PUC

ISSN 0103-9741

Monografias em Ciência da Computação

n° 25/09

Modeling User Preferences into Agent Architectures: a Survey

Ingrid Oliveira de Nunes

Simone Diniz Junqueira Barbosa

Carlos José Pereira de Lucena

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. 25/09 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena September, 2009

Modeling User Preferences into Agent Architectures: a Survey [1]

Ingrid Oliveira de Nunes[1], Simone Diniz Junqueira Barbosa[1], Carlos José Pereira de Lucena[1]

[1] PUC-Rio, Computer Science Department, LES - Rio de Janeiro - Brazil

{ionunes,simone,lucena}@inf.puc-rio.br

Abstract. Personal agents are becoming popular for automating tasks for users. Given that these agents represent individuals, there is the need to capture user preferences and customize agents according to the users' specific needs. Several approaches have been proposed in order to model user preferences and use them in algorithms that generate an output specific for each user. In this paper we present a survey of these approaches and concepts proposed to represent user preferences, mainly into agent architectures. We also discuss how these concepts are used in the agent reasoning process.

Keywords: Multi-agent Systems, Personal Agents, User Preferences.

Resumo. Agentes pessoais estão se tornando populares por automatizar tarefas para usuários. Dado que estes agentes representam indivíduos, existe a necessidade de capturar as preferências dos usuários e customizar os agentes de acordo com as necessidades específicas de cada usuário. Muitas abordagens têm sido propostas a fim de modelar as preferências do usuário e usá-las em algoritmos para lidar com essa necessidade. Neste artigo, apresentamos uma pesquisa dessas abordagens e os conceitos que elas propuseram para representar preferências do usuário, principalmente em arquiteturas de agentes. Nós também discutimos como estes conceitos são usados no processo de raciocínio dos agentes.

Palavras-chave: Sistemas Multi-agentes, Agentes Pessoais, Preferências do Usuário.

[1]This work has been sponsored by Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

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

Multi-agent Systems (MASs) have emerged as a promising technology in order to build modern software systems, whose characteristics include pro-activity, autonomy, high interactivity and situatedness in dynamic environments. From a software engineering viewpoint, MAS is a paradigm in which the problem space is decomposed into autonomous and pro-active entities, named agents. Given the characteristics of modern software, agents become a higher level abstraction to model these complex and distributed systems, reducing the gap between the problem space and the solution space.

One of the research topics in MASs is the development of personal agents. These agents act on behalf of users, automating tasks that were previously performed by them. Examples are search agents, which search the Internet to find relevant information for users; scheduler agents, which help users to manage their schedules; and notification agents, which notify users about important events. The development of personal user agents has several challenges. First, user preferences must be captured (*preferences elicitation*) and modeled (*preferences representation*) in the system. In addition, once these preferences are modeled in the system, it must know how to use them in order to adapt its behavior according to the preferences. Therefore, given that personal agents represent individuals, they must be customized based on user preferences and needs. Another relevant issue to be taken into account in this context is how to define how agents should interact with their users in order to confirm or request relevant information, without being annoying or intrusive.

In (Nunes, Lucena, Cowan & Alencar 2009), we presented our first research work in this direction. We have proposed an approach for building customized user agents. The approach takes advantage of the interplay of MASs, Software Product Lines (SPLs) and Service-oriented Architectures (SOAs). The main idea is to capture the domain variability into a variability model, and to develop an agent SPL that supports this identified variability. Next, a user is able to choose a configuration of the variability model, and then a customized agent is deployed into a MAS in order to provide a service for the user. The variability that we have addressed in our approach, which is directly associated with user preferences, lies in services, goals and plans. These concepts were chosen because we have developed agents that follow the belief-desire-intention (BDI) model. However, there are other user preferences that cannot be represented only with these concepts. For instance, if a user preference is *"to minimize cost"* when the user requests the agent to buy a book, this is neither a service, nor a goal, nor a plan.

