Resource-dependent contextual planning in AmI

Arthur Casals, Amal El Fallah-Seghrouchni, Anarosa Alves Franco Brandão, Carlos Eduardo Pantoja, Jose Viterbo

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 151 (2019) 485–492

Procedia Computer Science

www.elsevier.com/locate/procedia

The 10th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 29 - May 2, 2019, Leuven, Belgium

Resource-dependent contextual planning in AmI

Arthur Casals[a,b,*], Amal El Fallah-Seghrouchni[a], Anarosa Alves Franco Brandão[b], Carlos Eduardo Pantoja[c,d], José Viterbo[c]

[a] *Sorbonne Université, Paris, France*
[b] *Universidade de São Paulo, São Paulo, Brazil*
[c] *Universidade Federal Fluminense, Rio de Janeiro, Brazil*
[d] *CEFET/RJ, Rio de Janeiro, Brazil*

Abstract

When we consider ambient intelligence (AmI) environments, contextual planning for multiple agents requires efficient management of resources in order to deploy multi-agent plans. We propose a contextual planning framework to be used by agents in such environments. The proposed framework covers the entire cycle from modeling, multi-agent contextual planning, and deployment of generated plans while ensuring concurrent access and sharing of physical resources. For efficient and effective management of resources, the framework includes a resource management layer. This layer works as a service that provides the virtualization and registration of environments and devices informing the resources availability. The added value of this framework resides in formalizing resource-dependent contextual states in order to allow existing physical resources to be properly perceived as part of the context. Consequently, it allows for plan feasibility to be verified according to any eventual resources requirements related to the deployed plans. The paper goes on to present a proof of concept implementing a scenario based on real-world conditions. In that, we extrapolate the internal execution and monitoring mechanisms that should be required in the case of the framework practical application.

Keywords: Contextual planning; Ambient Intelligence; AmI; BDI

1. Introduction

Ambient Intelligence (AmI) refers to electronic environments in which systems (or electronic devices) can not only communicate with each other but also perceive and respond to the presence of people and other devices or systems [1]. Information representing the environment can be referred to as contextual information [2]. Systems that are capable of capturing contextual information and use it to adapt their functions accordingly are called context-aware systems[13].

* Corresponding author.
E-mail address: arthur.casals@lip6.fr

1877-0509
10.1016/j.procs.2019.04.066

Adopting Multiagent Systems (MAS) to deal with contextual information in AmI applications is of particular interest when the agents follow the BDI (Belief-Desire-Intention) architecture [24], since they can use contextual information in the form of *beliefs*. This information can be used in their planning process to adapt their actions to dynamic environmental changes. In this case, we refer to the planning process as *contextual planning*. The planning process used by a BDI agent [24] can be described as choosing an ordered set of actions to be executed over the environment. Environmental conditions may also include physical resources. Agents deployed in such environments may require resources such as sensors or smart devices to implement *smart applications* [8]. Such resources, on the other hand, may possess various features like capacity or durability. In this paper, we will consider an AmI environment as a physical space (such as a room) with a limited number of physical devices that can be used by agents (thermostats, light switches, cameras, etc.).

The objective of this paper is to propose a framework that allows contextual planning for multiple agents in AmI environments. In order to achieve this objective, we present an overview of contextual planning structures and mechanisms that are used to model the proposed framework. We also present the formalism involved in the planning structures in order to illustrate how physical resources can be used in the contextual planning process. This formalism will also be used in the framework implementation in order to allow the determination of optimal executable plans that depend on physical resources. In order to illustrate the proposed framework, we will consider scenarios where abstractions of AmI environments can be used by virtual agents.

Multiple agents possessing individual planning mechanisms could pose a challenge in terms of concurrent access to resources and their sharing. For this reason, we model the planning mechanism as a stand-alone structure, allowing it to be completely detached from the agent and used in conjunction with a resource provider layer. We present the planning and resource layers as part of a resource-capable planning framework. The objective of this structure is to allow the use of the planning mechanism as a service by multiple agents while using a resource management layer in order to abstract any physical resource existing in the environment and allow their shared use in a controlled manner.

This paper is organized as follows: Section 2 discusses some of the previous work related to our research. Section 3 presents the necessary formalism and describes the model structure of the resource-capable planning framework. An implementation of the proposed framework is shown in Section 4, along with a scenario execution for validation purposes. Finally, Section 5 concludes this paper and presents some perspectives for future work.

