engenharia de software magazine

DevMedia group Ano I - Edição 05

Planejamento
Alguns Cuidados que Devem Ser Observados na Aplicação da Análise de Pontos de Função

Verificação, Validação & Teste
Melhores Práticas na Automação de Testes

Usabilidade de Software

Conheça alguns de seus fundamentos e algumas técnicas para sua avaliação

Metodologias Ágeis
Analisando o Desenvolvimento de Software Orientado a Aspectos no contexto de Extreme Programming

Processo
Modelagem de Processos de Negócio com NetBeans

ISSN 1983-27-7

Aulas desta edição:

- Introdução à Engenharia de Requisitos - Parte 13
- Introdução à Engenharia de Requisitos - Parte 14
- Introdução à Engenharia de Requisitos - Parte 15
- Introdução à Engenharia de Requisitos - Parte 16
- Introdução à Engenharia de Requisitos - Parte 17
- Introdução à Engenharia de Requisitos - Parte 18
- Introdução à Construção de Diagramas de Classes da UML - Parte 1
- Teste Unitário com Objetos Mock
- Teste em Bancos de Dados com DBUnit
- Refatoração de código no Eclipse

Requisitos
Alguns Exemplos Prontos de Casos de Uso que Você Pode se Basear ao Especificar seus Requisitos

Gestão de TI
Gestão de Serviços Operacionais Aplicada à TI

Conhecimento faz diferença!

Mais de 60 mil downloads na primeira edição!

Faça já sua assinatura digital! | *www.devmedia.com.br/es*

Faça um *up grade* em sua carreira.

Em um mercado cada vez mais focado em qualidade ter conhecimentos aprofundados sobre requisitos, metodologia, análises, testes, entre outros, pode ser a diferença entre conquistar ou não uma boa posição profissional. Sabendo disso a DevMedia lança para os desenvolvedores brasileiros sua primeira revista digital totalmente especializada em Engenharia de Software. Todos os meses você irá encontrar artigos sobre Metodologias Ageis; Metodologias tradicionais (*document driven*); ALM (*application lifecycle management*); SOA (aplicações orientadas a servicos); Analise de sistemas; modelagem; Métricas; orientação à objetos; UML; testes e muito mais. **Assine já!**

DevMedia GROUP

engenharia de software magazine

Ano 1 - 5ª Edição 2008 **Impresso no Brasil**

Atendimento ao Leitor

A DevMedia conta com um departamento exclusivo para o atendimento ao leitor. Se você tiver algum problema no recebimento do seu exemplar ou precisar de algum esclarecimento sobre assinaturas, exemplares anteriores, endereço de bancas de jornal, entre outros, entre em contato com:

Carmelita Mulin – Atendimento ao Leitor
www.devmedia.com.br/central/default.asp
(21) 2220-5375

Kaline Dolabella
Gerente de Marketing e Atendimento
kalined@terra.com.br
(21) 2220-5375

Publicidade

Para informações sobre veiculação de anúncio na revista ou no site entre em contato com:

Kaline Dolabella
publicidade@devmedia.com.br

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique a vontade para entrar em contato com os editores e dar a sua sugestão!

Se você estiver interessado em publicar um artigo na revista ou no site SQL Magazine, entre em contato com os editores, informando o título e mini-resumo do tema que você gostaria de publicar:

Rodrigo Oliveira Spínola - Colaborador
editor@sqlmagazine.com.br

EDITORIAL

"Um produto, seja um sistema de software ou outro qualquer, possui usabilidade que é um dos atributos da qualidade perceptível aos usuários. A usabilidade é uma característica de um produto que informa quão fácil de usar e aprender esse produto é. Em outras palavras, serve como um indicador de quão intuitivo é utilizar aquele produto. Essa característica é determinante no sucesso desse produto, pois ela influencia diretamente o interesse do usuário em utilizar ou não esse produto ou sistema (de software). Para tanto, em sistemas de software, assim com em outros produtos, a usabilidade é avaliada durante o processo de desenvolvimento visando assegurar o nível desejado de usabilidade."

Neste contexto, a Engenharia de Software Magazine destaca nesta edição duas matérias com foco em usabilidade. Em uma delas, são discutidos os principais conceitos sobre usabilidade como esta pode ser inserida em um processo de desenvolvimento de software.

Em complemento a esta matéria, o segundo artigo apresenta um conjunto de definições associadas à avaliação da usabilidade da interface de usuário de sistemas de software ou outros produtos. Neste cenário, é apresentado um conjunto de métodos de avaliação de usabilidade, classificados como testes, pesquisa e inspeção.

Além destas duas matérias, esta quinta edição traz mais seis artigos:

- Analisando o Desenvolvimento de Software Orientado a Aspectos no contexto de Extreme Programming;
- Especificação de Requisitos com Casos de Uso;
- Cuidados na Aplicação da Análise de Pontos de Função no Relacionamento entre Usuário e Desenvolvedores;
- Melhores Práticas na Automação de Testes;
- Modelagem de Processos de Negócio com Uso do NetBeans;
- Gestão de Serviços Operacionais Aplicada à TI.

Desejamos uma ótima leitura!
Equipe Editorial Engenharia de Software Magazine

Rodrigo Oliveira Spínola
rodrigo@sqlmagazine.com.br
Doutorando em Engenharia de Sistemas e Computação (COPPE/UFRJ). Mestre em Engenharia de Software (COPPE/UFRJ, 2004). Bacharel em Ciências da Computação (UNIFACS, 2001). Colaborador da Kali Software (www.kalisoftware.com), tendo ministrado cursos na área de Qualidade de Produtos e Processos de Software, Requisitos e Desenvolvimento Orientado a Objetos. Consultor para implementação do MPS.BR. Atua como Gerente de Projeto e Analista de Requisitos em projetos de consultoria na COPPE/UFRJ. É Colaborador da Engenharia de Software Magazine.

Marco Antônio Pereira Araújo
(maraujo@devmedia.com.br)
É Doutorando e Mestre em Engenharia de Sistemas e Computação pela COPPE/UFRJ – Linha de Pesquisa em Engenharia de Software, Especialista em Métodos Estatísticos Computacionais e Bacharel em Matemática com Habilitação em Informática pela UFJF, Professor dos Cursos de Bacharelado em Sistemas de Informação do Centro de Ensino Superior de Juiz de Fora e da Faculdade Metodista Granbery, Analista de Sistemas da Prefeitura de Juiz de Fora. É editor da Engenharia de Software Magazine.

Eduardo Oliveira Spínola
(eduspinola@gmail.com)
É Editor das revistas Engenharia de Software Magazine, SQL Magazine, WebMobile. É bacharel em Ciências da Computação pela Universidade Salvador (UNIFACS) onde atualmente cursa o mestrado em Sistemas e Computação na linha de Engenharia de Software, sendo membro do GESA (Grupo de Engenharia de Software e Aplicações).

Caro Leitor,
Para esta quinta edição, temos um conjunto de 10 vídeo aulas. Estas vídeo aulas estão disponíveis para download no Portal da Engenharia de Software Magazine e certamente trarão uma significativa contribuição para seu aprendizado. A lista de aulas publicadas pode ser vista abaixo:

Tipo: Verificação, Validação e Teste
Autor: Marco Antônio Pereira Araújo
Título: Teste Unitário com Objetos Mock
Mini-Resumo: Esta vídeo aula apresenta a aplicação de objetos Mock em testes unitários com JUnit, com o objetivo de substituir classes e métodos ainda não implementados, proporcionando a realização de testes unitários.

Tipo: Verificação, Validação e Teste
Autor: Marco Antônio Pereira Araújo
Título: Teste em Bancos de Dados com DBUnit
Mini-Resumo: Esta vídeo aula apresenta a aplicação de testes unitários em bancos de dados com DBUnit, uma vez que o estado de bases de dados é fundamental na condução de testes de unidade automatizados.

Tipo: Projeto
Autor: Rodrigo Oliveira Spínola
Título: Introdução à Construção de Diagrama de Classes da UML – Parte 1
Mini-Resumo: Esta vídeo aula apresenta as definições iniciais envolvidas na construção de diagramas de classes da UML.

Tipo: Projeto
Autor: Marco Antônio Pereira Araújo
Título: Refatoração de código no Eclipse
Mini-Resumo: Esta vídeo aula apresenta a utilização de técnicas de refatoração na IDE Eclipse com o intuito de manter a qualidade do código fonte de aplicações Java.

Tipo: Engenharia de Requisitos
Autor: Rodrigo Oliveira Spínola
Título: Introdução à Engenharia de Requisitos - Parte 13
Mini-Resumo: Esta vídeo aula é parte do curso de introdução à engenharia de requisitos. Nesta décima terceira parte daremos início à parte mais prática do curso. Nesta aula apresentaremos uma ferramenta case para elaboração de diagramas de casos de uso e daremos início a um estudo de caso para elaboração de um diagrama de casos de uso para um sistema de locadora de veículos.

Tipo: Engenharia de Requisitos
Autor: Rodrigo Oliveira Spínola
Título: Introdução à Engenharia de Requisitos - Parte 14
Mini-Resumo: Nesta décima quarta parte finalizaremos nosso primeiro estudo de caso referente à elaboração de um diagrama de casos de uso para um sistema de locadora de veículos.

Tipo: Engenharia de Requisitos
Autor: Rodrigo Oliveira Spínola
Título: Introdução à Engenharia de Requisitos - Parte 15
Mini-Resumo: Esta vídeo aula é parte do curso de introdução à engenharia de requisitos. Nesta décima quinta parte apresentaremos um estudo de caso para elaboração de um diagrama de casos de uso para um sistema de vendas de passagem aérea.

Tipo: Engenharia de Requisitos
Autor: Rodrigo Oliveira Spínola
Título: Introdução à Engenharia de Requisitos - Parte 16
Mini-Resumo: Nesta décima sexta parte, finalizaremos o estudo de caso para elaboração de um diagrama de casos de uso para um módulo de gestão de usuários de um sistema de informação fictício.

Tipo: Engenharia de Requisitos
Autor: Rodrigo Oliveira Spínola
Título: Introdução à Engenharia de Requisitos - Parte 17
Mini-Resumo: Esta vídeo aula é parte do curso de introdução à engenharia de requisitos. Nesta décima sétima parte apresentaremos um estudo de caso para elaboração de um diagrama de casos de uso para um módulo de gestão de usuários de um sistema de informação fictício.

Tipo: Engenharia de Requisitos
Autor: Rodrigo Oliveira Spínola
Título: Introdução à Engenharia de Requisitos - Parte 18
Mini-Resumo: Nesta décima oitava parte, finalizaremos o estudo de caso para elaboração de um diagrama de casos de uso para um módulo de gestão de usuários de um sistema de informação fictício.

ÍNDICE

kali software

Desde 2004 Trazendo Teoria para a Prática

Treinamento e Consultoria em Engenharia de Software

Profissionais com ampla experiência prática (Consusltores Certificados) e acadêmica (Professores Universitários, Mestres e Doutores)

Know-how para implementar processos de software de acordo com modelos de qualidade (MPS e CMMI) e maximizar o retorno de investimento

Cursos 2008: *Inscrições Abertas*

Certificados emitidos pela Kali Software

Outubro | Novembro	Testes de Software – **16 horas**
Outubro | Novembro	Projeto Orientado a Objetos com UML e Padrões – **24 horas**
Novembro	Engenharia de Requisitos – **24 horas**
Novembro | Dezembro	Linguagem Java com ênfase na certificação SCJP da Sun – **32 horas**

Confira nossos cursos in-company, elaborados de acordo com as necessidades de treinamento de sua empresa
Confira as ementas, investementos e condições de pagamento: *www.kalisoftware.com*
Inscrição e outras informações: *info@kalisoftware.com*

kalisoftware.com | info@kalisoftware.com

Extreme Programming

Analisando o Desenvolvimento de Software Orientado a Aspectos no contexto de Extreme Programming

Rafael do Espírito Santo
rafael.santo@ufrj.br
Estudante de Ciência da Computação da Universidade Federal do Rio de Janeiro (UFRJ).
Ex-membro da Empresa Júnior de Consultoria em Microinformática da UFRJ (EJCM).
Atua no desenvolvimento de sistemas em projetos de consultoria na COPPE/UFRJ

Rodrigo Pereira dos Santos
rps@cos.ufrj.br
É bacharel em Ciência da Computação formado na Universidade Federal do Lavras. Atualmente cursa o mestrado na linha de Engenharia de Software pela COPPE/UFRJ.

Paulo Sérgio Medeiros dos Santos
pasemes@cos.ufrj.br
É bacharel em Ciência da Computação formado na Universidade Federal do Rio de Janeiro. Atualmente cursa o mestrado na linha de Engenharia de Software pela COPPE/UFRJ onde atua, principalmente, nas áreas de serviços Web, integração de ferramentas, gestão desconhecimento e engenharia de software experimental. Possui 7 anos de experiência em desenvolvimento de software sendo 5 anos trabalhando com Java e 2 anos com JavaServer Faces.

De que se trata o artigo?
Este artigo apresenta como os conceitos de programação orientada a aspectos influenciam a metodologia XP para desenvolvimento de software.

Para que serve?
Indicar pontos positivos e riscos associados do uso destas abordagens em conjunto.

Em que situação o tema é útil?
Para aqueles que tenham interesse em trabalhar com POA no contexto de XP e saber quais são os benefícios e principais riscos do uso destas abordagens em conjunto.

Considerando que *time-to-market* e volatilidade de requisitos de software compreendem fatores cruciais para a criação de valor para o processo de desenvolvimento de software, cada vez mais novas estratégias são pesquisadas a fim de atender satisfatoriamente os diferentes pontos de vista dos *stakeholders* envolvidos [Boehm, 2006]. Neste cenário, destacam-se os processos de desenvolvimento baseados em metodologias ágeis [Shore e Warden, 2007] [Beck, 2000], os quais focam em comunicação constante, projeto minimalista e codificação simplificada. Por essas características, tais processos têm despertado o interesse de empresas e de grupos de pesquisa, com destaque para o *Extreme Programming* (XP), devido às práticas e aos valores adotados durante o desenvolvimento, tais como integração contínua, programação em pares, refatoração de código e programação baseada em testes (*Test-Driven Development* – TDD) [Paulk, 2000]. A estrutura do processo XP se baseia nos princípios ágeis, que contemplam melhorias nas relações com os clientes e entre os membros da equipe, realçando o ambiente e os recursos disponíveis na organização e buscando "quebrar" problemas grandes em um conjunto de vários sub-problemas, com o intuito de melhorar a qualidade das soluções [Teles, 2005].

Por outro lado, um dos fatores que impactam o desenvolvimento de software

está na escolha adequada de metodologias e tecnologias durante a etapa de planejamento de um projeto, sobretudo perante o aumento da complexidade dos domínios de aplicação [Travassos *et al*., 2001]. Isso tem levado empresas e grupos de pesquisa a voltarem a sua atenção para o desenvolvimento e adoção de tecnologias que auxiliem o processo de desenvolvimento. Particularmente em XP, tem-se buscado por tecnologias que melhorem a elaboração e organização de produtos de software e que aumentem a produtividade dos desenvolvedores, evitando desviar a sua atenção dos reais objetivos do negócio do cliente, representados pelos requisitos funcionais – documentados nas chamadas *histórias* em XP. Nessa linha, o Desenvolvimento de Software Orientado a Aspectos (DSOA) surge como uma abordagem que objetiva ampliar a reusabilidade e a manutenibilidade de software, ao se basear no isolamento de certas funcionalidades em unidades encapsuladas [Elrad *et al*., 2001].

Na concepção de Kiczales *et al.* (1997), nem as técnicas de programação orientada a objetos (POO) nem as técnicas da programação procedural são suficientes para implementar com clareza importantes decisões do projeto. A Orientação a Aspectos (OA) estende a orientação a objetos (OO) com o intuito de tratar o entrelaçamento e o espalhamento de certos requisitos (interesses transversais ou *crosscutting concerns*) [Filman *et al*., 2004]. No entanto, apesar das melhorias almejadas, alguns pontos podem dificultar a reutilização e a manutenção, tais como a inversão de dependência e a inconsciência [Santos *et al*., 2007], fato este decorrente do estágio atual do DSOA rumo à aplicabilidade em projetos reais e na indústria, visando seu amadurecimento.

Dessa forma, faz-se necessária uma análise acerca de quais são os possíveis impactos positivos e negativos que uma abordagem como DSOA provocaria em ambientes reais (empresas de pequeno e médio porte), especialmente naqueles que adotam os princípios e práticas de XP, onde isso se torna fundamental, dado que esse processo se baseia na agilidade e não é dirigido por planos. Este trabalho tem por objetivo analisar características (valores, princípios e práticas) de XP de forma a confrontá-las com algumas questões críticas relativas ao DSOA, o que representa um primeiro passo para se pensar em avaliar a aplicabilidade de DSOA em diferentes contextos, buscando por possíveis impactos positivos e/ou negativos.

Conhecendo XP

Dado que mudanças, comunicação e pessoas representam fatores inerentes ao desenvolvimento de software de qualidade [Sato, 2007], o final da década de 1990 presenciou o surgimento das metodologias ágeis. O desenvolvimento ágil foi criado com o intuito de melhorar o desempenho de projetos de software ao buscar a diminuição da ineficiência, da burocracia de métodos dirigidos a planos e de qualquer outra coisa que não adicione valor ao produto de software [Tichy, 2004]. O conceito de agilidade ganhou destaque a partir do **Manifesto do Desenvolvimento Ágil de Software** [Beck *et al*., 2001], o qual aponta quatro valores principais: (i) *indivíduos e interações vêm antes de processos e ferramentas*; (ii) *software funcionando vem antes de documentação abrangente*; (iii) *colaboração do cliente vem antes de negociação de contrato*; e (iv) *resposta à modificação vem antes do plano em andamento*.

Dentre as metodologias ágeis mais utilizadas, destaca-se o *Extreme Programming* (XP), atribuído a Kent Beck, Ron Jeffries e Ward Cunningham [Beck, 2000]. XP tem como premissa técnica a combinação de: (i) inserção do cliente no projeto; (ii) liberações de pequenos incrementos ao longo das iterações no desenvolvimento; (iii) projeto simples; (iv) programação em pares; (v) refatoração; e (vi) integração contínua, visando uma curva mais branda do custo de mudanças com relação ao tempo [Beck, 2000]. XP apresenta maior aplicabilidade em pequenos projetos, de risco relativamente baixo, envolvendo pessoal altamente capacitado e requisitos passíveis de mudanças [Boehm, 2006].

No entanto, estudos mostram que é possível aplicar XP em projetos de maior porte, através de extensões que enfatizem *feedback* constante, projeto de software e criação de novos papéis no ciclo de desenvolvimento [Lipert *et al*., 2003] [Eckstein, 2006]. Estas extensões foram aplicadas em projetos de diferentes perspectivas (*web*, desktop, processamento em *batch*, dispositivos móveis), localização (localizadas dentro das instalações do cliente ou em ambientes externos), número de integrantes e qualificação dos profissionais envolvidos, e tem como objetivo prover meios de lidar com projetos de domínio de negócio complexo sem limitar as vantagens oferecidas pelo processo de desenvolvimento ágil de software.

A base de XP está sedimentada sobre quatro valores: (i) *comunicação*; (ii) *simplicidade*; (iii) *feedback*; e (iv) *coragem* [Beck e Andres, 2004]. Além destes, recentemente observou-se a importância de se considerar um novo valor, o *respeito*. XP preza pela comunicação constante entre os membros da equipe e destes com os clientes, e pelo desenvolvimento do sistema focado no atendimento às necessidades do cliente da maneira mais simples possível. O processo XP enfatiza que o *feedback* agrega três perspectivas: do sistema (através da utilização constante de testes de unidade), do cliente (por meio de testes de aceitação) e da equipe de projeto (através da comunicação constante). Por outro lado, a coragem reforça o fato de que as tarefas não devem ser adiadas, ou seja, devem ser desempenhadas o quanto antes, colaborando para que os desenvolvedores se sintam mais confortáveis com a refatoração do código. Por fim, o respeito se mostra, por exemplo, por meio do respeito mútuo entre os membros da equipe, com o objetivo de evitar a desvalorização do trabalho de qualquer membro da equipe de trabalho (e nem prejudicá-lo) e almejar uma equipe mais confiante e motivada.

Beck e Andres (2004) afirmam que os cinco valores representam critérios que visam uma solução de sucesso para projetos de software. Entretanto, eles colocam que esses valores são vagos demais para auxiliarem na escolha das práticas que suportam XP e apontam

vários princípios concretos derivados desses valores. Além disso, XP se utiliza de determinadas práticas que, se bem organizadas, oferecem um reforço mútuo com o intuito de se complementarem [Beck e Andreas, 2004]:

• **Jogo do planejamento**: consiste na rápida determinação do escopo da próxima *release* (versão), combinando as prioridades de negócio com as estimativas técnicas. O cliente define o escopo, a prioridade e as datas de uma perspectiva de negócios, enquanto a equipe técnica estima e rastreia o progresso;

• **Pequenas versões:** visa colocar um sistema simples em operação rapidamente, ou seja, liberar novas versões em um tempo muito curto, normalmente duas semanas;

• **Histórias (metáfora):** pretendem guiar todo o desenvolvimento por meio de uma história simples e compartilhada de como funciona o sistema como um todo;

• ***Design* simples (ou projeto minimalista):** busca manter o projeto o mais simples possível, a qualquer momento;

• **Teste**: alerta para que desenvolvedores escrevam continuamente testes de unidade que devem ser executados com a maior precisão possível. Além disso, os clientes devem realizar o teste de aceitação para demonstrar que as funcionalidades requeridas foram contempladas (i.e., "testar, depois codificar");

• **Refatoração de código:** implica na reestruturação do sistema sem modificar o seu comportamento, para remover duplicações, melhorar a comunicação, simplificar ou adicionar flexibilidade;

• **Programação em par**: reflete o fato de que todo o código produzido deve ser escrito por duplas de programadores;

• **Código coletivo (propriedade coletiva):** realça que qualquer desenvolvedor pode melhorar qualquer código do sistema, a qualquer momento e em qualquer lugar;

• **Integração contínua:** consiste em integrar e construir o sistema várias vezes ao dia (i.e., toda vez que uma tarefa é concluída). Nesse sentido, testes de regressão contínuos previnem a ocorrência de erros na implementação das funcionalidades quando os requisitos mudam;

• **40-horas por semana (folga):** destaca a importância da carga horária de trabalho não ultrapassar 40 horas por semana, sempre que possível, a fim de evitar o estresse dado que XP busca ser um processo altamente disciplinado. Horas-extras são permitidas desde que não ultrapassem duas semanas seguidas e foquem na conclusão das histórias correspondentes a uma dada iteração;

• **Cliente presente:** busca ter um cliente (usuário real) integrado à equipe durante o desenvolvimento do projeto, de forma que seu papel seja avaliar e mediar as funcionalidades implementadas.

• **Padrões de codificação:** objetiva a definição de regras que enfatizam a legibilidade e que permitam a comunicação entre os desenvolvedores através do código (padrões sintáticos e semânticos, comentários, etc.), de forma que o código seja visto como a própria documentação do produto de software.

Dessa forma, XP busca explicitar práticas conhecidas pelos *stakeholders* do desenvolvimento de software através de propriedades emergentes que, uma vez utilizadas em conjunto, adicionam valor e produtividade ao processo de desenvolvimento, conforme demonstrado em entrevistas realizadas com profissionais da área de Engenharia de Software (pesquisadores e analistas) [Santos, 2008].

Porém, nada impede que empresas adotem um pequeno conjunto de práticas de XP focadas em resolver os problemas de maior impacto no desenvolvimento de software, desde que a equipe assuma a responsabilidade pela flexibilização do processo [Paulk, 2001]. Por outro lado, Beck (2000) afirma que a aplicação das práticas de XP pode gerar lições aprendidas que auxiliam os desenvolvedores a dominarem as práticas, mas permitindo sua modificação caso necessário.

Estudos já realizados

Procurando abranger mais características de XP e suas relações com questões de DSOA, Kircher *et al*. (2002) realizam um estudo sobre os impactos da adoção de DSOA em XP, concentrando-se apenas no levantamento de influências positivas de DSOA que pudessem ser confrontadas com características de XP. De forma sintetizada, os autores entendem que a *comunicação* é influenciada de forma positiva enquanto os valores *coragem* e *feedback* não sofrem interferência direta pela adoção de DSOA. Quanto ao valor simplicidade, os autores concordam com a existência de pontos positivos, mas não questionam a existência de possíveis pontos negativos. Considerando os princípios de XP, notou-se que somente alguns deles foram abordados diante de questões de DSOA. Já nas práticas de XP, os autores afirmam que a *programação em pares* não apresenta impacto direto com relação à adoção de DSOA, enquanto a *publicação de versões curtas* é influenciada positivamente, devido à possibilidade de integração das diferentes versões por meio de um conjunto de aspectos. Para as demais práticas, não houve um questionamento sobre os possíveis impactos negativos acarretados pela adoção de DSOA. No entanto, a análise realizada não contempla muitas características de XP e não se mostra profunda em seu tratamento.

Por fim, os autores concluem que certas medidas devem ser adotas a fim de facilitar a integração entre DSOA e XP, tais como treinamento e uso intenso de Programação Orientada a Aspectos (POA), além do uso de ferramentas que auxiliem na identificação dos aspectos utilizados no produto de software.

Características de XP diante de questões críticas de DSOA

Nesta seção, serão analisadas algumas características do processo XP (valores, princípios e práticas), visando entender os possíveis impactos de algumas questões críticas de DSOA, com base nos trabalhos citados nas seções anteriores bem como a experiência dos autores no tema abordado. A seguir, cada subseção destaca uma característica que pode ser afetada pela adoção de DSOA em projetos que aplicam o processo XP. Para algumas características de XP (valores – comunicação, respeito e *feedback*; princípios – benefício mútuo, economia, diversidade, falha, fluidez, humanismo, oportunidades, reflexão e responsabilidade aceita; práticas – *build*

de dez minutos, ciclo semanal, ciclo semestral, equipe integral, folga, histórias, trabalho energizado, cliente presente, e programação em par) que não são referidas neste artigo, nenhuma relação direta entre DSOA e XP foi identificada durante a análise realizada.

Coragem

Observa-se que, no início do processo de desenvolvimento, equipes que adotam XP podem ter o grau desse valor diminuído, devido ao desconhecimento das abstrações da OA influenciaria negativamente no valor da simplicidade. Alguns fatores que interferem nessa diminuição consistem na possível inexperiência dos membros da equipe com programação orientada a aspectos (POA), nas dificuldades de entendimento e depuração do código, no princípio da inconsciência e na forma como a combinação (*weaving*) é realizada. Por exemplo, em POA, durante o desenvolvimento dos aspectos em si, a criação dos *pointcuts* deve ser realizada com cautela, a fim de verificar que somente as classes desejadas estejam sendo interceptadas pelo aspecto em questão. Por outro lado, após o período de adaptação com o DSOA, as equipes se sentem mais seguras, pois passam a desenvolver trechos de código melhor organizados e com menos tendências a erros. Com isso, novas funcionalidades (ou requisitos não funcionais) podem ser inseridas no produto de software sem que haja alteração significativa no código principal, impactando positivamente na adoção de DSOA em XP.

Simplicidade e Melhoria

As novas abstrações introduzidas pela OA e as dificuldades de se identificar requisitos que devem ser implementados com POA levanta a necessidade de uma etapa mínima de análise, focada na extração de aspectos a partir das histórias de XP. No entanto, dado que XP foca na etapa de implementação, em detrimento da produção de documentação extensiva de análise e projeto, a adoção da OA influenciaria negativamente no valor da simplicidade. Por outro lado, se bem realizado, o processo de identificação de aspectos permite melhorar o desenvolvimento, mediante a separação de interesses do sistema, e tornar o código tradicional (OO puro) estruturado, minimizando o espalhamento e o entrelaçamento de requisitos, e impactando o valor de melhoria positivamente, ao serem buscadas novas oportunidades de melhorar a organização do produto de software.

Auto-semelhança

O principio da auto-semelhança sugere que, uma vez encontrada uma solução para determinado contexto, esta solução deve ser adotada em situações semelhantes. A utilização de aspectos permite que o código principal possa ser reutilizado em outras situações bem como outros sistemas, devido ao baixo acoplamento impactando esta característica de forma positiva. No entanto, é necessária uma atenção extra por parte dos desenvolvedores com o intuito de identificar comportamentos que possam ser tratados de maneira semelhante, visando evitar a replicação de código no sistema. Baseado nesta idéia, existem iniciativas de facilitar o princípio de auto-semelhança, por meio da utilização de *frameworks* transversais (*frameworks* que encapsulam um único interesse transversal) [Camargo & Masiero, 2005] e dos *frameworks Spring* e *JBoss AOP*. No entanto, a imaturidade de metodologias e padronizações no levantamento dos aspectos a serem utilizados acarretam em um impacto negativo na característica de auto-semelhança.

Baby steps

O processo XP prega que a codificação deve ser feita de forma incremental, através de "passos de bebê", em que o desenvolvedor pode ganhar confiança ao ver que cada passo produz o resultado esperado. Nesse ponto, a adoção de DSOA permite que o desenvolvedor insira gradualmente novos comportamentos no sistema até atingir o objetivo desejado gerando um impacto positivo na adoção de DSOA em XP. No entanto, a inversão de dependência em aspectos impacta esta característica de XP de forma negativa devido ao fato de que os desenvolvedores de código em OO, por exemplo, podem estranhar resultados errados obtidos após a integração entre o código OO e o código OA.

Ambiente informativo

A todo instante, deve ser possível conhecer o estado das atividades em andamento. A POA possibilita a inserção de comportamentos no sistema, de forma que possa alterar o funcionamento do sistema. Ambientes de desenvolvimento (e.g., Eclipse) têm reduzido o impacto do princípio da inconsciência ao utilizarem marcações no código como forma de alertar ao desenvolvedor que um aspecto está sendo inserido no trecho de código analisado. Pensando na idéia de radiadores de informação (meio), a identificação de aspectos impactaria esta característica de forma positiva, já que serviria como base para prover um mapa entre quais aspectos estão sendo implementados e por quais desenvolvedores, melhorando o ambiente informativo.

Design incremental

Através do *design* incremental, pode-se entregar o produto de software ao cliente por etapas. Dado o desenvolvimento das funcionalidades de maior interesse, o cliente tem a visibilidade de parte de seu sistema em uso, enquanto as demais funcionalidades estão em construção. No processo de adaptação do produto de software para o atendimento de novos requisitos, trechos de código podem ser alterados com freqüência (através da refatoração do código). Com isso, as técnicas de refatoração devem ser estendidas ou revistas, a fim de verificar se o aspecto continua produzindo o comportamento desejado e impedir a manifestação de impactos negativos como modificações dos *pointcuts* que acarretem na modificação destes comportamentos. Por outro lado, novas funcionalidades podem ser inseridas no produto de software sem que haja alteração significativa no código principal, gerando uma influência positiva de DSOA nesta característica.

Integração contínua

A adoção de DSOA requer o aumento de disciplina com relação à prática de integração contínua, dado que os aspectos interferem em diversas partes do sistema. De certa forma, isso impacta na maneira de se realizar testes de integração no sistema, sobretudo em qual ordem as entidades (classes OO e aspectos)

devem ser construídas e testadas. Ainda, a partir do momento que o desenvolvedor observe que o aspecto codificado irá influenciar em outras partes do sistema (além daquelas que ele esteja desenvolvendo), ele deverá integrar os aspectos ao sistema e avisar aos demais membros da equipe. Este aumento de disciplina, aliado às imaturidades de metodologias e padronizações durante a identificação dos aspectos a serem utilizados e combinado com as dificuldades da utilização de novas abstrações, acarretam em uma influência negativa de DSOA em XP.

Código coletivo

A separação de interesses através de aspectos faz com que um mesmo aspecto possa influenciar diversas partes do sistema. XP prega que o código pertence a toda equipe e, assim, todos possuem a liberdade de modificá-lo, quando necessário. Dessa forma, a separação de interesses deve ser realizada com cautela, a fim de evitar um impacto negativo, pois a alteração da estrutura de um aspecto, visando atender um objetivo de certa parte do sistema, pode modificar o comportamento de outras partes do sistema e interferir no funcionamento do mesmo. Por outro lado, pode-se alcançar uma melhor visualização da implementação dos requisitos, gerando uma influencia positiva, por permitir maior modularização do produto de software produzido.

Desenvolvido dirigido a testes (TDD)

TDD deve ser analisado com cuidado pelos desenvolvedores, dado que o acoplamento entre aspectos pode inserir comportamentos indesejados, influenciando negativamente esta característica. Alguns pesquisadores levantam a hipótese de que a inserção de chamada a métodos em componentes que foram concebidos sem pensar em tal interação poderia apresentar um novo comportamento [Resende & Silva, 2005]. Por outro lado, a separação de interesses proporcionada pela POA permite também um planejamento diferenciado dos testes para as funcionalidades presentes nos aspectos e no código OO puro, acarretando em um impacto positivo da utilização de DSOA em XP.

Combinando XP e DSOA

Buscando explorar e entender os desafios inerentes à união entre XP e DSOA, questões de DSOA relacionadas por Santos *et. al* (2007) foram aplicadas diante das características de XP discutidas na seção anterior. São consideradas seis questões críticas de DSOA: (i) utilização de novas abstrações – envolve o entendimento das estruturas de OA na projetação de novas soluções para construção de produtos de software; (ii) identificação de aspectos – visa definir o que deve ser um aspecto em um projeto de software; (iii) imaturidade de metodologias e padrões – a falta de metodologias e técnicas consolidadas pode ser um risco ao desenvolvimento de software; (iv) depuração – mecanismos de depuração devem estar presentes para identificar erros no produto de software, principalmente quando vários aspectos podem atuar em um mesmo trecho de código (*join point*); (v) inversão de dependência – permite que um mecanismo externo controle o fluxo de execução de um sistema; (vi) inconsciência – consiste em não transparecer, para o programador desenvolvedor e para o código OO e, existência de comportamentos transversais encapsulados em aspectos. A **Tabela 1** relaciona cada questão às características de XP.

As características de DSOA que influenciam XP de forma positiva e negativa impactam o esforço e a qualidade da implementação. Pode-se observar que os impactos negativos sobre as características de XP se devem aos problemas do DSOA em relação a: rastreabilidade entre projeto e codificação de aspectos; surgimento de comportamentos indesejados; dificuldades de depuração de código; e visão da equipe de desenvolvimento sobre os aspectos utilizados no produto de software. A possibilidade de produzir incrementos de software sem grandes modificações no código orientado a objetos puro, a utilização de aspectos para o desenvolvimento de testes, a extração de trechos de código semelhantes e a separação de interesses trazem impactos positivos quanto à adoção de DSOA junto a XP.

	DSOA / XP	Novas abstrações	Identificação de aspectos	Imaturidade de metodologias	Depuração em POA	Inversão de dependência	Inconsciência
VALORES	Coragem						
	Simplicidade						
PRINCÍPIOS	Auto-semelhança						
	Melhoria						
	Baby steps						
PRÁTICAS PRIMÁRIAS	Ambiente informativo						
	Test-Driven Development						
	Design incremental						
	Integração contínua						
	Código coletivo						

Legendas: Influência positiva de DSOA em XP; Influência positiva e negativa de DSOA em XP; Influência negativa de DSOA em XP; Indiferente (sem influência percebida)

Tabela 1. Lista dos impactos de DSOA sobre características de XP

É possível perceber, através da análise da tabela, a existência de um maior número de impactos negativos (19) frente a um número expressivamente menor de impactos positivos (10). Além disto, algumas características são influenciadas de maneira negativa e positiva (7), ainda que grande parte das características de XP não seja influenciada pelas questões de DSOA (24). Desta forma, o emprego de DSOA no contexto de XP deve ser avaliado de maneira cuidadosa, procurando estimar quando os aspectos positivos de DSOA agregam mais valor diante dos riscos associados aos impactos negativos. Neste sentido, são identificados, na **Figura 1**, alguns desafios que devem ser explorados no sentido de minimizar os riscos da utilização de DSOA. Estes desafios têm como objetivo servir como um primeiro passo em direção à diminuição dos impactos negativos causados pela adoção de DSOA em projetos baseados no processo XP.

Figura 1. Principais desafios da adoção de DSOA em XP.

