Regras Contextuais para Aplicações Sensíveis ao Contexto: Modelagem e Realização na Plataforma Ginga

Igor Magri Vale[1], Izon Thomaz Mielke[1], Filipe Bosi Guaitolini[1], Patrícia Dockhorn Costa[1]

[1]Departamento de Informática – Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari 514 – Vitória – ES – Brasil

{igormagrivale,izomtm,bosiwaldo}@gmail.com, pdcosta@inf.ufes.br

Abstract. *This work aims at covering the design trajectory of context-aware application development in the domain of Digital Television (DTV) considering the artifacts for modeling context and situations, the specification of reaction rules and the realization in the Ginga platform. A prototype has been implemented in order to identify the specificities of the platform and of the domain of DTV. As a result, this work identifies requirements to adapt the current development tools to the domain of DTV.*

Resumo. *Este trabalho visa cobrir a trajetória de desenvolvimento de aplicações sensíveis ao contexto no domínio de TV Digital (TVD), considerando desde os artefatos de modelagem de contexto, situações e especificação de regras à realização na plataforma Ginga. Foi desenvolvido um protótipo que permitiu identificar as particularidades desta plataforma e do domínio de TVD. Como resultado, foram identificados requisitos para adaptar as atuais ferramentas de desenvolvimento ao domínio da TVD.*

1. Introdução

Sensibilidade ao contexto permite que aplicações utilizem informações sobre o ambiente no qual o usuário se encontra (contexto) a fim de adequar os serviços de acordo com a situação e as necessidades atuais do usuário. No domínio da TVD, a utilização de informações contextuais pode enriquecer a interação humano-máquina em vários tipos de cenários. Por exemplo, informações contextuais podem ser usadas para detectar a presença de crianças na sala, o que permitiria apenas a exibição de programas adequados para aquela faixa etária [Vale et al. 2009].

Contudo, o desenvolvimento de aplicações sensíveis ao contexto é uma tarefa desafiadora. Aplicações sensíveis ao contexto devem ser capazes de: (i) detectar o contexto do ambiente, (ii) observar, coletar e compor as informações contextuais (iii) automaticamente detectar mudanças relevantes no contexto, e (iv) reagir a estas mudanças, adaptando seu comportamento ou invocando serviços. Portanto, o suporte necessário para o desenvolvimento de aplicações sensíveis ao contexto envolve desde artefatos de modelagem de contextos, situações e regras [Dockhorn Costa 2007] a padrões de implementação.

Como diversos aspectos desta tecnologia envolvem características da plataforma de implementação e do domínio de aplicações, neste caso, TVD, existe a necessidade de adaptar as atuais ferramentas de desenvolvimento para este novo domínio. Este artigo

descreve avanços nas áreas supracitadas de forma a cobrir a trajetória de desenvolvimento para aplicações sensíveis ao contexto em TVD.

Um cenário de aplicação foi utilizado para descrever essas tecnologias e salientar as especificidades relativas ao domínio de TVD, em particular considerando a plataforma Ginga [Telemídia 2007] como alvo de implementação.

Este artigo é estruturado da seguinte forma: Seção 2 discute o domínio de aplicação, introduz o cenário utilizado como estudo de caso e apresenta os modelos de contexto e situação que descrevem as características do cenário; Seção 3 apresenta ECA-DL, uma linguagem de definição de regras usada para especificar comportamentos reativos de aplicações sensíveis ao contexto; Seção 4 introduz a plataforma Ginga, Seção 5 descreve o protótipo, Seção 6 discute trabalhos relacionados e a Seção 7 apresenta conclusões e indica trabalhos futuros.

