

Engenharia de software baseado em componentes: uma abordagem quantitativa

(resumo de investigação – Março, 2004)

Miguel Goulão

Área de investigação

Engenharia de software empírica, aplicada ao desenvolvimento baseado em componentes.

Nota prévia

Por questões relacionadas com a organização da exposição do resumo de investigação, optamos por não seguir à risca o conjunto de tópicos proposto no apelo a comunicações enquanto estrutura de documento, embora tenhamos tido o cuidado de procurar incluir todos os tópicos solicitados neste documento. Por forma a facilitar uma mais fácil localização desses tópicos no documento, apresentamos aqui uma breve lista comentada das secções onde se pode encontrar informação relativa a esses tópicos.

- identificação e relevância do problema técnico que pretende resolver – secção 1;
- ideia fundamental que a dissertação pretende veicular – secção 2;
- identificação das razões porque a investigação conduzida por outros investigadores não resolveu ainda esse problema – nas secções 3.1 a 3.6, as insuficiências do estado da arte vão sendo apresentadas à medida que os vários aspectos da arquitectura da solução proposta vão sendo introduzidos
- detalhes sobre a arquitectura da solução proposta – secção 3;
- outras contribuições científicas que a dissertação poderá vir a conter, para além da resolução do problema identificado – no final das secções 3.1, 3.2 e 3.7, identificam-se algumas contribuições adicionais, já concretizadas nuns casos e estimadas noutros;
- metodologia de investigação adoptada para conduzir a investigação, nomeadamente no que toca à validação dos resultados – secção 4;
- plano de trabalho, evidenciando a estratégia adoptada para a divulgação dos resultados à comunidade científica – secção 5.

1 Motivação

A Engenharia de Software Baseado em Componentes (ESBC) tem como objectivos fundamentais o desenvolvimento e melhoria de práticas relacionadas com o Desenvolvimento Baseado em Componentes (DBC). Entre outros desideratos, pretende-se que a ESBC possa dotar as organizações produtoras de software com a capacidade de previsão das características finais do software a desenvolver com base nas características dos diversos componentes de software que nele são integrados. Para que tal seja possível, é

necessário desenvolver mecanismos de avaliação, quer de componentes de software, quer de infraestruturas com eles construídas.

De acordo com diversos estudos independentes referidos em [1, 2], o DBC tem vindo a adquirir um peso crescente na indústria de software, impulsionado sobretudo por motivações de índole económica. O DBC é visto como uma abordagem à construção de sistemas de software que permite obter reduções quer nos custos de desenvolvimento de software, quer no tempo necessário a esse desenvolvimento e melhorar a qualidade do software desenvolvido [3]. Estas alegadas melhorias resultam, sobretudo, da reutilização de componentes. Numa primeira análise, reutilizar componentes bem testados é mais eficiente, quer do ponto de vista do esforço, quer do tempo, do que construir de raiz software com uma funcionalidade semelhante. Além disso, o esforço despendido no ensaio de um componente acaba por ser mitigado com as suas sucessivas reutilizações, porventura por diferentes utilizadores. Isso permite que a sua qualidade seja superior à que se conseguiria obter desenvolvendo software de raiz com um orçamento de valor semelhante ao gasto na aquisição do componente.

Não nos devemos deter nesta visão algo idealista e simplificada sobre a reutilização de componentes. O seu processo de selecção levanta uma série de problemas: tipicamente, os componentes reutilizáveis são construídos de modo a satisfazer uma necessidade do mercado, tal como ela é identificada pelos produtores do componente. Cabe depois aos produtores de software que integra componentes proceder à selecção dos componentes mais adequados às suas necessidades específicas.

Desde logo, é muito natural que exista algum desencontro entre as características específicas procuradas por quem selecciona os componentes e as características que estes oferecem [4]. O problema coloca-se não apenas a um nível funcional, mas também a um nível extra-funcional. A investigação relacionada com componentes tem focado, sobretudo, o primeiro lado da questão, ou seja, os problemas relacionados com a definição dos aspectos funcionais e a composição de componentes [5]. No entanto, observa-se um crescente interesse pela segunda vertente do problema: a avaliação de características extra-funcionais de componentes e infra-estruturas de componentes. Em particular, a qualidade de componentes e infra-estruturas de componentes, bem como o impacto dessa qualidade na qualidade global dos sistemas em que eles se integram, são tópicos de investigação bastante activos.

Relativamente ao desenvolvimento tradicional de software, o DBC apresenta alguns desafios novos que resultam, sobretudo, de os componentes reutilizáveis serem desenvolvidos frequentemente por organizações diferentes das que os reutilizam. Isso conduz a dificuldades acrescidas no que toca à avaliação desses componentes, dado que muito frequentemente eles são distribuídos como uma caixa negra. Esta característica limita a capacidade dos integradores de componentes os avaliarem, dado que apenas a informação disponibilizada pelos produtores do componente fica disponível. Uma análise à informação disponibilizada pelos produtores de componentes revela uma grande variabilidade na sua natureza e grau de detalhe [6], o que reduz, ou praticamente anula a capacidade de efectuar estudos comparativos.

