A Reasoning Framework for Ambient Intelligence *

Theodore Patkos, Ioannis Chrysakis, Antonis Bikakis, Dimitris Plexousakis, and Grigoris Antoniou

Institute of Computer Science, FO.R.T.H.
{patkos, hrysakis, bikakis, dp, antoniou}@ics.forth.gr

Abstract. Ambient Intelligence is an emerging discipline that requires the integration of expertise from a multitude of scientific fields. The role of Artificial Intelligence is crucial not only for bringing intelligence to everyday environments, but also for providing the means for the different disciplines to collaborate. In this paper we describe the design of a reasoning framework, applied to an operational Ambient Intelligence infrastructure, that combines rule-based reasoning with reasoning about actions and causality on top of ontology-based context models. The emphasis is on identifying the limitations of the rule-based approach and the way action theories can be employed to fill the gaps.

1 Introduction

The Ambient Intelligence (AmI) paradigm has generated an enabling multidisciplinary research field that envisages to bring intelligence to everyday environments and facilitate human interaction with devices and the surrounding. Artificial Intelligence has a decisive role to play for the realization of this vision promising commonsense reasoning and better decision making in dynamic and highly complex conditions, as advocated by recent studies [1]. Within Ambient Intelligence environments human users do not experience passively the functionalities of a smart space, instead participate actively in it performing actions that change its state in different ways. At the same time, the smart space itself and its devices are expected to perform actions and generate plans either in response to changes in the context or to predict user desires and adapt to user needs.

In this paper we describe the design of a reasoning framework intended for use in an AmI infrastructure that is being implemented in our institute. The framework integrates Semantic Web technologies for representing contextual knowledge with rule-based and causality-based reasoning methodologies for supporting a multitude of general-purpose and domain-specific reasoning tasks imposed by the AmI system. Given that during the first phases of this project we have fully implemented the rule-based features of the reasoner and applied them to practice, the present study concentrates primarily on the limitations identified

* This work has been supported by the FORTH-ICS internal RTD Programme "Ambient Intelligence and Smart Environments".

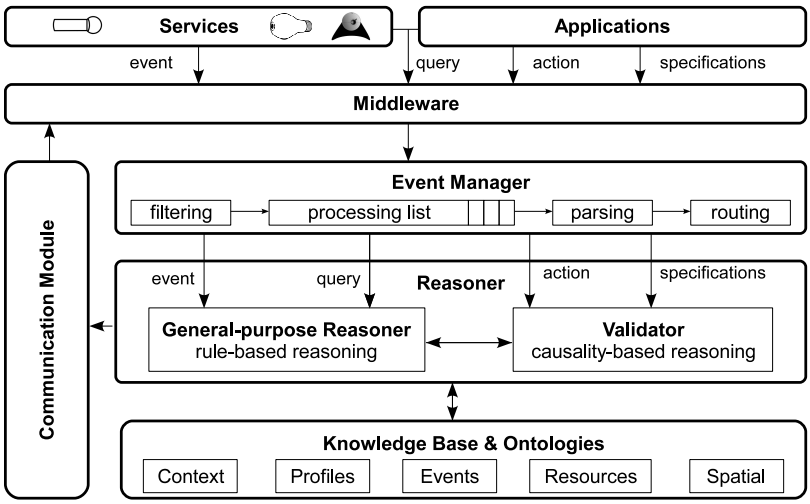

Fig. 1. Event management framework architecture.

concerning certain challenging issues of this new field, and the way action theories can be employed to offer efficient solutions. Action theories, a fundamental field of research within KR&R, are formal tools for performing commonsense reasoning in dynamic domains. In this paper we present how they can contribute to the AmI vision and what types of problems they can resolve. We report our experiences in developing each of the proposed solutions as distinct functionalities, but also describe how we plan to integrate them in the overall framework to provide a powerful hybrid reasoning tool for application in real-world conditions. Our objective is to illustrate the impact of combining logic-based AI methods on addressing a broad range of practical issues.

The rest of the paper is as follows. We first describe the overall architecture of the framework and continue with the tasks assigned to the rule-based reasoning component. Section 4 elaborates on the contribution of causality-based approaches to AmI. The paper concludes with a discussion of related work.

