A Semantic Data Grid for Satellite Mission Quality Analysis

Reuben Wright[1], Manuel Sánchez-Gestido[1], Asunción Gómez-Pérez[2], María S. Pérez-Hernández[2], Rafael González-Cabero[2], Oscar Corcho[3]

[1] Deimos Space, Ronda de Poniente, 19, 28760 Tres Cantos (Madrid), Spain
{reuben.wright, manuel.sanchez}@deimos-space.com
[2] Facultad de Informática, Universidad Politécnica de Madrid, Spain
{asun, mperez}@fi.upm.es, rgonza@delicias.dia.fi.upm.es
[3] School of Computer Science, University of Manchester, England
oscar.corcho@manchester.ac.uk

Abstract. The use of Semantic Grid architecture eases the development of complex, flexible applications, in which several organisations are involved and where resources of diverse nature (data and computing elements) are shared. This is the situation in the Space domain, with an extensive and heterogeneous network of facilities and institutions. There is a strong need to share both data and computational resources for complex processing tasks. One such is monitoring and data analysis for Satellite Missions and this paper presents the Satellite Mission Grid, built in the OntoGrid project as an alternative to the current systems used. Flexibility, scalability, interoperability, extensibility and efficient development were the main advantages found in using a common framework for data sharing and creating a Semantic Data Grid.

Keywords: S-OGSA, WS-DAIOnt-RDF(S), OWL, Satellite Mission, Grid, Space Domain, CCSDS

1 Introduction

We describe here the development of a system to enable flexible monitoring and investigation of the operation of satellite missions. We look briefly at the industry and its general operation, then in more detail at the requirements for such a system. The technical issues encountered, and solved, are given in detail and then a summary of the key advantages of the semantic approach taken. The paper concludes by considering the next steps to be taken in uptake of this system, and semantic technologies more generally in the space domain.

2 Requirements

The scope was to replicate some features of an existing Satellite Mission Quality Analysis program and to demonstrate some extensions. The analysis centres on the comparison of planned activity against the production of data by the satellite and the

further processing of that data in ground systems. The features that interest us are in the specific files, but these files are great in number, and size, and the metadata is locked into implicit forms. The approach was to extract this metadata to build an explicit semantic representation of the data which allows both the existing analysis and analysis not yet possible with that system. We will describe in this section the industry, the existing system, and use cases which describe the behaviour of this semantic system.

Earth Observation Satellite Systems Earth Observation can be defined as the science of getting data from our planet by placing in orbit a Hardware/Software element with several observation instruments, whose main goal is to obtain measurements from the Earth surface or the atmosphere. These scientific data are sent to Ground Stations and then processed in order to get meaningful information.

The working of an Earth Observation Satellite System consists of a simple process repeated over time. The instruments on board the satellite act like cameras that can be programmed taking "pictures" (images) of specific parts of the Earth at predefined times. Parameters for the instrument operations and the general satellite configuration constitute the Mission Plans issued by the Mission Planning System which is sent to the Flight Operation Segment (FOS). This, in turn, sends equivalent information to a Ground Station and from there to the satellite antenna of the spacecraft.

A computer on board the satellite will store the list of MCMD (MacroCommands) that ask an instrument or any other part of the satellite to perform an action. These include loading a table, triggering an operation and getting internal status information.

Images from each of the instruments are stored onboard (in the satellite computer memory) as raw data and when the satellite over-flies the Ground station that data is sent to the Ground Station antenna (Data downlink). Conversion from the raw data to “products” is performed in a sequence, at the Ground Station and various Payload Data Segment facilities. These add such things as identification labels and geolocation data to each of the images. Fig. 1 shows graphically the overall scenario. A more detailed explanation of the whole system can be found in [1].