2. Related work

Different aspects related to agent planning were studied in previous work [15, 4, 20]. Nevertheless, several aspects related to MAS planning and the use of environmental information are still being researched [17], and there is still a lack of solutions to certain problems, including cooperative planning mechanisms sensible to environmental changes.

Extensive research has also been conducted in the field of intelligent agent cooperative planning [7, 10, 19]. However, most of the existing work relies on central coordination or communication mechanisms, which can have a negative impact on AmI scenarios where agents possess individual goals and need to cooperate dynamically. A planning mechanism was proposed in [5] to allow cooperation between multiple intelligent agents without the need for previous commitment. This mechanism, however, is part of the structure of each individual agent, and it requires that each agent had previous knowledge about other agents present in the environment.

Using a MAS approach with physical resources has also been proposed by various authors in domains such as traffic management, city resilience [14, 18], autonomous vehicles [3], and smart cities [23]. In most cases, however, there was a direct coupling between the agents and the physical resources, resulting in specialized agents with deep knowledge about the accessed hardware.

Abstracting the devices accessed by an agent can be achieved through the use of artifacts [25], which solves the problem of requiring specialized knowledge and other hardware immediate related problems such as access concurrency. While the abstraction problem is solved, however, the same is not necessarily valid for scalability. The number of physical resources in AmI environments can drastically grow, depending on the scenario involved. A solution for that was proposed in [9] as a middleware that deals with communication and connectivity of multiple devices in high-scalable scenarios [22] and has been used in several domains, such as vehicle tracking applications [28], mobile things [27], and agent-based autonomous devices [21].

3. Collective Contextual Planning Framework

This section describes a resource-capable contextual planning framework that can be used by multiple intelligent agents to obtain the optimal plans among the ones available. It considers the feasibility of each plan according to resource availability. Before detailing the proposed framework, we provide the background of some theoretical concepts involved, as well as the necessary formalization.

3.1. Contextual Planning

The planning process of an agent provides a set of ordered actions to be executed. In the case of BDI agents, this process also involves considering the existent information in the environment (perceptions), which can affect which and how the actions will be chosen to be executed [24]. Triggering the planning process according to environment changes allows the agent to re-evaluate its goals and adapt its plans accordingly, so it can adopt the best course of action at that moment.

Considering the original BDI agent model, the agent possesses an interpreter to manage the states related to its beliefs (B), desires (D), and intentions (I) in order to achieve goals through planning. Let us consider an agent A_i. Chaouche *et al* [6] proposed a model to represent the plans associated with the multiple intentions of A_i. It proposes a representation of an agent plan $\tilde{P}(A_i)$ as a tree structure composed of a set of intention plans $\{\hat{P}_{i,j} : j = 1, \ldots, m\}$ and elementary plans $\{P_{j,k} : j = 1, \ldots m, k = 1, \ldots, n\}$. Each intention plan corresponds to the achievement of an agent's intention, and it can be an alternate of multiple sequence of elementary plans: $\hat{P}_{i,j} = \{P_{j,0}, \ldots, P_{j,n}\}$. This alternate composition relationship is captured by using the operator $\diamondsuit$. The $\odot$ operator will be described in detail in section 3.2. Alternating elementary plans allows achieving a specific intention in different manners, each one expressed by a sequence of elementary plans.

Elementary plans $P_{j,k}$ are described by behavior expressions $E_{j,k}$ of LOTOS [16], composed by an ordered finite set of observable actions a to be executed by the agent. Any behavior expression is associated with contextual information related to the (current) BDI state of an agent. Alternating elementary plans allows the planning process to consider different ways to achieve their associated intentions.

Based on the agent plan model presented above, a predictive contextual planning mechanism called CPS was proposed [6]. This mechanism was presented as a planning process method to be used by an agent to select optimal (and feasible) plans among the ones available. This process requires (i) analyzing each plan according to the previously defined structure and (i) reviewing any restrictions that might be associated with each of the actions to be performed by their respective plans. As a result, the formal structure obtained from the CPS (henceforth referred to as "CPS structure") represents all the potential paths that can be taken by the agent given the current contextual restrictions. The planning mechanism then selects the best possible plan to be executed by the agent given the current context conditions.