2 Event-based Architecture

The design goals of the reasoning framework have been the efficient representation, monitoring and dissemination of any low- or high-level contextual information in our Ambient Intelligence infrastructure, as well as the support for a number of general-purpose and domain-specific inferencing tasks. For that purpose we deploy the hybrid event-based reasoning architecture shown in Fig. 1, that comprises four main components; the Event Manager that receives and processes incoming events from the ambient infrastructure, the Reasoner that can perform both rule-based and causality-based reasoning, the Knowledge Base that

stores semantic information represented using ontology-based languages, and the Communication Module that forwards Reasoner requests for action execution to appropriate services. A middleware layer undertakes the role of connecting applications and services implemented by different research groups and with different technologies. *Services* denote standalone entities that implement specific functionalities about world aspects, such as voice recognition, localization, light management etc., whereas *applications* group together service instances to provide an Ambient Intelligence experience in smart rooms.

The semantic distinction between *events* and *actions* is essential for coordinating the behavior of AmI systems. Events are generated by services and declare changes in the state of context aspects. In case these aspects can be modified on demand, e.g., $CloseDoor(doorID)$, the allocated service provides also the appropriate interface for performing the change, otherwise the service plays the role of a gateway for monitoring environmental context acquired from sensors or devices. Actions, on the other hand, reflect the desire of an application for an event to occur. In a sense, events express atomic, transient facts that have occurred, while actions can either be requests for atomic events or complex combination of event occurrences according to certain operators that form compound events, i.e., sets of event patterns. It is the responsibility of the reasoner to examine individual actions before allowing their execution and guarantee that the state of the system remains consistent at all times.

Implementation: Our reasoning framework is part of a large-scale AmI facility that is being implemented in FORTH institute and has completed its first year of life. It expands in a three-room set up, where a multitude of different - hardware and software- technologies contribute services, such as camera network support for 2D person localization and 3D head pose estimation, RFID, iris and audio sensors for person identification and speech recognition, multi-protocol wireless communications etc. The middleware responsible for creating, connecting and consuming these services is CORBA-based and provides support for C++, Java, .NET, Python, and Flash/ActionScript languages. The rule-based component of the reasoner module uses Jess[1] as its reasoning engine, while the Validator component uses both Jess and DEC Reasoner[2], a SAT-based Event Calculus reasoner. The former is responsible for run-time action validation, while the latter performs more powerful reasoning tasks, such as specification analysis and planning. The available knowledge has been encoded in OWL ontologies using the Protégé platform for editing and the Protégé-OWL API for browsing and querying the corresponding models.

3 Context Information Modeling and Reasoning

The task of context management in an Ambient Intelligence environment requires an open framework to support seamless interoperability and mutual understanding of the meaning of concepts among different devices and services.

[1] Jess, http://www.jessrules.com/
[2] DECReasoner, http://decreasoner.sourceforge.net/

Ontology-based models are widely considered the most enabling approach for modeling contextual information, satisfying the representation requirements set by many studies in terms of type and level of formality, expressiveness, flexibility and extensibility, generality, granularity and valid context constraining [2, 3]. In our framework we design ontologies that capture the meaning and relations of concepts regarding low-level context acquired from sensors, high-level context inferred through reasoning, user and device profiling information, spatial features and resource characteristics (Fig. 1).

An aspect of the framework that acknowledges the benefit of ontologies is the derivation of high-level context knowledge. Complex context data inferred by means of rule-based reasoning tasks on the basis of raw sensor data or other high-level knowledge, is based on ontology representations and may concern a user's emotional state, identity, intentions, location etc. For instance, the following rule specifies that a user is assumed to have left the main room and entered the warmup room only if she was standing near the open main-room door and is no longer tracked by the localizer, which is installed only in this room:
(user (id ?u) (location DOORMAIN)) $\wedge$ (door (id DOORMAIN) (state OPEN)) $\wedge$ (event (type USERLOST) (user ?u)) $\Rightarrow$ (user (id ?u) (location WARMUP))
Under different context the same event will trigger a different set of rules, capturing for example the case where the user stands behind an obstacle.

Rule-based reasoning is a commonly adopted solution for high-level context inference in AmI [4]. Within our system it contributes to the design of enhanced applications and also provides feedback to services for sensor fusion purposes, in order to resolve conflicts of ambiguous or imprecise context and detect erroneous context. Furthermore, rule-based reasoning is also employed to coordinate the overall system behavior and offer explanations; the operation is partitioned into distinct modes that invoke appropriate rulesets, according to relevant context and the functionality that we wish to implement. Finally, this component provides also procedures for domain-specific reasoning tasks for search and optimization problems driven by application demands, such as for determining the best camera viewpoint to record a user interacting inside a smart room.

