



UNIVERSIDADE FEDERAL DE SANTA CATARINA
SISTEMAS DE INFORMACAO
INE 5632 – PROJETOS II

**APLICAÇÃO DA METODOLOGIA U-SCRUM PARA MELHORIA DE
PROCESSOS EM UM SISTEMA DE GESTÃO ACADÊMICA E
ADMINISTRATIVA.**

Marisete Martello

Gabriela Dal Berto

Orientadora: Patricia Della Mea Plentz

Florianópolis - SC

2010/1

MARISETE MARTELLO
GABRIELA DAL BERTO

**APLICAÇÃO DA METODOLOGIA U-SCRUM PARA MELHORIA DE
PROCESSOS EM UM SISTEMA DE GESTÃO ACADÊMICA E
ADMINISTRATIVA.**

Trabalho de Conclusão de Curso desenvolvido como requisito parcial para obtenção do título de Bacharel do Curso de Graduação em Sistemas de Informação.

Orientadora: Patrícia Della Mea Plentz

Florianópolis - SC

2010/1

BANCA EXAMINADORA

Profa. Dra. Patrícia Della Mea Plentz

Prof. Dr. João Candido Dovicchi

Profa. Dra. Marta Maria Leite

DEDICATÓRIA

À DEUS que está sempre presente em todos os momentos de nossas vidas.

Aos nossos pais e irmãos, pela dedicação, amor e confiança depositados em nossa caminhada.

Aos nossos amigos pela paciência, apoio e auxílio em mais esta etapa de nossas vidas.

AGRADECIMENTOS

À nossa orientadora Patrícia Della Mea Plentz por todo apoio, incentivo e paciência dados no decorrer deste trabalho, possibilitando o desenvolvimento do mesmo da melhor maneira possível.

Aos professores do Departamento de Sistemas de Informação pela formação acadêmica proporcionada.

Aos amigos, àqueles poucos, porém verdadeiros. Como diz a frase, amigos são a família que podemos escolher.

Agradecemos especialmente aos amigos que fizemos durante o curso, onde eu 'Marisete' ressalto o Leonard Chucre de Moraes, que esteve comigo e me apoiou em momentos importantes, e eu 'Gabriela' agradeço à todos que, de alguma maneira, contribuíram para minha formação e caminhada até aqui.

Agradecemos, enfim, a todos aqueles que estiveram ao nosso lado neste período, oferecendo apoio, amizade, confiança e que direta ou indiretamente contribuíram para esse momento.

"Acrescentemos alguma coisa a uma qualidade, e ela parecerá defeito. Tiremos alguma coisa de um defeito, e ele parecerá qualidade."

Émile Faguet

RESUMO

Para que a qualidade seja controlada e avaliada, são necessários métodos. Estes métodos atualmente compõem a área de controle da qualidade, definida como organização ou sistemática de todos os procedimentos necessários à confirmação de que o módulo ou produto está de acordo com as necessidades estabelecidas em sua fase de planejamento.

Neste contexto, o objetivo deste trabalho concentra-se em propor melhorias na qualidade de produtos de software, utilizando técnicas de usabilidade e demonstrando por meio de experiência prática, que a incorporação dessas técnicas em um ambiente de desenvolvimento ágil é factível, produz bons resultados e permite o ganho de qualidade e produtividade na confecção de software.

Palavras Chave: Qualidade, Usabilidade, Desenvolvimento ágil.

ABSTRACT

For quality control and evaluation, methods are needed. Nowadays those methods compose the quality control area, defined as the organization or the group of all the necessary procedures to confirm that the product was made according to the needs defined in it's planning phase.

In this context, this work's objective is to offer an improvement in software products, by using usability techniques and demonstrating with practical experiences that those techniques applied in an agile development environment allow us to produce excellent results in software quality and productivity.

Keywords: Quality, Usability, Agile development.

LISTA DE SIGLAS E ABREVIATURAS

AUCD: *Agile Usage-Centered Design*

CRUISER: *Cross Discipline User Interface and Software Engineering Lifecycle.*

Inmetro: Instituto Nacional de Metrologia, Normalização e Qualidade Industrial

ICP: *Inicial Conceptual Phase* (Fase Inicial Conceitual)

CTP: *Construction and Test Phase* (construção e teste do sistema)

ES: Engenharia de Software

IHC: Interação Humano Computador

IRUP: *Initial Requirements Up-Front*

MSN: *Microsoft Service Network* (Software de troca de mensagens instantâneas pela internet)

UI: *User Interface*

XP: *Extreme Programming* (Programação Extrema)

XPU: *Extreme Programming Usability* (Programação Extrema Usabilidade)

LISTA DE FIGURAS

Figura 1: Processo de projeto de UI.....	26
Figura 2: Projeto, testes e implementação proposta por Mayhew	29
Figura 3: ISO 13407.....	32
Figura 4: Sequência de reuniões proposta pelo Scrum.....	40
Figura 5: Ciclo proposto pelo Scrum	43
Figura 6: Fase inicial conceitual (ICP)	52
Figura 7: Fase de construção e testes (CTP).....	53
Figura 8: Estrutura do processo XPU	56
Figura 9: Papéis e artefatos da interação em SCRUM tradicional (Fonte: U-SCRUM).....	58
Figura 10: Papéis e artefatos da interação em U-SCRUM tradicional (Fonte: U-SCRUM)	59
Figura 11: Modelo conceitual de mais alto nível do Trevio	64
Figura 12: Product Backlog.....	68
Figura 13: Wireframe da tela de Lembrete	70
Figura 14: Wireframe da tela de Registro de Cobrança	71
Figura 15: Sprint Backlog.....	72
Figura 16: Aba Interna - Tela de Registro de Cobrança.....	73
Figura 17: <i>Wireframe</i> com alteração da tela Tipo de Entrega	75
Figura 18: Wireframe tela Controle de Entrega e Re-entrega	76
Figura 19: Wireframe tela Controle de Entrega Diária	77
Figura 20: Wireframe tela Controle de Entrega – Várias.....	78
Figura 21: Wireframe tela Controle de Entrega - Várias.....	79

LISTA DE TABELAS

Tabela 1: Contribuição das disciplinas para levantamento dos requisitos iniciais em CRUISER .51

SUMÁRIO

INTRODUÇÃO.....	14
TEMA.....	16
1. Delimitação do tema.....	16
2. Objetivos.....	17
2.1. Objetivo geral.....	17
2.2. Objetivos específicos.....	17
3. Motivação.....	17
4. Objeto.....	18
4.1. Problema.....	18
4.2. Metodologia.....	18
FUNDAMENTAÇÃO.....	19
1. Qualidade e produtividade.....	19
1.1. Qualidade e satisfação.....	20
1.2. Necessidades e requisitos.....	21
2. Engenharia de usabilidade.....	23
2.1. Processo de projeto de interface com o usuário.....	25
3. Desenvolvimento ágil de software.....	36
3.1. Princípios do manifesto ágil.....	37
3.2. Algumas metodologias ágeis.....	38
4. Usabilidade X Desenvolvimento ágil.....	47
4.1. Agile Usage-Centered Design (AUCD).....	48
4.2. CRUISER.....	50
4.3. XPU – Um modelo para o desenvolvimento de sistemas centrado no usuário	53
4.4. U-SCRUM: Uma metodologia ágil para promoção da usabilidade.....	57
4.5. Experiência prática de integração de usabilidade em um ambiente de desenvolvimento ágil.....	61
ESTUDO DE CASO.....	66

1. Gennera: missão e visão da empresa.....	66
2. Sistema ASP Gestão Educacional Online.....	66
3. Problema a ser estudado.....	67
CONCLUSÃO	84
SUGESTÕES DE TRABALHOS FUTUROS	87
REFERÊNCIAS BIBLIOGRÁFICAS.....	88

INTRODUÇÃO

A Qualidade¹ é hoje um dos maiores motivadores em qualquer área de trabalho, afinal, todos querem receber e também oferecer produtos e/ou serviços de qualidade.

Desta forma, é possível dizer que a qualidade de um produto ou serviço pode ser medida com base no grau de satisfação dos usuários/clientes em relação a sua utilização. No entanto, esta definição é um tanto subjetiva, se partirmos do princípio que um determinado produto ou serviço, para algumas pessoas possui qualidade, enquanto que para outras não, ou seja, para algumas pessoas o produto é ótimo, e, portanto, tem maior qualidade, enquanto que para outras pessoas, o produto é ruim, e em consequência, tem menor qualidade.

Mas então, como definir o que é Qualidade?

O dicionário da Língua Portuguesa Priberam [\[1\]](#) define qualidade como: “a maneira de ser boa ou má de alguma coisa”. Já a Norma ISO 8402 [\[2\]](#) define qualidade como “a totalidade de características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas”.

Aplicando esses conceitos à produção de software, é possível definir a qualidade de um sistema computacional como a capacidade de suprir necessidades dos usuários/clientes da maneira mais satisfatória.

Existem inúmeras maneiras de promover a qualidade em softwares, entretanto, para o escopo desse trabalho, foi escolhida a usabilidade como fator importante e determinante na melhora da qualidade de sistemas computacionais.

Em paralelo à busca pela satisfação dos clientes, que implica diretamente na boa qualidade do sistema, as empresas com o intuito de

¹ Boa qualidade do sistema. Conjunto de características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas (Norma ISO 8402 [2]).

potencializar seus ganhos e resultados, preocupam-se também com a redução dos custos e prazos de entrega.

À primeira vista, todavia, parece existir uma contradição, pois como é possível aumentar a qualidade dos produtos sem aumentar os investimentos e recursos ou o prazo de entrega? Esta questão foi o agente motivador deste trabalho, pois com a aplicação de grandes investimentos o ganho de qualidade pode ser obtido facilmente, porém, o desafio é conseguir isso apenas com a melhoria dos processos de desenvolvimento de software e com o mínimo de investimento possível.

Assim, foi utilizado como base para a construção desse trabalho a pesquisa sobre qualidade de software focada em usabilidade. A partir disso, buscou-se métodos de desenvolvimento de software capazes de promover a redução dos custos e prazos sem implicar em prejuízos no produto final. Dessa forma, chegou-se às metodologias ágeis que, desde seu surgimento, vêm ganhando cada vez mais o mercado de empresas de sucesso devido à qualidade dos seus resultados. Por fim, foram encontradas algumas metodologias já propostas sobre integração de usabilidade aos processos de desenvolvimento ágil e com base em uma dessas metodologias efetuou-se o desenvolvimento do estudo de caso que permitiu a aplicabilidade prática do conhecimento obtido durante as pesquisas.

TEMA

O desenvolvimento ágil de projetos vem ganhando cada vez mais espaço dentro das empresas de software que buscam construir projetos com qualidade no menor tempo possível [26].

A possibilidade de agregar aos processos de desenvolvimento de projetos a abordagem metodológica da engenharia de usabilidade é uma proposta que pretende potencializar a qualidade dos projetos de software. Essa visão aparentemente causa uma impressão de redução da agilidade no desenvolvimento dos projetos, já que a Engenharia de Usabilidade propõe uma série de práticas que muitas vezes são encaradas como supérfluas perante as demais necessidades do projeto [27].

Atualmente o *Scrum*² ocupa uma posição significativa dentre às metodologias de gerenciamento de software, já que sua abordagem propõe agilidade na gestão e planejamento de projetos, isso pode ser notado pela enorme quantidade de empresas que utilizam essa metodologia para gerenciamento de seus projetos.

Então, a proposta deste trabalho é incluir nos ciclos (*Sprints*) do *Scrum* conceitos e técnicas de usabilidade, visando melhorias na qualidade do sistema sem implicar em aumento de prazos e custos e, também, sem prejudicar sua abordagem chave.

1. Delimitação do tema

Este trabalho se limitará à pesquisa de técnicas ágeis de desenvolvimento de projetos, com foco principal na metodologia *Scrum*, bem como nos conceitos e metodologias de qualidade e engenharia de usabilidade.

² *Framework* utilizado para gerenciamento de projetos.

2. Objetivos

2.1. Objetivo geral

- Proporcionar melhorias na qualidade do software por meio da implantação de práticas da engenharia de usabilidade junto aos ciclos do *Scrum*.

