International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 6, June - 2014

A Novel Information Model For Reusable Software Components Management

José L. Barros J.
E.S.E.I.
Universidade de Vigo,
32004 Orense, Spain.

***Abstract*—Nowadays there is a variety of models for the representation of reusable software elements. International agencies, large manufacturers and academic researchers have proposed models for the purpose of reaching a global agreement and achieve a standard for representing these kind of components. Such standard would facilitate the intercommunication of different tools, both consuming and producing reusable software elements and, certainly, would provide a major boost to software development methodologies based on reuse, both in its compositional side and the generative one.Unfortunately, there is no such standard, and the models proposed so far only apply locally, within institutions that promote them, or in a very small domain. The complexity and the incompatibility between standards have proved to be the more restrictive factors towards a uniform model accepted by most manufacturers and developers.This article proposes a novel information model: configurable, extensible, which can adapt to the specific needs of any software development organization, large or small. The model is based on the categorization of meta-information associated with reusable software elements, expressed in the XML standard and providing direct support to the processes of representation, classification, storage, search and retrieval. The model allows the representation of any information relating to the software element, both the functional characteristics and the non-functional or structural aspects and the different relationships that may exist between them.**

Keywords—software reuse, information models, software components

I. INTRODUCTION

A model is a reduced representation of a system. Its purpose is to help to understand a complex problem or its solution. In this way a model helps us to communicate/analyse ideas about the problem or the solution, and to guide the implementation of the final system. A good model should be:

• Abstract: it emphasizes the important elements and hides the irrelevant ones.
• Understandable: easy to understand by all observers.
• Accurate: faithfully represents the real system.
• Predictive: you can use it to draw conclusions about reality.
• Cheap: much more cheap and easy to build than the real system.

A good model should be useful to:

• Detect possible errors or omissions during the design phase, before compromising resources to the implementation.
• Analyse, experiment and compare various alternative solutions.
• Minimize risks.
• Communicate with: customers, users, and development staff.
• Adequately guide the implementation phase.

In summary, an information model is a tool that allows us to describe a Reusable Software Element (RSE) using the set of attributes that characterise it, abstracting from the real entity. The reason for using a model is to reduce the complexity in the management of the actual RSE, and associated: storage, search and retrieval processes.

Existing models of component information can be classified into three different categories according to their use:

• Models for the description/classification of the components: they try to represent the component in an understandable way for customers and users, facilitating the classification and recovery processes. In essence these models are the data models of the repositories, which describe all the information needed to: find, understand, select and adapt the components. There is a good amount of these models, among the most representatives within the scope of software reuse we can mention:

• SIB [1],
• REBOOT [2],
• RSHP [3] and
• RAS (Reusable Asset Specification) [4]

• Models for the specification/composition of the components: these models try to specify the functionality of the components. We can use these models to specify and build components at the design level. Among the most representatives we had:

• LILEANNA [5],
• Rapide [6] and
• CSCM [7]

• Models for the implementation of the components: they try to help you in deciding how to deploy the components in a given programming language. They are important for reuse because at the end the reusable components should be implemented in an executable system/application. Commonly used ones are:

- COM/COM+/DCOM y .NET [8],
- CORBA [9] and
- Enterprise JavaBeans [10]

The importance of models in software development has caused the emergence in recent years of model-driven development (MDD). It is an approach to the software development process in which the key artifacts are models rather than programs. The code is generated automatically from models, using specific tools for automatic generation and modelling languages. Thus, the model becomes the implementation.

Unfortunately, the proliferation of specifications of models, technologies and development methods in recent times, has caused great confusion among developers, and a big difficulty to integrate and share data, due to the lack of a standard for the exchange of information.

One of the proposed solutions to this problem consists in the transformation of models, a process that is possible if models derive from the same meta-model. Currently these attempts at standardization of a meta-model are directed towards 11 [11]. On the other hand, the use of a standard language for the exchange of data between applications, such as the XML/XMI [12] language, would eliminate inconsistencies between different tools.

Our proposal is based on the assumption that any by-product of the process of software development, with potential for reuse, can be described by a small set of properties that are expressed in a Document describing the RSE (DRSE). The DRSE document is stored in XML format in a repository, from which can be: accessed, retrieved and shared with other applications that are compatible with the XML standard.