4 Causality-based Reasoning in Ambient Intelligence

Event-based architectures offer opportunities for flexible processing of the information flow and knowledge evolution over time. Rule-based languages provide only limited expressiveness to describe certain complex features, such as compound actions, therefore they do not fully exploit the potential of the event-based style to solve challenging event processing tasks raised by ubiquitous computing domains. The Event-Condition-Action (ECA) paradigm that is most frequently applied can be used for reacting instantly to detected events, viewing them as transient atomic instances and consuming them upon detection. They do not consider the durative nature, how far into the past or future their effects extend nor do they investigate causality issues originating from the fact that other events are known to have occurred or are planned to happen. Paschke [5] has

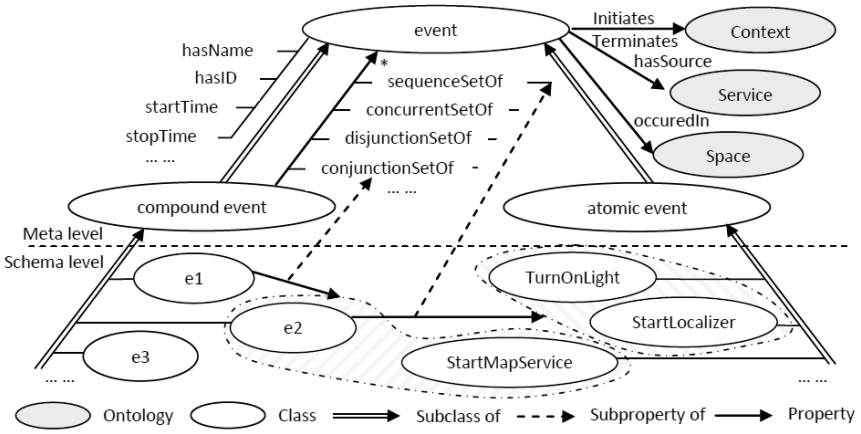

Fig. 2. Event ontology sample. Event operators, such as sequenceSetOf and concurrentSetOf, are implemented as $rdf : Seq$, $rdf : Bag$ and $rdf : Alt$ container elements.

already shown that the ECA treatment may even result in unintended semantics for some compositions of event patterns. However, real-life systems demand formal semantics in order to support verification and traceability purposes.

To this end, we apply techniques for reasoning about actions and causality. Action theories are formal tools, based on the predicate calculus, particularly designed for addressing key issues of reasoning in dynamically changing worlds by axiomatizing the specifications of their dynamics and exploiting logic-based techniques in order to draw conclusions. Different formalisms have been developed that model action preconditions and effects and solve both deduction and abduction problems about a multitude of commonsense reasoning phenomena, such as effect ramifications, non-deterministic actions, qualifications and others. For our purposes we apply the Event Calculus formalism [6, 7], which establishes a linear time structure enabling temporal reasoning in order to infer the intervals in which certain world aspects hold. The notion of time, inherently modeled within the Event Calculus, as opposed to other action theories, is a crucial leverage for event-based systems, e.g. to express partial ordering of events or timestamps.

In the remaining subsections we describe our approach to integrate action theories with our Ambient Intelligence semantic infrastructure and present the types of reasoning problems that have been assigned to them.

4.1 Event Ontology and Complex Actions

For any large-scale event-based system it is important to identify event patterns that describe the structure of complex events built from atomic or other complex event instances. Their definition and processing must follow formal rules with well-defined operators, in order for their meaning to be understood by all system entities. In an Ambient Intelligence infrastructure in particular, this need

is even more critical due to the multidisciplinary nature and the demand for collaborative actions by entities with significantly different backgrounds.

In order to promote a high-level description of the specifications of applications and to achieve a high degree of interoperability among services we design an event ontology to capture the notions of atomic and compound events and define operators among event sets, such as sequence, concurrency, disjunction and conjunction (Fig. 2). An event is further characterized by its initiation and termination occurrence times (for atomic events they coincide), the effect that it causes to context resources, the physical location at which it occurred, the service that detected or triggered it etc. Our intention is to focus only on generic event attributes that satisfy the objectives of our system, based on previous studies that define common top ontologies for events (e.g. [8]), rather than reproduce a complete domain-independent top event ontology, which would be in large part out-of-focus and result in a less scalable and efficient implementation.

