2010 Third International Conference on Communication Theory, Reliability, and Quality of Service

Ontology-based Representation of UPnP Devices and Services for Dynamic Context-aware Ubiquitous Computing Applications

Konstantinos Togias[1,2], Christos Goumopoulos[1], Achilles Kameas[1,2]

[1]DAISy Group
Computer Technology Institute
Patras, Greece
ktogias@cti.gr, goumop@cti.gr, kameas@cti.gr

[2]School of Science and Technology
Hellenic Open University
Patras, Greece
ktogias@eap.gr, kameas@eap.gr

***Abstract*—**Ontology and related technologies have been introduced into the Ambient Intelligence domain as a mean to provide declarative formal representations of the domain knowledge. The range of devices available in the scope of an Ambient Intelligence space becomes increasingly heterogeneous and at the same time ubiquitous. Hence there is a need to link the discovery, description and deployment of these ambient devices and their services with context and domain knowledge representations in order to facilitate an Ambient Intelligence space experience. The contribution of this work is an approach for bridging the gap between the non-semantic description mechanisms of XML based devices description protocols, such as UPnP, and the AmI domain knowledge representation. For this we design a prototype ontology-based representation for UPnP devices and services that provide a semantic linking between human-centric abstract description, and the software-centric concrete description that derives from the UPnP descriptors and is necessary to remotely execute method calls on devices. We also demonstrate the benefits of its use with a prototype implementation.

Keywords-ontology; semantics; ubiquitous computing; service description; device description; UPnP

I. INTRODUCTION

The ability to automatically discover, configure, control and monitor the available devices is mandatory for an Ambient Intelligence (AmI) space [1] in order to provide intelligent and adaptive behavior. There are several systems that aim at providing the device and service description and discovery mechanisms needed in pervasive systems. Jini [2], Zero Configuration Networking (Zeroconf) [3], Universal Plug and Play (UPnP) [4], Digital Living Network Alliance (DLNA) [5], Device Profile for Web services (DPWS) [6] are the most prominent of them. Some of them are based on central repositories while other utilize a peer-to-peer architecture. Most of them use Extensible Markup Language (XML) [7] descriptions for devices, services and application programming interfaces (APIs) [8]. The PalCom project also proposes the description of services in XML format [9] .

UPnP is the most well established industry-wide protocol for device advertising, discovery controlling and monitoring. UPnP devices advertise their presence and their services through UPnP XML descriptions while UPnP access points can subscribe and invoke UPnP actions on them. The UPnP device description contains several pieces of vendor-specific information, definitions of all embedded devices, URL for presentation of the device, and listings for all services, including URLs for control and eventing. The UPnP service description defines actions and their arguments, state variables and their data type, range, and event characteristics [4]. The UPnP forum has developed specifications for some device classes, so-called "Device Control Protocols" (DCPs) [10]. Each DCP defines a common (machine-level) interface for a class of UPnP devices with embedded services in terms of mandatory and optional actions and state variables. Control points that have foreknowledge of the DCP that a device is using, can thus easily make use of its interface. Considering though, that the specified device classes cover only a small portion of the types of the available UPnP devices, that the protocol does not make it mandatory for manufacturers to adhere to it, as it provisions custom device and service types [4], and the lack of semantics from such XML-based description models, proves UPnP nearly unusable to reason and make adaptation decisions.

Ontologies have become the means of choice for knowledge representation in recent years. In general, ontologies do add a layer of semantics that provides a common and formal understanding of domain concepts on top of the syntax modeling provided by the existing schema languages, such as XML [11]. Ontologies and related technologies have been introduced into the AmI domain as a means to provide declarative formal representations of the domain knowledge and thus to enable intelligent behaviors, such as context-based device discovery and selection, service deployment and adaptation.

While many researchers have proposed architectures for semantics-based context-aware dynamic service composition, they either propose domain custom low-level discovery and configuration mechanisms that are not based on UPnP [12][13] or link specific UPnP DCPs (e.g., multimedia service DCP) with specific domain ontologies (e.g., MPEG-21 DIA Ontology) [11]. To the best of our knowledge there is no generic ontology-based representation for the UPnP protocol.

