Nesta seção você encontra artigos voltados para as práticas e métodos ágeis.

Scrum e MPS.BR: Um caso prático do uso em conjunto destas abordagens

O caso da AltoQi

André Luiz Banki
banki@altoqi.com.br
Engenheiro Civil pela Universidade Federal de Santa Catarina (UFSC), Mestre em Engenharia Civil (na área de Estruturas), também pela UFSC, e Especialista em Engenharia de Software pelo SENAI/Florianópolis. Trabalha como Gerente de Desenvolvimento na empresa AltoQi Tecnologia Aplicada à Engenharia, com mais de 15 anos de experiência no gerenciamento de projetos de desenvolvimento de software, incluindo atividades de pesquisa técnica, levantamento de requisitos e coordenação de equipes. Responsável por todo o processo criativo no desenvolvimento de complexas ferramentas de software para Engenharia em ambiente CAD, adotadas por mais de 15.000 profissionais em todo o território nacional, bem como pela concepção e implantação de um processo de desenvolvimento baseado em metodologias ágeis e pela adequação desse processo ao nível G do modelo MPS.BR. Possui certificação PMP, RUP e Scrum.

De que se trata o artigo?

Relato de experiência, englobando quatro anos de investimento de uma empresa nacional de desenvolvimento de software na implantação de técnicas de Engenharia de Software, com destaque para a automação de testes, e de uma metodologia de desenvolvimento iterativo baseada no Scrum e avaliada pelo modelo MPS.BR. Está focado na aplicação prática desses conceitos em um modelo de desenvolvimento colaborativo e apresenta os resultados alcançados pela empresa com esse projeto.

Em que situação o tema é útil?

A escolha da metodologia mais adequada para o desenvolvimento de software em uma organização, levando em consideração os inúmeros fatores envolvidos, não é uma tarefa trivial. A publicação de estudos de caso, como este, procura apontar algumas dificuldades práticas envolvidas com a implantação de processos e auxiliar as empresas a traçar o seu caminho rumo à melhoria contínua.

Resumo:

A AltoQi é uma empresa no mercado brasileiro de software para Engenharia, com mais de 20 anos de experiência tanto no desenvolvimento como na comercialização de seus produtos. Ainda assim, a partir de um dado momento, o custo de manutenção desses sistemas tornou-se elevado, comprometendo a sua capacidade de desenvolvimento. Neste contexto, neste artigo será abordado o investimento que a empresa efetuou para controlar o desenvolvimento dos seus sistemas legados, iniciando pelo controle de solicitações, controle de versões e automação de testes. Será discutida a necessidade de investimento em boas práticas de Engenharia de Software como subsídio a uma efetiva implantação de metodologias ágeis de desenvolvimento.

A partir desse investimento em infra-estrutura, a aplicação de uma metodologia de desenvolvimento iterativo, baseada no Scrum, permitiu à empresa retomar o seu ritmo de crescimento e estender esse conceito a uma nova forma de interação com seus usuários. Com seus mais de 20.000 clientes, a empresa precisava de uma forma de levantar as reais necessidades dos seus clientes para criar uma nova versão de seu principal produto.

O conceito do desenvolvimento iterativo foi aplicado na prática, em um projeto de oito Releases, no qual cada um dos recursos foi discutido com os clientes em um blog de desenvolvimento. Com esse feedback, os requisitos desse produto puderam ser definidos com mais clareza e refinados a cada release liberado. O resultado foi positivo para os clientes, para a equipe de desenvolvimento e, especialmente, para a empresa.

A AltoQi (www.altoqi.com.br) atua desde 1989 no desenvolvimento e comercialização de software para projetos de Engenharia, possuindo mais de 23 mil clientes no Brasil e no exterior. O mercado da construção civil engloba mais de 12 especialidades diferentes, sendo que, destas, as principais são atendidas pelos produtos da empresa: AltoQi Eberick (para projeto de estruturas em concreto armado), AltoQi Hydros (para projeto de instalações prediais hidráulicas, sanitárias, de gás e de prevenção e combate a incêndio), AltoQi Lumine (para projeto de instalações elétricas prediais e de infra-estrutura para cabeamento) e QiCAD (plataforma independente de CAD, voltada à complementação dos desenhos gerados).

Além dos produtos de software e dos serviços de suporte técnico e especializado aos seus usuários, a AltoQi também atua, através do seu canal QiSat (Sistema Integrado de Ensino a Distância, Presencial e de Serviços Dirigidos ao Mercado da Construção Civil, www.qisat.com.br), para prover os profissionais ligados à área de construção civil de um mecanismo de acesso constante e atualizado ao conhecimento, capacitação, outras oportunidades e serviços.

A AltoQi está presente em todos os Estados da Federação, totalizando uma base de relacionamento com mais de 400 mil profissionais, através de atendimento comercial, convênios com mais de 100 universidades e com os principais CREAs do país, e dos mais de 6.000 alunos atendidos no canal de ensino do QiSat.

Forma de atuação

A organização AltoQi tem como objetivo principal de negócio o desenvolvimento e comercialização de produtos de software para o mercado da Construção Civil. Esses produtos de software organizam-se em Linhas de desenvolvimento, representadas por um Produto principal e seus diversos Módulos opcionais.

Os produtos desenvolvidos pela AltoQi são pacotes fechados, cuja customização pelo cliente é limitada à escolha dos módulos a serem adquiridos. Por isso, não há um "cliente" definido em nenhum projeto de desenvolvimento. No seu lugar, cabe ao Departamento de Produtos e Serviços (DPS) representar o cliente, exercendo o papel de *Product Owner* em relação ao Departamento de Desenvolvimento. O DPS é o responsável pelas atividades de análise dos produtos da empresa e do seu posicionamento no mercado, avaliação de novos produtos e novos recursos para os produtos atuais da empresa, suporte técnico aos clientes externos e treinamento técnico para clientes externos e internos. O Departamento de Desenvolvimento é responsável por planejar, executar e controlar as atividades que terão como resultado o produto de software selecionado pelo DPS. Cabe ao Departamento de Desenvolvimento a criação de novos produtos ou versões, as quais possam vir a ser comercializadas pela empresa, junto à sua extensa base de clientes.

