Verificação e Validação para Desenvolvimento Distribuído de *Software*: Uma Revisão Sistemática

Geraldo F. C. Neto[1], **Thelma E. Colanzi**[1], **Elisa H. M. Huzita**[1]

[1]Departamento de Informática – Universidade Estadual de Maringá
Av. Colombo, 5790 – 87020-900 Maringá – PR – Brasil

geraldneto@gmail.com, {thelma, emhuzita}@din.uem.br

Abstract. *Nowadays, due to search for the productivity increase, costs reduction and quality improvement, some companies have invested in distributed software development (DSD). This approach has some inherent characteristics to the distribution, that need to be considered to validation and verification activities of the software developed. However, this subject still shows to be incipient and has been explored for few research groups. This paper aims at presenting a systematic review about verification and validation for DSD. Therefore, this work contributes with the academic community supplying subsidies on the scientific production related to verification and validation activities in DSD.*

Resumo. *Atualmente, devido à busca pelo aumento de produtividade, redução de custos e melhoria de qualidade, várias empresas têm investido no desenvolvimento distribuído de software (DDS). Entretanto, esta abordagem tem algumas características inerentes à distribuição, que precisam ser consideradas quando da validação e da verificação dos produtos de software desenvolvidos. Contudo, pode-se perceber que esse assunto ainda se mostra incipiente e tem sido explorado por alguns poucos grupos de pesquisa. Sendo assim, este artigo tem o objetivo de apresentar uma revisão sistemática sobre verificação e validação para DDS. Desta forma, este trabalho contribui com a comunidade acadêmica fornecendo subsídios sobre a produção científica relacionada a atividades de verificação e validação em DDS.*

1. Introdução

Visando o ganho de produtividade, a redução de custos e às melhorias na qualidade, o número de empresas com processos de desenvolvimento distribuído de software tem aumentado. Huzita *et al.* (2007) definem o Desenvolvimento Distribuído de Software (DDS) como a ocorrência de atividades de projeto executadas por membros de equipes que podem estar em locais geograficamente distintos. Essa prática tem se tornado comum, uma vez que as empresas têm explorado um ambiente globalizado, enfatizando a importância dos sistemas de informação nas organizações e, inclusive, adotando práticas de terceirização (*outsoucing*) [AUDY & PRIKLADNICKI 2007]. Dessa forma, os Ambientes de Desenvolvimento de Software (ADS) têm ganhado cada vez mais importância [WIESE *et al.* 2005]. Os ADS buscam combinar técnicas, métodos e ferramentas para apoiar o engenheiro de software na construção de produtos de software, abrangendo todas as atividades inerentes ao processo, tais como de gerência, desenvolvimento e controle da qualidade. Esses ADS devem estar aptos a contribuir para a integração das equipes de desenvolvimento, considerando as suas características de distribuição.

Uma infra-estrutura distribuída eleva a complexidade do software desenvolvido, trazendo a este algumas características desejáveis especificadas por Coulouris (2005), como tolerância a falhas, transparência, abertura, compartilhamento de recursos, paralelismo e escalabilidade. Da mesma forma que as características de sistemas não-distribuídos, estas características devem passar por um processo de verificação e validação específico, visando o cumprimento da especificação do sistema em desenvolvimento.

Dentro das atividades de verificação e validação, uma das mais utilizadas é a atividade de teste, constituindo-se em um dos elementos para fornecer evidências da confiabilidade do software em complemento a outras atividades. O objetivo da atividade de teste é projetar testes que descubram sistematicamente diferentes classes de erros com o mínimo de tempo e esforço. Sendo assim, a atividade de teste é de fundamental importância para a identificação e eliminação de erros que persistem [PRESSMAN 2002]. Desta forma, convencionou-se denominar de atividades de VV&T todas as iniciativas de verificação, validação e teste de software.

O grupo de estudos de Engenharia de Software Distribuído do Departamento de Informática da UEM tem desenvolvido várias pesquisas abordando o desenvolvimento distribuído de software e, atualmente, tem dedicado esforços iniciais em avaliar atividades de VV&T para ambientes de desenvolvimento distribuído de software (ADDS). Por este motivo, foi realizada uma revisão sistemática sobre esse assunto.

