A Model-Driven Approach for Incorporating Reactive Rules in Declarative Interactive TV Applications

Patrícia Dockhorn Costa, João Paulo A. Almeida, Igor Magri Vale, Izon Thomaz Mielke
Computer Science Department, Federal University of Espírito Santo (UFES)
Av. Fernando Ferrari, s/n, Vitória, ES, Brazil
pdcosta@inf.ufes.br, jpalmeida@ieee.org, igormagrivale@gmail.com, izontm@gmail.com

Abstract— **This paper discusses an approach for incorporating ECA rules in interactive TV applications described in the multimedia declarative language adopted for the Brazilian Digital TV (SBTVD) platform. We discuss the limitations which are inherent in this platform and instead of proposing non-conformant extensions to the platform, we propose a model-driven approach which takes a rule model at a high level of abstraction and transforms it into the multimedia declarative language used in the Digital TV platform. This ensures the inclusion of context-aware rules into interactive TV applications such that they can be reactive to the user's context and alter their behavior according to rules referring to events and conditions from the user's environment. We illustrate the approach with a running example that considers contextual information to selectively display certain private information depending on presence detected by context-sources connected to the set-top box.**

Reactive rules; model-driven approach; interactive television; context-awareness;

I. Introduction

Context-aware applications use and manipulate context information to detect the situations of their users and adapt application behaviour accordingly. Since context-awareness allows applications and services to become tailored to the user's current situation and needs, several areas of research consider context-awareness as an important tool to enhance user experience [7, 13].

In the interactive television domain, in particular, context information has been used in various types of applications (see, e.g., [2, 4]), ranging from TV content adaptation (based on user's preferences) to home banking applications (selectively displaying certain private information based on presence). In a digital television application, for example, context information can be used to detect presence of children in the living room, which can be used to change the behavior of the TV to display "child safe" content.

Given the reactive nature of context-aware applications and the flexibility that rule-based solutions offer, it has been argued that context-aware application behaviors can be successfully described as a composition of reactive rules (see, e.g., [5, 8, 15]). In addition, rule-based approaches for context-aware applications have been considered applicable both at a (platform-independent) specification level and at a (platform-specific) implementation level [8]. At the implementation level, native platform support for rules enables one to reflect the overall structure of the specification at the implementation level. While this is beneficial with respect to traceability and simplicity of the design trajectory, direct support for rules is required of the target platform.

In this work, we address the problem of transforming a rule-based specification at the platform-independent level into an implementation platform that is not rule-oriented: the Brazilian Digital TV (SBTVD) platform. We propose a model-driven approach which takes an ECA rule model at a high level of abstraction and transforms it into the declarative language supported by the SBTVD platform [1].

Our aim is to ensure the inclusion of context-aware rules into digital TV applications such that they can be reactive to the user's context, while refraining from affecting the platform standard. This is particularly important considering that the standard is adopted in its current form by the Brazilian TV industry and is soon to be shipped in set-top boxes and TV sets. We illustrate the approach with a running example that considers contextual information to selectively display certain private information depending on presence detected by context-sources connected to the set-top box.

The ECA rule modeling foundations we employ have been (partially) discussed in previous works [8, 11, 10], in which we have proposed a comprehensive rule-based approach to support context-aware application development. As part of this approach, we have defined a domain specific language, coined ECA-DL, to support the definition of reactive behaviours in the scope of context-aware applications. This paper introduces ECA-DL TVD, a dialect of ECA-DL to allow complete implementation of such rules for interactive TV applications.

II. Application Scenario

We consider a home banking application, which allows users to check their bank account information using an interactive TV application. We have focused on solving a major privacy issue in such application: displaying private bank account information on the TV while others are also present. We propose that the context-aware application should detect presence to automatically omit private information when more than one person is in front of the TV.

