J Supercomput
DOI 10.1007/s11227-014-1296-5

ERRATUM

Erratum to: Exploiting smart spaces for interactive TV applications development

M. Mohsin Saleemi · Natalia Díaz Rodríguez · Johan Lilius

Erratum to: J Supercomput
DOI 10.1007/s11227-014-1183-0

The authors would like to correct authorship credits of the original publication.

Natalia Díaz Rodríguez contributed to the research presented in this article but was mistakenly excluded from the authorship list. The correct list is given in this erratum.

The online version of the original article can be found under doi:10.1007/s11227-014-1183-0.

M. M. Saleemi (✉) · N. Díaz Rodríguez · J. Lilius
Åbo Akademi University, Turku, Finland
e-mail: mohsin.saleemi@gmail.com

N. Díaz Rodríguez
e-mail: ndiaz@abo.fi

J. Lilius
e-mail: johan.lilius@abo.fi

Published online: 11 September 2014

J Supercomput
DOI 10.1007/s11227-014-1183-0

Exploiting smart spaces for interactive TV applications development

M. Mohsin Saleemi · Johan Lilius

Abstract The integration of semantic technologies and TV services is a substantial innovation to improve the services to users in an environment that is extended beyond the fixed home environment. But currently, this integration is mainly limited to provide personalized recommendation services and systems by matching user static preferences. Designing and development of interactive TV (iTV) applications using semantic technologies are not realized yet. In this work, we explore the potential of introduction semantic technologies and smart spaces in design and development of iTV applications. We use an example scenario to show how future iTV applications include the mesh-up of information from different sources. We proposed a methodology and show how ontology-driven approach can help to design and develop these iTV applications. We demonstrate the suitability of our ontology-driven application development tools and rule-based approach for the development of highly dynamic context-aware iTV applications.

Keywords Interactive TV · Smart Space · Semantic technologies · Ontology · Context-aware · Ubiquitous

1 Introduction

With the advent of new standards and technologies for transmission, development and execution environments, interactive TV (iTV) evolves rapidly as a reality in all of its forms, such as digital TV, mobile TV and Internet protocol TV (IPTV), etc. The term interactive TV applications mean different things to different people and no single definition is presently accepted by all researchers. European Broadcasting Union

M. M. Saleemi (✉) · J. Lilius
Åbo Akademi University, Turku, Finland
e-mail: mohsin.saleemi@gmail.com

Published online: 29 April 2014

defines iTV applications as enhanced or interactive services with digital Television [1]. BBC defines iTV as follows: iTV is the content and services (in addition to linear TV channels) which are available for digital viewers to navigate through on their TV screens. In practice, this means giving viewers control over some video, audio, graphical and text elements, or allowing them to use simple games and quizzes or send simple communication back to broadcasters [2]. These definitions are usually supported by broadcasters providing iTV applications and are generally related to and bounded to specific TV programs.

Due to the recent technological developments, ICT landscape is evolving into a highly interactive distributed environment that demands integration of information in heterogeneous technologies and systems. Information in this environment is accessed using a range of different devices. These devices include portable devices (such as mobile phones, PDAs, smart phones, tablets) and fixed or non-portable devices (such as TV, set-top boxes, desktop computers, Personal video recorders). These devices provide new possibilities of interaction and all of them have the capacity to execute applications and share information with each other. With the birth of IPTV, Television and Web came closer to each other by sharing a substantial set of methodologies to provide the users immerse interactive experience. Users now have more control over data and content creation, consumption and sharing. It is foreseen that the future interactive TV applications would involve not only a wide range of digital devices in highly interactive, dynamically changing and context-aware environment but also data/information from different sources, such as web. Connecting up all kind of information and content by being much more dependent on the environment (physical and social environment) could enable whole universe of converged iTV applications. For example, assume an iTV banking application that shows users account balance and other details on TV screen (or mobile TV screen). User can use this application to pay his bills, etc. The application could detect the presence of other persons in the room and could automatically hide the balance details when the living room TV is being used to display application content. The same application while using on smartphone could detect it as a personal device and show all banking details.

It is clear that creating such interactive applications requires establishing concrete development infrastructures and methodologies that can provide a sufficient level of abstraction to hide the complexity. Currently, there is no commonly agreed suitable method for the development of iTV applications and different organizations have their own platforms and approaches and APIs (JavaTV API, MHP, OCAP API, etc.). Several companies such as Aircode, Alticast, ItVBox and Cardinal, etc. use their tools for the development of iTV applications. Their tools provide graphical environment to easily create simple iTV applications. These environments and tools are too limited for creation of complex iTV applications that involve information and resources from many sources. To make full use of the power of interactivity and content consumption, data and device interoperability issues must be solved and data must be structured in a way that could enable multiple devices to consume and share data between them. Moreover, traditional development methods and techniques must be replaced by scalable, agile and configurable methodologies.

Smart Spaces provide a solution for the interoperability problem by standardizing how to describe data formats. A *Smart Space* is an abstraction of space that

encapsulates both the information in a physical space as well as the access to this information, such that it allows devices to join and leave the space. In this way, a *Smart Space* becomes a dynamic environment whose identity changes over time when the set of entities interacts with it to share information between them. Smart Spaces could take advantage of digital TV/IPTV technologies to deliver content/-data to the receiving devices and these data could be shared between heterogeneous devices present in the Smart Space. Smart Space application development tools could be used to develop interoperable interactive TV applications that employ mesh-up of information from different sources and devices rather than a standard remote control. As a result of this convergence, a whole new universe of applications could be possible.

