Universidade Federal de Pernambuco
Graduação em Ciência da Computação
Centro de Informática

GINGAWAY – UMA FERRAMENTA PARA CRIAÇÃO DE APLICAÇÕES GINGA-NCL INTERATIVAS PARA TV DIGITAL

Trabalho de Graduação

Mauro Fernando de Holanda Beltrão Filho

Recife, 3 de dezembro de 2008

Universidade Federal de Pernambuco
Graduação em Ciência da Computação
Centro de Informática

GINGAWAY – UMA FERRAMENTA PARA CRIAÇÃO DE APLICAÇÕES GINGA-NCL INTERATIVAS PARA TV DIGITAL

Trabalho de Graduação

Monografia apresentada ao Centro de Informática da Universidade Federal de Pernambuco por Mauro Fernando de Holanda Beltrão Filho, sob a orientação do Prof. PhD. Carlos André Guimarães Ferraz, como requisito para a obtenção do grau de bacharel em Ciência da Computação.

Folha de aprovação

GINGAWAY – UMA FERRAMENTA PARA CRIAÇÃO DE APLICAÇÕES GINGA-NCL INTERATIVAS PARA TV DIGITAL

TRABALHO DE GRADUAÇÃO

Banca examinadora:

_____

Carlos André Guimarães Ferraz – Orientador

_____

André Luís de Medeiros Santos – Avaliador

Resumo

A televisão é um meio de comunicação da maior importância, sobretudo no Brasil, onde passa atualmente por uma revolução, com a implantação do SBTVD, o Sistema Brasileiro de TV Digital. Entre as principais vantagens da nova tecnologia está a interatividade, que permite aos telespectadores o uso de aplicações através do controle remoto.

Com a iminente consolidação da TV digital brasileira, há a expectativa de uma grande demanda por aplicações interativas. Para atender a tal demanda, fazem-se necessárias ferramentas especializadas nessa nova tecnologia e que supram a necessidade de eficiência no desenvolvimento dessas aplicações.

Este trabalho realiza um estudo das necessidades envolvidas no desenvolvimento desse tipo de aplicação bem como dos principais ambientes de desenvolvimento disponíveis, propondo o Gingaway: um ambiente de desenvolvimento de aplicações interativas para a TV digital brasileira que visa a atender às necessidades colocadas.

Palavras-chave: TV digital, SBTVD, interatividade, aplicações, ambiente de desenvolvimento.

Abstract

Television is a mass media of highest importance, mostly in Brazil, where it is actually going through a revolution, with the deployment of the SBTVD, the Brazil's digital TV system. Among the main advantages of the new technology is interactivity, that allows viewers to use applications throught the remote control.

Given the imminent consolidation of the Brazil's digital TV, there is the expectation of a great demand for interactive applications. In order to attend such a demand, it is necessary tools specialized in the new technology, that fill the needs of efficiency of development of these applications.

This work realizes a study of the needs related to the development of this kind of applications as well as of the main available development environments, and proposes Gingaway: a development environment for interactive applications for Brazilian digital TV that intends to address the imposed needs.

Keywords: digital TV, SBTVD, interactivity, applications, development environment.

Agradecimentos

A Deus, por tudo.

Aos meus familiares, bem como a Veruska e sua família, por todo o amor, apoio e paciência, e por tornarem a minha vida melhor a cada dia.

Aos meu amigos, tão importantes para mim.

Ao meu orientador, Carlos Ferraz, por me ajudar a conduzir este trabalho da melhor maneira.

Aos meus amigos do trabalho, com os quais eu divido preciosas horas diariamente, em especial a Andrino, pela orientação, e a Caio, pela ajuda.

Enfim, a todos que contribuíram com este trabalho.

Índice

Lista de quadros

Lista de figuras

1. Introdução

A televisão é um forte meio de comunicação, presente no cotidiano de grande parcela da população em todo o mundo. Isso se verifica sobretudo no Brasil, onde mais de 90% dos domicílios possui ao menos um aparelho televisivo [1], e as pessoas assistem à TV durante cerca de cinco horas por dia em média [2]. Trata-se, portanto, de um grande mercado, que movimenta bilhões de reais por ano [3].

A televisão brasileira passa atualmente por um importante momento de sua história: a transição do padrão de transmissão analógico para o digital. Trata-se da maior revolução na área desde a sua implantação [4]. O padrão digital traz diversas vantagens em relação ao analógico, entre as principais estão: melhor qualidade de imagem e de som, possibilidade de aumento do número de canais e interatividade.

A interatividade permite novas experiências no uso da televisão. Os telespectadores podem passar a usar a TV de forma mais ativa, explorando conteúdo além do áudio-visual tradicional. Tal conteúdo é provido por *software* que pode chegar à TV junto com o áudio e o vídeo. Desse modo, usando o controle-remoto, o usuário passa a poder, por exemplo: navegar por informações, realizar compras, participar de enquetes ou promoções etc.

A adição de interatividade à programação pode agregar valor, podendo representar um importante diferencial competitivo. Tal diferencial pode ser explorado não só pelas emissoras, para enriquecer o conteúdo de seus programas, como também pelos anunciantes, que podem veicular propagandas interativas que atraiam os telespectadores. Formas alternativas de interatividade já tem sido exploradas pelas emissoras ainda na TV analógica, através de meios alternativos, sobretudo o celular [5].

1.1. Sistema Brasileiro de TV Digital Terrestre

Baseado no ISDB-T (*Integrated Services Digital Broadcasting Terrestrial*), o padrão de transmissão japonês, o padrão brasileiro de TV digital é conhecido como SBTVD-T (Sistema Brasileiro de TV Digital Terrestre). Ele está sendo desenvolvido de maneira conjunta pelo Fórum do SBTVD-T, que é formado por emissoras, indústria e entidades de ensino e pesquisa [6].

A principal particularidade do padrão brasileiro é o Ginga [7], sua plataforma de interatividade, sobre a qual rodam as aplicações interativas. O Ginga é definido por uma norma da ABNT [8], e consiste numa especificação de *middleware* baseada em padrões abertos. Os aparelhos

receptores de TV digital são capazes de executar as aplicações interativas baseadas no padrão brasileiro desde que contenham *software* que contemple a especificação do Ginga.

O Ginga se divide em dois subsistemas: o Ginga-NCL e o Ginga-J. Cada um deles provê um ambiente de execução distinto para as aplicações interativas. No caso do Ginga-NCL, as aplicações são escritas nas linguagens NCL [9] e Lua [10, 11], enquanto no caso do Ginga-J elas são escritas em Java [12]. Já que os dois subsistemas se comunicam, é possível também escrever aplicações que rodem sobre ambos os subsistemas, feitas parte em NCL e Lua, e parte em Java.

1.1.1. Ginga-NCL

NCL (*Nested Context Language*) é uma linguagem declarativa, baseada em XML [13], especializada na descrição de aplicações interativas. Documentos NCL definem, através de seus elementos, o comportamento das aplicações, sendo a base das aplicações do padrão Ginga-NCL.

Por ser uma linguagem declarativa, NCL possui a limitação de não permitir a descrição de procedimentos. Para contornar tal limitação, o Ginga-NCL adotou também a linguagem Lua.

Lua é uma linguagem procedural pequena e leve, muito utilizada para estender com *scripts* aplicações mais complexas. É justamente nesse sentido que Lua é usada no Ginga-NCL. *Scripts* Lua adicionam procedimentos às aplicações baseadas em NCL, permitindo computações complexas e facilitando a atividade de programação.

1.2. Demanda por ferramentas

Em geral, estão disponíveis ferramentas que facilitam muito o trabalho dos programadores, como é o caso das IDE (*Integrated Development Environment*). As IDE são aplicações sofisticadas que integram muitas das ferramentas utilizadas durante o processo de desenvolvimento. Entre essas ferramentas, geralmente se incluem: editores de código-fonte, compiladores e ferramentas de *build* e de *debug*. Além disso, as IDE automatizam muitas das atividades cotidianas, facilitando o trabalho e tornando-o mais ágil.

Entre as IDE mais populares, destacam-se as IDE Eclipse [14], por ter código aberto, ter propósito geral e ser facilmente extensível, possuindo uma arquitetura de suporte a extensões via *plug-ins*.

Com a consolidação da TV digital brasileira, há a expectativa de uma grande demanda por aplicações interativas. Para atender a tal demanda, fazem-se necessárias ferramentas especializadas nessas novas tecnologias e que supram a necessidade de eficiência no desenvolvimento dessas

aplicações, tal como as IDE.

1.3. Gingaway

Este trabalho apresenta o Gingaway, uma ferramenta especializada no desenvolvimento de aplicações interativas para a TV digital brasileira, especificamente para o padrão Ginga-NCL. O Gingaway, tal qual uma sofisticada IDE, procura contemplar o trabalho de implementação com uma gama de ferramentas integradas, proporcionando a requerida eficiência.

Para esse fim, são estudadas as necessidades relacionadas à implementação das aplicações interativas e as dificuldades encontradas pelos desenvolvedores com o uso das ferramentas disponíveis. São analisados também os recursos mais importantes presentes nas principais IDEs.

As seções seguintes apresentam o trabalho realizado no desenvolvimento do Gingaway:

- Seção *2. Trabalhos relacionados* – O estudo de *software* relacionado existente;
- Seção *3. Requisitos* – O estudo dos requisitos para o Gingaway;
- Seção *4. Implementação* – Os detalhes da implementação do Gingaway;
- Seção *5. Exemplo de utilização* – Um exemplo de utilização do Gingaway que ilustra suas ferramentas;
- Seção *6. Conclusões e trabalhos futuros* – Conclusões e sugestões de trabalhos futuros.

2. Trabalhos relacionados

Esta seção aborda o estudo realizado acerca do *software* existente relacionado ao Gingaway.

O Gingaway, por ser uma ferramenta de desenvolvimento, pertence a uma vasta classe de programas, que apesar de muito variados, tem o objetivo comum de tornar eficiente a atividade de programação. Este trabalho realizou então um estudo das principais ferramentas de desenvolvimento relacionadas, a fim de reunir no Gingaway as melhores características encontradas. Esse estudo é resumido a seguir, sem profundidade de detalhes; o intuito é apresentar apenas o que for importante para a compreensão do trabalho realizado.

Todas as ferramentas estudadas são multi-plataforma e foram analisadas em ambiente GNU/ Linux, Ubuntu Desktop 8.10.

2.1. Ambientes integrados de desenvolvimento (IDE)

Um avanço significativo pode ser observado com o passar do tempo, de modo que nota-se hoje a existência de ferramentas de alto nível, capazes de agilizar em muito o trabalho, bem mais do que em tempos atrás [15]. Entre as melhores ferramentas, destacam-se as IDE, que procuram prover aos programadores um amplo ferramental integrado em um mesmo ambiente, facilitando o trabalho e automatizando várias tarefas repetitivas, tornando ágil a atividade de desenvolvimento de *software*.

Entre as ferramentas inclusas nas IDE, geralmente se encontram: editores especiais para diversos tipos de arquivos, sobretudo arquivos de código-fonte, compiladores, ferramentas de *build* automático, de depuração avançada, emuladores, assistentes para criação de novos arquivos e de novos projetos, entre outras. Os editores de código-fonte geralmente possuem recursos avançados, específicos ao tipo de código ao quais se destinam, tais como: coloração de texto e marcação de problemas conforme a sintaxe, sugestão de código, auto-formatação e refabricação (*refactoring*) automáticas do código, navegação avançada pelo código etc.

2.2. Eclipse

O Eclipse é um projeto que produz diversos sub-projetos de *software* de código aberto (licença EPL – *Eclipse Public License*), sendo o principal deles a Plataforma Eclipse (*Eclipse Platform*), na qual se baseiam as IDE Eclipse.

A Plataforma Eclipse é uma plataforma de *software* madura, cujo propósito é servir de base

para criação de uma variedade de tipos de aplicações, principalmente ferramentas de desenvolvimento. A plataforma possui uma arquitetura bastante sofisticada, extremamente modular e extensível, baseada em *plug-ins*. Ela não é, em si, um produto final; sua finalidade é prover infra-estrutura para a criação de produtos a partir da adição de *plug-ins* específicos [16]. As próprias IDE Eclipse consistem num conjunto de ferramentas adicionadas como *plug-ins* sobre a plataforma básica. Por exemplo, no caso da IDE Eclipse para Java, o conjunto de *plug-ins* que adicionam as ferramentas Java à plataforma é chamado de JDT (*Java Development Tools*) [17].

2.2.1. Estendendo o Eclipse

A arquitetura estensível da Plataforma Eclipse baseia-se em *pontos de extensão* e *extensões*. Um ponto de extensão define um ponto onde pode ser adicionada funcionalidade; uma extensão é a implementação de funcionalidade para um ponto de extensão específico.

O núcleo da plataforma consiste em um engenho de execução que carrega os *plug-ins* disponíveis. Os *plug-ins* estendem a plataforma provendo extensões para pontos de extensão já definidos, e usam as API disponíveis. Por sua vez, *plug-ins* também podem definir pontos de extensão e prover API para que outros *plug-ins* os estendam, e assim sucessivamente.

O *Plug-in Development Environment* (PDE) [18] é um projeto do Eclipse que provê à IDE Java do Eclipse uma série de ferramentas que tornam fácil o desenvolvimento de *plug-ins* para a plataforma.

Os *plug-ins* consistem principalmente em quatro elementos: o arquivo *plug-in manifest*, código Java, bibliotecas e recursos (arquivos texto, multimídia etc.). O arquivo *plug-in manifest* é um arquivo de nome “plugin.xml” presente em todo *plug-in*, que descreve os detalhes do *plug-in* para a plataforma, entre outros: suas extensões, pontos de extensão e dependências. Durante o desenvolvimento, o PDE cria um projeto Java para cada *plug-in*.

A plataforma possui também o conceito de *features*, que são simplesmente conjuntos de *plug-ins*. *Features* são usadas quando um grupo de *plug-ins* provê em conjunto uma certa funcionalidade. Assim como os *plug-ins*, cada *feature* também corresponde a um projeto Java.

2.2.2. IDE Eclipse

As IDE baseadas na plataforma Eclipse, principalmente a IDE Java, estão entre os mais avançados e populares ambientes de desenvolvimento. Utilizando os pontos de extensão e a API providos pela plataforma, é possível construir IDE avançadas para diferentes propósitos. O JDT, por

exemplo, adiciona ao Eclipse um amplo e avançado ferramental para desenvolvimento Java.

São apresentados a seguir os principais pontos de extensão utilizados na construção de IDE baseadas na Plataforma Eclipse.

- Assistente de criação de projeto – Um assistente de criação de projeto é uma ferramenta que cria um novo projeto no ambiente baseando-se nas informações entradas pelo usuário. A Figura 1 ilustra o assistente de criação de projeto Java contribuído pelo JDT.

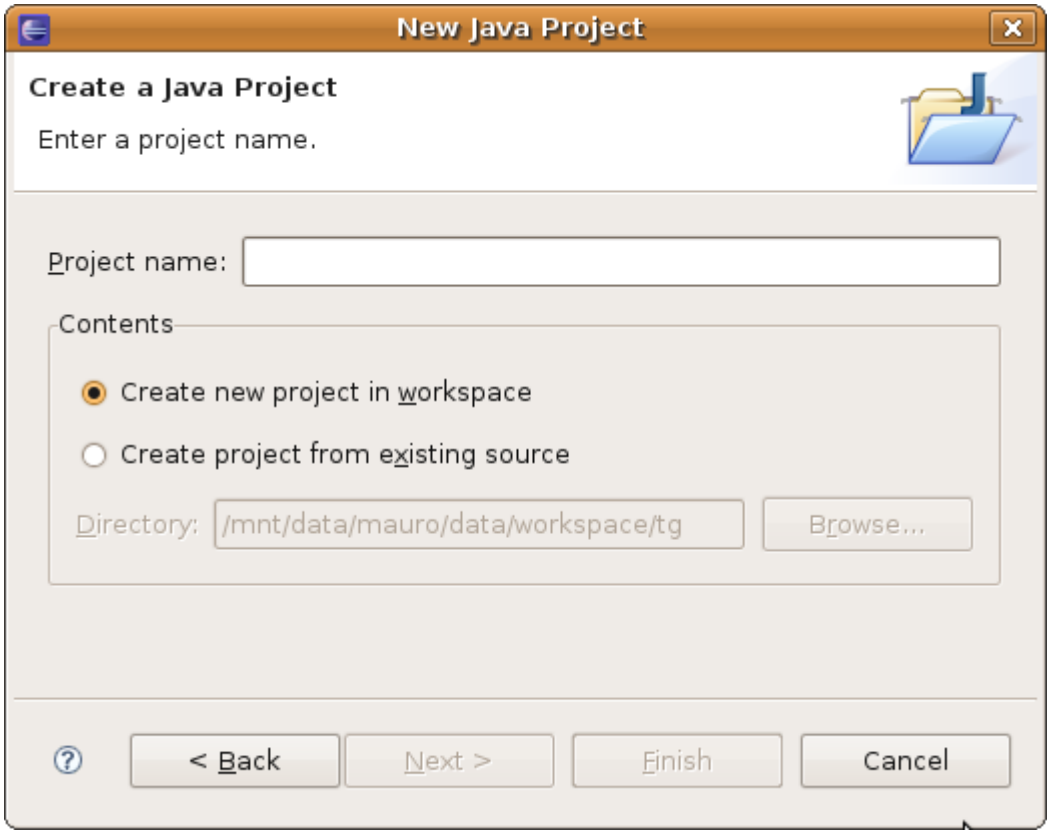

Figura 1: Assistente de criação de projeto Java

- Assistente de criação de arquivo – Um assistente de criação de arquivo é uma ferramenta que cria um novo arquivo no ambiente baseando-se nas informações entradas pelo usuário. A Figura 2 exibe o assistente de criação de classe Java provido pelo JDT.

