AstroGrid-D

Deliverable

A Broker for a Robotic Telescope Network[1]

Deliverable	5.5
Authors	Thomas Röblitz
Editors	Thomas Röblitz
Date	2008-05-02 15:27:58 +0200 (Fri, 02 May 2008)
Document Version	1.0.1
Current Version	1.0.1
Previous Versions	0.1.0, 0.2.0, 0.3.0, 0.4.0, 0.5.0, 0.5.1, 0.6.0, 0.6.1

A: Status of this Document
Published.

B: Reference to project plan
This document refers to task TA V-3 "'Entwicklung der Anbindung an robotische Teleskope"'. In particular, it documents the approach of the broker for observation requests considering static information only.

C: Abstract
Robotic telescopes are described in the *Remote Telescope Markup Language* (RTML), a XML dialect which may be converted to RDF and thereby stored in Stellaris, the AstroGrid-D's information

[1]This work is part of the AstroGrid-D project and D-Grid. The project is funded by the German Federal Ministry of Education and Resarch (BMBF).

service. Given the rather technical description of telescopes in RTML and potential observation requests – *Which telescope may observe galaxy X?* – it became clear, that the broker should be based on well-known reasoning techniques. Therefore, we have chosen *cwm* as the tool to match observation requests with robotic telescopes. Cwm is a rule interpreter based on *Notation 3*, which is a representation of RDF, but adding capabilities for expressing rules. In some sense, Notation 3 can be compared to the combination of RDF and SPARQL. Both, Notation 3 and cwm are developements of the World Wide Web Consortium (W3C).

D: Changes History

Version	Date	Name	Brief summary
0.1.0	2008-02-18		Initial draft
0.2.0	2008-02-25		Sections 2 and 3
0.3.0	2008-03-15		Sections 4 and 5
0.4.0	2008-03-15		Section 6
0.5.0	2008-03-16		Conclusion and outlook
0.5.1	2008-03-16		Schema URI fixed
0.6.0	2008-04-10		Added criteria for pixel sampling
0.6.1	2008-04-10		otm package version fixed
1.0.0	2008-05-02		Public release
1.0.1	2008-05-02		Title updated (Frank Breitling)

E:

Contents

1 Introduction

Robotic telescopes are described in the *Remote Telescope Markup Language* (RTML) [5, 4], a XML dialect which may be converted [3] to the *Resource Description Format* (RDF) and thereby stored in Stellaris [8, 6, 7], the AstroGrid-D's information service. Given the rather technical description of telescopes in RTML and potential observation requests – *Which telescope may observe galaxy X?* – it became clear, that the broker should be based on well-known reasoning techniques compared to the classical matchmaking provided by Condor ClassAds [9]. Therefore, we have chosen *cwm* [2] as the tool to match observation requests with robotic telescopes. Cwm is a rule interpreter based on *Notation 3* [1], which is a representation of RDF, but adding capabilities for expressing inference rules. In some sense, Notation 3 (N3) can be compared to the combination of RDF and SPARQL. Both, Notation 3 and cwm are developements of the World Wide Web Consortium (W3C).

Our approach builds on key technologies of the semantic web and logic programming. Given a database of facts about robotic telescopes and observation requests, these technologies allow to answer questions like "‘Which telescopes are capable of executing a certain observation request?"’ The matching mechanism is defined through a set of *rules* which iteratively generate new facts until the question is answered. Hence, the main objective of this document will be to introduce which type of rules are implemented and how they may be extended to support additional concepts (e.g., new features of telescopes).

Outline. In Section 2, we describe the general mechanism for brokering observation requests. Thereafter, we present a simple schema for specifying abstract observation requests in Section 3. The core of the broker – the inference rules – is defined in Section 4. Particularly, we show which rules currently exist and how rules may be added to cope with new features of telescopes or requests. In Section 4, we introduce command-line tools to be used for brokering observation requests. The steps necessary for setting up the software environment and obtaining input data, especially the static descriptions of robotic telescopes, are described in Section 6. We conclude in Section 7.