Segundo Kitchenham (2004), o método de revisão sistemática (RS) é utilizado no intuito de conhecer, de maneira imparcial, o estado-da-arte de um determinado assunto, técnica ou método. Para que essa análise seja realmente executada de modo imparcial, os critérios de busca, inclusão e exclusão de trabalhos são definidos antes mesmo do início das buscas, possibilitando e incentivando a repetição da pesquisa segundo os mesmos critérios, para que ela seja considerada completa.

A revisão sistemática sobre Estratégias de Verificação e Validação para Desenvolvimento Distribuído de Software foi realizada segundo os padrões propostos por Kitchenham (2004). Tal revisão se justifica pela dificuldade em obter material referente a esse assunto e pela inexistência de outra RS equivalente. Seu objetivo foi o de identificar o estado-da-arte no que tange a VV&T de DDS.

Este trabalho apresenta parte da revisão sistemática realizada, abordando alguns pontos do protocolo definido e alguns resultados obtidos no decorrer da revisão. O texto está estruturado da seguinte maneira: na Seção 2, estão as informações referentes ao planejamento e à condução da revisão; na Seção 3 são descritos os resultados obtidos na revisão sistemática; um quadro comparativo é mostrado na Seção 4, classificando os trabalhos encontrados segundo os critérios definidos na revisão; a Seção 5 contém as considerações finais e indicações para trabalhos futuros.

2. Planejamento e Condução da Revisão

A presente revisão sistemática foi planejada segundo o modelo de protocolo apresentado por Ferrari (2006), com base no modelo proposto por Kitchenham (2004). O relatório completo, no qual estão explicitadas características mais específicas do protocolo de revisão, como detalhes dos critérios de inclusão e de exclusão, estratégias de busca específicas para cada máquina de busca, as referências de todos os trabalhos encontrados, está disponível no *site* do grupo de pesquisa: www.din.uem.br/disen .

O protocolo foi revisado por pesquisadores do nosso grupo de pesquisa e algumas sugestões foram analisadas e incorporadas ao protocolo de revisão. Na seqüência apresenta-se parte do protocolo utilizado na revisão, incluindo: os objetivos principais da pesquisa; as questões e critérios que influenciaram na inclusão ou exclusão dos resultados; as estratégias de busca; a seleção preliminar dos trabalhos observados; a seleção final dos trabalhos obtidos e a condução da revisão.

2.1. Objetivo

O objetivo desta revisão sistemática foi identificar e analisar técnicas e critérios de VV&T aplicáveis a software distribuído (SD). Devido às necessidades do grupo de pesquisa, consideramos também como objetivos da revisão identificar técnicas e/ou critérios específicos em VV&T para DDS, e identificar técnicas de VV&T aplicáveis a ADDS. O objetivo principal está relacionado a SD devido à possibilidade de adaptação para o contexto de DDS e por se tratar de uma área um pouco mais amadurecida.

2.2. Questões de Pesquisa:

As questões que guiaram a condução da RS foram:

- Questão Primária: Quais técnicas de VV&T tem sido utilizadas em Software Distribuído?
- Questão Secundária 1: Quais características específicas de Software Distribuído são abordadas pelas técnicas encontradas?
- Questão Secundária 2: Quais das técnicas de VV&T encontradas podem ser utilizadas em Desenvolvimento Distribuído de Software?
- Questão Secundária 3: Quais das técnicas de VV&T encontradas podem ser aplicadas a Ambientes de Desenvolvimento Distribuído de Software Orientado a Objetos (OO)?

A questão secundária 3 foi criada considerando que nosso grupo de pesquisa está desenvolvendo um ADDS orientado a objetos e que, portanto, é do nosso interesse identificar técnicas de VV&T para ADDS desenvolvidos sob este paradigma.

2.3. Critérios de inclusão:

Considerando as questões abordadas nesta revisão sistemática, os critérios de inclusão dos trabalhos foram:

- Questão Primária: Apresentação de técnicas de VV&T utilizadas em Software Distribuído;
- Questão Secundária 1: Avaliação de características específicas de Software Distribuído a serem consideradas em VV&T;
- Questão Secundária 2: Apresentação de técnicas de VV&T utilizadas em Desenvolvimento Distribuído de Software;
- Questão Secundária 3: Apresentação de técnicas de VV&T utilizadas em Ambientes de Desenvolvimento Distribuído de Software OO.