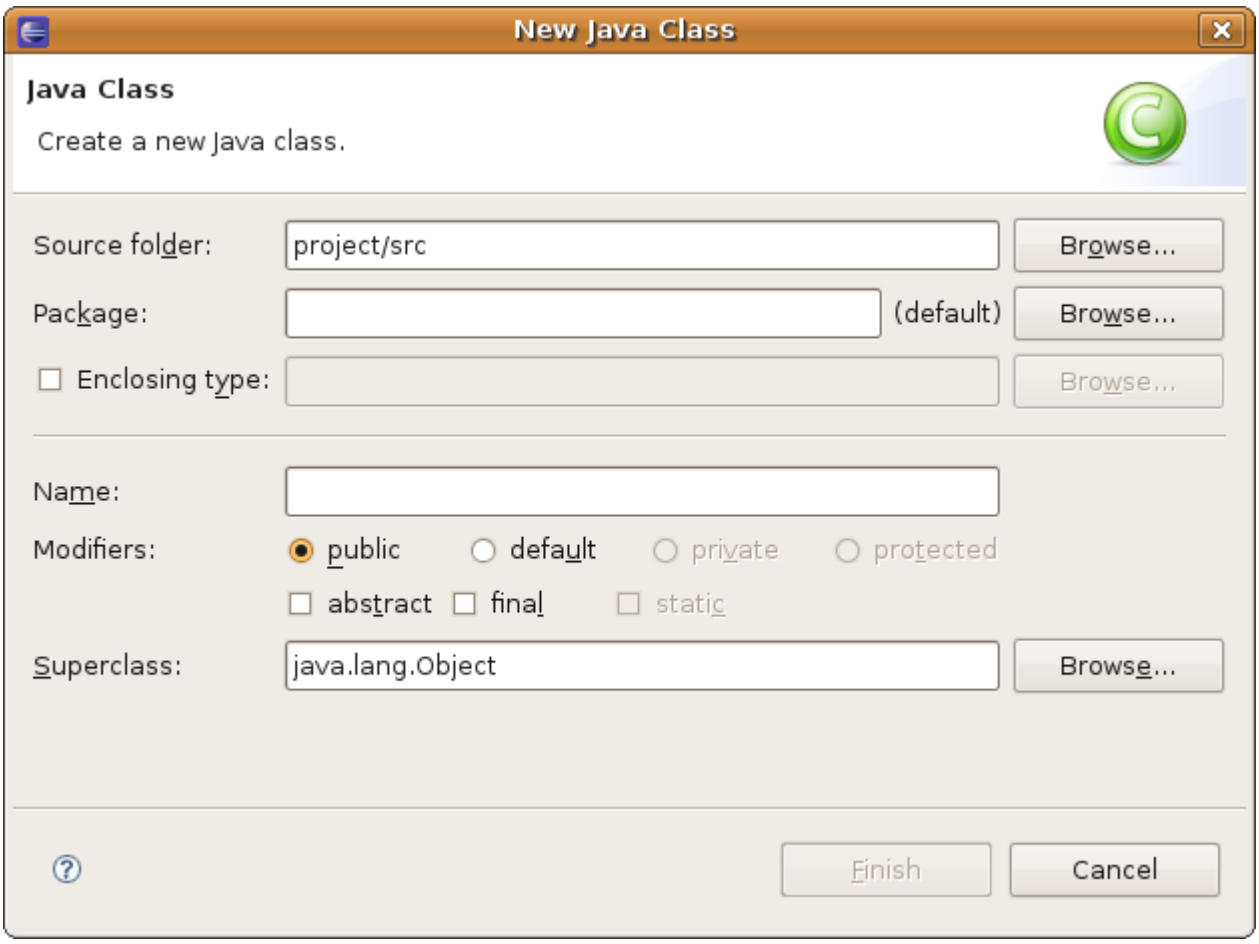

Figura 2: Assistente de criação de classe Java

- Natureza de projeto – Uma natureza de projeto associada a um projeto associa a ele determinadas propriedades e funcionalidades. Projetos Java tem ícone distinto, associado à natureza de projeto Java provida pelo JDT, como pode ser visto na Figura 4 a seguir.

- Perspectiva – Uma perspectiva define a disposição dos elementos visuais do ambiente. Trocando-se de perspectiva, é possível alternar rapidamente a disposição das ferramentas. Isso é muito útil para alternar ente diferentes atividades que requerem diferentes ferramentas. A Figura 3 a seguir destaca a barra de seleção de perspectivas, que fica por padrão no canto superior direito do *workbench*, como pode ser visto mais adiante na Figura 4.

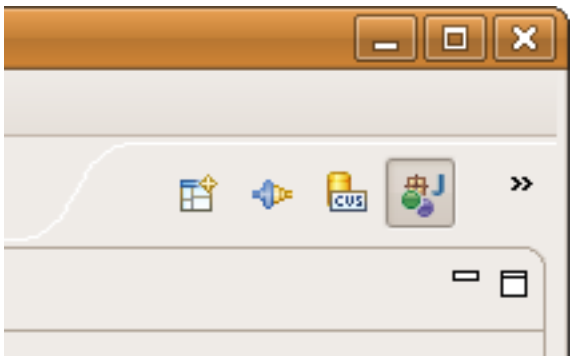

Figura 3: Barra de perspectivas

- Editor de arquivo – Um editor de arquivo permite ao usuário modificar o conteúdo do arquivo aberto no editor. Um editor pode variar muito no grau de complexidade, podendo ser apenas textual ou podendo conter elementos gráficos como rótulos, campos de texto e botões, e podendo conter ferramentas avançadas como marcação de problemas, sugestão de código, auto-formatação etc. A Figura 4 exibe o *workbench* com o editor Java do JDT na posição central.

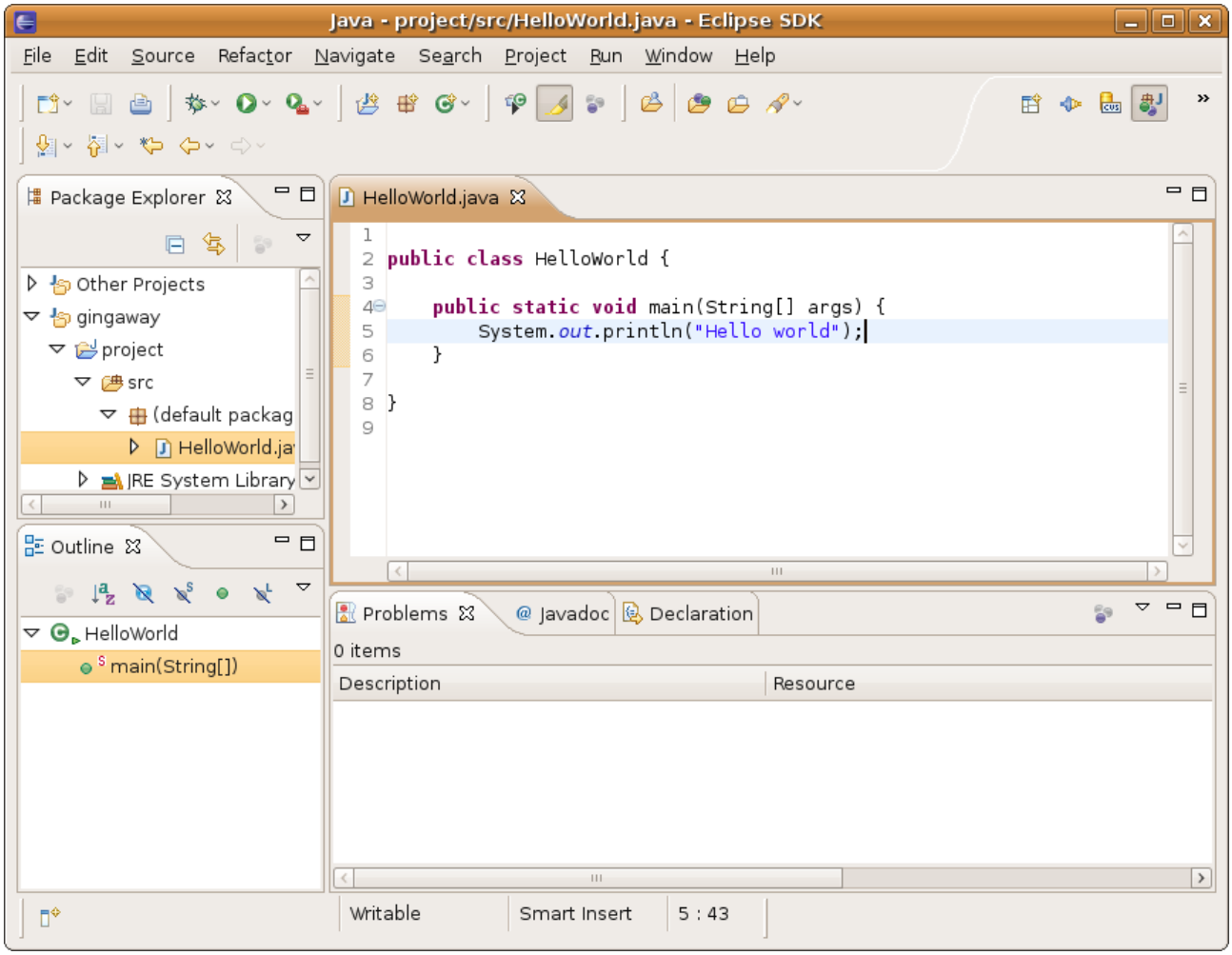

Figura 4: IDE Eclipse para Java

- *Outline* de arquivo – Geralmente um *outline* de arquivo é uma visão simplificada do conteúdo do arquivo aberto no editor, que também permite que o usuário modifique o conteúdo do mesmo. O *outline* de arquivo Java do JDT é exibido no canto inferior esquerdo do *workbench* na Figura 4.

- Configuração de execução – Uma configuração de execução permite que o usuário execute um programa de um determinado tipo, de dentro do ambiente, com determinadas opções. É possível salvar e modificar uma configuração. A Figura 5 mostra uma configuração de execução Java provida pelo JDT no diálogo de configurações de execução.

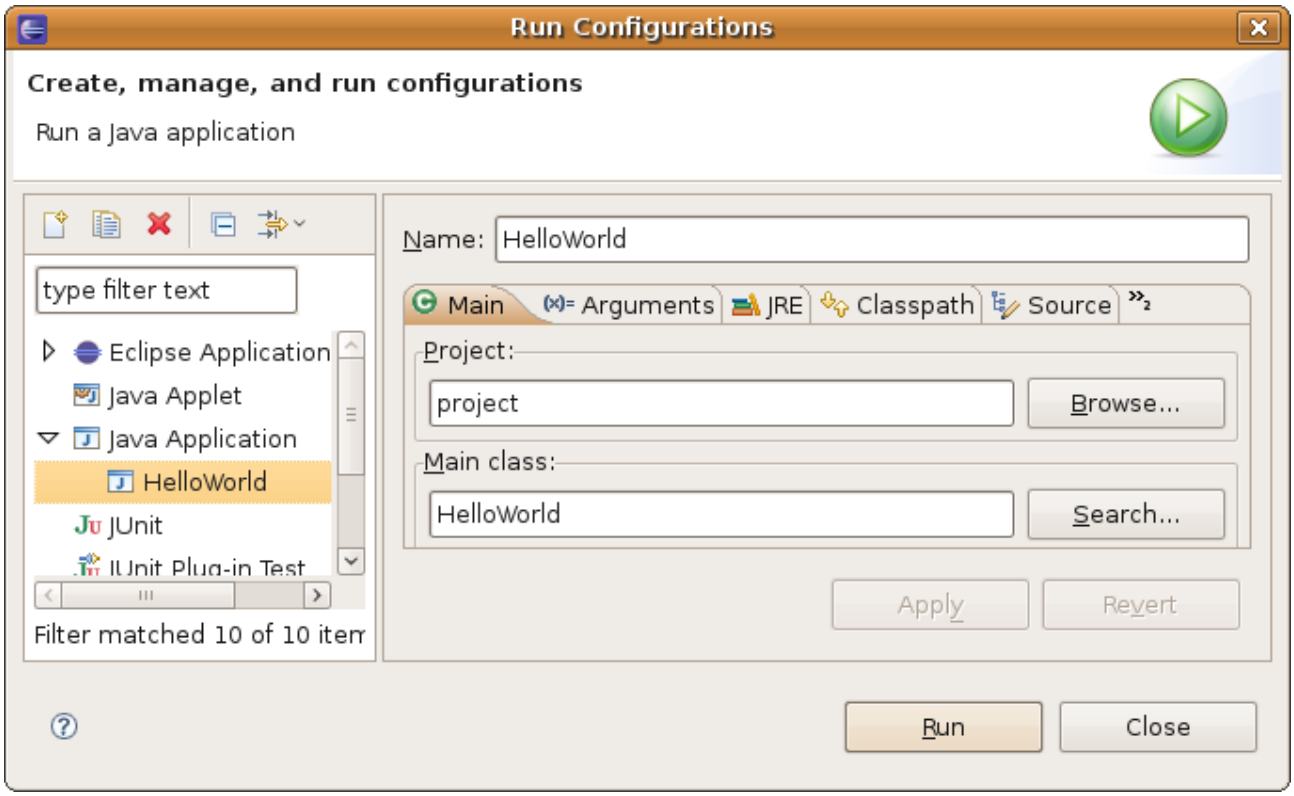

Figura 5: Configuração de execução Java

- Execução de arquivo – Este ponto de extensão permite que o usuário execute um arquivo de dentro do ambiente. Pode (ou não) envolver a criação automática de uma configuração de execução com valores padrão ou baseados em contexto. A execução é disparada pelo comando contido no centro da barra de tarefas de execução destacada na Figura 6.

Figura 6: Barra de tarefas de execução

- Marcadores de problemas – Um marcador de problema sinaliza a ocorrência de um determinado problema, de um determinado tipo, em um recurso contido em algum projeto no ambiente. Essa sinalização é bem integrada ao ambiente, podendo aparecer em diversos elementos da interface gráfica: no local específico do arquivo aberto no editor (por exemplo, na linha de texto do arquivo), no ícone do editor, no ícone do arquivo dentro do projeto, no ícone do projeto, na lista de problemas etc., como mostrado na Figura 7.

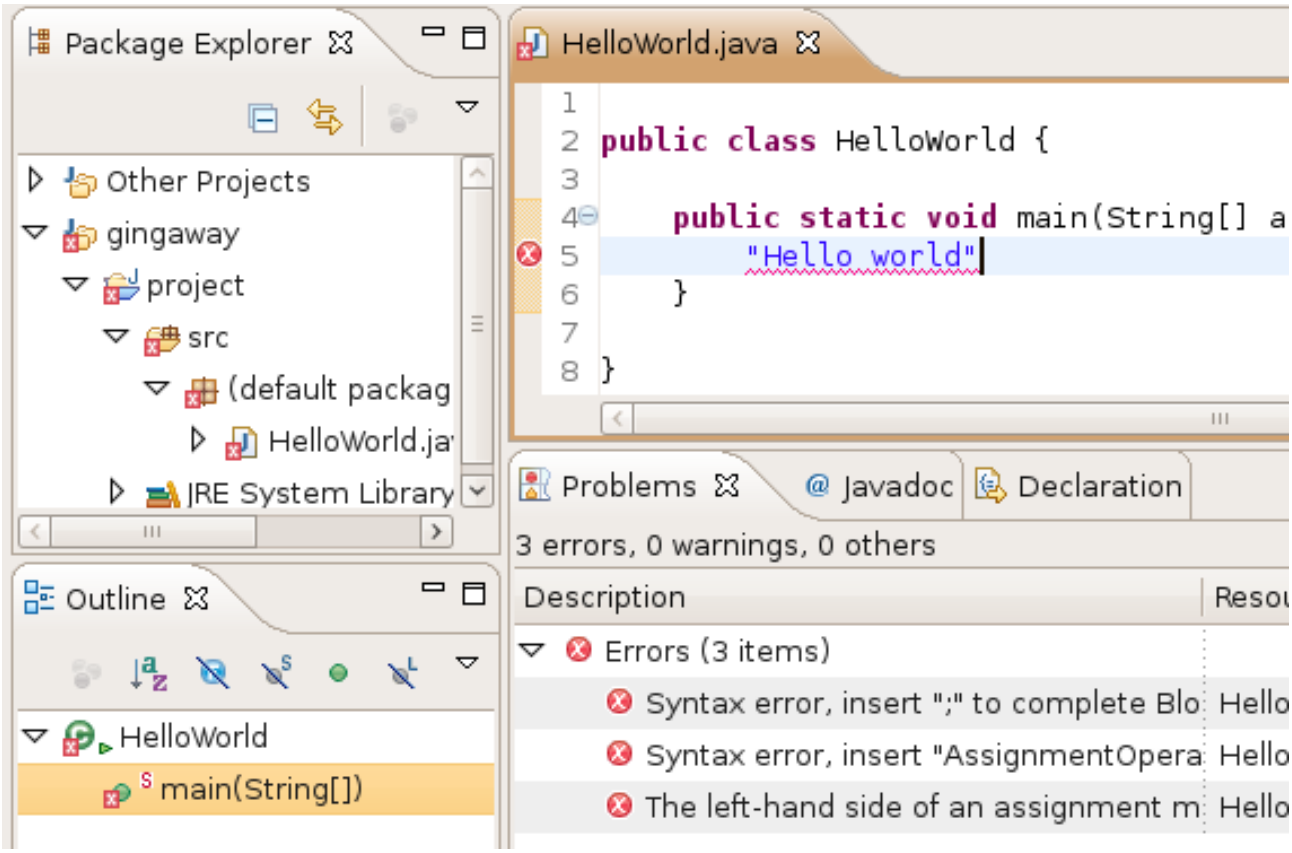

Figura 7: Marcadores de problemas Java

- Barra de tarfas – Uma barra de tarefas fornece atalhos para certos comandos. Por exemplo, a barra de tarefas Java contribuída pelo JDT inclui atalhos para os assistentes de criação de novos artefatos, como projeto, pacote e classe, como mostra a Figura 8.

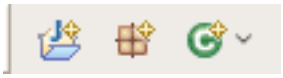

Figura 8: Barra de tarefas Java

- *Working sets* – Um *working set* agrupa projetos de uma determinada natureza, para fins de organização. Dois *working sets* são exibidos na Figura 7: *Other Projects* e *gingaway*.

- Ajuda – A ajuda fornece documentação sobre funcionalidades integrada ao ambiente,

como exibe a Figura 9.