This paper aims to extend the AmI space's capability to reason and make adaptation decisions by bridging the gap between generic XML-based description protocols (such as UPnP) and ontologies. We propose an ontology-based representation for UPnP devices and services that provide a semantic linking between human-centric abstract description, and the software-centric concrete description that derives from the UPnP XML descriptors and is necessary to remotely execute method calls on devices. We also show its use with a prototype implementation.

The paper is organized as follows. We first give a motivating example in Section 2. Section 3 presents the

978-0-7695-4070-2/10 $26.00 © 2010 IEEE
DOI 10.1109/CTRQ.2010.44

proposed ontology-based representation of UPnP devices and services. Section 4 describes an implemented system that makes use of this representation, and Section 5 presents conclusions and suggests possible future work.

II. Motivating Example

We present below a motivating example to show how linking UPnP devices and services with semantical information is non-trivial and how a mapping of the XML device and service descriptions to an ontology will be helpful.

Imagine that Alice has equipped her house with a UPnP compatible AmI system. In order to support Alice's activities in a dynamic environment where various UPnP smart devices are entering, leaving, becoming available or unavailable in non-predictable ways, the system has to get answers to questions such as:

- Which of the present UPnP devices offer a service that is appropriate for supporting a specific task? (e.g., reading a book)
- What UPnP actions or state variables can be used to monitor that service? (e.g., to find out whether the text displaying service is activated or not)
- What UPnP actions can be used to control that service? (e.g., to activate or deactivate the text displaying service)
- What parameter values must the actions be invoked with in order to set a service to the appropriate state? (e.g., to activate the text displaying service)

We assume that the AmI system has access to domain and upper level ontologies that gather knowledge about devices and services, user profiles, privacy policies, etc. The system also is equipped with ontology management software that can query and inference on these ontologies [14]. Therefore, for providing answers to the above questions, there is need to combine the information provided by each device's UPnP description with the knowledge incorporated in ontologies. This is not a trivial task because the UPnP XML descriptors do not provide semantics and are not expressed in an RDF based language. In addition if Alice owns various UPnP devices from different manufacturers there is no guaranty that similar devices, services and actions will not be implemented in different ways and advertised with different UPnP names, types and ids, making answering the above questions even harder.

Suppose that Alice has two ontology enabled UPnP e-reader devices, from different manufacturers, each of which carries its own local ontology that describes it in an abstract way. As there is no standard UPnP DCP for this type of devices, the manufactures have described their devices in heterogeneous ways. The one e-reader is advertised as a device of type *"urn:manufacturer1-domain:device:e-book:1"* with a service of type *"urn:manufacturer1-domain:service:switchpower"* that provides an action named *"getstatus"* that returns a binary value, while the other is a device of type *"urn:manufacturer2-domain:device:tabletpc:2"* and provides a service of type *"urn:manufacturer2-domain:service:ereading"* with an *"isactive"* action that also returns a binary value. Even if a standard UPnP DCP existed for this type of devices and manufacturers had followed it, so their descriptions were identical, there is a need for those UPnP descriptions to be linked with the domain knowledge contained in the device ontology in order to facilitate the binding of abstract concepts (such as *"reading"*, *"monitor"* e.t.c.) to the actual devices, services and action invocations. Once such links and the appropriate semantic descriptions are provided by the devices' ontologies, the system, can use them to infer that both the devices can be used for *"reading"* and that in order to *"monitor"* the state of the device of type *"urn:manufacturer2-domain:device:tabletpc:2"* it has to invoke the *"isactive"* action of the service of type *"urn:manufacturer2-domain:service:ereading"*, while in order to *"monitor"* the state of the device of type *"urn:manufacturer1-domain:device:e-book:1"* it has to invoke the *"getstatus"* action of the service of type *"urn:manufacturer1-domain:service:switchpower"*.
Furthermore, those software-centric semantic descriptions can easily be linked with concepts deriving from already defined ontologies for pervasive computing and AmI spaces or domain ontologies that gather higher level knowledge about devices and services, user profiles, privacy policies, locations, etc.. Such links can be defined using the equivalence and sub-classing elements provided by the Web Ontology Language (OWL) [15] (e.g. "*owl:sameAs*", "*rdfs:subClassOf*" and "*rdfs:subPropertyOf*"). After such relations are defined, OWL reasoners such as Pellet [16] and Jena [17] can use them to answer complex questions.

