AMF : a new design pattern for complex interactive software ?

F. Tarpin-Bernard, B.T. David

Lab. GRACIMP, ICTT Research group, Ecole Centrale de Lyon,
36 av. Guy de Collongue, 69131 Ecully, France
Email: tarpin@ec-lyon.fr, david@ec-lyon.fr

INTRODUCTION

An appropriate architectural model should fulfill three main objectives. Firstly, it organizes software structure to improve implementation, portability and maintenance. Secondly, it helps identify the functional components (e.g. MVC [1]), which is essential during analysis and design process. Its third role is to help the understanding of a complex system, not only for designers, but also for end-users. The architectural model is one of the three key elements needed to achieve efficient and good developments: methods - models - tools.
This paper presents AMF (French acronym for MultiFaceted Agent) an architectural model for interactive software which fulfills all these objectives. AMF is a generic and flexible model that can be used with design and implementation tools. It includes a graphical formalism that expresses the structures of software, and a run-time model that allows dynamic control of interactions.
The current trend in software engineering is to build handbooks of design patterns. Patterns help developers share architectural knowledge, help people learn a new design paradigm or architectural style, and help new developers avoid traps and pitfalls traditionally learned only by costly experience. Therefore, a new technique can be documented as a pattern, but its value is know only after it has been tried. The longer a pattern has been used successfully, the more valuable it tends to be [2].
AMF is a relatively new model that has been implemented in C++ and tested only on few applications. However, we consider that it contains all the requirements needed to become a real design pattern. That is the reason why, we want to present it following *the guidelines* proposed by Gamma* [3], as if it was already an improved design pattern.

1. AMF: A DESIGN PATTERN

Intent: The AMF approach structures interactive software in a set of multifaceted agents. Each agent is composed of facets that communicate through control administrators. AMF authorizes dynamic evolution of all its components.

---

* In this paper, we often refer to Gamma's works. In particular, we briefly compare parts of AMF to Gamma's design patterns. As we cannot develop here these comparisons, the reader should refer to [3] to better understand the interest of these related patterns.

Motivation

Multiagent models organize an interactive system in sets of agents which collaborate to realize the man - machine dialogue. Most of them use agents composed of three components mapped on the HCI paradigm (presentation to the user, functional core, and control of the interaction). These models present two main lacks:

1. They use macroscopic facets combining different thematic functions ;
2. They provide very few mechanisms to express interaction control and dynamics ;

Applicability

Use the AMF pattern when you build complex interactive software, heterogeneous software (using several programming languages), or dynamic software (in terms of interaction rules).

