A Formal Approach for Contextual Planning Management: Application to Smart Campus Environment

Ahmed-Chawki Chaouche, Amal El Fallah-Seghrouchni, Jean-Michel Ilié, Djamel Eddine Saïdouni

A Formal Approach for Contextual Planning Management: Application to Smart Campus Environment

Ahmed-Chawki Chaouche[1,2], Amal El Fallah Seghrouchni[1], Jean-Michel Ilié[1] and Djamel Eddine Saïdouni[2]

[1] LIP6 Laboratory, University of Pierre and Marie Curie
4 Place Jussieu, 75005 Paris, France
{ahmed.chaouche,amal.elfallah,jean-michel.ilie}@lip6.fr
[2] MISC Laboratory, University Constantine 2
Ali Mendjeli Campus, 25000 Constantine, Algeria
saidouni@misc-umc.org

Abstract. In this paper, we address the building of ambient systems as autonomous and context-aware intelligent agents. The original contribution is an algebraic language, namely AgLOTOS, used to automatically build the plan of an agent from its intentions. As plans are formally conceived as a structured set of concurrent processes, we show how to define a guidance service, helping the agent to maximize the satisfaction of its intentions. The underlying structure, called Contextual Planning System (CPS), takes the contextual information into account to predict the ability to execute the processes in plans. The last part of the paper talks about our current experiment integrating the proposed technique to assist user in a smart campus university.

Keywords: context-awareness, BDI agent, planning language, planning guidance, smart campus.

1 Introduction

Multi-agent System (MAS) approaches offer interesting frameworks for the development of ambient intelligence (AmI) systems, since their agents are considered as intelligent, proactive and autonomous [1]. This paper introduces an efficient planning management process into the architecture of the agent. In particular, we aim at offering to each AmI agent, a powerful predictive service. Like in other recent MAS approaches, e.g. [2, 3], which are dedicated to the planning and the validation of agents in MAS, we focus on one agent rather than on the whole MAS. Regarding AmI systems, this eases us to consider whatever dynamic features in the environment for the agents and to propose solutions consistent with the openness of the system.

Belief-Desire-Intention (BDI) are well-known intentional structure emphasizing the reasoning tasks of the agent, up to obtain rationality properties. Such properties are really appreciate in AmI systems since this enforces the confidence

on the system. The authors of [4] took profit from the fact that the plan of a BDI agent can be derived from its intentions, themselves resulting from the reasoning of the BDI interpreter [5]. In this context, the AgLOTOS language was defined to specify the plans accordingly to the following two criteria: *(1)* enhance the modular and concurrent aspects related to the execution of plans, up to see the plan as composed of concurrent processes, *(2)* handle the well-ordered composition of intentions, i.e. an agent can attribute weights to privilege execution with respect to some intentions.

In this paper, the AgLOTOS language is considered again but its semantics is enriched to automatically produce a state transition structure, namely *Contextual Planning System* (*CPS* for short). The aim is to capture the evolution of the plan in a predictive way, meaning that actions are supposed to be run successfully, but also that the context of the agent can evolve under the execution of actions. Automatic searches on this structure will allow us to propose guidance services, particularly helpful for the decision of agent in expected contexts.

Moreover, we aim at showing how our formal approach can be embedded in the development of the AmI agents. To our opinion, this contributes to making the operational bridge between AmI software engineering and formal approaches. Our project consisting in the design of a smart university campus is the very right place to embed our AmI agents architecture through a float of smart-devices dedicated to assisting users. In this project, the discovery of physical locations and the moves of users are taken into account. Unlike pure MAS approaches, this cannot be reduced to social problem and communication between software agents.

The outline of the paper is the following: Section 2 recalls the agent software architecture we consider, namely the HoA architecture, and its specific planning language used to associate plans with intentions. In Section 3, a contextual planning management is presented based on the building of the CPS structure. Section 4 details the concrete stages of the smart-campus project to conceive the AmI systems. A realistic scenario is given as an illustration of the concepts proposed in the paper. The last section concludes and brings out our next perspectives.