A avaliação de componentes é importante quer do ponto de vista dos produtores quer do ponto de vista dos utilizadores desses componentes. Do ponto de vista dos produtores de componentes, a avaliação da

complexidade de um componente é interessante numa perspectiva de gestão de projecto, dado que ajuda a estimar o esforço necessário à construção, evolução e depuração do componente. Aspectos como a facilidade de reutilização de um componente, a compreensão da sua interface, a sua usabilidade e a sua fiabilidade são, a par da riqueza de funcionalidades por ele oferecida, factores a ter em conta na selecção de um componente. Deste modo, são factores relevantes quer para o produtor, quer para o utilizador de componentes.

A avaliação de componentes deve ser realizada de modo a poder justificar a confiança dos utilizadores nos componentes que selecciona. Esta avaliação deverá ser feita, na medida do possível, a partir de informação que está disponível não apenas para os produtores, como para os utilizadores de componentes. Assim, a riqueza da especificação do contrato de utilização de um componente assume uma importância primordial. Por exemplo a facilidade de reutilização e adaptação de componentes poderá ser estimada a partir da complexidade da sua interface.

2 Contribuições fundamentais da dissertação

Com esta dissertação pretende-se contribuir para o estabelecimento de práticas de avaliação quantitativa de componentes e infra-estruturas como práticas correntes na ESBC. A nossa linha de acção passa, fundamentalmente, pela definição formal de métricas para avaliação de componentes e infra-estruturas de componentes, validação dessas métricas e produção de heurísticas para o desenho de software baseado em componentes, com base num modelo de qualidade também a definir no âmbito desta dissertação. É ainda nosso objectivo contribuir para a disseminação dessas práticas de avaliação através do desenvolvimento de ferramentas que as suportem e se integrem tanto quanto possível com ambientes de desenvolvimento normalmente usados pelos produtores de software baseado em componentes, dado que tal integração nos parece fundamental para a transferência das práticas de avaliação para um ambiente de produção.

3 Abordagem Proposta

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 QAOOSE, 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) [7]. 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/QUASAR/Resources/publications.htm>). Tal como acontece na ESBO, também na ESBC são necessários estudos quantitativos, quer sobre componentes, quer sobre infra-estruturas de componentes. Ao longo deste trabalho de doutoramento, pretendemos contribuir para o desenvolvimento de métodos, técnicas e ferramentas que suportem essa actividade.

A Figura 1 apresenta uma representação abstracta das interações entre as várias actividades necessárias à obtenção dos objectivos a que nos propomos na dissertação.

