ISSN 0103-9741

Monografias em Ciência da Computação

n° 14/10

On the Development of Personalized User Agents: A Model-driven Approach

Ingrid Oliveira de Nunes

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 14/10 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena October, 2010

On the Development of Personalized User Agents: A Model-driven Approach [1,2]

Ingrid Oliveira de Nunes

ionunes@inf.puc-rio.br

Abstract. Many modern computer systems are providing assistance to several of our usual tasks, by incorporating features with a proactive and autonomous behavior. A generalized and ambitious idea underlying such systems is the personalized user agents, which are personal assistants acting on the users' behalf. Even though significant research effort has been invested on developing user agents, we are far from their massive adoption. Research work in the context of human-computer interaction have criticized agent-based methodologies that seem to produce systems not easily accepted by the user: one of the main reasons is the autonomy of the agents that can cause a loss of control by the user. Different users need different kinds of user agents. In addition, a large group of users is willing to adopt user agents only if they know exactly what the agent is going to do.

Our research addresses this group of users. We aim at tackling two main problems: (i) how to empower end-users to instruct their personal agents; and (ii) how to build this family of applications, considering software architecture quality attributes. Our solution to these issues is to investigate a *virtual separation of concerns*. The main idea is to propose a virtual user model that is a high-level global view of user customizations, which is implemented by an underlying service-oriented multi-agent infrastructure. This user model might be able to drive runtime adaptations in the user agents, based on a model-driven approach. Our approach most likely will comprise: (i) a reference architecture that provides a general structure for user-customizable applications; (ii) a user Domain-specific Model, and a corresponding end-user Domain-specific Language, to model user configurations and preferences; (iii) a model-driven mechanism that supports the runtime adaptation of user agents; and (iv) a framework based on the components of our approach, which provides an infrastructure to build applications of our target domain, i.e. personal assistance software systems.

Keywords: Personalized User Agents, Multi-agent Systems, Model-driven Development, Domain-specific Modeling, Personalization, Dinamic Adaptation.

[1]This work has been partially supported by CNPq 557.128/2009-9 and FAPERJ E-26/170028/2008.

[2]This document was presented on April 20, 2010 as the PhD proposal of Ingrid Nunes, approved by the following committee: Prof. Carlos Lucena (president), Prof. Simone Barbosa and Prof. Hugo Fuks.

Resumo. Muitos sistemas de computação atuais estão provendo assitência a várias das nossas tarefas cotidianas, através da incorporação de funcionalidades com um comportamento pró-ativo e autônomo. Uma idéia generalizada e ambiciosa por trás desses sistemas são os agentes de usuário personalizados, os quais são assistentes pessoais atuando em nome do usuário. Mesmo que um esforço significativo de pesquisa tem sido investido no desenvolvimento de agentes de usuário, nós estamos longe da sua adoção massiva. Pesquisa no contexto da interação humano-computador tem criticado metodologias baseadas em agentes que parecem produzir sistemas que não são facilmente aceitos pelo usuário: um dos principais motivos é a autonomia dos agentes que causa perda de controle pelo usuário. Diferentes usuários precisam diferentes tipos de agentes. Além disso, um grande grupo de usuários desejam adotar agentes de usuário se eles souberem exatamente o que o agente irá fazer.

Nossa pesquisa tem como alvo este grupo de usuários. Nós pretendemos atacar dois problemas principais: (i) como dar poder a usuários finais para intruírem seus agentes pessoais; e (ii) como contruir esta família de aplicações, considerando atributos de qualidade de arquiteturas de software. Nossa solução para essas questões é investigar uma *separação virtual de interesses*. A idéia central é propor um modelo virtual de usuário que é uma visão global de alto nível das customizações do usuário, que é implementada por uma infraestrutura subjacente multi-agente e orientada a serviços. Este modelo de usuário deve ser capaz de guiar adaptações em tempo de execução em agentes de usuário, baseado numa abordagem dirigida a modelos. A abordagem irá conter: (i) uma arquitetura de referência que provê uma estrutura geral para aplicações customizáveis por usuários; um modelo específico de domínio de usuário, uma uma linguagem específica de domínio para usuários correspondente, para modelar configurações e preferências de usuário; (iii) um mecanismo dirigido a modelos que suporta adaptações dinâmicas de agentes de usuário; e (iv) um *framework* baseado em componentes da nossa abordagem, o qual provê uma infraestrutura para construir aplicações do nosso domínio-alvo, i.e. sistemas de software de assistência pessoal.

Palavras-chave: Agentes de Usuário Personalizados, Sistemas Multi-agentes, Desenvolvimento Dirigido a Modelos, Modelagem Específica de Domínio, Personalização, Adaptação Dinâmica.

In charge of Publications:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Contents

1 Introduction

Recently, Rogoff claimed that the Artificial Intelligence (AI) area is going to be the next big driver of global growth (Rogoff 2010). He argues that computers are going to automatically perform a growing number of tasks in the next 50 years, ranging from driving taxis to performing routine surgery. Many modern computer systems are already providing assistance to several of our usual tasks, by the incorporation of features with a proactive and autonomous behavior. Typical examples include product recommendations based on our purchase history and automatic generation of playlists based on songs we listen. These systems are increasingly becoming part of our everyday life. Furthermore, as web applications become increasingly interactive, accessible, and pervasive there is a need for mechanisms to help users extend their mental and physical capabilities. The web now provides access to huge amounts of well-organized information and supports social interactions well beyond our physical limitations. Thus there are new challenges in managing both the quantity of information and the complexity and timeliness of relationships.

