Ambiente Integrado para Verificação e Teste da Coordenação de Componentes Tolerantes a Falhas

Mestranda: Simone Hanazumi
Orientadora: Profa. Dra. Ana Cristina Vieira de Melo

Instituto de Matemática e Estatística
Universidade de São Paulo

17 de Novembro de 2008

Dados do Projeto

- **Área:** Engenharia de Software
- **Sub-áreas:** Testes, Verificação Formal e Coordenação de Componentes.
- **Público Alvo:** Desenvolvedores de Software na linguagem Java que fazem uso da coordenação de componentes e que têm interesse em realizar testes e verificação formal de seus sistemas.
- **Apoio:** FAPESP - Processo: 2008/03235-0, com vigência de Ago/08 a Jul/10.

Introdução

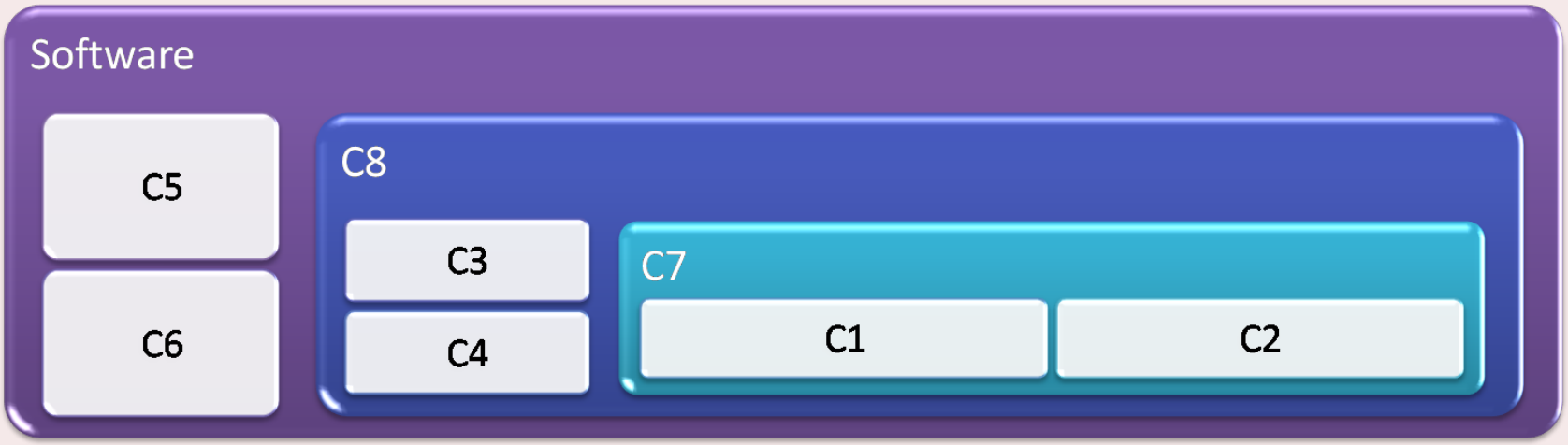
Motivação

1. Mercado competitivo;
2. Necessidade de atender a diferentes requisitos do cliente;
3. Rapidez no desenvolvimento de software.

Desenvolvimento Baseado em Componentes (DBC)

Idéia

Construir software utilizando componentes pré-existentes, ganhando assim rapidez no seu desenvolvimento.



Problemas

1. Garantir que a **integração de componentes não falhe** (sistema construído deve ser robusto e confiável);
2. Uso de **Testes e Verificação Formal** são os mais aconselhados para este fim, mas ambas as atividades possuem **custo alto** para serem desenvolvidas, além da **viabilidade prática** de sua execução ser questionável.

Proposta de Trabalho

Para ajudar na resolução dos problemas citados, este trabalho propõe construir **um aplicativo, que consiste em um ambiente que facilite a realização tanto de testes quanto de verificação de sistemas baseados em componentes tolerantes a falhas, escritos na linguagem de programação Java**. Para isto, as seguintes atividades serão realizadas:

- Definição de um conjunto de propriedades desejáveis sobre a coordenação do comportamento execepcional de componentes;
- Integração de duas ferramentas utilizadas para teste e verificação de programas Java: *OConGraX* e *JavaPathFinder Exceptions*.