II. THE INFORMATION MODEL

The DRSE is composed of three sections which group together those features related with:

1. Non-functional aspects of the RSE: covers aspects of administrative nature, such as: provider or author of the RSE, quality standards, physical location, access restrictions, hardware platform, phase of the life cycle in which the RSE was built, RSE type (see below the valid types of RSE), creation date, last modified date, reuse potential, and, in general, any other information that might be useful according to the criteria of the organization.
2. Functional aspects: describing the functionality of the RSE, its behavior, what it does. This section will have the highest percentage of searches, since the reusers (developers) often express their search as a list of required functional features.
3. Relationships: it contains relationship links between different RSE. Its purpose is to offer the possibility of recovering a set of related RSE, for example: once an RSE of type requirement, that satisfies the query of the developer is found, we are also interested in retrieving the RSE models, designs, code, tests and documentation, related to this requirement (first retrieved RSE).

For the purposes of determining the most appropriate structure for the DRSE, and the most relevant content to be included in each section, we have done an analysis of all the possible RSE that may appear during the process of software development. These potential RSE were grouped in categories and a set of properties was established both: for any type of RSE (general properties) and for specific types of RSE (restricted properties). On the previous set a process of factoring was applied and using the inheritance mechanism the common properties were extracted to build abstract classes of RSE. The resulting set of valid RSEfrom the categorization process is shown in the following figure (abstract classes).

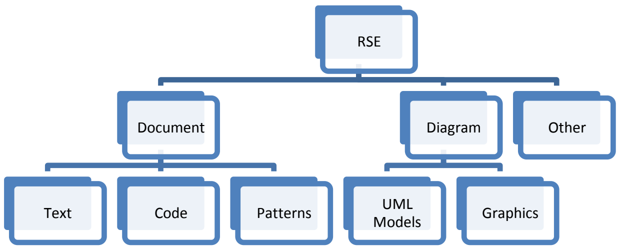

Figure 1 Model of abstract classes representing all possible valid RSE

The following figures show some of the possible valid types of RSE in two well-known categories.

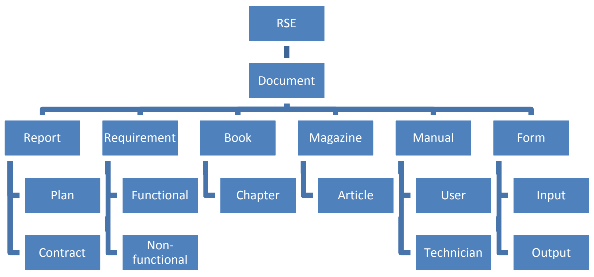

Figure 2 Valid RSE in Document category and the inheritance relationship

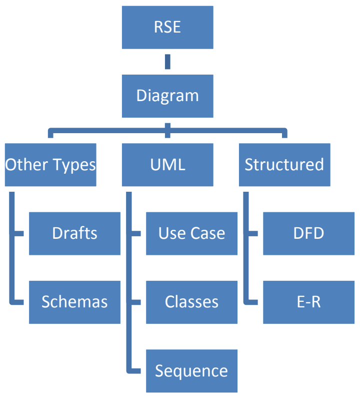

Figure 3 Valid RSE in Diagram category and the inheritance relationship

Based on this information we build the structure of the three sections of the document DRSE:

1. Administrative Description (AD): Non-functional content (for management activities, including control, quality assurance and so on).

2. Functional Description (FD): consisting of tuples of the form <descriptor:importance>, where descriptor is also a tuple<Term1;Term2>. Where:
   Term1 refers to an actiónwhich is apply on Term2; and
   Importanceis a numeric value that indicates the relevance of the descriptor within the set of all descriptorsthat comprise the FD.

3. Relationships Description (RD): consisting of tuples with the syntax <TypeOfRelation;Related_RSE>

III. Main Processes

Procedures for the semi-automatic extraction of information from the RSE, in order to populate the properties specified by the model in each of the "Description" sections were designed. Such procedures facilitate the generation of the documents DRSE and, in some cases, completely automate it.

DRSE documents are stored in an XML native database in which a mechanism for grouping based on the calculation of similarity between documents, organizes them (together) in clusters of similarity [13].

