AN INFORMATION SYSTEM DEVELOPMENT TOOL BASED ON PATTERN REUSE

Agnès Front-Conte, Ibtissem Hassine, Dominique.Rieu et Laurent Tastet
LSR-IMAG Laboratory, SIGMA team, BP 72, 38402 SAINT MARTIN D'HERES CEDEX – France

Keywords: Pattern, patterns system, reuse, pattern-based development tool.

Abstract: A pattern is a general and consensual solution to solve a problem frequently encountered in a particular context. Patterns systems are becoming more and more numerous. They offer product patterns or process patterns of varied range and cover (analysis, design or implementation patterns, and general, domain or enterprise patterns). New application development environments have been developed together with these pattern-oriented approaches. These tools address two kinds of actors: patterns engineers who specify patterns systems, and applications engineers who use these systems to specify information systems. Most of the existing development environments are made for applications engineers; they offer few functionalities allowing definition and organization of patterns systems. This paper presents AGAP, a development environment for defining and using patterns. Not only does AGAP address applications engineers, but it also allows patterns engineers to define patterns systems.

1 INTRODUCTION

For several years, in many domains, the increasingly permanent use of the information systems has revealed the need to capitalize and reuse the professional know-how. From this need emerges, in information systems engineering, a new activity that is the components reuse. A wide variety of reusable component models have already been proposed to integrate reuse in all applications development processes: generic domain models (Maiden, 1994), analysis patterns (Fowler, 1997), design patterns (Gamma, 1995), frameworks (Johnson, 1992), etc. In this paper, particular interest is given to the pattern approach.

A pattern is considered as a tested and accepted solution to a problem which occurs frequently in information systems development. Patterns are generally organized in patterns systems (Rieu, 2002). A patterns system defines a pattern collection with rules allowing to combine them (exp. Gamma patterns system, Ambler process patterns system (Ambler, 1998) or Gzara PIS patterns system (Gzara, 2000).

Several works integrate patterns in application development environments (Borne, 1999). This integration aims to automate their use and their application in concrete design contexts. Different kinds of pattern-based tools exist, research prototypes (Pattern Tool (Meijers, 1996), FACE (Meijers, 1997), etc) as well as commercial tools (Rational XDE, TogetherJ, etc). Nevertheless, the majority of existing tools only integrate E. Gamma's patterns system and don't take into account the patterns engineers needs.

This paper presents a reaserch project: AGAP, a development environment based on pattern reuse which combines the needs of applications engineers who specify information systems by patterns applications, and the needs of patterns engineers, who specify patterns and patterns systems by using the same formalism or by reusing items of existing formalisms. The second section presents, functional specification of AGAP, its architecture and resumes its features and its functionalities by some screens examples. To conclude, section 4 presents two AGAP industrial experiences and proposes future prospects of our research.

2 AGAP : A PATTERN-BASED TOOL

The main goal of AGAP is to meet needs of the pattern and application engineer.

Front-Conte A., Hassine I., Rieu D. and Tastet L. (2004).
AN INFORMATION SYSTEM DEVELOPMENT TOOL BASED ON PATTERN REUSE.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 548-551
DOI: 10.5220/0002643605480551

2.1 Actors

- The **patterns engineer**'s goal (figures 1) is to create and define pattern. Moreover, he aims to define patterns formalisms and patterns systems (PS) described according to these formalisms. A pattern system is applied to a given applicative domain (ex. banking IS) according to a given technological domain (ex. object-oriented).

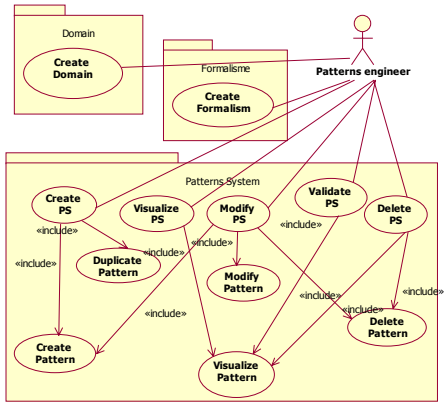

Figure 1: A few patterns engineer use cases

- The main goal of the **applications engineer** is to imitate pattern in order to model and design information systems (IS). This imitation is based on one or more PS available in AGAP. (figure 2).

