Uma Ferramenta de Autoria para Aplicações NCL e NCLua: uma Abordagem Orientada a Templates e com Suporte a Serviços Web

Thales Pordeus Ferreira
Trabalho de Mestrado
Início: março de 2011 – Término: dezembro de 2012
Programa de Pós Graduação em Informática (PPGI)
Universidade Federal da Paraíba (UFPB)
thales@lavid.ufpb.br

Raoni Kulesza
(Coorientador)
Centro de Informática (CIN)
Universidade Federal de Pernambuco (UFPE)
rk@cin.ufpe.br

Guido L. de S. Filho
(Orientador)
Laboratório de Aplicações de Vídeo Digital (LAVID)
Universidade Federal da Paraíba (UFPB)
guido@lavid.ufpb.br

RESUMO

As aplicações multimídia interativas tipicamente oferecem uma interface com o usuário (UI – User Interface) sofisticada e integrada com diferentes objetos de mídia. Complementar à UI multimídia, é comum a necessidade de apresentar informações visuais associadas a algum processamento de lógica de aplicação, uma característica comum na área das aplicações de informação, entretenimento ou jogos de computador. Neste contexto, a especificação da aplicação é feita em conjunto pelas equipes de software e UI, normalmente utilizando respectivamente ambientes de programação e ferramentas de edição visual. Atualmente, é possível identificar uma lacuna na definição de ferramentas de autoria que considerem a integração entre UI e lógica de aplicação complexa, por exemplo, os serviços web. O objetivo deste trabalho é definir uma ferramenta de autoria orientada a templates que integre melhor objetos de mídia e lógica de aplicação complexa, de forma a diminuir o tempo de desenvolvimento das aplicações. É empregada uma abordagem de desenvolvimento dirigida a modelos para gerar automaticamente as aplicações, reusando estruturas do template e funcionalidades presentes em serviços web.

PALAVRAS-CHAVE

Ferramenta de Autoria, Desenvolvimento Orientado a Templates, NCL, Lua, Serviços Web, TV Digital.

1. CONTEXTO TEÓRICO

As mudanças radicais da TV para tornar-se numa plataforma digital e interativa, como por exemplo, a TV Digital Interativa (TVDI) e as TVs Conectadas [1], possibilitam ao telespectador vivenciar experiências de informação e interação aproximadas daquelas já difundidas em aplicações presentes na Web. No cenário da TVDI, as formas já presentes de interação com os programas de TV (como votação, envio de mensagens, entre outras) podem ser aprimoradas com o uso das aplicações sobrepostas aos programas de TV [2]. Dessa forma, a interatividade torna mais transparente a interação do telespectador com o programa de TV.

Uma aplicação multimídia é definida como um software que possui interface com o usuário (UI – *User Interface*) com pelo menos um objeto de mídia (áudio, vídeo, imagens, animações, gráficos 2D ou 3D). Adicionalmente à UI multimídia, é comum em algumas das aplicações na TVDI o envolvimento com algum processamento de lógica de negocio, característica comum na área das aplicações informativas [4]. Estas aplicações possuem novos requisitos de ter objetos de mídia que estão intimamente vinculados à lógica de aplicação.

Um cenário atual que envolve este tipo de integração são os serviços web. Os serviços web são um tipo de código procedural disponibilizando funcionalidades para qualquer tipo de aplicação. As aplicações utilizando serviços da web, a exemplo das aplicações web *Mashups*, estão se tornando cada vez mais comuns e o reuso de serviços em aplicações multimídia também se faz necessário.