Fundamentação Teórica

Escolha da linguagem Java

1. Linguagem utilizada pelas ferramentas a serem integradas neste trabalho, *OConGrax* e *JavaPathFinder Exceptions*.
2. Linguagem Orientada a Objetos;
3. Mecanismos para tratamento de exceção bem definidos na linguagem (*try*, *catch*, *throw*, *finally*);

Componentes de Software

Definição

Componente é uma parte reutilizável de software, desenvolvida independentemente e empregada na construção de unidades maiores, por meio da adaptação do mesmo.

(D'Souza & Wills, 1999)

Componentes de Software

Abordagens para desenvolvimento baseado em componentes:

- *UML Components*;
- *Catalysis*;
- *Architecture Description Languages* (ADLs):
  - *Wright*;
  - *Rapide*.

Integração de Componentes

- Fase do DBC;
- Deve ser:
  - Harmoniosa e Transparente, do ponto de vista dos componentes;
  - Composicional;
  - Preservar propriedades existentes;
  - Lidar com propriedades que emergem das composições e que não podem ser observadas isoladamente.

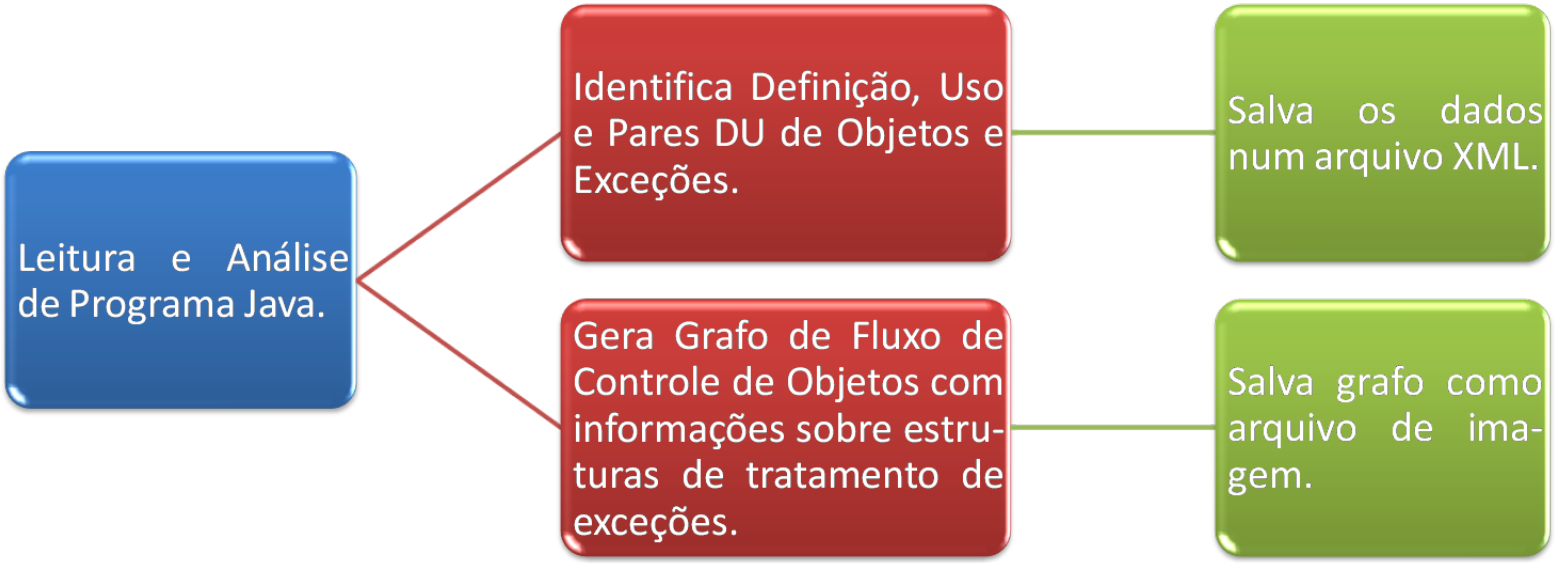
Coordenação de Componentes

- Descreve a forma como componentes e conectores são dispostos para concluir um conjunto de tarefas coletivas ou objetivos comuns.
- Conectores são os responsáveis por ligar componentes; possuem serviços de controle específico para coordenação de componentes.