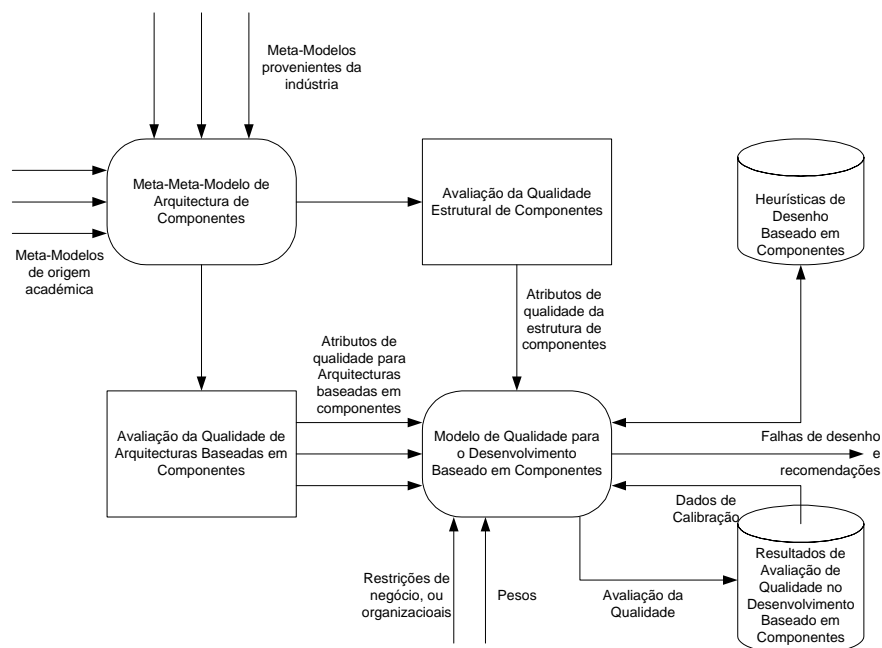


Figura 1 – Avaliação de arquitecturas de componentes

Pretendemos ter capacidade para a avaliação de componentes e infra-estruturas de componentes provenientes de várias origens, quer académica quer da indústria de software. A experiência prévia na avaliação de desenho orientado a objectos, sintetizada em [8], mostrou a utilidade de poder contar com uma representação independente para esse mesmo desenho, como facilitadora no desenvolvimento e validação de métricas para desenho orientado a objectos e de ferramentas que suportem a sua recolha. Deste modo, a utilização de um meta-meta-modelo para suportar a representação de componentes e arquitecturas de um modo padronizado afigura-se-nos como uma base de trabalho extremamente importante. Pretendemos extrair métricas a partir dessa representação, quer para a avaliação de componentes de forma isolada, quer para a avaliação de infra-estruturas de componentes. Estas métricas deverão ser definidas de modo a permitir a aferição de atributos de qualidade a utilizar num modelo de qualidade para o desenvolvimento baseado em componentes, que também pretendemos desenvolver. São ainda entradas para esse modelo os pesos a atribuir aos diferentes atributos, restrições de negócio e heurísticas de desenho baseado em componentes. O modelo deverá ser realimentado com os resultados de anteriores avaliações, por forma a ser correctamente calibrado. Como resultados práticos mais relevantes para o desenvolvimento baseado em componentes, destacamos a identificação automática de potenciais falhas de desenho e a produção de recomendações para as resolver.

Nas secções seguintes detalhamos os veios de investigação que estamos a explorar neste contexto.

3.1 Representação de infra-estruturas de componentes

Por forma a facilitar a avaliação de componentes, um primeiro passo na nossa pesquisa consistiu em procurar uma representação de infra-estruturas de componentes que capture os aspectos fundamentais dos actuais modelos (quer os de cariz académico, quer os usados em abordagens industriais).

Com um cariz sobretudo académico, destacam-se as linguagens de descrição de arquitectura (LDAs), tais como o Acme [9], Aesop [10], Adage [11], C2 [12], Darwin [13, 14], Rapide [15], SADL [16], UniCon [17], MetaH [18], e o Wright [19]. Embora cada uma destas LDAs seja especializada em diferentes aspectos da arquitectura, elas partilham um conjunto de conceitos e abstracções, em particular as de cariz estrutural, que estão factorizados na linguagem Acme.

Paralelamente, existem diversos modelos de componentes usados pela indústria de software, tais como o Java-RMI [20], o CORBA [21], o EJB [22] e o DCOM [23].

Na nossa pesquisa de uma representação que possa de algum modo capturar o essencial de abordagens tão diversas à representação de infra-estruturas de componentes, a utilização de um meta-meta-modelo afigura-se-nos como uma aproximação interessante. Ao modelo dos conceitos utilizados num modelo de componentes, chama-se “meta-modelo” de componentes. Um modelo em que os tais meta-modelos são representados é, portanto, um meta-meta-modelo.

Quando esta pesquisa se iniciou, não existia, tanto quanto sabemos, um candidato forte a desempenhar esse papel, embora já existissem algumas aproximações a um nível de abstracção bastante elevado (e.g. [24]). Embora tenhamos ponderado a possibilidade de criar um tal meta-meta-modelo, com o surgimento da proposta de especificação do meta-modelo do UML 2.0[25, 26], tornou-se evidente que a falta de expressividade de versões anteriores para representar convenientemente conceitos como a decomposição hierárquica de componentes, ou a noção de conector, entre outros, bem patente em [27-29], poderia ser ultrapassada pela nova versão.

Numa tentativa de avaliação da expressividade do novo meta-modelo do UML, apresentámos uma proposta de mapeamento de arquitecturas expressas em Acme em UML 2.0, recorrendo a expressões OCL para formalizar a semântica do mapeamento, em que se demonstra a sua adequação sobretudo na representação estrutural de infra-estruturas de componentes [30]. Por outro lado, o meta-modelo do UML 2.0, cuja publicação está actualmente em processo de finalização vem acompanhado de perfis para a representação de componentes (e.g. Enterprise Java Beans). Deste modo, optou-se por usar o novo meta-modelo do UML 2.0 como forma de representação de arquitecturas baseadas em componentes, até pelo suporte de ferramentas que se espera que esta nova norma venha a ter num futuro próximo.

Ainda que o objectivo fundamental da dissertação não seja o de transportar a modelação arquitectural, tradicionalmente mais associada ao mundo da investigação, para uma linguagem usada de um modo bastante generalizado pelos produtores de software, a formalização de um mapeamento de Acme para UML 2.0 não deixa de ser um contributo para mitigar este problema.

3.2 Definição formal de métricas para avaliação de componentes

O estado da arte no que toca à definição de métricas para a avaliação de componentes encontra-se ainda bastante aquém das necessidades da indústria de software. Diversos autores reconhecem a necessidade de definir métricas especialmente dedicadas à avaliação de componentes [6, 31, 32], dado que métricas tradicionais, quer sobre desenho orientado a objectos, quer sobre a complexidade interna de operações não são adequadas, por diversos motivos, entre os quais destacamos:

- a impossibilidade de serem recolhidas na ausência do código fonte, frequente em componentes distribuídos como caixa negra;
- a inadequação dos atributos avaliados, face às necessidades específicas do desenvolvimento baseado em componentes. Por exemplo, para o integrador de componentes, a facilidade de reutilização de um componente é mais relevante do que a sua complexidade interna, dado que o integrador de componentes se vai dedicar à construção de conectores entre os componentes que constituem uma determinada infra-estrutura, e não construir os componentes em si. As métricas a definir devem ser dirigidas aos aspectos do desenvolvimento baseado em componentes em que possam ser úteis.

