Um *framework* baseado em Componentes de Software para Aplicações de Gerência de Falhas em Redes de Computadores

Raissa Dantas Freire
Jacques Philippe Sauvé
Departamento de Sistemas e Computação - UFPB/Campus II
Rua Aprígio Veloso, CEP 58109-970, Campina Grande - PB
Fone: +55 (83) 3101119 Fax: +55 (83) 3101124
{raissa, jacques} @dsc.ufpb.br

Abstract

This work presents a component-based framework which allows the visual development of fault management applications for computer networks. Differently from other approaches, this solution frees the applications developer from the details of network management communication protocols and provides the necessary abstractions that enable him/her to concentrate on the desired solutions. As a result, the development of fault management applications becomes an easier and faster task.

Com o nível de abstração fornecido pelas APIs e linguagens atualmente disponíveis - exemplos podem ser encontrados em [Lima, 1998], [Schõnwãlder & Langendórfer, 1995], [Leinen, 1999], [Mellquist, 1997], [Sun Microsystems, 1999], [Hewlett Packard, 1998] e [Simões *et al.*, 1994] - o desenvolvimento de aplicações de gerência não tem sido uma tarefa fácil. Ao utilizar estas APIs, programadores precisam tratar de aspectos de baixo nível que não estão diretamente associados ao domínio de problema considerado. De um modo geral, as APIs disponíveis fornecem uma abstração para a realização de operações SNMP (*Simple Network Management Protocol*), por exemplo, mas deixam de fornecer abstrações de mais alto nível que atendam a necessidades de gerência específicas, como mecanismos mais elaborados para a identificação de falhas no caso da gerência de falhas.

Com o objetivo de fornecer um nível de abstração mais adequado para o desenvolvimento de aplicações de gerência de falhas, permitindo que o programador possa concentrar seus esforços na implementação da funcionalidade básica de sua aplicação, especificamos *um framework* CO (*Component-Oriented*) que permite a construção *visual* de aplicações de gerência a partir dos componentes fornecidos.

Num *framework* CO, cada componente implementa uma parte bem definida da funcionalidade geral *do framework* e da cooperação entre os componentes utilizados, obtém-se a funcionalidade da aplicação. Através da utilização de **componentes de software reutilizáveis** [D'Souza & Wills, 1999], é possível construir a aplicação através de um **ambiente gráfico,** ao **configurar e interconectar** os componentes disponíveis.

Assim, para permitir tratar características específicas de cada aplicação, um *framework* CO fornece componentes que podem ser **visualmente** configurados. Além disso, ele pode fornecer **componentes semiprontos** a partir dos quais novos componentes podem ser criados. Cada novo componente criado passa a fazer parte do *framework* e, uma vez disponível, pode ser também configurado através de um ambiente *gráfico, permitindo,* finalmente, a construção visual da aplicação. A figura 1 mostra esta arquitetura.

gerar um alarme toda vez que a falha ocorrer (GeradorDeAlarmes G_A). Para que o operador da rede seja notificado toda vez que a falha (evento) descrita ocorrer, o GeradorDeEventos G_E deve ser cadastrado no Monitor M, de modo que possa receber a informação de gerência decorrente da monitoração e, assim, avaliar a ocorrência da falha; o GeradorDeAlarmes G_A deve ser cadastrado no G_E, de modo que possa ser informado toda vez que a falha ocorrer, notificando o operador da rede. O fluxo de execução descrito na figura 2 reflete esta situação.

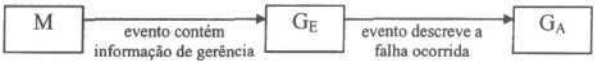

Figura 2. Fluxo de eventos Monitor-GeradorDeEventos-GeradorDeAlarmes

A tabela 1 lista alguns dos componentes até agora disponíveis. Cada componente é instanciado graficamente e cada um tem seu próprio conjunto de propriedades que podem ser configuradas pelo programador. O Monitor, por exemplo, tem uma propriedade chamada periodo que indica a periodicidade com que o Monitor deve obter informação na rede; o ElementoGerenciadoSnmp tem um endereço IP que identifica o agente SNMP por ele representado; e assim por diante. Os componentes utilizam esta configuração para realizar as operações de gerência propriamente ditas, escondendo do programador os detalhes de mais baixo nível associados ao protocolo de gerência utilizado.