2 The HoA Architecture and its Planning Language

Figure 1 highlights the agent architecture we consider for AmI systems. Called *Higher-order Agent architecture* (*HoA*), it enhances a clear separation in three processes:

- The *Context process* is in charge of the context information of the agent. It is triggered by new perceptions of the environment and also by internal events informing about the executions of actions. At a low level, it is in charge of observing the realization of the action executions, in order to state they are successfully achieved or not.
- The *Mental process* corresponds to the reasoning part of the agent. It is notified by the context process so that it can be aware of the important context

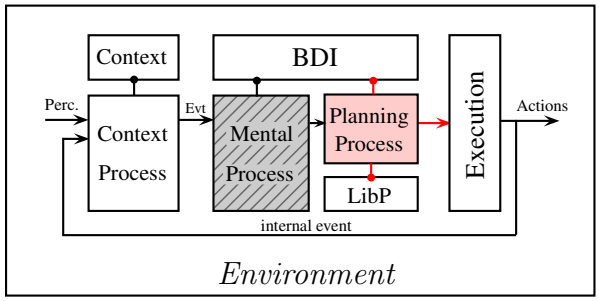

Fig. 1. Higher-order Agent architecture

changes and can provoke possible revisions of the beliefs (B), desires (D), and intentions (I) data.
As highlighted in Figure 1, the mental process represents the reasoning mechanism, which manages the BDI states of the agent. Triggered by the perceived events, it updates the B,D and I structures. In order to organize its selected intentions, the mental process is able to schedule them by associating with each one a given weight.

– The *Planning process* is called by the mental process. Helped by a library of plans (*LibP*), it mainly produces a plan of actions from the set of weighted intentions, but also offers some services related to the management of plans (see Section 3).

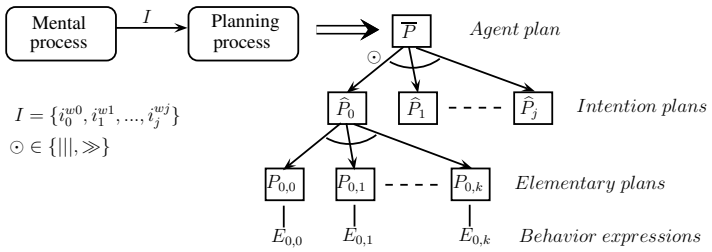

Fig. 2. Agent planning structure

In our approach for each BDI state, the plan of the agent is described by using the *AgLOTOS* language, as detailed in [4]. The language itself extends the LOTOS language [6] in order to specify concurrency between actions in plans. In addition as schemed by Figure 2, it refers to *two level planning structures*: *(1)* the *Agent plan* is made of sub-plans called *Intentions plans*, each one dedicated to achieve the associated selected intention; *(2)* each intention plan is an alternate of several sub-plans, called *Elementary plans*, extracted from the LibP library. This allows one to consider different ways to achieve the associated intention.

Further, we assume that the LibP library is indexed by the set of all the possible intentions for the agent.

Syntax of Elementary Plans. Each elementary plan is specified by a pair composed of a name to identify it and an AgLOTOS expression to feature its behavior. Consider that the names of elementary plans are ranged over $P, Q, ...$ and that the set of all the possible behavior expressions is denoted $\mathcal{E}$, ranged over $E, F, ...$ The AgLOTOS elementary expressions are written by composing (observable) actions through the LOTOS operators. The syntax of an elementary plan P is defined inductively as follows:

$$
\begin{array}{lll}
P ::= E & & \textit{Elementary plan} \\
E ::= & \textit{exit} \mid \textit{stop} & \\
 & \mid a; E \mid E \odot E & (a \in \mathcal{O}) \\
 & \mid \textit{hide } L \textit{ in } E & \\
\mathcal{H} ::= & \mid \textit{move}(\ell) & (\mathcal{H} \subset \mathcal{O}, \ell \in \Theta) \\
 & \mid x!(\nu) \mid x?(\nu) & (x \in \Lambda, \nu \in \mathcal{M}) \\
\odot & = \{ \; |||, |[L]|, ||, [\,], \gg, [> \; \} &
\end{array}
$$

The expression of an elementary plan refers to a (finite) set $\mathcal{O}$ of observable actions which are practically described as instantiated predicates, below ranged over $a, b, ...$ This set includes the subset $\mathcal{H}$ of the so-called AmI primitives which represent the mobility and communication, based on the two following assumptions about the AmI system: *(1)* every agent can perceive the enter and leave of other agents in the AmI system, *(2)* it can suggest some move between the AmI system locations and *(3)* it can communicate with another agent in the system. In the syntax, the primitive $\textit{move}(\ell)$ is used to represent the move of the assisted user to some location ℓ ($\ell \in \Theta$, a finite set of locations). The action $x!(\nu)$ specifies the emission to the agent x ($x \in \Lambda$, the set of neighbor agents) of the message ν ($\nu \in \mathcal{M}$, the set of possible messages), whereas, the expression $x?(\nu)$ means that the message ν is received from some agent x.