In this context, this paper presents a survey of the work done in modeling user preferences into agent architectures. Most of agent approaches are inspired by human nature. For instance, one widely adopted agent architecture, the BDI architecture (Rao & Georgeff 1995), is based on Bratman's studies on modeling human reasoning (Bratman 1987). As a consequence, the users' mental attitudes can be directly modeled into this agent architecture. However, some studies reported other mental attitudes that are important to human reasoning, for example motivations in the goal deliberation process. Therefore, the aim of our research is to identify the diverse concepts proposed in the literature, mainly related to agent architectures, used to model user concerns and preferences. Besides presenting selected research work, we discuss their role in the agents' reasoning process.

The remainder of this paper is organized as follows. From Section 2 to Section 7, we present and describe research work whose aim is to represent user concerns and preferences, most of them related to agents' "mental attitudes". Next, we discuss these presented approaches in Section 8. Finally, Section 9 concludes this paper.

2 Tropos

Tropos (Bresciani, Perini, Giorgini, Giunchiglia & Mylopoulos 2004) is a software development methodology that aims to allow the exploitation of the flexibility provided by agent-oriented programming. Two key features of Tropos are:

1. The notion of agent and related mentalistic notions are used in all software development phases, from early requirements analysis down to the actual implementation. The mentalistic notions are founded on BDI agent architectures (Rao & Georgeff 1995). Inspired by human reasoning, the BDI architecture is composed of three mental attitudes, which are: (i) beliefs – the agent's view of the world; (ii) desires – states that an agent wants to achieve; and (iii) intentions – desires that the agent is committed to achieve; and

2. A crucial role is given to early requirements analysis that precedes the prescriptive requirements specification of the system-to-be. In Tropos, there are earlier phases of the software development process than those supported by other agent or object-oriented software engineering methodologies.

The idea of focusing on the activities that precede the specification of software requirements, in order to understand how the intended system will meet organizational goals, has been first proposed in requirements engineering and specifically by Eric Yu in his $i*$ model. This model offers actors, goals and actor dependencies as primitive concepts. The main motivation underlying this earlier work was to develop a richer conceptual framework for modeling processes that involve multiple participants (both humans and software systems). The rationale of the $i*$ model is that, by doing an earlier analysis, one can capture not only what or how, but also why a piece of software is developed. This, in turn, supports a more refined analysis of system dependencies and encourages a uniform treatment of the system's functional and non-functional requirements.

The key concepts in Tropos are acquired as instances of a conceptual metamodel resting on the following concepts/relationships:

Actor. An actor represents a physical, social or software agent, as well as a role or position. It models an entity that has strategic goals and intentionality within the system or the organizational setting. The definition of software agent adopted is the one of classical artificial intelligence, i.e. a software having properties such as autonomy, social ability, reactivity, and proactivity. In addition, in Tropos, a role is defined as an abstract characterization of the behavior of a social actor within some specialized context or domain of endeavor, and a position represents a set of roles, typically played by one agent. An agent can occupy a position, while a position is said to cover a role;

Goal. A goal represents the actors' strategic interests. However, in Tropos, hard goals (referred to as goals) are distinguished from softgoals. The latter have no clear-cut definition and/or criteria for deciding whether they are satisfied or not. According to

(Chung, Nixon, Yu & Mylopoulos 1999), this different nature of achievement is underlined by saying that goals are *satisfied*, whereas softgoals are *satisficed*. Softgoals are typically used to model non-functional requirements;

Plan. At an abstract level, a plan represents a way of doing something. The execution of plan can be a means for satisfying one or more goals or softgoals;

Resource. It represents a physical or an informational entity;

Dependency. Between two actors, it indicates that one actor depends for some reason on the other, in order to attain some goal, execute some plan, or deliver a resource. The former actor is called the *depender*, while the latter is called the *dependee*. The object around which the dependency centers is called *dependum*. In general, by depending on another actor for a *dependum*, an actor is able to achieve goals that it would otherwise be unable to achieve on its own, or not as easily, or not as well. At the same time, the *depender* becomes vulnerable. If the *dependee* fails to deliver the *dependum*, the *depender* would be adversely affected in its ability to achieve its goals;

Capability. A capability represents the ability of an actor to define, choose and execute a plan for the fulfillment of a goal, given certain world conditions and in presence of a specific event; and

Belief. An actor's knowledge of the world.