2.2. Objetivos específicos

- Melhorar a qualidade do sistema;
- Atingir a harmonia na utilização da engenharia de usabilidade juntamente com o desenvolvimento ágil de projetos;
- Potencializar os ganhos em qualidade e produtividade.

3. Motivação

A Engenharia de Usabilidade, assim como o desenvolvimento ágil de projetos de software, faz parte do desenvolvimento de softwares no mundo de hoje. A união destas duas abordagens metodológicas, no entanto, é uma proposta mais ousada e pouco utilizada de maneira geral.

Este trabalho pretende despertar a visão para o desenvolvimento de softwares com qualidade e agilidade por meio do uso de uma metodologia unificada. Isso deve contribuir de forma significativa com a melhora dos processos de desenvolvimento de projetos de software, pois consegue agregar os atributos qualidade e agilidade que muitas vezes só ocorrem em contextos antagônicos.

4. Objeto

4.1. Problema

Tornar as práticas da Engenharia de Usabilidade mais ágeis ao ponto de agregá-las ao processo de desenvolvimento, neste caso, do sistema ASP Gestão Educacional Online³, permitindo a melhora da qualidade do software, sem prejudicar os prazos.

4.2. Metodologia

Este trabalho foi desenvolvido por meio de pesquisa em bibliografias relacionada a assuntos pertinentes ao tema, tais como: qualidade, engenharia de usabilidade e desenvolvimento ágil de software. Utilizou-se também entrevistas feitas à pessoas especialistas nas áreas pesquisadas.

³ O ASP Gestão Educacional Online é um sistema de gestão acadêmica e financeira, utilizado neste trabalho como estudo de caso.

FUNDAMENTAÇÃO

1. Qualidade e produtividade

Segundo Alfredo Lobo, ex-diretor de Qualidade do Inmetro [3], qualidade e produtividade são fatores chave para a competitividade, e em maior ou menor escala, sempre foram a grande preocupação dos setores produtivos.

Ainda segundo Alfredo Lobo [3], a qualidade ao longo do tempo observou diferentes abordagens, porém é até hoje fator chave de sucesso para as empresas. Com o acirramento da competição em função da globalização da economia, a adequada abordagem no trato da qualidade passou a ser uma questão de sobrevivência para as empresas. Ao longo do tempo, desde a fase de produção artesanal até os dias atuais, a qualidade teve ao menos, três diferentes abordagens, sendo:

- **Produção Artesanal:** Esta fase caracterizou-se pela aproximação entre produtor e consumidor. A interação plena entre os mesmos propiciava ao consumidor passar suas necessidades e expectativas diretamente ao produtor, e até hoje, os produtos produzidos neste período são conhecidos como “melhores” ou de “mais qualidade”, comparados aos produtos atualmente produzidos.
- **Revolução Industrial:** Esta fase provocou grandes mudanças em termos de abordagem da qualidade. O aumento da produção introduziu o chamado controle da qualidade, cujo foco inicial era a inspeção final do produto, todavia, este processo de controle da qualidade ao longo do tempo sofreu uma série de aperfeiçoamentos, a exemplo: Inspeção em diferentes etapas do processo produtivo e Controle estatístico da qualidade, com ênfase na detecção de defeitos. Este abordagem do processo acabou provocando o distanciamento entre produtor e consumidor, e a segmentação do controle da qualidade, como consequência da produção em série, diluiu a responsabilidade pela qualidade e problemas com qualidade dos produtos surgiram com maior intensidade.

- Exploração espacial: Os programas nucleares e mais recentemente a exploração de petróleo em águas profundas, provocaram uma nova e importante mudança na abordagem da questão da qualidade nas empresas, uma vez que se verificou que grande parte dos problemas relacionados à qualidade tinha origem em falhas gerenciais e não técnicas. Essa constatação acabou por originar os atualmente conhecidos sistemas de gestão da qualidade, porém, agora associando programas de detecção de defeitos a ações de administração da qualidade, que enfatizam, por sua vez, a prevenção de defeitos.

Qualificar fornecedores, elaborar procedimentos de execução e inspeção, treinar e qualificar os recursos humanos da empresa, melhorar a análise dos projetos, identificar expectativas e avaliar o grau de satisfação dos clientes, dentre outras, são ações que por si só incluem a prevenção de defeitos e também a administração ou gestão da qualidade [3].

Com a globalização dos mercados e economias, novas abordagens em termos de qualidade estão surgindo e são necessárias, uma vez que, a gestão de qualidade adequada é instrumento decisivo para alavancar a competitividade e com isso, garantir a sobrevivência das empresas, nos ambientes atuais de grande competição [3].

1.1. Qualidade e satisfação

Segundo Pressman (1994) [4], "Qualidade de software é a conformidade de requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais".

Neste ponto, pode-se dizer então que o termo qualidade está associado à conformidade com as especificações e a visão de satisfação do cliente, considerando ainda, fatores como prazos, pontualidade e flexibilidade, que também são importantes e devem ser considerados na avaliação de qualidade do produto.

Entretanto, é sabido que desenvolver software com qualidade tem sido o grande desafio do mercado atual, uma vez que, a obtenção da mesma depende de bons resultados não somente no desenvolvimento do produto, mas em todo o ciclo de vida do projeto, que envolve cinco grupos de processo: “Iniciação, Planejamento, Execução, Monitoramento/Controle e Encerramento” [5].

Segundo Lélío Varella (PMP) [5], “é preciso considerar também que o conceito de qualidade em projetos abrange não apenas a qualidade do produto ou serviço entregue pelo projeto, como também a qualidade do próprio projeto, como um empreendimento, e suas atividades gerenciais”.

Ainda segundo Lélío Varella (PMP) [5], “o primeiro passo no caminho da qualidade é um bom entendimento do problema, ou oportunidade central do projeto. E daí a compreensão das necessidades a serem atendidas e o estabelecimento dos requisitos da qualidade”.

1.2. Necessidades e requisitos

Kano et al [6] classificam os requisitos da seguinte forma: "atendimento ao requisito" e "sentimento de satisfação". Com base nisto, os requisitos de qualidade seriam classificados da seguinte maneira:

- Requisitos necessários: São requisitos que se atendidos passam despercebidos, porém, se não atendidos geram insatisfação do cliente.
- Requisitos normais: São requisitos que geram insatisfação do cliente se não forem atendidos, mas geram satisfação se forem atendidos.
- Requisitos atrativos: São requisitos que geram satisfação do cliente se forem atendidos, mas se não forem atendidos passam despercebidos.

As necessidades de um produto precisam ser entendidas e explicitadas, assim como os requisitos de qualidade deste produto devem ser formulados, especificados, documentados e incorporados ao projeto. Não se pode garantir a qualidade de um produto, mesmo que o mesmo esteja de acordo com sua

especificação ou que cumpra, em princípio, os objetivos esperados pelo cliente, uma vez que, muitos dos requisitos implícitos podem não ter sido tratados. Desta forma, o esforço do planejamento deve ser efetuado de forma a identificar e tornar explícito o maior número possível de requisitos.

Neste contexto, surge o conceito de usabilidade, o qual está associado a diversas características, que se implementadas, contribuirão para a qualidade do produto.

2. Engenharia de usabilidade

De acordo com a proposta deste trabalho, que consiste em promover a qualidade nos produtos de software por meio da incorporação de técnicas de usabilidade nos processos de desenvolvimento de software, este capítulo trata do conceito de usabilidade, bem como da importância de sua utilização na confecção de softwares em geral. São apresentadas também algumas técnicas de desenvolvimento de interface com o usuário visando a usabilidade e ainda algumas recomendações propostas pela ISO 13407 e ISO 18529.

A engenharia de usabilidade se ocupa do desenvolvimento da interface com o usuário que é um componente do sistema interativo. Este componente é formado por apresentações e estruturas de diálogo que lhe conferem um comportamento em função das entradas dos usuários.

A interface com o usuário apresenta painéis com informações, dados, controles, comandos e mensagens, e é por meio dessas apresentações que a interface solicita e recebe as entradas de dados, de controles e de comandos dos usuários. Por fim, ela controla o diálogo, interligando as entradas dos usuários com as apresentações de novos painéis. Uma interface é definida segundo uma *lógica de operação* que visa que o sistema seja agradável, intuitivo, eficiente e fácil de operar [7].

O projeto cuidadoso de interface com o usuário é uma parte essencial de todo o processo de projeto de software. Se um sistema de software deve atingir todo o seu potencial, é essencial que sua interface com o usuário seja projetada para combinar as habilidades, experiências e expectativas dos usuários previstos. Um bom projeto de interface com o usuário é crítico para a confiabilidade do sistema. Muitos dos chamados “erros de usuário” são causados pelo fato de que as interfaces de usuário não consideram as capacidades dos usuários reais e seu ambiente de trabalho. Uma interface com o usuário projetada inadequadamente significa que os usuários serão, provavelmente, incapazes de acessar algumas das características do sistema, cometerão erros e sentirão que o sistema os limitará em vez de ajudá-los a atingir o objetivo para o qual o sistema é utilizado [8].

Para apoiar a construção de uma interface capaz de implementar satisfatoriamente a interação com os usuários, alguns princípios devem ser considerados:

- Princípio da *Familiaridade do Usuário* – a interface deve ser projetada fazendo uso de termos e conceitos pertinentes ao ambiente de trabalho dos usuários, ou seja, eles devem sentir-se confortáveis e familiarizados na interação com o sistema [8].
- Princípio da *Consistência* – a consistência da interface diz respeito ao comportamento da mesma, que deve manter uma equivalência quando as operações são semelhantes; isso acelera o processo de aprendizado do usuário, pois permite o estabelecimento de um padrão de utilização [8].
- Princípio da *Surpresa Mínima* – seguindo o raciocínio do princípio da consistência, o princípio da surpresa mínima propõe que um usuário nunca deve ser surpreendido por um comportamento inesperado do sistema, isso pode causar irritação, desconforto e confusão no usuário [8].
- Princípio da *Facilidade de Recuperação* – esse princípio está relacionado à capacidade de permitir ao usuário a reparação de erros cometidos por ele. O projeto de interface com o usuário pode facilitar a recuperação de erros por meio de três mecanismos:
 - Confirmação de ações destrutivas: o sistema solicita confirmação do usuário antes de executar uma ação de caráter destrutivo.
 - Disponibilidade de um recurso do tipo refazer: permite recuperar o estado anterior ao atual.
 - *Checkpointing*: mecanismo capaz de gravar estados do sistema em determinados períodos, permitindo o retorno à estados anteriores [8].
- Princípio *Guia do Usuário* – esse princípio diz respeito à disponibilidade de recursos de ajuda para o usuário, deve ser

integrado ao sistema a fim de facilitar o acesso do usuário em qualquer situação [8].

- Princípio *Diversidade de usuário* – esse princípio prevê que um sistema interativo pode possuir diversos tipos de usuários e, portanto, a interface deve ser projetada de maneira a atender satisfatoriamente todos os tipos de usuários do sistema [8].

A engenharia de usabilidade, portanto, torna-se uma peça chave e, em alguns casos, essencial no processo de desenvolvimento de software. Isso porque pode-se dizer que a interface com o usuário é um fator determinante na produtividade e no conforto do usuário.

O que é muito comum em projetos de desenvolvimento de software é a construção das interfaces com o usuário à cargo dos próprios programadores ou designers que focam apenas na funcionalidade ou na estética do sistema, sem preocupar-se com a ergonomia e usabilidade do software. Isso ocorre porque incorporar um processo de engenharia de usabilidade no ciclo de desenvolvimento de um software exige recursos especializados, envolvimento muito próximo com os usuários e ferramentas especializadas, esses fatores conseqüentemente demandam mais tempo e dinheiro para o projeto [7].

