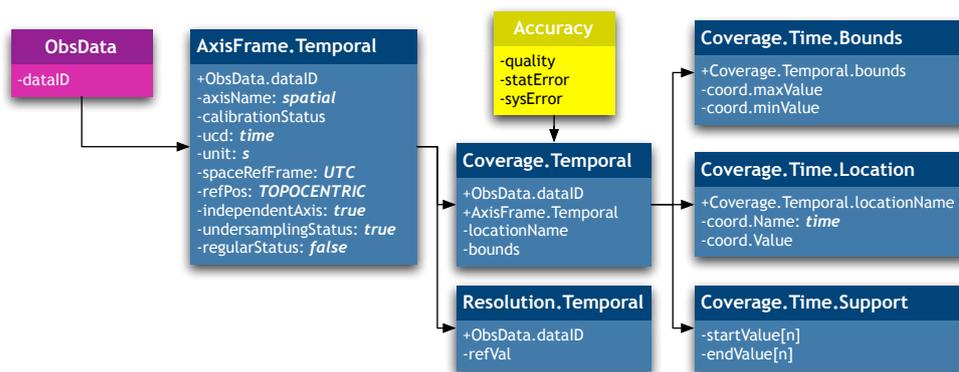


Figure 6.3: Temporal axis and related coverage metadata.

Coverage.Temporal.Bounds Bounds instances hold the maximum and minimum values for the axis. In the temporal axis, this is defined as the complete observation interval; alternatively, the total integration time can be used.

Coverage.Temporal.Support Coverage.Support instances provide the regions of an axis holding observational data. In the temporal axis, this are the time intervals actually devoted to flux collection².

Coverage.Temporal.Sensitivity In cases where an instrument had a sensor ramp-up, where some time is needed before 100% sensitivity is achieved, this class would describe such a ramp, and would allow for convenient scaling of data taken during those intervals. If that were never the case, Temporal.Sensitivity could just be dropped.

Resolution.Temporal Holds the temporal resolution of the data. Most of the time, telescope control system time-stamps are expressed in UTC or LST in seconds, so temporal resolution would be of the order of a second, but normally the instrument could deliver better temporal resolution. We should address this after the initial setup.

We will not develop the SamplingPrecision.Temporal class, because we are not sampling the temporal axis.

Accuracy.Temporal This class has to collect systematic and statistical errors associated with the temporal coordinates of the data. The Quality attribute gives additional information about the temporal data quality. We could use time re-syncing statistics in order to evaluate accuracy.

AxisFrame.Spectral and Coverage.Spectral

In this subsection, we describe the classes that configure the description of the spectral axis (AxisFrame.Spectral), and the characterisation of the coverage in such axis (Coverage.Spectral). Figure 6.4 shows the classes and their relationships.

AxisFrame.Spectral Describes the properties of the spectral axis, such as whether the axis is calibrated or not (calibrationStatus), units, reference system (RefPos), or the mathematical definition used for the Doppler effect (DopplerDef), for velocity calibrated spectra³. It is clear that

²Actual execution time for each individual sub-scan.

³For systems in the local standard of rest (LSR), emission lines occur at a precise frequency. Due to the Doppler effect, systems receding from the LSR will show a lower frequency, and those approaching will show higher frequencies. In the case of complex dynamics, the net effect is a broadening of the line, and the Doppler effect allows us to calibrate emission at different frequencies as emission at different velocities.

Table 6.8: AxisFrame.Temporal metadata.

Attribute	FITS Keyword	UCD	Description
axisName	assign	meta.id; meta.main	Axis name.
calibrationStatus	assign	time; obs.calib; meta.code	Calibration status from a controlled vocabulary: uncalibrated, calibrated, relative ^a , normalized ^b
ucd	assign	time; meta.ucd; meta.main	Main UCD for the axis.
unit	assign	time; meta.unit; meta.main	Main units for the axis.
refPos	assign	time; meta.ref; meta.id	Identification of the origin position from a controlled vocabulary.
independentAxis	assign	time; obs.param; meta.code	Boolean flag indicating whether the axis is independent of the rest or not.
undersamplingStatus	assign	time; obs.param; meta.code	Boolean flag indicating whether the data are sampled in this axis or not.
regularStatus	assign	time; obs.param; meta.code	Boolean flag used in case of sampled data, indicating whether sampling is regular or not.
numBins	assign	time; meta.number	Number of time samples.

^arelative refers to calibrated data, except for an additive or multiplicative constant.

^bnormalized refers to dimensionless quantities, such as those resulting from the division between two commensurable datasets.

Table 6.9: Coverage.Temporal.Location metadata.

Attribute	FITS Keyword	UCD	Description
coord.name	assign	time; meta.name	Name of the coordinate whose value goes in coord.value; in this case, time with respect to refPos.
coord.value	assign	time; meta.number	Numeric value for the time location; usually, a MJD or decimal UTC time.

Table 6.10: Coverage.Temporal.Bounds metadata^a.

Attribute	FITS Keyword	UCD	Description
coord.name	assign	time; meta.name	Name of the coordinate whose maximum and minimum values go in coord.maxValue and coord.minValue.
coord.maxValue	assign	time; meta.number; stat.max	Minimum numeric value for the observation time.
coord.minValue	assign	time; meta.number; stat.min	Maximum numeric value for the observation time.

^aThe UCDS for coord.maxValue and coord.minValue could have been, respectively, time.obs.start and time.obs.end, but we think the proposed UCDS are more consistent with the rest of the axes. Besides, we can defer the election, by using time.obs.start and time.obs.end at the beginning of the UCD, and appending meta.number; stat.max or meta.number; stat.min as additional qualifiers.

spectral information is indeed sampled, and the number of channels will be coded by numBins.

Coverage.Spectral Collects the different spectral properties of the data.

Coverage.Spectral.Location Coverage.Location holds the most representative value for the axis. In the spectral axis, and being a sampled axis, we will use the frequency for the central sample of the spectrum.

Coverage.Spectral.Bounds Holds the spectral limits of the data, that is, the starting and ending frequency for the spectrum.

Coverage.Spectral.Support Holds the spectral support for the data. We could choose to register just the actual bandwidth, or better, to set

Table 6.11: Coverage.Temporal.Support metadata^a.

Attribute	FITS Keyword	UCD	Description
code	assign	time; meta.code	Code for the time interval where we will be defining support; an array of [coord.code, startValue, endValue] tuples can be used to define temporal support.
startValue	assign	time.expo.start	Time interval start value.
endValue	assign	time.expo.end	2DTime interval end value.

^aCoverage.Temporal.Support could be alternatively defined by using just one value, the exposure time, with UCD `time.expo`. The chosen definition, apart from being more consistent across axes, allows for discontinuous temporal support.

Table 6.12: Coverage.Temporal.Resolution metadata.

Attribute	FITS Keyword	UCD	Description
referenceValue	assign	time; meta.number; meta.ref	Resolution reference value.

an array of different intervals where the instrument is sensitive. Better yet, we could register a sensitivity profile for each frequency. We will initially choose the array of intervals as the way to express the spectral support.

Coverage.Spectral.Sensitivity Holds the sensitivity profile for the instrument in frequency; in other words, this would be equal to the detailed frequency response of the whole antenna-receiver-backend set.

Resolution.Spectral Holds the spectral resolution information for the data. For filter banks, it is the filter bandwidth for each filter, or an average filter bandwidth. In the case of spectra obtained from the Fourier transform of the autocorrelation function, it is equal to the bandwidth divided by the number of channels.

SamplingPrecision.Spectral In the case of the Robledo antenna, this value should be equal to the value stored at Resolution.Spectral. For other instruments, SamplingPrecision.Spectral defines the frequency steps from one frequency bin to the next.

Table 6.13: Accuracy.Temporal metadata.

Attribute	FITS Keyword	UCD	Description
quality	assign	time; meta.code.qual	Quality code for time coordinates; invalid data are flagged with a quality code of 1.
sysError	assign	time; stat.error.sys	Systematic error for time coordinates.
sysErrorHigh	assign	time; stat.error.sys; stat.max	Maximum systematic error for time coordinates.
sysErrorLow	assign	time; stat.error.sys; stat.min	Minimum systematic error for time coordinates.
statError	assign	time; stat.error	Statistical error for time coordinates.
statErrorHigh	assign	time; stat.error; stat.max	Maximum statistical error for time coordinates.
statErrorLow	assign	time; stat.error; stat.min	Minimum statistical error for time coordinates.

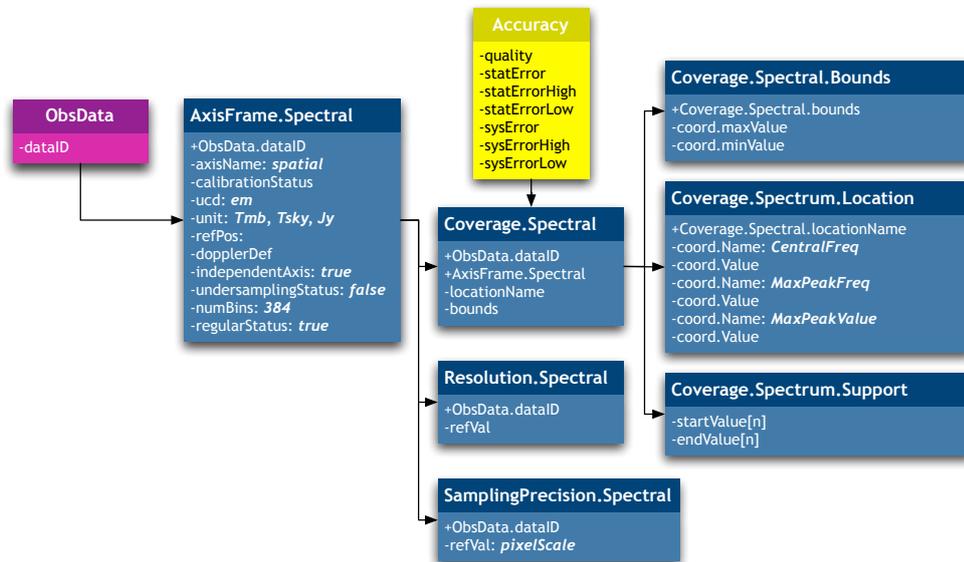


Figure 6.4: Spectral axis and related coverage metadata.

Accuracy.Spectral Errors associated with the spectrum. In the same way as the support could change along the spectrum, because of different sensitivities, we might need to characterise the accuracy along the spectrum. At least in the case of Robledo, the SNR is global, and considered equal for all frequencies. If not, a first approximation can be using an average SNR, and maximum and minimum SNR.

AxisFrame.Observable and Coverage.Observable

The Observable axis is, finally, the one directly related to the stored data, instead of data headers. In this subsection, we describe the classes that configure the description of the observable axis (`AxisFrame.Observable`), and the characterisation of the coverage in such axis (`Coverage.Observable`). Figure 6.5 shows the classes and their relationships.

AxisFrame.Observable Describes the properties of the observable axis (whether the observable is flux, polarisation state, or any other), such as whether the data is calibrated or not (`calibrationStatus`), applicable units, et cetera. Flux information is sampled at the digital conversion stage, and the number of available different flux levels is codified in `numBins`. By specifying units in this class, we define whether the observable data are provided in flux units, or in brightness or antenna temperature. Additional information, such as the polarisation, can be accommodated in the same axis in the same way we extended the Spatial axis for 2D or 3D coordinates, or by means of (an) additional observable (axis) axes.

Coverage.Observable Collects all information that directly characterises observable data.

Coverage.Observable.Location `Coverage.Location` instances hold the most representative value for that axis. In this case, the average spectrum flux is not significant, specially if we have a strong line detection. Hence, for this axis we will take the maximum flux as `Observable.Location`.

Coverage.Observable.Bounds `Coverage.Bounds` instances record the maximum and minimum values for the axis. In the Observable axis, it records minimum and maximum flux for the spectrum.

Coverage.Observable.Support This instance should give us the intervals of flux observed in the spectrum, and we could choose between making it equal to `Coverage.Observable.Bounds`, or defining an interval that holds a given dispersion from the `Coverage.Observable.Location` value. We will initially consider `Coverage.Observable.Support` as equal to `Coverage.Observable.Bounds`.

Table 6.14: AxisFrame.Spectral metadata.

Attribute	FITS Keyword	UCD	Description
axisName	assign	meta.id; meta.main	Axis name (frequency or velocity).
calibrationStatus	assign	em.radio; obs.calib; meta.code	Calibration status from a controlled vocabulary: uncalibrated, calibrated, relative ^a , normalized ^b .
ucd	assign	em.radio; meta.ucd; meta.main	Main UCD for the axis.
unit	TUNITn	em.radio; meta.unit; meta.main	Main units for the axis.
refPos	assign	em.radio; meta.ref; meta.id	Identification of the origin position from a controlled vocabulary.
dopplerDef	VELDEF	spect.dopplerParam; meta.code	Code defining the type of Doppler shift definition used for frequency and/or velocity calibration; from a controlled vocabulary: optical, radio, relativistic.
independentAxis	assign	em.radio; obs.param; meta.code	Boolean flag indicating whether the axis is independent of the rest or not.
undersamplingStatus	assign	em.radio; obs.param; meta.code	Boolean flag indicating whether the data are sampled in this axis or not.
regularStatus	assign	em.radio; obs.param; meta.code	Boolean flag used in case of sampled data, indicating whether sampling is regular or not.
numBins	assign	em.radio; meta.number	Number of spectral samples.

^arelative refers to calibrated data, except for an additive or multiplicative constant^bnormalized refers to dimensionless quantities, such as those resulting from the division between two commensurable datasets.

Table 6.15: Coverage.Spectral.Location metadata.

Attribute	FITS Keyword	UCD	Description
coord.name	assign	em.radio; meta.name	Name of the coordinate whose value goes in coord.value; in this case, frequency with respect to refPos.
coord.value	OBSFREQ	em.radio; em.freq; stat.mean	Numeric value for the central frequency location.

Table 6.16: Coverage.Spectral.Bounds metadata.

Attribute	FITS Keyword	UCD	Description
coord.name	assign	em.radio; meta.name	Name of the coordinate whose maximum and minimum values go in coord.maxValue and coord.minValue.
coord.maxValue	assign	em.freq; meta.number; stat.max	Minimum frequency value.
coord.minValue	assign	em.freq; meta.number; stat.min	Maximum frequency value.

Table 6.17: Coverage.Spectral.Support metadata.

Attribute	FITS Keyword	UCD	Description
code	assign	em.radio; meta.record	Code for the interval where we will be defining support; an array of [coord.code, startValue, endValue] tuples can be used to define spectral support.
startValue	assign	em.freq; stat.min	Frequency interval start value.
endValue	assign	em.freq; stat.max	Frequency interval end value.

Table 6.18: Coverage.Spectral.Sensitivity metadata.

Attribute	FITS Keyword	UCD	Description
numChannels	assign	em.radio; meta.number	Number of channels of the frequency response of the filter that describes spectral sensitivity.
channel[n]	assign	em.freq	Frequency for the n th channel of the filter that describes spectral sensitivity.
response[n]	assign	arith.factor	Filter response for the n th channel.

Table 6.19: Coverage.Spectral.Resolution metadata^a.

Attribute	FITS Keyword	UCD	Description
numChannels	assign	em.radio; meta.number	Number of channels for which resolution is provided.
referenceValue[n]	FREQRES	em.freq; spect.resolution	Resolution reference value for the n th channel.

^aWhen the Sensitivity class is provided, Resolution.numChannels has to be equal to Sensitivity.numChannels, and each channel can support different resolutions. When the Sensitivity class is not provided, Resolution.numChannels has to be set to 1, and Resolution.referenceValue represents the average channel resolution.

Table 6.20: SamplingPrecision.Spectral metadata^a.

Attribute	FITS Keyword	UCD	Description
numChannels	assign	em.radio; meta.number	Number of frequencies that we are sampling.
referenceValue[n]	FREQRES	em.freq; spect.resolution	Resolution reference value for the n th channel.

^aWhen SamplingPrecision.numChannels is set to 0, the number of spectral channels is given by AxisFrame.Spectral.numBins, and SamplingPrecision provides the sampling step in a single referenceValue. Otherwise, SamplingPrecision.Spectral.numBins must be equal to AxisFrame.Spectral.numBins, and when the Sensitivity class is provided, SamplingPrecision.Spectral.numBins must be equal to Spectral.Sensitivity.numChannels, and Spectral.SamplingPrecision.referenceValue[n] equal to Spectral.Sensitivity.channel[n].

Table 6.21: Accuracy.Spectral metadata.

Attribute	FITS Keyword	UCD	Description
quality	assign	em.freq; meta.code.qual	Quality code for frequency coordinates; invalid data are flagged with a quality code of 1.
sysError	assign	em.freq; stat.error.sys	Systematic error for frequency coordinates.
sysErrorHigh	assign	em.freq; stat.error.sys; stat.max	Maximum systematic error for frequency coordinates.
sysErrorLow	assign	em.freq; stat.error.sys; stat.min	Minimum systematic error for frequency coordinates.
statError	assign	em.freq; stat.error	Statistical error for frequency coordinates.
statErrorHigh	assign	em.freq; stat.error; stat.max	Maximum statistical error for frequency coordinates.
statErrorLow	assign	em.freq; stat.error; stat.min	Minimum statistical error for frequency coordinates.

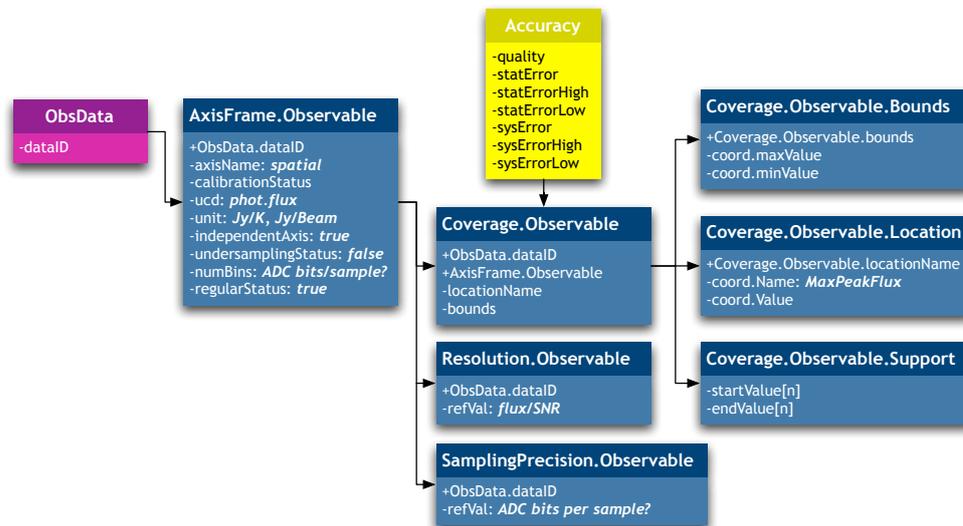


Figure 6.5: Observable axis and related coverage metadata.

Coverage.Observable.Sensitivity This would store the change rate in the observable axis for a given observed flux.

Resolution.Observable We keep the flux resolution for the data, defined as mean flux divided by the SNR, or mean noise.

SamplingPrecision.Observable Instances of this class indicate the sampling precision. For the observable (flux) axis, this should be equal to the Analog to Digital Converter (ADC) resolution, or worse if more processing is involved, and should be tabulated by instrument setup. We might also consider this axis as non-sampled, as `Resolution.Observable` is usually well above ADC resolution, and drop this class.

Accuracy.Observable Collected flux associated errors. Errors on the observable axis can be dependent on received flux, and so we should give an average value, and either variance or maximum and minimum accuracy on the observable axis.

If the observable variable were a different one, the main difference would correspond to the change of UCDs. In the particular case of polarisation, `phot.flux` would be replaced by `phys.polarization`. Other changes would include an additional characterisation of the parameters been actually returned by the instrument, and their relationship with Stokes parameters.

6.2 Target

target

noun

- a person, object, or place selected as the aim of an attack.
- a mark or point at which someone fires or aims, especially a round or rectangular board marked with concentric circles used in archery or shooting.

on target

phrase

- accurately hitting the thing aimed at.

The New Oxford American Dictionary, 2nd Edition

It is necessary to have a systematic storage of the targets being studied, because we have to allow queries by previously studied targets, as it is very likely that this query will be the most used one.

The ObsDM distinguishes between several types of targets:

Table 6.22: AxisFrame.Observable metadata (when the observed variable is flux).

Attribute	FITS Keyword	UCD	Description
axisName	assign	meta.id; meta.main	Axis name.
calibrationStatus	assign	em.radio; obs.calib; meta.code	Calibration status from a controlled vocabulary: uncalibrated, calibrated, relative ^a , normalized ^b .
ucd	assign	phot.flux; meta.ucd; meta.main	Main UCD for the axis.
unit	TUNITn	phot.flux; meta.unit; meta.main	Main units for the axis.
refPos	assign	phot.flux; meta.ref; meta.id	Identification of the origin position from a controlled vocabulary.
independentAxis	assign	phot.flux; obs.param; meta.code	Boolean flag indicating whether the axis is independent of the rest or not.
undersamplingStatus	assign	phot.flux; obs.param; meta.code	Boolean flag indicating whether the data are sampled in this axis or not.
regularStatus	assign	phot.flux; obs.param; meta.code	Boolean flag used in case of sampled data, indicating whether sampling is regular or not.
numBins	assign	phot.flux; meta.number	Number of spectral samples (if the axis is sampled)

^arelative refers to calibrated data, except for an additive or multiplicative constant^bnormalized refers to dimensionless quantities, such as those resulting from the division between two commensurable datasets.

Table 6.23: Coverage.Observable.Location metadata (when the observed variable is flux).

Attribute	FITS Keyword	UCD	Description
coord.name	assign	phot.flux; meta.name	Name of the coordinate whose value goes in coord.value; in this case, flux with respect to refPos.
coord.value	assign	phot.flux; stat.max	Numeric value for the maximum flux (flux location).

Table 6.24: Coverage.Observable.Bounds metadata.

Attribute	FITS Keyword	UCD	Description
coord.name	assign	phot.flux; meta.name	Name of the coordinate whose maximum and minimum values go in coord.maxValue and coord.minValue.
coord.maxValue	assign	phot.flux; stat.max	Minimum flux value.
coord.minValue	assign	phot.flux; stat.min	Maximum flux value.

Table 6.25: Coverage.Observable.Support metadata (when the observed variable is flux).

Attribute	FITS Keyword	UCD	Description
code	assign	em.radio; meta.record	Code for the interval where we will be defining support; an array of [coord.code, startValue, endValue] tuples can be used to define spectral support.
startValue	assign	em.freq; stat.min	Frequency interval start value.
endValue	assign	em.freq; stat.max	Frequency interval end value.

Table 6.26: Coverage.Observable.Resolution metadata.

Attribute	FITS Keyword	UCD	Description
referenceValue	assign	phot.flux; stat.snr; arith.ratio	Flux resolution (Flux/SNR), as $\frac{\text{Flux}}{\text{SNR}} = \text{flux} \times \frac{\text{fluxNoise}}{\text{fluxSignal}}$ Equivalent to average noise.

Table 6.27: SamplingPrecision.Observable metadata^a.

Attribute	FITS Keyword	UCD	Description
referenceValue	assign	phot.flux; instr.precision	Flux sampling precision; maybe expressed as the number of bits per sample.

^aSomewhat redundant, given the presence of AxisFrame.Observable.numBins. However, we include it for completeness.

Astronomical object It is a target for which some properties are known before the observation.

Source An entity created after analysing an observation; represents the actual detection by an instrument.

Field A region of the sky, not a single point or collection of points.

Pointing target A particular location for an observation. It can be part of an Astronomical object, a Field, or other entities, such as a Calibrator.

IVOA has not developed the Target class as a whole, but the authors of the SDM [79] have developed a set of possible metadata. We will initially adopt that set for the RADAMS, and we can see the associated metadata in Figure 6.6.

Another data model dealing with Targets is the Resource Data Model [80], which considers additional targets for observations, such as radio flux calibrators, pointing calibrators, et cetera.

We have to take into account that most of the time the archive will store ObsData associated to Pointing targets, but said Pointing targets can also be related to Astronomical Objects. For instance, an astronomer might have a

Table 6.28: Accuracy.Observable metadata.

Attribute	FITS Keyword	UCD	Description
quality	assign	phot.flux; meta.code.qual	Quality code for the observed flux; invalid data are flagged with a quality code of 1.
sysError	assign	phot.flux; stat.error.sys	Systematic error for the observed flux.
sysErrorHigh	assign	phot.flux; stat.error.sys; stat.max	Maximum systematic error for the observed flux.
sysErrorLow	assign	phot.flux; stat.error.sys; stat.min	Minimum systematic error for the observed flux.
statError	assign	phot.flux; stat.error	Statistical error for the observed flux.
statErrorHigh	assign	phot.flux; stat.error; stat.max	Maximum statistical error for the observed flux.
statErrorLow	assign	phot.flux; stat.error; stat.min	Minimum statistical error for the observed flux.

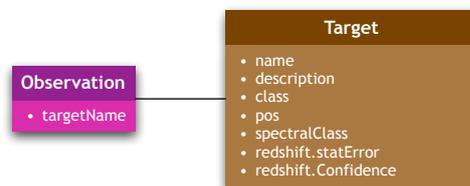


Figure 6.6: Target data model.

program in which s/he is looking for water masers as tracers of shocked regions, and specifies several Pointing targets. Those Pointing targets, however, will be most likely found within an Astronomical object, such as a planetary nebula. The association of the Astronomical object and the Pointing target will be made by means of the Target class.

6.3 Conclusions

In this chapter we have shown that the already specified IVOA data models for Characterisation, together with the Target data model, do not need to be altered in order to be used for radio astronomical archives.

However, the UCDs for data in the ObsDM, CharDM, or in the Target class had never been specified before, and this contributes to the better inter-operation of archives, as data with the same UCDs can be interesting for use in automatic data mining and data discovery.

In the next chapters we will analyse the parts of the RADAMS which need to be created in order to support radio astronomical archives.

Chapter 7

RADAMS: Curation, Packaging and Policy

In this chapter we will follow with our detailed description of the RADAMS, making emphasis on several classes which were introduced in the ObsDM, but which were not actually defined.

These classes are the Curation class, the Packaging class, and the Policy class. There is still another class introduced by the ObsDM pending of definition, the Provenance class, but due to its relevance we will devote two separate chapters to it: one to the concept of Data Provenance in e-Science, and one to the actual implementation of Data Provenance in the RADAMS.

7.1 Curation

curator

noun

- a keeper or custodian of a museum or other collection.

curate

verb [trans.] (usu. **be curated**)

- select, organize, and look after the items in (a collection or exhibition): *both exhibitions are curated by the museum's director.*

The New Oxford American Dictionary, 2nd Edition

If we conceive sets of astronomical observations (for instance, a particular observation program, a whole sky survey, et cetera) as *collections* or *exhibitions*, we can also generalise the VO concept (or any other data based e-Infrastructure) as a special kind of digital *museum*.

In that case, the definitions above point us to the fact that, for digital data collections, Curation is related to:

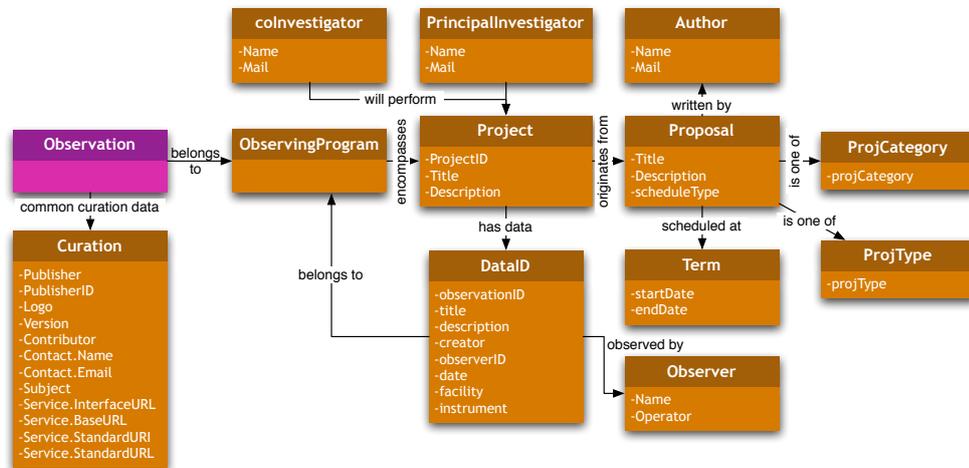


Figure 7.1: Curation data model.

- the final decision on what elements do finally enter the collection, and those who do not; that is, which digital copies of data will be offered as members of a particular dataset.
- the organisation on how the data collection is finally presented; that is, which services will be offered which contain a particular dataset, and which will be the access end-points to them.
- the care-taking of the items put in that collection; that is, keeping the datasets accessible, and updated if the archive admits updating.

So, the most important things the Curation data model has to deal with are grouping together all mandatory metadata for resources to be published in a VO Registry. We will make use of several additional classes in order to deal with Curation specifics.

One of the first sources for data curation in the first place is the *Resource metadata for the VO* [81] IVOA Recommendation. VO Registry entries are actually the *virtual exhibitions* we are exposing by means of VO data access protocols, so being able to know who curates, who is responsible for the *exhibition* is one of the needs for the Registry.

The Curation data model groups all metadata mandatory for resources to be published by a VO Registry. Several additional classes are needed for particular instances of data. Such classes and metadata are described by “Resource Metadata for the VO”, an IVOA Recommendation [80]. Figure 7.1 encompasses all classes related to metadata for curation.

Observation The Observation class was discussed earlier. All observations are related to a single Curation class.

Curation This is the main class that will group all curation related metadata with relationships to complementary classes such as DataID and the rest. It is the same for all data contained in the VO resource (archive) being described.

DataID There is an instance of this class for each different piece of data stored, or at least for each different way to store data. The Spectral Data Model advocates for this separation.

ObservingProgram Instances of this class keep common information about different observations, describing its scientific or technical goals. They also specify which is the project and/or proposal to which this observing program belongs. This class allows for easy querying by project or by common goals, such as maser surveys.

While there is debate about whether this class belongs to the Curation or to the Provenance Data Model, we believe it belongs to the Curation Data Model, as it is an organisational unit, and in our view it has nothing to do with how the data were taken.

Project This class is the main class from an organizational point of view. All observations belong to a single project through a single proposal. A Project instance is related to a collection of Proposal instances.

Proposal As stated in the Project class, a Project can be divided in a series of Proposals. A Proposal instance can be related to one or more observations. No observation can be related to more than one Proposal instance.

Author An Author instance contains information about Proposal authors.

Observer Observer instances represent the observer/s assigned to a Proposal. It also contains a relationship with a Contact acting as the operator for the instrument.

ProjCategory This instance specifies the scientific category of the project, from a controlled vocabulary; possible values, as specified by the Resource Data Model, are: *instrumental, galactic, terrestrial, solarSystem, extragalactic, stellar...*

ProjType This instance specifies the type of observation performed for a particular proposal. Possible values are specified by the Resource Data Model, and form a controlled vocabulary that we have derived from Lamb's, ATNF and NRAO proposals [69, 68, 75]: *astrometry,*

bandwidthSynthesis, circularPolarization, continuum, engineering, fillerTime, highTimeResolution, imaging, instrumental, linearPolarization, mapping, monitoring, mosaic, multibeam, pointSource, phasedArray, polarimetry, snapshot, solar, spectroscopy, survey, timeBinning.

Term Specifies start and end dates (ISO dates, with time stamp) for the proposal *observing term* or *scheduling block*.

7.2 Policy

policy

noun

- a course or principle of action adopted or proposed by a government, party, business, or individual.
- archaic prudent or expedient conduct or action.

The New Oxford American Dictionary, 2nd Edition

The general definition above translates into the VO as *data access policy*, and could be defined as the expedient action of granting or denying access to data following some principles proposed by data providers. Those principles take into account who is responsible for the data generation, for the data curation, and their relationship to the user trying to access the data.

Moreover, those principles (policies) can change from institution to institution, and also for different datasets curated by those institutions. Hence, we need to generalise a way to specify those different policies, both regarding the different users' relationships with the datasets, and the different ways to implement policies, from *everything is accessible*, to *PI's eyes only*.

The role of the Policy data model is to allow for very different policies to be applied to the data. In the case of a VO archive where the data are not immediately available (they have to be manually incorporated to the archive), Policy can become simpler, as in *everything in the archive is available for everyone*. Figure 7.2 shows the different classes needed to characterise the archive policy.

Policy instances attached to raw data shall need to identify the different roles of agents —not just people, but software packages too— accessing the archive, and give them appropriate access rights to pieces of data.

In the case of the Robledo Archive, the policy to be implemented should be the standard NRAO policy: 18 months since the end of the observations. Such a simple policy is better accomplished by not entering data into the archive until the proprietary period has expired.

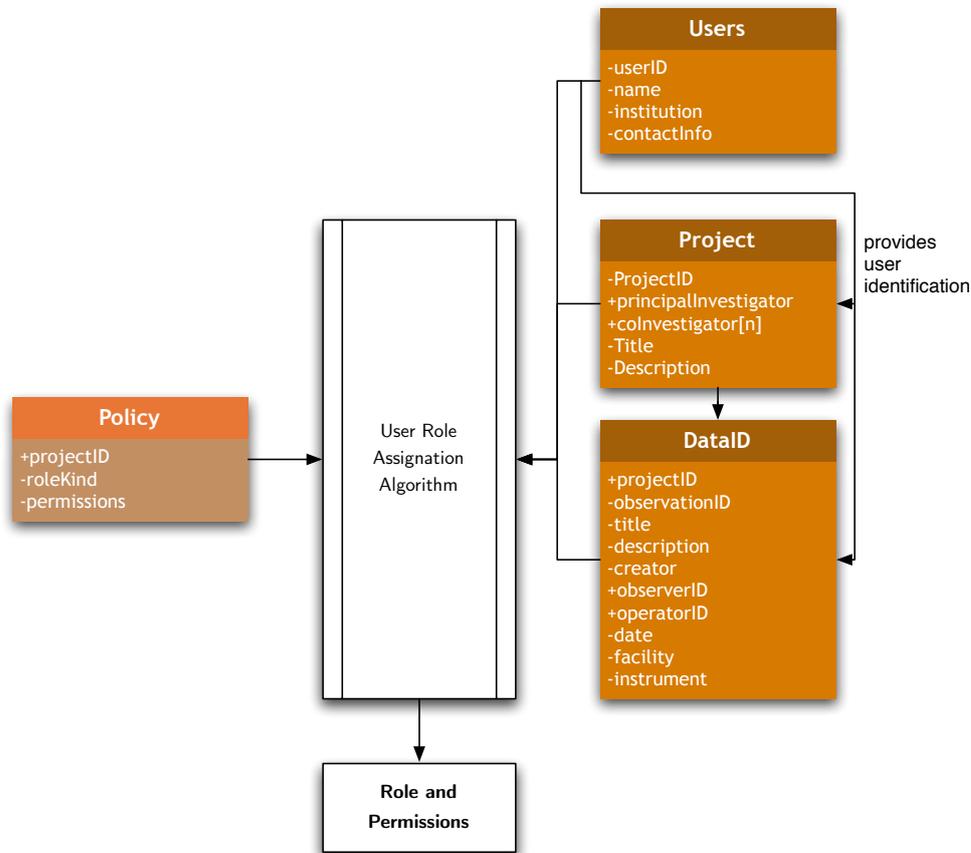


Figure 7.2: Policy data model, with role and permissions.

A more flexible solution would make use of an array of permissions by role, where roles are derived from the relationship between the principal investigator and/or the observer, and observatory staff.

Such roles would be chosen from a controlled vocabulary (`principalInvestigator`, `observer`, `coInvestigator`, `observatoryStaff`, `none`); those roles are derived from user/project relationships, so that people not belonging to the observatory, and which have nothing to do with the project, would default to `none`.

Each project in the archive should have, at least, an explicit policy of what is allowed for some with none relationship with the Principal Investigator, and people with `principalInvestigator` roles have all access rights to the archive; people with roles other than `principalInvestigator` would fallback to the `none` role, if their role's permissions are not explicitly declared.

We will use `Policy`, `Users` and `ObsData` metadata in order to select the corresponding role for the agent just logged in. Figure 7.3 shows the flow

diagram for the role selection.

Policy metadata and attributes are specified in Table 7.1, but also we will specify a subset of Curation attributes needed for successful Policy attribution.

We are also considering of establishing with the Policy data model data-oriented policies, instead of user-oriented. Data-oriented policies allow each datum —possibly by means of the DataID or Project classes— to be searched and found, but no additional information, or only partial information —possibly having to do with the owner of the datum— can be retrieved, depending on the data policy in place. This needs at least an additional attribute in the DataID class.

This could be easily changed into a role-enabling algorithm that enables different roles for the same user, and displays all the different roles the user can access. If this is not needed, we will stick to the proposed algorithm.

7.3 Packaging

packaging

noun

- materials used to wrap or protect goods.
- the business or process of packing goods.
- the presentation of a person, product, or action in a particular way.

The New Oxford American Dictionary, 2nd Edition

Digital data are usually thought of as free of *bit rot*: digital copies are always bit perfect, but that is so because a lot of care is put into providing enough redundancy on the storage and interchange processes, so that possible bit flips can be detected and corrected.

But changes in content can also be produced on intermediate states for a dataset: imagine reduced data products for which the calibration has been changed, producing at the same time a change in characterisation. The new data product being provided might link to an updated data product, but its characterisation can be the old one. Thus, we need a way to provide a permanent set of data products, which can be easily combined without further service queries by means of attached characterisation instances.

The Packaging class is used to specify how data from different sources will be presented together. For instance, if we wanted to retrieve data belonging to our already mentioned maser survey, a VO system could reply with a Multi-Beam FITS file containing all the scans and sub-scans that conform the On-Off patterns for a single issued observation, or with a .zip file with all the FITS files belonging to the survey, or with just a single On-Off pair, et cetera.

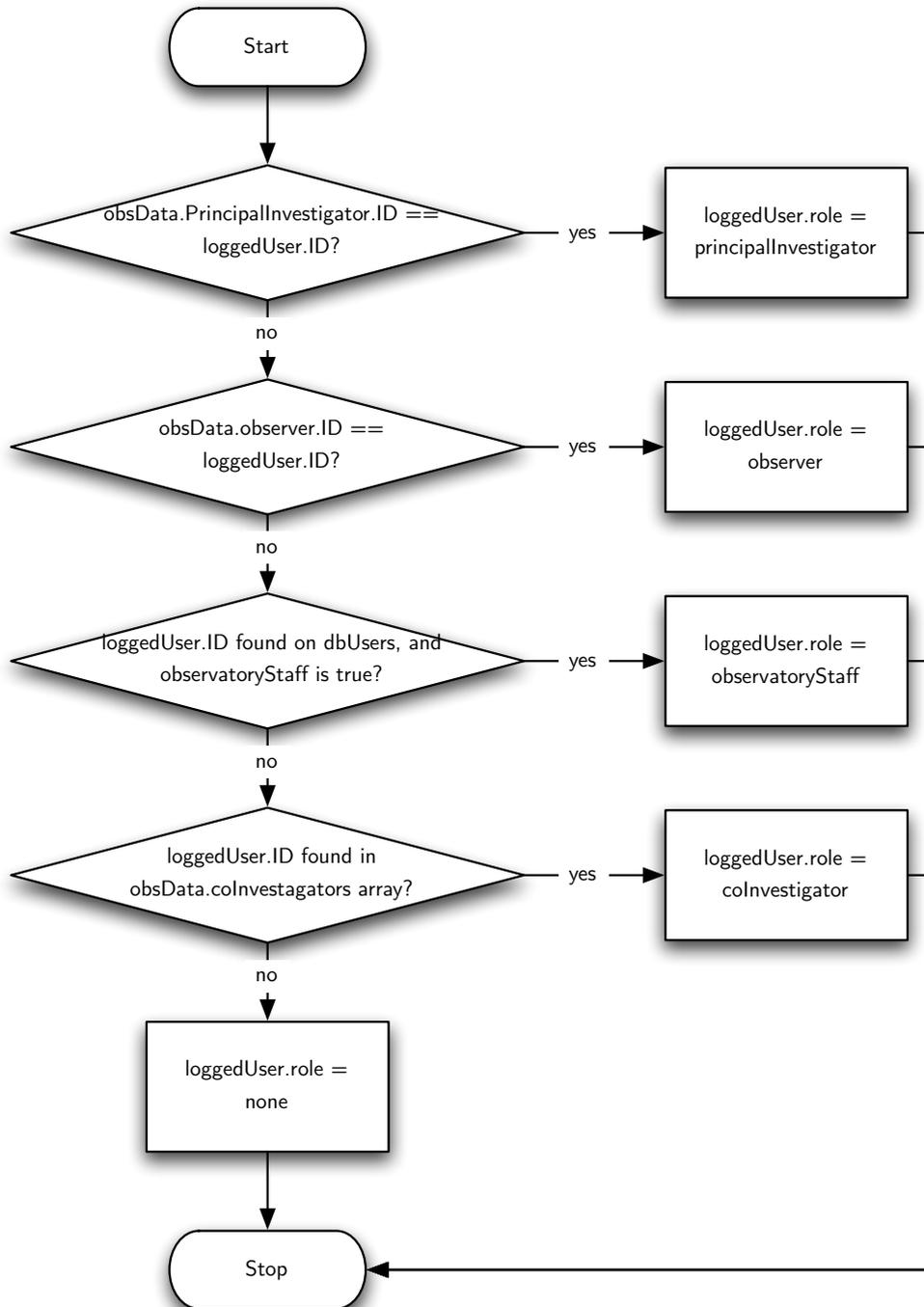


Figure 7.3: Flow diagram for the role determination algorithm.

Table 7.1: Policy metadata.

Attribute	FITS Keyword	UCD	Description
projectID	PROJID	meta.curation.project; meta.id ^a	Project identifier.
roleKind	assign	meta.policy.role; meta.code; meta.id ^b	Role kind for which permissions will be provided.
permissions	assign	meta.policy.permissions; meta.code ^a	Permissions provided for roleKind users of this project. The particular permissions to be provided are to be discussed by IVOA.

^aThere is no meta.curation.project UCD, but we propose the inclusion of one.

^bThere are no meta.policy.* UCDs, but we propose, at least, the inclusion of the meta.policy.atom.

Table 7.2: Policy related Users metadata.

Attribute	FITS Keyword	UCD	Description
userID	COMMENT	meta.id	User identifier for all user related operations in the archive.
name	COMMENT	meta.name	Real name of the user. Any known user of the archive has to be registered, or be anonymous.
institution	COMMENT	meta.name	Name of the institution to which the user belongs.
contactInfo	COMMENT	meta.note	Contact info (probably e-mail) for this user.

Table 7.3: Policy related Project metadata.

Attribute	FITS Keyword	UCD	Description
projectID	PROJID	meta.curation.project; meta.id ^a	Project identifier.
principalInvestigator	COMMENT	meta.id	User identifier for the principalInvestigator of the project.
coInvestigator[n]	COMMENT	meta.id	User identifier for the n th co-investigator.
title	COMMENT	meta.curation.project; meta.title ^a	Project title.
description	COMMENT	meta.curation.project; meta.note ^a	Project description.

^aThere is no meta.curation.project UCD, but we propose the inclusion of one.

An instance of the Packaging class would describe the contents of the data retrieved, in terms of project organisation, and of the particular files being actually delivered.

Unfortunately, there is no Packaging class defined at the VO level, so we have to resort to other packaging description mechanisms available in other archiving tools.

First, we have to state the requirements for this class:

- The Packaging class will describe the type of compression or grouping, if any, applied to the result of a query. So, possible results should form a controlled vocabulary, such as FITS, VOTable, zip, tar, gz, bz, bz2, tgz, tbz, tbz2, corresponding to: raw files —a FITS file or a VOTable—, a zip file, a tar file, a raw file compressed by gzip, bzip, bzip2, and a gzipped, bzipped or bzipped2 tar file.
- We believe a standard VO packaging scheme is needed in order to facilitate distribution. We propose one such scheme, that we call VOPack —Virtual Observatory Package; .vopack file extension—. We will define the VOPack as a “tarred and gzipped” folder (.tgz file) with an XML content descriptor, which acts as VO-aware manifest of contained assets.

The VOPack is a way of distributing VO-compliant content, in a way that makes it easy to reuse and point to existing content, either remote or locally.

Table 7.4: Policy related DataID metadata.

Attribute	FITS Keyword	UCD	Description
projectID	PROJID	meta.curation.project; meta.id ^a	Project identifier.
observationID	OBSID	obs; meta.dataset; meta.id	User identifier for the principalInvestigator of the project.
coInvestigator[n]	COMMENT	obs; meta.id	User identifier for the n th project co-investigator.
title	COMMENT	meta.curation.project; meta.title ^a	Project title.
description	COMMENT	meta.curation.project; meta.note ^a	Project description.
creator	AUTHOR	meta.curation; meta.id	User identifier for the creator of the data entry.
observerID	OBSERVER	obs.observer; meta.id	User identifier for the person performing the observation producing this data entry.
operatorID	OBSERVER	obs.operator; meta.id ^b	User identifier for the person performing operator duties while performing the observation.
date	DATE-OBS	time.obs.start	Date of observation.
facility	TELESCOP	instr.obsty	Facility (observatory) where the telescope/instrument resides in.
instrument	INSTRUME	instr.tel	Instrument performing the observation ^c .

^aThere is no meta.curation.project UCD, but we propose the inclusion of one.

^bWe propose the inclusion of either obs.operator or instr.operator as new UCDs to characterise operator-related data. However, obs.observer can be used when providing both observer and operator at the same time.

^cWe have to study whether this should contain an instrument-backend pair, or have different attributes for both.

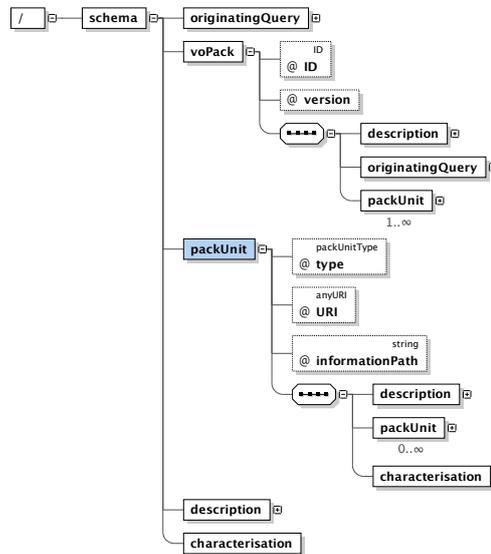


Figure 7.4: VOPack structure. Diagram generated by **Oxygen** from the XML schema.

A VOPack consists of a compressed file that contains at least a `voPack.xml` file—following the VOPack XML Schema—that describes all the additional content of the compressed VOPack, and their relationships between them. Figure 7.4 shows the structure diagram of a VOPack.

In that diagram, the `voPack` element is the root for the XML document. It includes a description, the originating query, and one or more `packUnits`, which actually point to the information being retrieved. The `originatingQuery` element contains the string with the URI that allows the retrieval of the `voPack`. Additional `characterisation` elements, following the Characterisation schema, can be used to further specify properties on the data being delivered with the VOPack.

The `packUnit` corresponds to a single piece of data, or to another `packUnits`, in case of more structured data. The depth of inclusion is arbitrary.

`packUnits` have a `type` attribute that can be one of: `votable`, `fits`, `vopack`, `compressedFolder`, `folder`, `otherXML`, `otherNonXML`. Table 7.5 specifies the meaning of this attribute.

For the `vopack`, `folder` and `compressedFolder` values, a new `vopack.xml` file has to be provided for their description. This allows for meta-packaging of ready-made VOPacks.

For the first three types, the `informationPath` attribute gives an XPath to the actual data being pointed, just in case the `packUnit` contains several tables, and not all of them are to be considered. In the case of FITS files, the

Table 7.5: Meaning of the different valid values for attributes of the `packUnit` data type.

<code>packUnit</code>	Description
<code>votable</code>	The packed unit is a VOTable.
<code>fits</code>	The packed unit is a FITS file.
<code>vopack</code>	The packed unit is itself a VO pack. The characterisation of the referencing VO pack encompasses all packed units, while each particular one will have its own, <i>narrower</i> characterisation.
<code>compressedFolder</code>	The referenced packed unit is a compressed directory.
<code>folder</code>	The referenced packed unit is a directory in the same file-system as the referencing VOPack.
<code>otherXML</code>	An XML representation, other than a VOTable, is used.
<code>otherNonXML</code>	A non-XML representation, also different from a FITS file, is used. This kind of representation should be avoided, but would be useful for packing instrument specific raw data formats, which are correctly characterised.

`informationPath` looks XPath-like, but points to the HDU or Image holding the data.

Figure 7.5 shows the complete listing for the VOPack XML Schema.

The VOPack XML Schema has been inspired by the concepts of Digital Items, Digital Item Containers, and Digital Item Components from MPEG-21 [72].

7.4 Conclusions

In this chapter we have dealt with generic astronomical concepts, non-specific of radio astronomy, but which were needed both for implementing the suggested ObsDM entities, but also for being able to successfully use IVOA data models as blueprints for archive development, as it will be shown when applying the data model to the archives of DSS-63 and IRAM 30m in chapter 12.