2.4. Estratégia de Busca para Seleção dos Estudos Primários

A estratégia de busca definida levou em conta as fontes utilizadas, os idiomas pesquisados e as palavras-chave que formaram as *strings* de busca, detalhados a seguir:

- Fontes: Máquinas de busca eletrônica (Google, Scirus), bases de dados eletrônicas indexadas (ACM, IEEE), consulta a especialistas e anais de eventos (SBES - Simpósio Brasileiro de Engenharia de Software, ICSE - International Conference on Software Engineering, ISESE - International Symposium on Empirical Software Engineering, etc.).
- Idiomas: Foram considerados os resultados em Inglês, por ser o idioma mais aceito no meio internacional; e em Português, por ser de conhecimento dos revisores a existência de grupos de pesquisa atuantes na área de DDS no Brasil, além de ser a língua mãe dos autores.
- Palavras-chave: *Distributed Software, validation and verification, software testing*. Variações e termos relacionados a estas palavras-chaves são:
  - *Distributed Software: distributed software system, distributed development, distributed development software;*
  - *Verification and Validation: validation, verification, requirement validation, software validation, software verification, VV&T;*
  - *Software testing: testing, testing criterion, testing technique.*

2.5. Seleção Preliminar

Foram geradas *strings* de busca utilizando combinações relevantes das palavras-chave, de acordo com os objetivos da revisão. As *strings* foram então submetidas às máquinas de busca, e os resultados obtidos puderam ser avaliados prosseguindo para a leitura dos resumos dos trabalhos e posterior classificação quanto à inclusão ou não no conjunto de trabalhos posteriormente avaliados.

2.6. Seleção Final e Extração dos Resultados

A seleção final consistiu na leitura total dos trabalhos previamente selecionados. Após a leitura dos trabalhos, o revisor produziu uma síntese destacando as técnicas encontradas e suas principais características.

2.7. Condução da Revisão

A revisão sistemática foi realizada no período de outubro de 2007 a fevereiro de 2008, prosseguindo, então, para a fase de estudo dos trabalhos selecionados e classificação dos mesmos segundo os critérios estabelecidos.

Foi construída uma *string* de busca padrão, utilizando conjunções das palavras-chave definidas e suas variações, como segue:

> *(distributed software OR distributed system OR distributed development OR distributed development software) AND (validation and verification OR validation OR verification OR requirement validation OR software validation OR software verification OR VV&T) AND (software testing OR testing OR testing criterion OR testing technique)*

Devido às especificidades das máquinas de busca utilizadas, foram necessárias adaptações da *string* de busca original, ou partes menores da *string*.

Para a etapa seguinte foram selecionados 12 dos 174 resultados obtidos pela Scirus, cujos assuntos satisfizeram as questões do presente trabalho. Dos 27 resultados da

máquina de busca do portal ACM, apenas 3 satisfizeram as questões, e um destes já havia sido referenciado pelos resultados da Scirus. Dos 10 resultados retornados pela máquina de busca do portal IEEE, 3 foram aceitos nesta revisão, mas já haviam sido referenciados nas máquinas de busca anteriores. As buscas efetuadas no Google, por ser uma máquina com propósito mais genérico, referenciaram mais de 15000 resultados, dentre eles os mesmos resultados obtidos pelas máquinas já citadas, grades curriculares de instituições de ensino, além de resultados diversos cujo assunto não atendia aos critérios da busca.

Assim, de mais de 200 referências encontradas nas máquinas de busca mais específicas, e dos mais de 15000 resultados encontrados no Google, 14 trabalhos foram classificados para a etapa de seleção final da revisão.

Foi efetuada a busca nos anais de eventos, incluindo os eventos ICSE, ISESE e SBES, e também os *workshops* WDDS (*Workshop* de Desenvolvimento Distribuído de Software) e SAST (Brazilian Workshop on Systematic and Automated Software Testing), nos anos de 2004 até 2007, quando disponíveis. Foram encontrados 5 resultados, dos quais 2 satisfaziam os critérios da busca. Dentre os trabalhos recomendados pelos especialistas, 3 foram incluídos.

3. Visão Geral dos Trabalhos Analisados

Os trabalhos recuperados na etapa de busca da RS passaram pela fase de análise, e foram classificados de acordo com os critérios derivados dos objetivos dessa RS. Grande parte desses trabalhos contemplava o uso de *frameworks* de VV&T de software distribuído, baseados na especificação formal dos sistemas.

