HighFrame: Uma solução integrada para desenvolvimento em alto nível e deployment automático de sistemas distribuídos heterogêneos baseados em componentes

Artêmio Oliveira de Andrade Junior
Felipe Oliveira Carvalho
Saulo Eduardo Galilleo Souza dos Santos
Tarcísio da Rocha
Victor Machado Pasquialino

São Cristóvão – SE
2014

Sumário

Lista de Figuras

Capítulo 1

HighFrame

Neste trabalho apresentaremos o HighFrame. Inicialmente será contextualizada uma visão geral do papel do *framework*, onde será possível identificar a forma como o HighFrame irá tratar a problemática da complexidade do desenvolvimento de sistemas distribuídos baseados em componentes. Em seguida é apresentada a arquitetura do *framework*. Na última seção é destacado o funcionamento do HighFrame detalhando as etapas do processo de desenvolvimento.

1.1 Visão Geral do *Framework*

O *HighFrame* tem como objetivo simplificar o desenvolvimento de sistemas distribuídos baseados em componentes, permitindo ao desenvolvedor manter o foco no negócio do sistema. Para isso, o *framework* propõe uma solução integrada que inclui: (i) desenvolvimento de componentes baseado em implementações genéricas focada no negócio do sistema; (ii) definição da arquitetura do sistema baseado em um modelo gráfico de alto nível; (iii) *deployment* automático da arquitetura nos nós distribuídos disponibilizando o sistema na sua forma funcional.

O HighFrame fornece uma camada de abstração para o desenvolvimento de componentes. Esta camada é baseada no modelo de anotações Fraclet (ROUVOY; MERLE, 2009. O Fraclet foi o modelo de anotações adotado devido possuir recursos para representar as principais características de diversos modelos de componentes. As anotações disponibilizadas através do HighFrame permite que o usuário desenvolva componentes genéricos, sem ter que conhecer características específicas de um determinado modelo de componentes que é suportado no *framework*.

Para desenvolver os componentes genéricos, as anotações devem ser inseridas em clas-

ses Java simples que implementam o negócio do sistema. Cada anotação fornecida representa um comportamento que é interpretado no *framework* para geração de componentes específicos. O código anotado é inserido em um repositório de componentes genéricos do *HighFrame*, este repositório centraliza todos componentes genéricos em um servidor para que sejam disponibilizados para posteriores reutilização.

Após desenvolver os componentes genéricos individuais e submetê-los ao HighFrame Servidor, o desenvolvedor pode compor a arquitetura do sistema distribuído em alto nível através de um Planejador Gráfico, este planejador é uma ferramenta gráfica cliente do HighFrame Servidor. O Planejador Gráfico permite a construção de um modelo de arquitetura do sistema e o plano de *deployment* em alto nível.

A arquitetura construída através do planejador permite definir os componentes do sistema e para quais modelos de componentes específicos cada um deles serão transformados (Ex. OpenCom, Fractal), bem como, a associação entre os componentes e qual a forma de comunicação entre eles, por exemplo, *bind* local ou remoto (*RMI ou Web Services*). Na arquitetura, é definida também a localização dos nós distribuídos que serão instalados os componentes. Obrigatoriamente, componentes que se comunicam entre nós diferentes da rede devem utilizar alguma forma de *bind* remoto. A interoperabilidade a ser realizada entre modelos de componentes heterogêneos e realizada pelo HighFrame através do framework de interoperabilidade InteropFrame.

1.1.1 HighADL

O HighFrame permite a composição de sistemas através de componentes genéricos para posterior transformação em componentes específicos. Assim como, possui um planejador que faz o plano de *deployment* do sistema distribuído. Neste plano de *deployment* são inseridas as seguintes informações: endereço ip do nó distribuído; modelo de componente que serão criados os componentes específicos no nó distribuído; e, método de comunicação remota distribuída a ser utilizado entre nós distribuídos. O plano de *deployment* é representado através de um arquivo XML. Possui as informações necessárias para que o HighFrame efetue o *deployment* distribuído e a interoperabilidade entre componentes heterogêneos distribuídos.

