Benchmark Suite para Ginga-NCL Execução e Análise na Plataforma Android

Alex Servino
aservino@inf.ufes.br

Magnos Martinello
magnos@inf.ufes.br

Guilherme M. Nogueira
gmaio@ime.usp.br

Laboratório de Pesquisas em Redes Multimídia (LPRM)
Departamento de Informática/Centro Tecnológico
Universidade Federal do Espírito Santo - UFES
Campus Universitário de Goiabeiras, Prédio: CT-VII
Av. Fernando Ferrari, 514, 29060-970 - Vitória/ES, Brasil

Instituto de Matemática e Estatística
Dep. de Ciência da Computação
Universidade de São Paulo
Rua do Matão, 1010 - São Paulo/SP

ABSTRACT

Recently, there has been in Brazil a few number of devices capable of receiving the digital television signal. However, a few number of devices has the support of the Brazilian Digital Television System's *middleware*, named Ginga, and usually there is no benchmark suite for validation tests, which affects the execution of interactive multimedia content. This paper reports a *benchmark suite* performed on the implementation of Ginga-NCL for Android based devices, developed at UFES. The obtained results allow to quantify some aspects of the middleware's performance, and particularly to sugest best practices for developing NCL applications in mobile scenarios.

RESUMO

Hoje já existem no Brasil aparelhos que permitem a recepção do sinal de TV digital. No entanto, poucos dispositivos estão equipados com o Ginga, *middleware* adotado pelo Sistema Brasileiro de Televisão Digital - SBTVD, sem necessariamente haver uma bateria de testes de conformidade, o que afeta a execução de conteúdo multimídia interativo. Esse trabalho descreve um *benchmark suite* realizado sobre a implementação do Ginga-NCL para dispositivos baseados no sistema Android, desenvolvida pela UFES. Os resultados obtidos permitem quantificar alguns aspectos referentes ao desempenho do middleware, e particularmente determinar boas práticas para desenvolvedores de aplicações NCL em cenários móveis.

Categories and Subject Descriptors

I.7.2 [**Document Preparation**]: Miscellaneous

General Terms

Middleware, Design, Benchmark

Keywords

Benchmarking, Ginga NCL, SBTVD, Android OS

WebMedia'11: Proceedings of the 17th Brazilian Symposium on Multimedia and the Web. Short Papers.
October 3 -6, 2011, Florianópolis, SC, Brazil.
ISSN 2175-9650.
SBC - Brazilian Computer Society

1. INTRODUÇÃO

O acesso ao conteúdo televisivo interativo a partir de dispositivos portáteis multifuncionais, como celulares e smartphones, é uma das grandes apostas para o crescimento da TV digital aberta no Brasil. Hoje já existem no país diversos fabricantes de dispositivos portáteis, tais como a Samsung e LG, cujos aparelhos permitem a recepção do sinal de TV digital. No entanto, esses dispositivos não estão equipados com o Ginga [9, 5], software nacional adotado como padrão de *middleware* pelo Sistema Brasileiro de Televisão Digital - SBTVD. O Ginga, em suas duas modalidades, uma declarativa, o Ginga-NCL [9], e outra imperativa, o Ginga-J [5], fornece uma série de facilidades para a construção e a execução de aplicações interativas em ambiente de TV digital aberta. Dessa forma, os dispositivos portáteis disponíveis no mercado brasileiro, embora se aproveitem da qualidade superior de áudio e vídeo inerente às tecnologias digitais, não suportam a interatividade.

No cenário de mobilidade, a escolha do sistema operacional embarcado no dispositivo portátil, assim como de suas plataformas de desenvolvimento associadas, tornam-se questões proeminentes no processo de desenvolvimento de novas implementações Ginga. Iniciativas acadêmicas têm implementado a máquina de execução Ginga-NCL em dispositivos móveis, tendo diferentes plataformas base, por exemplo o sistema operacional Symbian [8] em [7] e a plataforma Android em [6].

A plataforma Android lançada em 2008 pelo Google, através do consórcio *Open Handset Alliance* [3] é uma plataforma de código aberto e que não está vinculada a apenas um fabricante. Por essa razão, mas também pelo fato do código fonte estar publicamente disponível em [10], o presente trabalho tem como ponto de partida a implementação do Ginga-NCL na plataforma Android.