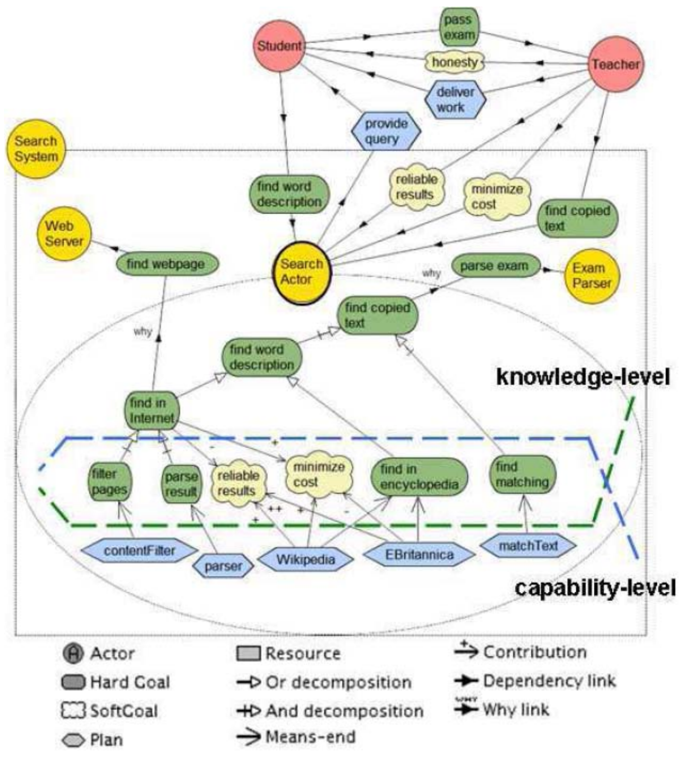

Figure 1: Tropos architectural design: Agent knowledge and capability levels

Figure 1 (Penserini, Perini, Susi, Morandini & Mylopoulos 2007) shows most of the concepts previously described. It illustrates a fragment of a goal-oriented Tropos specification of a search system's requirements. The system is intended to support students and teachers in exam-related activities.

Tropos allows to capture requirements in terms of high level concepts, such as goals, beliefs and plans, which are usually used to model agents. One deficiency of Tropos is the gap between the analysis and design models and the implementation model. As a consequence, implementing a MAS modeled with Tropos is not straightforward. In order to solve this problem, in (Penserini et al. 2007) a strategy is proposed for implementing a MAS specified with Tropos with the Jadex agent platform.

Another issue that we have identified in Tropos models is that they provide a relationship between softgoals and plans, whose aim is to express how a plan contributed to achieving a softgoal. Contributions are labeled with one or more "+", or one or more "-", indicating whether the plan contributed positively or negatively for achieving the softgoal, and the intensity of this contribution. The main problem that we have identified is that this is not static information as Tropos models define, but the contribution of a plan to a softgoal may depend on the context.

Finally, Tropos models typically have several concepts with several relationships among them. Consequently, models are sometimes difficult to understand.

3 Incorporating Motivations into BDI Agents

As presented in previous section, the BDI architecture defines that agents have three mental attitudes: beliefs, desires, and intentions. However, there are some extensions of this architecture. One of them incorporates motivations into agent architectures.

In (Norman & Long 1995), motive and motivation are defined and their difference from a goal is also described. According to (Norman & Long 1995), in human terms a "motive" may be described as a need or a desire that causes a person to act. In information-processing terms, it is a variable that has influence on the decision-making process and may cause action. "Motivation" is a driving force (dependent both on the internal state of the agent and the state of the external world) that arouses and directs action towards the achievement of goals. To illustrate this distinction, all humans have the motive of hunger, but no person will have sufficient motivation to act to satisfy their hunger at all times. These motivations reflect the context within which the agent is acting and ensure that any decisions have some relevance to this context.