A relação entre os dois departamentos e os clientes pode ser observada na **Figura 1** a qual ilustra:

- Clientes interagindo com a equipe de Suporte Técnico, para sanar suas dúvidas. Eventuais problemas e sugestões são levantados para futura análise.
- Interação do Suporte com a equipe de Produto, responsável por analisar os problemas e sugestões reportados e convertê-los em requisitos para os próximos projetos.
- Interface entre a equipe de Produto, após priorizar as necessidades para o programa, com o departamento de Desenvolvimento, para planejar e executar os projetos relacionados a esses recursos.

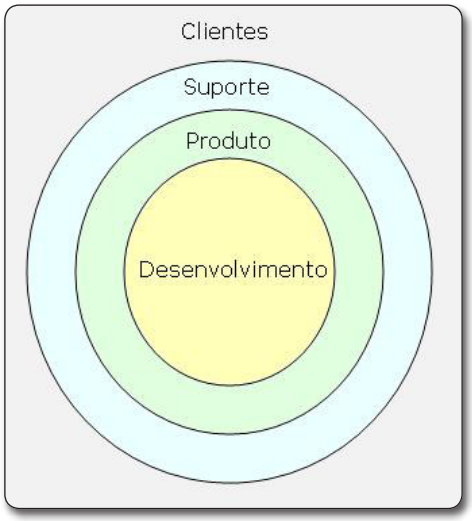

Figura 1. Relacionamento entre os departamentos e o cliente

Nesse cenário, um fator que diferencia a AltoQi de outras empresas de software é a forte necessidade do desenvolvimento interno dos requisitos. As necessidades dos clientes referem-se a problemas no campo de conhecimento da Engenharia Civil e, especialmente pela natureza gráfica e tridimensional das aplicações envolvidas, não são descritas adequadamente nem pelos próprios clientes. A empresa precisa ter, em sua equipe, corpo técnico adequado para entender as necessidades dos clientes, convertê-las em requisitos de software, detalhá-las em grau adequado para permitir a sua implementação e conferir cada um dos resultados apresentados.

Implantação de processos

Atualmente, o Departamento de Desenvolvimento da AltoQi possui o nível G do MPS.BR, o Modelo de Melhoria de Processos de Software Brasileiro, mas nem sempre este foi o cenário. Durante muitos anos, os quais englobaram o desenvolvimento dos produtos da empresa que são mantidos até hoje, o processo poderia ser descrito apenas como "programar e corrigir". Existia uma separação entre os Departamentos de Desenvolvimento e de Suporte Técnico, isolando a equipe de desenvolvimento de um contato direto com a base de clientes, mas não havia um trabalho sistematizado de Produto nem a aplicação de técnicas adequadas da Engenharia de Software. Cabe explicar que os produtos da empresa sofreram manutenção evolutiva desde sua criação, com o lançamento de Versões comerciais,

a intervalos de um a dois anos, e manutenção corretiva entre cada uma dessas versões, com a liberação de Revisões sem custo para os clientes. A mesma equipe de desenvolvimento sempre esteve responsável tanto pelo desenvolvimento das novas versões como pela manutenção das versões existentes.

Esse trabalho de evolução do programa era conduzido apenas com o apoio de testes exploratórios manuais, executados pela equipe de Suporte Técnico. Os resultados obtidos foram satisfatórios até 2006, quando o lançamento de uma nova versão do AltoQi Eberick, contendo mudanças muito significativas em relação à sua versão anterior, provou que essa sistemática era insuficiente para lidar com uma alteração daquela magnitude. A versão lançada apresentou um elevado índice de defeitos e, pelos dois anos seguintes, a maior parte do esforço de desenvolvimento teve que ser direcionada para a correção de problemas. Após várias Revisões, o produto foi estabilizado, mas esse custo de manutenção causou um atraso significativo no ciclo de desenvolvimento da empresa.

Para entender esse problema, é importante avaliar que, em 2007, a equipe de desenvolvimento lidava com um sistema que possuía uma elevada complexidade técnica (envolvendo computação gráfica, análise estrutural, prescrições normativas de Engenharia e um grande leque de conceitos técnicos inacessíveis aos programadores), sobre uma base de código legado desenvolvida continuamente desde 1994 e contendo um elevado grau de acoplamento entre todas as suas partes, sem o apoio de técnicas adequadas para lidar com essa situação.

A reversão dessa situação pôde ser obtida apenas por um investimento da empresa na melhoria do seu processo de desenvolvimento, começando pelos testes, problema primário da empresa na época. Existem duas estratégias principais para teste de software: os testes de "caixa preta" e os testes de "caixa branca". Os testes de caixa branca, ou testes estruturais, são utilizados para avaliar o comportamento interno do software, usualmente para verificar funcionalidades individuais, através de testes unitários. Essa abordagem foi descartada, pelo grande volume de código legado e reduzido prazo disponível para a obtenção de resultados.

No seu lugar, a empresa investiu nos testes de caixa preta, ou testes funcionais, utilizados para avaliar o comportamento externo do sistema, ignorando detalhes específicos de sua implementação e procurando testar o sistema como um todo, ao representar a interação do usuário com o software. Para isso, implantou um sistema próprio de automação de testes e instituiu, dentro do Desenvolvimento, o papel específico de Engenheiro de Testes, responsável por planejar e construir a base de testes que viria a garantir uma regressão completa do sistema a cada modificação efetuada. Esse trabalho foi transferido da equipe de Suporte para uma nova equipe dedicada apenas a essa tarefa. Hoje, cerca de metade da equipe de desenvolvimento é composta por profissionais da área da Computação e a outra metade por profissionais da área de Engenharia.

Em paralelo a isso, a equipe implantou um sistema e política de controle de versões (antes feito apenas informalmente) e um sistema para controle das solicitações (voltado, inicialmente, para gerenciamento dos problemas reportados pelos clientes e das correções efetuadas). Quando foi possível combinar as três técnicas (controle de versões, controle de mudanças e testes automatizados), a equipe finalmente pôde estabelecer um conceito fundamental para apoiar o gerenciamento do projeto: a rastreabilidade. Segundo o MPS.BR, a rastreabilidade deve ser estabelecida desde um requisito fonte, passando pelos seus requisitos de baixo nível, até o nível de decomposição mais baixo do produto (código fonte) e vice-versa. Isso permite associar qualquer modificação efetuada no programa aos requisitos que a motivaram e, com isso, avaliar com muito mais rapidez qualquer defeito que tenha sido inserido por essa modificação (ver **Figura 2**).