Para o desenvolvimento das aplicações é empregado linguagens textuais de autoria especializadas no domínio de criação multimídia. Por exemplo, aplicações para o padrão brasileiro de TV Digital tem a sua apresentação visual especificada pela linguagem de marcação NCL (*Nexted Context Language*) e a parte de código imperativo (comumente chamado de *script*) descrita pela linguagem Lua (NCLua) [3]. Uma deficiência da linguagem textual é demandar um esforço de codificação para transcrever em forma de texto a especificação da aplicação. A especificação feita por linguagem textual como NCL está relacionada basicamente a duas tarefas: (i) definição da sintaxe do conceito da linguagem a ser utilizado e (ii) de que forma eles serão usados (isto é, seus atributos preenchidos com valores). O trabalho [10] contorna em parte esse problema, pois tais ferramentas, apesar de acelerarem a escrita da sintaxe, não resolvem o segundo problema, que é a edição rápida dos valores dos atributos relacionados à especificação da aplicação. Ferramentas de autoria e uma sintaxe mais enxuta ajudariam a poupar tempo na especificação do documento [11].

Uma linguagem de autoria, por ser voltada para especificação da apresentação visual da aplicação, costuma ser abstraída utilizando ferramentas de autoria. Ferramentas de autoria oferecem técnicas simples e intuitivas para a construção visual de uma aplicação [5]. Dessa forma, uma das grandes vantagens é ajudar na redução da quantidade de código que os programadores precisam produzir, possibilitando uma melhoria considerável no tempo de construção da aplicação.

Outra forma para aumentar a produtividade do software é o uso dos métodos de autoria orientado a templates. Um template reaproveita partes de estruturas de aplicações que já foram

desenvolvidas previamente em novos casos de aplicações. Atualmente, as linguagens TAL (Template Authoring Language) [6] e XTemplate [7] são as principais linguagens para desenvolvimento baseado em templates de aplicações NCL. Aqui também se faz pertinente esconder os detalhes de implementação sintáticos e semânticos dessas linguagens textuais com o uso de ferramentas visuais.

Uma das metodologias que procura diminuir a complexidade do desenvolvimento de aplicações de domínio específico é o Desenvolvimento Generativo de Software (GSD – *Generative Software Development*). A principal característica do GSD é ser uma abordagem que busca modelar e desenvolver famílias de produtos de software ao invés de sistemas individuais [8]. Este trabalho aplica alguns dos conceitos existentes no trabalho [9] para o desenvolvimento de aplicações multimídia baseada em templates.

2. IDENTIFICAÇÃO DO PROBLEMA

O desenvolvimento de aplicações multimídia envolve basicamente a coordenação entre dois tipos de projetos: (i) projeto da interface com o usuário e (ii) projeto do software, que é a lógica da aplicação. Pelo fato deles serem comumente desenvolvidas independentemente (por profissionais e equipes distintas) ambas precisam ser integradas. Sendo assim, um requisito importante é a coordenação dos diferentes grupos de desenvolvedores e a integração dos seus artefatos advindos dos seus diferentes resultados produzidos.

Quando interfaces multimídia com o usuário tornam-se mais sofisticadas, as dependências entre objetos de mídias, interface e lógica de aplicação aumentam. Neste sentido, existe uma dependência da UI para a lógica e vice versa. Tais inter-relações são um gargalo critico dentro do desenvolvimento da aplicação e requer uma coordenação entre os diferentes grupos de desenvolvedores [15].

As ferramentas de autoria para NCL existentes não suportam a integração entre o projeto de software e o projeto de interface. As ferramentas tratam as tarefas de especificação da interface e do relacionamento entre as mídias utilizando uma única visão, o que dificulta o trabalho independente do designer e do programador. Por exemplo, no Composer [12] a visão estrutural permite criar e editar a aparência da UI como também definir lógica da apresentação.

Como visto anteriormente, o Composer [12] e o EDITEC [13] são algumas das ferramentas de autoria existentes para auxiliar na criação rápida de aplicações NCL. Entretanto, elas requerem do usuário conhecimento especializado da linguagem NCL para desenvolver uma aplicação. Em ambos, a notação visual faz-se uso de muitos conceitos presentes na NCL, os quais são bastante específicos ao domínio de aplicações multimídia interativas.