Para atender as particularidades da definição da arquitetura do sistema distribuído com o HighFrame foi criada uma ADL específica. Esta ADL foi denominada de HighADL e a principal característica que diferencia esta de outras ADL's é o conceito de SubArquiteturas. Uma SubArquitetura define uma composição de componentes que exportam interfaces, as interações com outras composições somente são realizadas através das interfaces exportadas. As SubArqueturas são referenciadas no plano de *deployment* para

definição dos parâmetros necessários para efetuar o *deployment* distribuído. O metamodelo da HighADL é apresentada na figura 1.1.

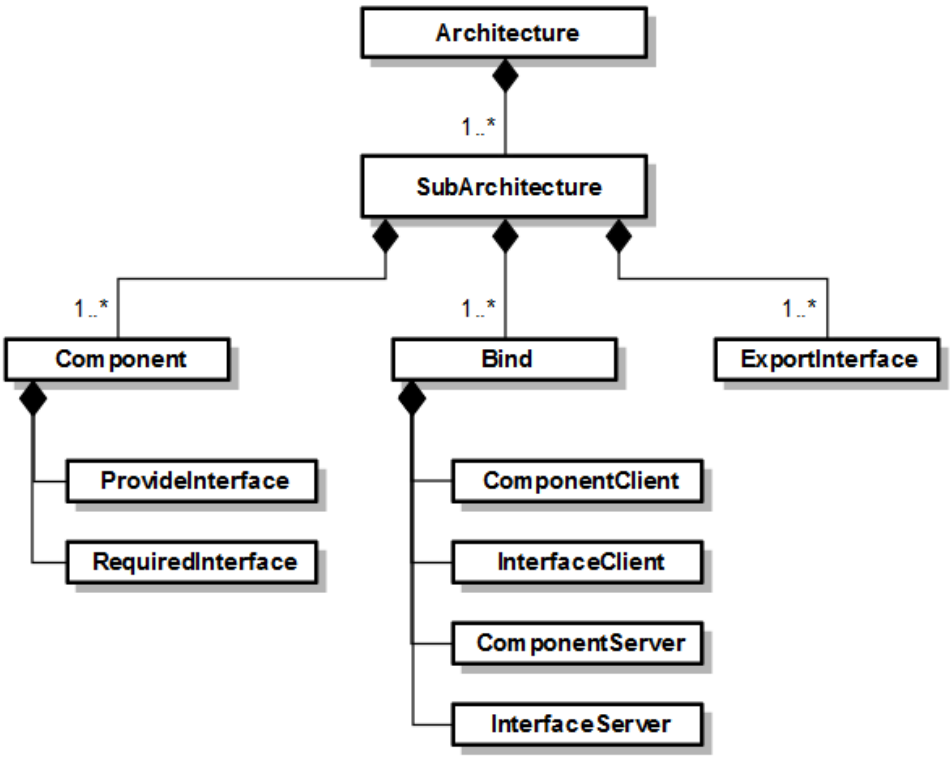

Figura 1.1: *Metamodelo da HighADL.*

O Metamodelo da HighADL possui como elemento raiz a definição *Architecture* e é composta por *SubArchitectures* que possuem os seguintes elementos: *component*, *bind*, *exportinterface*. Com o elemento *component* podem ser definidos os componentes que compõe uma *SubArchitecture*, com o elemento *bind* são definidas as conexões entre os componentes de uma *SubArchitecture*. No elemento *ExportInterface* são definidas as interfaces a serem exportadas, estas interfaces permitiram a conexão de *SubArchitectures* pertencentes a *Architectures* distintas em nós distribuídos.

1.2 Arquitetura do HighFrame

A solução incorpora um conjunto de tecnologias que, aliadas, proporcionam um alto nível no desenvolvimento de sistemas distribuídos baseados em componentes. O *framework* possui módulos distribuídos que concentram responsabilidades bem definidas e a interação entre estes módulos é transparente ao usuário desenvolvedor. A arquitetura distribuída do HighFrame é representada em três níves que possuem características distintas. A figura 1.2 apresenta a arquitetura distribuída do HighFrame, bem como destaca os módulos que compõe a solução.