In addition, two non-observable actions are also introduced, so that the total set of actions is denoted $Act = \mathcal{O} \cup \{\tau, \delta\}$, where $\tau \notin \mathcal{O}$ is the internal action and $\delta \notin \mathcal{O}$ is a particular observable action which features the successful termination of the considered elementary plan.

The basic expression *stop* specifies a plan behavior without possible evolution and *exit* represents the successful termination of some plan. In the syntax, the set $\odot$ represents the standard LOTOS operators: $E\,[\,]\,E$ specifies a non-deterministic choice, $\textit{hide } L \textit{ in } E$ a hiding of the actions of L that appear in E with L being any subset of $\mathcal{O}$, $E \gg E$ a sequential composition and $E\,[>\,E$ the interruption. The LOTOS parallel composition, denoted $E\,|[L]|\,E$ can model both synchronous composition, $E\,||\,E$ if $L = \mathcal{O}$, and asynchronous composition, $E\,|||\,E$ if $L = \emptyset$. In fact, the AgLOTOS language exhibits a rich expressivity such that the sequential executions of plans appears to be only a particular case.

Syntax of Agent Plans. The building of an agent plan requires adding the following AgLOTOS operators to compose some elementary plans:

– at the agent plan level, the parallel $|||$ and the sequential $\gg$ composition operators are used to build an agent plan from the intentions of the agent and the associated weights.
– the *alternate composition* operator, denoted $\Diamond$, allows to specify an alternate of elementary plans. In particular, an intention is satisfied iff at least one of the associated elementary plans is successfully terminated.

Let $\widehat{\mathcal{P}}$ be the set of names used to identify the possible intention plans: $\widehat{P} \in \widehat{\mathcal{P}}$ and let $\overline{\mathcal{P}}$ be the set of names qualifying the possible agent plans: $\overline{P} \in \overline{\mathcal{P}}$.

$$\begin{array}{lll} \widehat{P} ::= & P \mid P \Diamond \widehat{P} & \textit{Intention plan} \\ \overline{P} ::= & \widehat{P} \mid \widehat{P} \,|||\, \overline{P} \mid \widehat{P} \gg \overline{P} & \textit{Agent plan} \end{array}$$

With respect to the set of intentions I of the agent, the agent plan is formed in two steps: *(1)* by an extraction mechanism of elementary plans from the LibP library, *(2)* by using the composition functions called *options* and *plan*:

– $options : \mathcal{I} \rightarrow \widehat{\mathcal{P}}$, yields for any $i \in \mathcal{I}$, an intention plan of the form $\widehat{P}_i = \Diamond_{P \in libP(i)}\ P$.
– $plan : 2^I \rightarrow \overline{\mathcal{P}}$, creates the final agent plan $\overline{P}$ from the set of weighted intentions I. Depending on how I is ordered, the intention plans yielded by the different mappings $\widehat{P}_i = options(i)$ $(i \in I)$ are composed by using the AgLOTOS composition operators $|||$ and $\gg$.

The function $weight : I \longrightarrow \mathbb{N}$ that defines the weights of the intentions, in fact yields the way to compose the corresponding intention plans. The intention plans corresponding to the same weight are composed by using the concurrent parallel operator $|||$. In contrast, the intention plans corresponding to distinct weights are ordered by using the sequential operator $\gg$. For instance, let $I = \{i_0^1, i_1^2, i_2^1, i_3^0\}$ be the considered set of intentions, such that the superscript information denotes a weight value, and let $\widehat{P_0}, \widehat{P_1}, \widehat{P_2}, \widehat{P_3}$ be their corresponding intention plans, the constructed agent plan could be viewed (at a plan name level) as: $plan(I) = \widehat{P_1} \gg (\widehat{P_0}|||\widehat{P_2}) \gg \widehat{P_3}$.

A Simple AmI Example. Let us consider the following AmI scenario presented in [7], where Alice and Bob are two users of some University, each one assisted by a HoA software agent. The proposed problem of Alice is that she cannot make the two following tasks in the same time: *(1)* to meet with Bob in the location ℓ_1, and *(2)* to get her exam copies from the location ℓ_2. Clearly, the Alice's desires are conflicting since Alice cannot be in two distinct locations simultaneously. However, after having perceived that Bob is in ℓ_2, meaning in the same location as the exam copies, Alice asks for his help to bring her the copies.

The intentions of Alice and Bob are specified separately within their respective agents. These last ones can pervasively coordinate to help achieving the intentions of their assisted users. Here, the actions in plans are simply expressed by using instantiated predicates, like $get\_copies(\ell_2)$. Intention plans are composed from elementary plans which are viewed as concurrent processes, terminated by *exit*, *a la LOTOS*.

