ENGENHARIA DE REQUISITOS EM METODOLOGIAS ÁGEIS

Sérgio de Rezende Alves, André Luiz Alves
Universidade Católica de Goiás (PUC – Goiás)
Goiânia - Go – Brasil
sergiorezendealves@gmail.com, andre.luiz@ucg.br
André Luiz Alves – Especialista

Abstract. *Processes and professionals of the requirements area engineering must have competencies to meet the new ways of thinking and producing software in order to avoid results such as: Building and the wrong product. The advances in computational resources and the very dynamic of human activities have expanded the use of software applications to meet before facing internal processes of organizations. Identify, understand, organize and attend this environment where changes may occur faster than the ability to accompany them, undeniably, a difficult task. We present an analysis of good practices in addressing requirements suggested by agile modeling.*

Keywords*: Requirements Engineering, Agile Methodologies, Best Practices.*

Resumo. *Processos e profissionais da Engenharia de Requisitos devem possuir competências para corresponder aos novos modos de se pensar e produzir software de forma a evitar resultados como: Construir bem o produto errado. Os avanços dos recursos computacionais, de comunicação e a própria dinâmica das atividades humanas ampliaram o uso das aplicações de software antes voltado a atender processos internos das organizações. Identificar, entender, organizar e atender a este ambiente onde as mudanças podem ocorrer mais rápido que a capacidade de acompanhá-las é, inegavelmente, uma tarefa difícil. Neste contexto, apresentamos uma análise das boas práticas na abordagem de requisitos sugeridas pela modelagem ágil.*

Palavras-Chave: *Engenharia de Requisitos, Metodologias Ágeis, Boas Práticas.*

1. Introdução

A produção de software é marcada desde o início por dificuldades ligadas a natureza abstrata deste produto. Não se trata de algo físico para o qual uma comparação com modelos conhecidos possa fornecer uma idéia de sua qualidade ou estado. Entregar um produto que atenda as expectativas dos usuários em termos de funcionalidades, custos e prazos ainda é um desafio nos dias atuais. Estes problemas são evidenciados em dados do estudo do *Standish Group* – 2000 [1], indicando que 66% dos projetos de *software* desenvolvidos não atenderam as necessidades dos usuários. Entretanto o estudo chamou a atenção para um fato: muitos projetos que tiveram êxito foram superestimados chegando, em alguns casos, acima de 150%.

A construção de uma base metodológica forte, confiável e eficiente que oriente a execução dos projetos é fundamental para favorecer os bons resultados. A formação desta base exigirá dos profissionais de software um sério e disciplinado comprometimento, a exemplo de áreas como a construção civil que possui uma invejável maturidade metodológica elaborada ao longo de várias décadas pela apropriação de conhecimentos, adquiridos empiricamente, em um conjunto de técnicas, métodos, padrões que auxiliam a construção e manutenção de projetos.

É comum que em um processo de amadurecimento várias tendências disputem espaço em sua área, o que não é diferente para a Engenharia de Software. As metodologias tradicionais orientadas a documentação e as metodologias ágeis baseadas na interatividade das pessoas e no mínimo de documentação são dois exemplos destas orientações.

A Engenharia de Requisitos participa deste contexto ao abordar um dos principais problemas do desenvolvimento de sistemas: Os requisitos. Frederick P. Brooks em seu artigo "No Silver Bullet - Essence and Accidents of Software Engineering" [3] destaca a dificuldade que envolve os requisitos, ao afirmar: "*A parte mais difícil da construção de um sistema é decidir precisamente o que construir. Nenhuma outra parte do trabalho conceitual é tão difícil como estabelecer os requisitos técnicos detalhados, incluindo todas as interfaces para as pessoas, às máquinas, e outros sistemas de software. Nenhuma outra parte do trabalho gera tanto prejuízo ao o sistema resultante se feita de forma errada. Nenhuma outra parte é mais difícil de corrigir mais tarde*".

É importante ter clareza do contexto ao analisar qualquer aspecto envolvendo a produção de software, sejam ferramentas, processos ou métodos. Isto significa submeter tais proposições a um contexto de natureza não simples. Este artigo propõe uma análise da abordagem de requisitos em metodologias ágeis.

O artigo inicialmente apresenta os aspectos da natureza do software, fator cultural e engenharia de requisitos. Em seguida aborda o principio das metodologias ágeis seguindo a análise de suas boas práticas. E, por fim conclusões e sugestões de estudos complementares.

2. A Natureza do Software

Quando um problema apresenta vários aspectos, as soluções propostas podem hora privilegiar um ou outro. Frederick P. Brooks definiu quatro dificuldades inerentes a natureza do software: Complexidade, modificabilidade, conformidade e visibilidade.