To define formal semantics for the operators, we implement them as container elements that collect resources and translate them to Event Calculus axioms. Notice the compound event e_1 in Fig. 2, for instance, that expresses the partially ordered event type $[[TurnOnLight; StartLocalizer] \wedge StartMapService]$ where (;) represents the sequence operator. In order to utilize e_1 for reasoning tasks we axiomatize its temporal properties in Event Calculus:

$$\begin{aligned} Happens(Start(e_1), t) \equiv \\ Happens(TurnOnLight(l), t_1) \wedge Happens(StartLocalizer(), t_2) \wedge \\ Happens(StartMapService(), t_3) \wedge (t_1 < t_2) \wedge (t = min(t_1, t_3)) \end{aligned}$$

(respectively for $Happens(Stop(e_1), t)$), as well as its causal properties:

$$Initiates(Start(e_1), LightOn(l), t) \wedge Terminates(Stop(e_1), TrainingMode(), t)$$

We may formalize the effects of compound events to act cumulatively or canceling the effects of their atomic components and also we can specify whether certain effects hold at beginning times or at ending times. We currently model the duration of compound events in terms of their *Start* and *Stop* times, but also investigate the potentials of other approaches, such as the interval-based Event Calculus [5] or the three-argument *Happens* Event Calculus axiomatization [9].

Central advantages of the Event Calculus, in comparison to rule-based approaches, are its inherent ability to perform temporal reasoning considering both relative and absolute times of event occurrences and that it can reevaluate different variations of event patterns as time progresses. With ECA style reactive rules events are consumed as they are detected and cannot contribute to the detection of other complex events afterwards. Finally, the combination of semantic event representation and causality-based event processing in the framework makes much more convenient the process of describing the specifications of applications for non-AI-expert developers in our system, without undermining the system's reasoning capabilities. As we show next, these descriptions are expressive enough to enable inferences about future world states during application execution, as well as identification of potential system restriction violations.

Table 1. Defined specification axioms for application verification.

Specifications	Sample Event Calculus Axiomatization
Service Specifications	// Sorts retrieved from the *resource* ontology *sort light: object* *light Light01_MainRoom* // Events retrieved from the *event* ontology *event TurnOnLight(application,light)* *event TurnOffLight(application,light)* // Fluents retrieved from the *context* ontology *fluent LightOn(light)* // Effect Axioms based on the *event* and *context* ontology connection *∀l,t Initiates(TurnOnLight(l),LightOn(l),t).* *∀l,t Terminates(TurnOffLight(l),LightOn(l),t).*
System Restrictions	*∀serv,light,t HoldsAt(LocalizationRunning(serv),t)∧* *HoldsAt(LocatedIn(light,MainRoom),t)⇒ HoldsAt(LightOn(light),t).*
Application Specifications	*∀chair,l,t Happens(SitOn(chair),t) ⇒* *Happens(TurnOffLight(l),t) ∧ Happens(StartPresentation(<file>,TV01),t).* *∀chair,l,t Happens(StandUpFrom(chair),t) ∧ ¬HoldsAt(LightOn(l),t) ⇒* *Happens(TurnOnLight(l),t) ∧ Happens(StopPresentation(<file>,TV01),t).*

4.2 Design-time Application Verification

It is of particular significance to the management of an AmI system to separate the rules that govern its behavior from the domain-specific functionalities in order to enable efficient and dynamic adaptation to changes during development. We implement a modular approach that distinguishes the rules that express *system policies and restrictions* that guarantee a consistent and error-free overall execution at all times, from *service specifications* that change in frequent time periods and by a multitude of users, as well as from *application specifications* that are usually under the responsibility of non-experts who only possess partial knowledge about which the system restrictions are. Table 1 shows samples of the type of information that these specifications contain expressed in Event Calculus axioms. The specifications of services, for instance, retrieved from the different ontologies, describe the domains and express inheritance relations, instantiations of entities, and potentially context-dependent effect properties. Application descriptions express primarily the intended behavior of a developed application as a narrative of context-dependent action occurrences. Finally, system restrictions capture assertion about attributes of system states that must hold for every possible system execution (sometimes also called *safety properties* [10]).