Outro problema mais pontual das ferramentas de autoria existentes é a inexistência do suporte a linguagem NCLua, a qual possibilita a integração do código procedural dentro da aplicação NCL. Por exemplo, hoje não é possível que o programador possa de forma visual e rápida visualizar ou realizar uma chamada dos métodos existentes nas mídias procedurais e que um valor de retorno seja utilizado para ser apresentado por outra mídia. Um cenário atual que envolve bastante este tipo de integração é a utilização dos serviços web dentro da aplicação. A automatização do uso do código procedural dos serviços web facilitaria bastante a vida do programador.

3. OBJETIVO

Buscando solucionar os problemas relacionados anteriormente, este trabalho endereça a integração entre o projeto da interface com o usuário e o projeto do software utilizando uma ferramenta de autoria. Procuramos aplicar os conceitos e ideias existentes na área de GSD em uma ferramenta para a autoria de *templates* (famílias de aplicações multimídia) suportando a geração automática de aplicações NCL. Duas visões distintas e integradas suportam a

O objetivo geral desta dissertação é propor e desenvolver uma ferramenta de autoria orientada a *templates* (família de aplicações). O intuito é que a ferramenta coordene os artefatos produzidos pelo designer e pelo programador durante o processo de desenvolvimento da aplicação. Complementar a autoria visual da aplicação a especificação do *template* também é feita visualmente.

Um dos principais requisitos deste trabalho é abstrair os conceitos envolvendo tanto a linguagem NCL como a definição do template. Aspectos como a definição da interface e o relacionamento entre as mídias NCL e NCLua, ou seja, sincronismo e interatividade são feitas utilizado um mecanismo visual. A ferramenta diminui o esforço de criação da aplicação ganhando tempo em relação às linguagens textuais.

4. CONTRIBUIÇÕES

A principal contribuição esperada nesse trabalho é uma ferramenta de autoria para suportar o projeto da UI e projeto de software de maneira integrada e independente da sintaxe NCL. Como contribuições secundárias, mostramos como aplicar técnicas de GSD ao desenvolvimento de aplicações multimídia baseadas em templates.

5. METODOLOGIA

Como descrito anteriormente, este trabalho tenta resolver o problema de integração entre as atividades envolvidas no projeto da UI e no projeto de software (UI e Software).

A metodologia de pesquisa consiste em três fases: (i) análise do problema; (ii) proposta e projeto da solução e por fim a (iii) validação da proposta.

A primeira fase analisa os trabalhos existentes e investiga seus problemas. Para isso é feito uma revisão de literatura para determinar os requisitos da nossa abordagem que são advindos de outros trabalhos. Mais ainda, ela serviu para descobrir algumas limitações que precisam ser resolvidas. Tais limitações foram discutidas na seção 2 e constituem nos principais problemas endereçados. Primeiro foi feito um estudo sobre as principais abordagens para tratar o desenvolvimento de aplicações multimídia com foco na integração entre UI e Software. No segundo estudo, analisaram-se os métodos e ferramentas de autoria orientadas a templates.

Na segunda fase, os requisitos e problemas da etapa anterior são usados para projetar uma solução. O principal requisito é entender quais as tarefas e necessidades de integração o projeto de UI e software requerem. Baseado neles foi definido uma ferramenta de autoria com duas visões integradas e propomos um método de autoria baseado em template.

Finalmente, na terceira fase, é feito um estudo para investigar em que medida a ferramenta de autoria pode melhorar no tempo de desenvolvimento de aplicações multimídia em comparação aos métodos existentes de autoria orientada a templates. O objetivo é avaliar se a questão do nível de abstração utilizando um editor visual deve aumentar a produtividade, diminuindo o esforço de codificação durante o desenvolvimento de aplicações multimídia.

6. ESTÁGIO ATUAL DO TRABALHO

Como descrito na seção anterior, o trabalho proposto possui as etapas para analisar o problema e em seguida propor e validar a solução. Destas, encontra-se feito a análise do problema (seção 2) e uma proposta inicial da solução (seção 7).