Complexidade:

- Não há duas peças de software idênticas (exceto sub-rotinas em bibliotecas);
- O número de estados em software é enorme. Existe dificuldade de enumerar e entender todos os possíveis estados, gerando ameaças para a segurança do sistema e falta de confiança no produto;
- Para aumentar um software não basta repetir os mesmos elementos em tamanho maior, mas sim acrescentar um grande número de outros elementos complexos;
- Uma descrição de um software que abstraia sua complexidade tende a abstrair também a sua essência. Da complexidade surge a dificuldade de comunicação na equipe de desenvolvimento, o que aumenta atrasos, custos e erros. Funções complexas são difíceis de usar, de serem estendidas sem gerar efeitos colaterais e de serem gerenciadas.

Conformidade: O software deve ser integrado em um ambiente composto de hardware, pessoas e processos. Boa parte da complexidade do software é resultado de agentes externos com os quais o software deve ter interfaces apropriadas.

Modificabilidade: O software é a parte de um sistema que incorpora suas funções e as funções são as partes de um sistema que mais sofrem pressões para mudar. Existe uma percepção errônea de que o software, por ser um artefato puramente conceitual, é facilmente modificável. Todo software de sucesso sofre modificações.

Invisibilidade: O software é invisível e não-visualizável. Por não estar sujeito a relações físicas ou espaciais, o software não pode ser adequadamente representado por gráficos ou diagramas geométricos. Os vários diagramas que representam o software são visões parciais de aspectos específicos, que não produzem uma visão global e integrada.

3. Fator Cultural

O fator cultural exerce um papel determinante sobre qualquer atividade humana contribuindo positiva ou negativamente na formação das competências. Guy le Boterf [4], descreve o desenvolvimento de competências como sendo a passagem pelos estados de *incompetente inconsciente*, no qual o sujeito não sabe que não sabe alguma coisa; de *incompetente consciente*, onde o sujeito sabe que não sabe algo; de *competente consciente*, no qual o sujeito sabe o que sabe sobre algo; e de c*ompetente inconsciente*, onde o sujeito não sabe o que sabe, pois teria recursos cognitivos mobilizáveis em situações-problema que ainda não conhece. A palavra incompetente pode parecer pejorativa, mas não é esse sentido usual dado ao termo aqui.

As informações de requisito são obtidas em um ambiente organizacional definido, com características culturais próprias de competência e consciência na execução de seus processos. Não é difícil afirmar que o melhor contexto para se obter uma boa especificação de requisitos é aquele em que as atividades são realizadas com competência e consciência.

4. Engenharia de Requisitos

"Caminhar sobre a água e desenvolver software a partir de uma especificação de requisitos é fácil se ambos estão congelados"-E. Berard. A mudança é um dos aspectos que afetam os requisitos. Mesmo quando são fixos ainda existe a possibilidade de que sejam imprecisos, comprometendo a qualidade do software produzido. Quando as características de um produto não são estruturadas em um documento e são transmitidas apenas oralmente, há uma grande possibilidade de sejam compreendidas de diversas formas por diferentes indivíduos ou, pior, as informações serem simplesmente esquecidas [1]. Este é campo de aplicação da Engenharia de Requisitos.

4.1. A necessidade da Engenharia de Requisitos

No início da década de 70, com o rápido crescimento da demanda, a complexidade e a inexistência de técnicas estabelecidas para o desenvolvimento de sistemas, somados a imaturidade da engenharia de software como profissão culminou com a chamada **crise do software.** O termo citado Edsger Dijkstra [5] é caracterizado por situações como:

- Projetos estourando o orçamento;
- Projetos estourando o prazo;
- Software de baixa qualidade;
- Software muitas vezes não atingiu os requisitos;

• Projetos ingerenciáveis e o código difícil de manter.

O estudo do *Standish Group - Chaos Report* de 2000, citado Wagner Zaparoli[2] indica que em grande parte dos projetos que fracassam os motivos relacionam-se com:

• Objetivos não esclarecidos;

• Ausência de planejamento;

• Requisitos e especificações incompletos;

• Falta de controle na mudança de requisitos.

Entre 40% e 60% dos problemas encontrados em um projeto de software, estão relacionados a falhas na fase de levantamento de requisitos [6]. Muitos destes erros poderiam ser evitados se as organizações dispusessem de um processo de requisitos definido, controlado, medido e aprimorado [7]. Pressman [8] afirma que: *"... entender os requisitos de um problema está entre as tarefas mais difíceis enfrentadas por um engenheiro de software. Quando você começa a pensar sobre isso, a engenharia de requisitos não parece tão difícil. Afinal de contas, o cliente não sabe o que é necessário? Os usuários finais não deveriam ter um bom entendimento das características e funções que vão oferecer benefícios? Surpreendentemente, em muitos casos, a resposta a essas perguntas é não. E mesmo que clientes e usuários finais sejam explícitos quanto as suas necessidades, essas vão se modificar ao longo do projeto. A engenharia de requisitos é difícil".*

Os sistemas computacionais, inicialmente utilizados como ferramentas de apoio, passaram a ser parte essencial e estratégica do processo de tal forma que o processo não mais ocorre havendo falha no sistema que o suporta, bem como a melhoria do processo está diretamente ligada a melhoria do sistema computacional.