Conclusões

Apesar das vantagens oferecidas pela adoção de XP em projetos de software, não se pode afirmar que seus valores, princípios e práticas se aplicam em todos os cenários, uma vez que seus melhores casos de sucesso se referiam à sua aplicabilidade em pequenos projetos, de risco relativamente baixo, envolvendo pessoal altamente capacitado e requisitos passíveis de mudanças [Boehm, 2006]. No entanto, estudos mostram que é possível aplicar XP em projetos de maior porte através de extensões que enfatizam no constante feedback, projeto de software e criação de novos papéis no ciclo de desenvolvimento [Lipert *et al*., 2003][Eckstein, 2006]. Nesse sentido, novas abordagens e tecnologias associadas devem ser analisadas visando verificar sua influência em projetos que utilizam XP, como DSOA. Por outro lado, apesar de suas vantagens, DSOA apresenta algumas questões críticas [Santos *et al*., 2007], as quais devem ser analisadas a fim de se pensar em avaliar a aplicabilidade de DSOA em diferentes contextos, buscando por possíveis impactos positivos e/ou negativos.

Referências

Beck, K. (2000) "Extreme Programming Explained". Addison-Wesley, 190p.

Beck, K. et al. (2001) "Manifesto for Agile Software Development". Disponível em: www.agilemanifesto.org. Acessado em: 11/07/2008.

Beck, K.; Andres, C. (2004) "Extreme Programming Explained: Embrace Change, Second Edition". Addison-Wesley, 2ª ed., 224p.

Boehm, B. (2002) "Get Ready for Agile Methods, with Care", In: IEEE Computer, v. 35, n. 1 (January), pp. 64-69.

Boehm, B. (2006) "A View of 20th and 21st Century Software Engineering", In: Proc. of the 28th ICSE, Shanghai, China, pp. 12-29.

Elrad, T.; Kiczales, G.; Aksit, M.; Lieberher, K.; Ossher, H. (2001) "Discussing Aspects of AOP". Communications of the ACM, v. 44, n. 10 (October), p. 33-38.

Filman, R. E.; Elrad, T.; Clarke, S.; Aksit, M. (2004) "Aspect-Oriented Software Development". Addison-Wesley Professional, 1ª ed., 800p.

Hanenberg, S.; Oberschulte, C.; Unland, R. (2003) "Refactoring of Aspect-Oriented Software", In: Net.ObjectDays, Erfurt, Germany.

Jutta Eckstein, Agile Software Development in the Large - OOPSLA 2006 Disponível em: http://www.oopsla.org/2006/submission/tutorials/agile_software_development_in_the_large.html). Acessado em 24/08/2008 .

Kiczales, G.; Lamping, J.; Mendhekar, A.; Maeda, C.; Lopes, C. V.; Loingtier, J. M.; Irwin, J. (1997) "Aspect-Oriented Programming". In: 11th European Conference On Object-Oriented Programming (ECOOP'1997), Jyväskylä, Finland. LNCS, v. 1241, 531p., pp. 220-242.

Kircher, M.; Jain, P.; Corsaro, A. (2002) "XP + AOP = Better Software?", In Proc. of the eXtreme Programming and Agile Processes in Software Engineering (XP'2002), Alghero, Italy, 2002.

Laddad, R. (2003) "AspectJ in Action". Manning Publication, 512p.

Lippert, Martin et al. Developing Complex Projects Using XP with Extensions. IEEE Computer, Nova Iorque, v. 36, n.6, Jun 2003. p. 57 66.

Paulk, M. C. (2001) "Extreme Programming from a CMM Perspective", In: IEEE Software, v. 18, n. 6 (November/December), pp. 19-26.

Resende, A. M. P.; Silva, C. C. (2005) "Programação Orientada a Aspectos em Java". Brasport, 1ªed., 208p.

Santos, R. P. (2008) "Uma Visão Sociotécnica sobre o Desenvolvimento de Software com Extreme Programming", In: IV WOSES, VII SBQS, Florianópolis/SC, pp. 25-36.

Santos, R. P.; Silva, M. C. O.; Werner, C. M. L.; Murta, L. G. P. (2007) "Questões e Desafios de Orientação a Aspectos no Contexto da Reutilização de Software", In: Proc. of the I Latin-American Workshop on Aspect-Oriented Software Development (LA-WASP'2007), João Pessoa/PB, p. 185-185.

Sato, D. T. (2007) "Uso Eficaz de Métricas em Métodos Ágeis de Desenvolvimento de Software". Dissertação (Mestrado). IME/USP, São Paulo/SP, 139p.

Shore, J.; Warden, S. (2007) "The Art of Agile Development". USA: O'Reilly, 438p.

Teles, V. M. (2005) "Um Estudo de Caso da Adoção das Práticas e Valores do Extreme Programming". Dissertação (Mestrado). IMNCE/UFRJ, Rio de Janeiro/RJ, 179p.

Tichy, W. F. (2004) "Agile Development: Evaluation and Experience", In: Proc. of the International Conference on Software Engineering (ICSE'2004), p. 692.

Travassos, G. H.; Shull, F.; Carver, J. (2001) "Working with UML: A Software Design Process Based on Inspections for the Unified Modeling Language". Advances in Computer, v. 54, pp. 36-99.

Dessa forma, realizou uma análise sobre características de XP, visando confrontá-las com algumas questões críticas relativas ao DSOA, levantadas em [Santos *et al*., 2007]. Essa análise faz-se necessária para que possíveis impactos (positivos e negativos) que uma abordagem como DSOA provocaria em ambientes reais sejam bem delineados e previamente apontados. Em ambientes que têm maior probabilidade de sucesso com XP (empresas de pequeno e médio porte), isso se torna fundamental, dado que esse processo se baseia na agilidade e não é dirigido por planos [Boehm, 2002]. Seguindo este raciocínio, este trabalho tem como objetivo servir como um complemento aos estudos já realizados [Kircher *et al*., 2002], procurando levantar desafios que auxiliem na diminuição dos impactos negativos da adoção de DSOA em XP.

A adoção de DSOA em projetos com XP produz efeitos positivos devido a modularização do software, melhor divisão de responsabilidades, possibilidade de estabelecimento de padrões de codificação e realização de testes. No entanto, para que essa adoção obtenha sucesso, é necessário haver investimento em treinamento da equipe com o intuito de permitir a identificação dos possíveis aspectos a serem utilizados no sistema além de ambiente e ferramentas informativas sobre a existência de aspectos e local de atuação.

Dessa forma, para que DSOA seja adotad o em cenários ágeis, deve ser submetido primeiro a aplicações práticas em cenários tradicionais. Ou seja, devido aos riscos envolvidos em XP, suas bases metodológicas e tecnológicas devem estar bem sedimentadas para que suas promessas sejam garantidas, dado que XP se beneficia de experiências e carências das abordagens baseadas em planos [Paulk, 2001]. ●

Profissional de TI

Conheça o **IBM Rational Team Concert,** a ferramenta que permite a criação de aplicações de negócio de forma rápida e eficaz, acelerando o seu ciclo de desenvolvimento.

Disponível em 3 versões, o **IBM Rational Team Concert** é **gratuíto** para até 3 usuários (versão Community Edition).

Para mais informações, entre em contato conosco e solicite um DVD com conteúdo exclusivo!

PRIMEUP

IBM Business Partner

Endereço:
R. Marquês de São Vicente, 225
Instituto Gênesis, Sala 27B
PUC-Rio, Gávea

Tel: (21) 2512-6005

atendimento@primeup.com.br
www.primeup.com.br

Especificação de Requisitos com Casos de Uso

Alguns exemplos prontos que você pode tomar como base ao especificar seus requisitos

Rodrigo Oliveira Spínola
rodrigo@sqlmagazine.com.br
Doutorando em Engenharia de Sistemas e Computação (COPPE/UFRJ). Mestre em Engenharia de Software (COPPE/UFRJ, 2004). Bacharel em Ciências da Computação (UNIFACS, 2001). Colaborador da Kali Software (www.kalisoftware.com), tendo ministrado cursos na área de Qualidade de Produtos e Processos de Software, Requisitos e Desenvolvimento Orientado a Objetos. Consultor para implementação do MPS.BR. Atua como Gerente de Projeto e Analista de Requisitos em projetos de consultoria na COPPE/UFRJ. É Colaborador Engenharia de Software Magazine.

Eduardo Oliveira Spínola
eduspinola@gmail.com
É Colaborador das revistas Engenharia de Software Magazine, Java Magazine e SQL Magazine. É bacharel em Ciências da Computação pela Universidade Salvador (UNIFACS) onde atualmente cursa o mestrado em Sistemas e Computação na linha de Engenharia de Software, sendo membro do GESA (Grupo de Engenharia de Software e Aplicações).

De que se trata o artigo?
Este artigo apresenta alguns exemplos reais de especificação de casos de uso.

Para que serve?
Seu objetivo é explicitar de forma prática como estas descrições podem ser efetuadas em um nível de detalhe tal que informações importantes para outras etapas do desenvolvimento como planejamento de testes, projeto e desenvolvimento não sejam omitidas.

Em que situação o tema é útil?
O assunto abordado é útil no dia a dia do analista de requisitos na realização de suas atividades.

A atividade de identificação e especificação de requisitos do software é uma atividade bastante desafiadora. É complexo:

- Identificar as reais necessidades do cliente;
- Lidar com clientes;
- Formalizar as necessidades do cliente através da especificação de requisitos de forma que esta seja de fácil entendimento para o cliente e forneça as informações requeridas pela equipe de desenvolvimento;
- Lidar com domínios desconhecidos. Entende-se por domínio o contexto para o qual o software está sendo desenvolvido. Por exemplo: contabilidade, medicina, controle de estoque. Para ilustrar esta dificuldade, imagine-se elaborando a especificação dos requisitos de um módulo estatístico para um sistema do mercado financeiro.

A lista apresentada não é completa, mas fornece indícios reais da complexidade desta atividade. Existem diversas abordagens que podem ser trabalhadas em conjunto para lidar com estas dificuldades, por exemplo: análise de domínio, prototipação, casos de uso.

O objetivo deste artigo não é apresentar um referencial teórico sobre como lidar com cada questão envolvida nas atividades diárias de um analista de requisitos.

Focaremos em um ponto específico de seu trabalho que é a atividade de descrição (especificação) dos requisitos. Faremos isto de forma totalmente prática através da apresentação de um conjunto de casos de uso especificados que poderão servir de inspiração para suas atividades como analista de requisitos.

Contextualização dos Exemplos

As especificações de casos de uso apresentadas neste artigo são fruto da experiência prática dos autores como analistas de requisitos. Eles não objetivam servir de gabarito para futuras atividades de especificação, ao invés disso, podem ser considerados um bom ponto de partida para que possamos ver na prática como podemos escrever casos de uso efetivos. Entende-se por efetivo neste caso o fato do caso de uso conter informações necessárias para que as equipes de projeto e desenvolvimento possam desempenhar satisfatoriamente suas atividades a partir da descrição dos casos de uso.

É importante estar atento também ao fato de que os casos de uso apresentados neste artigo refletem as características de escrita dos autores. Estas características que envolvem itens como nível de detalhamento, informações contidas nos casos de uso, dentre outras, costumam mudar também de organização para organização. Mesmo assim, os exemplos são bastante válidos uma vez que o núcleo básico envolvido na descrição de casos de uso (fluxos principal e alternativos) está presente.

Consideraremos neste artigo o conjunto de definições apresentadas na **Tabela 1** como sendo importantes na descrição de um caso de uso.

A partir de agora serão apresentados quatro exemplos de especificação de casos de uso. Para cada uma delas faremos comentários adicionais para que o entendimento do requisito descrito faça sentido. Por fim, é importante também destacar que os exemplos foram escolhidos de forma a contemplar diferentes situações comumente presentes em especificações de requisitos (busca, cadastro, envio de e-mail, extensões).

Exemplos de Casos de Uso

Para os exemplos de casos de uso apresentados iremos considerar a lista de requisitos funcionais apresentada na **Tabela 2**. O diagrama de casos de uso correspondente pode ser visto na **Figura 1**.

Consideramos para este artigo a especificação dos casos de uso: UC 1 – Administrar Clientes, UC 2 – Consultar Projeto, UC 3 – Listar Projetos a Vencer, UC 4 – Listar Propostas de Projeto Criadas.

No caso de uso UC 1 – Administrar Clientes, apresentamos umas das situações mais comuns de encontrarmos em sistemas de informação, que é o cadastro de alguma entidade no sistema. Neste exemplo, estamos considerando as interfaces apresentadas nas **Figuras 2** e **3** como

Objetivo:	Contém uma breve descrição do objetivo do caso de uso.
Requisitos:	Neste campo indicamos a qual requisito funcional o caso de uso em questão está associado (ler **Nota 1**).
Atores:	Neste campo definimos a lista de atores associados ao caso de uso. Ator é qualquer entidade externa que interage com o sistema (neste caso, com o caso de uso em questão).
Prioridade:	Informação identificada junto ao usuário que auxilia na definição dos casos de uso que serão contemplados em cada iteração do desenvolvimento do software.
Pré-condições:	Neste campo devemos informar as condições que devem ser atendidas para que o caso de uso possa ser executado.
Freqüência de uso:	Informação identificada junto ao usuário que auxilia na definição dos casos de uso que serão contemplados em cada iteração do desenvolvimento do software.
Criticalidade:	Informação identificada junto ao usuário que auxilia na definição dos casos de uso que serão contemplados em cada iteração do desenvolvimento do software.
Condição de Entrada:	Neste campo definimos qual ação do ator dará início à interação com o caso de uso em questão.
Fluxo Principal:	Esta é uma das seções principais do caso de uso. É onde descrevemos os passos entre o ator e o sistema. O fluxo principal é o cenário que mais acontece no caso de uso e/ou o mais importante.
Fluxo Alternativo:	Fluxo alternativo é o caminho alternativo tomado pelo caso de uso a partir do fluxo principal, ou seja, dada uma condição de negócio o caso de uso seguirá por outro cenário que não o principal caso essa condição seja verdadeira.
Extensões:	Nesta seção colocamos todos os casos de uso que estendem o caso de uso base e em quais pontos eles são chamados dentro dos fluxos de eventos.
Pós-condições:	Neste campo devemos informar o estado em que o sistema (ou entidade manipulada no caso de uso) estará depois que o caso de uso for executado.
Regras de negócio:	Nesta seção descrevemos todas as regras funcionais que o caso de uso deve cumprir durante sua execução.

Tabela 1. Conceitos envolvidos na descrição de um caso de uso.

Nota 1. Requisito Funcional

São requisitos diretamente ligados a funcionalidade do software, descrevem as funções que o software deve contemplar. Alguns exemplos são:

- O software deve permitir o cadastro de clientes.
- O software deve permitir a geração de relatórios sobre o desempenho de vendas no semestre.
- O software deve permitir o pagamento das compras através de cartão de crédito.

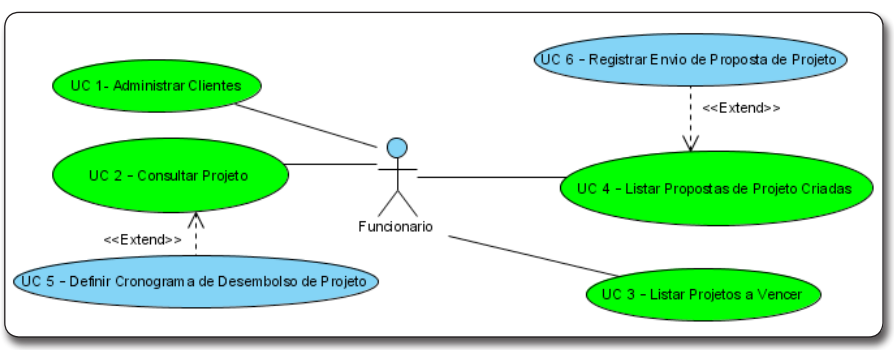


Figura 1. Diagrama de Casos de Uso.

RF1	O software deve permitir que funcionários adicionem, consultem, removam ou alterem dados de clientes de projetos.
RF2	O software deve permitir que funcionários acompanhem o ciclo de vida de seus projetos.
RF3	O software deve apresentar ao usuário a lista de projetos cuja data de término esteja dentro de parâmetros informados no sistema.
RF4	O software deve permitir que funcionários visualizem as propostas de projeto que estejam no estado "Proposta Criada".
RF5	O software deve permitir que funcionários definam as parcelas que irão compor o projeto, seus valores e eventos que serão associados ao pagamento de cada parcela.
RF6	O software deve permitir que funcionários registrem o envio de Proposta de Projeto ao cliente do projeto, anexando a versão final do documento referente à proposta do projeto no sistema.

Tabela 2. Lista dos requisitos funcionais.

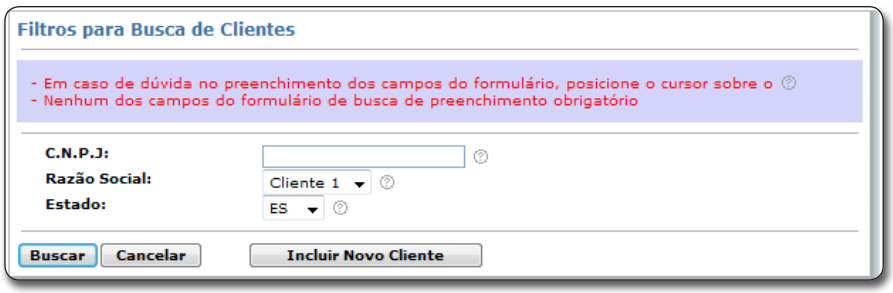


Figura 2. Tela 1 do caso de uso UC 1 – Administrar Clientes

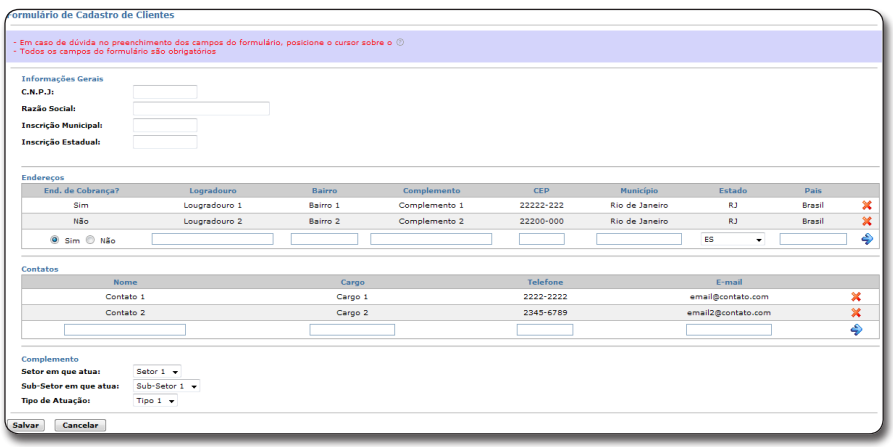


Figura 3. Tela 2 do caso de uso UC 1 – Administrar Clientes.

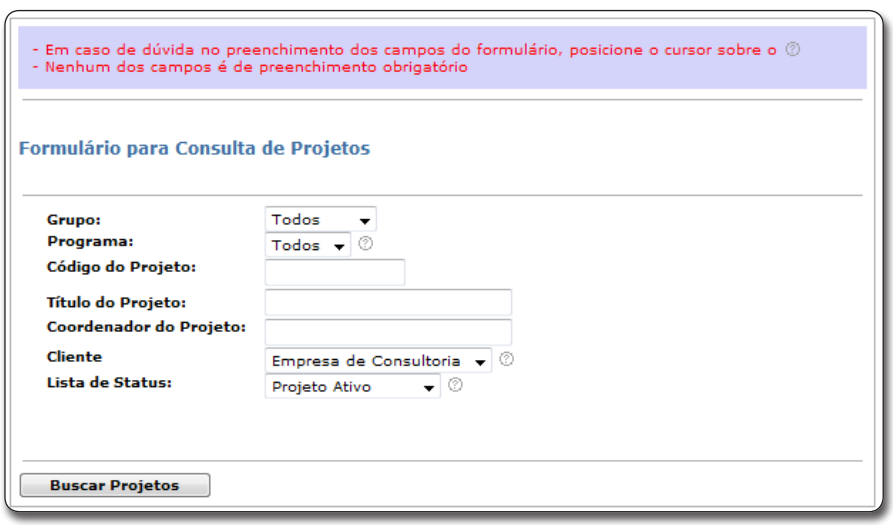


Figura 4. Tela 1 do caso de uso UC 2 – Consultar Projeto

base para especificação do caso de uso. A especificação pode ser vista na **Tabela 3**.

No caso de uso UC 2 – Consultar Projeto, apresentamos também umas das situações mais comuns de encontrarmos em sistemas de informação: consulta a uma dada entidade e disponibilização de opções de manipulação desta entidade. Neste caso, destacamos o uso de pontos de extensão com a opção do autor definir um cronograma de desembolso (definido em outro caso de uso).

Neste exemplo, estamos considerando as interfaces apresentadas nas **Figuras 4, 5** e **6** como base para especificação do caso de uso. A especificação pode ser vista na **Tabela 4**.

No caso de uso UC 3 – Listar Projetos a Vencer, apresentamos uma situação em que o sistema retorna o conjunto de projetos próximos do vencimento. Destacamos neste exemplo o fato de termos uma funcionalidade de e-mail contemplada e a necessidade de definir um modelo para descrição das informações que estarão presentes no e-mail.

Neste exemplo, estamos considerando a interface apresentada na **Figura 7** como base para especificação do caso de uso. A especificação pode ser vista na **Tabela 5** e a definição das informações de e-mail pode ser vista na **Tabela 6**.

No caso de uso UC 4 – Listar Propostas de Projeto Criadas, apresentamos uma situação em que o sistema retorna o conjunto de propostas de projeto criadas. Destacamos neste exemplo mais uma vez o uso de pontos extensão (a opção de Registro de Envio de Proposta de Projeto).

Neste exemplo, consideramos a interface apresentada na **Figura 8** como base para especificação do caso de uso. A especificação pode ser vista na **Tabela 7**.

Considerações Finais

Este artigo apresentou alguns exemplos reais de especificação de casos de uso. Seu objetivo foi explicitar de forma prática como estas descrições podem ser efetuadas em um nível de detalhe tal que informações importantes para outras etapas do desenvolvimento como planejamento de testes, projeto e desenvolvimento não sejam omitidas. ●

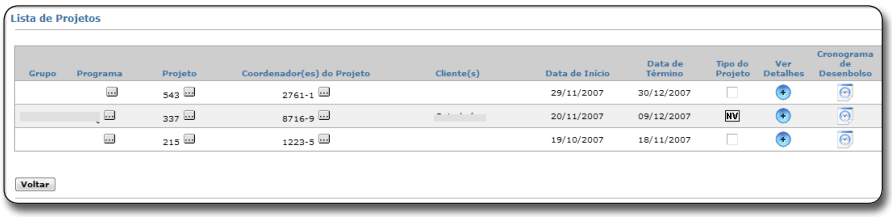


Figura 5. Tela 2 do caso de uso UC 2 – Consultar Projeto

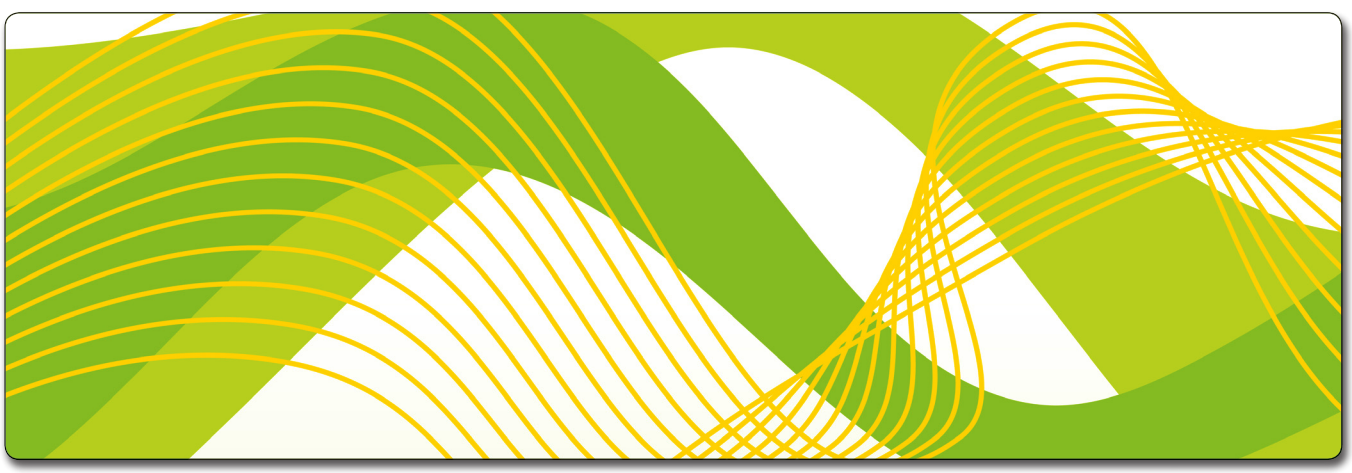
UC 1 Ð Administrar Clientes

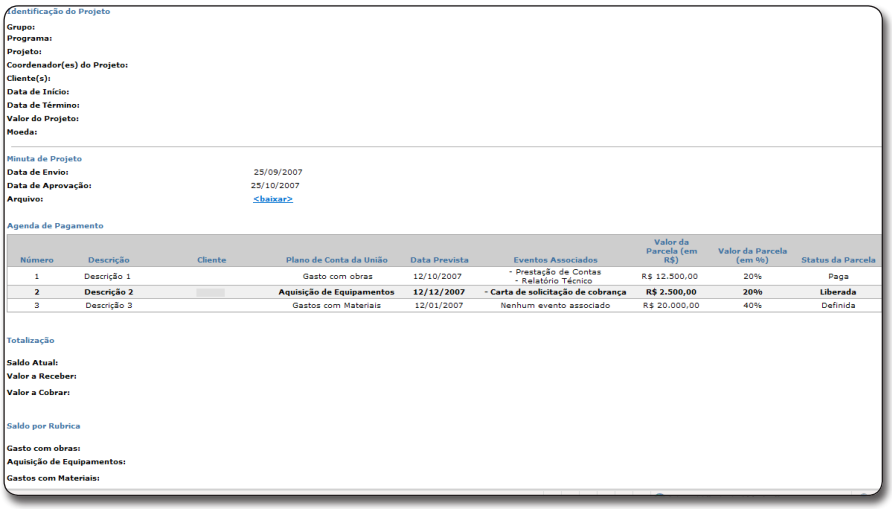
Objetivo:	Permitir que funcionários adicionem, consultem, removam ou alterem dados de clientes de projetos.
Requisitos:	RF1
Atores:	Funcionário
Prioridade:	Baixa
Pré-condições:	
Freqüência de uso:	Eventual
Criticalidade:	Baixa
Condição de Entrada:	O Funcionário seleciona a opção Administrar Cliente.
Fluxo Principal:	1. O sistema apresenta formulário de busca de clientes contendo as informações: - C.N.P.J (campo editável) - Razão Social (campo editável) - Estado (lista dos Estados brasileiros) - As opções Buscar e Cancelar - A opção Incluir Novo Cliente 2. O ator escolhe a opção Incluir Novo Cliente **[A1][A2]** 3. O sistema apresenta formulário de cadastro de Clientes contendo os seguintes campos a serem preenchidos: (Informações Gerais) - C.N.P.J (campo editável) - Razão Social (campo editável) - Inscrição Estadual (campo editável) - Inscrição Municipal (campo editável) (Endereços) - Tabela com os endereços cadastrados para o cliente que está sendo criado (cada endereço adicionado possui as seguintes informações: Indicador de Endereço de Cobrança, Logradouro, Bairro, Complemento, CEP, Município, Estado e País) e a opção de Excluir Endereço **[A5]** - Campos: Indicador de Endereço de Cobrança (opções Sim ou Não), Logradouro (campo editável), Bairro (campo editável), Complemento (campo editável), CEP (campo editável), Município (campo editável), Estado (lista de estados brasileiros) e País (campo editável) e a opção de Incluir Endereço **[A6][RN2] [RN7]** (Contatos) - Tabela com os contatos cadastrados para o cliente que está sendo criado (cada contato adicionado possui as seguintes informações: Nome, Cargo, Telefone e e-mail) e a opção de Excluir Contato **[A5]** - Campos: Nome (campo editável), Cargo (campo editável), Telefone (campo editável), e-mail (campo editável) e a opção de Incluir Contato **[A6][RN3]** (Complemento) - Setor em que atua (lista de setores previamente cadastrados no sistema) (campo editável) - Sub-Setor em que atua (lista de sub-setores previamente cadastrados no sistema) (campo editável) - Tipo de Atuação (lista de tipos de atuação previamente cadastrados no sistema) (campo editável) - Opções de Confirmação: Salvar e Cancelar 4. O Ator preenche o formulário apresentado 5. O Ator seleciona a opção Salvar **[A7]** 6. O sistema valida os dados preenchidos **[RN4][RN5][A8]** 7. O sistema salva os dados do Cliente 8. O caso de uso é encerrado.

Tabela 3 (parte1). Especificação do caso de uso UC 1 – Administrar Clientes.

Fluxo Alternativo	**[A1]** O Ator seleciona a opção Cancelar 1. O sistema retorna à tela principal e o caso de uso é encerrado. **[A2]** O Ator informa os dados de busca e seleciona a opção Buscar **[RN1]** 1. O sistema recupera os Clientes e apresenta para cada um, ordenados de forma ascendente pela Razão Social: - Razão Social (somente leitura) - CNPJ (somente leitura) - Município (somente leitura) - Estado (somente leitura) - Opção de Editar - Opção de Excluir 2. O sistema seleciona um dos clientes e clica na opção Excluir **[RN6][A3]** 3. O sistema apresenta tela de confirmação de Exclusão contendo as opções: Confirmar e Cancelar 4. O Ator seleciona a opção Confirmar **[A4]** 5. O Sistema efetua a exclusão do Cliente 6. O caso de uso retorna ao passo 2 do fluxo alternativo **[A2]**. **[A3]** O Ator seleciona a opção Editar 1. O sistema segue a partir do passo 3 do fluxo principal, porém os campos já vêm preenchidos com os dados do Cliente selecionado. **[A4]** O Ator seleciona a opção Cancelar 1. O sistema retorna ao passo 2 do fluxo alternativo **[A2]**. **[A5]** O Ator seleciona a opção Excluir 1. Sistema solicita a confirmação da exclusão com as opões confirmar e cancelar 2. O ator confirma a exclusão 3. O item é removido da lista e o sistema retorna ao passo 3 do fluxo principal. **[A6]** O Ator seleciona a opção Incluir 1. O sistema armazena os dados do item inserido, atualiza a tabela e retorna ao passo 3 do fluxo principal. **[A7]** O Ator seleciona a opção Cancelar 1. Sistema retorna ao passo 1 do fluxo principal. **[A8]** O Ator preenche dados inválidos 1. Sistema informa quais os campos foram preenchidos incorretamente e retorna ao passo 3 do fluxo principal.
Extensões:	Nenhum
Pós-condições:	Os dados de clientes estão armazenados no sistema.
Regras de negócio:	**[RN1]** Todos os campos de filtro para a consulta de Clientes são opcionais. **[RN2]** Apenas um endereço cadastrado pode ser indicado como endereço de cobrança. **[RN3]** Ao menos um contato deve ser cadastrado para o cliente. Ao inserir um contato, todos os dados são obrigatórios. **[RN4]** Todos os campos do formulário são obrigatórios. **[RN5]** Não podem ser adicionados dois clientes com o mesmo número de CNPJ. **[RN6]** Um cliente só pode ser excluído caso ele não esteja associado a nenhum projeto e contrato/convênio. **[RN7]** Pelo menos um endereço deve ser inserido. Ao inserir um endereço, todos os dados são obrigatórios, com exceção do campo Complemento.

Tabela 3 (parte2). Especificação do caso de uso UC 1 – Administrar Clientes.





Identificação do Projeto

Grupo:
Programa:
Projeto:
Coordenador(es) do Projeto:
Cliente(s):
Data de Início:
Data de Término:
Valor do Projeto:
Moeda:

Minuta de Projeto

Data de Envio: 25/09/2007
Data de Aprovação: 25/10/2007
Arquivo: <baixar>

Agenda de Pagamento

Número	Descrição	Cliente	Plano de Conta da União	Data Prevista	Eventos Associados	Valor da Parcela (em R$)	Valor da Parcela (em %)	Status da Parcela
1	Descrição 1		Gasto com obras	12/10/2007	- Prestação de Contas - Relatório Técnico	R$ 12.500,00	20%	Paga
2	**Descrição 2**		**Aquisição de Equipamentos**	**12/12/2007**	**- Carta de solicitação de cobrança**	**R$ 2.500,00**	**20%**	**Liberada**
3	Descrição 3		Gastos com Materiais	12/01/2007	Nenhum evento associado	R$ 20.000,00	40%	Definida

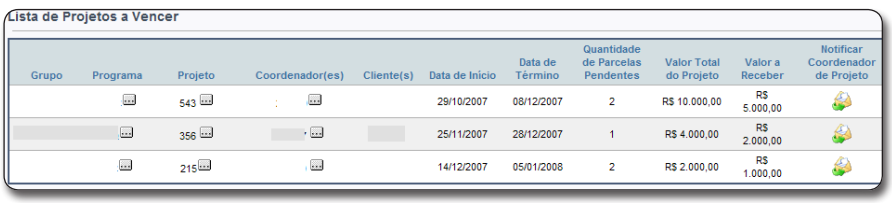
Totalização

Saldo Atual:
Valor a Receber:
Valor a Cobrar:

Saldo por Rubrica

Gasto com obras:
Aquisição de Equipamentos:
Gastos com Materiais:

Figura 6. Tela 2 do caso de uso UC 2 – Consultar Projeto



Lista de Projetos a Vencer

Grupo	Programa	Projeto	Coordenador(es)	Cliente(s)	Data de Início	Data de Término	Quantidade de Parcelas Pendentes	Valor Total do Projeto	Valor a Receber	Notificar Coordenador de Projeto
		543			29/10/2007	08/12/2007	2	R$ 10.000,00	R$ 5.000,00	
		356			25/11/2007	28/12/2007	1	R$ 4.000,00	R$ 2.000,00	
		215			14/12/2007	05/01/2008	2	R$ 2.000,00	R$ 1.000,00	

Figura 7. Tela 1 do caso de uso UC 3 – Listar Projetos a Vencer

UC 1 Ð Consultar Projeto

Objetivo:	Permitir que Funcionários acompanhem o ciclo de vida de seus projetos.
Requisitos:	RF2
Atores:	Funcionários
Prioridade:	Média
Pré-condições:	
Freqüência de uso:	Eventual
Criticalidade:	Média
Condição de Entrada:	Funcionário seleciona a opção Consultar Projeto.
Fluxo Principal:	1. O Sistema apresenta as seguintes opções para visualização dos projetos **[RN1]:** - Grupo (opção Todos + lista de grupos) (campo editável) - Programa (opção Todos + lista de programas) (campo editável) - Número do Projeto (campo editável) - Título do Projeto (campo editável) - Coordenador de Projeto (campo editável) - Cliente (campo editável) - Lista de status: todas (Projeto Ativo + Projeto Encerrado), Projeto Ativo, Projeto Encerrado. (campo editável) 2. O ator preenche os filtros de busca **[RN2].** 3. O Sistema apresenta uma lista de projetos, de acordo com os filtros fornecidos, ordenada por Código do Projeto **[A1]**: - Grupo (somente leitura) - Programa (somente leitura) - Projeto (Número + Título) (somente leitura) - Coordenador(es) do Projeto (somente leitura) - Cliente(s) (somente leitura) - Data de Início (somente leitura) - Data de Término (somente leitura) - Tipo do Projeto (somente leitura) - A opção Detalhar - A opção Definir Cronograma de Desembolso **[E1]**

Tabela 4 (parte1). Especificação do caso de uso UC 2 – Consultar Projeto.

Fluxo Principal:	4. O Sistema apresenta ao final a opção Voltar 5. Ator seleciona a opção Detalhar. **[A2]** 6. O Sistema apresenta tela contendo as informações: *Identificação* - Grupo - Programa - Projeto (Número + Título) - Coordenador(es) do Projeto - Cliente(s) - Data de Início - Data de Término - Valor do Projeto - Moeda - Tabela com as parcelas já cadastradas para o projeto selecionado (ordenadas pelo número da parcela), onde cada parcela adicionada possui as seguintes informações: * Número (somente leitura) * Descrição (somente leitura) * Cliente (somente leitura) * Plano de conta da união (somente leitura) * Data prevista (somente leitura) * Eventos associados (somente leitura) * Valor da Parcela (somente leitura) * Percentual da parcela em relação ao valor total (somente leitura) * Status da parcela (definida, pendente para cobrança, liberada para cobrança ou cobrada) (somente leitura) *Totalização* - Saldo Atual (somente leitura) - Valor a Receber (somente leitura) - Valor a Cobrar (somente leitura) (Saldo por Rubrica) (apenas se o projeto for Vinculado) - Tabela de Rubricas contendo Rubrica e Saldo (somente leitura) *Minuta de Projeto* - Data de Envio (somente leitura) - Data de Aprovação (somente leitura) 7. O Sistema apresenta ao final do formulário a opção voltar 8. O Ator seleciona a opção voltar 9. O Sistema retorna ao passo 3 do fluxo principal.
Fluxo Alternativo	**[A1]** Projeto não encontrado 1. O Sistema apresenta uma mensagem informando que não foi encontrado nenhum projeto com os dados fornecidos e, em seguida, apresenta o mesmo formulário com os dados previamente fornecidos 2. O sistema retorna ao passo 2 do fluxo principal. **[A2]** Ator selecionou a opção voltar 1. Sistema retorna ao passo 1 do fluxo principal.
Extensões:	**[E1]** Abrir caso de uso "UC 5- Definir Cronograma de Desembolso".
Pós-condições:	
Regras de negócio:	**[RN1]** Caso seja preenchido o Programa e Código do Projeto, a busca será feita usando estas informações, ignorando as demais opções. **[RN2]** Nenhum dos filtros de visualização é obrigatório.