2. Domínio de Aplicação

Este trabalho usa como domínio de aplicação a área de TVD, vislumbrando o desenvolvimento de aplicações inteligentes e sensíveis ao contexto do usuário que visam enriquecer as interações entre o usuário e seu aparelho de televisão. Por exemplo, o cenário apresentado em [Vale et al. 2009] demonstra como uma simples aplicação sensível ao contexto pode capturar informações do ambiente e, com base nessas informações, consegue tomar decisões autonomamente: "João possui uma aplicação sensível ao contexto instalada no seu aparelho de televisão que monitora o comportamento do microondas e a programação exibida na televisão. Quando a pipoca estiver pronta no microondas e houver um filme sendo exibido, João é notificado em sua TV por meio de uma mensagem na tela, e o filme tem sua exibição pausada até que João busque sua pipoca e continue a exibição".

A Figura 1 mostra a modelagem contextual que representa as entidades e contextos relevantes para a aplicação, que foram identificados analisando o cenário proposto. Para a modelagem de informações contextuais, foram utilizadas as abstrações propostas por [Dockhorn Costa 2007]. Nestas abstrações, identificam-se tipos de *entidades* (e.g., pessoas, carros e dispositivos), e tipos de *contexto* (e.g., Contexto Intrínseco e Contexto Relacional). Contexto Intrínseco refere-se às informações contextuais inerentes a uma única entidade, como por exemplo, a *localização* de uma pessoa ou a *luminosidade* de uma sala. Já o Contexto Relacional refere-se às informações contextuais que surgem da relação entre múltiplas entidades, como por exemplo, a *amizade* entre pessoas e a *presença* de uma pessoa em uma sala.

Na Figura 1 podem ser vistos as entidades `TV`, `TVProgram`, `MicrowaveOven`, os contextos intrínsecos `MicrowaveOvenStatus` e `TVProgramKind`, que são inerentes ao forno microondas e ao programa de TV, respectivamente, e o contexto relacional entre `TV` e `TVProgram`, denominado `Displaying`.

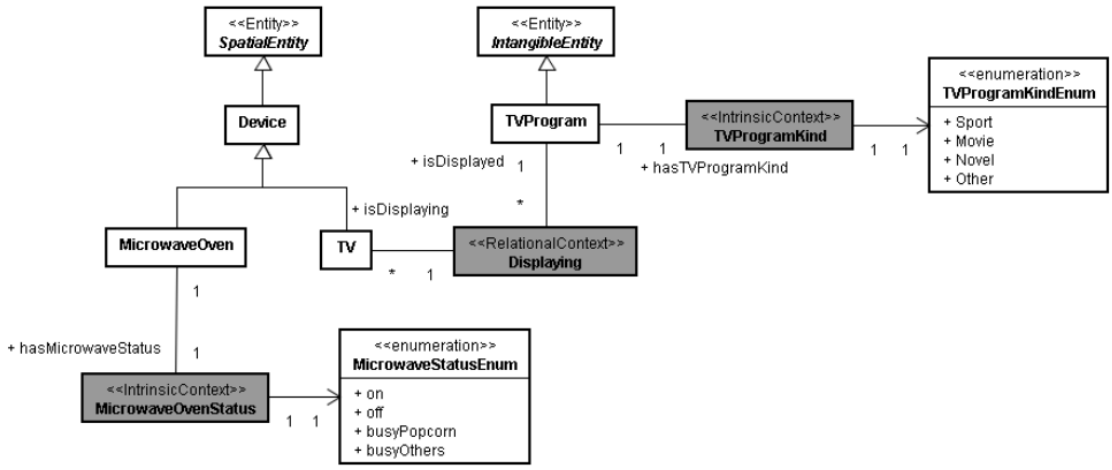

Figura 1. Tipos entidade e contexto relevantes.

Em [Vale et al. 2009], também são modeladas as *situações* nas quais essas entidades e contextos podem estar envolvidos. Situações modelam condições específicas nas quais entidades e contextos estão envolvidos. A Figura 2 mostra duas dessas situações.