A "goal", in turn, is distinct from a motive; it denotes a state that the system can achieve/prevent through action. A goal may be satisfied by the agent achieving/preventing the state in the required conditions, where a motive is a force biasing the decision-making mechanism and therefore cannot be satisfied in the same way. For example, the goal to have eaten a sandwich may be derived from the motive of hunger, for instance. This goal is satisfied once the meal is finished and the bill paid, but the motive will not disappear; the satisfaction of the goal will result in mitigating the influence of the hunger motive on the decision-making mechanisms.

Based on this proposal of adding motivations to the BDI architecture, research work was proposed. Norman & Long (Norman & Long 1995) proposed a goal creation system that uses a set of motives to trigger the "motivated agent" into considering the creation of a goal. These motives act as a filtering mechanism, extracting the contexts that may warrant the creation of a goal, and hence minimizing the computational resources required for creating sensible proactive goals. In (Coddington & Luck 2003), it is described how modeling the motivations of an agent can affect AI planning in terms of generating goals, assigning values indicating the priority of each goal, and choosing the best plan (i.e. the plan that best supports the agent's motivations). In (Meneguzzi & Luck 2007), Meneguzzi & Luck argue that, although seldom considered, a flexible meta-level reasoning component is a valuable addition to any agent architecture. So they describe such a component for use in BDI architectures, underpinned by a model of motivation and a motivation-based description language, and empirically demonstrate its effectiveness.

4 Incorporating User-given Constraints and Objectives into BDI Agents

AgentSpeak(L) is one of the most influential abstract languages based on the BDI architecture. It is an agent framework/language with explicit representations of beliefs and intentions for agents. This agent programming language was initially introduced by Rao (Rao 1996). AgentSpeak(L) is a programming language based on a restricted first-order language with events and actions. In (Dasgupta & Ghose 2006), this language is extended with constraint-handling capabilities. According to (Dasgupta & Ghose 2006), incorporating constraints into a reactive BDI agent programming language can lead to better expressive capabilities as well as more efficient computation (in some instances). More interestingly, the use of constraint-based representations can make it possible to deal with explicit agent objectives (as distinct from agent goals) that express the things that an agent may seek to optimize at any given point in time.

The concept of using constraints in AgentSpeak(L) was introduced by Ooi et al. (Ooi & Ghose 1999) where they show that the integration of constraints in a high-level agent specification language yields significant advantages in terms of both expressivity and efficiency. This work was extended in (Dasgupta & Ghose 2006) by incorporating explicit objectives beside the constraints to form the language Constraint AgentSpeak with Objectives (CASO). In addition, they describe some efficient plan and intention selection methods which would result in higher expressivity and more efficient computation, which has not been addressed in either AgentSpeak(L) introduced by (Rao & Georgeff 1995) or by (Ooi & Ghose 1999).

Informally, an agent program in CASO consists of a set of beliefs B, a set of constraints C, an objective function O, a set of events E, a set of intention I, a plan library P, a constraint store CS, an objective store OS and three selection functions S_E, S_P, S_I to select an event, a plan and an intention, respectively, to process, and n_p and n_i are the two parameters which denote the number of steps to look-ahead for the plan and intentions selection, respectively. In CASO, a constraint-directed improvisation is incorporated into the computation strategy employed during the interpretation process. Constraint logic programming (CLP) combines the flexibility of logic with the power of search to provide high-level constructs for solving computationally hard problems such as resource allocation.

In (Dasgupta & Ghose 2008), it is presented an implementation of CASO, which provides the user with the flexibility of adding explicit objectives and constraints to achieve final goals. CASO uses a modified version of Jason, the well-known BDI AgentSpeak(L) interpreter, together with another open-source constraint solver ECLiPSe, thereby combining reactive combining agent programming with constraint solving techniques. In the future, Dasputa & Ghose plan to extend CASO to incorporate user preferences as c-semiring.

