Building Conceptual Models by Knowledge Management Methodology

Nicolas Blanc, Thomas Montroig, Joseph Roumier, Laurent Romary, Jacek Szymanski, Denis Reboul

Building conceptual models by knowledge management methodology

N. Blanc, T. Montroig, J. Roumier, L. Romary, J. Szymanski, D. Reboul

PROTEUS ITEA European Project

URL : http://www.proteus-iteaproject.com

***Abstract* - The article proposes to use an alternative approach to design and validation processes of loosely integrated distributed co-operating systems. The example of such processes constitute the basic subject of ITEA project PROTEUS**

The PROTEUS project is intended to integrate without major functional modifications existent software tools which originally are not designed to work in co-operation. The tools cannot be modified profoundly solely in view of their integration, only a supplementary layer of data interpretation can be introduced

The fact that the tools mustn't be modified in order to follow the integration requirements creates the need of a top level description. Commonly used UML representation of overall tool service is not sufficient to provide distributed view on the co-operation scenarios. Most naturally the UML description is the best adapted when one attempts to obtain a semi automatic code generation. It is not good although look for coherence proofs. In order to conciliate the divergent criteria it is proposed to use a higher level description language.

This article intends to show the potential of knowledge management methodology used to build conceptual models of domains as a means for integration of UML representations of distributed systems.

I. INTRODUCTION

The Proteus project is aimed to integrate existent software tools not designed to work in co-operation in a complete system based on information exchange. By this way, we are considering that each tool brings services that will be used to help the end-user to deal with its tasks within the maintenance operations. So, Proteus is a Service Oriented Application dedicated to the maintenance field, thanks to the use of existent software it is intended to gather and manage.

From the end-user point of view, all the operations are performed by the same tool: the general Proteus Application. But actually, taking all the architecture into account leads the fact that each action implements many calls to services spread among the various tools used in the field of maintenance. Moreover, whichever is the concrete Proteus application, the main functionalities will be the same in spite of the differences revealed on the tools that are chosen for the application.

These observations reveal that even if Proteus is designed as a fully distributed system, a centralised organisation needs to be brought as a keystone of all the architecture. This keystone is identified as the "Proteus Information Repository" (PIR) and clearly puts the stress on the close co-operation needed between tools by furnishing them an unique way to communicate and by listing the possible existing link between resources thanks to a representation of the application domain.

From the project point of view, all the partners working on the research tasks need to specify how the tools they are describing will communicate (this is the ideal description of a tool). To do so, they also need to know how the other tools communicate when they need to use information they do not handle. The problem raised here is a computer-oriented vision of the application needed to describe the tools and the services. Here again, the notion of "Proteus Information Repository" helps the researchers to have a common basis on which to work: it provides common languages to describe architectural resources and services.

The main key-point of the Proteus Information Repository is its ability to combine two complementary approaches designed to be used at different points but describing the same notions from different points of view. These approaches and their full ability to cohabit are the subject of this paper which will first of all describe the different aspects of the description before showing how the strong link is built between them. Then, the use of these models will be shown at different steps of the Proteus platform implementation.

II. TWO ASPECTS OF A DESCRIPTION

A. Describing distributed services: a common language

Since the project includes many partners working on different issues but with the same goal, a common language needs to be used in order to describe every processes. This language must enables every partner to:

- have a quick general view of a process,
- define precisely the tools on which the partners are working,
- get information without having to be an computer expert.

These needs lead to use a tool as simple as possible which could describe all the tools to enable efficient communication within the project. The language proposed is UML (Unified Modelling Language) because of its graphical view understandable by a non computer-specialist but very close to the structures it describes.