Tabela 4 (parte2). Especificação do caso de uso UC 2 – Consultar Projeto.

UC 1 Ð Listar Projetos a Vencer

Objetivo:	Apresentar ao funcionário a lista de projetos cuja data de término esteja dentro de parâmetros informados no sistema.
Requisitos:	RF3
Atores:	Funcionário
Prioridade:	Baixa
Pré-condições:	
Freqüência de uso:	Diária
Criticalidade:	Baixa
Condição de Entrada:	O ator entra no Sistema
Fluxo Principal:	1. O Sistema recupera os projetos (ordenado pela data de término – da mais próxima da data corrente, para a mais distante) cujo vencimento se encontre dentro do período definido, contendo as informações **[RN1]**: - Grupo (somente leitura) - Programa (somente leitura) - Projeto (Número + Título) - Coordenador(es) do Projeto (Código + Nome) - Cliente(s) (somente leitura) - Data de Início (somente leitura) - Data de Término (somente leitura) - Quantidade de Parcelas Pendentes (somente leitura) - Valor Total do Projeto (somente leitura) - Valor Recebido (somente leitura) - Valor a Receber (somente leitura) - Opção Notificar Coordenador de Projeto **[A1]** 2. Caso de uso é encerrado.
Fluxo Alternativo	**[A1]** A opção Notificar Coordenador é selecionada 1. O Sistema apresenta tela de confirmação com a mensagem e as opções: - Mensagem: "Deseja enviar ao Coordenador de Projeto um e-mail indicando que a data de término do projeto está se aproximando e perguntando se há necessidade de se elaborar um aditivo do projeto?" - Opções Sim e Não 2. O Ator seleciona a opção Sim **[A2]** 3. O Sistema envia e-mail ao Coordenador do Projeto (ver e-mail na **Tabela 6**). **[A2]** A opção Não é selecionada 1. Caso de uso retorna ao passo 1 do fluxo principal.
Extensões:	
Pós-condições:	
Regras de negócio:	**[RN1]** Para identificar se o projeto está dentro do período para emissão de aviso, o sistema deve recuperar o parâmetro quantidade de dias de antecedência para aditivo no sistema e verificar se a data do projeto em questão se encontra no intervalo entre a data atual e a data atual subtraída da quantidade de dias de antecedência.

Tabela 5. Especificação do caso de uso UC 3 – Listar Projetos a Vencer.

Figura 8. Tela 1 do caso de uso UC 4 – Listar Propostas de Projeto Criadas.

Descrição:	Este e-mail é enviado para o Coordenador de Projeto indicando que a data de término do projeto está se aproximando e perguntando se há necessidade de se elaborar um aditivo do projeto.
Destinatário:	Coordenador do Projeto
Assunto:	Sistema X – Solicitação de Aditivo para o Projeto YYY XXX Onde YYY é o código do programa do projeto Onde XXX é o código do projeto
Corpo:	Caro Coordenador, O Projeto XXX, descrito abaixo, encontra-se com o vencimento próximo de ocorrer: - Grupo (código) - Programa (código) - Número do Projeto - Título do Projeto - Coordenador(es) do Projeto (Nome) - Cliente(s) (Razão Social) - Quantidade de Parcelas - Valor Total do Projeto - Valor a Receber - Número de Parcelas Pendentes de Recebimento - Data de Início - Data de Término Caso haja necessidade de geração de aditivo para o projeto, por favor, realizar tal solicitação através do Sistema ou entrando em contato com o setor responsável pelo acompanhamento do projeto. Este e-mail foi gerado automaticamente pelo Sistema. Caso necessário, entrar em contato diretamente com o setor responsável pelo acompanhamento do projeto. XX de XXXXXXX de XXXX

Tabela 6. Detalhamento das informações de e-mail.

UC 1 Ð Listar Propostas de Projeto Criadas

Objetivo:	Permitir que Funcionários visualizem as propostas de projeto que estejam no estado "Proposta Criada".
Requisitos:	RF3
Atores:	Funcionário
Prioridade:	Alta
Pré-condições:	
Freqüência de uso:	Diária
Criticalidade:	Baixa
Condição de Entrada:	Ator entra no sistema.
Fluxo Principal:	1. O Sistema recupera as propostas de projeto que estejam no estado "Proposta Criada" e apresenta cada uma, ordenada por data de solicitação (da mais antiga para a mais atual): - Grupo (somente leitura) - Programa (Código + Nome) - Proposta de Projeto (Número + Título) - Data de criação da proposta (somente leitura) - Coordenador(es) do Projeto (número + nome) - Cliente(s) (somente leitura) - Opção de Registrar Envio de Proposta de Projeto **[E1]** - Opção de Rejeitar Proposta de Projeto **[A1]** - Opção de Remover Proposta de Projeto **[A3]**
Fluxo Alternativo	**[A1]** O Ator seleciona a opção Rejeitar Proposta de Projeto 1. O Sistema apresenta tela de confirmação de Rejeição de Proposta contendo as opções: Confirmar e Cancelar 2. O Ator seleciona a opção Confirmar **[A2]** 3. O Sistema registra a proposta com o estado "Proposta Não-Aprovada" 4. Caso de uso retorna ao passo 1 do fluxo principal. **[A2]** O ator seleciona a opção Cancelar 1. Nada acontece e o caso de uso retorna ao passo 1 do fluxo principal. **[A3]** O Ator seleciona a opção Remover Proposta de Projeto 1. O sistema remove todos os registros associados à proposta de projeto selecionada e em seguida retorna ao passo 3 do fluxo principal.
Extensões:	**[E1]** Ator seleciona a opção Registrar Envio Ver caso de uso "UC 6 – Registrar Envio de Proposta de Projeto".
Pós-condições:	A lista de propostas criadas e ainda não enviadas ao cliente é apresentada.
Regras de negócio:	

Tabela 7. Especificação do caso de uso UC 4 – Listar Propostas de Projeto Criadas.

Usabilidade de Software

A Importância da Usabilidade no Desenvolvimento de Sistemas Interativos

Antonio Mendes da Silva Filho
antoniom.silvafilho@gmail.com
Professor e consultor em área de tecnologia da informação e comunicação com mais de 20 anos de experiência profissional, é autor do livros Arquitetura de Software e Programando com XML, ambos pela Editora Campus/Elsevier, tem mais de 30 artigos publicados em eventos nacionais e internacionais, colunista para Ciência e Tecnologia pela Revista Espaço Acadêmico com mais de 60 artigos publicados, tendo feitos palestras em eventos nacionais e exterior. Foi Professor Visitante da University of Texas at Dallas e da University of Ottawa. Formado em Engenharia Elétrica pela Universidade de Pernambuco, com Mestrado em Engenharia Elétrica pela Universidade Federal da Paraíba (Campina Grande), Mestrado em Engenharia da Computação pela University of Waterloo e Doutor em Ciência da Computação pela Univesidade Federal de Pernambuco.

De que se trata o artigo?
Usabilidade e Interação Humano-Computador (IHC) nos sistemas de software. Este artigo discute a importância da IHC e usabilidade no processo de desenvolvimento de sistemas de software

Para que serve?
Identificar atributos da usabilidade e aspectos da interação humano-computador que podem influenciar a facilidade de uso e aprendizagem de uma aplicação, os quais são determinantes para elevar o desempenho na realização de tarefas e grau de satisfação de usuários.

Em que situação o tema é útil?
Essencial no processo de desenvolvimento de sistemas interativos, onde objetiva-se desenvolver software que torne produtiva e satisfatória a realização de tarefas pelos usuários.

Um sistema de software, assim como qualquer outro produto, precisa ser simples, fácil de usar e deve otimizar o tempo de seu usuário de modo que ele(a) possa realizar uma tarefa de maneira eficiente e com satisfação. Um sistema de software ou produto com essas características tem usabilidade, um atributo de qualidade perceptível aos usuários e determinante no sucesso de um produto. O desenvolvimento de sistemas de software coloca a usabilidade como um dos atributos de qualidade que norteia o processo de construção do sistema, tópico discutido neste artigo.

Usabilidade

Hoje em dia, interagir com sistemas de software não é mais privilégio de profissionais de computação e tornou-se uma necessidade comum a toda população. Os projetistas e desenvolvedores de sistemas de software e, notadamente, dos sistemas interativos (i.e. sistemas caracterizados pela interação entre o sistema e o ser humano) têm sido colocados em posição de destaque, pois eles são os responsáveis pelo desenvolvimento

desses produtos. Entretanto, é imperativo que eles (projetistas e desenvolvedores) possam desempenhar bem seu papel, especificamente, no projeto de sistemas de software interativos, onde há interação entre ser humano e computador e têm como propriedade essencial prover suporte à atividade humana. Tal sistema habilita o ser humano a realizar suas tarefas mais rapidamente, com menos erros, com aprendizado menor, com qualidade resultante e satisfação maiores. Isto, contudo, depende da usabilidade do sistema.

Usabilidade é uma palavra que tem feito cada vez mais parte do vocabulário dos projetistas de sistemas de software. A usabilidade é um conceito chave no campo da Interação Humano-Computador (IHC), sendo um atributo de qualidade de sistemas que são fáceis de usar e de aprender. Em outras palavras, diz quão intuitiva é a interface gráfica de usuário ou, simplesmente, interface de usuário. Trata-se, portanto, de uma característica pela qual o usuário expressa seu interesse ou não em utilizar um sistema. Na grande maioria dos casos, os usuários preferem um sistema de fácil uso, mesmo com funcionalidades mais simples, a um sistema recheado de funcionalidades, porém de manipulação complexa e não intuitiva.

É importante também observar que a usabilidade é determinante no sucesso ou insucesso de qualquer produto. Portanto, o usuário sempre tem a última palavra ao expressar sua satisfação ou não no uso de um sistema ou produto. Perceba que produtos com usabilidade, resultante de interface (de usuário) bem projetada, permitem:

- Maior grau de eficiência quando os usuários realizam suas tarefas;
- Custos reduzidos de apoio ao usuário, tais como treinamento, ou atendimento ao usuário;
- A inserção de sistemas ou produtos mais naturalmente no ambiente de trabalho do usuário, facilitando a utilização do produto na realização de suas tarefas.

A usabilidade é uma característica da qualidade resultante do projeto de interface com o usuário, o qual compreende parte das atividades do processo de desenvolvimento de um sistema de software, discutido a seguir.

Desenvolvimento de Sistemas Interativos

Desenvolver um sistema interativo requer uma equipe de projeto atuando numa variedade de tarefas, as quais são estruturadas num processo. Tal processo compreende um conjunto de atividades que transforma entradas em saídas. Em determinado instante do processo, pode-se ter o projetista ou engenheiro de software esboçando parte da interface com usuário e/ou realizando uma entrevista com possíveis usuários a fim de derivar algum modelo e, posteriormente, tentar definir a arquitetura do sistema. Depois, mudanças de requisitos e projeto, naturais de acontecer no início do desenvolvimento, podem ser incorporadas a um protótipo o que permite realizar testes pelos usuários.

Um sistema interativo pode ser decomposto em duas grandes porções de software: o software da aplicação e o software da interface com usuário. A aplicação compreende toda a funcionalidade do sistema enquanto que o software da interface com usuário é encarregado de mediar a comunicação entre o usuário e a aplicação do sistema. Esta divisão de um sistema interativo é ilustrada na **Figura 1**.

Uma visão genérica de processo de desenvolvimento que contemplam apenas as principais atividades desses dois componentes é mostrada na **Figura 2**.

A parte superior da figura engloba atividades pertinentes ao desenvolvimento da aplicação, onde se tem atividades de desenvolvimento de software (encontradas na Engenharia de Software) resumidas apenas em três atividades (análise de sistema, desenvolvimento e testes de software). Perceba que a parte inferior da **Figura 2** compreende as atividades necessárias ao desenvolvimento

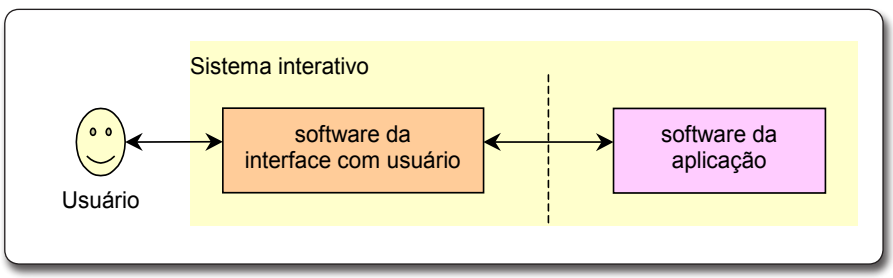


Figura 1. Decomposição de um sistema interativo.

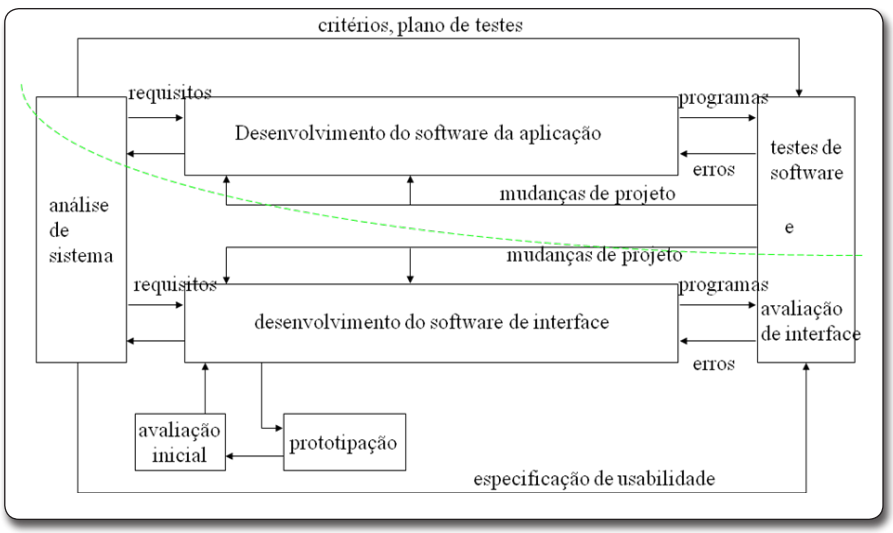


Figura 2. Processo de desenvolvimetno de um sistema interativo (esta figura poderia ser um pouco mais explicada no texto).

do software da interface de usuário. Há, também, três atividades básicas (análise de sistema, desenvolvimento e avaliação da interface). Todavia, adicionalmente, têm-se atividades de prototipação e avaliação inicial da interface do protótipo. Vale ressaltar que não houve qualquer intenção em ser completo, mas de destacar as atividades essenciais. Isso se deve ao fato de haver propostas distintas de processo de desenvolvimento para diferentes tipos de sistemas de software.

Neste contexto, o foco principal do desenvolvimento de sistemas interativos recai sobre o software da interface. Vale ressaltar que o processo de desenvolvimento de sistemas interativos, similarmente a de outros sistemas, consiste de três fases genéricas – definição, desenvolvimento e manutenção – conforme ilustrado na **Figura 3**.

A fase de definição compreende a identificação de informações que deveriam ser processadas, funções e desempenho desejados, tipo de interface a ser utilizada, tarefas que o sistema deveria prover suporte, perfil de usuários do sistema, dentre outras.

A fase de desenvolvimento concentra-se no projeto de estruturas de dados e arquitetura de software do sistema, conversão do projeto para alguma linguagem de programação, realização de testes e avaliação.

Finalmente, a manutenção considera modificações e/ou correções necessárias no sistema a fim de que este atenda aos requisitos do sistema.

Perceba que o processo de desenvolvimento de um sistema interativo tem dois grandes aspectos de interesse, que envolve o desenvolvimento das porções de software relativo à interface com usuário e da aplicação. Uma ilustração de processo que considera o desenvolvimento do componente do software de interface é mostrada na **Figura 4**.

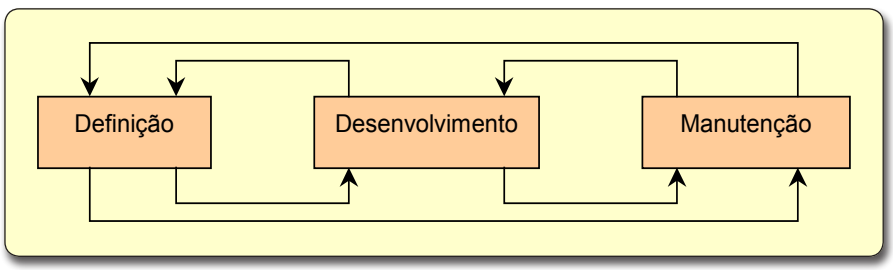


Figura 3. Fases genéricas no processo de desenvolvimento de software.

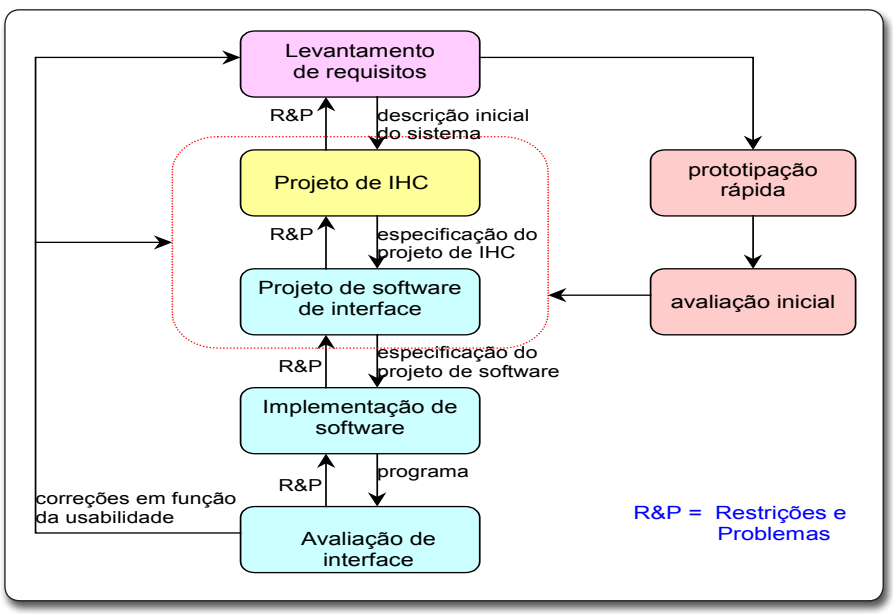


Figura 4. Desenvolvimento de software de interface.

Dentre as atividades que compõem o processo de desenvolvimento ilustrado na **Figura 4**, duas delas têm grande importância na qualidade obtida no sistema final. Estas são: o projeto de IHC (interação humano-computador) e projeto de software (de interface de usuário). Associado a essas atividades, tem-se a arquitetura de software que será dependente das características do sistema a ser desenvolvido, bem como dos atributos de qualidade desejados. Note que as atividades de desenvolvimento do software de interface, mostradas na **Figura 4**, podem conter restrições ou problemas que motivam retornar à atividade anterior visando modificação e/ou correção.

É comum o uso de ícones em interfaces de usuário que servem como dica para lembrar a qual comando está associado. Entretanto, se é feito uso de um ícone que não é do conhecimento dos usuários, então eles podem ficar perdidos ou até mesmo frustrados por não entenderem o significado do comando. Numa situação dessas, há a necessidade de modificar ou corrigir a proposta inicial do uso de ícone a fim de tratar esse problema. Uma correção recomendada é usar o ícone acompanhado da palavra que denota o comando, como mostrado na **Figura 5**, que ilustra botões de Start e Clear.

Cabe ressaltar que a interação humano-computador (IHC) é determinante para o projeto de interface (de usuário) e, portanto, para a usabilidade do sistema, como discutido na seção seguinte.

Interação Humano-Computador (IHC)

Interação Humano-Computador (IHC) é uma área multidisciplinar que envolve as áreas de Ciência da Computação, Psicologia, Fatores Humanos, Lingüística, dentre outras. IHC está voltada para a aplicação

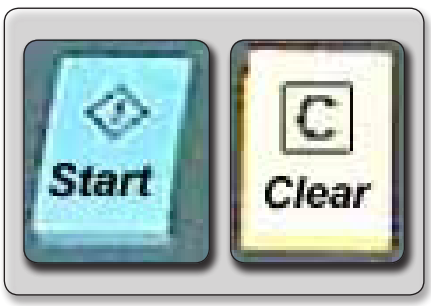


Figura 5. Exemplos de ícones com rótulos.

do conhecimento destas disciplinas para produzir interfaces ditas "amigáveis" (ou user-friendly). Em outras palavras, prover interfaces que ofereçam suporte a usabilidade. Isso é o que diz a Norma 9241 da ISO quando diz que a usabilidade é a capacidade que um sistema interativo oferece a seu usuário em determinado contexto de uso para realizar tarefas com eficácia, eficiência e satisfação. Quando a interface oferece essa três características, ela é comumente dita "amigável" (tradução do termo user-friendly).

IHC não é uma disciplina essencialmente voltada para o estudo de computação ou do ser humano, mas para a comunicação entre estas duas entidades. Conhecimento sobre as limitações da capacidade humana e restrições da tecnologia existentes devem ser ponderados para oferecer ao usuário um meio adequado através do qual eles podem interagir com os computadores.

Um dos fatores relevantes para a interface de usuário é a percepção humana. Todavia, antes de discuti-la, faz-se necessário explicar a diferença entre IHC e Interface de Usuário. É comum ambos os termos serem usados indistintamente e erroneamente. IHC é tudo que ocorre entre o ser humano e um computador utilizado para realizar algumas tarefas, ou seja, é a comunicação entre estas duas entidades. Interface de usuário é o componente (software) responsável por mapear ações do usuário em solicitações de processamento ao sistema (aplicação), bem como apresentar os resultados produzidos pelo sistema.

É comum encontrarmos interfaces que são difíceis de usar, confusas, e até mesmo frustrantes em alguns casos. Apesar de os projetistas gastarem tempo para desenvolver essas interfaces e que seja improvável que eles façam isto propositadamente, os problemas com interfaces acontecem. Recentemente, com o lançamento da nova versão do Office da Microsoft, o botão de apresentação de slides do Power Point, que era há décadas no canto inferior esquerdo, foi levado ao canto inferior direito.

Quando consideramos um sistema interativo, o termo fator humano assume vários significados. Dentro do nível fundamental, deveríamos entender a percepção visual, a psicologia cognitiva da leitura, a memória humana e os raciocínios dedutivo e indutivo. No outro nível, deveríamos entender o usuário e seu comportamento. Por fim, é preciso entender as tarefas que o software executa para o usuário e as tarefas que são exigidas do usuário quando da interação com o sistema. Note que o interesse recai, especificamente, na importância da percepção humana para o desenvolvimento de interface de usuário.

Os seres humanos percebem as coisas através de seus sentidos, isto é, visual, auditivo e tato. Estes sentidos habilitam o usuário de um sistema interativo a perceber a informação, armazená-la (em sua memória) e processar a informação usando o raciocínio dedutivo ou indutivo.

Grande parte da IHC ocorre através do sentido da visão, como por exemplo: relatórios, gráficos, etc. Neste caso, os olhos e o cérebro trabalham juntos a fim de receber e interpretar a informação visual baseada no tamanho, forma, cor(es), orientação e movimento. Muitos elementos discretos de informação são apresentados simultaneamente para as pessoas absorverem. Assim, uma especificação apropriada de comunicação visual é o elemento chave de uma interface amigável.

Embora haja uma tendência para se utilizar manipulação/comunicação gráfica no projeto de interface, muito da informação visual ainda é apresentada na forma textual. A leitura - o processo de extrair informação do texto - é a atividade chave na maioria das interfaces. Os seres humanos precisam decodificar os padrões visuais e recuperar o significado das palavras e frases. Para tanto, o processo de leitura tem sua velocidade controlada pelo padrão de movimento dos olhos, que escaneia o texto em alta velocidade.

Adicionalmente, o tipo de caractere fonte, o comprimento de linha do texto e cor(es) afetam a facilidade na qual o processo de leitura ocorre. Por exemplo, se considerarmos o uso da cor na busca e identificação de objetos, pode-se dizer que o uso excessivo de cores aumenta o tempo de busca e dificulta a identificação e memorização de objetos. Entretanto, a cor é útil na identificação de estados de objetos. Considere, por exemplo, que você esteja usando o ambiente Windows e abre uma pasta de documentos com vários arquivos. Neste caso, o ambiente permite que você selecione o modo de exibição (ícones grandes, ícones médios, ícones pequenos, detalhes, lista e lado a lado). Em tal situação, quando você clica sobre um arquivo, selecionando-o, a cor do objeto selecionado é modificada para identificar o estado do objeto (um arquivo) selecionado naquela pasta.

Quando a informação é extraída da interface, ela deve ser armazenada para ser recuperada (lembrada) e utilizada posteriormente. Além disso, o usuário precisa lembrar-se de comandos e seqüências operacionais de uso. Tais informações são armazenadas na memória humana (que é um sistema complexo) composto de duas partes: a memória de curta duração que possui capacidade de

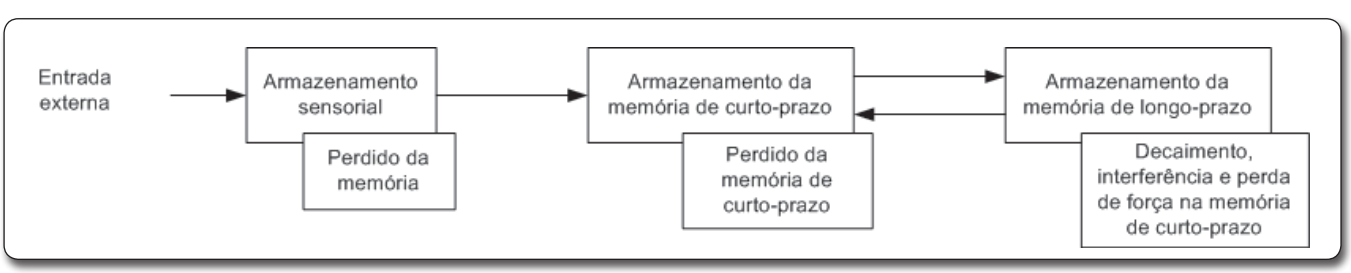


Figura 6. Modelo da memória humana.

armazenamento e tempo de recordação limitado, e a memória de longa duração que possui capacidade de armazenamento e tempo de recordação maior e onde se tem o conhecimento do ser humano, como ilustrado na **Figura 6**.

As três formas de armazenamento mostradas acima têm capacidades de armazenamento de informação distintas, como caracterizado abaixo:

• Armazenamento sensorial (específico para cada sentido – décimos de segundos);

• Armazenamento da memória de curta duração (informação limitada - poucos segundos);

• Armazenamento da memória de longa duração.

Assim, se o projetista especifica uma interface que faz solicitações indevidas dessas duas memórias (isto é, não levando em conta as limitações humanas), então o desempenho do usuário quando usando o sistema será degradado. Uma forma de avaliar a usabilidade é através do uso métricas (de usabilidade) que funcionam como critérios de medição da usabilidade do sistema considerado. Um conjunto desses critérios é apresentado na **Figura 7**.

Outro aspecto a considerar é que o projeto de interface de usuário deve ser centrado nas tarefas do usuário de modo a tornar mais natural sua implementação num sistema. Algumas observações são apresentadas abaixo:

• *IHC é baseada em tarefas* - quando um usuário interage com uma máquina, ou mais especificamente com um computador, ele está interessado em realizar uma ou mais tarefas. Portanto, pode-se considerar a tarefa como o elemento central da interação humano-computador;

• *Necessidade de seqüenciar tarefas secundárias* - usuários que desejam realizar uma tarefa, normalmente, têm a necessidade de saber uma ordem na qual executar as tarefas secundárias relevantes mesmo que eles possam e, normalmente, trabalhem em diversas tarefas de modo concomitante e assíncrono;

• *Usabilidade* - A abordagem baseada em tarefas permite ao projetista se concentrar em especificações precisas de tarefas realizadas pelos usuários a fim de assegurar uma comunicação natural entre usuário e computador.

Necessidades no Desenvolvimento de Sistemas Interativos

O desenvolvimento de um sistema interativo possui a peculiaridade de ter na interface de usuário um fator determinante da utilidade e aceitabilidade do sistema de software ou produto. Portanto, inicialmente, é importante observar que as técnicas desenvolvidas em Engenharia de Software não são sempre aplicáveis diretamente ao desenvolvimento desses sistemas. Isto ocorre porque a interface de usuário não é um componente de um sistema de software como um componente que implementa uma funcionalidade de busca, ordenação, controle e autenticação de usuário, ou qualquer outro componente funcional. A interface de usuário também requer, adicionalmente às atividades que se tem num projeto de software qualquer, o projeto de interação no qual se identifica e define objetos de interação (como menus, botões, caixas de diálogo, etc.) e definição da sintaxe do diálogo entre usuário e sistema, dentre outras coisas.

1. Tempo para realizar um tarefa.
2. Percentual de tarefa concluído.
3. Percentual de tarefa concluído por unidade de tempo.
4. Taxa de sucessos/falhas.
5. Tempo consumido com erros.
6. Percentual de erros.
7. Número de comandos utilizados.
8. Número de comandos disponíveis não utilizados.
9. Freqüência de uso de *ajuda* (help) ou documentação.
10. Número de vezes que o usuário expressa satisfação ou frustação.

Figura 7. Critérios de medição de usabilidade.

Nesse sentido, torna-se necessário buscar uma metodologia de projeto de interface de usuário que possa ser integrada às práticas atuais da engenharia de software. Dentro desse contexto, há um foco em técnicas de baixo custo.

Vale ressaltar que opiniões mais ortodoxas defendem que o projeto de interfaces deva ser executado por especialistas de outras áreas, como Psicologia e Ergonomia, para se obter sistemas de maior usabilidade. Todavia, tal percepção tem mudado aos poucos, uma vez que ela não é muito realista para a grande maioria das empresas de desenvolvimento de software. Dentre as razões, pode-se destacar:

• É oneroso manter múltiplas equipes com funções bem delimitadas;

• Apenas empresas com grande base de usuários podem se dar ao luxo de manter equipes especializadas;

• Sistemas críticos, onde vidas humanas podem estar em risco, consideram tal premissa.

Além disso, mesmo em grandes empresas, é cada vez mais aceito dispor de uma equipe com integrantes de perfil técnico mais abrangente e capaz de executar múltiplas atividades.

Outra observação é que múltiplas equipes deparam-se com problemas de comunicação entre elas. Nesse sentido, ocorre passagem de informações com ambigüidades e o produto obtido quase sempre não corresponde ao concebido.

As propostas atuais reconhecem tais dificuldades e têm o processo orientado para as tarefas do usuário. Dessa forma, ocorre apenas o envolvimento de uma equipe com membros ou engenheiros de software multifuncionais. Esses profissionais tanto criam o projeto quanto realizam a implementação. Eles compreendem, geralmente, especialistas em Computação com algum treinamento adicional em técnicas de projeto de interação.

Este profissional é responsável por planejar e realizar a inserção da tecnologia num ambiente de trabalho do usuário. O objetivo dele é compreender as tarefas realizadas no ambiente de trabalho do usuário a fim de levar essa informação para o sistema a ser desenvolvido. Adicionalmente, observa-se que:

• A usabilidade deve permear todo o processo;
• Não existe um método seguro e direto que garanta uma boa usabilidade já que muitas variáveis estão envolvidas;
• O processo é altamente iterativo, onde testes e avaliações representam atividades centrais.

Os modelos de desenvolvimento de sistemas interativos, via de regra, são norteados pela necessidade de ter o usuário como participante do processo de desenvolvimento, além de considerar a necessidade de oferecer suporte à usabilidade. Nesse sentido, o processo de desenvolvimento de sistemas interativos compreende várias atividades de projeto. Dentre elas, as duas mais importantes na qualidade obtida do produto final são o projeto IHC e o projeto de software de interface de usuário.

Juntamente com estas fases, tem-se a necessidade de uso de técnicas de especificação do projeto. Perceba que o meio pelo qual o projetista de interação comunica o projeto IHC ao projetista de software (ou engenheiro de software) é comumente realizada através de especificações, i.e., um conjunto formal de instruções sobre o projeto a partir do qual o código será desenvolvido. Objetivando auxiliar o processo do projeto de interação, os seguintes princípios são considerados:

• O desenvolvimento de um sistema de software deveria incluir teste empírico inicial e contínuo centrado em usuários apropriados que realizem tarefas representativas de modo a avaliar se os critérios de usabilidade estão sendo atendidos;
• À medida que o desenvolvimento procede, deveria incorporar procedimentos subseqüentes através de refinamento interativo e análise custo/benefício para determinar as modificações mais efetivas sob o ponto de vista do custo a serem realizadas no projeto de interação.

Por outro lado, o projeto de software de interface de usuário visa transformar o projeto do domínio do problema em um programa de computador. Nesta etapa do desenvolvimento é feito, por exemplo, o projeto das estruturas de dados e de algoritmos.

O processo de desenvolvimento de interface de usuário, envolvendo projeto de interação e projeto de software de interface de usuário, constitui-se num desafio. As razões pelas quais a interface é vista como sendo a parte mais difícil e desafiadora do desenvolvimento de um sistema interativo é, em parte, porque ela requer uma combinação de diversas áreas de conhecimento com respectivos especialistas para desenvolver uma interface de qualidade.

É importante observar que desenvolver interface de usuário não é simplesmente desenvolver uma fatia do software do sistema. Ela requer, além disso, outras atividades, tais como:

• Projeto da comunicação entre usuários e computador;
• Identificação e representação de tarefas dos usuários e informações pertinentes;
• Projeto gráfico e textual da interface;
• Projeto de software de interface através do qual outras decisões de projeto serão implementadas;
• Avaliação da interface.

Observe que o desenvolvimento de uma interface de usuário é a parte do sistema que mais requer do projeto tanto em termos de tempo de desenvolvimento quanto da proporção de software dedicada a esta fatia do sistema. Além disso, tal desenvolvimento tem uma atividade essencial que é, geralmente, integrada ao processo de desenvolvimento do sistema de software. Trata-se da avaliação da usabilidade, atributo da qualidade do sistema de software. Três categorias de métodos existem: inspeção, investigação e teste. Para cada uma dessas categorias, há técnicas que são empregadas para avaliar a usabilidade de um sistema de software, tema que será abordado num artigo futuro. Os leitores interessados no tema podem encontrar informações em http://www.usabilityhome.com/

Conclusão

Para finalizar, deve-se observar que nesse universo de dispositivos, dos mais variados tipos e objetivos, um aspecto determinante na aceitabilidade (leia-se também adoção) e uso deles é o *design* de suas interfaces. Dentro deste contexto, a usabilidade de uma interface envolve vários fatores como a facilidade de uso e aprendizagem por parte do usuário, bem como maior desempenho e satisfação do usuário na realização de suas tarefas. Esses fatores podem ser mensurados através de critérios de usabilidade como discutido no artigo. Todavia, enquanto a diversidade humana é um aspecto positivo no que tange ao enriquecimento sócio-cultural e troca de experiências entre os seres humanos, ela constitui-se num desafio aos projetistas de interface. Nesse sentido, um conjunto de características que o projetista deve considerar compreende:

• habilidades de percepção e cognição: capacidade de memorização, atenção e solução de problemas;
• fatores que afetam o desempenho motor e perceptivo: fadiga, ansiedade, medo, envelhecimento;
• diferenças culturais: descrição de datas, horário, unidades de peso (e outras medidas), formas de descrever endereços, bem como significado de cores e ícones;
• deficiências auditiva, motora, cognitiva e de fala, nos usuários de equipamentos.

