

# **Estudo Empírico Comparativo de Modelos de Componentes para o Desenvolvimento de Software com Suporte à Evolução Dinâmica e não Antecipada**

**Aluna: Nádia Milena Barbosa**

**Orientador: Angelo Perkusich**

Coordenação de Pós-Graduação em Informática – COPIN  
Departamento de Sistemas e Computação – Universidade Federal de Campina Grande  
Av. Aprígio Veloso, 882, Caixa Postal: 10.106, CEP: 58.109-970, Campina Grande-PB

nadia@dsc.ufcg.edu.br, perkusic@dee.ufcg.edu.br

Nível: Mestrado

Ano de Ingresso: 2005

Previsão de Conclusão: Fevereiro de 2007

Aprovação da proposta: 21/03/2006

**Resumo.** *Os sistemas de software atuais apresentam características que exigem cada vez mais do processo de desenvolvimento, tais como: dinamicidade, robustez, possibilidade de mudanças de requisitos e execução ininterrupta. Neste contexto, uma área recente de pesquisa é a evolução dinâmica e não antecipada de software. Existem alguns modelos de componentes com suporte a esta característica. Porém, não há até então uma base de conhecimento capaz de determinar qual desses modelos melhor se adequa a um determinado sistema. Nisto se baseia este trabalho, um estudo empírico comparativo dos modelos de componentes com suporte à evolução dinâmica e não antecipada de software. O estudo será realizado utilizando métricas para capturar informações sobre o sistema em termos dos atributos fundamentais do software. Por meio deste estudo experimental será formada uma base de conhecimento para a escolha do modelo de componentes, de forma adequada aos requisitos do sistema que se pretende desenvolver.*

**Palavras-chave:** *Desenvolvimento baseado em componentes, modelo de componentes, evolução dinâmica de software, métricas.*

# 1. Caracterização do problema

## 1.1. Introdução

O desenvolvimento de software baseado em componentes (DBC) é caracterizado pela integração, composição e adaptação de componentes de software pré-existentes. Esta abordagem enfatiza o reuso e apresenta vantagens no sentido de possibilitar o aumento da produtividade e da qualidade do software [Crnkovic 2001, Szypersky 1998]. O DBC facilita a manutenção dos sistemas por possibilitar que eles sejam atualizados pela integração de novos componentes e/ou substituição dos componentes existentes.

As razões para o interesse por DBC, segundo [Brown and Wallnau 1998], vêm da maturidade das tecnologias que permitem a construção de componentes e a combinação destes para o desenvolvimento de aplicações, bem como o atual contexto organizacional e de negócios que apresenta mudanças em como as aplicações são desenvolvidas, utilizadas e mantidas.

Neste contexto, um tópico recente de pesquisa é a evolução dinâmica e não antecipada de software. A motivação para o estudo nesta área de pesquisa é que os requisitos iniciais para a construção de sistemas são, em geral, modificados e, com isso, os seus componentes também sofrem alterações. Estas mudanças não antecipadas afetam diretamente o projeto existente [Ebraert et al. 2005], sendo uma das principais causas para problemas técnicos e aumento de custos para evolução de software [Kniesel et al. 2002].

Além disso, a evolução dinâmica e não antecipada de software objetiva atender a construção de sistemas que necessitam se adaptar em tempo de execução a novos requisitos de software ou a mudanças dos requisitos existentes. É imprescindível que tais mudanças possam ser realizadas sem grande impacto no projeto do sistema.

Existem vários modelos e tecnologias que oferecem algum tipo de suporte à evolução dinâmica e não antecipada de software, possibilitando desta forma a integração de novos componentes, substituição de componentes existentes e redefinições arquiteturais, originadas das potenciais mudanças de requisitos não previstas no projeto inicial, inclusive em tempo de execução.

Alguns destes modelos de componentes e tecnologias que poderão ser avaliados são: COMPOR [Almeida et al. 2004], modelo de componentes CORBA (CCM) [OMG 1999], EJB [Microsystems 1999], Fractal [Bruneton et al. 2003], .NET [Cazzulino 2005] Gravity [Hall and Cervantes 2003] e Jini [Gupta et al. 2002].

