7
Conclusão

Produtores e distribuidores de conteúdo estão imersos em diferentes ambientes e em cada ambiente existem diferentes perfis de atores com necessidades específicas. Esses atores passam por comunicadores, jornalistas, programadores e operadores de rede para transmissão e até mesmo o usuário final. Cada um desses atores tem uma visão particular sobre todo o processo, o que geralmente resulta em ferramentas focadas em atender as necessidades de um perfil específico de autor.

Considerando o uso de NCL no desenvolvimento de aplicações interativas para TVD, existem três grupos principais:

i) programadores de software, que geralmente desenvolvem aplicações complexas (inclusive as aplicações residentes nos *set-top-box*), que detém conhecimento técnico profundo sobre NCL e sua linguagem de script LUA;

ii) produtores de conteúdo, dentre os quais se encontram designers, comunicadores e jornalistas, focados na produção das mídias que farão parte da aplicação.

iii) Usuários finais, no caso de TVDi os telespectadores, que produzem conteúdo anotado com suas visões sobre determinada cena ou programa, essa anotação deve ser possibilitada por uma ferramenta embarcado no seu receptor de TVDi.

A participação de todos esses perfis de autores é fundamental para o sucesso da interatividade em sistemas de TVD. Este sucesso depende de aplicações interativas interessantes e atrativas. Estas aplicações só começam a aparecer quando o conteúdo começa a ser produzido por especialistas, como designers gráficos, comunicadores ou jornalistas. Atualmente, pelo menos no Brasil, as aplicações ainda estão sendo desenvolvidas, quase que exclusivamente, por detentores de conhecimento técnico das linguagens.

A fim de permitir o aparecimento desses produtores de conteúdo e não limitar a criação de aplicativos a programadores, é necessário o desenvolvimento de ferramentas de autoria eficientes e que tenham como alvo as necessidades desse tipo de perfil de autor.

7.1. Contribuições da dissertação

Uma ferramenta de autoria eficiente não deve prover apenas suporte aos requisitos funcionais, mas também deve levar em consideração os requisitos não-funcionais de maneira a auxiliar os autores nas tarefas relacionadas ao desenvolvimento de aplicativos. Visto anteriormente, muitas ferramentas têm o seu foco nos aspectos funcionais da autoria e falham na tentativa de prover eficiência, escalabilidade, portabilidade, etc.

Uma das contribuições deste trabalho é o levantamento de diversos requisitos não-funcionais, que foram sendo identificados na análise das várias ferramentas de autoria para documentos hipermídia existentes. Além disso, foi feita uma revisão das principais reclamações existentes nos fóruns da comunidade Ginga[3] sobre o Composer I, e percebeu-se que a maioria das reclamações girava em torno dos requisitos não-funcionais discutidos por este trabalho.

De posse desses requisitos não-funcionais, foi possível criar uma arquitetura para cobrir e suprir os aspectos não-funcionais levantados. A arquitetura proposta combina as menores funcionalidades comuns em um micro-núcleo, que serve de hospedeiro para futura extensões do sistema. Essas extensões são feitas através de plug-ins.

O micro-núcleo se torna responsável por troca de mensagens, no qual são acoplados diversos plug-ins que proveem as funcionalidades específicas aos usuários finais. Como resultado, obtém-se um ambiente altamente extensível e adaptável. Como uma das principais vantagens em relação às ferramentas existentes, essa arquitetura permite um sincronismo incremental entre as visões dos diversos plug-ins, além da validação das mensagens trocadas e controle de transação.

[3] Comunidade Ginga presente no Portal do Software Público Brasileiro. Atualmente com mais de 10.000 membros.

Além de todas as vantagens dessa arquitetura discutidas ao longo deste trabalho, uma que deve ser mencionada é a simplicidade da implementação deste micro-núcleo, que gera uma facilidade para embarcar em dispositivos com escassez de recursos, como o receptor de TVD (*set-top box*). Este micro-núcleo integrado em um STB abre um gama de possibilidades com foco em plug-ins para os telespectadores, inclusive em aplicações voltadas para TV Social.

A fim de provar o conceito e fazer uma avaliação da arquitetura proposta, o Composer II foi desenvolvido. O Composer II é a segunda versão da ferramenta de autoria voltada para documentos em NCL. Além da integração do Composer II com a implementação do micro-núcleo, foram desenvolvidos uma série de plug-ins, dessa maneira o mecanismo de comunicação, assim como aspectos de eficiência podem ser analisados.