Foram identificados alguns grupos de pesquisa, atuantes na área de VV&T, que têm envidado esforços no sentido de abordar o uso de métodos e/ou ferramentas de VV&T para software distribuído. Dentre os grupos encontrados, está o grupo IRISA, na França, responsável pelo desenvolvimento das ferramentas TGV e UMLAUT, e do *framework* VALOODS [JERON *et al.* 1998], que figuraram em 3 resultados obtidos. Pesquisadores da *University of Massachusetts Amherst* também têm abordado a área de VV&T, por meio do FLAVERS [CLARK & OSTERWEIL 1998], um sistema de verificação para sistemas seqüenciais ou distribuídos, abordado em 2 trabalhos classificados pela revisão. Outro grupo que também tem manifestado interesse em VV&T para software distribuído é representado pelos desenvolvedores do UniFrame [CAO *et. al.* 2005], um *framework* para a integração de componentes distribuídos, classificado na revisão com 3 artigos. Um grupo brasileiro também foi identificado, com 2 trabalhos referentes à ferramenta FIONA [GERCHMAN *et al.* 2005], desenvolvida para a injeção de falhas de comunicação em aplicações de rede.

A seguir, é apresentada uma descrição dos trabalhos realizados por estes grupos.

3.1. FLAVERS

Desenvolvido no LASER (*Laboratory for Advanced Software Engineering Research, University of Massachusetts Amherst*), FLAVERS – *Flow Analysis for Verification of Systems* – [CLARK & OSTERWEIL 1998] é um sistema utilizado para verificar propriedades especificadas pelo usuário, em programas seqüenciais ou distribuídos.

Usando FLAVERS, o analista precisa primeiro definir um conjunto de eventos

do programa e, em seguida, formular as propriedades a serem verificadas como consequências de tais eventos.

As propriedades são especificadas na forma de expressões regulares quantificadas (ERQ, ou originalmente *Quantified Regular Expression – QRE*). Estas expressões consistem do alfabeto da propriedade, que é o conjunto de eventos referenciados na propriedade, um indicador para definir se a propriedade deve ser mantida em todas ou em nenhuma execução do programa, e uma expressão regular que descreve a seqüência de eventos. Depois que o analista escreve uma propriedade em linguagem ERQ, esta é submetida ao FLAVERS para uma verificação sintática e posterior tradução para o formato de autômato finito determinístico (AFD). O componente de raciocínio FLAVERS utiliza representação de propriedades por AFD, assim, qualquer linguagem de especificação que possa ser traduzida em um AFD poderia ser utilizada para representar propriedades.

Com as propriedades a serem testadas já definidas, o sistema FLAVERS deve verificar se a execução do programa pode, em algum momento, violar alguma das regras definidas. O sistema deve avaliar também se, dada uma propriedade que deve ser mantida em todas as execuções, esta realmente acontece. Esta verificação é efetuada utilizando técnicas de análise de fluxo de dados, podendo gerar resultados conclusivos (mantém ou não mantém a propriedade) ou inconclusivos, os últimos podendo ser causados por falhas na especificação do modelo testado [CLARK & OSTERWEIL 1998].

3.2. UniFrame

Desenvolvido em um trabalho colaborativo entre Indiana University-Purdue University Indianapolis, University of Alabama at Birmingham e Naval Postgraduate School, o UniFrame – *Unified Framework for Seamless Integration of Heterogeneous Distributed Software Components* – [CAO *et. al.* 2005] visa facilitar o agrupamento de componentes heterogêneos, tendo como base a especificação dos meta-modelos dos componentes.

O UniFrame é composto pelos meta-modelos dos componentes, que representam as integrações e restrições dos componentes; pela geração automática de código *Glue/Wrapper* [1], aumentando a interoperabilidade; parâmetros para especificação e verificação de componentes individuais; um mecanismo para descoberta de componentes apropriados em uma rede; uma metodologia para desenvolvimento de sistemas distribuídos baseados em componentes e orientados a serviços; e mecanismos de avaliação do resultado da junção dos componentes.

Os trabalhos analisados, relacionados ao UniFrame, abrangem informações relativas à ferramenta de geração de código *Glue/Wrapper*, com poucas informações relativas aos mecanismos de avaliação dos componentes e/ou do resultado do agrupamento, que são indicadas em alguns estágios das pesquisas do grupo como trabalhos posteriores.

3.3. VALOODS e UMLAUT

Segundo Jeron *et al.* (1998), o desenvolvimento de software OO correto é uma tarefa árdua, devido a características como latência e recuperação de erros, que podem gerar *deadlocks*, condições de disputa, entre outros fatores que dificultam a detecção de erros, situação esta para qual a tecnologia OO não está bem preparada.