In our previous work, we have developed a programming interoperability solution [3,4] for rapid application development in Smart Spaces and it is based on the open source Smart-M3 architecture [5]. This paper discusses an approach for ontology-driven iTV application development and incorporating context-aware Event Control Action (ECA) rules in iTV applications such that they can be reactive to the user's context, while refraining from affecting the Smart-M3 platform standard. We further provide evaluation of our rule-based implementation by demonstrating the suitability of this approach for context-aware iTV applications.

2 Background and motivations

Due to the IP-based TV services, TV and web came closer to each other and the distinction between iTV applications and web services is becoming even harder as the operators combine and develop different technologies to serve specific situations. Moreover, iTV could use web as information space, i.e., utilizing web as an ultimate information source by means of variety of technologies, such as semantic web RDF, etc.

In view of recent advances, the definition of iTV applications has been changed and the definitions specified in previous section need to be modified to reflect the changes. We define iTV applications as services that are context aware and that could actively engage users to interact using multiple devices to participate in the application that may or may not be bounded to TV program and can be delivered and consumed through any medium, such as broadcast, cable, IP, web, etc. Moreover, information from heterogeneous sources could be used seamlessly. This definition covers all important aspects such as content, device and platform independence but highly context aware to adapt to user preferences. This brings the creation of such iTV applications closer to the creation of regular software as we know if for PC and mobile devices.

We believe that future interactive TV application would increasingly involve not only a wide range of digital devices in highly interactive, dynamically changing environment but also data/information from different sources, such as web. Moreover, they would also take benefits from pervasive computing environment to deliver highly personalized context-aware TV applications to the users. This is due to the increase in number of digital appliances embedded in the users surrounding. It gives to rise

pervasive interactive space that interconnects user, physical resources and computational entities. For example, iTV banking application given in the previous section. Hence there is a need to shift to new application development methodologies that can cope with issues such as dynamicity in terms of adding new devices and services, context-awareness, inferring new knowledge and sharing it with others.

We propose to use ontology-driven iTV applications development because

- Ontologies are perfect candidates for modeling context information which is highly desirable in future iTV applications to provide personal services dependent on the environment. Users at different contexts have different needs and expectations and ontologies can model the users context in effective way.
- As the concept of iTV has been evolving after the advent of IPTV, users are expecting highly dynamic systems where they can join and leave anytime to consume services. Smart Space provides this dynamic environment and ontologies are important concept in Smart Space-based infrastructure.
- Reasoning and inferring new knowledge from available information and searching and querying for their desired services are becoming essential part of iTV usages as TV came closer to the web in recent years. Ontology-driven architectures can provide reasoning capabilities in an effective way.
- Users now have range of devices in addition to the TV in their personal space to interact and consume iTV applications. It makes it a ubiquitous system where information from heterogeneous information sources such as sensors, digital appliances, web, smartphones, TV, PVR, etc. is used for realization of truly interactive applications. Ontologies are immediate solution for handling heterogeneity and provide information level interoperability to the users.

It is perceived that the use of ontologies will essentially change the way in which software systems/applications are built and that software designers will have libraries of ontologies from which they can choose relevant ones. Use of ontologies in application development provides competitive advantages over traditional approach enabling greater information sharing and reuse. Ontology-driven development (ODD) additionally exploits knowledge exploitation using reasoning over the maintained ontology.

In this work, we explore the potential of introduction of Smart Spaces in the design and development of interactive TV applications. In the previous work [3,4], we have developed ontology-driven tools and frameworks for rapid application development for Smart Spaces. We are now applying our ideas and methodologies of Smart Spaces to interactive TV domain as we believe this convergence could provide potential benefits in terms of value-added applications to the users. Our tools and methodologies provide benefits which are not currently realized in iTV domain, such as (i) abstracting underlying platform (ii) porting applications to different devices and platforms (iii) reducing efforts in learning APIs. Our approach for developing highly interactive applications deals with the key issues such as flexibility with respect to adding new devices and services to the Smart Space, high level of abstractions, rule-based reasoning, task-based and recommendation-based design and automatic code generation from application ontology.

3 Literature review

In the recent years, there has been coordinated efforts from multiple organizations and research groups towards inclusion of semantics in interactive TV services and platforms. Work presented in [6] describes a semantics-aware platform for interactive TV services to distribute, process and consume the media content. They proposed an interactive TV receiver framework capable of collecting, extending and reasoning semantic data related to broadcast multimedia content. The work presented in [7] outlines video annotation technique, ontology-based modeling, multimedia meta-data and user profiling through semantic reasoning. The main goal of this work is to create a personalized digital TV recommender based on meta-data. Other works in the direction of personalized TV programs recommendation systems based on semantics include [8,9]. All these approaches use semantics information for reasoning purpose to deliver personalized TV content.

Model-based approaches have been widely used for model-based user interface development [10]. Work presented in [11] and [12] describes approaches for model-driven development of interactive user interfaces, similar to researches like [13] and [14] which apply modeling concepts for creating platform independent user interfaces.

To the best of our knowledge, there is no work done on the creation of ontology-driven application for interactive TV. Our approach for developing highly interactive applications deals with the key issues, such as flexibility with respect to adding new devices and services for interaction, high level of abstractions, rule-based reasoning, task-based and recommendation-based design and automatic code generation from application ontology to facilitate application programmer. We have developed tools and frameworks for ontology-driven application development and applied them to interactive TV domain to realize the scenarios with mixture of technologies, systems, information and devices.

