revista MUNDOJAVA®

Artigo publicado
na edição 18

julho/agosto de 2006

O reuso de software envolve uma variedade de técnicas utilizadas desde as etapas de design até a implementação. O principal objetivo é evitar o retrabalho no desenvolvimento de um novo projeto, capitalizando trabalhos anteriores, fazendo com que as soluções já desenvolvidas sejam imediatamente implementadas em novos contextos. Para uma aplicação efetiva da reutilização de soluções prontas, deve-se conhecer bem a fundamentação, ferramentas, tecnologias e as métricas através das quais se pode avaliar o ganho de produtividade e economia obtido.

O reuso na prática

O reuso como diferencial competitivo em produtividade e qualidade no desenvolvimento de software.

Em todo mundo, as áreas de TI buscam cada vez mais agilidade, economia e inovação tecnológica, que constituem o tripé da competitividade da indústria de software. Essa tendência foi confirmada recentemente na Conferência Anual do Gartner, um dos maiores grupos de desenvolvimento de pesquisa e análises de TI no mundo, realizada no final de março de 2005 e que reuniu especialistas de diferentes países. E dentro de inúmeras tecnologias e soluções que aparecem todos os dias, a última edição do evento do Gartner enfocou a Integração de Aplicativos e Serviços na Web, demonstrando que grande parte dos especialistas aposta no reuso e em novas abordagens de construção como arquitetura orientada a serviços - SOA (Service-Oriented Architecture) e Web services, buscando ganho de produtividade e otimização de investimentos.

Reuso pode ser considerado uma denominação genérica, envolvendo uma variedade de técnicas utilizadas desde as etapas de design até a implementação. O principal objetivo é evitar o retrabalho no desenvolvimento de um novo projeto, capitalizando trabalhos anteriores, fazendo com que as soluções já desenvolvidas sejam imediatamente implementadas em novos contextos. Dessa forma, tem-se melhores produtos em um menor intervalo de tempo, com redução nos custos de manutenção, pois as partes do sistema são independentes, e facilitando a inclusão de novas funcionalidades. Além disso, a qualidade aumenta devido ao reuso de componentes previamente bem testados. [D'Souza et al., 1999], [Jacobson et al., 1997]. Uma das principais motivações na utilização de desenvolvimento baseado em componentes é exatamente a premissa de reutilização. Isso implica no reaproveitamento de partes já desenvolvidas. Esta abordagem, por reduzir o tempo de desenvolvimento e aumentar a qualidade do produto final é, de fato, bastante atraente para empresas de tecnologias [Krueger, 1992]

Este artigo apresenta diversos conceitos relacionados à componentização, principal prática de reuso na área de TI. Indica o CBD (Component-Based

Marcílio Oliveira
(marcilio@cit.com.br)
É pesquisador e consultor na área de reuso da DA Inc, empresa do grupo Ci&T Software. Bacharel em Ciência da Computação pela Universidade Federal de Viçosa e mestrando no Instituto de Computação da Unicamp, atuou como líder de pesquisa e desenvolvimento no Laboratório de Inovação em Software Ci&T/Unicamp durante dois anos, com realização de P&Ds envolvendo tecnologias como J2EE, Web services, redes P2P e componentização.

Kleber Bacili
(krbacili@cit.com.br)
É gerente de Produto da Ci&T Software, formado em Engenharia de Computação pela Unicamp. Atua há mais de seis anos em projetos de e-Business em corporações como Motorola, Natura, SKY, BankBoston e Petrobras. Por dois anos, atuou no grupo de arquitetura J2EE e como coordenador da Estratégia de Componentes e Reutilização da Ci&T. Atualmente, é responsável pela definição dos novos produtos da empresa e docente no curso de pós-graduação em Engenharia de Software do IBTA.

Development) como uma forte tendência no mercado de software, visando aumento de produtividade e qualidade no desenvolvimento de sistemas. São apresentados conceitos sobre essa técnica e alguns dados que mostram o ganho de produtividade e qualidade no desenvolvimento, assim como as métricas adotadas para análise de tais dados. Além disso, apresentamos também alguns conceitos iniciais de tecnologias promissoras como SOA e MDA, diretamente ligadas ao reuso de software.

Benefícios da reutilização

No cenário atual, encontramos duas principais necessidades que podem ser decisivas na avaliação final da qualidade e competitividade de uma equipe de desenvolvimento. Duas necessidades (vide figura 1) chamam a atenção:

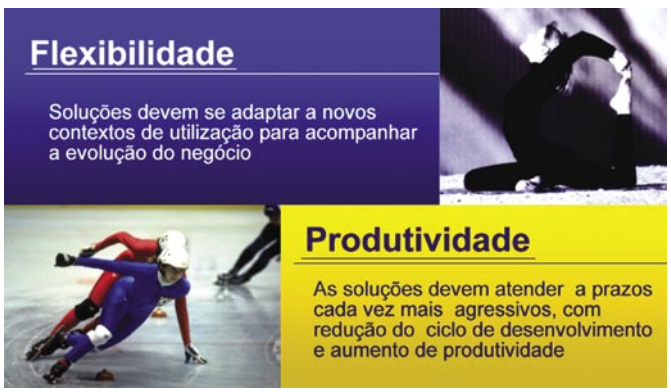

Figura 1. Flexibilidade e produtividade: necessidades decisivas na avaliação da competitividade.

O reuso demanda conhecimento e disciplina no desenvolvimento. No entanto, seus benefícios são rapidamente observados em empresas que reutilizam soluções em diferentes contextos. Dentre os principais benefícios da reutilização, podemos citar:

- redução de custos e prazos de novos projetos através do aumento de produtividade proporcionado pelo reaproveitamento de componentes/serviços;
- padronização da arquitetura dos produtos desenvolvidos pela empresa;
- oportunidade de reutilização de serviços legados através da exposição de suas interfaces como novos serviços;
- agilidade na manutenção das aplicações devido à padronização de sua arquitetura;
- aumento na qualidade dos produtos gerados através da reutilização de partes prontas e testadas.

A importância estratégica da tecnologia da informação para a competitividade em qualquer segmento de mercado é inquestionável. No entanto, a pressão pela racionalização dos investimentos em tecnologia também é grande. Afinal, a tecnologia por si só não tem valor, mas somente quando atrelada a objetivos de negócio bem definidos. No cenário atual, encontramos necessidades que se encaixam com a aplicação de técnicas de reutilização.

Sobretudo, o conhecimento do acervo digital é fator fundamental frente às necessidades de reuso. Se considerarmos e dividirmos alguns aspectos que influenciam negativa e positivamente na produtividade e qualidade (vide figura 2) fica notável que o conhecimento do acervo digital e o reuso estão estreitamente relacionados ao bom desempenho no desenvolvimento.