As expectativas de risco e de qualidade são determinantes sobre a exigência de uma base metodológica bem fundamentada para o desenvolvimento de um projeto. Quem, em sã consciência, moraria em um prédio de 20 andares construído sem um projeto bem elaborado? E se este prédio estivesse no Japão, onde tremores de terra são comuns? Organizações podem se sentir confortáveis com a economia obtida em projetos artesanais quando não entendem os riscos.

Os resultados negativos das pesquisas, as dificuldades que as empresas têm em lidar com os requisitos, os riscos envolvidos e as exigências de qualidade dos processos suportados por sistemas, justificam a necessidade de criação da disciplina de Engenharia de Requisitos em 1993, com a realização do *International Symposium on Requirements Engineering*, regida pela Engenharia de *Software,* que tem o objetivo prioritário de normatizar o uso e a gestão dos requisitos.

4.2. Conceitos de Requisito e Engenharia de Requisitos

Segundo Dorfmann e Thayer [9], requisito de software representa a capacidade requerida pelo usuário que deve ser encontrada ou possuída por um determinado produto ou componente de produto para resolver um problema ou alcançar um objetivo ou satisfazer a um contrato, padrão, especificação ou a outros documentos formalmente impostos.

A engenharia de Requisitos é definida por Rocha [10] como uma subárea da engenharia de software que tem por objetivo tratar o processo de definição dos requisitos de software. Na visão de Sommerville [11] a Engenharia de Requisitos é definida como o processo de descobrir, analisar, documentar e verificar as funções e restrições do sistema.

Mesmo com o desenvolvimento rápido agregando metodologias, técnicas e ferramentas para tratar requisitos, alguns problemas que persistem em grande parte nos projetos atualmente desenvolvidos. Alencar [12] cita alguns deles:

- Falta de envolvimento das partes participantes;
- Falta de gerenciamento e rastreamento de requisitos;
- Falta de definição das responsabilidades;
- Falta de comunicação entre os envolvidos.

4.3. Processo de Engenharia de requisitos

A elaboração e a manutenção de uma especificação de requisitos são realizadas através de um conjunto estruturado de atividades denominado Processo de Engenharia de Requisitos, destinadas a elicitar, analisar, documentar e validar os requisitos [13].

Elicitar refere-se à atividade voltada para descobrir (identificar, deduzir, extrair, evocar, obter) os requisitos de um sistema. Pode se utilizar de entrevistas com os usuários interessados pelo sistema, de documentos de processos e sistemas existentes, da análise do domínio do problema ou de estudos de mercado.

Na análise, os requisitos elicitados são compreendidos e detalhadamente analisados por todos os interessados no sistema. Nessa atividade surgem muitos conflitos, sendo comum haver a necessidade de negociação para que os requisitos sejam aceitos por todos.

Uma vez compreendidos, analisados e aceitos, os requisitos devem ser documentados com um nível de detalhamento adequado, produzindo a especificação de requisitos do software. Pode ser utilizada a linguagem natural ou diagramas, como os propostos pela UML.

Após terem sido documentados, é necessário que os requisitos sejam cuidadosamente validados, principalmente quanto à consistência e a completude. Esta atividade visa identificar problemas nos requisitos, antes do início da construção. A importância desta atividade é caracterizada pelo fato de que a correção de um erro nesta fase possui um custo muito inferior do que a correção nas fases mais adiantadas do processo de desenvolvimento (até 200 vezes menor [2]).

A estruturação destas atividades e dos resultados esperados é definida conforme a metodologia adotada pela organização. Em linhas gerais estas metodologias se diferem quanto ao maior ou menor rigor formal aplicado na execução das atividades e nos artefatos produzidos.

Orientações, modelos e padrões atribuem referencias de qualidade ao estabelecerem que os processos devem apresentar resultados gerados como:

- A Rational-2000 [14] afirma que projetos de *software* eficazes possuem as seguintes características em relação a requisitos:

  • Os requisitos realmente devem refletir as necessidades dos clientes;

  • Os requisitos devem ser compreendidos de forma coesa;

  • As expectativas dos clientes devem ser gerenciadas com eficácia;

  • As mudanças de requisitos devem ser gerenciadas.

- O Guia de Implementação do MPS.BR Nível –G [15] define que os processos relacionados a gerencia de requisitos devem produzir os seguintes resultados:

  • GRE1 - Os requisitos são entendidos, avaliados e aceitos junto aos fornecedores de requisitos, utilizando critérios objetivos;

  • GRE2 - O comprometimento da equipe técnica com os requisitos aprovados é obtido;

  • GRE3 - A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida;

  • GRE4 - Revisões em planos e produtos de trabalho do projeto são realizadas visando a identificar e corrigir inconsistências em relação aos requisitos;

  • GRE5 - Mudanças nos requisitos são gerenciadas ao longo do projeto.

