Querying Framework Documentation

Mauricio Sansano
LIFIA - Universidad Nacional de La Plata
La Plata, Argentina.

50 y 115 - 1er. Piso
(CP 1900) - La Plata, Argentina
Phone / Fax: (++54) 221 - 4228252

Marcos Godoy
LIFIA - Universidad Nacional de La Plata
La Plata, Argentina.

50 y 115 - 1er. Piso
(CP 1900) - La Plata, Argentina
Phone / Fax: (++54) 221 - 4228252

Juan Cappi
LIFIA - Universidad Nacional de La Plata
La Plata, Argentina.

50 y 115 - 1er. Piso
(CP 1900) - La Plata, Argentina
Phone / Fax: (++54) 221 - 4228252

Gustavo Rossi
LIFIA - Universidad Nacional de La Plata
La Plata, Argentina.

50 y 115 - 1er. Piso
(CP 1900) - La Plata, Argentina
Phone / Fax: (++54) 221 - 4228252
*also CONICET and UNLM

Abstract

In this paper we discuss about the combination of to powerful tools for documenting object oriented frameworks: hypermedia navigational models and hypermedia queries. We also present a way for translating queries performed on top of framework documentation applications, into XML representations, as a way for exchanging framework documentation not only through the web, but through different information systems.

Specially, we concentrate in integrating OOHDM, a model for designing object oriented hypermedia applications, with framework documentation applications having hypermedia features, and with OOHQL, an object oriented hypermedia query language that allows to perform queries on top of these kind of applications.

Our intent is to describe an architecture that, integrating this elements, will allow to create easy-to-use framework documentation, capable of being exchanged through other systems and work teams.

Keywords: OO-Frameworks, hypermedia queries, XML representation, documentation.

1. Introduction.

Nowadays, maybe the most important issue in software engineering is to achieve a substantial degree of reusability. This leads us to the state-of-the-art in object-oriented reusable assets, i.e. object-oriented frameworks.

Object-oriented frameworks are the state-of-the-art solution for building high quality applications in a particular domain, by systematically reusing an abstract design for that domain [Fayad99]. A framework provides a set of abstract and concrete classes that, when instantiated, work together to accomplish a given task within the framework domain. We can assume that an object-oriented framework is the implementation of an abstract design representing a specific domain model. We use frameworks in order to create new applications belonging to the same domain the framework represents. This way, an application engineer reuses design, improving software quality.

However, a framework is not an easy tool to understand and use. Learning how to use a framework requires both a deep comprehension of the framework domain, and a clear understanding of the framework architecture. The framework design, composed by concrete and abstract classes and the interrelationships between their instances, must be understood by the framework user, in order to accomplish the task of, for example, build a new application using the framework.

For this reason, framework documentation has become an important issue in the past few years. Moreover, we need not only a strong documentation for any application framework, but we also need efficient documentation techniques. We need a well designed organization of the documentation and a rapid and easy way to find and access those documents. The possibility of performing queries on top of framework documentation, let framework users to clearly specify which documentation is he looking for. This search must be more powerful then a simple text-oriented search. Lastly, it is also required to produce an exportable framework documentation, in order to have the possibility of exchanging documentation through other users or systems. An XML representation of the documentation may be used for achieving this goal.

This paper is organized in the following way: in section 2 we briefly discuss about the main issue of documenting frameworks, and the integration of different documentation techniques is explained as a way of improving framework documentation. In section 3 we present a hypermedia model for framework documentation. In section 4 we present a way for querying framework documentation navigational models and improve documentation usability. In section 5 we discuss about how to translate those queries to an XML representation. Finally in section 6 we give our conclusions and we talk about future work we are planning to do in this field.

2. Integrating documentation techniques.

Framework documentation should address the requirements of different kind of users, each of them expecting and requesting different things from the documentation. These users will need different documentation artifacts that help them understanding the framework according to their needs, and allowing them to accomplish a given task. In this context, many different documentation techniques needs to be integrated in order to achieve the requirements of each documentation user. For example, we may need to combine cookbook recipes with class diagrams and interaction diagrams. We may also want to combine hot spot cards [Pree95] with

cookbook recipes and patterns used for documenting the framework design [Jhonson92], etc. Moreover, we expect documentation users to be able to combine different documentation techniques on-the-fly, allowing users to customize the documentation in order to fit his/her specific needs.
Combining different active documentation techniques requires some way to organize and link these elements of on-line information. One way to achieve this characteristic is to organize the framework documentation as a hypermedia application. A hypermedia application allows us to create nodes representing each of the documentation artifacts (e.g. UML Sequence Diagrams [UML], cookbook chapters, CRC cards [Bellin97], etc.) and connect them using links. We have designed a complete architecture that allows us to develop these kind of applications. This architecture details are described in [Sansano2000]

3. A hypermedia model for organizing documentation.

As it has already been explained, the framework documentation will be organized by using the hypermedia paradigm. In this paper we concentrate in the documentation required for the standard user of a framework, the one who needs to build an application instantiating the framework.
We will now describe one possible hypermedia model for the framework documentation application that fulfils this type of user's needs. Of course, other models can be designed to achieve this goal.