OConGraX

Ferramenta escrita em Java, para programas em Java. Auxilia na realização de testes estruturais (*White-Box*).

- **Funcionalidades:**



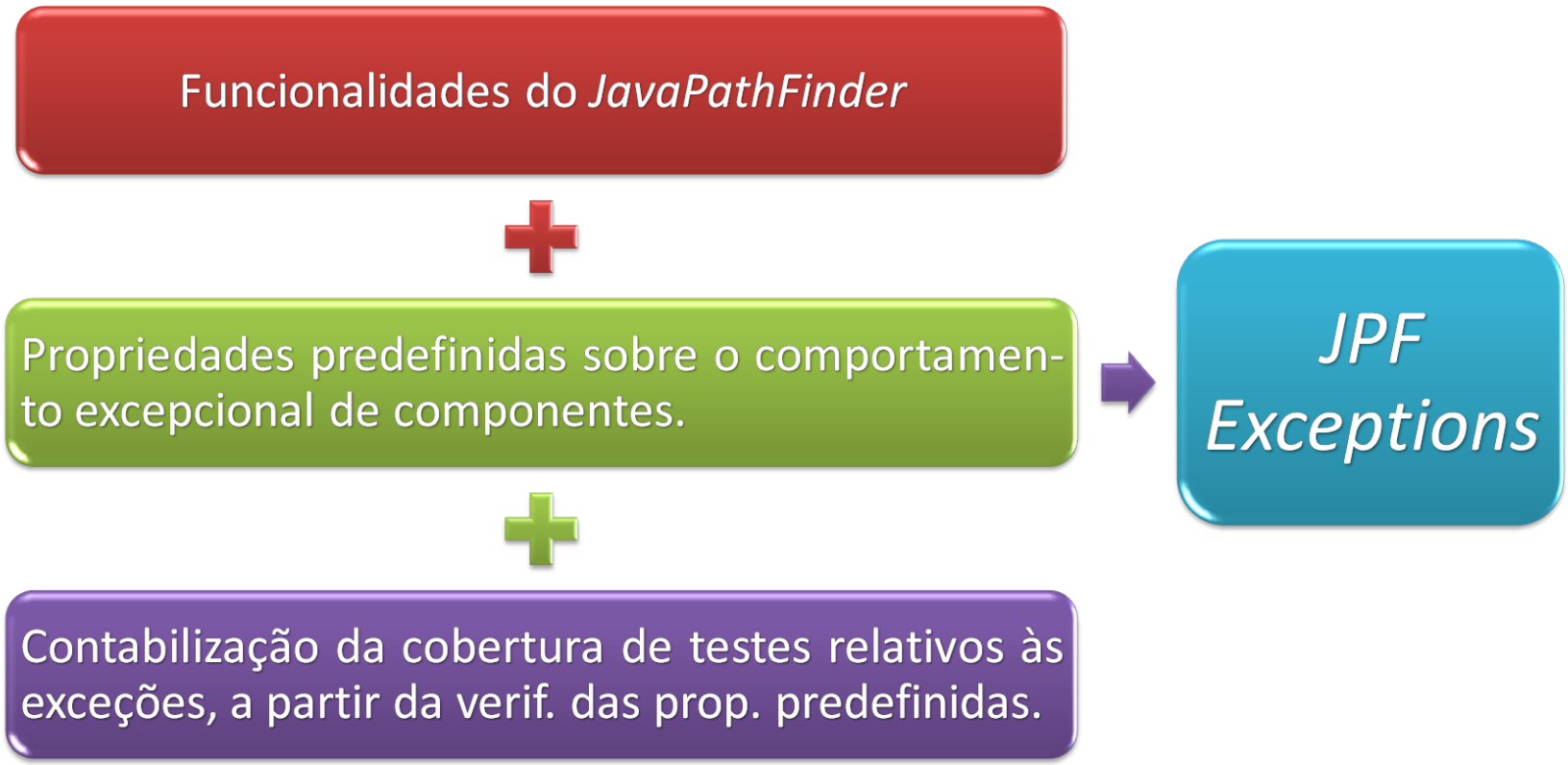
JavaPathFinder (JPF):

- Ferramenta desenvolvida pelo centro de pesquisas da NASA.
- É um verificador de modelo que funciona como uma JVM (*Java Virtual Machine*) executando um programa alvo de todas as maneiras possíveis, procurando por violações de propriedades como *deadlocks* ou exceções não tratadas. Caso um erro seja encontrado, todos os passos seguidos até a ocorrência do erro são exibidos.