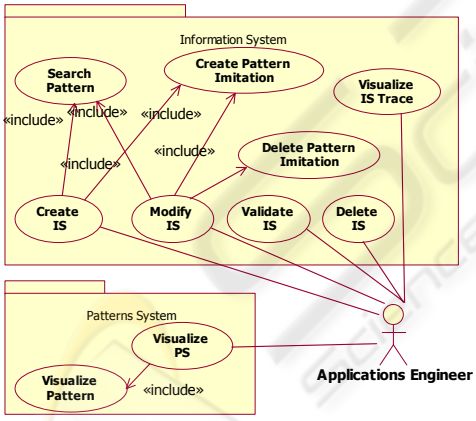

Figure 2: Applications engineer use cases

- The **administrator** manage AGAP by assuring : connection to the data base, user management and finally deletion of validated components, for example, a validated pattern system can be deleted only by the administrator who will make sure that no more user wants to use it.

2.2 Functional Specification

AGAP is composed of 7 business components: Tool, User, Field-type, Domain, Formalism, Patterns system and Information System. Only Tool and Patterns system components will be described in this paper. The structure of each component conforms to the business components structuring of the Symphony process (Hassine, 2002).

2.2.1 Component « Tool »

The component « Tool » is the base component of AGAP. It plays the role of a mediator between the user and the other business components. The interface includes the features expected by the user. These features correspond to all the use cases quoted in the section 0.

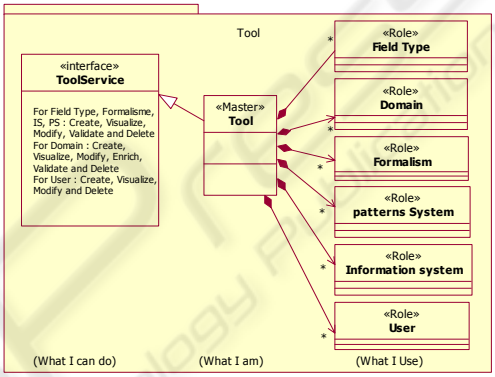

Figure 3: Component « Tool »

2.2.2 Component « Patterns system »

A pattern system is composed of patterns. described in a given formalism.

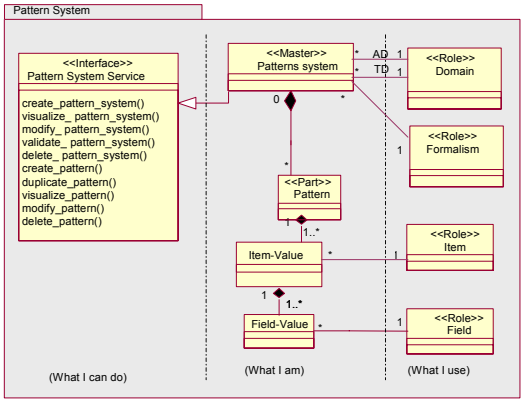

Figure 4: Component "Pattern system"

Each pattern has a given number of items whose fields are defined in the associated formalism. The value of every field has to respect the definition given in the field type associated to the field.

Application : The Figure 5 shows the creation screen of «GAMMA pattern system».