Cada um destes modelos possui características que melhor atendem ao desenvolvimento de software com determinados requisitos. Por exemplo, EJB e .NET têm sido apontados como adequados para aplicações corporativas. CCM, por sua vez, tem sido utilizado para desenvolver sistemas nos quais a heterogeneidade de linguagem é um requisito imprescindível. Sendo assim, a escolha do modelo de componentes mais adequado a um sistema pode ser determinante no processo de desenvolvimento.

Entretanto, na investigação relacionada com componentes tem-se enfatizado, sobretudo, a definição dos seus aspectos funcionais e a composição de componentes na construção de sistemas [Heineman and Council 2001]. A avaliação da qualidade dos componentes e do seu impacto nas arquiteturas em que são integrados tem sido pouco explorada, dificultando a escolha do modelo de componentes mais adequado [Goulão et al. 2002].

## 1.2. Objetivo

O objetivo principal desse trabalho é realizar um estudo empírico comparativo de modelos de componentes para a construção de sistemas de software que suportem evolução dinâmica e não antecipada. A análise será realizada com alguns dos modelos de componentes existentes, tais como: COMPOR, Jini, EJB, .NET, Gravity, Fractal e CCM.

Este estudo será baseado principalmente em métricas como: acoplamento, coesão, tamanho, desempenho.

A partir do objetivo principal são derivados outros secundários. São eles:

- Analisar os modelos de componentes existentes e identificar quais farão parte do estudo. Isso é necessário porque algum deles pode apresentar características que não possibilitem sua análise, tais como: falta de documentação, licenças de uso, dentre outras.
- Definir as métricas para avaliação, possibilitando o estudo empírico comparativo dos modelos de componentes.
- Identificar um sistema que contemple os requisitos que possibilitem a análise através das métricas. Esse sistema deve ser implementado utilizando todos os modelos previamente escolhidos e servirá como estudo de caso para o trabalho.
- Obter os resultados da avaliação empírica.

## 2. Fundamentação teórica

O desenvolvimento de software baseado em componentes é uma abordagem utilizada para estruturar soluções, organizar o desenvolvimento de aplicações e promover a convivência entre tecnologias [Heineman and Council 2001]. Além disso, DBC se preocupa com a criação de componentes que possam ser reutilizados em outras aplicações. Lidar adequadamente com questões como especificação e projeto de unidades de software altamente coesas e fracamente acopladas pode reduzir o custo relacionado ao entendimento dos problemas e do desenvolvimento das soluções. Existem diversas definições para o que vem a ser um componente de software. Neste trabalho, os componentes são tratados como artefatos de software, sendo utilizados como blocos de construção de uma solução. Assim, componentes podem ser definidos como unidades de composição com interfaces especificadas e com dependências explícitas de contexto [Szypersky 1998].

Um modelo de componente define um conjunto de padrões e regras para implementar, interoperar, customizar e implantar componentes. Neste trabalho, a característica utilizada como filtro para escolher os modelos de componentes a serem estudados neste trabalho foi a evolução dinâmica e não antecipada de software. Alguns modelos de composição possuem esta característica que trata das possíveis mudanças de requisitos do sistema. Quanto mais cedo for possível prever uma mudança no sistema, mais facilmente essa adaptação pode ser realizada. Porém, há muitos requisitos que só são notados depois da implementação do sistema, ou há a necessidade de modificar uma funcionalidade. Então uma evolução do sistema pode não ter sido prevista durante o projeto.

Além disso, a evolução dinâmica de software pode tratar da substituição de componentes sem alterar a arquitetura do sistema, que é composta pelas interfaces dos componentes e seus relacionamentos. Modificações deste tipo geralmente são motivadas pela necessidade de adicionar novas funcionalidades a um sistema.

No que diz respeito à forma de comparação entre os diversos modelos de componentes, neste trabalho são utilizadas métricas de software. Métricas têm sido largamente utilizadas em estudos empíricos para capturar informações sobre o sistema em termos dos atributos fundamentais do software.

De acordo com estudos sobre a aplicação de métricas, um processo de medição tem que ser focado em objetivos específicos. A abordagem *Goal-Question-Metric* (GQM) se baseia nesta idéia e apresenta vantagens no sentido de ajudar na identificação de métricas relevantes e apóia a análise dos dados coletados.