Alice's scenario
$I_A = \{meeting(Bob, \ell_1), asking(Bob, get\_copies(\ell_2))\}$
$\overline{P_A} = Bob!(get\_copies(\ell_2)); exit \gg meet(Bob); exit$

Bob's scenario
$I_B = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$
$\overline{P_B} = get\_copies(\ell_2); exit \mid\mid\mid move(\ell_1); meet(Alice); exit$

The mental process of an HoA agent can order its set of intentions, according to some preferences of the assisted user. For instance, the intention set related to Alice $I_A = \{meeting(Bob, \ell_1), asking(Bob, get\_copies(\ell_2))\}$ can be ordered such that $weight(meeting(Bob, \ell_1)) < weight(asking(Bob, get\_copies(\ell_2))$. The corresponding agent plan expression of Alice is: $\overline{P_A} = Bob!(get\_copies(\ell_2)); exit \gg meet(Bob); exit$, which is built by using the *options* and *plan* mappings. Pay attention that some actions can be processed concurrently, so is the case in the agent plan $\overline{P_B}$, for the intention plans $get\_copies(\ell_2); exit$ and $move(\ell_1); meet(Alice); exit$.

3 Contextual Planning System

We show now how to build the *Contextual Planning System*, denoted *CPS* for short, from the specification of an agent plan. It is a transition system representing all the possible evolutions of the plan. The building of these last ones are formally driven by a semantics of AgLOTOS constrained by contextual information. As a service instance that can be defined at the planning process level, a guidance mechanism is defined, that works over the evolutions represented by the CPS.

Building of the Contextual Planning System. The AgLOTOS operational semantics is basically derived from the one of LOTOS. A pair (E, P) represents a process identified by P, such that its behavior expression is E. Basic LOTOS semantics is detailed in [7] which formalizes how a process can evolve under the execution of actions. In particular, the rule $\frac{P:=E \quad E \xrightarrow{a} E'}{P \xrightarrow{a} E'}$, specifies how (E, P) pair is changed to (E', P) under any action a. Actually, $P := E$ means to consider any (E, P) source pair and $P \xrightarrow{a} E'$ means changing E to E' for P under the execution of a. As far as AgLOTOS is concerned, these rules also represent the operational semantics of elementary plans, viewed as processes.

The next definition specifies how the expression of an *agent plan* is formed compositionally from the expressions of the *intentions plans* of the agent, themselves built from an alternate of *elementary plans* and their behavior expressions. With respect to some agent plan $\overline{P}$, we introduce a notion of configuration of plans in order to specify that a part of the plan can already be executed. Further, the notation $[\overline{P}]$ represents the configuration of the agent plan $\overline{P}$, it is an AgLOTOS expression, which is obtained by composition of the different intention plan configurations of the agent, like $(E, \widehat{P})$.

Definition 1. *Any agent plan configuration* $[\overline{P}]$ *has a generic representation defined by the following two rules:*

1. $$\frac{\overline{P}::=\widehat{P} \qquad \widehat{P}::=\Diamond^{k=1..n}\, P_k \qquad P_k::=E_k}{[\overline{P}]::=(\Diamond^{k=1..n}\, E_k,\ \widehat{P})}$$
2. $$\frac{\overline{P}::=\overline{P_1}\ \odot\ \overline{P_2} \qquad \odot\in\{|||,\gg\}}{[\overline{P}]::=[\overline{P_1}]\ \odot\ [\overline{P_2}]}$$

The planning state of the agent is now defined contextually, taking into account the agent location and the termination information about the different intention plans defined for the agent.

Definition 2. *A (contextual) planning state is a tuple* $(\mathcal{C}, \ell, T)$*, where* $\mathcal{C}$ *is an agent plan configuration* $[\overline{P}]$*,* ℓ *corresponds to an expected location for the agent, and* T *is the subset of intention plans which will be terminated in this state.*

