

Intenção de Doutoramento

de

Miguel Carlos Pacheco Afonso Goulão
(Mestre)

**Engenharia de Software Baseado em Componentes:
uma Abordagem Quantitativa**

Sob a orientação de

Prof. Dr. Fernando Manuel Pereira da Costa Brito e Abreu

Dezembro de 2002

1. Identificação

Nome:	Miguel Carlos Pacheco Afonso Goulão
Categoria:	Assistente do Departamento de Informática
Endereço:	Departamento de Informática Faculdade de Ciências e Tecnologia Universidade Nova de Lisboa 2829-516 Caparica
Email:	Miguel.Goulao@di.fct.unl.pt
Telefone:	212948536 (ext. 10731)
Fax:	212948541
Orientador:	Prof. Dr. Fernando Manuel Pereira da Costa Brito e Abreu (Prof. Auxiliar)
Área científica:	Ciência e Tecnologia da Programação
Sub-áreas envolvidas:	Engenharia de Software, Desenvolvimento Baseado em Componentes, Qualidade de Software

2. Motivação

O desenvolvimento de software baseado em componentes é uma actividade com um peso crescente na indústria de software, de acordo com estudos realizados por diversas organizações, tais como o *Gartner Group*, a *Gica*, a *Ovum* ou a *PriceWaterhouseCoopers* [1, 2].

A principal força justificativa deste crescimento é económica. O desenvolvimento de software baseado em componentes é encarado como uma abordagem que permite simultaneamente reduzir custos no desenvolvimento, encurtar o período necessário ao desenvolvimento do software e aumentar a sua qualidade [3]. As poupanças ao nível dos custos e período de desenvolvimento resultam da reutilização de componentes já existentes. Dado que esses componentes são reutilizados por diversos utilizadores, é expectável que, a sua qualidade seja superior à que seria possível atingir, a um custo semelhante ao da sua aquisição, se um conjunto equivalente de funcionalidades fosse desenvolvido de raiz.

Nem sempre o termo componente é utilizado com significados coincidentes. Frequentemente encontramos referências a componentes comerciais prontos a reutilizar, conhecidos por *Commercial Off-The-Shelf* (COTS). O termo componente também aparece associado a conceitos tais como os de abstracção de desenho, unidade de projecto, item de configuração num sistema de gestão de configurações, ou porção de código fonte passível de ser reutilizado.

No contexto deste trabalho, a definição proposta pelo *Software Engineering Institute* (SEI), parece-nos ser a mais adequada: “um componente de software é um implemento opaco de funcionalidades, sujeito a composição por parte de terceiros e construído de acordo com um determinado modelo de componentes” [4].

A Engenharia de Software Baseado em Componentes (ESBC) estuda as problemáticas envolvidas no desenvolvimento de sistemas de software construídos

com componentes. Para que tal seja possível, tanto os componentes como os modelos de componentes que lhes estão subjacentes devem ser avaliáveis. Pretende-se, em particular, que as propriedades do sistema final possam ser previstas com base nas propriedades dos componentes que nele são compostos.

A investigação relacionada com componentes tem focado, sobretudo, a definição dos seus aspectos funcionais e a composição de componentes na construção de sistemas [5]. Uma outra visão, também importante, tem sido menos explorada: a avaliação da qualidade dos componentes e do seu impacto nas arquitecturas em que são integrados [6]. É nesta área que nos propomos desenvolver a nossa investigação.

É conveniente distinguir entre duas perspectivas fundamentais (e complementares) na investigação e prática de abordagens para o aumento da qualidade do software: uma, mais centrada no **processo** de desenvolvimento, outra na avaliação do **produto**.

A primeira perspectiva baseia-se no seguinte princípio: a qualidade do processo de desenvolvimento determina a qualidade no produto [7]. Modelos de maturidade para o processo de desenvolvimento, tais como o CMM [8, 9], SPICE [10], BOOTSTRAP [11] e o OOSPICE [12] ou normas de qualidade ISO 9000-3 [13] apoiam-se nesse princípio. Destas abordagens, apenas o OOSPICE considera a utilização de componentes. Actividades como a estimação de riscos, planeamento do projecto, garantia da qualidade dos componentes e dos sistemas com eles construídos, ou a avaliação da capacidade dos fornecedores dos componentes, são alguns dos focos do OOSPICE.

