Providing Context-Sensitive Access to the Earth Observation Product Library

Stephan Kiemle[1] and Burkhard Freitag[2]

[1] German Aerospace Center (DLR), German Remote Sensing Data Center (DFD)
Oberpfaffenhofen, D-82234 Weßling, Germany
Stephan.Kiemle@dlr.de
[2] University of Passau, Department of Computer Science and Mathematics
D-94030 Passau, Germany
Burkhard.Freitag@uni-passau.de

Abstract. The German Remote Sensing Data Center (DFD) has developed a digital library for the long-term management of earth observation data products. This Product Library is a central part of DFD's multi-mission ground segment Data and Information Management System (DIMS) currently hosting one million digital products, corresponding to 150 Terabyte of data. Its data model is regularly extended to support products of upcoming earth observation missions. The ever increasing complexity led to the development of operating interfaces which use a-priori and context knowledge, allowing efficient management of the dynamic library content. This paper presents the development and operating of context-sensitive library access tools based on meta modeling and online grammar interpretation.

Keywords: context sensitivity, meta modeling, earth observation, object query language, information management.

1 Introduction

The Product Library developed and operated at the German Aerospace Center DLR manages ever increasing amounts of digital earth observation products. The data growth rates are challenging, and even more so the increasing diversity of data structures and formats. Currently the Product Library already hosts about 80 different product types.

To be able to efficiently manage the huge amount of heterogeneous data, comprehensive human-machine interfaces and tools have been developed at DLR. This paper addresses the development of context-sensitive, interactive operating interfaces and presents operational experiences in the domain of earth observation data management. In particular, we describe an interactive query editor that makes intensive use of static a-priori information and meta information about the dynamic library data model to help the user formulating his or her queries and ensure valid interactions.

The following two sections give an overview of the architecture of the DIMS system and describe the problem to be solved by providing context-sensitive access to the Product Library. We will then discuss the underlying data model. Next, the interactive query editor supporting ad-hoc metadata queries is presented, followed by an evaluation section and a conclusion.

2 The Data and Information Management System DIMS

The Data and Information Management System (DIMS) has been developed as a distributed multi-mission infrastructure for production, cataloguing, archiving, ordering, accounting and distribution of earth observation products [1]. Fig. 1 shows an overview of the system architecture with the main service components EOWEB® user services, ordering control for the processing of user orders, production control for the organization of production workflows, different flavors of processing systems for the ingestion, value adding and publishing of earth observation data, the Product Library, the product generation and delivery component and components for monitoring and control.

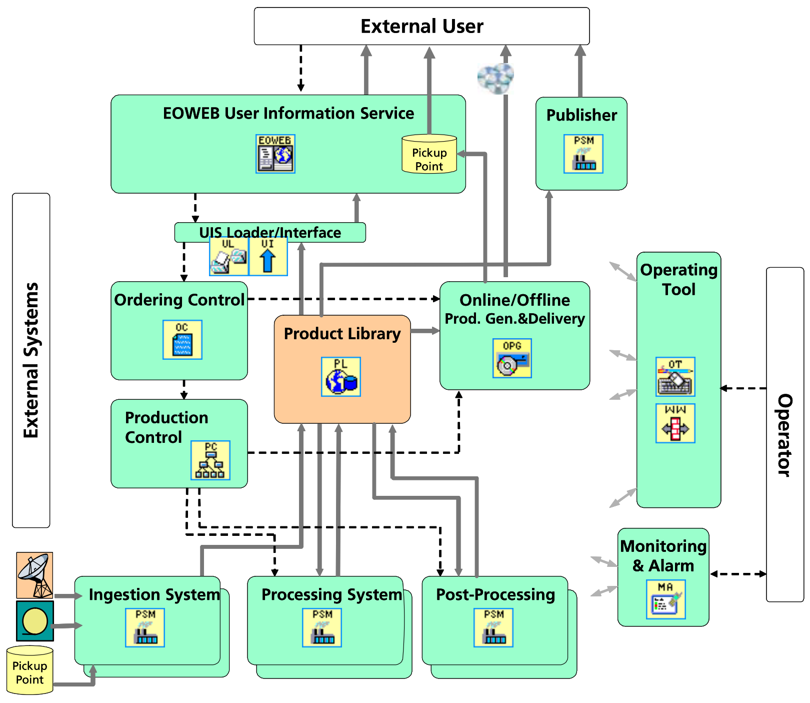
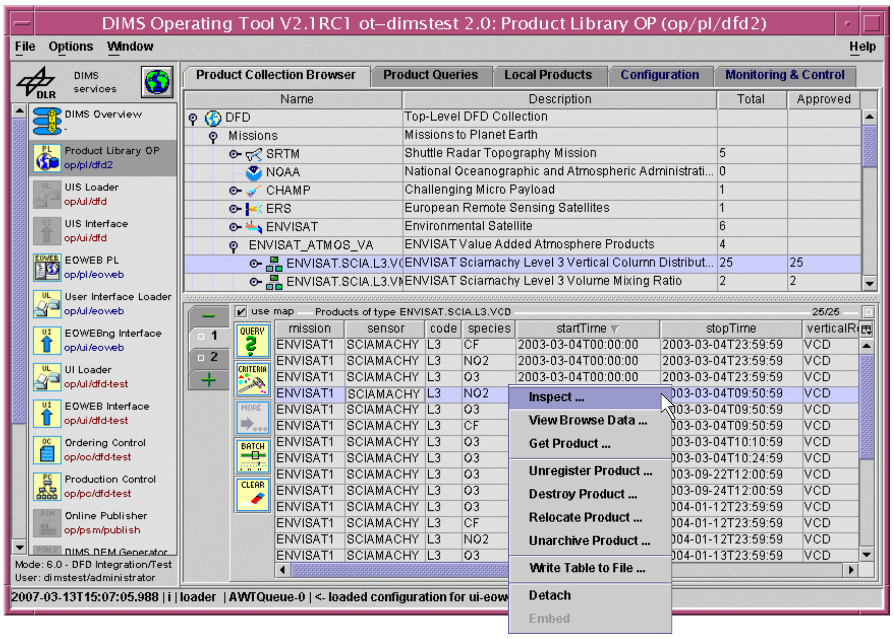