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- voPack schema, by Juan de Dios Santander Vela, IAA-CSIC -->
<!-- Ideas for voPack schema

voPack as a list of packUnits; packUnit is an abstract type
packUnits: fits, votable, folder, compressedFile, voxml, non-voxml, other
voxml types: sed, characterisation, stc, stcoords
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import namespace="http://www.w3.org/2001/XMLSchema" ></xs:import>
  <xs:complexType name="anyText" mixed="true">
    <xs:sequence>
      <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="packUnitType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="votable"/>
      <xs:enumeration value="fits"/>
      <xs:enumeration value="otherXML"/>
      <xs:enumeration value="otherNonXML"/>
      <xs:enumeration value="vopack"/>
      <xs:enumeration value="compressedFolder"/>
      <xs:enumeration value="folder"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- voPack is the root element of the schema -->
  <xs:element name="voPack">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="description" minOccurs="1" />
        <xs:element ref="originatingQuery" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="packUnit" minOccurs="1" maxOccurs="unbounded"/></xs:element>
      </xs:sequence>
      <xs:attribute name="ID" type="xs:ID"/>
      <xs:attribute name="version" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="0.1"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="description">
    <xs:complexType mixed="true">
      <xs:complexContent>
        <xs:extension base="anyText">
          <xs:attribute name="ID" type="xs:ID" />
          <xs:attribute name="value" type="xs:string" use="required" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="originatingQuery">
    <xs:complexType>
      <xs:attribute name="value" type="xs:string" use="required"/>
      <xs:attribute name="URL" type="xs:anyURI" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="characterisation">
    <xs:complexType>
      <!-- To be defined; should follow the characterisation schema -->
      <xs:sequence>
        <!-- Sequence of axis characterisations for the given pack unit -->
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="packUnit">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="description" minOccurs="1" />
        <xs:element ref="packUnit" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="characterisation" minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="type" type="packUnitType" use="required"/>
      <xs:attribute name="URI" type="xs:anyURI" use="required" />
      <xs:attribute name="informationPath" type="xs:string" use="optional"/></xs:attributes>
      <!-- The URI provides information either for the local file system, or remotely -->
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Figure 7.5: VOPack XSD schema listing.

Chapter 8

Data Provenance

provenance

noun

- the place of origin or earliest known history of something.
- the beginning of something's existence; something's origin.
- a record of ownership of a work of art or an antique, used as a guide to authenticity or quality.

The New Oxford American Dictionary, 2nd Edition

The dictionary definition above points to the main three elements to which data provenance in e-Science has to deal with: Attribution, i.e. proper acknowledgement of authorship and or origin; Data Quality; and Replication of results, i.e., the chain of processes needed to derive a result, beginning with data sources.

In the VO context, we can synthesise a definition by stating that Provenance (data provenance) is the record of the earliest known history of an astronomical data set, to be used as a guide to data quality and of data ownership.

Astronomical datasets use the data coming from different photons¹ to derive physical properties from the observed objects. Therefore, data provenance —also known as *lineage* information— in astrophysics is, in fact, the answer to the question *What was done to this collection of photons, and which systems did it?*

¹Their arrival rate, their energy distribution, the positions from the sky, the difference with properties of photons not coming from the observed region; in addition, the response to photons, and to their absence, has also to be determined.

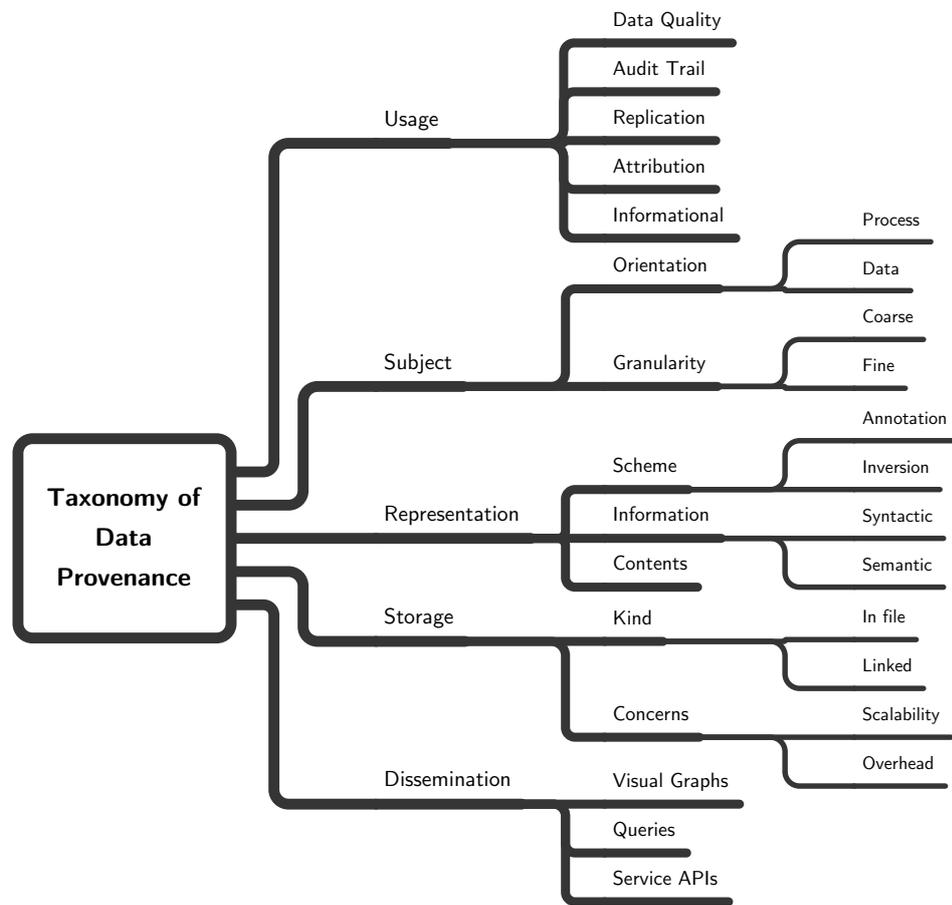


Figure 8.1: Taxonomy of data provenance techniques in e-Science. Those techniques can be classified depending on the use of the provenance data collected, on the actual subject of data provenance, or the way data provenance is represented, stored, or disseminated. Based on *A Survey of Data Provenance in e-Science* [82].

8.1 Provenance in e-Science

As we have already established, data Provenance in the Virtual Observatory is just a specific form of the problem of data Provenance in e-Science, so in order to establish how to model astronomical data provenance, we will study first the different data provenance recording and determination techniques already available.

In the paper *A Survey of Data Provenance in e-Science* [82] by Simmhan et al. many different data provenance collection techniques are studied, and a taxonomy of them is created —see figure 8.1—, depending on:

Usage Data provenance information can be used for estimating data quality and data reliability, or to provide an audit trail of the data transformations. It can also be used to replicate an experiment, to attribute the creation of a dataset to a set of original (in the sense of originating) data, and to the set of processes needed to derive the new dataset. Or it can be informational, without providing enough information to assess any of the above.

Subject Data provenance can be focused on the data themselves, collecting lineage metadata from each data product, or on the processes on the data. And in any case the recorded provenance can be coarse grained (for instance, describing a whole set of data) or fine grained (describing each single datum and/or process; for instance, database tuples' attributes).

Representation The way data provenance can be represented is also diverse, and some of them depend on the processing system being studied. The two main approaches in the literature are annotations and inversion. Annotations are *a priori* provenance metadata: the transformed data are accompanied by provenance metadata which has been created prior to the transformation. On the other hand, inversion data provenance is created *a posteriori* from the data products of a transformation which can be inverted, with provenance metadata being created from the documentation of the process and the differences between inputs and outputs. It is clear that inversion metadata are more compact, while annotations can include parameters of processes, their versions, and even references to publications.

As some of the lineage information implies relationships between datasets and processes, that information can be captured in data models about the processes, or using semantic web technologies, such as the RDF and OWL languages, in order to describe such relationships. In this way, the process semantics are documented. In any case, the process syntax is specified from the input data, the output data, and annotations, if present.

Storage As provenance metadata can be generated, collected and/or transmitted at many different places, there must be a way to keep those metadata stored, while keeping the relationship with the data themselves. Depending on the grain of the metadata collection process, and on whether the lineage information is just updated or versioned, the amount of provenance metadata can grow several orders of magnitude above the original data. Both the overhead of provenance metadata (percentage of storage devoted to provenance versus data, and cost of metadata management), and the scalability of the system, that is, how

to deal with the provenance metadata if the data rate increases an order of magnitude.

Dissemination Finally, provenance metadata are gathered for applications to use them. Typical ways of disseminating lineage information are by means of derivation graphs, but in many cases where the provenance metadata are stored inline with the data the form is just a list of time-stamped annotations. If semantic web tools are used, workflows can determine the input provenance information and create the dissemination graph at runtime. In addition, specially in cases where lineage metadata are stored in databases or XML documents, provenance specific queries can be performed, and even provenance query APIs can be created so that different systems can access such information.

Another classification [83] of Provenance data can be performed regarding how it is collected or computed:

Why provenance Refers to the reason why a datum is in a given dataset, i.e., the query that was performed, including data sources. For data coming from database queries, different fields can have different sources, but all fields, and many rows derived from the same query, will share that query. In a sense, *Why* provenance is a proof that a datum belongs to a processed (queried) data set, as it corresponds to the minimum set of sources and sources' entries which, together with the query being performed, will provide that datum as an answer.

Where provenance Refers to the actual progenitors of each particular datum, i.e., what particular database tuple, and field in the tuple, provided the datum we are analysing.

The difference between both kinds can be better illustrated using a database originated example. Let us imagine the SQL query:

```
SELECT name, telephone
FROM employee
WHERE salary > (SELECT AVERAGE salary FROM employee)
```

And let us say one of the tuples in the result is ("John Doe", 555123467). As all rows in `employee` were needed for the calculation of the salary average, a change in any row could make our target tuple disappear from the result list, so the *Why* provenance is the query plus the salary, name and telephone fields of all rows of the `employee` table. The *Where* provenance, however, is only concerned with the actual progenitor of each of the tuple's datum, and the answer is the name and employee fields of the row containing John Doe's entry in the `employee` table.

Finally, another difference can be established for data provenance collection systems, depending on the data collection strategy:

Eager collection Lineage information is created/computed after each transformational or derivational step.

Lazy collection Provenance information is calculated on demand, using specific knowledge of the transformations, and parameters saved.

We will end up our review of data provenance systems and classifications by showing table 8.1 of different data provenance management systems in different application areas, and see their characteristics.

We can see that most of the systems assume a relational database infrastructure, and rely on annotations for carrying on provenance information. Only systems completely contained within a relational database systems use inverse queries or functions for their recovery of data provenance.

So, in order to establish a Data Provenance framework for the Virtual Observatory we will have to establish all of the items above: how to collect the information, which will be the intended use, and how is the information going to be represented, stored, and presented to intended users and systems; whether we need a *Why* or *Where* provenance system depending on typical VO use cases; and if such system must be *Eager* or *Lazy*, again depending on use cases.

8.2 Provenance in astronomy and astrophysics

After the classification above, we will study the different data provenance techniques in use within the astronomy, together with their intended application.

Typically, one of the uses of header cards in FITS files is storing COMMENTS and HISTORY cards. Figure 8.2 shows an example of a FITS file processed by the AIPS² interferometric data reduction software.

In that figure we can see an example of how typical astrophysical tools have been dealing with data provenance. Each different application can make use of the FITS header cards, specifically of the HISTORY keyword, in order to provide feedback of the steps being performed on the data.

In this case, the data provenance being provided is of the *Why* kind, albeit not complete. Apart from the text *annotation* on the FITS headers, tables of task Parameters are stored within the FITS file itself, so this provenance information is collected *eagerly*, and corresponds to *Why* provenance. If all the operations were invertible, it would also provide an invertible *Where* provenance, but some of the data is lost during each processing step.

However, that is the case for a particular package. The FITS headers added by the difmap package³, for example, only indicate that difmap has touched the file somehow, without any details:

²<http://www.aips.nrao.edu/>

³<ftp://ftp.astro.caltech.edu/pub/difmap/difmap.html>

Table 8.1: Properties of different data provenance management systems for different scientific domains, as compiled by [82].

	LIP	Chimera	myGRID	CMCS	PASOA	ESSW	Tioga	Buneman	Trio
Domain ^a	GIS	Physics, Astronomy	Biology	Chemistry	Biology	Earth sciences	Atmospheric Sciences	Generic	Generic
Processing ^b	Commands	Services	Services	Services	Services	Scripts	RDBMS	RDBMS	RDBMS
Application ^c	IR	IRAP	IR	IU	IR	I	IE	IU	IU
Orientation	Data	Process	Process	Data	Process	Data & Process	Data	Data	Data
Granularity	Spatial layers	Abstract datasets (files)	Abstract sources	Files	Workflow Parameters	Files	DB tributes	DB tributes & Tuples	DB Tuples
Representation	Annotations	Annotations	Annotations	Annotations	Annotations	Annotations	Inverse functions	Inverse queries	Inverse queries
Semantics	No	No	Yes	Limited	No	Proposed	No	No	No
Storage ^d	RDBMS	RDBMS	RDBMS	RDBMS	RDBMS + File System	RDBMS	RDBMS	N/A	RDBMS
Overhead	user commands; MD entry	user initialization; automatic WF	def-service calls	service calls; working port	manual	manual	manual registry of inverse functions	N/A	Automatic generation of inverse queries
Scalability	No	Yes	No	No	Proposed	Proposed	Yes	N/A	No
Dissemination ^e	Queries	Queries	Semantic browser; Lineage graph	Queries; Browsing	Queries	Browsing	Queries; Graph	N/A	Queries

^aGIS: Geographical Information System
^bRDBMS: Relational Data Base Management System
^cKind of Provenance application: I: Informational; R: Regeneration; A: Audit; E: Error Tracking; U: Information Update; P: Planning
^dRDBMS: Relational Data Base Management System; N/A: Not available
^eN/A: Not Available

- allows queries on metadata which are relevant in order to find datasets which have given common properties;
- integrate not only software processing data provenance, but also instrumental data provenance, and observation configuration information.;
- and all of this has to be modular enough so that instrumental data provenance plus observation configuration can be adapted for many different observatories.

8.3 Properties of an IVOA Data Provenance proposal

From the requirements above, plus what we have learned about the typical uses of data provenance in astrophysics, we can say that an IVOA proposal for data provenance should:

- Be domain specific, but taking into account existing Provenance models for similar e-Science initiatives, such as GIS.
- Independent of RDBMS, as many astronomical datasets do not belong to databases.
- Traditionally, astrophysical data provenance has been used for informative uses, and sometimes for user error tracking. However, in the VO many data providers want (and even need) their services to be properly acknowledged (attribution), and data mining tools can make use of data provenance information to perform planning of data processing.
- It should be oriented both to the data and the data workflows.
- Provenance has to be provided, at least, at the level of complete observations, but it would be sensible to be able to provide provenance information at the scan level.
- Given the nature of astronomical datasets, and the fact that they usually go through a processing pipeline, Provenance information should be Eagerly collected, to avoid intensive computations later on,
- Provenance semantics in the VO are guaranteed by the use of several techniques: a precise Provenance data model (at least, for radio astronomical observations); the use of UTypes to link attributes with specific data model parts; and the use of UCDs to identify similarities in meaning. In addition, the IVOA Vocabularies being proposed⁴ can be used in order to clarify precise meanings for terms known to astronomical/instrumental literature, but still not unified within the IVOA.

⁴Based on several astronomical thesauri, such as the IAU Thesaurus.

- Techniques for storing data provenance should be archive-specific, but the expression of data provenance should be in XML, both in a custom ObsDM XML format (to be developed), and serialised in VOTables using UTypes and UCDS as pointers.
- Given that data provenance is to be expressed in XML, either in custom XML format, or in VOTable format, XML-related tools can be used to disseminate the provenance. Besides, XHTML or HTML4 can be used to express the data provenance when it is considered to be informational, instead of being available for queries on Provenance data. This last case, however, should be avoided, as the role of Provenance in the IVOA should be letting astronomers query on the parameters related to observation acquisition.
- The scalability of this approach to data Provenance, where each archive provides data provenance for the data it provides, allows the creation of a VO-wide Provenance infrastructure... but introduces the problem of being able to retrieve that information through the network from many different providers. This problem deserves further studies.

One additional concern for any proposal of astronomical data provenance is the keeping of all provenance information which is delivered, with annotations added for additional steps. In many present VO tools VOTables are converted into an application-specific intermediate representation, and when exported again as VOTables those metadata are lost.

8.4 Conclusions

Data provenance, by definition, it is completely dependant on the instrumental setting, and thus a VO-wide data provenance mechanism has to provide ways for both data archives and software packages to provide provenance information.

However, as most of the data provenance information is used mostly for data quality assessment and data selection, and less so for data processing, Provenance information should be easily accessible for astronomers and applications, but should be modular enough as to separate instrumental settings, environmental information, and signal processing description.

We will describe these modules in the next chapter.

Chapter 9

RADAMS: Data provenance

In this section, we will deal with the data provenance part of the RADAMS. This sub data model provides support for the description of how data have been generated.

As we saw in the RADAMS overview —chapter 5—, we have divided data provenance in three parts:

Instrumental Has to do with the instrumental setup, i.e., the configuration of all observation elements between the original photon source and the photon detection equipment.

Environmental Has to do with the elements in the path of the photon source which cannot be controlled by the instrumental setup, but that nonetheless affect the photon collection (i.e., by causing turbulence, changing the refraction index, or absorbing photons). We will register measurable environmental parameters in order to estimate possible effects and/or defects in the detections.

Processing Once raw data are recorded, many different processing steps need to be performed in order to provide science-ready data, or as it is sometimes said, data are provided with the instrument signature removed as much as possible. Processing Provenance records the different processes and their inputs performed in order to achieve the result being offered by the archive.

We will study each one in detail in the following sections.

9.1 Instrumental provenance

Figure 9.1 shows the classes associated with the instrumental configuration for the observation.

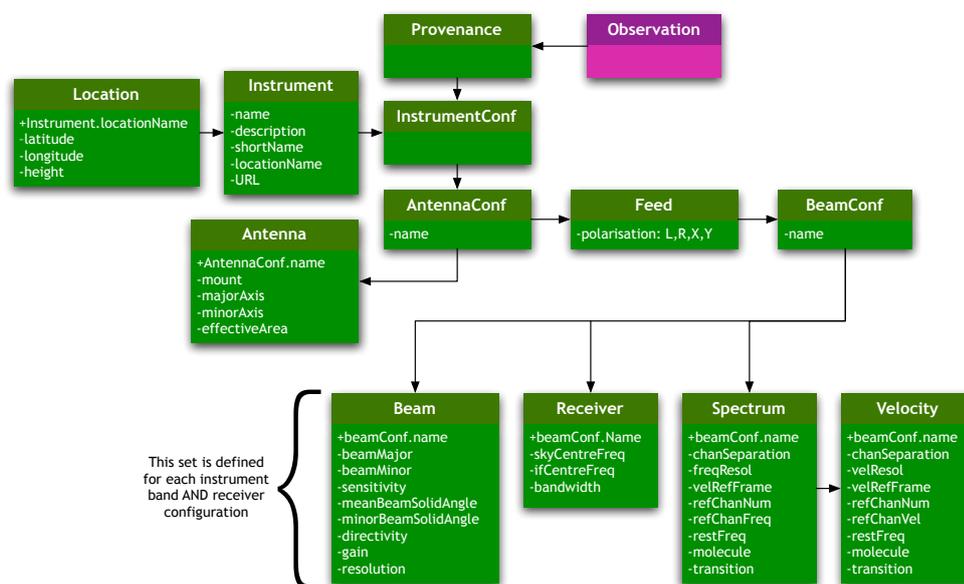


Figure 9.1: Provenance.Instrument data model.

InstrumentConf Each observation is associated to a particular instrumental configuration, which results from the particular configuration of the instrument + antenna + feed system. InstrumentConf instances group those settings.

Instrument Instances of this class specify the instrument configuration, as used for the observation.

Instrument.Location This is an instance of a Location class, used for specifying the location for the instrument.

AntennaConf In the same way InstrumentConf allows the grouping for all the instrumental settings, AntennaConf instances group together Antenna and Feed info (regarding polarisation), plus BeamConf—another aggregator or class—. Several AntennaConf instances can provide information for different antennas and feeds. Each possible antenna configuration will be labelled by a name.

Antenna Instances of this class specify the general properties for each given antenna. We will also use these instances to specify the type of scan being performed on the source from a controlled vocabulary.

Feed Instances of this class—one or more per AntennaConf—specify each of the feed horns used for the observation, and their corresponding

Table 9.1: Provenance instrument metadata.

Attribute	FITS Keyword	UCD	Description
name	INSTRUME	meta.id; instr; meta.main	Instrument name.
description	assign	meta.note; meta.main	Instrument description.
shortName	assign	instr; meta.id	Short name for the instrument.
locationName	assign	instr; pos; meta.id	Localisation of the instrument.
URL	COMMENT	instr; meta.ref.url	URL for the instrument (website for the instrument, documentation, or any other type of instrument description).

polarisation from a controlled vocabulary: L, R, X, Y. We cannot make use of the Stokes polarisation parameters, as they cannot be directly measured via the feed configuration: instead, they have to be derived by means of data processing steps.

BeamConf This class is used to group antennas, feeds, and beams. The relationship between BeamConf and Feed instances allows the specification of different beams, formed by the combination of different feeds. In a single-dish single-feed configuration, there is only one beam associated to a given receiver. We can also use this association for a single-dish, multiple-feed configuration where each feed goes to a different receiver.

Beam Metadata for this class specifies the actual beam for the telescope, associated to a given spectral band.

Receiver This metadata are used to describe the most relevant properties of the receiver, such as receiver type, intermediate frequency—in the case of heterodyne stages—, et cetera.

Spectrum In case of spectroscopic observations, the spectral analyser that has been used is specified by instances of this class.

Velocities This class mirrors the Spectrum class, and is preferred for those cases where velocities are used, instead of frequency.

Table 9.2: Instrument location metadata.

Attribute	FITS Keyword	UCD	Description
locationName	assign	instr; pos; meta.id	Name of a particular instrument location.
latitude	SITELAT	instr; pos.earth.lat	Instrument location latitude.
longitude	SITELONG	instr; pos.earth.lon	Instrument location longitude.
altitude	SITELEV	instr; pos.earth.altitude	Instrument location altitude.

Table 9.3: Antenna configuration metadata.

Attribute	FITS Keyword	UCD	Description
name	assign	instr.telescope; meta.title; meta.id	Name of the particular antenna.
scanType	assign	instr.setup; meta.code	Type of scan being performed by this antenna, from a limited vocabulary (suggested by the RDM [68]): beamSwitch, cal, cross, dopplerTrack, dwell, focus, frequencySwitch, holography, mosaic, onOff, onTheFly, point, positionSwitch, pulsar, raster, skyDip, tiedArray, track, wobblerSwitch.
mount	assign	meta.note	Mount type for the telescope from a limited vocabulary: azimuthal, equatorial, alt-azimuthal, dobson, german equatorial.
majorAxis	assign	instr; phys.size.smajAxis	Major axis dimensions.
minorAxis	assign	inst; phys.size.sminAxis	Minor axis dimensions.
effectiveArea	assign	instr; phys.area	Effective instrument area.

Table 9.4: Feed configuration metadata.

Attribute	FITS Keyword	UCD	Description
polarisation	STOKES	pos.posAng; phys.polarization; meta.code	Polarisation value from a controlled vocabulary: L, R, X, Y

Table 9.5: Beam configuration metadata.

Attribute	FITS Keyword	UCD	Description
beamMajor	BMAJ/HPBW	instr.beam; phys.size.smajAxis	Major axis HPBW of the main lobe of the beam.
beamMinor	BMIN/HPBW	instr.beam; phys.size.sminAxis	Minor axis HPBW of the main lobe of the beam.
sensitivity	BEAMEFF	instr.beam; instr.sensitivity	Beam average sensitivity.
mainBeamSolidAngle	assign	instr.beam; pos.posAng; meta.main	Main lobe's beam solid angle.
totalBeamSolidAngle	assign	instr.beam; pos.posAng; stat.max	Total beam solid angle, including secondary lobes.
directivity	assign	instr.beam; instr.setup; arith.factor	Directivity percentage.
gain	ANTGAIN	instr.beam; instr.setup; arith.factor	Beam gain ^a .

^aWe still have to clarify if the gain attribute is related to the directivity concept or not, and if it is related with the receiving stages or not.

Table 9.6: Receiver metadata.

Attribute	FITS Keyword	UCD	Description
type	BACKEND	instr.setup; meta.note	Receiver type (HEMT, Bolometer, SIS, et cetera).
skyCentreFreq	assign	src; em.radio; em.freq	Antenna tuning frequency.
IFCentreFreq	assign	instr.setup; em.freq	Heterodyne receiver intermediate frequency (or list of frequencies).
bandwidth	BANDWID	instr.bandwidth	Filter-bank total bandwidth.

Table 9.7: Spectrum metadata. It might be necessary to change the MOLECULE and TRANSITI keywords by LINE, for better CLASS compatibility.

Attribute	FITS Keyword	UCD	Description
numChannels	NAXISn	spect; em.freq; meta.number	Number of spectral channels.
chanSeparation ^a	assign	spect; em.freq	Mean channel separation (in frequency units), or channel frequency separation array.
freqResolution	FREQRES	spect.resolution; em.freq	Frequency resolution.
refChanNum	assign	spect; em.freq; meta.number; meta.ref	Spectral reference channel.
refChanFreq	OBSFREQ	spect; em.freq; meta.code; meta.ref	Spectral reference frequency (observed frequency).
restFreq	FREQn or RESTFREQ	spect.line; em.freq	Observed spectral line rest frequency.
molecule	MOLECULE ^b	spect; phys.mol; meta.id	Molecule name.
transition	TRANSITI ^c	spect; phys.atmol.transition; meta.id	Transition.

^aThere is a certain redundancy between the Provenance.Spectrum.chanSeparation attribute and the Coverage.Spectral.Resolution attributes.

^bIt might be necessary to change the MOLECULE keyword by LINE, for better CLASS compatibility.

^cIt might be necessary to change the TRANSITI keyword by LINE, for better CLASS compatibility.

Table 9.8: Velocity metadata.

Attribute	FITS Keyword	UCD	Description
numChannels	assign	spect; phys.veloc; meta.number	Number of velocity channels.
chanSeparation ^a	assign	spect; phys.veloc	Mean channel separation (in velocity units), or velocity channel separation array.
velResolution	assign	spect.resolution; phys.veloc	Velocity resolution.
velRefFrame	assign	spect; phys.veloc; pos.frame; meta.id	Identification of the reference system used for the velocity.
refChanNum	assign	spect; phys.veloc; meta.number; meta.ref	Velocity reference channel.
refChanFreq	assign	spect; phys.veloc; meta.code; meta.ref	Frequency for the velocity reference channel.
restFreq	RESTFREQ	spect.line; em.freq	Observed spectral line rest frequency.
molecule	MOLECULE ^b	spect; phys.mol; meta.id	Molecule name.
transition	TRANSITI ^c	spect; phys.atmol.transition; meta.id	Transition.

^aThere is a certain redundancy between the Provenance.Velocity.chanSeparation attribute and the Coverage.Spectral.Resolution attributes.

^bIt might be necessary to change the MOLECULE keyword by LINE, for better CLASS compatibility.

^cIt might be necessary to change the TRANSITI keyword by LINE, for better CLASS compatibility.

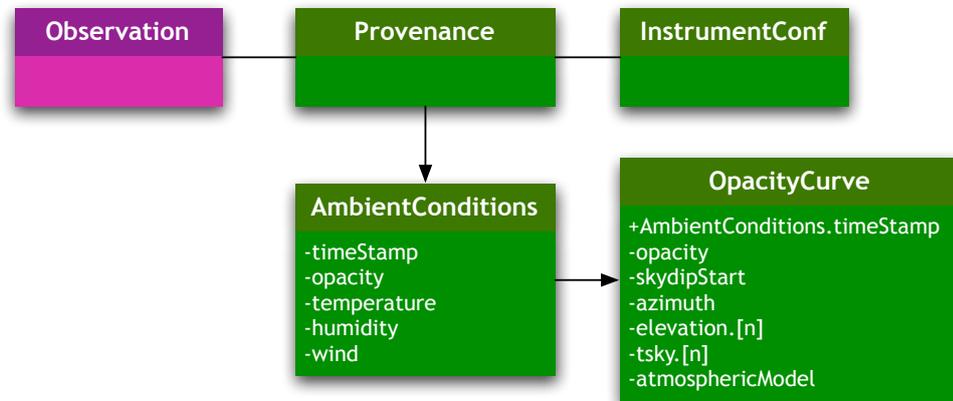


Figure 9.2: Provenance.AmbientConditions data model.

9.2 Environmental provenance

Environmental provenance is the part of the provenance dealing with ambient conditions, and as such the main class is called `Provenance.AmbientConditions`: it encompasses all metadata needed to specify weather conditions, air mass, opacity, et cetera. Figure 9.2 shows the corresponding classes and their relationships.

AmbientConditions Holds all metadata related with weather conditions for the observation, such as humidity, wind speed, opacity at zenith, et cetera.

OpacityCurve Includes the opacity curve (linked as a VOTable file) associated to the observing term where the data were observed (which we will derive from the nearest two skydip scans performed before and after the observation). We propose the inclusion of an array of [elevation, Tsky] pairs, together with the azimuth and the starting time of the skydip. Calculation of the opacity curve is different for bolometric or heterodyne observations. It is also necessary to include information on the atmospheric model and/or software used for opacity fitting (the atmospheric model used by MOPSIC and MIRA, the data reduction packages at the IRAM 30m antenna, is the *Atmospheric Transmission at Microwaves*, ATM, by Pardo et al. [84]).

Table 9.9: AmbientConditions metadata.

Attribute	FITS Keyword	UCD	Description
opacity	TAUZEN	phys.absorption.coeff	Opacity at zenith estimated at the observation frequency.
airMass	AIRMASS	obs.airMass	Air mass at zenith at the observing site.
temperature	TAMBIENT	phys.temperature	Ambient temperature.
humidity	HUMIDITY	obs.atmos; phys.columnDensity	Ambient humidity.
waterVapour	TAU_WPATH_RD<freq>	obs.atmos; phys.pressure	Equivalent pressure of the water vapour column.
tauFrequency	TAU_WPATH_RD<freq>	obs.atmos; em.freq	Tau radiometer frequency.
wind	WINDSPEE	obs.atmos; phys.veloc	Wind speed.

Table 9.10: Opacity metadata.

Attribute	FITS Keyword	UCD	Description
opacity	TAUZEN	phys.absorption.coeff	Opacity at zenith estimated at the observation frequency.
skydipStart	DATE-OBS	time.obs.start	Skydip starting time.
azimuth	AZIMUTH	pos.az	Skydip azimuth.
elevation[n]	ELEVATIO	pos.el	Skydip scan elevation.
tsky[n]	assign	instr.skyTemp	Sky temp at n th skydip.
atmosModel	assign	meta.modelled; obs.atmos; meta.id	Atmospheric model identification.

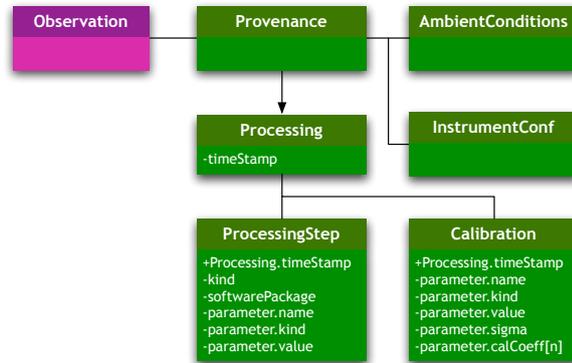


Figure 9.3: Provenance.Processing data model.

9.3 Processing provenance

The Provenance.Processing class will enable the specification of processing processes applied to the data before archival, including some processes necessary for the actual observation, such as the determination of the background signal via `frequencySwitching` or `positionSwitching` for background/source data comparison.

The RADAMS will make use of just two classes, Processing and Calibration —this is a subclass of processing—. Order is relevant, and it should be possible to reconstruct the pipeline by the ordering of Processing and/or Calibration instances. Figure 9.3 shows these classes.

Processing It holds information specifying the type of processing applied to data before archival. This includes pseudo-observational techniques such as position switching or frequency switching, as well as the type of data averaging, data weighting, et cetera. Table 9.11 provides minimal initial metadata, using arrays of parameter keywords for extensibility at the expense of complexity.

Calibration It is a subclass of Processing, where the type of processing is `dataCalibration`. In this class there are additional attributes to specify the type of calibration, and the axes to where this calibration will be applied. Table 9.12 provides minimal initial metadata, using arrays of parameter keywords for extensibility at the expense of complexity.

We still have to develop a calibration and/or pointing model; maybe based upon IRAM-Multi-Beam-FITS, or GBT FITS calibration tables.

Table 9.11: Processing Step metadata.

Attribute	FITS Keyword	UCD	Description
timestamp	DATE-RED	obs.param; time.epoch	Timestamp for the processing step being performed.
type	assign	obs.param; meta.code	Type of processing applied to source data; comes from a controlled vocabulary: unprocessed, noiseWeightedAverage, nonWeightedAverage.
softwarePackage	assign	meta.software; meta.id	Software package used for data processing; should come from a controlled vocabulary: CLASS, AIPS, AIPS++, CASA, MOPSIC, GILDAS, MIRA, MIR, other. In the case of other, the actual package that was used should be added as a parameter, with parameter.name as softwarePackage and the parameter.value as the package name.
parameter[n].name	assign	obs.param; meta.code	Additional processing parameter name, whose value will be in parameter.value; eventually, we will have a controlled list of possible parameter.name values.
parameter[n].type	assign	obs.param; meta.code	From a controlled vocabulary: integer, float, string, et cetera. At least all of FITS data types should be present.
parameter[n].value	assign	obs.param ^d	Value for the parameter indicated by parameter.name.

^dThe final UCD to mark parameter[n].value will be calculated when writing the VOTable, as it depends on parameter.type; it will be obs.param; meta.number most of the time, but it could be obs.param; meta.name or obs.param; meta.code, depending on the context.

Table 9.12: Calibration metadata^a.

Attribute	FITS Keyword	UCD	Description
timestamp	DATE-RED	obs.param; time.epoch	Timestamp for the calibration step being performed.
parameter.name	assign	obs.calib; obs.param; meta.id	Keyword defining the parameter that we will characterise with the remaining attributes.
parameter.type	assign	obs.calib; obs.param; meta.code	Type of calibration parameter used, from a controlled vocabulary: additive, factor, polynomial, exponential, logarithmic.
parameter.value	assign	obs.calib; obs.param; meta.number	Value for the main calibration parameter, where parameter.type is not polynomial.
parameter.sigma	assign	obs.calib; obs.param; meta.number	Value of sigma, for exponential calibrations.
parameter.calCoeff.[n]	assign	obs.calib; obs.param; meta.number	n th degree coefficient for a polynomial calibration parameter; polynomial degree is derived from the maximum n.

^aIt is mandatory that at least one [parameter.name, parameter.type, parameter.value] triplet appears, with fluxScale as parameter.name, and one of antennaTemperature, mbBrightnessTemperature, or S_nu as the parameter.value, with a parameter.type of string.

9.4 Conclusions

With this chapter we have finished our task of defining the modules suggested for the ObsDM, with the objective of being able to create a complete data model which could be used as a blueprint for archive development.

The Provenance data model is the most instrument dependant of the RADAMS models, but of the three sub-models, only the Instrument part is strictly specific to radio astronomy. This is a strength, however, as the Environment and Processing can be considered part of the atmosphere, and part of the workflow, and they are equally needed for all kinds of observations. By having the Instrument specific signature encapsulated in the Provenance.Instrument sub-model, it can be replaced by different Provenance.Instrument descriptions, meaningful for the different data analysis packages, which are the ones that need the instrument-specific information.

Part III

Bringing legacy tools to the VO

Chapter 10

Legacy astronomical packages and the VO

legacy

noun

- a thing handed down by a predecessor: *the legacy of centuries of neglect.*

legacy

adjective (computing)

- Denoting software or hardware that has been superseded but is difficult to replace because of its wide use.

The New Oxford American Dictionary, 2nd Edition

The physical properties we can ascertain from remote astronomical objects are the result of careful computations on observed datasets, which tend to be observatory, telescope, instrument, and even observing mode specific. Such computations include background noise estimations, electron counts to incident flux conversions, instrument signal removal, et cetera.

Many different software packages have been developed for performing those operations (which are commonly known as data reduction), and obtaining science-ready data products (i.e., images which have been flux calibrated, given precise astrometry, et cetera), and to analyse them to get additional physical information. For instance, once a spectrum has been calibrated on the local standard of rest of the source, the width of an emission line can be directly correlated to an expansion or contraction velocity of the observed object.

The development effort invested on these applications is enormous, and even when they have been developed using techniques available many years ago, and can be considered *technologically obsolete*, the recreation of all the algorithms in modern languages, or on more modern GUI framework foun-

dations is prohibitive, due to the extensive development and testing that would be needed for such a replacement.

Examples of 30-year old software packages still in heavy use include AIPS¹, a radio astronomical imaging package from the late seventies originally written in FORTRAN IV; or GIPSY², a 3D data analysis package for obtaining kinematic properties, whose development started in the early seventies, and is also written in FORTRAN.

But the goal set for the VO is to become the entire framework for future astronomical and astrophysical computing, both for its development and its execution, becoming completely transparent to astronomical users.

We cannot expect, then, astronomers to embrace the VO if they are not able to carry over the tools they are accustomed to, and we cannot recreate all the existing legacy applications. We must, therefore, find a way to bring these legacy applications to operate within the VO framework. That is the scope of this chapter: answering *How to bring legacy applications into the VO environment*.

10.1 VO-enabling applications

We must start by answering the question *What is a VO-enabled application?* From the many VO standards and protocols issued by the IVOA³, which and how many of them have to be implemented for us to consider an application is VO-enabled?

We will understand that a VO-enabled application is a software package which:

1. Can read and/or write data in VOTable (and possibly FITS) format.
2. Can query the VO Registry in order to find VO services or data resources, at least of one particular kind.
3. Can call a subset of VO-registered services for at least one of IVOA data access protocols⁴.

A somehow orthogonal capability is that of being able to communicate with other VO applications running in the same machine, via the Simple Application Messaging Protocol (SAMP) [85].

However, an application which can use SAMP to communicate with others can, in fact, make use of their capabilities, and retrieve data which have been

¹<http://www.aips.nrao.edu/>

²<http://www.astro.rug.nl/~gipsy/>

³<http://ivoa.net/Documents/>

⁴ConeSearch, Simple Image Access, or Simple Spectra Access. See appendix D.

processed by other applications. Conversely, the processed results can be mixed with those of any other application.

We can, then, consider that a VO application is just one which:

1. Can read and/or write data in VOTable (and possibly FITS) format.
2. Can communicate with other VO applications through messaging, both for sending and receiving data from those applications.

As we are leaving out the Registry search capability, and protocol calls, in order to truly VO-enable an application we will need to ensure that VO Registry access services, and ways to call VO services, are provided. That can be solved by either implementing or using external modules which can query the registry, perform data access queries, and return their results back to processing applications.

Let us compare, then, the two ways we have to create VO-enabled applications:

In-application VO compatibility In this scenario, we have to create a VO interface within every application we want to VO-enable. The following has to be implemented:

VO Registry interface A GUI for querying the VO registry, looking for the kind of services the application can make use of. Such a GUI can be reused only for applications using the same programming language and UI framework. We must consider it a *per application* development.

Changes into the User Interface The application user interface (be it command-line or graphical) has to be changed in order to be able to start the new VO functions: registry query, image/spectra/table queries and retrieval, messaging with other applications, et cetera. By definition, this is a *per application* development.

VO Data Access Layer interface The application must be able to obtain data from existing VO services (images, spectra, or tables). Code to query them must be included. There are query interfaces written in different programming languages, so we will consider this a *per language* development.

VOTable and FITS handling The two approved data formats for the VO are the VOTable and the FITS file formats. Any VO application has to be able to handle both (many VOTables include links to FITS files, which hold the actual data). There are libraries written for many different languages for FITS and VOTable handling, so there is a small *per application* development cost associated.

DAL to internal model interface Data retrieved from the VO correspond to the DAL data model, while the legacy application will have an internal data model of its own. This is responsible for the VOTable/FITS data conversion into the actual internal representation, and it is completely application dependent, and as such it represents a *per application* cost.

Messaging interface (optional) The application can also include a messaging interface. In that case, the DAL to the internal data model interface is also used by the data being retrieved from other applications. The messaging interface is based on the XML-RPC⁵ specification [86], for which many implementations exist in many different languages. The adaptation between the DAL model and the internal model it is the same as for the VO query interfaces, and includes a small amount of *per application* adaptation in order to support messaging.

In-application VO messaging In this other scenario, we leave the interface with the registry, and the DAL interface to other applications, and we rely on messaging to get the data in and out of the application. In this case, the modules to be implemented are:

Minimal changes to the UI (Views) and Controller Applications must be able to send data to compatible VO applications (applications who understand the kind of messages the host application wishes to provide), and must declare the kind of messages it can receive from other applications. The main Controller must be able to receive updates from the Messaging part controller.

Messaging interface In this case, messaging is the main building block for the application: other applications will perform the Registry queries, and the calls to data access protocols. The results will be later messaged to the application.

VOTable and FITS handling The messaging protocols use VOTables, which can include links to FITS files, as any other VO protocol. As mentioned above, there are extensive libraries for many different languages which can be used. We can consider, also, that FITS handling is a capability of any legacy application, so only VOTable handling is a true addition.

DAL to internal model interface This interface can be simpler than the one in the prior scenario, because the data origins will be the messaging applications. As we will see later, the messaging protocols impose a more strict semantics to the data being exchanged, allowing for a simpler interface.

⁵<http://www.xmlrpc.com/>

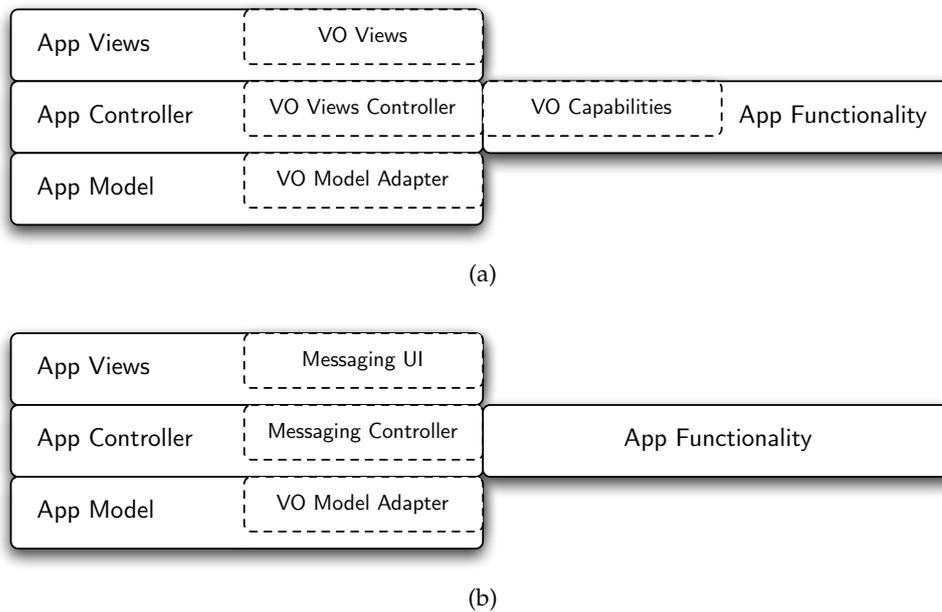


Figure 10.1: Comparison between the functionality to be added and/or modified in a VO-enabled application, both for a monolithic approach (a), and for a messaging-based approach (b).

We can see these two different approaches illustrated in figure 10.1. Figure 10.1a shows the monolithic approach to bringing legacy applications to the VO, with two different legacy applications and an already VO native application. We can see that there is a lot of redundancy and duplication in the development.

Figure 10.1b, on the other hand, shows how a modular interface, which connects legacy applications via messaging protocols, decreases development effort, decouples the development of VO functionality, and provides tools which can be used with any VO messaging enabled application. If we assume that all legacy applications are able to handle FITS files, only messaging, VO Table handling, and DAL to internal data model modifications will have to be performed per application, while the interfaces to the VO, and all external, plug-in like, capabilities, can be left to external modules, common to all applications we might wish to update.

There will be cases of applications where no modifications are possible, because no source code is available for them, or the language they are written in does not support web or XML-RPC interaction. In those cases, a VO downloader application, —which can act as a VO file consolidator— is the *greatest common denominator*, and remains the only way to use non-VO enabled legacy applications within a VO workflow.

10.2 Inter-application messaging in the VO

In order to VO-enable applications through a messaging system, we wish to be able to send messages which entail particular actions: examples of messages and their actions would be:

- Issuing a *data load* message, and having data loaded on the remote application.
- Issuing a *highlight data* message, and have the data highlighted on the remote application.

Without taking into account the nature of the data (in the VO, data is passed by means of VOTables, which might contain data, or link to data), it is clear that there might be several receivers for messages of this nature, so a mechanism for registering potential receivers of messages is needed. If the number of possible messages is moderate to large, an application would also need to declare the kind of messages it can deal with.

We can see that a possible solution to this are publish-subscribe mechanisms, where several parties can act as information publishers, and several (possibly different) parties act as information subscribers. The benefits in scalability and modularity of publish/subscribe systems, together with a thorough study of their mechanisms, taxonomy, and predecessors, can be found in the review by Eugster et al. [87].

The first messaging mechanism within the VO was an experiment by the VOTech project⁶, and was the PLatform for AStronomical Tool InterConnec-tion⁷ (PLASTIC) [88], a client-side messaging protocol.

PLASTIC was based on XML-RPC messaging between client applications and a central hub which had to start before the applications could connect to it. Applications would register the messages they support, and would provide handler functions for incoming messages (callbacks).

The number of defined messages was not very large, and by being implemented on top of XML-RPC it was easily ported to different languages and platforms. In fact, in spite of never being an IVOA standard, the number of applications supporting PLASTIC was very high, as the cost of implementing PLASTIC capabilities was very low.

However, PLASTIC sported a number of shortcomings:

Hard-coded message types and parameters The messages types were hard-coded in the protocol, making messages fixed, and not extensible. Small

⁶In turn, inspired by the XPA (uniX Public Access, <http://hea-www.harvard.edu/RD/xpa/intro.html>) protocol used for communication between tools written for the X11 windowing system, or Tcl/Tk, or Perl packages, such as IRAF, or the SAOImage DS9 FITS viewer. IRAF, for instance, can use DS9 as its imaging package thanks to XPA.

⁷<http://plastic.sourceforge.net/>

modifications to an existing message were not possible, as all parameters were fixed by the message definition.

Non-uniform messages Each message had a number of parameters that depended entirely on the message type, without any governing rule.

Java-based typing Data types were based on Java data types, instead of relying on platform-independent data type definitions.

Hard-coded transport type PLASTIC uses XML-RPC as its transport protocol, making it impossible to use different messaging protocols if the need might arise (for instance, usage of SOAP, e-mail, or other kind of transfer protocol).

The answer to that was taking the best of PLASTIC, which was never an IVOA standard, in spite of its success, and develop a new protocol which answered all of the above shortcomings, while trying to be a drop-in replacement for PLASTIC.

That protocol, an IVOA Recommendation, is the Simple Application Messaging Protocol (SAMP) [85]. SAMP provides a messaging mechanism which is both independent of the actual messages being sent (which are identified by a unique code, the MType, and which have to be standardised between applications), and independent on the transport mechanisms by creating different profiles. The Standard profile, however, uses XML-RPC as its transport mechanism, and resembles PLASTIC by using an XML-RPC based hub where applications register, but with enhanced message semantics.

We will cover SAMP in more detail in the following section.

10.3 SAMP: the Simple Application Messaging Protocol

As its predecessor, PLASTIC, SAMP is a hub-based messaging system, by which a intercommunication hub has to be started before any clients can start sending or receiving messages. Once the hub is started, it waits for client events, until shutdown condition is reached.

Figure 10.2 shows the complete life-cycle for a SAMP hub. First, the hub determines if there is no running, alive hub, before writing (or overwriting) a new SAMP hub discovery record. This discovery record depends on the actual profile (or profiles) supported by the hub⁸.

Once the stage has been set up, the hub waits for events from the clients. Supported operations are client registering (which gives each client a unique

⁸For the standard profile, the hub discovery record is a file called `.samp`, in key=value format, stating the XML-RPC endpoint of the hub, among other properties.

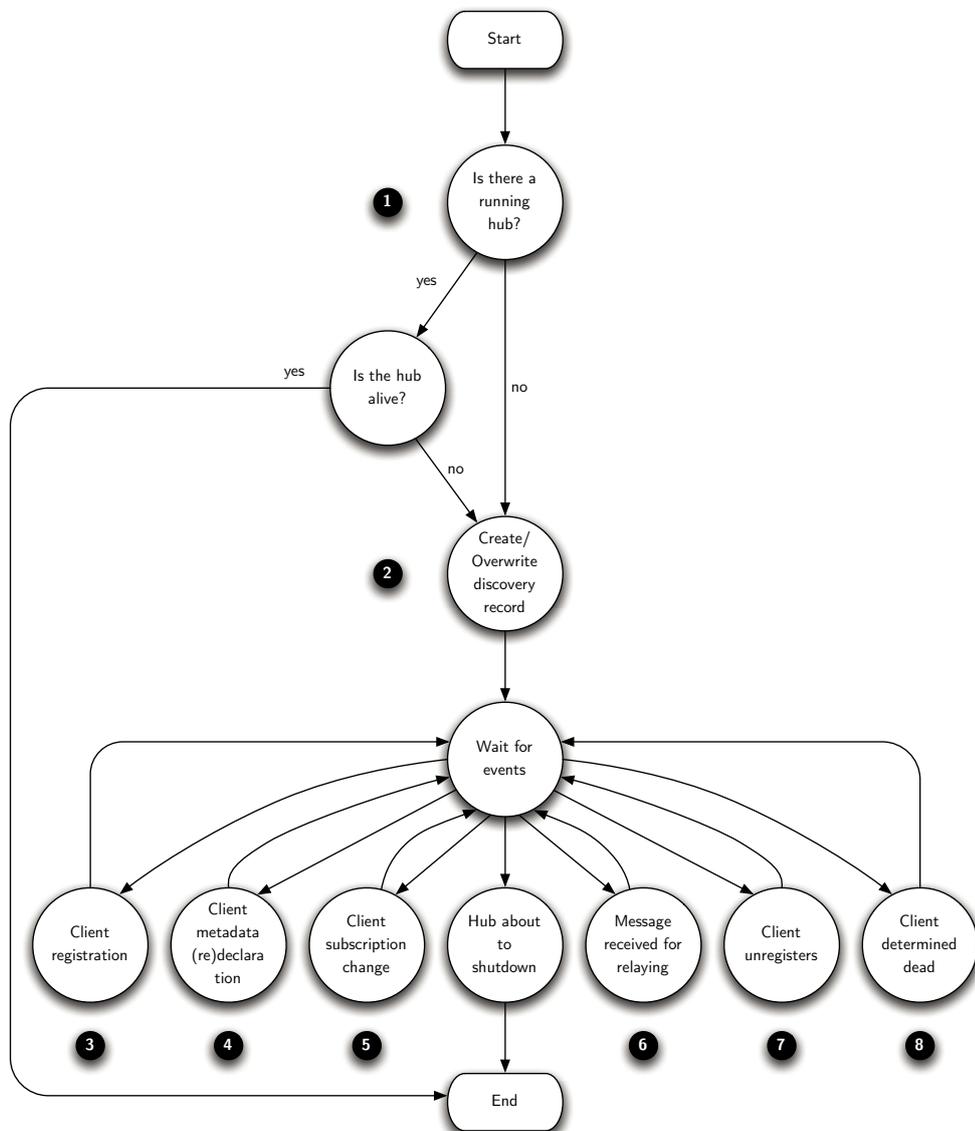


Figure 10.2: Life-cycle of a SAMP hub. Once determined there is no running, alive hub, the discovery record is created, and the hub waits for the different events it supports, until shutdown. The numbers on the black circles will be used to refer to particular steps throughout the text.