Três classes de problemas fundamentais que já eram detectáveis em métricas definidas para outros contextos constituem desafios que ainda estão por resolver também no âmbito das métricas para componentes. Apresentamos de seguida uma breve discussão desses problemas, com referências para conjuntos de métricas (ou propostas preliminares para a definição de métricas) em que eles são notórios.

- Um primeiro problema é o da ambiguidade na definição de métricas, resultante da utilização, com graus variáveis de definições informais das métricas. Isto conduz a diferentes interpretações das definições, seguidas de diferentes implementações dessas métricas que podem conduzir a diferentes resultados para uma mesma métrica, aplicada sobre o mesmo artefacto, consoante a implementação da métrica. Exemplos podem ser encontrados em [6, 31-34].
- Outro tipo de problema surge quando, apesar de as métricas serem formalmente definidas, o formalismo usado nessa definição é demasiado complexo para que possa ser compreendido por quem vai ter de as usar, o que, naturalmente, limita a sua disseminação.
- Finalmente, destaca-se a insuficiente ou mesmo totalmente inexistente validação das métricas propostas, sem a qual o grau de confiança em recomendações nelas baseadas é, necessariamente, baixo [6, 31-33]. Em [35] e sobretudo na proposta de Washizaki [34], já é notória uma maior maturação das propostas de métricas neste aspecto, apesar de os esforços de validação serem ainda manifestamente insuficientes.

Pretendemos contribuir para a melhoria do estado da arte, através da formalização de propostas de métricas e desenvolvimento de suporte computacional para a sua recolha, com vista ao esforço de validação de métricas que pretendemos levar a cabo. Utilizaremos o meta-modelo do UML 2.0, populado por meta-objectos sobre os quais recolhemos métricas definidas formalmente em OCL. Como prova de conceito sobre a definição de métricas para componentes, formalizámos o conjunto de métricas de Washizaki [36]. Além de formal, a definição de métricas em OCL apresenta outros benefícios interessantes. A linguagem é

de compreensão fácil para programadores, apesar de formal (resolvendo com isso os dois primeiros problemas que afectam algumas das propostas de métricas para componentes aqui referidas) e a definição é ao mesmo tempo executável por um avaliador de OCL. Este último aspecto é sobretudo interessante por facilitar a disseminação da definição de métricas e se enquadrar bem no objectivo de disponibilizar suporte computacional à recolha de métricas que seja integrável nas ferramentas de desenho usadas por quem define as arquitecturas baseadas em componentes. Não sendo um objectivo central da dissertação proceder à formalização e validação de conjuntos de métricas definidos por outros autores, estas tarefas afiguram-se nos simultaneamente como contribuições acessórias da dissertação para o estado da arte e necessárias para procedermos a uma avaliação comparativa das nossas próprias propostas.

3.3 Definição formal de métricas para a avaliação de infra-estruturas de componentes

Avaliar componentes, por si só, não é suficiente. É importante levar em conta que os componentes são desenvolvidos de forma independente. Na prática, isso significa que o produtor de cada componente faz um conjunto de suposições acerca da utilização que esse componente vai ter, que não corresponde necessariamente à utilização que de facto o componente tem. Esse desencontro pode resultar quer de uma insuficiente percepção por parte do construtor do componente sobre as condições em que o componente irá operar, na prática, quer da visão incompleta que o utilizador de um componente tem sobre o componente que vai reutilizar. Quando se constroi uma infra-estrutura de componentes de diversas proveniências, gerir todas as interações entre os diversos componentes é uma tarefa complexa. É perfeitamente plausível que componentes que analisados individualmente apresentem uma qualidade aceitável possam em conjunto não satisfazer os requisitos. Um exemplo típico é a utilização de recursos de memória, que ao ser analisada para toda uma infra-estrutura de componentes se pode revelar excessiva, embora tal situação não fosse detectável numa análise isolada a cada componente.

Há uma outra motivação para explorar a avaliação das infra-estruturas de componentes. Com a crescente utilização de componentes COTS, a complexidade desloca-se da construção dos componentes para a sua selecção e integração, quando observada do ponto de vista dos utilizadores de componentes. No que respeita à integração, essa complexidade pode ser melhor avaliada sobre as infra-estruturas criadas.

Autores como Wallnau e Stafford, num capítulo de [37] e Simão *et al*, em [38] têm defendido que a avaliação de infra-estruturas de componentes poderá trazer maiores benefícios do que a avaliação dos componentes de uma forma isolada. Partilhamos desta visão, que deverá ser comprovada de forma empírica.

A técnica de definição formal destas métricas é semelhante à descrita na secção anterior, variando apenas o foco da definição de métricas, que deixam de estar centradas nos componentes para se centrarem nos conectores que os ligam.

3.4 Definição de um modelo de qualidade para software baseado em componentes

A avaliação de componentes e infra-estruturas de componentes deverá usar como referência um modelo de qualidade adequado. 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. Esses atributos representam influências positivas na qualidade do software. Cabem dentro desta categoria modelos de qualidade tais como o ISO9126 [39], o COCOMO II [40], os modelos de Arthur [41], Grady [42], McCall [43], Meyer [44] e Abreu [8].