Fig. 1. Data and Information Management System Overview

As a central component the Product Library is responsible for the consistent long-term preservation of digital data products. It consists of an archive for the long-term storage of primary data and an inventory for efficient storage of metadata. A middleware encapsulates these components providing a comprehensive library interface for the consistent management of digital data products. This middleware decouples high-level information modeling and evolving low-level storage structures such as the robot-driven media library and hierarchical storage management for primary data and a relational database management system for metadata [2].

All product data can be accessed via the object oriented query language, extending the OQL standard [3]. The Product Library provides several functions required for data management. The Operating Tool (Fig. 2) is used to access these functions. All items of a collection can be listed using the incremental query mechanism, individual items can be searched by entering object query language conditions (including spatial operators) and item details (metadata, component structure, browse images) can be viewed. Items can be inserted, updated, retrieved and destroyed. Items can be registered and unregistered (manipulating only the metadata but not the data files), items can be re-located and un-archived (manipulating only the data files but not the metadata).



Name	Description	Total	Approved
DFD	Top-Level DFD Collection		
Missions	Missions to Planet Earth		
SRTM	Shuttle Radar Topography Mission	5	
NOAA	National Oceanographic and Atmospheric Administrati...	0	
CHAMP	Challenging Micro Payload	1	
ERS	European Remote Sensing Satellites	1	
ENVISAT	Environmental Satellite	6	
ENVISAT_ATMOS_VA	ENVISAT Value Added Atmosphere Products	4	
ENVISAT.SCIA.L3.V(	ENVISAT Sciamachy Level 3 Vertical Column Distribut...	25	25
ENVISAT.SCIA.L3.V|	ENVISAT Sciamachy Level 3 Volume Mixing Ratio	2	2

✓ use map — Products of type ENVISAT.SCIA.L3.VCD — 25/25

mission	sensor	code	species	startTime	stopTime	verticalR
ENVISAT1	SCIAMACHY	L3	CF	2003-03-04T00:00:00	2003-03-04T23:59:59	VCD
ENVISAT1	SCIAMACHY	L3	NO2	2003-03-04T00:00:00	2003-03-04T23:59:59	VCD
ENVISAT1	SCIAMACHY	L3	O3	2003-03-04T00:00:00	2003-03-04T23:59:59	VCD
ENVISAT1	SCIAMACHY	L3	NO2		003-03-04T09:50:59	VCD
ENVISAT1	SCIAMACHY	L3	O3		003-03-04T09:50:59	VCD
ENVISAT1	SCIAMACHY	L3	CF		003-03-04T09:50:59	VCD
ENVISAT1	SCIAMACHY	L3	O3		003-03-04T10:10:59	VCD
ENVISAT1	SCIAMACHY	L3	O3		003-03-04T10:24:59	VCD
ENVISAT1	SCIAMACHY	L3	O3		003-09-22T12:00:59	VCD
ENVISAT1	SCIAMACHY	L3	O3		003-09-24T12:00:59	VCD
ENVISAT1	SCIAMACHY	L3	O3		004-01-12T23:59:59	VCD
ENVISAT1	SCIAMACHY	L3	CF		004-01-12T23:59:59	VCD
ENVISAT1	SCIAMACHY	L3	NO2		004-01-12T23:59:59	VCD
ENVISAT1	SCIAMACHY	L3	O3		004-01-13T23:59:59	VCD