A visibilidade alavanca a qualidade e economia, melhorando o time-to-market, eliminando o desenvolvimento duplicado, diminuindo custos de manutenção, etc. Por outro lado, o desconhecimento do acervo

digital acarreta no desenvolvimento duplicado, manutenção cara, conflito de TI e Negócios, projetos mais caros e demorados, arquitetura sem padrões e diversos outros aspectos, apresentados na figura 2.

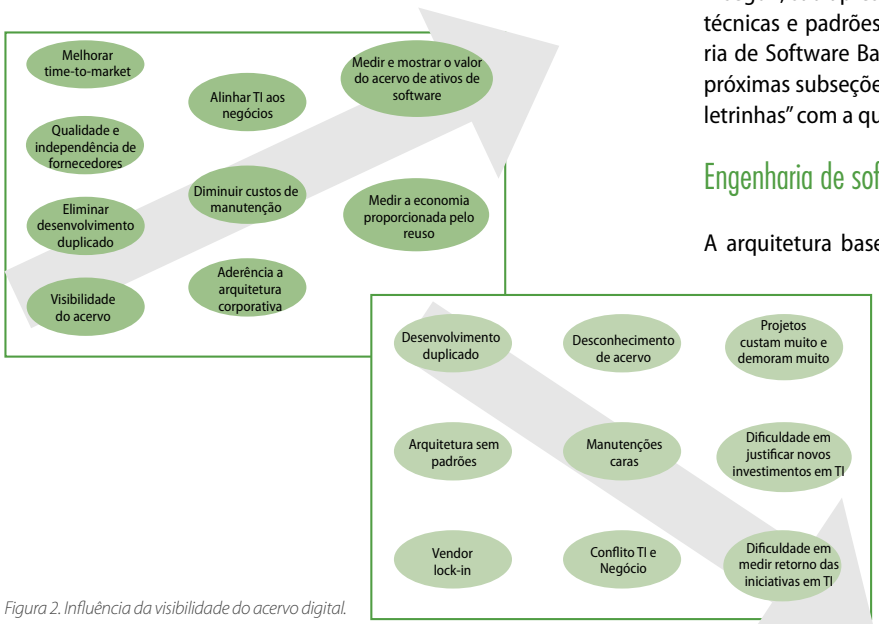

Figura 2. Influência da visibilidade do acervo digital.

Hoje, a produtividade e economia são alvos perseguidos pelas fábricas de software. Poderíamos listar alguns aspectos que tendem a influenciar diretamente a produtividade no desenvolvimento e, como apontado em [Barry,1999], o impacto sobre a economia de custos é descrito da seguinte forma:

- Trabalhando mais rápido (ex.: automação, ferramentas e ambientes): 8% de ganho.
- Trabalhando de forma mais inteligente (ex.: melhoria dos processos): 17% de ganho.
- Evitando o trabalho desnecessário (ex.: reuso dos artefatos de projeto): 47% de ganho.

No entanto, frente a tantas promessas, o que nos faz acreditar que agora é o momento apropriado para acreditarmos no reuso como principal responsável pela economia e produtividade? Bem, é possível perceber alguns fatores como o amadurecimento da arquitetura de sistemas distribuídos (J2EE, MS.Net), Servidores de Aplicações e Ferramentas, o mercado crescente de componentes prontos para uso (COTS – Commercial Off The Shelf) e momento econômico favorável

Do ponto de vista técnico, quando falamos da reutilização de ativos digitais, podemos estar nos referindo a diferentes tipos de ativos, como Web services, Componentes Java (Java beans, EJBs ...), componentes Microsoft (VB, COM, Dll, Activex ...), schemas e padrões de mensagem XML, componentes de banco de dados (Stored Procedures, Packages, etc.), componentes em sistemas legados (Subrotinas COBOL/ADABAS ...), dentre outros.

A componentização, ou desenvolvimento baseado em componentes, é a prática mais utilizada como reuso de software, mas outras fortes tendências se apresentam como SOA e MDA (Model-Driven Architecture).

Técnicas e padrões.

A seguir, são apresentadas, conceitualmente e de forma superficial, as técnicas e padrões relacionados a reuso de software, como Engenharia de Software Baseada em componentes, CBD, MDA, SOA e RAS. As próximas subseções apresentam mais informações sobre esta "sopa de letrinhas" com a qual vamos conviver por um bom tempo.

Engenharia de software baseada em componentes

A arquitetura baseada em componentes implica num elevado nível de modularização, reduzindo significativamente os custos de manutenção e evolução da solução. Novas funcionalidades podem ser facilmente adicionadas através da extensão de componentes preexistentes ou da criação de novos componentes.

Uma analogia que pode auxiliar na compreensão sobre componentes de software é o brinquedo Lego. As peças de Lego, clássico brinquedo de montar, podem simbolizar a idéia de componentes de software em um sentido mais geral. As peças do Lego permitem que montagem de automóveis, casas, aviões, navios, pessoas e muitas outras mais, limitada apenas pela criatividade dos montadores.

A montagem no Lego consiste em reunir diversas peças de diferentes formatos, encaixá-las entre si e obter, no final, uma figura totalmente nova. Existem algumas peças que podem e provavelmente serão utilizadas na construção de qualquer uma das montagens. As pequenas peças retangulares (aquelas que vêm em maior número) provavelmente estarão presentes em todas as montagens feitas com o brinquedo. Outras peças, por sua vez, têm funções mais específicas e serão utilizadas dentro de sua especialidade, dificilmente se adaptando a outros contextos. Um eixo, por exemplo, pode ser utilizado em automóveis e aviões, mas provavelmente não o será na montagem de casas. Nada impede, porém, que o montador visualize uma nova função para o eixo em um novo contexto e o utilize em uma montagem.

O exemplo é simples, mas é possível extrair dele algumas características básicas de componentes:

- componentes são reutilizáveis;
- alguns componentes são de uso mais geral e outros têm uso mais específico, mas todos são genéricos dentro do escopo considerado;
- componentes interagem com outros componentes.

Componentes são módulos de software pré-construídos, pré-testados e reutilizáveis que oferecem um conjunto de serviços através de uma interface bem definida. Representam unidades funcionais auto-suficientes que se comunicam umas com as outras através de interfaces padronizadas e podem ser combinadas para formar complexos sistemas como peças de um quebra-cabeça ou, em outra clássica analogia, como os inúmeros componentes em uma linha de montagem de automóveis.

Utilizam-se componentes para a criação de sistemas mais complexos, por meio da composição/integração de componentes. As técnicas

relacionadas à componentização do software, portanto, visam à otimização dos recursos de desenvolvimento de sistemas de software, com o reaproveitamento sistemático de componentes já desenvolvidos. A Engenharia de Software Baseada em Componentes, por conta disso, define diversas características fundamentais e estuda as vantagens e desvantagens da adoção de componentes.