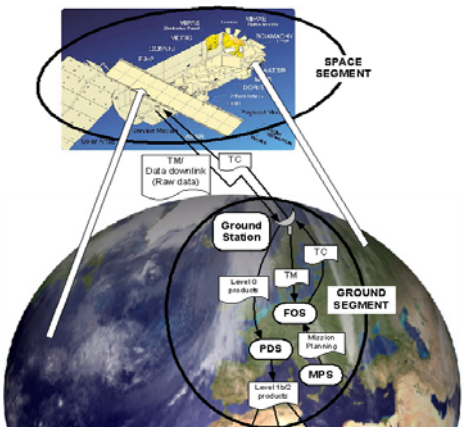

Fig. 1. General overview of an Earth Observation Satellite system (Envisat)

The Envisat satellite was launched in March 2002 with an ambitious, combined payload that ensures the continuity of the data measurements of the European Space Agency (ESA) ERS satellites. ENVISAT data supports Earth Science research and allows monitoring of the evolution of environmental and climatic changes. Furthermore, the data facilitates the development of operational and commercial applications. The list of instruments for this mission is composed of 10 instruments: RA-2, MWR, MIPAS, MERIS, GOMOS, ASAR, AATSR, DORIS, SCIAMACHY and LRR. Reference [2] contains extensive information describing the Envisat mission.

Mission planning for the instruments and for the Satellite operations is issued regularly, nominally on a weekly basis, and can be later modified, before it is frozen for sending to the satellite. If FOS receives requests for some instruments with High Rate modes (e.g. MERIS and ASAR) they have to be accommodated in the previous valid planning. A catastrophic event (earthquakes, volcanic eruptions, hurricanes, …) or a specific demand from the scientific community are examples of events that can cause last minute re-planning.

Existing System for Analysis of Satellite products: QUARC

Data circulates within the system as various Plan and Product Files, with well-defined structures. QUARC is a system that checks off-line the overall data circulation process and in particular the quality of the instrument product files. This process needs as input the product files, the MCMD and the mission planning, provided to the system by other facilities. Apart from the product files in the PDS it needs to retrieve information from other parts of the system to crosscheck that the planning has been transformed properly into MCMD´s, the instrument has performed successfully the measurements (taking images of the Earth), that these images have been stored onboard and transmitted as Raw Data to the Ground station and the processing from Raw Data to Level 0 and then to Level 1B and Level 2 was correct

QUARC returns reports and plots, which help in the production of new plans. Additionally, the QUARC system is designed to assist when taking decisions in the situation where an instrument or the whole system begins to malfunction and to detect, in a semi-automated fashion that something incorrect has occurred in one part of the product generation or data circulation.

The operational QUARC system is located in a single location (ESA-ESRIN, in Italy) that communicates with the archive containing all the products generated from the beginning of the mission and with all the other facilities. The Data Ingestion Modules, one per file type, read the files and convert their contents into parameters that are meaningful to the QUARC data model. The system has been built specifically for this purpose and has bespoke user interfaces. It took several years to build and there are significant ongoing maintenance and development costs as new reports are required and new missions are launched.

Use Cases

In addition to functional and non-functional requirements from the existing system we produced Use Cases to support incremental, distributed development. These translated directly into Test Cases for evaluation of the system.

Use Case 1: Instrument unavailability This is a Use Case to ensure our new system is capable of replicating the core functionalities of the existing system. A user needs to find out what planned events and generated products exist for a given time period and instrument, and to plot these results against each other in a timeline. A simple interface is needed, with no underlying complexity exposed to the user.

Use Case 2: Check for the quality of the products in Nominal mode Certain sorts of products have internal parameters giving a measure of quality of data. The specific situation for this use case at present would be extraction of one of these quality parameters, over a period of time, for an instrument in a particular mode, being "Nominal". The product files we were to work with didn't include this quality data so we expanded this to the more general requirement to be able to extract any piece of metadata from a set of product files.

Use Case 3: Update of functionalities with no software update A crucial perceived advantage of the semantic approach was the flexibility with which the system could be adapted. A mission may last 10/15 years and since we are largely investigating anomalous behaviour not all useful queries will be known ahead of time. We needed to know how easily we could develop new queries over our data, and parameterise them for use by ordinary users.

Use Case 4: Data lifecycle The satellite plans are not static and the system needed to be able to remove or update metadata from its stores. This needed to be done automatically, and only in the correct circumstances of a new plan covering the same time period and from the provider of the original plan. When querying, the user must be given information about the final, executed, plan.