To bridge the gap between the non-semantic description mechanism of UPnP protocol and the domain knowledge representation, we designed a prototype ontology-based representation for UPnP devices and services that provide a semantic linking between human-centric abstract descriptions and the software-centric concrete description that derives from the UPnP XML descriptors and is necessary to remotely execute method calls on devices. We have also developed a component in the context of ATRACO project [18] that uses instances of this representation for binding UPnP devices to abstract service descriptions and providing ontology-based device selection, service invocation and device failure adaptation.

III. Ontology-based representation of UPnP devices and services

In order to describe a device in a way that can be interpreted by software components, we have introduced an ontology-based representation of devices and services. To support consistent AmI space functionality and human-centric operation, a device should be described in two ways: a human-centric and a software-centric. The human-centric perception of a device takes into consideration the human that uses a device, the services it provides to him and its characteristics as a physical object. The software-centric view of the device focuses on the software API it provides to other software components and describes operations, parameters, variables and return values. The proposed ontology represents and links together both the human-centric and the software-centric descriptions of a device and is built in a modular fashion. The use of OWL [15] import element allows the separation of the levels of a domain description into independent documents, which can then be imported as needed. This technique helps writing clearer

descriptions, by separating the definitions according to the sub-domains they describe. It also maximizes the ability to reuse ontologies of all kinds. As a result, ontologies structured in this way are easier to be used and maintained. The proposed ontology consists of three modules. One module aims at providing a basic human-centric description of the device while the two other modules describe the software-centric characteristics focusing at the Device and Service levels respectively. For the software-centric description we focus on devices that implement the UPnP protocol, but modules describing the aspects of other protocols can be also developed in the future. From now on, we will refer to the group of the three modules as the *"Device Ontology"*, while the modules will be referred as *"Physical Device Ontology"*, *"UPnP Device Ontology"* and *"UPnP Service Ontology"* respectively when considered as ontologies by their owns.

A. The Physical Device ontology

The Physical Device ontology describes the generic human-centric aspects of a device such as it's name, it's owner, description tags, its location, the services it provides , the states each service can have and the services' current states (from a human-centric view). It also provides properties that relate the described device, its services and it's states to its corresponding UPnP Device and UPnP Services that are described by the *"UPnP Device"* and *"UPnP Service"* ontologies. The main concepts defined in the Device Ontology are the following:

- *Device:* Represents a device in its general human-centric notion. Each physical device is considered as an instance of this class.
- *Service:* Represents an abstract service that a device can provide to a human. e.g., An e-reader can provide reading service to a human, while a lamp can provide lighting services.
- *State:* Represents the abstract states of a Service. e.g., *"enabled"* and *"disabled"* can be the possible states of the reading service of an e-reader, while the lighting control service of a dimmable light may have three possible states: *"Low"*, *"Moderate"*, and *"High"*.

Figure 1 depicts the main classes and properties of the Physical Device ontology.

A Physical Device may embody a UPnP Device. The *upnpdevice:Device* class that represents the UPnP essence of a device is defined in the "*UPnP Device*" ontology. The *embodiesupnpdevice* property describes this relation. The same way an abstract service that is provided by the device can be related to an UPnP Service provided by the UPnP device through the *isRelatedToupnpservice* property. The *upnpservice:Service* class that represent a UPnP Service is defined in the UPnP Service ontology. The *hasStateGetUpnpAction* and *hasStateSetUpnpAction* properties link the abstract service with the exact UPnP actions that must be invoked in order to get and set its state. Similarly the *hasStateGetUpnpArgument* and *hasStateSetUpnpArgument* properties link the service with the UPnP arguments that must be provided for invoking the get and set actions. The *hasStateUpnpVariable* property links a service with the UPnP state variable that can be used for getting notifications about current state. The *upnpservice:Action*, *upnpservice:Argument* and *upnpservice:StateVariable* classes that represent UPnP actions, arguments and state variables are defined in the UPnP Service ontology. A service is connected with its possible states with the *hasState* property. The current state of the service can also be retracted by the ontology, without needing to invoke some UPnP method, through the *hasCurrentState* attribute, provided that the device has an internal mechanism for keeping its ontology instance up to date as its services change states. The *hasUpnpValueMapping* and *hasUpnpRangeMapping* properties map a state with the actual argument or return values that represent it in the UPnP context.