ElementoGerenciadoSnmp (eg)	Utilizado para representar as entidades gerenciadas da rede. Fornece componentes do tipo GrupoInfoGerencia.
GrupoInfoGerencia (gig)	Representa a informação de gerência utilizada pela aplicação.
Monitor (monitor)	Responsável pela monitoração síncrona. Controla a freqüência com que um ElementoGerenciadoSnmp deve fornecer determinada informação.
ReceptorDeTrapsSnmp (rTraps)	Responsável pela monitoração assíncrona. Recebe todos os *traps* gerados na rede.
GeradorDeEventoFalha(gEventos)	Identifica falhas na rede. O programador deve "completar" este componente para determinar o tipo de falha a ser identificado. Há três subtipos: GeradorDeEventoFalha Limiar (gLimiar), GeradorDeEventoFalhaCom Histerese (gHisterese) e GeradorDeEvento FalhaTrap (gTraps)
CorrelatorDeEventoFalha (correlator)	Faz correlação de eventos.
GerenciadorDeAlarmes (gAlarmes)	Gera alarmes em resposta à ocorrência de falhas.
GerenciadorDeLogs (gLogs)	Gera *logs* em resposta à ocorrência de falhas.

Tabela 1. Componentes do *framework*

Uma vez instanciados e configurados, os componentes devem ser interconectados de acordo com o modelo fonte/consumidor anteriormente descrito. Cada componente gera seus próprios eventos e determina quais tipos de componentes podem se cadastrar junto a ele para obterem a informação produzida. A figura abaixo mostras as interconexões possíveis. As setas representam o envio de um evento para os componentes cadastrados.

Na figura abaixo, tem-se um *framework* CO que possui seus próprios componentes (CP1, CP2, CP3) e que fornece componentes semiprontos (CS1, CS2) a partir dos quais o programador pode construir novos componentes. À medida em que uma nova aplicação é construída, necessidades particulares podem ser identificadas, por exemplo, a necessidade de verificar a ocorrência de uma determinada falha. Neste momento, para fazer com que o *framework* atenda às novas necessidades identificadas, o programador pode acrescentar o código necessário (P1, P2) *ao framework*, construindo novos componentes (CN1, CN2) a partir dos componentes semiprontos fornecidos. Uma vez que os componentes necessários à aplicação estejam disponíveis, o programador pode simplesmente instanciá-los através de um ambiente gráfico, configurando suas propriedades e interconectando-os (i_1, i_2, ...) conforme necessário.