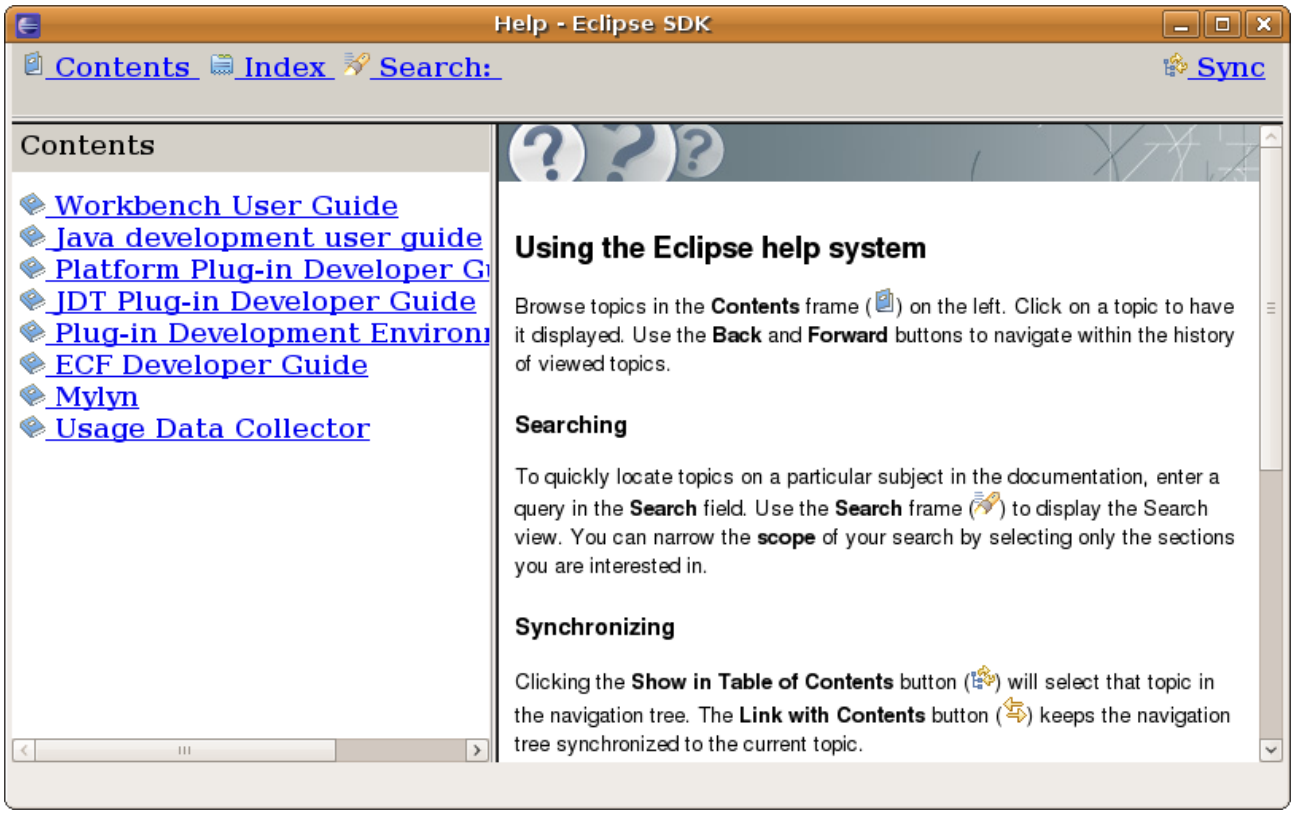

Figura 9: Ajuda integrada

2.3. *Ferramentas de desenvolvimento NCL*

2.3.1. NCL Eclipse

O NCL Eclipse [19] é uma extensão para o Eclipse, constituída de apenas um *plug-in,* que provê ferramentas para desenvolvimento de aplicações Ginga-NCL, mais voltadas a documentos NCL. Ele foi desenvolvido pelo Laboratório de Sistemas Avançados da Web, da Universidade Federal do Maranhão, e possui licenciamento duplo, com licenças GPLv2 e comercial. A versão estudada (a última então disponível) foi a 1.0.0.

Entre as extensões providas pelo NCL Eclipse, incluem-se: assistente de criação de documento NCL, editor de arquivo NCL, *outline* de arquivo NCL, marcadores de problemas e execução integrada de arquivos NCL através do Ginga-NCL Emulator [20]. O editor NCL possui os seguintes recursos avançados: coloração sensível à sintaxe, sugestão de código, marcação de erros e formatação automática.

Apesar de o NCL Eclipse prover extensões para diversos dos pontos de extensão listados na seção *2.2.2. IDE Eclipse*, faltam-lhe ainda recursos importantes como: natureza de projeto,

assistente de criação de projeto, configuração de execução e perspectiva. Além disso, há o acoplamento ao Ginga-NCL Emulator, que é incluso no pacote do *plug-in*. Esse acoplamento é negativo, pois o usuário pode desejar, por exemplo, usar uma ferramenta de execução diferente, e no caso do lançamento de uma nova versão do emulador, o *plug-in* passa a trabalhar com uma versão antiga do mesmo.

2.3.2. Composer

O Composer [21] é um ambiente de desenvolvimento de documentos NCL implementado em Java pelo laboratório Telemídia da PUC-Rio e possui licenciamento duplo, com licenças GPLv2 e comercial. Faz parte de seu propósito oferecer recursos avançados de interface gráfica, com editores visuais de documentos NCL de diferentes perspectivas do código: estrutural, temporal e de layout, além da visão textual. No entanto, o Composer possui vários problemas, citando apenas alguns:

- Problemas com a interface gráfica e usabilidade:
  - Elementos mal dispostos, como pode ser observado no exemplo da Figura 10;
  - Inconsistência com o padrão do ambiente desktop, como pode ser observado comparando padrões das interfaces na Figura 10 e na Figura 12 por exemplo;
- Informações deficitárias sobre erros, como mostrado na Figura 11;
- Suporte restrito à edição de arquivos do tipo NCL; não permite o manuseio do projeto com todos os seus arquivos;
- Baixa performance.

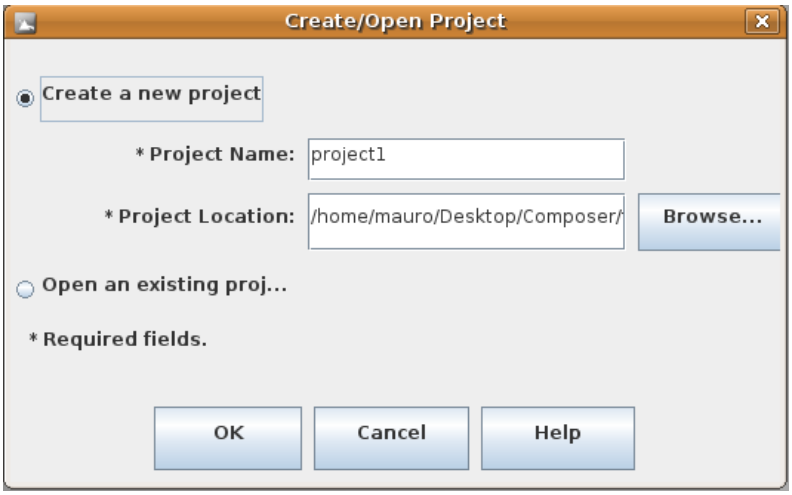

Figura 10: Janela de diálogo do Composer

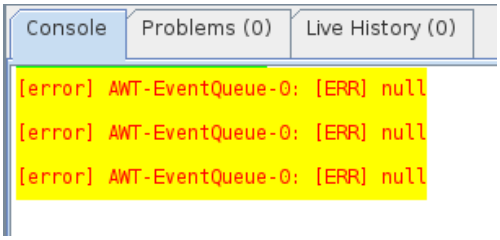

Figura 11: Mensagens de erro no Composer

2.4. Ferramentas de desenvolvimento Lua

2.4.1. Dynamic Languages Toolkit (DLTK)

O DLTK [22] é um projeto do Eclipse que provê ferramentas que estendem a Plataforma Eclipse, facilitando o trabalho de estender a plataforma com ferramentas voltadas ao desenvolvimento com linguagens dinâmicas. O projeto inclui extensões para Ruby e Python, entre outras, desenvolvidas usando o próprio DLTK, que podem servir como exemplo e que também foram estudadas. O DLTK tem licença EPL e a versão estudada (a última então disponível) foi a 0.95.

O Gingaway poderia se beneficiar do DLTK no desenvolvimento das ferramentas relacionadas à linguagem Lua, já que esta é uma linguagem dinâmica. Entretanto constatou-se no estudo que as ferramentas inclusas no DLTK são ainda pouco maduras, com quase nenhuma documentação, o que dificulta em muito o seu uso. E apesar de o *toolkit* facilitar o desenvolvimento, criar ferramentas avançadas requerem muito trabalho: as extensões para Python, por exemplo, somam cerca de 200 classes.

2.4.2. LunarEclipse

O LunarEclipse [23] é uma extensão para o Eclipse baseada no DLTK que provê ferramentas para o desenvolvimento de aplicações Lua, incluindo: assitentes de criação de projeto e de arquivo Lua, editor e outline de arquivo Lua, perspectiva Lua e execução integrada de *scripts* Lua. Esse projeto, que tem liceça de uso EPL, seria uma boa opção de reuso para o Gingaway, no entanto trata-se de um projeto descontinuado, que suporta apenas uma versão antiga do Eclipse (3.3, Europa), mal documentado e houve dificuldade para executá-lo sem a ocorrência de erros durante o estudo. A versão estudada (a última então disponível) foi a 1.2.8.

2.4.3. LuaEclipse

O LuaEclipse [24] é uma extensão para o Eclipse que provê ferramentas para desenvolvimento Lua, possui licença GPL e a versão estudada (a última então disponível) foi a 1.3 beta. Entre as extensões providas pelo LuaEclipse, incluem-se: assistentes de criação de projeto e de arquivo Lua, natureza de projeto Lua, editor de arquivo Lua, configuração de execução, execução integrada de arquivos Lua e marcadores de problemas. O editor Lua possui os seguintes recursos avançados: coloração sensível à sintaxe, marcação de erros e formatação automática. A ferramenta contém ainda duas funcionalidades: *Lua Actions*, que são atalhos para os assistentes de criação, e *Lua Profiler*, ferramenta que expõe detalhes internos de programas Lua em execução.

Como observado, o LuaEclipse inclui extensões para diversos dos pontos de extensão listados na seção *2.2.2. IDE Eclipse*, faltando-lhe apenas *outline* de arquivo Lua e perspectiva dentre os recursos mais importantes.

2.5. Ambiente NCL/Lua com NCL Eclipse e LuaEclipse

Uma opção de ambiente de desenvolvimento Ginga-NCL pode ser obtido com a instalação do NCL Eclipse e do LuaEclipse em uma IDE Eclipse. Fez parte deste estudo a análise deste ambiente, que reúne as ferramentas NCL providas pelo NCL Eclipse e as ferramentas Lua providas pelo LuaEclipse. Dessa forma, este ambiente contempla o desenvolvimento Ginga-NCL com ferramentas para as duas tecnlogias envolvidas, no entanto foram observados alguns problemas, entre eles: falta de funcionalidades importantes, presença de funcionalidades dispensáveis, instalação de mais de uma extensão e falta de integração. Cada um desses problemas é detalhado a seguir.

- Falta de funcionalidades importantes – Este ambiente herda a falta de algumas ferramentas importantes, constatada nos *plug-ins*, como: assistente de criação de projeto Ginga-NCL, natureza de projeto Ginga-NCL, perspectiva Ginga-NCL, *outline* de arquivo Lua, configuração de execução de aplicação Ginga-NCL.

- Presença de funcionalidades dispensáveis – Este ambiente herda das extensões algumas ferramentas indesejadas por não agregarem valor ao desenvolvimento Ginga-NCL, cuja presença tem o efeito negativo de ocupar espaço tanto na interface gráfica (impactando na usabilidade) quanto na memória. Exemplos são Lua Profiler e Lua Actions do LuaEclipse. Também a presença do Ginga-NCL Emulator embarcado no NCL Eclipse pode ser negativa caso o usuário opte por não utilizá-lo.

- Instalação de mais de uma extensão – Outro problema desta solução, inerente ao uso de mais de uma extensão, refere-se ao trabalho duplicado de obter, instalar, configurar e manter separadamente o NCL Eclipse e o LuaEclipse.

- Falta de integração – Por fim, constata-se que este ambiente sofre pela falta de integração entre as ferramentas providas de forma independente pelas duas extensões, sendo isso também decorrência dos problemas citados anteriormente. Este problema impacta na usabilidade do ambiente. Um exemplo disso pode ser observado na Figura 12, que mostra os assistentes de criação providos pelo NCL Eclipse e pelo LuaEclipse, agrupados em categorias separadas; o ideal seria uma única categoria denominada Ginga-NCL, já que este é o tipo de aplicação sendo desenvolvida.

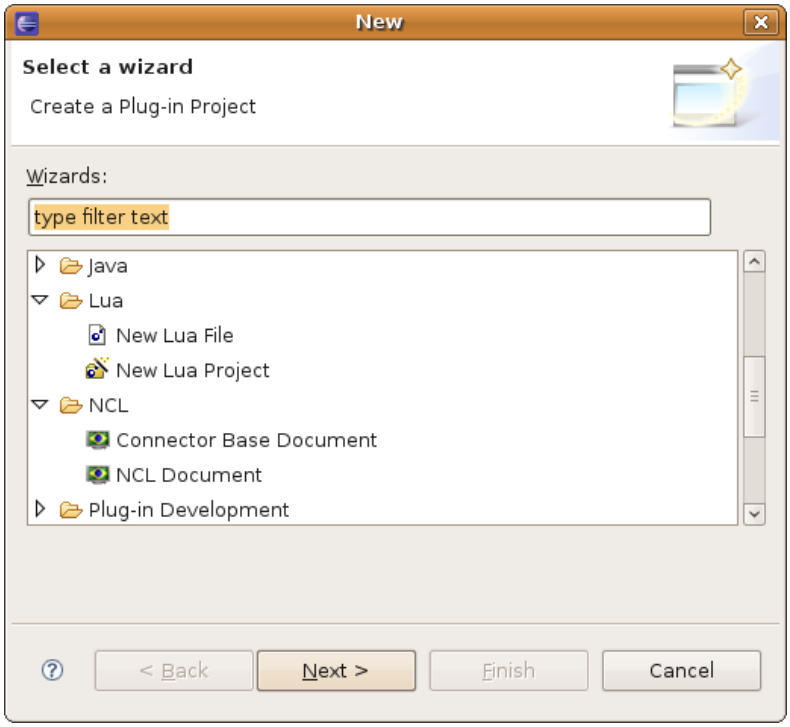

Figura 12: Assistentes de criação do NCL Eclipse e do LuaEclipse

2.6. Ambientes de execução de aplicações Ginga-NCL

Há atualmente dois ambientes de execução de aplicações Ginga-NCL de referência disponíveis: o Ginga-NCL Emulator e o Ginga-NCL Virtual Set-top Box. Assim como o Composer, ambos foram desenvolvidos pelo laboratório Telemídia da PUC-Rio e possuem licenciamento duplo, com licenças GPLv2 e comercial. O intuito dessas ferramentas é permitir a previsão do comportamento das aplicações tal como elas rodarão nos *set-top boxes*. Elas fazem parte, portanto, do conjunto de ferramentas de desenvolvimento Ginga-NCL, já que testar a aplicação em

desenvolvimento é parte fundamental da implementação.

2.6.1. Ginga-NCL Emulator

O Ginga-NCL Emulator [20] é desenvolvido em Java e apresenta problemas de fidelidade na apresentação de documento NCL, pois não oferece suporte a *alfa blending*, transparência, efeitos de transição, entre outros. A versão analisada (a última então disponível) foi a 1.1.1. A interface do programa é exibida na Figura 13.

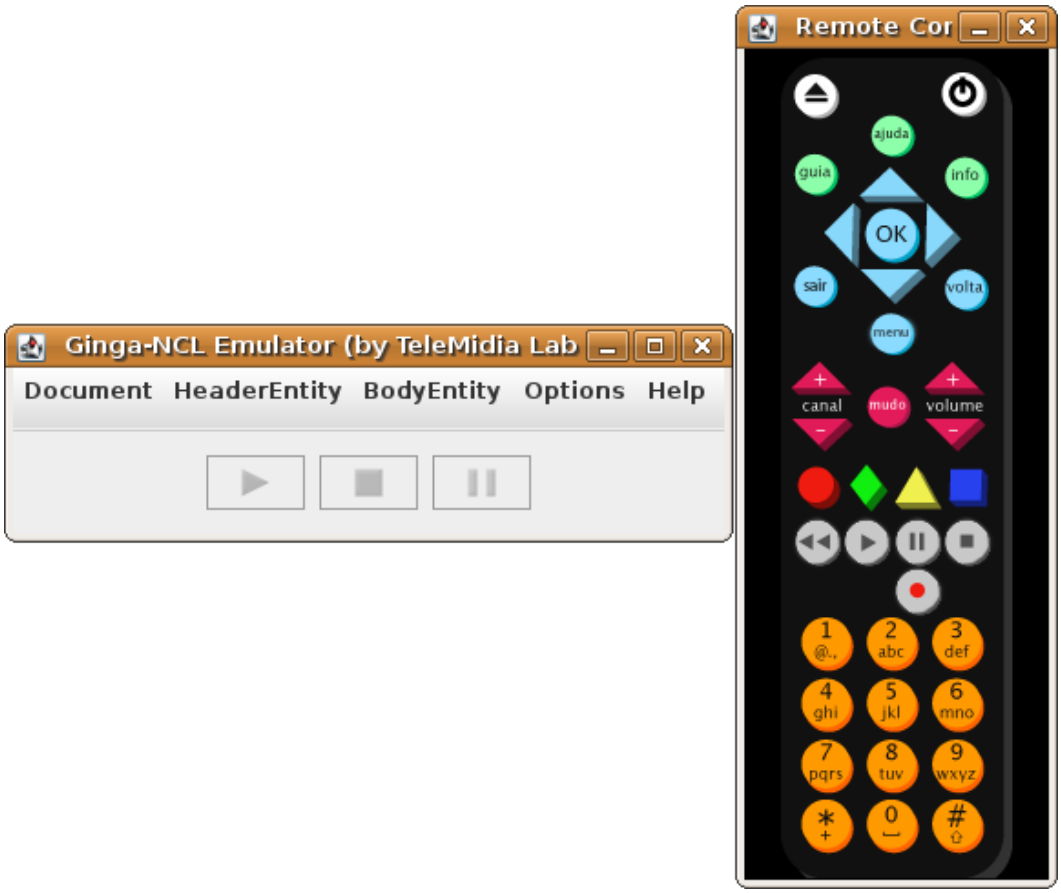

Figura 13: Ginga-NCL Emulator

2.6.2. Ginga-NCL Virtual Set-top Box

O Ginga-NCL Virtual Set-top Box [25] consiste numa máquina virtual contendo sistema operacional Linux e uma implementação do Ginga-NCL implementada em C++, sendo assim um ambiente mais próximo à realidade de *set-top boxes* reais, apresentando uma maior fidelidade de apresentação de documentos NCL em comparação ao Ginga-NCL Emulator. A versão analisada (a última então disponível) foi a 0.9.28. A tela inicial do *set-top box* virtual é exibida na Figura 14.