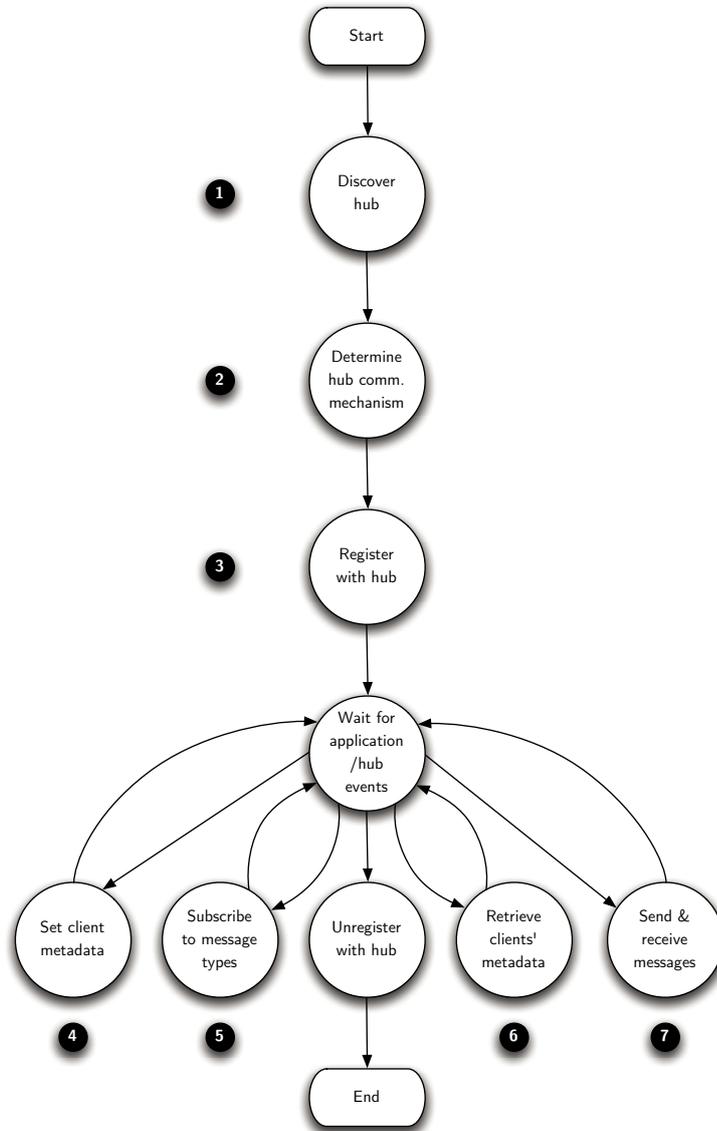


Figure 10.3: Life-cycle of a SAMP client. Once a client has found a hub, it registers with it, and at any time after registration it declares (or changes) metadata, subscriptions, and receives messages based on its subscriptions, until it unregisters with the hub before quitting. The numbers on the black circles will be used to refer to particular steps throughout the text.

id for communication, and identification of subsequent calls), metadata declaration (as many times as each client wishes), subscription to particular message types, message relaying to one, several, or all clients, et cetera. All exchanges between SAMP applications are mediated by the hub, including synchronous calls between SAMP applications.

Finally, when the hub is about to close, notifies all clients of that condition, and finally removes the hub discovery record, so that a new hub instance can start afresh.

Figure 10.3, on the other hand, reflects the life-cycle of each SAMP client. If a hub is discovered⁹, it immediately enters the registration stage, and receives a unique identifier which allows its identification¹⁰ for subsequent messages. For instance, other applications might wish to send individual messages to applications with certain declared capabilities.

The main difference with PLASTIC is at the message definition and subscription level: in SAMP, MTypes can be arbitrarily defined between the applications which understand them, and other applications can be connected to the hub without support for any messages other than those mandatory by the SAMP protocol.

As the application has declared which messages does it subscribe to, and the function which will deal with them, it will only receive messages it can handle. And before the application is ready to quit, it should unregister with the hub. All application quitting events should handle this unregistering process, in order not to leave fake registered applications with the hub.

Given that MTypes can be arbitrarily defined, and semantics and parameters are bound together by agreement between VO developers within the IVOA (for public MTypes), or between application modules (for private MTypes), SAMP can also be understood as a form of type-based publish/subscribe system [87].

10.4 Implementing SAMP into an existing application

We will assume the application we wish to make compatible with SAMP follows the Model-View-Controller (MVC) design pattern, because that is normally the case for GUI applications, and it allows for an easier discussion of the modifications needed.

As the application will become a SAMP client, we need to add the following modules:

SAMP Registration module This module would be added to the start-up code of the application, and would perform the discovery of the hub,

⁹For instance, in the standard profile, by finding a `.samp` file at the user's home directory.

¹⁰Apart from that unique identifier, different profiles might choose additional measures in order to ensure there is no client spoofing. In the standard profile, a private key is generated by the hub and sent back to the client upon registration.

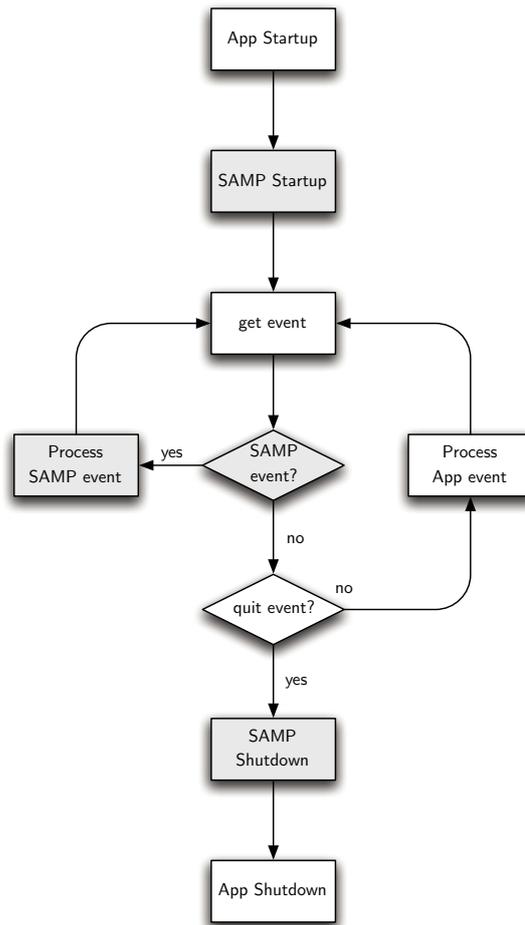


Figure 10.4: Simplified event flow of a SAMP-enabled application. In gray, the modules that have been introduced in order to provide SAMP-compatibility to an existing application. After the application has performed its start-up, but before entering the event-loop, we perform the setup of the SAMP infrastructure. We intercept SAMP events, in order to process them, letting the normal event-handling processing the rest.

the determination of the communication mechanism (in fact, testing that the hub corresponds to the same profile as the client; as we will use XML-RPC for communications, check the hub corresponds to the standard profile), and the registration with the hub (steps 1, 2 and 3 in figure 10.3). In addition, in this phase we can perform an initial declaration of the client metadata (step 4), and of the messages it subscribes to (step 5).

SAMP Message sending Depending on the application capabilities, only a subset of possible MTypes will be sent. We need to create UI elements (buttons, pop-up menus) which provide the user with the possibility of sending data to other applications. Those pop-ups will only show applications accepting the messages we intend to deliver, and for that we will query the hub for clients' metadata (step 6).

SAMP Message handling We need to implement the handlers for the messages we are subscribed to. For similitude with PLASTIC, and for extra modularity, a message dispatching object needs to be implemented, which will handle registered MTypes, which then dispatches the actual message to the corresponding handler, which in turn will make use of existing application functionality to either display or manipulate the received message. This corresponds to handling of step 7.

In addition, some small modifications to the main view Controller will have to be performed, so that incoming messages with data can be dealt with as if an *open file* event had been issued. If the application is well factored, changes to the controller can be inexistent.

The changes needed to the application flow are shown in figure 10.4. The main simplification is the **Process SAMP event** box, which apart from possibly reissuing application-specific events in order to complete event processing (i.e., for finishing a *table load* message with an actual data load, in the internal data format of the application; if the application load messages handle FITS files, the changes would be minor).

10.5 Benefits of a SAMP-based API

By implementing SAMP on an existing astronomical application, we have given it the opportunity to interact with other VO applications, and leave data selection in the VO to external applications.

However, given that SAMP, as a publish, subscribe, and messaging facility allows any kind of messages, by defining ad-hoc MTypes we can create new functionality that responds in particular ways to given MTypes and their parameters. This way, we can create a complete Application Programming Interface, in which instead of providing actual, byte-compiled functions, the functionality is provided via SAMP messages calls and responses.

Of course, given that the standard profile for SAMP uses XML-RPC, we could have created such an API as XML-RPC instances. However, bringing the API to SAMP has the following advantages, both over an XML-RPC API or pure binary API:

System decoupling By building a complete publish/subscribe system, we gain a three-way decoupling of system components [87]:

Spatial decoupling The *spatial* term refers to the fact that neither publishers nor subscribers need to share any kind of space, or shared knowledge. Publishers only need to know how to publish, and subscribers how to subscribe to events.

Temporal decoupling Publishers and subscribers do not need to orchestrate their interaction, and publishing is independent of the delivery of events to subscribers, and the receipt of an event to a subscribers does not need any interaction with the publisher.

Synchronism decoupling In a true publish/subscribe system, such as that provided by SAMP, publishers are not blocked while producing events, and subscribers can obtain asynchronous notifications (via callbacks) of events: neither production nor consumption of messages happen in the main flow of control of the publishers and subscribers, and do not therefore happen synchronously.

Modularity The building blocks for a SAMP-based API are the supported MTypes, or families of closely related MTypes. But in any case, any module can provide support for one or more MTypes. It brings the classical *do one thing well* motto typical from UNIX tools to the VO¹¹.

Service discoverability In order to be able to receive the MType messages which conform the API, functional modules need to register with the SAMP hub. Any application can query the hub, and request a list of the applications (modules) which support particular messages.

Available for all SAMP applications By being based on SAMP, a standard that many VO applications will implement, and thanks to the discoverability of SAMP-based services, we are in fact able to create a plug-in API for all SAMP-enabled, VO applications.

Easy module building SAMP-based computing modules can be built in any computing language and operating system which provides XML-RPC support. In fact, as XML-RPC is an HTTP based RPC, with XML payloads, any language which can create sockets, and establish an HTTP

¹¹When PLASTIC was announced, it was *marketed* in similar terms, but many of the shortcomings of PLASTIC did not allow for a radical, modular development.

Table 10.1: JSAMP message latency tests. We have performed several timing tests with the CalcStorm testing suite of the JSAMP package, in a variety of situations.

clients	queries	messaging kind ^a	time per message
20	50	random	10.2 ± 0.3 ms
20	50	random	14.4 ± 0.2 ms ^b
50	20	notify	10.7 ± 0.2 ms
25	40	random	12.0 ± 0.3 ms
1	1000	random	10.8 ± 0.4 ms
1	2000	sync	10.3 ± 0.3 ms
1	2000	async	12.6 ± 0.6 ms
1	2000	notify	7.9 ± 0.3 ms

^a*sync* stands for synchronous calls; *async* for asynchronous; *notify* for asynchronous notifications, without callbacks; and *random* indicates all message kinds above were issued at random.

^bThis result was obtained with an extra load on the hub caused by an additional testing procedure.

connection, can in principle communicate with SAMP services¹², just by creating the XML as strings, and sending them over the wire in HTTP. This allows for SAMP module creation in the language the developer is more accustomed to, or having the best library for a particular problem.

The main drawback for such a message-based API when compared with a binary API is message latency. Function calls operate at the processor level (or Virtual Machine (VM) level, for VM-based languages such as Java, C#, et cetera), while messaging needs many layers built on top of that.

However, for interactive tasks the latency is well below perception limits, and is the actual computation being performed on the received data which will consume most of the time.

In order to have an actual perception of the kind of time involved, we have used the CalcStorm testing suite found on the JSAMP¹³ Java package. When running CalcStorm, many small clients connect to the hub, which understand messages for adding, subtracting, multiplying and dividing floating point numbers, and start sending calculation messages to all the rest. After execution, the total time elapsed is divided by the number of messages issued.

¹²In that case, the most difficult part is implementing the callback functions.

¹³<http://deployer.astrogrid.org/software/jsamp/>

The results are shown on table 10.1, and were obtained on an Intel Core 2 Duo machine at 2.4GHz, running Mac OS X 10.5.6 and a Java 1.6.0_07 64-bit VM. We can see that in all cases messaging delivers performance which is perfectly in line with interactive needs: even in the slowest case, more than 68 messages per second could be sent and received; for computations taking less than messaging time to complete, real time updates can be provided more than 30 times per second¹⁴. In the best case, more than 130 notifications per second can be delivered.

We can see two additional things: First, the average cost for asynchronous messaging, where the calling application does not sync to the response, is just 22% higher than for synchronous communications. Clearly, for non-immediate results asynchronous messaging is the preferred, more robust solution, but for fast enough calculations synchronous messaging works better.

Second, as the notify test just provides a message, without waiting for computation results, it can be shown that messaging overhead for this very simple computation is around 77%. This includes the creation of the message, parsing of the arguments, and return of the message in XML format. For heavier computations, the corresponding messaging overhead would strongly decrease.

We have to take into account that CalcStorm timing takes into account the time needed to create the clients, register them with the hub, perform all queries, and unregister them. Even when that is taken into account, all of that can be performed in 176 ± 9 ms. Most of the time, there are additional, higher latencies, involved in the kind of interactivity supported by SAMP.

One more thing to note: messaging is eminently parallel, a very desirable feature in the days of multi-core: going from one core to two cores¹⁵, latency jumped from 7.56 ± 0.14 ms to 13.7 ± 0.4 ms, resulting in 1.8 times slowdown, well in line with the core decrease. As each execution unit can reside in different cores, and message parameters have to be copied in order to create the actual XML message, and then parsed again by the receiving end, there are far less opportunities for exploiting core-locality, providing many more opportunities for gains by parallelisation.

10.6 Conclusions

In this chapter, we have shown that, in order to bring astronomical legacy application into the VO one of the less intrusive techniques is building VO-compatible messaging in them, and let other applications and modules perform the actual interfacing with the VO.

¹⁴For comparison, PAL refresh rate is 25 frames per second, and NTSC refresh rate is 30 frames per second, which allow for perfectly smooth animation.

¹⁵With a monitoring application on to see load on each CPU

That results in a faster adaptation of legacy applications to the VO environment, and in a much more modular, and more parallel, development of VO functionality.

However, in order to actually bring applications into the VO we need the following items to exist:

VO Downloader Or file consolidator. As mentioned earlier, all legacy astronomical packages are able to read and/or write FITS files. Being able to retrieve both FITS files, and VOTables for later conversion, are the bare minimum for compatibility with the VO.

VO Registry and DAL module For applications which have been enhanced with SAMP messaging, but have not implemented queries to the VO, a GUI providing access to data services in the VO is the way to access to VO data. In this module data sources will be selected, and once queried on a single or multiple cones, the data provided will be sent by SAMP messaging to accepting parties.

VOTable to FITS converter Similar to the VODownloader, a VOTable to FITS converter is needed for providing VO compatibility with applications which cannot be modified.

Additional API The modules above conform the bare minimum to provide VO compatibility for GUI, SAMP based applications. However, as we can provide units which perform arbitrary computations on the parameters provided (asynchronous messaging supports long execution times, and the called module could issue a different message to the calling application, depending on the semantics of the message sent), we can built arbitrary modules.

In this thesis, we will demonstrate support for the following modules:

Name to coordinates resolver Sesame is a web-service hosted by the CDS which provides coordinates for galactic and extragalactic objects, based on the name of the object. We will provide a module which will understand a series of object solving messages, and will deliver different kinds of answers.

FFT module An FFT module will be developed, which uses both custom MTypes, but also `table.load.*` and `image.load.fits` messages, and sends back the result as corresponding table or image load messages to the calling application, so that arbitrary SAMP applications can make use of this computation facility.

We will provide a Python wrapper to call these modules from Python source code, and some of the modules will be written in Python as well.

However, as the API is based on SAMP messages, any language can be used to write the modules, and the modules can be run simultaneously or one by one, as only the modules supporting the messages and computations we are interested in need to be registered with the hub.

In the following chapter, we will define the actual API, and the implementation, for the modules established above.

Chapter 11

MOVOIR: MOdular Virtual Observatory InteRface, and VO APIs

modular

adjective

- employing or involving a module or modules as the basis of design or construction.

module

noun

- each of a set of standardized parts or independent units that can be used to construct a more complex structure.
- Computing any of a number of distinct but interrelated units from which a program may be built up or into which a complex activity may be analyzed.

interface

noun

- a point where two systems, subjects, organizations, etc., meet and interact.
- Computing a device or program for connecting two items of hardware or software so that they can be operated jointly or communicate with each other.

The New American English Dictionary, 2nd Edition

In the previous chapter we have seen that by creating applications which support VO messaging, using the SAMP messaging protocol, the application can be completely VO-enabled as long as there are outside modules performing certain functions.

In this chapter, we will show a set of such modules, which we call MOVOIR (as acronym for MOdular Virtual Observatory InteRface). We will see which are the messages used by already existing applications, and how can we create our own messages for supporting an interface to the VO, but also the messages defined by the existing applications can be rethought so that the support special services, defining that way a plug-in API for the VO.

An additional remark: in this, and the following chapters, the meaning of the words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL”, when they appear in capital letters, are to be interpreted as described by Internet Engineering Task Force (IETF) RFC 2119¹ [89].

11.1 SAMP Messages and MTypes

SAMP is a hub-based messaging system which supports many different mechanisms for messaging: message broadcasting, point-to-point messaging, and a publish/subscribe scheme where messages are sent to all interested applications, which have declared such interest when registering with the hub.

All of these operations are performed by sending MTypes, which are messages with particular codes, so that they have an associated meaning. In the SAMP standard profile, they correspond to XML-RPC calls to the hub to the corresponding `call`, `callAndWait`, or `notify` methods, provided as XML-RPC services by the hub. In those calls, there is a message parameter which correspond to a map (in the sense of a set of key-value pairs, where the keys are strings) with the following keys:

samp.mtype A string which defines the meaning and parameters of the message. All messages sent with the same `samp.mtype` need to provide the same mandatory parameters, with the same data types, with the same expected behaviour.

samp.params A map of the parameters needed for the correct interpretation of a message with the specified `samp.mtype`. As mentioned above, when an MType is defined, their mandatory parameters have to be defined, too. Keys are strings representing the parameter name, and the data type of the value depends on the actual definition of the MType, but has to be one of the supported data types.

When a message of a given type returns values, they are returned as map with the following keys:

¹<http://www.ietf.org/rfc/rfc2119.txt>

Table 11.1: SAMP Data Types. The corresponding Backus-Naur Form (BNF) for each data type can be found on section 3.3 of the *SAMP IVOA Recommendation* [85]. Data range for SAMP int or SAMP float types depends on the encoding and decoding applications.

Data type	Description
string	alphanumeric data.
list	ordered array of data items of the same type.
SAMP int	a string containing a representation of an integer number.
SAMP float	a string containing a representation of a floating point number.
SAMP boolean	a string containing either 0 or 1, for false or true logical values, respectively.
map	an unordered associative array of key-value pairs, in which each key is a string, and each value is given in one of the supported data types above.

samp.status This is a REQUIRED key. Its value is a string summarising the result of the processing. It may take one of the following predefined values:

samp.ok Signals total success. In this case, the `samp.result` key SHOULD be present, and the `samp.error` key SHOULD NOT appear.

samp.warning Partial success. Both `samp.result` and `samp.error` keys SHOULD be present.

samp.error Processing of the message failed. The `samp.error` key MUST be present, and the `samp.result` MUST NOT appear.

samp.result This key is REQUIRED in case of full (`samp.status` equal to `samp.ok`) or partial (`samp.status` equal to `samp.warning`) success. The value is a map containing the values for the named return values, which are determined by the value of `samp.mtype` (the MType). Even for MTypes which return no value, the key must be present, with its value set to empty.

samp.error This key is REQUIRED in case of full (`samp.status` equal to `samp.error`) or partial (`samp.status` equal to `samp.warning`) error. The value for this key is a map with the following keys:

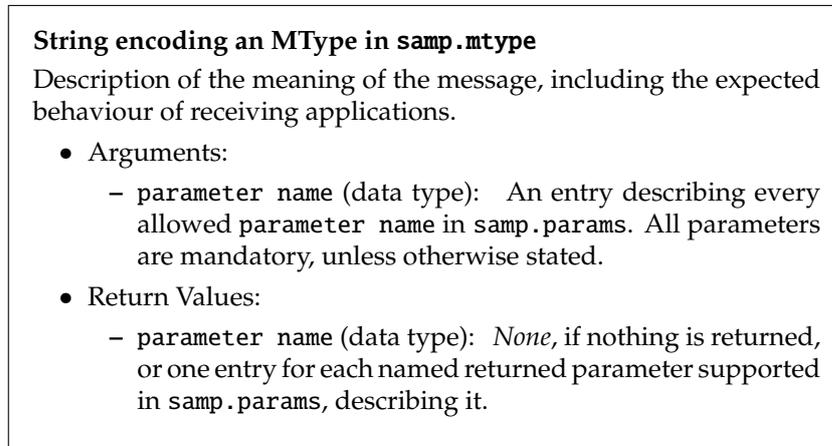


Figure 11.1: Format for describing MTypes.

samp.errortxt This key is REQUIRED in this map. Its value is short string describing the problem, to be presented to the user.

samp.usertxt This key is optional, and its value is a free-form string, with additional text the called application wishes to append to the error. It could be appended to the `samp.errortxt`, but it is undefined what to do with it.

samp.debugtxt This key is optional, and its value is a free-form string of interest for debugging purposes (e.g. a stack trace).

samp.code This key is optional, and its value is a string containing a code (numeric or textual) identifying the error.

In order to enhance interoperability, SAMP data types are specified as encoded strings, instead of having a binary encoding, or using platform specific types such as native XML-RPC int or float types. Allowed data types are shown in table 11.1.

11.2 Standard SAMP message types (MTypes)

As SAMP is an evolution of the PLASTIC messaging protocol, there have been defined some MTypes which represent the kind of messages, with their parameters, that PLASTIC applications were capable of sending.

The Applications Working Group of the IVOA has created a wiki page² for declaring the MTypes being publicly supported by different applications, so that applications can open up for ad-hoc messages. Once this thesis is published, the MTypes for the MOVOIR will be incorporated to this page.

²<http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/SampMTypes>

table.load.votable

Load (possibly display, or otherwise acknowledge the receipt of) a table in VOTable format.

- Arguments:
 - `url` (string): URL of the VOTable document to load.
 - `table-id` (string) *optional*: Identifier which may be used to refer to the loaded table in subsequent messages.
 - `name` (string) *optional*: Name which may be used to label the loaded table in the application GUI.
- Return Values:
 - none.

Figure 11.2: Description of the `table.load.votable` MType.

table.load.fits

Load (possibly display, or otherwise acknowledge the receipt of) a data table in FITS format.

- Arguments:
 - `url` (string): URL of the FITS file to load.
 - `table-id` (string) *optional*: Identifier which may be used to refer to the loaded table in subsequent messages.
 - `name` (string) *optional*: Name which may be used to label the loaded table in the application GUI.
- Return Values:
 - none.

Figure 11.3: Description of the `table.load.fits` MType.

In particular, only the following MTypes are officially supported by clients such as TOPCAT and Aladin, and maintained by the IVOA Applications WG: `table.load.votable`, `table.load.fits`, `table.highlight.row`, `table.select.rowList`, `image.load.fits`, `coord.pointAt.sky`, and `spectrum.load.ssa-generic`. They are shown, following the format of figure 11.1, in figures 11.2, 11.3, 11.4, 11.5, 11.6, 11.7, and 11.8.

We can see in the message descriptions that applications developers' flexibility is encouraged, but an additional property of all of these messages collaborates in that flexibility: none of the MTypes above return anything, and the actions to be carried out by the receiving application upon receipt of

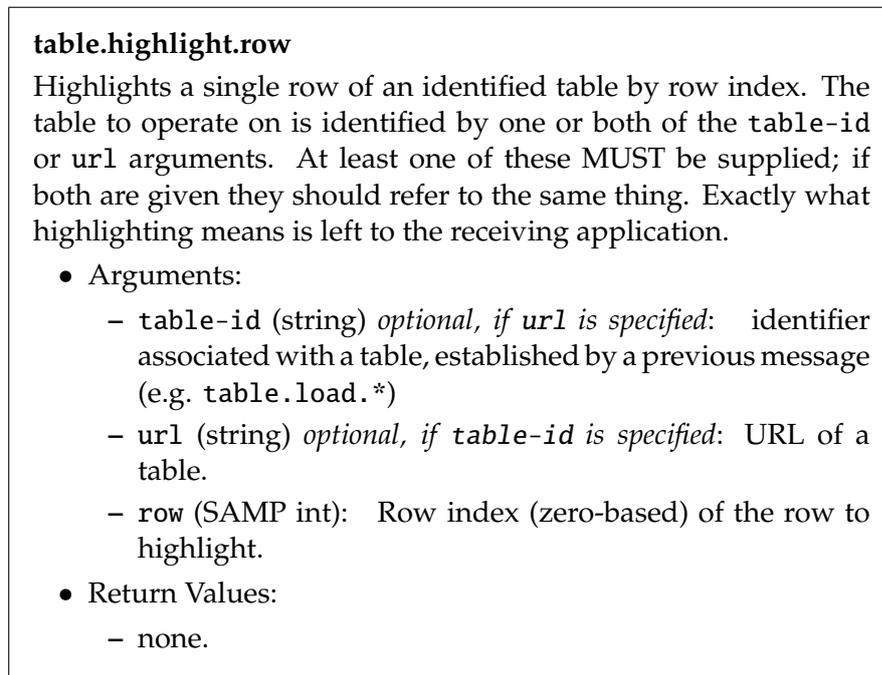
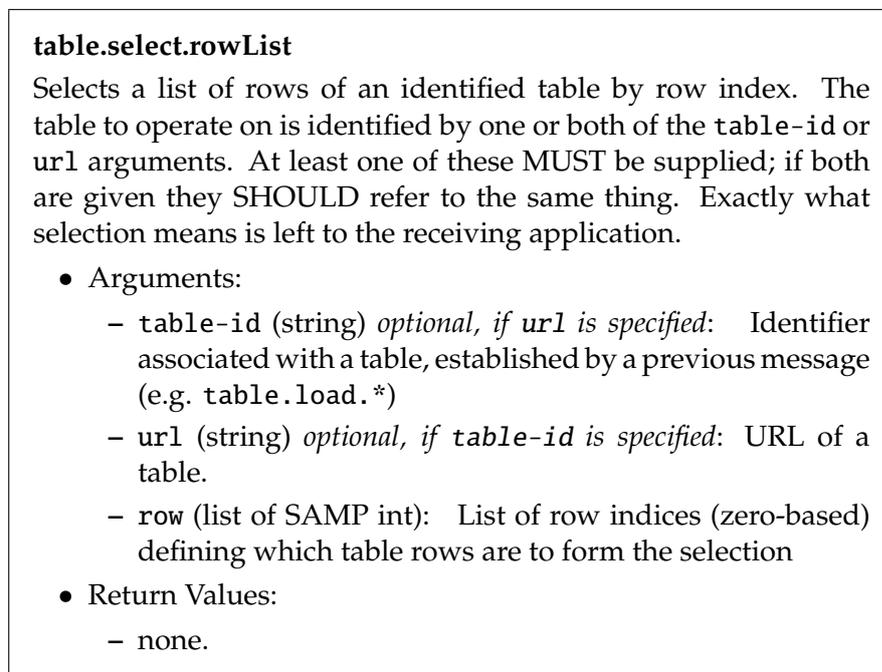
Figure 11.4: Description of the `table.highlight.row` MType.Figure 11.5: Description of the `table.select.rowList` MType.

image.load.fits

Load (possibly display, or otherwise acknowledge) a two-dimensional FITS image.

- Arguments:
 - `url` (string): URL of the FITS image to be loaded.
 - `image-id` (string) *optional*: Identifier which may be used to refer to the loaded FITS image in subsequent messages.
 - `name` (string) *optional*: Name which may be used to label the loaded FITS image in the application GUI.
- Return Values:
 - none.

Figure 11.6: Description of the `image.load.fits` MType.

coord.pointAt.sky

Directs attention (e.g. by moving a cursor or shifting the field of view) to a given point on the celestial sphere.

- Arguments:
 - `ra` (SAMP float): Right ascension in degrees.
 - `dec` (SAMP float): Declination in degrees.
- Return Values:
 - none.

Figure 11.7: Description of the `coord.pointAt.sky` MType.

these messages can be completely arbitrary.

We will use this flexibility to define a set of *behaviours*, upon receipt of standard MTypes, which will aid in calculations, and will provide the API for the MOVOIR operations.

11.3 Creating alternative response patterns

We have seen in the previous section that there are a number of already standardised MTypes, together with their corresponding responses upon receipt

spectrum.load.ssa-generic

Load (possibly display, or otherwise acknowledge) a spectrum or SED. The name refers to the fact that the metadata passed with this MType is based on the Simple Spectral Access protocol, but not on any particular version of it. The arguments are chosen such that it is convenient to use this MType for passing the results of an SSA query from an SSA client to a spectrum viewer (particularly to an SSA-capable spectrum viewer). However it is not necessary for SSA to be involved; SSA-like metadata may be faked and used to message loading of a spectrum from any source. In the latter case it is RECOMMENDED to provide at least the `Access.Format` entry in the meta map.

- Arguments:

- `url` (string): URL of the spectrum to load
- `meta` (map): Additional metadata describing the spectral data found at the URL. Key/Value pairs represent either Utypes or UCDs as defined or used in some version of the SSA specification or its predecessors. Example map keys are `Access.Format` (SSA 1.0 MIME type Utype) or `VOX:Spectrum.Format` (pre-1.0 SSA MIME type UCD). It is up to the recipient to make sense of these and, for instance, deal with the possibility that given expected keys are present or that apparently contradictory information is presented. Most existing SSA-aware spectrum viewing clients already contain this functionality.
- `spectrum-id` (string) *optional*: Identifier which may be used to refer to the loaded spectrum in subsequent messages.
- `name` (string) *optional*: Name which may be used to label the loaded spectrum in the application GUI.

- Return Values:

- none.

Figure 11.8: Description of the `spectrum.load.ssa-generic` MType.

of those MTypes. For all of those messages applications returned nothing³, and there were clear indications that applications should perform certain tasks upon receipt of such MTypes, but there is nothing the caller application can do in order to ensure a particular response, and this is a feature of this messaging system: applications are notified of an event, and what to do upon reception is completely up to them.

One clear case of SAMP-based application which subscribes to the messages above, but does not really perform the actions specified by the MType description, are logger application. A typical logger application would register with a SAMP hub, and subscribe to every possible message. However, for all messages received the message handler is the same, and just logs the `samp.mtype` and `samp.params` of the call, together with information about the caller application.

The lesson to learn from this example is that applications which subscribe to broadcasted messages, and specially those who require no answer, are completely transparent to caller applications: we can perform any action it makes sense to perform upon receipt of these messages, as long as the formal behaviour expected by the calling application (in terms of answering a synchronous message, not letting the calling application blocked, filling all required keywords in the response...) is fulfilled.

Let us explore a few examples of new functionality which can be added by two means: performing a somewhat different action upon receiving one of the standard messages, or letting called applications see who has called them, and send messages back to applications which support them.

Modifying or enhancing response actions

The first case (modifying what is done by the receiving applications) was illustrated by our previous example of a logger application, and we will also use this way to create a VODownloader application.

A VODownloader is, essentially, an application which, instead of actually working with the VOTables and FITS files which are sent to it, it downloads and saves them to a particular folder. This requires being able to respond to the `table.load.votable`, `table.load.fits`, and `image.load.fits`, and use the `url` parameter as a data source to download the file sent. This changes somehow the meaning of the `table.load.*` and `image.load.fits` MTypes, but in a sense which is compatible with the message definition, and provides a valuable service for integrating legacy applications for which there is no access to source code, and which cannot be integrated into the VO. For them, a VODownloader application is the only possible data access bridge to the VO.

³Formally, asynchronously called modules which return nothing SHOULD call the callers' reply method with a map with keys `samp.status` set to `samp.ok` and `samp.result` set to an empty map, but applications SHOULD NOT behave incorrectly if they do not receive it.

Performing same-message call-backs

When sending an asynchronous message, the caller application provides a function the called service needs to call back in order to provide the actual result.

However, that is only possible for asynchronous messages with MTypes expecting return values, and none of the standard messages provide nothing back. Of course, we can create our own, custom MTypes, but those will only be useful for the subset of applications which know about those additional MTypes. If we want to provide an extension mechanism for already existing VO applications, and for those applications whose only connection to the VO is messaging, we need to provide extensions through the existing MTypes.

Nonetheless, SAMP provides the mechanisms for that: as described in section 3.12, a callable client must support the `receiveNotification`, `receiveCall`, and `reply` methods, and all of them receive a `sender-id` parameter.

This allows called applications to send messages to the clients who called them first, upon receipt of a particular message.

In this way, we can think of different possibilities upon receipt of a given MType which can be replied by sending back the same MType:

image.load.fits In this message, only the `url` parameter is mandatory.

Upon receipt of such a message, the called module would perform some processing: for instance, image inversion, rotation, reflection, automatic level adjustments, adaptive wavelet filtering, FFTs, et cetera. Then, it would send the application which sent the original message another `image.load.fits` MType with the processed image.

table.load.votable Again, in this message only the `url` parameter is mandatory. Upon receipt of such a message, the called module would perform some processing: for instance, FFT of VOTable columns (in order to get spectra from auto-correlation data), statistics on the columns (min, max, average, median, standard deviation, et cetera), et cetera. Then, it would send the application which sent the original table another `table.load.votable` MType with the processed table.

In order for users to discover the actual behaviour of each module, the actual processing being performed (including default parameters being applied) SHOULD be available via the `samp.description` metadata declared with the hub.

Performing different-message call-backs

We are not restricted to create responses as messages of the same time to be back to the calling client. We might have, for example, a module which

takes an `image.load.fits` MType, and uses SExtractor⁴ to create a catalogue of sources detected on that FITS. The catalogue could be sent to other table loading applications (e.g. TOPCAT, but it could be broadcast to all subscribers of the `table.load.votable` MType) for display via a `table.load.votable` message, or simply stored as a file in a particular folder.

Another interesting possibility is that of receiving `ra` and `dec` coordinates via a `coords.pointAt.sky` MType, and performing a ConeSearch query on those coordinates for selected services, sending results back as a `table.load.votable` message.

With the schema described up to now we could only build modules limited to process just one input, be it a table, an image, or a pair of coordinates.

However, we can push this concept further, so that SAMP modules which need to operate on two inputs would wait for two messages of the same type from the same client, with different data, in order to use both data inputs.

One such example would be a module for cross-matching sources from two different catalogues. The sequence, in this case, would be:

1. An application (e.g. TOPCAT) sends a VOTable to an application (which we can call XMatch) which supports the `table.load.votable` MType.
2. XMatch starts preparing for a XMatch of that first table with another table yet to be sent by the same client.
3. The same application sends a second VOTable to the XMatch application via the `table.load.votable` MType.
4. XMatch performs a cross-matching of both tables, using default parameters, creating a third table, with all fields from both tables for sources which were successfully cross-matched.
5. When the cross-matching table has been created, a message is sent to the caller application using the same `table.load.votable` MType.

Such an XMatch module would be able to provide cross-matching services to every SAMP-enabled application able to send `table.load.votable` MTypes, as most applications being able to send `table.load.votable` MTypes can handle them as well. This means that just by supporting the standard messages, applications can be enhanced by modules which can act upon receipt of standard messages. We have, in fact, created a plug-in API for the VO.

This operation model has the drawback of needing a series of default parameters which cannot be specified by the calling applications, as they are using a generic message for loading data.

⁴<http://terapix.iap.fr/soft/sextractor/>

That problem could be overcome in two ways: first, as the data is sent in the VOTable format, if there are additional metadata on spatial resolution, we can build *intelligence* into the cross-matching module to let it adjust its default parameters from the available information.

Second, specially formatted VOTables could be sent to the modules via the same `table.load.votable` messages. The formatting would include precise field names corresponding to the parameters to be set for processing data from subsequent messages.

A final goal of this thesis would be the development of an IVOA-sanctioned plug-in architecture, where specific MTypes would be created for getting application capabilities (default values set-up, methods for discovery of plug-in messages, et cetera) the creation of MTypes specific for handling these module properties.

In that case, another tool which could be performed would be the automatic creation of messages for applications, by just selecting the message, and providing links to the data being created.

In addition, by publishing into the IVOA wiki the messages used by the MOVOIR, they could become more generalised, and available from many more modules.

11.4 Describing MType parameters

The main idea permeating this part of the thesis is that of using SAMP as the way to provide a VO API to enhance existing VO applications, and those applications which have been updated to make use of SAMP messaging.

In order for applications and developers to have access to the different functions and MTypes available from different active SAMP applications and/or modules, a discovery mechanism is needed.

The SAMP hub aids in the discovery of available modules and of the different MTypes supported, but provides no way for applications to discover mandatory and optional parameters for particular MTypes. And the ability to get the data types, and possibly meanings, for each parameter, should also be taken into account.

Hence, we propose a new MType, `movoir.describe.mtype`. The name has been chosen using the guidelines for MType identifiers set on the SAMP MTypes wiki page⁵. This MType definition is shown in figure 11.9.

We will show examples of use of the `movoir.describe.mtype` message, with both the constructed and received message map shown. For instance, the result for the `table.load.votable` message is shown in figure 11.10.

Figure 11.11, on the other hand, shows the result of calling `movoir.describe.mtype` with the additional `verbose` parameter, while figure 11.12 shows

⁵<http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/SampMTypes>

movoir.describe.mtype

Applications receiving this message need to provide a description of the MType specified by the `mtype` parameter, including message parameters, their type, and whether they are optional or not, and the result map, also including message type. If the application does not support that MType, it **MUST** return an error condition.

- Arguments:
 - `mtype` (string): MType identifier to be described.
 - `verbose` (SAMP boolean) *optional*: Boolean string stating whether the answer should be plain (`false`), or verbose (`true`), indicating an extra description is desired for parameters and the MType. Default value is `false`.
- Return Values:
 - `parameters` (map): Map where the keys are the different parameters supported by the specified `mtype`, and values are maps with three mandatory keys: `type`, giving the SAMP type —as in table 11.1— of the parameter; `optional`, a SAMP boolean stating whether the parameter is optional or not; and `ucd`, which provides a UCD for the kind of parameter. If `verbose` is set to `true`, an additional `movoir.description` key **SHOULD** appear for each parameter.
 - `movoir.result` (map) *optional*: If present, it indicates the return values provided by the `mtype` MType, in the same way as parameters are described. It **MUST** appear for MTypes with return values.
 - `movoir.description` (string) *optional*: If present, it provides a general description of the MType being analysed. It **SHOULD** appear when the `verbose` parameter is set to `true`.

Figure 11.9: Description of the `movoir.describe.mtype` MType.

```

samp.status : "samp.ok"
samp.result :
  parameters :
    url :
      type : "string"
      optional : "false"
      ucd : "meta.url"
    samp.mtype :
      "mouvoir.describe.mtype"
    samp.params :
      mtype :
        "table.load.votable"
        (a)
      table-id :
        type : "string"
        optional : "true"
        ucd : "meta.id"
      name :
        type : "string"
        optional : "true"
        ucd : "meta.name"
        (b)

```

Figure 11.10: Example of use of the `mouvoir.describe.mtype` message to learn how a particular message works for a particular application. Subfigure (a) shows the map built to send the `mouvoir.describe.mtype` message, while (b) shows the resulting map. Compare this result with the description of the `table.load.votable` MType in section 11.2.

the result of calling `mouvoir.describe.mtype` with itself as MType. This example allows us to see how to handle both maps as parameters and results. For maps, additional information needs to be provided in the form of UCD so that applications can learn automatically how to handle the map.

In any case, the textual descriptions allow any SAMP developer either to provide a similar support for a given MType, or to create clients able to send those messages.

The most interesting feature of the `mouvoir.describe.mtype` message is that it depends on the application being called: in particular, the descriptions of the message correspond to the use of the parameters and of the return values (if any) by that particular module.

Figure 11.12 also shows how to complement the UCD1+ vocabulary with special vocabulary for parameters very specific to SAMP: for instance, there is no UCD1+ —and there should be none— to specify a SAMP result, so having either its own form (as in `mouvoir.result`), or juxtaposing UC1+ terms with custom terms (as in `meta.code`; `samp.mtype`), allows to maintain UCD semantics as much as possible, while adding a few atoms for clarity.

Thus, the controlled vocabulary for the `ucd` key is the union of the UCD1+

```

samp.status : "samp.ok"
samp.result :
    movoir.description :      "Load (possibly
                                display, or otherwise acknowledge
                                the receipt of) a data table in FITS
                                format."
    parameters :
        url :
            movoir.description : "URL of the
                                    FITS file to be loaded."
            type : "string"
            optional : "false"
            ucd : "meta.url"
        table-id :
            movoir.description : "Identifier
                                    which may be used to refer
                                    to the loaded table in
                                    subsequent messages."
            type : "string"
            optional : "true"
            ucd : "meta.id"
        name :
            movoir.description :      "Name
                                    which may be used to label
                                    the loaded table in the
                                    application GUI."
            type : "string"
            optional : "true"
            ucd : "meta.name"
samp.mtype :
    "movoir.describe.mtype"
samp.params :
    mtype :
        "table.load.fits"
    verbose :
        "true"
    (a)
    (b)

```

Figure 11.11: Another example of use of the `movoir.describe.mtype` message. Subfigure (a) shows the map built to send the `movoir.describe.mtype` message with the `verbose` parameter set to true, while (b) shows the resulting map. In this case, we have additional `movoir.description` keys both for parameters and the message itself.

```

samp.status : "samp.ok"
samp.result :
  movoir.description : "Applications receiving this message need to
    provide a description of the MType [...]"
  parameters :
    mtype :
      movoir.description : "MType code to be described."
      type : "string"
      optional : "false"
      ucd : "meta.code; samp.mtype"
    verbose :
      movoir.description : "Whether to add movoir.description
        [...]."
      type : "SAMP boolean"
      optional : "true"
      ucd : "meta.code"
  movoir.result :
    movoir.description :
      type : "string"
      optional : "true"
      ucd : "meta.note"
    movoir.description : "Provides a general description of
      the MType being analysed[...]"
    parameters :
      type : "map"
      optional : "false"
      ucd : "movoir.parameters"
    movoir.description : "Map whose keys are the different
      parameters supported by the specified mtype [...]."
    movoir.result :
      type : "map"
      optional : "false"
      ucd : "movoir.result"
    movoir.description : "If present, it indicates the return
      values provided by the mtype MType [...]"

```

Figure 11.12: A more complex example of use of the `movoir.describe.mtype` message, where the MType being describe has parameters and provides results which themselves are maps. Compare this (abbreviated) result to the description of `movoir.describe.mtype` at the beginning of this section.

vocabulary with all terms starting with `samp.*`—interpreting the asterisk (*) as a wildcard— defined in the SAMP Recommendation [85], plus the following terms specific for the MOVOIR (but which might be adopted/adapted by the IVOA):

`movoir.result` Identifies a map of MType result keys.

`movoir.parameters` Identifies a map of MType parameter keys.

11.5 Providing default values and settings to modules

We had also identified the problem of setting default values for data processing algorithms implemented on top of SAMP/MOVOIR. For that, we propose two messages:

`movoir.configuration.set` This message sends a `url` to a VOTable with the different settings to be used by the module. An optional `mtype` parameter can be used to restrict settings to the behaviour for a particular `mtype`. In this case, the application SHOULD NOT take into account settings which configure behaviours for different `mtypes`. In the case of asynchronous or synchronous calls to this message, the return value SHALL contain a `url` to the new current values. If settings for different `mtypes` were discarded, the application `samp.status` should be set to `samp.warning`.

`movoir.configuration.get` This message will return a `url` key pointing to a VOTable containing the different parameters which can be set, and their current values, similar to that returned after a `movoir.configuration.set` message.

In particular, the MType definitions for these two messages are shown in figures 11.13 and 11.14.

11.6 MOVOIR modules to implement

After having presented how different SAMP-based processing strategies can be built, we will present here the different MOVOIR modules, and their corresponding MTypes they support with the actual behaviour being implemented.

In particular, we have implemented the modules which were suggested in section 10.6 of the previous chapter:

VO Downloader The VO Downloader is a SAMP application which can receive `table.load.votable`, `table.load.fits`, and `image.load.fits`

movoir.configuration.set

Applications receiving this message will use the `url` parameter to retrieve a `VOTable` with two fields: parameter name, and parameter value (using `SAMP` types). These will be used to set those parameters affecting the behaviour of the module receiving the message. to provide a description of the `MType` specified by the `mtype` parameter, including message parameters, their type, and whether they are optional or not, and the result map, also including message type. If the application does not support that `MType`, it **MUST** return an `samp.error` condition.

- Arguments:

- `url` (string): URL to the `VOTable` containing the values to be set. `FIELDS` with a `utype` attribute equal to `movoir.parameter.key` will be considered to contain parameter names to be set, and `FIELDS` with `utype` equal to `movoir.parameter.value` will be considered as the value to set. Any additional fields will be discarded.
- `mtype` (string) *optional*: String containing a single `MType` to which the parameters to be set conform

- Return Values:

- `url` (string) *optional*: For synchronous or asynchronous messages, the returned `url` parameter holds a URL to a `VOTable` with the new settings of the application. In addition to the two fields with `movoir.parameter.key` and `movoir.parameter.value` `UTypes`, a `FIELD` with name equal to `type`, and `utype` equal to `movoir.parameter.type` specifies the `SAMP` type of the parameter. If the parameter is only relevant to a particular `MType`, an additional field with name equal to `mtype`, and `utype` equal to `samp.mtype` should be provided.

Figure 11.13: Description of the `movoir.configuration.set` `MType`.

movoir.configuration.get

Applications receiving this message will provide a result with a `url` key, which should point to a `VOTable` with four fields: parameter name, parameter value, parameter type (using SAMP types), and parameter `mtype` (empty for parameters not specific upon particular `mtypes`). If a non-supported `mtype` is specified in the optional `mtype` parameter, the `samp.status` is set to `samp.warning`.

- Arguments:
 - `mtype` (string) *optional*: String containing a single `MType` to which the parameters of interest relate to.
- Return Values:
 - `url` (string): URL to a `VOTable` with the current settings of the module. In addition to the two fields with `movoir.parameter.key` and `movoir.parameter.value` `UTypes`, a `FIELD` with name equal to `type`, and `utype` equal to `movoir.parameter.type` specifies the SAMP type of the parameter. If the parameter is only relevant to a particular `MType`, an additional field with name equal to `mtype`, and `utype` equal to `samp.mtype` should be provided.

Figure 11.14: Description of the `movoir.configuration.get` `MType`.

`MTypes`, and downloads the received file. If a `name` parameter is provided, the file is renamed to that filename. A `movoir.vodownloader.dlpath` configuration key in a `.movoir` file is used to set the download path, while the current directory from where the VO Downloader has been launched. In addition, the VO Downloader application can send the downloaded file back to any other `table.load.votable`, `table.load.fits`, or `image.load.fits` subscriber, depending on file format.

VO Registry and DAL module As the VO Desktop application has been ported to SAMP, we will use the VO Desktop as Registry access and DAL module: only the data results will need to be sent to SAMP applications. Any other SAMP-enabled registry and VO data access layer front-end can be used instead.

VOTable to FITS converter Similar to the VO Downloader, the `VOTable` converter is an application which can receive messages with the `table.load.votable` `MType`, and creates and saves the converted FITS

file in the same path as the VO Downloader. In addition, the VOTable to FITS converter interface allows the user to send the converted FITS file to SAMP applications supporting the `table.load.fits` MType.

AMIGA ConeSearch module This module uses the `coord.pointAt.sky` MType to provide results of a ConeSearch of 0.5 degrees radius around the ra and dec coordinates provided. It is also able to understand an optional, non-standard radius parameter, to set the radius for the ConeSearch in degrees.

Sesame query module MOVOIR provides a multi-object Sesame query message, as support for a future multi-cone search service, in order to resolve multiple object names at once. This is another example of a service which can be easily implemented in any language—in this case, Python has been used—and which can be called from languages with support for XML-RPC calls.

