Ontology-driven Development of interactive TV Applications using Smart Space Approach

M. Mohsin Saleemi, Natalia Diaz Rodriguez, Johan Lilius

[1]Department of Information Technologies,
Åbo Akademi University, Turku, Finland
[2]Turku Centre for Computer Science (TUCS)
email: (msaleemi, ndiaz, jlilius) @abo.fi

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 applications using semantic technologies are not realized yet. In this work, we explore the potential of introduction of semantic technologies and smart spaces in design and development of interactive TV (iTV) applications. We use an example scenario to show how future iTV applications include the combination 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.

1 Introduction

With the advent of new standards and technologies for transmission, development and execution environments, interactive TV is evolving 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 means different things to different people and no single definition is presently accepted by all researchers. European Broadcasting Union defines iTV applications as enhanced or interactive services with digital Television [2]. 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 [1]. 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 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. are using their tools for 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. In order 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 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 interact 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 this data could be shared be-

tween heterogeneous devices present in the smart space. Smart space application development tools could be used to develop interoperable interactive TV applications that employ mixture 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 [4] [8] for rapid application development in Smart Spaces and it is based on the open source Smart-M3 architecture [9]. 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 cover 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 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.

- 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.

We propose to use ontology driven iTV applications development because

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 [8] [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) reduce 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 a coordinated efforts from multiple organizations and research groups towards inclusion of semantics in interactive TV services and platforms. Work presented in [10] describes a semantics-aware platform for interactive TV services in order 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 [15] outlines video annotation technique, ontology-based modeling, multimedia metadata and user profiling through semantic reasoning. The main goal of this work is to create a personalized digital TV recommender based on metadata. Other work in the direction of personalized TV programs recommendation systems based on semantics include [7], [11]. 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 [12]. Work presented in [13] and [3] describes approaches for model-driven development of interactive user interfaces. similarly in researches like [6] and [5] which applies 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 Enhanced iTV Application Scenario

We chose AuctionTV example scenario given in [14] 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 number of ways to recommend 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 users 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 by 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 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 users calendar when the auction will happen. At the time of the auction, the banking application on users 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 users profile. In this scenario, information between different sources and devices is communicated through the smart space. Smart space allows the fusion 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 metadata 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.

4.1 System Architecture

In this section, the overall system architecture is outlined for the application described in the previous section. Figure 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 metedata. This modules require that the information on TV content and advertisement must be defined using some standard for representing metadata

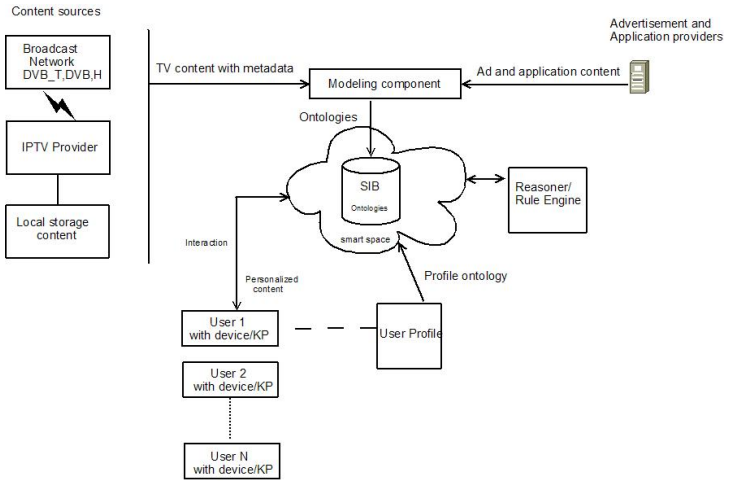

Fig. 1. System structure

such as MPEG-7. The metadata for TV content includes 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 SIB and reasoning engine. This infrastructure is the core component which used to store the ontologies and provide reasoning. It also facilitate interaction and communication with the user devices through KPs for sharing and updating information between 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 (SIB) to share information between them.

5 Ontology-driven Development Methodology

The pervasiveness in iTV applications increases the complexity of application development due to the extended context space. This requires development approaches based on 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, ontology driven development (ODD) follows similar approach as MDA by 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 ontology-driven development a suitable choice for building pervasive software applications.

– 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.

– 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 Smart space independent application model.

Context Ontology: Context ontology defines different user's context concepts such as location, time, audience, etc and 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.

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

Smart Space Specific App. Model: The Smart Space independent application model is then converted to Smart Space 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.

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 enable users to program their own environment to some extent which appear be possible by having iTV applications and users sharing 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 Semantic Information Broker (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 >>) 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 are requirement 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 >>). 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 [4] 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 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.

   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 section IV. Figure 2 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 consist of the concepts related to TV content such as category, title, actors, schedule etc. The domain ontology can be automatically generated using the metadata available for each TV program. The figure 3 describes the an example TV program ontology.