O nosso trabalho centrar-se-á na segunda perspectiva. Estamos particularmente interessados na avaliação quantitativa do produto em si, ou seja, dos componentes de software e de arquitecturas neles baseadas. Um tipo de avaliação que pretendemos promover é a da complexidade do componente. No âmbito da gestão de projectos, ela pode ser usada em actividades como a previsão de esforços de construção, evolução ou depuração de componentes, bem como na calendarização desses esforços.

A avaliação de componentes é útil para as organizações envolvidas em actividades fundamentais relacionadas com componentes, como sejam a sua construção e publicação, a pesquisa de componentes, sua adaptação e reutilização. Do ponto de vista dos construtores de componentes, é interessante que essa avaliação possa de algum modo premiar os bons componentes, de acordo com critérios como a sua fiabilidade, facilidade de reutilização (que poderá estar associada a contratos bem definidos, tornando o componente mais compreensível para os utilizadores) ou riqueza de funcionalidades, por exemplo. A pesquisa de componentes beneficia da avaliação dos mesmos na medida em que esta pode auxiliar a escolher entre componentes alternativos provenientes de múltiplas fontes. A facilidade de adaptação e reutilização de componentes poderá ser avaliada com base na complexidade dos contratos associados aos componentes.

3. Objectivos

A Engenharia de Software é tradicionalmente qualitativa, tornando-se complicado verificar até que ponto os benefícios reivindicados pelos defensores das diversas práticas, modelos e teorias são efectivos. Workshops tais como o QUAOOSE,

4 Engenharia de Software Baseado em Componentes: uma Abordagem Quantitativa

realizado no âmbito da ECOOP, ou outros realizados com objectivos semelhantes em conferências como a OOPSLA têm servido como um fórum privilegiado para a divulgação e debate de abordagens quantitativas à Engenharia de Software Baseado em Objectos (ESBO) [14]. A nossa actividade de investigação ao longo dos últimos anos tem sido dedicada precisamente a este tipo de abordagens (mais detalhes sobre a actividade do grupo de investigação a que pertencemos podem ser consultados em <http://ctp.di.fct.unl.pt/OUASAR/Resources/publications.htm>). Tal como acontece na ESBO, também na ESBC são necessários estudos quantitativos, quer sobre componentes, quer sobre arquitecturas neles baseadas. Ao longo deste trabalho de doutoramento, pretendemos contribuir para o desenvolvimento de métodos, técnicas e ferramentas que suportem essa actividade. Em particular, vamos explorar os seguintes veios de investigação:

- i. Definição de uma representação de arquitecturas baseadas em componentes que capture os aspectos fundamentais dos actuais modelos (quer os de cariz académico, quer os usados em abordagens industriais). São exemplos de modelos de componentes o Java-RMI [15], o CORBA [16], o EJB [17] e o DCOM [18]. Ao modelo dos conceitos utilizados num modelo de componentes, chama-se “meta-modelo” de componentes.
- ii. Para modelar os conceitos utilizados no conjunto de meta-modelos que iremos abordar, é necessário criar um meta-meta-modelo de componentes.
- iii. Definição formal de métricas para avaliação de componentes sob um ponto de vista estrutural, com base em instanciações do meta-meta-modelo.
- iv. Definição formal de métricas para a avaliação de arquitecturas baseadas em componentes com base em instanciações do meta-meta-modelo.
- v. Definição de um modelo para a qualidade de software baseado em componentes e dos próprios componentes de software. Esse modelo utilizará diferentes pesos para os diversos atributos que vier a considerar. Esses pesos serão calibrados com base em experiências empíricas e ainda heurísticas sobre a qualidade do desenvolvimento baseado em componentes. A validação deste modelo deverá ser feita com recurso a experiências controladas.
- vi. Pretendemos que o modelo a produzir inclua informação que permita a detecção de insuficiências de desenho, com base no modelo anteriormente referido, bem como a produção de recomendações para a melhoria dos componentes e sistemas baseados em componentes a avaliar.
- vii. Construção de ferramentas integradas de suporte à recolha de métricas e produção de resultados com base na sua análise.

Os objectivos dos veios de investigação referidos visam criar as condições que para seja possível avaliar componentes e arquitecturas de um modo quantitativo, sistematizado, formal e repetível. Ou seja, diferentes equipas de avaliação deverão chegar aos mesmos resultados quando avaliam o mesmo componente ou arquitectura à luz de um mesmo modelo teórico. Este tipo de avaliação tem várias aplicações, tais como: servir de base para a comparação de soluções alternativas; facilitar a identificação de eventuais falhas de desenho; pode ser usado na geração automática de recomendações para um aumento da qualidade.