Use Case 5: Modularity of metadata service The desire to be able to change the metadata store comes from wanting flexibility in extending the system. The approach was to design and build a loosely-coupled, service-orientated architecture. In particular we would ensure we could change the metadata store and query engine, but more generally we use modular components defined by their interfaces. Choices between components can be made on various characteristics including cost, scalability, reliability, and performance. Crucially the user shouldn't have to worry about implementation details.

3 Technical Issues and Solutions

The following diagram shows the geographical deployment and component breakdown of the developed system. Software was deployed at 3 sites – Manchester, Madrid and Athens, and Atlas further uses the Everlab cluster of machines throughout Europe. The number actions 1-5 and 6-8 show the activity flow for annotating and querying data respectively.

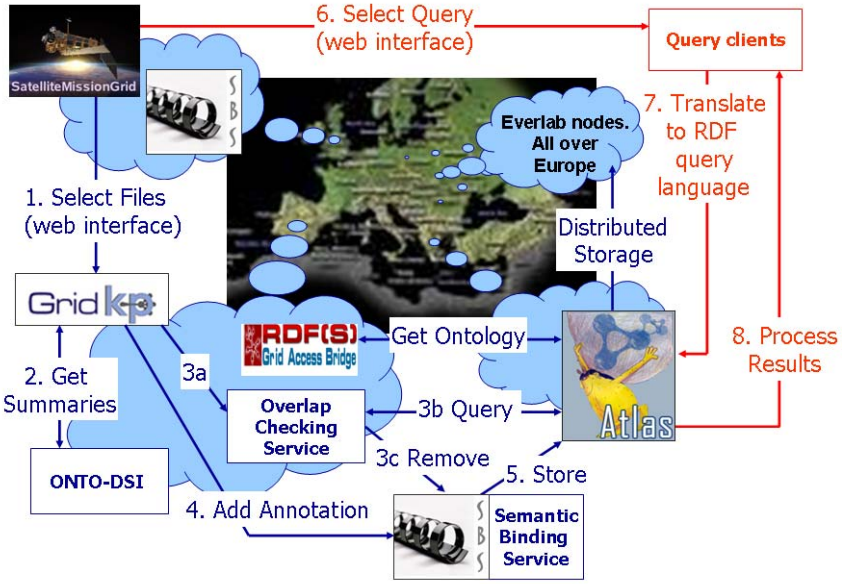

Fig. 2. System architecture

Infrastructure: a distributed, scalable architecture

S-OGSA [3] is the reference Semantic Grid architecture developed in Ontogrid and used for the Satellite Mission Grid. The basis of S-OGSA is an information model of semantic resources, which extends the OGSA model [4]. S-OGSA also anticipates that a set of capabilities will be required from Grid middleware in order to address the new services with varying degrees of semantic capabilities. For this, S-OGSA includes two service categories called Semantic Provisioning Services and Semantically Aware Grid Services. Below are details of the Information Model and the Service Categories, and their use in the Satellite Mission Grid.

S-OGSA Information Model The S-OGSA model identifies three types of entities:
Grid Entities - anything that carries an identity on the Grid, including resources and services [5]. In this system they include planning systems, planning files, satellite in-

struments, product files and product processing facilities. These entities would be considered similarly in any non-semantic implementation of the process.
Knowledge Entities - special types of Grid Entities that represent or could operate with some form of knowledge. Examples of Knowledge Entities are ontologies, rules, knowledge bases or even free text descriptions that encapsulate knowledge that can be shared. In this system we had a single ontology including classes for times, planning systems, macrocommands, satellite instruments, and the details of the various plan and product file metadata. Ultimately there needs to be a set of ontologies to cover the whole satellite mission domain. In the Satellite Mission Grid an annotation process creates knowledge entities (sets of RDF statements) for the different types of files.
Semantic Bindings - Knowledge Entities that represent the association of a Grid Entity with one or more Knowledge Entities (that is, they represent semantic metadata of a Grid Entity). Existence of such an association transforms the subject Grid entity into a Semantic Grid Entity. Semantic Bindings are first class citizens as they are modeled as Grid resources with an identity and manageability features as well as their own metadata. Grid Entities can acquire and discard associations with knowledge entities through their lifetime. In our system the files are made into Semantic Grid Entities by attaching the created annotations.