A core task of our framework is to verify that the specifications of AmI applications are in compliance with the overall system restrictions and detect errors early in the development phase. This *a priori* analysis is performed at design-time and can formally be defined as the abductive reasoning process to find a set P of permissible actions that lead a consistent system to a state where some of its constraints are violated, given a domain description D, an application description AP_i for application i and a set of system constraints C:

$$D \wedge AP_i \wedge P \models \exists t \neg C(t) \text{ where } D \wedge AP_i \wedge P \text{ is consistent}$$

In fact, if such a plan is found it acts as a counterexample providing diagnostic information about violated safety properties. Apparently, such inferences are computationally expensive and most importantly semidecidable. Nevertheless, Russo et al. [10] proved that a reduction considering only two timepoints, current (t_c) and next (t_n), can transform such an abductive framework to fully decidable and tractable under certain conditions (no nesting temporal quantifiers):

$$D(T) \wedge AP_i(T) \wedge C(t_c) \wedge P \models \neg C(t_n) \text{ given a 2-timepoint structure T}$$

This way, we do not need to fully specify the state at time t_c; the generated plan P is a mixture of $HoldsAt$ and $Happens$ predicates without requiring a complete description of the initial system state, in contrast to similar model-checking techniques. We plan to even expand the type of system specifications to allow for optimizing the process of application designing, capturing for instance inefficient action executions that for the developer may seem harmless or unimportant.
Example. A developer uploads an application description file to the system containing, among others, the two axioms shown in Table 1. The new application must first be examined for consistency with respect to the set of restrictions already stored in the system by service engineers. The developer executes the *ApplicationCheck* functionality of the Validator accessible through the middleware, which identifies a potential restriction violation whenever a user sits on a chair; the event causes the $TurnOffLight$ action to occur that conflicts with the Localization being at a *Running* state (any substantial change in lighting destabilizes the localization process). As a result, the developer needs to review the application, pausing for instance the Localizer before closing the lights. □

4.3 Run-time Validation

Although application analysis can be accomplished at design-time and in isolation, action validation must be performed at run-time considering the current state of the system, as well as potential conflicts with other applications that might share the same resources. For that purpose, action validation is not implemented as an abduction process as before. Instead, a projection of the current state is performed to determine potential abnormal resulting states, which is a more efficient approach for the needs of run-time reasoning. We have identified the following situations where action theory reasoning can contribute solutions:

Resource management. An Ambient Intelligence reasoner needs to resolve conflicts raised by applications that request access to the same resource. We introduce axioms to capture integrity constraints, as below:
$\forall app1, app2, t\ HoldsAt(InUseBy(Speaker01, app1), t) \wedge$
$HoldsAt(InUseBy(Speaker01, app2), t) \Rightarrow (app1 = app2)$
We also intend to expand the Validator's resource allocation policies with short-term planning based on known demands.

Ramifications and priorities. Apart from direct conflicts between applications, certain actions may cause indirect side-effects to the execution of others. Terminating a service may affect applications that do not use it directly, instead

invoke services that depended on it. Since the reasoner is the only module aware of the current state of the system as a whole it can detect such unsafe effect ramifications and take measures to prevent unintended situations to emerge, either by denying the initial actions or by reconfiguring certain system aspects. Towards this direction, actions are executed in terms of prioritization policies.

Uncertainty handling. Imagine a system constraint requiring for the main-room door to be in locked state iff no user is located inside. The multi-camera localization component may lose track of users under certain circumstances, e.g. if they stand behind an obstacle. In such a case where the reasoner is not aware of whether the user has left the room or not, it needs to perform reasoning based on partial knowledge. As a result, the state constraint must be formulated as:
$\forall user, t\ HoldsAt(Knows(\neg UserInRoom(user)), t) \Leftrightarrow$
$HoldsAt(Knows(DoorLocked(DOORMAIN)), t)$
Situations of ambiguous knowledge are very common in AmI systems. For that purpose, we plan to integrate a recent extension of the Event Calculus axiomatization that accounts for knowledge-producing actions in partially observable domains, enabling knowledge update based on sense actions and context [11].

5 Discussion and Conclusions

Responding to the need for event processing, rule-based approaches have contributed in manifold ways to the development of event-driven IT systems over the last years, from the field of active databases to distributed event-notification systems [12]. Within the field of ubiquitous computing, this paradigm is most frequently applied to the design of inference techniques for recognizing high-level context, such as user activity in smart spaces. For instance, FOL rules for managing and deriving high-level context objects and resolving conflicts is applied in SOCAM [13], whereas [14] describes a system that executes rules for in-home detection of daily living activities for elderly health monitoring.