Os modelos de qualidade genéricos aqui referidos não são totalmente adequados à avaliação de componentes ou sistemas neles baseados. Essa inadequação cria a necessidade de modelos de qualidade desenvolvidos especificamente para componentes. A título de exemplo, refira-se o COCOTS [45], uma extensão do COCOMO-II para contemplar os aspectos relacionados directamente com o desenvolvimento baseado em componentes. O conjunto de atributos passíveis de ser avaliados, quer por via da sua aplicabilidade ao universo dos componentes, quer pela sua visibilidade para o utilizador alvo do modelo de qualidade não é totalmente coincidente com o conjunto que seria utilizável para software em geral [46]. 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. Embora isso tenha vantagens, por exemplo no que respeita à facilidade de substituição de componentes, também tem consequências no que respeita à facilidade de avaliação dos componentes. A actividade de certificação de componentes por entidades independentes torna-se mais complicada, uma vez que essas entidades não têm acesso ao código fonte dos componentes. Uma discussão interessante sobre estes problemas pode ser encontrada em [6].

Uma crítica por vezes apontada aos modelos hierárquicos relaciona-se com a ausência de uma especificação de relações de dependência entre diferentes atributos de qualidade. Melhorias de um determinado atributo de qualidade são por vezes conseguidas sacrificando um outro. 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 interações. 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 é a dificuldade de avaliação do impacto real dos artefactos de software (ou seja, os entregáveis do processo de software) na qualidade do sistema de software como um todo. De acordo com Zhu [47], um modelo de qualidade deve:

- 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. As arestas representam as relações de dependência entre dois 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 levanta problemas quanto à generalização deste tipo de solução.

A abordagem mais centrada no processo também tem sido investigada. Um exemplo pode ser encontrado em [48], onde se descreve um projecto que tinha como objectivo a criação de normas para o processo de desenvolvimento de software baseado em componentes.

Pretendemos construir um modelo de qualidade centrado no produto que resolva problemas como os até aqui avançados. O seguinte conjunto de questões sintetiza as preocupações por detrás do modelo a construir:

- 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 resolver o conflito entre um grau de generalidade do modelo que permita comparações entre componentes ou sistemas baseados em componentes distintos e um que possibilite um fácil mapeamento desse modelo a componentes e sistemas concretos?

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 com heurísticas sobre a qualidade do desenvolvimento baseado em componentes. A validação deste modelo deverá ser feita com recurso a experiências controladas.

3.5 Produção de recomendações para o desenho baseado em componentes

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. A ideia fundamental passa por especificar em OCL heurísticas de desenho que permitam identificar, de forma automatizada, situações em

que essas heurísticas estejam a ser desrespeitadas. Tipicamente, para cada métrica, será estabelecido um intervalo de valores considerados normais, com base nas experiências empíricas de validação dessa mesma métrica. Essencialmente, uma heurística será então uma função em OCL que compara o valor calculado para a métrica com os valores típicos e indica se existe ou não uma violação da heurística.

A principal vantagem de se poder implementar este tipo de heurísticas com OCL é a facilidade de integração desta funcionalidade em ferramentas de desenho UML que suportem o OCL. Este factor é determinante, se tivermos em conta que um potencial factor de insucesso na aplicação de métricas é precisamente a insuficiente integração entre as ferramentas de desenho e as ferramentas de recolha e análise de métricas. A flexibilidade desta aproximação é também um factor interessante a ter em conta. Um utilizador de métricas terá disponível de forma perfeitamente integrada com o seu ambiente de desenvolvimento, a possibilidade de alterar as heurísticas, ou mesmo de definir novas heurísticas particularmente adequadas ao seu caso específico.

3.6 Construção de ferramentas de suporte à avaliação

A construção de ferramentas de suporte à avaliação quantitativa de componentes e suas infraestruturas é um factor essencial para que esta avaliação se possa fazer de uma forma repetível e independente. Uma vez que a representação escolhida para as arquitecturas baseadas em componentes é o UML 2.0, e que a linguagem escolhida para expressar as métricas e heurísticas a utilizar, o suporte computacional a utilizar baseia-se, naturalmente, em ferramentas que suportem tanto o UML como OCL. Aqui, levanta-se um problema prático que o tempo se encarregará de resolver. A médio prazo, é expectável que as ferramentas de desenho em UML suportem o novo meta-modelo, bem como a avaliação de expressões OCL. No entanto, actualmente, tal ainda não acontece. A versão final do novo meta-modelo ainda não foi concluída e o suporte ao OCL não é oferecido por muitas das ferramentas existentes no mercado.

Sintetizando, os requisitos fundamentais para a ferramenta de avaliação de componentes e suas infraestruturas são:

- Capacidade para gerar uma representação abstracta de meta-objects compatíveis com o meta-modelo do UML 2.0, a partir de uma dada infra-estrutura de componentes;
- Capacidade para instanciar as meta-classes do UML 2.0 com esses meta-objects;
- Capacidade para analisar e executar expressões OCL sobre o meta-modelo do UML 2.0, populado com os acima referidos meta-objects.