Considerar os aspectos acima mencionados significa orientar o processo de desenvolvimento para o usuário, buscando atender as necessidades de trabalho dos mesmos, o que implica em aumentar as chances da aceitabilidade do sistema de software ou produto pelo usuário, já que a usabilidade está sendo considerada no processo de desenvolvimento. ●

Links

Human-Computer Interaction (HCI) Bibliography
http://hcibib.org/

Bad Human Factors Design
http://www.baddesigns.com/

User Interface Engineering
http://www.uie.com/

Heuristics for User Interface Design
http://www.useit.com/papers/heuristic/heuristic_list.html

The History of the Graphical User Interface
http://www.mackido.com/Interface/ui_history.html

Usability Evaluation Methods
http://www.usabilityhome.com/

Avaliação de Usabilidade

Foco no Desempenho de Usuários

Antonio Mendes da Silva Filho
antoniom.silvafilho@gmail.com
Professor e consultor em área de tecnologia da informação e comunicação com mais de 20 anos de experiência profissional, é autor do livros Arquitetura de Software e Programando com XML, ambos pela Editora Campus/Elsevier, tem mais de 30 artigos publicados em eventos nacionais e internacionais, colunista para Ciência e Tecnologia pela Revista Espaço Acadêmico com mais de 60 artigos publicados, tendo feitos palestras em eventos nacionais e exterior. Foi Professor Visitante da University of Texas at Dallas e da University of Ottawa. Formado em Engenharia Elétrica pela Universidade de Pernambuco, com Mestrado em Engenharia Elétrica pela Universidade Federal da Paraíba (Campina Grande), Mestrado em Engenharia da Computação pela University of Waterloo e Doutor em Ciência da Computação pela Univesidade Federal de Pernambuco.

Um produto, seja um sistema de software ou outro qualquer, possui usabilidade que é um dos atributos da qualidade perceptível aos usuários. A usabilidade é uma característica de um produto que informa quão fácil de usar e aprender esse produto é. Em outras palavras, serve como um indicador de quão intuitivo é utilizar aquele produto. Essa característica é determinante no sucesso desse produto, pois ela influencia diretamente o interesse do usuário em utilizar ou não esse produto ou sistema (de software). Para tanto, em sistemas de software, assim com em outros produtos, a usabilidade é avaliada durante o processo de desenvolvimento visando assegurar o nível desejado de usabilidade. Este artigo discute um conjunto de mecanismos de avaliação da usabilidade.

De que se trata o artigo?
Avaliação de Usabilidade e Interação Humano-Computador (IHC) nos sistemas de software.

Para que serve?
A usabilidade pode ser considerada um elemento norteador do processo de desenvolvimento, isto é, o desenvolvimento acontece sob a óptica do usuário, buscando apoiar de maneira natural a realização de suas atividades. Neste contexto, percebe-se a importância do uso de técnicas para apoiar a avaliação de usabilidade do software.

Em que situação o tema é útil?
Essencial no processo de desenvolvimento de software onde podemos ter atividades específicas para apoiar a avaliação da usabilidade da interface com o usuário.

Usabilidade

Usabilidade tem sido um dos fatores utilizados por diversas empresas para conquistar novos usuários (consumidores). Este artigo trata da usabilidade de produtos e, mais especificamente, da usabilidade de sistemas de software. Assim, os dois termos serão utilizados indistintamente neste texto.

A usabilidade é uma característica através da qual o usuário expressa sua vontade e satisfação em utilizar um produto ou serviço como, por exemplo, a compra de um livro ou DVD numa loja virtual devido à simplicidade e agilidade. Dentre esses dois motivadores, a simplicidade predomina e impacta diretamente na agilidade.

Na maioria das situações, os usuários preferem um sistema (ou produto) de fácil uso, mesmo com funcionalidades mais simples, a um sistema com mais funcionalidades, porém de manipulação complexa e não intuitiva. Dessa forma, a usabilidade é considerada no processo de desenvolvimento de software.

Embora a usabilidade seja um atributo da qualidade perceptível aos usuários quando o sistema está em uso, é trabalhada ao longo do desenvolvimento do software para que este suporte as características desejadas. O conjunto de atividades, geralmente, encontradas num processo de desenvolvimento de interface de usuário (IU) é ilustrado na **Figura 1.**

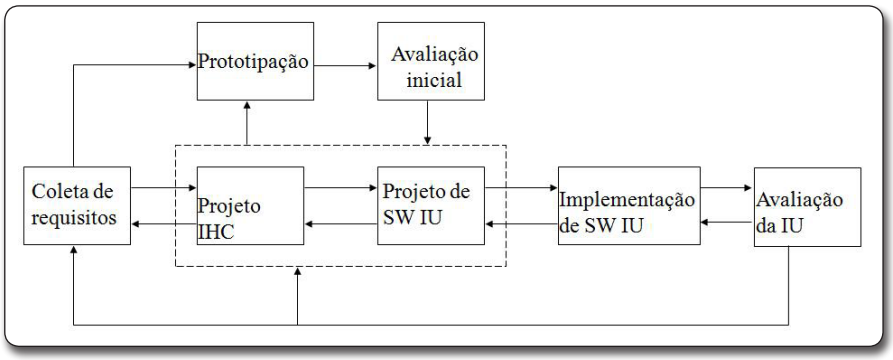


Figura 1. Atividades do desenvolvimento de interfaces de usuário.

1. Tempo para realizar um tarefa.
2. Percentual de tarefa concluído.
3. Percentual de tarefa concluído por unidade de tempo.
4. Taxa de sucessos/falhas.
5. Tempo consumido com erros.
6. Percentual de erros.
7. Número de comandos utilizados.
8. Número de comandos disponíveis não utilizados.
9. Freqüência de uso de *ajuda* (help) ou documentação.
10. Número de vezes que o usuário expressa satisfação ou frustação.

Figura 2. Critérios de medição de usabilidade.

Essas atividades podem ser encontradas em alguns processos com outras denominações. A *coleta de requisitos* tem por objetivo identificar o conjunto de funcionalidades e perfil de usuários que estarão fazendo uso delas. Trata-se de identificar as necessidades dos usuários e como elas podem ser apoiadas através da interface.

O *projeto de IHC* (interação humano-computador) ou simplesmente *projeto de interação* é uma atividade de suma importância no desenvolvimento de uma interface de usuário. O projeto de interação se apóia na modelagem (ou projeto) conceitual, onde um conjunto de tarefas dos usuários e de mecanismos de acesso a elas são identificados. No projeto de interação, o projetista faz o mapeamento de tarefas de usuários em formas específicas de diálogo entre usuário e a interface do sistema. O projeto de interação consiste de:

• *Modelagem conceitual:* onde é feita análise de elementos da aplicação e tarefas dos usuários.

• *Projeto de diálogo:* no qual se define o conjunto de ações na interface para a realização de tarefas do usuário.

• *Projeto de apresentação:* etapa onde se define o *layout* da interface e objetos de interação utilizados para apoiar a interação do usuário com a interface.

Ainda considerando a **Figura 1**, perceba que o projeto de interação é transformado no *projeto de software da interface* e, posteriormente, implementado. Nesse ínterim, é comum desenvolver o protótipo da interface, bem como fazer a avaliação preliminar da usabilidade da interface. Além disso, quando o sistema de software é finalizado, uma nova avaliação é conduzida a fim de checar se os critérios (ou métricas) de usabilidade inicialmente definidos foram atendidos.

Note que a usabilidade será um elemento norteador do processo de desenvolvimento, isto é, o desenvolvimento acontece sob a óptica do usuário, buscando apoiar de maneira natural a realização de suas atividades. Para tanto, os projetistas consideram heurísticas de projeto de interface. Há um conjunto de 10 (dez) heurísticas propostas por Jakob Nielsen que estão disponíveis em http://www.useit.com/papers/heuristic/heuristic_list.html.

Essas 10 heurísticas servem como recomendações que um projetista deveria seguir, as quais compreendem:

• exibir o estado do sistema, oferecendo visibilidade do status;

• fazer o mapeamento *natural* entre o sistema e o mundo real;

• prover suporte de controle e liberdade de escolha para usuários;

• prover consistência e seguir recomendações de padrões;

• priorizar a prevenção de erros de usuários;

• minimizar a necessidade de memorização dos usuários, priorizando o reconhecimento à recordação;

• oferecer recursos de uso eficiente e rápido da interface, tornando-a flexível e customizada às necessidades dos usuários;

• eliminar informações irrelevantes, visando prover suporte ao projeto minimalista e à estética;

• oferecer aos usuários mecanismos de ajuda que lhe permitam reconhecer, diagnosticar e tratar erros;

• prover os usuários de documentação e recursos de ajuda baseadas em tarefas dos usuários.

Conforme ilustrado na **Figura 1**, o projetista realiza avaliações durante o desenvolvimento da interface para verificar se ele tem tido sucesso em atender, por exemplo, às heurísticas citadas. Para tanto, ele considera o uso de métricas que compreendem critérios através dos quais se podem avaliar a interface de usuário. A **Figura 2** apresenta um conjunto de critérios que podem ser utilizados para avaliar a usabilidade.

Objetivando avaliar a usabilidade da interface de usuário de sistemas de software ou outros produtos, há um conjunto de métodos de avaliação de usabilidade, classificados como testes, pesquisa e inspeção, os quais são apresentados nas seções seguintes.

Testes de Usabilidade

Teste de usabilidade é uma ferramenta de avaliação da usabilidade que visa medir a taxa de sucesso que usuários conseguem utilizar e/ou aprender a usar um produto, como um sistema de software, para realizar tarefas. Em outras palavras, um teste de usabilidade objetiva avaliar se o projeto da interface atende às necessidades dos usuários.

Para que um teste de usabilidade seja realizado, faz-se necessário planejar as atividades a serem desenvolvidas e, para cada atividade de teste, devem-se elaborar os procedimentos de teste, além de preparar o ambiente no qual o teste será realizado. Outra questão essencial é a seleção dos usuários participantes do teste. Por fim, os testes são realizados no ambiente (controlado) de teste e os resultados são coletados e analisados.

Por exemplo, um teste de usabilidade pode verificar se as cores e *layout* usado numa interface são apropriados, se existe dificuldade no uso de um site de uma livraria ou imobiliária, por exemplo. Considere a **Figura 3** que capturou apenas parte de um site de uma imobiliária. Agora, suponha que temos um grupo de usuários participando de um teste de usabilidade, onde eles estão interessados em alugar um apartamento. Para realizar esta tarefa, qual dos botões, dentre os exibidos na **Figura 3**, eles deveriam clicar para realizar essa tarefa?

Num teste feito com poucas pessoas, familiarizadas com consultas na Internet, cerca de 50% sugeriu clicar na opção "Como Alugar", quase 40% sugeriu clicar em "Localize seu Imóvel" e os demais optaram por "Busca rápida". Embora esse teste não tenha sido realizado num ambiente controlado (por exemplo, um laboratório), ele serve para mostrar como testes com usuários podem fornecer dados de quão intuitiva (isto é, qual fácil usar e aprender) uma interface é. Interessante observar que quase metade dos usuários tenha escolhido a opção correta que era "Localize seu Imóvel". Outra alternativa para esta tarefa é "Busca rápida" que oferece um quantidade menor de recursos na busca, comparativamente com "Localize seu Imóvel", pois essa última oferece recursos de consulta detalhada.

Note que os dados obtidos com a realização de um teste são usados para melhorar o desempenho dos usuários por intermédio de um re-projeto, isto é, através da revisão do projeto de modo a tratar os critérios de usabilidade cujo resultado não tenha sido satisfatório. Dentro desse contexto, as heurísticas de usabilidade, propostas por Nielsen, para o projeto de interface servem como parâmetros para guiar tanto o processo de desenvolvimento quanto o de avaliação.

Os testes de usabilidade podem ser classificados ainda quanto ao propósito. Há quatro tipos que compreende:

- teste exploratório realizado no início do desenvolvimento de um produto para uma avaliação preliminar da interface;
- teste de acompanhamento que ocorre, praticamente, no meio do projeto para avaliar a interface (novamente antes de finalizar o projeto);
- teste de comparação que visa comparar dois ou mais projetos;
- teste de validação que tem por objetivo verificar a usabilidade do produto (acontecendo no final do projeto).

De um modo geral, os testes de usabilidade têm por objetivo interpretar os dados que são observados e coletados durante o teste, quando então se procede a análise desses dados checando:

- o desempenho dos usuários (i.e. a velocidade na qual eles conseguem realizar as tarefas);
- a taxa de erros cometidos quando executando tarefas;
- o tempo de aprendizagem para uso do produto ou software, no qual o usuário começa como iniciante até atingir domínio no uso do produto ou software;
- o grau de retenção das informações exibidas pela interface ao longo do tempo e nível de satisfação subjetiva (quando os usuários testados expressam sua satisfação em utilizar o produto ou software).

Embora o teste de usabilidade ajude o projetista a saber se a interface de usuário atende aos critérios, o grau de confiabilidade não é total pois você pode estar testando usuários atípicos e os resultados obtidos não podem ser generalizados a toda população de usuários. Além disso, o teste sempre ocorre num ambiente (i.e. numa condição) artificial.

Figura 3. Fragmento de interface de usuário de um site.

Investigação da Usabilidade

Investigação da usabilidade (ou *usability inquiry*) é outro método de avaliar interfaces de usuários. Trata-se de um método interpretativo uma vez que o avaliador faz a avaliação em campo, isto é, no ambiente do usuário. Diferentemente da abordagem anterior (teste de usabilidade) que privilegia a avaliação em ambiente controlado (i.e. laboratório),

o método investigativo considera:

• observar (geralmente sem qualquer intervenção) os usuários em seu ambiente de trabalho quando eles estão utilizando o produto ou software;

• o avaliador procura observar e incentivar o usuário quando ele está realizando tarefas a expressarem verbalmente como e porque eles realizam a tarefa daquela maneira;

• O avaliador também verifica e compara o que o usuário diz com o que ele, de fato, executar;

Perceba que esta técnica é não intrusiva no sentido que o avaliador tem papel passivo e cabe a ele apenas observar atentamente as ações do usuário enquanto este faz uso do sistema ou produto. Adicionalmente, o avaliador estimula o usuário para que ele realize suas tarefas:

• falando, explicando seus passos (quando executando uma tarefa);

• expressando sua satisfação ou não em usar o sistema ou produto;

• emitindo sua opinião sobre o produto ou sistema enquanto interage com ele.

Uma das formas de realizar a investigação da usabilidade é fazendo uma *investigação contextual*. Para tanto, o avaliador deve:

• selecionar um conjunto de usuários típicos;

• visitar e conhecer o ambiente no qual os usuários utilizam o produto ou sistema de software;

• solicitar aos usuários para eles mostrarem ao avaliador *como* eles executam suas tarefas (quando estão usando o produto ou sistema);

• documentar e analisar os resultados obtidos.

Note que o avaliador necessita entender e analisar as tarefas dos usuários e, para isso, trabalha com cenários de uso dos usuários e os decompõem inicialmente em tarefas discretas e, num segundo momento, em passos menores, pois esta atitude permite-lhe compreender as necessidades de interação e interface dos usuários. Além disso, ao proceder dessa forma, ele busca verificar se há situações de erro dos usuários nos cenários trabalhados.

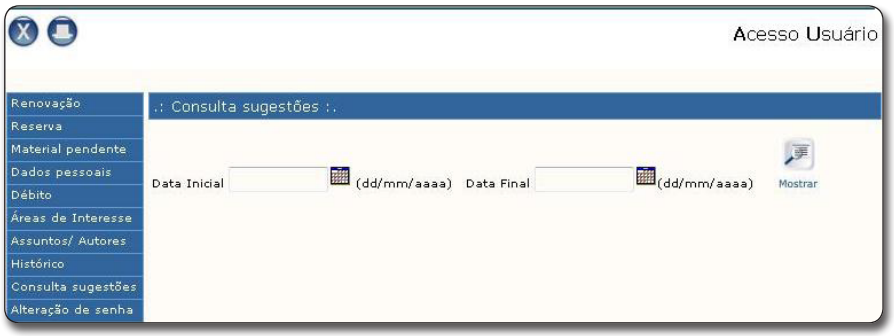


Figura 4. Fragmento de interface de usuário de um site.

Vale ressaltar que na investigação da usabilidade, o avaliador busca identificar os principais artefatos (como, por exemplo, objetos de interação e objetos físicos) usados pelos usuários quando executando tarefas. A intenção é identificar os objetos usados pelos usuários quando este está trabalhando. Além disso, observação remota pode ocorrer se houverem câmeras instaladas no ambiente do usuário. Em tal situação, a comunicação entre o avaliador e o usuário pode ser feita por telefone.

Outra atividade da investigação de usabilidade é a observação do padrão de interação entre usuário e produto ou sistema de software, procurando identificar as tarefas mais freqüentes, as quais são candidatas para ficarem em posição mais proeminente da interface ou serem acessadas por ícones ou teclas de atalho. Essa decisão visa facilitar o uso da interface. Considere, por exemplo, a **Figura 4** que ilustra parte de uma interface Web da biblioteca de uma universidade.

Durante a observação de um usuário, foi possível constatar que o usuário não conseguia localizar um botão que lhe permitisse sair do sistema, já que ele teve digitar sua matrícula e senha para ter acesso às funcionalidades mostradas na **Figura 4**. A opção encontrada foi um botão que contém um X (indicado com uma seta vermelha na **Figura 4**). Entretanto, essa opção não é a de sair ou fazer um *logout*, mas sim de fechar o *browser* (ou uma das guias do *browser*).

Note que esse tipo de observação pode ser feita tanto durante o desenvolvimento, quando ainda se tem um protótipo, como também após o sistema já estar em uso. Em ambas as situações, o objetivo é investigar se o produto ou software oferece suporte à usabilidade.

Inspeção de Usabilidade

A inspeção de usabilidade compreende técnicas como percurso cognitivo (*cognitive walkthrough*) e avaliação heurística, as quais são apresentadas a seguir. Tratam-se de técnicas onde se confia na perícia do avaliador. Neste caso, o avaliador é um especialista em usabilidade, geralmente, chamado de engenheiro de usabilidade.

Avaliação heurística

A avaliação heurística é uma técnica que permite o avaliador identificar problemas de usabilidade no projeto de interface de um produto ou sistema de software. Baseia-se em heurísticas que compreendem princípios aceitos em projetos de interface. A técnica é considerada econômica, comparativamente com testes de usabilidade, pois não requer laboratório ou qualquer outro ambiente controlado. Além disso, é fácil de ser empregada.

Essa técnica considera a participação de vários avaliadores. Recomenda-se que haja de 3 a 5 avaliadores. Inicialmente, cada avaliador realiza sua avaliação individual e apenas após terminada a avaliação é que permite-se a comunicação entre eles. Esse procedimento é adotado para se obter avaliações independentes e evitar que sejam viciadas.

O **procedimento da avaliação heurística** compreende:

• cada avaliador deve realizar sua avaliação individual;

• a avaliação individual tem duração de 60 a 100 minutos;

• durante a avaliação individual, o avaliador faz uso da interface, navegando diversas vezes pela interface com o objetivo de entender o escopo do produto ou software, para perceber o padrão de interação. Na primeira vez que o avaliador navega pela interface, sua intenção é de obter uma percepção global do produto ou software. Já num segundo momento de navegação, o avaliador inspeciona os objetos de interação e componentes de diálogo, verificando se eles atendem às heurísticas de projeto (como, por exemplo, as heurísticas de Nielsen apresentadas no início do artigo). Também, nessa avaliação individual, o especialista considera toda sua experiência;

• terminada a avaliação individual, os avaliadores se reúnem e discutem suas observações. Durante essa reunião, um especialista do domínio da aplicação pode também participar. O objetivo dessa participação é de esclarecer eventuais dúvidas pertinentes ao domínio;

• ao término da reunião dos avaliadores, uma avaliação de consenso consolida as observações deles. Essa avaliação consolidada relaciona o conjunto de problemas observados e justificativas, isto é, relaciona os problemas aos princípios (ou heurísticas) violados no projeto, segundo observações dos avaliadores.

Cabe destacar que a motivação para se ter de 3 a 5 avaliadores é devido ao fato que uma pessoa não seria capaz de perceber e achar sozinha todos os problemas.

Assim, os avaliadores trabalham com, por exemplo, as heurísticas de usabilidade propostas por Nielsen, com informações do perfil dos usuários e com cenários de uso dos usuários quando estes estão realizando tarefas típicas.

Adicionalmente, alguns especialistas sugerem que se façam testes usando, por exemplo, a técnica de *pensar em voz alta* (*Think Aloud*) como complemento à avaliação heurística. Nessa técnica, os usuários são requisitados a expressar verbalmente o que pensam e suas opiniões à medida que interagem com o sistema ou produto.

Percurso Cognitivo

Percurso cognitivo (*Cognitive Walkthrough*) é uma técnica de inspeção da usabilidade na qual de 1 a 4 avaliadores percorrem (isto é, interagem com) um conjunto de tarefas de um sistema de software ou produto avaliando quão compreensível e fácil de aprender essas tarefas são. Esta técnica prioriza a avaliação de uma interface de usuário quanto à facilidade de aprendizagem através da exploração. Isto decorre do fato que os usuários, via de regra, preferem aprender a funcionalidade de um software através de tentativas e erros a treinamentos.

A técnica de percurso cognitivo difere da avaliação heurística porque a primeira pode ser feita com um número menor de avaliadores podendo, inclusive, ser um único avaliador. Naturalmente, aplicar o percurso cognitivo com 2 a 4 avaliadores permite que problemas não percebidos por um avaliador sejam identificados por outro. Além disso, o percurso cognitivo é considerado uma técnica exploradora, pois o avaliador irá 'percorrer' a interface do produto ou software, explorando as funcionalidades oferecidas pela interface quando tenta realizar tarefas típicas dos usuários.

Vale ressaltar que esta técnica é mais comumente empregada durante o projeto de um produto ou software, uma vez que há o objetivo de verificar se os padrões de interação humano-computador atendem ou não aos critérios de usabilidade estabelecidos para o projeto. Todavia, esta técnica pode também ser aplicada em outras fases como implementação, testes e durante e após implantação do sistema de software no ambiente do usuário.

O **procedimento do percurso cognitivo** compreende:

• *identificar o perfil dos usuários do produto*: isto requer saber o grau de conhecimento que os usuários têm do produto avaliado e similares, além do conhecimento das funcionalidades oferecidas pelo produto;

• *identificar um conjunto de funcionalidades representativas*: as principais providas pelo produto ou software devem ser conhecidas, dado que elas serão percorridas (i.e. exploradas) na avaliação;

• *obter um conjunto de cenários de uso do produto ou software*: um conjunto de cenários de uso e respectivas ações para executá-los devem estar disponíveis para que eles sejam explorados durante a avaliação. Também, faz-se necessário o *feedback* ou respostas que a interface fornece às ações dos usuários. Isto visa informar aos avaliadores de como as funcionalidades podem ser acessadas para que eles possam avaliar quão intuitiva é, por exemplo, a seqüência de ações para realizar uma tarefa, tal como "Salvar como" quando o usuário intenciona atribuir um nome a um arquivo que irá salvar;

• *o avaliador percorre a interface do produto ou software*: o avaliador irá percorrer, isto é, explorar os cenários de uso daquela interface, verificar a resposta dada à sua ação, verificando a existência de problemas de usabilidade;

• *documentar problemas encontrados e elaborar sugestões*: o avaliador irá percorrer a interface do software, procurando identificar problemas de usabilidade como, por exemplo, o uso de ícones não intuitivos (ou seja, que não transmitem o significado apropriado da funcionalidade).

A identificação de problemas de usabilidade encontrados em interfaces de produtos se apóia no conhecimento que os usuários têm. Este conhecimento vem de estórias ou casos de sucesso e falhas quando usando outras interfaces.

Considere uma avaliação empregando a técnica de percurso cognitivo à interface Web de, por exemplo, uma imobiliária, como ilustrada na **Figura 5**.

Aqui, apenas parte da funcionalidade é mostrada e verifica-se que a interface oferece a funcionalidade de fazer a busca por um imóvel, que contém *layout* adequado e requer como entrada uma pequena quantidade de informações do usuário.

Entretanto, observa-se que é feito uso de dois botões: um com a função de *Buscar* e outro de *Limpar*. Enquanto no primeiro o projetista usou o termo BUSCA (substantivo), no segundo ele utilizou o termo LIMPAR (verbo).

Embora seja compreensível, não há

um padrão quanto ao uso de termos (se verbo indicando ação ou substantivo que pode representar um objeto ou coisa). Além disso, ao lado do rótulo BUSCA o projetista usou a figura de uma lupa, ícone esse geralmente é usado para este fim e de conhecimento da maioria dos usuários. Por outro lado, a figura de um √, geralmente usado em ações de Confirmar e Ok, é utilizada num contexto não adequado.

Observe que o percurso cognitivo é uma técnica na qual o avaliador coloca-se no lugar do usuário e procura 'imaginar' a experiência do usuário quando realizando a tarefa, como discutido no exemplo acima. No emprego desta técnica, o avaliador procura identificar erros ou dificuldades na navegação da interface, buscando ter acesso a funcionalidades, bem como detectar uso confuso e inconsistente de ícones, rótulos, imagens e opções.

Figura 5. Fragmento de interface de usuário de um site.

Conclusão

A usabilidade é e continuará sendo um fator determinante na escolha de produtos e sistemas de software. Sua apreciação não se limita ao apelo visual e à estética de uma interface de usuário. A usabilidade impacta o desempenho e satisfação dos usuários. Estes são aspectos importantes considerados quando da escolha de um produto. Este artigo discutiu a importância de considerar o emprego de métodos de inspeção da usabilidade necessários para verificar se o software, assim como qualquer outro, fornece as funcionalidades demandadas pelos usuários de uma maneira fácil de usar e fácil de aprender, isto é provendo suporte à usabilidade. Perceba que quão maior for a usabilidade oferecida por um software ou produto, maior a chance de sua aceitabilidade. ●

Links

Usability Net
http://www.usabilitynet.org/

Usability Gov
http://www.usability.gov/

Bad Human Factors Design
http://www.baddesigns.com/

Heuristics for User Interface Design
http://www.useit.com/papers/heuristic/heuristic_list.html

User Interface Engineering
http://www.uie.com/

Cuidados na Aplicação da Análise de Pontos de Função

Antecipando conflitos e aumentando a consistência entre contagens de pontos de função

Carlos Eduardo Vazquez

Sócio-fundador da FATTO Consultoria e Sistemas, um dos autores do livro "Análise de Pontos de Função: Medição, Estimativas e Gerenciamento de Projetos de Software". Possui 20 anos de experiência em TI, notoriamente na aplicação das disciplinas do desenvolvimento e sustentação de sistemas corporativos. Graduado em Processamento de Dados pela PUC-RJ em 1990, já passou com sucesso por quatro vezes pelo processo de certificação de especialista em pontos de função pelo IFPUG – International Function Point Users Group, tendo sido um dos primeiros brasileiros a conquistar essa certificação em 1996. Desde 1993, vem formando profissionais na aplicação da Análise de Pontos de Função, tendo sido professor da UFES, atuado como consultor de grandes projetos de tecnologia em empresas do setor financeiro, bancário e de telecomunicações.

De que se trata o artigo?

Orientação prática para um uso consistente da análise de pontos de função.

Para que serve?

Para antecipar, senão evitar conflitos, entre os diferentes tipos de usuários da técnica e permitir que os benefícios difundidos de sua aplicação possam ser efetivamente alcançados.

Em que situação o tema é útil?

Em todas as aplicações da análise de pontos de função, destacando-se para fins de estimativas e quando os pontos de função servem de insumo para monitoramento de qualidade e produtividade em contratos de desenvolvimento e manutenção de sistemas.

A técnica de Análise de Pontos de Função (APF) não é uma novidade nem tão pouco está na sua infância, afinal a pesquisa que a originou remonta ao ano de 1974 e a publicação do artigo resultante por Allan Albrecht, Measuring Application Development Productivity, ao ano de 1979. A novidade é a velocidade em que essa técnica vem ganhando adeptos para contagem de unidades para fins de remuneração das entregas de fábricas de software, fábricas de projetos e integradores de sistemas.

No Brasil, originalmente a principal aplicação da técnica é a criação de modelos de estimativa paramétrica e, ainda hoje, há quem associe imediatamente a análise de pontos de função como um sinônimo para estimativa. Estimar é determinar um valor possível para uma grandeza de interesse como o esforço para o desenvolvimento ou manutenção de sistemas, por exemplo. Um modelo de estimativa paramétrico é aquele em que esse valor é estimado a partir de um parâmetro, em contraste com os modelos de estimativa direta onde o analista estima diretamente o valor da grandeza de interesse.

Um modelo paramétrico típico é aquele em que a quantidade de pontos de função estimados é fornecida como entrada,

para o modelo fornecer a estimativa do esforço ou custo a partir dela. O mais simples desses modelos consiste em multiplicar essa entrada por uma constante que representa a produtividade, expressa normalmente pela taxa de entrega (métrica secundária que relaciona o esforço – quantidade de homens-hora despendidos na implementação de funções entregues – e a respectiva quantidade de pontos de função). Sempre que se trabalha com a taxa de entrega, é importante qualificar quais atividades e artefatos referem-se a esse esforço e observar que, quanto menor a taxa de entrega, maior a produtividade.

O processo pelo qual se obtém a constante de produtividade para um determinado contexto é chamado calibração, existindo diferentes técnicas e critérios para isso. A **Figura 1** ilustra um cenário onde o resultado da calibração da taxa de entrega é de 2,82 horas de construção e teste de unidade por ponto de função, utilizando a técnica dos mínimos quadrados.

Como a necessidade de calibração do modelo permite depreender, os pontos de função não representam esforço, prazo ou custo. Eles representam a ponderação das funcionalidades de armazenamento e transação fornecidas pelo software ou projeto aos seus usuários. Por esse motivo, a análise de pontos de função é considerada uma técnica de medição funcional. Além da confusão sobre o objeto da medição em pontos de função, os modelos de estimativa normalmente apresentam uma série de vícios na aplicação da técnica. Dentre as principais causas para isso está o conhecimento incompleto da mesma, da falta de alguns cuidados em sua aplicação e a falta de definição de premissas de contagem. Isto conseqüentemente:

(a) Leva a criação de modelos inadequados para os propósitos que deveriam endereçar. O uso da APF para fins de estimativa deve cumprir propósitos gerenciais e não operacionais. Propósitos gerenciais referem-se àqueles em que várias funções de um sistema ou projeto são avaliadas como um conjunto, enquanto propósitos operacionais referem-se àqueles onde as funções

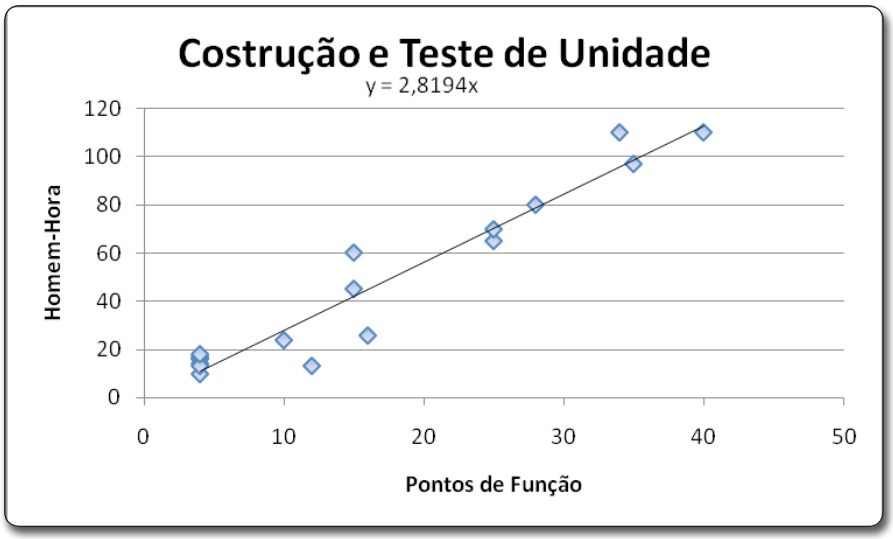


Figura 1. Gráfico de dispersão relacionando a quantidade de função entregue e o respectivo esforço necessário à sua codificação e teste unitário.

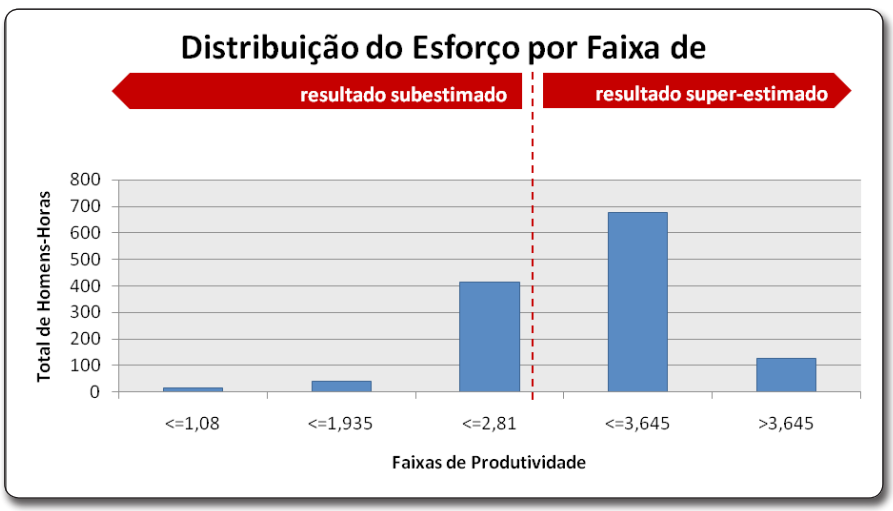


Figura 2. Distribuição do esforço total no atendimento de demandas por faixa de produtividade

individualmente, ou em pequeno número, são o objeto da avaliação. Cada função medida nessa última condição provavelmente terá uma produtividade real específica, senão única, e diferente da produtividade geral definida no modelo. Quando há determinada quantidade de funções em análise, existe uma tendência de haver uma compensação entre aquelas funções com maior produtividade real e aquelas outras onde uma menor produtividade foi aferida. Essa tendência faz com que, no todo e se bem calibrada, a produtividade do modelo aproxime-se da real.

Por exemplo, um modelo que é adequado para remunerar um desenvolvedor com base na medição de um conjunto de demandas ao longo de um mês, onde há compensação entre as diferentes produtividades aferidas, é usado para fins de planejamento de prazo e esforço de cada demanda individual, onde essa compensação não acontece. A **Figura 2** ilustra a distribuição do esforço despendido ao longo de um mês em função da produtividade expressa pela taxa de entrega. Apesar do indicador de 2,82 h/PF ser adequado para fins de estimar o esforço despendido ao longo de um mês de trabalho, se esse mesmo indicador for usado para estimar os prazos na implementação daquelas funções que colaboraram com horas à esquerda da linha vermelha pontilhada, o resultado tende a ser subestimado, enquanto os

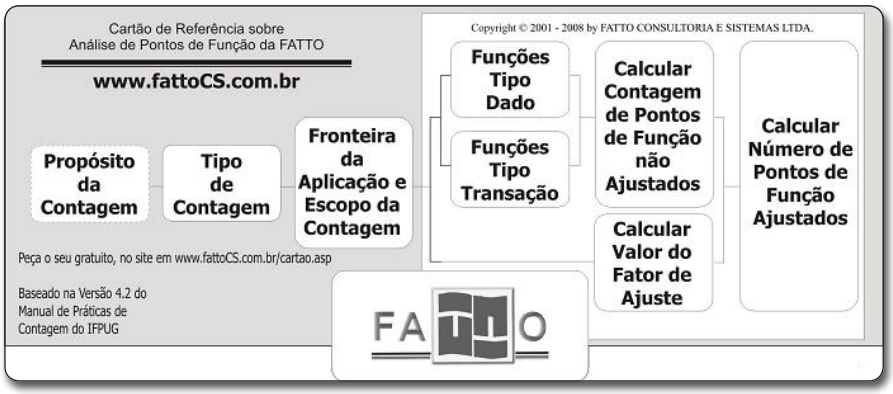


Figura 3. Processo de Contagem de Pontos de Função

prazos daquelas à direita tentem a ser superestimados. Não basta colocar mais profissionais para compensar essa tendência principalmente considerando-se projetos ou demandas pequenas como o caso em análise. A noção de que o prazo é o resultado da simples relação entre o esforço e a quantidade de pessoas que despenderão esse esforço é verdadeira dentro de certos limites. Conforme diminui a quantidade de funções em análise, mais estreito é esse limite. Isso acontece pela dificuldade em subdividir o trabalho em partes menores e da sobrecarga com as atividades de comunicação necessárias.