Embora discutido há bastante tempo, a reutilização de software encontrou solo fértil somente nos últimos anos com o amadurecimento da infra-estrutura tecnológica e o surgimento de plataformas maduras como J2EE e Microsoft .NET, além do surgimento do conceito de Arquitetura Orientada a Serviços (SOA) e dos serviços Web (Web services);

CBD – Component-based Development

O desenvolvimento baseado em componentes é a terceira via entre o desenvolvimento de aplicações sob encomenda (a partir do zero) e a busca de soluções através da implantação de grandes pacotes de software (tipicamente ERP, CRM, OLAP, etc.). Se no primeiro caso temos um elevadíssimo custo de desenvolvimento associado ao grande risco de insucesso (estimado hoje em cerca de 80%), no segundo caso, temos grandes (e em alguns casos intermináveis) esforços de implantação de soluções engessadas que raramente oferecem algum tipo de diferencial competitivo à empresa (visto que a solução de prateleira está sempre disponível para o concorrente). A terceira via, da componentização, permite a construção rápida e flexível de soluções customizadas através de componentes de software pré-construídos.

O conhecimento das aplicações, módulos e componentes de software já existentes bem como suas interdependências é fundamental para avaliar impactos gerados por mudanças, avaliar as entregas dos fornecedores a fim de garantir aderência a seus padrões e proporcionar reutilização desses ativos de software. Segundo constatou [Ambler, 05] um aspecto fundamental para que uma empresa consiga ter economia através da reutilização de ativos de software é o conhecimento do que já existe para ser reutilizado. Isso pode ser obtido através da adoção de repositórios para gerenciamento de componentes de software e também de uma estrutura que apresente detalhes de como o parque de aplicações e componentes de software é composto e está estruturado.

A utilização de partes prontas na construção de novas aplicações implica na existência de uma biblioteca de componentes prontos. As empresas podem organizar essas bibliotecas de diversas formas:

- diretório de rede mapeado;
- ferramentas de controle de versão;
- ferramentas específicas para a administração de componentes.

Diretórios de rede compartilhados, embora seja uma opção para armazenamento e distribuição de componentes, deixa a desejar por diversos aspectos: controle de versão das informações existentes, dificuldade para se encontrar componentes (carência de mecanismos de busca eficientes), inexistência de mecanismos de colaboração que favoreçam a reutilização de componentes entre diversos outros. Já as ferramentas de controle de versão oferecem alguns pontos positivos, mas ainda são insuficientes na tarefa de promover a reutilização de componentes. Então, a utilização de uma ferramenta específica para administração de componentes é altamente recomendada para empresas que buscam aumento de produtividade através da reutilização de componentes. Algumas ferramentas de gestão de componentes e ativos digitais são citadas mais a diante.

SOA – Service-Oriented Architecture

Uma arquitetura orientada a serviços é essencialmente uma coleção de serviços que se comunicam entre si. A comunicação entre estes serviços faz-se independente de contexto e estado, e é aplicada dentro de uma arquitetura de sistemas distribuídos.

Em muitas empresas de tecnologias que passam por alguma reestruturação ou evolução tecnológica, a utilização de SOA é vista com muito bons olhos como uma forma de ligar o "velho mundo" com o "novo mundo" de TI. E tudo isso sob a premissa da economia gerada pelo reuso.

A tecnologia dos Web services é a que mais se relaciona com SOA. Os Web services representam o início da comercialização de TI como serviços providos através da internet, como alternativa à aquisição e manutenção de todas as necessidades de hardware e software. De imediato, trata-se de uma forma simples e padronizada de utilizar a malha internet para implementar a integração entre sistemas heterogêneos, de forma bastante flexível, substituindo as tradicionais estratégias de EAI. Num segundo momento, o processo evolui para a descoberta (e aquisição) dinâmica de serviços disponíveis na rede, base da realização tecnológica dos aguardados e-marketplaces.

> "Em 2010, mais de 50% das grandes empresas terão estabelecido um portfólio de sistemas sob o padrão SOA em busca de uma plataforma unificada de negócios."
>
> *Gartner Group*

A aplicação do conceito de Web services em escala tem como fundação a implementação de soluções baseadas em componentes de software. A transformação da indústria de software em uma indústria de partes representa um salto qualitativo, permitindo que uma complexa solução seja projetada pela composição de componentes de software pré-construídos e criando uma poderosa abordagem chamada arquitetura de serviços distribuídos. Esta arquitetura, intrinsecamente aberta, interoperável, modular e de baixo acoplamento, substitui os tradicionais sistemas monolíticos, rígidos e fortemente acoplados.

Aplicação de técnicas de identificação de componentes de software permite extrair Web services das aplicações desenvolvidas pelas organizações, possibilitando um aumento do índice de reuso intra e extra-organização, se este for o objetivo desta. No entanto, a eficácia desta identificação é extremamente dependente da abordagem utilizada. Os serviços, por sua vez, são dependentes da tecnologia utilizada para o desenvolvimento do software do qual se deseja extrair os componentes. No Brasil, muito pouco se pesquisou sobre este assunto, a indústria ainda não tem como prática o uso destas técnicas, sendo que alguns projetos de pesquisa já vêm trabalhando nisso, mas ainda sem resultados comerciais efetivos.

MDA – Model Driven Architecture

Atualmente, durante o processo de desenvolvimento, desde o projeto (e, às vezes, até mesmo na análise) algumas decisões são baseadas no ambiente onde a aplicação será implantada e utilizada e qualquer alteração neste ambiente impacta diretamente, podendo inviabilizar melhorias e correções.

Neste contexto, MDA surge como uma nova maneira de especificar e construir sistemas, na tentativa de libertar a modelagem da aplicação totalmente da implementação, focando principalmente no modelo de negócio. Dessa forma, MDA implica em um novo papel da modelagem dentro do desenvolvimento de software, visando, sobretudo, proporcionar um novo salto na produtividade do desenvolvimento.

Baseada em padrões e mantida pelo OMG, Model-Driven Architecture tende a se tornar um padrão de modelagem no desenvolvimento de aplicações em um futuro próximo, tornando as aplicações cada vez mais independentes e flexíveis. As principais tecnologias relacionadas a MDA são:

- UML 2.0 (Unified Modeling Language). Uma notação padronizada mundialmente. Inclui a OCL (Object Constraint Language);
- XMI (XML Metadata Interchange), que é uma linguagem-padrão para troca de informações de modelos UML;
- CWM (Common Warehouse Metamodel). Promove a integração das últimas iniciativas de Data Warehouse;
- MOF (Meta-Object Facility). Provê a padronização de repositórios.

O principal diferencial de MDA está no fato do desenvolvimento ser padronizado e baseado nas atividades de modelagem. Uma visão inicial (e muito conhecida) de MDA é mostrada na figura 3.