Using the CPS structure, a mechanism called Collective Contextual Planning System (CCPS) was proposed [5]. This structure was designed to process plans from multiple CPS-capable agents. CCPS allows agents to ask for help or delegate part of a plan to other agents whenever it cannot execute the plan properly, avoiding re-planning and maintaining the consistency of its goals. At the same time, all the parallelism and concurrence taken into consideration by the original CPS mechanism are maintained. The idea behind this collaboration mechanism is similar to the contract net protocol [26], designed for distributed problem-solving. Nevertheless, the CCPS mechanism is based on the premise that all agents are CPS-capable: each of them must possess a predictive planning mechanism using CPS structures for cooperation to be established.

3.2. Formalization

Considering the nature of available resources in AmI environments, we propose a formalization to bind the environment resources to elementary plans. Resources can have various characteristics such as durability, capacity, or duration: e.g. a bridge can be open for a period of two hours during the day (duration), in which time a maximum amount of 10 cars can cross it (capacity). The plan configuration itself is not affected by the use of resources as long as the semantic rules for an agent plan $\tilde{P}(A_i)$ (henceforth referred to as $\tilde{P}$) are compliant with the semantic structure of the behavior expressions. As a consequence, the semantic rules at the intention plan level are also not affected. The

use of resources, however, can have an impact on the AmI environment. With that in mind, we propose the following formalism:

Environment: $Env(t)$ is a set of logical propositions representing contextual information that can be perceived by any agent at time t.

Resource: A resource r represents one of the physical resources present in an AmI environment. Differently from actions, resources do not possess sets of logical propositions. Instead, they possess a finite set of properties $Prop(r) = \{p_1, ..., p_n\}$ consisting of tuples $p_i = (k_i, v_i), 1 \leq i \leq n$. Each tuple represents a characteristic specific to that resource, identified by k_i and with an assigned value v_i. These properties refer to characteristics such as capacity or duration. As a consequence, a resource r is said to be *available* in an environment if (i) it exists in the given environment and (ii) all of its properties are satisfied in the same environment. The availability of a resource r is expressed by $avail(r)$.

Action: An action a represents the finest granular activity performed by the agent in the environment. If an agent A_i moves from a location X to a location Y, there is an associated action $move(X, Y)$. Each action a is subject to three sets of logical propositions (as in STRIPS [12]): preconditions list ($Pre(a)$); delete list ($Del(a)$); and post-conditions list ($Post(a)$). An action a is *feasible* if all of its preconditions $Pre(a)$ are satisfied by the conditions of $Env(t)$. Also, an action may need one or more resources in order to be feasible. In this case, the set of resources required by an action a is expressed by $R(a)$. Therefore, the feasibility of the action is also determined by the availability of the resources required. After an action a is executed, its conditions can be changed by $Del(a)$ or $Post(a)$- also altering $Env(t)$.

Plan: As mentioned before, an *agent plan* is composed of multiple *intention plans*, and each *intention plan* can be an alternate of multiple *elementary plans*. An *agent plan* denoted $\tilde{P}(A_i)$ represents the plans related to each goal agent A_i decided to achieve: $\tilde{P}(A_i) = \{\hat{P}_{i,1}, ..., \hat{P}_{i,j}\}$.

Each of these goals was chosen between the agent's intentions, and possesses an associated *intention plan*. Each *intention plan* $\hat{P}_q$ $(1 \leq q \leq j)$ is also composed of multiple elementary plans. Elementary plans, on the other hand, represent alternatives and are composed of a set of ordered actions to be executed by the agent. Therefore, an agent plan $\tilde{P}(A_i)$ can be defined as a set consisting of s partially ordered actions known by an agent A_i: $\tilde{P}(A_i) = \{a_1, a_2, ..., a_s\}$

As a partially ordered set, some actions in a plan must be executed before or in parallel with other actions. The execution of all actions in a plan necessarily requires the availability of all resources needed by each of the actions.

Plan Feasibility: Being a composite and partially ordered set of actions, the feasibility of an elementary plan P can be given by the feasibility of its actions: P is feasible in $Env(t)$ if and only if (i) all of its actions preconditions are satisfied by the conditions of $Env(t)$, and (ii) all of the resources needed by its actions are available in $Env(t)$. Therefore, an intention plan $\hat{P}$ is feasible if and only if at least one of its elementary plans is feasible. In the case of an agent plan $\tilde{P}$, we determine that it is completely feasible if and only if all of its intention plans are feasible.