Pretendemos validar as propostas apresentadas, quer quanto às métricas definidas, quer quanto à avaliação de componentes e arquitecturas, usando como referência o modelo de qualidade desenvolvido. Essa validação será realizada através de experiências controladas. O desenvolvimento das ferramentas referidas no ponto vii) visa suportar a actividade de experimentação.

4. Definição e delimitação do objecto de estudo

A Figura 1 é uma representação abstracta das interacções entre as actividades necessárias para atingir os objectivos propostos na secção 3.

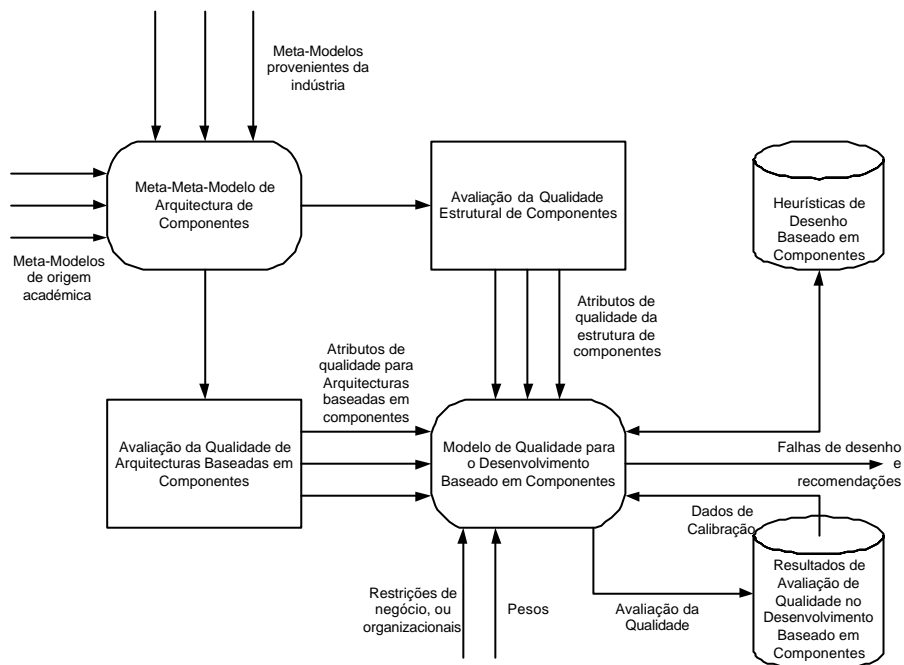


Figura 1 – Avaliação de componentes

Um primeiro aspecto a considerar é a representação do próprio componente. Consideramos que a utilização de um meta-meta-modelo de componentes é a forma mais adequada para definir essa representação. O meta-meta-modelo para arquitecturas de componentes constitui uma representação comum para os aspectos dessas arquitecturas que sejam considerados relevantes para a sua avaliação. Portanto, podemos usar este nível para efectuar avaliações comparativas de forma homogénea.

Esta opção resulta da nossa experiência prévia com a avaliação da qualidade do desenho de software orientado a objectos, durante a qual observamos a utilidade de

encontrar um formato de representação desse desenho independente das linguagens de programação ou desenho usadas [19]. No caso do desenho de software orientado a objectos, a UML [20] aparece como uma norma *de facto* e *de jure*, ainda que o aparecimento de propostas tais como a OML [21] procurem constituir alternativas a limitações encontradas na UML. O meta-modelo da UML constitui uma excelente base para a recolha de informação usada posteriormente na avaliação quantitativa do desenho de software orientado a objectos [22]. O desenho a avaliar pode ser obtido a partir de um qualquer formalismo de desenho OO, ou através de engenharia reversa, a partir do código fonte, e mapeado para UML, sendo depois avaliado nesta forma.

A linguagem usada pelo OMG (*Object Management Group*) na definição do meta-modelo da UML é o MOF (*Meta-Object Facility*) [23]. O meta-meta-modelo MOF é uma linguagem abstracta que permite especificar meta-modelos, tais como os da UML e o do próprio MOF. Actualmente, existe um perfil de UML para MOF que permite que a definição de meta-modelos com o MOF possa ser feita recorrendo à notação e ferramentas associadas à UML. Um dos objectivos do OMG é a fusão do meta-modelo da UML com o meta-meta-modelo do MOF, conseguindo-se assim definir uma linguagem à custa dela própria. Na prática, isso significa que deixará de ser necessário “subir” mais um nível meta para descrever a UML, ou seja, que a UML passará a ser definida em UML.