JavaPathFinder Exceptions (JPF Exceptions)

Extensão da Ferramenta *JavaPathFinder*.

- **Funcionalidades:**



Objetivos e Métodos

Objetivos Obrigatórios

1. Definição de propriedades sobre a coordenação do comportamento excepcional dos componentes;
2. Integração entre *OConGraX* e *JavaPathFinder Exceptions*;
3. Desenvolvimento do aplicativo em Java.

Objetivo Obrigatório 1:

Definição de propriedades sobre a coordenação do comportamento excepcional dos componentes

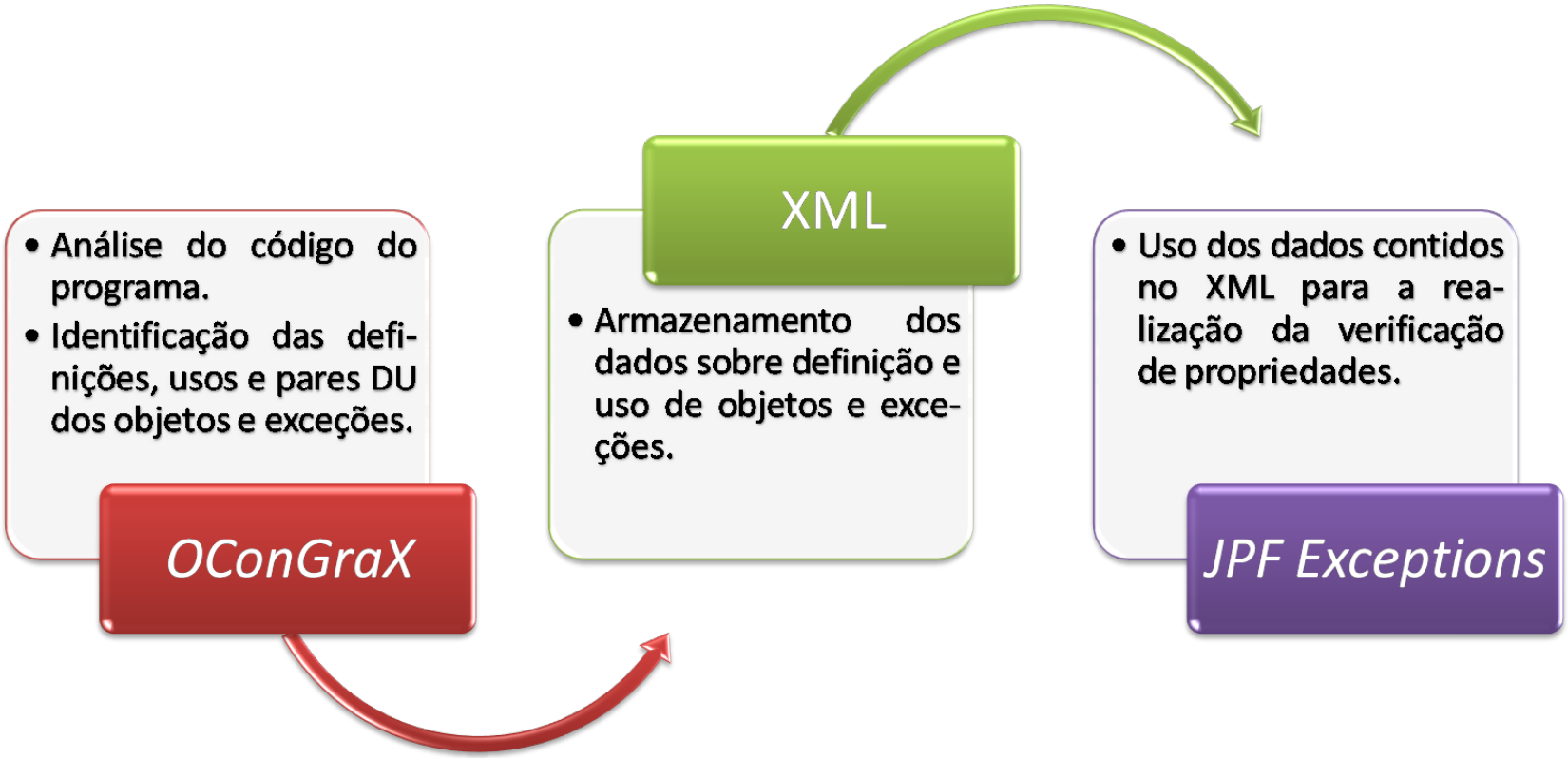
- Predefinir um conjunto de propriedades relativas à coordenação do comportamento excepcional dos componentes para que sejam utilizadas diretamente pelos desenvolvedores.