In order to meet the AmI requirements we need that agents: *(1)* perceive whenever other agents enter or leave the AmI environment; *(2)* communicate with each other while executing in the AmI environment; and *(3)* use resources available within the AmI environment. Therefore, AmI primitives for mobility and resources need are expressed as $move(X, Y)$ and $needs(r)$, respectively. X and Y are locations defined by a finite set of space localities Θ $(X, Y \in \Theta)$, and r belongs to a finite set of environment resources R $(r \in R)$.

3.3. Resource-capable Contextual Planning Framework

The proposed framework is intended to be used by multiple intelligent agents deployed into AmI environments. We focused our efforts on understanding how existing resources could be used by a shared planning mechanism. Studying the different agent architectures deployed into the environment is not part of the scope of this work. We focus solely on contextual planning and resource management capabilities, abstracting how the contextual information is perceived. We divided our considerations into (i) structuring a shared planning mechanism and (ii) abstracting and seamlessly exposing resources within an AmI environment to multiple agents.

In the AmI domain, plan feasibility is bound to contextual information. In this situation, a planning mechanism should be able to (i) identify the environment constraints attached to each of the agent's plans (including resource availability); and (ii) according to the contextual information available, decide which of the agent's plans can be executed in an optimal manner. Optimal plans according to environmental conditions can be identified by using a contextual planning mechanism such as the previously mentioned CCPS. This mechanism, however, was intended to be used internally by an intelligent agent, attached to its implementation. A shared planning mechanism requires a shared structure, not attached to any of the agents, and that can process plans from multiple agents using the same

contextual information. This structure should also possess a minimal coordination mechanism in order to avoid any conflicts when simultaneously dealing with requests from multiple agents.

Taking these considerations into account, our process of contextually identifying optimal plans was based on the CCPS mechanism. At the same time, we introduce the concept of shared resources into the available contextual information since a plan may require a set of resources in order to be executed. These resources, however, can be dynamically inserted or removed from the environment, and characteristics such as capacity and durability should also be taken into account.

With these considerations in mind, we propose a resource-capable contextual planning framework model. The framework is primarily composed of two loosely-coupled service layers: the Contextual Planning Layer (CPL), and the Resource Management Layer (RML). The necessary contextual information is modeled as a set of environmental conditions, which include a basic description of the available resources (shared among all layers). The CPL is responsible for receiving any plans that the agents intend to execute and check which of them are feasible. The RML, on the other hand, is responsible for managing the AmI environments and interacting with its resources when necessary. In order to do so, it uses AmI environment abstractions, which include characteristics of the environments (such as capacity) and their resources. We will refer to these abstractions simply as "environments" when detailing the framework model.

Multiple intelligent agents can access the CPL and provide their sets of plans, defined according to their individual intentions. Once the CPL receives a set of plans from an agent, it first verifies which of the composing plans are feasible according to the current context. This preliminary feasibility is determined only by the pre-conditions of each of the actions. After that, a second verification takes place in order to verify if any of these plans require physical resources.

In the case resources are required, they are grouped into sets (one for each elementary plan), which are then sent to the RML to be checked for availability. Each resource set is then checked against the available environments for compatibility (i.e., available environments that contain the resource sets required). The RML locks one compatible environment for each of the resource sets and answers the CPL with the identification of the resource sets that can be allocated.

According to the resources availability, the set of feasible plans is *filtered* and then sent back to the agent. In the possession of this information, the agent can then choose which plans will be executed. Any chosen plans are informed to the CPL, which then asks the RML to allocate the necessary resources. At this point, any locked environments that will not be used by the agent are also unlocked by the RML.

The service layers that compose the framework will be detailed in the following sections, as well as their composing modules and related algorithms.

3.4. Contextual Planning Layer

The CPL acts as an access layer for the agents, and it is responsible for (i) dealing with plan feasibility checks (sent by the agents); (ii) accessing the Resource Management Layer (RML) in order to verify the availability of any required resources; and (iii) determining which of the plans are unfeasible, according to the environmental conditions (including resource availability). It is also responsible for interacting with the RML to identify and allocate any resources ultimately required by the agents, once they inform which of the feasible plans were chosen for execution. From the CPL perspective, the process for checking an agent's plan for feasibility starts with an agent providing the CPL with a set of intention plans to be verified for feasibility, considering the environmental conditions and eventual resources availability. Plan feasibility is verified through the verification of feasibility for each action in the elementary plans. If any of the feasible plans depends on resources, the CPL transforms the resources description into a set of identified resources.

