# OPHELIA - Open Platform and metHodologies for devELopment tools IntegrAtion in a distributed environment

Maciej Hapke*
*Institute of Computing Science, Poznan University of Technology, Poland*

Andrzej Jaszkiewicz
*Institute of Computing Science, Poznan University of Technology, Poland*

Sergio Perani
*Omega, Milano, Italy*

**Abstract**

This paper outlines a research and demonstration project which addresses two important issues of software development: integration of software engineering tools in an open platform and the use of software engineering tools in distributed environment.

## 1. Introduction

Software engineering projects intensively utilize various kinds of software tools. They include CASE, requirements engineering, project management, measurement, bug report and documentation tools. Many of the tools use common information that is repeatedly entered and stored in different tools repositories. It results in additional workload and data inconsistency. Clearly, the value of particular tools may be increased by their integration.

* Praca finansowana z grantu KBN DS 43/639 i projektu EU „OPHELIA”

In addition many software projects are performed in distributed environment. Many large commercial projects run by international companies involve several outlets placed in distinct locations. Distributed development is also typical to open source projects. A disadvantage of many software engineering tools is their inability to work in distributed environment, where usually only leadership is centralized.

The goal of this paper is to outline a research and demonstration project which addresses two important issues of software development:

- integration of software engineering tools in an open platform,
- the use of software engineering tools in distributed environment.

The project is founded by the 5th Framework Programme of European Union. It is run by a consortium composed of four software companies - Centrisa (Spain), Gut Consult (Germany), Omega (Italy) and I.C.C.C. (Czech Republic), and two universities - Poznan University of Technology and Heriot-Watt University. The project has been registered under the acronym OPHELIA – and its full title is Open Platform and metHodologies for devELopment tools IntegrAtion in a distributed environment.

There are very few existing solutions that address the two issues. They are, however, closed commercial platforms. For example Rational produces several highly integrated tools for project management, system definition and design [8]. In addition there have been several attempts to define standard for the tool integration and the data exchange, but without obtaining appreciable results: for instance **ATIS** (A Tool Integration Standard) developed by Atherton Technology, which defined object oriented interfaces to the common data repository. It was used by DEC (CDD/Plus Repository) and by IBM (AD/Cycle Repository); then the ATIS experience has produced **CATIS** (Common Application and Tools Interface standard) and **CIS** (CASE Integration Services), but all those efforts have not been commercially successful. The **PCTE** (Portable Common Tool Environment) is a tool support interface specifically to provide a tool integration infrastructure in software engineering environments [5]. Despite extensive support for PCTE, especially from Europe, and international standardization, PCTE has not been commercially successful.

In contrary to the above projects the goal of the Ophelia project is to define a publicly available platform and implement it as open source software. Furthermore, for each main class of modules an exemplary tool (if possible open source) will be integrated with the platform. The specific goals of the projects are:

- Provide a common platform where the development tools can be easily integrated. This platform should be usable in distributed environment, independent from O.S., browser based, with an open source license and, when possible, auto-configurable.
- Define how existent tools can be integrated into this platform.

- Study the problems of interactions between the actors in the development phases; define criteria of analysis and improvements, so that the platform together with the new and old tools can be used inside existing development processes with reduced costs and increased efficiency.
- Extend existing tools, which will demonstrate how to have a complete integration with the OPHELIA platform.
- Prepare a software distribution plan with all the development results of the project.
- Promote the advantages of the OPHELIA platform.

## 2. Ophelia as an open integration platform

Ophelia's consortium has decided to use a new approach towards the creation of a platform for distributed development. In fact, the major attention is given to modules integration and tools' addition rather than to the implementation of a unique closed development environment. This last approach is normally followed by large organizations willing to impose their systems; Ophelia on the contrary will adopt the approach to focus only on the integration and to help organizations and tool developers to create their development environment (to use or to sell) based on the OPHELIA platform.

Another relevant characteristic of the Ophelia system is the willingness of creating an environment where tools implementation is kept apart from functionalities, thus enabling the substitution of a tool with a similar one without any troubles in the system functioning. This means an evident openness of the Ophelia environment towards users' available tools; customers indeed have much more possibilities of integrating their tools with few troubles and with minimal changes in methodological and practical working habits.