5 Reasoning about Soft Constraints

Many artificial intelligence synthesis problems such as planning or scheduling may be modeled as Constraint Satisfaction Problemss (CSPs). A CSP is typically defined as the problem of finding any consistent labeling for a fixed set of variables satisfying all given constraints between these variables. However, for many real tasks such as job-shop scheduling and time-table scheduling, all these constraints do not have the same significance and do not have to be necessarily satisfied. A first distinction can be made between hard constraints, which every solution should satisfy, and soft constraints, whose satisfaction does not have to be certain.

Soft constraints (Bistarelli, Montanari & Rossi 1997, Rossi 2008) model quantitative preferences by generalizing the traditional formalism of hard constraints. In a soft constraint, each assignment to the variables of a constraint is annotated with a level of its desirability, and the desirability of a complete assignment is computed by a combination operator applied to the local preference values. By choosing a specific combination operator and an ordered set of levels of desirability, a specific class of soft constraints can be selected.

Two main challenges in dealing with soft constraints are how to represent them and how to reason about them. In (Bistarelli et al. 1997), it is defined a constraint solving framework, named SCSP (for Semiring-based CSP), where all such extensions, as well as classical CSPs, can be cast. The main idea is based on the observation that a semiring (i.e., a domain plus two operations satisfying certain properties) is all that is needed to describe many constraint satisfaction schemes. In fact, the domain of the semiring provides the levels of consistency (which can be interpreted, for instance, as cost, or degrees of preference, or probabilities), and the two operations define a way to combine constraints together. More precisely, the notion of constraint solving over any semiring is defined. Specific choices of the semiring will then give rise to different instances of the framework, which may correspond to known or new constraint-solving schemes.

Soft constraints model quantitative preferences. In order to model qualitative preferences, (Boutilier, Brafman, Hoos & Poole 2003) proposed a graphical representation, CP-nets, that can be used for specifying preference relations in a relatively compact, intuitive, and structured manner using conditional *ceteris paribus* (all else being equal) preference statements. CP-nets can be used to specify different types of preference relations, such as a preference ordering over potential decision outcomes or a likelihood ordering over possible states of the world. However, it is mainly the first type – preferences over the outcomes of decisions – that motivates the development of CP-nets. The inference techniques for CP-nets described in (Boutilier et al. 2003) focus on two important, related questions: how to perform preferential comparison between outcomes, and how to find the optimal outcome given a partial assignment to the problem attributes.

Some approaches aim at combining soft-constraints and CP-nets in order to reason about both quantitative and qualitative preferences. For instance, in (Domshlak, Rossi, Venable & Walsh 2003), a framework is proposed to reason simultaneously about qualitative conditional preference statements and hard and soft constraints.

6 Acquiring User Tradeoff Strategies and Preferences

The work presented in (Luo, Jennings & Shadbolt 2006) focuses mostly on the problem of capturing user preferences and not on algorithms to reason about it, which are typically addressed by existing research work related with user preferences. This research work aims at starting bridging the knowledge acquisition gap between the negotiating agents' owners and the negotiation algorithms that their agents use. Specifically, in (Luo et al. 2006) the authors focus on exploring how the necessary knowledge about a user's negotiation tradeoffs can be acquired. In more detail, the term *tradeoff* among negotiation attributes refers to combinations of the attributes' values in which the worsening of some attributes is used to get other attributes' values improved. The set of all such potential combinations is called the *user's tradeoff strategy*, and its individual elements are called *tradeoff alternatives*. Some of these alternatives are more preferred by the users and this ordering is here referred to as a *user's tradeoff preference*. Such a preference is realized by assigning to each tradeoff alternative a satisfaction degree, where the more preferred an alternative is, the higher the corresponding satisfaction degree.