Semantic Provisioning Services These are Grid Services that provision semantic entities. Two major classes of services are identified:
Knowledge provisioning services - these can produce (and in some cases store and manage) Knowledge Entities. Examples of these services are ontology services and reasoning services. In this system the ontology services are implemented using RDF(S) Grid Access Bridge [6], an implementation of WS-DAIOnt [7].
Semantic Binding provisioning services - these can produce (and in some cases store and manage) Semantic Bindings. The system includes an annotation service that generates Semantic Bindings from planning and product files. This annotation service is implemented using the Grid-KP [8][9] tool. It also uses a storage service for the Semantic Bindings that are generated, so that they can be accessed at any time during their lifetime using a query language. This service [10] was implemented twice, once using the Atlas system [11][12] and once in Sesame [13]. These two services were “swappable” components.

Semantically Aware Grid Services This special type of Grid Services are able to exploit semantic technologies to consume Semantic Bindings in order to deliver their functionality [14]. Their role is complementary to the role of Semantic Provisioning Services since they consume the semantic entities held by Knowledge provisioning services and Semantic Binding provisioning services, and use their services. The user interface for the Satellite Mission Grid is a Semantically Aware Grid Service, making use of all the aforementioned elements in order to deliver its enhanced functionality.

Annotation: making metadata explicit

Data circulates in the existing systems as files with many common generic features. They are slightly different for planning and product files, and the information about these planned events and generated products is usually bound up with the data in-

volved. Standard ASCII formats encode the information in keyword-value pairs, which are stored as headers for the various files. This is a special format defined for the Envisat mission with an enormous amount of software and documentation generated through years of development. This structure can be simply translated to a fairly flat XML structure. Once this is performed on the planning and product files, the system uses XML software tools.

Product files consist of an ASCII header in the above format and a binary part encoded in an ESA proprietary format. This header is just a few Kbs out of an image file size of Gbs. The Onto-DSI [15] component was used to extract and provide just the headers from these files to avoid a large system overhead whilst annotating them.

Much of the metadata was encoded in specific, amalgamated identifiers. For example simple rules had to be created to process product filenames such as "RA2_MW__1PNPDK20060201_120535_000000062044_00424_20518_0349.N1". This is decomposed into an Event type (RA2_MW), Processing level (1P) and centre (PDK), a Sensing start time (2006-02-01:12.05.33) and so on. Generic metadata (applied across all captured metadata) and the ontology further add, for example, that the Event type (RA2_MW) is executed by a particular instrument, the Radar Altimeter. The ontology simplifies this procedure by making it a simple matter of a domain expert stating where the pieces of information are found. A parser extension was written in Grid-KP to carry out the extraction of the relevant properties from the files and this task was separated from other work such as the creation of query interfaces. While it was not used formally or automatically in this implementation, it is easy to force verification of the metadata against the ontology and this was done manually on occasion.

Another issue was conversion of units. One example of this was converting from date formats, as given above (and given to the users in the webforms) to another standard time format used in space missions, MJD2000. It is the number of seconds (and milliseconds) to have passed since the year 2000, including leap seconds. The conversion routine was wrapped as a webservice using SOAPLAB [16].

It is anticipated that migration of other data to the system would be much simplified by this process and these tools being in place. In addition the annotation services were deployed in different locations, which supported the distributed nature of the data sources.

Storage: managing a (meta)data lifecycle

The Annotation Service was able to use exactly the same mechanisms as the user interface to communicate with the Semantic Binding Service to ask if its current file overlapped with any existing Plan files. This design has two advantages; firstly, no new specific code needed to be written as we already had the query interfaces. Secondly, although the logic needed here was quite simple, we have allowed ourselves full access to the flexibility of RDF querying, which means that if more complex rules are needed in future we will be able to accurately encode them. Having established if an update was needed the RDF was updated (deleted and replaced), or not, using the standard mechanisms provided by the metadata stores.