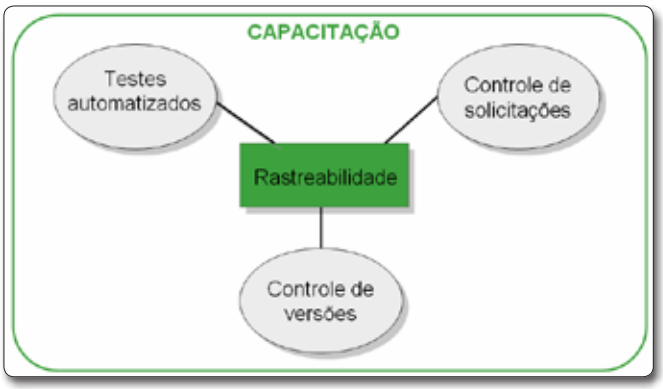

Figura 2. Técnicas da Engenharia de Software

Antes disso, dada a grande complexidade técnica do sistema, a equipe de desenvolvimento lidava com cada problema reportado pelos clientes apenas investigando-o diretamente, o que podia gerar impactos em outras partes do programa que não eram detectados rapidamente, pela dificuldade do teste. Com a automação dos testes, essa propagação de defeitos pôde ser limitada e o custo de cada correção foi reduzido drasticamente, uma vez que o defeito podia ser detectado rapidamente e rastreado até a sua origem.

Esse ciclo de melhoria levou dois anos para ser implementado. Com ele, a empresa pôde retomar o controle sobre o seu processo de desenvolvimento, recuperar o seu principal produto e passar da manutenção para a melhoria no processo de desenvolvimento de novos produtos. Para que isso pudesse ocorrer, é importante destacar também a necessidade de capacitação, exercida principalmente pela especialização do Gerente de Desenvolvimento, responsável pela definição de todo o processo e capacitação, por sua vez, de toda a equipe. Retirar o foco do dia-a-dia das atividades e estudar as melhores práticas do mercado é fundamental para entender como aplicá-las corretamente no seu ambiente de desenvolvimento.

A migração para um modelo de desenvolvimento iterativo, assunto do tópico a seguir, foi possível apenas tendo por base a correta aplicação de boas práticas da Engenharia de Software.

Modelo iterativo

Pode-se definir um Projeto como "um esforço temporário realizado com o objetivo de produzir um produto ou serviço único". Temporário significa que todo projeto tem início e fim explicitamente definidos e único indica o fato de que cada projeto possui características próprias e seu produto resultante é distinto em um ou mais aspectos de outros projetos existentes. Como características comuns para qualquer Projeto, são realizados por pessoas, possuem alguma restrição de recursos, são planejados, executados e controlados.

Embora as empresas de desenvolvimento de software, de forma geral, trabalhem na criação de Produtos, nem todas organizam efetivamente isso na forma de Projetos. O estabelecimento desse conceito, tendo por base a definição sistematizada do Escopo de um projeto antes do início do mesmo, bem como o controle sobre as mudanças nesse escopo ao longo do projeto, foi o primeiro passo na implantação de um modelo de desenvolvimento na AltoQi. O Departamento de Produtos e Serviços (DPS) foi criado nessa etapa, para separar as atividades de definição do escopo do projeto (necessidades dos clientes) das atividades de execução desse escopo (desenvolvimento de software).

No período 2009 / 2010, correspondente à implantação do processo de desenvolvimento descrito neste artigo, culminando com a avaliação nível G do MPS.BR em Dezembro / 2010, foram concluídos 20 projetos, em um total superior a 79000 horas de desenvolvimento. Esses projetos validaram o processo proposto e fornecem importante base histórica para o planejamento de projetos futuros.

O modelo de ciclo de vida adotado pela AltoQi, de natureza iterativa e incremental, se baseia no aumento e refinamento sucessivo de um sistema através de múltiplos ciclos de análise, projeto, implementação e teste, conforme colocado na **Figura 3**. Está fundamentado no Scrum, uma metodologia que tem alcançado reconhecimento como uma ferramenta eficaz para desenvolvimento produtivo de software. Segundo Sutherland, mesmo quando padrões de Scrum são combinados com outros padrões organizacionais existentes, eles são altamente adaptáveis, ainda que em organizações de desenvolvimento de software bem estruturadas.

O Scrum utiliza um conjunto de artefatos que suportam o processo de desenvolvimento e controlam o seu progresso. Para iniciar um projeto Scrum bastam uma Visão do produto, descrevendo a motivação do projeto e o seu resultado final esperado, e um *Project Backlog* (Lista de Recursos), documento que contém a lista completa dos requisitos do projeto, necessários para atender a Visão do produto. Na AltoQi, também é usado o termo *Release Backlog*, visto que cada Release de um Produto é gerenciado como um Projeto independente.

O processo apresenta três fases distintas: Preparação, Desenvolvimento e Fechamento. A fase de Preparação, também chamada "iteração zero", tem o objetivo de definir o Plano do Projeto, incluindo seu Escopo (recursos a serem desenvolvidos), Prazo e Custo, encerrando-se quando o sistema a ser desenvolvido está definido, sua viabilidade está confirmada e as ferramentas necessárias estão à disposição da equipe.

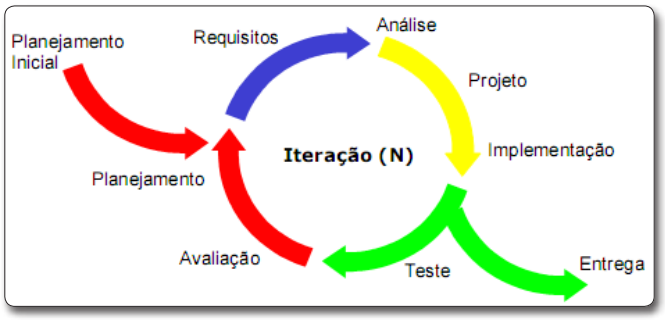

Figura 3. Diagrama simplificado do modelo iterativo

As estimativas de software fornecem subsídios para a definição de aspectos fundamentais dessa fase. Uma estimativa aproximada de tamanho das atividades que fazem parte do escopo permite que o Gerente de Projetos defina qual o cronograma do projeto, o esforço necessário, seu custo e os recursos que serão necessários para desenvolvê-lo. Esse assunto será tratado com mais detalhe no próximo tópico.