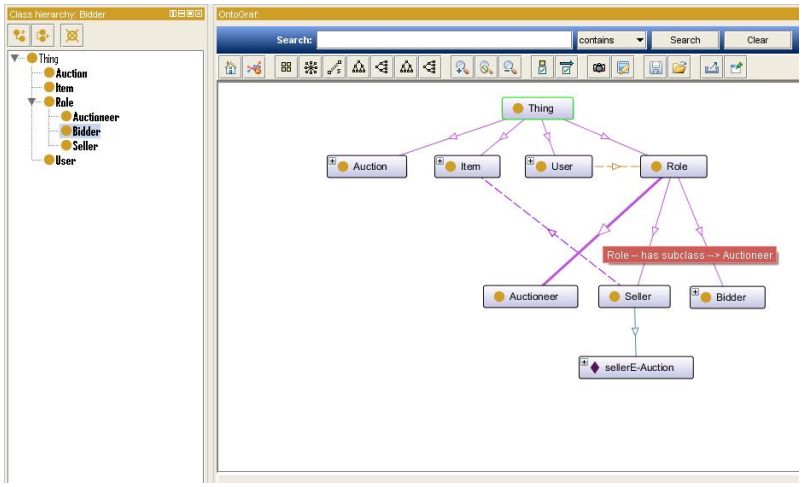

Fig. 2. Application Ontology

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
7     def createNewCalendar(self, title):
8         googleCalendar = Calendar()
9         googleCalendar.setTitle(XSDString(title))
10
11    def createUpdatedProgramEvent(self,title):
12        event = Event()
13        event.setTitle(XSDString(title))
14        event.setDtStart(XSDDateTime(datetime(2012, 8, 15,
              17,0,0)))
15        print "BBC Broadcaster just added a calendar event
              with updated programme"
16
17 def main(args):
```

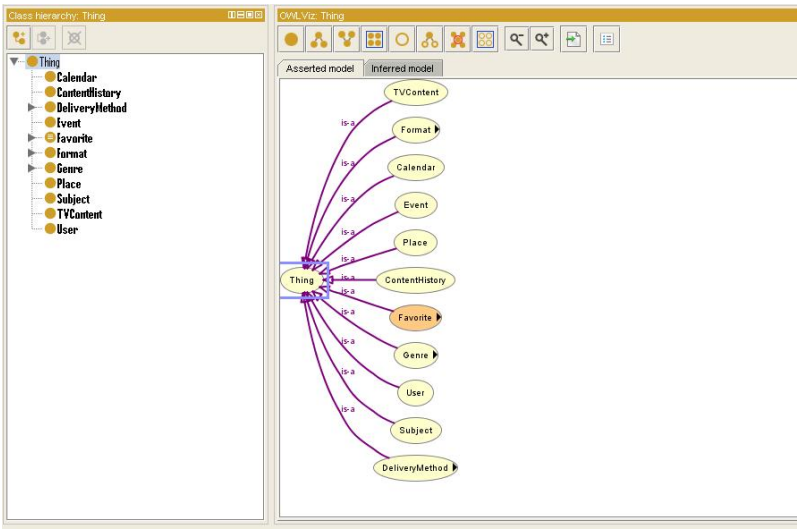

Fig. 3. iTV Program Ontology

```
18      app = QtGui.QApplication(sys.argv)
19      smartSpace = ('x', (TCPConnector, ('127.0.0.1', 10010)))
20
21      kp = TVBroadcaster_KP.create(smartSpace)
22      sys.exit(app.exec_())
```

Listing 1.1. Knowledge processor for TV Program

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

```
24 class AuctionItemsManager_KP(KnowledgeProcessor):
25
26      def initialize(self):
27          self.registerOntology(AuctionOntologyOntology())
28          self.addNewItemForSale("Cool100in1")
29
30      def addNewItemForSale(self,name):
31          event = Item()
32          event.setItemName(XSDString(name))
33          event.setDateOfStart(XSDDateTime(datetime(2012, 8,
                17, 17,0,0)))
34          #event.setHasStartingPrice(XSDInteger(200))
35          print "e-Auction Items Manager just added a new item
                for sale: ",name
36
37
38 def main(args):
39      app = QtGui.QApplication(sys.argv)
```

```
40     smartSpace = ('x', (TCPConnector, ('127.0.0.1', 10010)))
41
42     kp = AuctionItemsManager_KP.create(smartSpace)
43     sys.exit(app.exec_())
```

Listing 1.2. Knowledge Processor for Auction Application

7.3 Inference Rules

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 listing 7.3