Documentation oriented to the standard user
The standard user needs a kind of documentation that allows him to instantiate the framework in a rapid and correct way. This includes the customization of the hot-spots as well as the implementation of those hook methods that the framework defines, for the resulting application to work properly. This type of user needs to build-up a typical application that belongs to the framework domain, so for example, documentation will be composed by different examples of the framework instantiation. These examples would be linked to well-known documentation artifacts such as CRC Cards, Class Diagrams and Sequence Diagrams. From each cookbook item the user can navigate to the associated example, or to the Class Diagrams, or to the Sequence Diagrams, or to the related CRC Cards. Furthermore, we must provide browsing capabilities among these documentation artifacts (e.g. we could browse from the actor in a Sequence Diagram to the Class Diagram in which the class of this actor is present, and from the class in a Class Diagram to the CRC Card that specifies this class). The possibilities of navigating through these documentation tools are shown in the following navigational diagram:

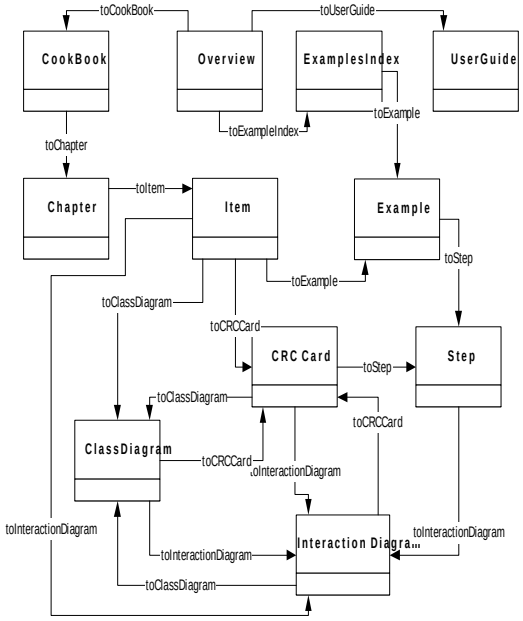

Figure 1. *Standard User navigational schema.*

Figure 1 shows a navigational diagram in OOHDM notation [Schwabe96]. This notation uses boxes to represent nodes, and arrows to represent typed links between those nodes. As complex documentation

involves sets of objects that are closely related, we need a way to easily navigate through these sets. We define a Navigational Context as a set of nodes that can be traversed sequentially (interaction diagrams where a class appear, CRC cards related with a requirement, etc.). Notice that the same object (node) may appear in different contexts so we need to separate the core information of that document (e.g. a CRC card) and context-related information (such as the following CRC in the same set, or some further explanation regarding the belonging of the CRC to the set). We use decorators [Gamma95] for this purpose. Navigational Contexts help organizing the navigational space of the documentation application as explained in [Schwabe98]. Some navigational contexts we can find in the previous schema are the following:

1. Cookbook Items related to each example
2. CRC Cards, Class Diagrams and Sequence Diagrams related to each Cookbook Item
3. Class Diagrams and Sequence Diagram sorted by class.
4. Cookbook Items sorted by CRC Card.
5. Class Diagrams and Sequence Diagrams sorted by CRC Card.

4. Querying the Hypermedia Documentation Model.

Despite the numerous advantages that organizing framework documentation as a hypermedia application has, we want to point out another powerful capability we are adding to framework documentation field. An important problem when navigating large hypermedia documentation webs, is the loosing of perspective. It is very easy to get lost and disappointed when trying to find the desired information about the framework. In fact, this problem is a consequence of a much more huge problem, that is navigating a hypermedia application where a lot of nodes are involved, trying no to get lost and finding the right information as soon as possible. Most of the time, the user doesn't follow a direct path to the node or set of nodes containing the information his is looking for.

Many techniques and tools have been proposed in order to help hypermedia users in this issue. But maybe one of the most interesting tools in this context are hypermedia queries. Having the possibility of querying a hypermedia application and select the desired information specifying a query, using some kind of declarative user-oriented query language, gives users a powerful tool for rapidly accessing hypermedia information, and saving time during navigation.

So we are going to aloud documentation users not only to organize framework documentation in a hypermedia application, but also to query this hypermedia application for directly access to documentation nodes.