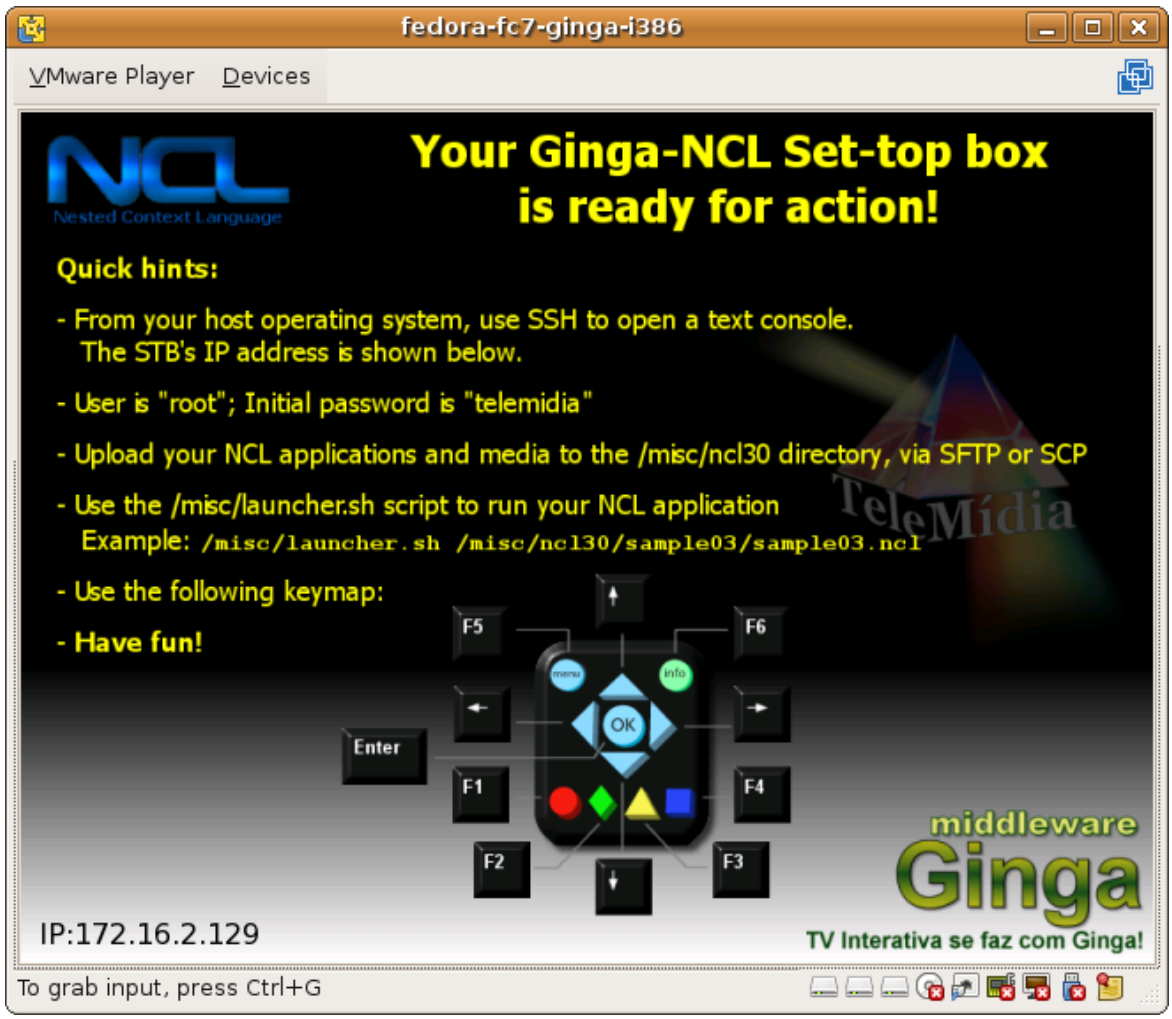

Figura 14: Ginga-NCL Virtual Set-top Box

2.7. Conclusões

É exibido a seguir um quadro comparativo que resume as características apresentadas sobre cada ambiente de desenvolvimento de referência para o Gingaway disponível.

NCL Eclipse	**Pontos positivos:** Editor NCL com coloração sensível à sintaxe, sugestão de código, marcação de erros e formatação automática. Benefícios da Plataforma Eclipse.	
	Pontos negativos: Emulador Ginga-NCL Emulator acoplado.	
	Ferramentas presentes: Assistente de criação de arquivo NCL Editor de arquivo NCL *Outline* de arquivo NCL Marcadores de problemas Execução integrada de arquivos NCL	**Ferramentas ausentes:** Natureza de projeto Assistente de criação de projeto Configuração de execução Perspectiva
Composer	**Pontos positivos:** Editores visuais de documentos NCL de diferentes perspectivas do código: estrutural, temporal e de layout, além da visão textual.	
	Pontos negativos: Problemas com a interface gráfica e usabilidade, inconsistência gráfica com o ambiente desktop , informações deficitárias sobre erros, suporte restrito a arquivos do tipo NCL e baixa performance.	
	Ferramentas presentes: Assistente de criação de projeto Editor de arquivo NCL Exucução integrada de arquivo NCL	**Ferramentas ausentes:** Ferramentas de suporte a Lua
LunarEclipse	**Pontos positivos:** Recursos do DLTK e da Plataforma Eclipse.	
	Pontos negativos: Projeto descontinuado, suporte a versão antiga do Eclipse (3.3), mal documentado.	
	Ferramentas presentes: Assitente de criação de projeto Assitente de criação de arquivo Lua Editor de arquivo Lua *Outline* de arquivo Lua Perspectiva Lua Execução integrada de *scripts* Lua	**Ferramentas ausentes:** Ferramentas de suporte a NCL
LuaEclipse	**Pontos positivos:** Editor Lua com coloração sensível à sintaxe, marcação de erros e formatação automática. Lua Profiler, ferramenta que expõe detalhes internos de programas Lua em execução. Benefícios da Plataforma Eclipse.	
	Pontos negativos: Ferramenta pode ainda amadurecer bastante, com a adição ferramentas ainda ausentes, como por exemplo sugestão de código e navegação avançada no editor. Há também *bugs*, como na interface de configuração de execução.	
	Ferramentas presentes: Assistente de criação de projeto Assistente de criação de arquivo Lua Natureza de projeto Lua Editor de arquivo Lua Configuração de execução Execução integrada de arquivos Lua Marcadores de problemas	**Ferramentas ausentes:** *Outline* de arquivo Lua Perspectiva de desenvolvimento Lua

<table>
<tr><td rowspan="3">NCL Eclipse/
Lua Eclipse</td><td colspan="2">Pontos positivos: Reunião de ferramentas NCL e Lua no mesmo ambiente.
Benefícios da Plataforma Eclipse.</td></tr>
<tr><td colspan="2">Pontos negativos: Presença de ferramentas indesejáveis, trabalho duplicado com a instalação de duas extensões independentes e falta de integração entre as ferramentas disponíveis.</td></tr>
<tr><td>Ferramentas presentes:
Natureza de projeto Lua
Assistente de criação de projeto Lua
Assistente de criação de arquivo NCL
Assistente de criação de arquivo Lua
Editor de arquivo NCL
Editor de arquivo Lua
Outline de arquivo NCL
Execução integrada de arquivos NCL
Execução integrada de arquivos Lua
Marcadores de problemas</td><td>Ferramentas ausentes:
Natureza de projeto Ginga-NCL
Assistente de criação de projeto Ginga-NCL
Perspectiva Ginga-NCL
Outline de arquivo Lua
Configuração de execução de aplicações Ginga-NCL</td></tr>
</table>

Quadro 1: Comparação dos ambientes disponíveis

3. Requisitos

Esta seção trata da análise de requisitos do Gingaway realizada neste trabalho.

O propósito primordial do Gingaway é prover aos desenvolvedores de aplicações Ginga-NCL um ambiente de programação de qualidade, que possibilite alta eficiência.

De modo a atender da melhor maneira às necessidades dos desenvolvedores, foram analisadas as dificuldades encontradas no uso das ferramentas disponíveis. Esse trabalho foi facilitado devido ao próprio autor ter participado de um projeto de desenvolvimento de uma aplicação interativa Ginga-NCL. Nesse projeto foram usadas duas ferramentas principais: um editor textual, para editar arquivos NCL e Lua, e um emulador, para testar a aplicação sob desenvolvimento.

O editor textual usado foi o ConTEXT [26], um avançado editor de texto em geral, com funções como: identação, macros, múltiplas abas, busca avançada, entre outras. Apesar disso, sua única função específica de linguagem é a de coloração do texto sensível à sintaxe, ligada ao tipo do arquivo sendo trabalhado. A falta de mais funções específicas, como marcação de problemas de sintaxe, sugestão de código, auto-formatação e refabricação (*refactoring*) de código, que tornam o trabalho mais eficiente, é em geral muito sentida pelos desenvolvedores.

O emulador usado para simular o ambiente do *set-top box* foi o Ginga-NCL Virtual Set-top Box, já abordado anteriormente. Com ele, foi necessário usar dois outros programas: o WinSCP [27], para incluir a aplicação em seu sistema de arquivos, e o PuTTY [28], para obter uma interface onde é executado o comando que inicia a aplicação. Como se vê, foi necessário o uso de três programas para o único propósito de testar a aplicação Ginga-NCL sendo desenvolvida.

O excesso de ferramentas independentes, com pouca integração entre elas, gera perda de tempo durante o processo de desenvolvimento, já que é preciso obter, instalar, configurar, abrir, fechar e principalmente manusear cada uma. Esse problema motivou, inclusive, o time a não usar uma ferramenta especializada na edição de código Lua, pois seria uma ferramenta a mais.

Constatam-se assim, nesse cenário, alguns problemas que levam a uma baixa produtividade: falta de especialidade, quantidade e falta de integração entre as ferramentas. Logo, fica clara a necessidade de um único ambiente que integre um amplo ferramental, com funções avançadas, especializadas nas tecnologias-alvo. E é aí que o Gingaway se insere, baseando-se em conceitos encontrados nas principais IDE.

3.1. Requisitos específicos

Segue-se a lista de requisitos funcionais e não-funcionais elicitados para o Gingaway. Eles se baseiam em conceitos bem estabelecidos por IDE maduras, sobretudo o Eclipse.

3.1.1. Requisitos funcionais

[RF01] Assistente de criação de projeto Ginga-NCL

O Gingaway deve prover um assistente para a criação de projetos Ginga-NCL.

[RF02] Assistentes de criação de arquivos NCL e Lua

O Gingaway deve prover um assistente para a criação de novos arquivos de código-fonte, tanto NCL como Lua.

[RF03] Natureza de projeto Ginga-NCL

O Gingaway deve prover aos projetos Ginga-NCL natureza própria, que lhes confiram suas características.

[RF04] Perspectiva Ginga-NCL

O Gingaway deve prover uma perspectiva própria para o desenvolvimento de aplicações Ginga-NCL.

[RF05] Editores de arquivos NCL e Lua

O Gingaway deve prover editores de código-fonte especializados, tanto para a edição de documentos NCL como de *scripts* Lua. Tais editores devem ter as funcionalidades avançadas de: coloração de texto e marcação de problemas conforme a sintaxe, sugestão de código, auto-formatação e refabricação (*refactoring*) automáticas do código e navegação avançada pelo código.

[RF06] *Outline* de arquivos NCL e Lua

O Gingaway deve prover o recurso de *outline* de arquivos NCL e Lua sendo editados.

[RF07] Configuração de execução de aplicações Ginga-NCL

O Gingaway deve prover uma interface para configuração de execução de aplicações Ginga-NCL.

[RF08] Execução de arquivos NCL e Lua

O Gingaway deve prover um meio direto de execução de arquivos NCL e Lua.

[RF09] Marcadores de problemas

O Gingaway deve prover marcadores de problemas de sintaxe de código-fonte tanto NCL como Lua.

[RF10] Barra de tarefas

O Gingaway deve prover uma barra de tarefas com atalhos para os assistentes de criação relacionados ao desenvolvimento Ginga-NCL.

[RF11] *Working sets*

O Gingaway deve prover tipo próprio de *working set*.

[RF12] Ajuda

O Gingaway deve prover documentação de ajuda integrada ao ambiente.

3.1.2. Requisitos não-funcionais

[RNF01] Integração entre as ferramentas dentro do ambiente

O conjunto de ferramentas provido pelo Gingaway deve primar pela integração, entre si e com o restante do ambiente provido pela plataforma e demais *plug-ins*, de modo que o acúmulo de tal número de funções não resulte em dificuldades aos usuários.

[RNF02] Seguir padrões da Plataforma Eclipse

O Gingaway deve seguir todos os padrões estabelecidos pela Plataforma Eclipse, sejam de aspecto visual, funcional, arquitetural etc. Essa exigência faz-se essencial para a manutenção da homogeneidade da IDE, ainda que com uma diversidade de *plug-ins* instalados sobre a plataforma.

4. Implementação

Esta seção apresenta o trabalho desenvolvido na implementação do Gingaway. Com a implementação, tentou-se construir a IDE que atende aos requisitos analisados na seção anterior.

O Gingaway consiste numa extensão para a Plataforma Eclipse. Assim ele se beneficia dos recursos fornecidos por essa avançada plataforma, detendo-se a adicionar as ferramentas específicas ao desenvolvimento de aplicações Ginga-NCL. Além da Plataforma Eclipse, o Gingaway usa como base duas extensões para a mesma plataforma já existentes: o NCL Eclipse e o LuaEclipse, apresentadas na seção *2. Trabalhos relacionados*.

4.1. Reuso de ferramentas

O Gingaway reusa elementos do NCL Eclipse e do LuaEclipse, se beneficiando do trabalho já realizado nesses dois projetos. Em vez de usá-los integralmente, o Gingaway aproveita apenas as funcionalidades que melhor contribuem para a realização dos requisitos tal como foram especificados. Assim, há elementos removidos ou modificados em algum ponto. Essas alterações não envolvem, contudo, modificações em código-fonte dos *plug-ins* existentes. As alterações se detiveram aos arquivos *plug-in manifest* de alguns dos *plug-ins*. Ou seja, o código-fonte aproveitado é aproveitado na versão original. Não fez parte do escopo deste trabalho melhorar o código dos *plug-ins* existentes. Houve adição de funcionalidades apenas no caso dos *plug-ins* desenvolvidos especificamente para o Gingaway.

Seguem-se os detalhes do trabalho realizado com esses projetos.

4.1.1. NCL Eclipse

Apesar de o NCL Eclipse ter licença livre, este trabalho não conseguiu ter acesso ao seu código-fonte; isso contudo não constituiu problema, já que não fez parte do escopo alterar o código-fonte dos projetos reusados. Foi criado, então, um novo *plug-in* dentro do Gingaway, o *plug-in* gingaway.ncleclipse, e a ele foram adicionados, como biblioteca, os binários do NCL Eclipse, assim como suas bibliotecas. O Ginga-NCL Emulator, incluso na distribuição do NCL Eclipse, foi excluso com o objetivo de desacoplar o emulador da IDE. O arquivo *plug-in manifest* sofreu as alterações apresentadas no Quadro 2.

Projeto	Modificação	Descrição
ncl_eclipse	Substituição	Ícone do editor NCL
		Ícone do assistente de criação de documento NCL
		Ícone do assistente de criação de documento de base de conectores
		Categoria do assistente de criação de documento NCL passa a ser a categoria Ginga-NCL definida pelo Gingaway
		Categoria do assistente de criação de documento de base de conectores passa a ser a categoria Ginga-NCL definida pelo Gingaway
	Remoção	Categoria de assistentes NCL

Quadro 2: Modificações no NCL Eclipse

O Quadro 3 detalha as extensões do *plug-in* gingaway.ncleclipse após as modificações feitas ao arquivo *plug-in manifest*.

***Plug-in*:** gingaway.ncleclipse Provê ferramentas NCL, baseadas no NCL Eclipse.	
Extensão	**Descrição**
org.eclipse.core.resources.markers	Provê os marcadores de problemas de código NCL.
org.eclipse.debug.ui.launchShortcuts	Provê atalho para execução de documentos NCL.
org.eclipse.ui.editors	Provê o editor de documentos NCL.
org.eclipse.ui.bindings	Provê atalho para o comando de formatação de código NCL, no editor.
org.eclipse.ui.newWizards	Provê assistentes de criação de documentos NCL e de base de conectores.
org.eclipse.ui.commands	Provê o comando de formatação de código NCL, no editor.

Quadro 3: *Plug-in* gingaway.ncleclipse

4.1.2. LuaEclipse

O código-fonte do LuaEclipse foi utilizado diretamente, mas não foi modificado, apenas os arquivos *plug-in manifest* dos *plug-ins* org.keplerproject.ldt.launch e org.keplerproject.ldt.ui, como mostrado no Quadro 4 (projetos identificados por “launch” e “ui”, respectivamente).

O Gingaway aproveitou apenas um subconjunto do total de *plug-ins* do LuaEclipse. O

Quadro 5 detalha os *plug-ins* utilizados, após as modificações.