2.1. Processo de projeto de interface com o usuário

Idealmente o projeto de UI (*User Interface* ou interface com o usuário), deve ser construído de forma iterativa e incremental onde o usuário está presente durante todo o ciclo [8]. O paradigma de desenvolvimento de uma interface com o utilizador deve permitir a realização de sucessivos ciclos de análise, concepção e testes, com a necessária retro-alimentação dos resultados dos testes, de um ciclo a outro. A estratégia consiste em, a cada ciclo, identificar e refinar continuamente o conhecimento sobre o contexto de uso do sistema e as exigências em termos de usabilidade da interface. Na sequência dos ciclos se constroem versões intermediárias da interface do sistema que são submetidas a testes de uso, em que os representantes dos utilizadores simulam a realização de suas tarefas. Inicialmente eles participarão

em simulações "grosseiras", usando maquetes⁴, mas, com o avanço do desenvolvimento, eles recorrerão a protótipos e versões acabadas do sistema, em simulações mais e mais fidedignas. O objetivo é avaliar a qualidade das interações e levar em conta os resultados dessas avaliações para a construção de novas versões das interfaces. Se implementada desde cedo no desenvolvimento, tal estratégia pode reduzir o risco de falhas conceituais do projeto, garantindo que, a cada ciclo, o sistema responda cada vez melhor às expectativas e necessidades dos usuários em suas tarefas [7]. O ciclo de desenvolvimento do projeto de UI pode ser observado na figura 1 [8], a seguir.

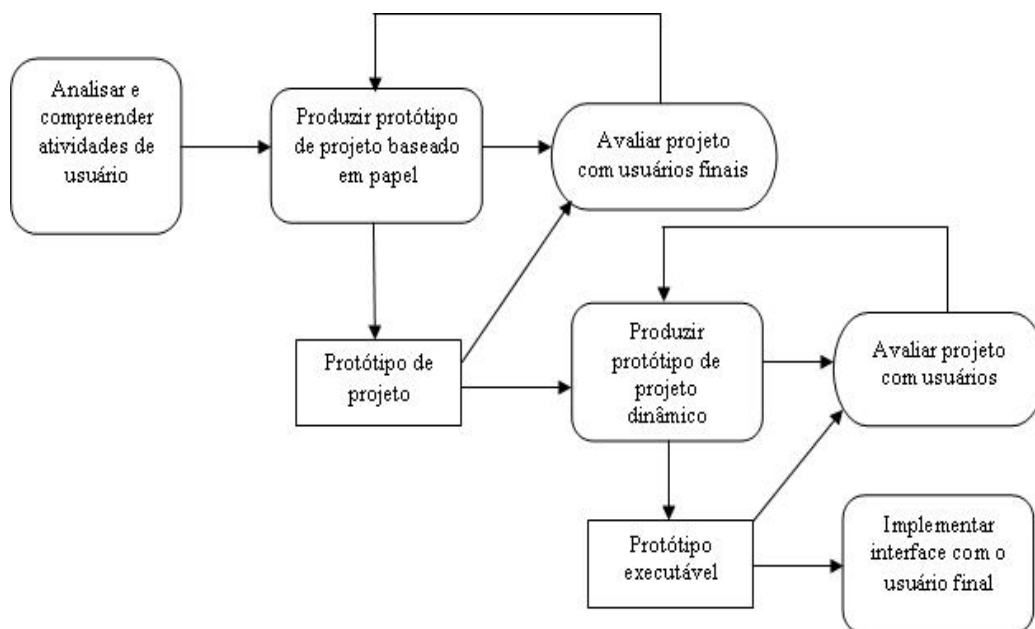


Figura 1: Processo de projeto de UI

São três os pontos principais que podem ser observados na ilustração anterior:

- Análise de usuário – a análise consiste em observar e capturar o máximo de informações sobre o usuário e suas interações com o ambiente de trabalho. Isso será útil para definir alguns padrões essenciais para o usuário os quais a interface deverá atender [8].

⁴ Modelo reduzido de um cenário [1].

- Prototipação⁵ de sistema – a prototipação tem como principal objetivo, guiar o usuário na definição do que é mais adequado na sua interação com o sistema, ou seja, o fato de visualizar e interagir com protótipos permite ao usuário identificar suas necessidades com maior clareza e objetividade por meio de experiências práticas e palpáveis [8].
- Avaliação de interface – a avaliação diz respeito aos feedbacks que o usuário pode dar perante a experiência do usuário com a interface [8].

2.1.1.Ciclo da engenharia de usabilidade

Seguindo o conceito de processo iterativo nos projetos de UI, Deborah Mayhew [9] propõe um modelo de ciclo de vida da engenharia de usabilidade. A proposta segue a mesma estrutura da norma ISO 13407, que consiste em atividades de análise, projeto e testes em cada ciclo do projeto de UI.

2.1.2.Análise de requisitos

Segundo Deborah Mayhew [9], a fase de análise de requisitos é composta por quatro atividades que resultarão na definição dos requisitos de usabilidade para o sistema. As atividades são:

- Análise do perfil do usuário – para todos os usuários do sistema, é necessário que os projetistas compreendam suas qualidades pessoais e habilidades no que diz respeito à tarefas e processos de trabalho dentro da organização.
- Análise do contexto da tarefa – para todas as tarefas implementadas pelo sistema, os projetistas devem tomar conhecimento de aspectos como: estrutura, duração, custos, objetivos, etc. de cada tarefa individualmente.

⁵ Protótipo: Modelo; exemplar; padrão [1].

- Análise das possibilidades e restrições da plataforma – as possibilidades e restrições neste momento referem-se aos aspectos físicos de ambiente e equipamentos que fazem parte do desenvolvimento do projeto.
- Análise de princípios gerais para o projeto – esta análise envolve o contexto de uso do sistema, ou seja, envolve a análise dos aspectos que são pertinentes à sua utilização, como: usuários, tarefas, equipamentos e ambiente.

Após as quatro fases de análise, parte-se para a construção de especificações da interface com base nas informações recolhidas até o momento. A partir disso, a autora sugere que todo o material produzido seja registrado em um documento oficial, denominado Guia de Estilo do Projeto. Este documento deve ser alimentado durante todo o ciclo de desenvolvimento por todas as fases executadas e deve também ficar disponível para acesso de todos os envolvidos no projeto.

2.1.3. Projeto, testes e implementação

A fase de projeto, testes e implementação, segundo Mayhew [\[9\]](#), deve repetir-se a cada ciclo do desenvolvimento da interface, passando por três níveis de implementação, conforme ilustra a figura 2 a seguir.

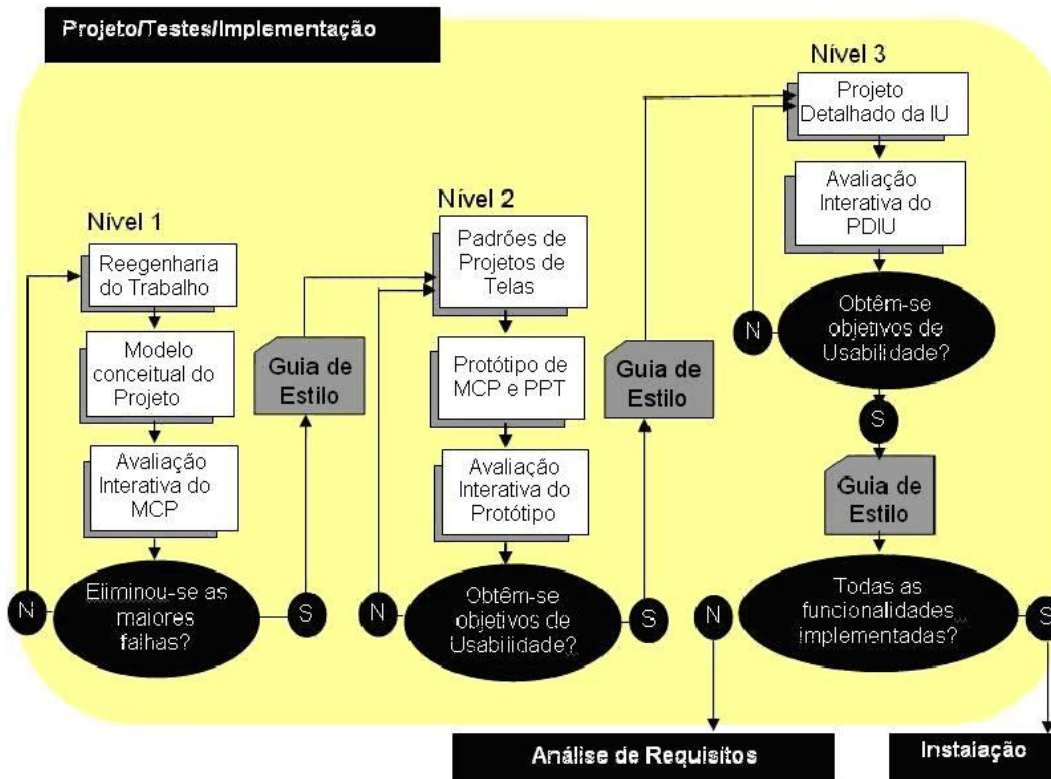


Figura 2: Projeto, testes e implementação proposta por Mayhew

2.1.3.1. Nível 1

O nível 1 é composto de três fases:

- Reengenharia do trabalho – com base nas especificações construídas na etapa de análise, os projetistas devem reformular as tarefas de interação entre homem e máquina visando eliminar as falhas no contexto da usabilidade.
- Modelo conceitual da interface – o modelo conceitual é definido pelos projetistas e contém as principais telas do sistema representadas por meio de fluxogramas. Estes fluxogramas são construídos com o objetivo de definir quais serão as principais telas, seus principais componentes e o fluxo de navegação entre as mesmas.
- Avaliação interativa do MCP – a avaliação do modelo conceitual é realizada com a participação de usuários e projetistas. De um

lado os usuários simulam uma interação com as maquetes do sistema feitas em papel e de outro lado os projetistas observam as dificuldades e limitações do usuário e propõem novas telas à medida que sejam necessárias.

O nível 1 poderá ser avançado somente quando as falhas maiores forem eliminadas neste contexto inicial de maquetagem e testes. Antes de seguir para o nível 2, todos os resultados obtidos no nível 1 devem ser registrados no documento Guia de estilo da interface.

2.1.3.2. Nível 2

Assim como no nível 1, o nível 2 também é composto de três fases:

- Padrões de projetos de telas – a definição de padrões resultará em um documento composto pelas regras no que diz respeito à escolha de controles, formato e localização de janelas, terminologia a ser utilizada, tipos de fontes e cores, etc. Este documento deve ser construído com base na análise de requisitos e na definição do modelo conceitual, uma vez que servirá de guia para a construção de interfaces consistentes e padronizadas.
- Protótipo de MCP e PPT – com base no modelo conceitual definido no nível 1 e no padrão de projeto de tela, é possível construir protótipos executáveis que simulem as funcionalidades do sistema, dessa maneira os usuários poderão interagir com os protótipos.
- Avaliação interativa do protótipo – os testes neste nível permitem avaliações mais realistas do sistema, já que podem ser realizados em protótipos. Os usuários conseguem simular sua interação com o sistema, o que permite a observação de medidas como facilidade de aprendizado, eficácia e taxa de erros durante a execução das tarefas.

Caso o nível 2 atinja os objetivos de usabilidade, é possível seguir para o próximo nível, antes disso, deve-se incorporar ao documento guia de estilo da interface o documento de padrões de telas.

2.1.3.3. Nível 3

No nível 3, o projeto da UI é feito de maneira detalhada, ou seja, neste momento o projetista integra ao projeto os detalhes que até então haviam sido desconsiderados. Isso resultará em um documento com opções não-essenciais de janelas, caixas de diálogos, itens de menu, formulários, etc.

Os testes nesse nível aproximam-se ainda mais da realidade, pois é possível avaliar integrações entre interfaces que até o momento eram avaliadas individualmente. Além disso, é possível medir tempos de tarefas para verificar se as respostas estão de acordo com o que foi definido na fase de análise de requisitos.

O documento resultante deste nível também deve ser incorporado ao documento guia de estilo da interface.

2.1.3.4. Instalação

Após concluir a etapa de projeto, testes e implementação com base nos três níveis apresentados anteriormente, parte-se para a instalação do projeto que consiste em disponibilizar o sistema em ambiente de produção, para que os usuários que já tiveram algum contato com as interfaces durante os testes anteriores, possam agora participar de avaliações objetivas de usabilidade por meio das diversas técnicas existentes como: questionários, entrevistas, teste formal de usabilidade, estudos de uso, etc.

O resultado disso é a realimentação do projeto de interface para futuras versões do produto ou de produtos familiares.