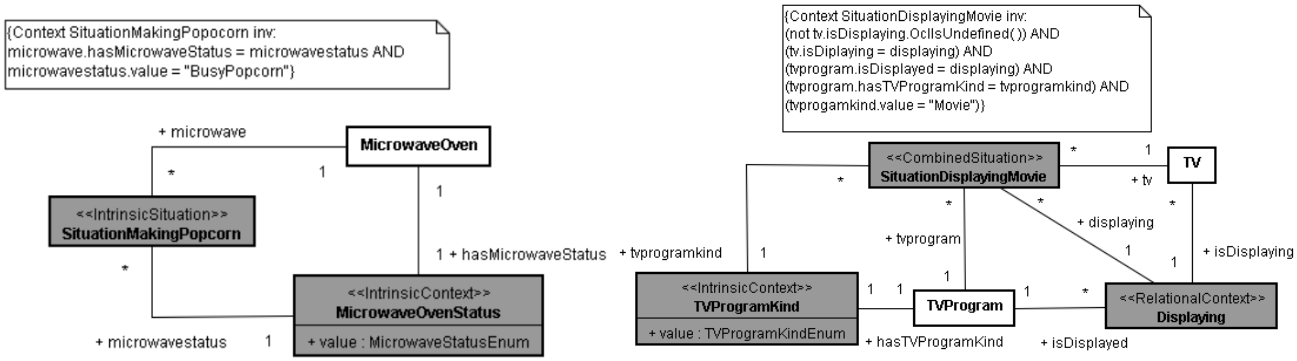

Figura 2. Situações SituationMakingPopcorn e SituationDisplayingMovie.

Na Figura 2, o tipo situação `SituationMakingPopcorn` especifica a condição sob a qual um forno microondas está ocupado preparando pipoca, enquanto que `SituationDisplayingMovie` detecta quando uma TV está exibindo um filme como seu programa de TV.

Esses modelos contextuais e de situação facilitam o projeto de aplicações sensíveis ao contexto por prover aos desenvolvedores de aplicação meios de estruturar eficientemente as situações de interesse da aplicação. Em [Dockhorn Costa 2007] é proposta uma abordagem baseada em regras para realização da detecção de contexto e situações contextuais. Nessa abordagem, os modelos contextuais e de situação são utilizados como base para especificação de regras que descrevem o comportamento reativo da aplicação sensível ao contexto. Seção 3 apresenta a linguagem de definição de regras que foi utilizada para especificar o comportamento reativo do cenário estudado.

3. ECA-DL

[Dockhorn Costa 2007] propõe uma linguagem específica de domínio baseada em regras, conhecida como *ECA-DL*, especialmente desenvolvida para especificação de comportamentos reativos de aplicações sensíveis ao contexto. Por ser específica de domínio, torna-se mais prática para desenvolvedores de aplicações sensíveis ao contexto do que outras linguagens de propósitos gerais.

Uma regra ECA-DL também é consistente com o padrão arquitetural Event-Control-Action (ECA) [Dockhorn Costa 2007] que provê uma estrutura de alto nível para aplicações que pró-ativamente reagem às mudanças no contexto. Dessa forma, uma regra ECA-DL é composta dos seguintes elementos:

- Eventos (*Events*), que permitem definir mudanças relevantes de situações por meio de combinações simples ou complexas de eventos;
- Condição (*Condition*), parte opcional que permite definir uma expressão lógica que deve permanecer verdadeira simultaneamente à ocorrência dos eventos e anteceder a execução de uma ação;
- Ação (*Action*), que permite a definição de invocações de operações a serem executadas devido à ocorrência de eventos e ao preenchimento das condições associadas a esses eventos.

Os elementos supracitados resultaram nas cláusulas `Upon`, `When` e `Do`, respectivamente. Eventos são definidos na cláusula `Upon`, enquanto que as condições são especificadas na cláusula `When` e, finalmente, ações são especificadas na cláusula `Do`. A cláusula `When` pode ser omitida se não há condições para serem especificadas. Dessa forma, uma regra ECA-DL tem a seguinte estrutura:

```
Upon<uponExpression>
When<conditionExpression>
Do <actionExpression>
```