O ambiente que temos vindo a desenvolver combina ferramentas de desenho em UML para definir a infra-estrutura de componentes com um conversor de XMI que transforma os desenhos gerados com essas ferramentas em especificações no formato usado pela ferramenta USE. É com esta última que procedemos à especificação e cálculo de métricas em OCL.

A ferramenta USE foi desenvolvida pela Universidade de Bremen para permitir a especificação de sistemas de informação usando um subconjunto da linguagem UML e permite exprimir restrições sobre esses modelos através de expressões OCL, que podem ser verificadas com o avaliador de expressões OCL que a

ferramenta inclui. Além disso, permite que o sistema especificado seja populado e que se possa obter informação detalhada sobre esses modelos, novamente recorrendo ao OCL. Outra característica interessante desta ferramenta é a sua capacidade para permitir carregar especificações de modelos. Em particular, podemos carregar o meta-modelo do UML 2.0, popular o meta-modelo com meta-objects e extrair informação sobre esses meta-objects. Na prática, isso significa que podemos recolher métricas formalizadas em OCL, navegando nos meta-objects criados, bastando para tal que os meta-objects representem a infra-estrutura de componentes que pretendemos avaliar.

Como desvantagens da ferramenta USE, destacamos o facto de o use dispor apenas de um subconjunto do UML, deixando de fora alguns elementos do meta-modelo que seriam úteis para os nossos objectivos (tais como, por exemplo os *packages*), complicando um pouco a tarefa de definição de métricas. Outra desvantagem resulta de a ferramenta dispor de um formato próprio para a representação dos modelos. Esse facto obriga-nos a fazer a tradução a partir de XMI para o formato usado pelo USE.

Numa primeira abordagem, explorámos a construção de um tradutor de uma ferramenta que usava um formato proprietário para USE. Com a crescente adopção do XMI como formato para expressar os modelos UML, optámos por apostar na construção de um tradutor de XMI para o formato USE. Este tradutor tem a capacidade de gerar dois tipos de saída: um com uma representação de um modelo no formato USE (que usamos para carregar o meta-modelo do UML 2.0), o outro com meta-objects gerados para popular o meta-modelo do UML 2.0. É sobre estes meta-objects que calculamos as métricas definidas em OCL.

3.7 Validação de propostas

Os objectivos dos veios de investigação referidos visam criar as condições que para seja possível avaliar componentes e infra-estruturas de componentes 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 infra-estrutura de componentes 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 proposta, usando como referência o modelo de qualidade desenvolvido. Essa validação será realizada através de experiências controladas. O desenvolvimento das ferramentas referidas no ponto 3.6 visa suportar a actividade de experimentação.

Pretendemos também disponibilizar os protótipos construídos à comunidade através do site do QUASAR, contribuindo assim para que outros possam usar a nossa arquitectura na formalização e validação de métricas. A concretização de tal objectivo oferece benefícios mútuos, dado que tal utilização por utilizadores independentes contribuirá, por seu turno, para a validação da arquitectura por nós proposta.

4 Metodologia

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

Uma primeira preocupação, por ser estruturante para o trabalho que se segue, foi a de identificar um meta-meta-modelo de arquitecturas de componentes. Embora, inicialmente, a possibilidade de construir um meta-meta-modelo para arquitecturas baseadas em componentes tenha sido considerada, à falta de alternativas interessantes para os nossos objectivos de investigação, o surgimento de uma proposta fortemente apoiada por consórcio de organizações com um grande peso na indústria de software que servia os nossos propósitos levou-nos a optar pela adopção do UML 2.0.

Trabalhamos agora na construção de conjuntos de métricas e de um 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, enquanto que o nosso trabalho incidirá sobretudo nos aspectos estruturais de arquitecturas baseadas em componentes. Há, pois, uma complementaridade de projectos que se insere bem no espírito do grupo de investigação de imprimir um cariz quantitativo às práticas de engenharia de software.

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/>), à qual o nosso grupo de investigação aderiu em Julho de 2003, 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.

5 Plano de Trabalho

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 arquiteturas de componentes. Procede-se depois ao início da implementação do suporte computacional para a recolha de métricas definidas sobre esse meta-modelo. Na segunda fase, exploraremos o referido meta-meta-modelo com a criação de métricas para a avaliação quantitativa de componentes e infra-estruturas de componentes. Na terceira fase desenvolveremos um modelo de qualidade que sirva de enquadramento para a referida avaliação. 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 (já cumprido)	<ul style="list-style-type: none">• Revisão bibliográfica sobre modelos de qualidade para componentes.• Revisão bibliográfica sobre meta-modelos de arquiteturas de componentes e sua construção.• Selecção de um meta-meta-modelo para arquiteturas de componentes (UML 2.0).• Redacção de um artigo com o mapeamento de uma ADL (Acme) para o meta-modelo do UML 2.0. [30]• Desenvolvimento de suporte computacional para a recolha das métricas com base no meta-meta-modelo (esta tarefa prolonga-se para além do final do 1º ano).
2º ano	<ul style="list-style-type: none">• Redacção de um artigo sobre a formalização de métricas sobre componentes, usando o OCL, sobre o meta-modelo do UML 2.0. [36]• Desenvolvimento de métricas para a avaliação estrutural de componentes. Desenvolvimento de métricas para a avaliação de infra-estruturas de componentes.• Concepção e realização de experiências empíricas de validação das métricas para a avaliação estrutural de componentes.• Concepção e realização de experiências empíricas de validação das métricas para a avaliação de infra-estruturas de componentes.• Redacção de um artigo com os resultados das experiências realizadas e a descrição das ferramentas nelas usadas, para publicação em conferência