Mode: 6.0 - DFD Integration/Test
User: dimstest/administrator

2007-03-13T15:07:05.988 | i | loader | AWTQueue-0 | <- loaded configuration for ui-eow

Fig. 2. DIMS Operating Tool, Product Library Collection Browser with Menu of Product Management Functions

The Operating Tool also provides configuration views allowing to manage the data collection hierarchy, the item spaces used for modular data modeling in the inventory and the archiving rules used to organize the file directory structure within the archive.

3 Problem Description

The tasks of product management include different operating use cases requiring interactive access to the library content. Knowledge about the evolving information model is absolutely essential e.g. when placing ad-hoc queries to the inventory or when browsing the library content and generating population reports.

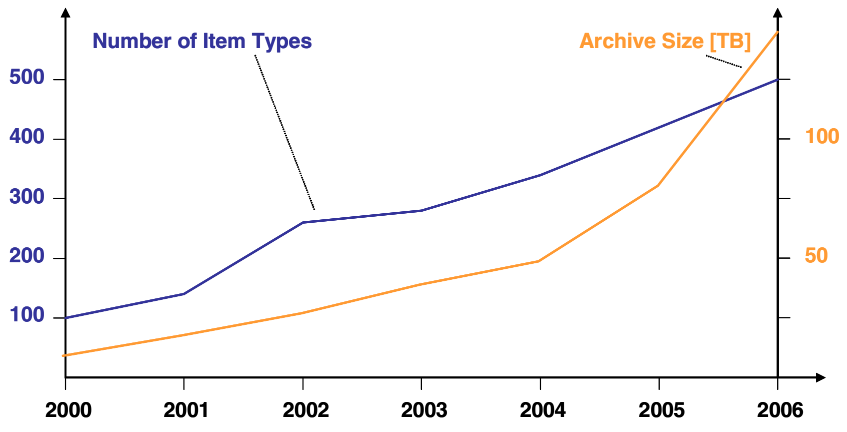


Fig. 3. Evolution of Number of Item Types and Library Archive Size

In the six years of operation of the DIMS Product Library, 500 item types (product and component collections) have been configured using a total of 1970 attributes. Growth and evolution of the Product Library is permanent. Fig. 3 shows how the size of the information model, represented by the number of item types, increased in parallel to the size of the library. The new German remote sensing mission TerraSAR-X [4], for example, added 30 new product item types and will multiply, together with other upcoming missions the current volume of the library by six in terms of number of products and archive size, leading to a total number of 10 million product items and 840 TByte of data in 2012. The variety of collections will significantly increase, making the management of the huge amount of heterogeneous data more and more difficult.

Operators use the DIMS Operating Tool, i.e., the graphical user interface for unified operating of all DIMS services, to browse the library content and place ad-hoc queries. Besides a hierarchical collection tree operators can view a textual documentation to get orientation in the bits and pieces of the information model and to get help on the use of the query language.

The librarian, i.e. one of the operators using the DIMS Operating Tool, is responsible for the management of the earth observation data in the Product Library. In this function he or she

- decides what products to store for the long term
- defines and maintains the data model
- configures the library according to the data model
- grants library access to users
- supervises data ingestion and access activity
- monitors library operations
- decides about data expiration and removal
- reports about library operations and use

Therefore the librarian needs comprehensive tools to access and monitor the Product Library. The librarian has to be able to browse the library content, retrieve individual items and perform actions on single or a set of identified products.

However, the available support for library operations and access turned out to be insufficient. Operators require aware client applications with knowledge about the underlying data model in order to cope with the increasing complexity in this dynamic application environment. Operating clients should also give support in the formulation of correct ad-hoc query expressions and take into account individual operator preferences, habits and the activity history.