As metodologias tradicionais têm uma orientação para que as atividades e artefatos sejam formalmente documentados e controlados. Por outro lado, as metodologias ágeis têm seu foco na iteratividade dos interessados no projeto e menos na documentação. Algumas organizações podem desenvolver estas atividades sem atender a nenhum processo estruturado, seja por desconhecimento ou falta de recursos, o que favorece a produção de software de baixa qualidade. Cabe observar que esta afirmação será verdadeira ou não, conforme o critério de qualidade estabelecido.

5. Metodologias Ágeis

Com o "Manifesto Ágil" em 2001, o termo "Metodologias Ágeis" se tornou conhecido e vem se difundido entre equipes desenvolvimento. O foco nas pessoas e interações cria novas condições para se abordar requisitos e algumas boas práticas são recomendadas para o sucesso desta atividade. Os princípios comuns compartilhados por todos esses métodos foram estabelecidos através do "Manifesto Ágil" [16] e se fundamentam em:

- **Indivíduos e interações** ao invés de processos e ferramentas;
- **Software executável** ao invés de documentação;
- **Colaboração do cliente** ao invés de negociação de contratos;
- **Respostas rápidas a mudanças** ao invés de seguir planos.

A maioria das metodologias ágeis nada possuem de novo [17]. O que as diferencia das metodologias tradicionais são o enfoque e os valores. A idéia das metodologias ágeis é o enfoque nas pessoas e não em processos ou algoritmos. Além disso, existe a preocupação de gastar menos tempo com documentação e mais com a implementação. O "Manifesto Ágil" não rejeita os processos e ferramentas, a documentação, a negociação de contratos ou o planejamento, mas simplesmente mostra que eles têm importância secundária quando comparado com os indivíduos e interações, com o software estar executável, com a colaboração do cliente e as respostas rápidas a mudanças e alterações. Esses conceitos aproximam-se melhor com a forma que pequenas e médias organizações trabalham e respondem a mudanças. Entre as metodologias ágeis a mais conhecida é a *Extreme Programming* (XP) e Scrum [1].

Alguns resultados efetivos já podem ser percebidos, mesmo considerando que as metodologias ágeis ainda estão em sua infância. Comparações entre métodos ágeis e metodologias tradicionais mostram que projetos utilizando os métodos ágeis obtiveram melhores resultados em termos de cumprimento de prazos, de custos e padrões de qualidade. O mesmo estudo demonstrou ainda que o

tamanho dos projetos e das equipes que utilizam as metodologias ágeis tem crescido. Apesar de as metodologias ágeis serem propostas para equipes pequenas (até 12 desenvolvedores), aproximadamente 15% dos projetos que as utilizam foram desenvolvidos por equipes de 21 a 50 pessoas, e 10% dos projetos, por equipes com mais de 50 pessoas, considerando um universo de 200 empresas avaliado no estudo [1].

O uso das metodologias ágeis em grandes projetos com grandes equipes pode exigir a adoção de soluções para problemas relativos a segurança e comunicação.

6. Abordagem de Requisitos em Metodologias Ágeis

Requisitos são fatores críticos na construção de sistemas, construir bem o produto errado é um dos piores resultados do desenvolvimento de software. Esta condição pode ocorrer quando existem distorções de entendimento dos requisitos entre usuários e desenvolvedores. A literatura da engenharia de requisitos coleciona várias técnicas que procuram minimizar tais desencontros, entre as quais as metodologias ágeis com seu foco na iteratividade do envolvidos e em respostas rápidas como forma de atender os requisitos dos usuários.

Em muitos momentos acreditou-se ter encontrado a “bala de prata” que resolveria os problemas do desenvolvimento de software. È comum ainda hoje ouvir profissionais que se promovem por estarem aplicando a última “onda”, chegando a classificar os demais, que ainda não aderiram a novidade, como ultrapassados. Um fato comum no seguimento de ferramentas e linguagens de programação. A falta de uma visão do contexto pode levar a uma ingênua conclusão de que uma solução que atenda a determinados aspectos do problema, é uma solução para o problema.

A abordagem de requisitos em metodologias ágeis, como qualquer proposta, pode permitir bons resultados a partir de sua combinação com aspectos favoráveis do contexto. Esta combinação pode requerer, por exemplo, que a engenharia de requisitos repense alguns de seus procedimentos principalmente devido ao fato que essas metodologias abdicam, em parte, de documentos e controles de artefatos muito presentes nesta disciplina. A figura-1 é uma proposta de visão da abordagem ágil de requisitos no contexto do desenvolvimento de software.