All MOVOIR modules will also support the `movoir.describe.mtype` message, as described in the previous section, in order to provide detailed description not only on available parameters, or

11.7 Salient features of the MOVOIR

By the way it has been developed, the MOVOIR has the following properties:

Multi-language support Astronomical data tools used to be written in FORTRAN, and compiled for UNIX-like systems. As of late, many astronomical applications, specially if they need to communicate with the VO, are being written in Java or in advanced scripting languages such as Python, which are available not only for UNIX-like systems, but for common platforms such as Windows or the Mac. The MOVOIR has parts written in Java, and in Python, but the interface is common to any kind of application supporting XML-RPC calls. In particular, it will support language-independent VO application messaging—see section 2.7—.

Multi-platform support Stemming from the usage of platform neutral languages, such as Java and Python, the MOVOIR can be used in every major platform (Linux, Mac, Windows), and in any other platform supporting any language with an XML-RPC library, or even just able to create communication sockets.

Reuse of existing packages The MOVOIR uses the STIL table library⁶ for its internal table operations, and the AstroRuntime (the server part of the

⁶<http://www.star.bristol.ac.uk/~mbt/stil/>

VO Desktop) for VO query support. The application-specific part of the MOVOIR uses also the STIL table library and the `nom.tam.fits`⁷ Java library for Java applications, and the standalone STILTS package⁸ and the PyFITS⁹ for Python applications. This allows for the combination of MOVOIR features with those independently exposed by the STIL and AstroRuntime, and the combination of scripts supporting both tools with those making use of MOVOIR's capabilities.

Single-operation, Multiple-targets The MOVOIR is able to work on multiple targets at the same time in every function which would normally expect a single target. This allows more complex workflows to be built, and to enhance query throughput.

Data Oriented Interface Most VO applications tend to be built around the existing Data Access Layer, or the VO protocols used for data access. They also treat the coordinate resolution systems such as Sesame, or Registry queries, as separate parts of the system. The MOVOIR interface is data focused, and filtering is performed from the coordinates first, with particular queries fitted to each particular service. This is also true for the scripting interface.

Dynamic module update As the MOVOIR uses a hub based mechanism for module registration, modules can be added or removed dynamically, and SAMP-enabled applications expect registered applications to change over time. This makes modules newly registered with the hub available to all applications able to send the MTypes the new modules are subscribed to. This feature eases development, deployment, and the updating of the MOVOIR, or any other SAMP-based package.

Publish/subscribe, message-based API As the MOVOIR uses the Simple Application Messaging Protocol (SAMP) as the main interaction bus, all operation requests are sent as SAMP messages, and the synchronous or asynchronous responses are also in the form of SAMP messages. This decision allows for the definition of an initial *plug-in API* which is message-based, instead of language- or library-based.

11.8 Main issues

The main issues with the MOVOIR approach to enhancing VO applications are the following:

⁷<http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/java/v0.9/>

⁸<http://www.starlink.ac.uk/stilts/>

⁹http://www.stsci.edu/resources/software_hardware/pyfits

SAMP UI scalability

The main problem with the MOVOIR is its scalability as a plug-in API. As the number of modules included with the MOVOIR, together with the number of SAMP applications, increases, so does the time needed by the user to find the correct module for sending the data.

The modularity of the MOVOIR helps, as modules do not need to be launched until they are needed, and are available right after launch in all applications, but still the problem of a likely increase of the number of SAMP-enabled applications and modules remains.

Possible UI solutions to that problem would be:

SAMP specific communication panes Instead of using a simple pop-up menu, a SAMP-specific pane could be used, either as a floating or auxiliary window, or as a pop-up drawer, which allowed for a better access to metadata descriptions not just in the hub UI, but also in the client.

Hierarchical display of SAMP Mtypes Applications could show available applications by the kind of message they can handle. However, only for SAMP standard messages there is a clear identification of the data type they can receive: for custom messages, such as those supported by the MOVOIR, only if all applications support the `movoir.describe.mtype` message can data types be discovered.

High delay, unexpected responses

Some users might think that the MOVOIR can create “non-causal” responses. That is, users might send a message to a module which takes a long processing time, and even reiterate the message, and then the application they are using might show two tables, or two images, that they are no longer expecting.

In order to avoid this processing time should be less than real time expectations. The typical golden rule for interactive systems is delivering a response in under 8 to 15 seconds¹⁰. Independently of the actual value, responses taking longer than that threshold might not be perceived as related to the initiating task, startling users. In any case, systems taking longer than one minute to process data should use messaging just to retrieve data, providing their own UI to initiate the processing and send results back. They should also provide progress information during the task processing, and either let the user send the result once it is available, or identify results with the originating task.

¹⁰<http://www.ariadneproject.org/index.php?id=63>

11.9 Conclusions

In this chapter we have shown how a data processing API can be built on top of the SAMP messaging protocol, and what ideas can be developed in order to create an extension mechanism for already existing VO applications.

The main achievement for the MOVOIR, being based on SAMP, is that it provides a way to create plug-ins for existing VO applications, which can be made independent of both the platform and language those modules can be written into, and completely independent of applications being enhanced.

We have also proposed a few extra messages which can be used to learn about plug-in parameters, and configurability, with the aim to start a discussion within the IVOA on what would be the best one to implement such a plug-in API.

Part IV

Thesis applications

Chapter 12

Implementations of RADAMS-based radio astronomical archives

The RADAMS was developed with two main objectives in mind:

- Providing a data model for the development of the DSS-63 and IRAM 30m antennas' VO-compliant archives; and
- Providing the VO with classes for supporting radio astronomical observations.

The latter objective was illustrated in the discussions of the Curation, Packaging, Policy, and Data Provenance parts of the Observation Data Model in chapters 7 and 9.

We will devote this chapter to the former objective, and we will show how the RADAMS has been used as a blueprint in order to create two different VO-compatible radio astronomical archives, and how features of each antenna have modified the RADAMS.

12.1 The Robledo DSS-63 archive

The DSS-63 is one of the antennas at the Madrid Deep Space Communications Complex. Three Deep Space Communications Complexes (DSCCs) were created by NASA in the late 50's, and were located at Canberra (Australia), Madrid (Spain) and Goldstone (USA), in order to allow for continuous monitoring of the incoming data from Earth-orbiting and interplanetary spacecraft missions, as well as radio and radar astronomy observations for the exploration of the Solar System and the Universe. The combined operation of the

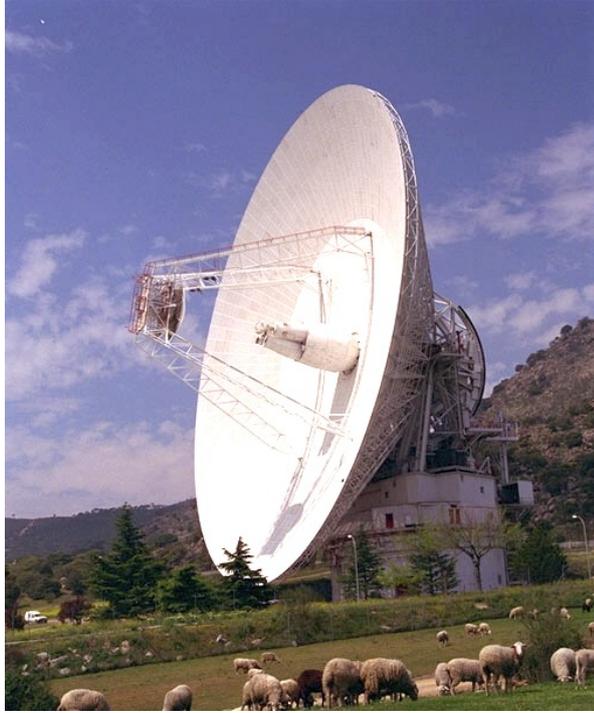


Figure 12.1: DSS-63 70-meter antenna at Robledo de Chavela, Madrid.

Table 12.1: DSS-63 properties, versus other antennas; values at 22 GHz.

Telescope	Diameter	Aperture efficiency (η_a)	Resolution	Sensitivity (K/Jy)
Effelsberg	100 m	29%	40"	0.8
GBT	100-110 m	55%	34"	1.5
Robledo DSS-63	70 m	49%	42"	0.7

Table 12.2: Antenna, receiver and spectrometer properties of DSS-63.

Antenna System^a	Name: Deep Space Station 63 Diameter: 70 m Type: parabolic Cassegrain Mount: azimuth/elevation Latitude: 40° 25' 52" N Longitude: 04° 14' 53" W Altitude: 865.5 m HPBW^b: 42" Aperture Efficiency (η): 49% maximum Sensitivity: 0.7 K/Jy Pointing accuracy: $\leq 10''$
K-band receiver	Amplifier type: cooled HEMT Frequency range: 18-26 GHz Polarization: LCP ^c or RCP ^d (default, LCP) T_{sys} (winter): 50 K T_{sys} (summer): 75 K
Spectrometers^e	Spaceborne-500 Type: Digital autocorrelator BW: 2, 4, 8 and 16 MHz Num. channels: 384 Observing mode: position switching Spectra-Data Type: Fourier-transform autocorrelator BW: 1, 2.5, 5 and 10 MHz Num. channels: 256 Observing mode: frequency switching SAO4K Type: Digital autocorrelator BW: 400 MHz Num. channels: 4096 Observing mode: position switching

^aHPBW, aperture efficiency, sensitivity and pointing accuracy measured at 22GHz, with 40° of elevation

^bHalf-Power Beam Width

^cLeft Circular Polarisation

^dRight Circular Polarisation

^eThe Spaceborne-500 is the correlator currently in use. It superseded the Spectra-Data, the first correlator available in this telescope (2001 to 2003), and is no longer in operation. The SAO4K, on the other hand, belongs to the Smithsonian Astrophysical Observatory, and is used for the SAMBA survey.

three DSCCs is what it is known as the Deep Space Network (DSN), which is managed by the Jet Propulsion Laboratory (JPL).

Each DSCC has at least four operational antennas:

- One 26-meter diameter antenna, originally built to support the Apollo missions to the Moon, presently used for communicating with Earth-orbiting spacecraft.
- One 34-meter diameter high efficiency antenna (HEF), designed around a precision-shaped reflector, for maximum signal sensitivity.
- One 34-meter diameter beam waveguide antenna (BWG), based on the HEF design, with five mirrors that reflect radio signals along a beam-waveguide tube from the antenna vertex to the equipment room, for easier maintenance access.
- One 70-meter diameter antenna, with the highest sensitivity, used for tracking the deepest space missions.

In the case of the Madrid DSCC, the 70m antenna is DSS-63, whose picture is displayed in figure 12.1. Due to their high sensitivity in centimetre wavelengths, they can be used to perform astronomical observations, when they are not following NASA vehicles .

Of all the time devoted for scientific observations, around 3% of the time at the Canberra and Madrid stations (up to 260 hours per year and per antenna) is available to Host-Country astronomers. The organisation responsible for the scheduling of this time at Madrid DSCC is the Centro de Astrobiología (INTA-CSIC), by arrangement with NASA.

Table 12.1 compares some properties of DSS-63 with those from other astronomically oriented radio telescopes, while table 12.2 summarises DSS-63 properties of the antenna system, the K-band receiver, and the different spectrometers which have been installed at the the DSS-63 antenna.

Nowadays, Host Country time at the MDSCC is devoted to perform spectroscopic observations at K-band (i.e., wavelengths around 10 cm), with the 70-m DSS-63 antenna. In particular, several projects for observing H₂O masers have been performed —see, for instance, Gregorio de Monsalvo’s thesis [90]—, and the team wished to make those spectroscopic archives public.

Spectral observations with the DSS-63 antenna

As the main scientific use of the DSS-63 antenna is the recollection of spectra, we will describe how spectroscopic observations are performed with this instrument.

The observing process for a spectrum is as follows:

Source selection First, a target source with a medium elevation at the time of observation is selected; extreme elevations introduce additional pointing errors and/or additional atmospheric effects.

Pointing calibrator selection Once the source has been selected, a strong pointing calibration source near the target is chosen, because minimising antenna motion between pointing calibration and the actual observation better maintains the mirror shape¹.

Pointing calibration The antenna will be moved up and down in elevation, and clockwise and counter-clockwise in cross-elevation, around the expected position for the pointing calibrator. As the profile for the telescope beam conforms to a Gaussian distribution, the data can be fitted with a Gaussian, and the pointing error adjusted by comparison between the expected position of the calibrator, and the fitted maximum flux position. This correction will be applied to the coordinates where the source is expected to be.

Focus calibration The same calibrator can be used to calibrate the focus of the instrument, defined as the position of the secondary mirror that maximises the power collected by the instrument. Again, the profile for the focus, when the mirror is moved along its axis, is assumed to be Gaussian, and the fit for the maximum provides the focus.

On/Off source observation Both the source and a nearby position with no emissions have to be observed, in order to discriminate the contribution from the instrument. This is performed either by changing the position of the antenna (position switching), or by moving the secondary mirror in such a way that the main feed is focused on a different region of the sky, with no radio sources (wobbler switching). Another possibility is to compare the power of the emission from the same source at slightly different frequencies, assuming that the antenna and atmospheric noise does not change with this frequency switch (frequency switching). In the case of the DSS-63, on/off observations are performed either by position switching or frequency switching.

Atmospheric corrections The amount of energy received by the instrument depends strongly on weather conditions, and on the length of the path of the signal through the atmosphere. In particular, at cm wavelengths the amount of water vapour in the atmosphere is the major contributor

¹Antenna geometry changes mostly due to gravitational effects which are minimum for changes in azimuth, a much more significant for changes in elevation. Large radio astronomical antennas are designed following the homology principle, so that deformations produce changes in focus, so in order to collect the maximum flux focus calibrations should be performed with sources at the same elevation as the source to be observed.

to atmospheric opacity (a quantity that is proportional to the probability of a photon being absorbed after travelling a given length in the atmosphere). Measurements of opacity at different elevations (tipping curves, or skydips), which correspond to different air masses (a measure of the amount of atmospheric gas in the line of sight of the instrument), are used to fit a curve that provides the atmospheric opacity². This is usually done at DSS-63 once per observing session and frequency setup.

There are other corrections and calibrations to consider, but most of them can be obtained from typical values for the instrument and particular configuration, and do not contribute to illustrate the observing process with the DSS-63 antenna.

Of particular interest will be parameters such as system temperature (T_{sys}), main beam solid angle (Ω_{mb}), aperture efficiency (η_a), and the antenna temperature scale.

The data output of the correlator is a 384-sample autocorrelation function, which by means of the Fourier transform (Discrete Fourier Transform, in this case) provides a function proportional to the power spectrum of the source³. The post-processing of the observation, together with the calibration procedures, will allow us to determine the actual spectrum scale, and frequencies for the salient features of the spectrum.

Archive requirements

The archive for the DSS-63 antenna, then, is an archive for single-point spectroscopic observations. The particular requirements for the archive were:

Support for two single-point modes The DSS-63 antenna archive would hold, as per the initial specification, only single-point on-off spectra using frequency switching, or single-point continuum measurements.

VO spectral access services The main data products of the archive are spectra, which are supported by the Spectrum Data Model (incorporated in the RADAMS, as we have seen), and continuum measurements, which are considered to be one-point, wide-band spectra, similar to photometric data points in the visible band. A SSAP service will be implemented on top of the database, as well as a ConeSearch on the Scans table.

Web access interface The web access interface would use the same infrastructure needed for the SSAP service, but providing a VO compatible spectral web-service, instead of using web forms.

²See section 7.2 in *Tools of radio astronomy* [91] for details.

³See section 4.1.2 of *Tools of Radio Astronomy* [91] for the derivation.

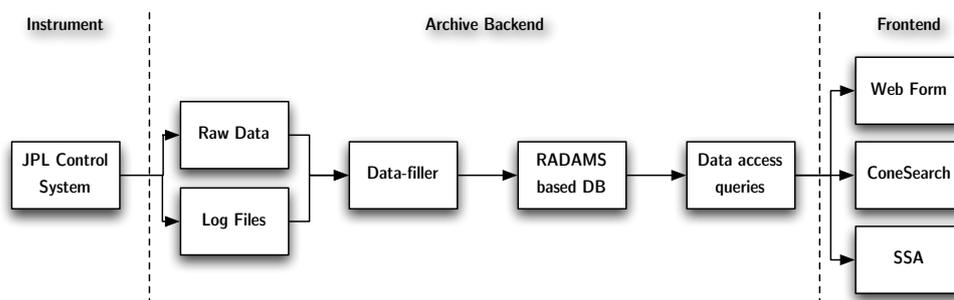


Figure 12.2: High level, layered architecture for the Robledo Archive. Dotted lines represent the logical separation between layers. Arrows represent data flow between sub-systems. Communications between layers are confined to the communications established between interfacing components.

No modification to the instrument control system All of the information stored in the archive should be available either from ingestion of the FITS files, or by harvesting the observation logs and control system output, but nothing should be added to the instrument control system. This is both a precautionary measure, so that we do not interfere with the telescope, but also works for making the project self-contained: tasks can be performed without need for external developers to modify another piece of software.

Simple data access policy The MSDCC data standard access policy is the straightforward NRAO policy: after 18 months, data are available to the public. However, as all data to be provided by these archive are older than that, there is no actual Policy module for this archive.

Archive architecture

With the requirements above, a layered architecture for the archive was devised where each layer correspond to a different subsystem:

Instrument With regards to the archive, the Instrument is represented by the Instrument Control System, which provides the links to observational data and configuration metadata.

Archive Backend The backend of the archive (not to be confused with any of the instrument's back-ends) is the module responsible for database and metadata access and maintenance. Includes the creation of entries in the database from the Raw data and observation log files, and the queries for supporting different query interfaces.

Frontend This is the actual accessible layer for human-computer or computer-computer interaction with the Archive Backend, using either a web form interface, or VO services such as ConeSearch for observation logs, and SSA for obtaining actual spectra.

Figure 12.2 shows that layered organisation, and how each layer maintains a single point of interaction with the next.

We have detailed several subsystems within the Archive Backend: the Data-filler, which waits for messages from the Instrument (in the form of entries on the observation log), ingesting them into the database, with links to the raw data storage; the database itself, which as we will see conforms to the RADAMS; and the archive queries to support the different use cases.

Apart from the automatic operation mode, the Data-filler can also be launched on its own, providing it with a set of FITS files to ingest, and the observing logs making reference to those FITS files.

For this archive, we have developed the complete Archive Backend, including the database implementation, which has been developed using the Django⁴ Python-based development framework. The database being used is Oracle, as per CAB prescriptions, and the user interface and VO data access modules built on top of the RADAMS will be developed by the SVO members of the LAEFF.

An interesting feature of the Archive Backend for the DSS-63 antenna, which has been also implemented for the IRAM 30m, is the way VO-related metadata (UCDs, UTypes, and other IVOA vocabularies) are provided to the Data access queries. We will describe that mechanism when discussing the implementation details of the IRAM 30m archive.

RADAMS implementation

Figure 12.3 shows the different database tables and their relationships⁵ used for the actual implementation of the DSS-63 archive.

If we compare that figure with the high-level overview of the RADAMS—figure 5.2—, we can see that in the archive implementation there are many more dependencies on the Observation or ObsData related tables in the archive database than those shown for the RADAMS.

That is so because we need to be able to perform different direct relationships with scans, as different search capabilities would be unfeasible if the whole dependency tree had to be traversed. However, in order to describe an observation (scan) or a set of associated scans, those additional dependencies are implicit, and need not be stated, as is the case with the high-level RADAMS data model.

⁴<http://www.djangoproject.com/>

⁵The complete .sql file implementing tables and relationships can be found at: http://www.iaa.es/~jdsant/thesis/dss63-sourceDM-v0_5.sql.

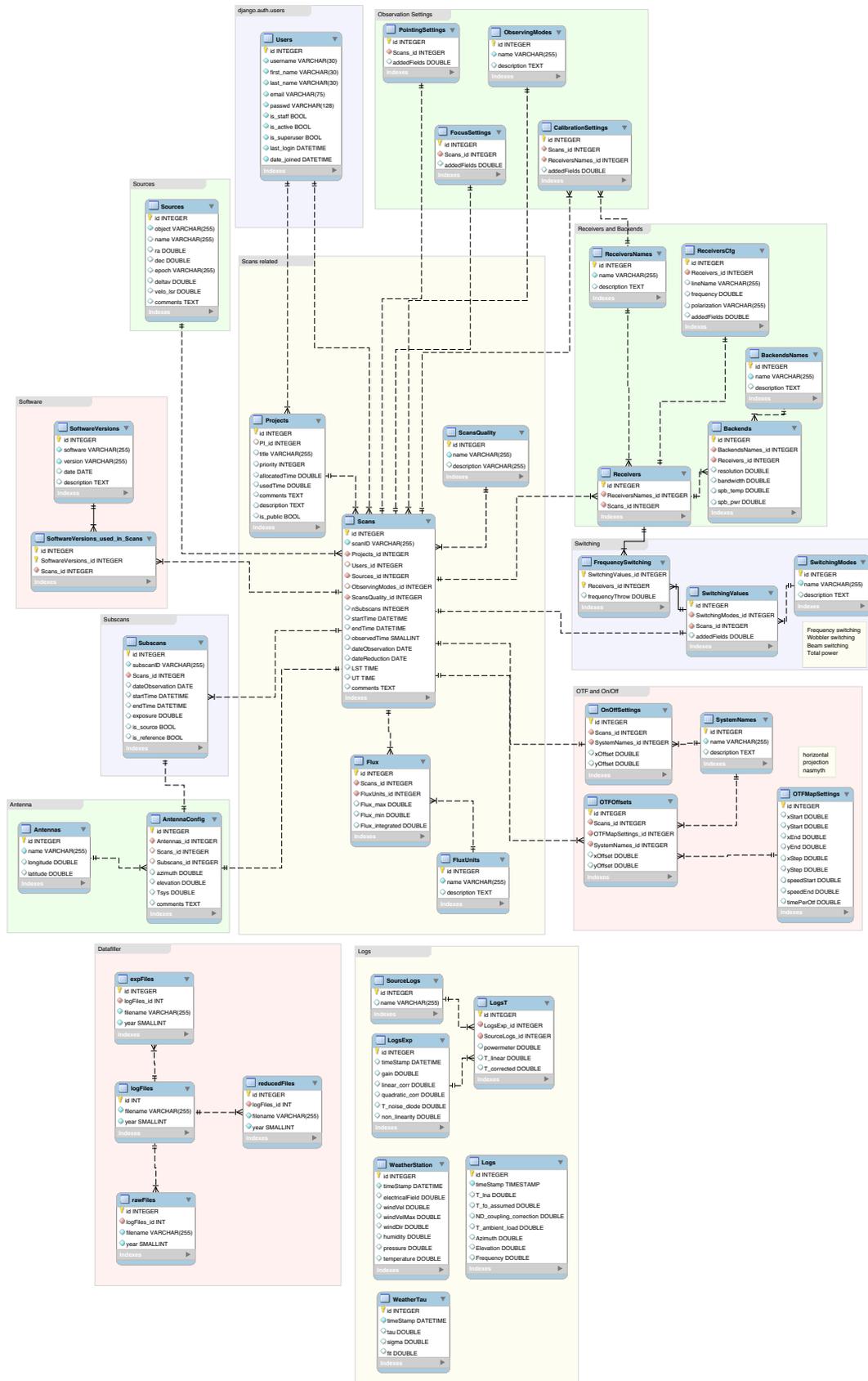


Figure 12.3: Implementation of the data model for the DSS-63 archive, generated from the actual SQL CREATE TABLE statements.

The Scan object is the cornerstone for all observations, as we discussed in section 6.1. From the Scan entity, all other relationships derive, except for those having to do with the Data-filler configuration—which are themselves outside of the scope of the RADAMS—and data which are related to Scans only by simultaneity, as is the case for weather station readings, whose only relation to Scans is their timestamp.

Sources implements a very simplified Target data model.

Project metadata is directly related to observations as part of the Curation data model, and to Users. Project and Users would be used by the Policy algorithm if DSS-63 archive’s policy were not fixed, as previously stated.

The Observations Settings, Receivers and Backends, Switching, Antenna and observation Settings are the tables supporting the Provenance.Instrument data model. Many of the tables are tables for instrument codes, or instrument setting codes.

The major mismatch between the RADAMS data model (developed for VO data query, and data description) and the actual data stored in the database (retrieved from the data available through the FITS file headers, and from the instrument control system), is found in the Characterisation data model: metadata such as spatial resolution, spectral resolution, et cetera, are derived from the values of the observation settings, and are not to be stored with the database, but will be, instead, generated on the fly by the VO interfaces. We will revisit this peculiarity when describing the IRAM 30m archive database and its relationship to the RADAMS.

The prototype web form interface to the archive can be seen in figure 12.4, where the Target data model (and CharDM Spatial axis) is queried by the Source name and position search box, the Project search box queries Observation metadata, Observation date is related to the CharDM in the Temporal axis, and Frequency/Velocity and Line names perform their searches on the CharDM Spectral and Velocity axes.

12.2 The IRAM 30m archive

The IRAM 30m radio telescope, located at the Loma de DÍlar, in the shoulders of the Pico Veleta in Sierra Nevada, Granada, is the leading millimetre-range radio telescope, due to its sensitivity and instrument capabilities. One objective measure of its importance is that it has generated more than 1100 papers since 1982⁶, when it started operations.

⁶Source: List of publications till 2008 making use of the IRAM 30m compiled by former IRAM-Spain director Rainer Mauersberger till 2008, and published through the SAO/NASA Astrophysics Data System:

http://adsabs.harvard.edu/cgi-bin/nph-abs_connect?library&libname=PV_Publ.&libid=45af8f27d3

Robledo DSS63 Scientific Archive

Home [Search](#) Results News Policy Help About Admin

Search

Source Name / Position

Source Name

Target Name

J2000 RA hh:mm:ss.ss

J2000 Dec dd:mm:ss.ss

Size degrees

Weather requirements

Opacity

Project

Project ID

Observation Date

From to

dd/mm/YYYY

Frequency / Velocity / Line Name

Frequency Range - GHz

Velocity Range - km/s

Line Name

Line Name Match

Figure 12.4: Prototype search form for the DSS-63 archive.

The IRAM 30m —shown on figure 12.5 next to the residence and control building— hosts several instruments, both coherent (heterodyne) and incoherent (bolometers), with different data reduction packages and techniques. There are also single-pixel and multi-pixel detectors, what makes the data handling even more particular.

As the data from the detectors can be fed to several analysis systems, the former are called front-ends, while the latter are called back-ends. Keeping the different frontend-backend combinations is one of the complications of data storage for the IRAM 30m.

And apart from the different frontend-backend combinations, one of the most complex parts of data handling for astronomical observatories is the handling of observing modes, compounded in radio astronomy with the combination of switching modes. They have been compiled, and briefly explained, in table 12.3. The definitive guide to the different observing modes, switching modes, front-ends, backends, and observing setup, is the guide to



Figure 12.5: IRAM 30-meter antenna, next to the residential and control building, near Pico Veleta (visible on the right side of the picture), at Sierra Nevada, Granada. Picture by the author.

the IRAM New Control System (NCS) user interface⁷ [92].

Archive architecture

The architecture of the IRAM 30m archive is very similar to that of the DSS-63 archive, and follows the same principles of independence from the control system operation (in this case, the IRAM NCS), and of layering of dependencies/responsibilities. Figure 12.6 shows that architecture, and can be easily compared with figure 12.2

The differences in the workflow and working architecture between the DSS-63 and IRAM 30m are essentially related to the difference in the control system: whereas the JPL control system used by all DSS antennas provides FITS files in a format common to all DSCCs, and other JPL sites, together with text logs in the same format, the IRAM 30m uses since 2005 the New Control System [92, 93, 94, 95, 96, 97], a Python-based real time telescope control system which writes each scan in a different IRAM Multi-Beam FITS [24] file, while at the same each observation is described in an XML file [94].

⁷<http://www.iram.es/IRAMES/documents/ncs30mPako/Current/PDF/pako.pdf>

Table 12.3: Available combinations of observing and switching modes at the IRAM 30m telescope. See notes and discussion on the text.

Observing mode	swTotal ^a	swBeam ^b	swWobbler ^c	swFreq ^d
Calibrate (Heterodyne) ^e	X			
Pointing ^f	X	X	X	
Focus ^g	X	X	X	
Tip (Skydip) ^h	X			
Track ⁱ				X
On-Off ^j	psw ^k		X	
OTF Map (Heterodyne) ^l	X			X
OTF Map (MAMBO Bolometer) ^m			X	
VLBI ⁿ	X			

^aIn total power observations there is, in fact, no switching. See exception at psw switching.

^bIn beam switching, the optical path is cut several times per observing cycle by the periodic interposition of a rotating blade (chopper).

^cIn wobbler switching observations, the secondary mirror *wobbles*, changing inclination slightly, effectively switching the beam to a different sky position.

^dIn frequency switching observations, the same position in the sky is observed at different frequencies.

^eCalibrate observations are performed for heterodyne receivers in order to be able to convert from voltages/counts to fluxes.

^fPointing observations are done to optimise the positioning of the telescope in Azimuth and Elevation. This is done by continuum observations of a cross scan in azimuth and elevation on a point source near the intended target source.

^gFocus observations are done to minimise the spread of received energy, maximising the collected energy.

^hTip, antenna tipping, or *skydip* observations are performed in order to estimate the actual opacity of the atmosphere at different air masses (amount of atmosphere in the line of sight).

ⁱIn Track or Tracking observations the position of the antenna does not change in celestial coordinates, and the off-source reference is taken by switching receiver frequency.

^jObservations made by comparison on the signal from a source and from a zero emission reference.

^kpsw: Position switching, where the antenna drifts from the on to the off position. For total power observations, the complete power pattern during the drift is also recorded.

^lOn-The-Fly observations record antenna signal across a drift in space along pre-determined paths. For heterodyne receivers, it creates data cubes.

^mOn-The-Fly observations with bolometers create intensity maps by recording intensities along pre-determined paths.

ⁿVLBI (Very Long Baseline Interferometry) observations are not stored by the IRAM.

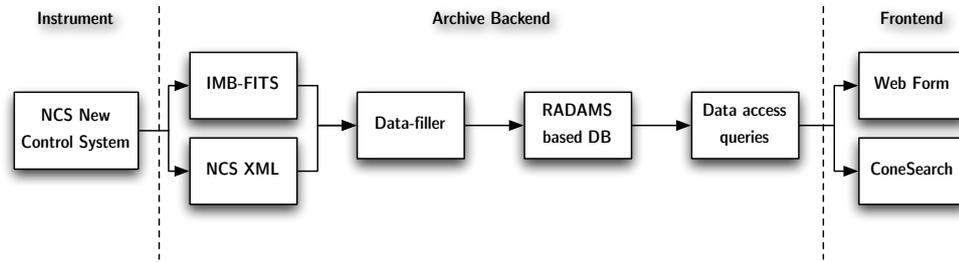


Figure 12.6: High level, layered architecture of the archive for the IRAM 30m antenna. Dotted lines represent the logical separation between layers. Arrows represent data flow between sub-systems. Communications between layers are confined to the communications established between interfacing components.

RADAMS implementation

Given that the high-level architecture is largely the same for both archives, the differences between them have to deal with the different source data format (JPL FITS versus IMB-FITS), observation logs (JPL text logs versus NCS XML files), and different database for supporting the additional observing modes, switching modes, and instruments available at the IRAM 30m.

This additional complexity could be easily imagined by comparing table 12.3 to the observation description for DSS-63 made in subsection 12.1: the observing modes possible with DSS-63 are just those available for heterodyne receivers, and wobbler-switching is not available.

Figure 12.7 shows the database tables and relationships for the IRAM 30m database, named TAPAS (Telescope Archive for Public Access System). If we compare it with the DSS-63 database in figure 12.3, we can see there are many more observing setting tables (each one holding different data for each different observing mode), and many more Project and Policy tables, in order to connect the archived data with the existing Pool database⁸.

Given the higher complexity of the TAPAS database, we will perform a detailed comparison of RADAMS entities with the archive tables.

Observation Observation is the root class for the data model, and serves to bind together all related information data and metadata. As such, it can be thought of as being embodied by the Scans table, with the caveats mentioned before about the extra number of dependencies.

⁸The Pool database is a parallel system used to record project data for observing projects to be managed under *pool* mode, that is, not having a fixed observing block, but instead allocating observing blocks following priority criteria for observable sources at any given time, using backup projects with less stringent weather conditions when high-priority projects cannot be observed.

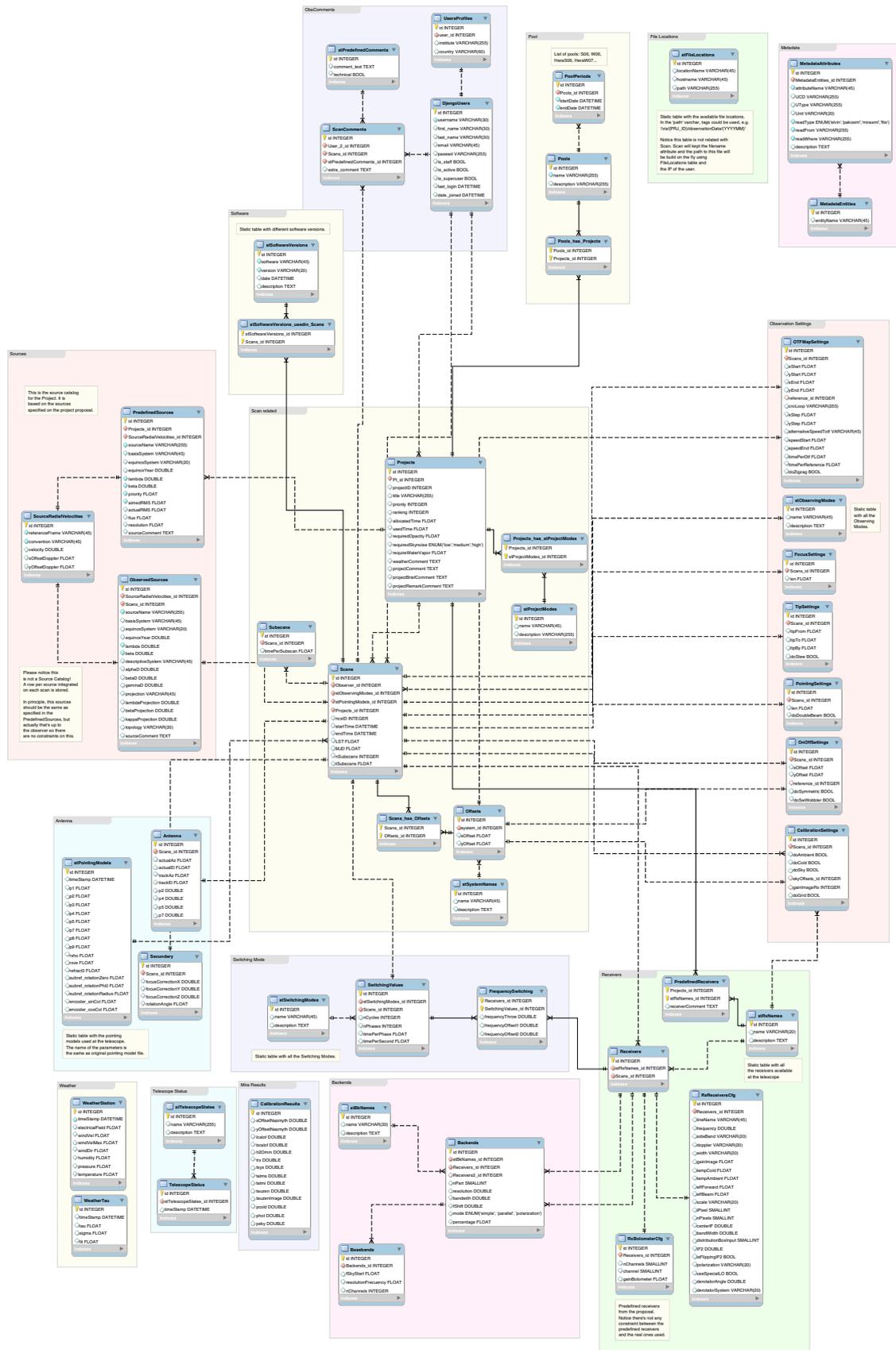


Figure 12.7: Implementation of the data model for the IRAM 30 archive, TAPAS (Telescope Archive for Public Access System). Generated by reverse engineering of the MySQL database.

ObsData ObsData is a proxy to the actual observation data. In the case of TAPAS, these are raw and reduced IMB-FITS files. A separate table, `stFileLocations`, is used to contain the prefix path to where data will actually be located (accessible from the TAPAS system). As TAPAS file names are systematic⁹, they are built from Scan metadata and the `stFileLocations` table on the fly.

Target Describes the target of the observation, providing as much information as available for already known targets, as discussed in section 6.2.

This part of RADAMS is supported in TAPAS by the tables `ObservedSources` and `PredefinedSources` tables. `PredefinedSources` contains the sources which have been proposed for an observing project, and are linked to it¹⁰, while `ObservedSources` contains the sources which have actually been observed, with the target-related observation settings required. In the case of `ObservedSources`, coordinates are given in a generic spherical system which is converted into equatorial or other systems following the `basisSystem` and `projection` settings.

Both tables are joined to a `SourceRadialVelocities` tables where recession velocities for well-known sources are stored, in order to properly set Velocity measurements.

Characterisation This is the core of the RADAMS and generic observation models, and corresponds fully to the Characterisation data model Recommendation [67], as we have discussed in section 6.1.

We will describe separately the origin of Characterisation data for each of the axes: Spatial, Temporal, Spectral, and Observable.

Spatial For most single-dish observations, `Location`, `Bounds` and `Support` are the same, and will be taken from the relationship between the `Scans` table and the `PredefinedSources` and `ObservedSources` tables. For OTF mapping, `Location` is taken from the `PredefinedSources` table, while `Bounds` will be calculated from projections of the `xStart`, `yStart`, `xEnd` and `yEnd` derived from the `ObservedSources` metadata, while if `Support` is to be specified it has to be calculated from the `xStep` and `yStep` attributes. For bolometer mapping, the `Bounds` and `Support` are the same¹¹, but they should

⁹The file name is of the form `iram30-backend-yyyyymmddsn-imb.fits`, where `backend` represents the backend name (i.e., `wilma`, `4mhz`, `vespa`...); `yyyyymmdd` represents a date in year-month-day format; and `nn` is the scan number, separated from the date by an `s`. An example filename: `iram30m-vespa-20090211s62-imb.fits`

¹⁰There is a CALIB observation project in which usual flux, focus and pointing calibration sources are included.

¹¹Support could be defined as a 2D function on the RMS achieved across the map.

be computed by the reduction software, or approximated by a convolution of observed coordinates with beam widths.

The coordinate computations for the observations have also to take into account the Offsets table, as those offsets can be added by the observer (following observing project instructions) to use the same observing scripts to scan an offset part of the sky. See section 6.1 of the *paKo user's guide* [92] for a discussion of NCS-supported coordinate systems.

Regarding spatial Resolution, this is a function of Antenna, Secondary, Observing mode, Receivers and Backends, and is looked up from engineering databases.

For spatial accuracy, all data for pointing corrections is collected for all scans, and statistically studied by the engineering team. The Antenna table contains the intended and actual elevation and azimuth of the antenna, which allows for taking into account the average tracking system error, while parameters p1 to p9 of the IRAM pointing model [98] are also stored, and can be used to optimise the pointing model. In particular, the observer adds the pointing corrections to the Antenna p2 —azimuth correction— and p7 —elevation correction— attributes.

Temporal Location in the Temporal axis will be defined as the Scan start`Time` attribute, with Bounds taken from start`Time` and map-keyend`Time`. In order to provide Support the dead times between Scans should be calculated, but are not provided.

Spectral The RADAMS defines the Location in the Spectral axis as the frequency for the central sample of the spectrum, which corresponds to the frequency attribute of the ReceiversC`fg` table. Bounds are defined as that frequency, plus/minus half the bandwidth attribute. Most of the metadata for the root AxisFrame.Spectral class is also recovered from the ReceiversC`fg` table. The Sensitivity will be compiled from the engineering measurements for the receiver, and will not be stored in TAPAS, while Resolution is compiled from the resolution attribute in the Backends table.

SamplingPrecision is also a function of the Backend, and will be stored and looked up from a static table.

Observable For flux observations, such as On/Off observations, Location is taken from the flux attribute of ObservedSource, while bounds is taken to be the interval at that value plus/minus the actualRMS value. We do not provide Support for the Observable axis, while flux resolution is calculated as flux divided by actualRMS.

As for flux accuracy, the values should be computed from statistics on historical data once the TAPAS database is in operation for at least a whole semester.

Provenance We will study the implementation of Provenance separately for each sub-model:

Provenance.Instrument This part of Provenance was discussed in section 9.1. It was defined as a hierarchical tree aggregating different subsystem configurations: InstrumentConf could hold several AntennaConf entries (for describing antenna arrays), one or several Feeds per AntennaConf, and one BeanConf per Feed stating Beam properties, Receiver properties, Spectrum properties and Velocity properties.

In TAPAS, the specific Instrument, Location and InstrumentConf classes are pre-computed and stored outside of the TAPAS database. The Antenna description part of the RADAMS needed to be updated for the IRAM 30m, and include the different switching modes, the configuration of the Secondary mirror (needed for wobblers switching modes). Beam metadata needs to be looked up in engineering configuration tables, and Receiver, Spectrum, and Velocity settings are obtained from the Receivers and Backends tables.

Provenance.AmbientConditions This part of the RADAMS is directly linked to the WeatherStation and WeatherTau tables, using timestamps to correlate scans with measurements.

In addition, the OpacityCurve is directly obtained from the results of observations of Tip kind¹². The CalibrationResults table holds that information for heterodyne receivers, reduced with the MIRA package.

Provenance.Processing Some parts of Provenance.Processing are implemented on the TAPAS database, in particular the Software package-related metadata. The calibration part is still being reviewed, as different packages (MIRA, MOPSIC) provide very different processing and calibration information.

Curation This part of the RADAMS is described in section 7.1. The common curation data for TAPAS (the Curation table) is held outside of this database, and will be used for creating the ConeSearch entry in VO Registries. Project and DataID are obtained from the Scans and Project tables.

¹²Scans with stObservingModes_id attribute corresponding to Tip or Bolotip observations.

There are plans to link the existing Proposal handling database to the TAPAS system, and in that case the remaining Curation data model will be implemented.

Policy As mentioned in section 7.2, the Policy determination algorithm needs to get access to Users, Project, and observation related information such as Observers, Operators, et cetera.

In TAPAS, Project information is stored in the Project table, with a link to the Principal Investigator, while Scan holds the Observer identifier, and Operators and Observatory staff have entries in the DjangoUsers table with the `isStaff` set to true.

The main difference with the proposed RADAMS architecture is that Operators are not identified with observations, and that there is no possibility to specify Co-Investigators.

Packaging The TAPAS archive has all the infrastructure needed for being able to provide data in the form of VO services. However, due to the actual IRAM Policy statement, only header information will be publicly available for observations, after a 12 years proprietary period, while data themselves will only be available for a selection of projects, and only after 18 months proprietary period. The actual implementation of the Packaging class has not been, therefore, adapted to the TAPAS archive.

The complete SQL file implementing the IRAM 30m archive data base can be downloaded from the following link:

http://www.iaa.es/~jdsant/thesis/iram30m-sourceDM-v0_8.sql

VO metadata attributes

We had mentioned that the Archive Backend not only stored and provided data to the Frontend layer, but that it also had a mechanism for providing the relevant VO metadata to any query mechanism.

This is achieved by means of the `MetadataAttributes` and `MetadataAttributes` tables. The `MetadataAttributes` table contains an entry for each of the different tables in the archive, to be identified by their name and a code, while the `MetadataAttributes` contains a row for every attribute of every table.

Among the fields in the `MetadataAttributes` table we provide UCDs, UTypes, and XPath or SQL expressions identifying how to retrieve a particular piece of information for each attribute.

In this way, the Data-filler can create database entries from the corresponding data sources, but also provides the corresponding UCDs and UTypes for exporting a particular attribute, and also includes documentation, help,

and even links to IVOA vocabularies [62], such as the IAU sanctioned thesaurus [99], or the IVOA thesaurus¹³, to help in the automatic discovery of interesting data by means of semantic tagging.

The Data-filler MetadataAttributes and MetadataAttributes tables content creation is bootstrapped by importing a JSON¹⁴ file which encodes all the information to be stored in said tables.

The original concept of having a metadata-oriented Data-filler, based on a mapping of attributes and entities was developed and implemented by Victor Espigares, and the inclusion of the ontologyLink attribute in order to incorporate arbitrary vocabularies was suggested by Juan de Dios Santander Vela. UCDs and UTypes were assigned following the RADAMS.

TAPAS interface

TAPAS provides two query interfaces: one web based query form¹⁵, and a VO-compatible ConeSearch service.

The TAPAS home screen—see figure 12.8—has a login feature, that either asks for login and password, or provides last login information. From this home screen users have access to the Search, previous search Results, News (updates to the archive, new datasets added, et cetera), Policy statement, Help, and a statement on the development of the archive.

The search form—figure 12.9—allows users to perform searches on the Target, Provenance.Instrument, Provenance.Environment, Characterisation.SpatialAxis, Characterisation.TemporalAxis, and Characterisation.SpectralAxis data models.

For specifying Targets, users can use either widely known names, such as those registered by NED for astronomical objects, or IRAM project object codes. If no suitable name is known, users can provide equatorial coordinates¹⁶ in sexagesimal format, and a cone angular size in decimal degrees. Object names, when not found in the projects, are resolved to coordinates using CDS' Sesame service.

The only weather requirement which can be imposed on searches is a maximum Opacity, as it can identify those datasets which have had a good enough sky for the kind of observation we intend to find.

Instrumental requirements include the specification of the instrument front-ends, and SpectralAxis requirements are specified either in the form of frequency ranges, of velocity ranges, or the name of the molecular or atomic line being observed (i.e. CO(1-0), HCN, et cetera).

In addition, TAPAS can be queried by project IDs, specially by the PI of the observation, and a Batch mode exists by which users can upload a

¹³<http://www.ivoa.net/rdf/Vocabularies/vocabularies-20081104/IVOAT/>

¹⁴JavaScript Object Notation, <http://www.json.org/>

¹⁵Presently available at: <https://mrt-lx3.iram.es/tapas/>

¹⁶Right Ascension and Declination in the J2000 equinox

The screenshot shows the home page of the TAPAS website. At the top, there is a blue header bar with the text 'TAPAS - Telescope Archive for Public Access System IRAM 30m Archive' on the left and 'Welcome Stephane' with 'Last logged: Sun, 29 Mar 2009' and a 'Logout' link on the right. Below the header is a light gray navigation bar with links for 'Home', 'Search', 'Results', 'News', 'Policy', 'Help', 'About', and 'Admin'. The main content area has a 'Home' section with a welcome message and a list of links: 'Search', 'News', 'Policy', 'Help', and 'About'. There is also an acknowledgment section with a small text block. On the right side of the main content area, there is a photograph of the IRAM 30m telescope. At the bottom right of the page, there is a footer with the text 'IRAM - IAA - CSIC'.

Figure 12.8: Home screen of TAPAS, with the main functions available from the menu.

pecially formatted file which contains a list of names and/or position pair coordinates. The names will be resolved by either NED or Simbad. If the internal IRAM name is to be provided, it should be enclosed by two asterisks (*; e.g. *M83A*). The positions are in the format $hh:mm:ss.ss \pm dd:mm:ss.ss$ (e.g. $12:12:12.12 +30:30:30.30$). Note that right ascension coordinates must be given in hours, but declination in sexagesimal degrees.

After searching for observations, the results are provided in HTML form, but also Comma Separated Values (CSV) ASCII files, and VOTables can be generated. Even a PDF of the results page can be downloaded —see figure 12.10—.

The available header information are the internal (IRAM) source name, object equatorial coordinates (J2000), project identifier (ID), receiver, sky opacity, and time-stamps for the start of the first and the latest scans for the observation.

When the user clicks on the Project link, a new page appears with project details —see figure 12.11—, including the time spanned by all project's scans, how many scans belong to the project, and how many have been performed using the MAMBO bolometer, or the heterodyne receivers.

For all project scans, a graph is provided where the taumeter reading (from the WeatherTau table) is plotted for each scan, giving users a view of the evolution of opacity throughout the different project's scans.

If users click on the number of scans for an observation on the Results

Figure 12.9: Search form of TAPAS. Queries can be performed on any of the blocks presented to the user.

page, a detailed view of all scans for a given observation is provided—see figure 12.12—, together with the common parameters for all scans: project ID, IRAM name of the source, velocity setting, equatorial coordinates, equinox, and receiver. This list can be also retrieved as HTML, CSV, or VOTable.

If a particular scan is clicked, the information for that scan is shown—see figure 12.13—, and obtains a link to the data (if available as per IRAM policy), Characterisation.TemporalAxis information (observation length, start, and stop), Target information (source name, coordinates), Provenance.Environment (weather conditions), Provenance.Instrument (antenna azimuth and elevation, pointing and focus corrections, observing mode, calibration settings, offsets, front-ends and back-ends), Provenance.Processing (software version, and calibration settings), and Characterisation.SpectralAxis (spectral line).

For different observing modes, the scan information provided is different. Compare figure 12.13 with figure 12.14. The data models are the same, but the information provided is different, corresponding to the differences in observation setup (Provenance.Instrument), and spatial and temporal coverage in the Characterisation.

Finally, figure 12.15 shows a listing of all sources observed for a given project, accessible from the Project details page.

Apart from the searching capabilities, the TAPAS web site provides links to help on the TAPAS user interface, and includes a link to the IRAM policy statement, as can be seen in figure 12.16.