No âmbito de uma iniciativa desenvolvida com o objectivo de conferir precisão à UML, foi proposta uma alternativa ao MOF, denominada MMF (*Meta-Modeling Facility*) [24]. A MMF é constituída por uma linguagem para meta-modelação – MML (*Meta-Modeling Language*) – e ferramentas de suporte a essa linguagem, tais como verificadores de satisfação de um modelo face ao respectivo meta-modelo e geradores de instâncias de meta-modelos. A MML suporta formas de representação de estruturas de objectos e construções para exprimir restrições sobre essas estruturas e assim garantir que elas são bem formadas. Além disso, também suporta construções para o empacotamento e composição de fragmentos da definição de uma linguagem, bem como de reflexividade. A reflexividade permite que uma entidade possa ser encarada como modelo e como instância, dependendo do ponto de vista. Por exemplo, no meta-modelo da UML, *Association* é uma classe a partir da qual se podem criar instâncias. Simultaneamente, a mesma *Association* é uma instância da construção *Class* ao nível do meta-meta-modelo.

Para a construção do meta-meta-modelo, estudaremos as alternativas MOF e MMF. A primeira tem a seu favor o facto de ser uma norma do OMG, o que poderá alargar o potencial público alvo do meta-meta-modelo. A opção pela MML poderá revelar-se tecnicamente mais aliciante, pelas vantagens da formalização que a acompanha. Em qualquer dos casos, a notação gráfica a usar será a da UML, enriquecida com expressões OCL [25], a linguagem formal que acompanha a UML.

Considere-se agora o problema da representação de arquitecturas de componentes. A Figura 2 apresenta a arquitectura de meta-níveis que pretendemos usar para a resolução deste problema. No nível mais baixo, temos as instâncias de componentes. No nível imediatamente acima, é feita a definição dos componentes. No terceiro nível, apresenta-se o modelo de componentes segundo o qual os componentes foram definidos. A este nível, podem-se colocar os modelos de componentes, em que se descreve a forma como os componentes se integram para formar um sistema baseado em componentes. É portanto aqui que se descrevem as tecnologias que permitem a

representação de componentes, bem como a sua composição, tais como o JavaBeans, o COM e o modelo de componentes do CORBA. Quando pretendemos representar estes diferentes modelos com uma linguagem comum, subimos ao meta-modelo de componentes. Se pretendemos generalizar os conceitos representados ao nível de um meta-modelo de componentes, chegamos finalmente ao meta-meta-modelo de componentes, que pretendemos desenvolver para depois, a exemplo do que se descreveu atrás para a avaliação de desenho OO, podermos usar esse meta-meta-modelo como base para a definição e recolha de métricas sobre componentes e arquitecturas baseadas em componentes.

	Conceitos Genéricos	Representação Genérica
Meta-Meta-Modelo de Componentes	Sim	Sim
Meta-Modelo de Componentes	Não	Sim
Modelo de Componentes	Não	Não
Componentes		
Instâncias de Componentes		

Figura 2 – Arquitectura de meta-modelos, por camadas

Com base no meta-meta-modelo, será possível navegar através da estrutura de componentes e avaliar essa mesma estrutura usando métricas adequadas, ainda por definir. O mesmo se pode dizer acerca de arquitecturas baseadas em componentes. As métricas serão expressas recorrendo à OCL. A solução técnica a usar na recolha de informação em meta-modelos para a avaliação de aspectos da qualidade, quer de componentes de software, quer de arquitecturas de componentes surge como uma extensão a um novo domínio do trabalho previamente desenvolvido, com sucesso, no contexto da avaliação do desenho de software orientado a objectos [26-28].

A avaliação de componentes e suas arquitecturas, usando métricas adequadas, deverá ter como referência um modelo de qualidade. Existem na literatura diversos modelos de qualidade de software. Uma abordagem, frequentemente explorada na criação de modelos de qualidade, passa pelo estabelecimento de uma visão hierarquizada desses modelos, em que se define um conjunto de características de qualidade que, por sua vez, podem ser decompostas em sub-características, ou atributos. Cabem dentro desta categoria modelos de qualidade tais como o ISO9126 [29], o COCOMO II, os modelos de Arthur [30], Grady [31], McCall [32], Meyer [33] e Abreu [26].