2 General Mechanism for Brokering Observation Requests

Fig. 1 shows the general mechanism for brokering observation requests. First, the observation request is described in a generic way using Notation 3 triples. Thereafter, the broker obtains information about the status of the robotic telescopes from an information service, e.g., Stellaris[2]. If the information is static it may be retrieved as first step, too. Next, an inference engine applies various rule to generate new facts in form of Notation 3 triples. Eventually, the results are post-processed for simple use with other components, e.g., a user interface.

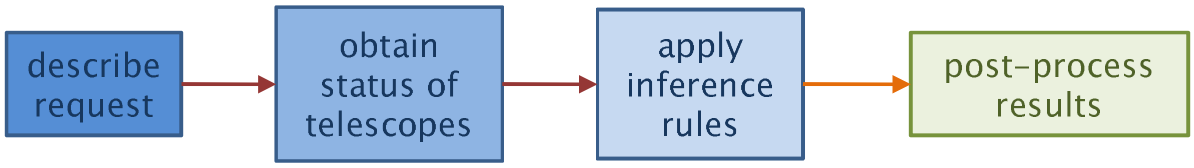

Figure 1: General mechanism for brokering observation requests.

3 Schema for Describing Abstract Observation Requests

The RTML allows to describe specific observation requests tied to a certain pre-selected telescope. However, it is not suitable for defining an observation request which translates into a query for finding matching telescopes. Hence, we developed a very basic schema for specifying *abstract observation requests*. For the sake of simplicity, the schema is based on Notation 3. Thus, it can directly be used by the broker without any conversion.

The schema uses the namespace `http://is.astrogrid-d.org/2008/02/14/opentelrequest` as defined by the triple

```
@prefix OTR: <http://is.astrogrid-d.org/2008/02/14/opentelrequest#>.
```

An abstract request is declared as a request by the statement

```
:req a OTR:REQ .
```

A request can be specified by several statements of the form

<request URI> <attribute> <value> .

Table 1: Supported attributes of the schema for describing abstract observation requests.

Attribute	Values	Description
`MinLatitude`	any number	Minimal value of the telescope's latitude
`MaxLatitude`	any number	Maximal value of the telescope's latitude
`MinLongitude`	any number	Minimal value of the telescope's longitude
`MaxLongitude`	any number	Maximal value of the telescope's longitude
`MinAltitude`	any number	Minimal value of the telescope's altitude
`MaxAltitude`	any number	Maximal value of the telescope's altitude
`MinGeoAperture`	any number	Minimal value of the telescope's camera geometric aperture
`MaxGeoAperture`	any number	Maximal value of the telescope's camera geometric aperture
`MinEffAperture`	any number	Minimal value of the telescope's camera effective aperture
`MaxEffAperture`	any number	Maximal value of the telescope's camera effective aperture
`Filter`	filter description	a subgraph containing the attributes `FilterName` and `FilterType`
`FilterName`	any string	name of the filter
`FilterType`	any string	type of the filter
`MinPixelSampling`	any number	Minimal value of the telescope's detector sampling property
`MaxPixelSampling`	any number	Maximal value of the telescope's detector sampling property

Currently, the schema supports 13 attributes listed in Table 1. The `Min/Max` attributes may be used to define range constraints on the values of certain properties of a robotic telescopes. The `Filter` attribute is satisfied if a telescopes features a filter with the name and/or type exactly matching the definition in the request.

For example, the request

```
@prefix OTR: <http://is.astrogrid-d.org/2008/02/14/opentelrequest#>.

:req a OTR:REQ .
:req OTR:MinLatitude 0 .
:req OTR:MaxLatitude 60 .
:req OTR:MinLongitude -100 .
:req OTR:MaxLongitude 12 .
:req OTR:MinAltitude 1200 .
:req OTR:MaxAltitude 3500 .
:req OTR:MinGeoAperture 0.6 .
:req OTR:MaxGeoAperture 1.3 .
:req OTR:Filter [ OTR:FilterName "B"; OTR:FilterType "Bessel_B" ] .
```

matches all telescopes with a longitude in [0,60], a latitude in [-100,12], an altitude from 1200 to 3500 meters, a camera geometric aperture from 0.6 to 1.3 and a filter of name `B` and type `Bessel_B`.