[1] Código *Glue/Wrapper* refere-se ao código gerado para agrupar componentes já desenvolvidos.

Apesar do sucesso obtido na utilização de Técnicas de Descrição Formal (TDF), em alguns casos, o custo da aplicação dos mesmos, juntamente com a semântica não tão trivial das especificações, as TDFs não têm sido utilizadas em larga escala.

Neste contexto, pesquisadores do grupo IRISA têm se esforçado para desenvolver um sistema de apoio à validação e geração de testes para software distribuído orientado a objetos – VALOODS, cuja semântica não fosse tão restrita quanto a das TDFs utilizadas.

Uma das ferramentas que contribuem para a validação de software distribuído é a UMLAUT (*Unified Modeling Language All pUrposes Transformer*), propondo-se a utilizar um subconjunto da UML, restringindo-a a fim de conferir-lhe uma sintaxe e semântica formal. Assim, por meio de uma linguagem de modelagem visual, a UML, a ferramenta pode derivar o código de validação escrito em linguagem Eiffel [PICKIN *et al.* 2007].

O *framework* VALOODS consiste de um conjunto de classes e padrões que definem a interação entre os objetos. A adaptação do modelo UML para que este seja compatível com o *framework* é efetuada pela ferramenta UMLAUT. Sobre o modelo adaptado são efetuadas a checagem de modelo, a simulação intensiva e a geração de casos de teste pela ferramenta TGV (*Test Generation with Verification technology)*, também desenvolvida pelo grupo IRISA.

3.4. FIONA

Para o desenvolvimento de sistemas de alta disponibilidade, torna-se necessária a aplicação de métodos que garantam seu funcionamento mesmo em situações adversas, como em falhas na troca de mensagens. Estes sistemas devem prover estratégias de tolerância a falhas, e estas estratégias devem ser corretamente validadas.

A fim de validar as estratégias de tolerância a falhas em sistemas de comunicação, FIONA – *Fault Injection Oriented to Network Applications* – [GERCHMAN *et al.* 2005] visa prover um ambiente distribuído de injeção de falhas em sistemas de rede. FIONA provê escalabilidade para a verificação de sistemas de grande porte, sem a necessidade de efetuar alterações no código da implementação sob teste. Além disso, provê a geração de *logs* da execução dos testes, para análise *post-mortem*.

A injeção de falhas é efetuada por meio de três componentes: o agente JVMTI (*Java Virtual Machine Tools Interface*), o protocolo de comunicação instrumentado e as classes auxiliares de injeção de falhas. A arquitetura de FIONA possibilita a execução dos testes em um ambiente distribuído, sendo que para cada nodo da rede deve haver um injetor local, sendo controlados por um injetor *Site*, que se comunica com um injetor principal. Esta arquitetura facilita a execução dos testes, mesmo em uma configuração na qual existam muitos nodos.

3.5. Outros trabalhos

Além dos trabalhos já citados, outros também foram selecionados para avaliação. Porém, por tratarem de assuntos que divergiam do foco inicial, sendo relativos a sistemas de hardware distribuído, sistemas embarcados ou processamento distribuído, não são detalhados neste artigo. Dentre eles, encontram-se os resultados obtidos por meio de consulta a especialistas, que são relativos a VV&T em processamento paralelo com troca de mensagens. Estas técnicas, apesar de não serem diretamente aplicáveis, talvez possam ser adaptadas para DDS.

Outros 2 artigos e 1 livro que satisfizeram às questões de busca também não são detalhados. Estes trabalhos não estavam disponíveis gratuitamente e quando da seleção preliminar, os *abstracts* não evidenciaram sua real importância para a RS. Portanto, decidiu-se por não efetuar o investimento para sua aquisição naquela ocasião.

4. Análise dos Dados Coletados

De posse dos resultados da revisão, foi possível efetuar a classificação dos resultados segundo os critérios de inclusão definidos no protocolo. A Tabela 1 apresenta a classificação destes resultados, separados por grupos de pesquisa.