The storyline for the application scenario is as follows: John uses an internet banking application through the TV. Every time he wants to check his bank account, he sits back on his couch and with the help of the remote control he logs into his bank account and performs the usual tasks. When someone else enters the room, the banking application automatically hides selected private information, such as his account balance. Everything goes back to normal when John is alone in the room again.

A. Modeling the Scenario

In our approach, application development starts with the definition of a context conceptual model, defining the relevant entity and context types. This context conceptual model defines the basic vocabulary to be used in the subsequent definition of (potentially complex) situations, events and context conditions for rule-structure behavior.

Figure 1 depicts the conceptual model that represents the entity and context types relevant to the scenario proposed. We use a UML class diagram with stereotypes to denote the classification of concepts with respect to the foundational distinctions proposed in [8]. In this model, an «Entity» is that which is said to exist independently of other entities. The entity types considered in this scenario are *Place*, *Person*, and *Account*. Context, on the contrary, is existentially dependent on entities. The context types considered in this model are *Presence* and *Access*, both of which are categorized as «RelationalContext», as they establish the relation between entities. *Presence* relates *Person* and *Place,* capturing the situation in which a *Person* is presently in a particular *Place*; *Access* relates *Person* and *Account*, capturing the situation in which a *Person* is logged into an *Account* for a home banking session.

In addition to modeling entity and context types, it may be necessary to model *situations*, which represent particular state-of-affairs that are of interest to applications [11]. Situations build upon context models since they can be composed of more elementary kinds of context conditions, and in addition can be composed of existing situations themselves. Examples of situations that are of interest to the scenario proposed are "there is more than one person in the room" and "there is one person in the room" (necessary to decide whether to display private account information or not). Both situations may be defined using the *Presence* «RelationalContext».

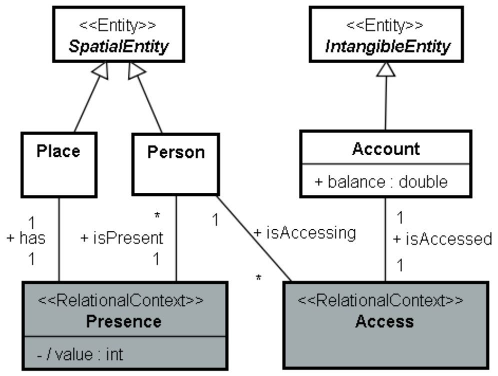

Figure 1. Context model

III. ECA-DL TVD

The next step in the design trajectory is the definition of rules for application behavior using a rule-based domain-specific language, called ECA-DL TVD. This language is a dialect of ECA-DL, which some of us have proposed in earlier work [8] and is specialized for specifying the reactive behavior of context-aware interactive TV applications.

An ECA-DL TVD rule is consistent with the Event-Condition-Action (ECA) architectural pattern, which provides a high-level structure for applications that react to changes in context. An ECA-DL rule is composed of an event clause, which defines relevant changes of situations through simple or complex combinations of events; an action clause, which defines the response of the context-aware application in terms of service invocations, and; a condition clause, which defines a logical expression that must hold for the action to be executed.

These elements are identified by the Upon, Do and When keywords, respectively. An <uponExpression> defines a combination of events, which may be TV events or situation events. TV events are considered primitive events such as the pausing of a video or a user click on the remote control. Situation events, in contrast, are defined in terms of situation transitions: an EnterTrue event occurs in the moment in which a situation begins to hold, and an EnterFalse event occurs when the situation ceases to hold.

The occurrence of the event (or combination of events) specified in the Upon clause triggers evaluation of the When clause. The actions described in the Do clause will be invoked only when the event in the Upon clause occurs and the conditions in the When clause are met (i.e., the <conditionExpression> evaluates to true).

Finally, an <actionExpression> consists of an invocation of a list of services. These services can be general services such as sending an SMS and TV services, such as pausing of a video.

In the case of the application scenario of this work, we can define four ECA-DL TVD rules to define the reactive behavior of the application. These rules are depicted in Figure 2.