Os modelos de qualidade genéricos, aqui referidos, não são totalmente adequados à avaliação de componentes ou sistemas neles baseados. Necessitamos de modelos de qualidade desenvolvidos especificamente para componentes. Aspectos como a selecção de um conjunto de atributos de qualidade relevantes, a sua avaliação e pesagem na tomada de decisões, estão em aberto [34]. Note-se que os componentes são normalmente disponibilizados sob a forma de uma caixa negra, o que retira a visibilidade de alguns dos seus atributos aos utilizadores dos componentes. Essa restrição condiciona o tipo de avaliação que se pode efectuar sobre os componentes. A avaliação a efectuar incide, portanto, sobre os conectores disponibilizados pelos

8 Engenharia de Software Baseado em Componentes: uma Abordagem Quantitativa

componentes e sobre o contrato a que o componente obedece. Uma discussão interessante sobre estes problemas pode ser encontrada em [35]. Este tipo de problemas afecta outros tipos de avaliação, além da avaliação da qualidade. Podemos encontrar um exemplo disso na estimação de custos no processo de software com o COCOTS [36]. Este modelo de estimação de custos foi desenvolvido como uma extensão do COCOMO-II [37] para contemplar os aspectos relacionados directamente com o desenvolvimento baseado em componentes.

Uma crítica por vezes apontada aos modelos de qualidade hierárquicos relaciona-se com a ausência de uma especificação de relações de dependência entre diferentes atributos de qualidade. O aumento da qualidade de acordo com um determinado atributo de qualidade é por vezes conseguido sacrificando outros atributos. Por exemplo, um reforço de aspectos relacionados com a segurança de um determinado componente poderá ser acompanhado de uma perda de eficiência desse mesmo componente.

Os modelos relacionais de qualidade procuram capturar este tipo de interacções [38]. No entanto, também estes modelos são insuficientes para capturar relações mais subtis que envolvam mais do que dois atributos de qualidade em simultâneo.

Quer se tratem de modelos hierárquicos ou relacionais, um problema comum nos modelos de qualidade genéricos é a dificuldade de avaliação do impacto real de cada artefacto de software (ou seja, cada entregável do processo de software) na qualidade do sistema de software de que faz parte. Para lidar com este problema, Zhu defende que um modelo de qualidade deve [39]:

- associar explicitamente atributos de qualidade aos artefactos do sistema;
- associar propriedades abstractas a fenómenos observáveis e verificáveis no sistema e seus artefactos;
- incluir a justificação que está na base da criação de relações entre os diversos atributos. Esta justificação pode ser específica a um determinado sistema e deverá ser verificável e validável no contexto desse sistema.

Zhu avançou com uma notação gráfica para a representação de modelos que apresentem estas propriedades. Essa notação utiliza grafos em que os nós são artefactos de software, aos quais estão associados o atributo de qualidade que se pretende analisar e uma descrição do fenómeno observado. Os ramos dos mesmos grupos representam as relações de dependência entre os artefactos, podendo ser etiquetadas com a respectiva descrição.

Este tipo de abordagem conduz a modelos de qualidade específicos. Estes modelos apresentam vantagens no que toca à avaliação de um sistema em particular, embora à custa da necessidade de repetir o esforço de construção do modelo para novos sistemas, o que pode levantar problemas quanto à generalização deste tipo de solução, pelo menos tal como apresentada por Zhu. Não obstante, este tipo de abordagem parece-nos promissor. Assim, procuraremos desenvolver uma evolução desta abordagem que resolva o referido problema da generalização. Em termos mais genéricos, pretendemos construir um modelo de qualidade centrado no produto que contemple respostas para as seguintes questões :

- Que atributos devem ser medidos?
- Como devem ser medidos?
- Quem deve medir esses atributos?
- Quando devem ser medidos?
- Como devem ser pesados, num processo de tomada de decisões ?
- Como garantir a generalidade do modelo para que permita comparações entre componentes ou sistemas baseados em componentes distintos e como mapear esse modelo a componentes e sistemas concretos?

Com base no modelo de qualidade a definir, pretendemos constituir um repositório de dados recolhidos durante as avaliações. Esses dados ficam disponíveis para calibração do modelo, abrindo-se assim caminho a tarefas como a detecção automática de potenciais falhas no desenho de sistemas baseados em componentes, ou a produção de recomendações com vista à melhoria desses sistemas. Para tal, o modelo deverá ser também enriquecido com heurísticas sobre o desenho.

5. Metodologia

O trabalho começará por uma revisão do estado da arte em áreas como os modelos de qualidade para componentes e os meta-modelos de arquitecturas de componentes .