Managing our RDF in identifiable, separate Semantic Bindings allows us to better manage the overlaps, and the lifetime of the metadata when several annotations may

be created. This also gives us confidence in the ability to incrementally migrate data into such a system.

Scalability was not explicitly investigated as part of this use case, but tests have been carried out separately on each of the separate components making up the system. This is the most sensible approach in this type of architecture, where components are combined without a need for large amounts of integration work.

Querying: exploring the data

Having created semantic bindings from our files to RDF statements about them, we had gained in flexibility in our querying. We used SPARQL and SeRQL when Sesame was the metadata store and RQL when Atlas was the store. The ability to choose which language we wanted was another reason for considering interchangeable metadata stores. We worked with a flexible "Free Querying" interface as we considered how the system would be incrementally improved and developed. This interface simply allowed the user to create queries (in the language of their choice) and get the results back in a tabular form.

As an example we looked at how we could abstract our queries over the data to a level where new sorts of data could be added to the system and still be identified by our queries. An initial implementation of one of the queries for Use Case 1 was looking for all planned events (DMOP event records) which were using the Radar Altimeter. We matched at the low level of Event identifiers using the implicit metadata that "events with identifiers containing RA are carried out by the Radar Altimeter instrument". The nature of RDF as a web of statements and the existence of an ontology to formalise the existence of different properties made it easy to move these queries to an improved, semantic level.

We were initially searching on the event_id property of the DMOP_er class (DMOP event records), which look like `"RA2_IE_00000000002372"`. It matches `REGEX(?EVENT_ID,".*RA.*")` in the SPARQL regular expression syntax. This query works, but we were able to see in the ontology that a better level was possible.

The individual data items about planned events use the event ids, but our system was able to augment that with the knowledge about which event types use which instruments. This was enabled by having an ontology which included instruments and types of events as objects independent of the individual events which they classify. The following diagram showing part of the Satellite Ontology shows that the `DMOP_er` class (top left) is related to the `Plan_Event` class by `represents_plan_event` property, and that `Plan_Event` instances have their own identifiers – `plan_event_id`. These look like `"RA2_CAL"` or `"RA2_IE"` and we could match them explicitly. They represent the different types of events that can be planned. We then looked one level further - moving up to the level of the `Instrument` class and see the Radar Altimeter is identified as one such, with `instrument_id` of `"RA"`.

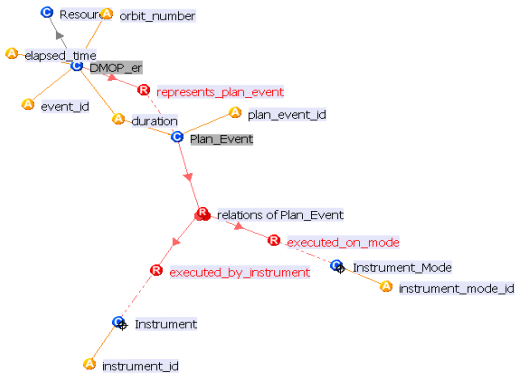

Fig. 3. A section of the Satellite Ontology showing the specific events (DMOP-er), their types (Plan_Event) and the Instruments which carry out those event types. Diagram from NeOn [17]

We moved in our SPARQL query from

```
?EVENT event_id ?EVENT_ID ;
FILTER ( REGEX(?EVENT_ID,".*RA.*"))
```

to

```
 ?EVENT event_id ?EVENT_ID ;
        represents_plan_event ?PLAN_EVENT_TYPE .
?PLAN_EVENT_TYPE executed_by_instrument ?INSTRUMENT .
     ?INSTRUMENT instrument_id "RA"
```

While this is longer it is both clearer to understand and to implement as a webform where the user will select an instrument. It is also likely to execute more quickly as it is looking for an exact match of `instrument_id` rather than having to rely on regular expression parsing of a string value.

The biggest gain is that it is much more robust in the face of changing data. We can continue to use these "semantic level" queries about instruments even if we add new event types which use this instrument or change the unique identifiers for individual DMOP event records. If further data in the system contained new sorts of events planned and carried out by the Radar Altimeter then our queries would automatically match them. In any of these extended cases a simple statement associates the new event type with an existing instrument or new events with an existing event type. The exact same query (for use of an instrument) will then also report about these new events. We shifted from talking about details of identifiers to the actual objects which the user is concerned about. i.e. we moved to a more semantic level. This process is shown in more detail in an OntoGrid demonstration video [18].