4 System architecture

In this section, the overall system architecture is outlined and Fig. 1 depicts this architecture. It consists of four main elements: first, the content provider sources, such as IPTV, mobile TV, digital TV, portable media providers, etc. Second, application and advertisement providers who provide application and advertisement content to be consumed by the users. Third, Modeling component that models the content and its meta-data. This modules require that the information on TV content and advertisement must be defined using some standard for representing meta-data, such as MPEG-7. The meta-data for TV content include title of the TV program, its category, actors, authors or anchor of the program, etc., and for the advertisement it includes name of the item, category, model, manufacturer, requirements and features, etc. We build ontologies based on this information which relate these concepts and their relations. Fourth, our Smart Space infrastructure including Semantic Information Broker (SIB) and reasoning engine. This infrastructure is the core component which used to store the ontologies and provide reasoning. It also facilitates interaction and communication with the user devices through KPs for sharing and updating information between

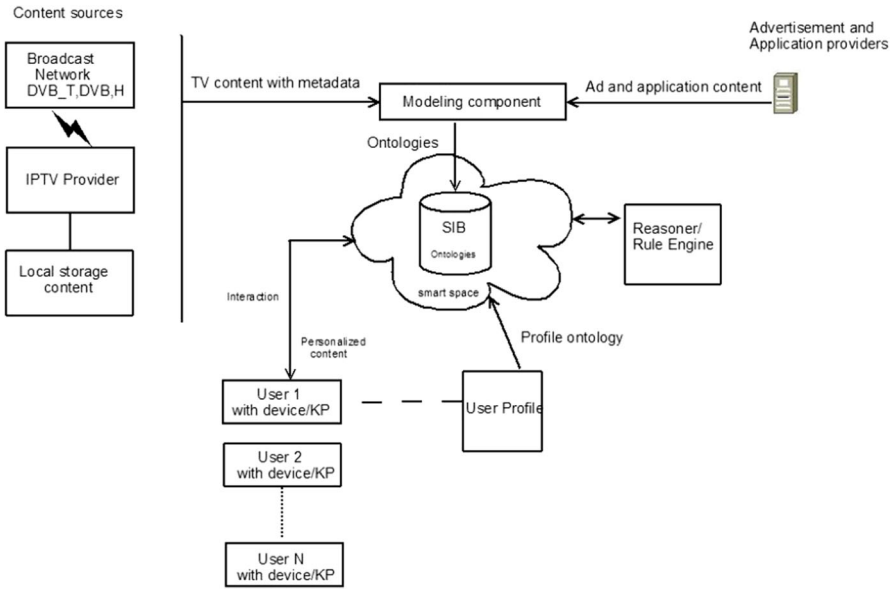

Fig. 1 Overall view of the system

them. Users profiles are also stored in the SIB as user profile ontology. Users receive personalized advertisement and applications based on their profile ontology. Users with heterogeneous devices can interact with the Smart Space to share information between them.

4.1 Enhanced iTV application scenario

We chose AuctionTV example scenario given in [15] because it exploits additional interaction and participation by iTV users. This application allows one of the participants to offer some item on sale through auction. This auctioneer get the role master and other users join afterwards get the role participant. Whenever a participant p bids on the item, the auctioneer raises the price confirming the bid. When the acceptable bid has been made and confirmed, the bidding process can be ended and the participant with highest bid gets the role winner.

We extended this basic scenario in a number of ways to recommend to users only particular items for bidding which are of their interest. This is done by observing users' TV viewing history, content consumption behavior, personal preferences, etc. and mapping all this knowledge to user's profile ontology. We assume that various digital devices in particular user space (e.g., home) can exchange information through the Smart Space. Assuming that there are different TV stations and programs embedding their schedules on their WebPages using some common semantics for program description. TV can then recommend TV programs for particular user based on the user profile ontology. It can further recommend particular interactive applications based on the preferences and profile ontology. For example, consider a user who has been

watching Shakiras new video song and has liked her page on Facebook. The system can recommend him an iTV application of bidding for her latest album based on harvesting and querying his content consumption behavior and personal preferences given on social networks. It can add an event to the user's calendar when the auction will happen. At the time of the auction, the banking application on user's smartphone can check the account detail and user can decide based on the information if he has to join the auction. The system can identify different users and give recommendations according to particular user's profile. In this scenario, information between different sources and devices is communicated through the Smart Space. Smart Space allows the mesh-up of all this information to enable intelligent ambient iTV applications which are not limited to only TV.

This interactive application exhibits important properties which enable it to be modeled and developed using our ontology-driven Smart Space approach. Firstly, inferencing the user's preferences by semantically matching user's profiles with meta-data of the content provided by the content providers. This activates appropriate services for the user from the available resources. Secondly, heterogeneous devices could be used for interaction with the system making it a multi-device environment. Thirdly, the application is driven by user's actions and time-based events could also be used. Fourthly, subscriptions could be used in the situations where one action could be performed before any other action, e.g., after a bid is made, the amount of next bid should be raised.

All these properties make Smart Space an ideal choice for the development of such kind of interactive TV applications as Smart Space addresses the issues of reasoning, heterogeneous devices, interoperability, subscription based and user-driven actions. Our approach for application development provides higher level of abstraction by automatically generating ontology API from application ontology by mapping OWL ontology concepts into object-oriented programming language concepts. This enables application developers to create innovative Smart Space applications using traditional object-oriented programming concepts without worrying about the complexity of OWL ontologies.