2.1.4. ISO 13407

A norma ISO 13407 - *Human centred design processes for interactive systems* (ISO 13407, 1999) também é baseada no desenvolvimento de interface com o usuário de forma iterativa, fornecendo orientação de como atingir a usabilidade por meio de atividades que incorporam fatores humanos e

ergonômicos, conhecimentos e técnicas com o objetivo de melhorar a eficácia e a produtividade, melhorando as condições de trabalho humano, e neutralizando os possíveis efeitos adversos da utilização na saúde humana, segurança e desempenho. O processo iterativo é composto por cinco fases que devem ser seguidas até que as necessidades dos usuário e da organização sejam satisfeitas [10]. A figura 3 a seguir ilustra o processo da ISO 13407:

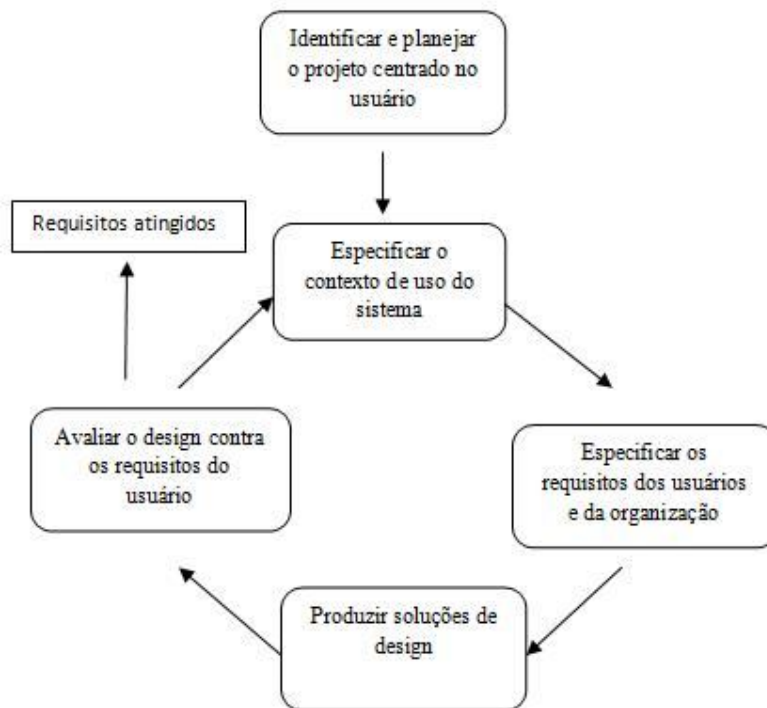


Figura 3: ISO 13407.

Como uma evolução da norma ISO 13407, surgiu a norma ISO TR 18529 - *Human-centred lifecycle process descriptions* (ISO TR 18529, 2000) com o objetivo de tornar o conteúdo da norma ISO 13407 acessível à especialistas em avaliação de processos de software e também para aqueles familiarizados ou envolvidos com modelagem de processos. Essa norma propõe um conjunto de sete práticas que serve de modelo para guiar os processos que deveriam ser executados por uma organização para atingir as metas de usabilidade [10]. As sete práticas são:

1. Certificar-se do conteúdo HCD (*Human Centered Design*) na estratégia do sistema
 - Representar os *stakeholders*⁶
 - Coletar inteligência de mercado
 - Definir e planejar a estratégia do sistema
 - Coletar *feedback* do mercado
 - Analisar tendências de usuários

2. Planejar e gerenciar o processo HCD
 - Consultar os *stakeholders*
 - Identificar e planejar o envolvimento dos usuários
 - Selecionar métodos e técnicas centradas no humano
 - Assegurar uma abordagem centrada no humano dentro da equipe
 - Planejar atividades de design centrado no humano
 - Gerenciar atividades centradas no humano
 - Defender abordagens centradas no humano
 - Fornecer suporte para o design centrado no humano

3. Especificar os *stakeholders* e os requisitos organizacionais
 - Esclarecer e documentar os objetivos do sistema
 - Analisar os *stakeholders*
 - Avaliar os riscos para os *stakeholders*
 - Definir a utilização do sistema
 - Gerar os requisitos dos *stakeholders* e da organização
 - Definir os objetivos de qualidade de uso

4. Compreender e especificar o contexto de uso
 - Identificar e documentar as tarefas do usuário
 - Identificar e documentar os atributos do usuário

⁶ Stakeholders: São indivíduos ou organizações que estão ativamente envolvidos no projeto, ou cujos interesses podem ser positiva ou negativamente afetados pelos resultados do projeto [PMI/PMBOK – Project Management Body of Knowledge (PMBOK® Guide – 2000 Edition)].

- Identificar e documentar o ambiente organizacional
- Identificar e documentar o ambiente técnico
- Identificar e documentar o ambiente físico

5. Produzir soluções de design

- Alocar funções
- Produzir um modelo de tarefa composto
- Explorar design do sistema
- Utilizar conhecimento existente para desenvolver soluções de design
- Especificar o sistema
- Desenvolver protótipos
- Desenvolver treinamento para o usuário
- Desenvolver suporte ao usuário

6. Avaliar o design contra os requisitos

- Especificar e validar o contexto de avaliação
- Avaliar os protótipos no início para definir os requisitos para o sistema
- Avaliar os protótipos a fim de melhorar o design
- Avaliar o sistema para verificar se os requisitos organizacionais e dos *stakeholders* foram cumpridos
- Avaliar o sistema a fim de verificar que as práticas exigidas têm sido seguidas
- Avaliar o sistema em uso a fim de garantir que continua cumprindo as necessidades dos usuários e da organização

7. Introduzir e operar o sistema

- Gestão de mudança
- Determinar o impacto sobre a organização e os *stakeholders*
- Customizar e localizar o design
- Entregar o treinamento dos usuários
- Fornecer suporte aos usuários nas atividades planejadas

- Assegurar a conformidade com a legislação ergonômica de trabalho.

3. Desenvolvimento ágil de software

Perante um cenário delineado pela pressão da constante inovação e concorrência acirrada imposta pelo mercado, ser competitivo hoje em dia significa entregar os produtos certos⁷ em períodos cada vez menores [26]. Mediante a este contexto, optou-se por trabalhar com metodologias ágeis neste trabalho, a fim de se obter resultados mais realistas e também de superar o desafio e o paradigma existentes hoje em muitas empresas, que limitam a utilização de técnicas de usabilidade em seus processos habituais de desenvolvimento de software por acreditarem que usabilidade é um aspecto secundário que pode ser deixado para depois.

Este capítulo apresenta os conceitos com “Desenvolvimento ágil de software” e o que motivou o surgimento dessa nova visão de desenvolvimento de software. Serão apresentadas também duas das mais conhecidas e utilizadas metodologias de desenvolvimento ágil: XP e Scrum. Estas duas metodologias serão referenciadas também no capítulo seguinte onde são apresentadas algumas propostas de metodologias de integração entre Usabilidade e Métodos ágeis.

Foi em meio à exigência pela dinamicidade que surgiu o Manifesto ágil em 2001, proposta que veio como uma reação contra os métodos tradicionais de desenvolvimento de projetos que exigiam uma carga pesada de burocracias, regulamentações e regimentos tornando o processo de desenvolvimento muito lento e pouco produtivo no novo cenário que veio se formando nos últimos anos.

Essa revolução não significa que qualquer documentação deve ser abolida durante o planejamento de um projeto de software. Trata-se de uma reformulação de processos e de gestão de projetos para que de fato somente os objetivos de real significância sejam o foco de todos os envolvidos no projeto.

⁷ Produto que está em conformidade com os requisitos levantados e que atende as características implícitas esperadas (Pressman 1994 [4]).

Manifesto ágil ou *Agile Software Development Manifesto*, foi criado nos EUA por um grupo de profissionais veteranos na área de software, que se reuniram a fim de discutir maneiras de melhorar o processo de desenvolvimento de projetos. O resultado disso foi um documento que reuniu os princípios e práticas propostas pela metodologia ágil, descritos a seguir.

3.1. Princípios do manifesto ágil

De acordo com o Manifesto ágil, são doze os princípios a serem considerados no desenvolvimento de software [\[11\]](#):

1. Nossa maior prioridade é satisfazer o cliente, por meio da entrega adiantada e contínua de software de valor.
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
3. Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
4. Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
5. Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
6. O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é por meio de uma conversa cara a cara.
7. Software funcional é a medida primária de progresso.
8. Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
9. Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.

11. As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis.
12. Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

Muitos dos autores do manifesto ágil fundaram metodologias que implementam os princípios e práticas propostas pelo documento. São exemplos de metodologias de desenvolvimento ágil: *Extreme Programming(XP)*, SCRUM, DSDM, *Adaptive Software Development*, *Crystal*, *Feature-Driven Development*, *Pragmatic Programming*.

Cada uma destas metodologias possui suas próprias diretrizes, porém todas cultivam aspectos comuns no que diz respeito ao desenvolvimento, à comunicação iterativa e ao foco na redução do esforço empregado em artefatos de software intermediários. Estas características são determinantes na diferenciação das metodologias tradicionais, em que o planejamento e o esforço aplicado aos artefatos intermediários é relativamente grande, o que implica no aumento do ciclo de vida do projeto.

Metodologias ágeis começam com a premissa de que projetos são imprevisíveis e que a incerteza do mercado vai provocar mudanças de escopo durante o seu andamento. Diante disso, é certo que o plano da organização deve se adaptar conforme as mudanças vão ocorrendo.

Os requisitos de projetos ágeis são escritos como fatias verticais do sistema como um todo e construídos de forma que eles sejam, na sua maioria, independentes, ou seja, que possam ser priorizados e implementados em qualquer ordem, permitindo assim, a entrega adiantada e contínua de software de valor. Esta é uma das características mais fortes no desenvolvimento ágil.

3.2. Algumas metodologias ágeis

3.2.1. SCRUM

Scrum é um *framework* utilizado para gerenciamento de projetos e sua proposta é focada no desenvolvimento iterativo e incremental. Foi abordado pela primeira vez em 1986 no artigo "*The New Product Development Game*"

escrito por Hirotaka Takeuchi e Ikujiro Nonaka e publicado na revista *Harvard Business Review*. Neste artigo eles relatam que projetos compostos por pequenos times multidisciplinares são capazes de produzir resultados melhores. Em 1995 este *framework* foi apresentado formalmente na conferência anual **OOPSLA** (*Object-Oriented Programming, Systems, Languages & Applications*).

O *Scrum* permite a inclusão de diversos processos e técnicas a fim de melhorar as práticas de desenvolvimento para a produção de produtos complexos. Ele utiliza uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar riscos. Sua estrutura é composta por: Times Scrum (Equipes), Eventos com duração fixa (*Time Boxes*), Artefatos e Regras [12].

3.2.1.1. Times Scrum (Equipes)

Um time Scrum corresponde à equipe que está diretamente ligada ao desenvolvimento do produto. É composta por três papéis:

- Scrum Master – possui a função de garantir que o processo seja compreendido e utilizado por todos os envolvidos. É de sua responsabilidade também contribuir com a evolução do Time treinando-o a fim desenvolver cada vez mais produtos de qualidade. É importante salientar que o Scrum Master não gerencia o Time, pois o Time é auto-gerenciável [12].
- Product Owner – papel que representa os interesses dos *stakeholders* do projeto, é responsável por definir as funcionalidades do produto e priorizá-las ao longo do desenvolvimento. Uma de suas funções também é aprovar ou não as tarefas entregues pela equipe de desenvolvimento [12].
- Time – o time é composto por um grupo de desenvolvedores com capacidades variadas, responsável por transformar os requisitos definidos pelo *product owner* em um produto funcional. Recomenda-se a quantidade de sete com mais ou menos duas pessoas para um time ótimo, sua composição deve ser multidisciplinar com conhecimento em áreas diversas, como: programação, controle de qualidade, análise de negócios, banco

de dados, etc. Outra característica importante de um time é o auto-gerenciamento, ou seja, ele é responsável por entregar os incrementos de funcionalidades dentro do período pré-definido, isso é possível por meio da sinergia que existe entre os membros e da multidisciplinaridade que permite a participação de cada um com um papel importante no desenvolvimento das tarefas [12].