A abordagem...

, tais como: desempenho, tamanho, acoplamento e coesão, e assim, indicar os principais aspectos da abordagem estudada [Santanna 2004]. O critério para a escolha dessas métricas, em geral, pode basear-se em demandas teóricas e práticas do processo de desenvolvimento [Chidamber and Kemerer 1994].

### **3. Caracterização da contribuição**

A partir do estudo empírico e das métricas definidas para a realização do estudo proposto, busca-se disponibilizar informações relevantes sobre cada modelo estudado. Desta forma, de acordo com os requisitos do sistema a ser implementado, tem-se um embasamento técnico para escolher qual dos modelos é o mais adequado para um sistema.

Além do resultado prático apresentado, este trabalho servirá como base conceitual para futuros investimentos na área de evolução dinâmica e não antecipada de software, assim como, para o desenvolvimento de aplicações neste contexto.

### **4. Etapas e estado atual do trabalho**

Para a realização do estudo empírico comparativo de modelos de componentes com suporte à evolução dinâmica e não antecipada de software foram definidas as seguintes etapas, a serem realizadas na ordem em que se apresentam.

1. Realização de levantamento bibliográfico sobre desenvolvimento baseado em componentes, modelos de componentes, métricas para avaliação de software e trabalhos relacionados. Nesta etapa busca-se o embasamento teórico necessário sobre cada item citado. Assim como, faz-se necessário averiguar quais modelos de componentes viabilizam o estudo proposto, isto é, se possuem alguns artefatos necessários, tais como: documentação, tutorial e licença de uso.
2. Esta etapa envolve a definição das métricas a serem utilizadas para obter informações sobre o sistema. O critério para a escolha das métricas terá como base demandas teóricas e práticas do processo de desenvolvimento. Através destas métricas serão indicados os principais aspectos da abordagem estudada.
3. Aqui deve ser feita a identificação de um sistema que contenha os requisitos a serem avaliados. Este servirá de estudo de caso no contexto deste trabalho. Em seguida, o sistema escolhido será implementado em cada modelo de componentes que fará parte do estudo e serão aplicadas no sistema as métricas para avaliação dos modelos.
4. A última etapa será a de análise dos dados obtidos e finalização do estudo empírico. Esta análise será feita através de gráficos gerados a partir dos dados obtidos

na etapa anterior. Estes dados servirão de base para a comparação entre os modelos de componentes que farão parte do experimento.

Atualmente o trabalho encontra-se na fase de definição das métricas de software. Ou seja, está sendo realizada uma análise de quais métricas são relevantes ao estudo, podendo assim, gerar dados que viabilizem o estudo empírico comparativo dos modelos de componentes com suporte a evolução dinâmica e não antecipada de software.

## 5. Trabalhos relacionados

Estudos experimentais foram realizados em [Santanna 2004] e foi apresentado um *framework* de avaliação para o desenvolvimento de software orientado a aspectos. Este *framework* é composto por dois elementos: um conjunto de métricas e um modelo de qualidade. O estudo empírico comparou uma abordagem orientada a objetos com uma abordagem orientada a aspectos para o projeto e implementação de um sistema multi-agentes. Em seguida o estudo envolveu a aplicação do framework proposto para avaliar as implementações em Java e AspectJ dos padrões de projeto da GoF.

Dos vários tipo de componentes existentes, [Washizaki et al. 2002] deu ênfase aos componentes de caixa-preta. Foram apresentadas métricas para este tipo de componente utilizando as informações limitadas que podem ser obtidas a partir dos resultados de estudos experimentais sobre os componentes. As métricas podem medir a reusabilidade do componente, bem como, testabilidade e portabilidade.

Em [Martín-Albo et al. 2003], métricas também são usadas, porém, para avaliar a qualidade dos componentes e até de sistemas, através de um modelo chamado *CQM model (Component Quality Model)*. Com isso, pretende-se ajudar na escolha dos diversos componentes existentes em um repositório.