4.2 Sequence diagram of the application scenario

Figure 2 depicts a sequence diagram illustrating the interaction between different modules of the architecture. It explains the operational flow and implementation of the afore-mentioned application scenario. The first step for any user is to register with the Smart Space using a unique id and the device profile. Only after registering he can consume and share information with other users through the Smart Space. Next, the user sends his profile information to be stored in the SIB in the form of profile ontology. It is required to consume the personalized contents that are of his interest. He can then request some content which is fetched from content provider using any channel, e.g., IPTV or broadcast. The content provider also send the content ontology which is stored in SIB. After the user gets the content, his profile ontology is updated to reflect the viewing history. Based on the reasoning of these ontologies, personalized advertisements and applications such as Auction of an item of user's interest are

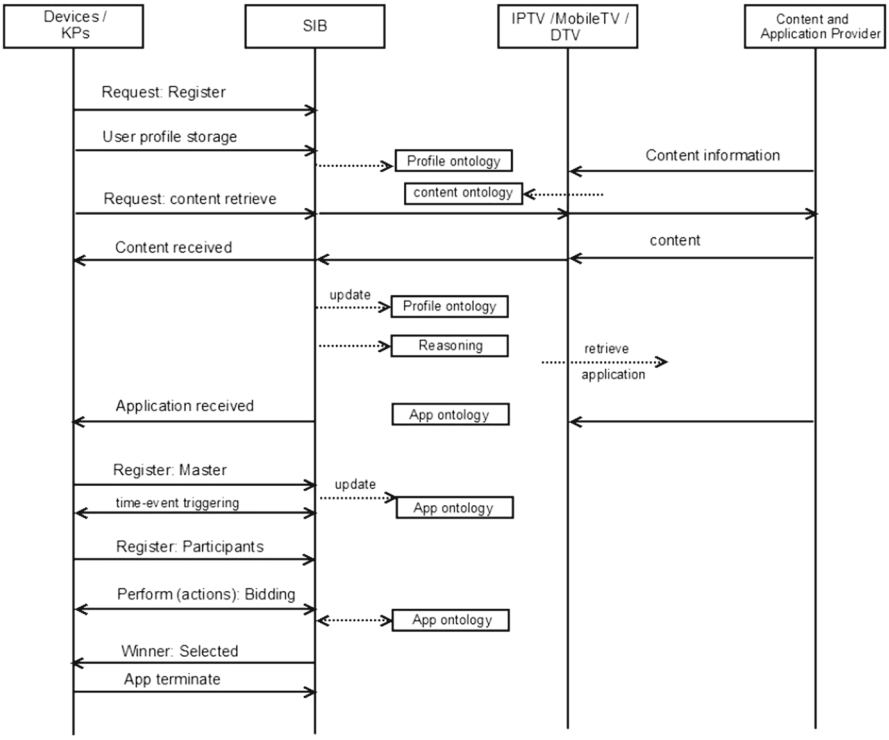

Fig. 2 Sequence diagram

suggested to the user. User can also start such as application if he wants to sell some item which is the case in our example scenario. The user registers as a seller to sell an item which updates application ontology in the SIB by inserting user's particulars.

5 Ontology-driven development (ODD) methodology

The pervasiveness in iTV applications increases the complexity of application development due to the extended context space. This requires development approaches based on the higher level of abstractions. Model-driven approach is promising as models are not only used for design, development, maintenance but also for generating executable code for specific applications and platforms. The main drawback of model driven approach for pervasive application development is its lack of support for reasoning tools. On the other hand, ODD follows similar approach as MDA using ontologies in Model driven engineering process but ODD additionally exploits knowledge exploitation using reasoning over the maintained ontology. There are several factors that make ODD a suitable choice for building pervasive software applications (Fig. 3).

– Languages for representing ontologies (OWL, etc.) are syntactically and semantically richer than common MDA approach of modeling in UML. UML models lack the formal semantics while ontologies are more explicit and precise.

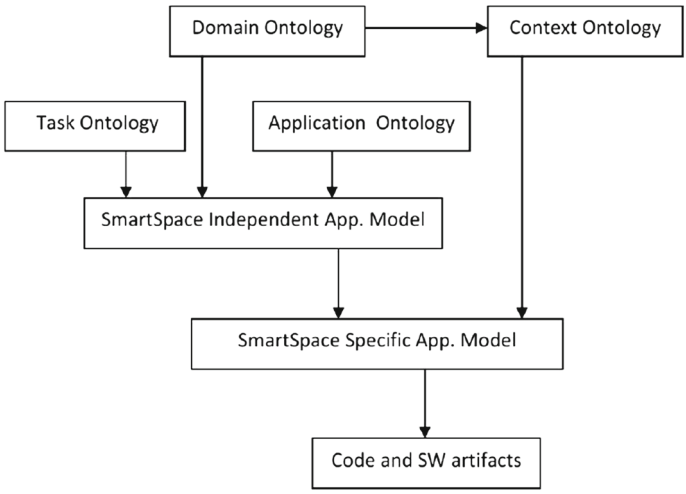

Fig. 3 Methodology

- Ontology-driven approach is theoretically found on logic. While ontology allows automated reasoning or inference, UML model does not.
- UML follows unique name assumption where same name always refer to the same object and different names refer to different objects. OWL, on the other hand, provides features to discipline names and two properties or two classes can be stated to be equivalent (equivalentClass, equivalentProperty).
- There are other developments related to OWL that are in progress, such as expressive rule language (SWRL, etc.) and OWL services.

In this section, we present an ontology-driven methodology for development of intelligent pervasive iTV applications.

Domain ontology: Domain ontology models a specific domain which represents part of the world. Particular meanings of the terms/concepts applied to the domain are provided by the domain ontology. Domain ontologies are computation independent and represent concepts of the particular domain in question. For example, in our example scenario, the domain is TV domain and the concepts in the domain such as viewers, program, etc. have particular meanings in this domain. In our proposed methodology, part of the domain ontology could be converted into Smartspace independent application model.