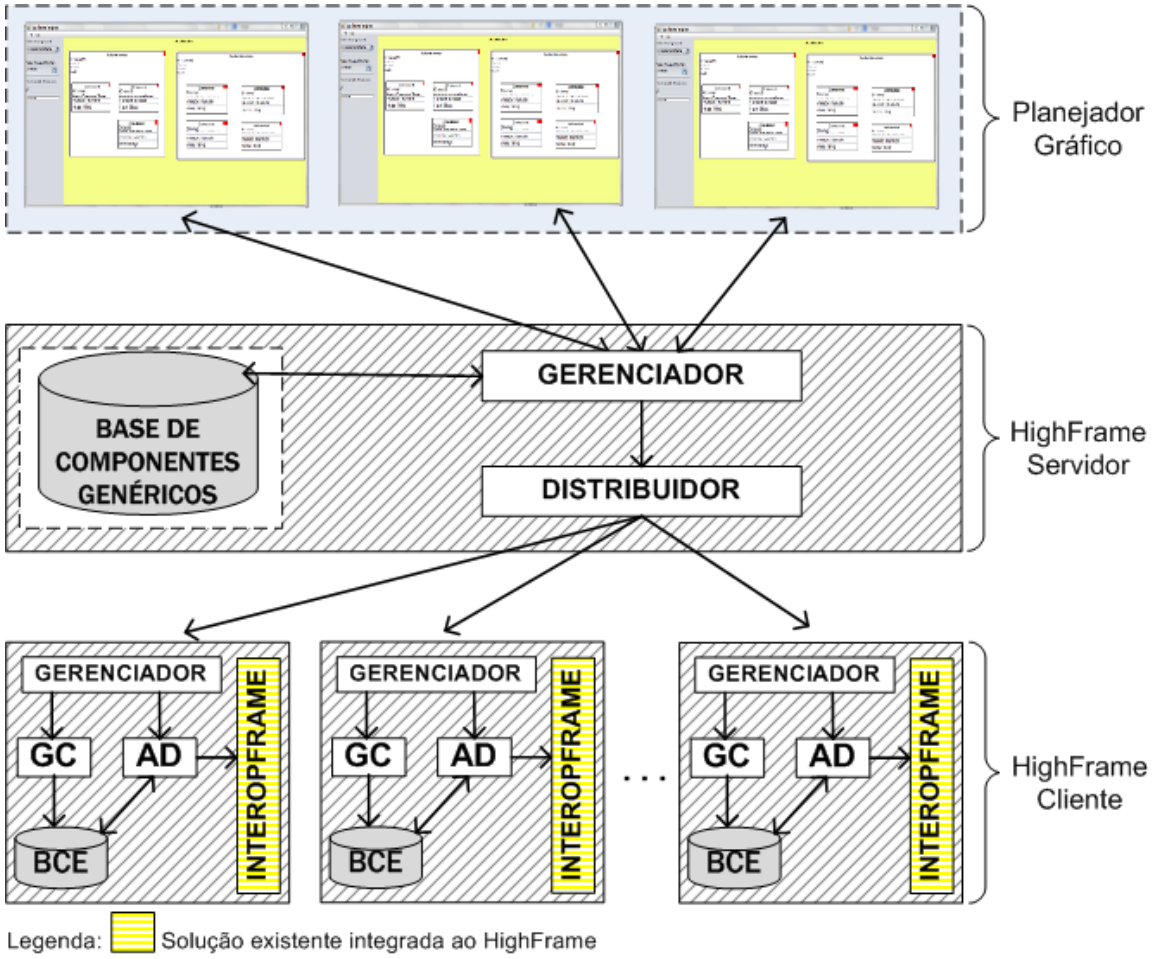

Figura 1.2: *Arquitetura distribuída do Framework.*

No primeiro nível temos os usuários do Planejador Gráfico. O Planejador Gráfico comunica-se com o HighFrame Servidor para obter informações de componentes genéricos existentes no servidor. Com esta informação o planejador disponibiliza ao o usuário os componentes genéricos para composição do sistema e criação do plano de *deployment*.