In the future the platform will allow integration of new, both commercial and open source tools. The tools could be integrated by either producers of the tools or third-parties that will adapt existing tools to Ophelia interfaces.

## 3. Architecture and technology

The main idea of the Ophelia platform is to integrate a number of tools into a single distributed software development system composed of several modules. The modules will share common data and information supplied through one of these modules will immediately be available to others, in a transparent way. The architecture of the platform is given in Fig. 1.

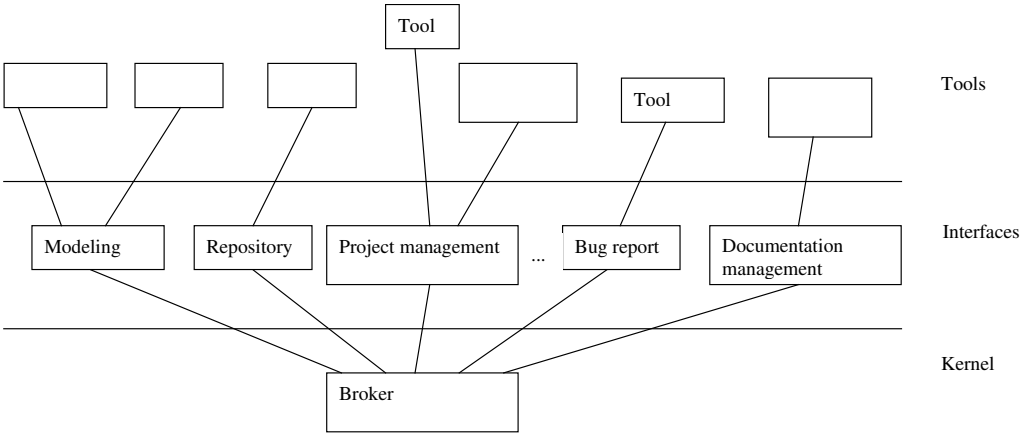



Fig. 1. The architecture of Ophelia platform

The platform will use XML based data formats and CORBA interface for modules communication. When possible we will use existing XML-based formats, e.g. XMI for the exchange of modeling data [7] and recently proposed project management formats [6].

The first level, kernel, defines the basis for the whole architecture. The only component of this level is the broker, in charge of synchronization and information exchange among different modules. There will be a broker for each server. The broker does not provide any storage functionality; it is only responsible for establishing connection between tools. Repository is one of the tools plugged in Ophelia platform.

The most important level is the intermediate level that defines the set of module interfaces. This is only a logic level (i.e. no application is contained in this level). The interfaces define:

- the module types,
- the specific behavior of a specific module,
- the data exchange with the broker or through other modules and the kind of available data.

The third level is composed of a number of applications, libraries etc. implementing or using interfaces of the previous level. There are 2 categories of tools:

- Pre-existing tools (e.g. MS Project), for those that are add-in designed. It will not be always possible in this case to adapt a tool, it might either be too expensive, or there will only be a limited set of functionalities available.
- New tools specifically developed for this architecture.

Particular partners of the Ophelia consortium are responsible for different modules of the system. Institute of Computing Science from Poznan University of Technology is responsible for project management module and its integration with modeling (CASE) module. In the next section we describe in more details the aspect of integration of the modules in Ophelia platform.

# 4. Integration of CASE and project management tools

Project management is a set of activities having crucial influence on the success of a software projects, especially in the case of middle and large size projects. Software companies intensively use commercial tools, e.g. Ms Project. An important disadvantage of such tools is their lack of integration with other software development tools. This fact increases the project manager workload and may result in lack of synchronisation between related information managed by different tools.

The project management (PM) module will be based on Ms Project, the market leader in this area. As at present no open source tool can be a real competitor of Ms Project, the best approach is to adapt Ms Project to the developed interface. The interface, however, will be general and will allow future adaptation of new project management tools to the OPHELIA environment.

The project management interface will support the following main functions:

- project specification - definition of tasks and their parameters, definition of resources, definition of previous constraints,
- project scheduling,
- project monitoring and tracking,
- access to the project specification and schedule.