Multi-agent Systems (MASs) (Weiss 1999), with roots not only in AI but also in distributed systems and Software Engineering (SE), have addressed this domain of applications, including web-based supply-network management, auction staging, medical-record processing, mission scheduling, and e-commerce (Jennings & Wooldridge 1998). MASs can incorporate autonomous behavior to support users in meeting many of these new barriers by freeing users from repetitive and tedious tasks. In addition, MASs by providing autonomous and proactive behavior, may be employed by web users to support access to information and decision-making.

A generalized and ambitious idea of systems with autonomous and proactive features are the personalized user agents. These are users' personal assistants that may act on their behalf in the virtual world. The concept of personal user agents was championed by Maes in 1994. In (Maes 1994), she discusses the large number of tasks that emerge from the use of computers and the web, and that autonomous agents may be personal assistants who are collaborating with the user in the same work environment. She suggests the existence of interface agents to provide assistance by monitoring the user's actions in the interface, i.e. "watching over the shoulder of its user", learning new "shortcuts", and suggesting better ways of doing the task. Moreover, user agents are not only responsible for reducing the effort of dealing with interfaces, but also may be in charge of the (semi-)automation of several other tasks, including planning trips for users and representing them in online auctions. Given that agents represent individuals in these scenarios, there remains a need to personalize an agent to meet specific needs of the users (Nunes, Lucena, Cowan & Alencar 2009).

This section introduces the specific problem we are looking at in the context of personalized user agents and the broad approach taken to solve it. In Section 1.1 we describe our problem statement and limitations of existing work. Next, we give an overview of our proposed solution in Section 1.2 and also define our specific aims. In Section 1.3, we detail the expected contributions for this thesis. In Section 1.4, we highlight aspects related to our work that will be left out of its scope. Finally, in Section 1.5, we present the structure of the remainder of this thesis proposal.[3]

[3]This document was presented on April 20, 2010 as the PhD proposal of Ingrid Nunes, approved by the following committee: Prof. Carlos Lucena (president), Prof. Simone Barbosa and Prof. Hugo Fuks.

1.1 Problem Statement and Limitations of Existing Work

Our research aims at tackling two main problems: (i) how to empower end-users to instruct their personal agents, i.e. computer systems that provide personal assistance to users and are able to act on their behalf; and (ii) how to build this family of applications (personalized user agents), considering software architecture quality attributes, i.e. our goal is to address high-quality systems from a SE perspective. Based on these two issues, we define our research question as follows.

Research Question. *How to develop high-quality personal assistance software that can be dynamically adapted to configurations and preferences expressed by a user?*

Our approach distinguishes user *configurations* from *preferences*, which we collectively refer to as *customizations*. A configuration is a setting that a user performs in a system, such as adding or removing a service or enabling an optional feature. These configurations can be related with environment restrictions, e.g. a device configuration. Preferences, in turn, refers to the set of assumptions relating to a real or imagined "choice" between alternatives, based on the degree of happiness, satisfaction, gratification, enjoyment, or utility they provide. Such information can be seen as a cognitive model of the user and is typically used in agents' reasoning process.

In this section, we detail these two problems we are addressing in our work and limitations of existing work. In Section 1.1.1, we detail research work that has been done in order to build user models, which are typically used to drive user agents' behavior. In Section 1.1.2, we describe current SE practices to develop user agents that aggregate user preferences.

1.1.1 User Modeling Approaches

Since their introduction in the middle 90s, several research work has been carried out in the context of user agents. These works are mainly concentrated in capturing and reasoning about user preferences. They include learning preferences by monitoring users (Schwab, Kobsa & Koychev 2000), eliciting their preferences by interacting with them (Luo, Jennings & Shadbolt 2006) and reasoning about preferences (Boutilier, Brafman, Domshlak, Hoos & Poole 2004). One of the largest projects related to the development of personal assistants is the Cognitive Assistant that Learns and Organizes (CALO) project[4], whose goal is to support a busy knowledge worker in dealing with the twin problems of information and task overload (Yorke-Smith, Saadati, Myers & Morley 2009). Even though significant research effort has been placed on developing user agents, we are far from their massive adoption.

Schiaffino & Amandi (Schiaffino & Amandi 2004) presented an empirical study that provides a solid basis for explaining this scenario. They claim that the "human-computer interaction (HCI) people have criticized agent-based methodologies that seem to produce systems not easily accepted by the user: one of the main reasons is the autonomy of the agents that can cause a loss of control by the user." Their study concluded that different users need different kinds of user agents. In addition, they showed that a representative

[4]http://caloproject.sri.com/

amount of users are willing to adopt user agents just if they know exactly what the agent is going to do.

Our research addresses this group of users. Our goal is to address the problem of empowering users to control their user agents, as opposed to approaches that implicitly build user models based on observations. These approaches usually rely on inference models that might reach the wrong conclusions about user preferences and cause agents to take inappropriate actions. This can lead to undesired situations such as the "agent hell" scenario described in (Zambonelli & Luck 2005). On the other hand, if the process of explicitly instructing an agent becomes a hard task, users would not make this effort. Therefore, this process must be as close as possible to natural language specifications. Existing explicit user modeling approaches force users to express their preferences in a particular way. Most of them rely on partial or total orders, which is not enough for users to express their preferences. Consequently techniques must be adopted to elicit user preferences. However, two arguments have been used for justifying why existing approaches take this direction:

1. *The expressiveness of user preferences are restricted due to computational limitations, therefore there must be a tradeoff between representation and reasoning.* Computational algorithms to reason about user preferences require specific inputs that constrain the way users express their preferences. A simple example is boolean preferences, in which users have to select items they like. In order to model user preferences to be processed by an algorithm there must exist a translation between how users express their preferences and how they are going to be represented in a computer system. In the previous example, this is in charge of the user. In more complex models, it is unacceptable that users make that effort, and consequently, there is a need for elicitation techniques to interpret answers to questions and build the user model. In this sense, we will try to show that it is possible to let users to express their preferences in a language as closer as possible to natural language, and then translate this language to a model that can be used by reasoning algorithms.