No tocante ao projeto, para o método de autoria orientado por template já foi desenvolvida uma prova de conceito através de um protótipo que realiza a geração automática de aplicações baseada em ferramentas tipo *wizard* usando 2 (dois) estudos de casos. Um deles instancia aplicações a partir de um template para aplicações do tipo Enquete e outro template para aplicações de S*lideshow*. Nos dois templates são tratados cenários que utilizam Serviços *web*.

Adicionalmente, foi realizado um estudo piloto através de metodologias de engenharia de software experimental com o objetivo de avaliar de forma quantitativa a ferramenta em relação a trabalhos relacionados que representam ferramentas de autoria existentes.

7. SOLUÇÃO PROPOSTA

A criação de uma aplicação multimídia NCL envolve uma cooperação entre o projeto da interface com o usuário e o projeto de software, que é a lógica de aplicação. O projeto da interface com o usuário corresponde à criação da UI por meio das tarefas de (i) escolher as mídias adequadas, (ii) posicionar corretamente as mídias (leiaute) e (iii) especificar o conteúdo e aspecto visual das mídias. O projeto da lógica de aplicação refere-se ao relacionamento entre os objetos de mídia podendo incluir a especificação (i) da sincronização temporal entre as mídias e (ii) da interatividade do usuário com a aplicação.

Estas atividades possuem relações de dependências entre seus artefatos desenvolvidos dentro do processo. Os fatores que influenciam na dependência estão: (i) criação, exclusão, alteração de objetos de mídia a partir da lógica de aplicação em tempo de execução; (ii) atualização dos objetos de mídia usando conteúdo dinâmico; e (iii) geração de eventos de objetos de mídia propagados para a lógica da aplicação e vice versa [15]. Sendo assim, é necessário que a ferramenta de autoria defina duas visões distintas, sendo uma para cada projeto, e que as inter-relações entre elas sejam suportadas.

O relacionamento entre os artefatos produzidos no projeto da UI e no projeto da lógica de aplicação exige uma comunicação entre o designer e o programador. As atividades que necessitam integrar os artefatos produzidos por estes dois desenvolvedores estão: (i) relacionamento dos elementos da interface com os objetos do domínio e (ii) uso dos elementos que foram definidos na UI pelo designer para realizar o projeto da lógica de aplicação, isto é, especificação do relacionamento entre os elementos.

Para cumprir estes requisitos, a ferramenta de autoria disponibiliza dois editores visuais voltados para modelar os aspectos relacionados ao projeto da interface com o usuário (UI), o projeto da lógica de aplicação e paralelamente a definição do template. Ambos suportam as tarefas de cada etapa e a partir do uso de modelos os dois projetos são integrados de maneira rápida.

A Figura 1 apresenta um *workflow* de integração entre os dois principais editores presentes na ferramenta. O fluxo de dados através da ferramenta mostra como o projeto da interface coordena-se com o projeto do software. No primeiro passo, utilizando o **editor de interface**, o designer define todos os elementos da UI que irão fazer parte da aplicação. Ele utiliza como base uma paleta com componentes visuais que podem ser arrastados para uma área de edição especifica. Assim que os elementos da UI são definidos, o projetista de software utiliza estes elementos como ponto de partida para iniciar o projeto da lógica da aplicação. Sendo assim, ele não necessita que a versão final da interface esteja totalmente pronta.

No design da interface ocorre o primeiro ponto de integração entre UI e a lógica de aplicação. Além das respectivas tarefas de edição da UI, o designer deve descrever qual será o conteúdo de cada elemento da UI. Esta tarefa acarreta na programação do mapeamento entre a UI e os objetos do domínio. O editor auxilia o designer disponibilizando uma interface para mapear o elemento da UI com o domínio do domínio.