(b) Aumenta a chance de haver um mesmo cenário com diferentes possíveis interpretações das regras, procedimentos e práticas de contagem conforme definidos pelo International Function Point Users Group – IFPUG – em seu manual de práticas de contagem (corpo de conhecimento da técnica atualmente em sua versão 4.2.1 e geralmente chamado de CPM – Counting Practices Manual – na comunidade de analistas de métricas).

Como conseqüência desses dois pontos, aqueles que conhecem a técnica apenas superficialmente julgam-na como muito subjetiva ou ineficaz. Esse artigo tem por objetivo destacar alguns cuidados ao usar pontos de função, principalmente como unidade de medição de contratos e, com isso, ajudar a evitar que os profissionais ou organizações habituados a usar a técnica em determinado contexto vejam-se frustrados ao usá-la em outro onde é exigido um maior rigor em sua aplicação.

Alguns Cuidados no Uso da Análise de Pontos de Função

O processo de análise de pontos de função é apresentado na **Figura 3** e destaca que existem algumas premissas que devem ser estabelecidas antes da identificação e ponderação das funções em análise (propósito da contagem, tipo de contagem e fronteira da aplicação). Contudo, ao aplicar a técnica muitas vezes, essas premissas não são estabelecidas ou então não há a consistência necessária entre diferentes contagens no que se refere a elas.

O estabelecimento dessas premissas minimiza a possibilidade de divergências na contagem, destacando o posicionamento das fronteiras entre aplicações relacionadas, a definição quanto aos seus usuários e os propósitos para a contagem.

Um caso típico e que permite o entendimento da importância disso é a identificação de funções de transação. Por exemplo, imagine uma sala em um prédio e uma pessoa saindo por uma porta dessa sala para atender a uma função do negócio. Se considerarmos a sala como fronteira da aplicação, a saída dessa pessoa (função "pessoa – sair da sala") será avaliada em termos de uma função e possivelmente considerada na ponderação (outros fatores também devem ser avaliados). Contudo, se a fronteira da aplicação for o prédio, não haverá função especificamente relacionada à pessoa sair da sala. Talvez isso seja parte de alguma outra função que se manifesta para os usuários quando algum produto ou resposta a uma solicitação saia do prédio. Daí a importância de conhecer quem são os seus usuários, definir as fronteiras com base naquilo que eles entendem e por onde os produtos e solicitações serão tramitados entre o mundo do usuário e a aplicação objeto de contagem. Observe que, para fins da APF, usuário não se restringe ao usuário final da aplicação, mas também aqueles que especificam requisitos funcionais e o que quer que interaja com o sistema – inclusive outra aplicação.

Além dessas definições intrínsecas à própria APF, outras afins à sua aplicação, à sua institucionalização em uma organização ou no relacionamento entre organizações em particular também devem ser feitas como: gerência de mudança; tratamento de requisitos técnicos e de qualidade não medidos diretamente pela APF; as fases do ciclo de vida, a relação das atividades e artefatos cujo esforço para execução ou produção está incluído no cômputo da produtividade; a definição dos momentos do ciclo de vida em que haverá medição; as particularidades na aplicação da técnica conforme a seleção do tipo de contrato-remuneração por estimativa ou por medição; dentre outros.

Para fins orçamentários, de nada adianta saber qual a área de um projeto de construção de uma casa se não for conhecido qual o tipo de obra e o respectivo custo unitário. Principalmente quando se adota um modelo de estimativa onde a quantidade de pontos de função é multiplicada por um indicador de produtividade, é necessário o planejamento das estratégias para a estratificação (definição de categorias de produtividade), calibração e monitoramento do desempenho dos indicadores de produtividade.

Por fim, um cuidado para adoção da APF é a manutenção de cenários com contagens exemplo de projetos e sistemas típicos com a documentação descrevendo os erros mais comuns no uso da técnica, acertadas entre todas as organizações que utilizarão a metrificação. Pela própria natureza particular desses cenários, não é possível tratá-los de forma geral neste artigo. A idéia de manutenção desses cenários é prover uma interpretação local uniforme quanto às regras gerais definidas pelo IFPUG.

Premissas modificadas conforme a conveniência de cada contagem

Para que um determinado conjunto de atividades possa ser avaliado como uma função, esse conjunto deve também interagir com usuários da aplicação (lembre-se que para a APF, outra aplicação pode também ser considerada usuária desde que interaja com aquela em análise). Por exemplo, surge uma demanda para que seja enviado um arquivo com a relação dos clientes incluídos, alterados e excluídos de um sistema de vendas para atualização das bases cadastrais de um sistema de CRM. Na medida em que a informação está saindo da aplicação Vendas, uma saída ou consulta externa será contabilizada e, na medida em que dados estão entrando na aplicação de CRM, três entradas externas serão contabilizadas. Considerar que uma aplicação seja usuária de outra em uma determinada contagem e, em outra contagem diferente, considerar que essa mesma aplicação seja parte da outra, é um erro. Uma vez estabelecido que CRM e Vendas são duas aplicações distintas, é um erro ora considerá-las dessa forma e ora considerá-las no conjunto como uma única aplicação. Por exemplo, surge uma demanda que envolve modificar e incluir funções em ambos os sistemas. Uma das funções alteradas é o envio do arquivo com as modificações no cadastro de clientes no sistema de vendas para o CRM. Nessa nova contagem essa função não é contabilizada e esse erro em particular consiste em confundir o conceito de escopo da contagem com o de fronteira da aplicação. Resumindo o problema, fronteiras entre as aplicações não são definidas previamente e usadas consistentemente entre as diferentes contagens.

Outro exemplo, fronteiras foram estabelecidas entre os sistemas de faturamento, contábil, relacionamento com clientes (CRM) e contas a receber, quando das contagens para calibração do modelo de estimativa. Uma demanda do usuário envolve modificar um programa que prepara e envia dados de faturamento acrescentando alguns campos que serão relevantes para os sistemas que processarão esses novos campos: Contábil, CRM e Contas a receber. A contagem dessa demanda, considerando as premissas originais, identificará as funções:

- Faturamento: preparar e enviar movimento;
- Contábil: processar faturamento;
- CRM: processar faturamento;
- Contas a receber: processar faturamento.

Contudo, se quando da contagem dessa demanda todos esses sistemas estiverem compreendidos numa mesma fronteira da aplicação, provavelmente teríamos apenas a função Nota Fiscal – Faturar, contada na medida em que será o único processo que interage com o usuário.

O impacto disso é a invalidação, ou pelo menos grande degradação, de toda a referência como a produtividade média, por exemplo. Afinal, o esforço envolvido é o mesmo, independentemente de como a fronteira seja posicionada, porém a quantidade de pontos de função pode mudar drasticamente e isso muda toda relação construída anteriormente quando do estabelecimento da curva de produtividade.

Valor do fator de ajuste usado para outros fins que não aqueles para os quais foi definido

O valor do fator de ajuste se propõe a ponderar os efeitos daqueles requisitos do usuário que não são específicos de uma função, mas gerais à aplicação como um todo (desempenho, processamento distribuído, usabilidade, suporte a vários protocolos, entre outros). Em termos práticos, o fator de ajuste visa ponderar os requisitos não funcionais, havendo no CPM orientações para a sua determinação. Apesar disso, muitas vezes esse fator é usado visando à inserção de uma margem de erro na estimativa. Quando se trabalha com uma estimativa para elaboração de uma cotação de preço global fixo isso é necessário, contudo, isso deve ser feito em função do conhecimento do problema, dos riscos de novas descobertas implicarem em esforço adicional, e não a partir de uma margem de teto e piso fixos de até 35%, arbitrados pelos criadores da APF para outros fins.

O próprio uso do valor do fator de ajuste não é um consenso entre a comunidade de analistas de métricas e, quando se diz que a APF é uma técnica de medição funcional, essa parte da técnica é desconsiderada. Existem aqueles que defendem que o seu uso aumente a relação entre tamanho e esforço; outros argumentam que o fator de ajuste é obsoleto, induz muita subjetividade e que os aspectos não funcionais devem ser ponderados diretamente no esforço pelo uso de modelos como o COCOMO II ou a maior estratificação nos indicadores de produtividade utilizados para elaborar estimativas.

Quando de sua adoção, é recomendado que, no guia de contagem, estejam destacadas as orientações do CPM, preferencialmente com exemplos e casos cotidianos na organização e do cenário em que a APF será utilizado. Quando não adotado, isso também deve estar definido no guia na medida em que ele é a referência interna para o uso da APF na organização.

Todo processo de desenvolvimento ou manutenção de sistemas é também um processo de descoberta. Quando é solicitada do analista uma estimativa de esforço em momentos preliminares do ciclo de vida, as funções ainda não descobertas como requisitos terão um impacto no esforço necessário ao empreendimento em questão. Isso é um risco que deve ser tratado e a esse tratamento é dada a denominação de contingência técnica. Os mecanismos para a sua determinação devem também estar definidos, destacando que, pela comparação do tamanho funcional em diferentes momentos no ciclo de vida, é possível quantificar quanto foi o crescimento em função dessas descobertas em projetos passados. A contingência técnica, referente a esse fenômeno (scope creep), passa a poder ser medida e utilizada no processo de estimativa do tamanho funcional. Em estimativa, o problema não é errar: é desconsiderar que se vai errar!

Desconsiderar aspectos do software que não são medidos pela técnica de Análise de Pontos de Função

Os pontos de função medem os requisitos funcionais e questiona-se a eficácia do uso do fator de ajuste para fins de ponderação dos requisitos não funcionais. Existem demandas cujo esforço necessário ao seu atendimento depende primordialmente desses últimos.

Existem aquelas que nem podem ser medidas diretamente em pontos de função como, por exemplo, uma manutenção perfectiva (atividade que não acrescenta funcionalidade alguma ao software e que visa melhorar aspectos como o seu desempenho, sua usabilidade, sua facilidade de manutenção, entre outros).

Não se deve deixar de definir como esses aspectos não funcionais devem ser ponderados e ter o seu esforço estimado. Atualmente tem se tornado quase que um padrão de fato a utilização de uma tabela de "itens não mensuráveis". Por exemplo, modificar um *layout* de uma tela não implica em modificar a sua funcionalidade. Um modelo bastante comum é considerar que a cada cinco campos cujo *layout* for alterado será considerado o equivalente a 0,20 pontos de função. Contudo, o esforço na execução de atividades relacionadas aos requisitos não funcionais de um projeto não será medido diretamente e estará incluído no indicador de produtividade aplicado aos requisitos funcionais, se isso não for explícito.

Confundir esforço com prazo

A produtividade expressa em pontos de função reflete uma tendência. A lógica da utilização de um modelo como esse é de que, na média, essa taxa seja representativa da relação entre esforço (custo) e a quantidade de pontos de função entregues numa demanda ou projeto. Mesmo em situações em que individualmente não haja a compensação na produtividade, quando há muitas pequenas demandas como um contexto de manutenção intensiva, no conjunto há essa tendência para compensação. Ao longo de um mês, usar um modelo de estimativas para estimar o esforço é válido, contudo isso provavelmente não se aplique ao prazo. Duas manutenções, ambas contando seis pontos de função, podem individualmente apresentar uma grande variação no esforço e prazo da sua entrega. Considere os exemplos de demandas de manutenção:

• **Incluir um novo campo na função de inclusão de cliente.** Atualmente, a classificação do cliente é feita totalmente de forma automática pelo sistema. Esse novo campo deve conter uma nota atribuída pelo responsável do cadastramento. Ela não será armazenada, porém o resultado da avaliação automática armazenada na ficha do cliente ao final do processo será multiplicado pelo valor informado nesse novo campo.

• **Acrescentar uma crítica no campo de limite de crédito, atualmente de preenchimento livre, no cadastro de clientes.** A solicitação é que esse campo passe a ser de preenchimento obrigatório. O valor informado não deve ser superior ao resultado de determinados cálculos. Ao atualizar o cadastro, deve-se comunicar o sistema de gestão de crédito e risco. Apenas deve-se permitir a conclusão da atualização se essa comunicação for bem sucedida e não deve ser possível enviar uma comunicação em separado do próprio cadastramento.

A segunda tende a demandar mais horas de esforço e um maior prazo de entrega que a primeira, independentemente de ambas pontuarem seis pontos de função. Portanto, muita atenção ao aceitar acordos de níveis de serviço em que o prazo seja estabelecido da mesma forma que o esforço. Por exemplo, o preço ou meta de produtividade é definida em 2,81 h/PF, mas isso não quer dizer que uma demanda de 6 PF seja entregue em um prazo de cerca de dois dias. Isso é um risco para a administração do contrato por parte do desenvolvedor e é um fator que incentiva a coleta de preços mais altos no mercado por parte dos clientes. A prática atual é estabelecer limites por patamares ou então usar o resultado de modelos como o COCOMO II que fornecem um modelo mais adequado para obtenção do prazo como um de seus produtos.

Vícios na Aplicação da APF em estimativas

Quando uma organização se propõe a trabalhar com pontos de função, alguns itens também devem ser considerados no que se refere às suas práticas de estimativa, afinal o preço do ponto de função não deixa de ser baseado numa estimativa da produtividade média. Este artigo destaca alguns deles:

• A contagem dos pontos de função habitualmente é feita uma vez e em momentos preliminares do projeto quando apenas estimativas do tamanho final podem ser obtidas pela aplicação da técnica: novas contagens em momentos intermediários entre esse início e o término do projeto, ou mesmo apenas nesse último momento, não são realizadas. Portanto, o modelo não é calibrado considerando a produtividade efetiva observada naquela organização para aquele tipo de projeto, nem tão pouco segrega o que seja risco das descobertas inerentes ao processo de desenvolvimento dos riscos que afetam a produtividade;

• A identificação das funções não é feita com base na lógica do usuário, mas sim com base nos artefatos tecnológicos utilizados: por exemplo, duas tabelas que, na visão do usuário, de acordo com a sua lógica, representam um único grupo de dados relacionados entre si são contadas como duas funções distintas de armazenamento. Na lógica da modelagem de dados, nada mais natural que segregar os dados de um pedido em duas tabelas de banco de dados, porém na lógica do usuário isso não é natural. O usuário não mantém um arquivo apenas com os dados dos cabeçalhos dos pedidos e outro arquivo onde mantém os dados dos itens dos pedidos.

• O uso de um modelo único de estimativas de esforço independentemente do tamanho da demanda: o uso da análise de pontos de função para fins de estimativa requer uma quantidade de funções tal que a produtividade média seja representativa. Observando pontualmente uma função isolada, a produtividade necessária à sua entrega pode ser muito diferente da produtividade média. Quando essa mesma função é analisada conjuntamente com outras em um projeto, a partir de certo ponto, essas diferenças – a maior ou menor – tendem a ser compensadas entre si. Se o escopo daquilo que se deseja estimar é menor que 50 ou 100 pontos de função, dificilmente haverá tal compensação e o modelo de estimativas apresentará um resultado com maior erro quando comparado com aqueles obtidos por estimativas diretas, que não usam uma unidade de tamanho como parâmetro para estimar o esforço. Esses são valores empíricos, fruto das observações das aplicações da APF. Para valores menores discrepantes, ensaios devem ser feitos no contexto em que a técnica será aplicada.

Todos esses vícios no contexto de um contrato levam a conseqüências para as partes envolvidas. Quando da análise de sua produtividade, um desenvolvedor incorre nesses erros, corre um risco de determinar uma taxa de entrega que não corresponderá àquela obtida considerando a medição da entrega apurada por seu cliente. Enquanto para uma demanda o desenvolvedor chega a 120 pontos de função, o cliente chega a apenas 80 e isso implicará em um impacto na sua produtividade e, conseqüentemente, em seu preço por ponto de função.

Guia de Contagem

O Guia de Contagem é um documento particular a uma organização com interpretações locais das regras, práticas e procedimentos de contagem definidos no CPM e fundamentados na "visão do usuário". Isto é, faz com que a técnica possa ser aplicada em uma ampla variedade de situações e independentemente de como os seus requisitos funcionais são atendidos. Por outro lado, dificulta a aplicação destes conceitos em situações em que essa visão de negócio do usuário, ou pelo menos princípios que permitam a sua materialização, não esteja documentada ou definida previamente, principalmente em situações com características distintas das fornecidos pelo IFPUG.

Por exemplo, o IFPUG define que uma função de transação seja: o menor conjunto de atividades completo, que deixa o negócio da aplicação sendo contada em um estado consistente; e reconhecido pelo usuário. Em outras palavras, que não exista subconjunto desse conjunto de atividades que também seja completo. A manutenção de um cadastro com dados de referência do negócio – tarifas, por exemplo – pode ser considerada completa e reconhecida pelo usuário, porém existe a inclusão, alteração e exclusão de tarifas, parte da manutenção citada, que também pode ser considerada completa, portanto, a manutenção de tarifas não pode ser considerada uma função de transação.

Para que uma tarifa seja excluída, é necessária a realização de uma atividade de validação. Não deve haver nenhum lançamento referente a essa tarifa. O usuário reconhece essa atividade, porém ela não é completa, ela é parte da exclusão de tarifas. O usuário não executa o processo de exclusão de tarifas com o propósito de consultar se existe ou não algum lançamento para ela.

A análise, a aplicação das regras de contagem, nesse caso em particular é bastante direta e contamos com exemplos e casos fornecidos pelo IFPUG com orientação para isso. Não é o caso de sistemas de controle de tráfego aéreo em que diferentes camadas de dados são apresentadas ao usuário, sistemas com consultas dinamicamente definidas pelo próprio usuário, sistemas de workflow, entre outros.

Também para as situações onde as regras do IFPUG não estão suficientemente claras ou objetivas (existem pontos da técnica que ainda são alvo de trabalho do comitê de práticas de contagem do IFPUG como, por exemplo, a determinação quanto à contagem de diferentes funções quando houver múltiplas mídias envolvidas em seu processamento), o guia de contagem irá tratar a situação de forma objetiva. Aproveitando a questão das múltiplas mídias, um guia local pode definir que diferentes funções de transação serão contadas quando houver um desenvolvimento específico para cada diferente mídia e que apenas uma função será considerada, caso contrário.

Conclusão

Quando a APF é utilizada para fins de medição da produção na relação entre usuários e desenvolvedores, e não apenas como um instrumento interno à organização de planejamento e controle, uma série de objetivos devem ser estabelecidos para:

• **Aumentar a consistência entre contagens feitas por diferentes profissionais:** uma vez que as situações mais comuns nas medições da organização estejam exemplificadas, menor a chance de profissionais diferentes usarem critérios distintos nas medições.

• **Centralizar a experiência da contagem envolvendo diferentes tecnologias e domínios de problema (por exemplo, sistemas em mainframe, múltiplas camadas, workflow, business intelligence, batch, entre outros):** mesmo para profissionais experientes no uso da APF, quando esses se deparam com situações não usuais, é comum o surgimento de dúvidas. Com a centralização desse conhecimento, um profissional experiente na APF, mas acostumado a medir somente sistemas de determinado tipo (por exemplo, mainframe-COBOL), será capaz de medir mais facilmente sistemas de natureza distinta (por exemplo, business intelligence).

• **Evitar o re-trabalho com a análise de questões recorrentes:** uma vez que determinada situação tenha gerado dúvida na medição, esta é analisada e decide-se qual a melhor forma de abordá-la. Essa dúvida e a abordagem adotada são documentadas no guia para que todos os envolvidos em medições não precisem gastar tempo analisando situações parecidas e nem correndo o risco de decidirem por uma abordagem diferente da adotada previamente. As medições tendem a ocorrer de forma mais rápida.

• **Criar um modelo de estimativa consistente com as características da APF, calibrado e definido para os propósitos específicos a que ele se destina.**

Os cuidados apresentados neste artigo têm por objetivo servir de base para a definição de uma pauta, de um agenda, para o sucesso na adoção da APF na realidade atual e para os propósitos que vêm tão intensamente alavancando a sua utilização. ●

Referências

IFPUG – International Function Point Users Group: *http://www.IFPUG.org*

Vazques, Carlos Eduardo; Simões, Guilherme Siqueira; Albert, Renato Machado. Análise de Pontos de Função: Medição, Estimativas e Gerenciamento de Projetos de Software. Ed. Érica, 2007.

Glossário Interativo da Análise de Pontos de Função: *http://apf.locaweb.com.br/mod/glossary/view.php?id=1*

Mapa Mental Interativo da Análise de Pontos de Função: *http://www.fattocs.com.br/freemind/APF.html*

Measuring Application Development Productivity - Allan J. Albrecht: *http://www.fattocs.com.br/artigos/MeasuringAplicationDevelopmentProductivity.pdf*

Melhores Práticas na Automação de Testes

Cristiano Caetano
c_caetano@hotmail.com
É certificado CBTS pela ALATS. Diretor da TestAnywhere, consultoria de teste de software (www.testanywhere.com.br). Possui mais de 10 anos de experiência, já trabalhou na área de qualidade e teste de software para grandes empresas como Zero G, DELL e HP Invent. Autor dos livros "CVS: Controle de Versões e Desenvolvimento Colaborativo de Software" e "Automação e Gerenciamento de Testes: Aumentando a Produtividade com as Principais Soluções Open Source e Gratuitas". Criador e mantenedor do portal TestExpert: A sua comunidade gratuita de teste e qualidade de software (www.testexpert.com.br).

A automação de testes é uma área em franca expansão, no entanto, é uma área ainda muito imatura. Muitos dos sucessos nos projetos de automação de testes são decorrentes de processos empíricos de tentativa e erro.

Para piorar a situação, a automação de testes ainda é objeto de mitos que geram percepções errôneas sobre os seus reais benefícios e limitações. Freqüentemente, as ferramentas de automação de testes são supervalorizadas gerando falsas expectativas, a infra-estrutura requerida é subestimada, entre outros problemas comuns que serão apresentados ao longo deste artigo.

Nas seções a seguir serão apresentadas algumas boas práticas e conselhos reunidos pelo autor durante os últimos anos (à custa do aprendizado empírico em projetos de automação de testes).

A automação de testes não é um processo de testes

Muitas empresas fracassam na implantação da automação de testes em virtude da inversão de prioridades causada pela busca da qualidade a qualquer preço. Neste cenário, as empresas adquirem uma ferramenta de automação de testes prematuramente sem, ao menos, possuírem um grau mínimo de maturidade no processo de testes. O processo de testes é informal, as responsabilidades não são bem definidas e os profissionais não possuem o perfil adequado ou não são qualificados.

Segundo Watts S. Humphrey, criador do CMM (Capability Maturity Model), a qualidade do produto final é diretamente proporcional à qualidade do processo utilizado no seu ciclo de vida. A implantação da automação de testes depende de testes manuais maduros e consistentes. Os projetos de automação de testes bem sucedidos são aqueles que se baseiam

em processos de testes formais e estruturados. Afinal, como é possível esperar alguma coisa de testes automatizados que são baseados em testes manuais errados, ambíguos ou inconsistentes. Não é possível automatizar o caos.

A rigor, o sucesso na implantação da automação de testes depende do entendimento de que a automação de testes ou uma ferramenta de automação, por si só, não vão melhorar ou organizar os testes existentes. É necessário antes, fazer a "faxina" e implantar um processo de testes formal. A necessidade de automatizar os testes virá naturalmente como resultado da evolução da maturidade do processo de testes.

Automatize os testes críticos primeiro

Nove em cada dez projetos de automação de testes cometem o erro de transformar todos os casos de testes manuais em *scripts* de testes automatizados. O leitor deve estar se perguntando: Mas este não seria o objetivo principal da automação de testes? – Sim e Não.

A automação de testes tem o objetivo de reduzir o envolvimento humano em atividades manuais repetitivas. Entretanto, isto não significa que a automação de testes deve se limitar apenas a fazer o trabalho repetitivo e chato. Os casos de testes manuais foram criados num contexto onde os testes seriam executados manualmente. É muito provável que estes casos de testes manuais não sejam muito eficientes em um contexto onde os testes serão executados automaticamente.

Freqüentemente, antes de iniciar a automação, os testes devem ser re-projetados a fim de aumentar a probabilidade de revelar um defeito que ainda não tenha sido encontrado. Afinal, o grande benefício da automação de testes não é a execução dos testes mais rápida e a qualquer hora do dia ou da noite, mas o aumento da amplitude e profundidade da cobertura dos testes.

Na prática, a automação de 100% dos testes manuais nem sempre é a melhor estratégia. A automação de testes é pouco eficaz quando os testes são complexos e exigem interações intersistemas ou validações subjetivas (estes tipos de testes devem permanecer manuais). Além disso, muitas vezes o custo e o tempo para automatizar os testes de um projeto são maiores que o custo e o tempo do próprio projeto de desenvolvimento (o que inviabilizaria a automação de 100% dos testes manuais).

A escolha dos testes automatizados candidatos, ou seja, os mais críticos, deve ser realizada com base no contexto do projeto de automação. No entanto, apesar de não existir uma categorização amplamente difundida, a experiência tem mostrado que os testes candidatos são normalmente agrupados em quatro áreas distintas:

- **Smoke Tests**: Um conjunto mínimo de testes é selecionado com o objetivo de validar um *Build* ou liberação antes do início de um ciclo de testes;
- **Testes de Regressão**: Os testes são selecionados com o objetivo de executar o re-teste de uma funcionalidade ou da aplicação inteira;
- **Funcionalidades Críticas**: Os testes são selecionados com o objetivo de validar as funcionalidades críticas que podem trazer riscos ao negócio;
- **Tarefas Repetitivas**: Os testes são selecionados com o objetivo de reduzir o envolvimento dos testadores em atividades manuais repetitivas e suscetíveis a erros, tais como cálculos matemáticos, simulações, processamentos, comparações de arquivos ou dados, etc.

Incorpore testabilidade ao aplicativo

Via de regra, os testes são automatizados utilizando a aplicação do jeito que ela é. Ou seja, os testes são criados sob o ponto de vista da interface gráfica e com o único objetivo de automatizar as ações dos usuários finais.

No entanto, testar por meio da interface gráfica nem sempre é a melhor opção para testes que exigem desempenho. Além disso, às vezes a interface gráfica não fornece evidências de que o teste foi realizado com sucesso, afinal, nem sempre a mensagem "Essa operação foi realizada com sucesso" significa que a operação foi realizada com sucesso de verdade.

A experiência tem mostrado que a incorporação de testabilidade na aplicação é um fator chave para o sucesso de um projeto de automação de testes. A testabilidade é um atributo que determina a capacidade de uma aplicação ser testada, ou seja, a facilidade de se testar uma aplicação está diretamente ligada ao nível de testabilidade incorporado nesta aplicação.

Normalmente é necessário realizar modificações na aplicação para torná-la mais fácil de testar. Essas modificações têm o objetivo de incorporar um conjunto de mecanismos que facilitam a observação e o controle do estado dos componentes internos da aplicação. Adicionalmente, estes mecanismos expõem ao mundo exterior as funcionalidades da aplicação por meio de APIs, Interfaces de Linha de Comando, Hooks, etc.

Isto torna, por sua vez, a aplicação muito mais fácil de se testar, sob o ponto de vista da automação de testes. O objetivo desses mecanismos (APIs, Interfaces de Linha de Comando, Hooks) é fornecer interfaces para o mundo exterior que não seja dependente da interface gráfica da aplicação. Dessa forma, é possível criar testes especializados para exercitar algumas funcionalidades da aplicação sem que seja necessário utilizar uma interface gráfica.

As ferramentas de automação de testes também têm defeitos

A automação de testes é, em última instância, a execução de um software para testar outros softwares. Dessa forma, podemos afirmar que as ferramentas de automação de testes sofrem dos mesmos tipos de problemas que as aplicações convencionais, ou seja: defeitos. É muito comum ocorrerem atrasos em projetos de automação em virtude de problemas causados pela ferramenta de automação de testes. Dentre os problemas mais comuns, devemos destacar:

- Manuais incompletos e ambíguos;
- Defeitos não reproduzíveis que ocorrem aleatoriamente;
- Vazamento de memória quando a ferramenta é executada durante muitas horas;
- Degradação do desempenho quando

a ferramenta é executada durante muitas horas;

• Travamentos durante a gravação ou execução dos testes;

• Relato incorreto de testes executados com sucesso quando ocorrem problemas ou vice-versa;

• Recursos que não funcionam como deveriam;

• Reconhecimento incorreto dos componentes (botões, campos, menus, etc) da aplicação em teste;

• Defeitos de regressão (funcionalidades que não funcionam mais corretamente após um *upgrade* da ferramenta);

• Defeitos críticos que impedem a utilização das principais funcionalidades da ferramenta.

Com base no que foi exposto, a experiência tem mostrado que as seguintes ações devem ser tomadas antes da compra e durante a utilização de uma ferramenta de automação de testes:

• Criar uma prova de conceito antes de comprar a ferramenta para certificar-se que ela não tenha defeitos críticos;

• Utilizar sempre a versão mais atual da ferramenta;

• Não utilizar a versão mais atual da ferramenta (*upgrade*) sem realizar um teste de regressão (para evitar que a correção de um problema crie problemas piores);

• Se não houver solução para os problemas ou limitações da ferramenta, considere a criação de soluções alternativas (*workarounds*) utilizando a própria ferramenta.

Demo não é prova de conceito

As maravilhas oferecidas pelas ferramentas de testes automatizados normalmente são apresentadas por demonstrações (Demos). Normalmente, essas ferramentas são compradas com base na boa impressão causada durante essas apresentações.

Triste engano, infelizmente. As demonstrações normalmente apresentam cenários de utilização muito simples em um ambiente controlado. Às vezes aquela ferramenta perfeita apresentada na demonstração pode ser um completo desastre para testar a sua aplicação. Neste contexto, a melhor prática é demonstrar os benefícios, limitações e restrições da ferramenta por meio de uma prova de conceito.

Em resumo, você deve escolher os testes mais importantes e os mais complexos que serão automatizados no seu projeto de automação e criar todos os *scripts* utilizando uma versão demo ou *trial* da ferramenta de automação.

Conforme mencionado anteriormente, as demonstrações apresentam as principais funcionalidades da ferramenta de automação por meio de cenários e testes simples. Uma prova de conceito, por sua vez, tem o objetivo de provar se a ferramenta atenderá as reais necessidades do *seu* projeto de automação.

Dentre os aspectos que devem ser analisados numa prova de conceito, devemos destacar:

• Avaliar se as funcionalidades da ferramenta de automação funcionam conforme apresentado na demonstração;

• Avaliar se os manuais são adequados e informativos;

• Avaliar se existem defeitos críticos que impedem a utilização das principais funcionalidades da ferramenta;

• Avaliar se existe degradação do desempenho ou vazamento de memória quando a ferramenta é executada durante muitas horas;

• Avaliar se a ferramenta reconhece os componentes e componentes personalizados (botões, campos, menus, etc) da aplicação que será testada;

• Avaliar se a ferramenta e a linguagem de *script* oferecida é robusta o suficiente para lidar com testes complexos;

• Avaliar se a ferramenta é realmente fácil de usar e se a ferramenta oferece recursos para a criação de bibliotecas de *scripts* a fim de facilitar a reutilização;

• Avaliar se a ferramenta funciona adequadamente na infra-estrutura/ambiente existente e avaliar se será necessário algum investimento adicional para adequar a infra-estrutura/ambiente aos requerimentos mínimos exigidos pela ferramenta;

• Identificar as limitações e restrições da ferramenta para evitar falsas expectativas.

Dimensione a infra-estrutura adequadamente

Muitos projetos de automação de testes fracassam em virtude do subdimensionamento da infra-estrutura. As ferramentas de testes automatizados exigem computadores de alto desempenho com processadores rápidos e grandes quantidades de memória. Além disso, estes computadores devem ser dedicados exclusivamente para a automação de testes (durante a execução dos testes o computador não pode ser utilizado para outro fim).

Os computadores onde os testes serão executados devem ser semelhantes ou com maior capacidade de processamento em relação aos computadores utilizados para criar os *scripts* de testes automatizados, caso contrário, ocorrerão problemas de baixa de performance.

A execução dos testes automatizados é muito sensível a influências externas. Devemos reforçar que diferenças sutis no ambiente (sistema operacional, rede, versões dos programas, dados, etc) podem prejudicar ou impedir a execução dos testes automatizados.

Planejar com detalhes a infra-estrutura necessária para criar e executar os testes automatizados é um fator chave para o sucesso de um projeto de automação de testes. Dentre os aspectos que devem ser considerados, devemos destacar:

• **Computadores de alto desempenho**: Os computadores que serão usados para criar e executar os testes automatizados deve estar em conformidade com os requerimentos mínimos exigidos pela ferramenta de automação;

• **Computadores dedicados**: Os computadores onde os testes automatizados serão executados devem ser dedicados para essa função, caso contrário os testes podem ser prejudicados e pode ocorrer degradação da performance;

• **Ambiente isolado**: O ambiente deve ser restrito ao time de automação de testes para evitar a desestabilização da integridade do ambiente por meio de influências externas;

• **Ambiente controlado**: Todos os detalhes do ambiente (sistema operacional, rede, versões dos programas, dados, etc) devem ser planejados e controlados, sob

pena de prejudicar ou impedir a execução dos testes automatizados;

• **Ambiente similar ao de produção**: O ambiente onde os testes serão executados deve ser igual ou similar ao ambiente de produção, sob pena de encontrar falsos positivos, ou seja, os testes são executados com sucesso no ambiente de testes, mas as funcionalidades apresentam defeitos em produção;

• **Massa de dados consistente**: Os dados utilizados nos testes devem ser conhecidos e representativos, além disso, devem ser armazenados numa linha de base (*baseline*) a fim de permitir que sejam recuperados todas as vezes que os testes automatizados forem executados.

Encare a automação de testes como um projeto

A automação de testes é uma combinação entre teste e desenvolvimento de software, afinal, a atividade de criação de *scripts* de teste é uma atividade de programação pura. Dessa forma, podemos afirmar que a automação de testes deve ser encarada como um projeto com características próprias, ou seja, exige um planejamento detalhado, assim como, atividades de projeto, desenvolvimento e testes, tal qual o desenvolvimento de um software convencional.

Via de regra, a criação de testes automatizados é realizada por testadores especializados em desenvolvimento, também conhecidos como Testador-Desenvolvedor ou Automatizador. Este profissional deve ter o perfil de desenvolvedor e ser treinado adequadamente para utilizar a ferramenta de automação de testes.

Com base no que foi exposto, a experiência tem mostrado que os seguintes aspectos devem ser considerados em projetos de automação de testes:

• O projeto de automação deve ser encarado como um projeto com características próprias, ou seja, exige um planejamento detalhado, assim como, atividades de projeto, desenvolvimento e testes;

• Os *scripts* de testes podem ter defeitos, logo, recomenda-se a utilização de uma ferramenta de gestão de defeitos (*bugtracker*) para gerenciar estes defeitos;

• Os *scripts* de testes devem ser submetidos a um processo de Gerência de Configuração de Software e controle de versões;

• Durante o desenvolvimento dos *scripts* de testes, deve-se aplicar as melhores práticas utilizadas em projetos de desenvolvimento de software convencionais;

• O Testador-Desenvolvedor deve ter o perfil e interesse em trabalhar com automação de testes, nem sempre um bom testador ou analista de testes se tornará um bom Testador-Desenvolvedor;

• Os recursos humanos que atuarão no projeto de automação de testes devem ser treinados adequadamente. Quanto mais capacitados, maior será a probabilidade de sucesso do projeto.

Alinhe as expectativas e garanta a colaboração de todos os envolvidos

Os projetos de automação de testes costumam fracassar porque as pessoas envolvidas têm percepções diferentes sobre os benefícios e as limitações de um projeto de tal natureza.

Normalmente a alta gerência acredita que a automação de testes é a solução de todos os problemas. Os arquitetos e desenvolvedores desconhecem a necessidade da criação de mecanismos para aumentar a testabilidade durante o projeto e desenvolvimento da aplicação.

O time de testes, por sua vez, costuma acreditar que a automação dos testes é uma tarefa simples que não exige capacitação e é resumida apenas pela transformação dos testes manuais em scripts de testes automatizados por meio dos recursos de gravação (*recording*) oferecidos pelas ferramentas de automação de testes.

Essas suposições errôneas são o resultado da falta de conhecimento dos benefícios e limitações de um projeto de automação de testes. As campanhas publicitárias das ferramentas de automação, por sua vez, pioram esse cenário em virtude de que vendem uma imagem de que a automação de testes é a solução de todos os problemas de qualidade. Dentre as artimanhas de marketing mais comuns usadas pelas empresas fabricantes das ferramentas de automação, devemos destacar:

• Desenvolva software de qualidade e aumente a receita e a lucratividade da sua empresa por meio da ferramenta de testes automatizados XYZ;

• Crie testes sofisticados com treinamento mínimo;

• Utilize o recurso XYZ para gravar os *scripts* e elimine o tempo de codificação dos *scripts*;

• Aumente a produtividade e diminua o tempo gasto em manutenção por meio do compartilhamento de *scripts* e bibliotecas;

• Identifique os problemas facilmente por meio do recurso XYZ utilizado nos nossos relatórios;

• Utilize o recurso XYZ de reconhecimento automático de objetos e garanta a execução dos testes mesmo quando a interface gráfica tenha mudado.