Thanks to this language, the description of the ideal tools becomes possible. Describing ideal tools is essential in the scope of the project: in the architecture of the final application, Proteus must be adapted to the concrete application. That is to say that without knowing which will be the functions of the tools that will be used in a concrete application, a corresponding ideal tool must be described. A tool may be defined as a set of services and in the scope of maintenance, we find the following tools:

- CMMS: Computerised Maintenance Management System
- SCADA: Supervisory Control And Data Acquisition
- ERP: Enterprise Resource Planning
- Expert System
- E-documentation server

Whichever will be the concrete tool, a component will be added to adapt it to the Proteus requirements. This component is an ICA (Intelligent Core Adapter) which will be the bridge between an ideal description and the concrete tool. Thus, from the Proteus point of view, there will be allways only ideal tools with well defined interfaces.

B. Describing a domain: a knowledge representation

a. Why an ontology ?

To ensure te consistency between the differents elements that compose an industrial domain, to permit their cooperation, it was mandatory to use a common representation of the information that lie in the domain [Huber02]. This knowledge representation has to be built to achieve a goal, this task is user-driven [Noy00]. This is why in the Proteus project, we were lead to propose an Ontology for Industrial Maintenance, in which the key concept is the Equipment.

The definition of an ontology we chose for Proteus is basically a "formal description of concepts in a domain of discourse […], properties of each concept describing various features and attributes of the concept(slots (sometimes called roles or properties)) and restrictions on slots(facets (sometimes called role restrictions))" [Noy00].

Moreover, an ontology or any other categorization system for knowledge is a way to access more easily and faster to the required information [Rosch78]. Industrial Maintenance deals with many different kind of concepts and services, and the amount amount of information to deal with might be very challenging, this is why using an ontology is required.

b. An ontology for Proteus

The ontology designed for Proteus is entitled the PIR (Proteus Information Repository). It is designed to allow the highest interoperability and compatibility with other systems and applications.

For these purposes, the PIR is modular following two axes :

- a vertical axe (cf. fig.x.x) devided in three layers of increasing specificity,
  - GMO (General Maintenance Ontology, cf fig y.y.) composed of the key concepts for industrial maintenance
  - Application Profile, which contains the information dedicated to a specific industrial domain (Plane Manufacturing, Electrical Engineering…)
  - Industrial Contexts which is particular to a given company (Cegelec, AKN, So,ny…)

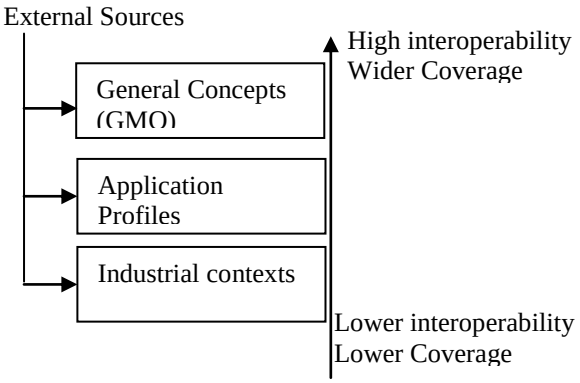


Figure x.x PIR vertical modularity

- A horizontal axe for the upper layer (GMO), composed of nine concepts(cf fig.y.y) that cover all the concepts used for deeper modelization.

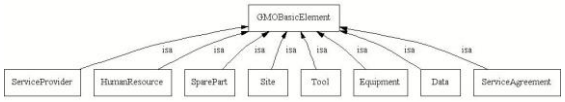


Figure y.y PIR horizontal modularity at GMO level

III. LINKING THESE TWO ASPECTS

A. Applying services to a concept

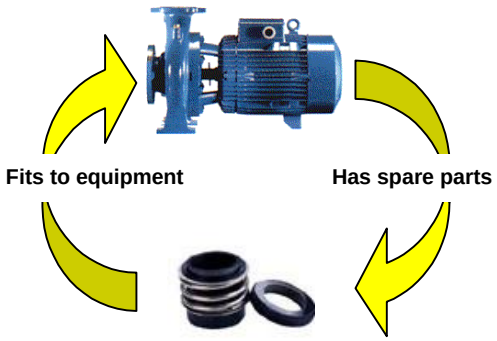
To link the UML representation with the ontological one, we need to list for each instance (resource) all the available applicable services.

This resource-oriented representation wills permit us to consider that the resources are handled by services furnished through external software tools.