A fase de Desenvolvimento, dividida em iterações, contém a execução das atividades técnicas necessárias ao cumprimento do escopo do projeto. Encerra-se quando todos os itens presentes no *Release Backlog* tenham sido implementados. Cada iteração é definida como uma "*time box*", com duração de duas semanas, na qual são executados parte dos itens definidos no *Release Backlog* (formando o "*Iteration Backlog*"). O resultado do item do trabalho a ser desenvolvido é um incremento do produto. A **Figura 4** ilustra o ciclo de desenvolvimento, no qual os requisitos são apresentados no *Iteration Backlog* e desenvolvidos na iteração, gerando um novo incremento que será entregue para a fase final do processo.

Ao final de cada iteração, deve-se ter um sistema completo, testado e documentado. Para isso, é fundamental o apoio de um programa de testes automatizados. A abordagem iterativa permite considerar a alteração nos requisitos, uma vez que eles normalmente serão alterados durante o processo. A abordagem tradicional, na qual todos os requisitos são determinados, de uma vez, no início do projeto, é válida apenas para projetos bastante triviais (KRUCHTEN, 2000). O ponto chave está na produção de versões do sistema final, de forma que, em cada uma, esteja presente um conjunto crescente de funcionalidades, já completamente integrada e testada. É importante frisar que as alterações nos requisitos devem ser controladas, mantendo-se o rumo do projeto.

A fase de Fechamento se inicia após a conclusão de todas as atividades técnicas do projeto (ou seja, após a finalização do escopo do projeto). Contém uma avaliação organizacional do resultado do projeto, com a coleta de lições aprendidas e informações históricas que serão úteis para o desenvolvimento dos próximos projetos (ver **Figura 5**).

Estimativas de software

Dentre os aspectos que foram importantes para a definição de um processo de desenvolvimento na AltoQi, o mais complexo deles foi a estimativa dos produtos de software.

Estimar e planejar são atividades críticas para o sucesso de qualquer projeto de desenvolvimento de software. Permitem guiar os investimentos no projeto, definir que equipe precisará ser alocada ao projeto em cada período e saber se o projeto está no caminho correto para entregar a funcionalidade desejada ao seu final.

Dentro do cenário específico da AltoQi, é importante relembrar que, em 2009, o Departamento de Desenvolvimento estava saindo de um ciclo vicioso de manutenção do seu principal produto, durante o qual a capacidade da empresa de entregar novos produtos, no prazo definido e com a qualidade necessária, estava sendo questionado. No aspecto da qualidade, a implantação das práticas já discutidas, com destaque para os testes automatizados, foi fundamental. No aspecto do cumprimento dos cronogramas, era preciso estabelecer uma forma adequada de estimar os projetos a serem desenvolvidos.

Vasquez (2009) destaca algumas dificuldades no ato de estimar:

• Ambiguidade, volatidade, falta de clareza: não é possível realizar uma estimativa adequada sobre um requisito que não esteja corretamente definido. Falta de clareza ou ambigüidade nos requisitos resultam em estimativas mal formuladas.

• Falta de medições adequadas: sem um acompanhamento do projeto, com a comparação entre os valores estimados e os valores realizados, não há retorno para a equipe quanto à qualidade e utilidade das suas estimativas.

• Falta de referências válidas: em termos de software, é usual tratar-se as estimativas em uma base comparativa, um critério que pode levar ao erro se forem adotadas referências inadequadas em relação ao projeto corrente.

• Visão distorcida do que seja estimar: é comum que se tenha dificuldade em diferenciar a estimativa dos conceitos de meta e compromisso. Uma estimativa é uma avaliação aproximada de algo (fornecer uma estimativa, para software, é o ato técnico de atribuir um tamanho a uma determinada grandeza). Com base nela, pode-se estabelecer uma meta, um objetivo relacionado a tempo e valor (estipular uma meta é um ato gerencial ou político) e assumir um compromisso, ou seja, o acordo final e responsabilidade, adquirida de forma verbal ou escrita, sobre o resultado do projeto. São três variáveis diferentes (ver **Figura 6**).