Em projetos de automação de testes, é necessário o alinhamento das expectativas e a colaboração entre a alta gerência, desenvolvedores, arquitetos e testadores. É de suma importância explicar com detalhes os objetivos e o escopo de um projeto de automação de testes e, acima de tudo, garantir que as expectativas de todas as partes interessadas sejam realistas. Dessa forma, conseguimos eliminar o folclore e os mitos criados pelas estratégias de marketing usadas pelas empresas fabricantes das ferramentas de automação.

A estratégia recomendada para garantir o alinhamento das expectativas com relação aos benefícios e limitações de um de projeto de automação de testes deve considerar as seguintes atividades:

• Demonstrar os benefícios, limitações e restrições da ferramenta por meio de uma prova de conceito;

• Planejar com detalhes a infra-estrutura necessária para criar e executar os testes automatizados para evitar problemas de subdimensionamento;

• Envolver desenvolvedores e arquitetos no projeto de automação de testes com o intuito de incorporar mecanismos para aumentar a testabilidade da aplicação;

		Teste Informal	Teste Manual	Teste Automatizado
Investimentos	Pessoal	0	60.000	60.000
	Infra-estrutura	0	10.000	10.000
	Ferramentas	0	0	12.500
	TOTAL	0	70.000	82.500
Fase de desenvolvimento	Defeitos encontrados	250	250	250
	Custo para corrigir	2.500	2.500	2.500
Fase de testes	Defeitos encontrados	0	350	500
	Custo para corrigir	0	35.000	50.000
Produção	Defeitos encontrados	750	400	250
	Custo para corrigir	750.000	400.000	250.000
Custo da Qualidade	Conformidade (investimento em prevenção)	0	70.000	82.500
	Não Conformidade (custo para corrigir os defeitos encontrados)	752.500	437.500	302.500
	TOTAL	752.500	507.500	385.000
Retorno de Investimento		N/A	350%	445%

Tabela 1. Retorno do investimento de testes informais, manuais e automatizados.

• Envolver os testadores para alinhar as expectativas em relação aos objetivos e estratégia da automação de testes, assim como, fomentar o treinamento e a capacitação;

• Envolver a alta gerência para alinhar as expectativas em relação ao retorno de investimento, benefícios, limitações e restrições de um projeto de automação de testes.

A automação de testes é um investimento de longo prazo

A automação de testes, sem dúvida, é uma boa prática em um processo de teste de software. No entanto, a automação de testes não é uma prática que se adota do dia para a noite. Conforme mencionamos anteriormente, a necessidade de automatizar os testes virá naturalmente como resultado da evolução da maturidade do processo de testes.

Apesar de todos os benefícios advindos da automação de testes, os investimentos são altos. Rex Black, guru na área de testes de software, no seu famoso artigo chamado "Successful Investing in Software Testing" apresenta um cenário hipotético com os custos requeridos para realizar testes informais, manuais e automatizados e o retorno de investimento (ROI) obtido em cada uma dessas abordagens.

Neste artigo, Rex Black, parte da premissa de que o custo para corrigir um defeito encontrado na fase de desenvolvimento custa cerca de $10, na fase de teste custa $100 e em produção custa $1.000, como pode ser visto na **Tabela 1**.

No exemplo proposto por Rex Black, a abordagem utilizando testes automatizados garante um retorno de investimento de 445%. A mensagem é clara: a automação de testes paga o investimento. O grande problema é quanto investimento é necessário e quando esse investimento vai começar a dar retorno.

Você poderá notar que o exemplo de Rex Black considera apenas um investimento de $12.500 para a compra da ferramenta de automação de testes. Por outro lado, devemos lembrar que uma ferramenta é apenas o meio pelo qual implementamos a automação de testes, mas a automação de testes não se limita apenas à utilização de uma ferramenta.

Como vimos anteriormente, a automação de testes deve ser encarada como um projeto com características próprias, ou seja, exige um planejamento detalhado, assim como, atividades de projeto, desenvolvimento e testes, tal qual o desenvolvimento de um software convencional. Ou seja, o investimento em automação de testes não se limita apenas aos custos da ferramenta de automação de testes.

Infelizmente, muitos projetos de automação de testes costumam fracassar porque a alta gerência costuma acreditar que o único investimento necessário para a automação de testes é a compra de uma ferramenta. Entretanto, existem muitos outros investimentos necessários, tais como:

• Contratação ou capacitação de pessoal para realizar a gestão do projeto de automação de testes;

• Contratação ou capacitação de pessoal para projetar/criar os casos de testes automatizados;

• Compra ou melhoria da infra-estrutura/ambiente para atingir os requerimentos mínimos exigidos pela ferramenta;

• Gastos relacionados à incorporação de testabilidade na aplicação para torná-la mais fácil de testar;

• Gastos relacionados à criação de provas de conceitos ou protótipos.

A automação de testes é um investimento com retorno garantido quando aplicada corretamente. No entanto, o retorno é de longo prazo, ou seja, não existe retorno imediato.

Conforme mencionamos anteriormente, a automação dos testes deve começar a partir de uma prova de conceito e gradualmente crescer até atingir maturidade por meio de um conjunto de *scripts* robustos e estáveis que garantam uma cobertura de testes adequada.

O caminho para a maturidade na automação de testes é tortuoso e demorado. É necessária muita persistência para atingir a maturidade na automação de testes. A melhor prática para aumentar as chances de sucesso é evoluir iterativamente. Dessa forma, os prejuízos advindos de uma decisão errada numa iteração serão baixos e administráveis.

O teste manual é insubstituível

Segundo Cem Kaner, autor do livro "Lessons Learned in Software Testing", o propósito da automação de testes pode ser resumidamente descrito como a aplicação de estratégias e ferramentas tendo em vista a redução do envolvimento humano em atividades manuais repetitivas.

Devemos reconhecer que a automação de testes não deve ser empregada como um substituto do teste manual. Ela deve ser introduzida como uma técnica adicional cujo objetivo principal é agregar

valor, sem, no entanto, invalidar o teste manual existente. Afinal, testes manuais e automatizados são abordagens de testes diferentes que se reforçam mutuamente.

Dessa forma, por meio da automação dos testes, os testadores poderão reduzir o seu envolvimento em atividades manuais repetitivas e focar em abordagens de testes complementares que exigem a aplicação da intuição e julgamento. O teste exploratório é uma estratégia de testes complementar interessante uma vez que faz uso intensivo da intuição e julgamento do testador.

O teste exploratório é, na sua definição mais básica, a criação e a execução ao mesmo tempo de um teste. Quando se realiza um teste exploratório, normalmente o testador não tem informações detalhadas sobre o que vai testar e como vai testar. O testador se baseia na sua experiência, assim como no conhecimento que ele vai adquirindo sobre o aplicativo durante a execução do teste exploratório.

A partir dessa perspectiva, podemos afirmar que o teste exploratório é uma atividade iterativa e empírica de exploração que exige idas e vindas num processo de investigação contínuo onde a intuição, a criatividade e a experiência do testador são indispensáveis para garantir a eficiência do teste.

O grande benefício da utilização de testes exploratórios está ligado à sua natureza empírica. Os testes automatizados repetem sempre os mesmos testes, os mesmos caminhos, as mesmas operações. Por meio dos testes exploratórios temos a oportunidade de empiricamente e iterativamente criar novos testes e seguir novos caminhos, que por sua vez, nos leva a descoberta de novos defeitos.

Conclusão

A automação de testes não deve ser empregada como um substituto do teste manual. Ela deve ser introduzida como uma técnica adicional cujo objetivo principal é agregar valor. O enfoque deve ser na melhoria do processo de testes utilizado na sua empresa. A necessidade de automatizar os testes virá naturalmente como resultado da evolução da maturidade do processo de testes.

Finalmente, devemos lembrar que a automação de testes é um investimento cujo retorno é de longo prazo. O caminho para a maturidade na automação de testes é tortuoso e demorado.

Nesta jornada, provavelmente surgirão problemas imprevistos, assunções incorretas, expectativas não alinhadas, defeitos na ferramenta de automação, falta de testabilidade na aplicação, problemas na infra-estrutura/ambiente, entre outros aspectos discutidos neste artigo. O importante é aprender com os próprios erros e seguir em frente, afinal, aqueles que não conseguem se lembrar dos erros do passado estão condenados a repetí-los (George Santayanna). ●

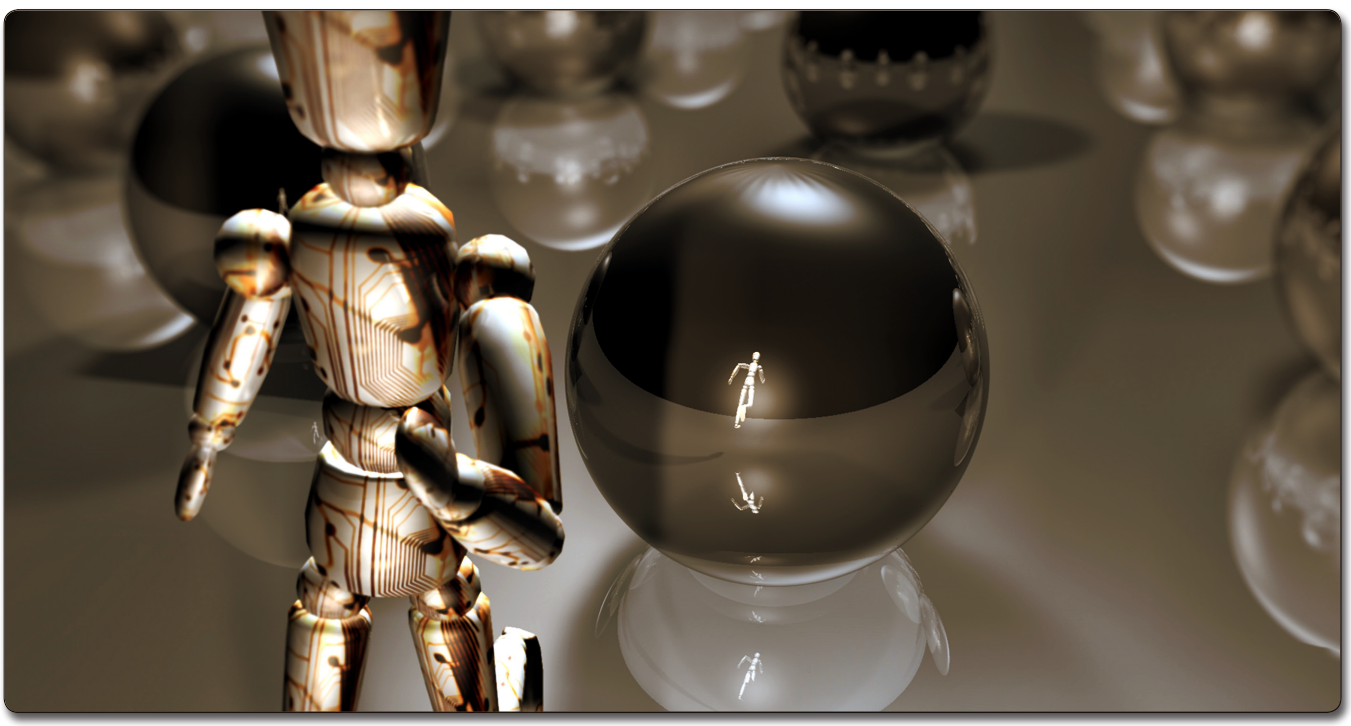
Links

Testes Exploratórios de A a Z
http://www.linhadecodigo.com.br/Artigo.aspx?id=1102

Rex Black - Successful Investing in Software Testing
http://www.rexblackconsulting.com/publications/Investing in Testing (Slides).pdf

Return on Investment Calculator for Test Automation
http://www.elbrus.com/services/test_automation_roi_calc/

Automação e Gerenciamento de Testes: Aumentando a Produtividade com as Principais Soluções Open Source e Gratuitas
http://www.linhadecodigo.com.br/EBook.aspx?id=2951



Modelagem de Processos de Negócio com Uso do NetBeans

Modelo prático em Business Process Execution Language (BPEL)

André Luiz de Castro Leal

andrecastr@gmail.com

É mestrando e especialista em Ciência da Computação pela Universidade Federal de Viçosa UFV, especialista em Gestão das Tecnologias da Informação pela Faculdade Machado Sobrinho, atua a mais de 14 anos no mercado de informática com projetos de software e atualmente coordena a equipe de desenvolvimento de sistemas computacionais do Grupo Mult unidade de Juiz de Fora. É também professor de disciplinas de Gerência de Projetos e Banco de Dados da Faculdade Estácio de Sá-JF.

Áreas de Interesse: Engenharia de Software, Qualidade de Software, Processos de Desenvolvimento de Software e SOA.

No artigo da edição anterior da Engenharia de Software Magazine, foi discutido a respeito da modelagem de processos de negócio com a utilização da notação *Business Process Modeling Notation* (BPMn) [BPMN, 2006] e a orquestração em linguagens executáveis como *Business Process Execution Language* (BPEL) [JURI, 2006] e *Web Services for Business Process Design* (XLANG) [WfMC, 2005]. Além da introdução a conceitos de BPM, foi descrito também o estado da arte da abordagem de modelagem de processos de negócio, os estudos do *Gartner* Group [NATI, 2006] com relação às expectativas e as "apostas" dos principais *players* do mercado com relação a essa abordagem.

Nesse artigo, será apresentado um paradigma prático dos conceitos apresentados anteriormente e utilizada uma IDE amplamente difundida no mercado, de fácil acesso a todos os desenvolvedores e empresas de pequeno e médio porte para que iniciem seus estudos a respeito de BPM.

De que se trata o artigo?

O artigo apresenta uma abordagem prática da utilização da IDE NetBeans para a modelagem de processos de negócio utilizando o BPEL.

Para que serve?

Orientar os desenvolvedores, através da conceituação de termos e da apresentação de um exemplo prático, de como utilizar a IDE na modelagem de processos de negócio.

Em que situação o tema é útil?

O tema tem aplicação na prática para desenvolvedores que iniciam os estudos de BPM e que podem utilizar uma IDE acessível no mercado para a modelagem e execução dos processos de negócio.

O NetBeans, desde sua versão 5.5, passou a incorporar ferramentas necessárias para escrever, testar e depurar aplicativos de arquitetura orientada a serviços (SOA), dinamizando o desenvolvimento e aprimorando a produtividade dos

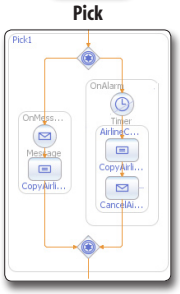
Elemento	Conceito
WSDL	*Service Web Definition Language*. Fornece um modelo em formato XML para descrição de serviços Web que possibilita uma separação da descrição das funcionalidades abstratas, oferecidas por um serviço, dos detalhes concretos, que descrevem um serviço, como "onde" e "como" as funcionalidades são oferecidas [WSDL 2003]. Dessa forma, permite uma linguagem padrão para a interação entre diferentes aplicativos.
Web Services	São componentes padronizados de parte de um aplicativo que são utilizados para a integração entre sistemas computacionais distintos. Dessa forma, os aplicativos interagem a partir de troca de mensagens em um formato padrão em XML. O principal objetivo da arquitetura *WebServices* é conectar aplicações desenvolvidas em diferentes plataformas através de um canal único e padronizado de comunicação.
Process 	É o processo principal que está modelado ou será executado, onde estão todas as relações dos elementos participantes do negócio. Atividades, regras, parceiros e iterações estão centralizados e serão executados em único fluxo. O processo será iniciado com um evento definido como *Process Start*, e será finalizado com um evento *Process End*.
Roles	São papéis que alguns elementos desempenham como participantes do processo.
Partners ou Partner Player	São elementos externos ao processo que através de papéis e regras interagem com o processo principal.
Partner Link 	Os *Partner Links* descrevem a interação entre o processo BPEL e *Web Services* externos. Nele devem estar descritos os principais papéis de cada *Partner Player* para que possam interagir através de troca de mensagens.
Partner Link Type	Utilizado para criar uma associação entre dois papéis e indicar o que cada papel deve implementar. Dessa forma, o *Partner Link Type* define as interações entre os diferentes papéis dos *Partners* no processo.
Sequence 	É a ordenação da execução das atividades do processo, as atividades são seqüenciadas e executadas de acordo com a ordem que aparecem no diagrama. As atividades são finalizadas quando a última atividade seqüenciada for completada. Podem ser adicionadas e seqüenciadas em um container de *Sequence* tantas atividades quanto forem necessárias.
Pick 	O bloco de elemento *Pick* é utilizado quando se faz necessário esperar pela execução de um determinado evento. Assim, o bloco só é executado após um evento específico ocorrer. O *Pick* possui duas ações: *OnMessage* e *OnAlarm*. *OnMessage* será utilizado para configurar o tipo de mensagem que é aguardado de um determinado *Partner*. *OnAlarm* é configurado com um *timer* indicando quando tempo o processo irá esperar um determinado evento.
Invoke 	Utilizado para invocar atividades entre o processo BPEL e um serviço do *Partner*. Esse serviço é invocado e trafegado por um endereço (porta) definido pelo *Partner*. Dessa forma, o *Invoke* permite a troca de mensagens entre o processo BPEL e os *Partners* envolvidos no processo.
Receive 	É utilizada para aguardar a chegada de uma mensagem de um determinado *Partner*. Enquanto a mensagem não é encaminhada pelo *Partner*, o processo entra em estado de espera e fica impedido de continuar ou ser finalizado.
Reply 	Utilizado pelo processo para responder uma mensagem ao serviço de um *Partner*, que tenha enviado uma mensagem anterior *(Receive)* para uma comunicação.
Assing 	Atribui valores às variáveis do processo. É também utilizado para copiar o conteúdo de uma variável para outra e para calcular expressões.
If .. elseif 	São estruturas condicionais que podem ser utilizadas, por exemplo, para a execução de uma determinada atividade caso uma determinada condição seja satisfeita.

Tabela 1. Principais termos e conceitos utilizados na modelagem BPEL.

principais aspectos dos aplicativos com a finalidade de SOA. Nesse artigo será utilizado o NetBeans 6.1 com o objetivo de se apresentar, de forma introdutória, os recursos disponíveis nesta versão para a criação de projetos com foco em modelagem de processos de negócio.

O artigo irá explicar o processo de registro de reserva de viagem que vem como exemplo na própria versão 6.1 [MAY, 2008] e que foi utilizado como exemplo de aplicação prática dos conceitos do artigo de BMP do artigo da edição anterior da revista. Dessa forma, o leitor poderá entender alguns dos principais elementos envolvidos no diagrama do processo e verificar como é possível manipular seus elementos com a IDE.

Vocabulário dos Principais Elemtnos Utilizados no Projeto

É importante, inicialmente, esclarecer alguns elementos e seus conceitos que estão direta ou indiretamente relacionados ao projeto a ser apresentado. Esses elementos estão disponíveis no *NetBeans* e são eles que tratam todo o fluxo da modelagem do processo de negócios do registro de reserva de viagem. A **Tabela 1** apresenta de forma sucinta esses elementos e seus conceitos.

Esses elementos estão disponíveis no *NetBeans* em uma paleta de ferramentas, que são utilizadas para modelar um

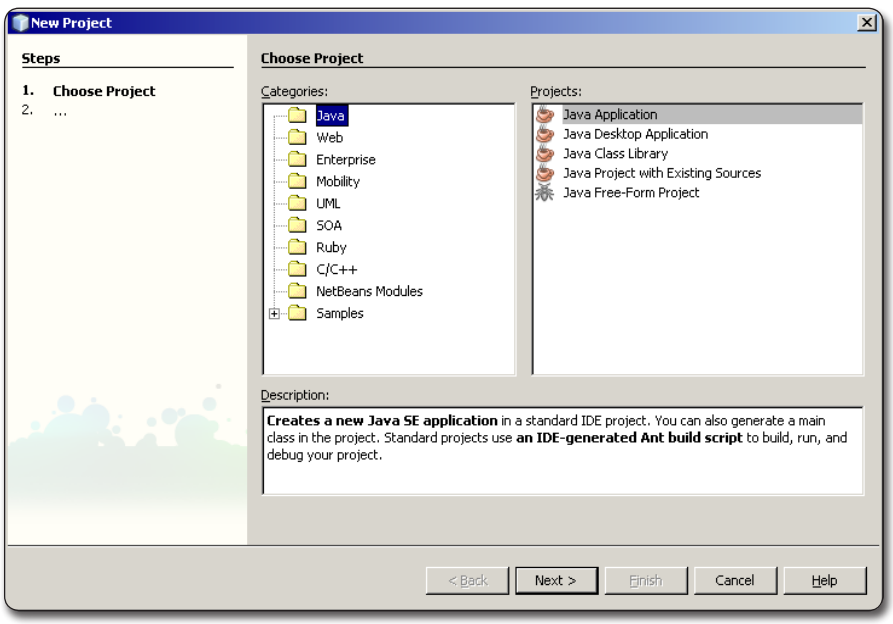


Figura 2. Tela para a criação de um novo projeto em Java.

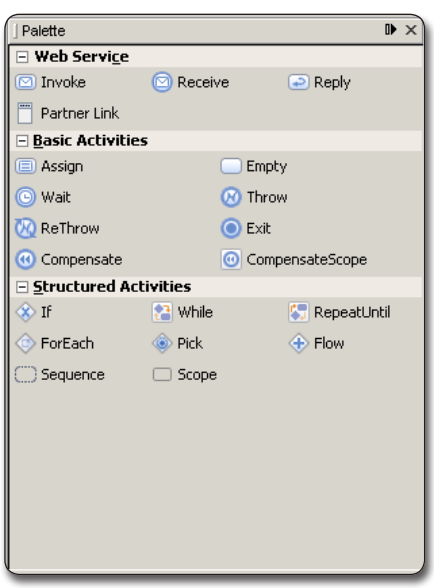


Figura 1. Paleta de ferramentas BPEL no *NetBeans*.

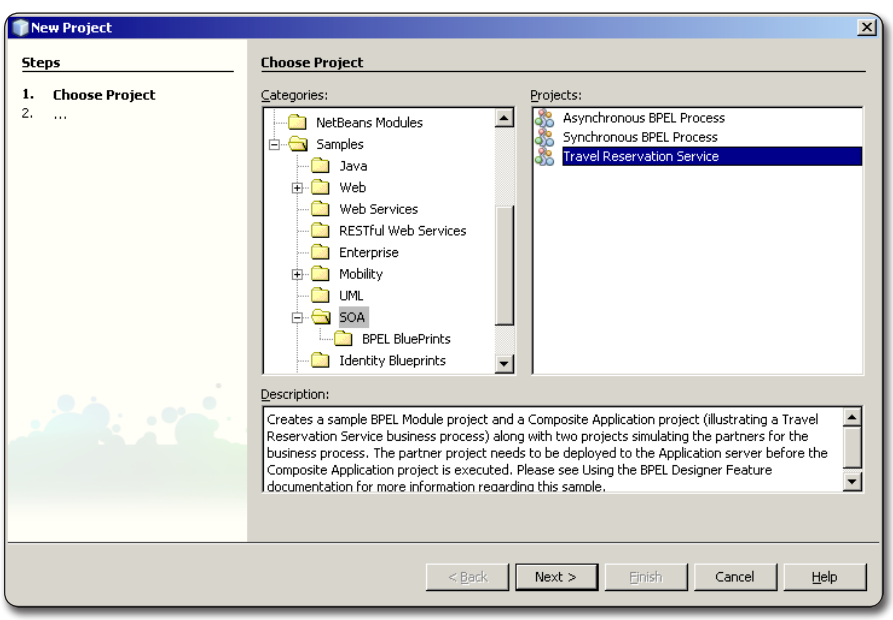


Figura 3. O projeto de exemplo.

projeto a partir de elementos do modelo BPEL. A **Figura 1** apresenta a paleta de ferramentas disponibilizadas pela IDE *NetBeans*. Caso a paleta de ferramentas não esteja disponível no projeto aberto, ela poderá ser acessada a partir do *menu Window* no sub-menu *Palette*, que poderá também ser acessada a partir da tecla de atalho *Ctrl + Shift + 8*.

Disponibilizando o Projeto de Exemplo

O *NetBeans* traz, em um dos seus exemplos, o projeto do registro de reserva de viagens implementado e disponível para obtermos as primeiras orientações sobre a modelagem na IDE.

Para disponibilizar o projeto, basta seguir os passos descritos adiante:

1) A criação de um novo projeto.

Com o *NetBeans* aberto, deve-se clicar no menu *File* e após essa ação clicar em *New Project*. A IDE irá abrir a janela apresentada na **Figura 2**.

2) Escolhendo o projeto de registro de reserva de viagem.

Após a janela aberta, o desenvolvedor deverá clicar no item do *Menu tree view* referente a exemplos do *NetBeans*. Em seguida deve escolher o item SOA. O projeto de registro de reserva de viagem irá estar disponível e pronto para ser aberto. A **Figura 3** apresenta a seqüência de opções até o projeto.

3) Parametrizando o projeto.

Após clicar no botão Next da janela anterior, o *NetBeans* irá fornecer uma tela para a parametrização do nome do projeto e a localização da pasta local onde serão criados os arquivos daquele projeto. As parametrizações podem ser informadas conforme apresentado na **Figura 4**. A pasta local deverá conter uma pasta física do ambiente do computador do desenvolvedor. Feitas as devidas configurações iniciais, o projeto estará pronto para ser aberto bastando, para isso, que o desenvolver clique no botão *Finish*.

4) O projeto aberto.

O *NetBeans* irá disponibilizar o projeto e todas as suas pastas e arquivos devidamente abertos para se iniciar a edição, alteração, execução ou qualquer ação necessária. Dessa forma, o projeto irá se iniciar conforme apresentado na **Figura 5**.

Principais Elementos do projeto de Registro de Reserva de Viagem

Para começar a entender o projeto BPEL do registro de reserva de viagens é necessário rever do que se trata esse projeto. O projeto é o modelo de um processo de registro de reserva de viagens efetuado por um cliente qualquer. Nesse processo estão envolvidas atividades do tipo fazer a reserva do hotel, da companhia aérea e do veículo.

No processo principal, serão inseridas as diversas ferramentas como: de comunicação, de condição, para invocar o processo, para receber mensagens, para dar respostas a mensagens recebidas, entre outras. O modelo, portanto, deve conter todos os elementos para que a comunicação entre os *Partners* e o processo principal aconteça e o processo possa ser executado.

Com o projeto aberto é possível encontrar e abrir o modelo BPEL principal,

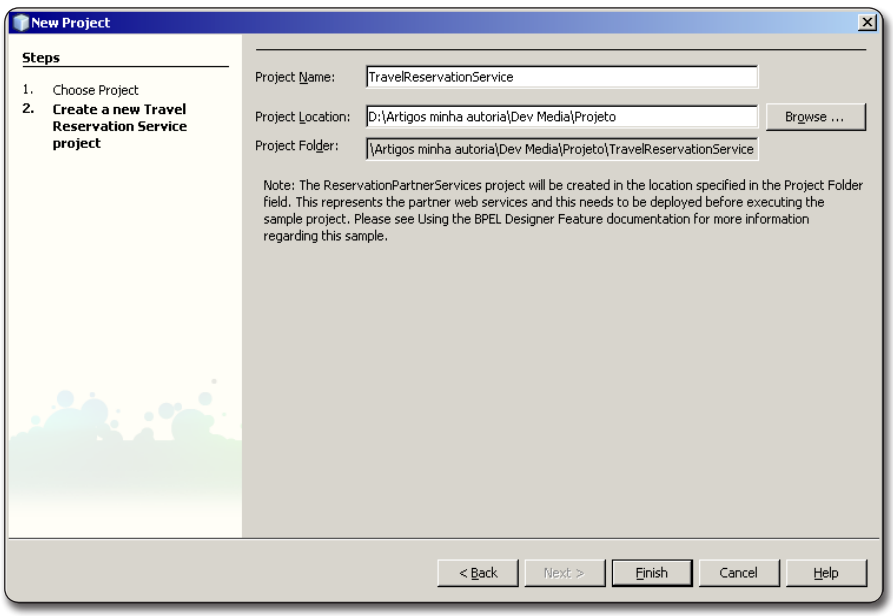


Figura 4. Parametrizações iniciais do projeto.

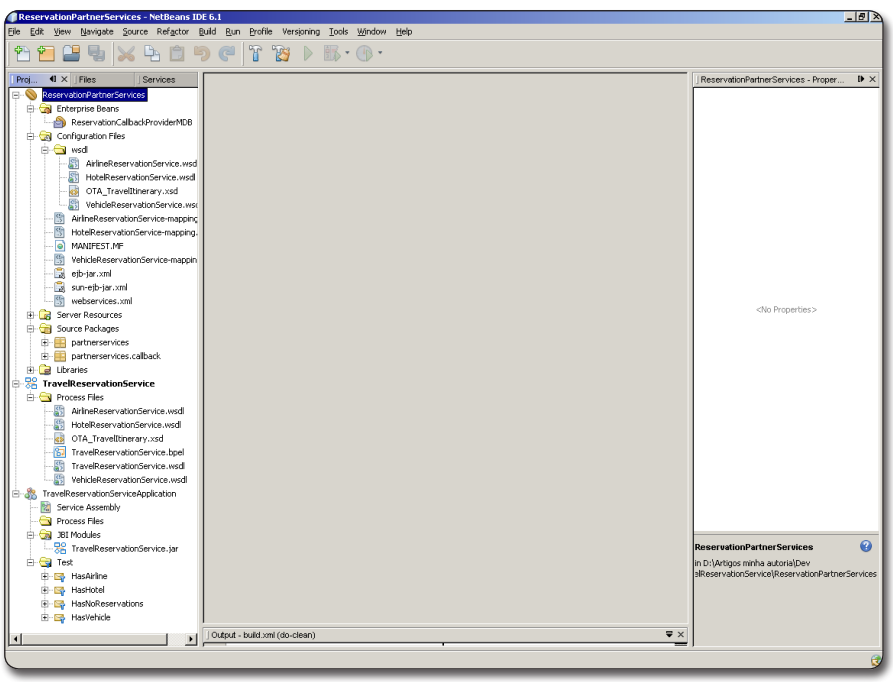


Figura 5. Apresentação de pastas e objetos disponíveis no projeto.

onde podem ser verificados todos os elementos de negócio envolvidos. Para abrir o modelo é necessário clicar na área de projetos sobre o nó da *tree view* que contenha a descrição *TravelReservationService*, após isso, clicar em *Process Files* e encontrar o objeto relativo ao *TravelReservationService.bpel*. A **Figura 6** apresenta o nó da *tree view* correspondente ao modelo BPEL. Para abrir o modelo, basta clicar com o botão da direta sobre o nó do modelo que uma caixa de diálogo irá aparecer com a opção *Open*, e em seguida clicar sobre essa opção.

O modelo BPEL será exibido conforme **Figura 7**. Note que nem todo o modelo está disponível para visualização, apenas o início do processo, os *Partners* do processo, uma *Sequence* de nome *HasAirline*, um *Pick* sem identificação e alguns elementos do processo principal.

Para continuar visualizando o processo é necessário arrastar a barra de rolagem vertical. À medida que a barra de rolagem é arrastada é possível verificar que os *Partner Links* ficam fixos no modelo e que somente o processo principal vai sendo apresentado. Dessa forma, é mais fácil a manutenção do projeto como um todo.

Os *Partners* envolvidos no processo do registro de reserva de viagem são: *Travel* (agente de viagem ou o cliente), *AirLine* (companhia aérea), *Vehicle* (Veículo) e *Hotel* (Hotel). A interação desses elementos é efetuada a partir de um processo único do serviço de reserva de viagens cujo nome no projeto é *TravelReservationService*.

Descrição do Fluxo do Processo Principal

Nesta seção será descrito o papel de cada elemento no processo. Serão abordadas as principais funcionalidades do processo e o que cada *Partner Link* aguarda do processo principal.

Descrição das Funções dos *Partner Links:*

⇨ ***Travel***: representa um cliente que envia uma requisição de viagem para o processo e, em seguida, espera recebê-la de volta após o processamento (**Figura 8**).

⇨ ***Airline***: faz uma interação com a companhia aérea para efetuar o serviço de reservas. O *Partner* irá trabalhar com três operações: a de receber um pedido de reserva do processo principal enviado pelo *Partner Travel*; receber uma mensagem de cancelamento da requisição do pedido de reserva, caso haja o esgotamento de tempo dessa requisição, e enviar uma mensagem de retorno ao processo principal no caso da reserva ser bem sucedida (**Figura 9**).

⇨ ***Vehicle***: faz uma interação com agenciadora de veículos para efetuar a reserva de locação de veículo. O *Partner* irá trabalhar com três operações: a de receber um pedido de locação do veículo do processo principal enviado pelo *Partner Travel*; receber uma mensagem de cancelamento da requisição do pedido da locação, caso haja o esgotamento de tempo dessa requisição, e enviar uma mensagem de retorno ao processo principal no caso a locação do veículo seja bem sucedida (**Figura 10**).

⇨ ***Hotel***: faz uma interação com a rede hoteleira para efetuar a reserva de hospedagem. O *Partner* irá trabalhar com três operações: a de receber um pedido de reserva hospedagem do processo principal enviado pelo *Partner Travel*; receber uma mensagem de cancelamento da requisição do pedido de hospedagem, caso haja o esgotamento de tempo dessa requisição, e enviar uma mensagem de retorno ao processo principal no caso a hospedagem seja bem sucedida (**Figura 11**).

Descrição do Fluxo do Processo Principal:

⇨ O processo é iniciado quando o *Partner Link Travel* encaminha uma requisição de reserva de viagem para o *Receive* denominado *ReceiveItinerary*, disponível no processo principal. A

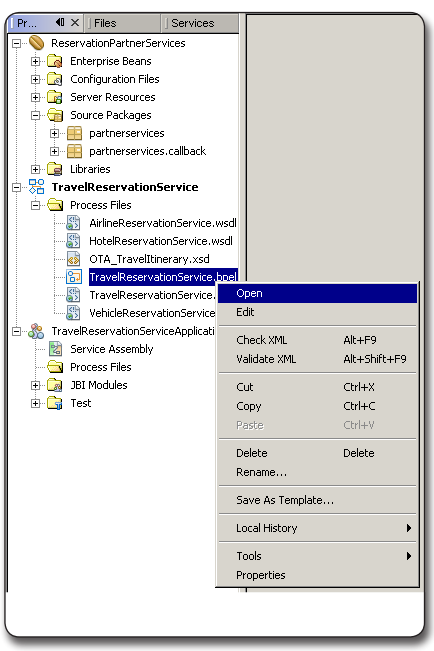


Figura 6. *Menu* de diálogo com opção para abertura do modelo BPEL.

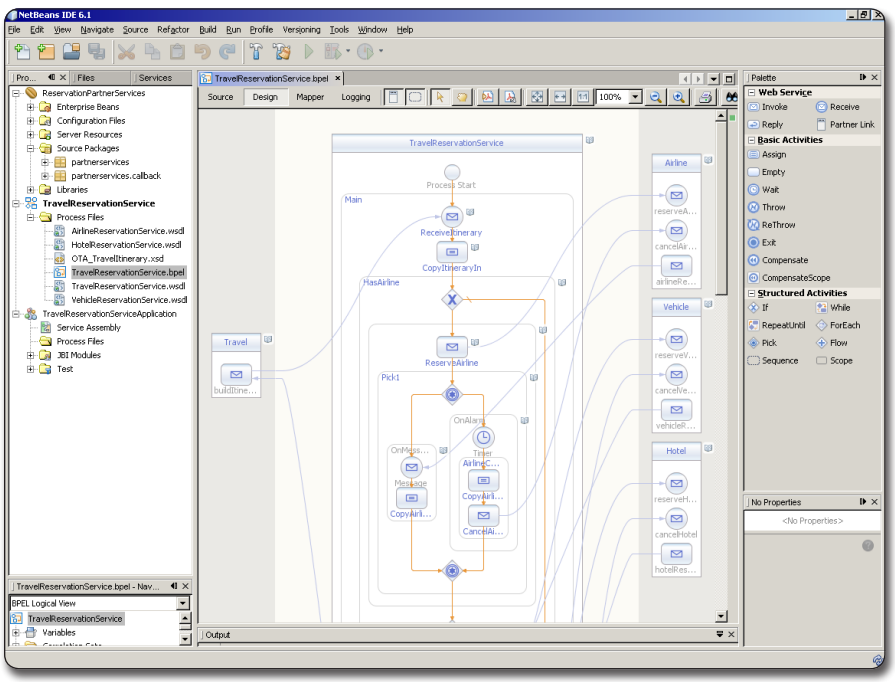


Figura 7. Modelo BPEL do registro de reserva de viagem.