4 Inference Rules for Brokering Observation Requests

Inference rules have the following structure

$$\mathbf{Q}_1 x_1, \ldots, \mathbf{Q}_n x_n : \bigwedge_j c_j(x_1, \ldots, x_n) \quad \longmapsto \quad t_1(x_1, \ldots, x_n), \ldots, t_k(x_1, \ldots, x_n)\,.$$

The $\mathbf{Q}_i$ are predicate logic quantifiers ($\forall$ or $\exists$) of variables x_i $(i = 1, \ldots, n)$. The terms $c_j(x_1, \ldots, x_n)$ are patterns – more precisely triples – which shall match facts in the information storage (e.g., RDF triples of Stellaris). If the premise is satisfied, the conclusions are added as (new) facts to the information storage.

We distinguish three types of rules: *request* rules, *match* rules and *satisfaction* rules. Request rules derive knowledge about the requested features of a request. In general, they have the form

`{ ?req OTR:`*attr* `?x } => { ?req :requires :feat`*akey* `} .`

The attribute *attr* is one of those listed in Table 1. The attribute short name *akey* is one of `Lat`, `Long`, `Alt`, `Ape` and `Cam`.

The match rules are specifically defined for each attribute. Typically, the premise of a match rule consists of a part to address a certain telescope's feature (such as its geographical location), a part to address the corresponding request's specification (for example its latitude) and a matching condition (the latitude is within the requested range). The addressing is done by defining multiple

[2] `http://stellaris.zib.de/`

triples which specify a path from the root node of description to the feature in question (cf. series of ?res_a, ?res_b, ...). For example, the following inference rule tests the latitude of a telescope.

```
{
  ?res_a a <http://www.rtml.org/v3.2#RTML> .
  ?res_a <http://www.rtml.org/v3.2#uid> ?res_b .
  ?res_a <http://www.rtml.org/v3.2#Telescope> ?res_c .
  ?res_c <http://www.rtml.org/v3.2#Location> ?res_d .
  ?res_d <http://www.rtml.org/v3.2#Latitude> ?res_e .
  ?res_e x2r1:value ?res_x .

  ?req_a a OTR:REQ .
  ?req_a OTR:MinLatitude ?req_b .
  ?req_a OTR:MaxLatitude ?req_c .

  ?res_x math:notLessThan ?req_b .
  ?res_x math:notGreaterThan ?req_c
}
=>
{ ?res_a :matches :featLat } .
```

Finally, a satisfaction rule evaluates if a telescope matches all required features. Particularly, it builds upon the facts generated by the request and match rules. Depending on the request – i.e., which features are given, a rule specific to this request is created. The following shows a rule of a request which only specifies information about the camera of the telescope.

```
{
  ?res :matches :featCam .
  ?req :requires :featCam .
}
=>
{ ?res :sats ?req }.
```

Request rules are contained in the file `${OTM_ROOT}/rules/request.n3`. The match rules are defined in the file `${OTM_ROOT}/rules/match.n3`. The file `${OTM_ROOT}/rules/satis.n3.in` contains the template of the satisfaction rules.

4.1 Extending the Set of Attributes

Adding new attributes requires the following steps:

- augment the *request* rules and define an appropriate name for the feature belonging to the attribute, *and*
- define *match* rules for determining which telescopes satisfy the additional requirement.

5 Command-line Tools Encapsulating the General Mechanism

The general mechanism is encapsulated into a single command-line tool `otm` – the **O**pen**T**el **B**roker. Its complete path is `${OTM_ROOT}/bin/otm`. It can be invoked without any parameters. In that case, it searches the current working directory for telescope descriptions in files with names `OT_`*string*`.n3` and a request description in `req.n3`. The telescopes' file name can be customized with any string *string*. Currently, the broker supports the single option `-req` *filename* to let it use a different file containing the request. The output of the broker either is a numbered list giving the names of the telescopes matching the request or the string `no telescope matches request`.