No segundo nível temos o HighFrame Servidor, que centraliza os componentes genéricos e coordena as ações de *deployment* do sistema distribuído. O HighFrame Servidor é composto pelos seguintes módulos:

- Gerenciador do Servidor: Faz a interface com o usuário desenvolvedor, toda solicitação oriunda do planejador é executada pelo módulo gerenciador. Este módulo, disponibiliza uma lista de componentes genéricos existentes no servidor e coordena todas as ações do HighFrame. Recebe como entrada uma HighADL e um plano de *deployment*, e é o responsável por separar e empacortar os recursos necessários a serem distribuídos entre os nós clientes.

- Base de Componentes Genéricos: Esta base aramazena os componentes genéricos desenvolvidos em Java com as respectivas anotações. Os componentes genéricos da

base ficam disponíveis para reutilização para o desenvolvimento de diversos sistemas.

- Distribuidor: Módulo responsável por enviar os componentes genéricos para os devidos nós distribuídos onde os componentes específicos serão gerados e instalados. O Distribuidor realiza a distribuição dos componentes na rede baseado no modelo de arquitetura submetido ao *framework* através do planejador.

No terceiro nível estão os nós clientes. Neste nível estão os HighFrame Clientes, que são os nós que receberão os componentes de acordo com o plano de *deployment* do sistema distribuído. Uma considerável parcela do processamento do HighFrame foi definida para este nível. Neste nível os componentes genéricos são transformados e implantados, tornando o sistema funional. Os módulos que compões o HighFrame cliente são:

- Gerenciador: Módulo que coordena as ações do HighFrame Cliente. Recebe pacotes do HighFrame Servidor com as definições dos componentes a serem criados e instalados. Responsável por interpretar os arquivos fornecidos pelo HighFrame Servidor, acionar o módulo Gerador de Componetnes (GC) e Agente de *Deployment* (AD).

- Gerador de Componentes (GC): Módulo responsável por gerar os componentes específicos com base nos componentes genéricos. Este é o módulo responsável por tratar a complexidade de cada modelo de componente específico. Os componentes específicos, depois de gerados, são colocados nos suas respectivas base de componentes específicos (BCE).

- Base de Componentes Específicos (BCE): Repositório que armazena os componentes específicos criados através do processamento do HighFrame Cliente. Estes componetnes são posteriormente utilizados para instalação e ativação nos respectivos *runtimes* dos modelos de componentes.

- Agente de *Deployment* (AD): Responsável por instalar e ativar os componentes nos *runtimes* de cada modelo de componente. É também o reponsável por instanciar e ligar os componentes locais do nó e solicitar ao InteropFrame que realize a comunicação entre componentes que se comunicam entre nós distribuídos.

- InteropFrame: É a solução de interoperabilidade entre componentes heterogêneos, realizando a ligação entre componentes de modelos distintos em nós distribuídos. O InteropeFrame gera automaticamente os *proxies* necessários para a ligação baseada em um pré-determinado modelo de comunicação (Ex. RMI, *web services*), instancia os *proxies* e conecta-os aos respectivos componentes para efetivar a comunicação.

O módulo de Gerador de componentes do HighFrame Cliente possui uma estrutura extensível, o que permite suportar novos modelos de componentes específicos. Inicialmente

O HighFrame provê suporte aos modelos de componentes Fractal e OpenCom e métodos de comunicação remota RMI e WebServices.

1.2.1 Tecnologias utilizadas na implementação

Nesta seção são especificadas as tecnologias utilizadas na construção do HighFrame. Será realizada uma abordagem *top-down* quanto aos níveis apresentados na seção anterior.

No primeiro nível está o Planejador Gráfico, a construção do Planejador Gráfico foi realizado com o Netbeans 7.4. O java Swing foi a tecnologia utilziada para construção das interfaces e a comunicação com o HighFrame Servidor foi realizada através da API Apache HTTP para implmentação do cliente HTTP.