Accessibility: providing tools for end users

Having developed the queries in a language such as SPARQL we very easily built webforms to allow users to provide the values for any such queries. A web application (running on Tomcat) was developed to serve Java Server Pages which presented interfaces and results to users, and converted to RDF queries, and back from RDF results sets. The results were provided in a simple XML structure and we generated from that either tables or graphs.

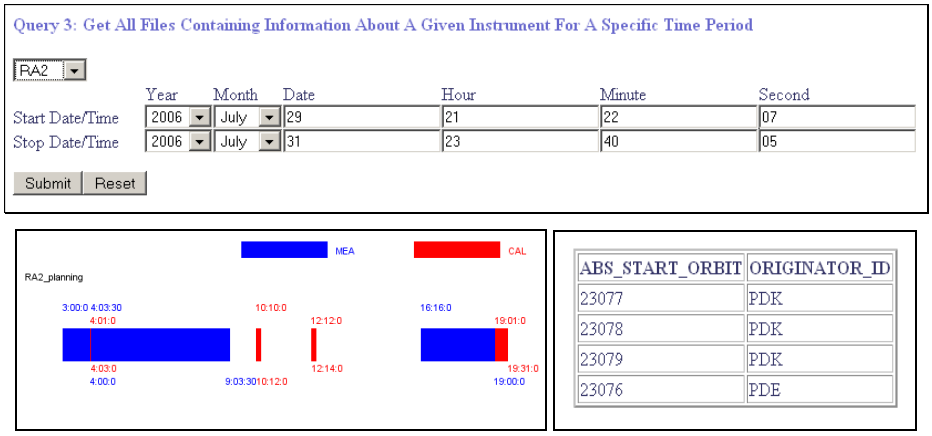

Fig. 4. Webform for user input and timeline and tabular outputs

4 Evaluation

The implementation shows several advantages with respect to a conventional approach, in terms of flexibility, reduction of software running costs, maintainability, expandability, and interoperability. A testing phase allowed us to evaluate in more detail the various aspects identified in the Use Cases.

Access to Data and Information

Use of Envisat data products or product related information is granted for the product metadata (in product headers) but not generally the science data itself, unless approved under the CAT-1 regulation [19]. This imposes a condition on the system, that it may sometimes be able to retrieve information for some parts within the files of the Envisat system, but this access has to be explicitly forbidden for the general users. These access concerns were taken care of in the system by making sure annotation components only processed the correct, public parts of the files.

The Grid infrastructure made possible the management of data in several locations.

Legacy formats/systems

One of the key issues to bear in mind when implementing a completely new design in a system of this size and complexity is how to manage migration of systems. Some parts of the system (probably belonging to other organisations or facilities) are not changed at all, or only partially updated and there is no simultaneous modernisation of all parts in the system. It is therefore necessary to gradually adapt the un-changed elements to the new development and to incrementally migrate functionality. Legacy systems can be considered in terms of data formats and software elements.

Data formats Although current data files are structured and in well documented formats there remain hundreds of different file types. Much of the metadata is only "implicit", such as the information stored within filenames or specialised code systems. For these files we have made this metadata "explicit", and much more easily accessible. This helps query developers, and also systems that must manage the lifecycle of metadata.

The use of the ontology to store information about properties will make migration of future data simpler. The mapping simply has to be done between the data and the ontology once. This should be especially easy as these existing systems have very strictly defined inputs and outputs and increasingly the formats used are XML based. The process of writing the specific code to extract the information from the datafile and re-encode it as RDF is much simplified, and can be more easily managed.

Software In the Envisat mission, complex software elements with well-defined interfaces are used in both planning and product generation. Some of these functionalities were incorporated in the prototype (e.g. time conversion utilities) by enabling them to be accessed as Web Services.

