Available online at www.sciencedirect.com
SCIENCE DIRECT®
The Journal of Systems and Software 74 (2005) 55–64
The Journal of Systems and Software
www.elsevier.com/locate/jss

"Computer, please, tell me what I have to do...": an approach to agent-aided application composition

Marcelo R. Campo *, J. Andrés Díaz Pace, Federico U. Trilnik

ISISTAN Research Institute, Faculty of Sciences, UNICEN University, Campus Universitario, Paraje Arroyo Seco, (B7001BBO) Tandil, Buenos Aires, Argentina

CONICET, Avda. Rivadavia 1917, (C1033AAJ) City of Buenos Aires, Argentina

Received 16 October 2002; received in revised form 1 February 2003; accepted 2 May 2003
Available online 21 January 2004

Abstract

The process of starting to use any reuse technology is usually one of the most frustrating factors for novice users. For this reason, tools able to reduce the learning curve are valuable to augment the potential of the technology to rapidly build new applications. In this work, we present *Hint*, an environment for assisting the instantiation of Java applications based on software agents technology. *Hint* is built around a software agent that has the knowledge about how to use a reusable asset and, using this knowledge, is able to propose a sequence of programming activities that should be carried out in order to implement a new application satisfying the functionality the user wants to implement. The most relevant contribution of this work is the use of planning techniques to guide the execution of instantiation activities for a given technology.

1. Introduction

It is a well-known fact that the more powerful the reuse technology the more knowledge is necessary to rapidly start to use it to produce applications. This aspect represents one of the most limiting factors of any reuse technology, but it is particularly crucial to object-based ones (Bosch, 2000). For this reason, composition tools are an invaluable complement. These tools can vary from simple wizards for generating code skeletons to complex graphical tools supporting the visual composition of applications. This kind of tools can dramatically improve the productivity in the case of applications that naturally fit into the scope of the target technology (Campo et al., 2002). However, when complexity grows, despite components are derived from a domain-specific framework, an integration framework or they are implemented following some interface standard, some kind of coding is always necessary in order to get a running application.

At this point a deeper knowledge of underlying design details can be necessary in order to use, or even more, adapt the behavior of existing components. Certainly, good quality documentation is a key issue. Nevertheless, none of the developed documentation techniques (Demeyer et al., 2000; Johnson, 1992) can be completely adapted to the different types of users, especially if considering the variations on knowledge and experience these users usually have (Helm et al., 1990). On one side, expert users may prefer to know about design details, and be able to make their own decisions. Many times this kind of users can adapt a framework (Fayad et al., 2000) in unexpected ways. On the other side, and perhaps the most important one, novice users may just want to be aware of higher-level aspects. This kind of users should be able to build an application without the need of understanding overwhelming details of design rationale. However, this is not always the case, producing a negative impact on the benefits that the technology can bring to enhance software development.

We believe that the problem resides not in how to provide a specific tool for a given technology, but in how

* Corresponding author. Tel.: +54-2293440363; fax: +54-2293440362.
E-mail addresses: mcampo@exa.unicen.edu.ar (M.R. Campo), adiaz@exa.unicen.edu.ar (J.A. Díaz Pace), ftrilnik@exa.unicen.edu.ar (F.U. Trilnik).

0164-1212/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2003.05.003

to make the documentation an active source able to guide a user in what should be done for building a specific application. On the basis of this documentation, a tool should take the principal requirements and design decisions from the user, and then propose a number of actions to follow in order to get a running application. This process can be approximated by means of what we have called the "Computer, please, tell me what I have to do..." paradigm. The rest of the paper presents an in depth description of this agent-based approach, organized as follows. Section 2 introduces the main concepts of the mentioned paradigm. Section 3 outlines the architecture of the *Hint* environment, illustrated with an example of framework instantiation. Section 4 covers the *Smartbooks* documentation method through which the instantiation knowledge is described. Section 5 shortly explains the use of planning capabilities to elaborate instantiation plans. Section 6 discusses preliminary results and lessons learned. Finally, Section 7 rounds up the conclusions of the work.

2. The "Computer, please, tell me what I have to do..." paradigm

Certainly, a pursued *ideal* is to have a system that (like in *Star Trek*) we could ask the computer to solve any problem by simply asking: "Computer, please, solve this problem...", and automatically obtain the result without any further effort. For example, let's suppose we want to build a graphical editor for Pert-like charts and there is a framework for building graphical editors available. In the ideal situation we would ask the computer to build a system that should satisfy the following informal specification:

> *"It should be possible to interactively create graphical objects representing events, and to relate these events through precedence links. The events should have visual representation for its attributes, and two of the attributes will be edited through the graphical interface. Besides, each attribute could be related with other ones, both from the same or related events".*

The computer would interpret these requirements and using the existing framework and components would produce the required application. Unfortunately, computer science is currently rather far of providing such a system, particularly in domains such as software development.

Despite this reality, we can approximate this ideal by using a paradigm in which the computer tells us what steps we should carry out in order to get an implementation using a given reuse technology. That is, given a set of functional requirements for an application, we should be able to ask the computer, "Computer, please, tell me what I have to do to implement this functionality using the existing software that you know how to use". In this approach, instead of providing the computer with the previous requirements, the computer shows the user the different functionality that could be derived from an existing reusable asset and the user can select the aspects he/she judges related with the required functionality. Next, using the knowledge about the existing assets, the computer answers the list of programming activities that should be carried out in order to implement such application. These activities can vary from the advice of which classes should be specialized, what methods should be overridden, what component should be used, what proxies should be implemented, what parameters should be set with specific values, etc. Note that an important aspect of the approach is the interaction with the user, so that he/she can provide the tool with the main design decisions to guide the generation of programming activities.

Following this idea, we developed *Hint*, *a Java* tool based on agent technology (Bradshaw, 1997) designed to provide semi-automated support to the process of application composition. *Hint* is based on the *Smartbooks* documentation method (Ortigosa et al., 2000), which extends common documentation techniques with *instantiation schemes* specifying how a piece of software should be specialized or used to implement a given functionality. Using *least-commitment planning techniques* (Weld, 1994), the *Hint* agent is able to build an implementation plan from the set of functionality that the user selected from the possible functionality that the documented reusable asset can provide. This plan consists of the sequence of programming tasks to be accomplished in order to produce a final application. When the developer starts executing these tasks, the agent observes the process and proceeds to modify the instantiation plan whenever new information, not previously available, can be deduced from the developer's behavior. These features distinguish *Hint* from other approaches in that developers, particularly novice ones, can start the development with a guide of what steps they have to carry out in order to build their applications.

3. The *Hint* environment

Hint represents the result of a three-year research effort on the subject (Ortigosa et al., 2000), incorporating in its last release enhanced functionality to deal with Java frameworks and components, CORBA adaptation and aspect-oriented development (Kiczales et al., 1997). In the current version it comprises four main components: *Documentation Tool*, *Rule Generator*, *Functionality Collector*, and the *Hint* agent, which is in turn composed by three components: *Planner*, *Task Manager*