Figura 8. *Partner Link: Travel.*

Figura 9. *Partner Link: Airline.*

Figura 10. *Partner Link: Vehicle.*

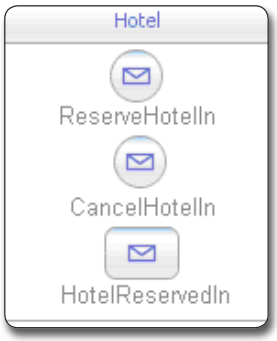


Figura 11. *Partner Link: Hotel.*

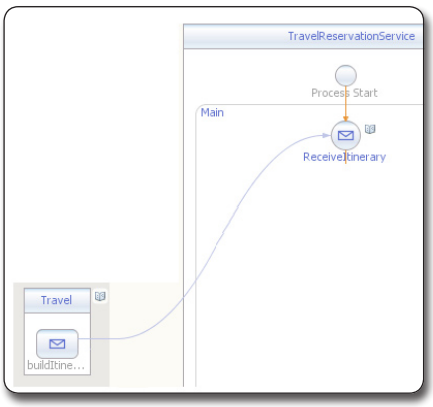


Figura 12. Chamada inicial do processo contendo as requisições de reserva.

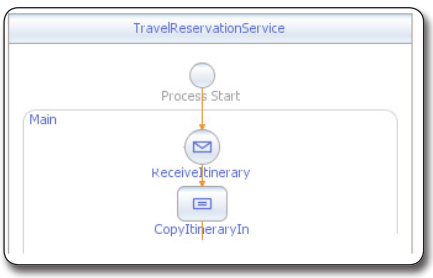


Figura 13. Passagem de dados de *input* do *Receive* para a atividade de atribuição de valores *Assign*.

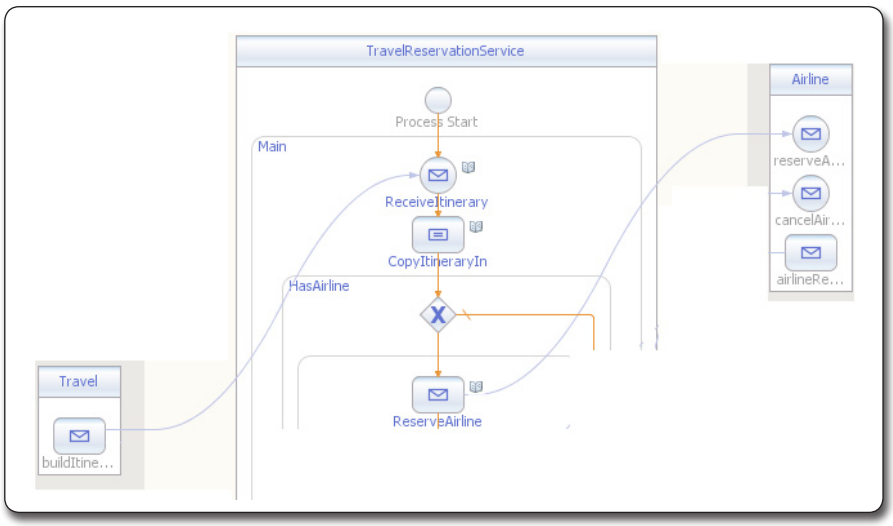


Figura 14. Chamada ao *Partner Link Airline*.

Figura 12 apresenta a chamada do *Travel* ao processo principal.

⇨ O *ReceiveItinerary* comunica com o elemento *CopyItineraryIn* (*Assign*) que copia os dados de entrada para 4 variáveis: uma variável de saída e três variáveis que serão processadas para os *Partner Links Airline, Vehicle* e *Hotel* com dados de pedidos de reserva do *Partner Travel*. É possível visualizar no destaque da **Figura 13** o *ReceiveItinerary* encaminhando os dados para a atividade de atribuição de valores *CopyItineraryIn*.

⇨ O processo principal irá verificar, a partir da atribuição de valores às variáveis, se o *Partner Travel* fez alguma solicitação de reserva.

. Se o *Travel* tiver feito uma solicitação de reserva de companhia aérea, o processo principal irá fazer uma chamada com o elemento *Invoke*, denominado *ReserveAirline*, para o *Parter Link Airline*, caso contrário o processo principal irá transpor o processamento para a próxima requisição que é a requisição de reserva de veículo. Os detalhes podem ser verificados na **Figura 14**.

. O processo principal irá repetir a mesma transação para o pedido de reserva de veículo e o mesmo acontecerá para o pedido de reserva de hotel. Da mesma forma anterior, se o *Travel* tiver feito uma solicitação dessas reservas, o processo principal irá fazer uma chamada com o elemento *Invoke* correspondente ao *Partner Link Vehicle ou Hotel*.

. Caso as requisições não tenham mensagem de reserva, o processo principal irá transpor o processamento para a próxima atividade que é efetuar uma resposta através de uma mensagem para o *Partner Travel* informando da situação que reservas não foram sucedidas e, para isso, irá acionar o elemento *Reply* denominado *ReturnItinerary* (**Figura 15**).

⇨ Para as reservas que forem bem sucedidas a partir de uma solicitação positiva de reserva do *Partner Travel*,

o processo principal irá proceder da seguinte forma:

. Para os três *Partner Links* (*Airline*, *Vehicle e Hotel*), o processo principal irá acioná-los com uma mensagem de pedido de reserva.

. Cada *Partner* irá processar a mensagem de solicitação, verificar disponibilidade de reserva e proceder com uma mensagem de resposta ao *Travel*. Os três *Partner Links* têm suas respectivas atividades de *Receive*, para o recebimento da solicitação do processo principal, e suas atividades de *Invoke*, para proceder com a resposta da solicitação para o processo principal. Dessa forma o processo principal poderá retornar ao *Partner Link Travel*, através do *Reply*, todas as mensagens recebidas dos parceiros externos ao processo.

A **Figura 16** apresenta o recebimento da chamada pelo Partner Link Airline e sua respectiva resposta ao processo principal.

⇨ No processamento das solicitações de reserva, cada elemento *Partner Link* está vinculado a um elemento *Pick*. Dessa forma, as respostas positivas ou negativas de confirmação de reserva serão encaminhadas a esse elemento e, em seguida, seguirão no fluxo do processo até o elemento *Reply*, de retorno ao requisitante *Travel*.

. No elemento *Pick* duas atividades serão executadas, no caso de uma confirmação de reserva, o elemento *Pick* irá receber a mensagem positiva em sua ação de *OnMessage*, onde o elemento *Pick* irá atribuir o valor da mensagem recebida do *Partner Link* à variável de saída para que o processo principal possa efetuar a resposta ao *Travel*.

. Cada elemento *Pick* irá também trabalhar com uma ação de temporização para aguardar uma resposta externa dos *Partners*, se essa resposta não for retornada em 20 segundos, a solicitação de reserva será cancelada.

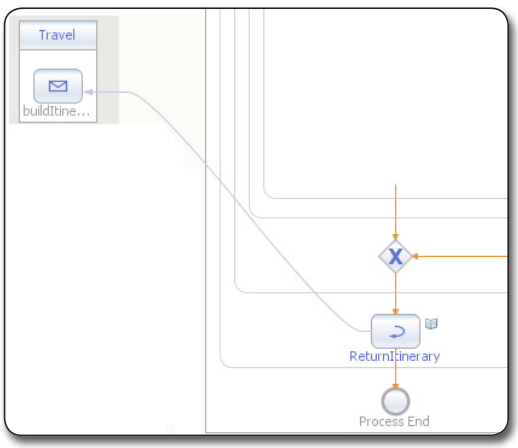


Figura 15. Retorno de resposta do *Partner Link Travel*.

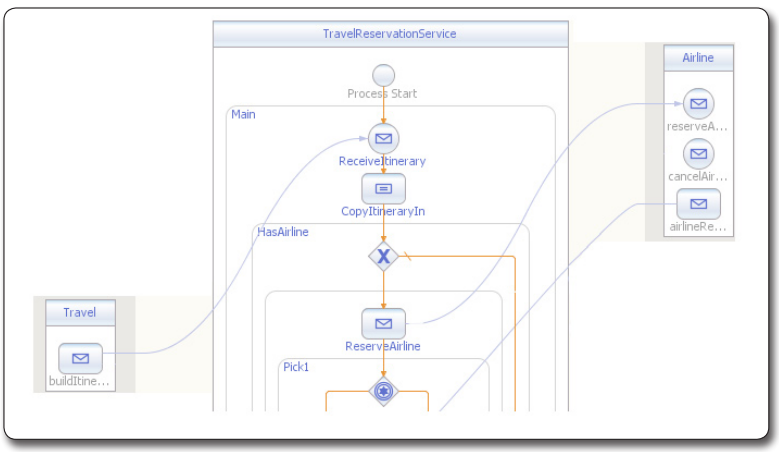


Figura 16. Fluxo de recebimento de mensagem e retorno pelo *Partner Link Airline*.

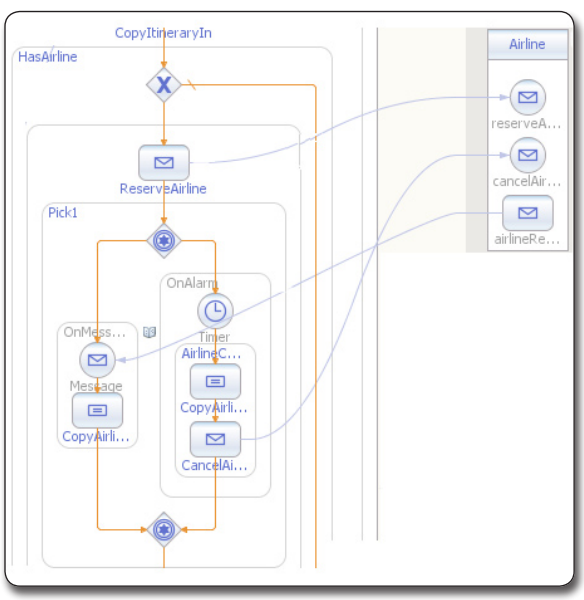


Figura 17. Comunicação entre *Airline* e elemento *Pick* do processo principal.

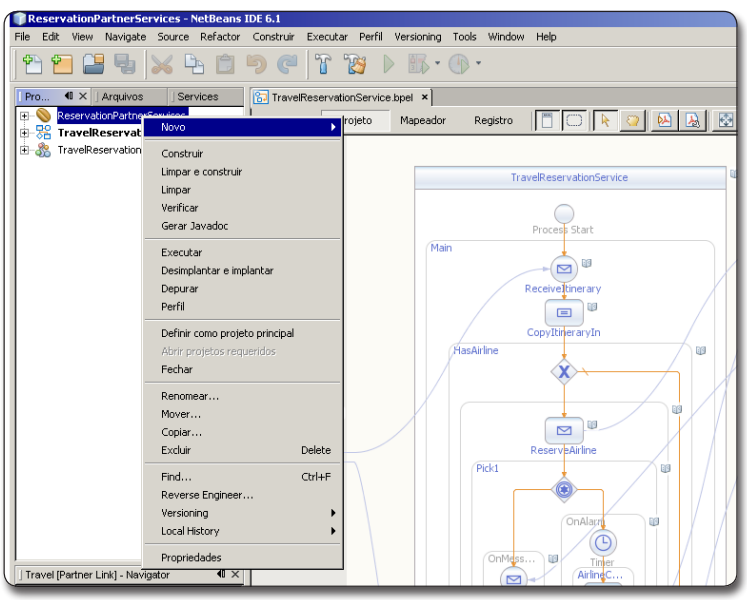
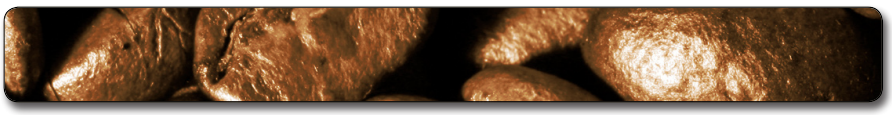


Figura 18. Menu de execução do projeto.



A **Figura 17** apresenta a transação entre o *Partner Link Airline* e o elemento *Pick* disposto no processo principal.

⇨ Após efetuado o *Reply* para o elemento *Travel*, o processo principal será encerrado.

Execução do Modelo

A execução do modelo irá fazer com que o processo principal faça a comunicação entre *Partner Travel* com os diferentes parceiros representados pelos *Partners Airline, Vehicle e Hotel*. Ou seja, independente da arquitetura de sistemas em que os parceiros externos ao processo estejam construídos, todos terão disponíveis mensagens em um padrão único de comunicação. Dessa forma, a execução do modelo BPEL permitirá uma interoperabilidade fazendo com que diferentes plataformas se comuniquem, disponibilizando para ambos os lados um padrão de mensagens em XML.

A execução do modelo poderá ser feita através do acionamento do *menu* sobre o item selecionado no projeto, conforme exibido na **Figura 18**.

Considerações Finais

O artigo trata de uma explicação a respeito dos principais elementos envolvidos no processo de registro de reserva de viagem. Seu objetivo foi esclarecer alguns conceitos utilizados na tecnologia de modelagem em BPEL e utilizar a IDE *NetBeans* para exemplificar esses conceitos.

Pelo objetivo do artigo e pela limitação de espaço desse meio de divulgação foram deixados de lado muitos elementos e conceitos. Mas a partir das explanações apresentadas, o desenvolvedor poderá iniciar a exploração da IDE *NetBeans* de forma mais rápida e avançar nos estudos dos conceitos de outros elementos que não estão sendo utilizados nesse exemplo. ●

Referências

[BPMN, 2006]
Business Process Modeling Notation Specification (2006). OMG Final Adopted Specification. February.

[JURI, 2006]
JURIC, M. B. (2006) **Business Process Execution Language for Web Services BPEL and BPEL4WS**. 2nd Edition. 372p. Packt Publishing Ltd.

[MAY, 2008]
MAY, B. (2008). **Understanding the Travel Reservation Service,** Maio 2008. Disponível em: http://www.netbeans.org/features/soa/index.html. Consultado em: 23/08/2008.

[NATI, 2006]
NATIS, Y. (2006). **Predicts 2007: SOA Advances.** EUA: Gartner, id: G00144445, Novembro.

[WfMC, 2005]
Process Definition Interface - XML Process Definition Language (2005). Workflow Management Coalition Workflow Standard. October.

[WSDL, 2003]
Web Service Description Language (2003), http://www.w3.org/TR/2003/WD-wsdl12-20030303, Outubro 2003.

Gestão de Serviços Operacionais Aplicada à TI

De que se trata o artigo?

Apesar da crescente importância da área de TI na estratégia de crescimento e ampliação dos negócios corporativos, executivos ainda encaram com muitas ressalvas o aumento dos custos dos projetos de tecnologia, dificultando a ampliação de investimentos que trarão mais agilidade e controle operacional a toda organização.

Para que serve?

Com a tendência dos custos de TI serem cada vez maiores, as organizações necessitam de mecanismos de tomada de decisões mais precisos, sobre pena de executar projetos de baixo valor agregado e deixar de executar projetos que trarão resultados financeiros mais significativos.

Estabelecer uma relação balanceada entre a área de negócios e a área de TI possibilitará uma melhor otimização dos recursos humanos e financeiros da empresa, reduzindo os riscos de solicitação de projetos de menor importância, pois todos serão cobrados pelo cumprimento das metas estabelecidas.

Em que situação o tema é útil?

Empresas que buscam novos modelos de gestão e necessitam alinhar seus projetos internos com as metas organizacionais, garantindo que o Planejamento Estratégico Corporativo está sendo seguido por todas as áreas da organização, buscando maior eficiência e rentabilidade operacional, forçando as áreas a requisitar "apenas" o que realmente trará ganhos financeiros.

Alexandre Bartie
alexandre_bartie@hotmail.com
Atua há 17 anos no gerenciamento de processos voltados à Qualidade e Engenharia de Software. Pós-Graduado em Capacitação Gerencial pela FEA-USP e em Gestão Empresarial pelo Instituto Trevisan. É Bacharel em Administração de Empresas pela Fundação Santo André. Diretor de Inovação da ALATS (Associação Latino-Americana de Teste de Software) e Engenheiro de Software responsável pelo Framework X-Zone. Autor do Livro Garantia da Qualidade de Software.

Tente imaginar uma empresa atuando em qualquer segmento de mercado, gerenciando o atual patamar de qualidade e complexidade exigidas pelos clientes, sem recorrer a nenhum tipo de estrutura computacional disponibilizada pela área de TI.

Esta empresa iria requerer uma estrutura gigantesca de profissionais, tornando a troca de informações um processo lento e altamente sujeito a falhas, resultando numa organização ineficiente e com alto nível de re-trabalho.

Apesar da evidente relevância da área de TI neste novo contexto econômico, existe um permanente descontentamento

da organização em relação ao nível de serviço prestado pela TI. Por mais avanços tecnológicos e conquistas alcançadas, parece que nunca conseguimos atingir o patamar de produtividade desejado pelos executivos.

Na verdade, algo muito semelhante ocorreu no início do século XX, quando a mão de obra profissional começou a ser substituída por máquinas automatizadas. A linha de montagem idealizada por Henry Ford organizou a cadeia produtiva em mini-etapas de simples realização, que foi gradativamente substituída por uma máquina especializada.

Apesar da infra-estrutura de produção automatizada ser muito mais cara e complexa, o nível de produtividade, velocidade e precisão deste novo modelo era incomparável. Mesmo assim, a área de produção era **constantemente massacrada** pela organização, exigindo níveis de produção cada vez maiores, exatamente para **justificar os altos investimentos** realizados. Fatores como falta de qualidade, re-trabalho, eficiência operacional eram ignorados pelos executivos, pois o que interessava eram apenas os **sucessivos recordes** de produção.

O resultado desta **pressão desenfreada** por "produção sem controle" foi o crescente aumento do estoque nas indústrias, mesmo quando estes não eram encomendados, gerando uma grave crise de rentabilidade organizacional. Com excesso de produtos em estoque, foi necessário baixar os preços para acelerar as vendas e recuperar parte dos investimentos. Enquanto isso, outros produtos que estavam sendo efetivamente solicitados pelo mercado não podiam ser produzidos por falta de matéria-prima específica ou indisponibilidade de recursos e máquinas.

As indústrias perceberam que as empresas mais lucrativas não eram as que mais produziam, mas aquelas que **inovavam suas formas de gestão operacional**, estabelecendo uma relação mais equilibrada entre o mercado e suas áreas operacionais, produzindo apenas o necessário.

Foi a partir dos anos 50, com o colapso da gestão industrial, que vários modelos foram criados para suportar uma nova dinâmica industrial voltada integralmente à **rentabilidade e eficiência corporativa**: ***Kanban***, ***Just-In-Time***, ***TOC*** (Teoria das Restrições), ***ABC*** (Custos baseado em Atividades), ***TQM*** (Gerenciamento da Qualidade Total) entre outros modelos, todos criados para otimizar toda a cadeia produtiva. Apesar de existirem a décadas, estes modelos ainda são a base das modernas formas de gestão industrial empregadas atualmente.

Esta experiência ensinou às organizações industriais a aceitarem com mais naturalidade os altos investimentos em infra-estrutura para ampliar e flexibilizar sua capacidade de produção, **valorizando o modelo de gestão operacional** como estratégia fundamental para maximizar a rentabilidade organizacional.

Como podemos observar, este **fenômeno repete-se novamente**, agora focado diretamente **na área de TI**. Assim como a área de produção no passado, nossos custos operacionais foram aumentando ano após ano e a pressão por produtividade é cada vez maior.

Da mesma forma que no passado, a área de TI deverá estabelecer um **novo modelo de gestão operacional**, nos mesmos moldes realizados pela área de produção no passado, de forma a conseguir estabelecer um maior equilíbrio entre **Negócios e TI**.

É objetivo deste artigo demonstrar como o modelo de **Gestão de Serviços Operacionais** enquadra-se perfeitamente não apenas na dinâmica de serviços da TI, mas em todas as demais áreas operacionais da organização.

Transformando a TI numa Fábrica de Serviços

Nossa tendência é acreditar que o aumento da produtividade gerado pelas evoluções tecnológicas derrubaria os custos de produção, quando na verdade, estes se tornaram cada vez maiores. Na verdade, os custos operacionais aumentam em função da demanda do mercado por produtos e serviços mais especializados e inovadores, forçando sua rápida obsolescência.

Este comportamento pode ser observado nos mais variados setores da economia - na **indústria** (com a produção de carros, aviões e navios), na **agricultura** e **pecuária** (produção de soja, milho e gado), na **medicina** (procedimentos cirúrgicos, exames e diagnósticos) ou na **engenharia** (civil, aeroespacial e militar). Os custos operacionais de um setor sempre serão maiores que no passado e tendem a tornarem-se ainda maiores com o passar dos anos.

O mesmo acontece com os custos operacionais da TI, onde manter os processos corporativos e agregar novas funcionalidades sem interromper as transações comerciais já estabelecidas, exige uma maior infra-estrutura de profissionais, ferramentas e processos de gerenciamento.

Na **Figura 1**, podemos ver que à medida que os projetos de TI tornam-se mais complexos, existe uma demanda por novos tipos de controles operacionais, que exigem da TI uma nova gama de conhecimento e técnicas a serem incorporadas nos futuros projetos.

A necessidade de áreas especializadas apoiando internamente a TI passa a ser cada vez mais estratégica ao longo dos anos, reduzindo a relevância das equipes de desenvolvimento e reforçando a importância estratégica das demais áreas operacionais.

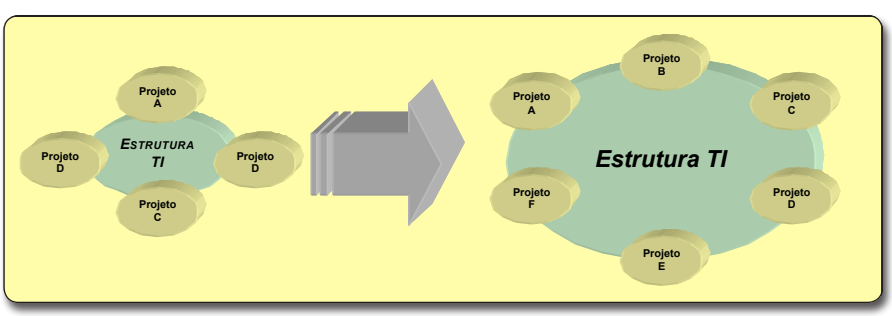


Figura 1. Tendência de aumento da estrutura da TI no longo prazo

Isto indica que as estruturas de apoio tendem a ser **maiores e mais especializadas**, oferecendo cada vez mais suporte, controle e qualidade a cada novo projeto de TI.

Entender isto como uma tendência que afeta todas as organizações permite que executivos encarem o crescimento dos custos operacionais da TI não necessariamente como algo negativo ou ligado a práticas inadequadas de gestão administrativa, mas como um custo necessário para manter-se competitivo.

A Inovação como Estratégia Corporativa

A solução para o fenômeno econômico chamado de **custos operacionais crescentes** é **ampliar a base de consumo**, objetivando incrementar as receitas corporativas. Empregando algumas estratégias de atuação de mercado, as empresas podem arcar com o crescente aumento dos custos operacionais, possibilitando manter a empresa lucrativa, sem perder sua competitividade.

Segue algumas estratégias adotadas mundialmente pelas empresas:

- **segmentação de mercado** (criação de produtos e serviços diferenciados);
- **atuação em novos mercados** (internacionalização do mercado de consumo);
- **fusões corporativas** (incorporar seu mercado e reduzir a concorrência).

Mesmo adotando todas as estratégias de ampliação do mercado de consumo, grande parte dos consumidores já possuem produtos e serviços equivalentes, reduzindo as chances de novas receitas operacionais. Desta maneira, as organizações necessitam **reduzir o ciclo de vida** de produtos e serviços, criando condições favoráveis dos consumidores substituírem seus atuais produtos e serviços por algo mais moderno e sofisticado. Trata-se de criar uma organização totalmente voltada à **inovação**.

A inovação é o **resultado de um esforço coletivo**, envolvendo diversas áreas e profissionais, que se materializa como uma **vantagem competitiva** da empresa. Poderá ser um novo produto ou serviço, um novo processo interno mais rápido e controlado, uma nova arquitetura de testes mais sofisticada e confiável, uma nova forma de reutilização de componentes, um novo acordo comercial. Enfim, a inovação é qualquer coisa que mantenha a empresa competitiva e pronta para o futuro.

A **capacidade de inovação** torna-se a característica mais importante das organizações do futuro, pois será através dela que as empresas irão surpreender seus concorrentes e ganhar espaço num mercado mais exigente e dinâmico. Disponibilizar rapidamente novos produtos e serviços, que agreguem melhorias significativas e que estejam alinhados com as necessidades e exigências do mercado é o caminho para o sucesso organizacional.

Priorizando as Inovações Corporativas

Quanto **maior importância e significado estratégico** uma inovação traz aos negócios, **mais prioritária** ela apresenta-se aos olhos da organização, tornando-se vital sua implementação. Do mesmo modo, as inovações de pouca relevância estratégica devem ser descartadas, evitando que estas concorram com inovações que efetivamente trarão maiores resultados financeiros.

Portanto, todas as inovações devem ser selecionadas e organizadas seguindo um critério de priorização, levando em consideração não apenas os benefícios gerados, mas também o esforço necessário para materializá-las.

Para facilitar o planejamento das inovações, muitas empresas utilizam-se de uma matriz de prioridades (**Figura 2**) para ser empregada no planejamento estratégico com a intenção de sistematizar o processo de decisão, auxiliando executivos a decidirem quais inovações devem ser priorizadas e quais devem ser descartadas.

Os Cuidados na Redução dos Custos Operacionais

Ao longo das últimas décadas, é inegável a contribuição da área de TI no aperfeiçoamento da qualidade e produtividade alcançada atualmente pelas organizações. Da mesma forma que as máquinas revolucionaram os meios de produção, a tecnologia da informação revolucionou a forma do gerenciamento dos serviços.

Mesmo assim, a TI ainda possui o estigma de ser um grande **custo-fixo organizacional**, associando sua imagem à de **despesas permanentes** que oneram a lucratividade. Portanto, é natural observarmos executivos gerando uma **forte pressão na redução dos custos** de TI.

O fato dos custos serem ainda o alvo preferido dos executivos demonstra que estes ainda estão **presos aos paradigmas** de uma economia de mercado pouco voltada à inovação.

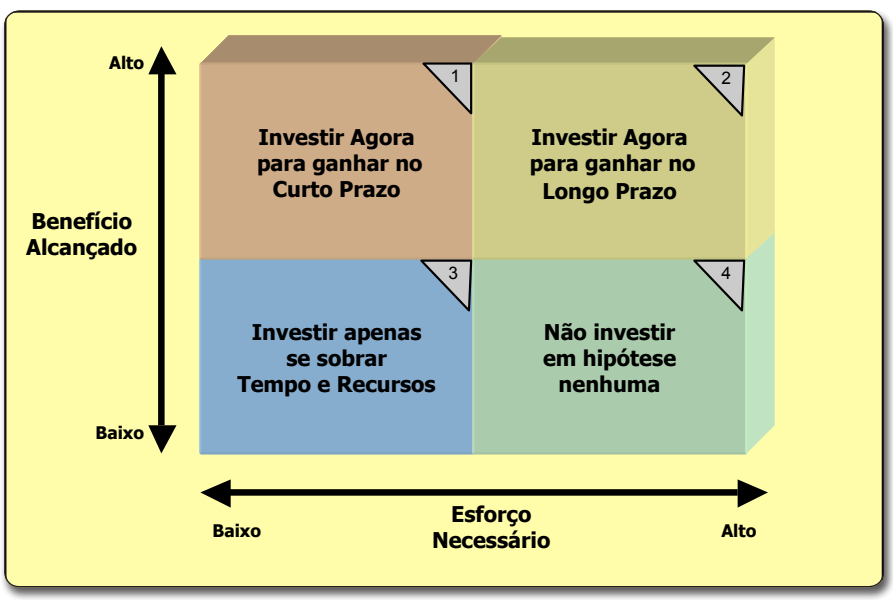


Figura 2. Matriz de Prioridades (Esforço X Benefício)

No entanto, empregar uma **política de redução de custos** sem um critério adequado pode produzir um efeito devastador na organização. Isto ocorre pelo fato de existir naturezas de custos operacionais diferentes, que devem ser compreendidas e avaliadas no momento de planejarmos uma redução dos custos corporativos.

Conforme podemos ver na **Figura 3**, existe uma grande diferença entre as **despesas operacionais**, necessárias para realização dos projetos e manter os níveis de qualidade estabelecidos, e as **perdas operacionais**, vinculadas à queda de produção, re-trabalhos constantes, erros de implementação e atrasos nas entregas.

Mesmo para **reduzir as perdas operacionais**, executivos devem estar dispostos a aumentar seus custos operacionais, realizando **investimentos** no setor que promoverão melhoria nos processos de trabalho, reduzindo as variações de qualidade e produtividades, além de inibirem falhas e erros operacionais.

Em resumo, focar exclusivamente na **redução dos custos corporativos** pode afetar diretamente na **capacidade de inovação** das diversas áreas operacionais, impossibilitando a empresa de gerar melhorias e alavancar novas oportunidades que seriam revertidas num significativo aumento da receita operacional, compensando todos os custos operacionais envolvidos.

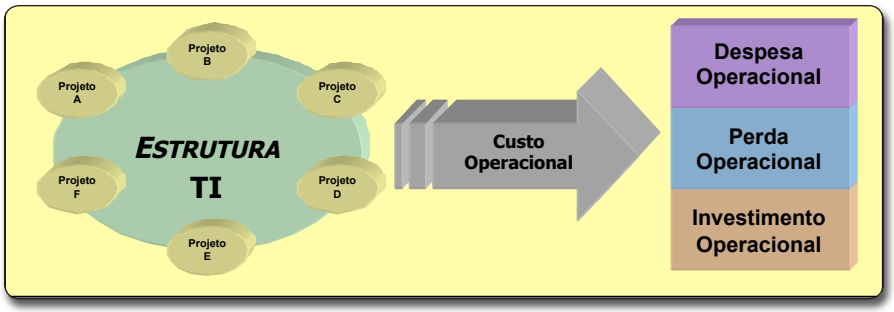


Figura 3. Classificação errada dos Custos podem afetar a Inovação Organizacional

Inovações Tendem a Aumentar Custos Operacionais

Este é um exemplo real de como uma organização deve estar aberta a ampliar seus custos operacionais, em troca de significativas vantagens no controle e agilidade de seus processos, permitindo gerenciar mais projetos e informações no longo prazo.

Neste exemplo, temos a área de TI de uma importante empresa brasileira, que possuía um gargalo operacional em seu processo de testes e homologação nas suas máquinas de auto-atendimento (ATMs). O serviço mais rentável desta empresa era oferecer a integração de sua infra-estrutura disponível, ampliando a cobertura de auto-atendimento de seus clientes em locais e regiões que não era vantajoso manter uma infra-estrutura própria.

Porém, este sistema de auto-atendimento compartilhado agregava uma alta complexidade tecnológica, pois reproduzir todas as condições de operação das várias instituições financeiras num único software exigiria uma aplicação única que suportasse um número crescente de novas funcionalidades, aumentando riscos de falhas e não-conformidades nos serviços compartilhados.

Num contexto onde novos clientes e novas transações são incorporadas permanentemente, as modificações no código-fonte eram freqüentes, aumentando os riscos de erros durante o processo de implantação. Garantir a conformidade do sistema exigiria testar todas as transações, o que demandava prazos cada vez mais longos de homologação (mais de 20 dias). O atual processo de testes e homologação restringia diretamente a capacidade da TI em gerenciar mais complexidades de forma rápida e escalável, impedindo o crescimento da rede compartilhada de auto-atendimento.

A única forma de validar uma nova versão do sistema de auto-atendimento compartilhado era exercitando o maior número de funcionalidades, incluindo as pré-existentes, possibilitando comprovar quais transações estão ou não em conformidade com os requisitos contratados. Porém, todos os testes eram feitos diretamente na máquina ATM (passar o cartão, digitar a senha, sacar o dinheiro, retirar o comprovante), o que tornava a realização dos testes lenta e sujeita a erros.

Durante todo o período de homologação do sistema, existia uma contínua interação entre as áreas de homologação e desenvolvimento, pois os relatos de problemas geravam novas versões que necessitam ser reavaliadas, tornando mais frágil toda a dinâmica de validação do aplicativo, pois novos erros poderiam ser inseridos em transações anteriormente testadas.

Alocar mais profissionais não resolveria o problema, pois além da baixa produtividade da execução de testes manuais, seria necessário ampliar o número de ATM´s disponíveis e revisar toda a infra-estrutura disponível. Repensar completamente a forma de como os testes eram organizados e executados era a única saída para este gargalo organizacional.

Para viabilizar o crescimento de longo prazo da organização e estabelecer um processo de testes e homologação mais escalável, rápido e confiável, a área de Qualidade da TI estabeleceu uma nova arquitetura de testes e homologação baseada em simuladores de dispositivos físicos e ferramentas de automação de testes.

Com isto, a empresa poderia executar todos os tipos de testes e combinações de forma contínua, construir novos casos de testes e incorporá-los a um inventário de testes, executar todas as funcionalidades mapeadas, coletar evidências dos testes e compará-las a "baselines" já aprovados, identificando automaticamente todas as não-conformidades.

Apesar da nova arquitetura de testes ser muito mais eficiente, ela traz uma carga de complexidade adicional que impacta diretamente no aumento dos custos operacionais do processo de testes e homologação. Foi necessário estabelecer um processo de testes totalmente automatizado e gerenciado por ferramentas, num ambiente controlado e de alta precisão, igual ao encontrado num ambiente industrial.

Antes, os testes eram executados manualmente por profissionais que necessitavam passar dias em pé, diante de máquinas para executar um número reduzido de casos de testes (400/dia). Agora, uma única pessoa conseguiria gerenciar facilmente 4.000 testes por dia, além de trabalhar com um processo totalmente controlado, padronizado e confiável. O prazo de homologação caiu pela metade e o nível de cobertura dos testes subiu para mais de 20 vezes.

Foram incorporados à estrutura operacional da TI novos elementos como simuladores e ferramentas de automação, que possuem custos de manutenção e licenciamento que encareceram a área de testes e homologação. Também foram necessários investimentos de consultoria para possibilitar incorporar conhecimentos e técnicas desconhecidas como automação e arquiteturas de testes baseadas em simuladores e processos automatizados.

Apesar dos **custos totais** tornarem-se **mais altos**, as inovações **reduziram as restrições operacionais** e deram um novo fôlego organizacional. Atingir este mesmo patamar de qualidade, empregando o modelo operacional de testes anterior, tornaria os custos proibitivos, inviabilizando sua implantação. Por isto, a melhor forma de avaliar os resultados de uma inovação é comparando os resultados alcançados analisando os **custos unitários** - ou seja, produzir um novo teste estava agora muito mais barato, o que permitiu a área ampliar em 20 vezes sua cobertura de testes.

Este exemplo demonstra como as inovações podem acarretar no aumento dos custos operacionais, sendo necessário sempre avaliar as vantagens competitivas que a organização obterá com este investimento, atacando as principais restrições operacionais que comprometem a qualidade e desempenho dos serviços disponibilizados.

Também demonstra que **aumentar custos operacionais não significa necessariamente ampliar o quadro de profissionais**. Na verdade, as inovações buscam sempre aumentar a produtividade, através da implementação de ferramentas, técnicas e metodologias que ampliam a capacidade dos profissionais.

Aumentar o **quadro de profissionais** é apenas uma forma simplista de aumentar o **volume de produção**, colocando mais profissionais para executar mais atividades sem nenhum compromisso no aperfeiçoamento operacional.

Recomenda-se **não ampliar** o quadro de profissionais em ambientes com **baixa produtividade**, pois além de um péssimo negócio, premia as áreas com menor eficiência e penaliza as áreas que buscaram melhorar seu desempenho operacional. Este é um dos erros mais clássicos que podemos observar nas organizações e demonstra uma total ausência de mecanismos de controle e aferição do desempenho operacional.

Entendendo o Conflito entre Negócios e TI

Com o aumento da dependência tecnológica dos negócios, a organização tende a perceber a área de TI como um **bem comum corporativo**, que pode ser **amplamente acionado**, sem restrições, uma vez que sua estrutura de profissionais, ferramentas e tecnologias estão à disposição da empresa. Este comportamento faz com que a estrutura TI seja paga na forma de **custo-fixo**, sem levar em consideração a fonte geradora das solicitações.