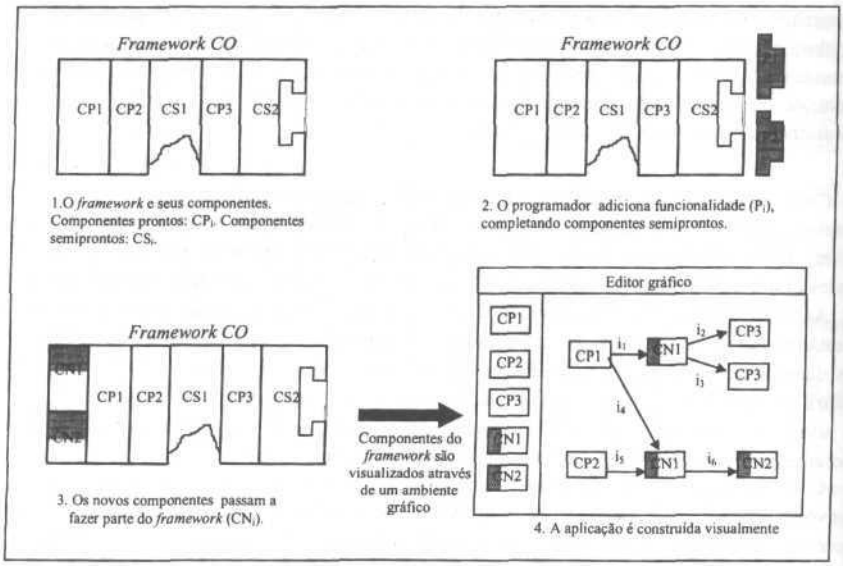

Figura 1. Construindo uma aplicação com base *num framework* CO

Para interconectar os componentes instanciados, um modelo de execução baseado em fontes e consumidores de eventos - também conhecido como "o padrão de projeto *Observer*" [Gamma *et al.*, 1994] - foi adotado. De acordo com este modelo, todo componente é um gerador potencial de eventos (um evento neste contexto, é qualquer resultado decorrente da ação de um componente). Se o componente A está interessado no evento gerado por outro componente B, A deve se "cadastrar" em B, declarando seu interesse em receber tal evento. B, por sua vez, compromete-se a enviar os eventos gerados para A e todos os demais componentes cadastrados. Um mesmo componente ainda pode ser, ao mesmo tempo, fonte e consumidor de eventos (componente-processador).

Suponha, então, que haja um componente responsável por fazer monitoração síncrona (Monitor M), outro responsável por verificar a ocorrência da falha específica "interface do roteador R com defeitos" (`GeradorDeEventos` G_E) e outro responsável por

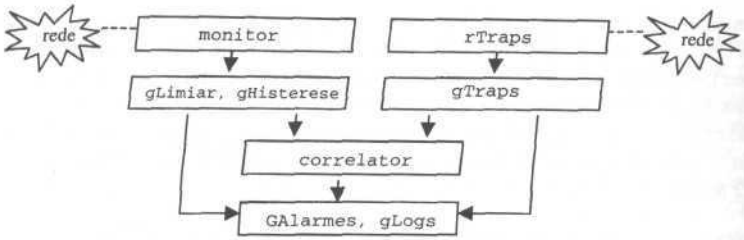

Figura 3: Interconexões possíveis

Os componentes monitor e rTraps são, estritamente, fontes *de* eventos e repassam para outros componentes a informação de gerência colhida na rede; gLimiar, gHisterese, gTraps e correlator são componentes-processadores que consomem a informação recebida para gerar novos eventos; e gAlarmes e gLogs são os consumidores de eventos que dão o tratamento adequado às falhas identificadas. O programador estabelece as conexões entre os componentes instanciados graficamente, configurando o fluxo de controle a ser executado pela sua aplicação. Os componentes disponíveis fornecem um nível de abstração mais elevado para a realização das tarefas de gerência de falhas, permitindo desenvolver aplicações de gerência de uma maneira mais simples e mais rápida.

6. Referências Bibliográficas

[D'Souza & Wills, 1999] D'Souza, D. F., Wills, A. C., *Objects, Components, and Frameworks with UML- The CatalysisApproach.* Addison-Wesley, 1999.

[Gamma *et al.*, 1994] Gamma, E., Helm, R., Johnson, R., Vlissídes, J., *Design Patterns: Elements of ReusableObject-Oriented Software.* Addison-Wesley, 1994.

[Hewlett Packard, 1998] Hewlett Packard Company. *SNMP Developer's Guide.* Edition l, November 1998.

[Lima, 1998] Lima, M. E., *LuaMan: Uma Plataforma para Desenvolvimento de Aplicações de Gerenciamento Extensíveis.* Dissertação de Mestrado, Depto. de Informática, PUC-Rio de Janeiro, Janeiro 1998.

[Mellquist, 1997] Mellquist, P. E., *SNMP++: An Object-Oriented Approach to Developing Network Management Applications.* Prentice-Hall, 1997.

[Schönwälder & Langendörfer, 1995] Schõnwàlder, J., Langendörfer, H., *Tcl Extensions for Network Management.* In Proceedings 3rs Tcl/Tk Workshop, Toronto, Canada, 1995.

[Leinen, 1999] Leinen, S., *SNMP Support for Perl 5.* 1999. *On-line:* http://www.switch.ch/misc/leinen/snmp/perl/index.html.

[Simões *et al.*, 1994] Simões, P. A. F., Brites, A. C. S. C., Leitão, P. M. C. , Monteiro, E. H. S., Fernandes, F. P. L. B., *A High-Level Notation for the Specification of Network Management Applications.* University of Coimbra, Portugal, May 1994.

[Sun Microsystems, 1999] Sun Microsystems Inc. *Java Management Extensions - SNMP Manager API.* August, 1999. Draft 2.0.