In a distributed architecture, such as that used in this Semantic Grid, encapsulation allows external systems to interact with individual parts of the system. For example, during a transitional period they might simply make use of a query interface to extract information or an RDF interface to provide some data to an RDF store or annotation component.

Standardised approach

The metadata format and query language defined purposely for the current QUARC implementation, although powerful, cannot be exported and directly used in other systems.

The Satellite Mission Grid uses the RDF standard for the storage of the metadata, and the specifics of the format are described by OWL and RDF(S) schemas. RDF allows us to use standard query languages like SPARQL or RQL which are incorporated in tools which have been already tested and proved adequate for re-use in other systems. The use of standard formats also allows different metadata stores to be used, depending on circumstances. For example, we used Atlas and Sesame. Another advantage is in not having to train developers in specific new skills but to be able to use what they already know. However, the existence of several query languages can add an overhead in terms of the required skills for developers.

Flexibility

The system allows a user who is familiar with one of the supported query languages to develop queries directly and iteratively. The process of creating forms which allow other users to provide the specific values for these queries is simple, and made even simpler by the existence of 'common' methods for converting between time formats. In this way we have been able to demonstrate how the system can be enhanced without any significant technical changes being required. This is crucial in an analysis system, where not all relevant queries are known or defined at the beginning of the use of the system.

It is also possible to add in some new relationships or properties without having to change the existing data at all. If a new way of grouping some part of the existing information was required by operators of the system then it could be added into the RDF directly. For example, a set of subtypes of instrument types could be created to describe their power consumption. Then each of the instruments could be labelled as belonging to one or other of the subtypes. This would take just a few statements in RDF and the millions of pieces of event information organised by which instrument generates them would now also be connected to this set of subtypes, allowing a user to make queries based on the power consumption.

Semantic Technologies

An Earth observation mission may last 10/15 years and the completely flexibly query interface allows exploring of data and development of new queries. This is crucial as anomalies are discovered and examined. Furthermore, new missions may provide information which is useful to combine with this existing data.

The developed Satellite Ontology can be extended and enlarged as required by the complexity of functionalities involved in the system. The structure developed so far has been sufficient for the data managed so far. The common ontology has enabled communication, and rapid extension of functionalities as demonstrated through Use Cases 2 and 3. The addition of "generic" information (such as the linking of event types to instruments) allows us to have semantically rich annotations. This semantic approach to queries where they are moved up to greater levels of abstraction gives us much more flexibility and robustness over time, as we are querying what the users need to know (usage of the instrument) rather than what we traditionally have recorded (the list of codes used for the event types). This shows the general technique of modelling what it is that users wish to search and discuss. As well as making development more efficient it reduces the time to acquaint new users with the system.

Data life cycle

We have shown that controlled updates can be made to stored data. These are automated but only allowed from authorised data sources. This ability supports the fact that data is not static, and that having access to the correct information can involve removing out-of-date statements as well as adding new ones.

More generally, we hope to be able to integrate data from many sites and missions and reuse the same system across these missions. As such we have created the methodology and tools for adding new data sources. Lightweight annotation components can convert from a legacy system to common RDF which is made available (via the semantic binding service) to the query engines.

There are high volumes of data being produced by Envisat – anticipated to reach a petabyte of data in 10 years of operation [20]. The extraction and management of just metadata (rather than all the data itself) is necessary for any ongoing analytical system. In a distributed system such as the Satellite Mission Grid, we create a single amalgamated dataset from many geographically dispersed data silos. Crucially, all the access is virtualised from the user perspective, i.e. they don't have to know about it at all. The resources and computation are distributed but the user has a simple, local browser interface for annotation and querying.

Modularity and Extensibility

The abstraction of components in a loosely coupled system means we have all the advantages of modularity. Interchangeable components can be selected depending on the particular application of the system. It allows the users to enjoy a single interface into whichever systems are determined to be best suited to the current scale of data and complexity of query. We also gain in extensibility; it opens up any further development to using "best-of-breed" components, be they new versions of existing ones, or completely new development. Separate testing of the various components will provide better metrics for deciding between them. We encountered the usual problems of distributed deployment but none proved insoluble and with experience these will be further reduced.