2. *People usually cannot state preferences up front but construct their preferences as they see the available options (Pu & Chen 2008).* The domain of applications that is explored in (Pu & Chen 2008) is recommender systems. In such systems, users must anticipate their preferences in order to receive a recommendation of a certain product in the future (that might not be of their interest). In addition, users typically have their preferences stored in several locations, such as different online stores. As a consequence there is a lack of motivation of the user to provide their preferences, and for several locations. The domain of applications we are looking at is different in two aspects: (i) the idea is to have a user personal computer system, i.e. users provide their preferences only once and for their particular purpose; and (ii) we are aiming at the automation of users' repetitive tasks, i.e. users have already done a task over and over again, and therefore they have already been exposed to decision-making situations associated with that task. We aim at providing a way for users explicitly specify this task to be done. The cost of performing this specification is amortized by reducing the cost of performing the task several times in the future.

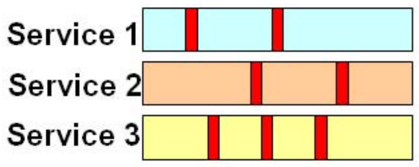
1.1.2 Software Engineering Practices to Develop User Agents

An essential characteristic of user agents is that they store information specific to each user. As previously discussed, most of existing research work focuses on eliciting and reasoning about this specific information (preferences), i.e. they focus on determining which are user preferences. However, how these preferences impact in computer systems is commonly not addressed in the literature. In some applications, such as recommendation systems, this impact tends to be low, because the user information is reflected only in a knowledge base. Nevertheless, this does not apply to systems that have their behavior changed due to user customizations.

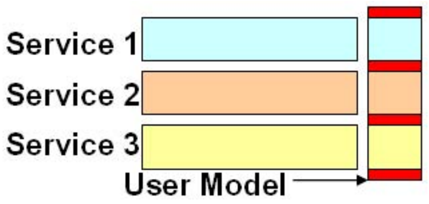
Current approaches to develop user agents typically adopt one of these two mechanisms for implementing user customizations: (i) a user model, which stores user information in a single location and is checked whenever a user-dependent action is performed; and (ii) control variables, which are inserted in the code to reflect user customizations and are used to make some decisions that indicate to an agent the right course of actions it should take. Both solutions are essentially the same, with the difference that the first solution concentrates all the user-specific data. Even though these solutions produce the desired behavior, they have drawbacks from a SE perspective.

Concentrating all user customizations in a single component creates a high coupling between this component and other system components. In addition, changes in this unique component may imply a lot of little changes applied to a lot of different classes. This characterizes the Shotgun Surgery bad code smell (Fowler, Beck, Brant, Opdyke & Roberts 1999). Moreover, in both solutions, a control variable will be used – in (i), it is retrieved from the user model – which is a program variable used to regulate the flow of control of the program. These control variables, i.e. user customizations, may be used in several system locations and are usually used in chained `if` or `switch` statements scattered throughout the system. If a new clause is added to the switch, all statements must be changed. This is another bad code smell, the Switch Statement (Fowler et al. 1999), and the object-oriented notion of polymorphism gives you an elegant way to deal with this problem.

Another software engineering issue related to user agents is that user customizations may be seen as a *concern* in a system that is spread all over the code. However, at the same time, each customization is associated with different services (also concerns) provided to users. Therefore, when developing such system one has to choose the dimension in which the software architecture will be modularized: in terms of services (Figure 1(a)) or modularizing user configurations and preferences in a single model (Figure 1(b)). It can be seen that it is not possible in either approach to modularize all concerns in separated



(a) Modularizing Services



(b) Modularizing User Model

Figure 1: Modularization Approaches.

Attitude	**Example**
Goal	I want to drink red wine.
Belief	I like red wine.
Motivation	Red wine is good for the heart.
Plan	In order to drink red wine either I go to the supermarket and buy a bottle (plan A) or I go to a restaurant and have some wine there (plan B).
Meta-goal	I want to drink red wine, but spending less money as possible (so I might choose plan A).

Table 1: User Preferences and their roles into agent architectures.

modules. In addition, without modularizing user customizations, as in Figure 1(a), they are buried inside the code, thus making it difficult to understand them as a whole.

Based on these arguments, we claim that there is a need for better software architectures to build personalized user agents, taking into account good software engineering practices. However, dealing with variable traits that emerge from user customization points is not a trivial task. These customization points are spread all over the system architecture and play different roles in agent architectures (Doyle 2004, Nunes, Barbosa & Lucena 2009). We illustrate examples of different roles that user preferences play into agent architectures in Table 1. If all this information is contained in a single user model, we have the problems discussed above and this model would aggregate information related to different concerns of the system (low cohesion among user model elements).

In summary, we are addressing two main problems not addressed by existing approaches:

(i) Current approaches mostly focus on the reasoning and elicitation processes of user preferences, relying on models that have a certain percentage of accuracy (not 100%). This prevents users to delegate (critical) tasks to their agents.

(ii) Current approaches focus on *which* are the user preferences, not addressing their impact in the system behavior and *how* their are realized.