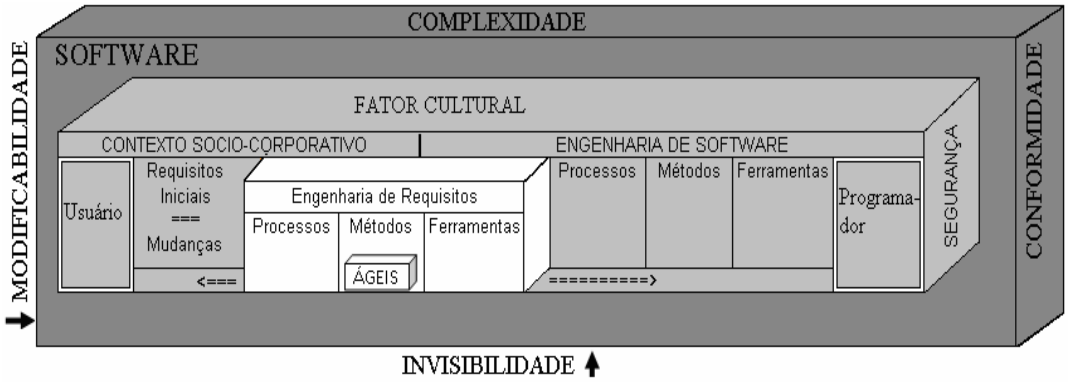

Figura-1. Abordagem ágil de requisitos no contexto do desenvolvimento de software

6.1. Desafios da Abordagem Requisitos em Metodologias Ágeis

Para que a modelagem de requisitos seja ágil é necessário que haja um ambiente propício. A modelagem de requisitos é prejudicada quando a cultura organizacional não favorece o desenvolvimento de software. Em muitos casos os interessados no projeto não entendem as implicações de suas decisões. Identificar o ambiente e os problemas que afetam a modelagem de requisitos deve ser uma primeira tarefa permitindo que, na condução do projeto, sejam discutidas possíveis soluções para lidar com esses problemas. A Agile Modeling(AM)[18] Identifica os seguintes desafios comuns:

- O acesso limitado ao Stakeholders;
- Separação geográfica dos interessados;
- Stakeholders do projeto não sabem o que querem;
- Mudanças de pensamento dos Stakeholders;
- Os conflitos de prioridades;
- Muitas partes interessadas no projeto querem participar;
- Stakeholders prescrever soluções de tecnologia;
- Stakeholders incapazes de ver além da situação atual;
- Stakeholders com receio de serem rebaixados profissionalmente;
- Stakeholders não compreendem os artefatos de modelagem;
- Os desenvolvedores não compreenderem o domínio do problema;
- Stakeholders são excessivamente centrados em um tipo de exigência;
- Stakeholders exigem formalidade significativa com relação aos requisitos;
- Os desenvolvedores não entendem as exigências.

6.2. Boas práticas na Abordagem Requisitos em Metodologias Ágeis

O amadurecimento metodológico é favorecido pela compilação de experiências aprendidas com o objetivo de oferecer um conjunto de recomendações referenciadas pelo termo “Boas Práticas”. É importante ressaltar que as boas práticas não impõem sua aplicação como única forma de se obter os resultados, e sim, sugerem que o sucesso de um projeto é favorecido por sua adoção. Assimilar estas práticas implica em uma mudança na cultura organizacional em função de seu grau de maturidade. É neste ponto que um bom entendimento do contexto sintetizado na Figura-1 pode orientar os envolvidos a perceberem os benefícios obtidos com as mudanças.

Em função das propostas ágeis serem efetivamente focadas no comportamento que levam a produção de um software executável mais que nos artefatos intermediários. O que se pode analisar é o quanto tais comportamentos contribuem para as atividades envolvendo requisitos. A Agile Modeling(AM) [18] fornece uma lista de boas práticas para abordagem de requisitos apresentadas e comentadas a seguir,

6.2.1. Participação ativa dos interessados - Stakeholders

A participação ativa é o elemento central da abordagem ágil. O cliente deve participar do projeto se sentindo parte do time. Um dos objetivos é tornar o cliente ciente das necessidades para que este contribua na remoção de obstáculos, reduzindo o desgaste da equipe de desenvolvimento em justificar dificuldades.

Em relação aspecto da modificabilidade do software, o cliente ativo terá mais condições e perceber os impactos de mudanças dos requisitos favorecendo a negociação.

Esta prática é necessária, porém não suficiente, para que os requisitos sejam entendidos, avaliados e aceitos junto aos fornecedores de requisitos.

A compreensão dos requisitos e sua real adequação as necessidades dos clientes são facilitadas quando estes acompanham o processo.

Um desafio é gerenciar as expectativas dos clientes para estas sejam realistas e exeqüíveis no escopo do projeto.

O envolvimento dos interessados no projeto pode parecer uma condição natural, afinal quem se beneficia de um bom projeto será o próprio usuário. Alencar [12] cita entre os problemas de projetos, a falta de envolvimento das partes participante. Algumas considerações podem justificar o não envolvimento:

- A disponibilidade do usuário é comprometida por suas atividades de rotina e a participação no projeto é vista como

um desvio de suas funções, principalmente por aqueles que ocupam cargos de decisão;

- Adaptar o projeto a disponibilidade do usuário pode comprometer mais tempo que o necessário para a conclusão de uma atividade;
- O envolvimento implica em assumir responsabilidades e desenvolver tarefas relativas ao projeto.