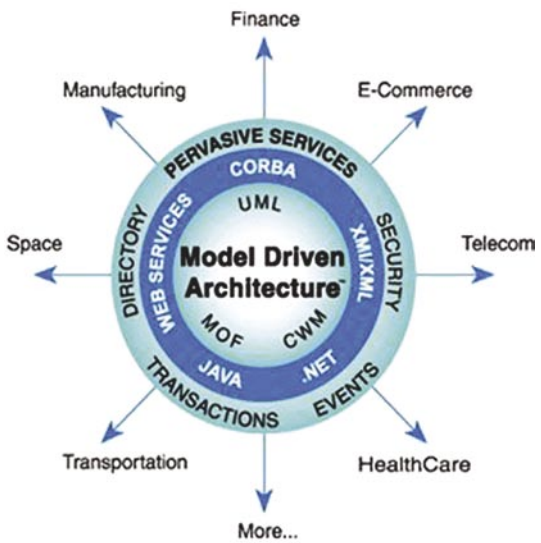

Figura 3. Visão de MDA [http://www.omg.org/mda/].

Os modelos MDA, de forma bastante resumida, são PIM e PSM. O ciclo de desenvolvimento compreende quatro fases essenciais:

- Computational-Independent Model (CIM). É o conjunto de requisitos, com uma visão inicial e geral do sistema a ser desenvolvido;
- Platform-Independent Model (PIM) representa a funcionalidade e o comportamento do negócio, sem influência dos detalhes tecnológicos. Consiste em um modelo detalhado, declarando pré e pós-condições em OCL e a semântica em Action Language;
- Platform-Specific Model (PSM). É um modelo de realização, específico para a plataforma a ser utilizada. Geralmente, é gerado a partir de um mapeamento da ferramenta MDA;
- Código-fonte. A ferramenta MDA gera todo ou a maioria do código da implementação para a tecnologia selecionada pelo desenvolvedor. Consistem em mapear o PSM para interfaces, código, descritores de GUI, SQL queries, etc.

O ciclo de desenvolvimento de uma arquitetura guiada a modelos é relativamente simples, como apresentado na figura 4.

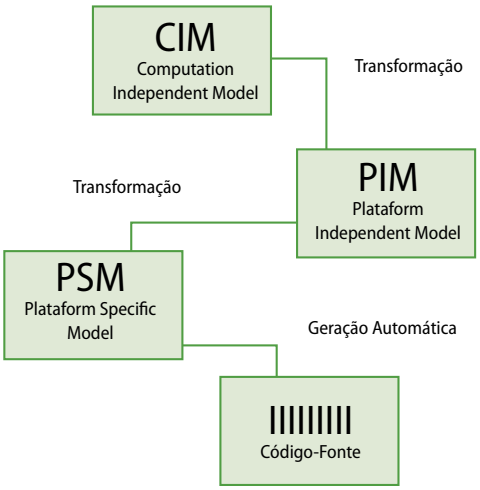

Figura 4. Ciclo de desenvolvimento MDA.

MDA mostra-se uma proposta bastante interessante para resolver muitos problemas enfrentados hoje em dia e, além de existir uma série de ferramentas que a implementam, por ser uma iniciativa independente de fornecedor, assegura a continuidade, objetividade e compatibilidade.

RAS – Reusable Asset Specification

O RAS é uma especificação adotada pela OMG para representação de ativos digitais, como componentes de software. Nesta especificação, o asset é estruturado em seções e pode estar relacionado a um determinado profile, possibilitando o armazenamento de características específicas para diferentes tipos de assets, como Web services, componentes, patterns e frameworks, definindo cenários distintos.

O RAS define uma forma-padrão para empacotamento de assets, que possuem coleções de artefatos que provêem a solução para um problema em um determinado contexto. Ele se diferencia dos demais artefatos devido ao fato de empacotar todos os arquivos que implementam uma solução e estruturar meta informações que definem e descrevem o seu comportamento por inteiro. O RAS surgiu de uma proposta conjunta das empresas Adaptive, Blueprint, ComponentSource, Flashline, IBM, LogicLibrary e OSTnet, a fim de fomentar o mercado de componentes, criando um padrão consolidado para representação de assets. A figura 5 ilustra uma definição em alto nível de um asset. As regras de utilização de um asset são as instruções descrevendo como o asset deve ser utilizado.

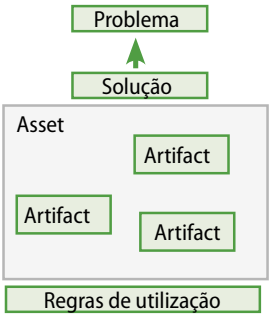

Figura 5. Estrutura de um asset segundo RAS. [OMG,05].

Os artefatos são os produtos gerados para serem utilizados no desenvolvimento, como documento de requisitos, modelos, arquivos com código-fonte, descritores, casos de teste, scripts, dentre outros. Em geral, o termo "artefato" está associado a um arquivo.

O principal objetivo é especificar a quantidade mínima de meta informações que um asset ou componente deve ter. Além disso, ele determina quais artefatos um componente possui, desde diagramas e casos de uso até os arquivos de teste, código-fonte e binário. Possibilita também classificar estes artefatos. As meta informações ficam armazenadas em um arquivo definido como "manifest", que é uma descrição em XML de várias características presentes no ativo representado.

Além da especificação fundamental do ativo, chamada de Core RAS, uma das características mais marcantes do RAS é a flexibilidade de se definir "papéis", ou profiles, para os ativos digitais. O RAS 2.1 disponibiliza profiles default, de forma que o construtor do ativo digital possa estendê-los e construir papéis específicos de acordo com o contexto da aplicação. Tanto os profiles default quanto os estendidos comunicam ao usuário do ativo quais domínios de utilização aquele ativo é convenientemente empregado.

A especificação 2.1 também padroniza os formatos das requisições e respostas em um cenário típico em que o cliente pesquisa por ativos em um repositório em conformidade com o RAS.

A especificação do RAS sugere o empacotamento do ativo digital em um arquivo de extensão .ras. Este arquivo constitui um container compactado, em formato ZIP, que reúne os demais arquivos. Todo ativo dentro do padrão RAS deve ser descrito por no mínimo um arquivo-manifesto (do inglês mainfest-file). Dessa forma, um arquivo .ras válido deve ter pelo menos um arquivo-manifesto e um artefato.

O arquivo-manifesto é um documento formato Extensible Markup Language (XML), que descreve os conteúdos de um ativo. O nome sugerido para este arquivo é rasset.xml, e cada artefato que compõe o ativo deve ser referenciado por este arquivo dentro do elemento XML <solution>. O arquivo-manifesto deve ser colocado na raiz do artefato e deve ser acompanhado de um arquivo de esquema do XML.

A figura 6 descreve o processo de construção do empacotamento RAS, a partir da especificação em alto nível (texto ou modelo UML). Um esquema XML é obtido a partir da especificação em alto nível, que por sua vez valida um arquivo-manifesto que é o ponto de entrada do pacote com o ativo descrito.