As one consequence, the DIMS Operating Tool had to be extended by a context-sensitive interactive OQL query editor for ad-hoc queries as presented in this paper.

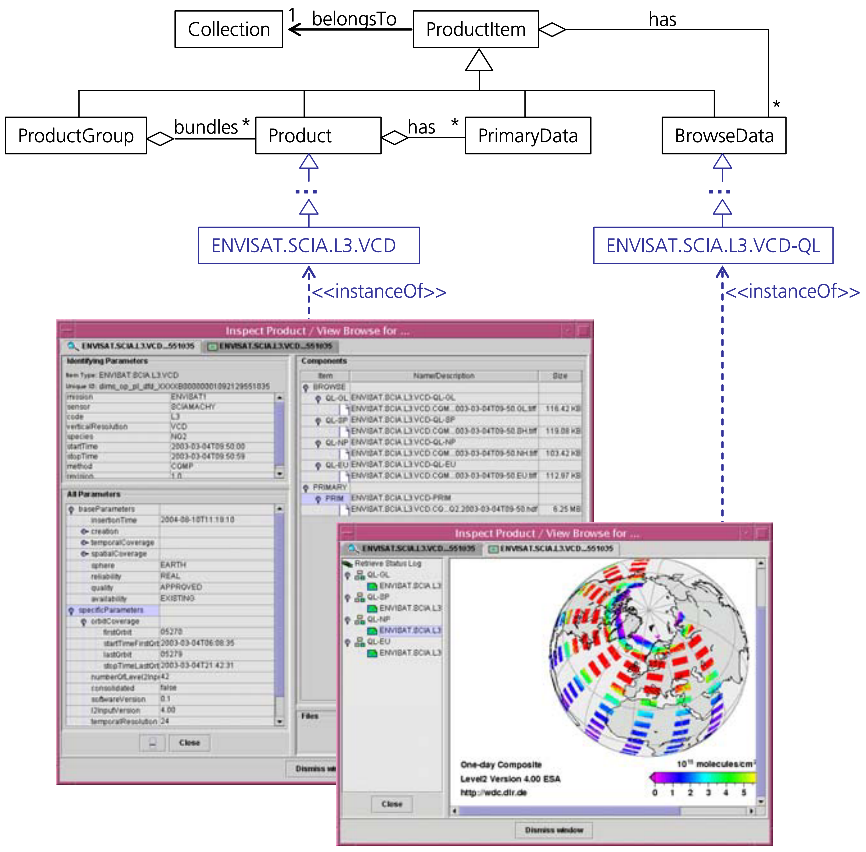
4 Data Model

In the following we will focus on the product data model to show how this knowledge can be used for context-sensitive management tools supporting the librarian.

4.1 Object Model

Object orientation as it can be defined with the Unified Modeling Language (UML 2.0) is particularly well suited for the domain of earth observation data products. Products are independent identifiable objects composed of other objects such as browse images, primary data, processing logs and quality maps. Different products of the same mission can inherit common properties. Higher level information products derived from raw data products are associated to their “predecessors”. Beyond the features described above there exist of course more properties of earth observation products which can also be represented very well using an object model.

In its upper part Fig. 4 shows the basic product model underlying the Product Library. Specific mission collections, products and product components are added by extending the classes *Collection*, *ProductGroup*, *Product*, *PrimaryData* and *BrowseData*,



Identifying Parameters

Item Type: ENVISAT.SCIA.L3.VCD
Unique ID: dims_op_pl_dfd_XXXXB00000001092129551035

mission	ENVISAT1
sensor	SCIAMACHY
code	L3
verticalResolution	VCD
species	NO2
startTime	2003-03-04T09:50:00
stopTime	2003-03-04T09:50:59
method	COMP
revision	1.0

All Parameters

baseParameters	
insertionTime	2004-08-10T11:19:10
creation	
temporalCoverage	
spatialCoverage	
sphere	EARTH
reliability	REAL
quality	APPROVED
availability	EXISTING
specificParameters	
orbitCoverage	
firstOrbit	05270
startTimeFirstOrt	2003-03-04T06:08:35
lastOrbit	05279
stopTimeLastOrt	2003-03-04T21:42:31
numberOfLevel2Inpu	42
consolidated	false
softwareVersion	0.1
l2InputVersion	4.00
temporalResolution	24