Em curto prazo, a cultura organizacional é mais determinante do que uma nova proposta metodológica. As mudanças podem ocorrer com o amadurecimento da organização. Caso não haja por parte da organização interessada uma postura favorável, será difícil envolver os usuários, o que é um forte indicativo de que o projeto não terá sucesso.

6.2.2. Adotar modelos inclusivos

Uma vez obtida a participação dos usuários, é importante o uso de modelos e ferramentas de modelagem e documentação de requisitos que reduzam as barreiras de comunicação de forma a manter o envolvimento. Algumas formas de representação, simples para um desenvolvedor, podem oferecer um baixo entendimento ao usuário que não tenha domínio dos conceitos envolvidos.

Estes modelos devem favorecem a visibilidade dos requisitos buscando um entendimento global e integrado, evitando a complexidade e representações excessivamente elaboradas e técnicas. Em função do aspecto da invisibilidade, própria da natureza do software, a utilização recurso que representem visões dos requisitos, aliados a um participação ativa do cliente contribuem para que os requisitos sejam entendidos e validados.

O uso de recursos simples como quadro branco, post-it e diagramas em alto nível podem são recomendados por pertencerem ao universo do s usuários. Técnicas de elicitação de requisitos como JAD-Joint Application Design, criam situações de reuniões que dão mais liberdade aos participantes para falarem e usarem salas com lousas ou papel flip chart para expressarem seus pensamentos.

O envolvimento dos usuários é pré-requisito para o sucesso desta prática. A cultura organizacional também assume papel determinante nesta atividade.

6.2.3. Fazer uma primeira abordagem de forma abrangente

Uma primeira abordagem abrangente tem o objetivo de se obter uma compreensão global dos requisitos que permita orientar o projeto e evitar a perda do esforço na especificação detalha de requisitos que podem ser alterados durante o projeto. Deve-se observa que:

- A abordagem deve permitir a definição do escopo do projeto. È importante definir o que será desenvolvido mesmo sem uma especificação detalhada dos requisitos;
- O cliente tendo receio de que suas necessidades não serão entendidas, procura imaginar todas as possibilidades no inicio do projeto;

6.2.4. Detalhe os requisitos Just In Time (JIT)

Os requisitos são trabalhados ao longo de todo projeto. Em ciclos incrementais e mais curtos atendendo ao princípio que o agislista deve abraçar as mudanças. O detalhamento dos modelos de requisitos deve ocorrer o mais próximo de sua implementação. Esta prática exige menos documentação, pois a memória dos detalhes está recente e não em especificações feitas no inicio do projeto. Este é o argumento dos agilistas. Cabe aqui uma questão: O desenvolvimento pode ocorrer próximo ao detalhamento, mas como fica a manutenção que não terá a disposição a documentação detalhada e talvez nem a lembrança do fornecedor do requisito? Uma possível alternativa é ter um código bem documentado.

6.2.5. Trate os requisitos em uma pilha prioridades

A atribuição de prioridades permite organizar o desenvolvimento e a sincronização com a expectativa do cliente. A atribuição de prioridade deve ser administrada, para muitos clientes tudo teria alta prioridade o que significa, na prática, que não existe uma atribuição de prioridades. A montagem de uma pilha de prioridades permite definir quais os próximos requisitos entram em desenvolvimento viabilizando a prática de detalhar os requisitos Just In Time.

6.2.6. O objetivo é implementar o requisito e não documenta-los

Produzir, manter e rastrear documentação de requisitos exige um esforço que na visão ágil pode comprometer a entrega do que realmente é esperado pelo cliente, o software funcionando. Uma documentação deve conter o suficiente para orientar o projeto e não o bastante que possa inviabilizá-lo.

Uma questão a ser considerada é que muitos processos da organização podem ser executados com competência, porém sem a consciência dos detalhes envolvidos. Ao buscar detalhamento dos requisitos torna-se obrigatório uma retomada consciente de detalhes do processo com os usuários que detém informações do negócio tornando-se o melhor momento para o registro deste conhecimento, considerando que, uma vez implementados, ocorrera um gradual esquecimento dos detalhes que só poderão ser descobertos por uma análise do código, caso não haja a devida documentação.

Não produzir uma documentação detalhada pode gerar o benefício em curto prazo de um software entregue de forma mais

ágil, mas deve-se cuidar para que em longo prazo não se crie dificuldade para sua manutenção.

6.2.7. Reconhecer que existem muitos interessados

Muitas pessoas em diversos papeis são afetadas pelo desenvolvimento e / ou implantação de um projeto de software. Consequentemente entre muitos interesses e opiniões surgem diferenças na visão do que deve ser feito, com qual prioridade e como deve ser feito. Neste caso um papel deve ser criado para intermediar e negociar com estes interessados servindo como ponto entre usuários e equipe de desenvolvimento. Este papel recebe o nome de Product Owner na metodogia ágil scrum.