O objetivo deste trabalho em andamento é projetar, executar e analisar um conjunto de aplicativos em NCL (*benchmark suite*) sobre a implementação do Ginga-NCL para dispositivos portáteis baseados no sistema operacional Android [6, 10]. O suite de testes é projetado para avaliar características de desempenho do middleware levando-se em conta fatores referentes i) aos recursos da linguagem NCL, ii) ao impacto dos objetos de mídia, e iii) ao ambiente de execução incluindo o sistema operacional e as características do dispositivo.

As demais seções do artigo estão organizadas da seguinte forma: Seção 2 descreve as características do *benchmark suite* proposto para avaliar o Ginga-NCL. Seção 3 discute a metodologia de análise proposta. Seção 4 contempla o estudo experimental realizado. Por fim, a seção 5 conclui o artigo apresentando as considerações finais e as perspectivas de trabalhos futuros.

2. BENCHMARK SUITE PARA GINGA-NCL

O universo das aplicações Ginga pode ser classificado como um conjunto composto por aplicações declarativas e aplicações procedurais. Aplicações declarativas são tratadas pelo subsistema Ginga-NCL e aplicações procedurais pelo subsistema Ginga-J. De acordo com as normas NBR-15606-4 e NBR-15606-5 [2, 1] definidas para o Sistema Brasileiro de Televisão Digital (SBTVD), dentre os dois subsistemas, apenas o declarativo é obrigatório para dispositivos portáteis. Esse trabalho concentra-se em realizar benchmarks no middleware declarativo Ginga-NCL, mais especificamente em dispositivos portáteis equipados com o sistema operacional Android.

Do ponto de vista conceitual, o *middleware* é uma camada na arquitetura que provê serviços para a camada de aplicação. No caso do Ginga-NCL, esse *middleware* é responsável pela execução das aplicações NCL, na qual pode-se ver a divisão em dois subsistemas [9]: a máquina de apresentação NCL e o núcleo Ginga. Considerando o sistema Android, sua componentização é ilustrada na Figura 1 [6].

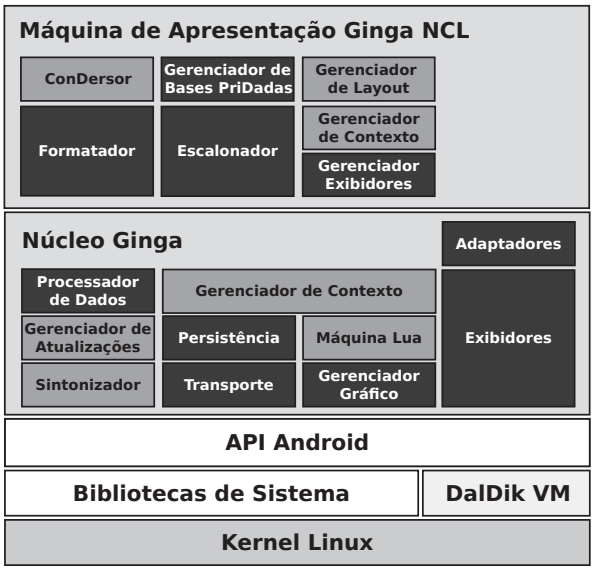

Figura 1: Arquitetura Ginga NCL para dispositivos portáteis

Como a máquina de apresentação Ginga-NCL é responsável pela execução de aplicações multimídia, o tempo de execução de uma determinada aplicação NCL depende de múltiplos fatores, dentre esses fatores os recursos providos pela linguagem NCL e os parâmetros referentes aos objetos de mídia. Já outros fatores, como o ambiente de execução incluindo características do sistema operacional, assim como recursos disponíveis no dispositivos tendem a influenciar notadamente o núcleo Ginga.

O *benchmark suite* projetado está dividido em três categorias: aplicativos que testam o suporte oferecido pela linguagem NCL; aplicativos que exploram as características dos objetos de mídia; e aplicativos que analisam o impacto do ambiente de execução;

No que se referente ao suporte oferecido pela linguagem NCL, os benchmarks projetados consideram a exibição de múltiplas imagens:

- Lançadas simultaneamente comparando-se uma exibição em série versus em paralelo;
- Sobrepostas, ocupando a mesma área do display do smartphone;
- Concatenadas em uma única imagem equivalente;
- Variando-se o tamanho da área de exibição.