5 Conclusions and Next Steps

A semantic approach, where metadata is created explicitly and managed as a "first class object" gives advantages of flexibility and extensibility. A Grid approach where data and computations are distributed across many sites and machines gives improvements of scalability and robustness. The prototype system has shown itself capable of carrying out the current functionality of mission analysis systems, but across a geographically distributed dataset. It has also shown itself to be easy to extend in capability without significant development effort.

In the Semantic Data Grid community we have helped focus on lightweight protocols and making components more easy to integrate with existing systems. This vision supports a movement towards SOKU – Service Oriented Knowledge Utilities. The next industry steps include the incremental updating and extension of existing systems, where we will add to the metadata we store and make explicit what was formerly implicit. There can also be new approaches to work on both internal tools and for external development work within ESA projects. The experience means we can actively seek out which new projects to build from the ground up in a semantically aware manner.

6 Acknowledgments

The authors would like to thank the other members of the OntoGrid consortium. They are also very grateful to the European Space Agency (ESA) representatives Olivier Colin (ESA-ESRIN) and Pierre Viau (ESA-ESTEC) for providing access to the actual products and auxiliary tools from the Envisat mission.

7 References

1. Sánchez Gestido, M.: OntoGrid Business Case and User Requirements Analysis and Test Set Definition For Quality Analysis of Satellite Missions. Deliverable D8.1, OntoGrid, http://www.ontogrid.net (2005)
2. ESA bulletin number 106, "EnviSat special issue", http://www.esa.int/esapub/pi/bulletinPI.htm
3. Corcho, O., Alper, P., Kotsiopoulos, I., Missier, P., Bechhofer, S., Kuo, D., Goble, C.: An overview of s-ogsa: a reference semantic grid architecture. Journal of Web Semantics 4 (2006)
4. Foster, I., Kishimoto, H., Savva, A., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J., Reich, J. V.: The open grid services architecture, version 1.0. Technical report, Open Grid Services Architecture WG, Global Grid Forum (2005)
5. Treadwell, J.: Open grid services architecture glossary of terms. Technical report, Open Grid Services Architecture WG, Global Grid Forum (2005)
6. RDF(S) Grid Access Bridge, OntoGrid, http://www.ontogrid.net
7. Estebán-Gutierrez, M., Gómez-Pérez, A., Corcho, O., Muñoz-García, O.: WS-DAIOnt-RDF(s): Ontology Access Provision in Grids, The 8th IEEE/ACM International Conference on Grid Computing, http://www.grid2007.org/?m_b_c=papers#2b (2007)
8. Grid-KP, OntoGrid, http://www.ontogrid.net
9. http://www.isoco.com/innovacion_aplicaciones_kp.htm
10. Corcho, O., Alper, P., Missier, P., Bechhofer, S., Goble, C.: Grid Metadata Management: Requirements and Architecture, The 8th IEEE/ACM International Conference on Grid Computing, http://www.grid2007.org/?m_b_c=papers#2b (2007)
11. Atlas, OntoGrid, http://www.ontogrid.net
12. Miliaraki, I., Koubarakis, M., Kaoudi, Z.: Semantic Grid Service Discovery using DHTs, 1st CoreGrid WP2 Workshop on Knowledge and Data Management (2005)
13. Sesame, Aduna Open Source project, http://www.openrdf.org
14. Alper, P., Corcho, O., Goble, C.: Understanding Semantic-Aware Grid Middleware for e-Science., Journal of Computing and Informatics (To be published)
15. ONTO-DSI, OntoGrid, http://www.ontogrid.net
16. Senger, M., Rice, P., Oinn, T.: Soaplab - a unified Sesame door to analysis tools. In: Proceedings of UK e-Science, All Hands Meeting, 2-4th September, Nottingham, UK (2003)
17. NeOn toolkit, http://www.neon-project.org/web-content/
18. Wright, R: Ontogrid Satellite Use Case demonstration video, http://www.youtube.com/watch?v=TSbb_8vmKvk
19. ESA Earth Observation Principal Investigator, http://eopi.esa.int/esa/esa
20. European Space Agency Information Note 13, http://www.esa.int/esaCP/ESA0MDZ84UC_Protecting_0.html