In order to adapt Ms Project to this interface an add-in will be developed. The interface will also support integration of Project Management module with other modules, e.g. with modeling module, i.e. it will support information exchange between modeling and project management tools. This kind of integration will assist the project manager by inducing information about project tasks and their precedence constraints from an UML based modeling tool repository. For example, information about a use case can be automatically transformed to definition of the following tasks: the use case design, implementation and testing. Furthermore, use cases relationships can correspond to some precedence constraints. Project manager will be able to see details of use cases corresponding to particular tasks. For example, it will be possible to see sequence diagrams associated with a use case. Such information may be helpful in defining duration and resource requirements of the tasks. On the other hand, the

user of modeling module will have access to information about scheduling and realization of the tasks related to selected use cases.

Task parameters and relationships between them depend on software development model used in a given project. It is planned to implement a specialized wizard tool (see Fig. 2) that would guide the project manager in defining a new project. The wizard is going to support waterfall, prototyping and incremental (evolutionary) software development models and possible hybrids. Moreover the tool will ensure information synchronization. If after transforming data to project management module a modification in CASE tool is done, an appropriate change in project definition will be also made. Events like these will be submitted and then served using CORBA services.

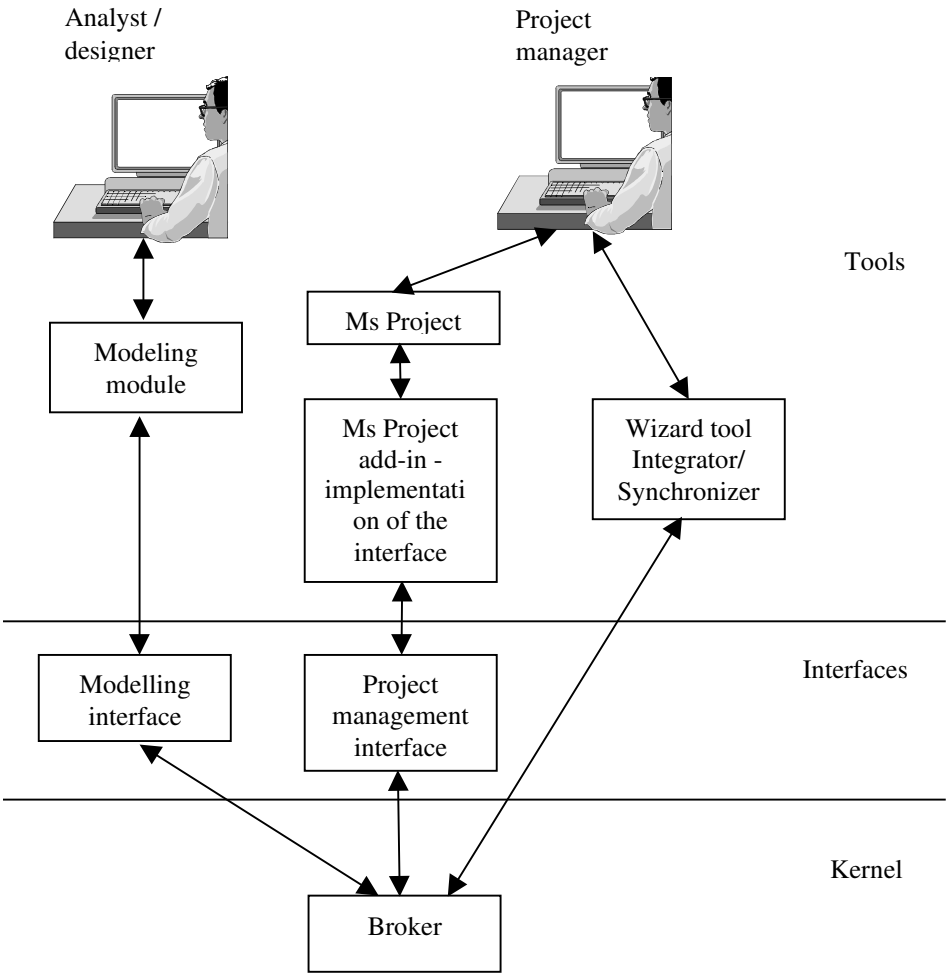
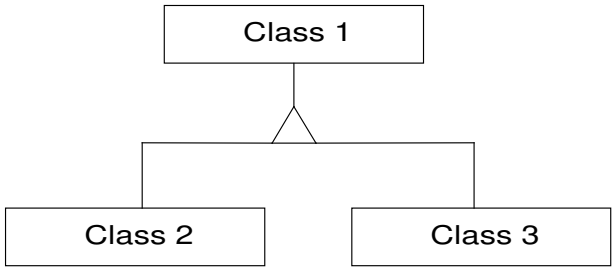