For allowing this querying capability stated above, we are going to use the OOHQL (Object Oriented Hypermedia Query Language) defined in [Diaz97] [Gordillo98]. OOHQL is a declarative hypermedia query language. Using OOHQL the user can easily perform queries on top of an object-oriented hypermedia navigational model, composed by node and link types, as defined in [Sansano99].

As an example of what we have recently stated, we present the following two queries, expressed in OOHQL, to be performed on top of the object-oriented navigational model presented before.

The first OOHQL expression retrieves all nodes of type *CookBookItem* (Item in figure 1) having a link connecting them with a node of type *CRCCard* whose *className* is Collection.

```
1)   select CookBookItem
     from cp: CookBookItem ,
          crc: CRCCard
     where (crc className = Collection)
           and
           Related_By(cp, toCRCCard, crc)
```

Here, *className* is a message (i.e. a getter message) who answers the name of the class the CRCCard node is representing.

The second OOHQL expression retrieves all nodes of type Example having an Interaction Diagram referencing the class ApplicationModel.

```
2)      select Example
        from e: Example,
             id: InteractionDiagram,
             crc: CRCCard
        where
        (crc className = ApplicationModel)
        and
        Related_By(crc,InteractionDiagram, id)
        and
        (exists (path(e , id)))
```

Note here the use of quantifier *exists* without having to specify a specific link types path.

Lastly, the third OOHQL expression retrieves all nodes of type *HotSpotCard* and all nodes of type *DocPattern*, documenting class Collection, where that hot spot was implemented using the Template Method Pattern [Gamma95]. This query is supposed to be performed on top of the navigational model corresponding to the framework designer view, that we did not present in this abstract, but we decided to include this query as another example of how to use the *path* operation when we want to specify the desired link types path.

```
3)      select HotSpotCard,
                DocPattern
        from hsc: HotSpotCard ,
             crc: CRCCard ,
             dp: DocPattern
        where (crc className = Collection)
               and
              (dp name = TemplateMethod)
               and
               path(crc, toHotSpotCard, hsc,
                     toDocPattern, dp)
```

Once executed, the query result can be viewed as a sub-hypermedia web, that is, a subset of the source hypermedia application, composed by nodes and links satisfying the query predicate. Of course, the framework documentation application has different user's views, as we stated before, and the query result has the same properties than the corresponding view. This way, the same query can have different results according to the user's view on top of which the query has been performed.

5. Towards the Web: Translating OOHQL query results to an XML representation.

In the past few years, XML [w3c] has evolved to become a standard format for exchanging information between systems, allowing to define flexible data definition schemas within a particular application domain. Many of these schemas has become standards for exchanging information in different domains. Another advantage of this new representation mechanism, it is that allows to clearly separate the way in which data is represented from the way in which the same data will be visualized. More over, we can define different visualizations for the same XML representation, depending on many different characteristics, as for example: the user profile, the visualization platform, the data ordering, etc.

In our case, we are interested in defining a way for representing a query result in XML. This will allow us to standardize the way in which we can interchange documentation between different systems, not only through the web, but through any other required communication platform. The query result will be composed by a set of nodes and their outgoing links, that we can consider as a subset of the queried hypermedia application, or in other words, a sub-hypermedia application. This XML schema will be more than a simple schema for representing nodes and links, will be enhanced with the elements required for expressing framework documentation, like for example, the way in which we are going

to represent CRCCard nodes, Class Diagrams nodes, Documentation Patterns nodes, etc.
So, the resulting architecture for exchanging framework documentation between different systems, or between people who work with the same application framework (of course, within the same application domain) will be composed by the following steps:

1. Create (if it doesn't exist), the hypermedia documentation application for a given framework, using the DocFramework presented in [Sansano2000]. This framework allows to instantiate a hypermedia application for documenting an object-oriented application framework, using well known documentation techniques.
2. Using the OOHQL Query Engine, perform a query in order to specify which specific documentation it is desired to be exchanged. After this we obtain the query result, that is a subset of the hypermedia application mentioned in the first step.
3. Apply the XML schema to the query result obtained before. In this step we obtain a general enough representation of the framework documentation for being exchanged through other entities.
4. Apply the desired XSLT for visualizing the framework documentation contained in the XML files, or export these files to another system in order to be interpreted by any other system. More over, XSLT can help filtering information after performing the OOHQL query, depending on the user profile or the query result client.

The above steps are presented in figure 2.

This architecture makes it possible the development of XML based documentation descriptions (based upon specialized DTD's) capable of representing complex design documents contained in the query result. This XML representation will allow the exchange of pieces of documentation across different teams and different application environments.

6. Conclusions and future work.

We have presented one possible way for organizing the navigational space of an object-oriented hypermedia application used for documenting a framework. These navigational design is formalized using a well known object-oriented hypermedia design model, named OOHDM.

We have shown a way for combining framework documentation organized into a hypermedia application, with hypermedia queries, allowing this way to rapidly access the desired information by querying the hypermedia application with a high-level declarative query language for hypermedia applications named OOHQL.