3.2.1.2. Eventos com duração fixa (Time Boxes)

Scrum possui uma série de eventos (reuniões) que ocorrem durante o ciclo de desenvolvimento, a maioria acontece segundo uma seqüência lógica proposta pelo *framework*, conforme pode ser observado por meio da figura 4 abaixo:

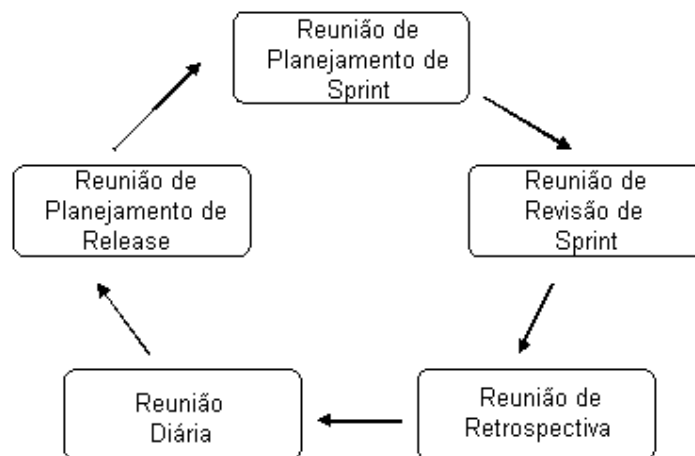


Figura 4: Sequência de reuniões proposta pelo Scrum

- Reunião de Planejamento de *Release* (*Release Planning Meeting*) – esta reunião visa a definição de um plano que compreende a meta da *release*, as maiores prioridades do *product backlog*⁸, os principais riscos

⁸ Product Backlog – é uma lista montada pelo product owner que contém todas as funcionalidades que deverão compor o produto final, essa lista é ordenada conforme a prioridade das tarefas. Cada tarefa do product backlog possui os atributos: descrição, prioridade e estimativa. Uma característica importante do product backlog é a dinamicidade, ou seja, à medida que o produto vai evoluindo, o product backlog evolui em paralelo com o objetivo de suprir as necessidades que vão surgindo para o produto, isso

e as características gerais e funcionalidades que estarão contidas na *release*. O plano também define uma data de entrega e os custos prováveis, a partir daí a organização pode inspecionar a evolução da *release* e fazer os devidos ajustes no plano a cada *Sprint* conforme a necessidade [12].

- *Sprint* – a *Sprint* corresponde à uma iteração do ciclo de desenvolvimento, com tempo de duração fixo. Uma *Sprint* compreende o período para o desenvolvimento de uma quantidade limitada de tarefas, ao final o resultado é um incremento de funcionalidades entregáveis do produto. As *Sprints* ocorrem uma após a outra, sem intervalo de tempo entre elas. É de responsabilidade do Scrum Master garantir que não ocorram mudanças que possam afetar a entrega da *Sprint*, porém o *product owner*⁹ possui autoridade para cancelar uma *Sprint* antes da sua conclusão [12].

- Reunião de Planejamento de *Sprint* (*Sprint Planning Meeting*) – Esta reunião é realizada antes do início de uma *Sprint*.

Para *Sprints* de 30 dias é recomendável uma reunião de 8 horas, para *Sprints* menores recomenda-se 5% do tempo total da *Sprint*.

Esse total de horas é dividido em duas partes: a primeira parte com a metade do tempo é utilizada para a definição do que será executado na *Sprint*, a segunda parte com a outra metade do tempo é utilizada para o Time decidir como irá implementar o que ficou definido na primeira parte da reunião.

Nesta reunião, o *product owner* apresenta quais são as próximas tarefas prioritárias do *product backlog* e a partir disso o Time decide quais tarefas ele terá a capacidade de implementar na *Sprint*. O

também ocorre devido à uma premissa do desenvolvimento ágil, que refere-se à capacidade de absorver mudanças de requisitos no meio do desenvolvimento de um produto [12].

⁹ Proprietário do Produto (cliente).

resultado é o *Sprint backlog*¹⁰ com a lista de tarefas que serão desenvolvidas na *Sprint* que está sendo planejada [12].

- Revisão da *Sprint* (*Sprint Review*) – reunião realizada ao término de cada *Sprint*. Para *Sprints* de 30 dias recomenda-se a duração de 4 horas, para *Sprints* menores recomenda-se no máximo 5% do tempo da *Sprint*. Participa dessa reunião o *Product Owner* que identifica o que foi entregue e o que não foi. O time discute os pontos positivos e os pontos negativos, bem como a solução aplicada para os problemas que ocorreram na *Sprint* que encerrou. Os pontos discutidos nessa reunião ajudam a identificar melhores decisões a serem tomadas nas próximas reuniões de planejamento de *Sprints* [12].
- Retrospectiva da *Sprint* (*Sprint Retrospective*) – essa reunião ocorre após a Revisão e antes do Planejamento da próxima *Sprint*. Possui duração fixa de 3 horas e visa inspecionar aspectos como: relacionamento entre as pessoas, processos e ferramentas utilizadas na última *Sprint*. Ao final da reunião, o Time deve apresentar um plano de melhorias a serem implementadas com o objetivo de tornar o processo de desenvolvimento mais produtivo e gratificante [12].
- Reunião Diária (*Daily Scrum*) – conforme o nome sugere, essa reunião ocorre diariamente com duração de 15 minutos. Recomenda-se que essa reunião aconteça todos os dias no mesmo horário e no mesmo local. Seu principal objetivo é aumentar a probabilidade do Time alcançar a meta da *Sprint*. É de responsabilidade do Time promover esta reunião na qual cada membro socializa os seguintes itens:
 - O que ele fez no dia anterior desde a última reunião diária;
 - O que ele fará no dia atual a partir da reunião que está ocorrendo;

¹⁰ Sprint Backlog – é uma parte do product backlog organizada também em forma de lista e contém todas as funcionalidades que serão implementadas em uma *Sprint*, o resultado de uma *Sprint backlog* é um incremento potencialmente entregável do produto, usa-se esse artifício pois o *product owner* pode resolver implantar a versão liberada ao final de uma *Sprint* [12].

- Quais os impedimentos que afetam a continuidade do desenvolvimento;

Com este acompanhamento diário, é possível promover a tomada rápida de decisões e manter todos os membros atualizados acerca do que está ocorrendo no projeto [12].

3.2.1.3. Regras

As regras são responsáveis por guiar o funcionamento correto do Scrum, são elas que fazem o elo entre os papéis, os artefatos e os eventos de duração fixa [12]. Scrum é uma metodologia adaptável a diversos ambientes de desenvolvimento, suas regras não são rígidas ao ponto de limitar sua utilização à ambientes pré-determinados. Essa é uma grande vantagem das metodologias de desenvolvimento ágil. A maioria propõe um processo com brechas que podem ser preenchidas com particularidades inerentes ao ambiente de desenvolvimento que serão utilizadas, conforme ilustrado na figura 5 [13] abaixo.

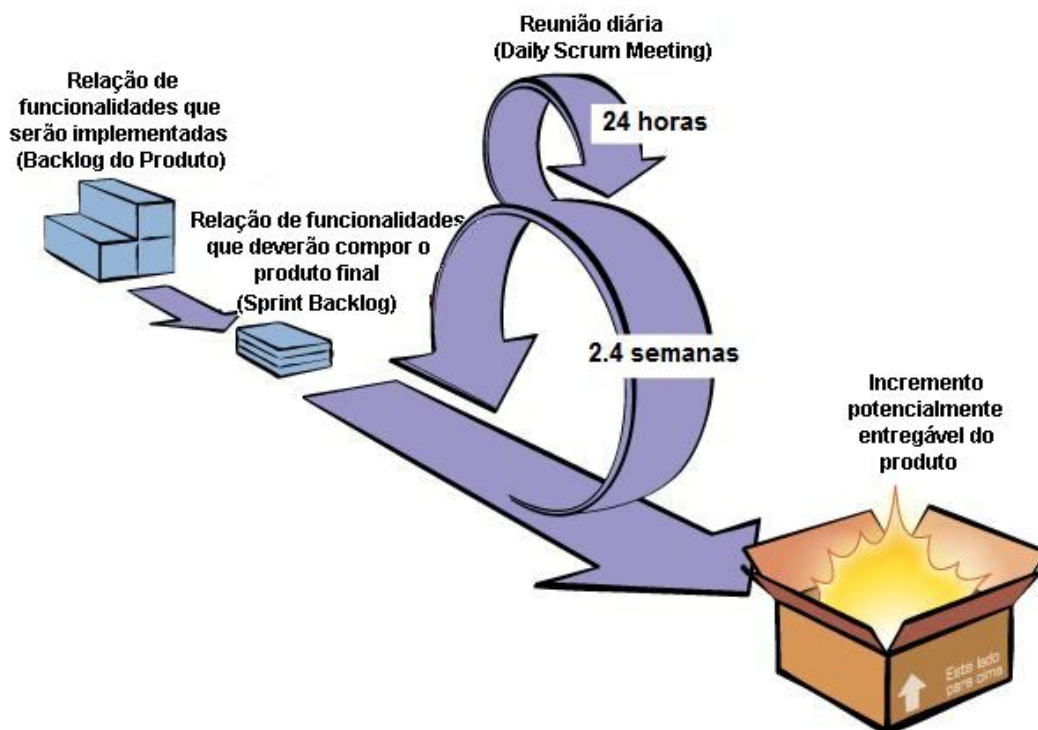


Figura 5: Ciclo proposto pelo Scrum

3.2.2. Extreme Programming (XP)

Extreme Programming ou XP é uma das metodologias ágeis mais conhecidas. Propõe a construção de sistemas de software com qualidade, redução de custos e entrega no menor tempo possível. Surgiu em 1996 nos Estados Unidos, foi idealizada por Kent Beck que definiu XP como uma maneira leve, eficiente, de baixo risco, flexível, previsível, científica e divertida para se desenvolver software.

A disciplina XP, como foi classificada por Beck, envolve a participação de dois papéis essenciais: O programador e o cliente. Os dois papéis se complementam e representam a estrutura principal do desenvolvimento. O programador é visto como o coração da XP, ele sabe como programar o que o cliente solicitou, portanto um depende do outro. Os demais papéis existentes em XP são [\[14\]](#):

- Testador – responsável por ajudar o cliente a escrever os testes funcionais e, se for o caso, executá-los apresentando os resultados em lugar bem visível a todos os envolvidos.
- Rastreador – responsável por dar *feedbacks* sobre as estimativas realizadas, visando a melhoria constante, por acompanhar o andamento das iterações, sinalizando qualquer alerta quanto ao prazo de conclusão e também é responsável por registrar informações pertinentes aos testes executados.
- Treinador – responsável por manter o andamento correto do processo, tomando as atitudes necessárias sempre que houver algum tipo de desvio nesse sentido.
- Consultor – especialista em determinada área técnica que fornece ajuda ao time quando o mesmo não possui o conhecimento suficiente naquela área.
- O Chefão – responsável por incentivar o time dando-lhes coragem, confiança.

XP baseia-se em cinco princípios fundamentais no que diz respeito ao desenvolvimento de projetos de software: simplicidade presumida, *feedback* rápido, mudanças incrementais, aceitação das mudanças e alta qualidade [\[14\]](#).

- Simplicidade presumida – este princípio sugere que todos os problemas que surgirem devem ser tratados como algo extremamente simples de ser resolvido. Isso pode ser feito pois XP recomenda que o trabalho de hoje seja suficiente apenas para resolver os problemas de hoje, pois considera que os recursos possuem habilidades suficientes para adicionar complexidade no futuro caso ela venha a ser necessária, independente de como sistema está implementado.
- *Feedback* rápido - tem por objetivo permitir uma aprendizagem mais rápida do que está sendo realizado, dessa forma é possível aplicar esse aprendizado o mais rápido possível a fim de melhorar constantemente. No contexto de XP, o *feedback* rápido permite que o cliente identifique melhorias, não conformidades e alterações de requisitos em dias ou semanas, ao invés de meses ou anos, da mesma forma os programadores conseguem obter um aprendizado sobre a melhor forma de projetar, implementar e testar o sistema com base nos *feedbacks* em escala de segundos e minutos ao invés de dias, semanas ou meses [2].
- Mudanças incrementais – XP sugere que mudanças ocorram aos poucos em quantidades pequenas de forma incremental, a fim de reduzir o risco de implementar uma mudança grande de alto impacto que possa fugir do controle previsível. Dessa maneira, o projeto é modificado um pouco de cada vez, o planejamento muda um pouco de cada vez, o time muda um pouco de cada vez.
- Aceitação das mudanças – as mudanças são previstas em XP e podem ocorrer durante o ciclo de desenvolvimento. É necessário estar preparado para aceitar essas mudanças preservando o maior número de opções para o cliente.
- Alta qualidade – este princípio relaciona-se com uma das variáveis de desenvolvimento de projetos proposta por XP: a qualidade, esta não é uma variável livre, possui como valores únicos possíveis: “Excelente” ou “Insanamente excelente”, se não for assim o projeto não terá sucesso.