6.2.8. Abordar requisitos independente de plataforma

Abordar requisitos orientados a uma tecnologia pode criar pré-restrições para necessidades dos clientes que são independentes de plataforma. Os termos requisitos orientados a objeto (OO), requisitos estruturados ou requisitos baseados em componentes são exemplos de abordagem relacionada a uma tecnologia de implementação.

6.2.9. Menor é melhor

Decompor processos em funcionalidades. Menores grupos de requisitos favorecem o entendimento, a estimativa, são mais fáceis de priorizar e consequentemente de gerenciar. O desenvolvedor tem mais domínio do que esta sendo feito.

6.2.10. Rastreabilidade

Rastreabilidade é a capacidade de estabelecer uma relação entre artefatos do projeto. Uma matriz de rastreabilidade de requisitos é o artefato criado para gravar essas relações. O mapeamento destas relações pode abranger desde a identificação do requisito passando por modelos de análise, modelos de arquitetura, modelos de design, código fonte, casos de teste e outros artefatos que se queira manter. A matriz de rastreabilidade é um artefato que pode ajudar controlar aspectos de complexidade do software bem como favorecer a visibilidade em projetos envolvendo grande numero de requisitos.

A manutenção da rastreabilidade consome um grande esforço que é aumentado quanto maior for o nível de detalhamento dos artefatos controlado. Rastreabilidade é uma capacidade que não se encaixa no paradigma ágil por se basear fortemente em documentação. Seu benefício é permitir a análise de impacto na mudança de um requisito.

Na visão ágil um profissional com domínio do negócio e familiarizado com o sistema poderia mapear o impacto da mudança. Mas mesmo neste caso pode haver a necessidade de uma mínima rastreabilidade que relacione o requisito ao seu fornecedor.

Este é um ponto difícil de ser tratado. Se a proposta ágil abraça as mudanças, faz entregas rápidas e produz partes menores e incrementais, conclui-se que uma mudança afetará um numero maior de artefatos produzidos e em produção. Como garantir que haverá uma percepção profissional do impacto sem o apoio de uma matriz de rastreabilidade?

Neste caso é aconselhável um debate aberto entre os envolvidos para se tornem conscientes dos custos e dos benefícios de se manter uma matriz de rastrabilidade e tomar a decisão em conjunto.

6.2.10. Explicar as técnicas

Explicar as técnicas é uma pratica que pode levar o usuário a se sentir mais envolvido no projeto ao desmistificar as técnicas que são usadas e os motivos por que são usadas. Esta prática como pode ser verificado foi sugerida para a abordagem de matriz de rastreabilidade no item 6.2.10 deste artigo e tanto neste caso como em outro seu benefício será o envolvimento do usuário.

6.2.11. Utilize palavras apropriadas ao negócio

Um vocabulário adequado facilita comunicação com os usuários e cria uma identificação do projeto de software com o negocio a que se destina. Um glossário com os termos específicos da área de negócio pode facilitar esta prática.

6.2.12. Crie um ambiente descontraido

Um ambiente de stress permanente não favorece a produtividade. Um ambiente descontraído não significa falta de foco nos objetivos.

6.2.13. Obtenha apoio da alta administração

O apoio da alta administração orienta o comprometimento e motiva as demais áreas da organização a se envolverem com as atividades, resultados e a adotarem as mudanças culturais que acompanham projetos de software. O fato é que sem apoio da alta gerencia qualquer mudança poderá encontrar muita resistência e provavelmente não terá sucesso.

7. Conclusão

O estudo da abordagem de requisito através do grupo de boas práticas aplicadas em metodologias ágeis indica que a flexibilidade e a interatividade as tornam o caminho natural a ser adotado por organizações que não possuem metodologias formais de trabalho e podem representar um grande salto de qualidade e satisfação em projetos de software. Também são práticas bem atraentes para pequenas equipes. O ambiente interativo proposto pelas metodologias ágeis é o ideal de todo profissional de requisitos considerando que a elicitação de requisitos e sua representação real das necessidades dos

usuários são fortemente favorecidos pelas aplicação das práticas abordadas. Não é descartada sua aplicação em projetos maiores, entretanto, em ambientes mais complexos envolvendo maiores riscos, é crescente a preocupação com a adoção de modelos de maturidade. Os padrões de qualidade e modelos de maturidade como MPS-BR tendem a ser cada vez mais utilizados como referencia de confiança do mercado para contratação e aquisição de serviços de TI. As metodologias ágeis não podem ignorar esta tendência o que sugere o que sejam realizados estudos que busquem compatibilizar estas correntes.