The preferences acquisition method works in the following way in (Luo et al. 2006). First, conduct a structured interview with the user in order to suggest the attributes between which the tradeoff could or should be made. Second, ask users to adjust the suggested default curve of the tradeoff strategy by improving some attribute to see how much worse the attribute being traded off can be made while still being acceptable. Finally, adjust the suggested default preference on the tradeoff alternatives by adjusting the parameters that configure the preference. This *default-then-adjust* approach is chosen for a number of reasons. First, in many negotiation scenarios it is not obvious between which negotiation attributes a tradeoff should be made. Second, according to standard business negotiation theory, users should be allowed to explore the space of tradeoff alternatives as exhaustively as they are willing to do. Third, the authors want to minimize the user's workload, but to maximize the flexibility to express complex tradeoff strategies and preferences where they exist.

In (Luo et al. 2006), the author first provides a formal definition for notions of user strategies and preferences with respect to negotiation tradeoffs between pairs of attributes that are considered the most common. Based on this, they present their default-then-adjust method for acquiring them. To demonstrate the effectiveness of the method for acquiring negotiation tradeoff strategies and preferences, they describe a prototype system for the accommodation renting scenario. Finally, they present an empirical evaluation of the prototype system.

7 Value-based Argumentation Frameworks

Along the past few years, argumentation theory has been of research interest in computer science. The most widely adopted definition of argumentation is as follows:

Argumentation is a verbal and social activity of reason aimed at increasing (or decreasing) the acceptability of a controversial standpoint for the listener *or reader, by putting forward a constellation of propositions intended to justify (or refute) the standpoint before a rational judge.* ((Grootendorst, Henkemans & van Eemeren 1996), page 5)

Forming and revising beliefs and decisions and rational interaction are two typical applications of argumentation in MASs. The notion of an Argumentation Framework (AF), introduced by Dung (Dung 1995), allows to explore a system of conflicting arguments, in which arguments are abstracted to entities whose role is solely determined by their relation to other arguments. In (Bench-Capon 2002), Bench-Capon has extended AF in order to incorporate values to its definition. This extended version of a AF consists of a Value-based AF.

The concept of values is inspired in moral and legal debates. In (Bench-Capon 2002), no formal definition for a value is given, but it is explained by means of an example: *"Suppose two people are debating whether income tax should be raised. One argues that it should because to do so would promote social equality, while the other argues that it should not, since enterprise should be rewarded. Certainly these arguments conflict, but neither party is likely to withdraw his argument in the face of the other. The point is that one believes that it is more important to promote equality than to reward enterprise, and the other ranks these purposes differently."* The idea is that arguments can promote or demote social values, and they can be influential to one's decision.

Value-based AFs have been used in the practical reasoning process of agents. In (Atkinson, Bench-Capon & McBurney 2006), the following argumentation scheme has been proposed with its associated critical questions: "In the circumstances R we should perform action A to achieve new circumstances S which will realize some goal G which will promote some value V." Based on this argumentation scheme and its critical questions, it is possible to reason about possible actions to achieve a certain goal considering the values that actions promote or demote.

8 Discussion

In previous sections, we presented several works that proposed the introduction of new concepts, most of them into agent architectures, in order to represent user concerns and preferences to allow the system to present a behavior according to specific user needs. Next, we summarize each one of these concepts:

Belief. Beliefs represent the informational state of the agent, i.e. its beliefs about the world (including itself and other agents). The term belief is adopted rather than knowledge because it recognizes that what an agent believes may not necessarily be true (and in fact may change in the future);

Goal. Goals represent the motivational state of the agent. They represent objectives or situations that the agent would like to accomplish or bring about. Goals may conflict with each other. Therefore, agents should choose which goals it wants to bring about. These selected goals that the agent is committed to achieve are named intentions;

Soft-goal. A soft-goal is used to model quality attributes for which there are no a priori, clear criteria for satisfaction, but are judged by actors as being sufficiently met;

Plan. Plans are sequences of actions that an agent can perform to achieve one or more of its intentions;

Motivation. Motivations are a driving force (dependent both on the internal state of the agent and the state of the external world) that arouses and directs action toward the achievement of goals;

Objective. Objectives are some utility function that needs to be maximized;

Constraint. Constrains are restrictions that determine which combinations of values are acceptable when finding assignments for variables of a problem;

Soft-constraint. Soft constraints model quantitative preferences by generalizing the traditional formalism of hard constraints;

(Social) Value. Social value orientations are individual differences in how people evaluate outcomes for themselves and others in interdependent situations;

Tradeoff preference. Tradeoff preference refers to an ordering on a set of tradeoff alternatives (this is realized through assigning each alternative a satisfaction degree). A tradeoff alternative is a combination of issues' values in which making some values worse can be counterbalanced by improving other values.