Uma `<uponExpression>` define uma combinação de eventos, que podem ser, dentre outros tipos, de *situação*. Eventos de situação são definidos em termos de transições de situação. Dois tipos de transições de situação são suportados: *EnterTrue*, que representa quando a situação S_1 passa a existir, e *EnterFalse*, que representa quando S_1 deixa de existir. A ocorrência do evento (ou da composição de eventos) especificado na cláusula `Upon` dispara a avaliação da cláusula `When`. Por exemplo, para o cenário apresentado na Seção 2, um evento de interesse seria quando a situação `SituationMakingPopcorn` deixa de existir, ou seja, a pipoca que estava sendo preparada no forno microondas de João está pronta. Isso é descrito da seguinte forma:

```
Upon EnterFalse (SituationMakingPopcorn(MicrowaveOven.MicrowaveOvenJohn))
```

Uma `<conditionExpression>` consiste de uma expressão booleana que pode ser composta de outras expressões booleanas. As ações descritas na cláusula `Do` serão invocadas apenas quando os eventos da cláusula `Upon` ocorrerem e as condições da cláusula `When` forem satisfeitas. No caso do cenário da Seção 2, a restrição seria que a TV do João deva estar exibindo um filme, ou seja, que a situação `SituationDisplayingMovie (TV.TVJohn)` exista. Portanto, a cláusula `When` para este cenário possui a seguinte estrutura:

```
When SituationDisplayingMovie (TV.TVJohn)
```

Finalmente, uma `<actionExpression>` consiste de uma lista de invocação de serviços. Para o caso do cenário estudado, os serviços identificados seriam: notificar o usuário (`Notify (Person.John, "A Pipoca está pronta!")`) e pausar o filme (`PauseMovie (TV.TVJohn)`). Portanto, a regra ECA-DL completa para o cenário da Seção 2 seria a seguinte:

```
Upon EnterFalse (SituationMakingPopcorn(MicrowaveOven. MicrowaveOvenJohn))
When SituationDisplayingMovie (TV.TVJohn)
Do Notify (Person.John, "A Pipoca está pronta!"), PauseMovie (TV.TVJohn))
```

4. Plataforma GINGA

Como mencionado na Seção 2, este trabalho baseia-se no domínio de aplicação de TVD, mais precisamente, na plataforma do Sistema Brasileiro de TVD (SBTVD) [ABNT 2007]. O SBTVD oferece um *middleware*, denominado de Ginga [ABNT 2007], que possibilita a execução de aplicações interativas no ambiente da TVD. Aplicações interativas podem ser desenvolvidas no Ginga em linguagem declarativa, através da linguagem NCL [ABNT 2007], ou em linguagem procedural, utilizando a linguagem Java [Souza Filho et al. 2007]. Ainda é possível utilizar *scripts* desenvolvidos em linguagem Lua para complementar as aplicações escritas em linguagem declarativa.

A Figura 3 apresenta a arquitetura do *middleware* Ginga. Na parte NCL do Ginga, (Ginga-NCL) encontra-se o formatador NCL, juntamente com o interpretador Lua. Já na parte Ginga-J tem-se a máquina virtual Java. Ambas as partes, NCL e Java, têm acesso à camada Ginga Common Core, na qual se encontram os tocadores de mídias, o canal de interatividade, o sintonizador, entre outros.

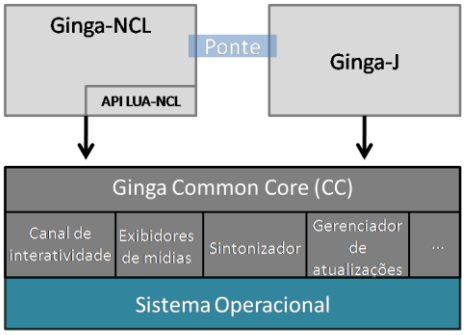

Figura 3. Arquitetura do *middleware* Ginga.