O RAS suporta mais de um tipo de empacotamento de ativo. A primeira abordagem é um pacote compactado que encapsula o manifesto e os artefatos em um só lugar, de forma a facilitar a distribuição. A outra abordagem, encontrada em servidores de controle de versão, é não ter encapsulamento em um arquivo compactado, mas ter um arquivo-manifesto em um local no servidor e os artefatos em outros locais, referenciados pelo arquivo. Uma variante da segunda abordagem, que é também suportada, é o arquivo-manifesto referenciar não os artefatos originais, mas sim referenciar as versões modificadas dos artefatos em um novo local.

A especificação do RAS inclui uma extensão do default profile para componentes de software, chamado default component profile. Este profile explicitamente adota os conceitos da metodologia UML Components, portanto especificando meta informações para os modelos e tipos construídos usando esta metodologia.

Atualmente, o RAS é o padrão de representação de componentes mais utilizado, muito disso devido à sua flexibilidade e adoção da OMG como padrão oficial de representação de ativos digitais. A documentação oficial está disponível em http://www.omg.org/technology/documents/formal/ras.htm.

Desafios

Embora as promessas sejam muitas, muitos também são os desafios encontrados na adoção do reuso em companhias que ainda não utilizem estas práticas. Muitas vezes, é necessária uma reestruturação na área de TI, envolvendo adaptação de diferentes setores da empresa. A mudança em alguns aspectos (vide figura 7) é inevitável.

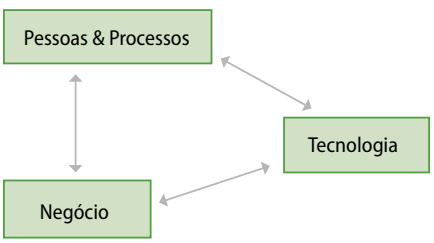

Figura 7. Áreas de desafios da implantação do reuso.

Pessoas

A implantação de práticas de reutilização requer mudança de mentalidade, visando desenvolvimento com reuso e para reuso.

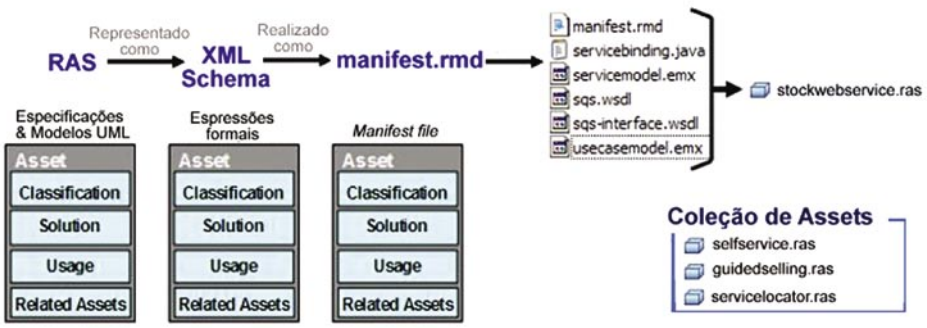

Figura 6. Processo de construção do arquivo .ras [OMG, 2005].

Muitas vezes, a adaptação requer direcionamento corporativo e patrocínio executivo. É preciso motivar as equipes de projeto a gerar novos componentes e reutilizar componentes prontos.

Além de uma mudança de mentalidade, também surgem novos papéis ligados à adoção de CBD, como o papel de administração de componentes e o de desenvolvimento e suporte a componentes.

Processos

É necessária uma adaptação no processo de desenvolvimento para tornar o reuso uma prática sistemática, através de novas atividades e responsáveis no processo utilizado.

Exemplos de atividades de um processo preparado para reuso seriam:

- etapa de decomposição da aplicação em componentes: modelagem do sistema considerando sua divisão e a aplicação de componentes de software no desenvolvimento;
- etapa para verificação do que já existe no acerv: nesta parte do processo, é importante se ter o conhecimento do acervo de componentes. O ideal é possuir alguma ferramenta para gestão efetiva de componentes de software;
- etapa de submissão dos componentes gerados para análise: as partes do sistema que forem desenvolvidas no projeto devem ser feitas de maneira componentizada, constituindo novos componentes, para serem reutilizados em projetos futuros. Os componentes gerados devem ser submetidos para alguma equipe responsável, geralmente um grupo de CBD, para validação do componente construído, visando sua reutilização futura;
- atividades de extração de medidas: a utilização de métricas para avaliação do potencial de reuso e do reuso real dos componentes é importante. Existem métricas que fornecem o percentual de reuso e o RCA (Reuse Cost Avoidance) na utilização de componentes em projetos específicos. Algumas métricas são apresentadas mais adiante neste artigo;
- processos de registro e atualização de componentes no catálogo: os componentes gerados no projeto devem ser registrados no mecanismo ou ferramenta de gestão de componentes. Além disso, durante algum projeto, um componente já existente pode ser evoluído e este deve ser atualizado no acervo.

Tecnologia

Plataformas e linguagens mudam com muita freqüência, o que pode dificultar a definição de uma tecnologia-padrão. Além disso, é necessário estabelecer uma Arquitetura Padronizada que favoreça o reuso. Por outro lado, a busca por "Arquiteturas Padronizadas" abre espaço para abordagens como SOA & Web services, o que promove a constante inovação e busca por tecnologias que se adaptem às novas tendências tecnológicas. Um excelente componente, flexível, bem documentado e que resolve um problema recorrente só pode ser considerado um bom componente se as pessoas conseguirem encontrá-lo, entendê-lo, avaliá-lo e reusá-lo, caso contrário, será um ótimo segredo!

Outras decisões também devem ser tomadas, considerando custos e benefícios. Sobretudo, vale lembrar que:

- **Fazer de forma componentizada custa mais:** deve-se bancar o investimento inicial (apostando, claro, no retorno em potencial que poderá ser colhido).
- **Integrar componentes prontos aumenta os riscos dos projetos:** existe também o investimento de integração (entendimento, configuração, testes, etc.).
- **Manter o acervo atualizado requer investimento:** é necessária a definição dos responsáveis pelos componentes quanto à documentação e à manutenção. Provavelmente, requer treinamento ou contratação de mão-de-obra especializada.

Embora a adoção de uma ferramenta de administração de ativos focada em reuso não garanta por si só o sucesso da iniciativa de reutilização da empresa, ela irá aumentar significativamente suas chances.

> "As pessoas não podem reusar o que não conseguem encontrar, é necessário saber onde as coisas estão."
>
> *Enterprise Unified Process (The Strategic Reuse Discipline: Scott Ambler, John Nalbone e Michel J. Vizdos)*

Ferramentas que gerenciem os ativos da empresa facilitam o controle, a organização, a visibilidade adequada, o controle de versões e a colaboração entre equipes.

A seguir, são listados também diversos aspectos (que muitas vezes podem ser considerados vícios) que dificultam a implantação de reuso com sucesso. São o que chamamos de "anti-patterns".

