The Lifecycle of an Astronomy Experiment - From a Description to the Result

1   Scenario and Problem Description

Within our scenario we try to control a telescope in Santiago de Chile remotely. To achieve this, a quite rich description of an astronomy experiment is needed which can be used by a program running on a computer that has access to the telescope to control the telescope and receive the results of the given experiment.

Currently we are facing several problems during the communication with the telescope:

- The adjustment of the telescope is still a problem since coordinates given to the telescope are not reflected correctly.

- The format of the images received from the telescope is still unclear and the decryption of the image is still a problem.

After solving this more or less technical problems, we need to specify how we plan to describe an experiment that we want to execute with a telescope. The next sections will provide first ideas for such a description by describing a possible lifecycle of such an experiment and by the suggestion of a DTD (Document Type Definition) for XML files. Since in the last section we will provide a possible architecture for this kind of scenarios based on web services XML is a very powerful way for describing those experiments.

2   The Lifecycle of an Astronomy Experiment in this Scenario

Within our scenario we need to stick to a certain lifecycle of an experiment to allow to build the software infrastructure for the experiments. This lifecycle will include the experiment description, the execution of the experiment, possible processing of the result and returning the result to the user. Figure 1 shows a proposed lifecycle which includes all the above mentioned steps.

The lifecycle usually starts with a description of the experiment given in XML. This description should include all the necessary information beginning from the position of the phenomena one wants to explore up to the format in which the result is propagated back to the user.

Afterwards this XML description of the experiment could be sent to the computer that is in charge of controlling the telescope. A service running on that computer can receive the XML file, parse it and than adjust all the necessary settings to perform this experiment.

Figure 1: A possible lifecycle of an astronomy experiment

After receiving the images from the telescope the service can process the received data if for this experiment image processing is necessary. Last but not least the data could be sent back to the user either way if it is in a processed format or not.

3 The Description of an Astronomy Experiment

To allow the description of an astronomy experiment in XML it is very natural to create a DTD file that describes the format of such an XML file. Additionally a DTD file of this kind allows later-on to validate the XML description files against a certain grammar defined in the DTD file.
An example of a DTD file that allows a quite rich description of an astronomy experiment could be like this:

```
<!ELEMENT AstroExperiment (Coordinates, ExposureTime,
                           Spectrum, ResultFormat,
                           Processing*)>
<!ELEMENT Coordinates EMPTY>
<!ATTLIST Coordinates northSouth PCDATA #REQUIRED
                      eastWest   PCDATA #REQUIRED
                      following  (true|false) "true">
```

```
<!ELEMENT ExposureTime EMPTY>
<!ATTLIST ExposureTime time              PCDATA #REQUIRED
                       timeBetweenImages PCDATA #REQUIRED>
<!ELEMENT Spectrum (ANY)>
<!ELEMENT ResultFormat (ANY)>
<!ELEMENT Processing (ANY)>
```

This description allows to define astronomy experiments by describing the *Coordinates* at which the phenomena takes place, the time when the phenoma will happen which is called *ExposureTime*, the *Spectrum* which should be used for the observation of the phenomena, the *ResultFormat* which describes the format in which the results of the telescope should be reflected to the user and last but not least it is possible to define *Processing* steps which should be performed on the data sent by the telescope.

The root element of a valid XML file concerning this DTD is an *AstroExperiment* element which has the coordinates, the exposure time, the spectrum, the description of the result format and the possible processing steps as children. Within the *Coordinates* element we can define the exact position of the phenomena as attributes and whether the telescope should follow the phenomena, so whether it has to adjust itself during the experiment. Furthermore this element has no child elements. The same goes for the *ExposureTime* element which has as attributes the exact time when the phenomena should happen and as a second argument the time that should be waited between two images. The *Spectrum* element has no attributes but may accept all kinds of XML subtrees as children just to provide a very free way for the description of the spectrum. It is also possible to define the format of the *Spectrum* element much closer but therefore more input about the different spectra is necessary. Almost the same goes for the *ResultFormat* element and the *Processing* element which are, up to now, not defined at all but maybe defined in the future.

4 Future Work and Outlook

For future work we should focus on the definition of the spectrum, result format and processing descriptions. Furthermore should the whole lifecycle be reflected in webservices running on the machine that is capable of controlling the telescope. In more detail, there must be at least two different methods available through a web service:

- A method for sending the XML description of the experiment.

- A method for receiving the result of the experiment, this could usually not be passed back as the return value of the above mentioned method just to allow asynchronous execution of experiments.

The usage of web services at this point in the scenario makes sense since we consider different kinds of clients to use the service of the telescope, e.g. web applications, a Cool Modes reference frame for astronomy experiments or stand-alone applications.