No segundo nível está o HighFrame Servidor, as anotações do Fraclet foram adotadas para possibilitar o desenvolvimento genérico. A API do Java *Annotation* foi utilizada para realizar a inspeção de anotações dos componentes genéricos, esta é uma API do próprio JAVA. O JDOM foi a API utilizada para tratar os arquivos em XML da HighADL e do plano de *deployment*. No HighFrame Servidor foi utilizado o *WebService* REST para possibilitar a comunicação com o HighFrame Cliente. O Jersey foi a tecnologia utilizada para implementação do WebService REST. O Servidor Web incorporado na solução para possibilitar o uso do Web Service foi o Jetty.

No terceirto nível está o HighFrame Cliente, que implementa através do Jersey as requisições via *WebService* REST. O módulo Gerador de Componentes (GC) utiliza o Velocity, que é um projeto da Apache para manipulação de dados de arquivos através de *templates*. O HighFrame Cliente incorpora os *runtimes* dos modelos de componentes Fractal e OpenCom por padrão. O InteropFrame foi a suloção incorporada para fornecer interoperablidade entre componentes heterogêneos, como também fornece automaticamente métodos para comunicação entre coponentes distribuídos.

1.3 Funcionamento do HighFrame

O HighFrame recebe como entrada os componentes genéricos em Java com as anotações e o modelo da arquitetura de *deployment* criado através do Planejador Gráfico. Após a submissão, o *framework* realiza uma verificação sintática do modelo de arquitetura fornecido e realiza também a validação das anotações fornecidas no código. Esta validação pode detectar entradas fornecidas de forma incorreta, o que alertará o usuário de possíveis erros existentes.

Para um melhor entendimento do funcionamento do framework relacionamos as etapas do processo que são necessária para o desenvolvimento com a solução proposta. As etapas elencadas no processo são: i) desenvolvimento de componentes genéricos; ii) definição da arquitetura através da Planejador Gráfico; iii) *deployment* automático da arquitetura nos nós distribuídos.

1.3.1 Desenvolvimento de componentes genéricos

Uma das grandes vantagens do HighFrame é permitir que o usuário desenvolvedor mantenha o foco do desenvolvimento no negócio da aplicação a ser desenvolvida, sem ter que preocupar-se com os detalhes técnicos de implementação de componentes em diversos modelos de componente. Com uma grande diversidade de modelos de componentes para construção de sistemas distribuídos, cada vez mais, desenvolvedores têm que lidar com modelos distintos, o que eleva o custo do processo de desenvolvimento do software.

O HighFrame trabalha com um metamodelo em alto nível para definição de componentes, o que permite que desenvolvedores realizem anotações no código Java de negócio para representar preocupações relacionadas a modelos de componentes. As principais anotações do Fraclet que podem ser usadas na implementação de componentes são:

- *@Component* identifica uma classe que define um componente;
- *@Provides* identifica uma interface provida de um componente;
- *@Requires* identifica uma interface requerida por um componente;

Um exemplo de um componente genérico desenvolvido é apresentado no código fonte a seguir.

```
1 @Component(provides = @Interface(name = "r", signature = Runnable.class)
    )
2 public class RequestReceiver implements Runnable {
3
4  private final Logger log = getLogger("comanche");
5
6  private int port;
7
8  @Requires
9  private Scheduler s;
10
11  @Requires
12  private RequestHandler rh;
13 ...
```

No exemplo apresentado temos a definição do componente *RequestReceiver* que fornece uma interface "r". Esta interface refere-se a implementação da classe *Runnable* do Java. O componente apresentado também define duas interfaces requeridas, a primeira é "s", que requer um componente que forneça *Scheduler* e a segunda é "rh" que requer um componente que forneça *RequestHandler*.

O código de negócio com as anotações referente aos componentes genéricos é inserido na base de componentes genéricos. Esta armazena componentes ainda em um modelo independente de tecnologia de componentes. A figura 1.3 representa o processo de inserção na base de componentes genéricos.

Figura 1.3: *Submissão de componentes genéricos no Framework.*

O módulo Gerenciador do HighFrame Servidor faz a interface com o usuário desenvolvedor. Durante a submissão são realizadas validações quanto ao uso correto das anotações disponíveis. O processo de validação faz com que todo o código genérico anotado esteja apto a ser transformado em um componente específico de qualquer modelo de componentes suportado pelo *framework*.