Com a necessidade de melhoria da qualidade e da produtividade dentro das organizações que adotam o desenvolvimento baseado e componentes, em [Gill and Grover 2004] apresentou uma técnica para medir a complexidade de um componente de software baseada no modelo de interface do componente. Isto inclui assinatura de interface, restrições de interface e empacotamento e configuração. Com base nas métricas utilizadas, a complexidade de componentes do software pode ser mantida em limites razoáveis.

## 6. Validação dos resultados

Espera-se validar este trabalho a partir dos resultados dos experimentos realizados. Será possível mostrar numericamente o desempenho de cada modelo de componentes com suporte a evolução dinâmica e não antecipada de software a partir das métricas utilizadas. E assim, a escolha do modelos que melhor se adequem aos requisitos do sistema a ser desenvolvido.

## Referências

Almeida, H. O., Perkusich, A., Paes, R. B., and Costa, E. B. (2004). Composição Dinâmica de Componentes para Aplicações com Mudanças Frequentes de Requisitos.

- In *Anais do IV Workshop de Desenvolvimento Baseado em Componentes*, volume 4, pages 9–14, João Pessoa, PB.
- Brown, A. W. and Wallnau, C. C. (1998). The Current State of CBSE. *IEEE Software*, 15(5):37–46.
- Bruneton, E., Coupaye, T., and Stefani, J. B. (2003). Recursive and Dynamic Software Composition with Sharing. volume 37, pages 8–18.
- Cazzulino, D. (2005). The Code Project. <http://www.codeproject.com/csharp/components.asp>. Acessado em novembro de 2005.
- Chidamber, S. and Kemerer, C. (1994). A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6):476–493.
- Crnkovic, I. (2001). Component-based Software Engineering - New Challenges in Software Development. In *Software Focus*, volume 4, pages 127–133. Wiley.
- Ebraert, P., Vandewoude, Y., D’Hondt, T., and Berbers, Y. (2005). Pitfalls in Unanticipated Dynamic Software Evolution. In *Proceedings of the Workshop on Reflection, AOP and Meta-Data for Software Evolution (RAM-SE’05)*.
- Gill, N. S. and Grover, P. S. (2004). Few Important Considerations for Deriving Interface Complexity Metric for Component-based Systems. *SIGSOFT Softw. Eng. Notes*, 29(2):4–4.
- Goulão, M., B., F., and Abreu (2002). The Quest for Software Components Quality. In *26th International Computer Software and Applications Conference, COMP-SAC’2002*, page 313, Oxford, Inglaterra.
- Gupta, R., Talwar, S., and Agrawa, D. P. (2002). Jini Home Networking: A Step Toward Pervasive Computing. *IEEE Computer*, 35(8):34–40.
- Hall, R. S. and Cervantes, H. (2003). Gravity: Supporting Dynamically Available Services in Client-Side Applications. In *ESEC / SIGSOFT FSE*, pages 379–382.
- Heineman, G. T. and Councill, W. T. (2001). *Component-Based Software Engineering - Putting the Pieces Together*. AddisonWesley.
- Kniesel, G., Noppen, J., Mens, T., and Buckley, J. (2002). First International Workshop on Unanticipated Software Evolution. In *ECOOP2002 Workshop Reader*, volume 2548 of *LNCS*. Springer Verlag.
- Martín-Albo, J., Bertoa, M., Calero, C., Vallecillo, A., Cechich, A., and Piattini, M. (2003). CQM: A Software Component Metric Classification Model. In *7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*.
- Microsystems, S. (1999). Enterprise JavaBeans Specification. <http://java.sun.com/products/ejb>. Acessado em novembro de 2005.
- OMG (1999). CORBA Component Model Joint Revised Submission. <http://www.omg.org>. Acessado em novembro de 2005.
- Santanna, C. N. (2004). Manutenibilidade e Reusabilidade de Software Orientado a Aspectos: Um Framework de Avaliação. Master’s thesis, Pontifícia Universidade Católica do Rio de Janeiro.
- Szypersky, C. (1998). *Component Software, Beyond Object-Oriented Programming*. Addison-Wesley.

Washizaki, H., Yamamoto, H., and Fukazawa, Y. (2002). Software Component Metrics and It's Experimental Evaluation. In *International Symposium on Empirical Software Engineering*, volume 2, pages 19–20.