Tabela 1. Avaliação dos resultados segundo os critérios da RS

Trabalho	VV&T em SD	Características SD	DDS	OO
FLAVERS	Sim	Não	Não	Não
UniFrame	Sim	Não	Sim	Não
UMLAUT	Sim	Não	Sim	Sim
FIONA	Sim	Tolerância a Falhas	Não	Sim
outros	Sim	Alguns trabalhos contemplam uma ou outra característica	Não	Não

A coluna VV&T em SD refere-se a métodos ou ferramentas de apoio a Verificação, Validação e Teste de Sistemas Distribuídos, não sendo restritos a sistemas de software. As características de sistemas distribuídos contempladas por Coulouris (2005), representadas pela terceira coluna, foram abordadas por um dos grupos de pesquisa considerados, tendo sido também encontradas em trabalhos não incluídos na revisão. Apenas dois grupos consideravam o desenvolvimento distribuído de software, apresentado na quarta coluna da tabela. A última coluna apresenta os resultados que tratavam especificamente sobre estratégias, métodos ou características relacionados a OO.

Durante a fase de leitura completa dos artigos, percebeu-se que em alguns dos resultados obtidos, indicados pela última linha da Tabela 1, os critérios que os incluíram na revisão não eram de fato abordados no corpo do texto. Estes artigos privilegiavam assuntos relacionados a testes de sistemas distribuídos de *hardware*, protocolos de comunicação ou sistemas embarcados, não explorando características de SD ou de DDS.

Os resultados referentes ao sistema FLAVERS, apesar de possibilitarem a verificação de programas seqüenciais ou distribuídos, não apresenta ferramentas específicas para DDS, não aborda especificamente alguma das características de software distribuído, e não aborda especificamente programação OO.

Os resultados encontrados sobre o *framework* UniFrame apresentavam técnicas de desenvolvimento de software distribuído baseado em componentes. Apesar de seus trabalhos não mencionarem especificamente testes, existem indicações da preocupação do grupo com a verificação na parte de integração de componentes.

O Grupo IRISA tem buscado definir métodos e ferramentas de teste para SD, como as ferramentas VALOODS e UMLAUT, que podem ser utilizadas em conjunto com ferramentas de modelagem e teste que exportem arquivos no formato XMI (XML Metadata

Interchange). Não foram identificadas características específicas de software distribuído abordadas pelo grupo.

Os gráficos a seguir apresentam uma classificação dos resultados da RS de acordo com a fonte do trabalho. Na Figura 1 pode-se notar que a maioria dos trabalhos incluídos foi obtida por meio de buscas eletrônicas e dentre eles 71% foi incluído na RS (Figura 2a). Quanto aos trabalhos selecionados a partir de anais de eventos somente 40% foi incluído (Figura 2b), evidenciando que dentre os artigos publicados em eventos, uma quantidade significativa deles não aborda características específicas de SD, DDS ou ADDS.

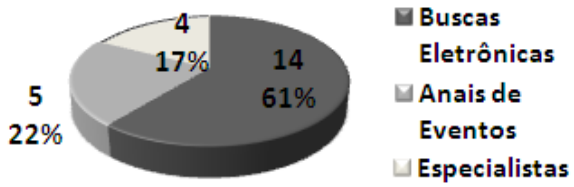

Figura 1. Classificação dos trabalhos incluídos de acordo com a fonte

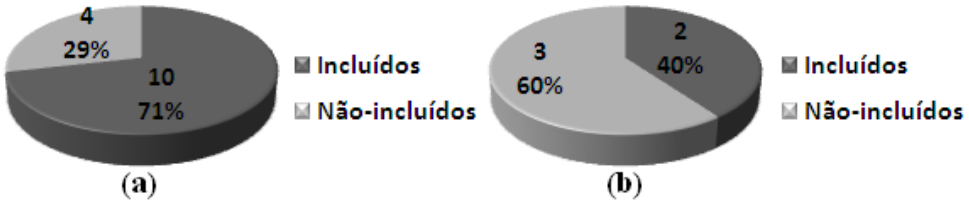

Figura 2. Classificação dos resultados (a) de buscas eletrônicas e (b) de Anais de Eventos

5. Conclusão e Trabalhos Futuros

Esta revisão sistemática buscou identificar técnicas, critérios ou métodos de teste aplicáveis a Desenvolvimento Distribuído de Software, especialmente os métodos aplicáveis a Ambientes de Desenvolvimento Distribuído de Software.

Os resultados da revisão mostraram que, mesmo sendo em estágio inicial, existe a preocupação de alguns grupos de pesquisadores em definir métodos e ferramentas que auxiliem nas atividades de VV&T para SD.