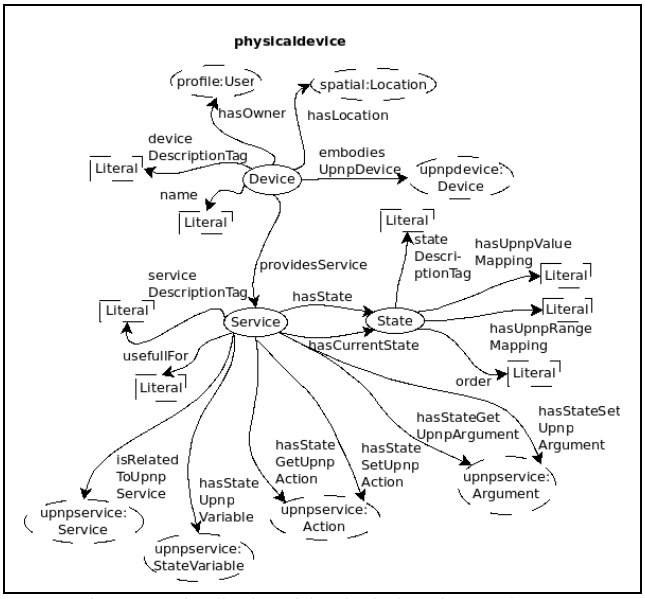

Figure 1. Visualization of the Physical Device Ontology

Finally the *hasOwner* and *hasLocation* properties can be used to link the device with a user and a location. The description of the concepts User and Location is out of the scope of this ontology, so we assume that they are defined in separate user profile and spatial ontologies that can be imported if used. The datatype properties *deviceDescriptionTag*, *serviceDescriptionTag* and *stateDescriptionTag* enables the tagging of devices, services and states from the manufacturer or the user. Services have also the *isUsefulFor* datatype property for tagging the uses a service may have.

B. The UPnP Device ontology

The upnpdevice ontology contains all the device attributes that are defined by the UPnP device architecture [4]. Such attributes are *friendlyName*, *Manufacturer*, *icon*, *model*, *embedded device*, *udn*, provided UPnP services, etc. It also relates the UPnP Device with the description of the UPnP Services it provides as it is presented by the upnpservice ontology.

The main class of upnpdevice ontology is the *Device* class that represents a UPnP device as it is defined by the

UPnP protocol. The classes *EmbeddedDevice*, *Manufacturer*, *Icon*, *Model* and their properties are direct mappings of the UPnP device XML template and their meaning is described in [4]. Figure 2 shows a visual representation of the UPnP Device ontology.

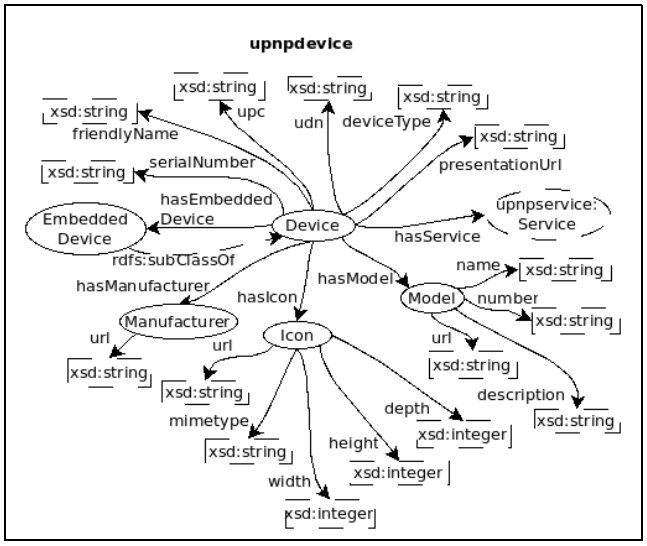

Figure 2. Visualization of the UPnP Device Ontology

C. *The UPnP Service ontology*

The upnpservice Ontology describes the UPnP Service attributes as they are defined by UPnP protocol. For example, a UPnP service has a *service type*, is related to *state variables* and provides *actions* that have *arguments*. The main class of upnpservice ontology is the *Service* class that represents a UPnP service as it is defined by the UPnP protocol. All other classes and properties are direct mappings of the UPnP service XML template and their meaning is described in [4]. Figure 3 shows a visual representation of the UPnP Service ontology.