TAPAS - Telescope Archive for Public Access System

IRAM 30m Archive

Welcome Stéphane
Last logged: Sun, 29 Mar 2009
Logout

Home Search Results News Policy Help About Admin

Sources List

Format HTML Return 50 rows

Total number of sources: 85 >>>

Source	Velocity	# Scans	J2000 RA	J2000 Dec.	Project	Receiver	Opacity	First Scan	Last Scan
0106+013	0.00	3	01:08:38	01:35:00	T02-06	A230	0.07	2009-02-20 13:40:29	2009-02-20 15:39:08
0106+013	0.00	3	01:08:38	01:35:00	T02-06	B230	0.07	2009-02-20 13:40:29	2009-02-20 15:39:08
0316+413	0.00	4	03:19:48	41:30:42	d13-08	A230	0.02	2009-02-10 22:59:45	2009-02-10 23:25:39
0316+413	0.00	4	03:19:48	41:30:42	d13-08	B230	0.02	2009-02-10 22:59:45	2009-02-10 23:25:39
0727-115	0.00	1	07:30:19	-11:41:12	monIt_oh231	A230	0.00	2009-03-09 22:30:22	2009-03-09 22:32:34
0851+202	0.00	3	08:54:48	20:06:30	125-08	A230	0.17	2009-02-27 02:32:44	2009-03-02 01:30:45
0923+392	0.00	1	09:27:03	39:02:20	d13-08	B230	0.05	2009-02-13 04:56:30	2009-02-13 04:58:45
0923+392	0.00	1	09:27:03	39:02:20	d13-08	A230	0.05	2009-02-13 04:56:30	2009-02-13 04:58:45
0953+254	0.00	1	09:56:49	25:15:15	d13-08	B230	0.07	2009-02-13 04:51:52	2009-02-13 04:54:06
0953+254	0.00	1	09:56:49	25:15:15	d13-08	A230	0.07	2009-02-13 04:51:52	2009-02-13 04:54:06
1039+811	0.00	7	10:44:23	80:54:39	d13-08	B230	0.05	2009-02-10 23:31:00	2009-02-16 04:29:34
1039+811	0.00	7	10:44:23	80:54:39	d13-08	A230	0.05	2009-02-10 23:31:00	2009-02-16 04:29:34
1226+023	0.00	3	12:29:06	02:03:08	d13-08	A230	0.05	2009-02-12 03:58:45	2009-02-12 04:41:31
1226+023	0.00	3	12:29:06	02:03:08	d13-08	B230	0.05	2009-02-12 03:58:45	2009-02-12 04:41:31
17455-2800	-15.00	21	17:48:42	-28:02:06	154-08	HERA1 Pixel 1	0.06	2009-02-23 06:54:37	2009-02-23 08:05:31
1757-240	0.00	9	18:00:30	-24:04:01	154-08	HERA1 Pixel 1	0.08	2009-02-21 05:55:33	2009-02-23 06:56:50
1757-240	0.00	1	18:00:30	-24:04:01	d16-08	HERA1 Pixel 1	0.05	2009-02-23 06:34:15	2009-02-23 06:49:56
1803+784	0.00	1	18:00:45	78:28:04	d13-08	B230	0.06	2009-02-12 04:51:39	2009-02-12 04:53:53
1803+784	0.00	1	18:00:45	78:28:04	d13-08	A230	0.06	2009-02-12 04:51:39	2009-02-12 04:53:53
18060-2005	13.00	58	18:08:56	-20:05:11	154-08	HERA1 Pixel 1	0.08	2009-02-21 06:01:07	2009-02-21 10:15:32
BODY Lutin	0.00	84			125-08	A230	0.20	2009-02-27 00:19:51	2009-03-02 03:02:47
core1	0.00	3	02:01:05	87:41:57	d13-08	B230	0.05	2009-02-12 04:51:39	2009-02-12 05:01:35
core1	0.00	3	02:01:05	87:41:57	d13-08	A230	0.05	2009-02-12 04:51:39	2009-02-12 05:01:35
G34.3+0.2	0.00	1	18:53:18	01:14:58	030-08	HERA1 Pixel 1	0.09	2009-02-21 05:26:35	2009-02-21 05:46:11
G34.3+0.2	0.00	6	18:53:18	01:14:58	154-08	HERA1 Pixel 1	0.09	2009-02-21 05:26:35	2009-02-22 04:56:31
HIP48541	0.00	22	09:53:59	27:41:43	d13-08	A230	0.06	2009-02-13 04:56:30	2009-02-15 05:00:01
HIP48541	0.00	22	09:53:59	27:41:43	d13-08	B230	0.06	2009-02-13 04:56:30	2009-02-15 05:00:01
HIP60074	0.00	9	12:19:06	16:32:53	d13-08	A230	0.05	2009-02-13 04:17:28	2009-02-13 04:49:23
HIP60074	0.00	9	12:19:06	16:32:53	d13-08	B230	0.05	2009-02-13 04:17:28	2009-02-13 04:49:23
IRC+10216	-27.00	4	09:47:57	13:16:42	125-08	A230	0.23	2009-02-27 00:42:57	2009-03-02 02:14:31
J1229+021	241.00	1	12:29:06	02:03:08	178-08	HERA2 Pixel 1	0.08	2009-02-21 03:06:35	2009-02-21 03:13:31
J1229+021	241.00	1	12:29:06	02:03:08	178-08	HERA1 Pixel 1	0.08	2009-02-21 03:06:35	2009-02-21 03:13:31
J1642+689	241.00	1	16:42:07	68:56:39	154-08	HERA2 Pixel 1	0.06	2009-02-23 08:02:14	2009-02-23 08:15:12
J1642+689	241.00	17	16:42:07	68:56:39	215-08	HERA1 Pixel 1	0.07	2009-02-21 03:35:46	2009-02-24 02:10:01
J1642+689	241.00	1	16:42:07	68:56:39	154-08	HERA1 Pixel 1	0.06	2009-02-23 08:02:14	2009-02-23 08:15:12
J1642+689	241.00	17	16:42:07	68:56:39	215-08	HERA2 Pixel 1	0.07	2009-02-21 03:35:46	2009-02-24 02:10:01
I1157	0.00	3	20:39:06	68:02:15	d13-08	A230	0.05	2009-02-11 02:11:33	2009-02-11 02:22:56
I1157	0.00	3	20:39:06	68:02:15	d13-08	B230	0.05	2009-02-11 02:11:33	2009-02-11 02:22:56
M101_1	241.00	79	14:03:17	54:24:12	215-08	HERA2 Pixel 1	0.07	2009-02-21 03:06:35	2009-02-24 02:02:19
M101_1	241.00	79	14:03:17	54:24:12	215-08	HERA1 Pixel 1	0.07	2009-02-21 03:06:35	2009-02-24 02:02:19
M101_3	241.00	29	14:03:17	54:24:21	215-08	HERA1 Pixel 1	0.07	2009-02-23 09:30:36	2009-02-23 10:51:16
M101_3	241.00	29	14:03:17	54:24:21	215-08	HERA2 Pixel 1	0.07	2009-02-23 09:30:36	2009-02-23 10:51:16
NGC7538	-57.40	1	23:13:45	61:28:10	030-08	HERA1 Pixel 1	0.07	2009-02-23 04:54:59	2009-02-23 05:05:42
NGC7538	241.00	3	23:13:45	61:28:10	215-08	HERA1 Pixel 1	0.06	2009-02-23 08:40:23	2009-02-23 08:50:28
NGC7538	-57.40	1	23:13:45	61:28:10	030-08	HERA2 Pixel 1	0.07	2009-02-23 04:54:59	2009-02-23 05:05:42
NGC7538	-57.40	10	23:13:45	61:28:10	d16-08	HERA1 Pixel 1	0.06	2009-02-22 08:46:30	2009-02-23 05:37:15
NGC7538	241.00	3	23:13:45	61:28:10	215-08	HERA2 Pixel 1	0.06	2009-02-23 08:40:23	2009-02-23 08:50:28
NGC7538	-57.40	10	23:13:45	61:28:10	d16-08	HERA2 Pixel 1	0.06	2009-02-22 08:46:30	2009-02-23 05:37:15
NGC7538IRS1	-57.40	32	23:13:45	61:28:12	d16-08	HERA1 Pixel 1	0.07	2009-02-22 09:09:58	2009-02-23 06:40:57
NGC7538IRS1	-57.40	32	23:13:45	61:28:12	d16-08	HERA2 Pixel 1	0.07	2009-02-22 09:09:58	2009-02-23 06:40:57

Edit search

Observation Date 01/01/2009 - 29/03/2009
 Line Name CO(2-1) Match Generic

IRAM - IAA - CSIC

Figure 12.10: TAPAS search results. Apart from the result list, the actual query is shown at the bottom of the results page.

TAPAS - Telescope Archive for Public Access System
IRAM 30m Archive
Welcome Stéphane
Last logged: Sun, 29 Mar 2009
Logout

[Home](#) [Search](#) [Results](#) [News](#) [Policy](#) [Help](#) [About](#) [Admin](#)

Project Info

ID d13-08
Observation Dates: 10/02/2009 - 16/02/2009

All Scans: 828
Scans with heterodyne receivers: 828
Scans with bolometer: 0
Observed sources with different receivers: 52

Opacity Curve
Time evolution of the opacity at 225 GHz for the project scans

Observation Date 01/01/2009 - 29/03/2009
Line Name CO(2-1) #Match Generic

IRAM - IAA - CSIC

Figure 12.11: TAPAS provides detailed information on projects, including access to observations in the project.

This is the official TAPAS policy statement:

TAPAS contains header information of all astronomical data taken [with the IRAM 30m antenna] after 01/01/2009. TAPAS does not contain any astronomical heterodyne or bolometer data. However, TAPAS provides the possibility to link the header data of individual scans to FITS files containing the uncalibrated astronomical data. TAPAS provides a Virtual Observatory facility for query and retrieval of header data.

Access rights to any astronomical data in TAPAS depend on the project. The PI and IRAM decide when to make astronomical data public. For all projects, after a proprietary period of one year, a subset of header data will be made accessible via web interfaces and search tools. IRAM staff have immediate and unrestricted access to all header data. For large programs,

TAPAS - Telescope Archive for Public Access System

IRAM 30m Archive

Welcome Stephane
Last logged: Sun, 29 Mar 2009
Logout

[Home](#) [Search](#) [Results](#) [News](#) [Policy](#) [Help](#) [About](#) [Admin](#)

Scans

Format HTML Return 50 rows

PROJECT: 215-08

SOURCE: M101_1
VEL: 241.00 km/s
RA: 14:03:17
DEC: 54:24:12
J2000

RX: HERA1 Pixel 1

Total number of scans: 79 >>>

Scan ID	# Sub.	Start	End	Type	Opacity	Frequency	Line	Azimuth	Elevation	pX	pY	fZ
2009-02-21.86	3	2009-02-21 03:06:35	2009-02-21 03:29:21	calibrate	0.12	230.538	12CO(2-1)	195.10	53.91	-5.80	-26.60	-2.52
2009-02-21.87	3	2009-02-21 03:06:35	2009-02-21 03:32:40	calibrate	0.10	230.538	12CO(2-1)	378.57	71.16	-5.80	-26.60	-2.52
2009-02-22.14	3	2009-02-22 01:12:36	2009-02-22 01:20:55	calibrate	0.06	230.538	12CO(2-1)	385.73	36.95	-4.90	-24.40	-2.36
2009-02-22.15	3	2009-02-22 01:12:36	2009-02-22 01:28:15	calibrate	0.07	230.538	12CO(2-1)	404.32	56.76	-4.90	-24.40	-2.36
2009-02-22.19	3	2009-02-22 01:29:26	2009-02-22 01:41:56	calibrate	0.06	230.538	12CO(2-1)	402.80	59.45	-4.90	-24.40	-2.36
2009-02-22.20	3	2009-02-22 01:29:26	2009-02-22 01:42:55	calibrate	0.06	230.538	12CO(2-1)	402.70	59.59	-4.90	-24.40	-2.36
2009-02-22.21	2	2009-02-22 01:43:13	2009-02-22 01:44:22	onTheFlyMap	0.09	230.538	12CO(2-1)	402.64	59.68	-4.90	-24.40	-2.36
2009-02-22.23	3	2009-02-22 01:44:39	2009-02-22 01:50:24	calibrate	0.08	230.538	12CO(2-1)	402.38	59.80	-4.90	-24.40	-2.36
2009-02-22.24	3	2009-02-22 01:44:39	2009-02-22 01:51:57	calibrate	0.08	230.538	12CO(2-1)	401.75	60.82	-4.90	-24.40	-2.36
2009-02-22.25	2	2009-02-22 01:52:16	2009-02-22 01:53:24	onTheFlyMap	0.09	230.538	12CO(2-1)	401.68	60.90	-4.90	-24.40	-2.36
2009-02-22.26	2	2009-02-22 01:53:42	2009-02-22 01:55:20	onTheFlyMap	0.09	230.538	12CO(2-1)	401.83	60.95	-4.90	-24.40	-2.36
2009-02-22.27	2	2009-02-22 01:55:39	2009-02-22 01:57:31	onTheFlyMap	0.08	230.538	12CO(2-1)	401.65	61.21	-4.90	-24.40	-2.36
2009-02-22.28	2	2009-02-22 01:57:50	2009-02-22 02:00:07	onTheFlyMap	0.09	230.538	12CO(2-1)	401.39	61.56	-4.90	-24.40	-2.36
2009-02-22.29	2	2009-02-22 02:00:28	2009-02-22 02:02:53	onTheFlyMap	0.12	230.538	12CO(2-1)	400.87	62.00	-4.90	-24.40	-2.36
2009-02-22.30	2	2009-02-22 02:03:11	2009-02-22 02:05:40	onTheFlyMap	0.09	230.538	12CO(2-1)	400.67	62.35	-4.90	-24.40	-2.36
2009-02-22.31	3	2009-02-22 02:03:11	2009-02-22 02:06:19	calibrate	0.10	230.538	12CO(2-1)	400.13	62.72	-4.90	-24.40	-2.36
2009-02-22.32	2	2009-02-22 02:06:37	2009-02-22 02:09:09	onTheFlyMap	0.10	230.538	12CO(2-1)	399.82	62.80	-4.90	-24.40	-2.36
2009-02-22.33	2	2009-02-22 02:09:30	2009-02-22 02:12:38	onTheFlyMap	0.12	230.538	12CO(2-1)	399.91	63.18	-4.90	-24.40	-2.36
2009-02-22.34	2	2009-02-22 02:12:57	2009-02-22 02:16:19	onTheFlyMap	0.13	230.538	12CO(2-1)	399.23	63.62	-4.90	-24.40	-2.36
2009-02-22.35	3	2009-02-22 02:12:57	2009-02-22 02:17:00	calibrate	0.10	230.538	12CO(2-1)	398.26	64.09	-4.90	-24.40	-2.36
2009-02-22.36	2	2009-02-22 02:17:19	2009-02-22 02:18:26	onTheFlyMap	0.10	230.538	12CO(2-1)	398.14	64.16	-4.90	-24.40	-2.36
2009-02-22.37	2	2009-02-22 02:18:45	2009-02-22 02:20:23	onTheFlyMap	0.10	230.538	12CO(2-1)	398.32	64.20	-4.90	-24.40	-2.36
2009-02-22.38	2	2009-02-22 02:20:43	2009-02-22 02:22:33	onTheFlyMap	0.09	230.538	12CO(2-1)	397.87	64.57	-4.90	-24.40	-2.36
2009-02-23.14	3	2009-02-23 01:18:34	2009-02-23 01:26:07	calibrate	0.07	230.538	12CO(2-1)	385.87	37.84	-5.10	-23.40	-2.50
2009-02-23.15	3	2009-02-23 01:18:34	2009-02-23 01:31:20	calibrate	0.06	230.538	12CO(2-1)	403.67	58.05	-5.10	-23.40	-2.50
2009-02-23.20	3	2009-02-23 01:31:49	2009-02-23 01:37:25	calibrate	0.05	230.538	12CO(2-1)	402.85	59.38	-5.10	-23.40	-2.50
2009-02-23.21	2	2009-02-23 01:37:43	2009-02-23 01:38:51	onTheFlyMap	0.05	230.538	12CO(2-1)	402.79	59.46	-5.10	-23.40	-2.50
2009-02-23.22	2	2009-02-23 01:39:09	2009-02-23 01:40:46	onTheFlyMap	0.05	230.538	12CO(2-1)	402.93	59.50	-5.10	-23.40	-2.50
2009-02-23.23	2	2009-02-23 01:41:05	2009-02-23 01:42:56	onTheFlyMap	0.10	230.538	12CO(2-1)	402.79	59.77	-5.10	-23.40	-2.50
2009-02-23.24	2	2009-02-23 01:43:16	2009-02-23 01:45:33	onTheFlyMap	0.08	230.538	12CO(2-1)	402.62	60.10	-5.10	-23.40	-2.50
2009-02-23.26	2	2009-02-23 01:48:37	2009-02-23 01:51:06	onTheFlyMap	0.04	230.538	12CO(2-1)	402.08	60.90	-5.10	-23.40	-2.50
2009-02-23.27	3	2009-02-23 01:48:37	2009-02-23 01:51:45	calibrate	0.06	230.538	12CO(2-1)	401.67	61.32	-5.10	-23.40	-2.50
2009-02-23.28	2	2009-02-23 01:52:03	2009-02-23 01:54:36	onTheFlyMap	0.06	230.538	12CO(2-1)	401.24	61.40	-5.10	-23.40	-2.50
2009-02-23.30	2	2009-02-23 01:58:23	2009-02-23 02:01:46	onTheFlyMap	0.07	230.538	12CO(2-1)	400.90	62.26	-5.10	-23.40	-2.50
2009-02-23.31	2	2009-02-23 02:02:04	2009-02-23 02:05:44	onTheFlyMap	0.05	230.538	12CO(2-1)	400.28	62.72	-5.10	-23.40	-2.50
2009-02-23.32	3	2009-02-23 02:02:04	2009-02-23 02:06:20	calibrate	0.06	230.538	12CO(2-1)	399.72	63.24	-5.10	-23.40	-2.50
2009-02-23.33	2	2009-02-23 02:06:41	2009-02-23 02:10:23	onTheFlyMap	0.05	230.538	12CO(2-1)	399.23	63.31	-5.10	-23.40	-2.50
2009-02-23.34	2	2009-02-23 02:10:42	2009-02-23 02:14:26	onTheFlyMap	0.05	230.538	12CO(2-1)	399.07	63.90	-5.10	-23.40	-2.50
2009-02-23.35	2	2009-02-23 02:14:45	2009-02-23 02:18:29	onTheFlyMap	0.06	230.538	12CO(2-1)	398.11	64.34	-5.10	-23.40	-2.50
2009-02-23.36	3	2009-02-23 02:14:45	2009-02-23 02:19:07	calibrate	0.07	230.538	12CO(2-1)	397.64	64.99	-5.10	-23.40	-2.50
2009-02-23.37	2	2009-02-23 02:19:26	2009-02-23 02:23:04	onTheFlyMap	0.07	230.538	12CO(2-1)	397.04	64.92	-5.10	-23.40	-2.50
2009-02-23.38	2	2009-02-23 02:23:22	2009-02-23 02:26:50	onTheFlyMap	0.06	230.538	12CO(2-1)	396.60	65.45	-5.10	-23.40	-2.50
2009-02-23.40	3	2009-02-23 02:27:08	2009-02-23 02:31:11	calibrate	0.08	230.538	12CO(2-1)	394.95	66.37	-5.10	-23.40	-2.50
2009-02-23.41	2	2009-02-23 02:31:30	2009-02-23 02:34:50	onTheFlyMap	0.08	230.538	12CO(2-1)	394.57	66.35	-5.10	-23.40	-2.50
2009-02-23.42	2	2009-02-23 02:35:09	2009-02-23 02:38:19	onTheFlyMap	0.09	230.538	12CO(2-1)	393.93	66.87	-5.10	-23.40	-2.50
2009-02-23.43	2	2009-02-23 02:38:36	2009-02-23 02:41:39	onTheFlyMap	0.05	230.538	12CO(2-1)	393.14	67.37	-5.10	-23.40	-2.50
2009-02-23.44	2	2009-02-23 02:41:58	2009-02-23 02:44:45	onTheFlyMap	0.06	230.538	12CO(2-1)	392.15	67.66	-5.10	-23.40	-2.50
2009-02-23.45	3	2009-02-23 02:41:58	2009-02-23 02:45:23	calibrate	0.09	230.538	12CO(2-1)	391.37	68.09	-5.10	-23.40	-2.50
2009-02-23.46	2	2009-02-23 02:45:41	2009-02-23 02:47:56	onTheFlyMap	0.09	230.538	12CO(2-1)	391.09	67.92	-5.10	-23.40	-2.50
2009-02-23.47	2	2009-02-23 02:48:13	2009-02-23 02:49:13	onTheFlyMap	0.06	230.538	12CO(2-1)	390.42	68.44	-5.10	-23.40	-2.50

>>>

Edit search

Source Name M101_1
Resolver IRAM Name

IRAM - IAA - CSIC

Figure 12.12: TAPAS provides a list of all of the scans composing an observation.

TAPAS - Telescope Archive for Public Access System
IRAM 30m Archive
Welcome Stéphane
Last logged: Sun, 29 Mar 2009
Logout

[Home](#) [Search](#) [Results](#) [News](#) [Policy](#) [Help](#) [About](#) [Admin](#)

Scan Info

SCAN 2009-02-11.62
PROJECT: [d13-08](#)

[iram30m-vespa-20090211s62-imb.fits](#)

DATE: 2009-02-11 03:44:24
SCAN DURATION 5.0 min,
START TIME: 2009-02-11 03:44:24 UT
END TIME: 2009-02-11 03:49:23 UT
LST: 13:00:36
SUBSCANS: 3

SOURCE

NAME: polotf
VELOCITY: 0.00 km/s
RA: 02:00:06
DEC: 07:42:04
J2000

WEATHER

	MIN	MAX
OPACITY @225 GHz	0.04	0.07
WIND VELOCITY	3.8 m/s	4.7 m/s
HUMIDITY	17 %	17 %
PRESSURE	723.9 mb	723.9 mb
TEMPERATURE	-2.0 °C	-1.9 °C

ANTENNA

AZIMUTH: 358.94°
ELEVATION: 35.09°

POINTING CORRECTION X: -1.00°
POINTING CORRECTION Y: -3.60°
FOCUS CORRECTION Z: -3.10 mm

OBSERVATION TYPE

calibrate

SETTINGS

CALIBRATION
AMBIENT: YES
COLD: YES
SKY: YES
GRID: NO

OFFSETS

RECEIVERS

RX FREQUENCY LINENAME	A100 86.847 GHz SIO(V0)
BACKEND	VESPA
RX FREQUENCY LINENAME	A230 220.399 GHz 13CO(2-1)
BACKEND	VESPA
RX FREQUENCY LINENAME	B100 89.189 GHz HCO+(1-0)
BACKEND	VESPA
RX FREQUENCY LINENAME	B230 220.399 GHz 13CO(2-1)
BACKEND	VESPA
CALIBRATION	
TREC	129.10 K
TSYS	246.59 K
TCAL	269.54 K
TATMS	259.81 K
PWV	1.198 mm
FREQUENCY IMAGE	228.915 GHz

SOFTWARE

PaKo v 1.0.9.3

[Edit search](#)

Observation Date 01/01/2009 - 29/03/2009
Line Name CO(2-1) Match Generic

IRAM - IAA - CSIC

Figure 12.13: Details of an on-off calibration scan in TAPAS.

TAPAS - Telescope Archive for Public Access System
IRAM 30m Archive
Welcome Stéphane
Last logged: Sun, 29 Mar 2009
Logout

[Home](#) [Search](#) [Results](#) [News](#) [Policy](#) [Help](#) [About](#) [Admin](#)

Scan Info

SCAN 2009-02-22.21
PROJECT: 215-08

[iram30m-wilma-20090222s21-imb.fits](#)
[iram30m-4mhz-20090222s21-imb.fits](#)

DATE: 2009-02-22 01:43:13
SCAN DURATION: 1.1 min.
START TIME: 2009-02-22 01:43:13 UT
END TIME: 2009-02-22 01:44:22 UT
LST: 11:37:49
SUBSCANS: 2

SOURCE

NAME: M101_1
VELOCITY: 241.00 km/s
RA: 14:03:17
DEC: 54:24:12
J2000

WEATHER

	MIN	MAX
OPACITY @225 GHz	0.09	0.09
WIND VELOCITY	0.9 m/s	2.3 m/s
HUMIDITY	18 %	18 %
PRESSURE	723.8 mb	723.8 mb
TEMPERATURE	-3.8 °C	-3.8 °C

ANTENNA

AZIMUTH: 402.64°
ELEVATION: 59.68°

POINTING CORRECTION X: -4.90"
POINTING CORRECTION Y: -24.40"
FOCUS CORRECTION Z: -2.36 mm

OBSERVATION TYPE
 onTheFlyMap

SETTINGS

OTF

X START: 792"
Y START: -323"
X END: 792"
Y END: 161"
X STEP: -11"
Y STEP: 0"
SPEED START: 16 "/s
SPEED END: 16 "/s
TIME PER OTF: 30 s
TIME PER REFERENCE: 10 s

OFFSETS

RECEIVERS

RX	HERA1 Pixel 1
FREQUENCY	230.538 GHz
LINENAME	12CO(2-1)
BACKEND	WILMA
RX	HERA2 Pixel 1
FREQUENCY	230.538 GHz
LINENAME	12CO(2-1)
BACKEND	4MHz
BACKEND	WILMA

SOFTWARE
 PaKo v 1.0.9.3

[Edit search](#)

Observation Date 01/01/2009 - 29/03/2009
 Line Name CO(2-1) Match Generic

IRAM - IAA - CSIC

Figure 12.14: Details for a scan belonging to an OTF map.

TAPAS - Telescope Archive for Public Access System
Welcome Stéphane
Last logged: Sun, 29 Mar 2009
Logout

Home Search Results News Policy Help About Admin

Sources List

Format HTML Return 50 rows

PROJECT: d13-08

Total number of sources: 52 >>>

Source	Velocity	# Scans	J2000 RA	J2000 Dec.	Receiver	Opacity	First Scan	Last Scan
0316+413	0.00	4	03:19:48	41:30:42	A100	0.02	2009-02-10 22:59:45	2009-02-10 23:25:39
0316+413	0.00	4	03:19:48	41:30:42	A230	0.02	2009-02-10 22:59:45	2009-02-10 23:25:39
0316+413	0.00	4	03:19:48	41:30:42	B100	0.02	2009-02-10 22:59:45	2009-02-10 23:25:39
0316+413	0.00	4	03:19:48	41:30:42	B230	0.02	2009-02-10 22:59:45	2009-02-10 23:25:39
0923+392	0.00	1	09:27:03	39:02:20	A100	0.05	2009-02-13 04:56:30	2009-02-13 04:58:45
0923+392	0.00	1	09:27:03	39:02:20	A230	0.05	2009-02-13 04:56:30	2009-02-13 04:58:45
0923+392	0.00	1	09:27:03	39:02:20	B100	0.05	2009-02-13 04:56:30	2009-02-13 04:58:45
0923+392	0.00	1	09:27:03	39:02:20	B230	0.05	2009-02-13 04:56:30	2009-02-13 04:58:45
0953+254	0.00	1	09:56:49	25:15:15	A100	0.07	2009-02-13 04:51:52	2009-02-13 04:54:06
0953+254	0.00	1	09:56:49	25:15:15	A230	0.07	2009-02-13 04:51:52	2009-02-13 04:54:06
0953+254	0.00	1	09:56:49	25:15:15	B100	0.07	2009-02-13 04:51:52	2009-02-13 04:54:06
0953+254	0.00	1	09:56:49	25:15:15	B230	0.07	2009-02-13 04:51:52	2009-02-13 04:54:06
1039+811	0.00	7	10:44:23	80:54:39	A100	0.05	2009-02-10 23:31:00	2009-02-16 04:29:34
1039+811	0.00	7	10:44:23	80:54:39	A230	0.05	2009-02-10 23:31:00	2009-02-16 04:29:34
1039+811	0.00	7	10:44:23	80:54:39	B100	0.05	2009-02-10 23:31:00	2009-02-16 04:29:34
1039+811	0.00	7	10:44:23	80:54:39	B230	0.05	2009-02-10 23:31:00	2009-02-16 04:29:34
1226+023	0.00	3	12:29:06	02:03:08	A100	0.05	2009-02-12 03:58:45	2009-02-12 04:41:31
1226+023	0.00	3	12:29:06	02:03:08	A230	0.05	2009-02-12 03:58:45	2009-02-12 04:41:31
1226+023	0.00	3	12:29:06	02:03:08	B100	0.05	2009-02-12 03:58:45	2009-02-12 04:41:31
1226+023	0.00	3	12:29:06	02:03:08	B230	0.05	2009-02-12 03:58:45	2009-02-12 04:41:31
1730-130	40.00	2	17:33:02	-13:04:49	A100	0.06	2009-02-14 06:01:37	2009-02-15 06:18:28
1730-130	40.00	2	17:33:02	-13:04:49	A230	0.06	2009-02-14 06:01:37	2009-02-15 06:18:28
1730-130	40.00	2	17:33:02	-13:04:49	B100	0.06	2009-02-14 06:01:37	2009-02-15 06:18:28
1730-130	40.00	2	17:33:02	-13:04:49	B230	0.06	2009-02-14 06:01:37	2009-02-15 06:18:28
1803+784	0.00	1	18:00:45	78:28:04	A100	0.06	2009-02-12 04:51:39	2009-02-12 04:53:53
1803+784	0.00	1	18:00:45	78:28:04	A230	0.06	2009-02-12 04:51:39	2009-02-12 04:53:53
1803+784	0.00	1	18:00:45	78:28:04	B100	0.06	2009-02-12 04:51:39	2009-02-12 04:53:53
1803+784	0.00	1	18:00:45	78:28:04	B230	0.06	2009-02-12 04:51:39	2009-02-12 04:53:53
core1	0.00	3	02:01:05	87:41:57	A100	0.05	2009-02-12 04:51:39	2009-02-12 05:01:35
core1	0.00	3	02:01:05	87:41:57	A230	0.05	2009-02-12 04:51:39	2009-02-12 05:01:35
core1	0.00	3	02:01:05	87:41:57	B100	0.05	2009-02-12 04:51:39	2009-02-12 05:01:35
core1	0.00	3	02:01:05	87:41:57	B230	0.05	2009-02-12 04:51:39	2009-02-12 05:01:35
HIP48541	0.00	22	09:53:59	27:41:43	A100	0.06	2009-02-13 04:56:30	2009-02-15 05:00:01
HIP48541	0.00	22	09:53:59	27:41:43	A230	0.06	2009-02-13 04:56:30	2009-02-15 05:00:01
HIP48541	0.00	22	09:53:59	27:41:43	B100	0.06	2009-02-13 04:56:30	2009-02-15 05:00:01
HIP48541	0.00	22	09:53:59	27:41:43	B230	0.06	2009-02-13 04:56:30	2009-02-15 05:00:01
HIP60074	0.00	9	12:19:06	16:32:53	A100	0.05	2009-02-13 04:17:28	2009-02-13 04:49:23
HIP60074	0.00	9	12:19:06	16:32:53	A230	0.05	2009-02-13 04:17:28	2009-02-13 04:49:23
HIP60074	0.00	9	12:19:06	16:32:53	B100	0.05	2009-02-13 04:17:28	2009-02-13 04:49:23
HIP60074	0.00	9	12:19:06	16:32:53	B230	0.05	2009-02-13 04:17:28	2009-02-13 04:49:23
I1157	0.00	3	20:39:06	68:02:15	A100	0.05	2009-02-11 02:11:33	2009-02-11 02:22:56
I1157	0.00	3	20:39:06	68:02:15	A230	0.05	2009-02-11 02:11:33	2009-02-11 02:22:56
I1157	0.00	3	20:39:06	68:02:15	B100	0.05	2009-02-11 02:11:33	2009-02-11 02:22:56
I1157	0.00	3	20:39:06	68:02:15	B230	0.05	2009-02-11 02:11:33	2009-02-11 02:22:56
polotf	0.00	134	02:00:06	87:42:04	A100	0.04	2009-02-10 23:31:00	2009-02-16 05:50:57
polotf	0.00	134	02:00:06	87:42:04	A230	0.04	2009-02-10 23:31:00	2009-02-16 05:50:57
polotf	0.00	134	02:00:06	87:42:04	B100	0.04	2009-02-10 23:31:00	2009-02-16 05:50:57
polotf	0.00	134	02:00:06	87:42:04	B230	0.04	2009-02-10 23:31:00	2009-02-16 05:50:57
Saturn	33.20	17			A100	0.05	2009-02-11 03:59:01	2009-02-16 04:22:38
Saturn	33.20	17			A230	0.05	2009-02-11 03:59:01	2009-02-16 04:22:38

>>>

Edit search

Observation Date 01/01/2009 - 29/03/2009
Line Name CO(2-1) Match Generic

IRAM - IAA - CSIC

Figure 12.15: TAPAS provides access to all sources observed in a given project.

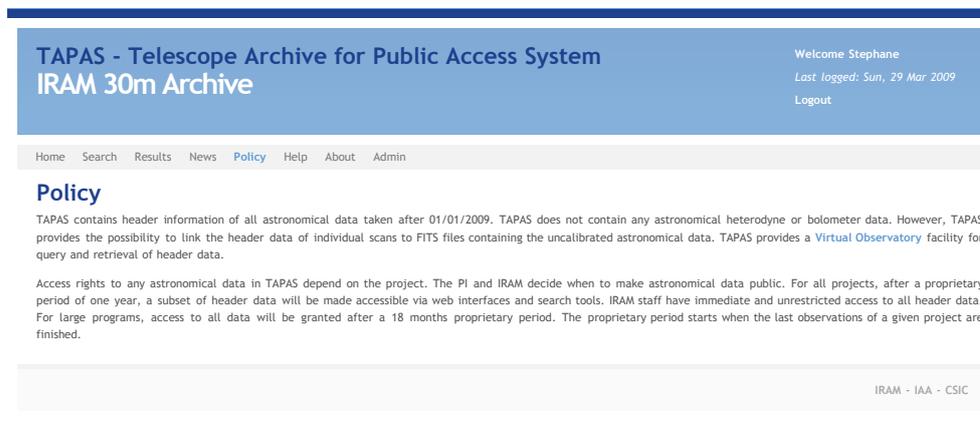


Figure 12.16: TAPAS policy, as specified by the IRAM.

access to all data will be granted after a 18 months proprietary period. The proprietary period starts when the last observations of a given project are finished.

The TAPAS archive has been led by Lourdes Verdes-Montenegro (IAA-CSIC), as PI of the coordinated project, and Rainer Mauersberger (IRAM), as co-PI for the project and leader for the IRAM part, with Stéphane Leon (IRAM) as the lead scientist.

On the development part, the development and design of the TAPAS archive infrastructure, data models, and interfaces has been the responsibility of the IAA-CSIC team, while integration issues have been solved by the IRAM team.

In particular, the data model was developed by Juan de Dios Santander Vela (IAA-CSIC) from the RADAMS, and it was adapted by Víctor Espigares (IAA-CSIC) to Django. Victor Espigares is also responsible for the Data-filler—the Archive Infrastructure module in figure 12.2—, and José Enrique Ruiz (IAA-CSIC) has developed the web interface and the VO Services—the Frontend layer in figure 12.6—.

Helmut Wiesemayer (IRAM) modified the calibration packages (MIRA, paKo) to produce extra XML files used by the Data-filler for online data ingestion of heterodyne calibration data, and Walter Brunswig (IRAM) provided the system integration services in the NCS for accessing FITS files, installing the Data-filler, and all supporting packages.

12.3 Conclusions

In this chapter we have shown how the RADAMS, developed and described in its different modules in chapters 5, 6, 7, and 9, is able to provide the

foundation for the development not just for data archives built *from scratch*, such as that for the DSS-63 antenna, but also for archives being built on top of an existing archival infrastructure, such as that of the IRAM 30m antenna.

The many different observing modes, frontends, backends, mappings between frontend and backends, and the need to support both raw and processed data, have enriched the RADAMS with respect to its initial incarnation, to the point of providing the desired basis.

In the case of the IRAM 30m archive, the IRAM data access policy (no actual data available through the archive interface at least for 18 months since the end of an observing project, and access to header —pointing and observing configuration— information for observations after 12 months) have contributed to validate the Policy data model, and the Instrument, Software Processing and Calibration Provenance.

Chapter 13

Enhancing massa to support VO datasets

In chapter 10 we discussed what constitutes a VO application, and we reached the conclusion that a VO application could be one that supported VO application messaging, and VO file formats. Such an application could rely on existing VO Data Access modules, and share the data products it creates with other tools.

Besides, in chapter 11 we further discussed what kind of messages should be implemented by MOVOIR applications, and saw two different kind of messages:

Standard VO data interchange messages These are messages from the standard suite of SAMP MTypes, that is, those shown on section 11.2.

MOVOIR specific messages We can further divide this kind of messages into:

Self-description messages Those used in order to provide a self-discovery, and self-description API which can be used by automated tools in order to create appropriate messages. These were proposed in section 11.4.

Function messages These are messages created for implementing direct access to the functionality offered by a MOVOIR module.

We further established in section 11.6 which should be the modules needed in order to be able to provide a complete VO environment for legacy applications:

VO Downloader Application to enable the download of VO Tables and FITS files, using `table.load.*` and `image.load.fits` messages.

VOTable to FITS converter Similar to the VO Downloader, the VOTable converter is an application which creates FITS files from `table.load.votable` messages, saving the converted FITS file.

VO Registry and DAL module This module is left to already existing applications, such as the VO Desktop.

In addition, two more modules can be created to show the flexibility of the MOVOIR framework:

AMIGA ConeSearch module This module is an example for using the `coord.pointAt.sky` MType to send back as a `table.load.votable` message the results of a ConeSearch of a fixed radius around the ra and dec coordinates obtained by clicking on an image with a World Coordinate System set on applications supporting both sending `coord.pointAt.sky` and receiving `table.load.votable`.

Sesame query module This module is an example of a non-standard MType supporting module, so that MOVOIR provides a multi-object Sesame query message, as support for a future multi-cone search service, which can resolve multiple objects at once.

In this chapter, we will show how we have built each of these modules, and which have been the key points to take into account during the implementation for each of them, with the final goal of incorporating massa into the VO.

13.1 VO Downloader

In order to create the VO Downloader, we started with the source code for a simple Java-based URL downloader application¹, consisting of only 4 classes: the `DownloadManager` class represents the main View and Controller classes; the `Download` class is the threaded Controller and Model for each actual download; the `DownloadTableModel` class is the Model for the Table View in `DownloadManager`; and `DownloadProgressRenderer` is a delegate class for the renderer sub-view.

In order to be able to download files sent from any application, the VO Downloader needs to respond to all data load messages—`table.load.votable`, `table.load.fits`, and `image.load.fits`— In addition, a generic `movoir.download.file` is supported, plus the `movoir.describe.mtype` MType.

¹<http://www.sourcecodesworld.com/source/show.asp?ScriptId=1185>

Implementation details

As the VO Downloader is a Java application, we will use the JSAMP² Java-based SAMP toolkit in order to implement JSAMP support.

In section 10.4 we indicated that, in principle, the changes to an existing application for the application to incorporate SAMP were minor, and were restricted to application setup, application shutdown, and the creation of the handlers for SAMP messages. We will show that in VO Downloader, and the solution is exactly the same for massa.

Figure 13.1 shows the `initSamp()` function, which performs the initialisation of the SAMP messaging module within VO Downloader. `initSamp()` is the last call in the VO Downloader creator of the `DownloadManager`.

The MOVOIR VO Downloader interface is show in figure 13.2, while the VO Downloader detail in the JSAMP HubMonitor can be seen in figure 13.3.

We can see that `initSamp()` provides an abstract SAMP message handler, `handleVotable()`, which receives three parameters:

HubConnection hubConnect This parameter holds an instance of the JSAMP `HubConnection` class, and is profile specific. In the case of the SAMP standard profile, this parameter holds the details for connecting to the hub via XML-RPC.

String senderId This parameter provides a string with the unique identifier of the application that sent (or broadcasted) a message to our handler. We will use this in order to create messages back to the sender with the result of our computations. In the case of VO Downloader, the results is always a `samp.ok` answer, or a `samp.error` if the message does not provide a `url` parameter within the `samp.params` map.

Message msg The `msg` parameter is the map described in section 11.1, and provides all the actual message information: `samp.mtype`, for choosing the function to be performed, and `samp.params` for the function to work.

In the case of VO Downloader, the messages which are implemented are `table.load.votable`, `image.load.fits`, `table.load.fits`, and the MOVOIR-specific `movoir.download.file` (passed as a list of `MType` strings in `loadMTypes`). All of those messages need to provide a `url` parameter.

If `url` is present, the already implemented `addAction(String urlString)` method is called in order to queue the download of the corresponding data. We can see that the `addAction` method already existed in VO Downloader, and our handler just delegates the actual behaviour to an already existing function.

²<http://deployer.astrogrid.org/software/jsamp/>

```

public void initSamp() {
    // Get standard instance for the StandardClientProfile
    // (static methods of the corresponding classes)
    if (theHubConnector == null) {
        ClientProfile theClientProfile = StandardClientProfile.getInstance();
        theHubConnector = new HubConnector(theClientProfile);
    }

    // Before connecting, set up metadata
    Metadata theMetadata = new Metadata();
    theMetadata.setName("movoirVODownloader");
    theMetadata.setDescriptionText("MOVIR VO Downloader module."+
        "Downloads FITS files and VOTables sent to it via url.file.download,"+
        "table.load.votable and image.load.fits MTypes.");
    theMetadata.setIconUrl("http://amiga.iaa.es:8080/FCKeditor/UserFiles/Image/composicion3size.gif");
    theMetadata.setDocumentationUrl("http://movoir.sourceforge.net/");

    // Use the Metadata map, prepare it for the connection
    theHubConnector.declareMetadata(theMetadata);

    // Use the corresponding MTypes and functions
    MessageHandler handleVotable = new AbstractMessageHandler(loadMTypes) {

        @Override
        public Map processCall(HubConnection arg0, String senderId, Message msg) throws Exception {
            // Get Message params
            // Prepare for download:
            HashMap sampResult = new HashMap();
            Map msgParams = msg.getParams();

            if (!msgParams.containsKey("url")) {
                // We need the URL parameter in the messages
                // we are subscribed to.
                sampResult.put("samp.status", "samp.error");
            } else {
                sampResult.put("samp.status", "samp.ok");
                actionAdd((String)msgParams.get("url"));
            }

            return sampResult;
        }
    };

    theHubConnector.addMessageHandler(handleVotable);

    // Once all handlers have been declared, prepare the subscription list
    theHubConnector.declareSubscriptions(theHubConnector.computeSubscriptions());
    theHubConnector.setAutoconnect(10); // Connect, with reconnection after 10s

    // We're done initing!
    // throw new UnsupportedOperationException("Not yet implemented");
}

```

Figure 13.1: Section of VO Downloader source code: `initSamp()` function, last in the VO Downloader creator.

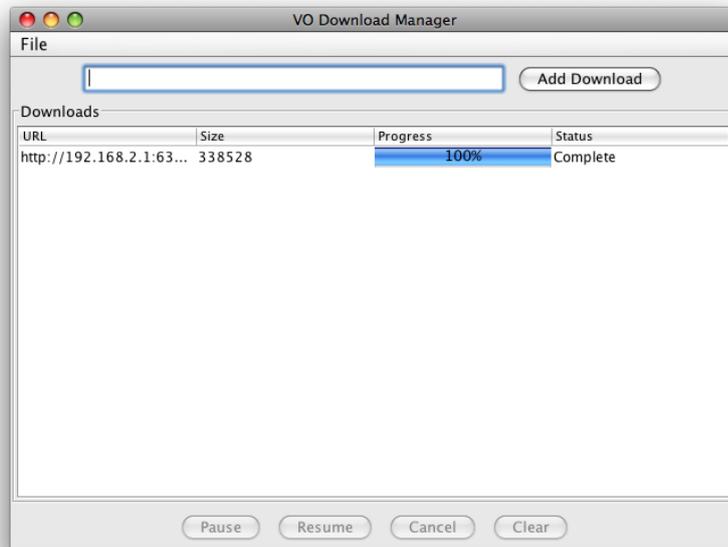


Figure 13.2: VO Downloader, with one download complete after receiving a `table.load.votable` from TOPCAT.

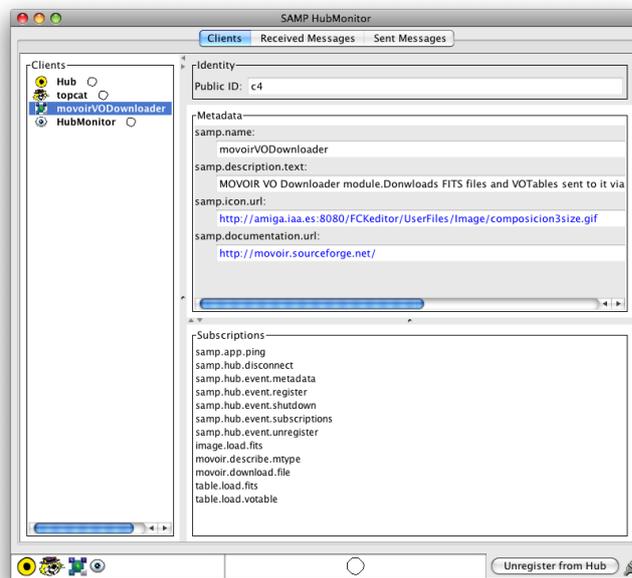


Figure 13.3: JSAMP Hub Monitor showing the supported messages of the VO Downloader

An additional `shutdownSamp()` method is called from the `actionExit()` method. Following the flow diagram shown in figure 10.4, we have just modified:

VO Downloader creator We have added an `initSamp` method to declare the messages we are prepared to handle, at the end of the creator of the `DownloadManager` class of the VO Downloader.

Event handling The event handling is modified by the same `initSamp` method, by adding listeners to SAMP messages with the `addMessageHandler` method. The message handler uses existing controller methods to delegate the actual implementation of the handler.

VO Downloader shutdown In the case of the VO Downloader, the original `DownloadManager` shutdown is handled by the `actionExit` method, and the shutdown of the SAMP messaging in VO Downloader is implemented as the first function in that method. For other applications, equivalent placements, if possible within a guaranteed execution thread, need to be chosen.

In order to implement the `movoir.describe.mtype` method, the easiest way is to create a separate handler (to be added with `addMessageHandler` of the `HubConnector` class), that use statically created maps. However, it is simpler, and more illustrative to create the result map within the handler itself. Figure 13.4 shows the listing of the `movoir.describe.mtype` method handler supporting just the `table.load.votable` MType.

13.2 VO Data Converter

The VO Data Converter is another MOVOIR module needed to support those legacy applications for which source code is not available. As legacy applications will usually provide zero support for VOTables, data sent to the VO Data Converter will only be directly useful for legacy applications in FITS or ASCII formats.

But in order to simplify that process, and eliminating the conversion step after download, we will create two separate VO Data Converters: one that converts VO tables into FITS tables, and another one which converts VO tables into CSV files.

Implementation details

As only VOTables will need to be converted (FITS files are usually supported by legacy applications), those two modules need to be subscribed just to the `table.load.votable` MType. The response they need to provide to that message is the conversion to to the corresponding data format.

```

MessageHandler handleMvoirDescribe =
    new AbstractMessageHandler(describeMTypes) {

    @Override
    public Map processCall(HubConnection hubConnection,
        String senderId, Message msg) throws Exception {
        HashMap sampResultMap = new HashMap();
        HashMap urlDescriptionMap = new HashMap();
        urlDescriptionMap.put("mvoir.description",
            "URL of the FITS file to be loaded.");
        urlDescriptionMap.put("optional", "false");
        urlDescriptionMap.put("type", "string");
        urlDescriptionMap.put("ucd", "meta.url");

        HashMap tableIdDescriptionMap = new HashMap();
        tableIdDescriptionMap.put("mvoir.description",
            "Identifier which may be used to refer to the loaded" +
            "table in subsequent messages.");
        tableIdDescriptionMap.put("optional", "true");
        tableIdDescriptionMap.put("type", "string");
        tableIdDescriptionMap.put("ucd", "meta.id");

        HashMap nameDescriptionMap = new HashMap();
        nameDescriptionMap.put("mvoir.description",
            "Name which may be used to label the loaded table in" +
            "the application GUI.");
        nameDescriptionMap.put("optional", "true");
        nameDescriptionMap.put("type", "string");
        nameDescriptionMap.put("ucd", "meta.name");

        HashMap tableLoadVotableParameterMap = new HashMap();
        tableLoadVotableParameterMap.put("url", urlDescriptionMap);
        tableLoadVotableParameterMap.put("table-id",
            tableIdDescriptionMap);
        tableLoadVotableParameterMap.put("name", nameDescriptionMap);

        Map msgParamsMap = msg.getParams();
        String errorNoMType =
            "The mvoir.describe.mtype needs a valid MType" +
            "parameter.";

        if (!msgParamsMap.containsKey("mtype")) {
            sampResultMap.put("samp.status", "samp.error");
            sampResultMap.put("samp.error", errorNoMType);
        } else {
            String theMType = msgParamsMap.get("mtype").toString();
            if (theMType.equals("table.load.votable")) {

                sampResultMap.put("parameters",
                    tableLoadVotableParameterMap);
                sampResultMap.put("mvoir.description",
                    "Download a table in VOTable format from a given" +
                    "URL in HTTP, FTP, or FILE protocols.");
            }
        }
        return sampResultMap;
    }
};

```

Figure 13.4: Partial listing of the `mvoir.describe.mtype` SAMP message handler, with the response map created on the fly.

We will write both modules (which will be quite similar) in Python, using the `SAMPy`³ module developed by Luigi Paioro. They will also make use of the `STILTS`⁴, by Mark Taylor, called as a command line tool from the Python scripts, which will also make use of the `curl` command for consolidating the download of the URL parameter.

By using a different language we are also validating the multi-language support of the MOVOIR, which allows to use the language which provides the most ease of implementation for each module.

The complete listing of the Votable to FITS converter module is shown in figure 13.5.

Taking into account that the `table.load.votable` message can be sent both as a notification (specially when broadcasted), or as a call (synchronous or asynchronous), the module needs to be able to receive that message in any of those two flavours.

That is achieved by means of a single processing function, `convertVotable2FITS`, which is called from the handlers for notifications or calls.