Structure

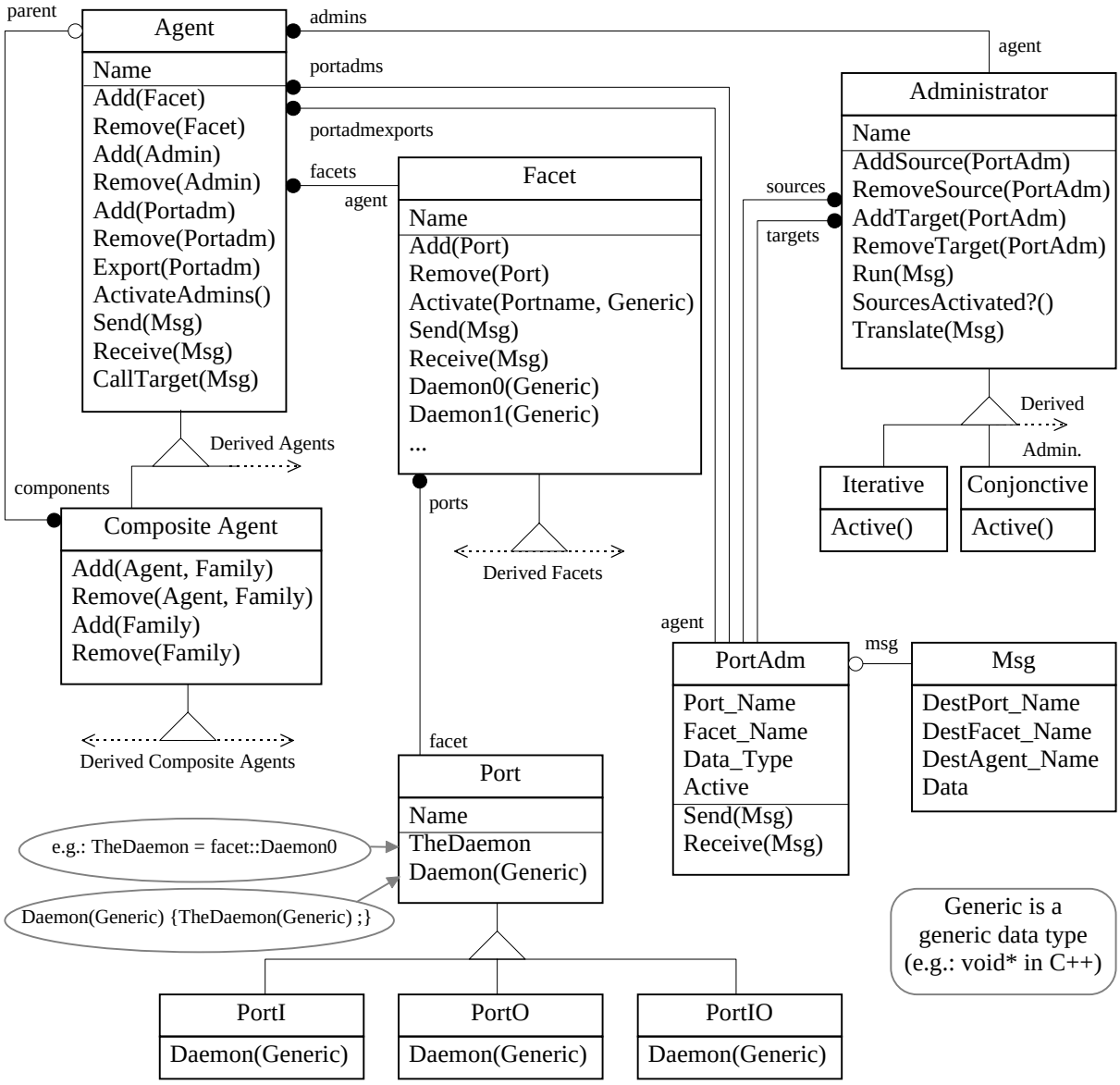

Figure 1: an OMT-based representation of AMF structure

Participants

The agents are organized according composition rules so that an agent can contain several other agents (application of the COMPOSITE pattern of Gamma [3]). A root agent contains all the agents of the system. Each *agent* is composed of a various number of *facets*. These facets can be *similar to the classical components* of PAC model [4] or come either from a *finest split of control component*, or from the *identification of new characteristics of agents* (e.g.: management of the user model), or from *duplication of classical facets* (several "presentation" facets). For instance, we can identify the following facets: "presentation" (I/O relations with the user), "abstraction" (logical data - functional core), "evaluation" (capture of the user's actions),"help" (contextual and on-line helps linked to a user model)...

Furthermore, AMF expresses interaction control with two kinds of components:

1. Each facet presents several *communication ports* (allowing inputs, outputs or both). These ports avoid to have a permanent binding between a service and its implementation. Moreover, it is possible to realize the body of the functions in heterogeneous languages. This technique looks like the BRIDGE pattern of Gamma [3].
2. The control is composed of *port administrators* (elements that store references on the communication ports of the facets) and *control administrators* (components that manage unidirectional links between several port administrators).

A control administrator has three roles:

1. To *connect:* managing logical relations between the communication ports (sources S and targets T) that are connected to it;
2. To *translate:* transforming the messages that come from the source ports in messages understandable by target ports. According to the translation technique, we distinguish *transferring, corresponding, assembling, computing* and *processing administrators*. This technique looks like the ADAPTER pattern of Gamma [3];
3. To express *behavior* (see MEDIATOR and STRATEGY patterns [3]): specifying activation conditions and doing extra work (e.g. communicate with other entities in a cooperative context). According to the different conditions, we identified administrators called: *simple* (if S is activated then activate T), *conjunctive* (if S_1 and S_2 then T), *disjunctive* (if S_1 or S_2 then T), *sequence* (if S_1, S_2 then T), *iterative* (if n*S then T)...