1.3.2 Definição de arquitetura através do Planejador Gráfico

O processo de definição da arquitetura do sistema permite realizar diversas definições quanto ao uso de componentes existentes na base de componentes genéricos. O *framework* disponibilizará uma ferramenta gráfica em alto nível para a composição da arquitetura, onde serão apresentados os componentes da base de componentes genéricos e permite definir um método comunicação remota distribuída suportados pelo *framework*. O módulo Gerenciador do HighFrame Servidor é o responsável por gerenciar e disponibilizar os componentes genéricos para o Planejador Gráfico. A figura 1.4 representa o processo de disponibilização dos componentes genéricos no Planejador Gráfico para composição da arquitetura do sistema.

A figura 1.4 apresenta o processo de disponibilização dos compoenenetes genéricos através do Planejador Gráfico, o que permite a composição em alto nível da arquitetura do sistema. Este processo de definição da arquitetura permite que o desenvolvedor crie SubArquiteturas reutilizáveis, estas serão sinalizadas para transformação em um deter-

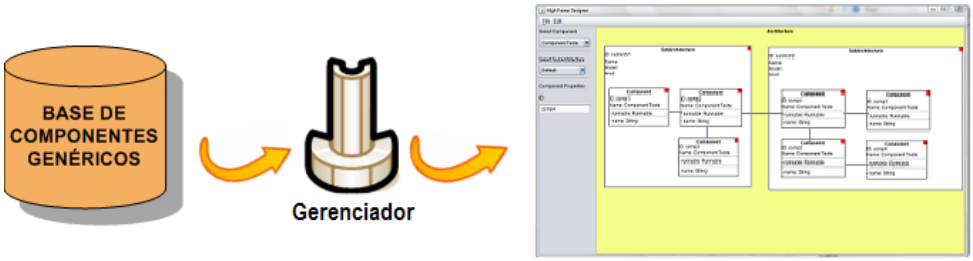

Figura 1.4: *Disponibilização de componentes no Planejador Gráfico.*

minado modelo de componentes, que serão implantadas em diferentes nós da rede, se comunicando através de um dos métodos suportados pelo HighFrame. Com isso o desenvolvedor não precisará conhecer detalhes técnicos da implementação de comunicação entre objetos distribuídos. Todo processo de criação da arquitetura pode ser realizado no modo "arrastar e soltar". Com isso é possível definir os nós distribuídos, seus componentes, interconexões e a forma de comunicação remota entre os componentes que interagem a partir de nós distribuídos.

Após o processo de definição da arquitetura, o módulo Gerenciador do HighFrame Servidor interpreta o arquivo XML referente a modelo de arquitetura construído através do Planejador Gráfico para realizar o processo de enpacotemento de componentes genéricos e o plano de *deployment* de cada nó, para posteriormente realizar a distribuição para os nós distribuídos. A responsabilidade de criação de componentnes específcios é do HighFrame Cliente que executa em um nó distribuído. Um exemplo da arquitetura montada através do Planejador Gráfico é apresentado na figura 1.5.