The first rule is triggered whenever the user logs into his/her bank account and there is only one person presently in the room (presumably the user himself). This rule results in showing the balance information (the "regular" application behavior). Conversely, the second rule is triggered whenever the user logs into his/her bank account and there is more than one person presently in the room. This rule results in hiding the account's balance (the "adapted" application behavior). The third and fourth rules ensure that the account's balance remains shown or hidden reflecting the number of persons in the room. Finally, when there is no person in the room, any activity with the account must cease, which is represented by the fifth rule.

The situations referred to in the clauses of Figure 2 can also be defined by ECA-DL TVD rules. In this case, actions are used to state when a situation starts to hold (do EnterTrue) and when it ceases to hold (do EnterFalse).

```
Upon EnterTrue SituationAccessing (person1)
When SituationOnePersonInRoom (room1)
Do showBalance()

Upon EnterTrue SituationAccessing (person1)
When SituationMoreThanOnePersonInRoom (room1)
Do hideBalance()

Upon EnterTrue SituationOnePersonInRoom (room1)
Do showBalance()

Upon EnterTrue SituationMoreThanOnePersonInRoom
(room1)
Do hideBalance()

Upon EnterTrue SituationNoPersonInRoom (room1)
Do logout()
```

Figure 2. ECA-DL TVD Rules for the Banking Application scenario

IV. TARGET PLATFORM & APPLICATION ARCHITECTURE

The Brazilian Digital TV System (SBTVD) implements a middleware platform, coined Ginga [1], which supports the development and execution of interactive applications in the SBTVD environment. Interactive applications can be developed upon the Ginga middleware using a declarative language, called NCL (Nested Context Language), in combination with Lua scripts (called, in this scope, NCLua).

NCL is a hypermedia authoring language that was originally developed to describe multimedia applications with space-time synchronization between media objects (e.g., video, audio, images, etc.). As such, it does not have built-in concepts for context-handling and reactivity. In order to implement these concepts in NCL we have identified a number of NCL document elements, which together with certain NCLua scripts provide the required support.

In particular, *capturing context from remote sensors* can be realized by means of a Ginga component called Interactivity Channel, which allows an NCL application to communicate with remote devices over a network. This is a general communication framework, which we have leveraged to provide a programming framework that supports capturing and distributing context events. This framework contains a number of reusable NCLua scripts and is available online at [http://code.google.com/p/itveventframework/]. The NCLua scripts for capturing context alter the so-called Property Anchors in an NCL document, thereby introducing *context information* in the scope of the NCL application.

Further, *processing rules* (both situation detection rules and reaction rules) can be realized in a declarative NCL document by means of the Connector element, which allows the specification of causality relations. Causality relations establish a cause and effect relationship between NCL elements (called medias). Conceived originally to represent causality relations between multimedia elements (e.g., establishing that a certain audio should be followed by a certain video), we have used these to specify relations between context conditions and actions, which can result in the invocation of NCLua scripts, the assignment of values to property anchors and the starting/stopping of videos.

Finally, *reacting upon context* can be realized by transformation of a general service action (in a Do clause) to NCL code. This transformation checks an action's repository, which defines whether the actions can be performed by invoking external services or whether the services can be translated directly into a specific NCL simpleAction element. The actions showBalance() and hideBalance() in our running example are instances of the latter case. These actions are implemented by simpleActions which set a property anchor that determines whether the balance should be displayed.

The high-level architecture of a context-aware TV application is depicted in Figure 3 (instantiated for our application scenario). The TV Application (deployed on a set-top box) is an NCL application, which is composed of an NCL document in combination with NCLua scripts. The scripts realize the communication (over a network) between the NCL application and the Presence Sensor (Context Source), as well as between the NCL application and the Bank Service.

