

Como atender os requisitos arquiteturais de software usando métodos ágeis como SCRUM e XP

*Pontifícia Universidade Católica – PUC-SP
Especialização em Engenharia de Software, novembro de 2009
Jefferson Lira - jefferson.l.vieira@gmail.com
Paulo Fernandes – paulofernandesjr@gmail.com*

Abstract

The proposal of this paper is show how we manage situations where the architecture is crucial for the developing software using agile methods like Scrum and XP. We define a fictional problem and comprehensive for explain how the agile methodologies can manage architecture situations and we cite some possible solutions for that.

1. Introdução

A proposta deste artigo é mostrar de forma abrangente como tratar as situações onde a arquitetura do software torna-se crucial para o desenvolvimento com a utilização de metodologias ágeis como SCRUM e XP (eXtreme Programming). Este artigo foi dividido de forma que facilite o entendimento.

Com o objetivo de informar os conceitos utilizados neste artigo, são citadas definições de **Arquitetura**, **SCRUM** e **XP**. Na segunda seção evidenciamos situações onde a arquitetura é ponto crucial para o sucesso do desenvolvimento do software, na terceira seção é demonstrado como lidar com essas situações, na quarta seção mostramos métricas e comparações com outras metodologias e para finalizar este artigo é apresentada a conclusão, baseando nos fatos aqui evidenciados.

Arquitetura

A arquitetura do software é um ponto de extrema importância no desenvolvimento e deverá ter uma maior atenção quando sua existência é de grande complexidade para o sucesso do software, pois a definição desta poderá não ser a mais apropriada para o negócio. A má escolha de uma arquitetura de software fará com que o projeto possa ser um desastre, já a

melhor escolha propicia uma maior chance para o sucesso do projeto.[1] Abaixo evidenciamos definições sobre o que é a arquitetura de um software.

A arquitetura de software de um programa ou de um sistema é a estrutura ou estruturas do sistema, que incluem elementos de software, propriedades externas e as suas relações. [5]

A arquitetura de software define a estrutura básica do sistema. A arquitetura é modulada em um alto nível de funcionalidades do sistema, gerenciamento e distribuição de dados, qual plataforma será usada, etc. [9]

Arquitetura de software é a estrutura dos componentes do sistema/programa, seus relacionamentos, princípios e diretrizes para o projeto e sua evolução. [10]

Devido as definições citadas acima, formalizamos que não existe uma definição mundial sobre o que é a arquitetura de software. As definições no geral enfatizam que a arquitetura é a descrição do sistema e a soma de pequenas partes dele, e como essas partes se relacionam e cooperam entre si para executar o trabalho do sistema. [2] A qualidade e longevidade do software são determinadas pela sua arquitetura. [1]

Não podemos confundir a arquitetura do software com o design. A arquitetura se preocupa com a seleção de elementos arquiteturais, suas iterações e restrições, já o design são as atividades que se preocupam com a modularização e detalhamento de interfaces, algoritmos, procedimentos e tipos de dados que darão suporte satisfatório a arquitetura. [8]

Um software tipicamente contempla requisitos funcionais e não funcionais, sendo que muitas das vezes um deverá refletir o comportamento do outro. Os requisitos funcionais descrevem as funções que o software deve ser capaz de realizar. Já os requisitos não-funcionais descrevem as qualidades e restrições de como o sistema realiza suas funções. Um software,

portanto, deve exibir atributos de qualidade que atendam aos seus requisitos. [12]

O ideal é que os atributos de qualidade do software sejam identificados e qual a sua influência na arquitetura, por fim relacionar estes atributos as decisões arquiteturais que os proporcionam.