Context Ontology: Context ontology defines different users' context concepts, such as location, time, audience, etc. and is used for the reasoning purpose in combination with the standard user profile to improve content consumption and interaction experience. In the proposed methodology, the context ontology could be derived from domain ontology. This is because some portion of the domain ontology might be used for reasoning purpose. Context ontology characterizes the state and situation of a user and is important for personalized ambient services by taking into account the context of the user; for example, if the user is in the home or office while listening a particular song? are there other people in the room ? etc. Users can use manage, link and

synchronize available functionalities and behaviors of applications and the resources according to available context information.

Application Ontology: Application ontology describes the concepts and their relationships in the application; for example, item, participant, winner, calendar, etc. in our example AuctionTV application scenario.

Task Ontology: Task ontology specifies a library of different tasks that the devices provide to the users. For example, tasks and services provided by different devices, such as mobile phone, PVR, etc. in a smart room. The business logic of the application could also be defined using task ontology. In this case, the total behavior is the combination of this emerged behaviors defined in the task ontology.

Inference rules: The context inference process requires deterministic inference rules to infer new context. These rules are either general or domain specific.

SmartSpace Independent App. Model: With particular domain, task and application ontologies, a Smartspace independent Application model is created.

SmartSpace Specific App. Model: The SmartSpace independent application model is then converted to SmartSpace specific application model. The context ontology and inference rules are used to make the system into a particular setting of Smart Space. That is, based on the context of a user at a given situation, a particular setting of the Smart Space specific to that situation will be applied. For example, if a user enter to his smart house, the setting of different digital devices will be automatically set according to his preferences, such as room temperature, light settings, music, etc. These settings are specific for each user in that smart space, that is for the other person in the house.

Code and software artifacts: This module contains the corresponding artifacts at the low level. In our case, it would consists of the SIB which contains task ontology and context ontology.

The split in the proposed methodology gives more structured design and allow reusability of task, domain and context ontologies for other applications in that domain. The methodology has to be mapped to the underlying smart-M3 architecture. Such methodology provides higher level of abstraction and changes the physical environment into a programmable space in which users can mange, synchronize, link and consume accessible functionalities and behaviors of iTV application devices and services in the environment according to context information.

Using this methodology and the reasoning rules enables users to program their own environment to some extent which appears to be possible by having iTV applications and users sharing the same conceptual world model. Abstract rule language could be used to do business logic of the applications and enable and synchronize information flow between devices and iTV applications according to the contextual information.

6 Inference rules using PythonRules module

We have developed a Python Module for easy definition of rules in our approach for Smart-M3. The interaction with the SIB is made so that the Python developer does not have to deal with RDF triples or semantic technologies like query languages to access the central repository of shared information. The Python Rule module makes use of the

Ontology Library Generator (OWL to Python) and its framework as abstraction of the interface with the SIB. The *PythonRules* module allows the programmer to write rules on the fly, i.e., can be executed directly and interpreted as any other Python statement. Secondly, the module allows rules to be stored in the class *RuleSoup* which handles the whole set of rules and runs them when needed. For the first case, operators (// and $\gg$) were overloaded for rule syntax clarification and expressivity. In the second case, declaring a rule does not implies its execution and delays it until the programmer desires it by calling the method execute() of *PythonRule* class or *runAll()* from the *RuleSoup* class.

With Clause Class: The Class With represents the With Clause of the Rule and the first parameter of the PythonRule Class. It contains a list of Individuals which is required to be present on the Smart Space.

- *__init__(assumptions)*: Provides to the class With the individuals that appear later on the When and Then clauses in the same rule. All the individuals to be used on different clauses in the same rule must be added as input parameter to this class except the instances of objects which are created in the Then clause.
- *evaluate()*: Evaluates, without executing the rule, the With clause returning True if all of the individuals in With clause are different than None and they exist in the SIB, this is, if they have already been created in the Smart Space.
- *getWithIndividuals()*: Returns all the instances representing the individuals participating on the rule and that have been specified previously when creating the With clause.

When Clause Class: The Class When represents the When Clause of the Rule and the second parameter of the PythonRule Class. It contains a condition which is requirement to be satisfied for the rule to fire.

- *__init__(condition)*: Initializes the class When with one or several boolean conditional statements to be satisfied. If they are more than one condition, they must be expressed as a single one through Python regular boolean operators.
- *evaluate()*: Evaluates, without executing the rule, the When condition.
- *getWhenConditions()*: Returns the rule condition.

Then Clause Class: The Class Then represents the Then Clause of the rule and the third parameter of the PythonRule Class. It contains a list of Python actions to execute if the With and When clauses hold.

- *__init__(consequent)*: Initializes the class Then with one or a list of sentences to be executed if the rule condition holds. Note that they are not executed until execute() is called (unless the rule is created on the fly with the operators // and $\gg$). An example of statement could be, e.g., creating a new individual.
- *execute()*: Executes the provided statements.
- *getThen()*: Returns the rule actions (or consequent of the rule).
- *getReturnValues()*: Returns the values (if any, in case of need) returned by each of the statements included in the Then clause.

The execution of the rules is achieved through the subscription capability of the SSAP protocol to the *Smart Space*. This generates asynchronous notifications when

changes occur in the *Smart Space*. However, this is a concrete implementation of *Smart Space* broker (*Smart-M3*) but *PythonRules* module aims at being independent of the information broker or repository used. The class *Individual* wraps, for this purpose, the Ontology class corresponding to the Python class whose objects are used within the classes *When*, *With* and *Then*. *Individual* also hides, by means of its helper methods, the use of RDF queries and namespaces to the programmer, who only needs logic Python expressions, i.e., basically any Python expression plus the added value of the rule construction.