1.2 Proposed Solution

Our proposed solution to the previously described issues is to provide a *virtual separation of concerns* (Kästner & Apel 2009), in which concerns-related code is not physically located into separated modules, but presented as a virtual view of the system. A concern is anything that is interesting from the point of view of a stakeholder. In our case, the concern that will be virtually modularized is the user model. The main idea is to structure the user-customizable system architecture in terms of service-oriented agents and physically modularizing each variability as much as possible into agent abstractions. In addition, we provide a high-level (i.e. virtual) user model, as Figure 2 illustrates. The model is virtual because it is not physically implemented into code, but provides a modularized view of user customizations. User preferences and configurations are not design or implementation abstractions, but they are implemented by typical agent abstractions (beliefs, goals, plans, etc.), i.e. they play their specific roles in the agent architecture. The virtual user model is a complementary view that provides a global view of user customizations. This model uses a high-level end-user language, and users are able to configure and adapt their agents

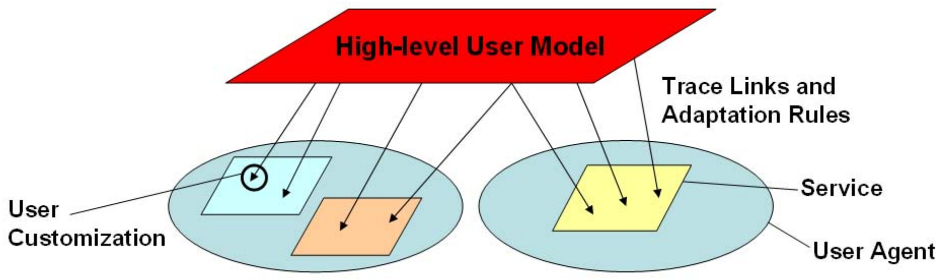


Figure 2: High-level User Model.

by means of this model. The adaptation process is achieved by means of trace links and adaptation rules that connects these two models – user model and agent architecture model, which are in different abstractions levels (end-user and implementation levels).

Based on this idea, we narrow our research question, by formulating our research hypothesis:

Research Hypothesis. *The adoption of an implementation-independent user model expressed in a high-level language for end-users, combined with adaption rules and trace links related to user customizations:*

(i) *provides an effective means for end-users to explicitly personalize their agents; and*

(ii) *improves the quality, mainly in terms of modularization, of user-customizable software architectures, in comparison with current practices to develop user model-based systems.*

In a nutshell, our approach consists of a model-driven reference architecture to building user-customizable applications as well as models and mechanisms to be used within this reference architecture. The domain of applications we are targeting is personal assistance software systems. They are composed of service-oriented user agents, which provide different personalized services to users. Agents are not only tailored to provide customized services to users, but also aggregate users' preferences model in order to act appropriately on their behalf. Our approach is to define a Domain-specific Model (DSM), with the aim of allowing the modeling of users' preferences and configurations using high-level abstractions, typically used in users' vocabulary, thus abstracting from the underlying implementation model. An instance of this DSM is a user model, which is related to an underlying multi-agent-based infrastructure by means of adaptation rules and trace links. In this sense, the user model can be seen as a *high-level view* of user variation points, including preferences and configurations, present in user agents. User agents are built using variability implementation techniques in order to support different user customizations. These trace links provide support to adapt user agents based on changes on the user model. Furthermore, our goal is also to build a domain-specific end-user-level language, which is based on this DSM, to allow end-users manipulating the user models. By means of this Domain-specific

Language (DSL), users will be able to dynamically program and personalize their agents (end-user programming).

We propose to adopt an agent-based approach to design and implement user-customizable systems for several reasons: (i) agent-based architectures are composed of human-inspired components, such as goals, beliefs and motivations, thus reducing the gap between the user model (problem space) and the solution space; (ii) plenty of agent-based AI techniques have been proposed to reason about user preferences, and they can be leveraged to build personalized user agents; and (iii) agent architectures are very flexible, thus facilitating the implementation of user customizations. For instance, there is an explicit separation of what to do (goals) from how to do it (plans).

In addition, by proposing our reference architecture, we also aim at providing guidance for the development of high-quality user-customizable applications. As previously discussed, most of existing approaches focused on determining which are user preferences and did not address how they are realized in software systems. Since the late 1980s, software architecture has emerged as the principled understanding of the large-scale structures of software systems (Shaw & Clements 2006). Therefore, while designing this architecture, we have taken into account software engineering issues identified as current practices to develop user-model-based applications. In this sense, we also contribute with the analysis of existing mechanisms to implement user customizations, which may result in low-quality architectures. Good (modular, stable, ...) architectures are essential to produce higher quality software which is easier to maintain. Otherwise, software architectures may degenerate over time, making their maintenance a hard task, including an increase of development costs due to refactorings.

Moreover, our approach leverages principles of Software Product Line (SPL) engineering. SPLs (Clements & Northrop 2001, Pohl, Böckle & van der Linden 2005) are becoming an essential software reuse technique that has been applied in the construction of mass-produced software systems. A SPL is a family of related software products built from a common set of software components, where each component typically implements a distinguishable feature (Clements & Northrop 2001). SPLs exploit common and variable features of a set of systems and lead to the development of flexible architectures that support the derivation of customized systems. Personalized user agents can be seen as a SPL of user agents that are tailored to specific users. As a consequence, SPL techniques are used to support the development of this family of agents. However, the main difference from personalized user agents and SPLs is that the later contemplates only user configurations, but not user preferences. In addition, there are other challenges in combination of agent-based approaches and SPLs (Pena, Hinchey & Ruiz-Cortés 2006), such as dealing with fine-grained variability into agent architectures (Nunes, Cirilo, Cowan & Lucena 2009).

In summary, our approach most likely will comprise: (i) a reference architecture that provides a general structure for user-customizable applications; (ii) a user DSM, and a corresponding end-user DSL, to model user customizations; (iii) a model-driven mechanism that supports runtime adaptation. This mechanism allows adapting user agents according to changes in the user model. User agents follow a defined agent architecture model composed of mental attitudes, whose aim is to reduce the gap with users' cognitive model; and (iv) a framework based on the components of our approach, which provides an infrastructure to build applications of our target domain, i.e. personal assistance software systems.