These resources are stored into a local database (resource store) for future reference and then sent to the RML. Each identified resource set represents the subset of resources required for a single elementary plan. The RML returns the resource ID tags for every resource where all resources are available. If even one of the resources described in the identified resource set is not available, its ID will not be sent by the RML. All resource ID tags received are sent to the resource store, so any feasible plans depending on unavailable resources can be filtered before being sent to the agent.

The resulting set of feasible plans is sent back to the agent, which can then choose and inform the CPL which of them will be executed. The CPL checks in its local database for which of the plans informed by the agent require any

of the previously mapped resources. If any resources are required, the previously mapped correspondent resource IDs are sent to the RML, which then allocates the resources as required. The CPL then sends a message back to the agent informing about the success of the allocation operation.

3.5. Resource Management Layer

The Resource Management Layer (RML) manages virtualization and registration of physical environments and devices, providing information on their availability and allocating them to an agent when necessary. The devices are dynamically registered in the RML, at which point they inform all of their existing functionalities and characteristics. Once a device is registered, it keeps updating the RML with the information gathered from its sensors. In case of disconnection, the device is unregistered from this layer. RML is built over an instance of ContextNet [9] in order to provide high scalability and connectivity of devices and environments.

The RML structure is composed of two independent modules: (i) the client, running on devices, and (ii) the cloud module, running on a fixed infrastructure. The cloud module is a server-side solution built upon a ContextNet instance, which virtualizes every device connected to it. The cloud module is capable of mapping different environments in real-time, each composed of devices and their capabilities. Mapping resources allow them to be used by agents through the CPL requisitions.

It is important to observe that devices perform a dynamic configuration and update process when they connect with the cloud module for the first time. At this point, a connected device sends information about all its available functionalities (including all sensing and actuators' capabilities) and the current environmental conditions (perceived according to its existing sensors). Once connected, the device keeps updating the cloud module with data gathered from the sensors (once per device's cycle). Then, it checks if there are any messages received from the cloud module and put them into a queue of actions to be executed at the end of the cycle. Devices implement a client of the ContextNet middleware to provide the communication between them and the cloud core.

4. Scenarios and Experiments

After defining how to monitor the framework execution, we implemented a proof-of-concept and validated it in conjunction with an application scenario. The implementation was done using Java, and both layers were deployed to the Web as RESTful services [11]. The objective was to observe if the framework would be able to properly deal with requests from multiple agents and multiple resources at the same time. Also, we would like to observe how the plan feasibility mechanism would behave with the addition of the physical resources to the context. For this reason, we will not detail the agent plans used in our experiments. Our scenario contained multiple agents, with multiple plans depending on common resources. We used the presented formalization as a frame of reference in order to determine which would be the behavior in both situations, so we could compare the outcome of the experiments with a theoretical baseline.

The proof-of-concept was evaluated with a simple scenario involving multiple agents. Each agent possessed different plans, and some of them depended on physical resources. We also considered different sets of resources grouped into five different environment abstractions. Each of these environments possessed a limited capacity and a subset of the following resources: thermostat, camera, illumination controls, and projector.

The scenario described above was executed in two different environmental conditions (contexts), considering starting time (enough time to achieve time-sensitive intentions), inherently unachievable intentions, and the presence of available resources. In the first context (1), each agent was given a set of plans that did not contain resource-dependent elementary plans. In this context, we expected that only the plans depending on other constraints than resources would be unfeasible. We used this execution in order to establish a baseline for the next execution

In the second context, resource dependency was added to the plans used in the previous execution. The resources bound to each plan were not necessarily available in the environment. Thus, in this context, we expected that some of the otherwise feasible plans would be marked as unfeasible due to resource constraints. We used six different sets of environment configurations as requirements (Table 1). Each environment configuration was designed to match none, one or multiple of the environment abstractions.