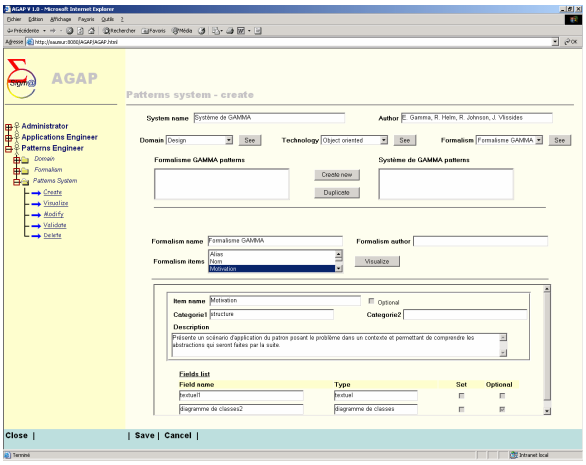

Figure 5 : Creation screen of «GAMMA PS»

2.3 Architecture

The current version of AGAP is an evolution of some last versions (Conte, 2001). This evolution takes into account applications engineer aims which is the creation of its information system. To realize this task, the application engineer needs to look for a solution described by a pattern specified in a given patterns system. Then, he realizes the imitation and the integration of this pattern in its information system. He must also be able to visualize the integration traceability.

2.3.1 Cooperation with other modeling tools

AGAP has to allow to manage various diagrams types : class diagrams, sequence diagrams, etc. He must also be able to modify and to manipulate their contents. In this purpose AGAP is coupled with other evolved existing tools specialized to realize these tasks (Rose, Objecteering, etc.).

A first cooperation mode proposes the following architecture (figure 7):

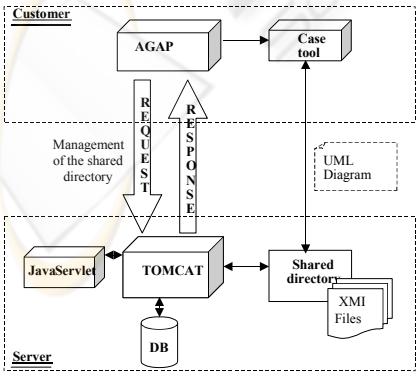

Figure 6: Cooperation AGAP/UML case tool

In the server part are defined a shared directory accessible by AGAP users, a common data base in which the link between the diagram file and the pattern will be protected (its name, its protection place) and Java servlets which ensure the management of the diagrams files within the shared directory: creation of pattern workspace, naming of the diagrams files, validation of the diagrams files, etc. Diagrams are saved in XMI files. XMI (XML Metadata Interchange) allows to exchange logical UML models between various distributors of UML case tools.

2.3.2 Pattern exchange format

Each pattern belongs to a patterns system, which itself represented in a particular formalism. One or several items of the formalism contain the solution of the pattern. To allow a better imitation and integration, a field is added to the item solution which will contain the pattern constraints. This solutions are created in case tools as ArgoUML, Rational Rose, etc. Each case tool allows to save UML diagrams in XMI. However, the result file is not completely compatible between the various tools. Indeed, it is based on a normalized DTD (UML 1.3 and UML 1.4), but each case tool adds extensions to allow for example the diagram drawing. To be case tool independent, AGAP generates an XMI file which contains only the necessary data for the UML diagram. To couple a case tool to AGAP, it is necessary to associate to it all the features allowing to generate this generic XMI file, as well as the invert features.

2.3.3 Imitation and Integration

An imitation is an adaptation of the solution preserving the essence of the pattern problem. Solution imitation is made from the generic XMI file of the solution. The integration is the result of the fusion of two XMI files, the pattern XMI file and the information system XMI file.

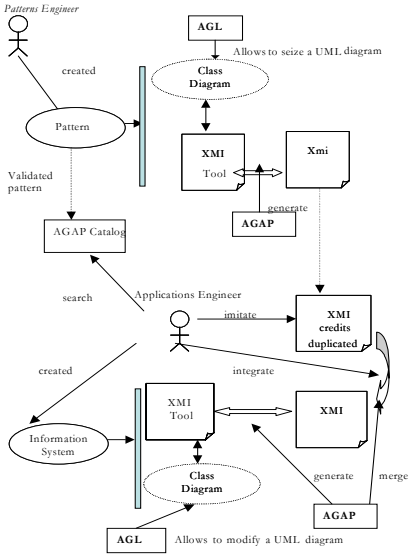

Figure 7: AGAP imitation and integration processes

Application : The shows the imitation screen of «Adapter pattern ».