The last part of the `mvoirVOT2FITSconvert` module is, in fact, the part that it is executed first. It starts with the declaration of metadata and the the

³<http://cosmos.iasf-milano.inaf.it/pandora/sampy.html>

⁴<http://www.starlink.ac.uk/stilts/>

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-
# movoirVOT2FITSconverter

import os
import sampy
import urllib
import pdb

downloadFolder="/Users/jdsant/Downloads/"

vot2fitsMD = {
    'samp.name': 'movoirV2FITS',
    'samp.description.text': 'Conversor de VOTable a FITS',
    'movoir.version': '0.1',
    'samp.icon.url': 'http://www.iaa.es/~jdsant/thesis/v2fitsIcon.png'
}

def convertVotable2FITS(votableUrl, pFileName):
    print "Entering convertVotable2FITS", votableUrl, pFileName
    # pdb.set_trace()
    # Download url 2 temp folder
    extension = '.fits'
    fileName = ""
    if pFileName=="":
        fileName = votableUrl.split('/')[-1]
    else:
        fileName = urllib.quote(pFileName)

    print "FileName:", fileName
    curlCommand = "/usr/bin/curl -s -o "+downloadFolder+fileName+" "+votableUrl+"
    print "Executing command:\n"+curlCommand+"\n"
    osCode = os.system(curlCommand)
    if osCode != 0:
        print "Download failed"
    else:
        # We succeeded
        # Lets convert!
        stiltsCommand = "./stilts tcopy "+downloadFolder+fileName
        stiltsCommand = stiltsCommand+" "+downloadFolder+fileName+extension
        stiltsCommand = stiltsCommand+" ifmt=votable ofmt=fits"
        print "Executing command:\n"+stiltsCommand+"\n"
        os.system(stiltsCommand)
        os.system("open "+downloadFolder)

    # Call STILTS to convert file
    # Open folder

def notificationHandler(private_key, sender_id, mtype, params, extra):
    #pdb.set_trace()
    print ("notifHandler", private_key, sender_id, mtype, params, extra)
    if mtype == "table.load.votable":
        if params.has_key('url'):
            tableName=""
            if params.has_key('name'):
                tableName=params['name']
            print "In the good branch",params['url'], tableName
            convertVotable2FITS(params['url'], tableName)

def callHandler(private_key, sender_id, msg_id, mtype, params, extra):
    # As this is a call, we should return an empty samp.result
    # after having processed it
    print ("callHandler:", private_key, sender_id, msg_id, mtype, params, extra)
    if mtype == "table.load.votable":
        if params.has_key('url'):
            tableName=""
            if params.has_key('name'):
                tableName=params['name']
            print "In the good branch",params['url'], tableName
            convertVotable2FITS(params['url'], tableName)
            replyMsg = {'samp.status': sampy.SAMP_STATUS_OK,
                        'samp.result': {}}
            vot2fitsClient.reply(msg_id, replyMsg)
        else:
            print "In the error branch"
            replyMsg = {'samp.status': sampy.SAMP_STATUS_ERROR,
                        'samp.result': {},
                        'samp.error': {'samp.error.txt': 'No url parameter provided.'}}
            vot2fitsClient.reply(msg_id, replyMsg)

def responseHandler(private_key, sender_id, msg_id, response):
    # Response received, typically an OK response,
    # we should not be receiving none of these
    print ("Receiving response", private_key, sender_id, msg_id, response)

# Start binding behaviours
vot2fitsClient = sampy.SAMPIntegratedClient(vot2fitsMD)
vot2fitsClient.connect()
vot2fitsClient.bindReceiveNotification("table.load.votable", notificationHandler)
vot2fitsClient.bindReceiveCall("table.load.votable", callHandler)
#movoirVOT2FITSconverter.vot2fitsClient.disconnect()

```

Figure 13.5: Complete listing of `movoirVOT2FITSconverter.py`.

registration of the module with the hub. Later, the calls and notifications to the `table.load.votable` message are bound, respectively, to the `callHandler` and `notificationHandler` functions, respectively.

The handler functions perform a sanity check of the called parameters, such as ensuring that the message is indeed a `table.load.votable` message, and that it provides the mandatory `url` parameter. Once this is ensured, the control is passed to the `convertVotable2FITS` function, which performs the actual conversion.

`convertVotable2FITS` is straightforward: the file name is obtained from the URL, a `curl` call for data download is setup first, preparing next the `stilts` call. Once setup, they are called sequentially. Finally, the `open` call is used⁵ to open the conversion folder and reveal both the original and converted files.

The only meaningful difference between the `movoirVOT2FITSconvert` and `movoirVOT2CSVconvert` modules is the output parameter sent the `stilts` command, and the file extension.

⁵The `open` command is specific to Mac OS X; an alternative in Linux, with the Gnome desktop installed, is the `gnome-open` command.

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-
# movoirAMIGAcone

import samPy
coordMessage = "coord.pointAt.sky"
coneSearchEndpoint='http://vizier.u-strasbg.fr/viz-bin/votable/?-source=J/A+A/472/121A'

amigaCSmd = {
    'samp.name': 'AMIGA isolation',
    'samp.description.text': 'coord.pointAt.sky client that provides a ConeSearch to the
AMIGA V. Isolation parameters (Verleye, 2007) at Vizier.',
    'movoir.version': '0.1',
    'samp.icon.url': 'http://thesaurus.maths.org/mkb/media/png/Cone.png'
}

def returnConeSearchUrl(sender_id, ra, dec, sr):
    # We want to send a "table.load.votable" message
    # with the corresponding ConeSearch query URL
    csUrl = coneSearchEndpoint+"RA="+str(ra)+"&DEC="+str(dec)
    csUrl = csUrl + "&SR="+str(sr)
    print "CS URL: ", csUrl
    tableLoadMsg = {'samp.mtype': 'table.load.votable', 'samp.params': {'url': csUrl,
'table-id': csUrl, 'name': 'AMIGA CS clickAt response'}}
    print "Message: ", tableLoadMsg
    senderMeta = amigaCSClient.getMetadata(sender_id)
    print "Sender_id: ", sender_id, "App: ", senderMeta['samp.name']
    amigaCSClient.call(sender_id, "clickAMIGA", tableLoadMsg)

def notificationHandler(private_key, sender_id, mtype, params, extra):
    #pdb.set_trace()
    print ("notifHandler", private_key, sender_id, mtype, params, extra)
    if mtype == coordMessage:
        if params.has_key('ra') and params.has_key('dec'):
            ra = params['ra']
            dec = params['dec']
            sr = 0.5 # Half a degree
            if params.has_key('sr'):
                sr = params['sr']
            print sender_id, ra, dec, sr
            returnConeSearchUrl(sender_id, ra, dec, sr)

def callHandler(private_key, sender_id, msg_id, mtype, params, extra):
    # As this is a call, we should return an empty samp.result
    # after having processed it
    print ("callHandler: ", private_key, sender_id, msg_id, mtype, params, extra)

if mtype == coordMessage:
    if params.has_key('ra') and params.has_key('dec'):
        ra = params['ra']
        dec = params['dec']
        sr = 0.5 # Half a degree
        if params.has_key('sr'):
            sr=params['sr']
        print sender_id, ra, dec, sr
        returnConeSearchUrl(sender_id, ra, dec, sr)
    else:
        print "In the error branch"
        replyMsg = {'samp.status': samPy.SAMP_STATUS_ERROR,
'samp.result': {},
'samp.error': {'samp.errortxt': 'No url parameter provided.'}}
        amigaCSClient.reply(sender_id, msg_id, replyMsg)

def responseHandler(private_key, sender_id, msg_id, response):
    # Response received, typically an OK response,
    # we should not be receiving none of these
    print ("Receiving response", private_key, sender_id, msg_id, response)

# Start binding behaviours
amigaCSClient = samPy.SAMPIntegratedClient(amigaCSmd)
amigaCSClient.connect()
amigaCSClient.bindReceiveNotification(coordMessage, notificationHandler)
amigaCSClient.bindReceiveCall(coordMessage, callHandler)
amigaCSClient.bindReceiveResponse("clickAMIGA", responseHandler)

movoirAMIGAcone.amigaCSClient.disconnect()

```

Figure 13.6: Complete listing of movoirAMIGAcone.py.

13.3 ConeSearcher

The ConeSearcher module is a very straightforward, Python-based module, which sends a `table.load.votable` message back with the results of a ConeSearch around a region of the sky where the user clicked in applications such as Aladin, which send a `coord.pointAt.sky` MType.

Given that the ConeSearch URL can be built by concatenating the archive endpoint, and the parameters RA, DEC, and SR, it is fairly easy to write the `table.load.votable` message in response to the `coord.pointAt.sky` message.

Implementation details

Figure 13.6 shows the complete listing for a ConeSearch client based upon the `coord.pointAt.sky` message and corresponding coordinates. In this case, we use the ConeSearch endpoint for the AMIGA catalogue of isolation parameters held at Vizier.

As this implementation is very similar to the VO Data Converter modules, we will just point out that the key is the `returnConeSearchUrl`, which creates the URL to send back via the `table.load.votable` message, and takes care of providing a suitable name for the new table.

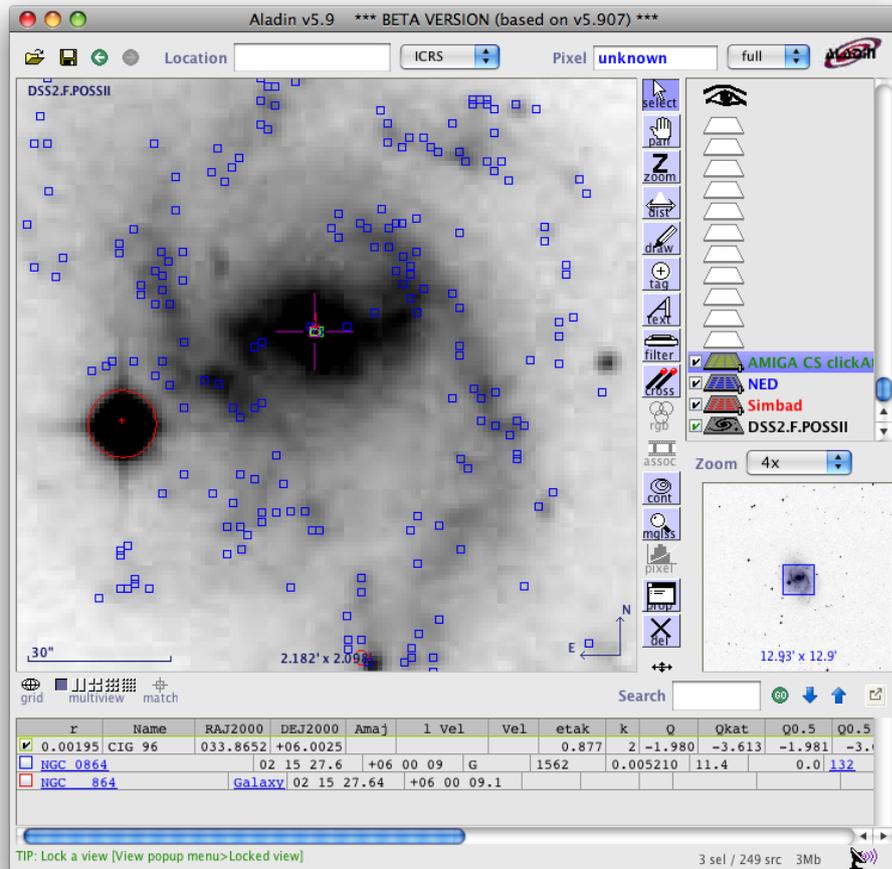


Figure 13.7: Aladin Sky Atlas showing the result of having clicked near NGC 804, which is one of the galaxies in the AMIGA sample, while the MOVOIR ConeSearch module is active.

Figure 13.7 shows a screen capture of the Aladin Sky Atlas displaying NGC 804, which is one of the members of the AMIGA sample of isolated galaxies⁶. Over the optical image, Aladin is overlaying data from NED (blue marks) and Simbad (red marks), but we can also see another layer in light green, AMIGA CS clickAt, which has been generated from a ConeSearch activated by a click on Aladin, which in turn generates the `coord.pointAt.sky`, which activates the ConeSearcher module.

It is trivial to update the module to perform ConeSearches in arbitrary endpoints, and indeed it would be straightforward to update the module in order to incorporate the `movoir.configuration.set` message, in order to set, for instance, a different endpoint, or a different default search radius.

Other trivial uses of `coord.pointAt.sky` listeners which provide `table.load.votable` messages as a result would include synthesising the answers from different ConeSearches (which can be performed in different threads to minimise wait time).

More sophisticated uses can be the development of scripts which directly query large survey's databases (for instance, SDSS queries using the `cas jobs` Java interface), overlaying the objects (and properties) on a certain part of the sky resulting from a given query. Of course, once the IVOA Table Access Protocol is finished, and compatible services start to appear, including remote cross-matching capabilities, the possibilities increase significantly.

13.4 AppleScript XML-RPC SAMP tester

This is not a proper MOVOIR module. Instead, this is a small testing module that was quickly implemented in order to show the bare XML-RPC interface of the SAMP protocol (in the standard profile).

Figure 13.8 contains the listing of a sample AppleScript client which connects to a SAMP hub, collects some information, and sends a `table.votable.load` message.

In order to do that, the `.samp` file is found, loaded, and parsed in order to get the values of the `samp.secret` and `samp.hub.xmlrpc.url` keys, while the **tell application** block performs all the XML-RPC communication is held.

The steps performed by this script are: registering with the hub, interchanging a secret common to hub and application; declare the application metadata; getting a list of all clients registered with the hub; getting a list of all clients subscribed to `table.load.votable`; create and broadcast a `table.load.votable` to a local VOTable; remain

Even without knowledge of AppleScript, this example illustrates the fact that SAMP messages are indeed pure XML-RPC messages, where parameters are sent by order, but that limitation can be overcome by using maps

⁶Also known as CIG 96, or K73 96

```

-- Variable setup
set samp_file_path to "/Users/jdsant/samp"
set samp_secret to ""
set samp_hub_xmlrpc_url to ""
set samp_client_key to ""

-- Read the contents of the .samp file
set theFile to POSIX file samp_file_path
open for access theFile
set sampFileContent to (read theFile)
close access theFile

-- Use an AppleScript trick to get the different sampleLines
set prevTIDs to AppleScript's text item delimiters
set myDelimiter to "\n"
set AppleScript's text item delimiters to myDelimiter
set sampleLines to text items of sampFileContent
set AppleScript's text item delimiters to prevTIDs

-- For each line, look if the key name corresponds with either
repeat with sampleLine in sampleLines
    if sampleLine contains "=" then
        set AppleScript's text item delimiters to "="
        set itemsToParse to text items of sampleLine
        set AppleScript's text item delimiters to prevTIDs
        if the first item of itemsToParse is equal to "samp_secret" then
            set samp_secret to the second item of itemsToParse
        end if
        if the first item of itemsToParse is equal to "samp_hub_xmlrpc_url" then
            set samp_hub_xmlrpc_url to the second item of itemsToParse
        end if
    end if
end repeat

set table_votable_load_subscribers to {}

-- Prepare client metadata to be sent to the hub
set client_metadata to {[samp.name]:"AS_SAMP", [samp.description.text]:"An
AppleScript XML-RPC based SAMP client.", [samp.documentation.url]:"http://
developer.apple.com/documentation/AppleScript/Conceptual/soapXMLRPC/",
[samp.icon.url]:"file:///Users/jdsant/Documents/IAA/MaterialesTesis/SAMP/
%20tests/SEScriptEditorX.png"}

if samp_hub_xmlrpc_url is not equal to "" then
    tell application "http://192.168.2.1:56483/xmlrpc"
        -- AppleScript cannot use a variable to set the
        -- XML-RPC destination, we need to set it by hand

        -- Perform the registration call, passing the .samp file's
        -- samp_secret; register_map is the result, and contains
        -- the shared hub-application secret key
        set register_map to call xmlrpc {method name:"samp.hub.register", parameters:
            {samp_secret}}

-- retrieve the private app-hub key
set samp_client_key to [samp.private-key] of register_map

-- declare metadata to the hub (nothing returned)
call xmlrpc {method name:"samp.hub.declareMetadata", parameters:
    {samp_client_key, client_metadata}}

-- obtain a list of registered clients
set registered_clients_list to call xmlrpc {method
    name:"samp.hub.getRegisteredClients", parameters:
    {samp_client_key}}

-- We do not need to loop through client to get those supporting an mtype
-- However, this call would have returned the list of
-- subscribers to the table.load.votable MType
set table_votable_load_subscribers to call xmlrpc {method
    name:"samp.hub.getSubscribedClients", parameters:
    {samp_client_key, "table.load.votable"}}

-- Prepare a message map to be broadcasted
set messageMap to {[samp.mtype]:"table.load.votable", [samp.params]:{[
    url]:"file:///Users/jdsant/CVS/devel/astrogrid/desktop/api/examples/
workflows/sample-conesearch-input_veron.vot", name:"ASS Veron
Ceti", [table-id]:"AS_SAMP#1"}}}

-- Broadcast the prepared message map: the result is that
-- all applications able to load the VOTable would read
-- that file
set notified_ids to call xmlrpc {method name:"samp.hub.notifyAll", parameters:
    {samp_client_key, messageMap}}

repeat 10000000 times
    get notified_ids
end repeat

-- We do a VERY LONG loop in order to be able to see
-- this script as a registered application
-- Unregister with the hub, so that we not leave a zombie
-- application
call xmlrpc {method name:"samp.hub.unregister", parameters:
    {samp_client_key}}

set all_results to {registered_clients_list, table_votable_load_subscribers,
    notified_ids}
get all_results
end tell
end if

```

Figure 13.8: Example AppleScript-based SAMP application, which shows XML-RPC calls for the XML-RPC profile of the SAMP protocol. AppleScript keywords are in blue, while variable names are set in green, and string literals in black. Comments are preceded by two dashes (--). Source code available at: <http://www.iaa.es/~jdsant/thesis/SAMP.applescript>

for setting named parameters, as SAMP does. It is also interesting to compare figures 13.8 and 13.1, in order to appreciate the much higher level of abstraction provided by the JSAMP library.

Figure 13.9 is a screen capture of the SAMP Hub Monitor screen, showing the AS_SAMP registered with the hub and selected, in order to see AS_SAMP metadata.

13.5 Bringing massa into the VO

The modules described in the previous sections have been implemented in order to be able to bring VO utilities to legacy applications without access to application source code. Apart from the facilities already implemented by those modules, the VO Data Converter module—see section 13.2—illustrates how to launch command-line applications after having received a SAMP message. We can, therefore, create SAMP wrappers for different VO messages. We can easily imagine that a mixture between the VO Downloader (to retrieve the remote data, and to launch a command-line application), and

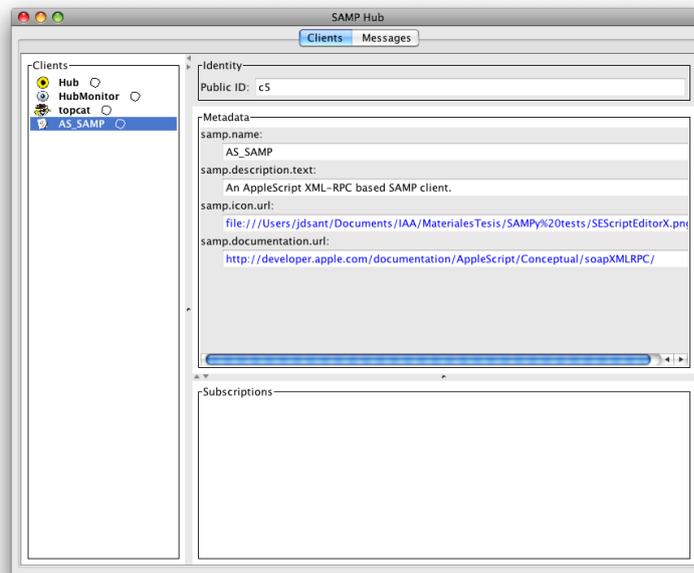


Figure 13.9: Screen capture of the SAMP Hub Monitor showing the active applications, and the metadata stored in the hub regarding the AS_SAMP application.

the VO Data Converter (to convert it into a format suitable to the application), can enable the VO-awareness of existing applications, by launching it with the received, copied, and possibly converted file as a parameter⁷.

In the case of *massa*, being a spectral analysis and transformation application, the modules should provide support for the `table.load.votable`, `table.load.fits` —for spectra expressed as tables—, as some of them are, and the more specific `spectrum.load.ssa-generic`.

However, if there is access to the source code, as is the case with *massa*, it is much more advisable to include SAMP support in the application itself.

The source code for both *massa* and *madcuba*⁸ is composed of 724 Java classes, and there are 159 additional files shared between HTML documentation, JPG images, PNG icons, and test `.fits` and `.30m` files. By concentrating in using SAMP in order to enable VO compatibility, we just need to touch a

⁷Besides, several operating systems, such as FreeBSD or Mac OS X, provide facilities for detecting changes in folder content, so that a script can be launched upon change, which opens the possibility of creating SAMP messages in order to send data created in particular folders to applications of interest.

⁸*madcuba* is a data cube viewer integrated in the same codebase as *massa*, so that *massa* can build, via a regridding algorithm, datacubes from irregularly sampled On-The-Fly observations to be later visualised with *madcuba*.

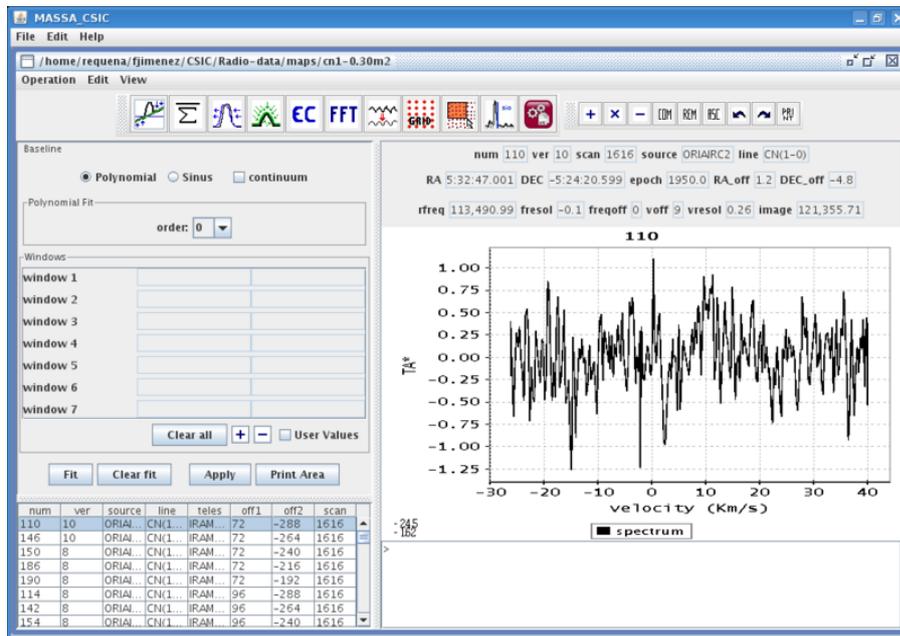


Figure 13.10: Screen capture of massa.

few of them. Figure 13.10 shows a screen capture of massa, with one spectrum selected among a set of observations.

In particular, and recalling both section 10.4, and the introduction of SAMP capabilities into the Java-based VO Downloader, we can see that the candidate classes to be changed are just a few.

In particular, we need to:

- Include SAMP initialisation after GUI initialisation.
- Include SAMP shutdown at the first stages of GUI shutdown.
- Create *.load.* handlers which make use of existing data creation mechanisms; in some cases (VOTables) they will need to change format.
- Create additional menu options for sending massa spectra to other VO applications, including the creation of VO metadata from the massa internal model.

Initialising SAMP

In massa, the GUI is initialised by the `initGUI()` method in the `MainApp` class, which holds the `main()` function for massa.

As `initGUI()` is called in the `MainApp` class creator, we will call our `initSamp()` method right afterwards, residing in the same `MainApp` class.

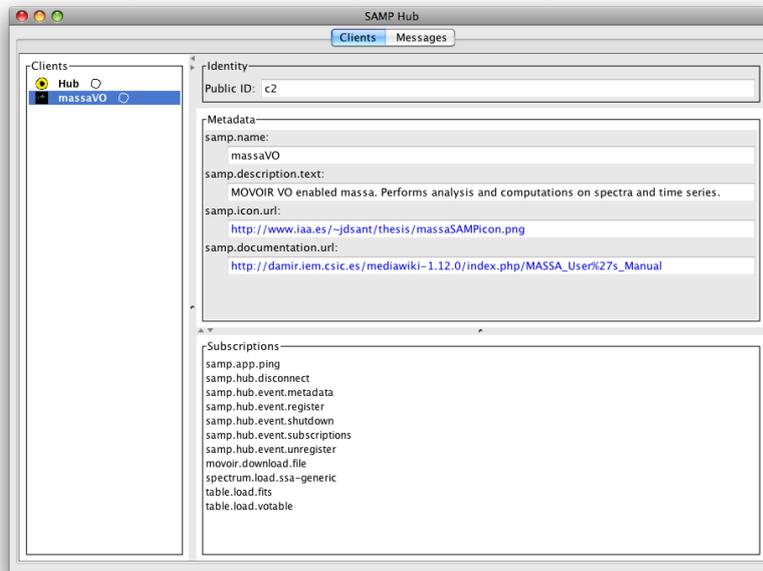


Figure 13.11: massa registered with the JSAMP Hub

In it, we will call a single handler functions for the different `*.load.*` methods. That function will call different procedures depending on the actual `MType` being received.

Figure 13.11 shows a screen capture of the JSAMP HubRunner with `massa` registered with the hub, metadata declared, and message handlers installed.

Shutting down SAMP

To shut SAMP down, we will choose the exit handler for `massa` as the place to install a `shutdownSamp()` method. By using the `JFrame` class as the base class for `MainApp`, setting up auto-quit on close, and having no other quit mechanism, the place to install the `shutdownSamp()` method is right after having confirmed that indeed we wish to close `massa`.

That way, `massa` will always correctly unregister with the hub on quit.

Once chosen the right place to install the `shutdownSamp()` method, the method itself is quite simple, and is identical to the one used for the VO Downloader.

Creating data load handlers

The simplest data load handler to be created in `initSamp()` could just use `massa`'s `Loader` class, bypassing the file chooser, and providing directly a path to the `loadFile` method in that class.

By doing that, however, one of the file types which can be understood by *massa*'s `SpectraLoaderFactory` class must be provided. Hence, for `table.load.fits` messages, if the URL sent corresponds to one of the FITS file formats understood by *massa*, that would be the most straightforward way to implement it.

For `table.load.votable` messages, either a linked FITS file can be passed to the already mentioned `loadFile` method, or the `VOTable` can be converted to FITS using a Java library such as `STIL` (the software library which is the foundation for the `stilts` command), and later to the internal model of the application. For `spectrum.load.ssa-generic` messages the URL parameter would point to either an XML serialisation of the spectrum or, most likely, to a FITS file, making this approach the easiest.

However, and specially for the recently incorporated `spectrum.load.ssa-generic` message, this would completely ignore the associated meta keyword, which corresponds to a map of `Spectrum` data model `UTypes` as keys.

The alternative is to modify the `SpectraLoaderFactory`, so that it is provided with a special `VOTable`, created on the fly by the `*.load.*` method handlers, and which handles by itself the creation and configuration of a FITS file containing all metadata provided in the original message.

In this way, we just modify one class, the `SpectraLoaderFactory`, create a few more classes for handling each different case, and the creation of internal structures is left untouched, and being able to survive future changes in the application internal data model.

The main issue with this approach is that, by the way *massa* is designed, incoming data would be shown first in one of *massa*'s data selection windows, such as the one displayed in figure 13.12. That is not a problem per se, but it reduces somehow the interactivity with other applications. On the other hand, it allows a first peek to the metadata of the spectra being received (even if it consist of just one spectrum).

Adding SAMP menu options

For *massa*, the easiest way to introduce SAMP data sending menu options would seem to add them to the global menu bar, in an Interop menu, mimicking the way the SAMP interface is built on applications like `TOPCAT`.

This is, however, not advised, because as shown in figure 13.10, and glimpsed in figure 13.12, there are different kinds of *massa* windows, each one with its own menu.

We believe that, in the case of *massa*, it is more sensible to provide selection based, context sensitive pop-up menus to provide options to:

- broadcast the selected spectra as a series of `spectrum.load.ssa-generic`, `table.load.fits`, or `table.load.votable` messages;

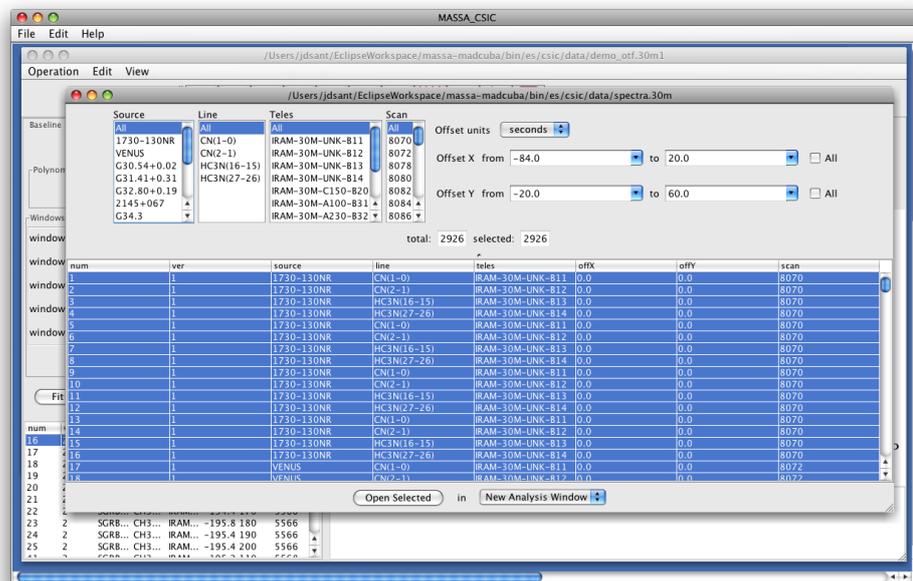


Figure 13.12: massashowing its data selection window, which opens right after having opened a file which contains multiple spectra.

- send the selected spectra to particular applications; applications supporting `spectrum.load.ssa-generic` will be shown first, with applications supporting `table.load.fits` next, and those supporting `table.load.votable` last; the sending method would be chosen accordingly, and data conversions will be performed as needed from the `SimpleSpectraBunch` internal format.

13.6 Conclusions

By developing a SAMP-based approach for the MOVOIR, we have been able to VO-enable `massa` with minimal changes, and at the same time we have provided a mechanism for enhancing not only `massa`, but any other existing SAMP-enabled application which is able to send and receive standard MTypes.

The lowest common denominator degree of VO compatibility is provided by the VO Downloader, which enables other VO applications to send `table.load.votable`, or `image.load.fits` messages, and consolidate those files into a folder where legacy applications can retrieve useful data; and the VO Data Converters, which are capable of providing FITS or CSV files created from VOTables sent to them.

We have also shown, however, than if source code is available, and there are SAMP libraries for the language and platform the legacy application is running on, that the amount of code which has to be modified is minimum: code has to be added to initialise the SAMP messaging, and SAMP event handling, but that is achieved by just adding a single method call in the application initialisation, and the shutdown of SAMP messaging as the first method in the `actionExit` method handler.

The only remaining issue to be solved, which is also limited to the classes performing data loads, is how to perform the adaptation of the FITS or VOTable data files to the internal structure of the application.

In the case of VOTables whose role is to describe a FITS file whose link is included, the data load handlers only need to access that FITS file, and use the regular FITS load capabilities included with the application. Finally, sending data to other applications needs the prerequisite of installing appropriate interface elements, with their corresponding `actionListeners`, which create a temporary VOTable/FITS file from the selected element, or a VOTable with links to FITS files for various elements, and send the appropriate kind of message.

By being so surgical in our approach to bringing VO capabilities to applications, we can ensure that an application which has been upgraded to use SAMP can evolve during time, and be refactored, and most changes in the SAMP-related modules will be restricted to the interface for creating datasets and sending datasets in the application, while the rest of the SAMP support code—SAMP startup and shutdown—will need no changes, unless support for additional messages is required by application evolution.

Conclusions and future work

In this thesis we have performed an extensive and inclusive review of what the Virtual Observatory means from the eyes of current radio astronomers: how does the Virtual Observatory enhance their interaction with astronomical archives, and how data in those archives become interoperable thanks to a unified description; on the other hand, we have seen how the tools they are accustomed to use need to be incorporated into the Virtual Observatory, what different means exist for that, and which one has the less impact on said applications in order to provide VO compatibility.

During that review of available literature, and of IVOA proposed and drafted standards, we have seen that the IVOA had not proposed any complete data model, and many radio astronomical specifics were not taken into account. As a result, no appropriate data model for radio astronomical observations existed.

Thus, a data model has been developed (RADAMS) to support, initially, observations made with single-dish radio telescopes. The RADAMS uses as a basis the IVOA draft proposal for an Observation data model [65], combined with the Characterisation data model [67]. On top of that model the Provenance, Policy, Curation, and Packaging classes have been created and specified, as no actual proposals existed.

Those added classes are completely modular, both in their expression in database form as in their XML serialisation, as they use their own UTypes, and as such they can be safely ignored by applications not able to understand such metadata.

As the most radio astronomy specific part of the RADAMS is found in the Provenance data model, and due to the already mentioned modularity, the RADAMS conforms a complete Observation data model proposal. In the case of the Provenance.Instrument sub-model, we have not only covered single-dish instruments, but also interferometers.

The RADAMS has been validated by being used as the basis for the development of two radio astronomical archives, those for the DSS-63 and IRAM 30m antennas. That development, in turn, serves to back-up the feasibility of using IVOA data models as the basis for the development of new astronomical archives, in general.

These archives have needed a complete proposal of metadata for their XML serialisations, consisting of FITS keywords for the Data-filler, and Unified Content Descriptors (UCDs) for all attributes. Many UCDs have been built by means of juxtaposition of existing ones, so that the resulting UCDs are more specific. A few UCDs have been proposed for addition to the UCD1+ vocabulary.

Complementarily, we have analysed the different options available for bringing legacy astronomical (not necessarily radio specific) tools into the VO, and we have found that by using a messaging protocol based on multi-platform and multi-language intercommunication technologies such as XML-RPC, we can provide VO compatibility and interoperability with minimal effort and code changes, as long as the tools to be updated conform to the Model-View-Controller (MVC) paradigm. A recently developed IVOA protocol, the Simple Application Messaging Protocol (SAMP), conforms to that description, and incorporates VO semantics in the way messages are created, providing a framework for the implementation of the proposed mechanism.

We have validated such analysis, by implementing SAMP, and VO-specific messages, in several applications, and showing their interoperability. In particular, we have implemented SAMP, and MOVOIR messages, into *massa*, the VODownloader, and several other MOVOIR modules.

Finally, the MOVOIR is the first proposal for a plug-in development architecture in order to allow the enhancement of the operation of existing, SAMP compatible, VO applications, without needing access to application source code, and with additional discovery capabilities for MOVOIR-aware applications.

Future work

In the future, I will work with the Archive Management Group of the Archive Department of the Data Management and Operations Division of the European Southern Observatory organisation. The scope of my work will be VO-related archive metadata management. In addition, I will remain a member of the IVOA Data Modelling Working Group, and will also be a member in charge of the development of a radio data cube data model in the framework of the EuroVO-AIDA Technology Forum.

The organisations above will allow me to continue my research on astronomical archives and VO technologies, and I plan to perform the following tasks, also in collaboration with the team at the IAA:

- Submit the different RADAMS modules for approval to the IVOA Data Modelling Working Group, in order to finally issue an Observation Data Model IVOA Recommendation. The RADAMS would serve as a reference implementation of that Observation Data Model, a requisite of the IVOA for their Recommendations.

- Extend the use of semantic web technologies in the ESO as a substitution mechanism for the ESO Hierarchic Keywords.
- Introduce the MOVOIR at the IVOA InterOp meetings, within the Applications Working Group, and present it at the ADASS (Astronomical Data Analysis Software and Systems) conference, compiling user suggestions for new MOVOIR modules.
- Extend and prepare for distribution existing MOVOIR modules.
- Propose the MOVOIR as a reference implementation for an IVOA sanctioned plug-in architecture, in order to allow better integration of MOVOIR modules with plug-in aware applications, and create RADAMS-enabled MOVOIR modules.
- Develop a data quality assessment metric for RADAMS compatible archives, and implement it as a MOVOIR tool.
- Use MOVOIR in order to bring VO capabilities to the GIPSY high-level radio data cube analysis and kinematic modelling package GIPSY al VO, within the AMIGA³ project framework.
- Generalise the RADAMS with support for interferometry, in collaboration with the ALMA Archive Team at ESO, within that same framework.

Conclusiones y trabajo futuro

En esta tesis hemos realizado un repaso extenso e inclusivo de qué significa el Observatorio Virtual (VO) desde el punto de vista de los radioastrónomos de hoy en día: por un lado, hemos visto cómo el VO mejora las capacidades de los archivos que utilizan, y cómo la descripción unificada de dichos datos permite su interoperabilidad; por otro, hemos visto la necesidad de que las herramientas software a las que están acostumbrados los radioastrónomos se adapten al VO, cómo se puede lograr esa integración, y cuál es la forma de menor impacto en dichas aplicaciones de lograr esa integración.

Durante dicho repaso a la literatura disponible, y a los estándares desarrollados por IVOA, hemos visto que no existía ningún modelo de datos sancionado por IVOA apropiado para todas las observaciones astronómicas. Además, los modelos existentes no soportaban las especificidades de las observaciones radioastronómicas.

Por ello, se ha desarrollado un modelo de datos, pensado inicialmente para observaciones con radiotelescopios de antena única (RADAMS), aprovechando de un lado el boceto de modelo de datos de Observaciones [65], y el modelo de datos de Caracterización [67]. Sobre ese modelo se han desarrollado las clases Provenance, Policy, Curation y Packaging, para las que no existían propuestas concretas.

Esos añadidos se realizan de forma modular tanto en su expresión en base de datos como en su serialización XML, de modo que pueden ser ignorados por aplicaciones que no los entiendan.

Puesto que la parte más específicamente radioastronómica del RADAMS se halla en el modelo de datos Provenance (Procedencia, o Linaje), y dado el ya mencionado carácter modular del RADAMS, el resultado es la creación de una propuesta completa de modelo de datos de Observación para IVOA. En el caso del sub-modelo Provenance.Instrument, se ha tenido en cuenta su adaptabilidad para la descripción de interferómetros, y no sólo de radiotelescopios de antena única.

Hemos validado este modelo al implementar dos archivos radioastronómicos basados en él, como son los archivos para las antenas DSS-63 e IRAM 30m. Esa validación es, a su vez, un respaldo a la viabilidad de los modelos de datos de IVOA como herramientas de diseño de archivos astronómicos en general.

Para esta última tarea hemos proporcionado metadatos completos para la creación de las serializaciones XML de los servicios del archivo, consistentes en palabras clave FITS y Unified Content Descriptors (UCDs) para todos los atributos. Se ha usado el mecanismo de yuxtaposición de UCDs para proporcionar UCDs más específicos de los ya existentes. Se ha propuesto también la adición de algunos UCDs al vocabulario UCD1+.

De otra parte, hemos analizado las opciones disponibles para la incorporación de herramientas astronómicas (no específicamente radioastronómicas) en el marco del VO, y hemos encontrado que utilizar un protocolo de mensajería basado en tecnologías multiplataforma y multilenguaje, como XML-RPC, nos permite hacer compatible e interoperable con el VO esas herramientas antiguas con un mínimo de esfuerzo, y de cambios en el código, siempre que esas herramientas sean conformes a un esquema Modelo-Vista-Controlador (MVC).

Hemos validado dicho análisis al implementar SAMP en varias aplicaciones, y haber demostrado su interoperabilidad. En concreto, hemos implementado SAMP, y los mensajes propios de MOVOIR, en las aplicaciones VODownloader, massa, y otros módulos de MOVOIR.

Por último, el MOVOIR es la primera muestra de una posible arquitectura para la creación de módulos de ampliación —*plug-ins*— que permitirían la ampliación de las aplicaciones existentes y que utilicen el protocolo SAMP, sin necesidad ninguna de tener acceso a su código fuente.

Trabajo futuro

En el futuro, voy a trabajar con el Grupo de Gestión de Archivos (Archive Management Group) del Departamento de Archivos (Archive Department) de la División de Gestión de Datos y Operaciones de la ESO (European Southern Observatory, Observatorio Europeo Austral) en la organización de sus archivos, y más concretamente de sus metadatos correspondientes en el marco del IVOA, lo que va a permitir la continuidad del presente trabajo. Además, seguiré siendo miembro del grupo de trabajo de IVOA dedicado a Modelado de Datos (IVOA Data Modelling Working Group), y uno de los técnicos a cargo del desarrollo de un modelo de datos para cubos de datos radio en el marco del proyecto europeo EuroVO-AIDA Technology Forum.

En concreto, espero realizar las siguientes tareas en ese marco, y en colaboración con el grupo AMIGA:

- Someter las diferentes partes del RADAMS para su aprobación dentro del IVOA Data Modelling Working Group, para la creación final de una IVOA Recommendation correspondiente al Modelo de Datos de Observación. Existirá así al menos una implementación de referencia del modelo, un requisito del IVOA para sus estándares.

- Extender el uso de tecnologías de web semántica, propuestas en la forma de extensión de vocabularios, en substitución de mecanismos alternativos, no estándar, como las ESO Hierarchic Keywords.
- Presentar el MOVOIR en los IVOA InterOp, y el ADASS (Astronomical Data Analysis Software and Systems), y obtener sugerencias de los usuarios para crear más módulos para el MOVOIR.
- Extender y dejar listos para distribución los módulos de MOVOIR.
- Proponer MOVOIR como implementación de referencia para una arquitectura de *plug-in* sancionada por IVOA. Si se sanciona una tal arquitectura, se permitiría una mejor integración de los módulos de MOVOIR, mediante el soporte de mensajes adicionales, con aplicaciones que implementen dichos mensajes.
- Desarrollar una métrica objetiva de calidad de datos basados en RADAMS, e implementar un módulo MOVOIR que sea capaz de proporcionarla.
- Usar MOVOIR para la adaptación del paquete de análisis de cubos de datos y modelado cinemático GIPSY al VO, en el marco del proyecto AMIGA³.
- Generalizar el RADAMS para que soporte datos interferométricos, en colaboración con el ALMA Archive Team de la ESO, dentro del mismo marco.

Part V

Appendices

Appendix A

IVOA structure

The International Virtual Observatory Alliance mission and high-level structure was described in section 2.9 of the introduction to the Virtual Observatory.

In this appendix, we provide a detailed overview of the different constituents of the IVOA: Working Groups, Interest Groups, and directive committees.

A.1 Working Groups

The existing IVOA Working Groups¹ (IVOA WGs) are the following:

Applications This WG is devoted to VO application development. Group members have specified several application interoperability protocols (first PLASTIC, the PLatform for ASTrophysical Tool Interoperability and Collaboration; later SAMP, the Simple Application Messaging Protocol), and used them to enable VO data sharing between applications running on the same machine.

Data Access Layer If we view the VO as a superposition of layers which provide services to the layers above them, the Data Access Layer would be the one allowing applications to access data by calling VO services. In particular, this WG is devoted to the development of the data access protocols for the VO. The Simple Cone Search [34] was the first DAL protocol, followed by the SIAP [31] and the SSAP [32] protocols².

Data Modelling This WG deals with the domain-specific data models needed to make astronomical information interoperable, defining the metadata needed for VO queries and automated data mining. The Characterisation data model [100] is the most successful product of this WG.

¹<http://www.ivoa.net/intranet/>

²It should be noticed that the SIAP is a protocol still in the Working Draft stage, and which is being adapted to the common principles used for SSAP.

Grid and Web Services This WG provides the liaison between the Virtual Observatory and the grid, and other forms of distributed computing. In particular, the GWS WG has created a web services wrapper, the Common Execution Architecture [44], for several data mining, processing, and retrieval tasks, so that they can be offloaded to more powerful servers for further processing. The services themselves can be implemented either on standalone machines, or on the grid, transparently to the user.

Resource Registry This WG is devoted to one of the key parts of the VO, the discovery of existing services via VO Registries. VO Registries comply with the Open Archive Initiative Protocol for Metadata Harvesting (OAI-PMH), so that readily available solutions exist for VO registries. Metadata harvesting allows VO registries to share newly registered entries, so that eventually any resource can be found by querying any Registry. This WG also maintains the Registry of Registries (RofR), a list of OAI-PMH endpoints which can be used by harvesting registries to know which registries to harvest, and for client applications to select between available registries.

Semantics This WG is dedicated to the mark-up of information so that data columns can be explicitly identified by their intended meaning, without needing to guess the context.

VO Event This WG provides the foundation for the VO to deal with transient events, such as Gamma-ray Bursts (GRBs), supernovae, lensing of microlensing events, transits, et cetera, so that any VOEvent aware system can decide whether the event is of interest. VOEvent packets [101] carry enough space-time coordinates to allow for automatic driving and aiming of robotics telescopes.

VO Query Language This WG aims to provide a universal query language for all table-oriented access protocols. Current IVOA protocols are target based, while many other astrophysical questions, specially on systems built around relational databases, can be expressed on terms of relational algebra. VOQL is based around SQL, with extensions for spatial regions, object cross-matching, and understanding of VO data models and semantics. The VOQL, also known as the Astronomical Data Query Language (ADQL) was finally standardised in 2008, and is an IVOA Recommendation [102].

VOTable This WG has already accomplished the standardisation of the VO data format. The VOTable, build around an initial XML data format by the CDS [27], is already mature enough, and the WG will stay dormant after the final release of the latest VOTable specification [103].

A.2 Interest Groups

Apart from the Working Groups, several Interest Groups have been formed. The Interest Groups are not part of the key IVOA infrastructure, but some of them are born to study particular needs of the IVOA, to later move them in production, or deciding which particular WG will have the responsibility to move development further. Interest Groups (IGs) could be seen as Working Group incubators. That was the case for the VOEvent interest group, which turned into a full-blown WG after all the initial work. This is a list of active and dormant IGs.

Theory The Theory IG was born to study how different kinds of simulated data could be fit within the VO infrastructure. Currently two kinds of efforts exist in it: a Numerical Simulations effort, concentrated on how to exploit large database-based simulations such as the Millennium Simulation³; and a micro-simulations group, devoted to more specific, parameter-based simulations (stellar models, Initial Mass Function distributions, et cetera). Within this IG the Simple Theoretical Access Protocol (STAP) [104], and the Simple Numerical Access Protocol were drafted.

Data Curation and Preservation This IG group was born recently, and its main aim is to study how to perform the long term, sustainable, preservation, and curation of data within the Virtual Observatory.

Open Grid Forum AstroRG This IG is the liaison of the IVOA with the Open Grid Forum⁴. The GWS WG focuses on delivering web-services-based distributed computing with an unspecified processing backend, while this IG specifically seeks to provide reference grid implementations of VO services.

Radio This IG was the discussion forum for radio astronomy specific provisions within the VO. However, this group is now considered dormant, and radio specific extensions are discussed within the different WGs.

A.3 Steering bodies and committees

Additionally to this WGs and IGs, which provide the core development work, within IVOA there are several steering bodies for its governance:

IVOA Executive Committee The IVOA Executive Committee is the main steering force of the IVOA, and it is formed by the representatives of

³<http://www.mpa-garching.mpg.de/galform/millennium/>

⁴<http://www.ggf.org/>

the different national and international VO projects, together with the IVOA chair and vice-chair.

Technical Coordination Group The TCG is a committee formed by the Chairs of the Working and Interest Groups as well as the IVOA chairs and vice-chairs. It has its own chair, and its role is to ensure proper technical coordination amongst the various IVOA WGs and IGs, as well as being the liaison between them and the IVOA Executive Committee.

Standing Committee on Standards and Processes After having drafted the processes by which IVOA standards were to be defined, the IVOA Standards and Processes WG was disbanded. However, from time to time the need to reform either approved standards or the standardisation process will arise, and in that case the role of the Standing Committee on Standards and Processes is to review and update IVOA processes in response to the needs of the IVOA community.

A.4 Standard definition process

IVOA standard documents start their life as Working Drafts within any of the existing Working Groups. When a consensus has been reached within the Working Group, the Working Draft is elevated to the IVOA Executive as a Proposed Recommendation, which has to be subject to a four weeks review period, or Request For Comments. If no major issues have been raised during the RFC review, or by the IVOA Executive, the Proposed Recommendation turns into an IVOA Recommendation.

IVOA Recommendations can be later evaluated by the IAU VO Working Group in order to become IAU Standards. Currently, the only IVOA Recommendation which is an IAU Standard is the VOTable data format.

IVOA Notes can be written by anyone and submitted to a suitable working group, but there is no formal process for their promotion, other than lobbying in the Working Group for the contents of one or several notes to become a Working Draft, or part of one.

In any of the promotion steps the documents **MUST** incorporate the changes agreed within the RFC period, while other comments, if duly answered, do not need to be incorporated. However, if in any of the stages a substantial revision of the document is needed, the proposal **MUST** go back to the Working Draft stage.

This standard promotion process is illustrated in figure [A.1](#), and is documented by the *IVOA Document Standards* [105].

It should be noted that in the IVOA InterOp meeting of Fall 2008 in Baltimore, Robert Hanisch announced an agreement with the NASA Astronomical Database System, responsible for tracking publications related to astronomy. This agreement included Working Draft as low-impact refereed publications

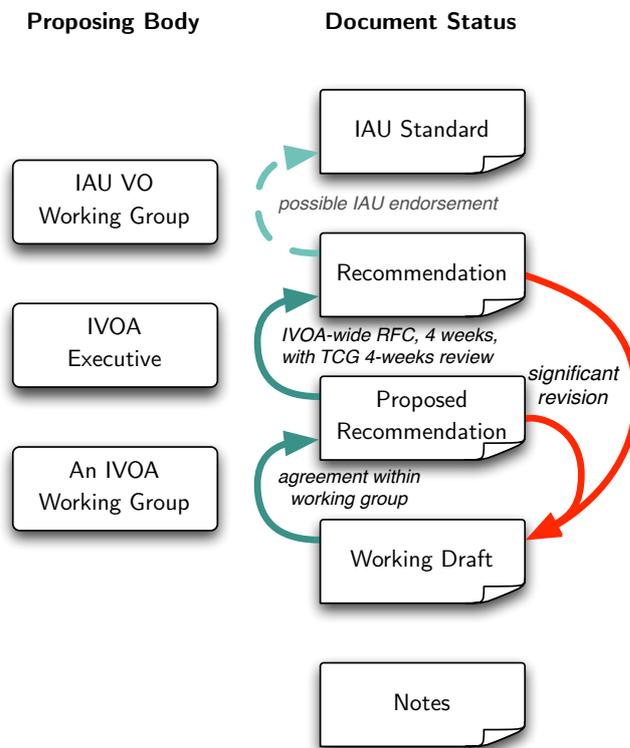


Figure A.1: IVOA Document Process. Standard documents of the IVOA can take the form of Working Drafts, Proposed Recommendations, and Recommendations, while IVOA Notes can be used to publish interesting comments on best practices, and might become part of a working draft if they are deemed interesting enough. IVOA Recommendations are evaluated by the IAU VO Working Group, which might propose an IVOA Recommendations into a IAU Standard.