Fig. 2. Integration of CASE and project management tools

In [3] the first two authors proposed a methodology for transformation of UML models to project management tasks. This methodology was implemented in SISOP prototype system that integrated Select CASE tool with Ms Project. In contrary to Ophelia approach, however, SISOP was a closed tool-specific system based on OLE interface. In Ophelia platform we will use this methodology updated and extended on the basis of comments and remarks gathered during pilot applications of SISOP.

We support several approaches to definition of project tasks on the basis of UML. The tasks may correspond for example to classes or use cases. With each class (use case) we associate three tasks – detailed design, implementation and testing of the class (use case). In addition, some relationships between classes (use cases) may be optionally transformed to precedence constraints. The project manager may require detailed design (implementation, testing) of a base class to be completed before detailed design (implementation, testing) of its specializations.




| ID | Task Name | Duration |
|---|---|---|
| 1 | Design of class 1 | 2d |
| 2 | Implementation of class 1 | 2d |
| 3 | Tests of class 1 | 2d |
| 4 | Design of class 2 | 3d |
| 5 | Implementation of class 2 | 2d |
| 6 | Tests of class 2 | 4d |
| 7 | Design of class 3 | 2d |
| 8 | Implementation of class 3 | 2d |
| 9 | Tests of class 3 | 3d |

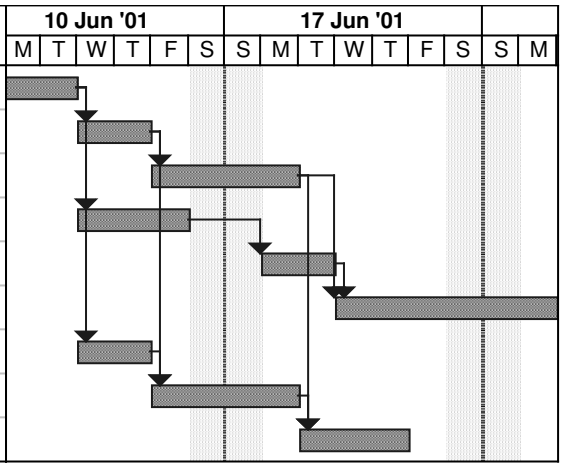



Fig. 3. Class diagram and corresponding tasks

In addition the project management module will allow planning and tracking to be updated in a semiautomatic way thanks to information got from other modules. For example:

- getting back those components not assigned through the modeling module,
- calculating estimations of project completion from the bug report module or from a possible test module

Moreover other modules can be used as well to get a more intuitive user interface. For example, one could use as well a sight on the model showing system components to associate developer activities: such sight gives something more with respect to a simple activity list.

# 5. Data storage

The data of the project management module will be stored in XML format. An XML schema for Project Management has been proposed by OASIS [6] but since it does not include information about resource requirements it has to be modified by Ophelia consortium.

On the other hand, XMI (XML Metadata Interchange Format) specifies an open information interchange model that is intended to give developers working with object technology the ability to exchange programming data over the Internet in a standardized way, thus bringing consistency and compatibility to applications created in collaborative environments. Among others XMI covers the transfer of UML models. It identifies standard XML DTD's to allow the exchange of UML information.

At present a number of commercial or open source projects use XMI, e.g.: IBM WebSphere, VisualAge for Java and XMI for Rose toolkit, Unisys UREP, IntegratePlus and XMI for Rose toolkit, Oracle Corporation Designer 2000, Rational Software Rose 2000, Softeam, Together/E, Meta Integration Technology Inc., metadata bridges, Argo/UML, NSUML library1.