Components

Item	Name/Description	Size
BROWSE		
QL-OL	ENVISAT.SCIA.L3.VCD-QL-OL	
	ENVISAT.SCIA.L3.VCD.COM...003-03-04T09-50.OL.tif	116.42 KB
QL-SP	ENVISAT.SCIA.L3.VCD-QL-SP	
	ENVISAT.SCIA.L3.VCD.COM...003-03-04T09-50.SH.tif	119.08 KB
QL-NP	ENVISAT.SCIA.L3.VCD-QL-NP	
	ENVISAT.SCIA.L3.VCD.COM...003-03-04T09-50.NH.tif	103.42 KB
QL-EU	ENVISAT.SCIA.L3.VCD-QL-EU	
	ENVISAT.SCIA.L3.VCD.COM...003-03-04T09-50.EU.tif	112.97 KB
PRIMARY		
PRIM	ENVISAT.SCIA.L3.VCD-PRIM	
	ENVISAT.SCIA.L3.VCD.CO...02.2003-03-04T09-50.hdf	6.25 MB

Files

Fig. 4. Basic Product Model (Extract) and Extension Example with Instances Symbolized by Operating Tool Product Inspection and Browse Views

4.2 Data Model Evolution

In the dynamic environment of the Product Library the data management task of configuring new product types for new missions is a nominal use case which has to be supported without interruption of operations. By configuring the library it is possible to define additional archiving rules extending the archive system as well as to extend the product metadata model in the inventory system.

The inventory of the Product Library allows to configure object data models. *Collections* are used to define the component structure of a product and to place products in a freely configurable collection hierarchy allowing easy navigation e.g. by mission, sensor or application domain. *Item types* are used to define product and component types by specifying name, meaning, a list of identifying attributes and describing attributes. Types can be set in a single-inheritance hierarchy, inheriting properties of parent types. *Attributes* are defined by specifying name, meaning, data type and other basic properties such as valids, value ranges and value constraints. Attributes can be structured, meaning that their data type is not primitive but a *structure* itself defining a list of slot attributes. Associations, aggregations and composition *references* can be defined to link item types.

In a huge digital library system like the DIMS continuous change at various levels has to be anticipated. In by far the most cases an existing data model will be extended. One way to achieve this is to extend already defined item types (see Fig. 4). Of course, also new item types can be defined as well. They can reuse already defined attributes, structures and references if the meaning matches. This simplifies modeling and leads to easier manageable data models.

4.3 Meta Model

The entities introduced above to define data models are called modeling elements. Of course these elements can again be modeled and managed within the library. The Product Library inventory therefore maintains the *meta item space*, a repository of all modeling elements ever defined to build application data models.

The advantages of managing modeling elements in a distinct repository are obvious: the modeling tools use this repository for safe persistence of configured data models and they can browse already defined modeling elements to allow their reuse.

In the OMG meta model architecture [5], different levels of modeling are defined. The M0 level represents the real world, in our case the earth observation products in the Product Library.

The M1 level represents the data models of M0, here the product data model consisting of types, attributes and references as described above. The M1 level of modeling gives a common formal view on reality, allowing to describe, compare, reuse and exchange application data items.

The M2 level represents the data models of M1, i.e., the meta model used to define application data models. The M2 level of modeling gives a standard and formal view on application data models, allowing to describe, compare, reuse and exchange application data model elements.

The OMG meta model architecture also defines a M3 level again abstracting the M2 level and intended to give an ubiquitous, generally applicable representation of meta models to be able to even represent and formally define different ways of defining meta models.

In practice, the Product Library uses the M0 level (product instances stored in the library), the M1 level (product data model) and the M2 level (repository of modeling elements). The M3 level is not used, since there is no need for different meta models and thus not need to further abstraction. However, the repository containing the modeling elements is self-contained, meaning the structures of the meta model are themselves defined as instances in the meta model. Thus the repository of modeling elements corresponds to both the M2 and the M3 level. This allows e.g. to access the repository of modeling elements using the same tools and interfaces (such as the object query language) as for accessing the product data models.

Fig. 5 shows an extract of the Product Library meta model hosting the modeling elements. Based on this meta information on the configured data models, the inventory is able to configure the physical storage layer, namely the underlying relational database management system. The inventory middleware maps all access to the product metadata such as object queries and insertions to corresponding statements to the database systems, thereby decoupling application level interfaces from the relational storage model. This allows an independent physical design, e.g. choosing normalization to save space or de-normalization to save access time.

In addition, client applications can access the repository of modeling elements to add model awareness and guide the user through the library data model.