1.2.1 Aims

The main goal of the proposed thesis is to define an approach that empowers end-users to customize their personal agents. There are other more specific aims, which are presented next:

- perform an experimental study to evaluate how end-users express their preferences in their well-know domains;
- define a reference architecture to build user-customizable personal agents taking into account quality attributes;
- propose a user Domain-specific Model, which allows modeling user customizations using high-level abstractions;
- build an end-user Domain-specific Language based on this Domain-specific Model as close as possible to natural language;
- propose a model-driven mechanism that allows dynamically adapting these user-customizable personal agents based on changes on the user Domain-specific Model;
- develop a framework to support the development of user-customizable personal agents based on the proposed approach; and
- evaluate the approach with an experimental study and qualitative analysis.

1.3 Expected Contributions

The main expected contributions for this thesis are:

- empowering end-users to dynamically customize their personal agents;
- improve the understanding about how users express their preferences;
- improve the quality of the development of user-model-based systems by the definition of a reference architecture;
- provide a framework to develop user-customizable personal assistance software based on a multi-agent infrastructure; and
- define a Domain-specific Model to model user preferences and configurations, which uses end-user abstractions and can be reused across different domains, and is associated with an end-user Domain-specific Language. This DSM can be leveraged in different situations:
    1. to provide a common language to users specify their customizations;
    2. to help on mixed initiatives in which users can verify the customizations inferred by the system; and
    3. to provide a modular reasoning about user customizations, which is performed using high level abstractions and is independent of implementation technologies.

1.4 Out of the Scope

As the problems we are addressing in this thesis are part of a broader context, a set of related aspects will be left out of its scope. Nevertheless, these aspects were envisioned since the initial definitions of the approach. They can be addressed in the future and be incorporated to our work. The following issues are not going to be directly addressed by this work:

Learning. Implicit user preferences learning methods can be leveraged to reduce the effort of users specifying their preferences (and also configurations). This method can be used in a complementary way of our approach. Our architecture was devised to accommodate a learning module, however we are not focusing on this module;

Security and Privacy. When we are dealing with user preferences, information security and privacy become an important issue. The software must be robust enough to not allow malicious systems stealing private information, and this information cannot be provided without the user permission. Even though our architecture was also devised to accommodate a module that addresses security and privacy issues, we are not going to provide methods to address this issue;

Explanations for Users. Personal agents take actions according to user specifications (or an inferred specification, if a learning approach is used). Therefore, these actions can be seen as a consequence of a user specification, and the analysis of a cause-and-effect path can be difficult for users understanding it. So, it is also important that personal agents are able to explain the reasons and motivations of their actions for users;

Human Error-proofness. As personal agents take actions according to user specifications, they can be the wrong actions if users do not specify their customizations correctly. Some mechanisms can be adopted to address this issue, such as: (i) use of learning techniques to compare learned facts about users and the specifications they make. In case of the user provide an "awkward" configuration or preference, the system can require confirmations; and (ii) use of consistence analysis techniques to review the user model and detect inconsistences; and

Runtime deployment. We are aiming to provide runtime adaptations to personal agents, i.e. users may change their customization at runtime and the system is adapted to these changes. However, we are not addressing situations in which, for instance, new agents with new services need to be dynamically introduced in the system. Therefore, there is no support for runtime deployment of new components or modifying existing ones.

1.5 Outline

The remainder of this thesis proposal is organized as follows.

Section 2 discusses related work. Section 3 presents the methodology adopted to perform this work, including a schedule for the next activities. It also briefly introduces results already achieved. Appendixes A and B present two papers (one published and one submitted), which describe part of the progress already made in solving the problem stated in this thesis proposal.

2 Related Work

The main expected contribution of this thesis is the proposal of an approach that support the development of user-customizable personal agents in many aspects. Therefore, approaches that address the development of personalized user agents are presented in this chapter (Section 2.1). We have analyzed these approaches and pointed out their identified strengths and weaknesses.

One interesting work that is important to mention is the situated-automata approach proposed by Kaelbling & Rosenschein (Kaelbling n.d., Kaelbling 1991, Rosenschein & Kaelbling 1995). This work emerged in middle 80s, when researchers have started to look at reactive agents, which simply react to an environment, without reasoning about it, but the "intelligent" behavior arises as a result of an agent's interaction with its environment. Situated-automata theory is a formal semantics of embedded computation, which gives a specification of the information content of the internal states of a computer system in terms of the external states of the environment in which that system is embedded (Kaelbling n.d.). In Kaelbling & Rosenschein's work, agents are implemented as a finite-state machine, expressed as a fixed sequential circuit, which is a low-level language that can make implementing complex agent programs quite tedious. Therefore, they propose two higher-level languages, Ruller and Gapps, intended to facilitate programming intelligent agents. Ruler and Gapps are both declarative languages that allow the programmer to write high-level static facts about the environment that are "compiled away." Compilation allows static facts to be taken into account in the structure of the agent's circuitry, but frees it from the necessity of representing them explicitly. The same idea of providing views of the same system in different abstraction levels is adopted in our work. End-users are usually not able to deal with lower-level languages, therefore we provide this high-level language to allow end-users programming their agents. However, our "compilation" process is not offline, as in (Kaelbling n.d.), but it is performed at runtime. It is important to highlight that our approach is not restricted to reactive agents. We are addressing agents that may not have the ability to implicitly learn about user preferences, but have an "intelligent" behavior that makes them able to act pro-actively to achieve users' goals taking into account restrictions (soft and hard constraints) imposed by users.