Metodologia:

1. Estudo de propriedades sobre a coordenação do comportamento excepcional de componentes tolerantes a falhas.
2. Definição, implementação e teste, no *Java PathFinder Exceptions*, das propriedades relativas à coordenação do tratamento excepcional de componentes

Objetivo Obrigatório 2:
Integração entre OConGraX e JavaPathFinder Exceptions

- Permitir uma comunicação entre ambos via um arquivo XML.



Objetivo Obrigatório 2:
Integração entre OConGraX e JavaPathFinder Exceptions

Metodologia:

1. Definição de um modelo XML que permita a comunicação entre os trabalhos a serem integrados;
2. Implementação, na *OConGraX*, de ajustes para geração do modelo XML definido.
3. Implementação, no *JavaPathFinder Exceptions*, de ajustes para geração do modelo XML definido.

Objetivo Obrigatório 3:
Desenvolvimento do aplicativo em Java

- Desenvolvimento da ferramenta proposta em Java, com uma interface gráfica que permita ao testador utilizar facilmente todas as funcionalidades de testes e verificação disponibilizadas.

Metodologia:

1. Criação de uma interface gráfica utilizando **Java Swing**, que unifique os trabalhos sob um único aplicativo.

Métodos de Avaliação

1. Para avaliar o trabalho realizado, serão feitos testes com programas Java que contenham todos os casos tratados pela ferramenta resultante relacionados a:
   - Coordenação de Componentes;
   - Verificação Formal;
   - Testes.
2. Por meio destes testes, ficará comprovado que a ferramenta está funcionando corretamente, fazendo o que foi especificado inicialmente no projeto.

Objetivo Desejável e Trabalhos Futuros

Objetivo Desejável

- Verificação da eficácia e eficiência da ferramenta resultante em sistemas Java reais e de maior porte.

Trabalhos Futuros

- Adição de novas funcionalidades relacionadas a testes, verificação e/ou coordenação de componentes à ferramenta;
- Criar novas versões da ferramenta, ou estendê-la de modo a viabilizar seu uso para sistemas escritos em outras linguagens além de Java.

Contribuição Científica

Ferramenta, que consiste num ambiente integrado para verificação e testes de sistemas Java baseados em componentes tolerantes a falhas, e que auxiliará os desenvolvedores de software Java na realização das atividades citadas.

- **Funcionalidades:**

Plano de Trabalho

Cronograma

Jul/08 a Dez/08

- Estudos Preliminares;
- Definição de um modelo XML que permita a comunicação entre a *OConGraX* e o *JavaPathFinder Exceptions*.

Jan/09 a Jul/09

- Implementação, na *OConGraX* e no *JavaPathFinder Exceptions*, de ajustes para geração do modelo XML definido;
- Definição, implementação e teste, no *JavaPathFinder Exceptions*, das propriedades relativas à coordenação do tratamento excepcional de componentes;
- Criação de uma interface gráfica utilizando *Java Swing*, que unifique os trabalhos sob um único aplicativo (1);
- Escrita da Qualificação do Mestrado.

Cronograma

Ago/09

- Defesa da Qualificação do Mestrado.