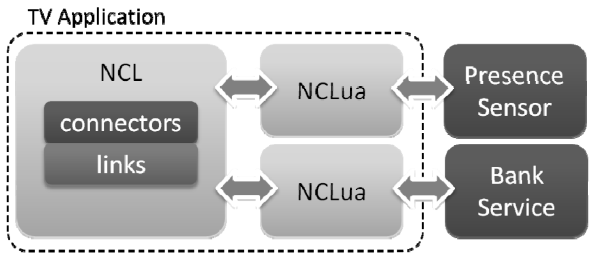

Figure 3. Application high-level architecture

We have implemented the Presence Sensor that detects the number of persons facing the TV screen by means of a camera and face recognition using an OpenCV library, which provides an algorithm for object detection [12]. Presence information is generated by the Presence Sensor and observed by the NCLua script, which keeps the corresponding NCL document property anchors up-to-date with respect to newly arrived presence information.

A model transformation is used to generate the NCL documents and NCLua scripts from Context Models and ECA-DL TVD rules. We have defined transformation rules at the metamodel level from ECA-DL TVD to NCL. In order to generate NCL code, the developer should specify the context-models and ECA-DL TVD rules, which conform to the ECA-DL TVD metamodel we have defined.

V. RELATED WORK

A number of approaches have employed ECA rules to describe the reactive behavior of context-aware applications. Some of these approaches directly implement context-aware applications on rule-based platforms without employing a model-driven approach (see, e.g., [15]), thus not benefiting from the use of high-level descriptions of (rule-based) application behavior.

In contrast, the works described in [6] and [8] focus specifically on model-driven approaches for context-aware application development. These approaches assume rule-based implementations (in a rule-based platform called Jess), an assumption we had to forego because of the characteristics of the SBTVD platform standard.

Similarly to our approach, [5] and [9] start the design trajectory from specifications in a platform-independent context-aware service description language. However, while we target a declarative multimedia language, their approaches target Java code instead (which is manually transformed in [5] and produced automatically in [9]).

The work described in [3] also proposes a Model-Driven Development method for context-aware pervasive systems resulting in Java code. They capture system reactions with general-purpose UML sequence diagrams and triggering conditions specified in OCL. The approach includes a transformation to Java, with methods "to check the triggering condition and to execute the actions that are specified in the sequence diagram" [3]. A significant difference with respect to our approach is the use of a general purpose language instead a domain-specific rule language that directly supports context events, situations and the ECA pattern.

Another approach that works on the transformation of ECA policies is [14]. The authors describe a model-driven approach to transform high-level graphical ECA policies into executable PonderTalk code. The Ponder2 Obligation Policy Interpreter is used to execute ECA policies, which is comparable to execution in a rule engine, similarly to the aforementioned approaches based on Jess. Again, this is different from our approach in that the target language we adopt does not support reaction rules explicitly (and thus our work addresses a more challenging transformation scenario).

VI. Concluding Remarks

We believe that a rapid approach to context-aware application development should provide high-level abstractions for designers to express application behavior conveniently in terms of context events and corresponding application reactions.

Our approach leverages the ECA-DL TVD language in the development of context-aware interactive TV applications, automatically producing the implementation of ECA rules in the declarative NCL multimedia language – as required for interactive TV applications in the Brazilian Digital TV (SBTVD) platform.

The main challenge we have addressed refers to transforming a rule-based specification at the platform-independent level into an implementation platform that is not rule-oriented, while preserving conformance to the standard.

We have implemented the transformations in an Eclipse plug-in, which takes as input an instance of the ECA-DL TVD metamodel and produces NCL and NCLua scripts. With respect to the implementation, we intend to focus on improving the ECA-DL TVD editor (which is currently the standard tree editor generated by Eclipse).

In addition, we intend to investigate the combined use of interactive TV applications and mobile context-aware applications, possibly using NCL at the set-top box and rule-based implementations on back-end servers. We believe that rule-based implementations should cope with the scalability challenges that arise from managing a large number of mobile devices and context sources, as demonstrated by the experiments described in [10].