2.1 Approaches for Developing Personalized User Agents

As discussed in the introduction (Section 1), Maes (Maes 1994) have made the initial steps on the development of personalized user agents. In (Maes 1994), she criticized the adoption of an end-user programming approach (Lai & Malone 1988) to develop user agents. However, Maes presented only qualitative arguments and claimed that this approach requires too much insight, understanding and effort from the end-user. Since then, most of the work we have found in the literature is related to an automated learning process. Our approach tends to minimize this user effort by providing a high-level language, with the goal of allowing users to express their instructions using a vocabulary as similar as possible to how they would instruct a human personal assistant. In addition, our language can be adopted in mixed-initiative approaches, in which users can understand, verify and make adjustments on their own model in order to increase the trust in the system. Too much effort is delegated for users (Lai & Malone 1988). Users are required to perform low-level specifications (domain modeling). Agents are programmed with rules, and are in charge of, for instance, processing incoming e-mail. These rules are similar to the ones used in nowadays e-mail clients, and advanced users do use them to facilitate processing e-mails.

A multi-agent infrastructure, named Seta2000, for developing personalized web-based systems is presented in (Ardissono, Goy, Petrone & Segnan 2005). It supports the development of recommender systems by the provision of two main behaviors: personalized suggestion of items, and the dynamic generation of user interfaces. The paper describes three main aspects of the infrastructure: (i) web communication; (ii) management of the interaction flow; and (iii) management of the user models. The main contribution from a personalization perspective of the Seta2000 is the reusable recommendation engine that can be customized to different application domains. The other components of the infrastructure deal with issues from general web-based systems and MASs, such as session management and communication among agents, which are extensively described, including implementation details. Even though this work provided a reusable infrastructure to build web-based recommender systems, it did not provide new solutions in the context of personalized systems: it leverages existing recommendation techniques and provides an extensible implemented agent-based solution. This can be seen in the related work reported in (Ardissono et al. 2005), in which Seta2000 is compared to general purpose agent platforms, such as JADE.

Huang et al. (Huang, Dai, Wei & Huang 2008) describes a personalized recommendation system based on multi-agents. The system provides an implicit user preferences learning approach, and distributes responsibilities of the recommendation process among different agents, such as learning, selection & recommendation and information collection agent. These agents are an underlying infrastructure of an intelligent user agent. As the previous discussed work, this system adopts existing techniques to build recommendation systems. It is important to highlight that our focus is not recommender systems. This family of systems typically aims to creating a user profile and based on this profile infer goods (products, food, etc) that might be interesting to users. On the other hand, our goal is to provide means for users, who already know what they want, to specify and delegate time-consuming repetitive tasks for a personal agent. Nevertheless, a cognitive model of the user is also necessary (user preferences) to enable agents to act appropriated on behalf of the users. But in recommendation systems preferences are very vague, because users are not contextualizing them for a certain situation they want to achieve.

One of the biggest, if not the biggest, projects in the context of personalized user agents is the Cognitive Assistant that Learns and Organizes (CALO) project[5] (Berry, Peintner, Conley, Gervasio, Uribe & Yorke-Smith 2006, Berry, Donneau-Golencer, Duong, Gervasio, Peintner & Yorke-Smith 2009, Yorke-Smith et al. 2009), funded by Defense Advanced Research Projects Agency (DARPA), whose goal is to support a busy knowledge worker in dealing with the twin problems of information and task overload, i.e. to create cognitive software systems, that is, systems that can reason, learn from experience, be told what to do, explain what they are doing, reflect on their experience, and respond robustly to surprise. Along the project, the research effort was mostly concentrated in the PTIME agent, which is an autonomous entity that works with its user, other PTIME agents, and other users, to schedule meetings and commitments in its user's calendar. PTIME addresses the problem of automating a repetitive task (scheduling) that we are addressing, and the approach also allows users to make explicit specifications in a natural language (meeting

[5]http://caloproject.sri.com/

constraints); however the solution is highly coupled with the domain being explored. In addition, as there is only one concern being addressed by PTIME (scheduling), there are not multiple concerns related to user customizations, which is a problem we are looking at. Furthermore, this research work did not address user configurations, i.e. the system that all users are managing are not tailored for their needs in the sense of features that the system provides. Despite this limitations, the CALO project substantially advanced on the development of user agents, also taking into account HCI issues that are essential to improve the chances of users adopting personal agents. Therefore, lessons learned from this project (Berry et al. 2009) can be leveraged in our work.

Finally, Schiaffino & Amandi made solid contributions to the development of personalized user agents from an HCI perspective. But means of an empirical study (Schiaffino & Amandi 2004), they showed what users really expect from user agents, such as the kind of interruptions they tolerate, when they are willing to delegate tasks to agents, and when agent mistakes are accepted. Nevertheless, their research focused in an issue that we are not directly addressing: when and how interrupt users. Their goal is to design agents that can provide context-aware assistance and make context-aware interruptions (Schiaffino & Amandi 2006). In this sense, Schiaffino & Amandi's work is complementary to our approach.

3 Methodology and Work in Progress

In this chapter we introduce and describe the activities to be accomplished along the next three years in order to achieve the goals defined in this proposal. In addition, we briefly describe the research work that we have already developed in the context of this thesis. Some papers associated with this research work are presented in Appendixes A and B.

A first study is presented in (Nunes, Lucena, Cowan & Alencar 2009), in which we have proposed an approach for building customized service-oriented user agents. The main idea was to capture the domain variability into a variability model, and to develop an agent SPL that supports this identified variability. A user is able to choose a configuration of the variability model, and then a customized agent is deployed into a MAS to provide a service for the user. However, the approach have not addressed dynamic adaptations, i.e. user agents do not evolve at runtime; and have not considered user preferences, but only user configurations. The case study used in this paper allowed us deriving interesting lessons learned related to the development of fine-grained variability into agent architectures, which were reported in (Nunes, Cirilo & Lucena 2009) and (Nunes, Cirilo, Cowan & Lucena 2009).