A vertente teórica da nossa investigação envolve, nomeadamente, a construção do meta-meta-modelo de arquitecturas de componentes, dos conjuntos de métricas e do modelo de qualidade para software baseado em componentes. Esta actividade constitui um ponto de contacto com o trabalho a desenvolver pela nossa colega Aline Lúcia Baroni, do Grupo QUASAR. No contexto do seu trabalho de doutoramento, a nossa colega desenvolverá métricas para a avaliação da modelação dinâmica de sistemas usando a UML. Consideramos fundamental que as propostas efectuadas no âmbito deste trabalho sejam validadas experimentalmente e pretendemos criar as condições para que essa validação possa ser levada a cabo quer por nós quer por outros, através da disponibilização de ferramentas necessárias que vierem a ser construídas.

A participação em eventos de carácter científico tais como conferências, *workshops*, simpósios, palestras e cursos relacionados com os temas a explorar ao longo do trabalho de doutoramento irá certamente assumir um papel muito relevante para a construção e validação de propostas no contexto deste trabalho.

Ainda tendo em conta os benefícios esperados do intercâmbio de experiências com outros investigadores, pretendemos ao longo deste trabalho estreitar laços com vários grupos prestigiados de investigação em engenharia de software experimental, sediados, nomeadamente com os quais existem já alguns contactos, em instituições tais como o grupo do Dr. Peter Kaiser no Institut Experimentelles Software Engineering do Fraunhofer Institute (<http://www.iese.fhg.de/>), na Alemanha, o Grupo Allarcos do Prof. Mario Piattini da Universidade de Castilla-La Mancha (<http://alarcos.inf-cr.uclm.es/>), em Espanha, o grupo do Prof. Houari Sahraoui da Universidade de Montreal, no CRIM (<http://www.crim.ca/>), Canadá, ou o Grupo do Prof. Victor Basili, da Universidade de Maryland, Software Engineering Lab

(<http://sel.gsfc.nasa.gov/>), nos Estados Unidos. A recente criação de redes de apoio ao desenvolvimento da engenharia de software experimental, como a ESERNET (<http://www.esernet.org/>) ou a CeBASE (<http://www.cebase.org/>) ilustram bem o interesse de várias organizações em combinar esforços para, por exemplo, construir repositórios de dados experimentais que possam ser partilhados por diferentes equipas de investigação afiliadas.

6. Plano de Actividades

O plano de trabalho proposto está dividido em 4 fases. Na primeira fase, pretendemos estabelecer as fundações para a realização de todo o estudo subsequente, com a criação de uma base de trabalho consubstanciada no meta-meta-modelo de arquitecturas de componentes. Na segunda fase, exploraremos o referido meta-meta-modelo com a criação de métricas e ferramentas para a avaliação quantitativa de componentes e arquitecturas de componentes. Na terceira fase desenvolveremos um modelo de qualidade que sirva de enquadramento para a avaliação de componentes e respectivas arquitecturas. Finalmente, na quarta fase, apresentaremos os resultados finais do projecto, com a redacção da dissertação.

Neste plano de trabalhos, está sempre presente a preocupação de participar activamente em eventos científicos por forma a melhor validar as nossas propostas.

O trabalho descrito nas secções anteriores deverá ser conduzido de acordo com o plano de actividades descrito na seguinte tabela:

Calendário	Actividades
1º ano	<ul style="list-style-type: none"> • Revisão bibliográfica sobre modelos de qualidade para componentes. • Revisão bibliográfica sobre meta-modelos de arquitecturas de componentes. • Revisão bibliográfica sobre a construção de meta-meta-modelos. • Proposta de um meta-meta-modelo para arquitecturas de componentes. • Redacção de um artigo descrevendo o meta-meta-modelo, para apresentação e publicação em conferência internacional. • Redacção de um artigo de síntese sobre a construção do meta-meta-modelo e mapeamento de meta-modelos de componentes existentes para publicação em revista científica. • Desenvolvimento de métricas para a avaliação estrutural de componentes.
2º ano	<ul style="list-style-type: none"> • Desenvolvimento de métricas para a avaliação de arquitecturas de componentes. • Desenvolvimento do suporte computacional para a recolha das métricas com base no meta-meta-modelo. • Concepção e realização de experiências empíricas de validação das métricas para a avaliação estrutural de componentes. • Concepção e realização de experiências empíricas de validação