Set/09 a Nov/09

- Criação de uma interface gráfica utilizando *Java Swing*, que unifique os trabalhos sob um único aplicativo (2);
- Realização de testes do aplicativo construído.
- Escrita da Dissertação do Mestrado.

Dez/09

- Defesa da Dissertação do Mestrado.

Locais para Possíveis Publicações

Conferência Nacional

- **SBES**: Simpósio Brasileiro de Engenharia de Software

Conferências Internacionais

- **ASE**: *IEEE/ACM International Conference on Automated Software Engineering*
- **ICSE**: *International Conference on Software Engineering*
- **SEFM**: *IEEE International Conferences on Software Engineering and Formal Methods*

Referências

–

Object State Testing for Object-Oriented Programs.
In *COMPSAC'95: Proceedings of the 19th International Computer Software and Applications Conference*, página 232, Washington, DC, USA, 1995. IEEE Computer Society.

ALLEN, R.

A Formal Approach to Software Architecture.
Tese de Doutorado, Pittsburgh, PA, USA, 1997, ISBN 0-591-64744-3.

ALLEN, R., GARLAN, D. e OCKERBLOOM, J.:

Architectural Mismatch or Why it's hard to build systems out of existing parts.
In *ICSE'95: Proceedings of the 17th international conference on Software engineering*, páginas 179-185, New York, NY, USA, 1995. ACM, ISBN 0-89791-708-1.

ANDERSON, T. e LEE, P. A.

Fault Tolerance: Principles and Practice.
Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 1990, ISBN 0387820779.

ARBAB, F.

Reo: a channel-based coordination model for component composition.
In *Mathematical.Structures in Comp. Sci.* 14(3):329-366, 2004, ISSN 0960-1295.

AUGUSTIN, L.M., BRYAN, D., KENNEY, J.J., LUCKHAM, D.C., MANN, W. e VERA, J.,

Specification and Analysis of System Architecture Using Rapide.
IEEE Transactions on Software Engineering, 21(4):336-355, 1995, ISSN 0098-5589.

BANATRE, J. e ISSARNY, V.:

Architecture-based Exception Handling.
In *HICSS'01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 9*, página 9058, Washington, DC, USA, 2001. IEEE Computer Society, ISBN 0-7695-0981-9.

BARBOSA, M.A. e BARBOSA, L.S.

An Orchestrator for Dynamic Interconnection of Software Components.
Electron. Notes Theor. Comput. Sci., 181:49-61,2007, ISSN 1571-0661.

CHEESMAN, J. e DANIELS, J.

UML components: a simple process for specifying component-based software.
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000, ISBN 0-201-70851-5.

CHEN, M.H. e KAO, H.

Testing Object-Oriented Programs – An Integrated Approach.
In *ISSRE'99: Proceedings of the 10th International Symposium on Software Reliability Engineering*, página 73, Washington, DC, USA, 1999. IEEE Computer Society, ISBN 0-7695-0443-4.

CRNKOVIC, I.

Building Reliable Component-Based Software Systems.
Artech House,Inc., Norwood, MA, USA, 2002, ISBN 1580533272.

CRNKOVIC, I., HNICH, B., JONSSON, T., e KIZILTAN, Z.

Specification, Implementation and Deployment of Components.
Communications of the ACM, 45(10):35-40, 2002, ISSN 0001-0782.

D'SOUZA, D.F. e WILLS, A.C.
Objects, components, and frameworks with UML: the catalysis approach.
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999, ISBN 0-201-31012-0.

DENG, X., DWYER, M., HATCLIFF, J., JUNG, G. e RANGANATH, V.P.
Cadena: An Integrated Development, Analysis, and Verification Environment for Component-based Systems.
Software Engineering, International Conference on, 0:160, 2003, ISSN 0270-5257.

DWYER, M. e HATCLIFF, J.
Using the Bandera Tool Set to Model-Check Properties of Concurrent Java Software.
In *CONCUR'01: Proceedings of the 12th International Conference on Concurrency Theory*, páginas 39-58, London, UK, 2001. Springer-Verlag, ISBN 3-540-42497-0.