Com um modelo de apoio para definir e organizar os atributos do software importantes para a avaliação de sua qualidade existe a norma ISO 9126. Esta norma é um padrão internacional para avaliação da qualidade do software. Os atributos utilizados para avaliar a qualidade do software são os seguintes:

Funcionalidade: é a capacidade do software realizar as funções que foram especificadas; [13]

Confiabilidade: é a capacidade do software ser seguro e tolerante a falhas; [13]

Usabilidade: é a medida da facilidade do usuário executar alguma funcionalidade do sistema; [13]

Eficiência: é a capacidade do sistema alcançar a resposta dentro do período de tempo especificado, está relacionado tanto ao desempenho quanto aos recursos usados; [13]

Manutenibilidade: é a medida de quanto o software é fácil ser alterado; [13]

Portabilidade: é a medida da facilidade do software ser portado para outro ambiente. [13]

Tendo definido as decisões e informações arquiteturais, seja performance, escalabilidade, arquitetura de referência, segurança, ou outros itens, estas devem ser armazenadas em um documento. O documento mais comum que encontramos é o DAS (Documento de Arquitetura de Software). Este documento é de grande utilidade para guiar a equipe de desenvolvimento.

SCRUM

O SCRUM é um processo iterativo e incremental para o desenvolvimento de qualquer produto e gerenciamento de qualquer projeto, a quem diga que esta mais para um framework que uma metodologia, ou até mesmo mais para atitude que um processo.

Para que o SCRUM seja utilizado com êxito, cada pessoa envolvida deve cumprir com seu papel, seguindo corretamente todos os processos, e a fim de viabilizar, utilizar como apoio algumas das ferramentas que lhe são oferecidas. Entretanto, para que seja possível seguir corretamente todos os processos, primordialmente é necessário que todos estejam de acordo com a cultura envolvida.

Segue abaixo uma breve descrição sobre os três papéis relacionados ao SCRUM:

Product Owner: responsável por garantir o retorno de investimento, este deve conhecer as necessidades do cliente;

Scrum Master: responsável por remover os impedimentos do time e garantir o uso do SCRUM;

Time (Equipe): equipe de desenvolvimento multidisciplinar e auto-gerenciável, responsável por produzir produto com qualidade e valor para o cliente.

As ferramentas como: **Product Backlog**, **Sprint Backlog**, **Burndown Chart** e **Scrum Board** são de grande utilidade durante o ciclo de vida do SCRUM. Para um melhor entendimento, segue abaixo uma breve descrição:

Product Backlog: uma lista com todos os requisitos que o Product Owner deseja, sem muitos detalhes técnicos, essa lista é ordenada por prioridade pelo Product Owner;

Sprint Backlog: contém uma lista com as tarefas decompostas sobre os itens extraídos do Product Backlog que foi definida no Sprint Planning Meeting e que deverá ser entregue ao Product Owner. Estas atividades não devem durar mais de 2 dias ou 16 horas;

BurnDown Chart: gráfico que mostra o trabalho da equipe dia a dia, avaliando assim se o Sprint está atrasado ou não. Caso o gráfico demonstre que a equipe acabará o Sprint antes, o Product Owner é consultado e novas funcionalidades são incorporadas àquele Sprint;

Scrum Board: é um quadro onde deverá contemplar todas as tarefas que serão realizadas dentro de um Sprint e listadas de acordo com as prioridades de cada item.

Existem três tipos de cerimônias no SCRUM, **Sprint Planning Meeting**, **Daily Scrum Meeting** e **Sprint Review Meeting**, estes três tipos de evento caracterizam bem o ciclo de vida de cada Sprint: início, meio e fim. Segue abaixo uma breve descrição sobre as cerimônias:

Sprint Planning Meeting: encontro para planejar o que será feito no Sprint. A equipe acessa o Product Backlog e detalha de forma mais ampla as atividades que serão executadas no Sprint de acordo com suas prioridades, avaliando tempo e complexidade. Após definido o que será feito, o Sprint Backlog é gerado;

Daily Scrum Meeting: encontros diários, com duração em média de 15 minutos, a idéia de ser uma reunião em que cada membro da equipe deve responder 3 perguntas: O que fiz ontem? O que farei hoje? O que está impedindo de que alcance o objetivo? Essa reunião é liderada pelo ScrumMaster;

Sprint Review Meeting: encontro realizado quando o Sprint chega ao fim. Este encontro é dividido em duas partes, na primeira parte é demonstrado ao ProductOwner quais atividades definidas no Product Backlog foram realizadas, o Product Owner lidera esse encontro e pode chamar todos os interessados no projeto. Após a demonstração o Product Owner e os interessados no projeto atualizam e repriorizam o

Product Backlog, definindo assim o próximo Sprint. Finalizada essa primeira parte o ScrumMaster toma a liderança e começa uma reunião com a equipe, onde a equipe avalia o que foi realizado positivamente e negativamente no Sprint, também avaliam o que poderia ser mudado para melhorar o próximo Sprint.