(as Working Drafts need to pass a peer-review among group members), while Proposed Recommendations and Recommendations will be taking into account as high-impact refereed publications (as they have passed an ever tougher peer-review process, and will create a IVOA Recommendation).

Appendix B

Introduction to XML

XML stands for eXtensible Markup Language. As the name implies, XML is a computer language for the tagging —markup— of data, or more precisely, an specification for writing particular markup languages. XML is Recommendation of the W3C since February 1998, and the latest XML 1.0 specification can be found at the W3C site¹. For people not used to interpreting standard references, a solid reference guide to XML and related specifications is *XML in a nutshell* [106].

The aim of this appendix is presenting astronomers, possibly with some development experience, with an introduction to XML, so that the VOTable appendix can be better understood.

B.1 Data markup

If we find the following set of data:

```
Juan de Dios Santander Vela
23/05/1972
Ingeniero en Electrónica
Instituto de Astrofísica de Andalucía
```

we, as resourceful humans with lots of common sense, might think that the first line corresponds to the full name of a person, the second one is a date (most likely, a birth date), the third one is a degree (in Spanish), and the fourth one is an institution to which said person might belong.

A non-ambiguous way to present the information above could be the following:

```
NAME:          Juan de Dios Santander Vela
BIRTHDATE:     23/05/1972
DEGREE(es_ES): Ingeniero en Electrónica
INSTITUTION:   Instituto de Astrofísica de Andalucía
```

¹<http://www.w3.org/TR/REC-xml/>

That, for instance, is the kind of convention used in FITS header cards [22]: using keywords for particular pieces of data, and putting the data after said keywords.

However, that system brings its own share of problems. For instance, what kind of keywords must be admitted? How many degrees can a person have, or how many institutions can that person belong to? If we wish to provide the same information for a large number of people, how does a computer recognise the end of each record?

B.2 XML markup

In XML, those problems are solved by following these rules:

Tags enclose marked up data A piece of data is marked up by enclosing it between `<TAG>` and `</TAG>`. For instance, a person's name would be a string enclosed between `<NAME>` and `</NAME>`, as in `<NAME>Juan de Dios Santander Vela</NAME>`.

Tags are case sensitive In an XML document, all of the following are different tags: `<NAME>`, `<name>`, `<Name>`. However, it is strongly recommended not to use tags which only differ in their capitalisation.

At the root of an XML document, one and only one tag exists That means that in XML documents, everything is inside a tag. If more tags need to be specified, they must be enclosed in a higher hierarchy tag. With this in mind,

```
<NAME>Juan de Dios Santander Vela</NAME>
<BIRTHDATE>23/05/1972</BIRTHDATE>
```

is not a valid XML document, but

```
<PERSON>
<NAME>Juan de Dios Santander Vela</NAME>
<BIRTHDATE>23/05/1972</BIRTHDATE>
</PERSON>
```

is.

All tags reside completely within other tags This condition means that for closing a tag, all other tags that might have opened inside must be closed. That is, `This is a text marked up with bold and italics ` is not valid XML, as the `` tag opens after the `` tag, but the `` tag is still open when trying to close ``.

The last two conditions are also commonly stated as: *XML documents must be well-formed*. That is, *well-formedness* is the property of an XML

document of having just a single root element, and having elements completely enclosed within other elements.

Additionally, XML tags can hold extra attributes. For instance, the `<DEGREE>` tag can have an `language` attribute which specifies the language and country in which the degree was awarded, i.e., `<DEGREE lang='es_ES'>Ingeniero en Electrónica</DEGREE>`.

With those rules, the information above might be written in XML as:

```
<PERSONS>
<PERSON>
<NAME>Juan de Dios Santander Vela</NAME>
<BIRTHDATE>23/05/1972</BIRTHDATE>
<DEGREE lang="es_ES">Ingeniero en Electrónica</DEGREE>
<INSTITUTION>Instituto de Astrofísica de
Andalucía
</INSTITUTION>
</PERSON>
</PERSONS>
```

An additional `<PERSONS>` tag has been included so that several `<PERSON>` records could be found in the same XML document.

B.3 XML validation

In the example above, we have devised an XML-based markup whose root element is the `<PERSONS>` tag, which can hold inside one or more `<PERSON>` tags, each one containing `<NAME>`, `<BIRTHDATE>`, `<DEGREE>` and `<INSTITUTION>` tags.

However, as there is been no specification of the language, how can a computer make sure that the `<INSTITUTION>` tag belongs in the `<PERSON>` tag? What can be put inside a `<BIRTHDATE>` tag?

In other words, how can we specify a particular XML markup language so that computers can validate particular XML document (instances) against that specification?

As originally XML was a subset of an existing markup language called Standard Generalized Markup Language (SGML), XML documents used the SGML mechanism for markup specification, called Document Type Definition (DTD). DTDs specify `ELEMENTs`, which are the particular tags which can appear in an XML document referencing the DTD, and `ATTLISTs` (attribute lists), which are the sets of attributes that can modify a particular tag. The `ELEMENT` declarations' syntax also allows for specifying which tags can occur, and how many times, inside others.

A DTD specifying the `ELEMENTs` and `ATTLIST` for the example above would be:

```

<!ELEMENT PERSONS (PERSON*)>
<!ELEMENT PERSON (NAME,BIRTHDATE?,DEGREE*,INSTITUTION*)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT BIRTHDATE (#PCDATA)>
<!ELEMENT DEGREE (#PCDATA)>
<!ELEMENT INSTITUTION (#PCDATA)>
<!ATTLIST DEGREE lang CDATA #REQUIRED>

```

The rules for DTDs are:

- `<!ELEMENT tag (tagcontents)>` indicates that `<tag>` tags are allowed, with `tagcontents` inside.
- `tagcontents` is either a single tag, a list of tags separated by commas (,, indicating several tags which may appear in sequence), a list of tags separated by the vertical bar (|, indicating alternative tags which might appear), the keyword ANY (indicating that any tag can occur inside), or the keyword EMPTY (indicating that the `<tag>` tag must have no content, and can be used as `<tag/>`). All the tags appearing inside a `tagcontents` must have a corresponding `<!ELEMENT tag>` entry.
- The number of times a tag specified in `tagcontents` can appear depends on certain modifier characters after the tag. The asterisk (*) indicates that a tag can appear any number of times, including none at all. The question mark (?) indicates that the tag is optional, and can appear once or none at all. A plus sign (+) indicates that the tag must appear at least once, but can appear many more times. If no symbol is specified, the tag appears just one time.
- `<!ATTLIST tag attrName attrType defaultValue>` specifies that the `<tag>` tag can have an attribute `attrName`. `attrType` is one of: CDATA (for strings); `(a|b|...|n)` (for an attribute which can be any one of a to n); ID (for a unique identifier); IDREF (for a reference to an ID of a different element); IDREFS (for a space-separated list of IDs to different elements); and additional specifiers for entities and entity lists (entities are symbols representing particular characters or strings, such as `&`; representing the ampersand (&)), shared controlled vocabularies (for attributes with the same list of possible values). And `defaultValue` is either "value", indicating the `attrName` can be set to any value, but if not value is applied by default; #IMPLIED for attributes which do not need to be set, and for which no default value is assumed; #REQUIRED for attributes which must be set with any value; or #FIXED "value" for attributes which must be set exactly to value.

If we take into account the rules above, our DTD states that:

1. The <PERSONS> tag contains any number of <PERSON> tags, but <PERSON> tags alone. The root <PERSONS> tag must exist. (tagname* denotes any number, including none, of <tagname> tags, while tagname alone means one and only one <tagname> tag appearance.)
2. The <PERSON> tag contains, in order, one <NAME> tag, and optionally a <BIRTHDATE>, any number of <DEGREE> tags, and any number of <INSTITUTION> tags.
3. Inside the <NAME>, <BIRTHDATE>, <DEGREE> and <INSTITUTION> tags (that is, enclosed between <tag> and </tag>), the content can be any string (#PCDATA).
4. The <DEGREE> tag has a required attribute, lang, which is a string (CDATA).

A valid XML document is one which is not only well-formed, but which also follows the prescriptions of the DTD regarding allowed tags, allowed tag content and cardinality restrictions (the number of times a tag can appear), as well as restrictions on allowed attributes.

B.4 XML semantics

To this moment, we have considered the validity of XML documents with regard to certain automatic checks to be performed regarding the contents of several tasks. But what about the semantics of XML documents?

Semantics, regarding data markup, is precisely the relationship between a marked-up datum and an existing object (real or virtual) for which a particular property is being stated. The relationship must be one-to-one from the data markup and the property of interest.

We can see that for particular XML languages, specified by a DTD, the <!ELEMENT elementName> tags introduce a hierarchy of elements for which data (or metadata) have to be provided, whereas <!ATTLIST attribName> indicate additional metadata for elements.

That way, DTDs establish a data model for describing data in a particular scope. In the case of astronomical data interchange, XML documents are called VOTables. Initially, VOTables were XML documents conforming to the VOTable DTD (see section C.1 in appendix C).

However, DTDs do not impose strong restrictions on particular datum a tag can hold, other than enumerating *a priori* all possible values. However, if a datum must be an URI (as in a href attribute within a <LINK> tag), there are practical restrictions on what can constitute a valid URI and what not, but they cannot be described in a DTD. That's what the XML Schema language is for: defining data types which carry their own semantics, and expressing tighter restrictions which help in defining such semantics.

B.5 The XML Schema language

The XML Schema language is a set of XML keywords which can be used to express everything a DTD can express, in terms of data modelling, but with many additional features:

Better defined restrictions DTDs can only express the restrictions on data exposed on section B.3. However, many times additional restrictions must be imposed on data, such as ensuring certain contents conforms to given regular expressions.

Data types In computer languages, data types are very much akin to physical units, in the sense that variables of a particular data type can only hold, be compared to, or added to values of that same (or compatible) data types. XML Schema can be used to create data types, and then assign those data types (which by themselves impose restrictions on data) to elements (or attributes) which must conform to those data types. This aids both in better reuse of datatypes across the schema, and a much more modular design.

Namespaces Namespaces are an XML feature quite related to semantics and modularity. Namespaces allow the usage of tags from different XML Schemata in the same XML document, so that the particular meaning for a tag within a given schema is preserved, together with the restrictions and relationships set by each schema.

XML Schema is expressed in XML This means that the file expressing the restrictions/data types for XML documents of the type conforming to said schema is, by itself, an XML document, so that XML tools can be used for verifying the schema validity, or for using the XQuery language to find usages of a data type, or the XSLT language for creating a DTD, or any other kind of representation, from the Schema document. However, that also means that XML Schemas tend to be much more verbose than DTDs for the same file type.

We will not enter in detail into the XML Schema language, as it is outside the scope of this appendix. The official W3C specification can be found at <http://www.w3.org/TR/xmlschema-1/>.

Appendix C

VOTable format definition

The VOTable is the main data interchange standard existing in the VO, and is an IVOA Recommendation since 2004. Its formal definition can be found in [103], or in the IVOA Documents site¹. However, we will provide in this appendix a brief description of the VOTable format which also aims to illustrate some parts of that document.

C.1 VOTable DTD

The initial definition of the VOTable, version 1.0, was made in terms of an XML Document Type Definition (DTD), which can be found in listing C.1. This document specifies ELEMENTS, which are the tags that can be used to markup data, and which tags can be included, and how many times, within any other tag; and ATTLISTS, or attribute lists, which are the attributes that can further refine a tag being applied to a piece of data.

C.2 VOTable XML Schema Definition

A new version of the VOTable, version 1.1, was a little bit more restrictive on what is an acceptable VOTable, by means of an XML Schema Definition (XSD). The XML schema can be seen in listing C.4.

The additional restrictions over the VOTable 1.0 standard (those defined by the DTD) are, among others:

- `ucdType` validates against UCD1 or UCD1+ Unified Content descriptors, allowing the rejection of VOTables with UCDs using not allowed characters.
- `astroYear` validates against Besselian or Julian years' specifications (e.g., J2000, B1950, but not 2000 or F2000).

¹<http://www.ivoa.net/Documents/latest/VOT.html>

Listing C.1: VOTable 1.0 Document Type Definition.

```

<!-- DOCUMENT TYPE DEFINITION for VOTable = Virtual Observatory Tabular Format
See History at      http://vizier.u-strasbg.fr/doc/VOTable
See Discussions at  http://archives.us-vo.org/VOTable
Reference DTD as    http://us-vo.org/xml/VOTable.dtd
                   or at http://cdsweb.u-strasbg.fr/xml/VOTable.dtd
XML Schema at      http://us-vo.org/xml/VOTable.xsd
                   or at http://cdsweb.u-strasbg.fr/xml/VOTable.xsd
.Version 1.0 : 15-Apr-2002
-->

<!-- VOTABLE is the root element -->
<!ELEMENT VOTABLE (DESCRIPTION?, DEFINITIONS?, INFO*, RESOURCE*)>
<!ATTLIST VOTABLE
    ID ID #IMPLIED
    version CDATA #IMPLIED
>

<!-- RESOURCES can contain other RESOURCES,
together with TABLES and other stuff -->
<!ELEMENT RESOURCE (DESCRIPTION?, INFO*, COOSYS*, PARAM*, LINK*,
    TABLE*, RESOURCE*)>
<!ATTLIST RESOURCE
    name CDATA #IMPLIED
    ID ID #IMPLIED
    type (results | meta) "results"
>

<!ELEMENT DESCRIPTION (#PCDATA)>
<!ELEMENT DEFINITIONS (COOSYS?, PARAM?)*>

<!-- INFO is a name-value pair -->
<!ELEMENT INFO (#PCDATA)>
<!ATTLIST INFO
    ID ID #IMPLIED
    name CDATA #IMPLIED
    value CDATA #IMPLIED
>

<!-- A PARAM is similar to a FIELD, but it also has a "value" attribute -->
<!ELEMENT PARAM (DESCRIPTION?, VALUES?, LINK*)>
<!ATTLIST PARAM
    ID ID #IMPLIED
    unit CDATA #IMPLIED
    datatype (boolean | bit | unsignedByte | short | int | long | char
    | unicodeChar | float | double | floatComplex | doubleComplex) #IMPLIED
    precision CDATA #IMPLIED
    width CDATA #IMPLIED
    ref IDREF #IMPLIED
    name CDATA #IMPLIED
    ucd CDATA #IMPLIED
    value CDATA #IMPLIED
    arraysize CDATA #IMPLIED
>

<!-- A TABLE is a sequence of FIELDS and LINKS and DESCRIPTION,
possibly followed by a DATA section -->
<!-- ELEMENT TABLE (DESCRIPTION?, LINK*, FIELD*, DATA?) -->
<!ELEMENT TABLE (DESCRIPTION?, FIELD*, LINK*, DATA*)>
<!ATTLIST TABLE
    ID ID #IMPLIED
    name CDATA #IMPLIED
    ref IDREF #IMPLIED
>

```

Listing C.2: VOTable 1.0 DTD: continued

```

<!-- FIELD is the definition of what is in a column of the table -->
<!-- A field may have 2 sets of VALUES: "legfal" and "actual" -->
<!ELEMENT FIELD (DESCRIPTION?, VALUES*, LINK*)>
<!ATTLIST FIELD
  ID ID #IMPLIED
  unit CDATA #IMPLIED
  datatype (boolean | bit | unsignedByte | short | int | long | char
    | unicodeChar | float | double | floatComplex | doubleComplex) #IMPLIED
  precision CDATA #IMPLIED
  width CDATA #IMPLIED
  ref IDREF #IMPLIED
  name CDATA #IMPLIED
  ucd CDATA #IMPLIED
  arraysize CDATA #IMPLIED
  type (hidden | no_query | trigger) #IMPLIED
>

<!-- VALUES expresses the values that can be taken by the data in a column. -->
<!ELEMENT VALUES (MIN?, MAX?, OPTION*)>
<!ATTLIST VALUES
  ID ID #IMPLIED
  type (legal | actual) "legal"
  null CDATA #IMPLIED
  invalid (yes | no) "no"
>
<!ELEMENT MIN (#PCDATA)>
<!ATTLIST MIN
  value CDATA #REQUIRED
  inclusive (yes | no) "yes"
>
<!ELEMENT MAX (#PCDATA)>
<!ATTLIST MAX
  value CDATA #REQUIRED
  inclusive (yes | no) "yes"
>
<!ELEMENT OPTION (OPTION*)>
<!ATTLIST OPTION
  name CDATA #IMPLIED
  value CDATA #REQUIRED
>

<!-- The link is a URL (href) or some other kind of reference (gref). -->
<!ELEMENT LINK (#PCDATA)>
<!ATTLIST LINK
  ID ID #IMPLIED
  content-role (query | hints | doc) #IMPLIED
  content-type CDATA #IMPLIED
  title CDATA #IMPLIED
  value CDATA #IMPLIED
  href CDATA #IMPLIED
  gref CDATA #IMPLIED
  action CDATA #IMPLIED
>

<!-- DATA is the actual table data, in one of three formats -->
<!ELEMENT DATA (TABLEDATA | BINARY | FITS)>

<!-- Pure XML data -->
<!ELEMENT TABLEDATA (TR*)>
<!ELEMENT TR (TD+)>
<!ELEMENT TD (#PCDATA)>
<!ATTLIST TD
  ref IDREF #IMPLIED
>

```

Listing C.3: VOTable 1.0 DTD: continued

```

<!-- FITS file, perhaps with specification of which extension to seek to -->
<!ELEMENT FITS (STREAM)>
<!ATTLIST FITS
    extnum CDATA #IMPLIED
>

<!-- Binary data format -->
<!ELEMENT BINARY (STREAM)>

<!-- Stream can be local or remote, encoded or not -->
<!ELEMENT STREAM (#PCDATA)>
<!ATTLIST STREAM
    type (locator | other) "locator"
    href CDATA #IMPLIED
    actuate (onLoad | onRequest | other | none) "onRequest"
    encoding (gzip | base64 | dynamic | none) "none"
    expires CDATA #IMPLIED
    rights CDATA #IMPLIED
>

<!-- Expresses the coordinate system we are using -->
<!ELEMENT COOSYS (#PCDATA)>
<!ATTLIST COOSYS
    ID ID #IMPLIED
    equinox CDATA #IMPLIED
    epoch CDATA #IMPLIED
    system (eq_FK4 | eq_FK5 | ICRS | ecl_FK4 | ecl_FK5 | galactic
        | supergalactic | xy | barycentric | geo_app) "eq_FK5"
>

```

- `arrayType` validates against the specification of fixed or variable array sizes (e.g., `12x23x*`, but not `12 by 23`).
- `precType` validates against the syntax for precision specification (e.g., `F10`, `E5` or `5`, but not `12F`).

And the use of XML Schema native data types allows an easier specification of data bounds, such as `xs:nonNegativeInteger`, or of date and time time stamps with `xs:dateTime`.

C.3 VOTable structure

Independently of whether a VOTable XML document specifies its document type via a DTD or the VOTable XML Schema, its structure can be seen in figure C.1.

A VOTable, then, is an XML document whose root element is the `<VOTABLE>` tag, and which might hold inside a `<DESCRIPTION>` tag, a `<DEFINITIONS>` tag², and a `<COOSYS>` tag. More importantly, a VOTable can contain several `<RESOURCE>` tags, each of one could contain any number of `<TABLE>` and/or `<RESOURCE>` tags. Figure C.3 shows the structure of a `<RESOURCE>` tag, while figure C.4 shows the structure of a `<TABLE>` tag.

²This tag is deprecated in the version 1.1 of the VOTable definition, which has made the `<COOSYS>`, `<PARAM>` and `<INFO>` tags first-class citizens under the `<VOTABLE>` tag.

Listing C.4: VOTable XML Schema. Notice the version numbering from the first schema definition, compatible with the DTD version, to the latest XML Schema.

```

<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema for VOTable = Virtual Observatory Tabular Format
.Version 1.0 : 15-Apr-2002
.Version 1.09: 23-Jan-2004 Version 1.09
.Version 1.09: 30-Jan-2004 Version 1.091
.Version 1.09: 22-Mar-2004 Version 1.092
.Version 1.094: 02-Jun-2004 GROUP does not contain FIELD
.Version 1.1 : 10-Jun-2004 remove the complexContent
-->
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://www.ivoa.net/xml/VOTable/v1.1"
  xmlns="http://www.ivoa.net/xml/VOTable/v1.1"
>

<!-- Here we define some interesting new datatypes:
- anyTEXT may have embedded XHTML (conforming HTML)
- astroYear is an epoch in Besselian or Julian year, e.g. J2000
- arrayDEF specifies an array size e.g. 12x23x*
- dataType defines the acceptable datatypes
- ucdType defines the acceptable UCDS (UCD1+)
- precType defines the acceptable precisions
- yesno defines just the 2 alternatives
-->

<xs:complexType name="anyTEXT" mixed="true">
  <xs:sequence>
    <xs:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="astroYear">
  <xs:restriction base="xs:token">
    <xs:pattern value="[JB]?[0-9]+([\.[0-9]*)?"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ucdType">
  <xs:restriction base="xs:token">
    <xs:pattern value="[A-Za-z0-9_.;\-\|]*"/><!-- UCD1 use also / + % -->
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="arrayDEF">
  <xs:restriction base="xs:token">
    <xs:pattern value="([0-9]+x)*[0-9]*[*]?(s\W)?"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="encodingType">
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="gzip"/>
    <xs:enumeration value="base64"/>
    <xs:enumeration value="dynamic"/>
    <xs:enumeration value="none"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="dataType">
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="boolean"/>
    <xs:enumeration value="bit"/>
    <xs:enumeration value="unsignedByte"/>
    <xs:enumeration value="short"/>
    <xs:enumeration value="int"/>
    <xs:enumeration value="long"/>
    <xs:enumeration value="char"/>
    <xs:enumeration value="unicodeChar"/>
    <xs:enumeration value="float"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="floatComplex"/>
    <xs:enumeration value="doubleComplex"/>
  </xs:restriction>
</xs:simpleType>

```

Listing C.5: VOTable XML Schema: cont.

```

<xs:simpleType name="precType">
  <xs:restriction base="xs:token">
    <xs:pattern value="[EF]?[1-9][0-9]*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="yesno">
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="yes"/>
    <xs:enumeration value="no"/>
  </xs:restriction>
</xs:simpleType>

<!-- VOTable is the root element -->
<xs:element name="VOTABLE">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="DESCRIPTION" minOccurs="0"/>
      <xs:element ref="DEFINITIONS" minOccurs="0"/><!-- Deprecated -->
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="COOSYS" />
        <xs:element ref="PARAM" />
        <xs:element ref="INFO" />
      </xs:choice>
      <xs:element ref="RESOURCE" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:ID"/>
    <xs:attribute name="version">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="1.1"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<!-- RESOURCES can contain DESCRIPTION, (INFO/PARAM/COSYS), LINK, TABLEs -->
<xs:element name="RESOURCE">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="DESCRIPTION" minOccurs="0"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="INFO" />
        <xs:element ref="COOSYS" />
        <xs:element ref="PARAM" />
      </xs:choice>
      <xs:element ref="LINK" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="TABLE" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="RESOURCE" minOccurs="0" maxOccurs="unbounded"/>
      <!-- Suggested Doug Tody, to include new RESOURCE types -->
      <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:token"/>
    <xs:attribute name="ID" type="xs:ID"/>
    <xs:attribute name="utype" type="xs:string"/>
    <xs:attribute name="type" default="results">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="results"/>
          <xs:enumeration value="meta"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <!-- Suggested Doug Tody, to include new RESOURCE attributes -->
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
</xs:element>

```

Listing C.6: VOTable XML Schema: cont.

```

<xs:element name="DESCRIPTION" type="anyTEXT" />

<xs:element name="DEFINITIONS">
  <xs:annotation>
    <xs:documentation>Deprecated in Version 1.1</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="COOSYS" />
      <xs:element ref="PARAM" />
    </xs:choice>
  </xs:complexType>
</xs:element>

<!-- INFO is a name-value pair -->
<xs:element name="INFO">
  <xs:complexType><xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="ID" type="xs:ID"/>
      <xs:attribute name="name" type="xs:token" use="required"/>
      <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:extension>
  </xs:simpleContent></xs:complexType>
</xs:element>

<!-- A PARAM is similar to a FIELD, but it also has a "value" attribute -->
<xs:element name="PARAM">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="DESCRIPTION" minOccurs="0"/>
      <xs:element ref="VALUES" minOccurs="0"/>
      <xs:element ref="LINK" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:ID"/>
    <xs:attribute name="unit" type="xs:token"/>
    <xs:attribute name="datatype" type="dataType" use="required"/>
    <xs:attribute name="precision" type="precType"/>
    <xs:attribute name="width" type="xs:positiveInteger"/>
    <xs:attribute name="ref" type="xs:IDREF"/>
    <xs:attribute name="name" type="xs:token" use="required"/>
    <xs:attribute name="ucd" type="ucdType"/>
    <xs:attribute name="utype" type="xs:string"/>
    <xs:attribute name="value" type="xs:string" use="required"/>
    <xs:attribute name="arraysize" type="arrayDEF"/>
  </xs:complexType>
</xs:element>

<!-- A TABLE is a sequence of FIELD/PARAMs and LINKS and DESCRIPTION,
possibly followed by a DATA section
-->
<xs:element name="TABLE">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="DESCRIPTION" minOccurs="0"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="FIELD" />
        <xs:element ref="PARAM" />
        <xs:element ref="GROUP" />
      </xs:choice>
      <xs:element ref="LINK" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="DATA" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:ID"/>
    <xs:attribute name="name" type="xs:token"/>
    <xs:attribute name="ref" type="xs:IDREF"/>
    <xs:attribute name="ucd" type="ucdType"/>
    <xs:attribute name="utype" type="xs:string"/>
    <xs:attribute name="nrows" type="xs:nonNegativeInteger"/>
  </xs:complexType>
</xs:element>

```

Listing C.7: VOTable XML Schema: cont.

```

<!-- FIELD is the definition of what is in a column of the table -->
<xs:element name="FIELD">
  <xs:complexType>
    <xs:sequence> <!-- minOccurs="0" maxOccurs="unbounded" -->
      <xs:element ref="DESCRIPTION" minOccurs="0"/>
      <xs:element ref="VALUES" minOccurs="0"/> <!-- maxOccurs="2" -->
      <xs:element ref="LINK" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:ID"/>
    <xs:attribute name="unit" type="xs:token"/>
    <xs:attribute name="datatype" type="dataType" use="required"/>
    <xs:attribute name="precision" type="precType"/>
    <xs:attribute name="width" type="xs:positiveInteger"/>
    <xs:attribute name="ref" type="xs:IDREF"/>
    <xs:attribute name="name" type="xs:token" use="required"/>
    <xs:attribute name="ucd" type="ucdType"/>
    <xs:attribute name="utype" type="xs:string"/>
    <xs:attribute name="arraysize" type="xs:string"/>
    <xs:attribute name="type">
      <!-- type is not in the Version 1.1, but is kept for
           backward compatibility purposes
      -->
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="hidden"/>
          <xs:enumeration value="no_query"/>
          <xs:enumeration value="trigger"/>
          <xs:enumeration value="location"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<!-- GROUP groups columns; may include descriptions, fields/params/groups -->
<xs:element name="GROUP">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="DESCRIPTION" minOccurs="0"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="FIELDref"/>
        <xs:element ref="PARAMref"/>
        <xs:element ref="PARAM"/>
        <xs:element ref="GROUP"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:ID"/>
    <xs:attribute name="name" type="xs:token"/>
    <xs:attribute name="ref" type="xs:IDREF"/>
    <xs:attribute name="ucd" type="ucdType"/>
    <xs:attribute name="utype" type="xs:string"/>
  </xs:complexType>
</xs:element>

<!-- FIELDref and PARAMref are references to FIELD or PARAM defined
in the parent TABLE or RESOURCE -->
<xs:element name="FIELDref">
  <xs:complexType>
    <xs:attribute name="ref" type="xs:IDREF" use="required"/>
    <!-- utype and maybe ucd could well be added there,
         will be if necessary -->
  </xs:complexType>
</xs:element>
<xs:element name="PARAMref">
  <xs:complexType>
    <xs:attribute name="ref" type="xs:IDREF" use="required"/>
    <!-- utype and maybe ucd could well be added there,
         will be if necessary -->
  </xs:complexType>
</xs:element>

```

Listing C.8: VOTable XML Schema: cont.

```

<!-- VALUES expresses the values that can be taken by the data
      in a column or by a parameter
-->
<xs:element name="VALUES">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="MIN" minOccurs="0"/>
      <xs:element ref="MAX" minOccurs="0"/>
      <xs:element ref="OPTION" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:ID"/>
    <xs:attribute name="type" default="legal">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="legal"/>
          <xs:enumeration value="actual"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="null" type="xs:token"/>
    <xs:attribute name="ref" type="xs:IDREF"/>
    <!-- xs:attribute name="invalid" type="yesno" default="no" -->
  </xs:complexType>
</xs:element>
<xs:element name="MIN">
  <xs:complexType>
    <xs:attribute name="value" type="xs:string" use="required"/>
    <xs:attribute name="inclusive" type="yesno" default="yes"/>
  </xs:complexType>
</xs:element>
<xs:element name="MAX">
  <xs:complexType>
    <xs:attribute name="value" type="xs:string" use="required"/>
    <xs:attribute name="inclusive" type="yesno" default="yes"/>
  </xs:complexType>
</xs:element>
<xs:element name="OPTION">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="OPTION" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:token"/>
    <xs:attribute name="value" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
<!-- The LINK is a URL (href) or some other kind of reference (gref) -->
<xs:element name="LINK">
  <xs:complexType mixed="true">
    <xs:attribute name="ID" type="xs:ID"/>
    <xs:attribute name="content-role">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="query"/>
          <xs:enumeration value="hints"/>
          <xs:enumeration value="doc"/>
          <xs:enumeration value="location"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="content-type" type="xs:token"/>
    <xs:attribute name="title" type="xs:string"/>
    <xs:attribute name="value" type="xs:string"/>
    <xs:attribute name="href" type="xs:anyURI"/>
    <xs:attribute name="gref" type="xs:token"/><!-- Deprecated in V1.1 -->
    <xs:attribute name="action" type="xs:anyURI"/>
  </xs:complexType>
</xs:element>
<!-- DATA is the actual table data, in one of three formats -->
<xs:element name="DATA">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="TABLEDATA"/>
      <xs:element ref="BINARY"/>
      <xs:element ref="FITS"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

```

Listing C.9: VOTable XML Schema: cont.

```

<!-- Pure XML data -->
<xs:element name="TABLEDATA">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="TR" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="TD">
  <xs:complexType><xs:simpleContent>
    <xs:extension base="xs:string">
      <!-- xs:attribute name="ref" type="xs:IDREF" -->
      <xs:attribute name="encoding" type="encodingType"/>
    </xs:extension>
  </xs:simpleContent></xs:complexType>
</xs:element>

<xs:element name="TR">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="TD" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- FITS file, perhaps with specification of which extension to seek to -->
<xs:element name="FITS">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="STREAM"/>
    </xs:sequence>
    <xs:attribute name="extnum" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>

<!-- BINARY data format -->
<xs:element name="BINARY">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="STREAM"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- STREAM can be local or remote, encoded or not -->
<xs:element name="STREAM">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="type" default="locator">
          <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
              <xs:enumeration value="locator"/>
              <xs:enumeration value="other"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="href" type="xs:anyURI"/>
        <xs:attribute name="actuate" default="onRequest">
          <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
              <xs:enumeration value="onLoad"/>
              <xs:enumeration value="onRequest"/>
              <xs:enumeration value="other"/>
              <xs:enumeration value="none"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="encoding" type="encodingType" default="none"/>
        <xs:attribute name="expires" type="xs:dateTime"/>
        <xs:attribute name="rights" type="xs:token"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

```

Listing C.10: VOTable XML Schema: cont.

```

<!-- Expresses the coordinate system we are using -->
<xs:element name="COOSYS">
  <xs:complexType><xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="ID" type="xs:ID" use="required"/>
      <xs:attribute name="equinox" type="astroYear"/>
      <xs:attribute name="epoch" type="astroYear"/>
      <xs:attribute name="system" default="eq_FK5">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="eq_FK4"/>
            <xs:enumeration value="eq_FK5"/>
            <xs:enumeration value="ICRS"/>
            <xs:enumeration value="ecl_FK4"/>
            <xs:enumeration value="ecl_FK5"/>
            <xs:enumeration value="galactic"/>
            <xs:enumeration value="supergalactic"/>
            <xs:enumeration value="xy"/>
            <xs:enumeration value="barycentric"/>
            <xs:enumeration value="geo_app"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension></xs:simpleContent></xs:complexType>
  </xs:element>
</xs:schema>

```

The <COOSYS> tag

The celestial coordinate systems to be used throughout a particular VOTable are specified by the <COOSYS> tag. Its structure is illustrated by figure C.2.

The valid values for the `system` attribute of the <COOSYS> tag are shown in table C.1, with small descriptions of the different coordinate systems which can be specified.

In the case of the `eq_FK?` and `ecl_FK?` coordinate systems, the `equinox` attribute is used to fix the systems (with default value J2000 for `*_FK5`, and B1950 for `*_FK4`). If necessary, the `epoch` attribute can provide the epoch for the positions.

In the future, the <COOSYS> tag might be deprecated in favour of an STC[66] serialisation (complete or simplified).

The <RESOURCE> tag

<RESOURCE> tags provide data values, preceded with a description, of some logically independent data structure. We can see the structure of a <RESOURCE> tag in figure C.3.

VOTable Data Types

In the following we will provide descriptions of the different data types used within VOTables, together with their definition.

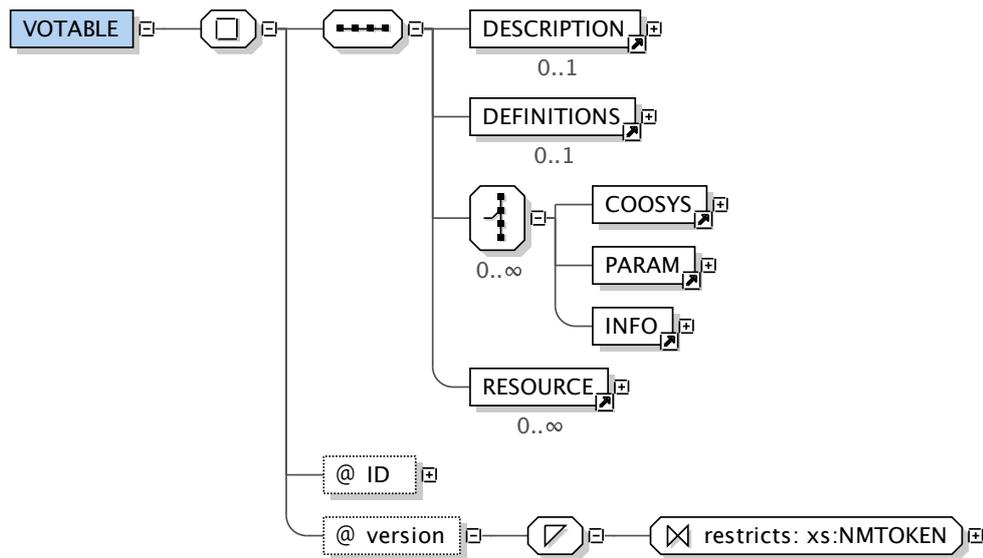


Figure C.1: VOTable high level structure, seen as elements of the `<VOTABLE>` tag. Inside the root `<VOTABLE>` tag, a `<DESCRIPTION>` tag can be used to describe in human readable form the contents of a particular VOTable. After that, in VOTable 1.0 the `<DEFINITIONS>` tag could be used to provide global information, coordinate systems, and params. In VOTable 1.1 it is recommended to specify global coordinate system definitions via the `<COOSYS>` tag right below the `<VOTABLE>` tag, followed by the `<PARAM>` tag (for specifying fixed values, such as a value for the Hubble constant, a global instrumental parameter, or similar constant values affecting the whole VOTable) and the `<INFO>` tag for documentation. Those tags can be used as many times as needed, and after them any number of `<RESOURCE>` tags can appear as many times as needed, specifying actual data resources. If more than one `<COOSYS>` tags are specified, their `ref` attribute must be set so that the relevant coordinate system can be referenced from within a `<TABLE>` tag. Generated with the XML Schema diagram generator of the oXygen XML Editor.

Table C.1: Meaning of the different valid values for attributes of the encodingType data type.

<i>system type</i>	Description
ICRS	Coordinates are given in the International Celestial Reference System.
eq_FK5	Equatorial coordinates, using the Fifth General Catalogue as reference frame.
eq_FK4	Equatorial coordinates, using the Fourth General Catalogue as reference frame.
ec1_FK5	Ecliptic coordinates, using the Fifth General Catalogue as reference frame.
ec1_FK4	Ecliptic coordinates, using the Fourth General Catalogue as reference frame.
galactic	Galactic coordinates, using the galactic reference frame, with the Sun at the centre, and the line in the galactic plane between the Sun and the centre of the Milky way used as baseline for determining galactic latitude and longitude.
supergalactic	Supergalactic coordinates, similar to galactic coordinates, but where the preferential plane is that containing the barycentres of the the Virgo, Perseo-Pisces, and Great Attractor galaxy clusters, while the preferential line is the intersection of the plane of the Milky Way with the supergalactic plane.
barycentric	Coordinates co-moving with the Sun, but with time measurements free from relativistic corrections due to the presence of gravity.
geo_app	Coordinates refer to the Earth longitude and latitude.
xy	Corresponds to a user-defined coordinate system, similar to the alpha and beta coordinates in the IRAM New Control System.

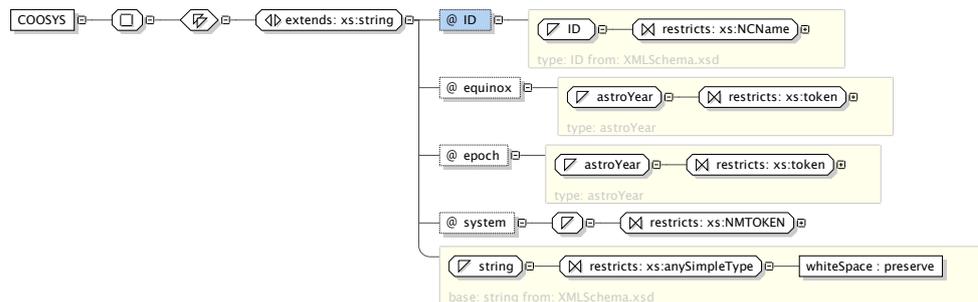


Figure C.2: The <COOSYS> tag attributes, with data types. The ID attribute can be used in ref attributes in coordinate fields in order to establish a particular coordinate system, if more than one <COOSYS> tags are specified. Both the epoch and equinox attributes correspond to an astroYear data type.

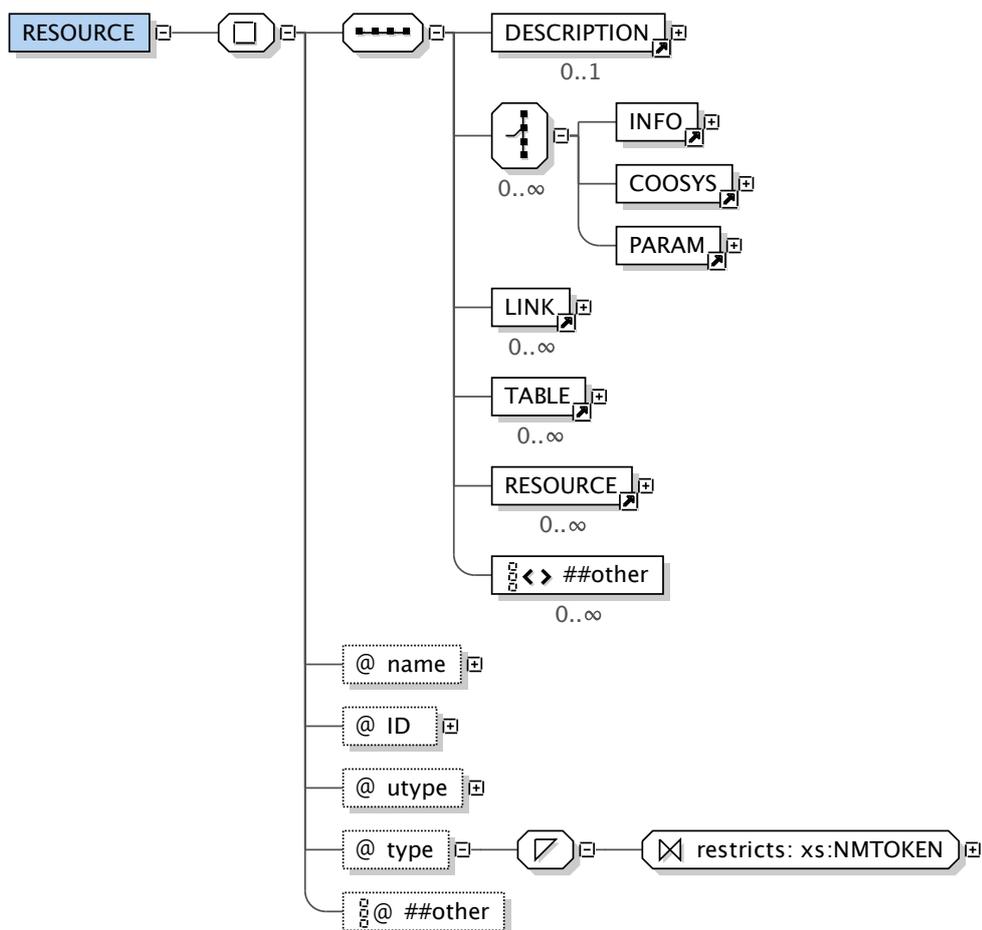


Figure C.3: Elements of the `<RESOURCE>` tag. `<RESOURCE>` tags can contain many of the tags the `<VOTABLE>` tag can hold, but additionally it can contain `<LINK>`, `<TABLE>` and `<RESOURCE>` tags. `<TABLE>` tags are used to include tabular data, while `<LINK>` tags are used to point to non-tabular data resources being described by `<DESCRIPTION>`. Generated with the XML Schema diagram generator of the oXygen XML Editor.

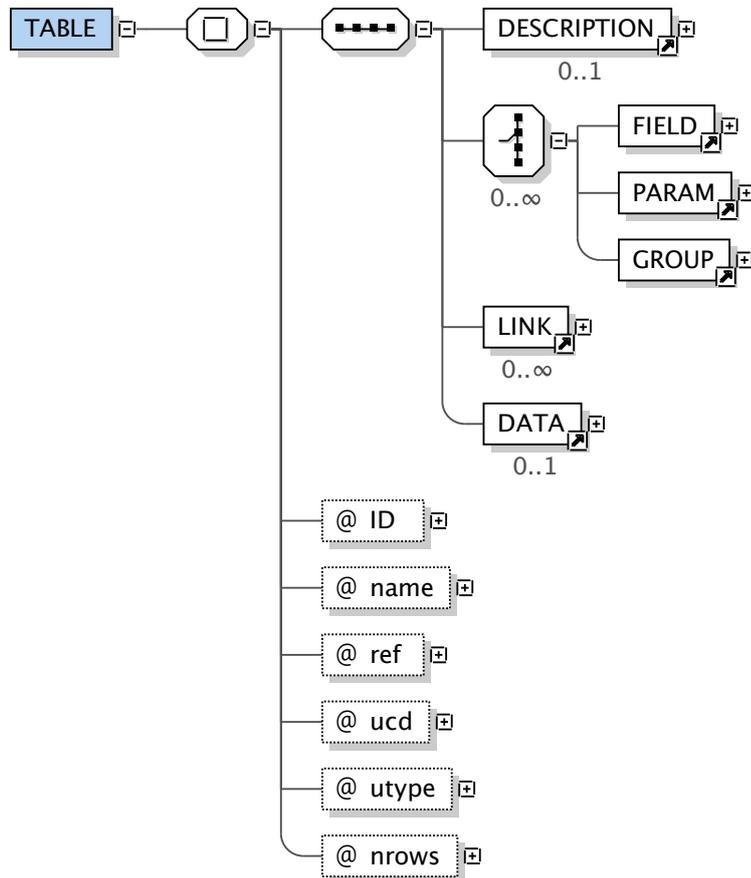


Figure C.4: Elements of the `<TABLE>` tag. `<TABLE>` tags might contain a `<DESCRIPTION>` tag, and then they can be followed by any number of `<FIELD>`, `<PARAM>` and `<GROUP>` tags. `<FIELD>` tags are used to specify the different fields of the `<DATA>`, in case the VOTable contains inline data. Otherwise, the `<LINK>` tag is used to reference remote data in other formats. The `<PARAM>` tag is used to specify data which is common to all rows of the table, as if it were a `<FIELD>` whose value is the same for all rows. Finally, the `<GROUP>` tag is used to group related fields or parameters, such as Right Ascension and Declination grouped into a coordinates `<GROUP>`. Generated with the XML Schema diagram generator of the oXygen XML Editor.

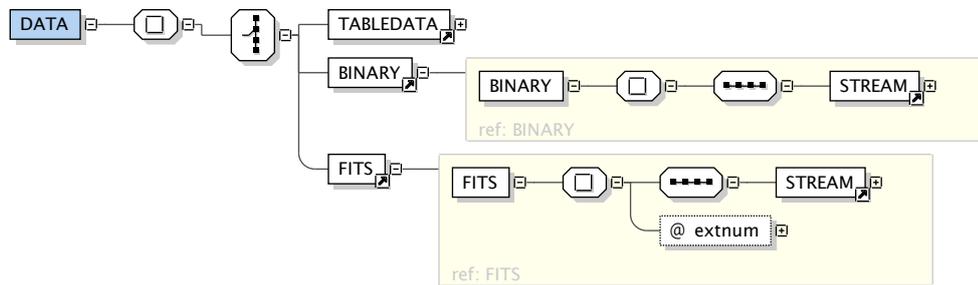


Figure C.5: Elements of the `<DATA>` tag. In a `<RESOURCE>` tag, tabular information is included within a `<DATA>` tag. That information can be pure XML tabular data within a `<TABLEDATA>` tag, a linked FITS file within a `<FITS>` tag, or embedded binary data within `<BINARY>` tag. The `<STREAM>` tag contains either the data or references to the data. See figure C.7 for more information. Generated with the XML Schema diagram generator of the oXygen XML Editor.

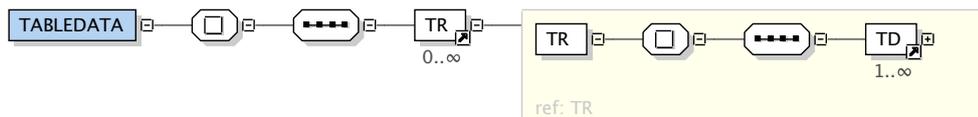


Figure C.6: Elements of the `<TABLEDATA>` tag. Inside the `<TABLEDATA>` tag several `<TR>` tags can exist, corresponding to different table rows. For each table row, there must be as many `<TD>` tags as `<FIELD>` tags in the `<TABLE>` tag containing a particular `<TABLEDATA>`.

anyText

`anyText` corresponds to a sequence of any characters of indeterminate length, with the additional restriction that they must not be processed in any way. This allows `anyText` elements to be able to embed XHTML, or any other kind of textual content.

astroYear

`astroYear` corresponds to a normalised string (a string where white space is collapsed, and line-feeds are removed) which conforms to the regular expression:

$$[JB]?[0-9]+([\.[0-9]*)?$$

The regular expression considers as a valid `astroYear` any string of one or more characters, optionally beginning either by `J` or `B`, followed by any

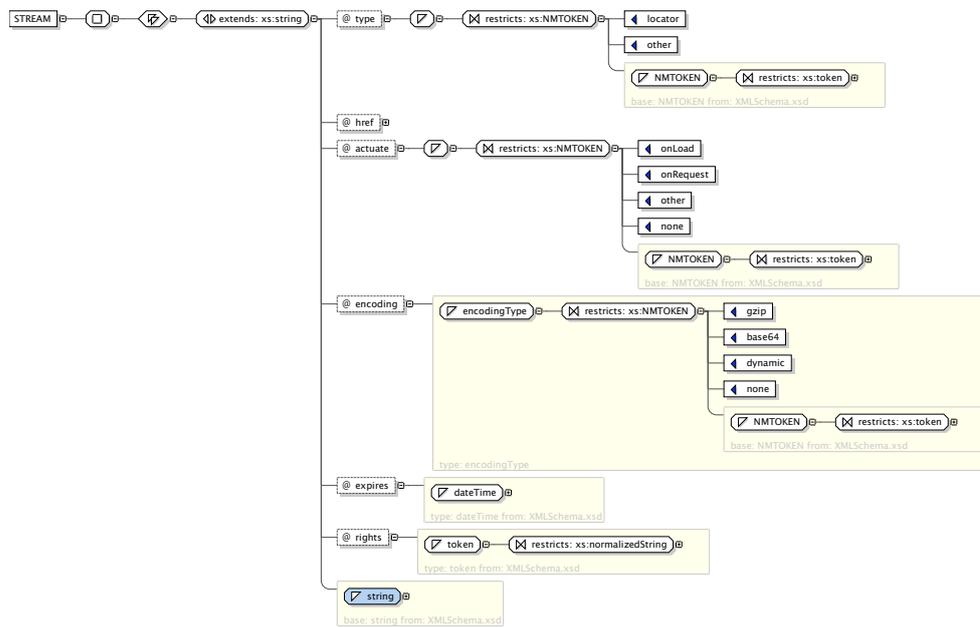


Figure C.7: Elements of the <STREAM> tag. In this figure we have expanded the different attributes for the <STREAM> tag in order to better understand what has to be provided. The <STREAM> tag is a string which can contain data encoded in the formats specified by the encoding attribute, only if the type attribute is not locator. If the type attribute were locator, the href attribute would contain an URI pointing to the actual data source. In any case, an expires attribute exists for transient datasets (for instance, on-demand synthetic datasets, image mosaics, or similar non-persistent datasets), the actuate attribute exists for **TODO: Find out what is the actuate attribute used for**, and the rights attribute can be used to store a string (or an escaped XML fragment) which specifies in a human readable form the access rights for such data. Generated with the XML Schema diagram generator of the oXygen XML Editor.

number (at least one) of decimal digits, optionally followed by a decimal dot and any number of decimal digits.