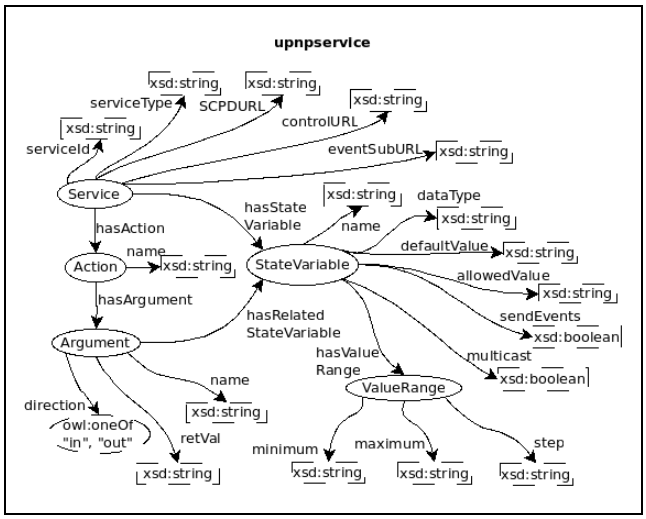

Figure 3. Visualization of the UPnP Service Ontology

D. *An example instance of the device ontology*

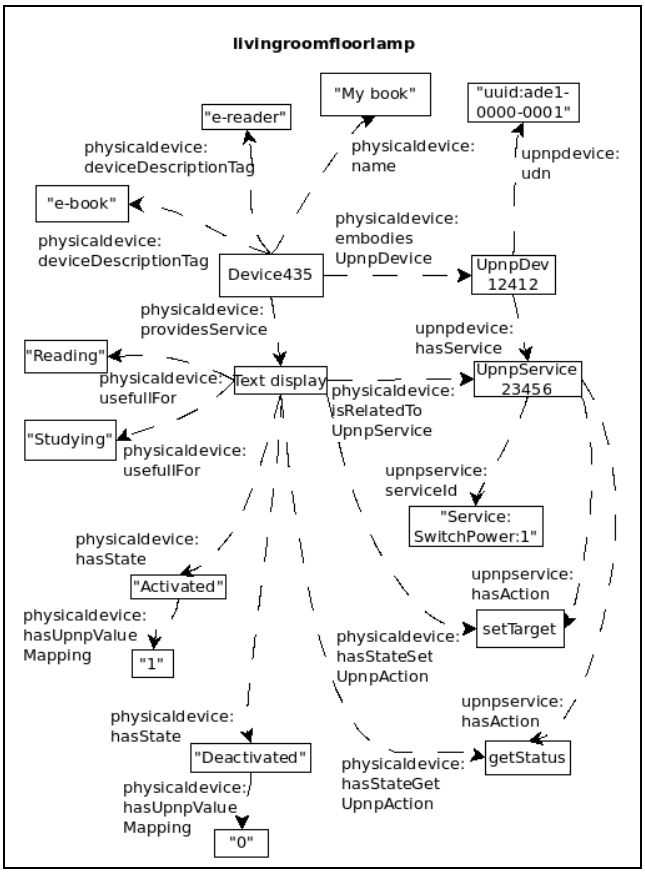

Figure 4. Partial visualization of an instance of the Device Ontology

The concepts defined in the Device, UPnP Device and UPnP Service ontologies can be used to create ontology instances for describing the devices that are available in an AmI space. Each device is supposed to have one local instance that specializes the classes defined in the three ontologies to its actual human-centric and software-centric properties. The visualization of a part of such an instance is shown in Figure 4.

In this example, we describe the one of the two e-reader devices that we presented at the previous section. The user views it as *"My book"* and it is recognized at the UPnP context as the device *"uuid:ade1-0000-0001"*. The device is tagged as *"e-reader"* and *"e-book"*. It provides a service named *"text display"* that is useful for *"reading"* and *"studying"* and has the states *"activated"* and *"deactivated"*. The *"text display"* service is related to the UPnP service with id *"Service:SwitchPower:1"*. It's *"activated"* state corresponds to the UPnP value *"1"* while it's *"deactivated"* state corresponds to value *"0"*. The *setTarget* UPnP action can be used for setting the state of the *"text display"* service, while the *getStatus* action returns information about *"text display"* service's current state.