No caso do conjunto de aplicativos referente aos objetos de mídia, foi considerado o tamanho e as dimensões da imagem que será exibida. No que concerne o ambiente de execução, foram projetados benchmarks levando-se em conta:

- o custo para a criação, gerência e escalonamento da thread relativa à mídia (custo de gerência de thread) na perspectiva do sistema operacional;
- a relação do tempo de preparo de uma mídia com a quantidade de mídias a serem exibidas simultaneamente;
- o impacto do intervalo entre chamadas do Garbage Collector (GC) no tempo de preparo.

3. METODOLOGIA DE TESTES

Na arquitetura Ginga-NCL, o processo de iniciar um evento do tipo Apresentação é custoso e envolve a preparação da mídia. A metodologia de testes proposta é destinada a investigar o comportamento do middleware considerando como métrica o tempo de preparação durante a execução de documentos hipermídia NCL, indicando o atraso relativo a esse processo.

Considerando que esse tempo de preparo do exibidor pode variar significativamente em função de inúmeros fatores, foram elaboradas baterias de testes envolvendo o benchmark suite discutido previamente, a fim de quantificar a variação do tempo de preparo. Após sucessivas execuções de cada benchmark, os dados são coletados e análises são feitas obtendo medidas estatísticas tais como mediana, desvio padrão e apresentando os resultados em gráficos do tipo boxplot representando os diferentes quantis. Para minimizar o efeito aleatório nos resultados, foram feitas pelo menos 20 execuções de cada benchmark.

Ao final das análises, algumas considerações são elaboradas procurando trazer explicações dos resultados obtidos. A idéia é que os resultados obtidos permitam auxiliar desenvolvedores de aplicações multimídia no projeto de aplicações NCL adequadas ao ambiente de TVD móvel.

4. EXECUÇÃO E ANÁLISE DOS TESTES

Essa seção relata os principais resultados na execução das baterias de testes usando como base o benchmark suite proposto. Os resultados obtidos buscam sugerir características de desempenho do ponto de vista do middleware e também recomendações para o desenvolvimento adequado de aplicações NCL em dispositivos portáteis.

4.1 Custo de gerência de threads

O Ginga-NCL vincula cada componente de mídia com uma thread do sistema operacional. Dessa forma, em cada componente de mídia existe um custo embutido envolvendo a criação, gerência das estruturas de dados, e escalonamento de sua thread. Para avaliar isoladamente esses recursos do sistema, o benchmark projetado tem seu foco concentrado basicamente no custo da thread ao construir imagens de dimensões 1x1 pixel para serem exibidas. A Figura 2 permite quantificar esse custo, no qual pode-se observar um aumento no atraso de preparação em função da quantidade de componentes de mídia sendo simultaneamente exibidos. A mediana do atraso é de aproximadamente 60ms para 20 imagens e de 80ms para 40 imagens, mas também percebe-se que há uma variação considerável do atraso (tamanho do desvio padrão) a partir de 5 imagens. Vale ressaltar que para um número maior do que 40 imagens, por exemplo o smartphone T-Mobile G1, não foi capaz de executar uma aplicação que exibe na tela 50 imagens de dimensões 1x1 pixel.

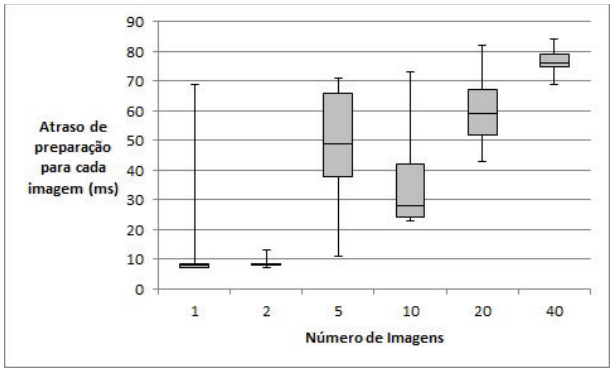

Figura 2: Exibição de imagens de 1x1 pixel.