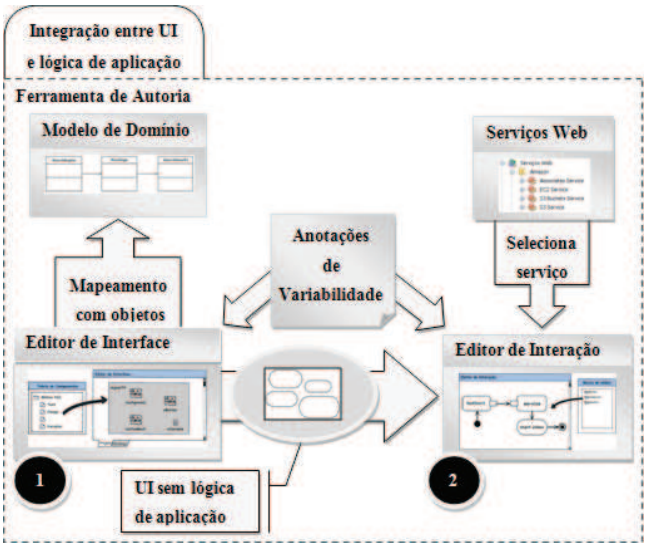

Figura 1 – Visão geral dos editores presentes na ferramenta.

No segundo passo, o **editor de interação** especifica a lógica de aplicação a partir dos elementos da UI presentes no editor de interface. O editor de interação modela visualmente o relacionamento entre as mídias NCL e Lua da aplicação tratando os aspectos de sincronização temporal e interatividade. Ele utiliza como base um grafo de blocos para modelar o comportamento da aplicação em decorrência da interação feita pelo usuário ou por eventos ocorridos nas mídias. A notação visual abstrai a sintaxe NCL/NCLua e é baseada no diagrama de ligação de sinais proposto em [14].

O editor inclui blocos para representar tanto os elementos da UI como o código imperativo de objetos NCLua (ou seja a lógica da aplicação). Eles são conectados para descrever o fluxo de ações a serem realizadas e dessa forma definindo a lógica da aplicação. Funcionalidades de serviços web também são reusadas em

termos de blocos e são selecionados a partir da paleta de serviços web.

Paralelamente a criação da aplicação, o desenvolvedor utiliza um conjunto de anotações de variabilidade para definir o template da aplicação. As anotações são usadas para descrever como os elementos do modelo realizam os aspectos comuns e variáveis do template. Um elemento pode ser anotado por dois tipos de anotações: (i) condição de presença (CP) – presença ou não de determinado elemento, por exemplo, elemento presente apenas quando selecionado para existir na aplicação instanciada; (ii) multiplicidade – repetição de elementos do template.

A aplicação final é gerada automaticamente com base na configuração escolhida pelo usuário a partir das características do template. As características da aplicação são os pontos comuns e variáveis do template em questão. A aplicação é obtida passando por duas etapas de transformação, sendo uma transformação entre modelos, para instanciar o template a partir da configuração, e por fim a geração textual, para transformar a instancia do template na aplicação NCL.

8. TRABALHOS RELACIONADOS

Quanto às ferramentas de autoria podemos destacar: EDITEC [12] e Composer [13]. O EDITEC é um editor gráfico de templates de composição hipermídia baseado na linguagem XTemplate 3.0. Seu objetivo é auxiliar o processo de criação de templates através de uma abordagem gráfica de fácil uso sem a necessidade de conhecimento da linguagem XTemplate ou NCL. O Composer não considera templates durante a autoria, mas disponibiliza recursos de edição visual e textual para auxiliar o desenvolvedor em tarefas repetitivas.

A grande vantagem destes editores é auxiliar na criação rápida de aplicações NCL. Entretanto, eles requerem do usuário conhecimento especializado da linguagem para desenvolver uma aplicação. Em ambos, a notação visual faz-se uso de muitos conceitos presentes na NCL, os quais são bastante específicos ao domínio de aplicações multimídia interativas.