Our framework applies rule-based techniques in a style similar to these systems, but as previously argued, a full-scale AmI system requires much more potent reasoning both for context-modeling and for regulating the overall system operation. The need for hybrid approaches is highlighted in many recent studies [3, 4]. Towards this direction, the COSAR system [15] combines ontological with statistical inferencing techniques, but concentrates on the topic of activity recognition. In [16] a combination of rule-based reasoning, Bayesian networks and ontologies is applied to context inference. Our approach, on the other hand, combines two logic-based approaches, namely rule- and causality-based reasoning and achieves a general-purpose reasoning framework for AmI, able to address a broad range of aspects that arise in a ubiquitous domain.

The proposed integration of technologies is a novel and enabling direction for the implementation of the Ambient Intelligence vision. In is in our intention, while developing the run-time functionalities of the framework, to contribute to the action theories research, as well. A Jess-based Event Calculus reasoner, offering efficient online inferencing, is already under implementation.

References

1. Ramos, C., Augusto, J.C., Shapiro, D.: Ambient Intelligence–the Next Step for Artificial Intelligence. IEEE Intelligent Systems **23**(2) (2008) 15–18
2. Thomas Strang, C.L.P.: A Context Modeling Survey. In: 1st International Workshop on Advanced Context Modelling, Reasoning and Management. (2004)
3. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A Survey of Context Modelling and Reasoning Techniques. Pervasive and Mobile Computing (2009)
4. Bikakis, A., Patkos, T., Antoniou, G., Plexousakis, D.: A Survey of Semantics-based Approaches for Context Reasoning in Ambient Intelligence. In: Proceedings of the Workshop Artificial Intelligence Methods for Ambient Intelligence. (2007) 15–24
5. Paschke, A.: ECA-RuleML: An Approach combining ECA Rules with temporal interval-based KR Event/Action Logics and Transactional Update Logics. CoRR **abs/cs/0610167** (2006)
6. Kowalski, R., Sergot, M.: A Logic-based Calculus of Events. Foundations of knowledge base management (1989) 23–51
7. Miller, R., Shanahan, M.: Some Alternative Formulations of the Event Calculus. In: Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II, London, UK, Springer-Verlag (2002) 452–490
8. Kharbili, M.E., Stojanovic, N.: Semantic Event-Based Decision Management in Compliance Management for Business Processes. In: Intelligent Event Processing - AAAI Spring Symposium 2009. (2009) 35–40
9. Shanahan, M.: The Event Calculus Explained. Lecture Notes in Computer Science **1600** (1999) 409–431
10. Russo, A., Miller, R., Nuseibeh, B., Kramer, J.: An Abductive Approach for Analysing Event-Based Requirements Specifications. In: ICLP '02: Proceedings of the 18th International Conference on Logic Programming, London, UK, Springer-Verlag (2002) 22–37
11. Patkos, T., Plexousakis, D.: Reasoning with Knowledge, Action and Time in Dynamic and Uncertain Domains. In: 21st International Joint Conference on Artificial Intelligence. (2009) 885–890
12. Paschke, A., Kozlenkov, A.: Rule-Based Event Processing and Reaction Rules. In: RuleML '09: Proceedings of the 2009 International Symposium on Rule Interchange and Applications. (2009) 53–66
13. Gu, T., Pung, H.K., Zhang, D.Q.: A Service-oriented Middleware for Building Context-aware Services. Journal of Network and Computer Applications **28**(1) (2005) 1–18
14. Cao, Y., Tao, L., Xu, G.: An Event-driven Context Model in Elderly Health Monitoring. Ubiquitous, Autonomic and Trusted Computing **0** (2009) 120–124
15. Riboni, D., Bettini, C.: Context-Aware Activity Recognition through a Combination of Ontological and Statistical Reasoning. In: 6th International Conference on Ubiquitous Intelligence and Computing. (2009) 39–53
16. Bulfoni, A., Coppola, P., Della Mea, V., Di Gaspero, L., Mischis, D., Mizzaro, S., Scagnetto, I., Vassena, L.: AI on the Move: Exploiting AI Techniques for Context Inference on Mobile Devices. In: 18th European Conference on Artificial Intelligence. (2008) 668–672