3.2.2.1. Processo de desenvolvimento

Seguindo um dos princípios do manifesto ágil, o processo de desenvolvimento em XP é iterativo e incremental, ou seja, o projeto é desenvolvido em ciclos periódicos. A cada iteração, é entregue um pacote de funcionalidades ao cliente com algum valor para o negócio, porém essa versão não contempla todas as necessidades do sistema.

Uma nova iteração começa com a criação de *User Stories* que são pequenas histórias escritas pelo cliente contando o que cada funcionalidade, prevista para aquele ciclo, deverá fazer. Essas histórias são escritas em cartões indexados sem a utilização de termos técnicos. A partir disso, a equipe de programadores irá propor as estimativas para cada *User Story*. Ao concluir o processo de estimativas, os cartões são devolvidos ao cliente para que seja feita a priorização das funcionalidades de acordo com o tempo disponível para o ciclo atual e a importância do desenvolvimento de cada funcionalidade [14].

Quanto às histórias que não puderam entrar nesta iteração, estas serão postergadas para o próximo ciclo que seguirá o mesmo processo descrito até então. Essas histórias só serão incluídas na próxima iteração caso possuam um grau de necessidade maior que as próximas histórias que serão escritas pelo cliente [14].

Um método polêmico adotado pelo XP é o *Pair Programming* que trata-se de reunir dois programadores em um único computador a fim de compartilhar idéias, conhecimento e código para obter melhores soluções. Por mais contraditório que possa parecer, esse método promete não demandar mais recursos, já que consegue potencializar a qualidade do código que é desenvolvido [14].

Outra prática diferenciada é manter o cliente em contato direto com a equipe de desenvolvedores, dando-lhe liberdade para opinar sobre determinado desenvolvimento antes mesmo da conclusão. Dessa maneira é possível absorver questões importantes e essenciais para o cliente que até então não haviam sido percebidas com nitidez [14].

Para garantir a qualidade excelente, XP propõe que os testes ocorram em paralelo com a programação, ou seja, os programadores são responsáveis por executar os testes de unidade que devem ser escritos antes mesmo do

início da programação. Outra frente de testes é de responsabilidade do cliente que prepara um conjunto de testes funcionais para verificar se o que foi desenvolvido está de acordo com as suas necessidades [14].

O final de cada iteração em XP é marcado pela entrega de uma versão ao cliente que equivale à uma parcial do produto final. É neste momento também que a equipe reúne-se para fazer uma retrospectiva do ciclo que se encerrou para identificar o que funcionou bem e o que pode ser melhorado [14].

4. Usabilidade X Desenvolvimento ágil

Conforme consta no capítulo III, o desenvolvimento ágil surgiu com o intuito de acelerar a entrega dos projetos de software, visto que o mercado competitivo que veio se formando, passou a exigir cada vez mais e em menor tempo e custo. Essa demanda faz crer que a utilização de técnicas de usabilidade em projetos de software perde prioridade perante outras necessidades impostas por esse novo cenário.

Segundo Jakob Nielsen [15], os métodos ágeis correspondem a uma proposta elaborada por programadores, estando assim diretamente relacionada à implementação dos sistemas, o que deixa a desejar no design e usabilidade dos mesmos. Outra questão preocupante para Nielsen, é que em metodologias ágeis, o desenvolvimento é iterativo, ou seja, pequenas partes do sistema vão sendo entregues gradativamente ao cliente, o que pode prejudicar o conceito de uma experiência de usuário integrada. Na pior das hipóteses, a interface de usuário pode acabar tornando-se uma colcha de retalhos [15].

Perante essa suposta ameaça de incompatibilidade entre Desenvolvimento ágil e Usabilidade, alguns autores propuseram metodologias de integração que fossem capazes de respeitar os princípios dos métodos ágeis sem prejudicar a qualidade da Interação Humano-Computador (IHC) da Usabilidade. Dentre os autores das propostas de integração, pode-se citar Constantine & Lockwood (2001), Memmel, Gundelsweiler e Reiterer (2007) e Vasconcelos (2004).

A seguir serão apresentadas algumas das metodologias identificadas durante as pesquisas realizadas para este trabalho, na busca de evidências de que é possível inserir conceitos e técnicas de usabilidade em ambientes de

desenvolvimento ágil sem perdas nos resultados, e que ainda implique em ganho de qualidade no produto final.

Ao final do capítulo, será apresentada também uma experiência prática baseada na incorporação de conceitos e técnicas de usabilidade em uma empresa que possuía um processo de desenvolvimento ágil. Essa experiência foi relatada por meio de uma entrevista realizada com o especialista em usabilidade Walter Cybis.

4.1. Agile Usage-Centered Design (AUCD)

Na década de 1990, Constantine e Lockwood criaram uma abordagem de projeto centrada no “uso” do sistema, a *Usage-Centered Design*, em que o objetivo maior era usar de quaisquer ferramentas e técnicas capazes de projetar sistemas mais utilizáveis em menos tempo. Mais tarde, em 2001, eles perceberam que poderiam adaptar a proposta inicial ao modelo ágil de desenvolvimento de software, e foi onde surgiu então a abordagem *Agile Usage-Centered Design* (AUCD) baseada na metodologia ágil XP.

A essência da proposta está fundamentada na criação de casos de tarefas (*task cases*) que orientam o design da interface de usuário e ao mesmo tempo são suficientes para contemplar as necessidades da equipe de programadores no que diz respeito à criação de classes, métodos, objetos e seus relacionamentos [16].

Para atingir os melhores resultados por meio da utilização da proposta AUCD, é fundamental a colaboração e participação dos usuários, especialistas do domínio e clientes. Caso estes não sejam participantes presentes no processo de design, é necessário que estejam disponíveis para revisão e validação do trabalho feito pela equipe do projeto. As atividades relacionadas com o processo de desenvolvimento proposto pela AUCD para uma equipe composta por designers, desenvolvedores e no mínimo um usuário ou representante do usuário, são: definição, refinamento e priorização de papéis; definição, refinamento, priorização, descrição e organização de tarefas; prototipação de papel; refinamento de protótipos e ainda construção da interface, conforme segue [16]:

1. **Definição de papéis** – esta atividade se refere à criação de uma lista de papéis que os usuários poderão desempenhar no sistema. Esta lista é efetuada por meio de cartões indexados, em sessões de *brainstorming*¹¹.
2. **Refinamento de papéis** – nesta etapa a lista de papéis é revisada e refinada. São descritos aspectos importantes de cada papel nos cartões indexados.
3. **Priorização de papéis** – os cartões de papéis dos usuários são ordenados, com a finalidade de se criar um ranking de prioridades para o sucesso do projeto.
4. **Definição das tarefas** – esta atividade prevê a construção de uma lista de casos de tarefa (*task cases*) ou casos de uso essenciais para dar suporte aos papéis dos usuários levantados na sessão de *brainstorming*, começando pelos papéis de maior prioridade (a construção deve ser efetuada também com cartões indexados).
5. **Priorização das tarefas** – nesta atividade, os cartões de casos de tarefa devem ser ordenados, primeiro pela frequência antecipada e por último pela importância em relação ao sucesso do projeto. Após, classificar os cartões indexados em três categorias: necessário (deve constar na primeira versão), desejado (fazer se houver tempo) e descartável (marcar os cartões com as suas categorias).
6. **Descrição das tarefas** – para os cartões vinculados a categoria “necessário”, juntamente com os desejados que são críticos, complexos, não-claros ou interessantes, escrever uma narrativa no cartão indexado da tarefa. Esta narrativa deve abordar o “caso de sucesso” (fluxo normal dos eventos) de forma essencial (abstrata, simplificada e livre de tecnologia) usando o formato padrão de duas colunas, uma para as intenções do usuário e outra para as responsabilidades do sistema.

¹¹O Brainstorming é um método de geração coletiva de novas idéias através da contribuição e participação de diversos indivíduos inseridos num grupo. A utilização deste método baseia-se no pressuposto de que um grupo gera mais idéias do que os indivíduos isoladamente e constitui, por isso, uma importante fonte de inovação através do desenvolvimento de pensamentos criativos e promissores [17].

7. **Organização das tarefas** – nesta etapa os cartões de casos de tarefa devem ser agrupados em grupos que possuam um cenário comum ou uma seqüência temporal.
8. **Prototipação de papel** – esta etapa prevê o tratamento de cada grupo de cartões como um conjunto de tarefas a serem apoiadas por um contexto de interação (tela, página, caixa de diálogo ou algo parecido) na interface do usuário. Em seguida deve-se esboçar a prototipação do papel para essa parte da interface, concentrando-se nos casos de tarefa necessários, sem esquecer-se dos demais, pois também são importantes para a harmonia da interface.
9. **Refinamento de protótipos** – nesta etapa os protótipos são inspecionados por usuários e clientes utilizando cenários derivados dos casos de tarefas (Revisar e refinar os protótipos de papel com base nos resultados da inspeção).
10. **Construção da interface** – inicialmente deve-se programar a interface do usuário ou a camada de apresentação com base nos protótipos de papel ou nos casos de tarefas associados.

Após, o desenvolvimento segue o padrão iterativo das metodologias ágeis, em que a cada ciclo novos papéis e casos de tarefas vão sendo adicionados ao projeto a fim de refinar e expandir a interface do usuário [\[16\]](#).

4.2. CRUISER

Visando preencher a lacuna existente entre a engenharia de software e a Interação Humano-Computador (IHC), Thomas Memmel, Fredrik Gundelsweiler e Harald Reiterer propuseram no ano de 2007 uma disciplina sob os princípios do desenvolvimento ágil que cruza conceitos de engenharia de software e IHC. Foi denominada CRUISER por abreviação de: *A Cross-Discipline User Interface and Software Engineering Lifecycle.*

O ciclo de desenvolvimento proposto por CRUISER inicia com a fase de levantamento dos requisitos iniciais do sistema (IRUP - *Initial Requirements Up-Front*). O desafio dessa fase é não ferir a agilidade proposta pelo

desenvolvimento ágil, já que faz uso de conceitos e técnicas de diferentes disciplinas [18]. A Tabela 1 a seguir, ilustra a composição da fase IRUP:

Engenharia de software ágil	IHC	Design autoritário
Casos de uso, Cenários de uso; Requisitos Técnicos; Objetiva a performance do usuário;	Papéis e modelos de tarefas; Cenários de usuário, tarefa, interação ; Casos de Uso Essenciais; Padrões UI; Objetiva a experiência do usuário;	Protótipos de alta fidelidade; Objetiva a qualidade Hedônica;

Tabela 1: Contribuição das disciplinas para levantamento dos requisitos iniciais em CRUISER

De acordo com um dos princípios da metodologia ágil: “Aplicar padrões de projeto, utilizar os recursos existentes”, CRUISER sugere o uso de guias de estilo provindos de IHC a fim de facilitar o processo de design e apoiar as definições a respeito dos padrões de interface, porém ao contrário dos guias de estilo existentes em IHC, estes devem ser simples e enxutos para adequarem-se ao aspecto ágil [18].

A segunda fase do ciclo de desenvolvimento é a fase inicial conceitual (ICP - *Initial Conceptual Phase*), que prevê a separação entre a criação dos protótipos de interface do usuário e os protótipos de arquitetura do sistema com o intuito de acelerar o processo, já que desta maneira é possível evoluir o desenvolvimento de ambos em paralelo [18].