4.2 Mídias Paralelas versus Sequenciais

Ao realizar a exibição de múltiplas mídias simultaneamente, a linguagem NCL permite que as mídias sejam preparadas de duas formas à escolha do desenvolvedor. A primeira forma é fazer exibição paralela das mídias, ou seja, requisitar instantaneamente que todas as mídias sejam preparadas. A idéia é utilizar o mesmo elo para todas as mídias, com o papel condicional idêntico para todas, e com a ação iniciadora vinculada a cada uma delas. A segunda maneira é fazer exibição em série das mídias, ou seja, requisitar que as mídias sejam preparadas uma a uma. Neste caso, utiliza-se o elo desejado apenas para a primeira mídia a ser exibida, e em seguida o elo "onBeginStart" nas demais mídias, sempre vinculando a condição "onBegin" com o componente de mídia anterior associado com a ação "start".

A Figura 3 mostra os testes realizados do benchmark que compara a exibição paralela com exibição em série. Para imagens pequenas (2,11kB), o tempo de preparação usando o recurso da linguagem NCL para exibição de mídias em paralelo é consideravelmente menor do que em série, enquanto que, para imagens grandes (170kB), a diferença entre as duas formas é basicamente desprezível. Apesar de se tratar de um resultado esperado, é interessante observar a magnitude da diferença no atraso. Para a exibição de 35 imagens pequenas seriais, o atraso é da ordem de 3.5 seg comparada com 2.1 segundos para imagens exibidas em paralelo. Apesar de haver uma variação maior no atraso para a exibição em paralelo, ainda assim é mais recomendável especialmente à medida que o número de imagens simultâneas aumenta.

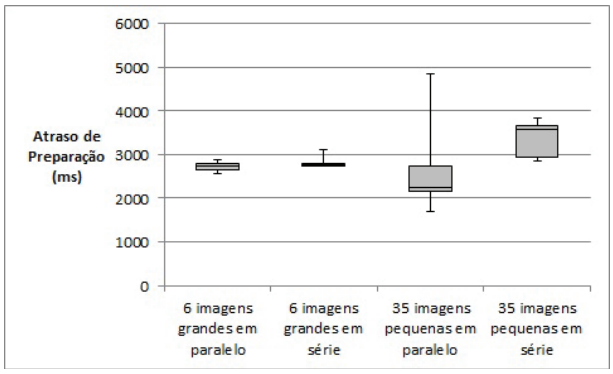

Figura 3: Exibição de imagens em série e em paralelo.

4.3 Concatenação de Imagens

Para analisar o efeito da concatenação de imagens, o benchmark testado mostra a exibição de múltiplas imagens apresentadas uma a uma em relação ao atraso de uma única imagem resultante da concatenação do mesmo conjunto de imagens. Essa imagem resultante terá o mesmo custo computacional gráfico que a soma das demais, porém terá apenas o custo de gerência de uma thread. A Figura 4 apresenta o resultado desse benchmark, na qual para o caso de 6 imagens, a diferença no atraso foi de basicamente 1 seg., e para 35 pequenas imagens (2,11kB em tamanho), o atraso de preparação foi 20 vezes menor para a exibição de uma única imagem resultante da concatenação das 35.

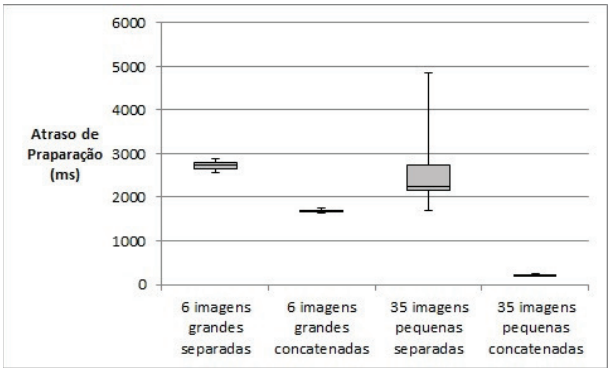

Figura 4: Imagens separadas versus imagens concatenadas.

4.4 Tamanho da Região

Para analisar o impacto do tamanho da região de um objeto de mídia, o benchmark testado varia o tamanho da região para uma determinada imagem. A Figura 5 apresenta o resultado, na qual pode-se notar que o tamanho da região não afeta substancialmente o atraso de preparação.