Como explicado na seção 2, as ferramentas de autoria existentes não suportam a integração entre o projeto de software e o projeto de interface (UI). Outro problema é a inexistência do suporte a integração entre o código NCL e o código procedural que trata a lógica da aplicação. O uso dos serviços *web* também faz parte deste cenário de integração.

Em relação às abordagens de desenvolvimento é possível destacar o trabalho [15]. Ele enfoca o desenvolvimento de aplicações multimídia interativas usando a MML (*Multimedia Modeling Language*) como linguagem para modelagem. O principal objetivo é permitir que o processo envolva diferentes especialistas pertencentes aos três tipos de projeto existente: projeto de software; interface com o usuário e produção de mídias. Apesar de ser uma linguagem voltada para integrar UI e software, a MML não considera templates, nem serviços web no seu metamodelo. Outra deficiência é que nenhum mecanismo é criado para abstrair os conceitos da UML durante a definição do projeto de software através do diagrama de atividades. Dessa forma, a linguagem é voltada apenas para especialista em UML.

9. AGRADECIMENTOS

O autores agradecem à CAPES e ao CTIC/RNP pela suporte financeiro a esse projeto de pesquisa.

10. REFERÊNCIAS

[1] Alfonsi B., “I Want My IPTV: Internet Protocol Television Predicted a Winner,” *IEEE Distributed Systems Online,* 2005.

[2] Bachmayer S., Lugmayr A. e Kotsis G., “Convergence of collaborative web approaches and interactive TV program formats,” *International Journal of Web Information Systems*, 2010.

[3] Soares L. F. G., Moreno M. F. e Sant'Anna F., “Relating declarative hypermedia objects and imperative objects through the NCL glue language,” em *Proceedings of the 9th ACM symposium on Document engineering*, 2009.

[4] Pleub A., “Modeling the user interface of multimedia applications,” em Proceedings of the 8th international conference on Model Driven Engineering Languages and Systems, 2005.

[5] Kaskalis T., Tzidamis T. D. e Margaritis K. G., “Multimedia Authoring Tools: The Quest for an Educational Package,” *Educational Technology & Society*, 2007.

[6] Soares Neto C., “Autoria de Documentos Hipermídia Orientada a Templates,” Tese de Doutorado, 2010.

[7] dos Santos J. e Muchaluat-Saade D. C., “XTemplate 3.0: spatio-temporal semantics and structure reuse for hypermedia compositions,” *Multimedia Tools and Applications*, 2011.

[8] Czarnecki K., “Overview of generative software development,” em I*nternational conference on Unconventional Programming Paradigms*, 2005.

[9] Czarnecki K. e Antkiewicz M., “Mapping Features to Models: A Template Approach Based on Superimposed Variants,” em *International Conference on Generative Programming and Component Engineering*, 2005.

[10] Azevedo R. G. A., Neto C. S. S., Teixeira M. M., Santos R. C. M. e Gomes T. A., “Textual authoring of interactive digital TV applications,” *EuroITV '11,* 2011.

[11] Soares Neto C., de Souza C. S. e Soares L. F. G., “Linguagens computacionais como interfaces: um estudo com nested context language.,” em *IHC*, 2008.

[12] Guimarães L. R., Guimarães R. M. R. e Soares L. F. G., “Composer: Authoring Tool for iTV Programs,” *Euro iTV,* 2008.

[13] Damasceno J., Santos J. e Muchaluat-Saade D. C., “EDITEC: Editor Gráfico de Templates de Composição para Facilitar a Autoria de Programas para TV Digital Interativa,” *Simpósio Brasileiro de Sistemas Multimídia e Web,* 2010.

[14] Faison T., Event-Based Programming: Taking Events to the Limit, 1ª ed., Apress, 2011.

[15] PleuB A., “MML: A Language for Modeling Interactive Multimedia Applications,” em *International Symposium on Multimedia*, 2005.