De acordo com [Soares and Castro 2008], a camada Ginga-J é opcional em dispositivos móveis, sendo as demais camadas obrigatórias a todos as implementações do Ginga para quaisquer dispositivos. Assim, optou-se por desenvolver o protótipo utilizando a linguagem declarativa NCL com elementos procedurais Lua. Desta forma, a aplicação sensível ao contexto desenvolvida neste trabalho é iniciada e executada pelo Ginga-NCL, juntamente com o interpretador de NCLua. Além disso, são realizados acessos à camada Ginga *Common Core* para, por exemplo, exibir as mídias existentes na aplicação, ou utilizar o canal de interatividade, uma vez que este componente possibilita a troca de informações entre a aplicação e os dispositivos utilizados para captura de contexto.

5. Protótipo

Foi desenvolvido um protótipo com objetivo de identificar as particularidades da plataforma Ginga de maneira a adaptar as atuais ferramentas de desenvolvimento para o domínio de TVD. O cenário da aplicação sensível ao contexto apresentado na Seção 2 foi implementado na linguagem NCL, com auxilio de *scripts* NCLua [Sant'Anna et al. 2008], sendo que as inferências contextuais e de reatividade especificados nos modelos de contexto, situações e regras ECA-DL foram implementadas usando recursos da própria linguagem NCL. A Figura 4 apresenta um diagrama de sequência que especifica o comportamento típico do protótipo. Neste são representados o Canal de TV, a Aplicação Sensível ao Contexto, o Microondas, o Telespectador e a sequência de interações entre essas entidades.

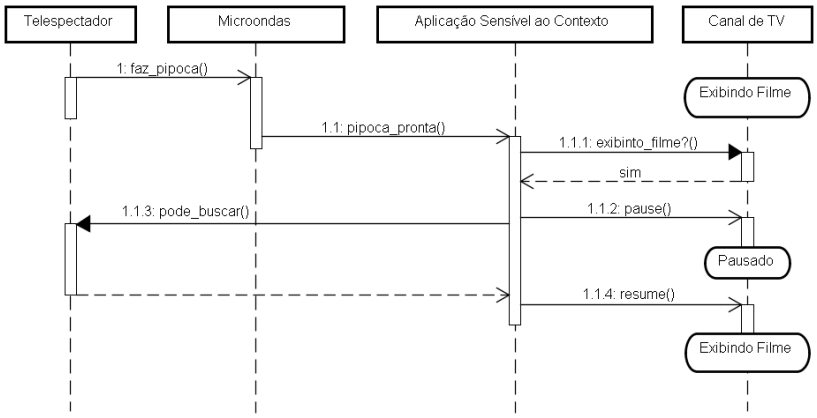

Figura 4. Diagrama de sequência típico.

Inicialmente, o Telespectador configura o Microondas para o preparo da pipoca (interação "faz_pipoca()" na Figura 4). A partir deste momento, a Aplicação Sensível ao Contexto espera a ocorrência do evento "pipoca_pronta". Assim que a pipoca estiver pronta, o Microondas notifica a Aplicação (interação "pipoca_pronta()") e a Aplicação verifica se um filme está sendo exibido no Canal de TV (interação "exibindo_filme?()"). Quando ambas as condições forem verdadeiras, a Aplicação dispara a ação que pausa o filme (interação "*pause()*"). Se o Telespectador desejar continuar assistindo o filme, a exibição é reiniciada no Canal de TV (interação "*resume*()").

5.1. Implementação

A implementação do protótipo é composta de três módulos: o *Sensor do Microondas*, o *Monitor*, e o *Intermediador de Dados*. O Sensor do Microondas é executado em um dispositivo remoto, enquanto os demais módulos são executados pelo receptor de TVD.

O *Sensor do Microondas* é uma aplicação criada para simular o comportamento de um microondas. Esta aplicação disponibiliza informações contextuais sobre seu estado (ocupado, disponível, etc.) por meio de *sockets*, atuando como servidor no qual possíveis clientes (por exemplo, o Intermediador de Dados) se conectam para obter informações contextuais do microondas. O *Monitor* é um documento NCL que implementa o comportamento principal do protótipo, sendo responsável por inferir as situações contextuais e acionar devidos serviços. Além disso, controla a exibição do vídeo na TV. O *Intermediador de Dados*, por sua vez, é um *script* NCLua representado

no documento NCL que implementa o Monitor. Este *script* é responsável por intermediar a comunicação entre o Sensor do Microondas e o Monitor. A existência deste componente vem da necessidade da troca de dados por rede, algo que não pode ser feito apenas utilizando a linguagem NCL.