A implementação desenvolvida por esse trabalho do micro-núcleo, assim como o Composer II e os plug-ins serão disponibilizadas na comunidade Ginga (www.softwarepublico.gov.br) sob a licença Eclipse[4]. O propósito dessa licença é fomentar o desenvolvimento de plug-ins por terceiros e ainda criar um ambiente no qual será possível criar distribuições especializadas para um determinado perfil de autor.

7.2. Avaliação dos RNFs

Com o objetivo de avaliar o quanto a arquitetura proposta e a nova implementação do Composer atendem os requisitos não-funcionais discutidos no Capítulo 3, foram criados alguns critérios de avaliação compilados na Tabela 4 abaixo.

Requisito não-funcional	Avaliação	Atendido pela arquitetura
Desempenho	Testar o desempenho das funções exercidas pelo micro-núcleo.	Foram escritos testes unitários sobre as funções do micro-núcleo e sua performance foi medida.

[4] http://www.eclipse.org/legal/epl-v10.html

Escalabilidade	Carregar diferentes plug-ins sobre o micro-núcleo.	Foram desenvolvidas 4 visões as quais foram agregadas sem problemas de performance ao núcleo.
Customabilidade	Desenvolver plug-ins com alvo em perfis de autores diferentes.	Temos uma visão textual voltada para programadores, e uma visão de leiaute e estrutural voltada para não-programadores.
Extensibilidade	Teste das APIs de extensão do núcleo	Foram desenvolvidos 4 plug-ins e 2 se encontram em desenvolvimento.
Portabilidade	Testar a implementação do micro-núcleo em diferentes plataformas	O novo Composer foi testado no Mac,Linux e Windows. QT dá suporte a plataformas embarcadas.
Confiabilidade	Testar a tolerância a falhas das funções do micro-núleo e da integração com os plug-ins.	Foram escritos testes de unidade para cada função do micro-núcleo. **Foram encontrados problemas de integração com plug-ins defeituosos**.
Manutenabilidade	Testar os passos para adição de um novo plug-in	Para adicionar um novo plug-in é necessário implementar apenas duas interfaces.

Tabela 4 - Avaliação dos RNFs sob a arquitetura

Pode-se observar, pela Tabela 4, que grande parte dos requisitos não-funcionais foram satisfeitos pela arquitetura e implementação atual do micro-núcleo e novo Composer. As soluções para os RNFs não completamente atendidos foram colocados como trabalhos futuros, detalhados a seguir.

7.3.Trabalhos Futuros

Um dos principais trabalhos futuros será o desenvolvimento de um repositório Web para os plug-ins desenvolvidos pela comunidade. Concentrando em um único lugar os plug-ins existentes para a ferramenta. A partir desse repositório será possível a criação de distribuições customizadas, onde o usuário poderá escolher os plug-ins que farão parte de sua distribuição.

Atualmente, o Composer II tem foco na edição de documentos NCL feitos somente por um único autor. Serão conduzidos estudos para a extensão dessa arquitetura de forma a dar suporte à edição colaborativa destes documentos.

O foco deste trabalho foi propor uma arquitetura para ferramentas de autoria hipermídia que suprisse as necessidades levantadas, provendo a implementação de uma ferramenta que valide essa arquitetura. Por isso, em um futuro próximo é necessário um estudo mais elaborado sobre a interface gráfica da ferramenta.

Uma falha encontrada no desenvolvimento foi a disposição espacial dos plug-ins. À medida que o número de plug-ins aumenta, a poluição visual da ferramenta se eleva. É necessário prover um mecanismo gráfico no qual o autor possa desabilitar rapidamente plug-ins para um determinado documento. Por exemplo, quando o autor está editando um documento NCL que descreve somente os conectores utilizados por um outro documento NCL, não existe a necessidade de plug-ins de leiaute ou temporal serem lançados para este documento.

Atualmente, o micro-núcleo é a única ponte de comunicação entre os plug-ins. Mais do que isso, essa comunicação está restrita às APIs de troca de mensagens entre os plug-ins e o micro-núcleo. É preciso a criação de um mecanismo de comunicação entre plug-ins de maneira direta e sem restrições. Uma extensão na arquitetura a fim de prover esta comunicação deverá ser estudada.

O principal problema encontrado na atual implementação do micro-núcleo é a confiabilidade na integração com plug-ins defeituosos. O micro-núcleo não tem tolerância a falhas levantadas por plug-ins, dessa maneira um plug-in de terceiro com falhas compromete toda ferramenta. Um trabalho futuro é desenvolver um micro-núcleo que utilize paradigma de comunicação via processos.