Acknowledgments

This work is partially sponsored with CNPq grant number 309059/2008-9 and the GingaFrEvo and GingaRAP project funded by CTIC/RNP. Igor M. Vale and Izon T. Mielke are supported by CAPES and FAPES, respectively.

References

[1] ABNT NBR. Digital Terrestrial Television - Data Coding and Transmission Specification for Digital Broadcasting - Part 2: Ginga-NCL for fixed and mobile receivers, Brazilian Standard 15606-2, Brazil, 2007. http://www.dtv.org.br/download/en-en/ABNTNBR15606_2D2_2007Ing_2008Vc2_2009.pdf

[2] A. Thawani, S. Gopalan, V. Sridhar. "Context Aware Personalized Ad Insertion in an Interactive TV Environment." Proceedings of Workshop on Personalization in Future TV, 2004.

[3] E. Serral, P. Valderas, and V. Pelechano, "Towards the Model Driven Development of context-aware pervasive systems," Pervasive and Mobile Computing, vol. 6, 2010, pp. 254-280.

[4] F. S. da Silva, L. G. P. Alves and G. Bressan. "PersonalTVware: A Proposal of Architecture to Support the Context-aware Personalized Recommendation of TV Programs". European Interactive TV Conference (EuroITV 2009), Leuven, Belgium, 2009.

[5] K. Yang, S. Ou, A. Liotta, and I. Henning, "Composition of context-aware services using policies and models," Global Telecommunications Conf. (GLOBECOM '05). IEEE, 2005.

[6] L. Ferreira Pires, N. Maatjes, M.J. van Sinderen, and P.D. Costa, "Model-driven approach to the implementation of context-aware applications using rule engines", Proc. Workshop on Model-driven Software Engineering, Darmstadt, Germany, 2007, pp. 24-32.

[7] M. Rosemann, J., Recker, C., Flender, "Designing Context-Aware Business Processes," R. H. L. Chiang, K. Siau, and B. C. Hardgrave (eds.): Systems Analysis and Design: People, Processes, and Projects. Chapter IV , M.E. Sharpe, Inc., Armonk, New York, 2010.

[8] P.D. Costa, Architectural Support for Context-Aware Applications: From Context Models to Services Platforms. Ph.D. Thesis, University of Twente, 2007.

[9] P.D. Costa, Towards a Service Platform for Context-Aware Applications, M.Sc. thesis, University of Twente, 2003.

[10] P.D. Costa, J.P.A. Almeida, L.F. Pires, and M. van Sinderen, "Evaluation of a Rule-Based Approach for Context-Aware Services," Global Telecom. Conf. (GLOBECOM 2008). IEEE, 2008, pp. 1-5.

[11] P.D. Costa, J.P.A. Almeida, L. Ferreira Pires, and M. van Sinderen, "Situation Specification and Realization in Rule-Based Context-Aware Applications," DAIS, LNCS 4531, Springer, 2007, pp.32-47.

[12] P. Viola and M Jones. "Rapid object detection using a boosted cascade of simple features," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 511–518, 2001.

[13] P. Hu, S. K. Chong, J. Indulska, and S. Krishnaswamy, "Context-aware and resource efficient sensing infrastructure for context-aware applications," IEEE PerWare 2010: Middleware support for pervasive computing workshop at Per-Com2010, Germany, April 2010.

[14] R. Romeikat, M. Sinsel, and B. Bauer, "Transformation of Graphical ECA Policies into Executable PonderTalk Code," Rule Interchange and Applications, LNCS Vol. 5858/2009, Springer 2009, pp.193-207.

[15] T. Beer, J. Rasinger, W. Höpken, M. Fuchs, and H. Werthner, "Exploiting E-C-A Rules for Defining and Processing Context-Aware Push Messages," Advances in Rule Interchange and Applications, LNCS Volume 4824/2007, Springer 2007, pp. 199-206.