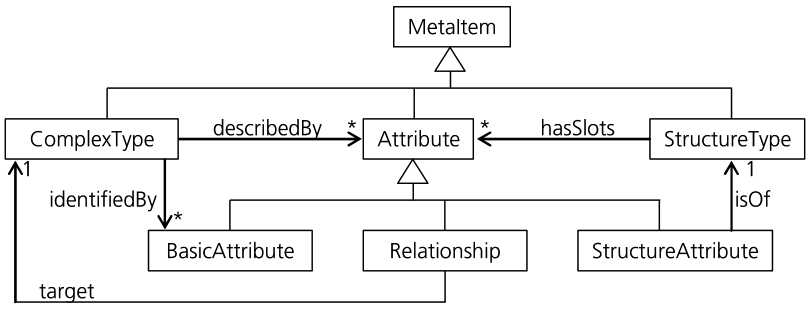


Fig. 5. Product Library Meta Model (Extract)

5 The Interactive OQL Query Editor

5.1 Requirements

To support users in formulating ad-hoc queries, a context-sensitive OQL editor had to be developed supporting

- query completion at an arbitrary input position
- error marking
- special token evaluation.
  If the proposal list includes special tokens which can be replaced with model elements, this will be done. For example the special token <CT> will be replaced with a list of available ComplexTypes stored in the repository.
- usage of meta model information
- template selection for the `select` or the `where` clauses of a OQL query.

5.2 Grammatical Context

The online interpretation of the query language grammar, i.e. the analysis of expressions during the editing process, allows the evaluation and validation of human inputs on three information levels.

Lexical correctness means that the query consists of valid tokens whereas syntactical correctness guarantees that it is valid with respect to the rules defining the grammar. Finally, semantical correctness ensures that the types, conditions and expressions are consistent and match the data model. For instance, in semantically correct queries comparisons of attributes with literal values use the same data type, and the attributes used in conditions refer to an object type which has actually been specified in the `from` clause.

The OQL query editor has been developed based on an EBNF definition of the query language and using the parser generator JavaCC [7]. The generated lexer and parser classes allow the addition of semantic actions required to distinguish equally defined identifier tokens in different contexts and to connect the repository of modeling elements in order to compute sensitive suggestions for the next inputs in the given context. Depending of the current position, the suggestions may include grammar terminals such as keywords, brackets or comparators, as well as names of types and attributes as defined in the data model. If the input is syntactically incorrect, the parser issues an error message listing the expected tokens, even if the input is still incomplete.

The following example shows a valid OQL query where **syntactical** and semantical correctness is highlighted.

```
select   min sceneIndex, max sceneIndex
from     SRTM1.X-SAR.IFDS
where
 (dataTakeOrbit = 61 and availability = 'PRELIMINARY'
  and there exists no corresponding SRTM1.X-SAR.IFDS
  with equal dataTakeOrbit and equal sceneIndex
  where availability = 'EXISTING')
```

This query computes the scene index range of the interferometric dataset products (Shuttle Radar Topography Mission) on orbit number 61, which have not been processed yet and therefore are catalogued only with a preliminary status.