Os "Anti-patterns"

Existem, sobretudo, diversos "vícios" em centros de desenvolvimento que se tornam verdadeiros obstáculos na implantação do reuso em ambientes corporativos.

- **Artefato não-reutilizável e declarado reutilizável:** algum ativo é declarado como reusável, mas na verdade não é. Uma opção é que outra pessoa, além do autor, verifique se é de fato. Outra, é que seja declarado reusável somente quanto tiver sido reusado. Reuso é qualidade e não quantidade!
- **Not-Invented-Here Syndrome** (Síndrome de "isto não foi inventado aqui"). Embora este seja um problema já contornado em diversos ambientes, ainda pode ser citado como um dos principais desafios. Os desenvolvedores e as equipes de projeto deixam de utilizar determinado ativo, pois não foram eles que criaram. Normalmente, equipes mais maduras, reutilizam – desde que o ativo tenha qualidade, esteja bem documentado e seja fácil de entender.
- **Organização singular:** muitas pessoas gostam de pensar que sua situação é radicalmente diferente da de todos os outros, que, de alguma forma, eles são únicos e, por isso, devem construir tudo a partir do zero! A solução é aceitar que vocês trabalham com as mesmas tecnologias e resolvem os mesmos problemas que milhares de outras pessoas.
- **O reuso acontece por acaso:** trata-se da crença de que o reuso acontece gratuitamente. Como temos visto, é preciso investimento e disciplina para se obter os benefícios!
- **Reuso apenas de código:** acreditar que somente código pode ser reutilizado. Vimos que "ativos" são muito mais abrangentes que apenas código.
- **Reuso depende de tecnologia:** a idéia de que a adoção de determinada tecnologia irá direcionar o reuso. De fato, algumas tecnologias podem trazer níveis elevados de reuso, no entanto, um ativo

é reutilizado se ele possui qualidade e não somente se tiver sido desenvolvido em Java, por exemplo. Determinada tecnologia pode facilitar ou dificultar o reuso, mas nunca impedir ou garantir.

- **Reuso depende de repositório:** acreditar que a adoção de um repositório específico para promover o reuso irá direcionar o reuso na empresa. Sim, as ferramentas auxiliam bastante e suportam o reuso, mas não garantem que ele ocorra!
- **Produzir antes de usar:** desenvolvedores que tentam criar ativos reutilizáveis sem uma experiência significativa acabam criando ativos não-reutilizáveis.
- **Reuso apenas no projeto:** o benefício obtido com a criação de ativos reutilizáveis dentro de um projeto é muito pequeno. O reuso é uma iniciativa multiprojeto!

Tendo em vista tantas barreiras, fica claro que a implantação de reuso com sucesso não acontece por acaso. Os desafios são muitos, são necessários inovação, disciplina e algum investimento. Precisa-se de ferramentas, processos e pessoas capazes de agregar de forma concreta, práticas como componentização e MDA em empresas de tecnologias.

Ferramentas

Embora os benefícios sejam conhecidos, poucas empresas conseguiram emplacar uma estratégia de reuso de software com sucesso e parte dessa dificuldade advém da carência de ambientes integrados que privilegiem o reuso.

Na literatura especializada, assim como na Conferência Anual do Gartner em 2004, são destacados como pontos fundamentais para que as iniciativas de reutilização sejam implementadas com sucesso, que as equipes de projeto estejam conscientes e motivadas a criar e reutilizar componentes de software, que os processos relacionados ao reuso estejam definidos e que a ferramental disponível forneça todo o suporte necessário de forma integrada e colaborativa.

> "As ferramentas de modelagem visual reduzirão a codificação 'em cinco vezes' nos próximos dez anos."
> *Simpósio do Gartner, março/2004*

A busca por produtividade nos ambientes corporativos é notável. As ferramentas têm papel decisivo neste contexto, gerando código, integrando fases e abstraindo camadas.

MDA, por exemplo, mostra-se bastante dependente das ferramentas de modelagem, sobretudo para criação e reutilização de modelos. De olho na oportunidade de acompanhar o bonde MDA, algumas empresas já disponibilizam ferramentas adequadas aos padrões de "arquiteturas guiadas a modelos". Algumas ferramentas de modelagem compatíveis com MDA estão surgindo com bastante força, visando um crescente mercado nesta área. É possível citar algumas ferramentas, maduras ou não, que pretendem fornecer um suporte a esta abordagem de desenvolvimento:

- EMF (Eclipse Modeler Framework) http://www.eclipse.org/emf;
- AndroMDA . Ferramenta open source. (http://www.andromda.org/);
- OptmalJ (http://www.compuware.com/products/optimalj/);
- Rational Software Modeler (http://www.ibm.com/products/br/);
- Borland Together (http://www.borland.com/br/products/together/index.html).

Os benefícios alcançados pela reutilização vêm se materializando e se tornando cada vez mais claros. Por exemplo: se para se desenvolver um determinado componente foram necessárias 200 horas da equipe de desenvolvimento, a cada vez que este é reaproveitado, a empresa tem uma economia de pelo menos parte das 200 horas (alguns fatores de desvio são considerados [Weber, 2002]). Para conseguir o sucesso, é fundamental que os componentes produzidos sejam divulgados e reutilizados sempre que possível, administrar o acervo e utilizar métricas que permitam comprovar a economia e direcionar os novos investimentos. E é neste contexto que a utilização de ferramentas de gestão de componentes podem fazer a diferença.

A respeito de ferramentas de gestão de ativos digitais, podemos afirmar que as ofertas de mercado ainda se encontram em consolidação (poucos players no mercado), o escopo de funcionalidades ainda não está totalmente definido (e por vezes conflitantes com ferramentas de SCM e IDEs) e o mercado nacional ainda se encontra pouco maduro.

Seguem algumas sugestões do que deve ser considerado na avaliação e escolha de ferramentas de gestão de ativos digitais (segundo o Forrester Research):

- **Usabilidade.** São usuários técnicos e de negócio, então deve ser user-friendly;
- **Escalabilidade.** Deve suportar grandes volumes de componentes e artefatos;
- **Processo.** Suportar workflow de aprovação e se integrar ao processo de desenvolvimento das empresas;
- **Integração com ferramentas de desenvolvimento.** Para que os desenvolvedores tenham mais produtividade;
- **Métricas e modelos de ROI (Return of Investment).** Para que os benefícios possam ser medidos de forma objetiva;
- **Indexação e busca.** Uma das principais features, pois é a entrada para todas as demais.