```
45 withClause = With([newGadgetItem])
46 whenClause = When(newGadgetItem.getProperty("ItemName") == "
       Cool100in1")
47 thenClause = Then([remindToBidEvent.new(Event),
48     remindToBidEvent.setProperty(Title = XSDString("Cool100in1
           in e-Auction, Remember to Bid!")),
49     remindToBidEvent.setProperty(DtStart = XSDDateTime(
           newGadgetItem.getProperty("HasDateOfStart"))),
50     remindToBidEvent.setObject("MemberOf", nataliaCalendar.get
           ()),
51     GoogleCalendar("smartspacecalendar@gmail.com", "smartspace
           ").addEvent("Remember to Bid for
                                        Cool100in1!","","e-
           Auction", dayBefore(XSDDateTime(newGadgetItem.
           getProperty("DateOfStart"))),
                                    dayBefore1hourAfter(
           XSDDateTime(newGadgetItem.getProperty("DateOfStart")))
           , None)])
52
53 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.

```
55 withClause = With([favouriteDocumentaryEvent])
56 whenClause = When(favouriteDocumentaryEvent.getProperty("
       Title") == "Spanish people around the world: Finland")
57 thenClause = Then([remindDocumentaryEvent.new(Event),
58     remindDocumentaryEvent.setProperty(Title = XSDString("
           Spanish people around the world in Finland!")),
59     remindDocumentaryEvent.setProperty(DtStart = XSDDateTime(
           favouriteDocumentaryEvent.getProperty("DtStart")-
           oneDay)),
```

```
60      remindDocumentaryEvent.setObject("MemberOf",
            nataliaCalendar.get()),
61      GoogleCalendar("smartspacecalendar@gmail.com", "
            smartspace").addEvent("Tomorrow is your favourite
                                   documentary","","BBC
            Broadcaster", dayBefore(XSDDateTime(
            favouriteDocumentaryEvent.getProperty("DtStart"))),
                       dayBefore1hourAfter(XSDDateTime(
            favouriteDocumentaryEvent.getProperty("DtStart"))),
            None)])
62
63  rule = PythonRule(withClause, whenClause, thenClause)
64
65  # Running the whole RuleSoup...
66  ruleSoup = RuleSoup()
67  ruleSoup.addRule(rule)
68  ruleSoup.runAllRules()
69  # Waiting for creation of new Events...
70  sys.exit(app.exec_())
```

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 PythonRules 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 evaluated 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.

Acknowledgment

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. British broadcasting corporation: Enhanced tv formats: http://www.bbc.co.uk/commissioning/interactive/.
2. European broadcasting union: Multimedia homepage of the european broadcast union (ebu), geneva, switzerland. http://www.ebu.ch/en/multimedia/index.php.
3. Model-driven development of advanced user interfaces.
4. Smart-M3 software at sourceforge.net, release 0.9.4beta, May 2010. [Online]. Available: http://sourceforge.net/projects/smart-m3/.
5. S. K. A. Coyette and J. Vanderdonckt. Applying model-based techniques to the development of uis for mobile computers. In *In Human-Computer Interaction INTERACT 2007*, page 150, 2007.
6. J. Eisenstein, J. Vanderdonckt, and A. Puerta. Applying model-based techniques to the development of uis for mobile computers. In *IN IUI 2001 International Conference On Intelligent User Interfaces*, pages 69–76. ACM Press, 2001.
7. F. Hopfgartner and J. Jose. Semantic user profiling techniques for personalised multimedia recommendation. *Multimedia Systems*, 16(4-5):255–274, 2010.
8. A. Kaustell, M. M. Saleemi, T. Rosqvist, J. Jokiniemi, J. Lilius, and I. Porres. Framework for Smart Space Application Development. In *Proceedings of the International Workshop on Semantic Interoperability, IWSI*, 2011.
9. I. Oliver and J. Honkola. Personal semantic web through a space based computing environment. In *Proceedings of the 2nd International Conference on Semantic Computing*, 2008.
10. A. Papadimitriou, C. Anagnostopoulos, V. Tsetsos, S. Paskalis, and S. Hadjiefthymiades. S.: 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*, 2007.
11. L. A. G.-J. H. Pieter Bellekens, Kees van der Sluijs and A. Kaptein. Semantics-based Framework for Personalized Access to TV Content: Personalized TV Guide Use Case.
12. V. d. B. J. H. H. S.-S. Pleu, A. Model driven development of advanced user interfaces.
13. A. Pleuss, D. Gračanin, and X. Zhang. 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*, pages 89–92. ACM, 2011.
14. J. Van Den Bergh, B. Bruynooghe, J. Moons, S. Huypens, K. Handekyn, and K. Coninx. 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, pages 21–30, Berlin, Heidelberg, 2007. Springer-Verlag.
15. A. G.-S. M. R.-C. B. B.-M. M. L.-N. J. G.-D. Yolanda Blanco-Fernndez, Jos J. Pazos-Arias. Avatar: An advanced multi-agent recommender system of personalized tv contents by semantic reasoning. In *Web Information Systems WISE 2004*.