A comunicação entre o Sensor do Microondas e o Intermediador de Dados é realizada por meio de um Canal de Interatividade, que consiste de um mecanismo de comunicação que fornece conexão entre o receptor de TV (Intermediador de Dados) e um servidor remoto (Sensor de Microondas). Desta forma, informações contextuais capturadas pelo Sensor de Microondas são enviadas através de eventos para o Intermediador de Dados.

A comunicação entre o Intermediador de Dados e o Monitor é feita por mudanças de valores de atributos que são compartilhados por ambos os módulos. Em NCL, estes atributos são realizados por âncoras de propriedades, como mostrado na Figura 5. Por exemplo, quando o Intermediador de Dados recebe um evento de "pipoca pronta", ele notifica o Monitor modificando a propriedade "pipoca" da âncora de propriedade para o valor "pronto".

```
<media id="sensor" src="microondas.lua" descriptor ="dSensor">
   <property name="pipoca" value="perguntar"/>
</media>
```

Figura 5. Nó de conteúdo representando o interpretador de dados e suas âncoras de propriedade.

5.1.1. Detectando contextos "pipoca pronta" e "exibindo filme"

Como mencionado, o Sensor do Microondas envia informações para o Intermediador de Dados, que por sua vez atualiza o Monitor com informações contextuais do microondas.

Para detectar a exibição de um filme na programação, NCL permite a especificação de *âncoras de conteúdo*, que definem segmentos específicos na programação. Este recurso permite, por exemplo, identificar quando um filme é iniciado e coletar informações sobre seu estado (ocorrendo, finalizado, etc.), ou o número de vezes que foi reproduzido. A Figura 6 mostra uma mídia de vídeo, que contém uma âncora de conteúdo (*tag* `<area>`), indicando que um filme inicia aos 100 segundos e termina aos 420 segundos.

```
<media id="video" src="video.mpg" descriptor="dVideo">
 <area id="filme" begin="100s" end="420s"/>
</media>
```

Figura 6. Exemplo de âncora de conteúdo como identificador de programação.

5.1.2. Implementação do comportamento reativo descrito pela regra ECA-DL

A regra ECA-DL foi implementada como parte do Monitor utilizando o recurso de conectores da linguagem NCL. Um conector especifica relações entre diferentes *nós* em um documento NCL definindo condições sobre as quais um nó pode ser ativado, e as ações que serão disparadas quando isto acontecer. Um *nó* geralmente está associado a um recurso multimedia, como por exemplo, um vídeo, uma imagem ou um código procedural. Um conector poderia definir, por exemplo, uma relação causal entre dois vídeos, de forma que ao término de um vídeo, outro seja iniciado.

O conector que implementa a regra ECA-DL discutida na Seção 3 é mostrado na Figura 7. Este conector é utilizado para definir o seguinte comportamento: *"uma vez que* a pipoca estiver pronta, *quando* um filme estiver sendo exibido, *faça* pausar o filme e exibir mensagem na tela".

```
<causalConnector id="onEndAttributionPropertyTestPause">
   <compoundCondition operator="and">
      <simpleCondition role="onEndAttribution"/>
      <assessmentStatement comparator="eq">
         <attributeAssessment role="pTest" eventType="attribution"
attributeType="nodeProperty"/>
         <valueAssessment value="pronto"/>
      </assessmentStatement>
      <assessmentStatement comparator="eq">
         <attributeAssessment role="rMovie" eventType="presentation"
attributeType="state"/>
         <valueAssessment value="occurring"/>
      </assessmentStatement>
   </compoundCondition>
   <compoundAction operator="par">
       <simpleAction role="pause"/>
        <simpleAction role="start"/>
    <compoundAction/>
</causalConnector>
```

Figura 7. Conector que representa a regra ECA-DL discutida na Seção 3.