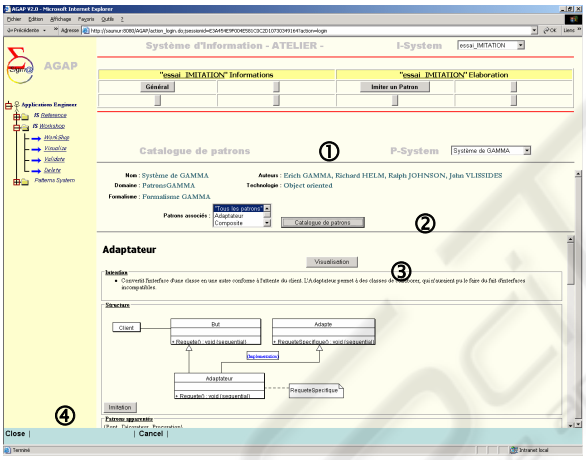

Figure 8 : Imitation screen of «Adapter Pattern»

3 CONCLUSION

This article presented AGAP, a development environment suited to two types of actors, applications engineers and patterns engineers. AGAP addresses therefore two types of processes:
- a process by reuse allowing the AE to define information systems by selecting, applying and integrating patterns applications,
- a process for reuse allowing the PE to define and to organize patterns systems.

AGAP evolved from a prototype model to a functional product and was used wright now to specify two formalisms: P-Sigma (Conte, 2001) and Gamma as well as Gamma patterns system (Gamma, 1995) and two patterns systems written in P-Sigma.

These patterns systems result from applied researches on two projects in collaboration with industrial companies. The first one focuses on the engineering of Product Information Systems (PIS) of industrial enterprises (Gzara, 2000) and was developed in collaboration with Schneider Electric company (project CNRS PROSPER-POSEIDON). The second patterns system concerns the specification of Symphony, a development process based on business components proposed by the UMANIS company.

From these first validated results, other research works were initiated to facilitate reuse in information systems engineering field and to guaranty a traceability between design choices and software products resulting from the design.

REFERENCES

Ambler, S.W., 1998. *Process Patterns building Large Scale Systems using Object technology*, SIGS Books, Cambridge University Press.

Borne, I., Revault, N., 1999. Comparaison d'outils de mise en oeuvre de design patterns, *Object-oriented Patterns*, Vol5, num2.

Conte. A, Giraudin J .P., Hassine I., Rieu D. 2001, Un environnement et un formalisme pour la définition, la gestion et l'application de patrons, Revue ISI vol 6 n°2.

Fowler, M., 1997. *Analysis Patterns – Reusable Object Models*, Addison-Wesley.

Gamma, E., Helm, R., Johnson, R.E., Vlissides, J., 1995. *Design patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley,

Gzara L., Rieu D. 2000. Tollenaere M., *Pattern Approach To Product Information Systems Engineering,* Requirements Engineering Journal, Editors: Peri Loucopoulos & Colin Potts, Springer- Verlag, London.

Hassine I., Rieu D., Bounaas F., Seghrnouchni O. , « Symphony : Un modèle conceptuel de composants métier» *Revue ISI*, volume 7, numéro 4, Hermès, 2002.

Johnson, R.E., 1992. Documenting Frameworks using Patterns, *OOPSLA'92*.

Maiden, N., Sutcliffe, A., Taylor, C., Till, D., 1994. A set of formal problem abstractions for reuse during requirements engineering, *ISI, Hermes, vol. 2, n° 6.*

Meijers M., 1996. Tools Support for Object-Oriented Design Patterns, *Master's Thesis, Utrecht University*.

Meijler, S. Demeyer, R. Engel, 1997. Making design patterns explicit in Face, *ESEC/FSE 97.*

Rieu, D., Giraudin, J.P., Conte A., 2002. Pattern-Based Environments for Information Systems Development, *The Sciences of Design, Lyon, France.*