7 Implementation and evaluation

Our application development tools are used as follows:

1. *Smart-M3 Ontology to Python API Generator*: First of all, the ontologies to be used need to be converted automatically to their corresponding Python classes. For this purpose, our Ontology Library [3] is used, generating classes for each Ontology class together with their properties and methods.
2. *Programming Knowledge Processors*: When the Ontology Library has generated the needed classes with the included middleware, containing getters and setters

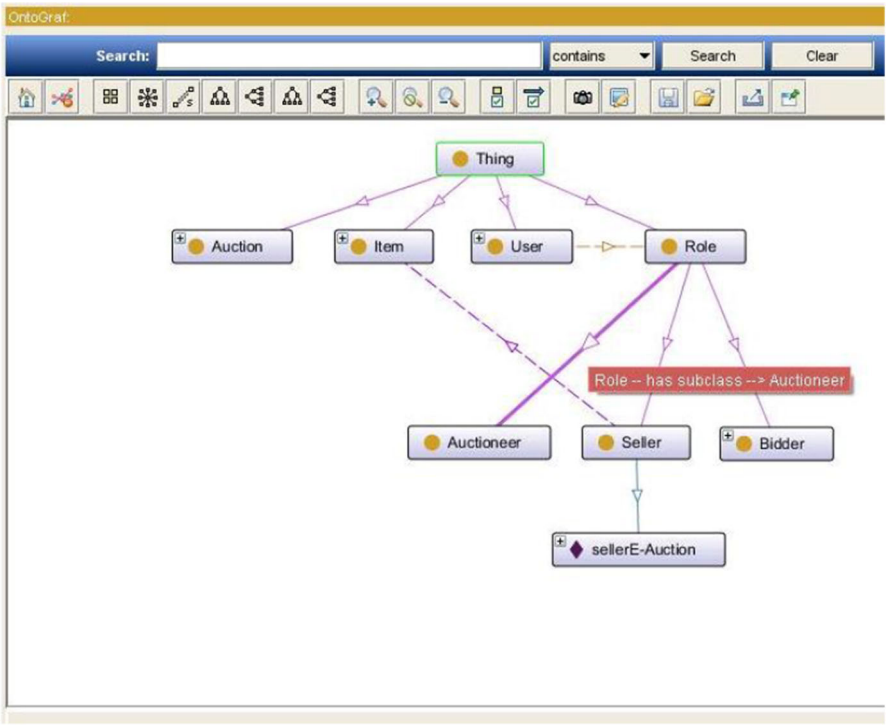

Fig. 4 Application ontology

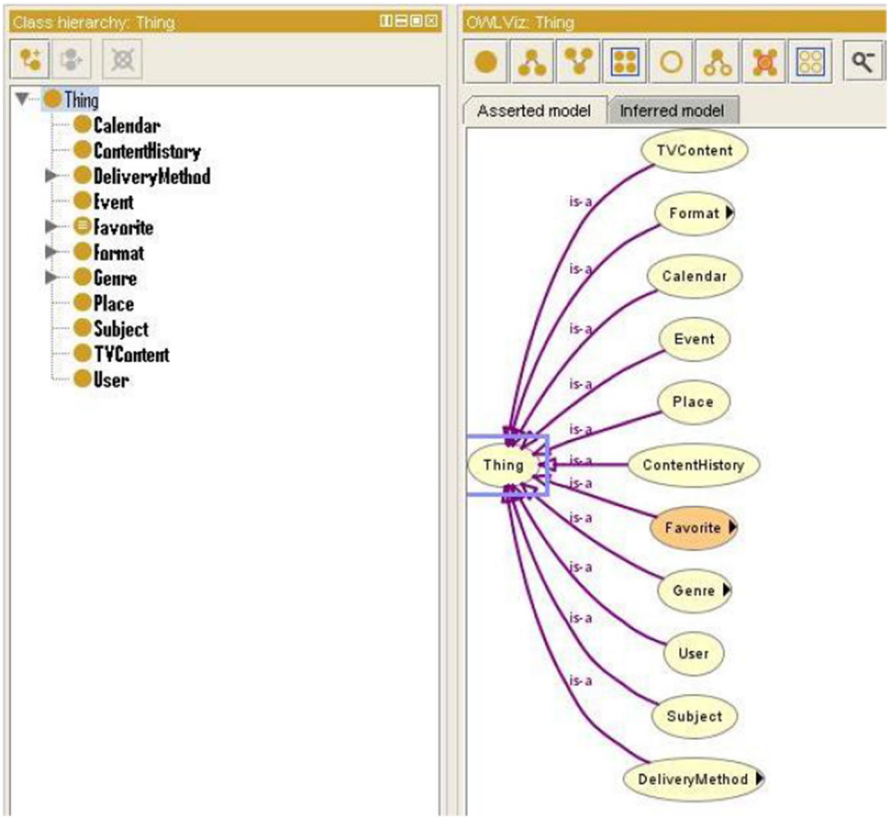

Fig. 5 iTV program ontology

methods, this middleware already abstracts the communication with the SIB allowing programming of KPs. The generated `EmptyKP.py` file can be used as a starting template; instance declarations automatically translate to RDF insertions into the SIB (after committing changes). This allows other applications connected to the same *Smart Space* to know about the existence of those individuals and to interact with them.

3. *Python Rules for Smart Space programming*: Since the previous middleware still requires a considerable number of calls before achieving interaction with the repository, as well as working with specific namespaces, *PythonRules* provides a higher abstraction layer for fast specification and configuration of the *Smart Space*'s behavior. needs to be imported.
   The rules can either be executed synchronously (when declared in real time) or stored together in the class *RuleSoup*. In the latter case, they can be run all at once and executed asynchronously (when their conditions are satisfied).