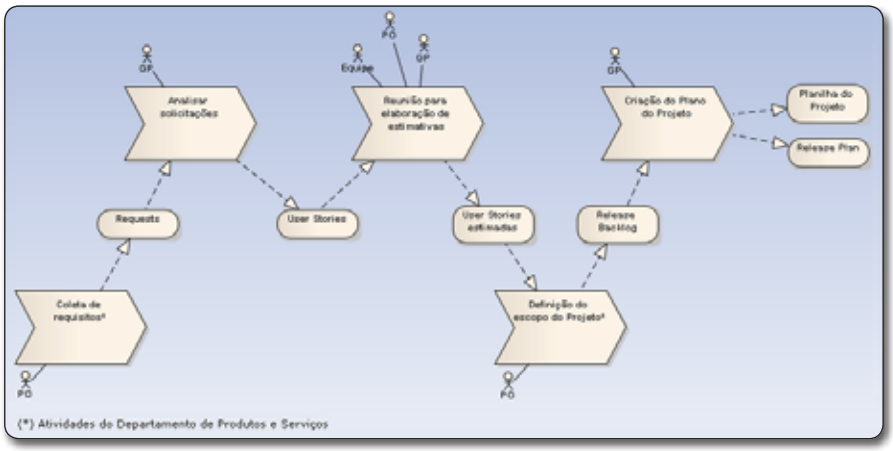

Figura 4. O papel das estimativas no desenvolvimento do plano de projeto

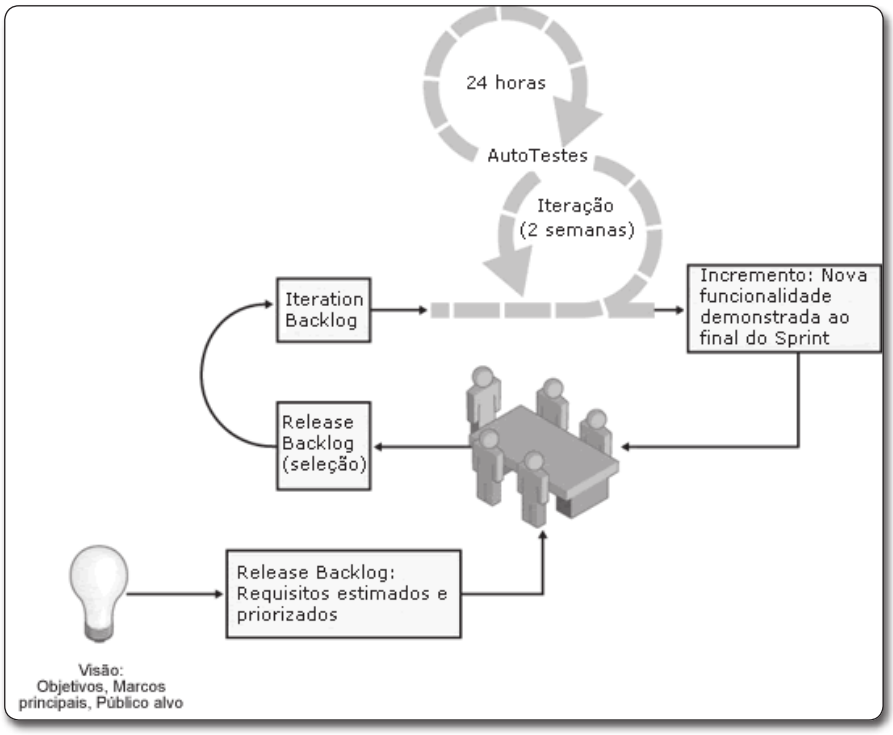

Figura 5. Ciclo de vida

Existem vários procedimentos disponíveis em bibliografia para estimativa de software. Quando o processo foi inicialmente estabelecido, no início de 2009, foi concebido para basear-se no Scrum e utilizar o procedimento mais comum nas metodologias ágeis, os *Story Points*. Uma *User Story* (estória de usuário) é uma breve descrição de uma funcionalidade a ser desenvolvida no projeto, do ponto de vista de um usuário do sistema, escrita de forma livre. Os Story Points são uma unidade de medida aplicada para representar o tamanho / complexidade de cada User Story, na qual o valor isolado de uma estimativa não tem qualquer importância, mas sim o tamanho relativo entre os diversos recursos do projeto. Costuma-se atribuir uma escala não linear de valores, como

a sequência de Fibonacci (1, 2, 3, 5, 8, 13...) ou a progressão geométrica (1, 2, 4, 8, 16...). Isso é importante para ressaltar o fato de que, quanto maior o tamanho da tarefa sendo estimada, menor a precisão da medida. Por exemplo, não há sentido em estimar uma tarefa com 8 e outra com 9 pontos.

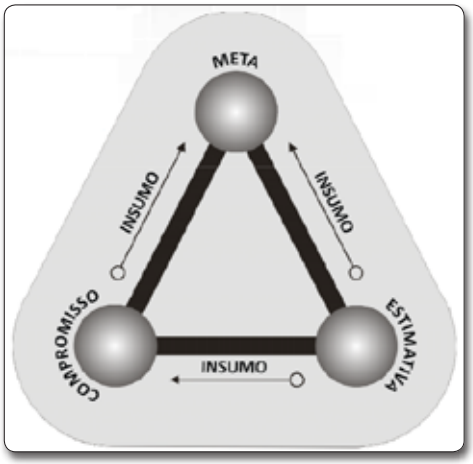

Figura 6. Meta, estimativa e compromisso

Esse critério foi usado durante o primeiro ano de implantação do processo, gerando resultados interessantes e trazendo bom grau de controle ao projeto no qual estava sendo aplicado, mas apresentou algumas deficiências. Em primeiro lugar, mesmo mantendo a mesma equipe na execução de uma sequência de projetos, esse procedimento permitia definir uma escala relativa de tamanho adequada entre as Stories desse projeto, mas apresentava dificuldades na comparação de um projeto com o outro, reduzindo a eficácia das informações históricas disponíveis. Em segundo lugar, e mais importante, como a empresa possuía outras equipes trabalhando em outras linhas de desenvolvimento, o critério das Story Points utilizava referências diferentes em cada equipe, inviabilizando um controle organizacional do desempenho dos diversos projetos.

Para melhorar o processo, foram avaliadas duas outras importantes técnicas de estimativa de software:

• Pontos de função (Function Points): técnica mais utilizada pelo mercado para mensuração do tamanho de projetos de desenvolvimento e melhoria de sistemas. Objetiva a determinação do tamanho funcional do sistema através da visão do usuário, independentemente da tecnologia utilizada. O método mais usado para contagem dos pontos de função é a Contagem Indicativa proposta pela NESMA (Netherlands Software Metrics Association), na qual as funcionalidades são separadas em cinco tipos: entradas, saídas, consultas, arquivos internos e interfaces externas.

• Pontos por casos de uso (*Use Case Points*): métrica apresentada ao mercado como uma solução mais simples para as estimativas de tamanho, baseada na contagem dos casos de uso utilizados para representar os requisitos de sistema e dos atores envolvidos com os mesmos, obtendo o número de pontos por caso de uso não ajustados. A esse valor são aplicados fatores de complexidade técnica e complexidade ambiental, obtendo o número de pontos por caso de uso ajustados.

Nas três técnicas, o principal conceito comum é o de utilizar o tamanho e não o esforço como estimativa de qualquer tarefa. O tamanho representa uma medida da atividade que deve ser independente do profissional que a executa. Para que se obtenha o esforço estimado (número de horas necessárias), deve-se conhecer a produtividade do profissional ou da equipe, valor que só pode ser obtido na execução de atividades anteriores que tenham sido estimadas da mesma forma (ver **Figura 7**).

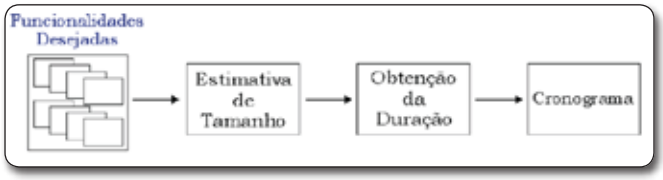

Figura 7. Estimativa por tamanho

À medida que as estimativas são realizadas, elas podem ser adicionadas em uma base histórica de estimativas, servindo de referência para novos projetos. Esse histórico de estimativas permite o refinamento das mesmas e a consistência entre as estimativas realizadas. Isso é válido desde que se use o mesmo critério de medida consistente para as diversas estimativas (ver **Figura 8**).