Projeto	**Modificação**	**Descrição**
launch	Substituição	Nome da configuração de execução passa a ser “Lua Script” (original: “Lua Standalone Application”)
		Perspectiva onde é habilitado o atalho de execução de arquivos Lua passa a ser a perspectiva Ginga-NCL (original: Java, Java Browsing, Java Hierarchy, Debug)
	Remoção	Atalho de teclado para execução de script Lua
ui	Substituição	Ícone do assistente de criação de arquivo de código-fonte Lua
		Nome do assistente de criação de arquivo Lua passa a ser “Lua Script” (original: “New Lua File”)
	Remoção	Assistente de criação de projeto Lua
		Categoria de assistentes Lua
		Atalhos para os assistentes de criação de projeto e arquivo Lua presentes no menu *File/New* no menu de contexto da IDE

Quadro 4: Modificações no LuaEclipse

Extensão	**Descrição**
***Plug-in*:** org.kepler.ldt.launcher Provê as ferramentas para a execução de scripts Lua.	
org.eclipse.debug.core. launchConfigurationTypes	Provê o tipo de configuração de execução de *scripts* Lua.
org.eclipse.debug.core. sourceLocators	Provê um *source locator* utilizado pela configuração de execução de *scripts* Lua.
org.eclipse.debug.ui. launchConfigurationTabGroups	Provê a interface gráfica para configuração de execução.
org.eclipse.debugui. launchConfigurationTypeImages	Provê o ícone do tipo de configuração de execução.
org.eclipse.debug.ui.launchShortcuts	Provê atalho para execução de documentos NCL.
org.eclipse.ui.preferencePages	Provê uma página de preferências que permite ao usuário indicar o interpretador Lua a ser utilizado pela IDE.
***Plug-in*:** org.keplerproject.ldt.core Núcleo do LuaEclipse.	
org.eclipse.core.resources.builders	Provê um *builder* de projetos Lua.
org.eclipse.core.resources.natures	Provê a natureza de projeto Lua, que adiciona funcionalidades Lua aos projetos.
***Plug-in*:** org.keplerproject.ldt.linux Provê a biblioteca Lua específica para a plataforma Linux. Não provê extensões.	
***Plug-in*:** org.keplerproject.ldt.ui Provê a interface gráfica das ferramentas básicas.	
org.eclipse.ui.editors	Provê um editor de arquivos de *script* Lua.
org.eclipse.ui.newWizards	Provê um assistente de criação de arquivos de *script* Lua.
org.eclipse.ui.preferencePages	Provê uma página de preferências para configuração de opções.
***Plug-in*:** org.keplerproject.ldt.ui.baseExts Provê tratamento de código Lua.	
org.eclipse.ldt.ui. ScannerRulesExtension	Provê um *scanner* de código Lua, útil para a marcação de erros.
org.eclipse.ldt.ui. SourceConfigurationExtension	Provê o recurso de sugestão de código ao editor Lua.
***Plug-in*:** org.keplerproject.ldt.win32 Provê a biblioteca Lua específica para a plataforma Windows. Não provê extensões.	
***Plug-in*:** org.keplerproject.luadoc Provê ferramentas de documentação de código Lua. Não provê extensões.	
***Plug-in*:** org.keplerproject.lualogging Provê ferramentas de log de código Lua. Não provê extensões.	

Quadro 5: *Plug-ins* do LuaEclipse

4.2. *Criação de novos* plug-ins

Além dos *plug-ins* relacionados aos projetos reusados, o Gingaway provê sete novos *plug-ins*, sendo três deles relacionados a funções básicas e os demais relacionados à execução integrada de aplicações Ginga-NCL, como explicado a seguir.

4.2.1. *Plug-ins* centrais

Os *plug-ins* centrais do Gingaway fornecem o núcleo de suas funcionalidades. Eles são detalhados no Quadro 6.

Extensão	**Descrição**
***Plug-in*:** gingaway.core Núcleo do Gingaway.	
org.eclipse.core.resources.natures	Provê a natureza de projeto Ginga-NCL, que identifica os projetos, adicionando funcionalidades específicas.
***Plug-in*:** gingaway.help Provê ajuda integrada.	
org.eclipse.help.toc	Provê o conteúdo da ajuda integrada.
***Plug-in*:** gingaway.ui Provê a interface gráfica das ferramentas básicas do Gingaway.	
org.eclipse.ui.newWizards	Provê o assistente de criação de projeto Ginga-NCL, além da categoria de assistentes Ginga-NCL, a qual contém todos os assistentes do Gingaway: novo projeto, novo documento NCL, novo documento de base de conectores, novo *script* Lua.
org.eclipse.ui.perspectives	Provê a perspectiva Ginga-NCL.
org.eclipse.ui.ide.projectNatureImages	Provê o ícone para a natureza de projeto Ginga-NCL, definida no *plug-in* gingaway.core. Esse ícone identifica visualmente os projetos Ginga-NCL no ambiente.
org.eclipse.ui.menus	Provê a barra de tarefas do Gingaway, visível na perspectiva Ginga-NCL, que contém atalhos para todos os assistentes do Gingaway: novo projeto, novo documento NCL, novo documento de base de conectores, novo script Lua.

Quadro 6: *Plug-ins* centrais do Gingaway

4.2.2. *Plug-ins* de configuração de execução

Para prover a execução integrada de aplicações Ginga-NCL, o Gingaway inclui *plug-ins* específicos para o Ginga-NCL Emulator e o Ginga-NCL Virtual Set-top Box. Esses *plug-ins*

permitem a utilização dessas ferramentas a partir da IDE, mas sem incluí-las; o usuário deve instalá-las separadamente em seu sistema. Dessa maneira as ferramentas de execução ficam desacopladas do Gingaway, facilitando a modificação ou remoção das mesmas. Em contraste, no NCL Eclipse, que consiste de apenas um *plug-in*, o ferramental de execução é acoplado, sendo o Ginga-NCL Emulator incluso na distribuição.

O Quadro 7 e o Quadro 8 mostram os *plug-ins* relacionados.

Extensão	**Descrição**
***Plug-in*:** gingaway.emulator.core Núcleo das ferramentas de execução de aplicações NCL.	
org.eclipse.debug.core. launchConfigurationTypes	Provê a configuração de execução.
***Plug-in*:** gingaway.emulator.ui Provê a interface gráfica das ferramentas.	
org.eclipse.debug.ui. launchConfigurationTabGroups	Provê a interface gráfica para a configuração de execução.
org.eclipse.debug.ui. launchConfigurationTypeImages	Provê o ícone do tipo de configuração de execução.

Quadro 7: *Plug-ins* para execução com o Ginga-NCL Virtual Set-top Box

Extensão	**Descrição**
***Plug-in*:** gingaway.vstb.core Núcleo das ferramentas de execução de aplicações NCL.	
org.eclipse.debug.core. launchConfigurationTypes	Provê a configuração de execução.
***Plug-in*:** gingaway.vstb.ui Provê a interface gráfica das ferramentas.	
org.eclipse.debug.ui. launchConfigurationTabGroups	Provê a interface gráfica para a configuração de execução.
org.eclipse.debug.ui. launchConfigurationTypeImages	Provê o ícone do tipo de configuração de execução.

Quadro 8: *Plug-ins* para execução com o Ginga-NCL Emulator

4.3. Features

Os *plug-ins* do Gingaway explicados acima foram agrupados em seis *features*, resumidas no Quadro 9.

Feature	**Descrição**	**Conteúdo**
***Features* principais:**		
gingaway.linux.feature	Inclui o Gingaway completo para a plataforma Linux.	gingaway.emulator.feature gingaway.luaeclipse.linux.feature gingaway.vstb.feature gingaway.core gingaway.help gingaway.ncleclipse gingaway.ui
gingaway.windows.feature	Inclui o Gingaway completo para a plataforma Windows.	gingaway.emulator.feature gingaway.luaeclipse.windows.feature gingaway.vstb.feature gingaway.core gingaway.help gingaway.ncleclipse gingaway.ui
***Features* de adaptação do LuaEclipse:**		
gingaway.luaeclipse. linux.feature	Inclui a porção do LuaEclipse utilizada, versão para a plataforma Linux 32 bits.	org.keplerproject.ldt.core org.keplerproject.ldt.launcher org.keplerproject.ldt.linux org.keplerproject.ldt.luadoc org.keplerproject.ldt.lualogging org.keplerproject.ldt.ui org.keplerproject.ldt.ui.baseExts
gingaway.luaeclipse. windows.feature	Inclui a porção do LuaEclipse utilizada, versão para a plataforma Windows 32 bits.	org.keplerproject.ldt.core org.keplerproject.ldt.launcher org.keplerproject.ldt.luadoc org.keplerproject.ldt.lualogging org.keplerproject.ldt.ui org.keplerproject.ldt.ui.baseExts org.keplerproject.ldt.win32
***Features* de execução de aplicações Ginga-NCL:**		
gingaway.emulator.feature	Provê execução de aplicações Ginga-NCL através do Ginga-NCL Emulator.	gingaway.emulator.core gingaway.emulator.ui
gingaway.vstb.feature	Provê execução de aplicações Ginga-NCL através do Ginga-NCL Virtual Set-top Box.	gingaway.vstb.core gingaway.vstb.ui

Quadro 9*: Features do Gingaway*

A *feature* principal do Gingaway é a *feature* gingaway.linux.feature ou a gingaway.windows.feature, dependendo da plataforma. Essas *features* apontam para todo o conjunto

de *plug-ins*, tanto diretamente, no caso dos *plug-ins* centrais do Gingaway, como indiretamente, nos casos em que aponta para outras *features*, que por sua vez apontam para os respectivos *plug-ins*. As *features* gingaway.luaeclipse.linux.feature e gingaway.luaeclipse.windows.feature foram criadas para agrupar os *plug-ins* do LuaEclipse. Por fim, o Gingaway inclui as *features* gingaway.emulator.feature e gingaway.vstb.feature que provem a execução integrada de aplicações Ginga-NCL.

4.4. Site de atualização

Por fim, o Gingaway inclui o projeto gingaway.site, gerado pelo PDE e usado para criar o site de atualização (*update site*), que permite ao usuário instalar ou atualizar a instalação do Gingaway de forma integrada à IDE Eclipse. O projeto relaciona as *features* disponíveis para distribuição (abordadas na seção anterior), cria o *build,* empacotando todos os arquivos em pacotes JAR [29] e os disponibiliza para *download* através de uma URL indicada.

5. Exemplo de utilização

Esta seção apresenta um exemplo de utilização do Gingaway numa IDE Eclipse, ilustrando suas funcionalidades. Neste exemplo é criado um projeto Ginga-NCL com um documento NCL e um arquivo de *script* Lua. A aplicação é executada com o Ginga-NCL Virtual Set-top Box. O Remote System Explorer [30] é usado para comunicação com essa máquina virtual.

- Assistente de criação de projeto Ginga-NCL – O projeto, chamado de "test", é criado na IDE com o assistente.

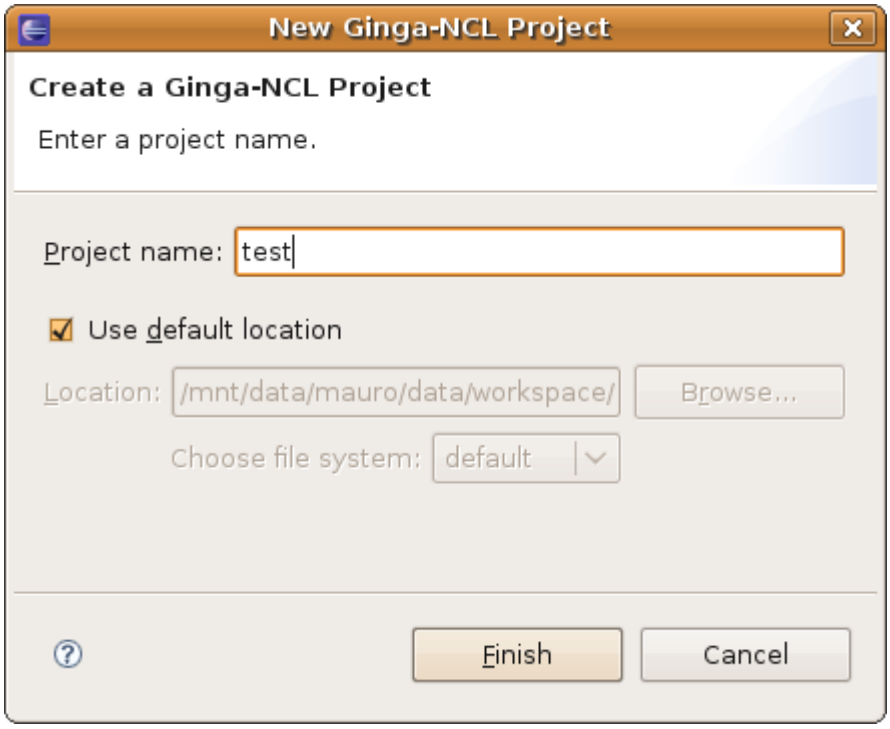

Figura 15: Assistente de criação de projeto Ginga-NCL

- Assistente de criação de documento NCL – O documento NCL é criado e adicionado ao projeto através do assistente.

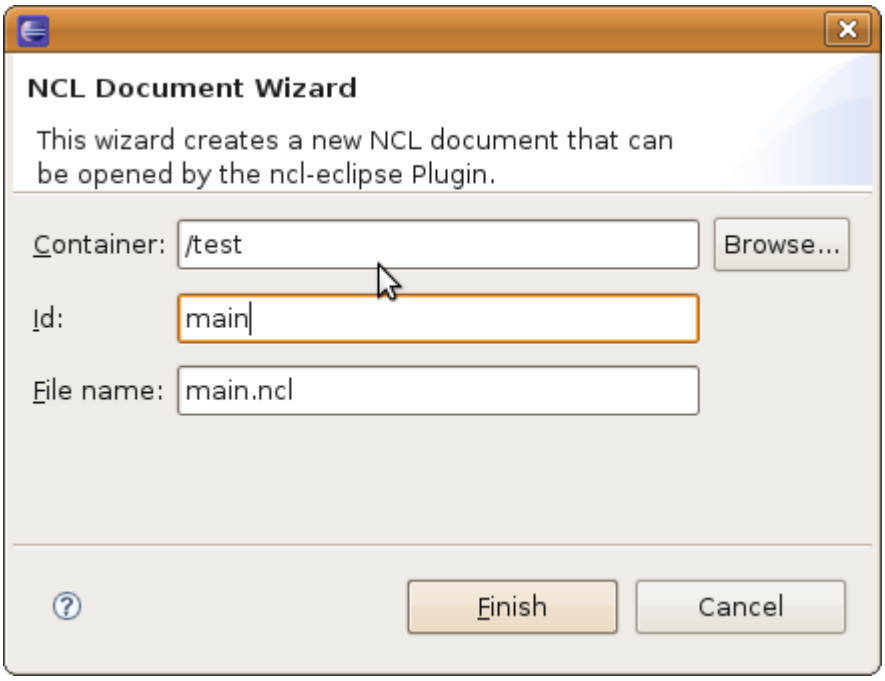

Figura 16: Assistente de criação de documento NCL

- Assistente de criação de arquivo Lua – O arquivo Lua é criado e adicionado ao projeto através do assistente.

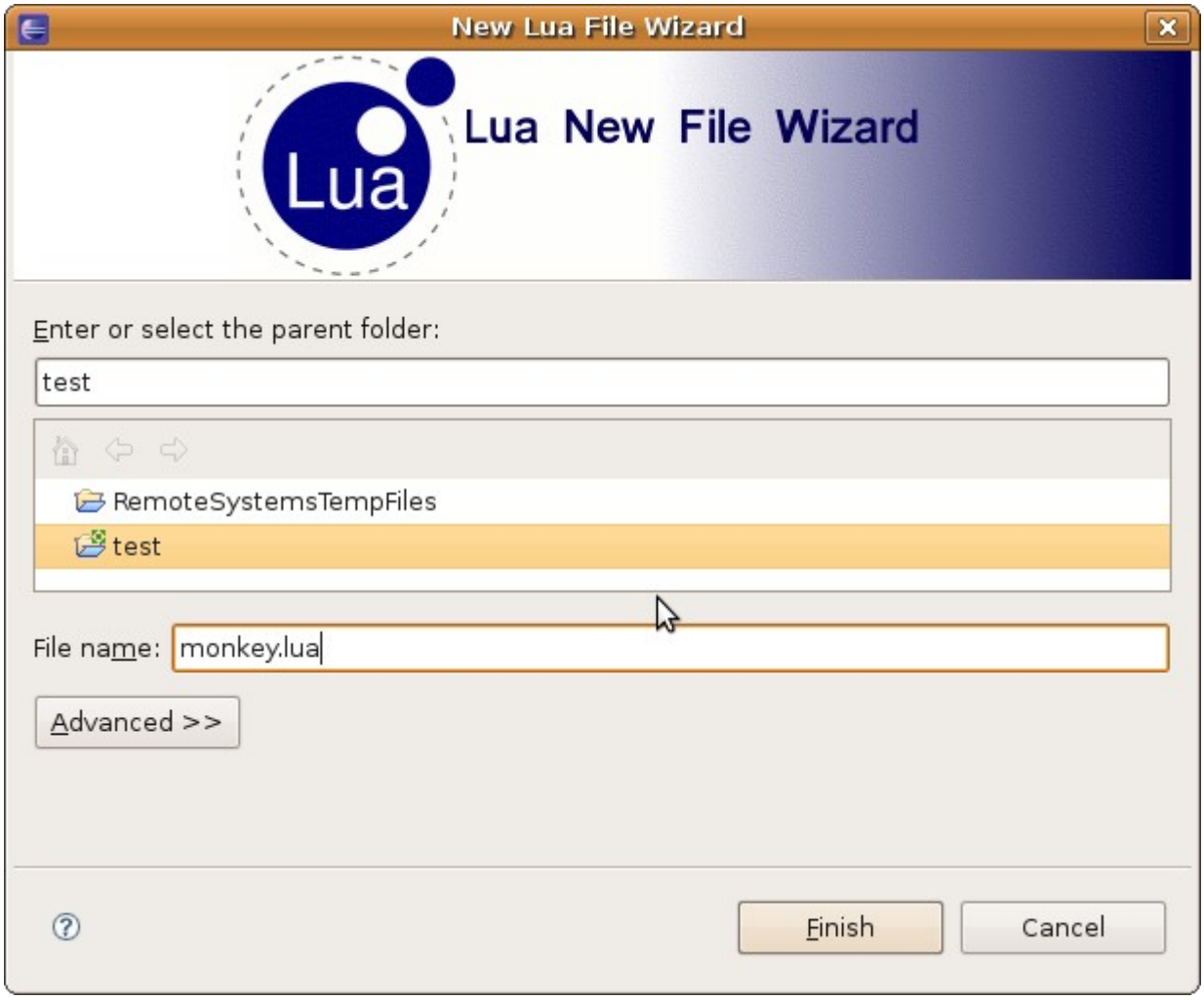

Figura 17: Assistente de criação de arquivo Lua

- Editor NCL – Após a criação com o assistente, o documento NCL é aberto no editor.