DWYER, M., HATCLIFF, J., JOEHANES, R., LAUBACH, S., PĂSĂREANU, C., e ZHENG, H.
Tool-supported program abstraction for finite-state verification.
In *Proceedings of the 23rd International Conference on Software Engineering*, páginas 177-187, 2001.

DWYER, M., HATCLIFF, J. e RODRÍGUEZ, E.
Checking strong specifications using an extensible software model checking framework.
In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, páginas 404-420. Springer, 2004.

DWYER, M., PĂSĂREANU, C. e VISSER, W.:
Finding Feasible Counter-examples when Model Checking Abstracted Java Programs.
In *TACAS 2001: Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, páginas 284-298, London, UK, 2001. Springer-Verlag, ISBN 3-540-41865-2.

ENGLER, D.R. e MUSUVATHI, M.

Static Analysis versus Software Model Checking for Bug Finding.
In *Verification, Model Checking and Abstract Interpretation - LNTCS 2937*. Springer-Verlag, 2004.

FOWLER, M.
UML Distilled: A Brief Guide to the Standard Object Modeling Language.
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003, ISBN 0321193687.

GUERRA, P.A.C.

Uma Abordagem Arquitetural para Tolerância a Falhas em Sistemas de Software Baseados em Componentes.
Tese de Doutorado, Universidade Estadual de Campinas, Jun. 2004.

HANAZUMI, S. e MELO, A.C.V.

Ferramenta para Análise do Tratamento Excepcional de Objetos.
Monografia. Disponível em: <http://www.ime.usp.br/~hanazumi/mac499-monografia.pdf>. Acesso em: 22 mar. 2008

HARROLD, M.J. e SINHA, S.

Analysis of Programs with Exception-Handling Constructs.
In *ICSM'98: Proceedings of the International Conference on Software Maintenance*, página 348, Washington, DC, USA, 1998. IEEE Computer Society, ISBN 0-8186-8779-7.

HARROLD, M.J. e SINHA, S.

Criteria for Testing Exception-Handling Constructs in Java Programs.
In *ICSM'99: Proceedings of the IEEE International Conference on Software Maintenance*, página 265, Washington, DC, USA, 1999. IEEE Computer Society, ISBN 0-7695-0016-1.

HAVELUND, K. e VISSER, W.

Model Checking Programs.
In *Automated Software Engineering Journal*, páginas 3-12. Press, 2000.

JAVA PATHFINDER TEAM

Java PathFinder.
Disponível em: <http://javapathfinder.sourceforge.net/>. Acesso em: 22 mar. 2008.

KORSON, T.D. e McGREGOR, J.D.

Integrated Object-Oriented Testing and Development Processes.
Commun. ACM, 37(9):59-77, 1994, ISSN 0001-0782.

MEDVIDOVIC, N., OREIZY, P., ROBBINS, J. E., e TAYLOR, R.N.

Using Object-Oriented Typing to Support Architectural Design in the C2 Style.
SIGSOFT Softw. Eng. Notes, 21(6):24-32, 1996, ISSN 0163-5948.

MELO, A.C.V. e NUNES, P.R.A.F.

OConGra - Uma Ferramenta para Geração de Grafos de Controle de Fluxo de Objetos.
In *Anais do 18o Simpósio Brasileiro de Engenharia de Software*, Brasília, Brasil, 2004.

MEYER, B.

Applying "Design by Contract".
Computer, 25(10):40-51, 1992, ISSN 0018-9162.

SUN MICROSYSTEMS

Java Technology.
Disponível em: <http://java.sun.com/>. Acesso em: 10 nov. 2007.

SZYPERSKI, C.
Component software: beyond object-oriented programming.
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002, ISBN 0201745720.

VOAS, J.M.
The Challenges of Using COTS Software in Component-Based Development.
Computer, 31(6):44-45, 1998, ISSN 0018-9162

WEYUKER, E.J.
The evaluation of program-based software test data adequacy criteria.
Commun. ACM, 31(6):668-675, 1988, ISSN 0001-0782.

XAVIER, K.S.
Estudo sobre Redução do Custo de Testes através da Utilização de Verificação de Componentes Java com Tratamento de Exceções.
Dissertação de Mestrado, Universidade de São Paulo, São Paulo, Brasil, 2008.

Dúvidas?

Fim da Apresentação