The fact of having an XML representation for framework documentation, or at least, for query results performed on top of a hypermedia framework documentation application, it is a powerful tool for exchanging and communicating, in a standard way, framework documentation knowledge through people working in the same field.

We are also working in a met-level notification mechanism that allows us to keep framework documentation updated

when changes in the target framework occur. This mechanism is quite implemented and working nowadays.

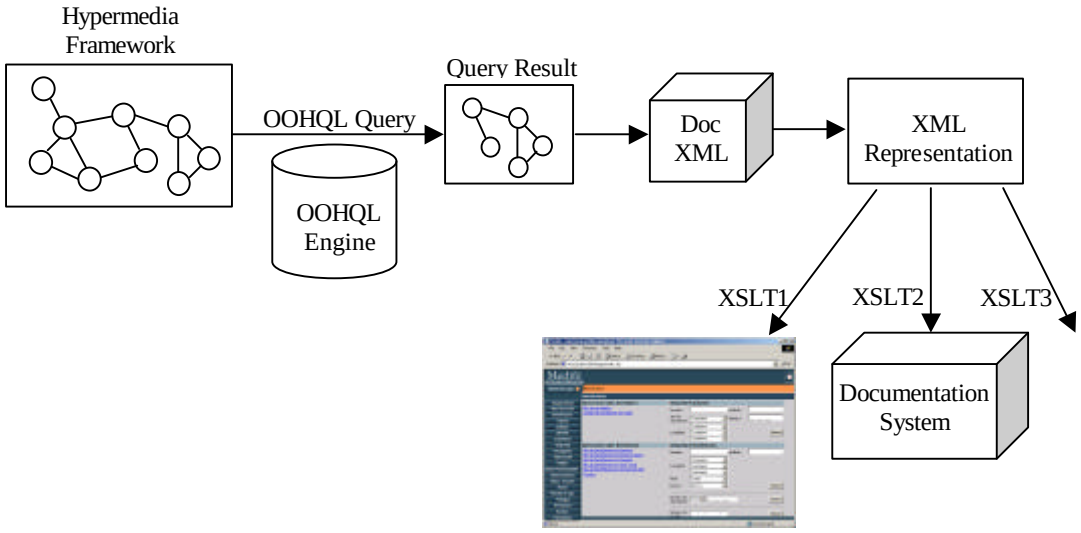

Figure 2. *Documentation interchange architecture.*

7. **References.**

[Bellin97] D. Bellin, S. Simone, "The CRC Card Book.", Addison Wesley Longman, 1997.

[Diaz97] Diaz A., Gordillo S. 1997 "Designing Navigational Contexts using an OO Query Language." Proceedings of DEXA'97. QPMIDS Workshop, Toulouse, France, September 1-5.

[Fayad99] M. Fayad, D. Schmidt and R-Johnson (editors): "Building Application Frameworks.", Wiley 1999.

[Gamma95] E. Gamma, R. Helm, R. Johnson and J. Vlissides: "Design Patterns. Elements of reusable object oriented software". Addison Wesley, 1995.

[Gordillo98] Gordillo S., Diaz A., "An object oriented model for querying hypermedia applications" published in Object Oriented Symposium of the ESDA Conference, vol. 2. pp 133-138, ASME., Montpellier - France, 1-4 July.

[Jhonson92] R. E. Jhonson. Documenting Framework using Patterns. In Proceedings of the OOPSLA'92, ACM SIGPLAN Notices, vol. 27, no. 10, October 1992, pages 63-76.

[Pree95] W. Pree: "Design Patterns for object-oriented software.", Addison Wesley, 1994.

[Sansano99] Sansano Mauricio, Arambarri Federi co, Diaz Alicia, Gordillo Silvia. "Query Execution on an Object Oriented Hypermedia System" MISRM'99 October 30 1999. Orlando, Florida.

[Sansano2000] Mauricio Sansano, Marcos Godoy, Luis Matricardi, Gustavo Rossi, "An architecture for documenting framewo rks.". Workshop on Information Technologies and Systems 2000, 9-10 December 2000, Brisbane, Australia.

[Schwabe96] D. Schwabe, G. Rossi, S. Barbosa, 1996, "Systematic Hypermedia Design with OOHDM". Proceedings of the ACM International Conference on Hyper text, Hypertext'96, pp. 116-128.

[Schwabe98] D. Schwabe, G. Rossi: "An object-oriented approach to web-based application design". Theory and Practice of Object Systems (TAPOS), October 1998.

[UML] J. Rumbaugh, I. Jacobson, G. Booch, "The Unified Modeling Language Reference Manual.", Addison Wesley, 1998.

[w3c] http://www.w3.org/XML/