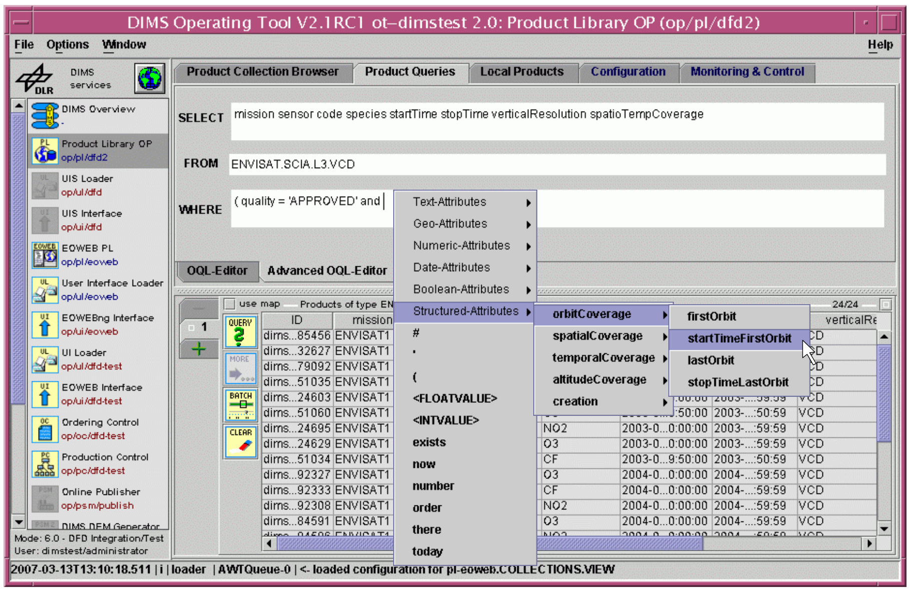
5.3 Repository-Related Context

Interactive editors supporting input completion based on a static information model are common in human machine interfaces. Less common is the capability to determine the correctness of partial expressions. Rather infrequent, however, is the capability to validate inputs against the dynamic data model.

To be able to give suggestions for elements of the data model and to check semantical correctness, the query editor retrieves information from the repository of modeling elements. The first information retrieved is the list of available object types which can be specified in the `from` clause of the query. Each object type is defined by its name and a list of identifying and describing attributes. In subsequent actions, the query editor retrieves meta information about selected attributes, such as data type, valid values, value ranges and structure slots.

Depending on the selected object type, the specific attributes describing this type are suggested to be included after `select` or within the condition of the query. Structured attributes can be expanded to their slot attributes. As each attribute has a well-defined data type, the editor can ensure the correct choice of operations, comparators and literals within expressions, e.g. not allowing the comparison of a numeric attribute with a date literal. The editor is able to recognise specified set attributes and references via properties of the corresponding modeling elements and is able to suggest appropriate conditions on the set elements or referenced object types.

Fig. 6 shows the context-sensitive OQL editor with a partial query. The object type has already been selected and a `where` condition has been partially specified. On typing a control key, the editor shows a pop-up menu with a collection of suggested valid language tokens or data model elements to be entered next. When the query has been completely specified this way, it is again validated and then sent to the inventory for execution. The Operating Tool displays the results in a table and on the map.



SELECT mission sensor code species startTime stopTime verticalResolution spatioTempCoverage

FROM ENVISAT.SCIA.L3.VCD

WHERE (quality = 'APPROVED' and

ID	mission				verticalRe
dims...85456	ENVISAT1				CD
dims...32627	ENVISAT1				CD
dims...79092	ENVISAT1				CD
dims...51035	ENVISAT1				CD
dims...24603	ENVISAT1		...00:00	2003-...:59:59	VCD
dims...51060	ENVISAT1		...:50:00	2003-...:50:59	VCD
dims...24695	ENVISAT1	NO2	2003-0...0:00:00	2003-...:59:59	VCD
dims...24629	ENVISAT1	O3	2003-0...0:00:00	2003-...:59:59	VCD
dims...51034	ENVISAT1	CF	2003-0...9:50:00	2003-...:50:59	VCD
dims...92327	ENVISAT1	O3	2004-0...0:00:00	2004-...:59:59	VCD
dims...92333	ENVISAT1	CF	2004-0...0:00:00	2004-...:59:59	VCD
dims...92308	ENVISAT1	NO2	2004-0...0:00:00	2004-...:59:59	VCD
dims...84591	ENVISAT1	O3	2004-0...0:00:00	2004-...:59:59	VCD

Mode: 6.0 - DFD Integration/Test
User: dimstest/administrator

2007-03-13T13:10:18.511 | i | loader | AWTQueue-0 | <- loaded configuration for pl-eoweb.COLLECTIONS.VIEW

Fig. 6. Context-sensitive OQL Editor Embedded in the DIMS Operating Tool

6 Evaluation

The object query language editor of the Product Library illustrates the benefits of context-sensitivity in interactive library applications. This editor supports an easy ad-hoc specification of complex search queries which are syntactically correct and semantically meaningful. The knowledge exploited for context-sensitivity consists of the static grammar of the query language and the dynamic data model of the library. Therefore the editor goes beyond classical language-directed or syntax-directed editors based on common compiler design [6].

The approach to develop a language based editor using a parser generator is very effective. One of its features is its declarative approach as opposed to developing a dedicated imperative program which of course would depend tightly on the grammar. It turned out in practice that grammar changes - which are more frequent than one would assume at first sight - could indeed be implemented in a more straight forward way as compared to a pure programmatic approach.