7.1 Ontology development

As our approach is based on ontology-driven application development, the first step is to create application and domain ontologies. We developed an application ontology for the application scenario described in Sect. 4. Figure 4 illustrates the excerpt from the application ontology. The ontology shows the semantic relationships between different concepts.

As we are dealing with the interactive TV domain in this particular scenario, the domain ontology consists of the concepts related to TV content, such as category, title, actors, schedule, etc. The domain ontology can be automatically generated using the meta-data available for each TV program. Figure 5 describes the example TV program ontology.

7.2 Programming knowledge processors: iTV use case

When the application programmer does not deal with RDF Triples directly, but mainly with logic Python ordinary statements, the translation of problems described with natural language into programs becomes much easier. This section describes two knowledge processors that are created for the evaluation of example scenario. The *TVBroadcasterKP* creates a new Calendar and a new Event. These knowledge processors use the APIs that are generated from the ontologies by our tool.

```
1  class TVBroadcaster_KP(KnowledgeProcessor):
2
3      def initialize(self):
4          self.registerOntology(CalendarOntology())
5          self.createUpdatedProgramEvent("Spanish people
               around the world: Finland")
6          #self.createNewCalendar("Natalia's Calendar")
7
8      def createNewCalendar(self, title):
9          googleCalendar = Calendar()
10         googleCalendar.setTitle(XSDString(title))
11
12     def createUpdatedProgramEvent(self,title):
13         event = Event()
14         event.setTitle(XSDString(title))
15         event.setDtStart(XSDDateTime(datetime(2012, 8, 15,
               17,0,0)))
16         print "BBC Broadcaster just added a calendar event
               with updated programme"
17
18 def main(args):
19     app = QtGui.QApplication(sys.argv)
20     smartSpace = ('x', (TCPConnector, ('127.0.0.1', 10010)
           ))
21
22     kp = TVBroadcaster_KP.create(smartSpace)
23     sys.exit(app.exec_())
```

Listing 1 Knowledge processor for TV Program

The *AuctionItemsManagerKP* is an example Class that creates new Items for sale at an Auction.

```
class AuctionItemsManager_KP(KnowledgeProcessor):

    def initialize(self):
        self.registerOntology(AuctionOntologyOntology())
        self.addNewItemForSale("Cool100in1")

    def addNewItemForSale(self,name):
        event = Item()
        event.setItemName(XSDString(name))
        event.setDateOfStart(XSDDateTime(datetime(2012, 8,
            17, 17,0,0)))
        #event.setHasStartingPrice(XSDInteger(200))
        print "e-Auction Items Manager just added a new
            item for sale: ",name


def main(args):
    app = QtGui.QApplication(sys.argv)
    smartSpace = ('x', (TCPConnector, ('127.0.0.1', 10010)
        ))

    kp = AuctionItemsManager_KP.create(smartSpace)
    sys.exit(app.exec_())
```

Listing 2 Knowledge Processor for Auction Application

7.3 Inference rules

Straight after the KP is created, the developers could define the Python rules related to the existing KPs. Then, connect the KPs to the Smart Space and run them is the only thing left.

If `EmptyKP.py` (provided by the Ontology-Python Generator) is used, instance declarations will automatically translate to insertions of Triples into the SIB. This allows other KP applications connected to the same Smart Space to know about those individuals' existence to interact with them. In the Python Rules Module, every KP application contains a `TripleStore` instance (produced by Ontology-Python Generator) representing the Smart Space' SIB. At last, the *With()*, *When()* and *Then()* clauses translate its Python statements to one of the implementation options given by the Ontology-Python Generator. These are SIB calls in *RDF* or *WQL* language. Our approach shows that learning OWL or query languages is not needed for interconnections with the SIB and interactions with other devices' KPs.

We created two simple rules for the evaluation purpose.

Rule 1: If in an electronic auction, the new gadget Cool100in1 is offered for sale, Natalia would like to be notified as soon as this item appears in the Auction. If this occurs, an event on her calendar should be created immediately to remind her to bid. The particular implementation of this rule using our PythonRule module is given in the following listing:

```
withClause = With([newGadgetItem])
whenClause = When(newGadgetItem.getProperty("ItemName") ==
     "Cool100in1")
thenClause = Then([remindToBidEvent.new(Event),
   remindToBidEvent.setProperty(Title = XSDString("
       Cool100in1 in e-Auction, Remember to Bid!")),
   remindToBidEvent.setProperty(DtStart = XSDDateTime(
       newGadgetItem.getProperty("HasDateOfStart"))),
   remindToBidEvent.setObject("MemberOf", nataliaCalendar.
       get()),
   GoogleCalendar("smartspacecalendar@gmail.com", "
       smartspace").addEvent("Remember to Bid for
       Cool100in1!","","e-Auction", dayBefore(XSDDateTime(
       newGadgetItem.getProperty("DateOfStart"))),
                                dayBefore1hourAfter(
       XSDDateTime(newGadgetItem.getProperty("DateOfStart")
       )), None)])

rule = PythonRule(withClause, whenClause, thenClause)
```