	<p>das métricas para a avaliação de arquitecturas de componentes.</p> <ul style="list-style-type: none"> • Redacção de um artigo com os resultados das experiências realizadas e a descrição das ferramentas nelas usadas, para publicação em conferência internacional. • Redacção de um artigo de síntese da concepção e validação de métricas, quer para componentes quer para as suas arquitecturas, para publicação em revista científica. • Construção de um modelo de qualidade para o desenvolvimento baseado em componentes.
3º ano	<ul style="list-style-type: none"> • Realização de uma experiência de validação do modelo desenvolvido. • Redacção de um artigo em conferência internacional descrevendo a construção do modelo de qualidade. • Redacção de um artigo para publicação em revista científica, com a descrição da integração dos diversos aspectos da abordagem para a avaliação de componentes e arquitecturas proposta. • Análise final dos resultados. • Escrita e submissão da dissertação

7. Enquadramento

A linha de investigação relacionada com a ESBC está enquadrada na estratégia do Grupo QUASAR (*QU*antitative *A*pproaches on *S*oftware *E*ngineering And *R*eengineering). O QUASAR integra-se na linha de acção de Engenharia de Software do CITI (Centro de Informática e Tecnologias de Informação – <http://www.di.fct.unl.pt/citi/>) do Departamento de Informática da Faculdade de Ciências e Tecnologia. A actividade de investigação do QUASAR inclui actualmente as seguintes linhas de investigação:

- formalização de métricas para o desenho orientado a objectos;
- avaliação quantitativa de arquitecturas de componentes;
- modelos de estimação para o desenvolvimento de software orientado a objectos;
- reengenharia de sistemas legados;
- engenharia de software automatizada.

Informação mais detalhada sobre as actividades de investigação do QUASAR pode ser consultada online em <http://ctp.di.fct.unl.pt/QUASAR/> .

8. Referências

- [1] L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord e K. Wallnau, “Volume I: Market Assessment of Component-Based Software Engineering”, Software Engineering Institute, Nota técnica CMU/SEI-2001-TN-007, Maio, 2001.
- [2] J. D. Williams, “Raising Components”, *Application Development Trends*, vol. 7, 2000.
- [3] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*. New York: ACM Press / Addison-Wesley, 1998.
- [4] F. Bachman, L. Bass, C. Buhman, S. Cornella-Dorda, F. Long, J. Robert, R. Seacord, e K. Wallnau, “Volume II: Technical Concepts of Component-Based Software Engineering”, Software Engineering Institute, Relatório Técnico CMU/SEI-2000-TR-008, Maio, 2000.
- [5] G. T. Heineman e W. T. Councill, *Component-Based Software Engineering - Putting the Pieces Together*. Boston, MA: Addison-Wesley, 2001.
- [6] M. Goulão e F. B. e Abreu, “The Quest for Software Components Quality”, 26th International Computer Software and Applications Conference, COMPSAC'2002, Oxford, Inglaterra, 2002.
- [7] W. Humphrey, *Managing the Software Process*: Addison-Wesley Publishing Company, 1989.
- [8] M. C. Paulk, B. Curtis, M. B. Chrissis e C. V. Weber, “Capability Maturity Model (Version 1.1)”, *IEEE Software*, 1993.
- [9] M. Paulk, S. Garcia, e M. B. Chrissis, “An Architecture for CMM Version 2”, SEPG'96, 1996.
- [10] M. Konrad, M. Paulk e A. Graydon, “An Overview of SPICE's Model for Process Management”, Fifth International Conference on Software Quality, Austin, Texas, EUA, 1995.
- [11] G. Koch, *Process Assessment: The BOOTSTRAP Approach*: Butterworth-Heinemann Ltd, 1993.
- [12] F. Stallinger, A. Dorling, T. Rout, B. Henderson-Sellers e B. Lefever, “Software Process Improvement for Component-Based Software Engineering: An Introduction to the OOSPICE Project”, 28th EUROMICRO Conference, Dortmund, Alemanha, 2002.
- [13] ISO9000-3, “Quality Management and Quality Assurance Standards. Part 3. Guidelines for the Application of ISO9001 to the Development, Supply and Maintenance of Software”, International Standards Organization, 1991.
- [14] F. B. e Abreu, H. Zuse, H. A. Sahraoui e W. Melo, “ECOOP Workshop on Quantitative Approaches in OO Software Engineering”, *ECOOP'99 Workshop Reader, Lecture Notes in Computer Science*, A. Moreira, Ed.: Springer-Verlag, 1999.
- [15] Sun, “Java Remote Method Invocation Specification”, Sun Microsystems, Inc. Revision 1.8, Java 2 SDK, Standard Edition v1.4, 2002.
- [16] OMG, “CORBA Components - Volume I”, 1999.
- [17] V. Matena e M. Hapner, “Enterprise JavaBeans Specification 1.1,” Sun Microsystems, Inc. 1999.
- [18] Microsoft, “DCOM Technical Overview”, Microsoft Corporation 1996.