A Figura 7 mostra que ao fim de uma atribuição à variável que contém o status da pipoca, se o valor contido em "pTest" for igual ao valor "pronto" e o estado de "rMovie" for "ocorrendo", então a ações de "*pause*" e "*start*" são disparadas paralelamente.

NCL permite o desacoplamento entre a especificação de um conector e a definição dos nós, de maneira que o conector apenas especifica a relação entre nós, mas não define os recursos de mídia sendo relacionados. A amarração entre um nó e um recurso de mídia é feita pelo elemento *link*, como mostrado na Figura 8. Por exemplo, este elemento *link* realiza a amarração entre a âncora de propriedade "pipoca" do Interpretador de Dados e o papel (*role*) "pTest" especificado no conector.

```
<link xconnector="onEndAttributionPropertyTestPause">
   <bind component="sensor" interface="pipoca" role="onEndAttribution"/>
   <bind component="sensor" interface="pipoca" role="pTest" />
   <bind component="video" interface="filme" role="rMovie" />
   <bind component="video" role="pause"/>
   <bind component="mensagem_pause" role="start"/>
</link>
```

Figura 8. Link entre elementos e conector.

6. Trabalhos Relacionados

A utilização de regras aplicadas ao cenário da TVD interativa ainda é pouco explorada. Alguns trabalhos que tratam desse tema são [da Silva et al. 2009], [Neto and Ferraz 2006], [Endler et al. 2006] e [Brackmann et al. 2009].

Em [da Silva et al. 2009], é proposto que programas de TV sejam sugeridos ao telespectador com base nas informações sobre o programa, enviadas pela emissora através do Guia Eletrônico de Programa, e nas informações coletadas do usuário. Neste trabalho, a sugestão de preparo de pipoca é feita com base nas informações enviadas pela emissora e informações referentes ao estado do forno microondas são capturadas

através de sensores. Em [da Silva et al. 2009], ao contrário, as informações sobre o usuário devem ser informadas pelo próprio usuário.

[Neto and Ferraz 2006] e [Endler et al. 2006] apresentam propostas para processamento de informações contextuais remotamente e não no cliente, devido à limitações de processamento. Neste trabalho, utiliza-se o próprio formatador NCL do *middleware* Ginga como uma máquina de regras e, portanto, não há uma preocupação com a limitação de processamento, uma vez que qualquer receptor de TVD com Ginga pode exectuar a aplicação.

[Brackmann et al. 2009] propõe o GingaSC que define um conjunto de APIs para captura e processamento de informações contextuais através de extensões da arquitetura do Ginga. Não foram descritos protótipos ou aplicações que demonstrassem a viabilidade de tais extensões. No protótipo apresentado neste trabalho, informações contextuais são capturadas e processadas utilizando os componentes oferecidos pela própria arquitetura atual do Ginga-NCL.

Pode-se observar que a utilização de regras no ambiente da TVD interativa ainda está em seus estágios iniciais. Poucos trabalhos utilizam máquinas de regras para oferecer algum tipo de serviço para o usuário. Além disso, apenas uma das propostas foi desenvolvida direcionadamente para o *middleware* Ginga.

7. Conclusões

Este trabalho apresentou avanços na trajetória de desenvolvimento de aplicações sensíveis ao contexto para a plataforma Ginga. A modelagem de uma aplicação sensível ao contexto foi apresentada, incluindo a definição de regras que permitem a especificação de comportamentos reativos. Foi desenvolvido um protótipo com objetivo de identificar as particularidades da plataforma Ginga de maneira a adaptar as atuais ferramentas de desenvolvimento a esta plataforma. Por exemplo, foi identificado que os mecanismos de *âncora de propriedade e âncora de conteúdo* (Seção 5.1.1) podem ser usados para capturar informações contextuais. Para realização de regras ECA-DL, foi identificada a necessidade do uso de *conectores* (Seção 5.1.2).