O **Sprint** é um conjunto de tarefas a serem executadas em um determinado tempo. A Figura 1 ilustra o ciclo de vida do SCRUM. [3][4]

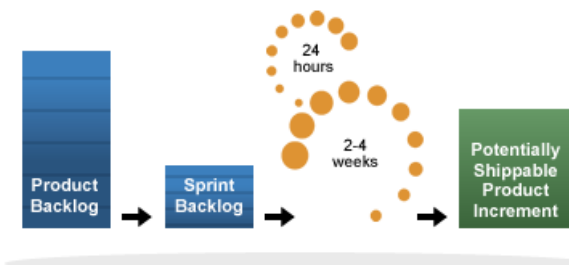


Figura 1 – Ciclo de vida do SCRUM

XP (eXtreme Programming)

Extreme Programming ou XP como é chamado é um processo de desenvolvimento de software baseado em valores de simplicidade, comunicação, feedback e coragem. O objetivo do XP é assegurar que o cliente receba o máximo de valor a cada dia de trabalho da equipe de desenvolvimento. Ele é organizado em torno de valores e práticas que atuam de forma harmônica e coesa para assegurar que o cliente sempre receba um alto retorno do investimento em software.

Os quatro valores fundamentais em que o XP se baseia são:

Feedback: fazer com que o cliente conduza o desenvolvimento diariamente a fim de garantir que a equipe direcione toda a sua atenção para aquilo que de fato irá gerar mais valor;

Comunicação: evitar o gasto de um valioso esforço na tentativa de trocar informações por meios de extensos documentos escritos que freqüentemente são interpretados de forma incorreta ou incompleta;

Simplicidade: garantir que seja desenvolvido apenas o suficiente para atender as necessidades atuais do cliente, desprezando qualquer funcionalidade não essencial;

Coragem: devido ao XP ser uma metodologia de software que se baseia em diversas premissas que contrariam os processos tradicionais de desenvolvimento de software, é preciso que todos da equipe tenham coragem para adotá-las e acreditar que, utilizando as práticas e valores do XP, serão capazes de fazer com que o software evolua com segurança e agilidade.

O XP tem alguns pontos fortes que auxiliam no processo de desenvolvimento, a citar:

Cliente Presente: a presença objetiva viabilizar a simplicidade dos processos, facilitar a comunicação com os desenvolvedores e permitir um ciclo contínuo e rápido de feedback;

Jogo do Planejamento: reunião com o cliente a cada nova release a fim de definir quais funcionalidades devem ser implementadas de acordo com suas prioridades;

Stand Up Meeting: reunir com a equipe de desenvolvimento a cada manhã para avaliar o trabalho que foi executado no dia anterior e priorizar aquilo que será implementado no dia que se inicia;

Refactoring: é utilizado para manter sempre o software o mais simples possível de ser manipulado sem que estas alterações no código possam afetar as funcionalidades que já estão implementadas;

Código Coletivo: a idéia é que o código seja comunitário a todos os desenvolvedores, permitindo assim que todos possam alterar o código quando necessário sem ter que pedir autorização de outra pessoa;

Código Padronizado: a fim de permitir que o sistema seja o mais homogêneo possível, a equipe deve estabelecer padrões de codificação, viabilizando assim a facilidade de qualquer manutenção futura;

Metáfora: técnica para transmitir idéias de formas simples, através de uma linguagem comum que é estabelecida entre a equipe e o cliente;

Ritmo Sustentável: é recomendável que os desenvolvedores trabalhem apenas 8 horas por dia a fim de garantir o máximo de rendimento e permitir a produção de software com a melhor qualidade possível;

Design Simples: optar sempre pela simplicidade do design, viabilizando a agilidade durante o desenvolvimento, dado que o feedback deve ser rápido ao cliente;

Integração Contínua: a equipe de desenvolvimento deve garantir a integração de seus códigos com o restante do sistema diversas vezes ao dia;

Releases Curtos: visa à disponibilidade de funcionalidades rapidamente ao cliente para que ele possa utilizar o software no dia-a-dia e se beneficiar dele.