IV. IMPLEMENTATION

In the context of ATRACO project [18] we developed a component, consisting of a control point and an ontology manager, that uses instances of the device ontology for selecting, controlling and monitoring UPnP devices and services based on semantic descriptions. Each device was equipped with a local instance of the ontology and with ontology management software that was used for automatically producing the UPnP Service and UPnP Device ontology instances from the UPnP XML descriptors of the device and for keeping the instance up to date regarding run-time attributes such as the service's current states and the device's location. The ontology manager was used for importing the devices' ontology instances and performing queries to the resulting combined graph. For the representation of ontologies we used the OWL ontology language [15n], while the software was written in Java [19].

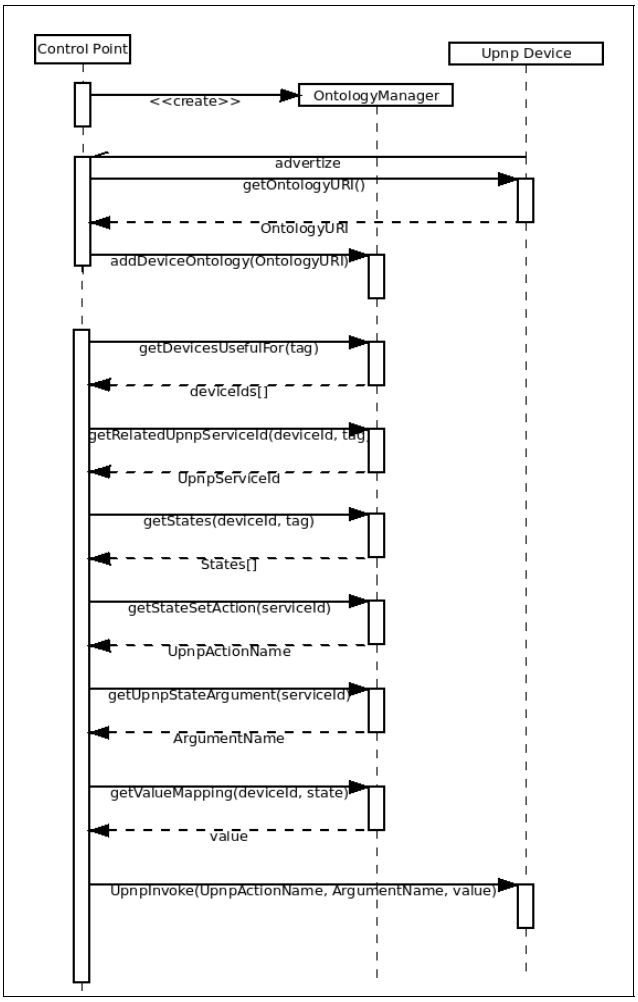

Figure 5. Interaction between the control point, the ontology manager

For testing the component we turned two laptop computers into two UPnP e-reader devices using software we developed for this purpose. Their UPnP device and service descriptors where crafted in heterogeneous ways, providing different device and service type declarations and different action and parameter names (see Section 2 for detailed description of the two UPnP devices). Then we equipped them with device ontology instances. We described the devices and their services and states with human-centric description tags and connected them with the corresponding UPnP level actions and values.

We started our experiment by bringing the first e-reader in the AmI space. After importing its ontology, the system successfully selected it as a device that provides services "useful for reading" and automatically invoked the appropriate action for activating it, utilizing the information encoded in the device ontology. Next we brought in the second e-reader. Once it was discovered, its ontology was also imported by the ontology manager. We then simulated a malfunction on the first e-reader and made the system select an alternative device for continuing the reading task. The second e-reader was automatically selected and activated although its UPnP description, action names and parameters where totally different. Thus, we showed that the proposed device ontology can be used to provide context-aware adaptation and support device selection and activation decisions.