This pattern has been created with extensibility in mind, but the most usual strings used for attributes with the astroYear data type will be B1950 and J2000.

ucdType

ucdType corresponds to a normalised string which conforms to the regular expression:

`[A-Za-z0-9_.;\-\]*`

This regular expression considers as a valid `ucdType` any string of any number of characters to choose any alphanumeric digit, string, with the addition of the underscore (`_`), the dot (`.`), the semicolon (`;`), the backslash (`\`) or the dash (`-`). This pattern is valid for all UCDs in version 1.1, and for most version 1 UCDs, even when these UCDs can make use of these symbols: slash (`/`), plus (`+`) and percentage (`%`) signs.

However, not all strings that match this pattern can be considered UCDs. That is, matching the pattern is a necessary, but not sufficient, precondition.

arrayDEF

`arrayDEF` corresponds to a normalised string which conforms to the regular expression:

$$([\d-9]+x)*[\d-9]*[*]?(s\W)?$$

This regular expression considers a valid `arrayDEF` any string formed by whole numbers separated by the `x` character, possibly followed by another whole number, possibly followed by an asterisk (`*`), optionally followed by an `s`, and a non-word character (neither letter nor digit).

This data type is used to specify textually multi-dimensional arrays with defined dimensions, as in `12x512` for an array of 12 rows and 512 columns, or `12x*` for an array of 12 rows and an indeterminate number of columns.

When the `s` suffix is used, the non-word character is used to specify the string separator. For instance, `1024s`, denotes an array of strings, of up to 1024 characters in total, where the string elements of the array are separated by a comma (`,`).

encodingType

`encodingType` corresponds to one of the following strings: `gzip`, `base64`, `dynamic`, or `none`. Table C.2 shows the different meanings for each string.

dataType

`dataType` corresponds to one of the following strings: The string identifiers specify data types for `<FIELD>` and `<PARAM>` tags, as show in table C.3

precType

`precType` corresponds to a normalised string which conforms to the regular expression:

$$[EF]?[1-9][\d-9]*$$

Table C.2: Meaning of the different valid values for attributes of the `encodingType` data type.

<code>encodingType</code>	Description
<code>gzip</code>	Indicates that the link of the <code><STREAM></code> tag has been compressed with <code>gzip</code> .
<code>base64</code>	Indicates a <code>base64</code> encoded link or embedded data.
<code>dynamic</code>	Indicates that the stream encoding will be delivered dynamically when retrieving the link as part of the MIME type response.
<code>none</code>	No encoding other than that used for the VOTable is used.

This regular expression considers a valid `precType` any string encoding an integer, without any leading zero, and optionally preceded by an E or an F (which is the default). F specifies a fixed digit precision, while E indicates a relative precision.

For instance, 5 or F5 indicates a fixed five decimal digits precision, while E5 specifies a relative precision of 10^{-5} , or 5 significant figures.

Table C.3: Meaning of the different valid values for data type identifiers, data range, and storage needs.

Data type	Description	FITS equivalent type	Bytes
boolean	Boolean data, codified either as a 0 or 1 bit.	L	1
bit	Bitwise data (usually a bit array, corresponding to the FITS type). Storage space is given in bytes for groups of 8 bits.	X	$\lceil b/8 \rceil^a$
unsignedByte	Unsigned integer number in the 0 to 255 range.	B	1
short	Signed integer number in the -32768 to 32767 range.	I	2
int	Signed integer number in the -2147483648 to 2147483647 range.	J	4
long	Signed integer number in the -9223372036854775808 to 9223372036854775807 range.	K	8
char	Binary serialisation an ASCII (7-bit) character.	A	1
unicodeChar	Binary UTF-16 representation of a Unicode character.	n/a ^b	2
float	Binary ANSI/IEEE-754 32-bit floating point number in big-endian order.	E	4
double	Binary ANSI/IEEE-754 64-bit double precision floating point number in big-endian order.	D	8
floatComplex	Pair of float, with the first element as the real part of the complex number, and the second as the imaginary part.	C	8
doubleComplex	Pair of double, with the first element as the real part of the complex number, and the second as the imaginary part.	C	16

^aWhere b is the number of bits in the bit array, and $\lceil x \rceil$ represents the nearest integer greater or equal to x .

^bThere is no equivalence to the Unicode character in the FITS standard, which was developed much earlier than the Unicode standard.

Appendix D

VO protocols

In this appendix we present a quick guide to the usage of the different available IVOA proposed Virtual Observatory protocols, with the aim of making easier to understand the official standard documents. It can be used as a fast reference of what is needed both for clients and servers compliant with these protocols.

D.1 Simple ConeSearch

Query syntax

ConeSearch v1.0 query with all parameters. Mandatory parameters in black. **Optional parameters in red** (light grey in black and white).

```
endPointURL?RA=rightAscension&  
DEC=declination&  
SR=searchRadius&  
VERB=verbosity
```

Parameter description

endPointURL represent the Uniform Resource Locator of a deployed ConeSearch web service. Usually of the form:

```
http://host.com/path/to/service
```

rightAscension codifies the Right Ascension of the region of interest. It has to be provided in decimal degrees, in the ICRS coordinate system.

declination codifies the Declination of the region of interest. It has to be provided in decimal degrees, in the ICRS coordinate system.

searchRadius is given in decimal degrees.

verbosity specifies reply verbosity; verbosity can be one of 1, 2 or 3. 1 provides the least verbose reply, and 3 the most verbose.

Service reply

A ConeSearch service reply consist of a VOTABLE, with complies with the following :

- The VOTABLE response contains just one RESOURCE, which contains just one TABLE.
- Table fields:
 - Exactly ONE field has UCD="ID_MAIN", with the string data type, which provides an unique ID for each table row/record.
 - Exactly ONE field has UCD="POS_EQ_RA_MAIN", with data type double, representing the Right Ascension of the observed source in the ICRS coordinate system.
 - Exactly ONE field has UCD="POS_EQ_DEC_MAIN", with data type double, representing the Declination of the observed source in the ICRS coordinate system.
 - An additional <FIELD> or <PARAM> tag with UCD="OBS_ANG-SIZE" can be used to specify a per-record (<FIELD>) or per-table (<PARAM>) positional error on source positions, or angular size for resolved observations.
 - The rest of the returned fields depend on the catalog, but all <FIELD> or <PARAM> tags should specify description, data type, and UCD.

D.2 Simple Image Access Protocol

Query syntax

Mandatory query parameters listed in black. Optional parameters listed in blue. Not all services implement all optional parameters, except for FORMAT=formatType. Use metadata as formatType to retrieve supported parameters.

```
endPointURL?POS=RA,DEC&
SIZE=searchRadius&
FORMAT=formatType&
INTERSECT=intersectionMechanism&
NAXIS=axisSizeVector&
CFRAME=coordFrame&
EQUINOX=equinoxSpec&
```

```

CRPIX=refPixCoordVector&
CRVAL=refPixValueVector&
CDELTA=pixScaleVector&
ROTANG=rotationAngle&
PROJ=projectionKind&
VERB=verbosity

```

Parameter description

endPointURL represents the Uniform Resource Locator of a deployed SIAP web service. Usually of the form:

```
http://host.com/path/to/service
```

RA,DEC is a pair of double quantities, separated by a comma (,), representing Right Ascension and Declination in decimal degrees. No official consideration of coordinate system, but ICRS can be assumed.

searchRadius is either a double quantity, or a pair of double quantities separated by a comma (,). If a pair of values is specified, the first is assumed to be an RA width, and the second a DEC width. The region thus defined has constant RA and DEC edges. The combination of POS and SIZE will be called *region of interest* (ROI) in the following.

formatType is a string representation of the MIME media type of files being asked for to the service. It will be one of:

- image/fits
- image/png
- image/jpeg
- text/html
- fits (equivalent to image/fits)
- graphic (equivalent to image/png; image/jpeg)
- all (default value)

Additionally, if **formatType** is metadata, the service will reply with a VOTABLE with all supported input parameters.

intersectionMechanism specifies how to evaluate the intersection of candidate images with the ROI specified. It has to be ONE of the following:

COVERS: returned images completely includes (covers) the entire ROI.

ENCLOSED: returned images are completely covered (enclosed) by the ROI. Uses the same mechanism of **COVERS**, reversing the role of ROI and test image.

CENTER: returned images include at least the center of the ROI. This is an special case of OVERLAPS.

OVERLAPS: returned images have some common points with the specified ROI. OVERLAPS should return at least all images returned by CENTER.

axisSizeVector is the size of the output image in pixels. This is a vector-valued quantity, expressed as `NAXIS=<width>,<height>`.

coordFrame is a string specifying a coordinate system reference frame. It is one of:

- ICRS (default)
- FK5
- FK4
- ECL
- GAL
- SGAL

equinoxSpec is the epoch of the mean equator and equinox for the coordinate system reference frame specified by `coordFrame`. It is not required for ICRS, and defaults to B1950 for FK4, and to J2000 otherwise.

refPixCoordVector provides the coordinates of the reference pixel, in pixel coordinates of the output image. This is a vector-valued quantity of the same size as `axisSizeVector`.

refPixValueVector is a vector of the world coordinates relative to `coordFrame` at the `refPixCoordVector`. Defaults to the ROI center coordinates, transformed to the output coordinate system reference if other than ICRS.

pixScaleVector provides the scale of the output image in decimal degrees per pixel. A negative value implies an axis flip. Since the default image orientation is N up and E to the left, the default sign of `pixScaleVector` is [-1,1]. Can be given as comma separated vector, or as a single value; if only one value is given, it applies to both image axes.

rotationAngle specifies the rotation angle for the output image in degrees relative to `coordFrame` (an image which is unrotated in one reference frame may be rotated in another). Rotation of the WCS declination or latitude axis with respect to the second axis of the image, measured in the counterclockwise direction (following FITS WCS convention, which in turn is based on the old AIPS convention). Defaults to 0 (no rotation).

projectionKind specifies the celestial projection of the output image; is one of TAN, SIN, ARC, et cetera. Defaults to TAN.

verbosity is an integer value, ranging from 0 to 3 that specifies reply verbosity (how much additional meta data are provided).

Optional parameters: query with all mandatory and optional parameters in version 1:

Remember that parameters not supported by the service will be discarded. Call the service with `FORMAT=metadata` to retrieve supported parameters (see below).

SIAP v2 extensions: query with mandatory and extended parameters:

```
endPointURL?POS=RA,DEC&
SIZE=searchRadius&
REQUEST=requestKind&
TIME=timeSpec&
ANGRP=angularResolution&
PUBDID=pubID&
CREATORID=creatorID&
COLLECTION=collectionID&
TOP=topNumResults&
MAXREC=maxRecords&
MTIME=incorporationTime&
COMPRESS=compressionFlag&
RUNID=jobRunID&
REDSHIFT=redShiftRange&
TARGETNAME=targetName
```

See SSAP for explanations on TIME, PUBDID, CREATORID, COLLECTION, TOP, MAXREC, MTIME, COMPRESS, RUNID, REDSHIFT and TARGETNAME.

requestKind specifies the kind of image services being required: cut-out, mosaicing, atlas archive, pointed archive. A service can support all or some (at least one) of these request kinds.

angularResolution specifies the angular resolving power of the image, for interpolated images.

SIAP multidimensional extensions: for SIAP to be extended to support multidimensional data sets, keywords MAXIS, CRPIX, CRVAL and CDELTA should be enhanced to specify N dimensions (typically, $N \in \{3,4\}$); first two should always be RA and DEC, with either time, frequency wavelength or velocity as the third dimension, and polarisation as a fourth axis, if present.

Service reply: a SIAP service reply consist of a VOTABLE, with complies with the following :

-

D.3 Simple Spectral Access Protocol

Query syntax

Parameters all SSAP compliant services need to implement are in black. Parameters whose implementation is optional, but recommended, appear in blue, and parameters whose implementation is completely optional appear in red.

```

endPointURL?POS=RA,DEC&
SIZE=searchRadius&
BAND=freqRange&
TIME=timeRange&
FORMAT=formatType&
SPECRP=specResol&
SPATRES=spatialResol&
PUBDID=pubID&
CREATORID=creatorID&
COLLECTION=collectionID&
TOP=topNumResults&
MAXREC=maxRecords&
MTIME=modificationTime&
COMPRESS=compressionFlag&
RUNID=jobRunID&
APERTURE=apertAngle&
TIMERES=timeResol&
SNR=signal2noise&
REDSHIFT=redShiftRange&
VARAMPL=amplitudeVariability&
TARGETNAME=targetName&
TARGETCLASS=targetClass&
FLUXCALIB=fluxCalibKind&
WAVECALIB=waveCalibKind

```

None of the service mandatory parameters is needed *per se* in a particular query: the following are legal, minimal SSAP queries:

```

endPointURL?POS=RA,DEC&
SIZE=searchRadius
endPointURL?TIME=timeRange&
BAND=freqRange

```

Parameter description

The following is a description of the different SSAP parameters. In SSAP, parameters can be qualified by juxtaposing additional qualifiers, separated by a semicolon (;). Appropriate qualifiers will be pointed out as needed.

endPointURL represents the Uniform Resource Locator of a deployed SIAP web service. Usually of the form:

```
http://host.com/path/to/service
```

RA,DEC is a pair of double quantities, separated by a comma (,), representing Right Ascension and Declination in decimal degrees, in the ICRS coordinate system unless otherwise specified.

POS=RA,DEC can be substituted by GALACTICPOS=Gallat,Gallong. Not all services need to support GALACTICPOS.

searchRadius is either a double quantity, or a pair of double quantities separated by a comma (,). If a pair of values is specified, the first is assumed to be an RA width, and the second a DEC width. The region thus defined has constant RA and DEC edges. The combination of POS and SIZE will be called *region of interest* (ROI) in the following.

freqRange specifies a frequency range the query should be restricted to, expressed as low/high wavelengths in meters: BAND=2.7E-7/0.13. A source qualifier can be used to specify that the frequencies are being specified at the source LSR: BAND=0.001/0.01;source

timeRange restricts the query to a given date, specified as an ISO 8601 [1] date-time string (e.g., TIME=1998-05-21), or a date-time range in UTC: TIME=1998-05-21/1999). Qualifiers can be used to specify Local Sidereal Time or other time zones:

```
TIME=2003-04-01T02:00:00/2003-04-01T06:00:00;LST.
```

formatType is a string representation of the MIME media type of files being asked for to the service. It will be one of:

- application/fits
- compliant (votable or xml)
- native (native format of the archive)
- graphic (PNG, JPEG or EPS)
- votable (compare with XML)
- fits (equivalent to application/fits)

- `xml` (SED XML serialisation)
- `all` (default value)

Additionally, if `formatType` is `metadata`, the service will reply with a VOTABLE with all supported input parameters.

`formatType` can be qualified by the conventions used for spectral reduction, such as `convention=STScI-STIS`.

specResol is a double specifying spectral resolving power, $\frac{\lambda}{\Delta\lambda}$; `specResol` is a dimensionless quantity.

spatialResol is a double specifying the minimum spatial resolution, in decimal degrees, corresponding to signal PSF.

pubID is a string holding the NASA ADS publication ID.

creatorID is a string specifying the IVORN for the data set creator.

collectionID is a string specifying an IVORN for a particular collection, or a short-name: for instance, the short-name `HST Spectra`, or its IVORN: `ivo://mast.stsci/ssap/hst`.

topNumResults is an integer specifying the maximum number of results to be returned by the query, but requesting them sorted by an *heuristic score* defined by the service. That score is built to be higher for better matching records. In any case, the same query should always return the same `topNumResults` records. Default value is up to the service, but should be small enough to deliver fast responses.

maxRecords is an integer specifying the maximum number of results to be returned by the query. The same query might return different records depending on service implementation. Default value is up to the service, but should be small enough to deliver fast responses.

modificationTime is similar to `timeRange` in syntax, but refers to record creation or modification date-time.

compressionFlag is a string containing either `true` or `false`. If `compressionFlag` is `true`, returned data might be compressed. Default is `false`.

jobRunID is a string with contains a client side request ID, so that records from different services, called with the same `jobRunID`, can be easily aggregated.

apertAngle is a double used to specify a synthetic aperture angle in decimal degrees in order to extract spectra from data cubes, event lists or other fundamental data. For extraction services, this parameter is mandatory, and the service should provide a sensible default.

timeResol is a double specifying minimum time resolution in seconds (typically, the bounds of the exposure time coverage).

signal2noise is a dimensionless double specifying the Signal to Noise ratio, as a number of times noise RMS.

redShiftRange is a string encoding a double or a range of doubles, which specifies a particular dimensionless redshift or range, calculated using the optical convention $z = \frac{\Delta\lambda}{\lambda}$: REDSHIFT=1.3/3.0 means $z \in [1.3, 3.0]$, REDSHIFT=1.3/ means $z \in [1.3, +\infty)$, while querying with REDSHIFT=/3.0 means $z \in [0, 3.0]$. Negative values can be used to indicate *blueshifts*.

amplitudeVariability is dimensionless double in the range [0.0,1.0] indicating amplitude variations from the maximum.

targetName is a string providing the name of target that will need to be resolved by SIMBAD, NED, or similar services. To be used instead of coordinates.

targetClass is a string providing comma delimited list of types of astronomical objects to be searched for. Valid types form controlled vocabulary coming from Simbad [3]. Eventually types should come from the IVOA Thesaurus.

fluxCalibKind is a string encoding a boolean value (true or false) indicating whether we require returned data fluxes to be calibrated. No distinction is made between absolute or relative flux calibration.

waveCalibKind is a string encoding a boolean value (true or false) indicating whether we require returned data wavelengths to be calibrated. No distinction is made between absolute or relative wavelength calibration.

Service reply

SSAP services reply with a VOTABLE, which complies with the following:

- VOTABLE condition

Bibliografía

- [1] A. Comte, *The positive philosophy of Auguste Comte*, vol. 1 of 2, ch. I: General View of Book II: Astronomy. London: G. Bell and sons, 1896. [1.1](#)
- [2] G. de Vaucouleurs, *Astronomical photography. From the daguerrotype to the electron camera*. London: Faber & Faber, 1961, 1961. [1.1](#)
- [3] P. Abrahams, "The early history of astrophotography." [1.1](#)
- [4] K. G. Jansky, "Electrical disturbances apparently of extraterrestrial origin," *Proceedings of the Institute of Radio Engineers*, vol. 21, pp. 1387–1398, October 1933. [1.1](#)
- [5] K. G. Jansky, "A note on the source of interstellar interference," *Proceedings of the Institute of Radio Engineers*, vol. 23, pp. 1158–1163, October 1935. [1.1](#)
- [6] R. W. Wilson, "The cosmic microwave background radiation," *Science*, vol. 205, pp. 866–874, Aug. 1979. [1.1](#)
- [7] A. A. Penzias, "The origin of the elements," *Science*, vol. 205, pp. 549–554, Aug. 1979. [1.1](#)
- [8] S. V. W. Beckwith, M. Stiavelli, A. M. Koekemoer, J. A. R. Caldwell, H. C. Ferguson, R. Hook, R. A. Lucas, L. E. Bergeron, M. Corbin, S. Joojee, N. Panagia, M. Robberto, P. Royle, R. S. Somerville, and M. Sosey, "The Hubble Ultra Deep Field," *The Astronomical Journal*, vol. 132, pp. 1729–1755, Nov. 2006. [1.1](#)
- [9] J. Gunn and D. Weinberg, "The Sloan Digital Sky Survey," in *Wide Field Spectroscopy and the Distant Universe* (S. J. Maddox and A. Aragon-Salamanca, eds.), pp. 3–+, 1995. [1.2](#)
- [10] D. G. York, J. Adelman, J. E. Anderson, Jr., S. F. Anderson, J. Annis, N. A. Bahcall, J. A. Bakken, R. Barkhouser, S. Bastian, E. Berman, W. N. Boroski, S. Bracker, C. Briegel, J. W. Briggs, J. Brinkmann, R. Brunner,

- S. Burles, L. Carey, M. A. Carr, F. J. Castander, B. Chen, P. L. Colestock, A. J. Connolly, J. H. Crocker, I. Csabai, P. C. Czarapata, J. E. Davis, M. Doi, T. Dombeck, D. Eisenstein, N. Ellman, B. R. Elms, M. L. Evans, X. Fan, G. R. Federwitz, L. Fiscelli, S. Friedman, J. A. Frieman, M. Fukugita, B. Gillespie, J. E. Gunn, V. K. Gurbani, E. de Haas, M. Haldeman, F. H. Harris, J. Hayes, T. M. Heckman, G. S. Hennessy, R. B. Hindsley, S. Holm, D. J. Holmgren, C.-h. Huang, C. Hull, D. Husby, S.-I. Ichikawa, T. Ichikawa, Z. Ivezić, S. Kent, R. S. J. Kim, E. Kinney, M. Klaene, A. N. Kleinman, S. Kleinman, G. R. Knapp, J. Korienek, R. G. Kron, P. Z. Kunszt, D. Q. Lamb, B. Lee, R. F. Leger, S. Limmongkol, C. Lindenmeyer, D. C. Long, C. Loomis, J. Loveday, R. Lucinio, R. H. Lupton, B. MacKinnon, E. J. Mannery, P. M. Mantsch, B. Margon, P. McGehee, T. A. McKay, A. Meiksin, A. Merelli, D. G. Monet, J. A. Munn, V. K. Narayanan, T. Nash, E. Neilsen, R. Neswold, H. J. Newberg, R. C. Nichol, T. Nicinski, M. Nonino, N. Okada, S. Okamura, J. P. Ostriker, R. Owen, A. G. Pauls, J. Peoples, R. L. Peterson, D. Petravick, J. R. Pier, A. Pope, R. Pordes, A. Prosapio, R. Rechenmacher, T. R. Quinn, G. T. Richards, M. W. Richmond, C. H. Rivetta, C. M. Rockosi, K. Ruthmansdorfer, D. Sandford, D. J. Schlegel, D. P. Schneider, M. Sekiguchi, G. Sergey, K. Shimasaku, W. A. Siegmund, S. Smee, J. A. Smith, S. Snedden, R. Stone, C. Stoughton, M. A. Strauss, C. Stubbs, M. Subbarao, A. S. Szalay, I. Szapudi, G. P. Szokoly, A. R. Thakar, C. Tremonti, D. L. Tucker, A. Uomoto, D. Vanden Berk, M. S. Vogeley, P. Waddell, S.-i. Wang, M. Watanabe, D. H. Weinberg, B. Yanny, and N. Yasuda, "The Sloan Digital Sky Survey: Technical Summary," *The Astronomical Journal*, vol. 120, pp. 1579–1587, Sept. 2000. 1.2
- [11] M. F. Skrutskie, R. M. Cutri, R. Stiening, M. D. Weinberg, S. Schneider, J. M. Carpenter, C. Beichman, R. Capps, T. Chester, J. Elias, J. Huchra, J. Liebert, C. Lonsdale, D. G. Monet, S. Price, P. Seitzer, T. Jarrett, J. D. Kirkpatrick, J. E. Gizis, E. Howard, T. Evans, J. Fowler, L. Fullmer, R. Hurt, R. Light, E. L. Kopan, K. A. Marsh, H. L. McCallon, R. Tam, S. Van Dyk, and S. Wheelock, "The Two Micron All Sky Survey (2MASS)," *The Astronomical Journal*, vol. 131, pp. 1163–1183, Feb. 2006. 1.2
- [12] R. H. Becker, R. L. White, and D. J. Helfand, "The VLA's FIRST Survey," in *Astronomical Data Analysis Software and Systems III* (D. R. Crabtree, R. J. Hanisch, and J. Barnes, eds.), vol. 61 of *Astronomical Society of the Pacific Conference Series*, pp. 165–+, 1994. 1.2
- [13] J. J. Condon, W. D. Cotton, E. W. Greisen, R. A. Perley, Q. F. Yin, and J. J. Broderick, "The NRAO VLA Sky Survey," in *Bulletin of the American Astronomical Society*, vol. 25 of *Bulletin of the American Astronomical Society*, pp. 1389–+, Dec. 1993. 1.2

- [14] M. P. Haynes and the ALFALFA Team, “The Arecibo Legacy Fast ALFA (ALFALFA) Extragalactic HI Survey,” *ArXiv e-prints*, vol. 806, June 2008. [1.2](#)
- [15] R. Giovanelli, M. P. Haynes, B. R. Kent, P. Perillat, A. Saintonge, N. Brosch, B. Catinella, G. L. Hoffman, S. Stierwalt, K. Spekkens, M. S. Lerner, K. L. Masters, E. Momjian, J. L. Rosenberg, C. M. Springob, A. Boselli, V. Charmandaris, J. K. Darling, J. Davies, D. G. Lambas, G. Gavazzi, C. Giovanardi, E. Hardy, L. K. Hunt, A. Iovino, I. D. Karachentsev, V. E. Karachentseva, R. A. Koopmann, C. Marinoni, R. Minchin, E. Muller, M. Putman, C. Pantoja, J. J. Salzer, M. Scodreggio, E. Skillman, J. M. Solanes, C. Valotto, W. van Driel, and L. van Zee, “The Arecibo Legacy Fast ALFA Survey. I. Science Goals, Survey Design, and Strategy,” *The Astronomical Journal*, vol. 130, pp. 2598–2612, Dec. 2005. [1.2](#)
- [16] G. E. Moore, “Cramming more components onto integrated circuits,” *Electronics*, vol. 38, April 1965. [1.3](#)
- [17] European Southern Observatory (ESO), “The ESO/ST-ECF Science Archive Facility: An overview of our services,” brochure, European Southern Observatory, Space Telescope European Coordination Facility, April 2004. [1.2](#)
- [18] A. S. Szalay and J. Gray, “The World-Wide Telescope,” *Science*, vol. 293, pp. 2037–2040, Sept. 2001. [1.3](#), [2.1](#), [2.2](#)
- [19] R. J. Brunner, S. G. Djorgovski, and A. S. Szalay, eds., *Virtual Observatories of the Future*, vol. 225 of *Astronomical Society of the Pacific Conference Series*, 2001. [1.3](#)
- [20] F. Genova, F. Bonnarel, P. Dubois, D. Egret, P. Fernique, S. Lesteven, F. Ochsenbein, and M. Wenger, “Mining the Sky: Proceedings of the MPA/ESO/MPE Workshop Held at Garching, Germany, July 31 - August 4, 2000 with the CDS Services,” in *Mining the Sky* (A. J. Banday, S. Zaroubi, and M. Bartelmann, eds.), pp. 674–+, 2001. [1.3](#)
- [21] F. Genova, “Interoperability,” in *Astronomical Data Analysis Software and Systems XI* (D. A. Bohlender, D. Durand, and T. H. Handley, eds.), vol. 281 of *Astronomical Society of the Pacific Conference Series*, pp. 41–+, 2002. [2.1](#)
- [22] D. C. Wells, E. W. Greisen, and R. H. Harten, “FITS: A Flexible Image Transport System,” *Astronomy & Astrophysics, Supplement Series*, vol. 44, pp. 363–+, June 1981. [2.3](#), [B.1](#)

- [23] C. Flatters, “The FITS Interferometry Data Interchange Format,” AIPS Memo 102, National Radio Astronomy Observatory, Associated Universities, Inc., Washington DC, USA, August 2000. [2.3](#)
- [24] D. Muders, E. Polehampton, and J. Hatchell, “Multi-Beam FITS Raw Data Format,” tech. rep., Max-Planck-Institut für Radioastronomie, December 2005. [2.3](#), [6](#), [6.1](#), [12.2](#)
- [25] R. W. Garwood, “SDFITS: A Standard for Storage and Interchange of Single Dish Data,” in *Astronomical Data Analysis Software and Systems IX* (N. Manset, C. Veillet, and D. Crabtree, eds.), vol. 216 of *Astronomical Society of the Pacific Conference Series*, pp. 243–+, 2000. [2.3](#)
- [26] FITS Working Group, “Definition of the Flexible Image Transport System (FITS),” FITS Standard Version 3.0, Commission 5: Documentation and Astronomical Data-International Astronomical Union, <http://fits.gsfc.nasa.gov/iaufwg/>, July 2008. [2.3](#)
- [27] F. Ochsenbein, M. Albrecht, A. Brighton, P. Fernique, D. Guillaume, R. J. Hanisch, E. Shaya, and A. Wicenec, “Using XML for Accessing Resources in Astronomy,” in *Astronomical Data Analysis Software and Systems IX* (N. Manset, C. Veillet, and D. Crabtree, eds.), vol. 216 of *Astronomical Society of the Pacific Conference Series*, pp. 83–+, 2000. [2.3](#), [A.1](#)
- [28] B. Thomas, E. Shaya, and C. Cheung, “Converting FITS into XML: Methods and Advantages,” in *Astronomical Data Analysis Software and Systems X* (F. R. Harnden, Jr., F. A. Primini, and H. E. Payne, eds.), vol. 238 of *Astronomical Society of the Pacific Conference Series*, pp. 487–+, 2001. [2.3](#)
- [29] K. Blackburn, A. Lazzarini, T. Prince, and R. Williams, *XSIL: Extensible scientific interchange language*, vol. 1593/1999 of *Lecture Notes in Computer Science*, pp. 513–524. Springer, 1999. [2.3](#)
- [30] E. Shaya, B. Thomas, J. Gass, J. Blackwell, and C. Cheung, “XDF - an XML-based Scientific Interchange Language,” in *Bulletin of the American Astronomical Society*, vol. 32 of *Bulletin of the American Astronomical Society*, pp. 1600–+, Dec. 2000. [2.3](#)
- [31] D. Tody and R. Plante, “Simple Image Access Protocol,” IVOA Working Draft 1.01, International Virtual Observatory Alliance, October 2008. [2.4](#), [A.1](#)
- [32] D. Tody and M. Dolensky, “Simple Spectral Access Protocol,” IVOA Proposed Recommendation Version 1.01, International Virtual Observatory Alliance, file://localhost/Users/jdsant/Documents/IAA/SSA-20070604.pdf 2007. [2.4](#), [A.1](#)

- [33] M. Dolensky and D. Tody, "The Simple Spectral Access protocol," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (P. J. Quinn and A. Bridger, eds.), vol. 5493 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 262–268, Sept. 2004. [2.4](#)
- [34] R. Williams, R. Hanisch, A. Szalay, and R. Plante, "Simple cone search," Proposed Recommendation Version 1.03, International Virtual Observatory Alliance, February 2008. [2.4](#), [A.1](#)
- [35] C. Lagoze, H. Van de Sompel, M. Nelson, and S. Warner, "The Open Archives Initiative Protocol for Metadata Harvesting," Tech. Rep. v2.0, Open Archives Initiative, June 2002. [2.8](#)
- [36] J. Kunze, R. Guenther, M. Hlava, C. Morgan, and J. Perkins, "The Dublin Core Metadata Element Set," American National Standard Z39-85-2007, ANSI/NISO, NISO Press, Bethesda, Maryland, U.S.A., May 2007. [2.8](#)
- [37] R. Plante, K. Benson, M. Graham, G. Greene, P. Harrison, G. Lemson, T. Linde, G. Rixon, and A. Stébé, "VOResource: an XML Encoding Schema for Resource Metadata," IVOA Recommendation v1.03, International Virtual Observatory Alliance, February 2008. [2.8](#)
- [38] R. Plante, A. Stébé, K. Benson, M. Graham, G. Greene, P. Harrison, T. Linde, G. Rixon, and IVOA Registry WG, "VODataService: a VO-Resource Schema extension for describing collections and services," IVOA Working Draft v1.01, International Virtual Observatory Alliance, October 2008. [2.8](#)
- [39] P. Quinn, R. Hanisch, and A. Lawrence, "IVOA Statement and Roadmap 2002-2005," June 2002. [2.9](#)
- [40] D. de Young, G. Djorgovski, F. Genova, K. Gorski, B. Hanisch, A. Kembhavi, A. Lawrence, T. Linde, O. Malkov, R. Moore, R. Norris, P. Quinn, B. Pirenne, D. Schade, A. S. Szalay, V. Vitkovsky, W. Voges, N. A. Walton, and R. Williams, "First IVOA Full Meeting," Minutes FM1, International Virtual Observatory Alliance, Garching bei München, Germany (<http://www.ivoa.net/internal/IVOA/IvoaRepMin/ivoa-fm1-20020613.pdf>), June 2002. [2.9](#)
- [41] P. J. Quinn and K. M. Górski, eds., *Toward an International Virtual Observatory*, 2004. [2.9](#)
- [42] P. J. Quinn, M. Allen, K. Andrews, T. Boch, F. Bonnarel, S. Derriere, M. Dolensky, P. Fernique, M. Hill, M. C. Leoni, A. Linde, A. Micol, B. Pirenne, A. M. S. Richards, A. Schaaff, G. Tissier, N. A. Walton, and A. Wicenec, "The AVO Prototype," in *Astronomical Data Analysis*

- Software and Systems (ADASS) XIII* (F. Ochsenbein, M. G. Allen, and D. Egret, eds.), vol. 314 of *Astronomical Society of the Pacific Conference Series*, pp. 304–+, July 2004. 2.9
- [43] R. Gutiérrez, C. Rodrigo, and E. Solano, “The Spanish Virtual Observatory (SVO),” in *Astronomical Data Analysis Software and Systems XV* (C. Gabriel, C. Arviset, D. Ponz, and E. Solano, eds.), vol. 351 of *ASP Conference Series*, pp. 19–+, July 2006. 2.9
- [44] P. Harrison, “A proposal for a Common Execution Architecture,” IVOA WG Internal Draft 2005-05-12 Version 1.2, International Virtual Observatory Alliance, May 2005. 2.10, A.1
- [45] F. Murtagh, A. Heck, and P. Benvenuti, *Astronomy from large databases - scientific objectives and methodological approaches. Proceedings of a conference, held at Garching, 12 - 14 October 1987*. 1988. 2.11
- [46] M. J. Kurtz, T. Karakashian, C. S. Grant, G. Eichhorn, S. S. Murray, J. M. Watson, P. G. Ossorio, and J. L. Stoner, “Intelligent Text Retrieval in the NASA Astrophysics Data System,” in *Astronomical Data Analysis Software and Systems II* (R. J. Hanisch, R. J. V. Brissenden, and J. Barnes, eds.), vol. 52 of *Astronomical Society of the Pacific Conference Series*, pp. 132–+, Jan. 1993. 2.11
- [47] M. Wenger, F. Ochsenbein, D. Egret, P. Dubois, F. Bonnarel, S. Borde, F. Genova, G. Jasiewicz, S. Laloë, S. Lesteven, and R. Monier, “The SIMBAD astronomical database. The CDS reference database for astronomical objects,” *Astronomy & Astrophysics, Supplement Series*, vol. 143, pp. 9–22, Apr. 2000. 2.11
- [48] B. F. Madore, G. Helou, H. G. Corwin, Jr., M. Schmitz, X. Wu, and J. Bennett, “The NASA/IPAC Extragalactic Database,” in *Astronomical Data Analysis Software and Systems I* (D. M. Worrall, C. Biemesderfer, and J. Barnes, eds.), vol. 25 of *Astronomical Society of the Pacific Conference Series*, pp. 47–+, 1992. 2.11
- [49] P. Boyce, “Journals, Data and Abstracts Make an Integrated Electronic Resource,” in *Bulletin of the American Astronomical Society*, vol. 28 of *Bulletin of the American Astronomical Society*, pp. 1280–+, Dec. 1996. 2.11
- [50] P. M. Rodríguez-Pascual, R. González-Riestra, N. Schartel, and W. Wamsteker, “The IUE INES System: Improved data extraction procedures for IUE,” *Astronomy & Astrophysics, Supplement Series*, vol. 139, pp. 183–197, Oct. 1999. 2.11
- [51] F. Ochsenbein, P. Bauer, and J. Marcout, “The Vizier database of astronomical catalogues,” *Astronomy & Astrophysics, Supplement Series*, vol. 143, pp. 23–32, Apr. 2000. 2.11

- [52] C. Imhoff, F. Abney, D. Christian, M. Donahue, R. Hanisch, T. Kimball, K. Levay, P. Padovani, M. Postman, M. Smith, and R. Thompson, "Resources Available through the Multimission Archive at Space Telescope (MAST)," in *Bulletin of the American Astronomical Society*, vol. 31 of *Bulletin of the American Astronomical Society*, pp. 968–+, May 1999. [2.11](#)
- [53] G. B. Berriman, J. C. Good, and C. J. Lonsdale, "The Infrared Science Archive (IRSA) at IPAC: Moving Towards the NVO," in *Bulletin of the American Astronomical Society*, vol. 32 of *Bulletin of the American Astronomical Society*, pp. 1601–+, Dec. 2000. [2.11](#)
- [54] A. Richmond and N. White, "HEASARC. The design and architecture of an astrophysics information system.," in *Bulletin of the American Astronomical Society*, vol. 26 of *Bulletin of the American Astronomical Society*, pp. 995–998, May 1994. [2.11](#)
- [55] I. Foster and C. Kesselman, *Computational Grids*, ch. 2. Morgan-Kaufman, 1999. [2.12](#)
- [56] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets," *Journal of Network and Computer Applications*, vol. 23, pp. 187–200, 2001. [2.12](#)
- [57] M. Graham, D. Morris, and G. Rixon, "VOspace specification," Proposed Recommendation Version 1.14, International Virtual Observatory Alliance, October 2008. [2.12](#)
- [58] P. Z. Kunszt, A. S. Szalay, and A. R. Thakar, "The Hierarchical Triangular Mesh," in *Mining the Sky* (A. J. Banday, S. Zaroubi, and M. Bartelmann, eds.), pp. 631–+, 2001. [2.12](#)
- [59] S. Koposov and O. Bartunov, "Q3C, Quad Tree Cube – The new Sky-indexing Concept for Huge Astronomical Catalogues and its Realization for Main Astronomical Queries (Cone Search and Xmatch) in Open Source Database PostgreSQL," in *Astronomical Data Analysis Software and Systems XV* (C. Gabriel, C. Arviset, D. Ponz, and E. Solano, eds.), vol. 351 of *Astronomical Society of the Pacific Conference Series*, pp. 735–+, July 2006. [2.12](#)
- [60] P. Padovani, M. G. Allen, P. Rosati, and N. A. Walton, "Discovery of optically faint obscured quasars with Virtual Observatory tools," *Astronomy & Astrophysics*, vol. 424, pp. 545–559, Sept. 2004. [2.13](#)
- [61] P. F. Ortiz, F. Ochsenbein, A. Wicenec, and M. Albrecht, "ESO/CDS Data-mining Tool Development Project," in *Astronomical Data Analysis Software and Systems VIII* (D. M. Mehringer, R. L. Plante, and D. A.

- Roberts, eds.), vol. 172 of *Astronomical Society of the Pacific Conference Series*, pp. 379–+, 1999. 4.2
- [62] S. Derriere, A. J. G. Gray, N. Gray, F. V. Hessman, T. Linde, A. Preite Martinez, R. Seaman, and B. Thomas, “Vocabularies in the Virtual Observatory,” IVOA Proposed Recommendation Version 1.13, International Virtual Observatory Alliance, July 2008. 4.2, 12.2
- [63] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1998. 4.4
- [64] P. P.-S. Chen, “The entity-relationship model—toward a unified view of data,” *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 9–36, 1976. 4.4
- [65] J. McDowell, F. Bonnarel, D. Giaretta, G. Lemson, M. Louys, and A. Miccol, “Data Model for Observation,” IVOA Data Modelling WG Internal Draft v0.23, International Virtual Observatory Alliance, April 2005. 4.5, 4.1, 4.5, 5.8, 13.6, 13.6
- [66] A. Rots, “Space-Time Coordinate metadata for the Virtual Observatory,” IVOA Recommendation 1.33, International Virtual Observatory Alliance, October 2007. 4.5, 4.5, 4.4, C.3
- [67] IVOA Data Modelling WG, “Data Model for Astronomical DataSet Characterisation,” IVOA Recommendation v1.13, International Virtual Observatory Alliance, March 2008. 4.5, 4.2, 4.5, 4.3, 5.4, 12.2, 13.6, 13.6
- [68] P. Lamb and R. Power, “IVOA Data model for raw radio telescope data,” *IVOA Radio Astronomy Interest Group Note for Discussion*, October 2003. 4.6, 6, 7.1, 9.3
- [69] T. Murphy, P. Lamb, C. Owen, and M. Marquarding, “Data storage, processing and visualisation for the ATCA,” *ArXiv Astrophysics e-prints*, Enero 2006. 4.6, 7.1
- [70] P. Warner, “NOAO Science Archive - Domain Model,” tech. rep., National Optical Astronomy Observatory, 2004. 4.6
- [71] F. Viallefond, “The Alma Science Data Model,” in *Astronomical Data Analysis Software and Systems XV* (C. Gabriel, C. Arviset, D. Ponz, and S. Enrique, eds.), vol. 351 of *Astronomical Society of the Pacific Conference Series*, pp. 627–+, July 2006. 4.6
- [72] J. Bormans and K. Hill, “MPEG-21 Overview,” Tech. Rep. v.5, ISO/IEC JTC1/SC29/WG11/N5231, October 2002. 4.6, 7.3
- [73] J. M. Martínez, “MPEG-7 Overview,” tech. rep., ISO/IEC JTC1/SC29/WG11, October 2004. 4.6

- [74] R. J. Hanisch, W. D. Pence, B. M. Schlesinger, A. Farris, E. W. Greisen, P. J. Teuben, R. W. Thompson, and A. Warnock, "Definition of the Flexible Image Transport System (FITS)," Tech. Rep. NOST 100-2.0, NASA/Science Office of Standards and Technology (NOST), NASA Goddard Space Flight Center, Greenbelt MD 20771, USA, 1999. [6](#)
- [75] R. M. Prestage and M. H. Clark, "Device and Log FITS Files for the GBT," tech. rep., NRAO Green Bank, December 2004. [6](#), [7.1](#)
- [76] A. Preite Martinez, S. Derriere, N. Gray, R. Mann, J. McDowell, T. McGlynn, F. Ochsenbein, P. Osuna, G. Rixon, and R. Williams, "The UCD1+ controlled vocabulary," IVOA Proposed Recommendation Version 1.11, International Virtual Observatory Alliance, December 2005. [6](#)
- [77] J. Schwarz and R. Heald, "Software glossary," Tech. Rep. Version 0.2, Atacama Large Millimeter Array, May 2003. [6.1](#)
- [78] A. Rots, "Space-Time Coordinate metadata for the Virtual Observatory," *IVOA Proposed Recommendation*, March 2005. [6.1](#)
- [79] J. McDowell, D. Tody, T. Budavari, M. Dolensky, F. Valdés, P. Protopapas, and A. Rots, "IVOA Spectral Data Model," *IVOA Data Access Layer WG Working Draft*, May 2006. [6.2](#)
- [80] R. Hanisch, G. Greene, A. Linde, R. Plante, A. M. S. Richards, E. Auden, K. T. Noddle, and W. O'Mullane, "Resource metadata for the Virtual Observatory," in *Astronomical Data Analysis Software and Systems XIII* (F. Ochsenbein, M. G. Allen, and D. Egret, eds.), vol. 314 of *ASP Conference Series*, pp. 273–+, 2004. [6.2](#), [7.1](#)
- [81] R. Hanisch, IVOA Resource Registry WG, and NVO Metadata WG, "Resource Metadata for the Virtual Observatory," IVOA Recommendation Version 1.12, International Virtual Observatory Alliance, March 2007. [7.1](#)
- [82] Y. L. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-Science," *SIGMOD Record*, vol. 34, pp. 31–36, septiembere 2005. [8.1](#), [8.1](#)
- [83] P. Buneman, S. Khanna, and T. Wang-Chiew, "Why and Where: A Characterization of Data Provenance," in *Proceedings of the 8th International Conference on Database Theory*, pp. 316–330, 2001. [8.1](#)
- [84] J. R. Pardo, J. Cernicharo, and E. Serabyn, "Atmospheric Transmission at Microwaves (ATM): an improved model for millimeter/submillimeter applications," *IEEE Transactions On Antennas And Propagation*, vol. 49, pp. 1683–1694, December 2001. [9.2](#)

- [85] M. B. Taylor, T. Boch, M. Fitzpatrick, A. Allan, L. Paioro, J. Taylor, and D. Tody, "Simple Application Messaging Protocol," IVOA Recommendation 1.11, International Virtual Observatory Alliance, April 2009. [10.1](#), [10.2](#), [11.1](#), [11.4](#)
- [86] D. Winer, "XML-RPC Specification," tech. rep., XML-RPC.com, 1999. [10.1](#)
- [87] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, 2003. [10.2](#), [10.3](#), [10.5](#)
- [88] T. Boch, M. Comparato, J. D. Taylor, M. B. Taylor, and N. Winstanley, "PLASTIC—a protocol for desktop application interoperability," IVOA Note v1.0, International Virtual Observatory Alliance, June 2006. [10.2](#)
- [89] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Internet Engineering Task Force, Mar. 1997. [11](#)
- [90] I. de Gregorio Monsalvo, *Radio Astronomical Study of the Physical Conditions, Kinematics, and Chemistry of the Environment Surrounding Low-Mass Young Stellar Objects*. PhD thesis, Universidad Autónoma de Madrid, Facultad de Ciencias Físicas, Departamento de Física Teórica, May 2006. [12.1](#)
- [91] K. Rohlf and T. L. Wilson, *Tools of radio astronomy*. Astronomy & Astrophysics Library, Springer-Verlag, Berlin, West Germany, enlarged fourth edition ed., 2004. [2](#), [3](#)
- [92] H. Ungerechts, "paKo—Observer's User Interface to the NCS at the 30-m Telescope," User Manual v 1.0.9, Institute de Radio Astronomie Millimetrique, Avenida Divina Pastora, Local 20, 18012 Granada, Spain, October 2007. [12.2](#), [12.2](#), [12.2](#)
- [93] W. Brunswig, H. Ungerechts, J. Penalver, and Sievers, "Synchronization of Switching Modes in the NCS," Tech. Rep. v1.1, Institute de Radio Astronomie Millimetrique, February 2002. [12.2](#)
- [94] W. Brunswig and H. Ungerechts, "Messages in the ncs," Tech. Rep. v1.6, Institute de Radio Astronomie Millimetrique, January 2004. [12.2](#)
- [95] P. Hily-Blant and H. Ungerechts, "Ncs 30m: Study—feasibility of new observing modes (om)," Tech. Rep. v1.2, Institute de Radio Astronomie Millimetrique, June 2001. [12.2](#)
- [96] A. Perrigouard, "NCS 30m Antenna Mount Drive Control," Tech. Rep. IRAM-COMP-007, v2.4, Institute de Radio Astronomie Millimetrique, November 2005. [12.2](#)

- [97] H. Ungerechts, W. Brunswig, and A. Sievers, "New Control System for the 30m Telescope: System Architecture: Overview," Tech. Rep. v1.3, Institute de Radio Astronomie Millimetrique, November 2002. [12.2](#)
- [98] J. Penalver, U. Lisenfeld, and R. Mauersberger, "Pointing with the IRAM 30-m telescope," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (H. R. Butcher, ed.), vol. 4015 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 632–640, July 2000. [12.2](#)
- [99] R. M. Shobbrook and R. R. Shobbrook, *The Astronomy Thesaurus*. 1993. [12.2](#)
- [100] J. McDowell, F. Bonnarel, I. Chilingarian, M. Louys, A. Micol, and A. Richards, "Data Model for Astronomical DataSet Characterisation," IVOA Working Draft v1.1, International Virtual Observatory Alliance, May 2007. [A.1](#)
- [101] R. R. White, A. Allan, S. Barthelmy, J. Bloom, M. Graham, F. V. Hessman, S. Marka, A. Rots, K. Scholberg, R. Seaman, C. Stoughton, W. T. Vestrand, R. Williams, and P. R. Wozniak, "Astronomical network event and observation notification," *Astronomische Nachrichten*, vol. 327, pp. 775–+, Sept. 2006. [A.1](#)
- [102] I. Ortiz, J. Lusted, P. Dowler, A. S. Szalay, Y. Shirasaki, M. Nieto-Santisteban, W. O'Mullane, P. Osuna, VOQL-TEG, and VOQL-WG, "IVOA Astronomical Data Query Language," IVOA Recommendation 2.00, International Virtual Observatory Alliance, VO Query Language Working Group, October 2008. [A.1](#)
- [103] F. Ochsenbein, R. Williams, C. Davenhall, D. Durand, P. Fernique, D. Giaretta, R. Hanisch, T. McGlynn, A. S. Szalay, M. B. Taylor, and A. Wicenec, "VOTable Format Definition," Proposed Recommendation Version 1.1, International Virtual Observatory Alliance, August 2004. [A.1, C](#)
- [104] C. Rodrigo, R. Gutiérrez, E. Solano, and M. Cerviño, "Theoretical models in the Virtual Observatory," *ArXiv e-prints*, vol. 711, Nov. 2007. [A.2](#)
- [105] R. J. Hanisch, C. Arviset, F. Genova, and B. Rino, "IVOA Document Standards," IVOA Proposed Recommendation Version 1.2, International Virtual Observatory Alliance, March 2009. [A.4](#)
- [106] E. R. Harold and W. S. Means, *XML in a nutshell*. O'Reilly & Associates, 2nd ed., June 2002. [B](#)