O conjunto de ferramentas desenvolvido pelo grupo IRISA é um dos que mais se destaca, por automatizar grande parte do processo de testes, através de uma formalização da UML, podendo ser utilizada em conjunto com outras ferramentas de modelagem.

Outro resultado a ser destacado é o sistema FIONA, por apresentar uma estratégia de VV&T – injeção de falhas de comunicação – abordando uma das características específicas de SD já citadas: tolerância a falhas.

Os resultados desta revisão sistemática estão sendo utilizados como parte de um Trabalho de Graduação, que tem como objetivos a identificação e possível utilização de técnicas de VV&T em um ADDS, e a busca por ferramentas CASE que automatizem as atividades de VV&T para ambientes distribuídos.

No contexto do grupo de estudos de Engenharia de Software Distribuído do Departamento de Informática da UEM, esta revisão poderá ser utilizada como base para novas

pesquisas na área de VV&T, devendo, se possível, ser repetida por outros pesquisadores para que seus resultados sejam complementados.

Referências

Audy, J. and Prikladnicki, R. (2007). Desenvolvimento Distribuído de Software. Rio de Janeiro: Elsevier.

Cao, F.; Bryant, B. R.; Burt, C. C.; Raje, R. R.; Olson, A. M.; Auguston, M. (2005). A Component Assembly Approach Based On Aspect-Oriented Generative Domain Modeling. In: Proceedings of the Software Composition Workshop (SC 2004). Electronic Notes in Theoretical Computer Science. vol.114. pp 119-136. Barcelona, Espanha.

Clark, L. A.; Osterweil, L. J. (1998) Verifying Properties of Distributed Systems: Prospects for Practicality. Laboratory for Advanced Software Engineering Research (LASER), Department of Computer Science at the University of Massachusetts Amherst, Technical Report.

Coulouris, G. F.; Dollimore, J.; Kindberg, T. (2005). Distributed Systems: Concepts and Design. Addison Wesley.

Ferrari, F. C.; Maldonado, J. C.(2006). Uma Revisão Sistemática sobre Teste de Software Orientado a Aspectos. In: III Workshop Brasileiro de Desenvolvimento de Software Orientado a Aspectos (WASP'2006). XX Simpósio Brasileiro de Engenharia de Software. Florianópolis, SC.

Gerchman, J.; Jacques-Silva, G.; Drebes, R. J.; Weber, T. S.(2005) Ambiente Distribuído de Injeção de Falhas de Comunicação para Teste de Aplicações Java de Rede. In: XIX Simpósio Brasileiro de Engenharia de Software. Uberlândia, MG.

Huzita, E. H. M.; Tait, T. F. C.; Colanzi, T. E.; Quinaia, M. A. (2007) Um Ambiente de Desenvolvimento Distribuído de Software – DiSEN. In: Workshop de Desenvolvimento Distribuído de Software. XXI Simpósio Brasileiro de Engenharia de Software. João Pessoa, PB.

Jeron, T.; Jezequel, J.-M.; Le Guennec, A. (1998) Validation and Test Generation for Object-Oriented Distributed Software. In: International Symposium on Software Engineering for Parallel and Distributed Systems (PDSE 1998). Kyoto, Japão

Kitchenham, B. (2004). Procedures for performing systematic reviews. Joint Technical Report TR/SE-0401 (Keele) – 0400011T.1 (NICTA), Software Engineering Group – Departmen of Computer Science - Keele University and Empirical Software Engineering – National ICT Autralia Ltd, Keele/Sta_s-UK and Eversleigh-Autralia.

Pickin, S.; Jard, C.; Jeron, T.; Jezequel, J.-M.; Le Traon, Y. (2007) Test Synthesis from UML Models of Distributed Software. IEEE Transactions on Software Engineering, vol.33, no.4, pp.252-269. Abril, 2007.

Pressman, R. S. (2002). Engenharia de Software. 5 ed. Rio de Janeiro: McGraw-Hill.

Wiese, I. S.; Amorim, E. F.; Huzita, E. H. M.; Steinmacher, I.; Pascutti, M. C. D.; Pozza, R. (2005). Uma Proposta de Arquitetura para Ambientes de Desenvolvimento Distribuído de Software. In: CACIC 2005 - XI, Congreso Argentino de Ciencias de la Computación, 2005. Workshop de Ingeniería de Software y Bases de Datos (WISBD). Concordia, Entre Ríos, Argentina.