Table 1. Semantic rules of intention and agent plan configurations

Intention plan level		
(Action)	$\dfrac{E \xrightarrow{a} E' \quad a\in\mathcal{O}\cup\{\tau\}}{(E,\widehat{P}) \xrightarrow{a} (E',\widehat{P})}$	$\dfrac{E \xrightarrow{\delta} E'}{(E,\widehat{P}) \xrightarrow[\widehat{P}]{\tau} (E',\widehat{P})}$
Agent plan level		
(Action)	$\dfrac{\mathcal{C} \xrightarrow{a} \mathcal{C}' \quad a\in\mathcal{O}\cup\{\tau\}}{(\mathcal{C},\ell,T) \xrightarrow{a} (\mathcal{C}',\ell,T)}$	$\dfrac{\mathcal{C} \xrightarrow[\widehat{P}]{\tau} \mathcal{C}'}{(\mathcal{C},\ell,T) \xrightarrow{\tau} (\mathcal{C}',\ell,T\cup\{\widehat{P}\})}$
(Communication)	$\dfrac{\mathcal{C} \xrightarrow{x!(\nu)} \mathcal{C}' \quad x\in\Lambda}{(\mathcal{C},\ell,T) \xrightarrow{x!(\nu)} (\mathcal{C}',\ell,T)}$	$\dfrac{\mathcal{C} \xrightarrow{x?(\nu)} \mathcal{C}' \quad x\in\Lambda}{(\mathcal{C},\ell,T) \xrightarrow{x?(\nu)} (\mathcal{C}',\ell,T)}$
(Mobility)	$\dfrac{\mathcal{C} \xrightarrow{move(\ell')} \mathcal{C}' \quad \ell\neq\ell'}{(\mathcal{C},\ell,T) \xrightarrow{move(\ell')} (\mathcal{C}',\ell',T)}$	$\dfrac{\mathcal{C} \xrightarrow{move(\ell)} \mathcal{C}'}{(\mathcal{C},\ell,T) \xrightarrow{\tau} (\mathcal{C}',\ell,T)}$
(Sequence)	$\dfrac{\mathcal{C}_1 \xrightarrow{a} \mathcal{C}_1' \quad a\in\mathcal{O}\cup\{\tau\}}{\mathcal{C}_1\gg\mathcal{C}_2 \xrightarrow{a} \mathcal{C}_1'\gg\mathcal{C}_2}$	$\dfrac{\mathcal{C}_1 \xrightarrow[\widehat{P}]{\tau} \mathcal{C}_1'}{\mathcal{C}_1\gg\mathcal{C}_2 \xrightarrow[\widehat{P}]{\tau} \mathcal{C}_1'\gg\mathcal{C}_2}$
(Parallel)	$\dfrac{\mathcal{C}_1 \xrightarrow{a} \mathcal{C}_1' \quad a\in\mathcal{O}\cup\{\tau\}}{\mathcal{C}_1\|\|\|\mathcal{C}_2 \xrightarrow{a} \mathcal{C}_1'\|\|\|\mathcal{C}_2}$ $\dfrac{\mathcal{C}_1 \xrightarrow{a} \mathcal{C}_1' \quad a\in\mathcal{O}\cup\{\tau\}}{\mathcal{C}_2\|\|\|\mathcal{C}_1 \xrightarrow{a} \mathcal{C}_2\|\|\|\mathcal{C}_1'}$	$\dfrac{\mathcal{C}_1 \xrightarrow[\widehat{P}]{\tau} \mathcal{C}_1'}{\mathcal{C}_1\|\|\|\mathcal{C}_2 \xrightarrow[\widehat{P}]{\tau} \mathcal{C}_1'\|\|\|\mathcal{C}_2}$ $\dfrac{\mathcal{C}_1 \xrightarrow[\widehat{P}]{\tau} \mathcal{C}_1'}{\mathcal{C}_2\|\|\|\mathcal{C}_1 \xrightarrow[\widehat{P}]{\tau} \mathcal{C}_2\|\|\|\mathcal{C}_1'}$

Table 1 shows the operational semantic rules defining the possible planning state changes for the agent. These rules are applied to produce the CPS, from an initial planning state, e.g. $([\overline{P}], \ell, \emptyset)$, meaning that the agent is initially at

location ℓ, and its plan configuration is $[\overline{P}]$. There are two kinds of transition rules:

Intention plan level: When an intention plan is assumed to be treated, the left hand side transition $(\mathcal{C}_1, a, \widehat{P}, \mathcal{C}_2)$, denoted $\mathcal{C}_1 \xrightarrow[\widehat{P}]{a} \mathcal{C}_2$, expresses a change of intention plan configuration, from $\mathcal{C}_1$ to $\mathcal{C}_2$, and assumes the execution of the action a from $E \xrightarrow{a} E'$ and $P := E$. The right hand side transition highlights the termination case, keeping trace of the intention plan $\widehat{P}$ that is going to be terminated. By calling $\mathcal{CN}$ the set of all the possible intention plan configurations for the agent, the transition relation is a subset of $\mathcal{CN} \times \mathcal{O} \cup \{\tau\} \times \widehat{\mathcal{P}} \times \mathcal{CN}$. For sake of clarity, the transition $(\mathcal{C}_1, a, nil, \mathcal{C}_2)$ is simply denoted $\mathcal{C}_1 \xrightarrow{a} \mathcal{C}_2$. Observe that due to the fact we consider a predictive guidance in this paper, only expected successful executions are taken into account, thus abstracting that a plan may fail. Moreover, the semantics of the alternate operator is reduced to a simple non-deterministic choice of LOTOS: $\Diamond^{k=1..n} E_k \equiv [\ ]^{k=1..n} E_k$, in order to possibly take into account every elementary plan to achieve the corresponding intention.