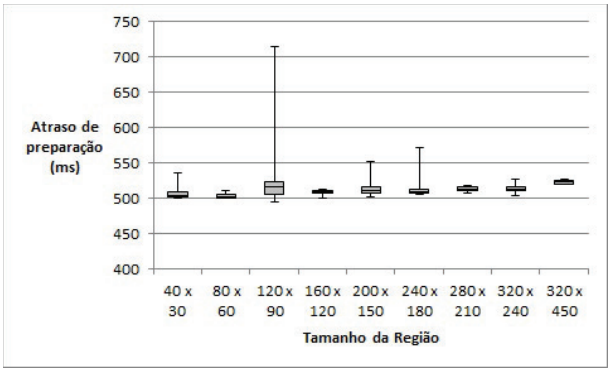

Figura 5: Variação do tamanho da região.

4.5 Impacto do tamanho de imagem

Para analisar o impacto que o tamanho de imagem (em kB) tem no atraso de preparação, o benchmark testado seleciona imagens de diferentes dimensões. Para uma mesma dimensão, são geradas imagens de diferentes tamanhos através da redução de qualidade. Para permitir uma estimativa estatística do efeito do tamanho da imagem sobre o atraso, um método de regressão linear foi utilizado sobre o conjunto de atrasos medidos. A Figura 6 apresenta um dos resultados desse benchmark para imagens de dimensões 560x374. Apesar do tamanho padrão de resolução *one-seg* ser 320x240, esse valor maior de resolução (560x374) foi escolhido por caber em um display de aparelho portátil e exigir um pouco mais da parte gráfica. Outros testes foram omitidos devido as restrições de espaço.

Na Figura 6, a equação de reta estimada pela regressão (y = 0.4528 x + 172.81), com um coeficiente de determinação da re-

gressão R^2 de 0.9737, mostra que a relação atraso/tamanho de imagem é de 45 ms/KB. Considerando uma série de regressões foi observado que, para o dispositivo testado (T-Mobile G1), a relação atraso/tamanho varia entre 40 e 60 ms/kB com intervalo de confiança de 95%. Outra observação é que quanto maior a dimensão da imagem original, maior é a parcela constante (b=172.81) do atraso obtido na equação da reta estimada pela regressão. Por exemplo, para resoluções idealizadas de 900x600 e 1024x768, os valores são b=240 e b=343, respectivamente.

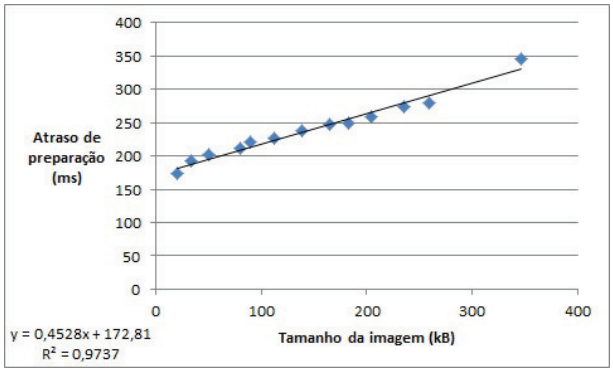

Figura 6: Variação do tamanho da imagem de dimensões 560x374 pixels.

4.6 Impacto do Garbage Collector

Para analisar o impacto da frequência de chamadas ao Garbage Collector, o benchmark testado analisa o intervalo de tempo entre chamadas do GC durante a execução de diversas aplicações da suite de testes. A Figura 7 ilustra o instante de chamada do GC em traços azuis pontilhados para uma aplicação NCL que exibe 14 imagens pequenas (2kB) simulteamente. Nota-se que o tempo de preparação das mídias está relacionado ao intervalo entre chamadas ao GC. Para um intervalo grande, há uma maior probabilidade de saturação dos recursos no dispositivo, isso leva a um aumento no tempo de execução da thread. Para intervalos menores, a liberação de recursos pelo GC pode permitir uma diminuição no tempo de execução, evidentemente se não houver uma concorrência acentuada entre as próprias threads do sistema. Outra observação é que o início e o final das mídias tendem a se concentrar em instantes próximos a uma chamada de GC. Essa característica peculiar é explicada pela liberação de recursos (memória) necessários para o término da execução dessas threads concorrentes.