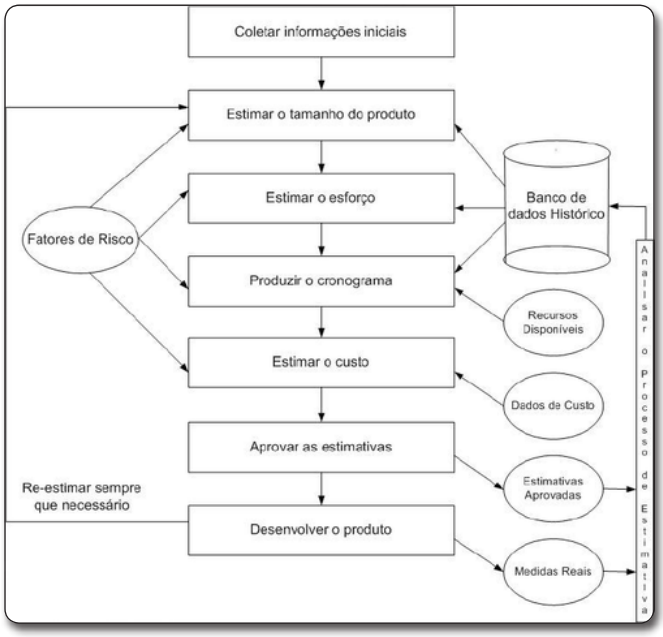

Figura 8. Processo de estimativas alimentando a base histórica

Na AltoQi, nenhuma das três técnicas pôde ser aplicada diretamente. Os *Story Points* apresentavam a dificuldade de comparar os resultados entre equipes diferentes. A métrica dos Pontos de Função objetiva justamente definir um critério de medida que seja válido entre diferentes organizações, mas o processo de contagem proposto pela NESMA (2004) não é bem definido para aplicações baseadas em computação gráfica, dificultando a adoção desse método. A aplicação dos Pontos por Caso de Uso está limitada a projetos de software

cuja especificação seja feita na forma de casos de uso. Além disso, não existe um padrão único para a escrita do caso de uso, deixando o resultado da contagem dependente da forma como cada profissional escreve os casos de uso. Essa definição também não seria suficientemente consistente entre equipes diferentes.

Procurando aproveitar o melhor de cada técnica, a equipe criou uma escala própria de decomposição funcional (pontos de função) para seus requisitos. Ao invés de estimar diretamente uma User Story, a mesma é dividida em quantos aspectos funcionais possam ser identificados (ferramentas gráficas, algoritmos de dimensionamento, geração de desenhos, etc.) e cada uma dessas partes é estimada. Ao reduzir um problema a aspectos mais genéricos, tornou-se possível manter uma consistência entre diversos projetos e mesmo entre as equipes que lidam com Produtos diferentes. Os resultados obtidos com as estimativas nunca são totalmente precisos, mas são úteis e servem ao propósito estabelecido pela organização. É com base neles que o desempenho de cada um dos projetos pode ser acompanhado e o conceito de desenvolvimento colaborativo, exposto a seguir, pôde ser aplicado com a confiabilidade necessária.

Desenvolvimento colaborativo

No início de 2009, a empresa preparava-se para iniciar o desenvolvimento de uma nova versão de seu principal produto, o AltoQi Eberick V6. O processo de desenvolvimento a ser adotado havia sido concebido e uma ferramenta para gerenciamento dos projetos havia sido pesquisada, customizada e implantada, suportando o processo definido. O escopo do produto estava sendo selecionado pelo DPS, dentre uma lista de cerca de 300 sugestões reportadas pelos clientes ao longo dos últimos anos. Era sabido que havia tempo e equipe disponível para atender apenas a uma pequena parte dessa demanda. O foco desse projeto, conforme definido pela Direção da empresa, deveria ser o de fornecer aos seus clientes recursos que maximizassem a sua produtividade e melhorassem a sua experiência com o produto.

Além de partir dessa definição bastante abrangente, era esperado um ciclo de desenvolvimento de 18 a 24 meses. Como a última versão do produto havia sido lançada havia mais de dois anos, o ciclo total superaria os quatro anos, uma demora que vinha sido percebida pelos clientes como incapacidade da empresa de criar novos produtos. Esperar o final do projeto para fazer a sua divulgação, como havia sido feito nos lançamentos anteriores, poderia causar uma evasão expressiva da base de clientes.

Nesse cenário, a empresa resolveu aplicar o conceito do desenvolvimento iterativo, concebido apenas para controlar internamente o andamento dos projetos, também para definir o produto junto aos seus clientes. Para isso, foi criado o programa AltoQi Eberick Next, um conjunto de 8 Releases, com duração de 2 a 3 meses cada. O conjunto desses projetos tinha por objetivo a criação da nova versão do produto AltoQi Eberick, através de um modelo de planejamento progressivo. Foi definido, apenas em alto nível, o escopo do produto completo. Foi planejado o primeiro Release, contendo os recursos considerados mais prioritários pela equipe, e pré-definido o escopo do Release 2 mas, para os demais, ainda seria aguardado o retorno dos clientes (ver **Figura 9**).

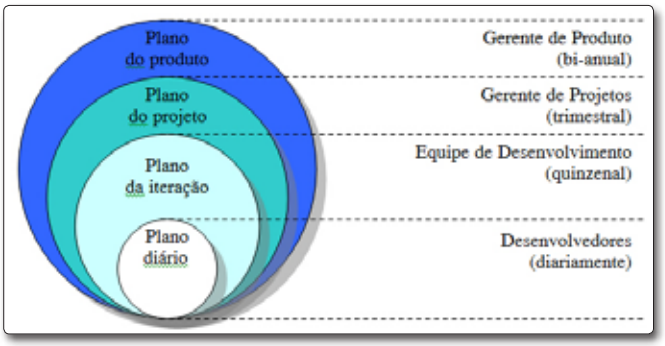

Figura 9. Planejamento progressivo

Cada um desses Releases deveria ser concebido como um projeto independente, ao final do qual o incremento do produto estaria completo, com os recursos selecionados tendo sido completamente especificados, implementados e testados. Ao final de cada Release, deveria existir um Produto parcial completamente estabilizado, de forma a poder ser enviado para a base de usuários participantes do programa Eberick Next. Cada cliente participante desse programa estaria habilitado a receber as versões intermediárias do produto, com a possibilidade de aplicá-las imediatamente nos seus projetos, sem as restrições de uma versão "beta". Essa proposta, evidentemente, só foi viabilizada pelo investimento que havia sido feito pela empresa, nos dois anos anteriores, na criação da sua base de testes automatizados.