We also have studied the role of user preferences into agent architectures. We have made a survey of research work in the context of MASs, which propose agents' internal structures or mental attitudes that can be used to represent user preferences. The technical report related to this part of the work can be seen in (Nunes, Barbosa & Lucena 2009). Moreover, we have already proposed a first version of our reference architecture and the user DSM in (Nunes, Barbosa & Lucena 2010*a*, Nunes, Barbosa & Lucena 2010*b*). In this paper, we identified the main SE issues in the development of user agents, which motivated the structure of our architecture. Furthermore, besides proposing our user DSM, which is a metamodel, we showed its instantiation for two different application domains (flight and computer domains). Furthermore, several research work related to this thesis proposal have already been studied, including the following research areas: (i) user agents, which were presented in this paper; (ii) user preferences; (iii) runtime model adaptations; and (iv) self-adaptive systems. Some of them have been extensively studied and some still need further investigation.

Table 2 presents a tentative schedule of the activities to be accomplished along the next three years. These activities emerged from results already achieved, mainly the first version of our reference architecture, which allowed us identifying the main subproblems that must be addressed. Black cells indicate when a certain activity will be performed, and gray cells indicate periods that were not taken into account (past periods or periods after the end of this phd). Next, each one of these activities is detailed, as far as it is possible in a research project.

Table 2: Schedule.

	2010											
	J	F	M	A	M	J	J	A	S	O	N	D
A1	▒	▒	▒	■	■	■	■					
A2	▒	▒	▒	■	■	■	■	■				
A3	▒	▒	▒						■	■		
A4	▒	▒	▒							■	■	■

	2011											
	J	F	M	A	M	J	J	A	S	O	N	D
A4	■											
A5		■	■									
A6				■	■							
A7					■	■	■	■	■	■		
A8											■	■

	2012											
	J	F	M	A	M	J	J	A	S	O	N	D
A8	■	■	■	■								
A9					■	■	■	■	■	■		
A10									■	■	■	■

	2013											
	J	F	M	A	M	J	J	A	S	O	N	D
A10	■	■		▒	▒	▒	▒	▒	▒	▒	▒	▒
A11			■	▒	▒	▒	▒	▒	▒	▒	▒	▒

A1 – Initial Implementation. A main goal for the present semester is to perform a simple implementation of the proposed approach in order to produce concrete results. The idea is to instantiate the reference architecture proposed in (Nunes et al. 2010*b*), with the development of not sophisticated modules to show the potential of our approach. The DSM, also proposed in (Nunes et al. 2010*b*), will be used as the metamodel of the user model, and simple adaptation rules will be adopted to adapt the agent architecture as well as a commonly known learning algorithm. JADE[6] is the chosen agent platform, because it is based in "pure" (not XML files) Java, thus enabling the use of Java annotations, which is a potential mechanism to be adopted to perform runtime adaptations. The application domain of this initial implementation is trip planning.

A2 – Experimental Study of End-user Preferences Specification. In parallel to ac-

[6]http://jade.tilab.com/

tivity A1, it will be performed an experimental study of how well users are able express their preferences related to well-known domains (involved with repetitive tasks), and if they need to be exposed to concrete choice situations to be able to express their preferences.

A3 – Definition of the User DSM (review). In (Nunes et al. 2010*b*), we have proposed a first version of the user DSM that allows modeling user customizations, i.e. configurations and preferences. This activity aims at reviewing this model using the information obtained in the activity A2.

A4 – Definition of the End-user DSL. The user DSM provides the necessary vocabulary to model user customizations, using high-level abstractions. This activity is to build a language based on this DSM to be used by end-users.

A5 – Tool Support for the DSM and the End-user DSL. The goal of this activity is to produce tool support for modeling and processing both the user DSM and DSL. There are Eclipse IDE[7] components that provide support to the development of DSMs, what makes this technology a strong candidate to be used for accomplishing this activity.

A6 – Agent Architecture Definition. The transformation rules necessary to adapt user agents according to changes in the user DSM must define operations over agent components. This activity is to define the agent architecture that will be adopted, with its internal components, such as goals, plans, meta-goals and motivations.

A7 – Model Transformation Rules. This activity aims at defining transformation rules over the user DSM and the agent architecture model in order to allow the runtime adaptation. The goal is to express these model transformation rules in a formal language.

A8 – Framework Development. The proposed reference architecture provides guidance for developing user-customizable systems. We also provide models and mechanisms to support the development of systems based on this architecture. In this activity, we will build a framework, which connects the proposed models and mechanisms and serves as an infrastructure that can be instantiated to different domains. We will also build an instance of the framework that will be used in next activity.

A9 – Approach Evaluation. This activity is to evaluate the proposed approach to build user-customizable personal agents. We aim at making an experimental study using the application developed in previous sections (an instance of the framework) and end-users, in order to evaluate the effectiveness of our approach in providing means for users instructing their agents. The quality attributes of the reference architecture will be evaluated by a qualitative analysis.

A10 – Thesis writing.

A11 – Phd Thesis Defence.

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

References

Ardissono, L., Goy, A., Petrone, G. & Segnan, M. (2005), 'A multi-agent infrastructure for developing personalized web-based systems', *ACM Trans. Internet Technol.* **5**(1), 47–69.

Berry, P. M., Donneau-Golencer, T., Duong, K., Gervasio, M., Peintner, B. & Yorke-Smith, N. (2009), Evaluating user-adaptive systems: Lessons from experiences with a personalized meeting scheduling assistant, *in* 'Proceedings of the Twenty-First Innovative Applications of Artificial Intelligence Conference (IAAI'09)', Pasadena, CA, USA, pp. 40–46.