The following shows two examples for calling the `otm`.

```
> otm
1 -- <http://is.astrogrid-d.org/context/OpenTel/Amadeus.aip.de/RTML>
2 -- <http://is.astrogrid-d.org/context/OpenTel/MONET.Uni-Goettingen.de/RTML>
3 -- <http://is.astrogrid-d.org/context/OpenTel/STELLA-I.aip.de/STELLA-I.rtml>
4 -- <http://is.astrogrid-d.org/context/OpenTel/STELLA-II.aip.de/STELLA-II.rtml>
5 -- <http://is.astrogrid-d.org/context/OpenTel/Wolfgang.aip.de/RTML>

> otm -req req9.n3
no telescope matches request
```

While the request file is typically edited manually or through some GUI, the telescope descriptions are likely to be downloaded from an appropriate instance of Stellaris and saved in the format N3. The latter will be covered in the following section.

6 Preparing the Software Environment and Input Data

The broker uses `cwm` as inference engine which in turn requires `Python 2.4` or higher with support for handling XML documents. The latest version of `cwm` can be downloaded from [2]. At the time of writing the current version was 1.2.1. We installed `cwm` with the proposed procedure (see `README` file of the cwm package) under the directory `/usr/share/opentel`.

Next, we describe the installation of the package `otm` version 1.1.0. Unpack the package and change to its main directory, e.g.,

```
> tar xzf otm-1.1.0.tgz
> cd otm-1.1.0
```

Run the script `install.sh` which installs the package under `/usr/share/opentel` (default) or the directory given as first parameter, e.g.,

```
> sh install.sh
```

or

```
> sh install.sh /my/root/dir/for/opentel
```

Add the path to the executables – e.g., `/usr/share/opentel/bin` or `/my/root/dir/for/opentel/bin` to the PATH environment variable. For example (bash shell),

```
> export PATH="${PATH}:/usr/share/opentel/bin"
```

6.1 Providing Static Telescope Descriptions

As described in the previous section, the broker searches the current directory for files describing telescopes. Since these descriptions are considered to be static, it is sufficient to download them manually. For example, the introspection interface of Stellaris can be used to find telescope descriptions, download and store them as N3 documents. Alternatively, well-known tools such as `wget` may be used to automate the provision of the telescope descriptions.

7 Conclusion and Outlook

We described our approach for matching abstract observation requests against descriptions of robotic telescopes. Because the latter descriptions are available in RDF (and N3), we choose the inference engine *cwm*, developed by the W3C, for the matchmaking step. Although the current set of rules is very basic, we believe that it may be easily extended to support future requirements – i.e., other criteria for matchmaking.

F: References / Bibliography

References

[1] Tim Berners-Lee. Notation 3 (n3) a readable rdf syntax. http://www.w3.org/DesignIssues/Notation3.html, March 2006.

[2] Tim Berners-Lee. cwm - a general purpose data processor for the semantic web. http://www.w3.org/2000/10/swap/doc/cwm, February 2008.

[3] F. Breitling. Providing Static Metadata of Robotic Telescopes to Stellaris. Technical Report D2.4, AstroGrid-D project, In preparation.

[4] Frederic V. Hessman. Rtml - remote telescope markup language. http://www.uni-sw.gwdg.de/~hessman/RTML/, November 2005.

[5] F.V. Hessman, C. Pennypacker, E. Romero-Colmenero, and G. Tuparev. Telescope networking and user support via remote telescope markup language. In *Advanced Software, Control and Communication Systems for Astronomy, SPIE*, pages 290–301, 2004.

[6] M. Högqvist. Stellaris: The AstroGrid-D Information Service. Technical Report D2.2, AstroGrid-D project, January 2007.

[7] M. Högqvist. Stellaris: The AstroGrid-D Information Service (2nd Version). Technical Report D2.5, AstroGrid-D project, January 2008.

[8] M. Högqvist, T. Röblitz. AstroGrid-D Information Service Requirements Specification and Architectural Design. Technical Report D2.1, AstroGrid-D project, July 2006.

[9] Rajesh Raman, Miron Livny, and Marvin Solomon. Matchmaking: Distributed resource management for high throughput computing. In *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing, Chicago, Illinois, USA*, pages 140–146. IEEE Computer Society Press, July 1998.