In order to show how our ontology can be linked with other ontologies in order to support higher level inference, we defined the *physicaldevice:Device* class as a sub-class of the *geo:SpatialThing* class and the *physicaldevice:hasLocation* class as a sub-property of *geo:location*. The *geo:SpatialThing* and *geo:location* are defined at the W3C WGS84 Geo Positioning ontology [20]. We also linked our ontology with the Friend of a Friend (FOAF) ontology [21] by defining *profile:Person* as a sub-class of *foaf:Person*. We then created FOAF profiles for two hypothetical persons Joe and Anni. The *foaf:Person* class is defined in the FOAF ontology as a subclass of *geo:SpatialThing*. We defined the location of Joe, Anni and the first e-reader as "*Livingroom*" and the location of the second e-reader as "*Kitchen*". We also defined Joe as the owner of the first e-reader and Anni as the owner of the second using the *physicaldevice:hasOwner* property. We then used the OWL reasoner provided by the Jena RDF framework [17] for answering complex queries such as "*What are the UPnP devices that provide a service useful for reading and are at the same location as their owners?*".

A detailed sequence diagram showing the interaction between the control point, the ontology manager and the UPnP device is shown at figure 5 while some of the queries that where executed on the combined graph during our runs, expressed in SPARQL, and the returned results are shown at Table 1.

TABLE I. EXAMPLE QUERIES AND THE CORRESPONDING RESULTS

Query	Result
SELECT DISTINCT ?upnpdeviceid FROM ?device a physicaldevice:Device . ?device physicaldevice:providesService ?service . ?service physicaldevice:usefulFor ?tag . ?device physicaldevice:embodiesupnpdevice ?upnpdevice . ?upnpdevice upnpdevice:udn ?upnpdeviceid . FILTER regex(?tag "reading", "i")	*?upnpdeviceid = "uuid:ade1-0000-0001"*

SELECT DISTINCT ?value FROM ?device a physicaldevice:Device . ?device physicaldevice:embodiesupnpdevice ?upnpdevice . ? upnpdevice upnpdevice:udn ?upnpdeviceid . ?service physicaldevice:hasState ?state . ?state rdfs:label ? statelabel . ?state physicaldevice:hasUPnPValueMapping ?value FILTER ((?upnpdeviceid="uuid:ade1-0000-0001") && ? statelabel = "Activated")	?value = "1"
SELECT DISTINCT ?upnpdeviceid FROM ?device a physicaldevice:Device . ?device physicaldevice:providesService ?service . ?service physicaldevice:usefulFor ?tag . ?device geo:location ?devloc . ?device physicaldevice:hasOwner ?user . ?user geo:location ?userloc . ?device physicaldevice:embodiesupnpdevice ?upnpdevice . ? upnpdevice upnpdevice:udn ?upnpdeviceid . FILTER (regex(?tag "reading", "I") && ?userloc = ?devloc)	*?upnpdeviceid = "uuid:ade1-0000-0001"*

V. Conclusions and future work

This paper presented a prototype ontology-based representation for UPnP devices and services that provide a semantic linking between human-centric abstract description and the software-centric concrete description that derives from the UPnP XML descriptors. The resulting ontology has rich semantics that can be combined with other upper level domain ontologies and used by ontology-based systems. We implemented an ontology-based system and showed that the use of UPnP devices that are equipped with instances of this ontology provide the necessary mappings for performing context-aware adaptation and device selection and activation decisions.

For the software-centric representation we focused on the prominent UPnP standard, but we would like to extend the ontology with modules that map other known device description protocols such as DPWS that is advertised as the successor of UPnP. In the near future, we would also like to try applying automatic alignment algorithms using ontology alignment frameworks such as Alignment API [22], heuristics and publicly available general purpose synonym lexicons, taxonomies and semantic networks, such as Wordnet [23] and ConceptNet [24], in order to align instances of the device ontology that use different tags to describe similar services. It will also be interesting to align and integrate the device ontology with existing context ontologies for smart spaces and domain ontologies, such as policy ontologies, user profile ontologies and spatial ontologies in order to enable the AmI system for richer more intelligent and more adaptive behavior.

Acknowledgment

The research described is partly supported by the ATRACO (ICT-216837) project.

References

[1] ISTAG in FP6: Working Group 1, IST Research Content, Final Report. Available at ftp://ftp.cordis.europa.eu/pub/ist/docs/istag-wg1-final_en.pdf. Retr. Feb 2010.

[2] J. Waldo. “The Jini Architecture for Network-Centric Computing.” Communications of the ACM, 1999, pp. 76 - 82 .

[3] D. Steinberg and S. Cheshire. “Zero Configuration Networking: The Definitive Guide.” O’Reilly Media, Inc., 2005.