The general procedure to generate code with the aid of the parser generator instead of writing it by hand is helpful and prevents code failures. The usage of context information and metadata, especially data provided by the meta model, makes it possible to develop an editor which provides the user with information matching at the current position of the query. As mentioned above, the editor is able to validate the input not only syntactically but also semantically. Our evaluation showed that the number of erroneous or unreasonable queries has decreased significantly. Users report that they found that the error messages produced by the syntax checker are well understandable and helpful. On the semantical level the type checks for literals, operators and attributes help constructing a valid query. Very good acceptance received the context-aware automatic suggestion of valid attributes that takes into account the current product type and other content-related contextual properties.

As the editor is based on a parser, it provides the possibility to support the user at every position of the query. Taking a look at very powerful IDEs like Eclipse, it can be seen that this is not the case in every IDE. The approach entails a lot of possible features that can be implemented to provide the user with more help and information. The editor turned out to be useful for both kinds of users, beginners and professionals. The former one gets help in any situation. The latter one, already knowing the syntax of the language, only takes the assistant to get metadata information, especially from the available models. In most cases the usage of meta model information takes place in the background. In any case there is no need to look up the documentation of the data model or the OQL syntax as in former days. Moreover, OQL features that have seldom been used are now more frequently “detected” by the users.

The query example provided in section 5.2 illustrates the power and expressiveness of OQL compared to some SQL catalogue look-up in a pure relational system: This query can easily be formulated with the help of the context-sensitive editor, but it corresponds to the following translated SQL code, which is quite hard to be specified and understood:

```
SELECT
  MIN(t.intermediateSceneI), MAX(t.intermediateSceneI)
FROM M_SRTM1XSARIFDS t
WHERE (
  t.dataTakeOrbit = '61' AND
  t.availability = 'PRELIMINARY' AND
  NOT EXISTS (
    SELECT unique_id
    FROM M_SRTM1XSARIFDS r
    WHERE (
      r.availability = 'EXISTING' AND
      r.dataTakeOrbit = t.dataTakeOrbit AND
      r.intermediateSceneI = t.intermediateSceneI ) ) )
```

Therefore the interactive query editor allows operators and library users to specify ad-hoc queries without requiring detailed knowledge about the data model and without being an expert on SQL.

The context-sensitive interactive query editor is an important constituent of the sustainability of the DIMS Product Library, which has not only to cope with a permanently growing amount of data and diversity of information, but also to integrate existing digital archives and provide application-level interfaces for other services to cover all earth observation ground system tasks of product processing, monitoring, long-term storage, ordering and delivery.

7 Conclusion

This paper addresses the problem of user support in a very large digital library system. In particular, extensions of the data model and dynamic library content put a heavy burden on the user who has to extract application-specific data using a standard query language. We have shown how a context-sensitive editor can be constructed and integrated into the system that supports the user in formulating his or her queries. The editor is aware of the underlying data model as well as (part of) the dynamic content of the library and thus is able to propose syntactically correct and semantically meaningful continuations of a query. An evaluation has shown that the context-aware editor significantly improves user-friendliness und usability of the DIMS digital library system.

Acknowledgments. Special thanks to our student Ulrich Frank who significantly supported this work by investigation, assessment and prototyping. We also thank Sven Kröger, DLR, for his support during prototyping and integration.

References

1. Mikusch, E., Diedrich, E., Göhmann, M., Kiemle, S., Reck, C., Reißig, R., Schmidt, K., Wildegger, W., Wolfmüller, M.: Data Information and Management System for the Production, Archiving and Distribution of Earth Observation Products. Data Systems in Aerospace 2000, EUROSPACE. ESA Publications Division, SP-457, Noordwijk (2000)
2. Kiemle, S.: From Digital Archive to Digital Library – a Middleware for Earth-Observation Data Management. In: Agosti, M., Thanos, C. (eds.) ECDL 2002. LNCS, vol. 2458, Springer, Heidelberg (2002)
3. Cluet, S.: Designing OQL: Allowing objects to be queried. Information Systems 23(5), 279–305 (1998)
4. German TerraSAR-X Radar Satellite Mission homepage at the German Aerospace Center, Available in the WWW at http://www.dlr.de/tsx/start_en.htm
5. Object Management Group (OMG): Common Warehouse Metamodel (CWM) Specification. Version 1.1, volume 1 (March 2003), Available in the WWW at http://www.omg.org/docs/formal/03-03-02.pdf
6. Grune, D., Bal, H., Jacobs, C., Langendoen, K.: Modern Compiler Design. John Wiley, Chichester (2002)
7. javaCC project home (last visited 13/03/2007), Available in the WWW at https://javacc.dev.java.net/