Podemos destacar algumas ferramentas conhecidas nacional e internacionalmente. Nacionalmente, a principal oferta é o DigitalAssets Manager (www.digitalassetsmanager.com.br), que é uma ferramenta para gerenciamento de ativos digitais, que utiliza métricas de RCA (Reuse Cost Avoidance), suporte a diferente tipos de ativos e tecnologias, como componentes, papers, fóruns e notícias. Apresenta integração com IDEs e possibilita a comunicação remota entre implantações, montando uma rede P2P. Internacionalmente, as principais ferramentas são Flashline (http://www.flashline.com) e Logidex Library (http://www.logiclibrary.com), que dividem a liderança no mercado internacional. Ambas provêm gerenciamento de repositórios de componentes de software, apresentam funcionalidades maduras e boa integração com IDEs. Também utilizam métricas de avaliação do reuso. Por outro lado, possuem preço proibitivo para pequenas, médias e boa parte das grandes empresas. Em outras oportunidades, podemos detalhar mais as ferramentas aqui citadas. É importante conhecer o acervo digital e mantê-lo atualizado e visível aos desenvolvedores. Dessa forma, a utilização de alguma ferramenta para gerenciamento de ativos digitais influencia diretamente na eficiência de aplicação de práticas de reuso de software

Métricas

Atualmente, a falta de padronização e a falta de confiabilidade nos valores compõem a principal barreira na utilização de métricas de forma confiável na validação do reuso.

A utilização de métricas agrega valor e torna mais efetiva a implantação de reuso no desenvolvimento de software. Essas métricas oferecem informações que podem ser decisivas sobre a eficiência com a qual o reuso é implantado, e os ganhos obtidos com esta prática. Para determinarmos as métricas de reuso propostas por Poulin [Poulin, 97], alguns conceitos iniciais são importantes:

- RCR (Relative Cost of Reuse): custo de entendimento, integração, testes, etc. A inclusão de componentes de software em um projeto adiciona novos custos, pois envolve a reutilização de ativos que foram produzidos por outras equipes e, possivelmente, para resolver diferentes problemas;
- RCWR (Relative Cost of Writing for Reuse): custo de desenvolver para reuso. Quando se pensa em reuso, deve-se ter em mente que o desenvolvimento também deve ser adaptado. O desenvolvimento de funcionalidades deve ser feito visando à utilização daquele componente por outros projetos, em diferentes contextos. Existem diversas técnicas para aumentar a reusabilidade de um determinado componente como, por exemplo, a utilização de Análise de Domínio (Domain Analysis) e Linhas de Produto de Software (Software Product Lines);
- RSI (Reused Source Instruction): o que deve ser contado como código reutilizável. São contados bibliotecas, frameworks, componentes de negócio, etc. Por outro lado, não são contados os códigos gerados por ferramentas, componentes de caixa branca (código copiado e colado), dentre outros. Normalmente, a métrica mais utilizada como medida de tamanho é a LOC (Lines of Code).

A primeira e mais simples métrica de reuso obedece ao padrão atual da indústria.

$$\%\ \text{Reuso} = \frac{\text{R.S.I.}}{\text{Software Total}} \times 100\%$$

Por exemplo, no desenvolvimento de uma aplicação A, o desenvolvimento foi feito utilizando componentes de negócios já existentes no portfólio da empresa e desenvolvidos em outros projetos. A aplicação foi finalizada com 50 mil linhas de código. Considerando que parte do código trata de problemas específicos, muita coisa precisou ser desenvolvida a partir do zero, porém 35 mil das 50 mil linhas de código da aplicação não foram desenvolvidas na aplicação A, e sim reutilizadas através da aplicação de componentes já desenvolvidos.

Dessa forma, o percentual de reuso da aplicação A foi de:

$$\%\ \text{Reuso A} = \frac{35\ \text{kLOCS} \times 100\%}{50\ \text{kLOCS}} = 70\%$$

O exemplo apresentado valida a fórmula básica de cálculo de reuso, porém para que possamos utilizar um padrão de métrica para componentes com características específicas, é necessário estabelecer um padrão de ajuste para tornar o índice de reuso mais real. Podemos estabelecer um fator de correção, de acordo com o "grau de reuso", e isso pode ser aplicado para todas as categorias de componente. A tabela 1 exibe um exemplo para componentes de serviço e componentes de Backoffice.

Tipo de Componente	Reuso básico	Reuso médio	Reuso completo
Serviços	25%	60%	100%
Backoffice	30%	60%	90%

Tabela 1. Exemplo de "graus de reuso" relacionados a diferentes níveis de reuso.

Utilizando o fator de ajuste, podemos usar a seguinte métrica para o percentual de reuso:

$$\%\ \text{Reuso} = \frac{\text{R.S.I.} \times (1 - \text{RCR}) \times \text{Fator de ajuste} \times (\$\text{LOC})}{\text{Software Total}}$$

Os dados relacionados ao RCR e RSI são utilizados na métrica de reuso para calcular o RCA (Reuse Cost Avoidance). Além do RCA, existem outras métricas para cálculo do ROI (Return of Investment) que é o valor total do benefício financeiro oferecido pelo reuso aplicado em um determinado projeto. O ROI também pode ser obtido através de métricas simples, como apresentado em [Poulin, 06].

$$\text{RCA} = \text{R.S.I.} \times (1 - \text{RCR}) \times \text{Fator de ajuste} \times (\$/\text{LOC})$$

Onde $/LOC é o valor considerado por linha de código da aplicação.

Esta fórmula de RCA é utilizada por algumas ferramentas comerciais, como o Digital Assets Manager, citado no tópico anterior, para avaliação do ganho obtido na utilização de cada componente cadastrado e gerenciado na ferramenta.

Enfim, os benefícios alcançados pela reutilização vêm se materializando e se tornando cada vez mais claros. Vimos que, se para se desenvolver um determinado componente foram necessárias 200 horas da equipe de desenvolvimento, a cada vez que este é reaproveitado, a empresa tem uma economia de pelo menos parte das 200 horas (vimos também que alguns fatores de desvio são considerados). Para conseguir o sucesso, é fundamental divulgar e reutilizar os componentes produzidos sempre que possível, administrar o acervo e utilizar métricas que permitam comprovar a economia e direcionar os novos projetos.

Caso de utilização e avaliação de impacto do reuso

Nacionalmente, não temos muitos casos de sucesso publicados sobre implantação do reuso de software em empresas, e vale a pena citarmos um exemplo. A seguir, apresentamos a avaliação do impacto do reuso no processo de desenvolvimento na Ci&T Software (www.cit.com.br), através de dados reais. Na Ci&T, o resultado da aplicação da componentização atingiu ganhos representativos nos projetos desenvolvidos. Neste contexto, considera-se ganhos em produtividade, economia e qualidade, devido à utilização e adaptação de todo o processo de desenvolvimento em torno do reuso.

Através das métricas utilizadas (RCA), é possível analisar o ganho em economia obtido. A tabela 2 apresenta os dados reais relacionados a componentes utilizados em um projeto da empresa.