- [19] F. B. e Abreu, L. M. Ochoa, e M. Goulão, “The GOODLY Design Language for MOOD2 Metrics Collection”, ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QUAOOSE'1999), Lisboa, Portugal, 1999.
- [20] OMG, “UML Notation Guide (version 1.4)”, Rational Software Corporation, Ed. Menlo Park, CA, EUA: Object Management Group, 2001.
- [21] D. Firesmith, B. Henderson-Sellers, e I. Graham, *OPEN Modeling Language (OML) Reference Manual*: Cambridge University Press, 1998.
- [22] A. L. Baroni e F. B. e Abreu, “Formalizing Object-Oriented Design Metrics upon the UML Meta-Model”, Brazilian Symposium on Software Engineering, Gramado - RS, Brasil, 2002.
- [23] OMG, “Meta Object Facility (MOF) Specification (Version 1.4)”, Object Management Group, Abril 2002.
- [24] T. Clark, A. Evans, S. Kent, S. Brodsky, e S. Cook, “A Feasibility Study in Rearchitecting UML as a Family of Languages using a Precise OO Meta-Modeling Approach (version 1.0)”, pUML Group / IBM. Setembro, 2000.
- [25] OMG, “Object Constraint Language Specification (version 1.1)”, Rational Software Corporation, Ed. Menlo Park, CA, EUA: Object Management Group, 1997.
- [26] F. B. e Abreu, “Engenharia de Software Orientado a Objectos: uma Aproximação Quantitativa”: Tese de Doutoramento, IST, 2001.
- [27] A. L. Baroni, S. Braz e F. B. e Abreu, “Using OCL to Formalize Object-Oriented Design Metrics Definitions”, ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QUAOOSE'2002), Málaga, Espanha, 2002.
- [28] A. L. Baroni, “Formal Definition of Object-Oriented Design Metrics”, : Vrije Universiteit Brussel - Bélgica, em colaboração com a École des Mines de Nantes - França e a Universidade Nova de Lisboa - Portugal, 2002.
- [29] ISO9126, “Information Technology - Software Product Evaluation - Software Quality Characteristics and Metrics”. International Organization for Standardization.
- [30] L. J. Arthur, *Measuring Programmer Productivity and Software Quality*: Wiley-Interscience, 1985.
- [31] R. B. Grady e D. L. Caswell, *Software Metrics: Establishing a Company-Wide Program*. Englewood Cliffs, NJ, EUA: Prentice-Hall, 1987.
- [32] J. McCall, “Quality Factors”, *Encyclopedia of Software Engineering*, vol. I+II, J. J. Marciniak, Ed.: John Wiley & Sons, 1994.
- [33] B. Meyer, *Object-Oriented Software Construction*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1997.
- [34] M. Goulão e F. B. e Abreu, “Towards a Components Quality Model”, Work in Progress Session of the 28th Euromicro Conference - Euromicro 2002, Dortmund, Alemanha, 2002.
- [35] M. Bertoa e A. Vallecillo, “Quality Attributes for COTS Components”, ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QUAOOSE'2002), Málaga, Espanha, 2002.

- [36] C. Abts, B. Boehm e E. Bailey, “COCOTS: A COTS Software Integration Lifecycle Cost Model - Model Overview and Preliminary Data Collecting Findings”, Relatório Técnico, University of Southern California, EUA, 2000.
- [37] B. W. Boehm, B. K. Clark, E. Horowitz, R. Maduchy, R. Selby, e C. Westland, “An overview of the COCOMO 2.0 software cost model”, Relatório Técnico, University of Southern California, EUA, 1995.
- [38] R. G. Crespo, “Seeking Quality in Software Systems Through the Properties of Software Development”, ERCIM Workshop, Pisa, Itália, 1992.
- [39] H. Zhu, Y. Zhang, Q. Huo, e S. Greenwood, “Application of Hazzard Analysis to Software Quality Modelling”, 26th Annual International Computer Software and Applications Conference, COMPSAC'2002, Oxford, Inglaterra, 2002.