Desenvolvimento Guiado pelos Testes: visa o desenvolvedor escrever testes para cada funcionalidade antes mesmos de começar a codificá-las, possibilitando eles aprofundar o entendimento das necessidades do cliente;

Abaixo a Figura 2 demonstra as práticas e os principais ciclos do XP:

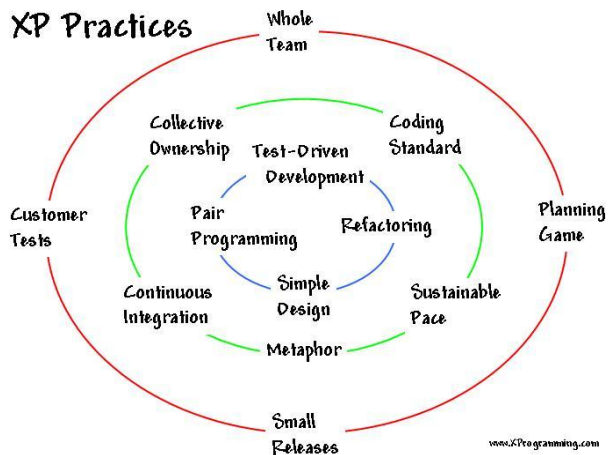


Figura 2: Práticas e princípios do XP

A idéia da utilização do XP é voltada para projetos cujos requisitos são vagos e mudam com frequência, desenvolvimento de sistema orientado a objeto, equipes pequenas e de preferência até 12 desenvolvedores. Este desenvolvimento deverá atender o modo iterativo ou incremental, objetivando que o sistema comece a ser implementado logo no início do projeto e ao longo do tempo adquirindo novas funcionalidades. [6][7]

2. Problema

O propósito deste artigo é tentar de forma abrangente mostrar como podemos lidar com situações em que a arquitetura do software torna-se ponto crucial para o sucesso no desenvolvimento de software utilizando as metodologias ágeis SCRUM e XP.

No desenvolvimento de um software, existem algumas situações importantes que precisamos estar atentos no aspecto arquitetural, geralmente são requisitos não funcionais. Esses requisitos são identificados durante a análise, de forma que sejam atendidas na medida do possível. Com o objetivo de simular uma situação em que a arquitetura do software é fundamental para o sucesso do projeto, foi criado um cenário, que, apesar de fictício não são difíceis de serem encontradas no dia a dia dos desenvolvimentos de softwares. Abaixo é descrito um possível cenário onde se aplica necessidades arquiteturais para se tratadas em desenvolvimento com metodologias ágeis SCRUM e XP.

Cenário

Durante uma reunião de planejamento da Sprint, o Product Owner informa a necessidade de algumas funcionalidades do software que ele gostaria que

fossem atendidas nessa próxima Sprint. Sobre o suporte do Scrum Master, os integrantes da equipe decompõem as tarefas de acordo com os itens de maior prioridade do Product BackLog, de forma que estes não ultrapassem dois dias para o seu término, atribuindo estes ao Sprint BackLog.

Entre essas tarefas, foram identificadas algumas situações em que requisitos não funcionais eram de grande importância para o Product Owner e para o sucesso do software:

1. O sistema deverá ser executado de qualquer computador com acesso a internet;
2. O sistema deverá possuir um mecanismo de tolerância a falhas;
3. O sistema deve sempre ter uma contingência e estar disponível 99% do tempo;
4. A interface do sistema deve ser muito fácil para os usuários, algumas requisições devem ser executadas sem o carregamento de toda a página;
5. O usuário não pode ficar esperando a resposta por mais de 7 segundos;

O cenário citado acima, seria muito crítico e difícil de ser atendidos com a utilização do SCRUM e XP, caso não tivessem sido identificados ainda na primeira iteração, mesmo com a técnica de refactoring, pois nem sempre o jeito mais simples, é o mais apropriado para algumas situações.