For this reason, a table will be use to summarise the set of methods that can be applied on a basic resource:

Resource	
Service 1	Method 1.1
	Method 1.2
Service 2	Method 2.1
	Method 2.2

Let's consider an equipment to maintain, a pump, and its spare parts. Some services are dedicated to provide them methods to be handled, as represented below:



The services applied on these two resources are:

Pump	
Spare Parts Management	getSparepart(idEquipment)

Spare parts	
Spare Parts Management	getEquipment(idSparepart)

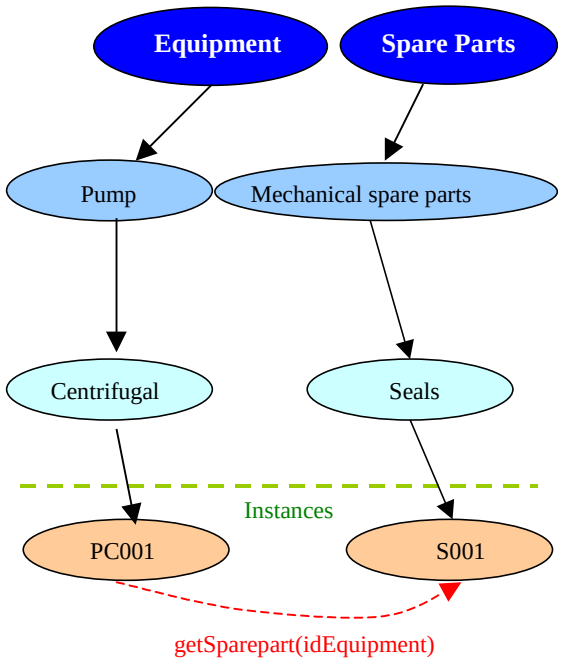
We will also suppose that an Equipment has an attribute giving a unique identifier enabling the various tools to identify it uniquely by always passing it as a parameter when the method of the service is called.

B. Setting top level concepts (GMO stabilisation)

The GMO should be the representation of the basics of maintenance resources that must be handled to perform a full maintenance process. This point of view introduces a maintenance-oriented description in order to have a more adapted tool.

The resources described in §3.1 will have to comply with the ones listed in the GMO.

The GMO representation makes it possible to describe and classify the resources with a hierarchical point of view. The services identified by the representation exposed in the previous sections then links the resources together: an attribute described in the ontology is by this way represented by a service furnished by a "Service Provider" (such as CMMS, E-Doc, …). By this way, at the implementation point of view, an attribute in the GMO is represented by a method found in a service set within a maintenance tool.



C. A continuous process

We will now consider the job of every partner in charge of the services described in §3.1 that will list the method that can be applied on the basic resources. For example each partner will find in the GMO a class called "Equipment" with no method listed at this starting point. Then, he will complete the table with the methods he provides.

The partners are organised in working group. The proposal will be validate for each working group and then widespread to all.

A specific task force, the PIR task force, does the validation process of the proposals. In each working group a corresponding mender of the PIR task force will help in the definition of the services. As the Task Force gathers application experts and modelling experts, the model will be consistent and very close to the description needed to perform a well definition of the domain.

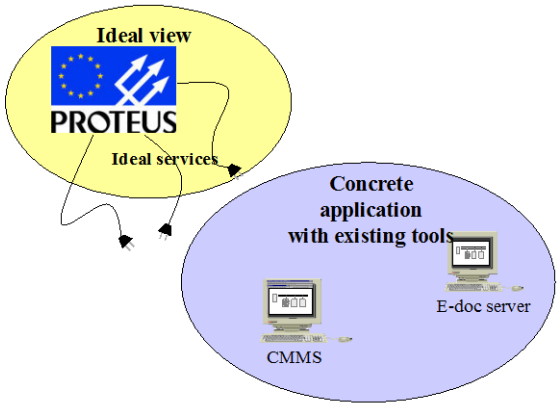
IV. USAGE OF THESE DESCRIPTIONS

One of the first usages of this multi-oriented approach is a more efficient organisation of the tasks performed by the various partners by having common basis on which to work. But another aim for the notion of a centralised information

repository is also to be ran in a production implementation. This concrete application may be used at two steps of the implementation: the configuration step and the running step.