	<p>internacional.</p> <ul style="list-style-type: none"> • Redacção de um artigo de síntese da concepção e validação de métricas, quer para componentes quer para as suas infra-estruturas, 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

Bibliografia

- [1] L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord, and K. Wallnau, "Volume I: Market Assessment of Component-Based Software Engineering", Software Engineering Institute, Technical Note CMU/SEI-2001-TN-007, May, 2000 2001.
- [2] J. D. Williams, "Raising Components", *Application Development Trends*, vol. 7, pp. 27-32, 2000.
- [3] C. Szyperski, D. Gruntz, and S. Murer, *Component Software: Beyond Object-Oriented Programming*, 2nd ed. New York: ACM Press - Addison Wesley, 2002.
- [4] A. Egyed, N. Medvidovic, and C. Gacek, "A Component-Based Perspective of Software Mismatch Detection and Resolution", *IEE Software Engineering*, vol. 147, pp. 225-236, 2000.
- [5] G. T. Heineman and W. T. Councill, *Component-Based Software Engineering - Putting the Pieces Together*. Boston, MA: Addison-Wesley, 2001.
- [6] M. Bertoa and A. Vallecillo, "Quality Attributes for COTS Components", 6th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'2002), Málaga, Spain, 2002.
- [7] F. B. e. Abreu, H. Zuse, H. A. Sahraoui, and W. Melo, "ECOOP Workshop on Quantitative Approaches in OO Software Engineering", in *ECOOP'99 Workshop Reader, Lecture Notes in Computer Science*, A. Moreira, Ed.: Springer-Verlag, 1999.
- [8] F. B. Abreu, "Engenharia de Software Orientado a Objectos: uma Aproximação Quantitativa": IST, 2001.
- [9] D. Garlan, R. T. Monroe, and D. Wile, "Acme: Architectural Description of Component-Based Systems", in *Foundations of Component Based Systems*, G. T. Leavens and M. Sitaraman, Eds.: Cambridge University Press, 2000, pp. 47-68.
- [10] D. Garlan, R. Allen, and J. Ockerbloom, "Exploiting style in architectural desing environments", SIGSOFT'94: The Second ACM Symposium on the Foundations of Software Engineering, 1994.
- [11] L. Coglianesi and R. Szymanski, "DSSA-ADAGE: An Environment for Architecture-based Avionics Development", AGARD'93, 1993.
- [12] N. Medvidovic, P. Oreizy, J. E. Robbins, and R. N. Taylor, "Using object-oriented typing to support architectural design in the C2 style", SIGSOFT'96: Fourth ACM Symposium on the Foundations of Software Engineering, 1996.
- [13] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer, "Specifying distributed software architectures", Fifth European Software Engineering Conference, ESEC'95, 1995.