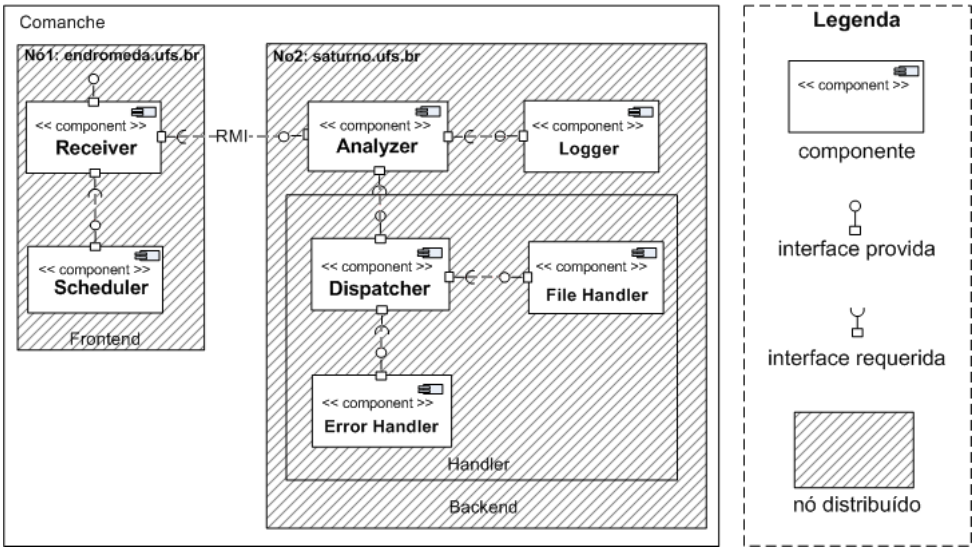

Figura 1.5: *Modelo de arquitetura gerado com o Planejador.*

O exemplo apresentado na figura 1.5 é baseado no *Comanche Web Server* (ROUVOY; MERLE, 2009) que oferece recursos básicos de um servidor web. Este servidor é composto pelos seguintes componentes: *Receiver*, *Shceduler*, *Analyzer*, *Logger*, *Dispatcher*, *FileHan-*

ler e *ErrorHandler*. Para o exemplo apresentado, temos transformações para o modelo OpenCom no *Frontend* e para o modelo Fractal no *Backend*. O componente *Receiver* localizado no nó endromeda.ufs.br conecta-se através de RMI ao componente *Analyzer* localizado no nó saturno.ufs.br ligando assim o *Frontend* ao *Backend*.

1.3.3 *Deployment* automático da arquitetura nos nós distribuídos

O processo de *deployment* automático de componentes distribuídos tem como objetivo realizar a distribuição dos componentes nos nós da rede para quais os mesmos foram destinados, bem como, disponibilizar o sistema em sual forma funcional. Para isso, o HighFrame Servidor interage com o HighFrame Cliente hospedado nos nós distribuídos com o objetivo de realizar a distribuição dos pacotes destinados a cada nó. O HighFrame Cliente recebe os componentes genéricos e plano de *deployment* para realizar a transformação, instalação e ativação dos componentes específicos no *runtime* do modelo de componentes. Em seguida é realizada a conexão remota entre componentes distribuídos de acordo com o modelo de arquitetura fornecido através do Planejador Gráfico.

O HighFrame integra em seu *framework* o InteropFrame, que é um *framework* de interoperabilidade entre modelos de componentes heterogêneos que é responsável por realizar o *Bind* entre os componentes heterogêneos distribuídos. Neste momento são gerados os *proxies* de comunicação remota para os componentes. A figura 1.6 representa o processo de *deployment* da arquitetura em nós distribuidos com diferentes tecnologias de componentes e de comunicação distribuída.

Neste exemplo é solicitado ao *framework* o *deployment* da arquitetura em quatro nós distribuídos. Os nós distribuídos possuem o HighFrame Cliente, onde existem os recursos necessários para supartar modelos de componentes OpenCom e Fractal. Para o exemplo apresentado é solicitada a transformação para o modelo de componentes Fractal nos nós Nó1 e Nó3, e OpenCom nos nós Nó2 e Nó4. A conexão remota entre um componente do Nó1 com outro componente do Nó2 é realizada através de *RMI*, enquanto que a conexão entre componentes do Nó1 com o Nó4 e do Nó3 com o Nó4 são realizadas através de *WebServices*.

O HighFrame permite a utilização de métodos de comunicação *RMI* e *WebServices*. Assim, o usuário desenvolvedor escolhe através do Planejador Gráfico qual método utilizar para conectar componentes distribuídos homogêneos ou heterogêneos. Por fim, o *framework*, instância e conecta todos os componentes deixando o sistema em sua forma funcional.