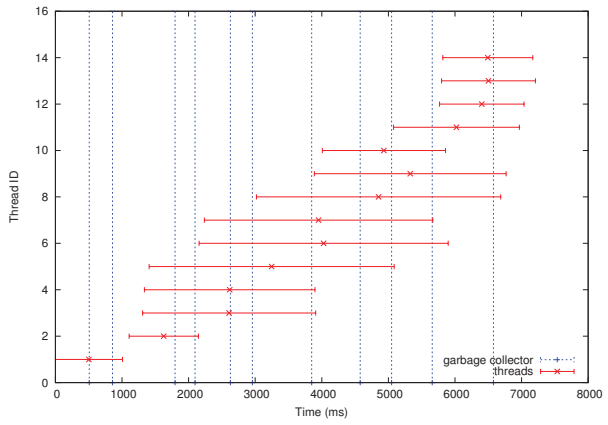

Figura 7: Intervalo entre chamadas de GC consecutivas para aplicação de 14 imagens pequenas simultâneas.

5. CONCLUSÕES E TRABALHOS FUTUROS

A avaliação funcional e de desempenho da implementação Ginga-NCL está ancorada em um série de experimentos e medições executados a partir de um conjunto de aplicações NCL projetadas para testar o middleware, denominado de *benchmark suite*. A implementação Ginga-NCL Android disponível em [10] tem sido usada como ponto de partida para a execução de baterias de testes sistêmicos para garantir a robustez do middleware. Os resultados obtidos permitem analisar alguns aspectos de desempenho da implementação Ginga-NCL em dispositivos portáteis Android e orientar os desenvolvedores NCL no projeto de suas aplicações.

Apesar da máquina de execução Lua ser requerida pelo padrão de middleware do SBTVD em dispositivos portáteis, e mesmo estando parcialmente implementada em Android [4], testes que combinem aplicações NCL com LUA ainda precisam ser incluídos no benchmark suite. Para trabalhos futuros, espera-se estender o benchmark suite incluindo novos testes, por exemplo para diferentes versões do Android, testes que usem diferentes recursos da linguagem NCL (como a resposta a input de usuário), testes do uso de bateria, avaliação de desempenho de vídeos utilizando como parâmetro a taxa de quadros-por-segundo (FPS) e a reprodução do benchmark em diferentes dispositivos.

Do ponto de vista de implementação, espera-se implementar uma versão do middleware Ginga-NCL em C++ no Android utilizando o *Native Development Kit (NDK)* a partir da versão 2.2 do Android.

Agradecimentos

Esse trabalho teve apoio financeiro da CAPES.

6. REFERÊNCIAS

[1] ABNT/NBR15606-4. *Televisão digital terrestre - Codificação de dados e especificações de transmissão para transmissão digital - Parte 4: Ginga-J - Ambiente para a execução de aplicações procedurais.*, 2010.

[2] ABNT/NBR15606-5. *Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital Parte 5: Ginga-NCL para receptores portáteis - Linguagem de aplicação XML para codificação de aplicações.*, 2011.

[3] Open Handset Alliance. `http://www.openhandsetalliance.com/press\_110507.html`.

[4] Guilherme de Maio Nogueira. GingaNCL for Android: Lua code execution with graphical support. *Monografia apresentada para obter grau B.Sc. em Ciência da Computação, pela Universidade Federal do Espírito Santo (UFES)*, 2011.

[5] Guido Lemos de Souza Filho, Luiz Eduardo Cunha Leite, and Carlos Eduardo Coelho Freire Batista. Ginga-j: The procedural middleware for the brazilian digital tv system. *Journal of the Brazilian Computer Society*, 12(4):47–56, march 2007.

[6] G. Daher e G. Nogueira e G. Comarela e F. Fabris e M. Martinello e J. G. Pereira Filho. Ginga-ncl em dispositivos portáteis: Uma implementação para a plataforma android. In *XVI Simpósio Brasileiro de Sistemas Multimídia e Web - WebMedia2010*, Belo Horizonte, MG, Brasil, 2010.

[7] Vítor Medina Cruz e Marcio Ferreira Moreno e Luiz Fernando Gomes Soares. Ginga-ncl: Implementação de referência para dispositivos portáteis. In *XIV Simpósio Brasileiro de Sistemas Multimídia e Web - WebMedia2008*, pages 67–74, Vila Velha, ES, Brasil, 2008.

[8] Symbian Developer Network. `http://developer.symbian.com`.

[9] Luiz Fernando Gomes Soares, Rogério Ferreira Rodrigues, and Marcio Ferreira Moreno. Ginga-ncl: the declarative environment of the brazilian digital tv system. *Journal of the Brazilian Computer Society*, 12(4), March 2007.

[10] Ginga Mobile Website. `http://gingamobile.lprm.inf.ufes.br`.