[4] Contrib. Members of the UPnP Forum. “UPnP™ Device Architecture 1.1.” Available at http://www.UPnP.org/specs/arch/UpnP-arch-DeviceArchitecture-v1.1.pdf. Retr. Feb 2010.

[5] N. Venkitaraman. “Wide-Area Media Sharing with UPnP/DLNA.” in Proc. of Consumer Communications and Networking Conference, CCNC 2008. 5th IEEE, pp. 294-298, 2008.

[6] OASIS, “Devices Profile for Web Services Version 1.1 – Public Review Draft 01”, Available at http://docs.oasis-open.org/ws-dd/dpws/1.1/wsdddpws-1.1-spec.html, Retr. Feb 2010.

[7] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. “Extensible Markup Language (XML) 1.0 (Fifth Edition)”, W3C, Available at http://www.w3.org/TR/REC-xml/, Retr. Feb 2010.

[8] Wikipedia, “Application programming interface.”, Available at http://en.wikipedia.org/wiki/Application_programming_interface, Retrieved Feb 2010.

[9] D. Svensson Fors, B. Magnusson, S. Gestegård Robertz, G. Hedin, and E. Nilsson-Nyman, “Ad-hoc composition of pervasive services in the PalCom architecture.”, in Proc. of the 2009 international conference on Pervasive services, New York, NY, USA, pp. 83-92, 2009.

[10] Contributing Members of the UPnP Forum, “UPnP Standardized DCPs.”, Available at http://www.UPnP.org/standardizeddcps/default.asp, Retr. Feb 2010.

[11] N. Li, A. Attou, S. De, and K. Moessner, “Device and service descriptions for ontology-based ubiquitous multimedia services.”, in Proc. of the 6th International Conference on Advances in Mobile Computing and Multimedia, New York, NY, USA, pp. 370-375, 2008.

[12] K. Fujii and T. Suda, T. “Semantics-based context-aware dynamic service composition.”, ACM Trans. Auton. Adapt. Sys., 4(2), pp. 1-31, 2009.

[13] A. Toninelli, A. Corradi, and R. Montanari, “Semantic-based discovery to support mobile context-aware service access.”, Computer Communications, 31(5), pp. 935-949, 2008.

[14] L. Seremeti, C. Goumopoulos, and A. Kameas, “Ontology-based modeling of dynamic ubiquitous computing applications as evolving activity spheres.”, Perv. Mob. Comp., 5(5), pp. 574-591, 2009.

[15] M. K. Smith, C. Welty, and D. L. McGuinness. “OWL Web Ontology Language Guide.”, W3C, Available at http://www.w3.org/TR/owl-guide/, 2004. Retr. Feb 2010.

[16] Pellet open-source Java based OWL DL reasoner, Available at http://www.mindswap.org/2003/pellet/. Retr. Feb 2010.

[17] Jena Semantic Web Framework for Java, Available at http://jena.sourceforge.net/. Retr. Feb 2010.

[18] A. Goumopoulos et al., “ATRACO: Adaptive and Trusted Ambient Ecologies, Self-Adaptive and Self-Organizing Systems (SASO).”, Workshop on Pervasive Adaptation (PERADA), IEEE CS, pp. 96-101, 2008.

[19] J. Gosling, B. Joy, and G. Steele, “The Java Language Specification, Third Edition.”, http://java.sun.com/docs/books/jls/third_edition/html/j3TOC.html . Retr. Feb 2010.

[20] WGS84 Geo Positioning RDF vocabulary, Available at http://www.w3.org/2003/01/geo/wgs84_pos. Retr. Feb 2010.

[21] FOAF Vocabulary Specification, Available at http://xmlns.com/foaf/spec/. Retr. Feb 2010.

[22] J. Euzenat, “An API for ontology alignment”, in Proc. of 3rd conference on international semantic web conference (ISWC), Hiroshima (JP), Lecture notes in computer science 3298, pp. 698-712, 2004.

[23] G. A. Miller, “WordNet: an on-line lexical database.”, International Journal of Lexicography, pp. 235-244, 1990.

[24] C. Havasi, R. Speer, and J. Alonso, “Conceptnet 3: a flexible, multilingual semantic network for common sense knowledge.”, In Recent Advances in Natural Language Processing, Borovets, Bulgaria, 2007.