Berry, P., Peintner, B., Conley, K., Gervasio, M., Uribe, T. & Yorke-Smith, N. (2006), Deploying a personalized time management agent, *in* 'AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems', ACM, New York, NY, USA, pp. 1564–1571.

Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H. & Poole, D. (2004), 'Cp-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements', *J. Artif. Int. Res.* **21**(1), 135–191.

Clements, P. & Northrop, L. (2001), *Software product lines: practices and patterns*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Doyle, J. (2004), 'Prospects for preferences', *Computational Intelligence* **20**, 111–136.

Fowler, M., Beck, K., Brant, J., Opdyke, W. & Roberts, D. (1999), *Refactoring: Improving the Design of Existing Code*, 1 edn, Addison-Wesley Professional.

Huang, L., Dai, L., Wei, Y. & Huang, M. (2008), A personalized recommendation system based on multi-agent, *in* 'WGEC '08: Proceedings of the 2008 Second International Conference on Genetic and Evolutionary Computing', IEEE Computer Society, Washington, DC, USA, pp. 223–226.

Jennings, N. R. & Wooldridge, M. (1998), Applications of intelligent agents, *in* 'Agent technology: foundations, applications, and markets', Springer-Verlag, pp. 3–28.

Kaelbling, L. P. (1991), 'A situated-automata approach to the design of embedded agents', *SIGART Bull.* **2**(4), 85–88.

Kaelbling, L. P. (n.d.).

Kästner, C. & Apel, S. (2009), 'Virtual separation of concerns - a second chance for preprocessors', *Journal of Object Technology* **8**(6), 59–78.

Lai, K.-Y. & Malone, T. W. (1988), Object lens: a "spreadsheet" for cooperative work, *in* 'CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work', ACM, New York, NY, USA, pp. 115–124.

Luo, X., Jennings, N. R. & Shadbolt, N. (2006), 'Acquiring user tradeoff strategies and preferences for negotiating agents: A default-then-adjust method', *Int. J. Hum.-Comput. Stud.* **64**(4), 304–321.

Maes, P. (1994), ‘Agents that reduce work and information overload’, *Commun. ACM* **37**(7), 30–40.

Nunes, I., Barbosa, S. & Lucena, C. (2009), Modeling user preferences into agent architectures: a survey, Technical Report 25/09, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brazil.

Nunes, I., Barbosa, S. & Lucena, C. (2010*a*), An end-user domain specific model to drive dynamic user agents adaptations, Technical Report CS-2010-05, University of Waterloo, Waterloo, Canada.

Nunes, I., Barbosa, S. & Lucena, C. (2010*b*), An end-user domain-specific model to drive dynamic user agents adaptations, *in* ‘International Conference on Software Engineering and Knowledge Engineering (SEKE 2010) (submitted)’, USA.

Nunes, I., Cirilo, E., Cowan, D. & Lucena, C. (2009), Fine-grained variability in the development of families of software agents, *in* ‘7th European Workshop on Multi-Agent Systems (EUMAS 2009)’, Ayia Napa, Cyprus.

Nunes, I., Cirilo, E. & Lucena, C. (2009), Developing a family of software agents with fine-grained variability: an exploratory study, *in* ‘V Workshop on Software Engineering for Agent-oriented Systems (SEAS 2009)’, Fortaleza, Brazil, pp. 71–82.

Nunes, I., Lucena, C. J., Cowan, D. & Alencar, P. (2009), Building service-oriented user agents using a software product line approach, *in* ‘ICSR ’09: Proceedings of the 11th International Conference on Software Reuse’, Springer-Verlag, Berlin, Heidelberg, pp. 236–245.

Pena, J., Hinchey, M. G. & Ruiz-Cortés, A. (2006), ‘Multi-agent system product lines: challenges and benefits’, *Communications of the ACM* **49**(12), 82–84.

Pohl, K., Böckle, G. & van der Linden, F. J. (2005), *Software Product Line Engineering: Foundations, Principles and Techniques*, Springer-Verlag.

Pu, P. & Chen, L. (2008), ‘User-involved preference elicitation for product search and recommender systems’, *AI Magazine* **29**(4), 93–103.

Rogoff, K. (2010), ‘Grandmasters and global growth’, *Project Syndicate* . Available at http://www.project-syndicate.org/commentary/rogoff64/English.

Rosenschein, S. J. & Kaelbling, L. P. (1995), ‘A situated view of representation and control’, *Artif. Intell.* **73**(1-2), 149–173.

Schiaffino, S. & Amandi, A. (2004), ‘User - interface agent interaction: personalization issues’, *Int. J. Hum.-Comput. Stud.* **60**(1), 129–148.

Schiaffino, S. & Amandi, A. (2006), ‘Polite personal agents’, *IEEE Intelligent Systems* **21**(1), 12–19.

Schwab, I., Kobsa, A. & Koychev, I. (2000), Learning about users from observation, *in* ‘In Proceedings AAAI 2000 Spring Symposium: Adaptive User Interface’, AAAI Press, pp. 241–247.

Shaw, M. & Clements, P. (2006), 'The golden age of software architecture', *IEEE Softw.* **23**(2), 31–39.

Weiss, G., ed. (1999), *Multiagent systems: a modern approach to distributed artificial intelligence*, MIT Press, Cambridge, MA, USA.

Yorke-Smith, N., Saadati, S., Myers, K. L. & Morley, D. N. (2009), Like an intuitive and courteous butler: a proactive personal agent for task management, *in* 'AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems', International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 337–344.

Zambonelli, F. & Luck, M. (2005), 'The agent hell: a scenario of worst-practices in agent-based software engineering', *Potentials, IEEE* **24**(2), 23–26.