- [14] J. Kramer, J. Magee, and S. Uchitel, "Software Architecture Modeling and Analysis: a Rigorous Approach", in *Formal Methods for Software Architectures: SFM 2003*, vol. 2804, M. Bernardo and P. Inverardi, Eds. Berlin Heidelberg: Springer, 2003, pp. 44-51.
- [15] D. C. Luckham, L. M. Augustin, J. J. Kenney, J. Veera, D. Brian, and W. Mann, "Specification and analysis of system architecture using Rapide", *IEEE Transactions on Software Engineering*, vol. 21, No.4, pp. 336-355, 1995.
- [16] M. Moriconi, X. Qian, and R. Riemenschneider, "Correct architecture refinement", *IEEE Transactions on Software Engineering*, vol. 21, No. 4, pp. 356-373, 1995.
- [17] M. Shaw, R. DeLine, D. V. Klein, T. L. Ross, D. M. Young, and G. Zelesnik, "Abstractions for Software Architecture and Tools to support them", *IEEE Transactions on Software Engineering*, vol. 21, No. 4, pp. 314-335, 1995.
- [18] P. Binns and S. Vestal, "Formal real-time architecture specification and analysis", Tenth IEEE Workshop on Real-Time Operating Systems and Software, New York, USA, 1993.
- [19] J. E. Robbins, N. Medvidovic, D. F. Redmiles, and D. S. Rosenblum, "Integrating Architecture Description Languages with a Standard Design Method", International Conference on Software Engineering (ICSE98), Kyoto, Japan, 1998.
- [20] Sun, "Java Remote Method Invocation Specification", Sun Microsystems, Inc. Revision 1.8, Java 2 SDK, Standard Edition v1.4, 2002.
- [21] OMG, "CORBA Components - Version 3.0", Object Management Group Inc., Specification formal/02-06-65, June 2002.
- [22] S. Microsystems, "Enterprise JavaBeans™ Specification, Version 2.1", November 2003.
- [23] M. Corporation, "DCOM Technical Overview", Microsoft Corporation 1996.
- [24] U. Rasthofer, "Modeling With Components - Towards a Unified Component Meta Model", Model-Based Software Reuse Workshop on ECOOP'2002, Malaga, Spain, 2002.
- [25] U2-Partners, "3rd revised submission to OMG RFP ad/00-09-01: Unified Modeling Language: Infrastructure - version 2.0", U2-Partners January 2003.
- [26] U2-Partners, "2nd revised submission to OMG RFP ad/00-09-02: Unified Modeling Language: Superstructure - version 2.0", U2-Partners January 2003.
- [27] D. Garlan and A. J. Kompanek, "Reconciling the Needs of Architectural Description with Object-Modeling Notations", <<UML>> 2000, York, UK, 2000.
- [28] D. Garlan, "Formal Modeling and Analysis of Software Architecture: Components, Connectors and Events", in *Formal Methods for Software Architectures*, vol. 2804, LNCS, M. Bernardo and P. Inverardi, Eds. Bertinoro, Italy: Springer, 2003, pp. 1-24.
- [29] N. Medvidovic, D. S. Roseblum, D. F. Redmiles, and J. E. Robbins, "Modeling Software Architectures in the Unified Modeling Language", *ACM Transactions on Software Engineering and Methodology*, vol. 11, pp. 2-57, 2002.
- [30] M. Goulão and F. B. Abreu, "Bridging the gap between Acme and UML for CBD", Specification and Verification of Component-Based Systems (SAVCBS'2003), at the ESEC/FSE'2003, Helsinki, Finland, 2003.
- [31] N. S. Gill and P. S. Grover, "Component-Based Measurement: Few Useful Guidelines", *ACM SIGSOFT Software Engineering Notes*, vol. 28, 2003.
- [32] S. Sedigh-Ali, A. Ghafoor, and R. A. Paul, "Software Engineering Metrics for COTS-Based Systems", *IEEE Computer*, 2001.
- [33] R. Dumke and A. Schmietendorf, "Possibilities of the Description and Evaluation of Software Components", *Metrics News*, vol. 5, 2000.
- [34] H. Washizaki, H. Yamamoto, and Y. Fukazawa, "A Metrics Suite for Measuring Reusability of Software Components", Metrics'2003, 2003.
- [35] M. A. S. Boxall and S. Araban, "Interface Metrics for Reusability Analysis of Components", Australian Software Engineering Conference (ASWEC'2004), Melbourne, Australia, 2004.
- [36] M. Goulão and F. B. Abreu, "Formalizing Metrics for COTS (submitted)", MPEC'2004, Edimburgh, 2004.
- [37] I. Crnkovic and M. Larsson, *Building Reliable Component-Based Software Systems*. Boston: Artech House Publishers, 2002.
- [38] R. P. S. Simão and A. D. Belchior, "Quality Characteristics for Software Components: Hierarchy and Quality Guides", in *Component-Based Software Quality: Methods and Techniques*, LNCS 2693, A. Cechich, M. Piattini, and A. Vallecillo, Eds.: Springer, 2003, pp. 184-206.

- [39] ISO9126, "Information Technology - Software Product Evaluation - Software Quality Characteristics and Metrics". Geneva, Switzerland: International Organization for Standardization.
- [40] B. W. Boehm, B. K. Clark, E. Horowitz, R. Maduchy, R. Selby, and C. Westland, "An overview of the COCOMO 2.0 software cost model", University of Southern California, Technical Report 1995.
- [41] L. J. Arthur, *Measuring Programmer Productivity and Software Quality*: Wiley-Interscience, 1985.
- [42] R. B. Grady and D. L. Caswell, *Software Metrics: Establishing a Company-Wide Program*. Englewood Cliffs, NJ, EUA: Prentice-Hall, 1987.
- [43] J. McCall, "Quality Factors", in *Encyclopedia of Software Engineering*, vol. I+II, J. J. Marciniak, Ed.: John Wiley & Sons, 1994, pp. 958-ff.
- [44] B. Meyer, *Object-Oriented Software Construction*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1997.
- [45] C. Abts, B. Boehm, and E. Bailey, "COCOTS: A COTS Software Integration Lifecycle Cost Model - Model Overview and Preliminary Data Collecting Findings", University of Southern California, Technical Report March 2000.
- [46] M. Goulão and F. B. e. Abreu, "Towards a Components Quality Model", Work in Progress Session of the 28th Euromicro Conference - Euromicro 2002, Dortmund, Germany, 2002.
- [47] H. Zhu, Y. Zhang, Q. Huo, and S. Greenwood, "Application of Hazzard Analysis to Software Quality Modelling", 26th Annual International Computer Software and Applications Conference, COMPSAC'2002, Oxford, England, 2002.
- [48] A. F. Ackerman, "Quality Grades for Software Component Source Code Packages - Informal Draft 0.1.1", Institute for Zero Defect Software 1997.