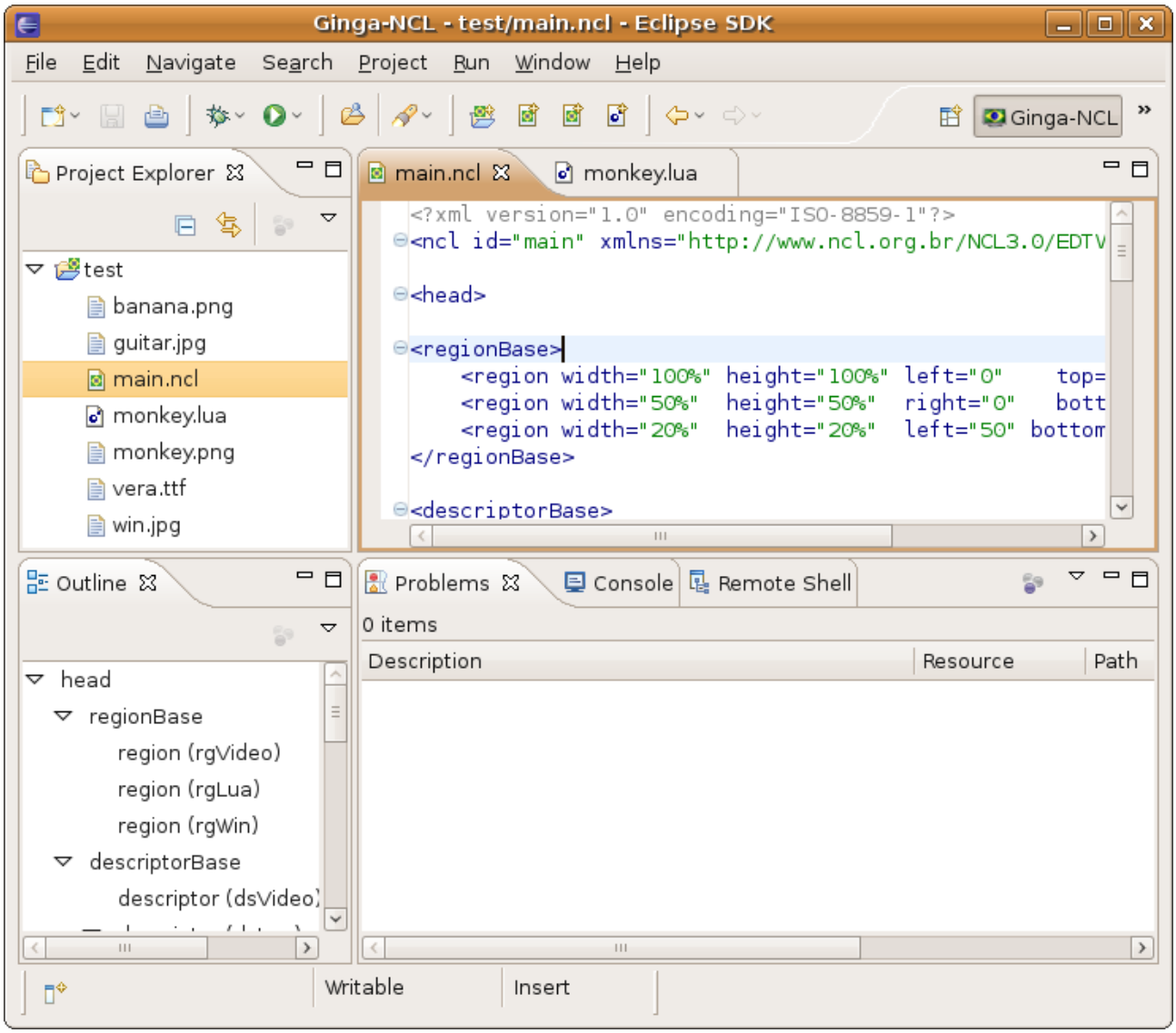

Figura 18: Editor NCL

- Editor Lua – Após a criação com o assistente, o arquivo Lua é aberto no editor.

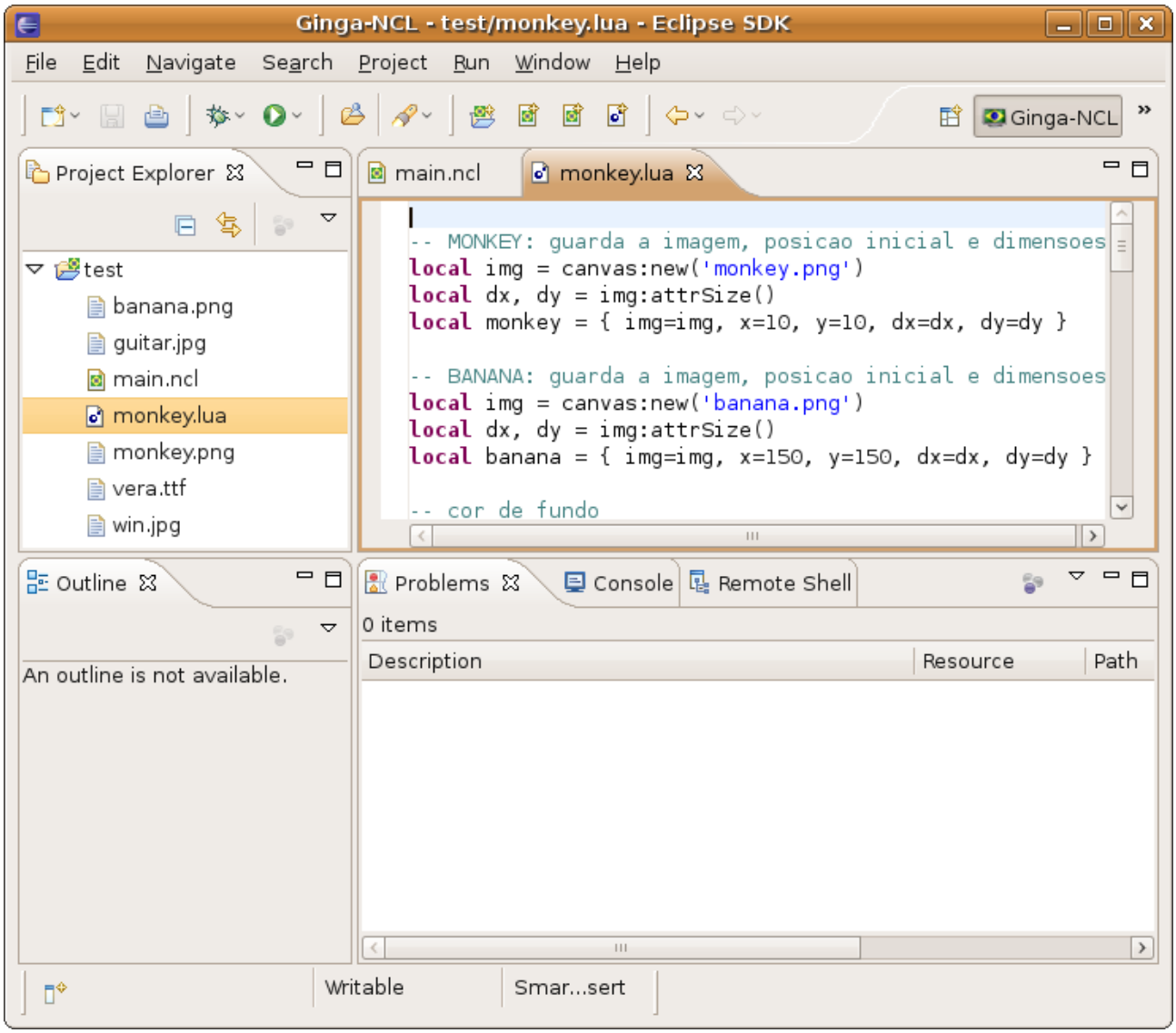

Figura 19: Editor Lua

- Marcadores de problemas NCL – Marcadores de problemas alertam sobre problemas de sintaxe no código NCL.

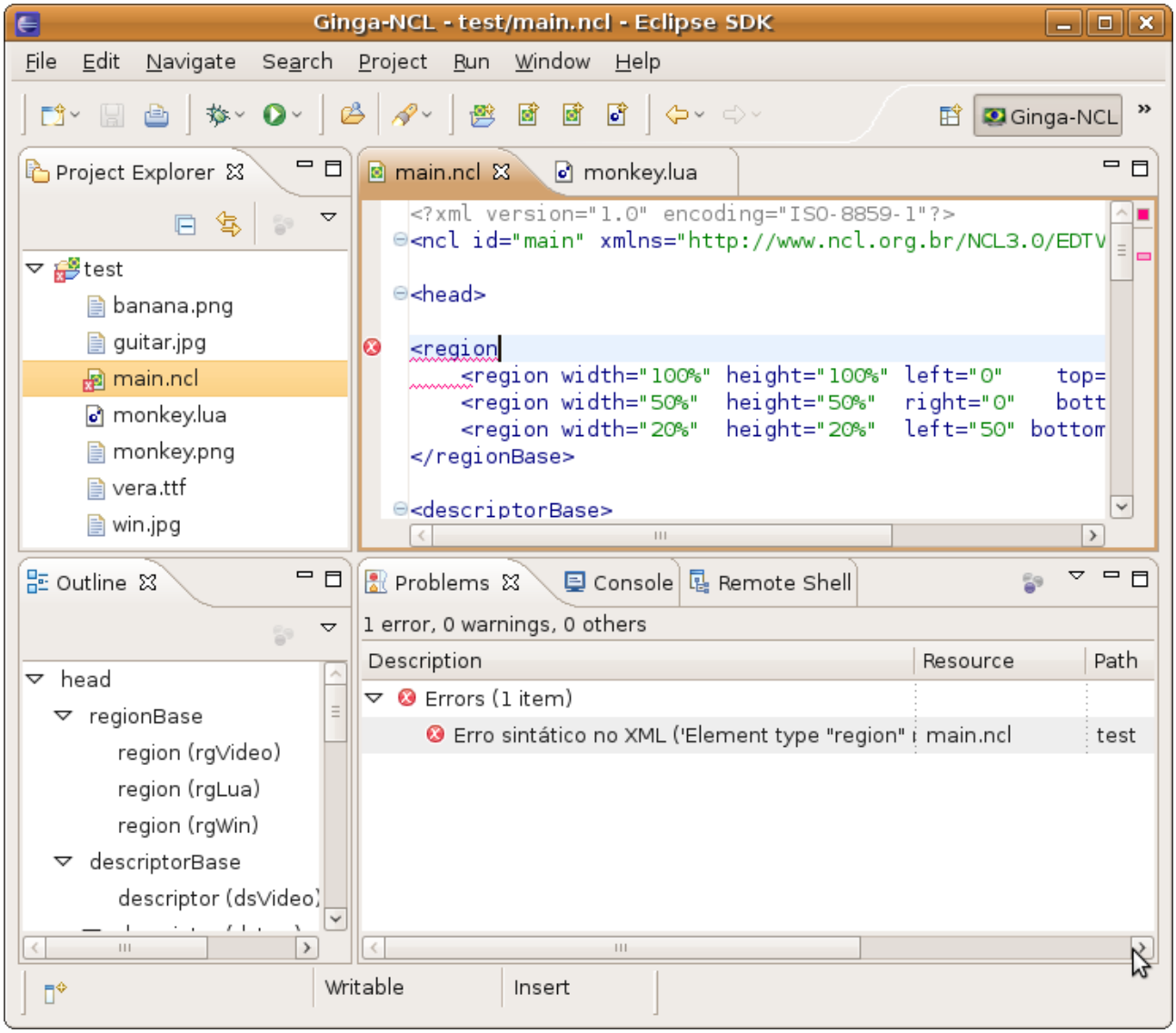

Figura 20: Marcadores de problemas NCL

- Marcadores de problemas Lua – Marcadores de problemas alertam sobre problemas de sintaxe no código Lua.

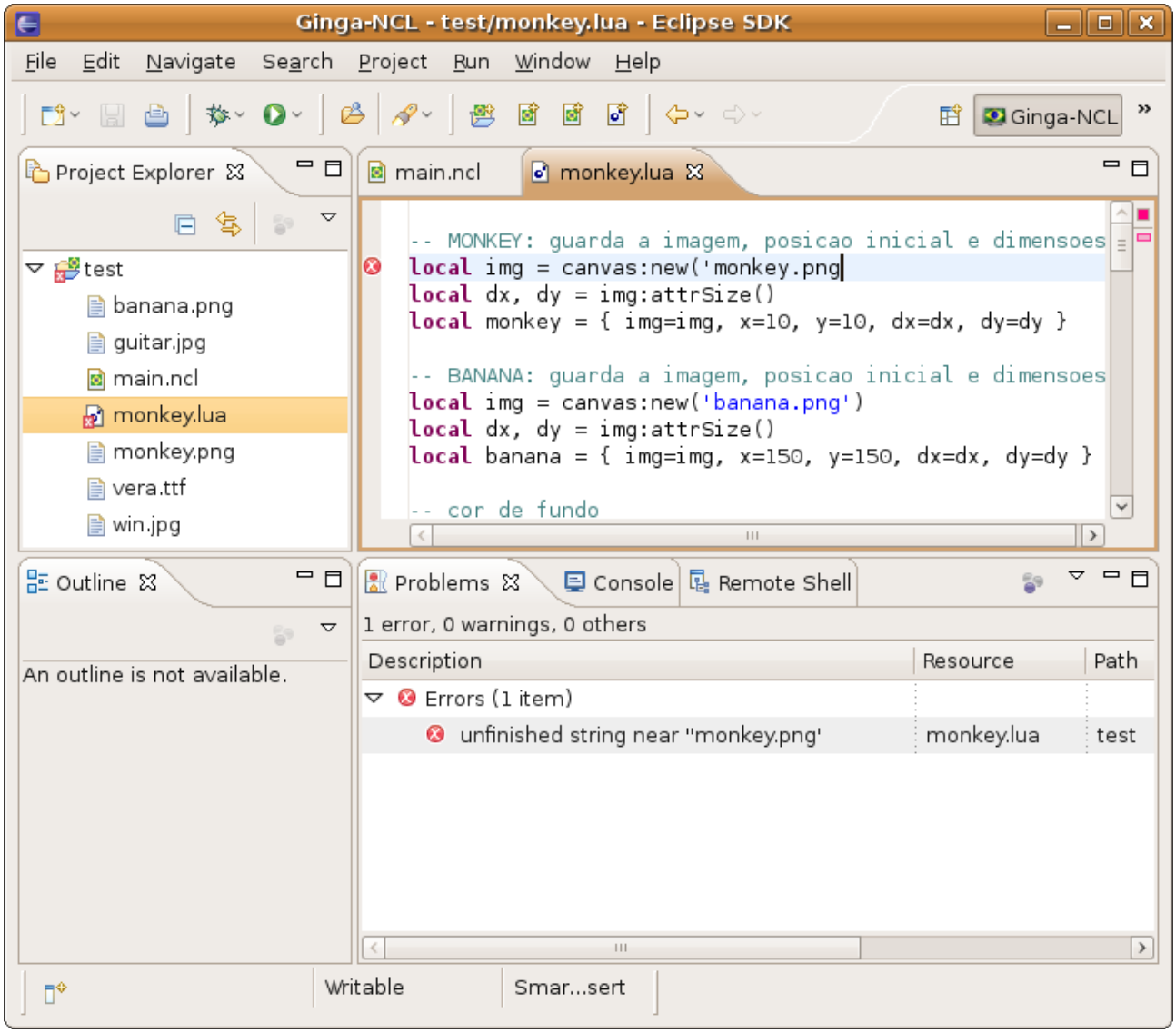

Figura 21: Marcadores de problemas Lua

- Configurações de execução de aplicações Ginga-NCL – Os tipos de configurações de execução do Gingaway são exibidos no diálogo de configurações de execução.

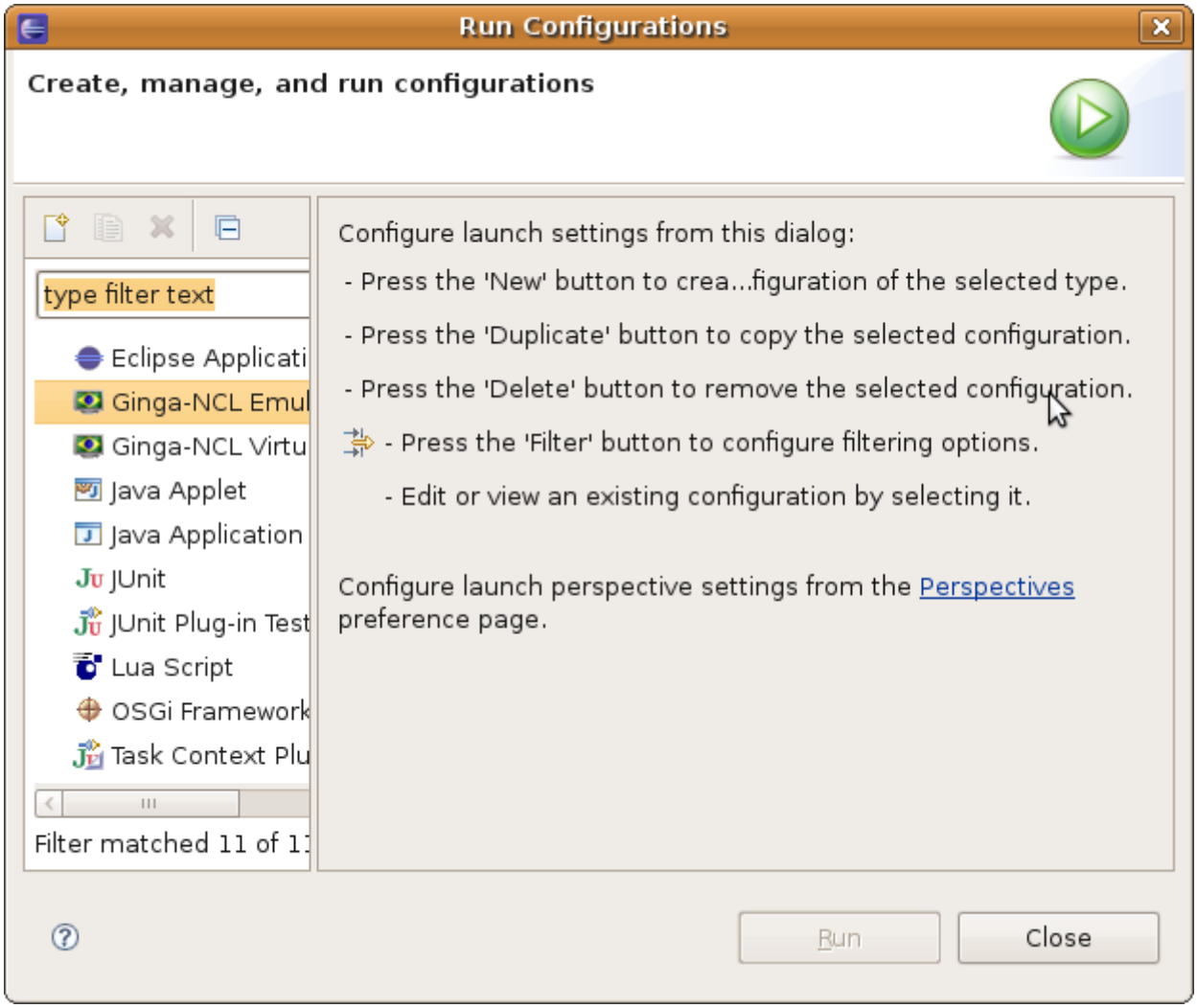

Figura 22: Configurações de execução de aplicações Ginga-NCL

- Configuração de execução com o Ginga-NCL Emulator – É criada uma configuração de execução para execução com o Ginga-NCL Emulator.