Além da liberação de versões intermediárias do produto, a empresa decidiu criar um ambiente de colaboração com seus usuários, na forma de um blog de desenvolvimento, o Blog Eberick Next. Nesse blog, eram publicados os recursos do projeto, conforme eram desenvolvidos, permitindo o *feedback* dos clientes mesmo antes do fechamento do Release. Isso permitia discutir com os clientes acerca dos critérios adotados e coletar opiniões a respeito de recursos que estavam planejados para os próximos releases (ver **Figura 10**).

No período de Abril/ 2009 a Novembro/ 2010, o Blog do Eberick Next teve 150 *posts* publicados, os quais foram acompanhados por 826 profissionais registrados, em mais de 72.000 visualizações, os quais deixaram mais de 5.600 comentários, todos tratados pela equipe de desenvolvimento do produto.

Resultados obtidos

Pode-se afirmar que o projeto Eberick Next foi um sucesso. A iniciativa de dividir o produto em Releases funcionais, disponíveis para os usuários, bem como o estabelecimento do Blog de desenvolvimento, pode ser avaliada sob diversas perspectivas:

• Do ponto de vista dos clientes, a oportunidade de participar da definição dos rumos do projeto foi muito bem acolhida.

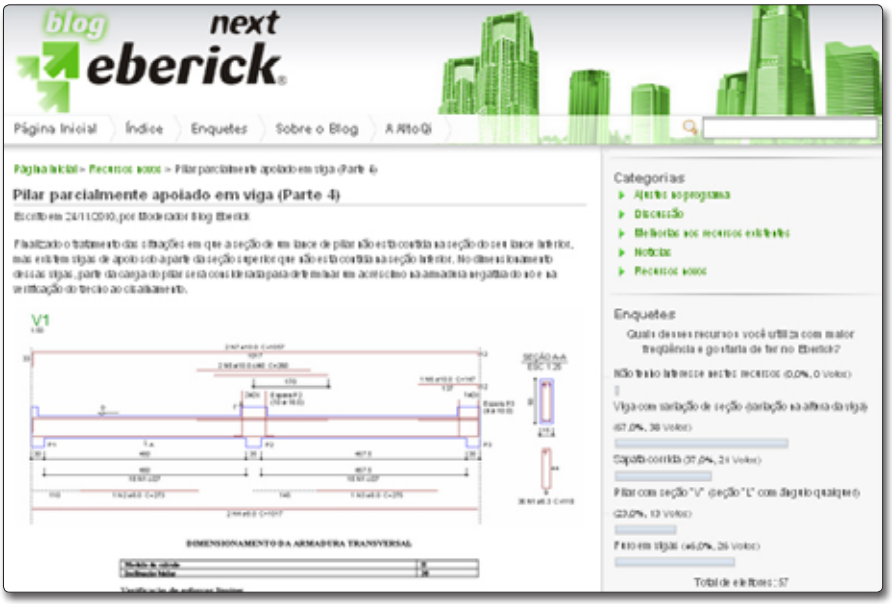

Figura 10. Blog do Eberick Next

Mesmo os clientes que não participaram diretamente do projeto Next reportaram sentir-se mais confiantes com a empresa e com o produto, por saber que teve a contribuição de outros usuários.

• Do ponto de vista da equipe de Desenvolvimento, garantiu um feedback antecipado para o projeto, resultando em um produto de qualidade superior e uma elevação do moral do grupo.

• Do ponto de vista da equipe Comercial, a divulgação prévia e continuada do produto, ao longo do seu desenvolvimento, facilitou bastante o posterior trabalho de argumentação comercial, mesmo junto a clientes que não haviam participado do projeto Next.

• Do ponto de vista da equipe de Marketing, essa iniciativa levou a um fortalecimento da marca e da imagem da empresa perante os seus clientes.

É interessante comentar que o desenvolvimento desse produto partiu de um banco de cerca de 300 solicitações registradas e pôde materializar 60 novos recursos no programa mas, ao final, a lista de solicitações registradas já superava 650. Cerca de metade dos recursos desenvolvidos nem estava na lista original de necessidades. Isso ressalta a importância de conhecer as necessidades atuais dos clientes e ilustra o fato de que os clientes conseguem expressar muito melhor essas necessidades depois de experimentar parte do produto já concluído.

Ao final, mais importante do que demonstrar a aplicação dos conceitos do desenvolvimento iterativo para a Direção da empresa, com o *feedback* positivo dos clientes participantes dessa iniciativa, foi o resultado comercial obtido pelo produto. Após completar os oito Releases, o produto foi oficialmente fechado como "AltoQi Eberick V7" e lançado comercialmente a toda a base de clientes da empresa. Dos 826 profissionais que participaram, mais de 600 efetuaram imediatamente a migração para a versão V7. Outros 220 profissionais que não participaram do projeto Next também adquiriram a licença no lançamento, apenas pelo histórico desse projeto.

Avaliação MPS.BR

O projeto Eberick Next teve como foco a preocupação da empresa pela melhoria dos produtos e serviços oferecidos. Como apoio à melhoria interna do processo de desenvolvimento, a empresa optou por alinhar essa iniciativa com um modelo destinado especificamente ao mercado de desenvolvimento de software brasileiro, o MPS.BR (modelo de Melhoria de Processos de Software Brasileiro), baseado nas normas NBR ISO/IEC 12207, ISO/IEC 15504 e no modelo internacional de melhoria de processos de software CMMI-DEV, com o objetivo de guiar a melhoria dos processos das micro, pequenas e médias empresas do mercado de software brasileiro.

O modelo MPS.BR define 7 níveis de maturidade de processos: A (em otimização), B (gerenciado quantitativamente), C (definido), D (largamente definido), E (parcialmente definido), F (gerenciado) e G (parcialmente gerenciado). A escala de maturidade se inicia no nível G e progride até o nível A. Em cada um dos níveis, há um conjunto de processos que indicam onde deve ser feito o esforço de melhoria. Os níveis são cumulativos, o que significa dizer que uma organização aprovada no nível F possui as capacidades de ambos os níveis F e G (ver **Figura 11**).