Agent plan level: the possible changes of the planning states, like $(\mathcal{C}, \ell, T)$, are expressed at this level. In the Communication rules, the action send $x!(\nu)$ (resp. receive $x?(\nu)$) is constrained by the discovery of the agent x in its neighborhood. In the Mobility rule, the effect of the $move(\ell')$ action yields the agent to be placed in ℓ'. The Action rules refer to the ones of the intention plan level. The left hand side one exhibits the case of a regular action, whereas the right hand side one specifies the termination case of some intention plan, which is added to T.

The building of the CPS takes the three following contextual information into account: *(1)* the reached location in a planning state, *(2)* the set of intention plans that are terminated when reaching a planning state, and *(3)* more globally, the set Λ of neighbors currently known by the agent.

Definition 3. *Let I be a set of weighted intentions for the agent. The Contextual Planning System (CPS) is a labeled kripke structure $\langle S, s_0, Tr, \mathcal{L}, \mathcal{T} \rangle$ where:*

- *S is the set of (contextual) planning states,*
- *$s_0 = ([\overline{P}], \ell, \emptyset) \in S$ is the initial planning state of the agent, such that $[\overline{P}]$ is the agent plan configuration of the agent and ℓ represents its current location,*
- *$Tr \subseteq S \times \mathcal{O} \cup \{\tau\} \times S$ is the set of transitions which are denoted $s \xrightarrow{a} s'$,*
- *$\mathcal{L} : S \to \Theta$ is the location labeling function,*
- *$\mathcal{T} : S \to 2^{\widehat{\mathcal{P}}}$ is the termination labeling function which captures the terminated intention plans.*

Application to the scenario. We reconsider the scenario of Section 2. The pairs $(E_m, \widehat{P_m})$ and $(E_g, \widehat{P_g})$ are two intention plan configurations corresponding to Bob. The first one corresponds to the intention $meeting(Alice, \ell_1)$ and the second one to $getting\_copies(\ell_2)$, such that $E_m = move(\ell_1); meet(Alice); exit$ and $E_g = get\_copies(\ell_2); exit$.

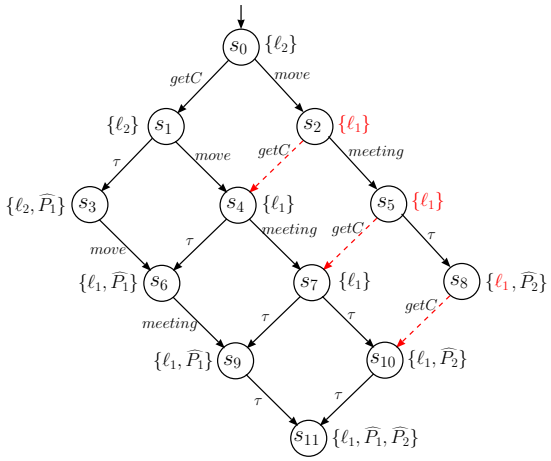

Fig. 3. The CPS_B corresponding to the agent plan $\overline{P_B}$

The CPS corresponding to Bob, denoted CPS_B, is illustrated in Figure 3. It is built from the initial CPS state, $s_0 = ([\overline{P_B}], \ell_2, \emptyset)$, taking into account the current location ℓ_2 of Bob. In the figure, the dashed edges represent the unrealizable transitions from the states $s \in \{s_2, s_5, s_8\}$, because $pre(getC) = \ell_2 \notin \mathcal{L}(s)$.

In a CPS, the transitions from any state s only represent actions that are realizable. Like in STRIPS description language [3], actions to be executed are modeled by instantiated predicates submitted to preconditions and effects. In this paper, the preconditions only concern the contextual information known in that state. Let $pre(a)$ be the precondition of any action a, then $pre(x!(\nu)) = pre(x?(\nu)) = (x \in \Lambda)$ and for any other action a, $pre(a(\ell)) = \ell \in \mathcal{L}(s)$.