Em IHC, os protótipos são utilizados para representar o futuro sistema, permitindo a simulação da interação dos usuários com o mesmo. Eles proporcionam oportunidades para que o projetista obtenha *feedback*¹² mais fidedigno sobre os problemas e vantagens da interface em desenvolvimento [7]. CRUISER sugere a utilização dos dois tipos, porém com a ressalva de

¹² Feedback é um termo inglês, introduzido nas relações vivenciais para definir um processo muito importante na vida do grupo. Traduz-se por realimentação ou mecanismo de revisão [19].

equilibrar as vantagens e desvantagens de cada um considerando sempre o ambiente ágil do projeto, conforme pode ilustra a figura 6 a seguir: [18]

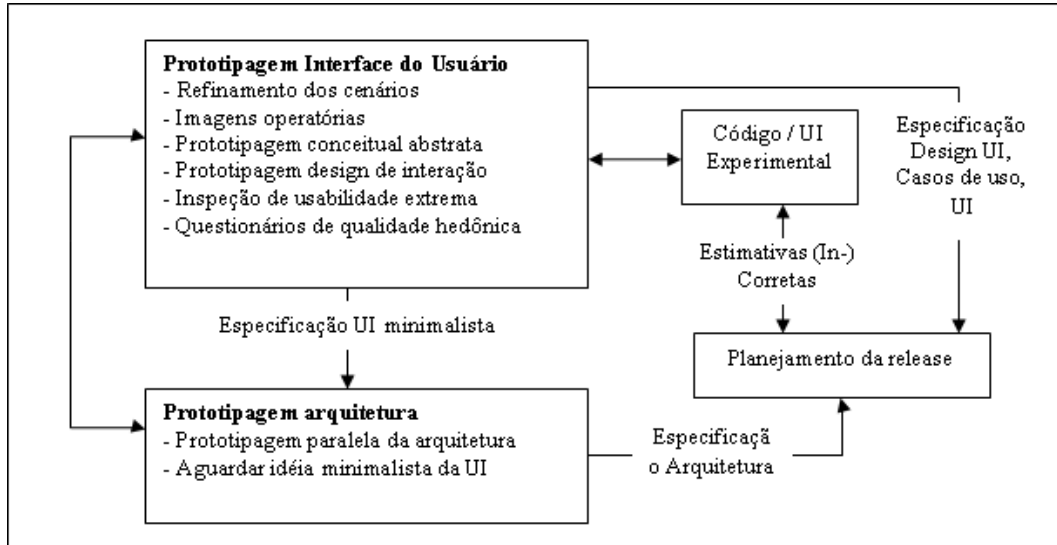


Figura 6: Fase inicial conceitual (ICP)

A fase seguinte diz respeito à construção e teste do sistema (CTP - *Construction and Test Phase*) que assemelha-se muito com o ciclo incremental e iterativo do XP. Neste momento, são definidas as funcionalidades que serão implementadas na próxima iteração. Assim como é feito na fase anterior de prototipagem, a codificação da arquitetura do sistema e da interface do usuário são desenvolvidas em paralelo. Os componentes de interface que impactam na arquitetura do sistema, podem ser desenvolvidos mais rapidamente e refinados nas próximas iterações. A fase CTP termina com a liberação de uma versão do sistema, da mesma forma que acontece no XP. Antes do planejamento da próxima iteração, a versão é avaliada para identificação de possíveis problemas de usabilidade, caso sejam encontrados, os defeitos são documentados em cartões indexados para posterior priorização e execução em iterações seguintes, conforme pode ser observado na figura 7 [18] a seguir.

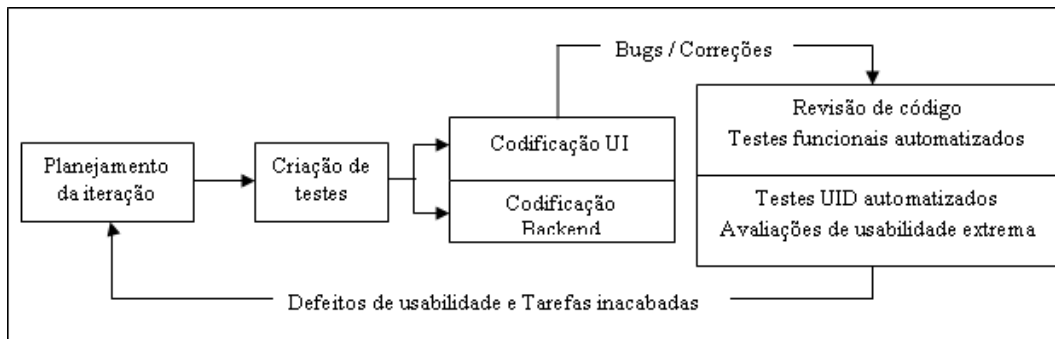


Figura 7: Fase de construção e testes (CTP)

O ciclo de vida CRUISER encerra com a fase de implantação. Neste momento o sistema é entregue ao cliente e os usuários passam a utilizá-lo. CRUISER prevê a identificação de melhorias por parte dos usuários mesmo depois da implantação do sistema, portanto é possível retornar a fases anteriores para implementar os novos requisitos [18].

4.3. XPU – Um modelo para o desenvolvimento de sistemas centrado no usuário

No ano de 2004, também com o intuito de prover um modelo para a construção de sistemas de software centrado no usuário em ambientes de desenvolvimento ágil, Vasconcelos criou a metodologia XPU (*eXtreme Programming* (minimizado) + Usabilidade) [20].

Os dois universos: Engenharia de Software (ES) e Interface Humano – Computador (IHC) foram combinados, uma vez que algumas das melhores práticas do desenvolvimento iterativo e incremental foram integradas com práticas de usabilidade na intenção de gerar-se um novo processo para a concepção de aplicações. A metodologia XPU é composta por sete fases, sendo elas [20]:

- Definição de papéis – esta fase diz respeito à definição do papel de cada membro da equipe de desenvolvimento. Esta prática é sugerida pela metodologia XP, e XPU utiliza-se da mesma e propõe novos papéis visando abranger também os aspectos de usabilidade. Os papéis recomendados são: Cliente, Usuário,

Gerente, Desenvolvedor, Testador e Especialista em usabilidade. O resultado da definição dos papéis é a construção de uma tabela composta pelo nome de cada componente, bem como o papel que o mesmo ocupará no desenvolvimento do projeto.

- Conversa com o cliente – por considerar de grande importância para o sucesso do projeto, XPU propõe nesta fase a criação de três artefatos da Engenharia de Software e da usabilidade que não são previstos pela metodologia XP. São eles: Documento de visão, Lista de objetivos do usuário e Documento de perfil do usuário.
- Inicialização – esta fase prevê o início de algumas atividades de planejamento, incluindo a construção de artefatos da ES, IHC e XP tais como: *User stories*, Modelo lógico de dados, Projeto arquitetural, Diagrama de classes, Modelo da tarefa.
- Planejamento de *releases* – conforme propõe a metodologia XP, em XPU a fase de planejamento de *releases* contempla a priorização de um conjunto de *user stories* que serão desenvolvidas e entregues na próxima versão do sistema. A duração de uma *release* é de quatro semanas, sendo que duas semanas formam uma iteração, é importante ressaltar que essa duração da *release* e das iterações em XPU é sempre fixa, o que varia são as atividades que serão priorizadas. Os artefatos gerados nesta fase são: Plano de *releases* e Modelo da tarefa.
- Planejamento da iteração – seguindo novamente a metodologia XP, o planejamento da iteração diz respeito à segregação das tarefas priorizadas na fase anterior, para que estas possam ser desenvolvidas em períodos menores (iterações) dentro da *release*. A particularidade nesta fase, é que as tarefas do usuário definidas no modelo de tarefas da fase anterior, não necessariamente precisam ser divididas já que o próprio modelo de tarefas possui um modelo hierárquico. Os artefatos sugeridos são: Tabela de Alocação de Tarefas, Modelo da interação e o Protótipo da interface do usuário.

- Implementação – esta fase representa o início da codificação do sistema com base em todos os artefatos criados nas fases anteriores. Visando a qualidade do código a ser produzido, XPU sugere a utilização de algumas práticas da metodologia XP, sendo elas: Integração contínua, Propriedade coletiva, Boas práticas de programação, Testes e Reunião de acompanhamento.
- Verificação de testes – esta fase é responsável por concluir uma iteração. Este é o momento de executar os testes propostos pelo cliente no início da iteração, bem como os testes de usabilidade a fim de validar se o resultado confere com os objetivos propostos ainda nas fases iniciais.

A seguir todas as fases comentadas anteriormente, são ilustradas na figura 8 que representa o fluxo de desenvolvimento da metodologia XPU.

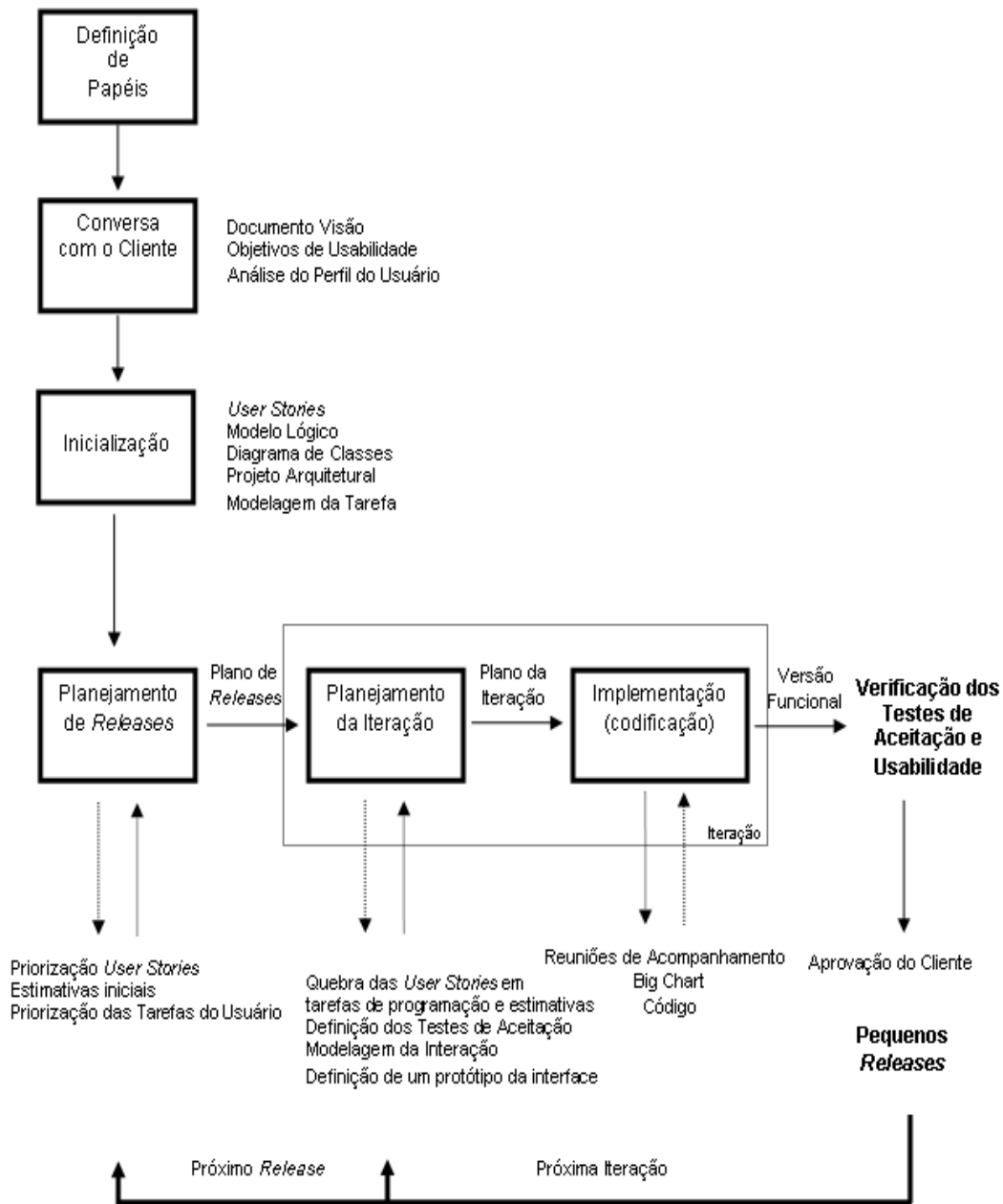


Figura 8: Estrutura do processo XPU

4.4. U-SCRUM: Uma metodologia ágil para promoção da usabilidade

U-SCRUM foi proposto por Mona Singh como uma variante da metodologia SCRUM tradicional, após comprovar por meio de experiências práticas que em projetos cuja usabilidade é um fator importante, a metodologia SCRUM não consegue atender adequadamente as necessidades dos usuários. Segundo Mona Singh, essa constatação pode ser confirmada pelos seguintes fatos relevantes [\[21\]](#):

- *Product owners* em SCRUM, geralmente são sobrecarregados com interesses de marketing e vendas, deixando à desejar a usabilidade do sistema;
- Muitas vezes falta competência e motivação para os *product owners* tradicionais incluírem no projeto experiências de usuário;
- Metodologias ágeis tradicionais, deixam pouco espaço para especificar uma visão de experiência do usuário, que impulsiona a arquitetura e é essencial para garantir uma experiência do usuário coerente.