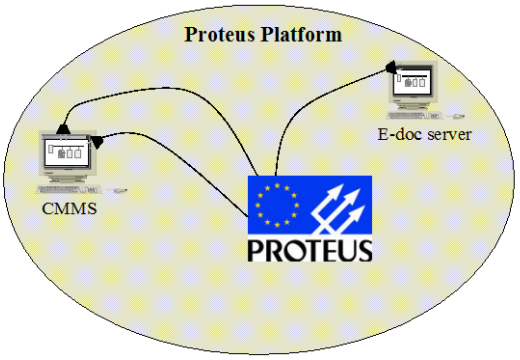
A. At the configuration step

The first step when setting up an installation is to configure it. In the case of the Proteus platform, the configuration stands for the listing of the available services and the comparison with the attended services.



Indeed, the conceptual Proteus platform describes a set of ideal services that need to be furnished by the software maintenance tools such as SCADA, e-documentation server, expert systems, CMMS and ERP in order for the platform to run efficiently. All the services described as explained in § 3 must then be located in the scope of the real implementation. Everything would be easy if all the software tools that can be found on the market had the same functionalities. But this is unfortunately not the case and depending on the installation that has already been performed before the decision to implement the platform on a concrete site, some services may be found in a tool or another (sometimes in several ones). Starting from the observation that the platform must be adapted at the application field without having to modify any of the existent tools, the issue becomes here: how to determine which real tool is the most representative of an attended function ?

The solution is here again to use the centralised repository known in the scope of Proteus as the PIR which will bring enough information to design a method to configure the platform.



As the PIR gathers a description of all the basic concepts on the instances of which the ideal listed services can be applied, the parallel must be done with the already installed tools. Actually, with a well implemented configurator, the configuration becomes an audit consisting in listing the available services with their location (the tool implementing the service). Then, thanks to the configurator tool, each ideal service will have an equivalent in the concrete tool and the linking of the “ideal description” and the “real description” will build the platform.

Each ideal service is consequently located and the platform is ready to be launched.

B. At a running step

As mentioned in §4.1, the PIR contains all what is necessary to have the platform running. But actually, another condition is required to get a fully functional platform: the knowledge of the structure of the services. This knowledge is here offered by the normalisation of the interfaces defined for every set of service: the ICA. Having the information of the location of a service and its description enables so to get a complete running system as show the following scenario.

Let’s consider a service called to order Spare Parts relative to a given Equipment. This service knows how to calculate the cost of a Spare Part, where to send the order and all other ordering relative operation, but it can’t find by itself the Spare Parts needed by an Equipment.

That’s why it needs to call the service dedicated to find this type of resource linked to an equipment. As it is designed to work in co-operation with this other service, the ordering service exactly knows how to call the service if it is available: it knows the method to call, the parameters to give and the structure of the returned result. The information it needs is: where can I find the service ? Is it available on this platform ?

The consequence of these question is that the service must search for the required information. As previously explained, this kind of information can be given by the PIR. So, the service we are dealing with will call the PIR to know if the type of information needed is available on this platform. If it is, the location (address) of the service will be used to call it.

The power of the PIR notion is here again demonstrated: by its full integration in the project (from the conception of the project to the running mode of the platform without forgetting its ability to get a platform configured), it gathers all the information needed to get quick and relevant access to specific data.

V. CONCLUSION

This approach of the description of a domain modelized differently depending on the intended application shows the ability to make cohabit two complementary visions. Thanks to this knowledge management methodology, a domain can be described by specialists of the application field in order for computer specialists to develop special tools on the same application field, without having to be experts in the domain. This brings a powerful collaboration enlarging the capabilities of tools dedicated to an application field (here industrial maintenance) with a strong link to the needs reported from the end-users.