Planning Guidance. In order to guide the assisted user, the planning process can select an execution trace through the CPS such that the number of intention plan terminations is maximized, in respect to the mapping $\mathcal{T}$ of the planning states. This can be captured with the notion of Maximum trace, based on a trace mapping $end : \Sigma \longrightarrow 2^{\widehat{\mathcal{P}}}$ used to specify the set $end(\sigma)$ of the termination actions that occur in a trace $\sigma \in \Sigma$. From an algorithmical point of view, the configurations having the maximum number of terminated intention plans could be straightforwardly detected by parsing the CPS structure, with regards to the set of terminated intention plans of each built planning state. By labeling these states with a specific proposition `MAX`, the search of maximum traces is reduced to the traces which satisfies the (LTL) temporal logic property `AF(MAX)`.

Considering again CPS_B corresponding to Bob, an example of maximum trace derived from s_0 is the following, expressing that Bob should get the copies before moving to the meeting with Alice:

$((E_g, \widehat{P_g})|||(E_m, \widehat{P_m}), \ell_2, \emptyset) \xrightarrow{getC} ((E'_g, \widehat{P_g})|||(E_m, \widehat{P_m}), \ell_2, \emptyset) \xrightarrow[\widehat{P_g}]{\tau} ((E_m, \widehat{P_m}), \ell_2, \{\widehat{P_g}\})$

$\xrightarrow{move} ((E'_m, \widehat{P_m}), \ell_1, \{\widehat{P_g}\}) \xrightarrow{meet} ((E''_m, \widehat{P_m}), \ell_1, \{\widehat{P_g}\}) \xrightarrow[\widehat{P_m}]{\tau} ((stop, \widehat{P_m}), \ell_1, \{\widehat{P_g}, \widehat{P_m}\})$

4 Experimentation: The Smart-Campus Project

We experiment our agent-based approach in a distributed system project called Smart-Campus. Our aim is to design a powerful system that assists users in their activities within a complex university campus to better interact and adapt to users' needs and demands. This project is in progress but we concretely equip a float of *Android Smart-Devices*[3] (SD) by the smart-campus application. In this application, the software architecture is composed of an HoA agent and a specific graphical user interface (GUI) to interact with the user to be assisted. Hence, this allows us an explicit presentation of the reasoning of an agent and the concrete use of the guidance service driven by the mental process, according to the change of context process information. From a smart-campus architecture, we now scheme the deployment of the smart-campus application in the SD, and the way to develop the HoA agent main processes.

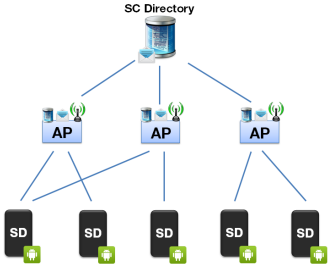

Fig. 4. Smart-campus architecture

Smart-Campus Architecture. The campus system is concretized by the smart-campus starting service which automatically runs the smart-campus application and connects the SD to the `"CAMPUS"` network, through one of the possible WiFi Access Points (AP). As illustrated in Figure 4, the SD can automatically access to the server *"SC Directory"* which is viewed as a middleware maintaining the persistence of contextual information like the discovery and the locations of other users (through their SD) and objects concerning the campus. The starting service is also dedicated to declare the public information of the user to the server, in particular its location. One of the specificity of this project is that the HoA agent embedded in the SD remains autonomous when the SC directory cannot be reached or when the user is exiting the campus. It can continue assisting the user, due to the context information and persistent data previously stored in the SD, can be pervasively updated with the help of other neighbor agents.

Context Process. The context process is based on services currently implemented over the smart-campus architecture, based on physical localization and (a)synchronous communication mechanisms. They are supported by the smart-device API facilities, in particular the WiFi API. As an example, the navigation

[3] Devices: Google Nexus 5, 7 − Android 4.4 KitKat (API level 19).

service takes profit from the underlyied localization service to determine on the fly, the position and the move of the assisted user.

Observe that the localization service must work over the campus ground as well as the different stairs of the buildings. The best localization indoor technique is currently a research in progress e.g. [8]. Currently, we use different WiFi access points within the campus to compute the geographical locations, since this works in both indoor and outdoor locations. Anyway, the localization process requires a tune calibration phase to store specific information in the SC directory, concerning a set of physical reference points that must be selected over the campus, as mentioned in the fingerprinting approach.

In our case, information includes the physical location of the reference point (GPS), its symbolic name (place/room/corridor) and above all the perceived signal attenuation (RSSI[4]) from that location, in respect to the different WiFi access points. The localization service on the SD can then compare its proper perceptions of the WiFi attenuation in respect to the same references stored in the SC directory, so that to deduce an approximation of its position through statistical computations and trilateration concepts.