These concepts have different roles into an agent architecture and into the agent reasoning process. The reasoning process can be divided into two sub-processes: (i) *Theoretical reasoning* is reasoning directed towards beliefs – concerned with deciding what to believe in; and (ii) *Practical reasoning* is reasoning directed toward actions – concerned with deciding what to do. (ii) can, in turn, be split into two parts: (i) Deliberation – deciding which state of affairs we want to achieve; and (ii) Means-ends reasoning – deciding how to achieve such state of affairs.

Concept	**Reasoning Process**	
	Input	**Output**
Belief	Practical reasoning	Theoretical reasoning
Goal	Means-ends reasoning	Deliberation
Soft-goal	Means-ends reasoning	-
Plan	Means-ends reasoning (plan library)	Means-ends reasoning (selected plan)
Motivation	Deliberation	-
Objective	Means-ends reasoning	-
Constraint	Means-ends reasoning	-
Soft-constraint	Means-ends reasoning	-
(Social) Value	Means-ends reasoning	-
Tradeoff preference	Means-ends reasoning	-

Table 1: Concepts and their role into the agent reasoning process.

Table 1 presents how the described concepts are typically used in the agent reasoning process. The table is not definitive, meaning that some concept may be used in other sub-processes of the reasoning process. The idea is to capture how they were used in the investigated research work. It is important to notice that most of the proposed concepts that extend the BDI model aim at helping the the means-ends reasoning, i.e. selecting an appropriate plan for an agent to achieve a goal (or selecting actions for creating a plan). However, these new concepts are not deliberated and reasoned by the agent itself (represented by an empty output column); they must be directly informed by users or be defined at design time. In particular, the main goal of the work proposed in (Luo et al. 2006) is to capture tradeoff preferences from users.

Table 1 shows that concepts are used as input or output of different parts of the process. Therefore, user preferences get spread and tangled into the agent architecture, and it is hard to understand, and consequently maintain, which components of the agent architecture represent user preferences. This motivates the proposal of a separate model that captures and represents user preferences, which is traced into the agent architecture.

9 Conclusion

Modern software systems tend to be autonomous, pro-active, context-aware and highly interactive. This autonomy property refers to systems acting without the intervention of humans. MASs have emerged as a promising technology to develop these complex systems. In this context, personal agents have been explored to automate tasks for users. In order to these agents act according to the user's needs, his preferences should be captured and modeled into the agent architecture.

In this paper we presented a survey of approaches that model high-level concepts into agent architectures that may represent user interests. These approaches aim at modeling user concerns that arise from specific domains and at proposing algorithms to take them into account into the agent reasoning process. One of the investigated approaches focused on the problem of capturing user preferences, and not in the algorithm to deal with them. The concepts that we introduced by the presented approaches are: (i) belief; (ii) goal; (iii) soft-goal; (iv) plan; (v) motivation; (vi) objective; (vii) constraint; (viii) soft-constraint; and (ix) value. We also presented discussions about these concepts, mainly related to their role into the agent reasoning process. Most of them are used in the means-end reasoning to determine which course of action an agent should take.

The survey presented in this paper motivates the development of a separate model to represent user preferences, so they do not get buried into the agent architecture. As future work, our goal is to define a model-driven approach composed of user preferences and agent architecture models. In addition, a traceability model allows to define relationships between these two models. With such models, we aim at developing agents that adapt their behavior based on changes in user preferences.

References

Atkinson, K., Bench-Capon, T. J. M. & McBurney, P. (2006), 'Computational representation of practical argument.', *Synthese* **152**(2), 157–206.