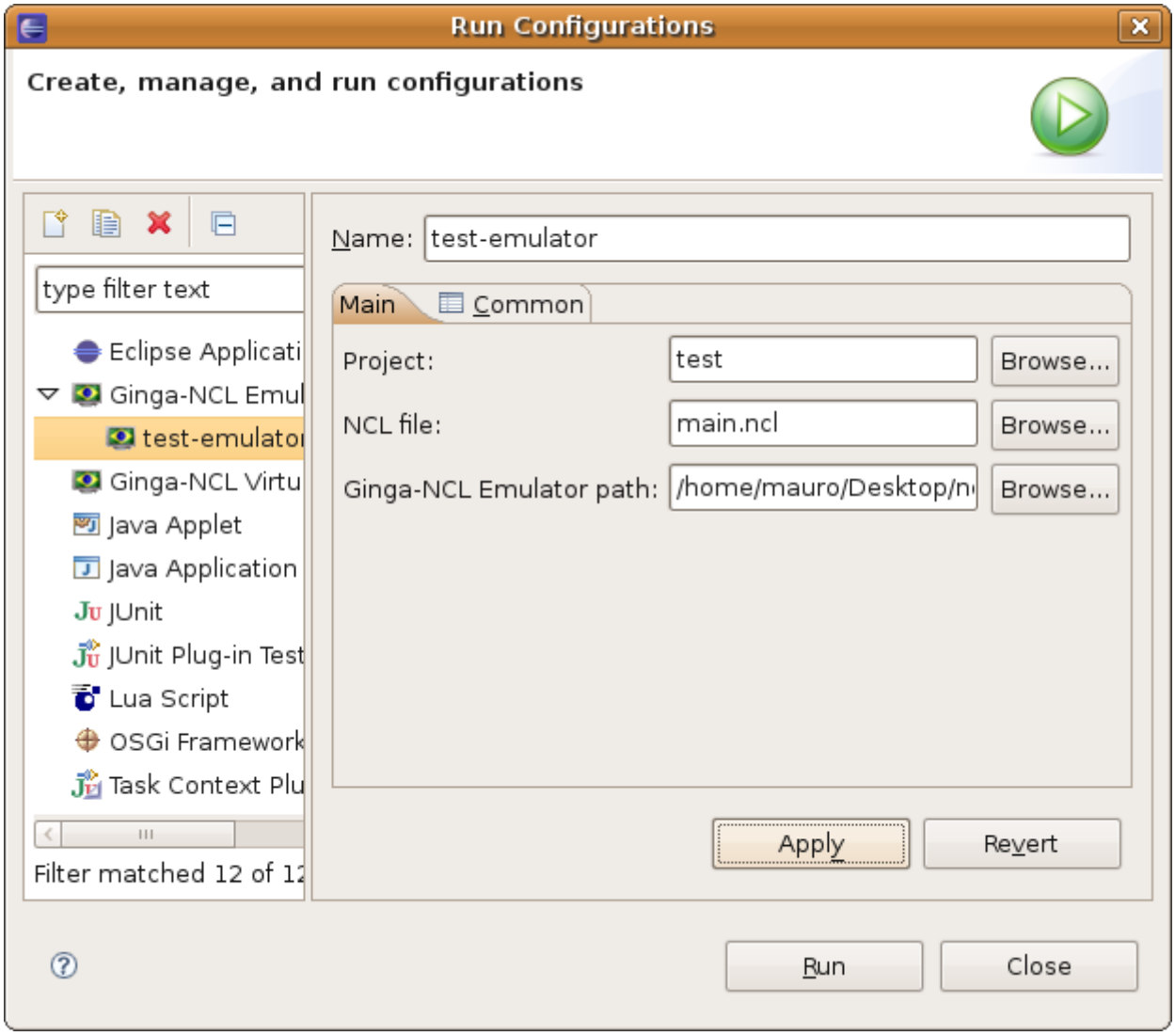

Figura 23: Configuração de execução com o Ginga-NCL Emulator

- O diálogo de seleção de projeto pode ser utilizado para selecionar o projeto que contém o arquivo NCL a ser executado.

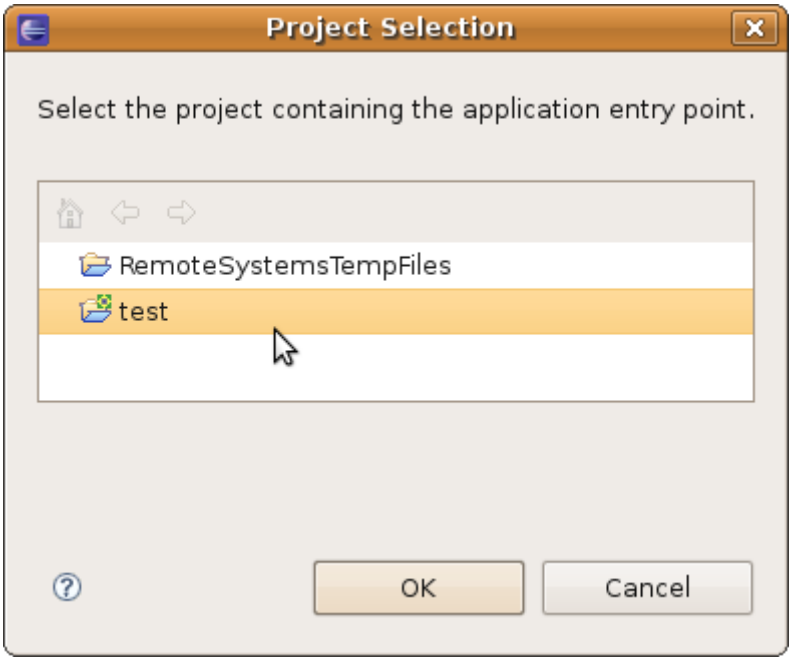

Figura 24: Diálogo de seleção de projeto

- O diálogo de seleção de documento NCL pode ser utilizado para selecionar o arquivo NCL a ser executado.

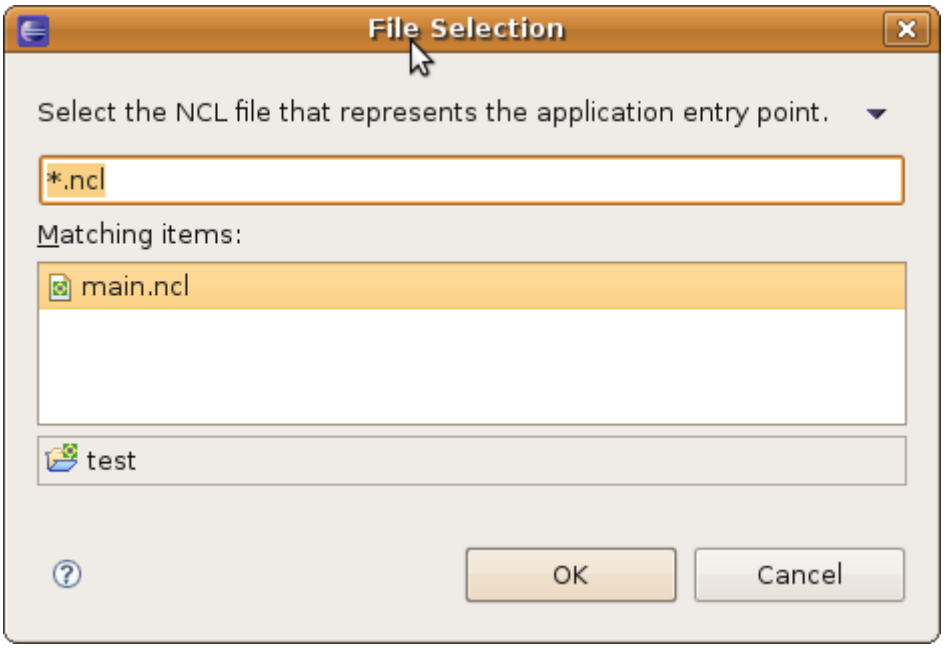

Figura 25: Diálogo de seleção de documento NCL

- Configuração de execução com o Ginga-NCL Virtual Set-top Box – É criada uma configuração de execução para execução com o Ginga-NCL Virtual Set-top Box.

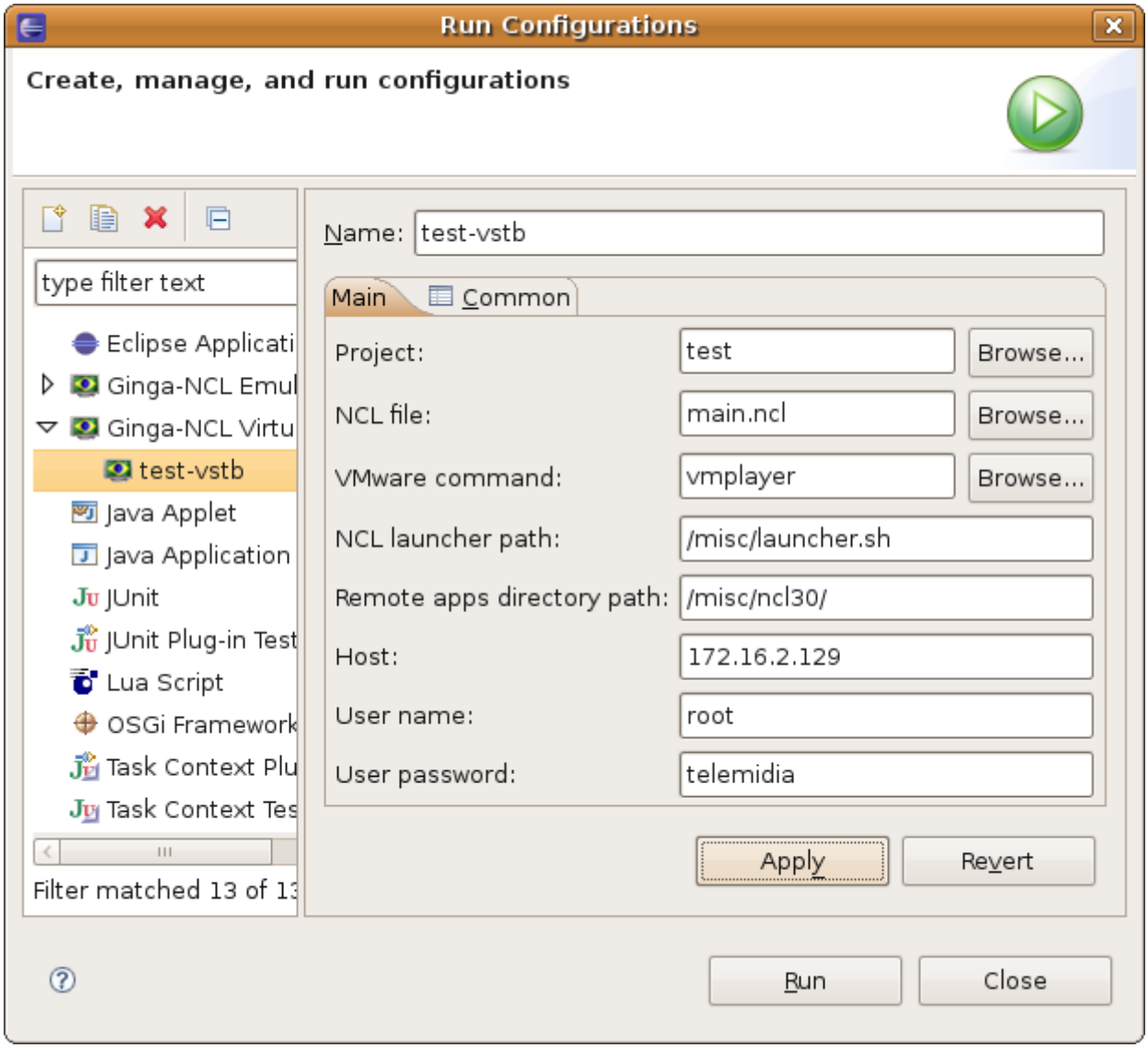

Figura 26: Configuração de execução com o Ginga-NCL Virtual Set-top Box

- A execução da última configuração envia os arquivos do projeto para o Ginga-NCL Virtual Set-top Box e em seguida inicia a aplicação.

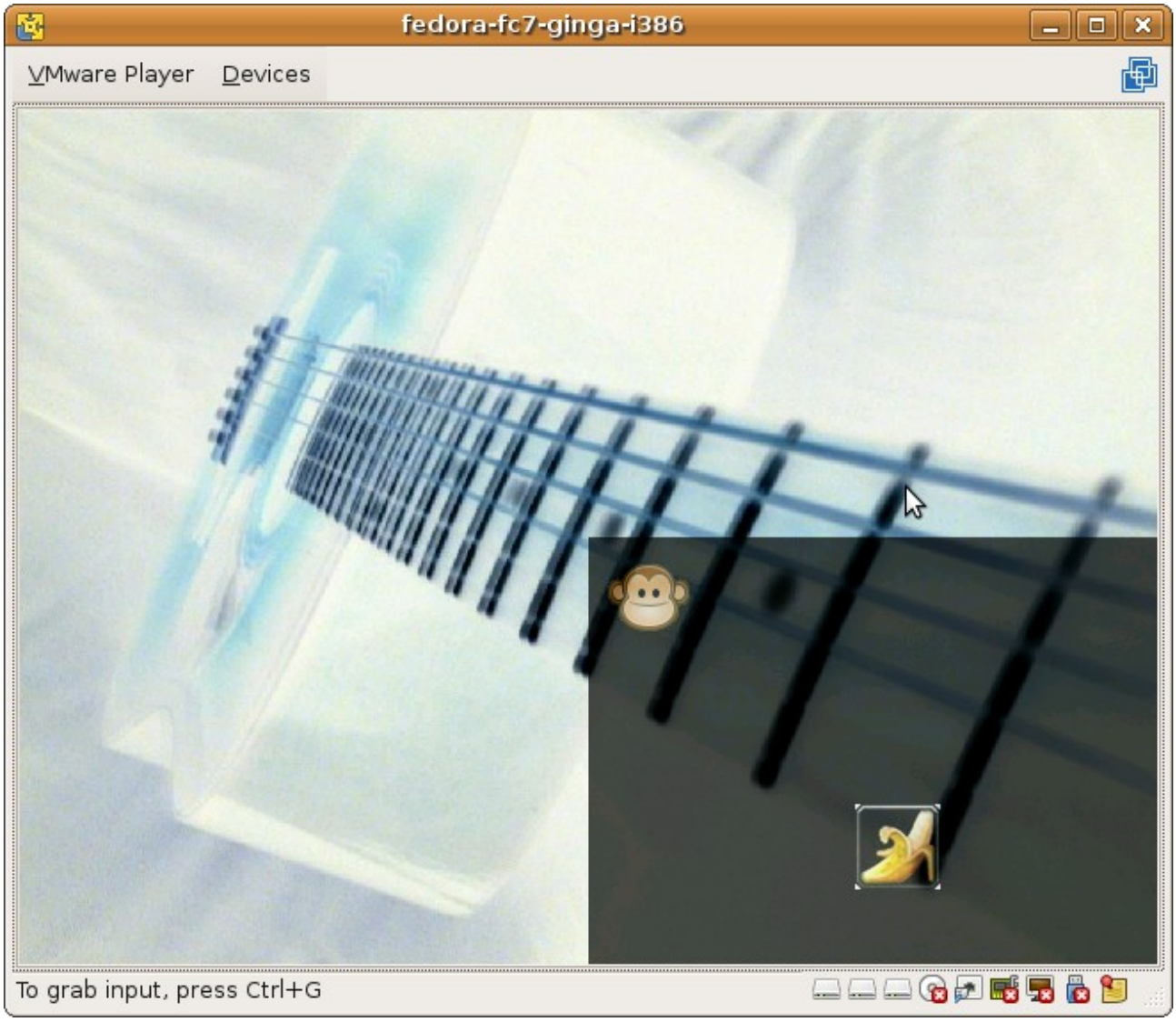

Figura 27: Execução da aplicação no Ginga-NCL Virtual Set-top Box

- Com a execução da aplicação, é criada uma console na IDE que exibe as mensagens da saída padrão da aplicação.

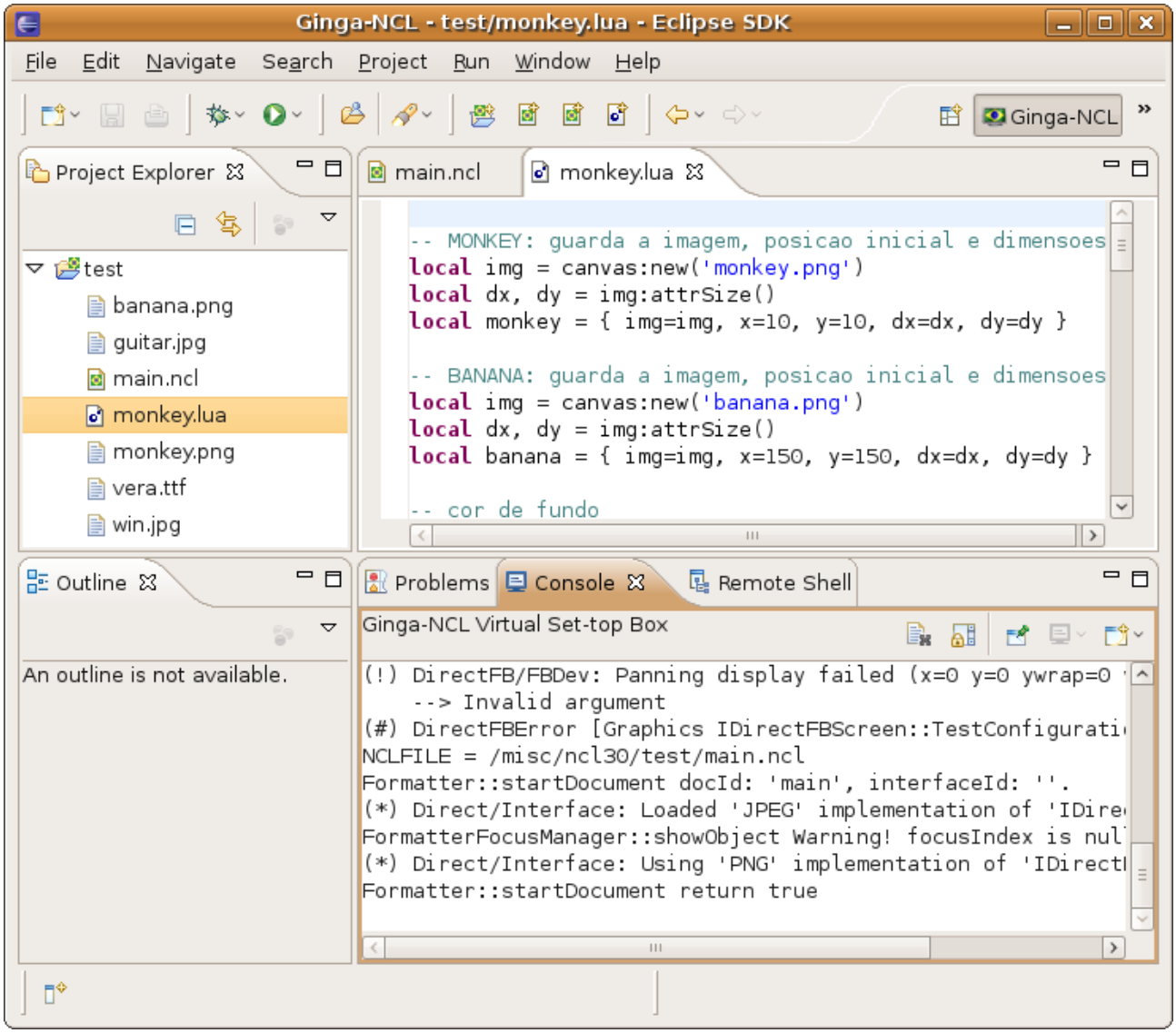

Figura 28: Log da execução na console da IDE

- Geralmente é necessário acessar diretamente a máquina virtual, por exemplo para manipular os arquivos dos projetos enviados ou para executar comandos que interrompem a aplicação. Isso pode ser feito dentro do ambiente, de forma integrada, utilizando o Remote System Explorer.