Table 1. Environment configurations

Config.	Capacity	Resources	Matches
1	20	thermostat, light controls	1,4
2	40	temperature	1
3	60	light controls	–
4	20	camera	3
5	20	light controls	1,2,4,5
6	20	temperature, computer	–

Once defined, the environment configurations were bound to the agent plans and executed in order to verify the resource allocation mechanism and its impact on plan feasibility. We divided the test objectives into (i) availability (checking if a given environment set was available upon request), (ii) concurrency (checking if the resource limit was observed, allowing plans depending on equivalent environment configurations to be executed accordingly) and (iii) multiple dependency (checking if plans depending on multiple environments could be fully or partially executed, and how the resource management layer would handle such situations). An example of test cases executed according to these directives is shown in Table 2. Each case is related to a given objective, the configuration sequence used (environments sequentially required by agent plans), and the expected allocation of the available environments. Both configuration sequences and the allocated environments refer to the tables previously shown. We used the operator "+" for multiple elements in the same requisition, as observed in line 3 of Table 2.

Table 2. Tests

	Objective	Config. Sequence	Alloc. Env.
1	Availability	3	–
2	Concurrency	4,4,1,1,1	3,–,1,4,–
3	Multiple dependency	2+4,2+4	1+3,–

5. Conclusion

In this paper we proposed a mechanism to allow collective contextual planning in AmI environments, considering the existence of various physical resources. This involved (i) reusing and adapting an existing process algebra based on LOTOS to represent plans in order to accommodate the concept of physical resources and (ii) modeling and deploying a contextual planning framework that could be simultaneously used by multiple agents. Formalizing resource-dependent behavior expressions was necessary in order to allow existing physical resources to be properly perceived as part of the context, and consequently allowing for plan feasibility to be verified according to any resource requirements related to these plans. Modeling and deploying a contextual planning framework adherent to this formalization was necessary to verify the proposed theoretical structure. It also illustrated how the proposed model can enable the use of resource-dependent contextual planning with other resource management mechanisms.

Our implementation worked as expected. The CPL was based on the original CCPS implementation [5], and we did not find any major difficulties when adapting it. The RML was implemented as a virtual-only representation of physical devices, managed by a simple queueing mechanism. We expect to considerably improve this part of the implementation, increasing the complexity level of both the physical resources and their relationships.

Finally, observing the implementation behavior in a scenario based on real-world conditions was fundamental in order to extrapolate the internal execution and monitoring mechanisms that should be required in the case of a practical application of this work. We expect to study and test these constraints in detail in future research, as well as other practical related topics such as communication protocols and distributed constraints management mechanisms.

References

[1] Aarts, E., Wichert, R., 2009. Ambient intelligence, in: Technology Guide. Springer, pp. 244–249.