Rule 2: If there is a new event in the broadcaster calendar which includes Natalia's favorite documentary, Spanish people around the world, happening in Finland, she would like to be notified on her calendar not to miss it.

```
withClause = With([favouriteDocumentaryEvent])
whenClause = When(favouriteDocumentaryEvent.getProperty("
    Title") == "Spanish people around the world: Finland")
thenClause = Then([remindDocumentaryEvent.new(Event),
    remindDocumentaryEvent.setProperty(Title = XSDString("
        Spanish people around the world in Finland!")),
    remindDocumentaryEvent.setProperty(DtStart =
        XSDDateTime(favouriteDocumentaryEvent.getProperty("
        DtStart")-oneDay)),
    remindDocumentaryEvent.setObject("MemberOf",
        nataliaCalendar.get()),
    GoogleCalendar("smartspacecalendar@gmail.com", "
        smartspace").addEvent("Tomorrow is your favourite
        documentary","","BBC Broadcaster", dayBefore(
        XSDDateTime(favouriteDocumentaryEvent.getProperty("
        DtStart")))), dayBefore1hourAfter(XSDDateTime(
        favouriteDocumentaryEvent.getProperty("DtStart"))),
         None)])

rule = PythonRule(withClause, whenClause, thenClause)

# Running the whole RuleSoup...
ruleSoup = RuleSoup()
ruleSoup.addRule(rule)
ruleSoup.runAllRules()
# Waiting for creation of new Events...
sys.exit(app.exec_())
```

The rules are application specific and a set of rules is to be specified for each domain. Based on the context information from multiple context dimensions, the system triggers when a change happens and the associated rules are activated to infer new context information. The underlying implementation of the Python Rules Module translates Python logic expressions to the SIB API main interface: *Query, Subscribe,*

Insert, Remove, Update. Thus, the Python Rules Module just needs to be imported to be used with the KP class where the DIEM application is coded.

8 Discussion

For this use case, we first developed the ontologies in Protege and then these ontologies are fed as input to our tool to generate ontology libraries. We then implemented the knowledge processors that reflect the functionality of the application. Knowledge processors use the generated ontology APIs. We then defined the rules using Python-Rules Module which describes the rule expressions embedded into Python language.

Our approach for ontology-driven iTV application development worked well for these simple rules. We aim to extend it to evaluate more complex scenarios and context-aware iTV applications.

9 Conclusions and future work

In this paper, we have presented how to develop interactive TV application using ontology-driven Smart Space approach. We have also demonstrated the suitability of our rule-based approach to support a highly dynamic context-aware service that includes reasoning for situation detection. We have developed a context-aware service in the domain of interactive TV and evaluated it using our developed ontology-driven tools. Future work includes development of more complex application scenarios that could benefit from pervasiveness. Moreover, performance and scalability of the approach will be accessed by increasing the number of entities, number of events generated and the number of rules. For this purpose, evaluation parameter *reaction time* will be defined. The reaction time will consist of the processing time that takes place between an event occurrence, and the invocation of the action, i.e., the time between the occurrence of an event, and triggering a rule associated with that.

Acknowledgments The research work presented in this paper is based on DIEM project and the authors would like to acknowledge all the partners of this project.

References

1. European Broadcasting Union: multimedia homepage of the European Broadcast Union (EBU), Geneva, Switzerland. http://www.ebu.ch/en/multimedia/index.php
2. British Broadcasting Corporation: enhanced TV formats. http://www.bbc.co.uk/commissioning/interactive/
3. Smart-M3 software at sourceforge.net, release 0.9.4beta, May 2010. [Online]. http://sourceforge.net/projects/smart-m3/
4. Kaustell a, saleemi MM, Rosqvist T, Jokiniemi J, Lilius J, Porres I (2011) Framework for smart space application development. In: Proceedings of the international workshop on semantic interoperability, IWsI
5. Oliver I, Honkola J (2008) Personal semantic web through a space based computing environment. In: Proceedings of the 2nd international conference on semantic, computing

6. Papadimitriou A, Anagnostopoulos C, Tsetsos V, Paskalis S, Hadjiefthymiades S (2007) A semantics-aware platform for interactive tv services. In: In the Proceedings of the 1st international conference on new media technology (I-MEDIA 07), Graz, Austria
7. Blanco-Fernandez Y, Pazos-Arias JJ, Gil-Solla A, Ramos-Cabrer M, Barragans-Martinez B, Lopez-Nores M, Garcia-Duque J (2004) Avatar: an advanced multi-agent recommender system of personalized tv contents by semantic reasoning. In: Web information systems WISE 2004
8. Bellekens P, van der Sluijs K, Aroyo L, Houben G-J, Kaptein A (2007)Semantics-based framework for personalized access to TV content: personalized TV guide use case
9. Hopfgartner F, Jose J (2010) Semantic user profiling techniques for personalised multimedia recommendation. Multimed Syst 16(4–5):255–274
10. Pleu A, Van den Bergh J, Humann H, Sauer S (2005) Model driven development of advanced user interfaces. In: CEUR workshop proceedings, vol 159. http://ceur-ws.org/Vol-159
11. Pleuss A, Gračanin D, Zhang X (2011) Model-driven development of interactive and integrated 2D and 3D user interfaces using mml. In: Proceedings of the 16th international conference on 3D web technology, pp 89–92. ACM, New York
12. den Bergh JV (2010) Model-driven development of advanced user interfaces. In: 28th of the international conference on human factors in computing systems (CHI). ACM, New York
13. Eisenstein J, Vanderdonckt J, Puerta A (2001) Applying model-based techniques to the development of uis for mobile computers. In: IN IUI 2001 international conference on intelligent user interfaces. ACM Press, New York, pp 69–76
14. Coyette A, Kieffer S, Vanderdonckt J (2007) Applying model-based techniques to the development of uis for mobile computers. In: In human–computer interaction INTERACT 2007, p 150
15. Van Den Bergh J, Bruynooghe B, Moons J, Huypens S, Handekyn K, Coninx K (2007) Model-driven creation of staged participatory multimedia events on TV. In: Proceedings of the 5th European conference on interactive TV: a shared experience, EuroITV'07. Springer, Berlin, pp 21–30