Vale observar que esta análise foi feita baseada em linhas de código e que foram aplicados fatores de ajuste. Aplicando-se a fórmula básica (sem ajuste), o percentual de reuso calculado é:

$$\%\ \text{Reuso B} = \frac{105\ \text{kLOCS} \times 100\%}{105 + 24\ \text{kLOCS}} = 81\%$$

Componente	Fator de ajuste	LOCs	Ajustado
CAF 3.0 (mecanismos)	12%	22.000	2.640
CAFX 1.0	30%	5.000	1.500
eFlow 1.2	30%	37.000	11.100
PageBrowser 2.0	30%	5.000	1.500
FWSecurity 2.2	30%	36.000	10.800
Totais		**105.000**	**27.540**
LOCs do projeto			24.000
% Reuso			**53%**

Tabela 2. Dados reais de um projeto B, desenvolvido na Ci&T.

Através dos dados da tabela 2 também podemos avaliar o Reuse Cost Avoidance do projeto, em que

- R.S.I. = 105 kLOCS
- RCR = 20%
- $/LOC = R$ 1,00 / LOC (considerando um real por linha de código).
- Fator de reajuste do projeto = 26,2% (média ponderada do fator de reajuste de cada componente, considerando o tamanho de cada componente como peso).

Temos:

RCA proj.B = 105.000 x (1 – 0,2) x 0,262 x (R$ 1,00) = R$ 22.008,00

Portanto, podemos avaliar objetivamente alguns dados do projeto B:

- o percentual de reuso (calculado sem ajuste) foi de 81%;
- o percentual de reuso real foi de 53%;
- o RCA foi de R$22.000,00.

Além do ganho com produtividade, a qualidade também é afetada positivamente, considerando-se o número de defeitos por linha de código. A qualidade foi sensivelmente impulsionada pela utilização de soluções prontas e adaptação dos processos para o desenvolvimento com reuso de componentes.

Se relacionados aos dados indicados pelas pesquisas sobre empresas com CMM4 e 5, dos Estados Unidos e da Índia, observamos um quociente bastante satisfatório. Segundo um benchmarking de mercado realizado pelo SEI (Software Engineering Institute) sobre defeitos pós-implantação, a média de defeitos pós-implantação nestas empresas é entre 0,2 e 0,4 defeito/KLOC.

Com a iniciativa de reuso, a Ci&T gera, em média, 25% de economia em seus novos projetos de desenvolvimento. Ou seja, quando um projeto é iniciado, cerca de 25% do esforço necessário para construí-lo a partir do zero é economizado através da reutilização de partes de software pronto – os componentes. Essa economia reflete-se diretamente na competitividade da empresa e boa parte da estratégia é fundamentada na ferramenta de gestão adotada. Além disso, investindo em reuso, a empresa obteve uma série de benefícios, como a geração de ativos, com mais de 80 componentes construídos (8 registros no INPI), aumento da produtividade de 15% ao ano e economia gerada em torno de R$1,5 milhões. Ultimamente, diversas pesquisas têm abordado os temas de componentização, SOA e MDA, e exemplos como este, de implantação e avaliação de reuso, são úteis para validar os conceitos e benefícios do reuso de software na prática.

Saber mais

- OMG: http://www.omg.org
- MDA: http://www.omg.org/mda
- Software Engineering Institute (SEI). http://www.sei.cmu.edu
- Ci&T Labs: http://labs.cit.com.br
- Gartner Group: o evento anual do Gartner de 2005 foi realizado nos dias 30 e 31 de março, em São Paulo. (Mais informações em http://www.gartner.com/2_events/conferences/2005/brl22l/brl22l.jsp)

Considerações Finais

Muito se vê sobre reuso em apresentações de workshops, eventos científicos, venda de produtos, etc. Porém, seus benefícios práticos são muitas vezes obscuros ou subjetivos. Neste artigo, pudemos apresentar alguns conceitos e tecnologias importantes relacionados a reuso de software, assim como métricas utilizadas para avaliar o impacto sobre a produtividade e economia da empresa e, por fim, apresentamos um case de implantação real de reuso em uma empresa de TI.

Em artigos futuros, pretendemos apresentar mais detalhadamente cada uma das tecnologias aqui citadas, dando maior visibilidade às promessas e capacidades dos entusiastas, que vêem o reuso como um trampolim em produtividade e qualidade de software, focados em satisfazer a necessidade de se produzir mais e melhor em um tempo cada vez mais curto, e cabe a nós aceitar que, atualmente, esta é a realidade nas fábricas de software• MJ

Referências

- *Ambler, S. W. Nalbone, J.. Vizidos, M. J. "The Enterprise Unified Process: Extending the Rational Unified Process", Printice Hall, 2005.*
- *Bacili, K. R.; "Gestão de Portifólio de Componentes", IBTA – Pós graduação Engenharia de Software – Setembro de 2005*
- *Barry B., "Managing Software Productivity and Reuse", IEEE Computer, vol. 32, nº. 9, pp. 111-113, Sept., 1999.*
- *D´Souza, D. F.; Wills, A. C. "Objects, Components, and Frameworks with UML, The Catalysis Approach", Addison-Wesley, USA, 1999.*
- *Gonçalves, E. M.; "Metodologias de Representação de Ativos Digitais e Componentes de Software", Relatório de pesquisa (IACS) - Ci&T Labs (http://labs.cit.com.br) - 22 de Março de 2006.*
- *Jacobson, L.; Griss, M. Jonsson, P. "Software Reuse: Architecture, Process and Organization for Business Success", Addison-Wesley, Longman, 1997*
- *Krueger, C. W., "Software Reuse", ACM Computing Surveys, Vol. 24, Nº. 02, pp. 131-182, 1992.*
- *Lanusse, Andreano. (Borland) "Maximize the Business Value of Software - Model Driven Architecture", Wire – Recife/PE - fevereiro de 2006.*
- *OMG, Reusable Asset Specification. http://www.omg.org/technology/documents/formal/ras.htm. (visitado em: 10/05/2006)*
- *Oliveira, M. S.; Garcia, I. C.; Nunes, A. "Resource Discovery em uma arquitetura Peer-to-Peer aplicada a distribuição e compartilhamento de componentes de software". I Workshop de Peer-to-Peer – XXIII Simpósio Brasileiro de Redes de Computadores, Fortaleza - Maio de 2005.*
- *Oliveira, M. S., "Promovendo e potencializando práticas de componentização através de um Laboratório de Inovação Tecnológica". IXSemana de Informática – Universidade Federal de Viçosa –Agosto de 2004.*
- *Poulin, Jeffrey S., "Measuring Software Reuse: Principles, Practices, and Economic Models". Addison-Wesley (ISBN 0-201-63413-9), Reading, MA, 1997.*
- *Poulin, Jeffrey S., "Metrics for Object-Oriented Reuse". http://www.flashline.com/Content/poulin/reuse Metrics.jsp. (visitado em: 10/05/2006)*
- *Weber, K. C.; Nascimento, C. J.; "Brazilian Software Quality 2002". The 24th IEEE International Conference on Software Engineering (ICSE), EUA, pp. 634-638, 2002.*