# 6. Prototype disoda! system

Poznan University of Technology has developed already a prototype system integrating ArgoUML and Ms Project called disoda! (**D**iagram-**I**n **S**chedule-**O**ut **D**istributed **A**pplication). The prototype is based on XML standard described in section 5. The prototype system supports specification of a new project schedule on the basis of UML design. Its functionality includes, however, only limited synchronization of the schedule and UML design after further changes.

The prototype system is composed of four modules. Server module is responsible to storing and accessing XML files constituting a project repository. It allows also managing projects and users through a WEB interface. ArgoUML interface module is used to uploading and downloading ArgoUML files (XMI and PGML) from the server. The module is relatively simple, since XMI and PGML are the main formats used by ArgoUML tool. The project management module allows storing and reading all scheduling data in XML format, as well as uploading and downloading the files from the server. The integrator module exchanges information between CASE and project scheduling tool. It works only with XMI and XML files and communicates with server module only, thus it is completely independent on the particular tools used in the prototype.

Note, that ArgoUML interface and project management modules allow distributed use of the two applications. In particular project management module may be seen as an alternative to Ms Project network installation. ArgoUML is at present a standalone application without any support for distributed work, thus our system significantly increases ArgoUML's functionality.

All modules except of PM module are implemented in Java. PM module is implemented in Visual Basic (Ms Project add-in responsible for reading scheduling data to and from XML file) and C++ (network communication module). The integrator module uses JAXP XML parser (**J**ava **A**PI for **X**ML **P**arsing) from Sun using DOM method. The project management module uses Ms Widnows XML parser.

More information about disoda! is available at www.disoda.z.pl. The system is open source software. The successful implementation of the system proves that it is possible to integrate commercial tools and open source tools in a common platform.

## 7. Planned project dissemination

The project starts in July 2001 and is going to be completed by June 2003. First results will be available after one year. In the second year the consortium will focus on further improvement and extension of the platform based on evaluation by potential users and on promotion of the platform. The promotion of the platform architecture is one of the main tasks. It will be important to have a feedback from the development community in order to estimate the results and quality of the partners work.

## 8. Summary

Although the work on OPHELIA platform is at very preliminary level, particular partners of the consortium work intensively on several prototypes which main goal is to test technical possibilities of integration of particular both

commercial and open-source tools in a single platform. The disoda! prototype system described in this paper illustrates the possibility of integrating open-source CASE tool ArgoUML with a commercial software Ms Project with the use of XML. Of course, integration of a given tool with the platform will not always be possible. It requires the tool to be able to save and read data in required XML format and/or to deliver an appropriate programming interface.

## Bibliografia


[1]. M. Hapke, A. Jaszkiewicz, P. Kominek (1999), Zintegrowane narzędzia harmonogramowania przedsięwzięć programistycznych w warunkach niepewności. Materiały I Krajowej Konferencji Inżynierii Oprogramowania, Kazimierz Dolny 11-13.10.1999, Informatyka Stosowana S4/99, Politechnika Lubelska, 65-76.

[2]. M. Hapke, P. Kominek, A. Jaszkiewicz, R. Slowinski (2000), Integrated tools for software project scheduling under uncertainty. In: P. Brucker, S. Heitmann, J. Hurink, S. Knust (Eds.) *Proc. 7th Int. Workhop on Project Management and Scheduling PMS'2000*, Osnabrueck, Germany, April 17-19, 2000, pp.149-151.

[3]. M. Hapke, A. Jaszkiewicz, R. Strugalski (1997), *Integracja narzędzi CASE z narzędziami harmonogramowania przedsięwzięć*, Raport naukowy Instytutu Informatyki Politechniki Poznańskiej, **Nr RB-0/97**.

[4]. http://www.argouml.org

[5]. http://www.ecma.ch/ecma1/STAND/ECMA-149.HTM

[6]. http://www.oasis-open.org

[7]. http://www.omg.org

[8]. http://www.rational.com