Conforme colocado no decorrer deste artigo, a melhoria de processos no Departamento de Desenvolvimento da AltoQi é algo que vem sido trabalhado desde 2007, até chegar ao modelo de desenvolvimento iterativo implantado em 2009, o qual ganhou destaque com o desenvolvimento do Blog do Eberick Next. A formalização dos procedimentos foi uma necessidade que a empresa observou no caminho de ampliação da sua equipe e fortalecimento do Departamento.

No final do ano de 2009, usando como piloto o próprio projeto Next, a empresa resolveu buscar para si essa certificação de qualidade, validando o investimento em processo que havia sido feito e visando posicionar a empresa no início dessa jornada em direção à excelência. O primeiro passo foi a implantação do nível G do MPS.BR, o qual envolve as áreas de Gerência de Projetos e Gerência de Requisitos. Esse trabalho transcorreu durante um ano e culminou em uma avaliação impecável, ao final de 2010.

O sucesso nessa avaliação vem trazendo diversos benefícios à empresa, mas o atendimento às melhores práticas do desenvolvimento de software é o principal fator a ser considerado, resultando em projetos mais eficientes e organizados, com benefícios diretos para os usuários dos produtos AltoQi.

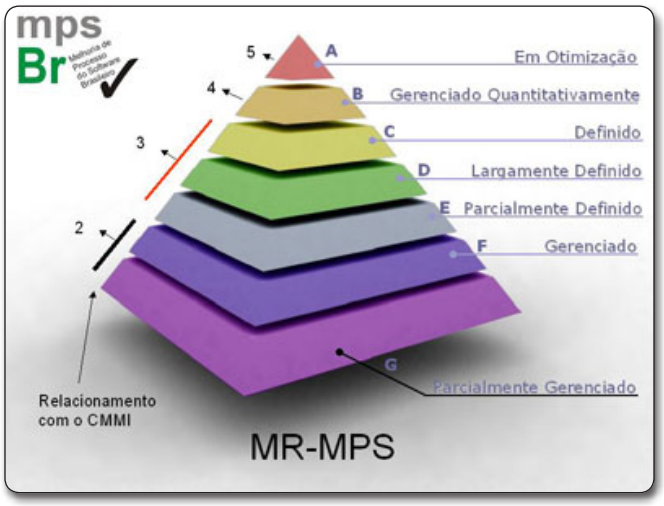

Figura 11. Modelo MPS.BR

Conclusões

Na área de desenvolvimento de software, muitos são os fatores que determinam o sucesso de um empreendimento. Certamente, sem a inspiração para a criação de um produto ou serviço diferenciado, ou sem contar com as pessoas adequadas e motivadas para o desenvolvimento do produto, não é provável que a empresa possa colher bons resultados. Mesmo o melhor dos processos não garantiria o sucesso dos projetos. Por outro lado, a adoção de um processo inadequado pode, certamente, minar a produtividade de uma equipe, desviá-la dos objetivos do negócio e reduzir a moral da empresa como um todo.

Neste artigo, foi mostrado como a aplicação de uma metodologia de desenvolvimento bem estruturada permitiu à AltoQi retomar o seu ritmo de crescimento e estabelecer uma nova forma de interação com seus usuários. Todavia, adotar apenas uma metodologia de gerenciamento como o Scrum não seria suficiente. Para que isso fosse possível, um investimento inicial na adoção de boas práticas de Engenharia de Software, com destaque para a automação dos testes, foi imprescindível. Essa experiência bem-sucedida validou o início de um novo projeto, nos mesmos moldes, e está sendo usada como base para todas as demais linhas de produto da empresa.

Além disso, comentou-se o fato de que, embora o processo de desenvolvimento adotado tenha se baseado em metodologias ágeis, isso não limitou, de forma alguma, a possibilidade de certificar esse processo em relação a um modelo de maturidade.

O mesmo processo que permitiu à empresa a adoção de um modelo de desenvolvimento colaborativo pôde ser avaliado, com sucesso, no nível G do MPS.BR.

Espera-se que este relato de experiência possa auxiliar outras empresas a definir seu próprio caminho rumo à melhoria contínua no seu processo de desenvolvimento e na maximização dos resultados dos seus projetos.

Referências

BONA, Cristina. Avaliação de Processos de Software: Um Estudo de Caso em XP e ICONIX. Dissertação de Mestrado em Ciências da Computação. Universidade Federal de Santa Catarina, 2002.

COHN, Mike. Agile Estimating and Planning. Prentice Hall, 2005.

HAZAN, Cláudia e VON STAA, Arndt. Análise e Melhoria de um Processo de Estimativas de Tamanho de Projetos de Software. 2005. *http://www.dbd.puc-rio.br/depto_informatica/05_04_hazan.pdf>*

JAKOBSEN, Carsten e JOHNSON, Kent. Mature Agile with a twist of CMMI. 2011. *http://jeffsutherland.com/scrumpapers.pdf*

KRUCHTEN, Philippe. The Rational Unified Process, An Introduction. 2ª ed. Reading, Massachusetts: Addison-Wesley, 2000.

MPS.BR (Melhoria de Processo do Software Brasileiro). Guia Geral. Sociedade SOFTEX, 2009.

_____. Guia de Implementação – Parte 1: Fundamentação para Implementação do Nível G do MR-MPS. Sociedade SOFTEX, 2009.

MYERS, Glenford J. et al. The Art of Software Testing. 2ª ed. John Wiley & Sons, 2004.

NESMA (Netherlands Software Metrics Association). Definitions and counting guidelines for the application of function point analysis. NESMA Manuals, 2004.

PMI (Project Management Institute). PMBOK (Project Management Body of Knowledge) - Um Guia do Conjunto de Conhecimentos em Gerenciamentos de Projetos. 2004.

SCHWABER, Ken. Agile Project Management with Scrum. Redmond, Washington: Microsoft Press, 2004.

VASQUEZ, Carlos Eduardo. Estimativas de Software – Fundamentos, Técnicas e Modelos... e o principal, integrando tudo isso. Apresentação realizada na Engenharia de Software Conference. DevMedia, 2009.