Entretanto, observou-se que uma simples regra ECA-DL gerou uma quantidade considerável de linhas de código e demandou horas de estudo e programação. A criação de regras mais complexas usando apenas os artefatos de NCL torna-se praticamente inviável. É indicado como trabalho futuro a definição de mecanismos mais eficientes para implementação de regras na plataforma Ginga. Por exemplo, pode-se vislumbrar a utilização de técnicas da área de Desenvolvimento Orientado a Modelos (MDA) [Almeida et al. 2004] a fim de permitir a geração automática de código NCL a partir de regras ECA-DL. MDA permite automatizar o mapeamento entre as linguagens supracitadas a partir da especificação dos metamodelos de ECA-DL e NCL e de frameworks de transformação de modelos [The Eclipse Foundation 2010].

Finalmente, no protótipo apresentado, a descoberta e captura de informações contextuais foram realizadas através de simulações. Como trabalho futuro, pretende-se estudar mais profundamente a camada Ginga *Common Core* para identificar abordagens eficientes de descoberta e captura de informações contextuais por meio de dispositivos

(sensores, contadores, medidores, entre outros), que estarão distribuídos pela residência do usuário.

Referências

ABNT. (2007). ABNT NBR 15606-2:2007: Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações.

Almeida, J. P. A., Pires, L. F. and van Sinderen, M. (2004). Costs and Benefits of Multiple Levels of Models in MDA Development. In Proceedings of the 2nd European Workshop on Model-Driven Architecture with Emphasis on Methodologies and Transformations, pages 12–20, Canterbury, UK.

Brackmann, C. P., Venecian, L. R., Luzzardi, P. R., & Yamin, A. C. (2009). GingaSC: Uma Proposta de Sensibilidade ao Contexto para TV Digital Brasileira. Seminfo 2009, Torres, Rio Grande do Sul.

da Silva, F. S., Alves, L. G. P., and Bressan, G. (2009). PersonalTVware: Uma Proposta de Arquitetura Sensível ao Contexto para Suporte a Recomendação Personalizada de Conteúdo no Cenário da TV Digital Interativa. SBCUP – I Simpósio Brasileiro de Computação Ubíqua e Pervasiva, Bento Gonçalves, Rio Grande do Sul.

Dockhorn Costa, P. (2007). Architectural Support for Context-Aware Applications – From Context Models to Services Platforms. PhD thesis, Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands.

Endler, M., Sacramento, V., Rubinsztejn, H., do Nascimento, F. N., da Rocha, R. C. A., Filho, J. V., and Baptista, G. L. B. (2006). Experiências no Desenvolvimento de uma Arquitetura de Middleware para Ciência de Contexto. Monografias em Ciência da Computação, PUC-Rio. Rio de Janeiro, Brasil.

Neto, F. C. A. and Ferraz, C. A. G. (2006). Uma arquitetura para suporte ao desenvolvimento de aplicações sensíveis a contexto em cenário de convergência. SBRC 2006, Curitiba, Paraná.

Sant'Anna, F., Cerqueira, R., and Soares, L. F. G. (2008). NCLua - Objetos Imperativos Lua na Linguagem Declarativa NCL. Webmedia 2008, Vila Velha, Espírito Santo.

Soares, L. F. G. and Castro, P. H. (2008). As múltiplas possibilidades do middleware Ginga. Revista Producão Profissional – Junho 2008, pages 76–83.

Souza Filho, G. L., Leite, L. E. C., and Batista, C. E. C. F. (2007). Ginga-J: The Procedural Middleware for the Brazilian Digital TV System. Journal of the Brazilian Computer Society, 13(4):47–56.

Telemídia, L. (2007). Middleware Ginga – TV Interativa se faz com Ginga! Fonte: http://www.ginga.org.br

The Eclipse Foundation. (2010). Eclipse.org. Fonte: http://www.eclipse.org/

Vale, I. M., Guaitolini, F. B., and Costa, P. D. (2009). Modelagem Contextual de um Cenário para a TV Digital. SIMTVD – 1º Simpósio Internacional de Televisão Digital, Bauru, São Paulo.