Collaborations

When a facet needs to use a distant service, it activates its corresponding output port. This port builds a message and sends it to its associated daemon (see figure 1). Then, the control facet of the owner agent wakes up all the control administrators which are connected to this source port. If this port is exported (connected to other agents), the activation is recursively transmitted to the parent agents. Then, each concerned administrator considers its activation conditions (see behavioral role). If these conditions are validated, the message is translated and sent to all the target ports. The activation of these ports runs their associated daemons.

Actually, the control administrators are not directly linked to the communication ports. They are linked to administrator ports which are kinds of string references of communication ports. These administrator ports are used to route messages and to introduce a large flexibility : at each moment, you can exchange two facets if theirs ports have the same names.

Other sections

We will not develop the other technical sections introduced by Gamma to describe design patterns, i.e. *consequences* (largely evoked before), *implementation* (details of C++ classes) and *sample code, known uses* (examples developed in our laboratory) and *related patterns*.

2. A GRAPHICAL FORMALISM

AMF proposes a very powerful graphical formalism which helps understand complex systems. This formalism is used as a design tool by editors and builders. It represents agents and facets with overlapped boxes, communication ports with rectangles which contain the associated services, and control administrators with symbols which express their behavior.
To briefly introduce this formalism, figure 2 shows the modeling of an elementary interaction. Generally, two administrators manage the relations between an action starting from the *Presentation* facet and the associated command defined in the *Abstraction* facet. In another situation, A_1 could be replaced by an iterative administrator if several activations are necessary to run the action.

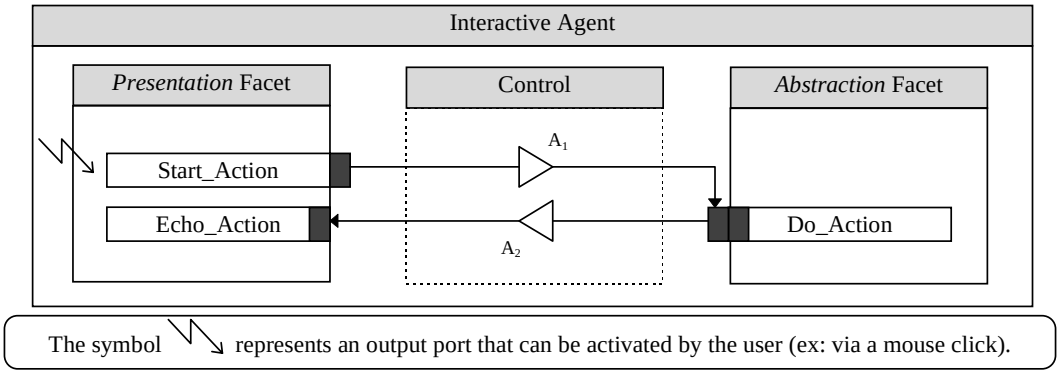

Figure 2: An interaction on a single-user agent modeled with AMF.

CONCLUSION

We presented a new multifaceted multiagent architectural model for complex interactive software. This model, AMF, is not only a conceptual model with a very useful graphical formalism, but also a guide of implementation which provides mechanisms of interaction and builder tools (C++ [6]). Currently used to model various applications like schedulers or CAD editors, AMF is a very interesting model for the evaluation of interactions. The definition of new facets lets us imagine new uses. For instance, we extend AMF for groupware design. For these reasons, we consider that AMF could become a design pattern as defined by Gamma.

REFERENCES

[1] G.E. Krasner & S.T. Pope, A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *Journal of Object-Oriented Programming,* 1(3) 26-49, Aug. 1988.
[2] D.C. Schmidt, M. Fayad & R.E. Johnson, Software patterns, *Communication of the ACM*, 39 (10) 37-39, Oct. 1996.
[3] E. Gamma, R. Helm, R. Johnson & J. Vlissides, Dessign Patterns: Elements of Reusable Object-Oriented Software, *Addison-Wesley*, Reading, Mass. 1995.
[4] J. Coutaz, Architecture Models for interactive software: Failures and Trends, *in Engineering for HCI, G. Cockton Ed., Elsevier Science Publ.*, 137-153, 1990.
[6] F. Tarpin, Architectures logicielles pour le travail coopératif, PhD ECL, France, 1997.