As metodologias ágeis efetivamente não tratam os requisitos com base em documentação. Este é um ponto polêmico, principalmente no confronto com metodologias tradicionais. Deve-se considerar que a Tecnologia da Informação é uma área que extrapola o aspecto da pura programação, sendo a cultura de uma organização fortemente impactada com adoção de projetos de software. Um efeito verificável se refere ao fato que os usuários, cuja área de negócio é sistematizada, tendem a esquecer gradativamente detalhes do processo. A não documentação do conhecimento na fase de requisitos pode representar uma dificuldade para futura manutenção, através da perda deste conhecimento em médio e longo prazo, o que reduz a condição de melhoria dos processos. A documentação será sempre referenciada como problema, seja quando ausente, incompleta, pesada ou não utilizada. O "problema" da documentação pode ser visto como um desafio merecedor do desenvolvimento de estudos e ferramentas que favoreçam a conciliação dos benefícios de uma documentação clara, com a dinâmica das mudanças dos requisitos. Um bom exemplo são as ferramentas Wiks, sendo a mais conhecida a Wikpedia: http//wikipedia.org.

A aplicação de metodologias ágeis não garante o sucesso de um projeto. Casos fracassos ou de prejuízos advindos de falhas na especificação farão com que a relação do cliente com o projeto seja fortemente testada. Cabe citar que tão sujeito a mudança quanto os requisitos, são as relações humanas. Situações negativas são fontes de conflito que exigem grande atenção dos gestores, sob pena de comprometerem os projetos. Os fatores culturais da organização devem ser bem avaliados para que se possa obter uma adequação entre a metodologia e o ambiente organizacional. A interatividade não pode tornar amador o processo de desenvolvimento de software. Em síntese, o foco na geração de resultados é uma grande lição da abordagem ágil e a práticas envolvendo requisitos devem favorecer a produção de software e agregar valor aos processos organizacionais e não se restringir a atender e pura programação e nem a pura exigência de processos.

Referências

[1] – Koscianski, André -Qualidade de Software: aprenda as metodologias e técnicas mais modernas para desenvolvimento de software / André Koscianski, Michcel dos Santos Soares -- 2ª Ed. -- São Paulo: Novate Editora, 2007;

[2] - Wagner Zaparoli – Engenharia de requisitos : Um fundamento na construção de sistemas de informação- Mestre em Ciência da Computação – MACKENZIE. Consultor de Sistemas; Colunista em Ciência e Tecnologia do jornal Gazeta de Bebedouro; Professor de-Lógica de Programação na UNINOVE;

[3] - No Silver Bullet -Essence and Accidents of Software Engineering Essence and Accidents of Software Engineering Computer Magazine; April 1987 Computer Magazine, abril 1987 by Frederick P. Brooks, Jr., por Frederick P. Brooks, Jr., University of North Carolina at Chapel Hill University of North Carolina em Chapel Hill;

[4] - Guy le Boterf. *L'ingénierie des compétences*.Paris: 1998. Pode ser encontrado resumido no site: http://www.adbs.fr/site/emploi/guide_emploi/competen.pdf;

[5] - Edsger Dijkstra, 1972 - Association for Computing Machinery Turing Award, intitulada "The Humble Programmer" (EWD340), publicada no periódico en:Communications of the ACM;

[6] – Leffingwell, D; Calculating the Return on Investiment from More Effective Requirements Management; American Programmer 10(4); 13-16; 1997;

[7] – José Roberto Blaschek – Gerência de Requisitos – O principal problema dos projetos de software;

[8] - Pressman, Roger S. Engenharia de software I Roger S. Pressman; tradução Rosângela Delloso Penteado, revisao tecnica Fernao Stella R. Germano, Jose Carlos Maldonato, Paulo Cesar Masiero. –6. ed. -- Sao Paulo: McGraw-Hill, 2006;

[9] - [DORFMANN e THAYER, 1990] DORFMANN, M. e THAYER, R. Standards, Guidelines, and Examples of System and Software Requirements Engineering.Los Alamitos, CA: IEEE Computer Society Press, 1990;

[10] - ROCHA, Ana Regina Cavalcanti et al. Qualidade de software. Teoria e prática. São Paulo: Prentice Hall, 2001;

[11] - [SOMMERVILLE, 2003] SOMMERVILLE, I. Engenharia de Software, Addison Wesley, 6a edição, 2003;

[12] - ALENCAR, F. M. R. Mapeando a Modelagem Organizacional em Especificações Precisas. 1999. Tese de doutorado. Centro de Informática.UFPE, Recife;

[13] Wiegers K.E.; Software Requirements; Microsoft Press; 1999;

[14] - RATIONAL Software Corp. O sucesso começa com o gerenciamento de requisitos. São Paulo: Rational, 2000;

[15] - MPS.BR - Melhoria de Processo do Software Brasileiro Guia de Implementação – Parte 1: Fundamentação para Implementação do Nível G do MR-MPS - 2009;

[16]- [Agile Manifesto] Disponível em http://agilemanifesto.org/, acessado em 01 de Setembro de 2009;

[17] - Cockburn, A. e Highsmith, J. "Agile Software Development: The Business of Innovation", IEEE Computer, Sept.,(2001), pp. 120-122;

[18] - Agile Modeling (AM) - Práticas eficazes de Modelagem e Documentação - http://www.agilemodeling.com/ acessado em 06/10/2009;

[19] http://en.wikipedia.org/wiki/History_of_software_engineering - acessado em 01 de outubro 2009.