Queries can be constructed in different ways, offering greater flexibility to developers:

1. Direct queries: using the query language of the database management systems (DBMS) and the featuresin AD and RD sections (which have been indexed in its entirety).
2. Queries in the form of a Functional Description (FD section): the reuser build up a new FD which is then compared with stored FD's. When a similar FD is located the system returns the whole DRSE and all the other RSE which have a relationship link with the one located (the similarity cluster is retrieved).
3. An UML diagram: the reuser builds a diagram (UML) that is exported to a file with XMI format (XML subset), then the established procedures extract the needed information to build a complete DRSE related to this diagram, the FD section of this document is then isolated and the technique described in step 2 is finally applied.

Another additional process consists of the dynamic creation of a controlled vocabulary, to which Term1 and Term2 in the FD are added. This vocabulary represents the domain of the application and is useful to the development organization for the purpose of reducing the problems of communication between members of the development team and final users.

IV. Conclusions

The research has shown, so far, that any by-product of software development can be stored in a file in a computer storage medium: HDD, database, Web server and so on; and that the contents of these files can be fully described by a text document using meta-information (a set of product properties). In particular, the text document may be created in XML format.

The developed information model allows representing any RSE, no matter: size, granularity, stage of life cycle where it was obtained, language, format or physical location. The model is homogeneous; it means that the very same model is used for any RSE (no special add-ons or changes).

The model is easily configurable to adapt to the specific needs of an organization's software development methodology. It is only necessary to redefine the content of the different sections that make up the DRSE, by adding or removing the required XML elements. This feature enables the scalability of the model.

The model covers all aspects that may be relevant in a query (requirements of the systems, developer's queries), both the functional and non-functional. In addition, the inclusion of a section with information on relationships allows recover, at the same time, a whole set of related RSE, expanding the probability to reuse (find something useful).

The process of grouping allows factorizing a RSE group common functionality, offering guides and recommendations for future development of more complex (generic) RSE.

The developers can access the DRSE by means of: formal queries, text queries, navigation, comparison of diagrams, keywords and so on. This flexibility allows different types of users, with varying degrees of expertise to be able to take advantage of the repository of RSE.

In relation to technology, to implement the model, repository, and processes of search and retrieval, all the necessary tools are open source software, providing independence from a manufacturer or a specific operational platform.

REFERENCES

[1] Constantopoulos, P., Dörr, M.: "Component Classification in the Software Information Base", Object-Oriented Software Composition. O. Nierstrasz and D. Tsichritzis (Eds), Prentice Hall, pp 177-200, 1995.
[2] Karlsson, E.A.: "Software Reuse: A Holistic Approach", John Wiley & Sons, 1996.
[3] Lloréns, J., Morato, J., Genova, G., Fuentes, M., Quintana, V., Díaz, I.: "RSHP: An information representation model based on relationships", In Ernesto Damiani, Lakhmi C. Jain, Mauro Madravio (Eds.), Soft Computing in Software Engineering (Studies in Fuzziness and Soft Computing Series, Vol. 159), Springer, pp. 221-253, 2004.
[4] http://www.omg.org/spec/RAS/
[5] Tracz, W.: "Lileanna: a parameterized programming language", In Proceedings, Second International Workshop on Software Reuse, pages 66-78, March 1993. Lucca, Italy.
[6] Luckham, D.: "Rapide: A language and toolset for simulation of distributed systems by partial orderings of events" In DIMACS Partial Order Methods Workshop IV, July 1996.
[7] http://doi.ieeecomputersociety.org/10.1109/EURMIC.2004.1333358
[8] http://www.microsoft.com/com/default.mspx
[9] http://www.omg.org/technology/documents/formal/components.htm
[10] http://docs.oracle.com/javase/7/docs/api/java/beans/package-summary.html
[11] http://www.omg.org/gettingstarted/overview.htm
[12] http://www.w3.org/TR/2000/REC-12-20001006
[13] Barros, J., Marques, J.: Conglomerados Multidimensionales: Un mecanismo simple de organización de Elementos Software Reutilizables. JISBD'02, Madrid, Noviembre, 2002. Pp.: 375-386.

AlltheInternet links werecheckedon9th of June, 2014