Na seção seguinte é proposta uma solução comumente sugerida pelas pessoas que utilizam SCRUM e XP para evitar esse tipo de problema.

3. Solução

Dado o SCRUM se tratar de uma cultura voltada para gestão de projetos, utilizado para viabilizar as necessidades reais do cliente, não permitindo que seja desenvolvido algo que não agregue valor ao cliente e o XP por sua motivação de agilidade durante o desenvolvimento do software, como citado nos pontos fortes na introdução deste artigo, todo o esforço é em desenvolver códigos simples, e apenas o que de fato são necessários para atender as necessidades identificadas pelo Product Owner a ser entregues na iteração atual.[3][6]

Por motivo da equipe de projetos ágeis ter a quantidade de recursos reduzidos e multidisciplinares, a equipe dificilmente terá um arquiteto ou um time de arquitetura trabalhando no projeto, geralmente o papel de arquiteto é desempenhado pelo desenvolvedor mais experiente da equipe.

Quando se trata de desenvolvimento de projetos ágeis usando SCRUM e XP, não existe uma iteração exclusiva para resolver as situações arquiteturais, caso

que ocasionam problemas como descritos no cenário da seção anterior. No cenário descrito, é óbvio que precisamos nos preocupar com a arquitetura, pois ela será de grande importância para atender os requisitos solicitados. Esses requisitos entre eles mesmos existem uma contradição quando diz respeito ao desenvolvimento de software, pois é muito difícil manter o **item 4 e 5**, sem nenhum *tradeoff* entre eles. Nos **itens 1, 2 e 3** é evidente a importância da componentização dos elementos, pois desta maneira irá permitir a flexibilidade em disponibilização dos componentes em servidores de aplicação distintos permitindo aplicabilidade de *load balance* e *failover*. Segundo Robert Nord e James Tomayko, devemos realizar a primeira iteração que tratará somente das definições das situações arquiteturais, essa iteração é chamada por muitos de iteração zero. [11] Essa prática é muito debatida pelos agilistas extremos, pois como já foi explicado o objetivo do SCRUM e do XP é entregar algo que agregue valor ao cliente, e para o cliente a arquitetura não é importante, ela só importa para a equipe de desenvolvimento.

Devido aos estudos e situações encontradas no dia a dia, onde se deve dar uma melhor atenção para arquitetura do software quando complexa, acreditamos que uma boa solução para esse problema, tanto para agradar os agilistas extremos, como manter a arquitetura do software, é manter um equilíbrio no desenvolvimento para cada **Sprint**, fazendo com que de fato seja dada uma maior atenção para arquitetura na primeira iteração e na medida em que novas iterações vão surgindo, vá diminuindo o tempo gasto na arquitetura e dedicando ao máximo aos requisitos funcionais, lembrando que a documentação e solução das situações da arquitetura de software sejam realizadas de maneira mais flexível possível para não gerar um tempo de esforço demasiado.

4. Métricas

Esses dois termos merecem atenção: desenvolvimento tradicional e desenvolvimento ágil.

O termo citado como desenvolvimento tradicional enfatiza os projetos de software que se baseiam tipicamente no desenvolvimento em cascata, este que embora seja antigo ainda é muito utilizado para o desenvolvimento de software. O desenvolvimento tradicional é eminentemente seqüencial, onde cada fase se utiliza do resultado da fase anterior, conforme ilustrado na Figura 3.

O desenvolvimento ágil, por sua vez, faz referência ao desenvolvimento iterativo, em espiral, conforme ilustrado na Figura 4.



Figura 3: Desenvolvimento tradicional (cascata).



Figura 4: Desenvolvimento iterativo (em espiral).

No desenvolvimento tradicional existem fases apropriadas para decidir e adequar uma melhor arquitetura para o projeto, isto ocorre desde a análise até a implementação, em contra partida, métodos desse tipo podem influenciar sobre o quanto de funcionalidades disponibilizadas para os usuários de fato serão utilizadas, conforme ilustrado na Figura 5.