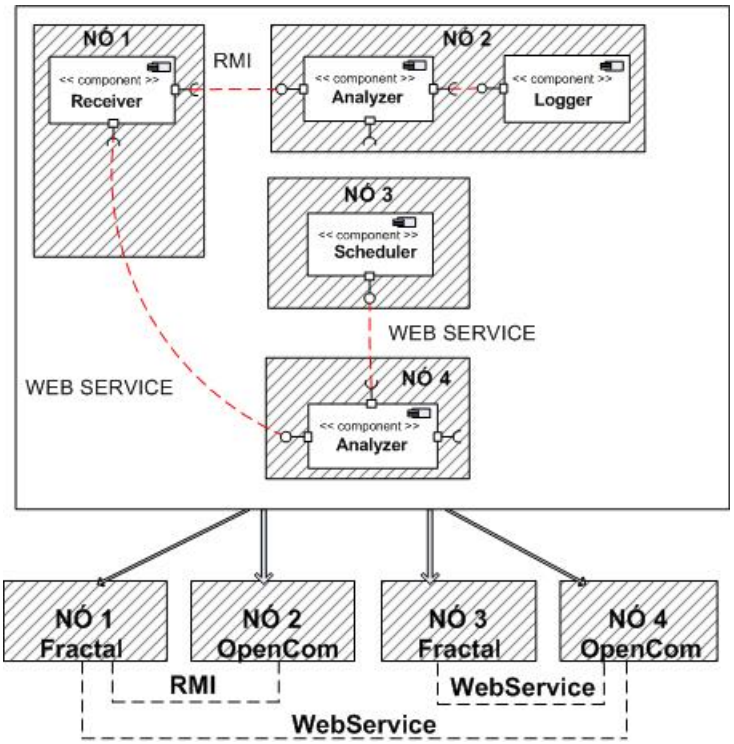

Figura 1.6: *Deployment da arquitetura em nós distribuídos.*

Referências Bibliográficas

BLAIR, G.; Paolucci, M.; Grace, P.; and GEORGANTAS, N. 2011. **Interoperability in Complex Distributed Systems**. In 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Connectors for Eternal Networked Software Systems. Springer.

BRUNETON, E.; COUPAYE, T. 2003. The Fractal Component Model, Speci?cation v2. [S.l.]

BURG, S. van der; DOLSTRA, E. Disnix: A toolset for distributed deployment. Science of Computer Programming, p. 118, abr. 2012. ISSN 01676423.

CALA, J. Adaptive Deployment of Component-based Applications in Distributed Systems. 218 p. Tese (Doutorado) Faculty Of Electrical Engineering, Automatics, Computer Science And Electronics, 2010.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. Distributed Systems: Concepts and Design. 4. ed. Addison–Wesley, 2007. 790 p.

COULSON, G. The design of a con?gurable and recon?gurable middleware platform. Distrib. Comput., Springer-Verlag, London, UK, UK, v. 15, n. 2, p. 109126, abr. 2002. ISSN 0178-2770. Disponível em: < http://dx.doi.org/10.1007/s004460100064>.

CRNKOVIC, I.; LARSSON, M. Building Reliable Component-Based Software Systems. [S.l.]: Artech House, 2002. (Computing Library). ISBN 9781580533270.

RIBA, N.; CERVANTES, H. A MDA tool for the development of service-oriented coomponent-based applications. Eighth Mexican International Conference on Current Trends in Computer Science, 2007.

ROUVOY, R.; MERLE, P. Leveraging component-based software engineering with fraclet. Annal of telecomunications, Springer Verlag, v. 64, p. 65–79, 2009. ISSN 0003–4347.

SANTOS, S. E. G. S; ROCHA, T. **Um Framework Livre para Desenvolvimento de Sistemas Baseados em Componentes Distribuídos.** XIII ERBASE – Escola Regional de Computação – Bahia, Alagoas e Sergipe. FreeBASE – WorkShop de Software Livre, 2013.

SENTILLES, S.; PETTERSON, A.; NYSTROM, D.; NOLTE, T.; PETTERSSN, P.; CRNCOVIC, I., Save–IDE – A tool for design, analysis and implementation of component-based embedded systems, Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on , vol., no., pp.607,610, 16—24 May 2009. doi: 10.1109/ICSE.2009.5070567

SZYPERSKI, C.; CHAUDRON, M. R. V.; REUSSNER. R. Component–based Software Enginnering. Proceedings, 11th International Symposium (CBSE 2008), Karlsruhe, Germany, October 14–17. Springer, 2008.