Bench-Capon, T. (2002), Value-based argumentation frameworks, *in* 'Proceedings of Non Monotonic Reasoning', pp. 444–453.

Bistarelli, S., Montanari, U. & Rossi, F. (1997), 'Semiring-based constraint satisfaction and optimization', *J. ACM* **44**(2), 201–236.

Boutilier, C., Brafman, R. I., Hoos, H. H. & Poole, D. (2003), 'Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements', *Journal of Artificial Intelligence Research* **21**, 2004.

Bratman, M. E. (1987), *Intention, Plans, and Practical Reason*, Cambridge, MA.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. & Mylopoulos, J. (2004), 'Tropos: An agent-oriented software development methodology', *Autonomous Agents and Multi-Agent Systems* **8**(3), 203–236.

Chung, L., Nixon, B. A., Yu, E. & Mylopoulos, J. (1999), *Non-Functional Requirements in Software Engineering*, Springer.

Coddington, A. & Luck, M. (2003), Towards motivation-based plan evaluation, *in* I. Russell & S. Haller, eds, 'Sixteenth International FLAIRS Conference', AAAI Press, pp. 298–302.

Dasgupta, A. & Ghose, A. K. (2006), Caso: a framework for dealing with objectives in a constraint-based extension to agentspeak(l), *in* 'ACSC '06: Proceedings of the 29th Australasian Computer Science Conference', Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 121–126.

Dasgupta, A. & Ghose, A. K. (2008), Implementing reactive bdi agents with user-given constraints and objectives, *in* 'Proc. of the AAMAS-2008 Workshop: From Agent Theory to Agent Implementation'.

Domshlak, C., Rossi, F., Venable, K. B. & Walsh, T. (2003), Reasoning about soft constraints and conditional preferences: Complexity results and approximation techniques, *in* 'IJCAI-2003', Morgan Kaufmann, pp. 215–220.

Dung, P. M. (1995), 'On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games', *Artificial Intelligence* **77**(2), 321–357.

Grootendorst, R., Henkemans, F. S. & van Eemeren, F. H. (1996), *Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Developments*, Lawrence Erlbaum.

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.

Meneguzzi, F. & Luck, M. (2007), Motivations as an abstraction of meta-level reasoning, *in* 'CEEMAS '07: Proceedings of the 5th international Central and Eastern European conference on Multi-Agent Systems and Applications V', Springer-Verlag, Berlin, Heidelberg, pp. 204–214.

Norman, T. J. & Long, D. (1995), Goal creation in motivated agents, *in* 'ECAI-94: Proceedings of the workshop on agent theories, architectures, and languages on Intelligent agents', Springer-Verlag New York, Inc., New York, NY, USA, pp. 277–290.

Nunes, I., Lucena, C., Cowan, D. & Alencar, P. (2009), Building service-oriented user agents using a software product line approach, *in* 'ICSR 2009'.

Ooi, B. H. & Ghose, A. K. (1999), Constraint-based agent specification for a multi-agent stock brokering system, *in* 'IEA/AIE '99: Proceedings of the 12th international conference on Industrial and engineering applications of artificial intelligence and expert systems', Springer-Verlag New York, Inc., Secaucus, NJ, USA, pp. 409–419.

Penserini, L., Perini, A., Susi, A., Morandini, M. & Mylopoulos, J. (2007), A design framework for generating bdi-agents from goal models, *in* 'AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems', ACM, New York, NY, USA, pp. 1–3.

Rao, A. S. (1996), AgentSpeak(L): BDI agents speak out in a logical computable language, *in* R. van Hoe, ed., 'Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World', Eindhoven, The Netherlands.

Rao, A. S. & Georgeff, M. P. (1995), BDI-agents: from theory to practice, *in* 'Proceedings of the First International Conference on Multiagent Systems (ICMAS 1995)', San Francisco.

Rossi, F. (2008), 'Preferences, constraints, uncertainty, and multi-agent scenarios'.