Figura 29: Diálogo de início de conexão

- O Remote System Explorer provê um terminal para execução remota de comandos.

Esse terminal pode ser utilizado com o *set-top box* virtual.

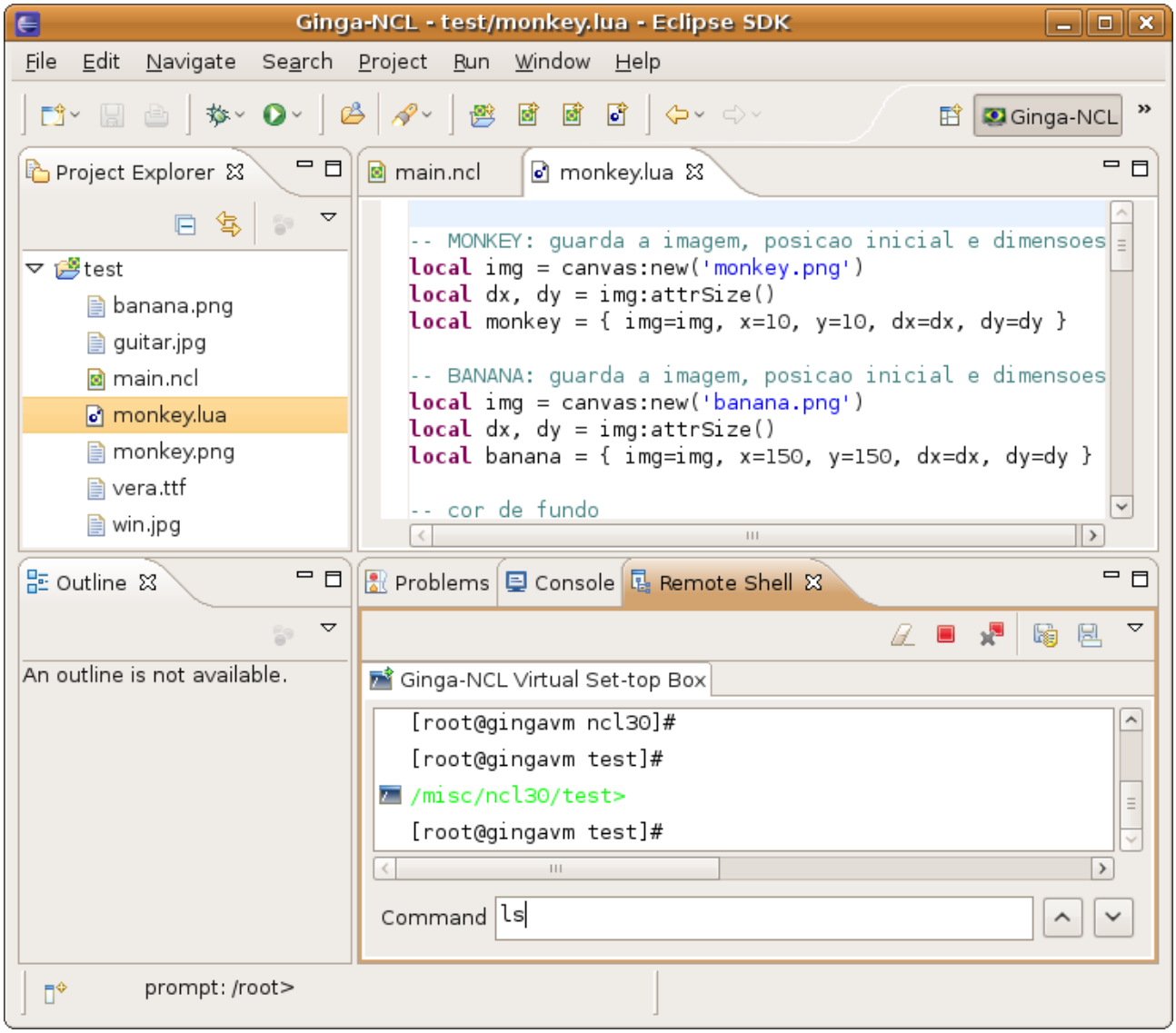

Figura 30: Terminal

6. Conclusões e trabalhos futuros

Este trabalho cumpriu seu objetivo de construir um ambiente de desenvolvimento avançado especializado no desenvolvimento de aplicações Ginga-NCL. Ao longo do estudo, foram analisadas as principais ferramentas e necessidades relacionadas ao desenvolvimento de aplicações Ginga-NCL. A partir do estudo das ferramentas presentes em uma das IDE mais avançadas e populares, a IDE Eclipse para Java, chegou-se aos requisitos do Gingaway. A implementação foi pautada pelo reuso, baseada na Plataforma Eclipse e em dois projetos que forneceram ferramentas centrais, as extensões NCL Eclipse e LuaEclipse.

O Quadro 10 abaixo relaciona os requisitos elicitados, indicando quais foram atendidos, quais foram atendidos parcialmente e também aqueles que não foram atendidos.

[RF01] Assistente de criação de projeto Ginga-NCL	✔ Atendido
[RF02] Assistentes de criação de arquivos NCL e Lua	✔ Atendido
[RF03] Natureza de projeto Ginga-NCL	✔ Atendido
[RF04] Perspectiva Ginga-NCL	✔ Atendido
[RF05] Editores de arquivos NCL e Lua	✔ Atendido
[RF06] *Outline* de arquivos NCL e Lua	✗ Atendido parcialmente
[RF07] Configuração de execução de aplicações Ginga-NCL	✔ Atendido
[RF08] Execução de arquivos NCL e Lua	✔ Atendido
[RF09] Marcadores de problemas	✔ Atendido
[RF10] Barra de tarefas	✔ Atendido
[RF11] *Working sets*	✗ Não atendido
[RF12] Ajuda	✗ Atendido parcialmente
[RNF01] Integração entre as ferramentas dentro do ambiente	✔ Atendido
[RNF02] Seguir padrões da Plataforma Eclipse	✔ Atendido

Quadro 10: Realização dos requisitos

Com a observação do Quadro 10 acima, conclui-se que a implementação do Gingaway atendeu à maioria dos requisitos elicitados. Ela atendeu parcialmente ao requisito *[RF06]* Outline *de arquivos NCL e Lua* porque apenas o *outline* de arquivo NCL está presente. O requisito *[RF12] Ajuda* foi parcialmente atendido porque a documentação incluída na ajuda é atualmente incompleta. O requisito *[RF11]* Working sets não foi atendido porque working sets não foram implementados por falta de tempo, sendo um requisito de mais baixa prioridade.

A seguir é apresentado o Quadro 11, que reexibe o resumo dos diversos ambientes estudados apresentado no Quadro 1 com a adição das características do Gingaway, com o propósito de compará-lo aos demais ambientes. Após observação, conclui-se que o Gingaway apresenta um maior número de funcionalidades e que as diversas ferramentas encontram-se mais bem integradas no ambiente em comparação às alternativas apresentadas.

Outra conclusão deste trabalho dá conta de que grandes são as dificuldades encontradas no desenvolvimento de uma ferramenta tão avançada como uma IDE, ainda que baseada em uma plataforma extensível e rica em recursos como a Plataforma Eclipse e em reuso de projetos de terceiros.

6.1. Trabalhos futuros

Como sugestões de continuidade deste trabalho, destacam-se: a implementação das funcionalidades elicitadas neste trabalho mas não realizadas, a melhoria do código, o aprimoramento das ferramentas já inclusas e a adição de novas ferramentas.

A principal funcionalidade elicitada não realizada é o *outline* de arquivo Lua. Também faltaram a implementação de *working sets* Ginga-NCL e uma documentação de ajuda mais completa.

A melhoria do código existente é importante porque, durante este trabalho, não houve tempo hábil para um trabalho próprio de manutenção do código.

As principais sugestões de aprimoramento referem-se aos editores NCL e Lua, que a exemplo do editor Java presente no JDT, podem evoluir incluindo um grande número de recursos avançados.

Por fim, a criação de versões do Gingaway para outras plataformas.

NCL Eclipse	**Pontos positivos:** Editor NCL com coloração sensível à sintaxe, sugestão de código, marcação de erros e formatação automática. Benefícios da Plataforma Eclipse.	
	Pontos negativos: Emulador Ginga-NCL Emulator acoplado.	
	Ferramentas presentes: Assistente de criação de arquivo NCL Editor de arquivo NCL *Outline* de arquivo NCL Marcadores de problemas Execução integrada de arquivos NCL	**Ferramentas ausentes:** Natureza de projeto Assistente de criação de projeto Configuração de execução Perspectiva
Composer	**Pontos positivos:** Editores visuais de documentos NCL de diferentes perspectivas do código: estrutural, temporal e de layout, além da visão textual.	
	Pontos negativos: Problemas com a interface gráfica e usabilidade, inconsistência gráfica com o ambiente desktop , informações deficitárias sobre erros, suporte restrito a arquivos do tipo NCL e baixa performance.	
	Ferramentas presentes: Assistente de criação de projeto Editor de arquivo NCL Exucução integrada de arquivo NCL	**Ferramentas ausentes:** Ferramentas de suporte a Lua
LunarEclipse	**Pontos positivos:** Recursos do DLTK e da Plataforma Eclipse.	
	Pontos negativos: Projeto descontinuado, suporte a versão antiga do Eclipse (3.3), mal documentado.	
	Ferramentas presentes: Assitente de criação de projeto Assitente de criação de arquivo Lua Editor de arquivo Lua *Outline* de arquivo Lua Perspectiva Lua Execução integrada de *scripts* Lua	**Ferramentas ausentes:** Ferramentas de suporte a NCL
LuaEclipse	**Pontos positivos:** Editor Lua com coloração sensível à sintaxe, marcação de erros e formatação automática. Lua Profiler, ferramenta que expõe detalhes internos de programas Lua em execução. Benefícios da Plataforma Eclipse.	
	Pontos negativos: Ferramenta pode ainda amadurecer bastante, com a adição ferramentas ainda ausentes, como por exemplo sugestão de código e navegação avançada no editor. Há também *bugs*, como na interface de configuração de execução.	
	Ferramentas presentes: Assistente de criação de projeto Assistente de criação de arquivo Lua Natureza de projeto Lua Editor de arquivo Lua Configuração de execução	**Ferramentas ausentes:** *Outline* de arquivo Lua Perspectiva de desenvolvimento Lua

	Execução integrada de arquivos Lua Marcadores de problemas	
NCL Eclipse/ Lua Eclipse	**Pontos positivos:** Reunião de ferramentas NCL e Lua no mesmo ambiente. Benefícios da Plataforma Eclipse.	
	Pontos negativos: Presença de ferramentas indesejáveis, trabalho duplicado com a instalação de duas extensões independentes e falta de integração entre as ferramentas disponíveis.	
	Ferramentas presentes: Natureza de projeto Lua Assistente de criação de projeto Lua Assistente de criação de arquivo NCL Assistente de criação de arquivo Lua Editor de arquivo NCL Editor de arquivo Lua *Outline* de arquivo NCL Execução integrada de arquivos NCL Execução integrada de arquivos Lua Marcadores de problemas	**Ferramentas ausentes:** Natureza de projeto Ginga-NCL Assistente de criação de projeto Ginga-NCL Perspectiva Ginga-NCL *Outline* de arquivo Lua Configuração de execução de aplicações Ginga-NCL
Gingaway	**Pontos positivos:** Reunião de ferramentas NCL e Lua de forma integrada em um mesmo ambiente. Benefícios da Plataforma Eclipse.	
	Pontos negativos: Grau de maturidade e consistência das ferramentas pode ser muito melhorado.	
	Ferramentas presentes: Natureza de projeto Ginga-NCL Assistente de criação de projeto Ginga-NCL Assistentes de criação de arquivo NCL e Lua Perspectiva Ginga-NCL Editores de arquivo NCL e Lua *Outline* de arquivo NCL Configuração de execução integrada de aplicações Ginga-NCL flexível Execução integrada de aplicações Ginga-NCL Barra de tarefas com atalhos para assistentes de criação Marcadores de problemas	**Ferramentas ausentes:** *Outline* de arquivo Lua

Quadro 11: Comparação dos ambientes com o Gingaway

7. Referências

[1]. IBGE. Síntese de Indicadores Sociais 2006. p. 136. 2006.

[2]. Midiativa – Brasileiro vê TV mais de 5 horas por dia. Disponível em: <http://www.midiativa.org.br/index.php/midiativa/content/view/full/2717>. Acesso em: 23 nov. 2008.

[3]. CARAMELLO, Érika Fernanda; TSUBOTA, Guilherme dos Santos. Publicidade no celular: fácil de agradar, difícil de aprovar.. In: INTERCOM 2007 - XXX Congresso Brasileiro de Ciência da Comunicação, 2007, Santos. INTERCOM 2007 - XXX Congresso Brasileiro de Ciência da Comunicação, 2007.

[4]. MENDES, L. L. SBTVD - Uma Visão sobre a TV Digital no Brasil. T&C Amazônia, v. 1, p. 48-59, 2007.

[5]. SARAIVA JÚNIOR, Silvio. A interatividade nos programas esportivos da Rede Bandeirantes de Televisão. In: XXX Congresso Brasileiro de Ciências da Comunicação, 2007, Santos. Anais do XXX Congresso Brasileiro de Ciências da Comunicação, 2007. v. CD-ROM.

[6]. Fórum – SBTVD. Disponível em: <http://www.forumsbtvd.org.br/index.php>. Acesso em: 1 nov. 2008.

[7]. Middleware Ginga. Disponível em: <http://ginga.org.br/>. Acesso em: 2 nov. 2008.

[8]. ABNT. NBR 15606-2: 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 . Rio de Janeiro, 2007.

[9]. Middleware Ginga. Disponível em: <http://www.ncl.org.br/>. Acesso em: 2 nov. 2008.

[10]. The Programming Language Lua. Disponível em: <http://www.lua.org/>. Acesso em: 2 nov. 2008.

[11]. IERUSALIMSCHY, Roberto; FIGUEIREDO, Luiz Henrique de; CELES, W.. The evolution of Lua. In: ACM History of Programming Languages, 2007, San Diego, California. Proceedings of the third ACM SIGPLAN conference on History of programming languages. New York, USA: ACM Press, 2007. p. 2-1-2-26.

[12]. Developer Resources for Java Technology. Disponível em: <http://java.sun.com/>.

Acesso em: 2 nov. 2008.

[13]. Extensible Markup Language (XML). Disponível em: <http://www.w3.org/XML/>. Acesso em: 2 nov. 2008.

[14]. Eclipse.org home. Disponível em: <http://www.eclipse.org/>. Acesso em: 19 ago. 2008.

[15]. BOEKHOUDT, Caspar. The Big Bang Theory of IDEs. New York, USA: ACM Press, 2003.

[16]. Help – Eclipse SDK. Disponível em: <http://help.eclipse.org/ganymede/index.jsp>. Acesso em: 2 nov. 2008.

[17]. Eclipse Java development tools (JDT). Disponível em: <http://www.eclipse.org/jdt/>. Acesso em: 23 nov. 2008.

[18]. PDE. Disponível em: <http://www.eclipse.org/pde/>. Acesso em: 23 nov. 2008.

[19]. NCL Eclipse. Disponível em: <http://www.laws.deinf.ufma.br/~ncleclipse/>. Acesso em: 2 nov. 2008.

[20]. Ginga-NCL Emulator.

[21]. Composer. Disponível em: <http://www.softwarepublico.gov.br/dotlrn/clubs/ginga/composer2/one-community?page_num=0>. Acesso em 23 nov. 2008.

[22]. Dynamic Languages Toolkit. Disponível em: <http://www.eclipse.org/dltk/>. Acesso em: 23 nov. 2008.

[23]. LunarEclipse. Disponível em: <http://lunareclipse.sourceforge.net/>. Acesso em: 2 nov. 2008.

[24]. LuaEclipse: An integrated development environment for the Lua programming language. Disponível em: <http://luaeclipse.luaforge.net/>. Acesso em: 2 nov. 2008.

[25]. Ginga-NCL Virtual Set-top Box.

[26]. Text Editor for Windows. Disponível em: <http://www.contexteditor.org/>. Acesso em: 4 nov. 2008.

[27]. WinSCP :: Free SFTP and FTP client for Windows. Disponível em: <http://winscp.net/eng/index.php>. Acesso em: 4 nov. 2008.

[28]. PuTTY: a free telnet/ssh client. Disponível em:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/>. Acesso em: 4 nov. 2008.

[29]. JDK 1.3.1 Java Archive (JAR)-related APIs & Developer Guides – from Sun Microsystems. disponível em: <http://java.sun.com/j2se/1.3/docs/guide/jar/>. Acesso em 24 nov. 2008.

[30]. Target Management Home page. Disponível em: <http://www.eclipse.org/dsdp/tm/>. Acesso em 25 nov. 2008.

* * *