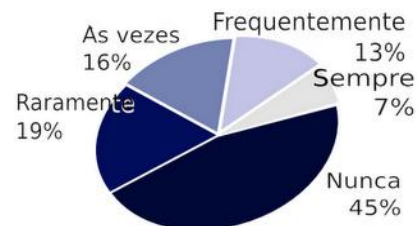


Figura 5: Utilização de Funcionalidade

A criação de funcionalidades não utilizadas pelos usuários gera custos desnecessários. Outra desvantagem quanto à utilização do desenvolvimento tradicional está relacionada ao custo de uma alteração

no projeto de acordo com suas fases, conforme ilustrado na Figura 6.

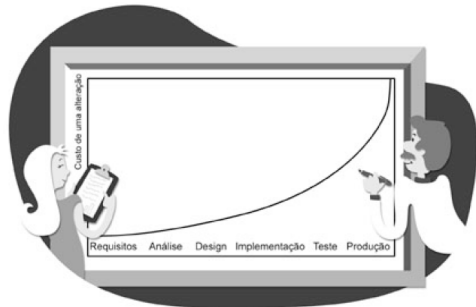


Figura 6: Custos de alterações durante o projeto em desenvolvimento tradicional.

No desenvolvimento ágil, apesar de obter vantagens sobre o desenvolvimento tradicional pertinente a funcionalidades disponibilizadas para os usuários, assim como os custos envolvidos, é necessário adaptá-lo para atender os requisitos arquiteturais de grande complexidade. [6]

5. Conclusão

É evidente que qualquer uma das metodologias, seja a tradicional ou a metodologia ágil, sempre nos deparamos com algumas vantagens e desvantagens para na sua escolha. A vantagem que propicia na escolha de uma metodologia ágil, esta no fato dele ser mais flexível quanto a sua adaptação, como foi evidenciada na solução proposta nesse artigo. Entretanto devemos ser cautelosos quanto à adaptação, pois este poderá descaracterizar a metodologia ágil, permitindo facilmente perturbar a eficiência da metodologia utilizada.

O que foi proposto nesse artigo foi exatamente mostrar de uma maneira clara que mesmo com metodologias ágeis é possível atender requisitos arquiteturais de grande complexidade, assim como documentar estes, sem descaracterizar a cultura do SCRUM e as práticas oferecidas pelo XP (eXtreme Programming).

6. Referência

[1] The Importance of Software Architecture, 2003 por Carnegie Mellon University. <http://sunset.usc.edu/GSAW/gsaw2003/s13/northrop.pdf> Acessado em 30 de Outubro de 2009 16:00

[2] Bachmann, Felix; Bass, Len; Carriere, Jeromy; Clements, Paul; Garlan, David; Ivers, James; Nord, Robert; Little, Reed; Software Architecture Documentation in Practice:

Documenting Architectural Layers, 2000 SPECIAL REPORT CMU/SEI-2000-SR-004

[3] Scrum Alliance http://www.scrumalliance.org/pages/what_is_scrum Acessado em 30 de Outubro de 2009 17:00

[4] ImproveIT <http://improveit.com.br/scrum> Acessado em 30 de Outubro de 2009 17:45

[5] Bass, Len; Clements, Paul; Kazman, Rick; Software Architecture in Proctice, Second Edition, 2003

[6] Teles, Vinicius; Extreme Programming – Aprenda como encantrar seus usuários desenvolvendo software com agilidade e alta qualidade, 2006

[7] Beck, Kent; Extreme Programming Explained, 2004

[8] Eden H., Amnon; Kazman, Rick; Architecture, Design, Implementation <http://www.eden-study.org/articles/2003/icse03.pdf>

[9] Hohmann, Luke; Beyond Software Architecture: creating ans sustaining winning solutions, 2003

[10] Resources for Software Architects <http://www.bredemeyer.com/definiti.htm> Acessado em 26/11/2009 as 21:00

[11] Nord, Robert; Tomayko, James; Software Architecture-Centric Methods and Agile Development, 2006

[12] Atributos de Qualidade <http://cnx.org/content/m17527/latest/> Acessado em 27 de Novembro de 2009 01:30

[13] ISO 9126 - <http://www.sqa.net/iso9126.html> Acessado em 29 de Novembro de 2009 23:00