No modelo tradicional, as áreas que acionam a TI não contabilizam os custos quando consomem os serviços, caracterizando como "**algo grátis"** oferecido pela organização. Desta forma, quem paga pelos projetos TI são todas as áreas, quando na verdade deveria ser **apenas a área solicitante**.

Porém, percebemos que encarar a TI como um **custo-fixo organizacional** impossibilita nosso entendimento sobre a real natureza de seus custos. Isto gera uma **inconsistência** no sistema de **gerenciamento dos custos operacionais** da empresa, uma vez que as áreas solicitantes **recebem os serviços da TI gratuitamente**.

O Modelo de Relacionamento como Explicação do Conflito

Cada novo contrato de prestação de serviços estabelece uma nova relação **Cliente-Fornecedor** entre nossa organização e o mercado. Este novo contrato é recebido pelos executivos com muito entusiasmo, uma vez que se trata de uma **fonte renovada de recursos financeiros**.

Como a área de negócios é avaliada de acordo com o volume financeiro envolvido em cada contrato negociado, suas metas internas direcionam seus profissionais a buscarem novas oportunidades e ampliarem seu mercado de atuação, mantendo uma perfeita sintonia com a estratégia corporativa de **ampliação das receitas financeiras**.

Na contramão desta tendência, a área de TI recebe este novo contrato com um peso adicional à sua **estrutura-fixa**, pois será mais um projeto a ser somado a outras dezenas em andamento. Mesmo os profissionais de TI tendo consciência de que mais projetos negociados significam mais recursos financeiros para toda a empresa, eles entendem que deverão arcar com toda a responsabilidade pelas novas entregas.

Como os executivos de TI possuem uma estrutura fixa de recursos (humana e financeira), eles têm consciência de sua **capacidade de produção limitada**. Mesmo com a possibilidade de acionar empresas terceirizadas, será a própria TI que deverá negociar este "aumento de custos", o que pode gerar alguns meses de análise e justificativas, até que venha a aprovação da diretoria.

Como a área de negócios pode acionar **infinitamente** a área de TI, temos esta situação de forma recorrente dentro das organizações, onde novos contratos passam a concorrer com o andamento dos demais projetos já existentes.

Trata-se de uma **inconsistência** do atual modelo de relacionamento, pois a TI deveria ter uma **estrutura financeira variável**, que possibilita **imediata** adequação às flutuações de demandas, sem a necessidade de aprovações financeiras de executivos e dos acionistas da organização.

O atual modelo de gestão torna a organização muito vulnerável às variações de demandas, que podem acontecer a qualquer momento, com a negociação e fechamento de novos contratos.

A **cultura da TI como um "serviço grátis" leva inevitavelmente à "cultura do desperdício"**, pois incentiva as áreas

de negócios a demandarem serviços que espelham mais "desejos" do que propriamente "necessidades", o que significa que muitas das solicitações podem gerar pouco retorno financeiro para a organização.

Existe uma frase muito conhecida e citada no mundo da administração chamada "não existe almoço-grátis", onde reforça exatamente que qualquer esforço produz um custo e este deverá ser repassado, numa analogia direta aos modelos de gestão de Custos ABC e modelos equivalentes de gestão de custos.

Talvez a forma mais simples de apresentar a natureza de conflito entre as áreas de Negócios e TI seja demonstrar como o modelo de relacionamento pode gerar comportamentos indesejáveis e prejudiciais à organização.

Para exemplificar a chamada "**cultura do desperdício**", podemos realizar um pequeno estudo estatístico, avaliando **dois grupos de controle** distintos. No primeiro grupo, 50 pessoas terão R$ 10,00 para gastar num restaurante onde é cobrado por quilo, enquanto que outras 50 pessoas entrarão no mesmo restaurante e pagarão os mesmos R$ 10,00, porém poderão consumir tudo o que quiserem (bebida, comida e sobremesa).

O resultado deste estudo será sempre o mesmo. O primeiro grupo apesar de ter os mesmos R$ 10,00 para gastar, necessitou fazer uma análise de custo-benefício e pedir apenas o necessário, **otimizando sua solicitação** e requisitando apenas o essencial. Após a refeição, as pessoas apresentam-se satisfeitas, pois **pediram tudo que precisavam** – em alguns casos receberam até o troco.

No segundo grupo, não havia restrições estabelecidas, e as pessoas sentiram-se totalmente à vontade em experimentar de tudo. Isto ocorre porque as pessoas buscaram **maximizar financeiramente** seu pedido, ou seja, quanto mais itens consumiam, maior valor teria seus R$ 10,00. Após a refeição, as pessoas apresentam-se satisfeitas, pois **pediram tudo que desejavam**.

Isto demonstra claramente que os dois grupos adotaram comportamentos diferentes por que o modelo de relacionamento estabelecido entre "restaurante e cliente" os conduziram inconscientemente para esta situação. Comparando os grupos de controle, fica fácil evidenciar que o **modelo de relacionamento pode gerar maior ou menor nível de desperdício,** mesmo que de forma involuntária e inconsciente.

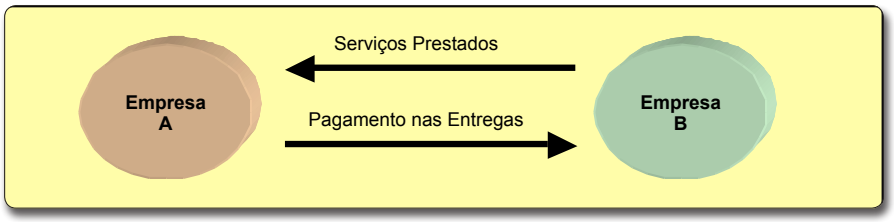


Figura 4. Relação Cliente-Fornecedor empregado entre Empresas

A Dinâmica de um Relacionamento Equilibrado

Podemos descrever a **dinâmica do mercado** como uma **relação de trocas contínuas entre empresas** que buscam atender suas necessidades diversas. O sucesso de uma organização pode ser medida pela capacidade de atender adequadamente necessidades geradas pelo mercado.

Ao atender uma necessidade, a organização estabelece uma **relação de troca**, fornecendo o **serviço requerido** e sendo adequadamente **remunerado pelo esforço demandado**. Para manter a competitividade na prestação de serviços, as organizações investem mais na qualificação de seus profissionais, de forma a atender cada vez mais o nível de qualidade, produtividade e controle exigidos pelos clientes.

Neste modelo de relacionamento, as empresas possuem um perfeito relacionamento, pois possuem um mecanismo de troca equilibrado. Quanto maior o volume de projetos solicitados pelo **Cliente**, mais recursos financeiros serão repassados ao **Fornecedor**. Aumentar o escopo e complexidade das solicitações também irá encarecer o projeto do Cliente, criando um natural mecanismo de limitação de exigências para quem solicita os serviços.

Conforme podemos observar na **Figura 4**, o conceito de **Gestão de Serviços** é naturalmente adotado entre as empresas, exatamente por propiciar uma relação financeira justa e equilibrada, possibilitando que flutuações de demandas do **Cliente** sejam automaticamente repassadas ao **Fornecedor** de serviços.

Classificando os Modelos de Relacionamento

Ao compararmos a dinâmica de relacionamento entre empresas, com a dinâmica entre as áreas de **Negócios e TI**, observamos que alguns fundamentos da relação **Cliente-Fornecedor** não são respeitados – a remuneração proporcional ao serviço executado.

A **remuneração proporcional** (também chamada de **variável**) é o mecanismo que equilibra a relação entre as partes, pois onera o cliente proporcionalmente à sua exigência, enquanto que o fornecedor receberá de acordo com o serviço demandado.

Não existindo a proporcionalidade, as exigências do cliente **tendem ao infinito**, enquanto que a estrutura do fornecedor **mantém-se limitada**. Este tipo de relacionamento desequilibrado é chamado de **Patrão-Empregado,** pois lembra os contratos CLT, onde profissionais recebem um valor fixo no final do mês, independentemente de seu volume de produção.

Abaixo, aprofundaremos mais sobre as características e comportamentos associados às duas formas de relacionamento identificadas:

Relacionamento Patrão-Empregado (Modelo Atual)

As relações **Patrão-Empregado** são baseadas no modelo **ganha-perde**, onde independente do volume de serviços prestados pelo empregado, o **Patrão** realiza periodicamente pagamentos de valor fixo, em troca apenas da permanente disponibilidade do **Empregado**.

Na perspectiva do **Patrão**, a relação mais vantajosa será adotar uma estratégia de

maximizar sua capacidade produtiva, criando um ambiente onde o **Empregado** possa produzir cada vez mais, sem que o preço-fixo estabelecido seja alterado.

Já na perspectiva do **Empregado**, a relação mais vantajosa será adotar a estratégia de **minimizar seu esforço produtivo**, exatamente o oposto do que objetiva o **Patrão**. Neste contexto, o **Empregado** busca criar um ambiente onde possa produzir o mínimo possível, uma vez que sua remuneração já estar estabelecida.

Neste tipo de relação, existe um **conflito permanente** entre **Patrão e Empregado**, pois ambos buscam maximizar os ganhos desta relação. Desta forma, apenas um terá uma relação mais favorável, uma vez que o **modelo financeiro não é dinâmico**, pois não está vinculado aos níveis de serviços estabelecidos.

Na relação **Patrão-Empregado**, existe sempre a promessa de recompensa futura para o bom desempenho, apesar de tratar-se sempre de uma análise subjetiva e não formalizada entre as partes.

Portanto, o **Empregado** assume integralmente o risco de ampliar seu trabalho e promover algum tipo de inovação para a organização. O **Patrão** define o momento da avaliação, estabelecendo os critérios que quantificam o valor da inovação, julgando o mérito ou não da recompensa.

Relacionamento Cliente-Fornecedor (Modelo Proposto)

As relações **Cliente-Fornecedor** são baseadas no modelo **ganha-ganha**, onde o volume de serviços prestados é proporcional ao volume financeiro a ser repassado, estabelecendo uma relação balanceada entre aumento de demanda e readequação da infra-estrutura para manter os níveis de serviços exigidos.

Neste tipo de relação, o **Fornecedor** procura continuamente oportunidades e novos serviços para oferecer ao seu **Cliente**, pois necessita manter-se viável financeiramente, ampliando seu mercado de atuação e reforçando sua atuação nos Clientes já conquistados.

Como outras empresas também buscam ampliar suas receitas financeiras, o **Fornecedor** sempre buscará encontrar formas mais eficientes de execução dos serviços prestados. Por iniciativa própria, o **Fornecedor** buscará melhorar seu desempenho, implementando sempre inovações que agreguem maior valor aos seus projetos.

A relação **Cliente-Fornecedor** é naturalmente estabelecida entre empresas, exatamente por incorporar uma dinâmica financeira que oscila em sintonia com as demandas de serviços – quanto mais serviços forem executados, mais pessoas são necessárias para realizar os projetos, maiores serão os esforços e custos envolvidos, maiores serão as receitas financeiras previstas.

Demonstrando os diferentes Modelos de Alocação de Custos

Para que possamos demonstrar como os modelos de relacionamento influenciam diretamente no modo de alocação dos custos, vamos empregar um exemplo

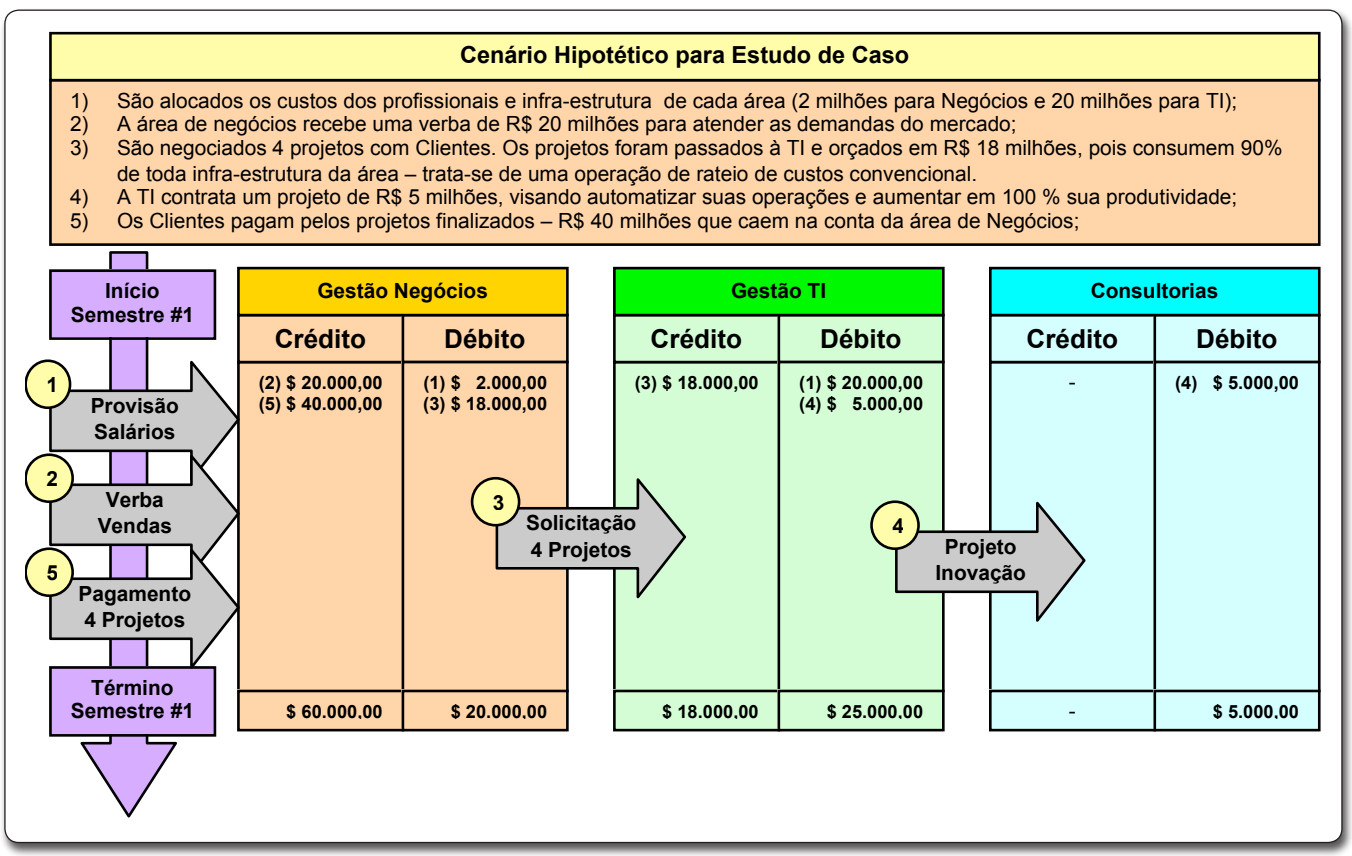


Figura 5. Cenário Inicial para análise do comportamento da Alocação dos Custos de TI

de uma empresa que oferece serviços ao mercado e necessita da área de TI para atender seus Clientes.

Na **Figura 5**, descrevemos um cenário que permitirá estabelecer uma base inicial financeira e compará-la com os demais modelos existentes, possibilitando analisar como um inadequado modelo de alocação de custos pode prejudicar a análise financeira das áreas da organização.

Com este cenário estabelecido, vamos simular um 2º. semestre empregando as duas diferentes abordagens de alocação de custos, porém respeitando as seguintes premissas:

- Custos Operacionais das áreas permanecem inalterados;
- Verba de Vendas permanece inalterada;
- A Produtividade da TI aumenta em 100% (projeto de inovação);
- Projetos Contratados saltam de 4 para 8 no novo semestre;
- Projetos são equivalentes na complexidade dos anteriormente contratados.

Dinâmica de Alocação de Custos no Modelo Tradicional

No modelo tradicional, a TI é encarada como um "**custo-fixo operacional**" e serve apenas como critério de rateio para as unidades de negócio existentes e auxiliar na composição dos custos dos serviços oferecidos ao mercado.

Apesar do conceito deste modelo ser atrativamente simples, ele gera uma grave inconsistência dos custos operacionais, dificultando identificar quais áreas estão sendo mais produtivas – a TI ou a área de Negócios.

O problema está no fato de que o custo do serviço está sendo contabilizado de acordo com o nível de esforço e não por sua produção. Como a produtividade da área de TI aumentou em 100%, executar o dobro dos projetos continua consumindo os mesmos 90% da estrutura interna da área, mantendo-se em R$ 18 milhões o custo interno da TI.

Isto significa que apesar de sua luta contínua para aperfeiçoar a qualidade e produtividade de seus projetos, a área de TI sempre será contabilizada de forma negativa (**custo-fixo**), pois deverá ser apenas uma fonte de rateio para a área de negócios.

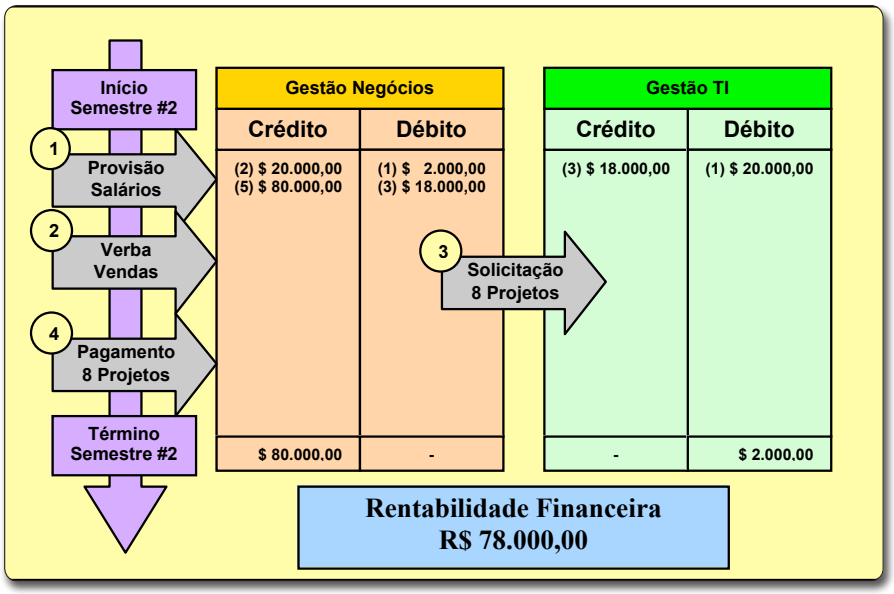


Figura 6. Alocação de Custos TI no modelo Tradicional

Na **Figura 6**, é fácil perceber que os ganhos de produtividade da TI não foram contabilizados em seu centro de custo, mas sim repassados diretamente ao centro de custos da área de negócios, **distorcendo a rentabilidade** de um produto ou serviço oferecido pela organização, o que pode acarretar prejuízos significativos em relação à análise e formação de preços no mercado.

Alocação de Custos no Modelo Proposto

No modelo proposto, a TI deixa de ser encarada como um "**custo-fixo operacional**" e passa a desempenhar um papel ativo dentro da estrutura financeira organizacional. Seu centro de custos passa a absorver todos os ganhos de produtividade, evidenciando os benefícios que investimentos em TI podem gerar no desempenho da organização.

Para entender a precificação dos serviços da TI foi estabelecido, devemos lembrar dos 4 projetos repassados pela TI no 1º semestre, que custaram para a área de negócios R$ 18 milhões.

Isto equivale a dizer que qualquer projeto TI, com nível de complexidade semelhante, custaria para a área de negócios R$ 4,5 milhões. Como foram contratados 8 projetos simultaneamente, o custo foi proporcional ao serviço solicitado – R$ 36 milhões.

Desta forma, os serviços de TI continuam sendo cobrados de forma fixa pelos seus serviços executados, independentemente do aumento dos custos internos ou das variações de produtividade da área.

O exemplo da **Figura 7** demonstra como é possível gerenciar a relação Cliente-Fornecedor empregando este modelo, transformando as áreas da empresa em **centros de custos independentes**. A alocação das receitas e das despesas operacionais permite aos executivos acompanharem mais facilmente a dinâmica de troca de serviços, possibilitando identificar áreas com maior volume de solicitações e melhor rentabilidade.

Os gerentes das áreas passam a administrar seus centros de custos como se fossem executivos de empresas, possibilitando negociar adiantamentos, prêmios de produtividade, punições em atrasos ou quebras de contrato, reembolso por falhas na prestação de serviço e diversos outros mecanismos contratuais, possíveis apenas quando operamos entre empresas.

Integrando o Modelo com o Planejamento Estratégico

Todas as empresas deveriam organizar periodicamente o **Planejamento Estratégico Corporativo** visando definir quais serão as metas corporativas (financeiras

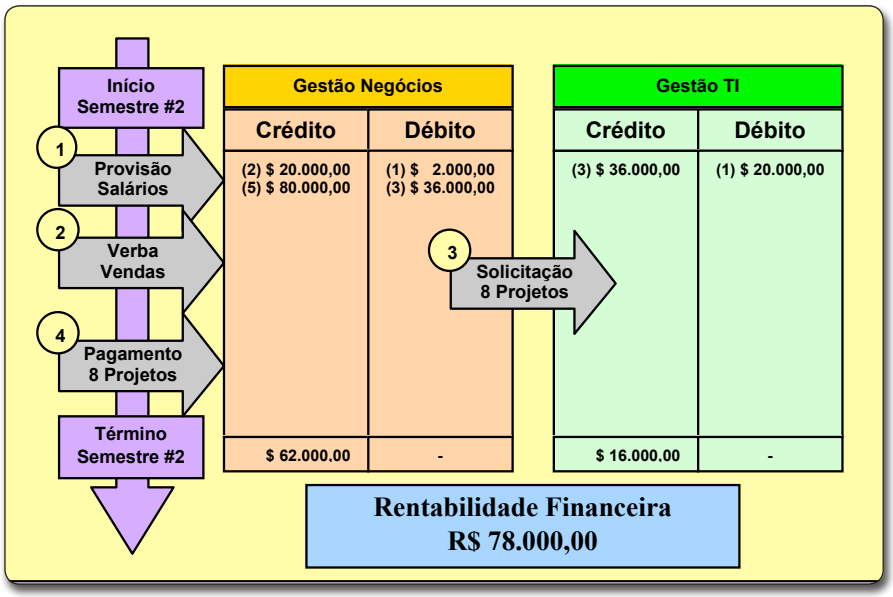


Figura 7. Alocação de Custos TI no modelo Proposto

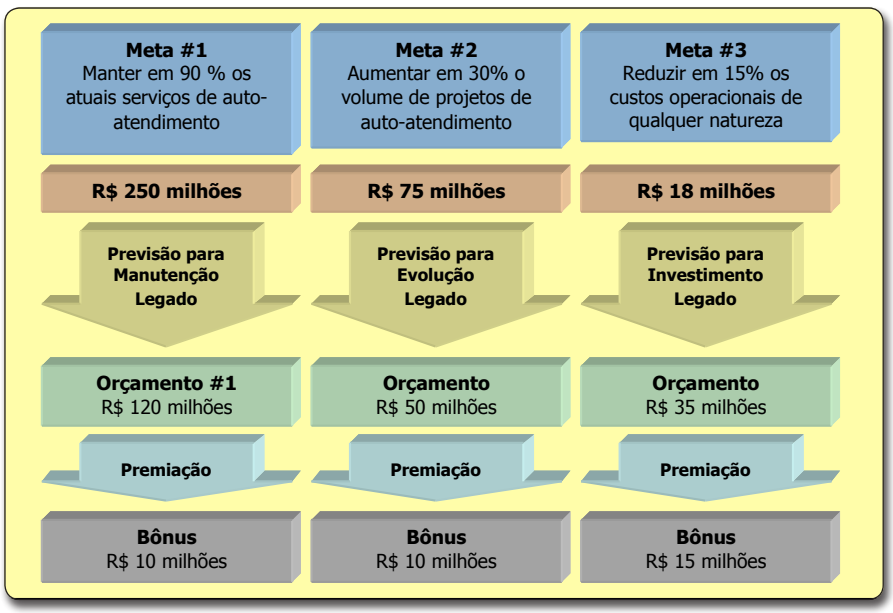


Figura 8. Exemplo de Planejamento Estratégico baseado em Metas e Orçamentos pré-definidos

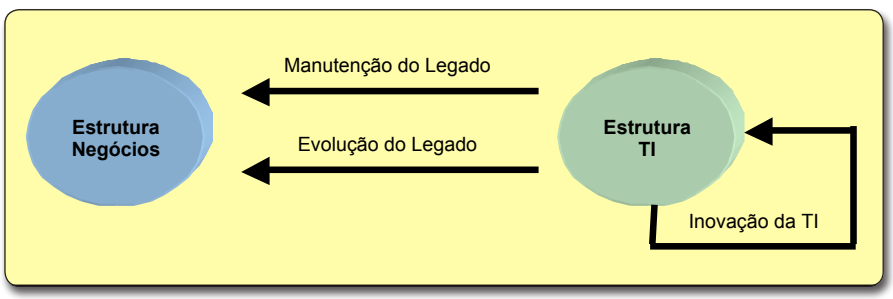


Figura 9. Natureza de Serviços de TI

e operacionais) a serem alcançadas pela organização num determinado prazo estabelecido.

Na **Figura 8** podemos ver como o **Planejamento Estratégico** deixa claro para todos da organização quais são as metas planejadas e qual será o montante financeiro que a empresa está disposta a investir para alcançar este objetivo.

Uma vez estabelecido o **Planejamento Estratégico**, este deverá ser desmembrado em planos mais detalhados (**Planejamento Tático e Operacional**) até que consigamos visualizar todos os projetos e distribuí-los para **todas as áreas operacionais**, visando criar um alto grau de **colaboração** de todos para o cumprimento das metas. A previsão de uma premiação reforça a idéia de que **todos ganham** atingindo os objetivos da empresa.

Com o **planejamento financeiro** estabelecido, sabemos quais são as verbas disponibilizadas para cada área de negócio existente, podendo estabelecer o **limite de gastos operacionais** (internos ou externos) para atingir determinada meta organizacional. Cada vez que um serviço é solicitado, o valor do serviço é debitado da área-cliente e creditado na área-fornecedora.

Neste cenário, a área de negócios deverá avaliar bem, antes de solicitar qualquer serviço para a TI, pois sua premiação não dependerá apenas da meta corporativa alcançada (modelo tradicional), mas no desempenho financeiro de sua área (custos realizados para alcançar a meta).

Na verdade, todas as áreas passam a ser avaliadas pelo fluxo financeiro de entradas e saídas gerados pelas trocas de serviços. Será como trazer a **dinâmica das empresas para dentro da organização**.

Identificando a Natureza dos Serviços de TI

Por maiores que sejam os investimentos corporativos em processos, metodologias, profissionais, treinamento, ferramentas e consultorias, o **perfil de relacionamento** entre as áreas torna-se o **principal freio** na produtividade e inovação empresarial, gerando enorme

impacto nos resultados corporativos.

Todas as organizações que desejam alcançar maior agilidade e controle operacional deveriam escolher modelos de relacionamento mais equilibrados e voltados à inovação, de forma a garantir o melhor desempenho e qualidade nos serviços disponibilizados. Com isto, torna-se necessário **institucionalizar a relação Cliente-Fornecedor** entre as áreas de Negócios e TI, de forma a gerenciarmos a área de TI como se fosse uma empresa independente.

Com a possibilidade de **cobrar pelos serviços prestados** às áreas de negócio, a TI tem a oportunidade de atuar como um **centro de lucros**, possibilitando até gerar um **superávit** e financiar seus próprios projetos de inovação, sem depender da disponibilidade financeira da organização.

Como podemos observar na **Figura 9**, existem três tipos diferenciados de serviços que deverão ser contabilizados e controlados de forma independente dentro da TI.

Manutenção do Legado

Trata-se do somatório de todos os custos de TI voltados única e exclusivamente para **manter a estrutura tecnológica em funcionamento**, incluindo todas as atividades relacionadas à monitoração dos ambientes de produção, manutenção corretiva e preventiva dos sistemas de comunicação e infra-estrutura, apoio tecnológico a clientes e fornecedores que compartilham o sistema de informações da empresa, homologação de novos produtos e serviços de clientes que desejam ser incorporados ao atual sistema legado.

A principal característica do custo de manutenção do legado é ser **acumulativo**. À medida que a empresa cresce, toda a estrutura legada suportada pela TI aumenta com os anos. Muitas empresas ainda são oneradas por terem que manter estruturas antiquadas de processamento (*mainframe*) e tecnologias ultrapassadas (*Unix, OS-2, DOS*), ampliando mais os custos do legado.

A melhor forma de lembrar a área de Negócios da existência deste **custo recorrente** é estabelecer alguns projetos de vigência anual (Manutenção de Equipamentos, Monitoração de Redes, Suporte Técnico e Problemas de Produção). Seu objetivo será englobar custos permanentes e cobrar anualmente por estes serviços prestados, sendo o valor a ser pago **ajustado diretamente pelas oscilações de indicadores** como nível de disponibilidade dos serviços e volume de transações.

Para os **custos eventuais** (Homologação de Produtos e Novos Clientes), os projetos devem ser negociados conforme a natureza e complexidade do serviço, sempre cobrando da área de negócio o serviço prestado.

Evolução do Legado

Trata-se de todos os custos de TI voltados única e exclusivamente para **modificar ou ampliar as funcionalidades dos sistemas legados**, incluindo todo e qualquer tipo de modificação de layout, inserção de novos campos e consistências de negócios, novas telas e transações comerciais, ajustes tributários ou financeiros provenientes de adequações legais, correção de "*bugs*", confecção de relatórios, automação de processos, entre outras solicitações.

Normalmente, a área de Negócios preocupa-se apenas com estes tipos de projetos, pois entendem que estes irão agregar melhorias e inovações aos produtos e serviços oferecidos pela organização, gerando mais oportunidades de vendas e comercialização dos mesmos. Inconscientemente, a área de Negócios acredita que toda estrutura de TI está focada exclusivamente para atender este tipo de demanda.

A área de Negócios negligencia os projetos de **Manutenção do Legado**, pois acredita que seja uma obrigação da TI manter esta estrutura em perfeito funcionamento, mesmo que o sistema legado exista exclusivamente para viabilizar as transações comerciais que beneficiam diretamente a área de negócios.

Inovação da TI

Trata-se de todos os custos de TI voltados única e exclusivamente para **renovar as técnicas, processos e tecnologias** a serem empregadas nos futuros projetos, incluindo todas as modificações tecnológicas, reestruturação de sistemas, incorporação de novas técnicas, inserção de novos procedimentos de controle operacional, adequações metodológicas, auditorias internas, treinamento e capacitação profissional, entre outras demandas.

No modelo tradicional, os projetos de inovação da TI são negociados e priorizados em conjunto com as demais demandas da organização, concorrendo com projetos das áreas de negócio, que são claramente mais importantes que as necessidades da TI, sobrando pouco para discutir sobre o aperfeiçoamento da TI.

No modelo proposto, o **Plano de Inovação da TI** será financiado tanto pela organização, que prevê um volume de investimentos para inovações, quanto pelo superávit operacional da TI, possibilitando que a mesma fique comprometida com o aumento da produtividade e melhorias, incorporando apenas os custos realmente necessários.

Com o superávit operacional, a TI poderá promover e elaborar programas de incentivos financeiros totalmente apartados das demais áreas corporativas, criando metas e políticas de incentivo próprias à sua realidade, reforçando o nível de autonomia e responsabilidade financeira dado a profissionais e executivos da TI.

Impacto da Gestão de Serviços nas Áreas Operacionais

A **Gestão de Serviços Operacionais** é uma modalidade de gestão baseada no **desempenho financeiro** das áreas operacionais, possibilitando equilibrar automaticamente os recursos empresariais de acordo com a alocação dinâmica financeira gerada pelas trocas de serviços.

Este conceito é uma adaptação do modelo de **custeio por atividades (Custo ABC)** aplicada nas industriais e uma evolução do modelo de Gestão de Centros de Custos, para o de Gestão de Centros de Lucros. Neste modelo, evitamos a aplicação do rateio e transformarmos as áreas em centros financeiros independentes, podendo gerar lucro ou prejuízo.

A **Gestão de Serviços Operacionais** empregam **métricas** para estabelecer a **tabela de preços** referentes aos serviços prestados (transações dia, pontos de função, casos de testes, scripts automatizados, eventos de negócio, ou outros indicadores que possam dimensionar o volume do serviço a ser executado).

Porém, a avaliação das áreas operacionais ocorre apenas com um **único indicador financeiro**, a **lucratividade local**, gerada pela diferença entre as receitas e despesas produzidas. Desta forma, existe uma total transparência do mecanismo de avaliação do desempenho das áreas, sendo facilmente entendido e acompanhado pelos profissionais, requisito fundamental para garantir o engajamento de todos na busca da eficiência corporativa. A transparência do modelo revela as áreas que estão sendo mais focadas no aumento da produtividade e redução de custos operacionais.

Para exemplificar, quando a TI melhora sua produtividade, ela tem maiores condições de atender mais solicitações sem aumentar seus custos internos, o que cria um superávit financeiro no seu Centro de Custo, o que seria um bom sinal de desempenho. Da mesma forma, a área de TI poderia ter optado pela decisão mais simples, de aumentar a capacidade de produção através da contração de terceiros ou aumento do quadro de funcionários, gerando um aumento de seus custos internos.

Portanto, este modelo revela as conseqüências financeiras das decisões das lideranças, quando focam exclusivamente no aumento da produção (curto-prazo), quando deveriam focar no aumento da produtividade (longo-prazo).

Não adianta melhorarmos os indicadores (atender mais solicitações) sem que consigamos melhorar nosso desempenho financeiro. Isto evita que sejam feitos investimentos que não ataquem as **restrições sistêmicas corporativas**, ou seja, não afeta diretamente a "corrente crítica" da organização.

O indicador financeiro é tão eficiente como mecanismo de avaliação que disciplina tanto as áreas que consomem quanto aquelas que fornecem os serviços.

Disciplina quem consome os serviços

No modelo tradicional, todas as **requisições de serviços eram gratuitas**, o que não exige nenhuma contrapartida financeira ao centro de custo solicitante, não exigindo dos gerentes de áreas uma preocupação com o melhor aproveitamento dos serviços prestados.

Neste modelo, cada área solicitante passa a avaliar mais criteriosamente o que pede, estabelecendo sempre uma relação custo-benefício e administrando o saldo de investimentos que sua área ainda possui, alinhando suas requisições às metas corporativas.

Este modelo inibe as solicitações desnecessárias, evitando que tempos e recursos sejam alocados para realizar projetos que geram pouco ou nenhum valor agregado, afetando diretamente a rentabilidade corporativa. Afinal, **não existe esforço sem custo financeiro**.

Disciplina quem executa os serviços

Ao contrário do modelo tradicional, todos os gerentes administram suas áreas como se fossem empresas, tendo um interesse real em tornar seu centro de custo rentável, trabalhando na forma mais rápida, eficiente e econômica possível, sem desperdícios.

Assim, cada área buscará novas formas de ampliar suas **receitas** com o objetivo de bancar sua infra-estrutura, cursos de aperfeiçoamento, salários, bonificações, aquisições e licenciamento de ferramentas e outras despesas que podem gerar impacto direto na produtividade da área.

Neste modelo, os próprios profissionais buscam manter uma estrutura compatível e balanceada com a realidade das demandas de serviços. Ter muitos profissionais provoca aumento dos custos operacionais e inviabiliza aumentos salariais e bonificações. Ter poucos profissionais impede o total atendimento dos serviços solicitados e restringem as receitas operacionais.

Conclusão

Para suportar o maior nível de exigência e profissionalismo, a área de TI deve deixar de ser encarada como uma simples área de apoio para tornar-se efetivamente uma área estratégica dentro das organizações.

Para isto, a área de TI deverá adotar padrões de governança mais adequados à sua dinâmica operacional, livrar-se do tradicional orçamento fixo e remunerar-se de acordo com o volume de solicitações requeridas pelas áreas de negócio, deixando de lado a relação patrão-empregado e apostar na relação cliente-fornecedor.

A solução aqui proposta é transformar a área de TI como uma "empresa independente" e adotar o conceito de "Fábrica de Serviços", cobrando por todo e qualquer acionamento e atendendo da forma mais rápida e controlada possível. Desta maneira, mesmo sendo uma área interna da organização, ela terá o mesmo comportamento de uma estrutura terceirizada, remunerada pelo volume de solicitações demandadas.

Desta forma, a área de TI passa a ter não apenas condições de adaptar-se com eventuais oscilações de demanda de serviços, mas garantir as inovações internas necessárias para suportar os futuros projetos da organização, investindo no aperfeiçoamento e controle de seus processos de manutenção e evolução dos sistemas, trazendo maior vantagem competitiva a toda organização. ●