Mental and Planning Processes To interact with the assisted user, the GUI is an important issue of our application. Figure 5 brings out an instance of three relevant screenshots of the developed GUI. Bob is here the assisted user, being notified on his SD in real time, of the evolution of its intentions, its current location and the (best) direction to meet Alice.

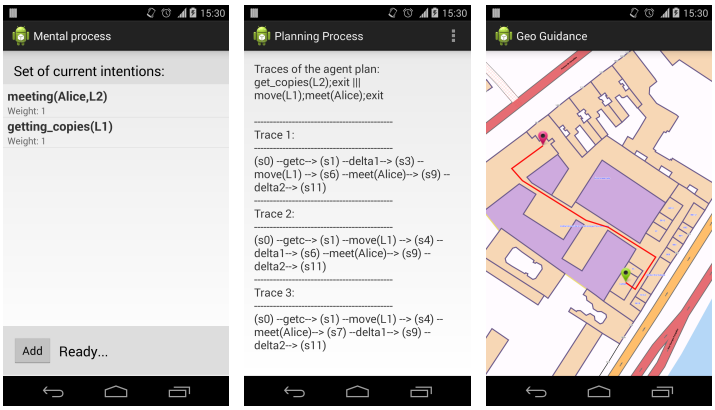

Fig. 5. Smart-campus scenario

- The first one (left hand side) shows the current weighted intentions managed by the mental process, coming from the assisted user desires or the pervasive activity of the HoA agent.
- The second screen is a debug view showing the agent plan and all the possible CPS traces. The contextual guidance service allows the agent to assist

[4] RSSI: Received Signal Strength Indication

the user in realizing his desires in proposing different alternatives of plans, optionally inducing the proposition of spatial paths.
– The last screen (right hand side) highlights the used navigation interface showing a global view of the campus during the execution of the Bob's scenario. As a specific GUI, graphical maps are modern and useful interfaces for the users. The application is able to manage the maps of the campus, over which additional layers are used to render maps interactive and to show different locations and paths.

5 Conclusion

The algebraic language AgLOTOS appears to be a powerful way to express an AmI agent plan as a set of concurrent processes, helped by an adapted plan library describing elementary plans. The proposed operational semantics of AgLOTOS allows one to build a Contextual Planning System (CPS), for any BDI state of the agent.

In respect to the current set of the agent intentions, the CPS structure allows to evaluate all the possible plans. Despite the concurrent execution of plans, the predictive mechanism we propose, allows to guide the agent contextually, over its next possible executions. The problem to search an optimal solution maximizing the number of (sub) plans to be executed, is reduced to a reachability problem over the CPS structure.

In the smart campus project, the presented formal predictive technique is applied to assist users in their daily activities, based on basic contextual information corresponding to spatial location and dynamic neighborhood. Hence, it can also be viewed as a concrete spatial guidance over the campus.

References

[1] Olaru, A., Florea, A.M., El Fallah Seghrouchni, A.: A context-aware multi-agent system as a middleware for ambient intelligence. MONET **18**(3) (2013) 429–443
[2] Sardina, S., de Silva, L., Padgham, L.: Hierarchical planning in BDI agent programming languages: a formal approach. In: AAMAS '06. (2006) 1001–1008
[3] Meneguzzi, F., Zorzo, A.F., da Costa Móra, M., Luck, M.: Incorporating planning into BDI agents. Scalable Computing: Practice and Experience **8** (2007) 15–28
[4] Chaouche, A.C., El Fallah Seghrouchni, A., Ilié, J.M., Saïdouni, D.E.: A dynamical plan revising for ambient systems. Procedia Computer Science **32** (2014) 37 – 44
[5] Rao, A.S., Georgeff, M.P.: An abstract architecture for rational agents. In Nebel, B., Rich, C., Swartout, W.R., eds.: KR, Morgan Kaufmann (1992) 439–449
[6] Brinksma, E., ed.: ISO 8807, LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. (1988)
[7] Chaouche, A.C., El Fallah Seghrouchni, A., Ilié, J.M., Saïdouni, D.E.: A Higher-order Agent model for ambient systems. Procedia Computer Science **21** (2013) 156 – 163
[8] Galvn-Tejada, C.E., Garca-Vzquez, J.P., Garca-Ceja, E., Carrasco-Jimnez, J.C., Brena, R.F.: Evaluation of four classifiers as cost function for indoor location systems. Procedia Computer Science **32**(0) (2014) 453 – 460