[2] Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P., 1999. Towards a better understanding of context and context-awareness, in: International Symposium on Handheld and Ubiquitous Computing, Springer Berlin Heidelberg. pp. 304–307.
[3] Barros, R.S., Heringer, V.H., Lazarin, N.M., Pantoja, C.E., Moraes, L.M., 2014. An agent-oriented ground vehicle's automation using Jason framework, in: th$ International Conference on Agents and Artificial Intelligence, pp. 261–266.
[4] Boukharrou, R., Chaouche, A.C., Seghrouchni, A.E.F., Ilié, J.M., Saïdouni, D.E., 2015. Dealing with temporal failure in ambient systems: a dynamic revision of plans. Journal of Ambient Intelligence and Humanized Computing 6, 325–336.
[5] Casals, A., Belbachir, A., Seghrouchni, A.E.F., Brandão, A.A.F., 2018. Fostering agent cooperation in ami: A context-aware mechanism for dealing with multiple intentions, in: Del Ser, J., Osaba, E., Bilbao, M.N., Sanchez-Medina, J.J., Vecchio, M., Yang, X.S. (Eds.), Intelligent Distributed Computing XII, Springer International Publishing, Cham. pp. 225–234.
[6] Chaouche, A.C., El Fallah-Seghrouchni, A., Ilié, J.M., Saidouni, D.E., 2015. From intentions to plans: A contextual planning guidance, in: Intelligent Distributed Computing VIII. Springer International Publishing, pp. 403–413.
[7] Di Febbraro, A., Sacco, N., Saeednia, M., 2016. An agent-based framework for cooperative planning of intermodal freight transport chains. Transportation Research Part C: Emerging Technologies 64, 72–85.
[8] Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.C., 2001. Scenarios for ambient intelligence in 2010. Office for official publications of the European Communities .
[9] Endler, M., Baptista, G., Silva, L., Vasconcelos, R., Malcher, M., Pantoja, V., Pinheiro, V., Viterbo, J., 2011. ContextNet: context reasoning and sharing middleware for large-scale pervasive collaboration and social networking, in: Proceedings of the Workshop on Posters and Demos Track, ACM. p. 2.
[10] Engesser, T., Bolander, T., Mattmüller, R., Nebel, B., 2017. Cooperative epistemic multi-agent planning for implicit coordination. arXiv preprint arXiv:1703.02196 .
[11] Fielding, R.T., Taylor, R.N., 2002. Principled design of the modern web architecture. ACM Transactions on Internet Technology (TOIT) 2, 115–150.
[12] Fikes, R.E., Nilsson, N.J., 1971. Strips: A new approach to the application of theorem proving to problem solving. Artificial intelligence 2, 189–208.
[13] Hong, J.y., Suh, E.h., Kim, S.J., 2009. Context-aware systems: A literature review and classification. Expert Systems with Applications 36, 8509–8522.
[14] Li, Z., Al Hassan, R., Shahidehpour, M., Bahramirad, S., Khodaei, A., 2017. A hierarchical framework for intelligent traffic management in smart cities. IEEE Transactions on Smart Grid .
[15] Logan, B., Thangarajah, J., Yorke-Smith, N., 2017. Progressing intention progression: A call for a goal-plan tree contest, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017, ACM. pp. 768–772.
[16] Lotos, I., 1989. A formal description technique based on the temporal ordering of observational behaviour. ISO8807, 1XS989 .
[17] Meneguzzi, F., De Silva, L., 2015. Planning in bdi agents: a survey of the integration of planning algorithms and agent reasoning. The Knowledge Engineering Review 30, 1–44.
[18] Mustapha, K., Mcheick, H., Mellouli, S., 2016. Smart cities and resilience plans: a multi-agent based simulation for extreme event rescuing, in: Smarter as the New Urban Agenda. Springer, pp. 149–170.
[19] Nigon, J., Gleizes, M.P., Migeon, F., 2016. Self-adaptive model generation for ambient systems. Procedia Computer Science 83, 675–679.
[20] Nunes, I., Schardong, F., Filho, A.E.S., 2017. Bdi2dos: An application using collaborating BDI agents to combat ddos attacks. J. Network and Computer Applications 84, 14–24. URL: https://doi.org/10.1016/j.jnca.2017.01.035, doi:10.1016/j.jnca.2017.01.035.
[21] Pantoja, C.E., Soares, H.D., Viterbo, J., Fallah-Seghrouchni, A.E., 2018. An architecture for the development of ambient intelligence systems managed by embedded agents, in: The 30th International Conference on Software Engineering and Knowledge Engineering, Hotel Pullman, Redwood City, California, USA, July 1-3, 2018., pp. 215–214.
[22] Pardo-Castellote, G., 2003. Omg data-distribution service: Architectural overview, in: Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on, IEEE. pp. 200–206.
[23] Pribyl, O., Svitek, M., 2015. System-oriented approach to smart cities, in: Smart Cities Conference (ISC2), 2015 IEEE First International, IEEE. pp. 1–8.
[24] Rao, A.S., Georgeff, M.P., et al., 1995. Bdi agents: from theory to practice., in: ICMAS, pp. 312–319.
[25] Ricci, A., Piunti, M., Viroli, M., Omicini, A., 2009. Environment programming in CArtAgO, in: Seghrouchni, A., Dix, J., Dastani, M., Bordini, H.R. (Eds.), Multi-Agent Programming: Languages, Tools and Applications. Springer US, Boston, MA, pp. 259–288. doi:10.1007/978-0-387-89299-3_8.
[26] Smith, R.G., 1980. The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on computers , 1104–1113.
[27] Talavera, L.E., Endler, M., Vasconcelos, I., Vasconcelos, R., Cunha, M., e Silva, F.J.d.S., 2015. The mobile hub concept: Enabling applications for the internet of mobile things, in: Pervasive computing and communication workshops (PerCom workshops), 2015 IEEE international conference on, IEEE. pp. 123–128.
[28] Vasconcelos, I., Vasconcelos, R., Baptista, G., Seguin, C., Endler, M., 2013. Desenvolvendo aplicações de rastreamento e comunicação móvel usando o middleware sddl, in: Salão de Ferramentas, Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2013).