O que diferencia a variante U-SCRUM é a inclusão de um *product owner* focado em usabilidade, além do *product owner* responsável pelas suas funções tradicionais. Portanto o projeto passa a ter uma dupla de *product owners* [\[21\]](#).

O papel do *product owner* de usabilidade é apresentar ao grupo uma visão da experiência do usuário, com o auxílio de *peessoas* que representam o perfil dos usuários do sistema. Nesta apresentação, são levantadas as metas de alto nível do usuário, o modelo de navegação do produto, incluindo sua relação com outros produtos, e também é efetuada uma descrição (em alto nível) da experiência do usuário com o produto [\[21\]](#).

Outros requisitos do sistema são liderados pelo *product owner* tradicional, ou seja, é de sua responsabilidade tudo que está relacionado ao *backend* do sistema. Cabe aos dois *product owners* especificarem as histórias do usuário conjuntamente a fim de manter o equilíbrio e as prioridades de desenvolvimento do projeto [\[21\]](#).

4.4.1. Limitações do SCRUM tradicional

Uma limitação do SCRUM é que durante os *Sprints*, não há mecanismos de *feedback* do usuário, dessa maneira pode-se dizer que há pouca oportunidade para construção de protótipos e validação de requisitos antes de começar o desenvolvimento. Outra limitação básica da metodologia SCRUM, é o foco da equipe, ou seja, os objetivos que motivam todos, são direcionados para a conclusão e entrega do projeto e não para a experiência do usuário, mesmo quando a equipe inclui pessoal de usabilidade, é comum despriorizar requisitos de usabilidade de um *Sprint* para o outro quando o prazo está apertado [21].

É muito frequente em projetos que utilizam a metodologia SCRUM, uma preocupação focada na entrega de algo funcionando, mesmo que não esteja adequadamente utilizável, isso porque o SCRUM tradicional enfatiza a produção de resultados apenas no que diz respeito à funcionalidade com o argumento de que a usabilidade pode ser melhorada depois da entrega, conforme ilustra a figura 9 abaixo [21].

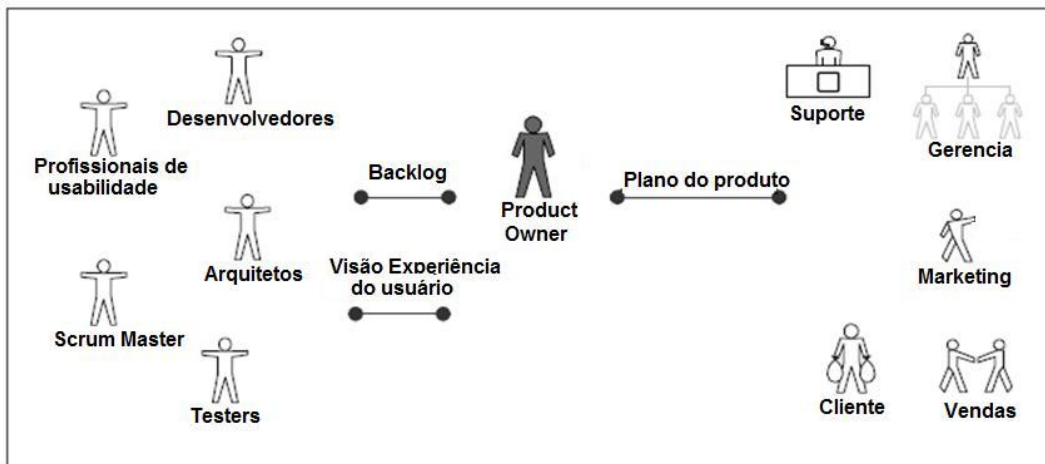


Figura 9: Papéis e artefatos da interação em SCRUM tradicional (Fonte: U-SCRUM)

4.4.2. A proposta U-SCRUM

Segundo Mona Singh, a inspiração para a criação do U-SCRUM veio pelas limitações citadas anteriormente no SCRUM tradicional e também pelas circunstâncias especiais de um projeto que teve desde o princípio a qualidade da experiência do usuário como um fator crucial. Sob o novo paradigma de desenvolvimento, um *product owner* de usabilidade foi incluído na equipe do projeto e a partir disso os dois *product owners* trabalharam conjuntamente para alcançar um primeiro acordo sobre a experiência do usuário. Isso foi possível por meio de observações dos usuários em seus lugares habituais de trabalho [21]. A figura 10 a seguir demonstra como ficou a estruturação de papéis e artefatos em U-SCRUM:

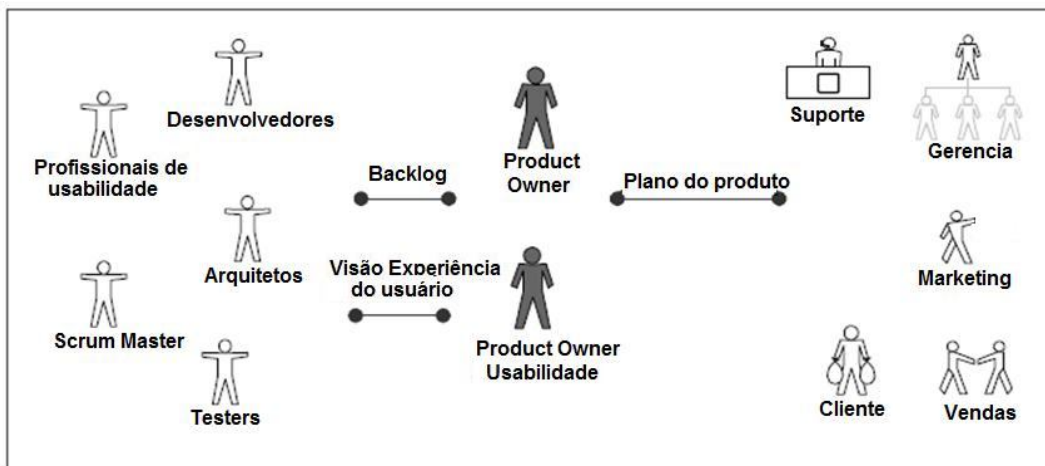


Figura 10: Papéis e artefatos da interação em U-SCRUM tradicional (Fonte: U-SCRUM)

Diferente do SCRUM tradicional, U-SCRUM utiliza-se de três artefatos principais: plano de projeto, visão da experiência do usuário e um *backlog*. Estruturalmente, o *backlog* assemelha-se muito ao *backlog* tradicional, porém inclui maior consideração para aspectos de usabilidade. O plano de projeto é construído em conjunto pelos dois *product owners*. A visão da experiência do usuário fornece uma base para discutir a concepção do produto com as principais partes interessadas, tais como marketing de clientes (internos e externos), vendas, suporte técnico e desenvolvedores. Esta visão contém as

informações do que é pedido e do que é entregue, facilitando assim o processo de comunicação entre as partes interessadas e melhorando a produtividade [21].

Em resumo, o fluxo de desenvolvimento da metodologia U-SCRUM no que diz respeito à usabilidade, acontece da seguinte forma: O *product owner* de usabilidade com a visão em mãos desenvolve um fluxo de tarefas como um *storyboard*¹³, após produz artefatos como *wireframes*¹⁴ a partir das entradas do *backlog* construído juntamente com o *product owner* tradicional. As *personas*¹⁵, como já dito anteriormente, são utilizadas nas reuniões com desenvolvedores e outros interessados no projeto. Para validar o produto em desenvolvimento, uma série de testes com usuários são utilizados, os usuários são convidados a avaliar protótipos de papel simples do *storyboard* e *wireframe* diferentes. Os resultados de tais avaliações alimentam as próximas iterações do projeto [10].

A estrutura da metodologia U-SCRUM, exige atenção especial para alguns fatores importantes e cruciais para o sucesso de sua aplicação. São eles [21]:

- Os dois *product owners* devem trabalhar conjuntamente. A fim de evitar possíveis conflitos, em nenhum momento as intenções de um devem sobrepor as do outro;
- Pode ser necessário uma coordenação adicional quando há dois *product owners*, todavia isto é variável;
- Há poucos profissionais que desempenham o papel de *product owner* com formação suficiente para assumir o papel de *product owner* de usabilidade;

¹³ O *storyboard* é utilizado para representar as janelas/páginas que serão inseridas na aplicação, bem como os seus elementos, antes do seu efetivo desenvolvimento [23].

¹⁴ *Wireframe* é um desenho básico, como um esqueleto, que demonstra de forma direta a arquitetura de como o objeto (interface, página da internet, modelo, etc.) final será de acordo com as especificações relatadas [24].

¹⁵ A *persona* é um perfil psico-social de um usuário e reflete os objetivos, habilidades e atitudes do mesmo.

- O uso de *personas* pode ser arriscado pois apresenta-se para a equipe de desenvolvimento como uma criação artificial;
- A visão da experiência do usuário deve fornecer um quadro completo do projeto e não um ciclo de vida do produto, pois se corre o risco de os desenvolvedores não enxergarem o objetivo final do projeto.

Os resultados obtidos nos projetos que utilizam a metodologia U-SCRUM têm sido bastante satisfatórios relata Mona Singh, a prova disso são os freqüentes comentários de usuários finais e outros interessados no projeto como: gestores de produto, pessoal de apoio técnico e equipe de vendas. Foi constatado também melhorias na produtividade dos desenvolvedores graças aos artefatos específicos de usabilidade que permitem um conhecimento mais global do produto [21].

4.5. Experiência prática de integração de usabilidade em um ambiente de desenvolvimento ágil

A empresa Interage que atua no ramo de segurança da informação em Porto Alegre, teve a iniciativa de aplicar em seus processos de desenvolvimento de software uma integração entre usabilidade e desenvolvimento ágil. Essa iniciativa foi apoiada pelo engenheiro de usabilidade Walter de Abreu Cybis autor do livro “Ergonomia e Usabilidade – Conhecimentos, Métodos e Aplicações” e visa a adoção de práticas de usabilidade a fim de enriquecer a qualidade dos seus produtos adaptando-as ao ambiente ágil de desenvolvimento de software.

Antes do início do primeiro projeto, foi feito um trabalho de conscientização dentro da empresa Interage por meio de *workshops*¹⁶, para explicar os conceitos, ferramentas, critérios e vantagens da usabilidade para todos os envolvidos no projeto. O resultado foi muito positivo, pois todos se sensibilizaram e compraram a idéia apresentada por Cybis, e isso sem dúvida

¹⁶ Workshop é uma reunião de grupos de trabalho interessados em determinado projeto ou atividade para discussão e/ou apresentação prática do referido projeto ou atividade [22].

foi um fator muito importante para o sucesso da utilização do método de integração.

O primeiro projeto baseado na proposta de integração, denominado Trevio, iniciou sem a participação do especialista em usabilidade, o que impediu a elaboração de modelagem conceitual e prototipagem, porém assim que o especialista em usabilidade foi inserido no projeto, algumas técnicas de usabilidade passaram a ser utilizadas em paralelo com o desenvolvimento iterativo e incremental proposto pela abordagem ágil.

O Trevio é um sistema que oferece controle, bloqueio e monitoramento do MSN para públicos variados visando a segurança na comunicação via MSN. Isso é possível por meio de uma série de configurações que o sistema possui permitindo customizações específicas para cada público alvo.

Tida como um fator determinante para o sucesso do produto, a usabilidade foi aplicada na interface do painel de controle do sistema. Essa interface é utilizada para configuração do serviço Trevio e permite a definição das configurações desejadas para o controle da comunicação. A versão 1.0 foi implementada priorizando a definição das regras de utilização do MSN, ou seja, a interface principal apresentava os campos para definir quais regras seriam aplicadas durante as conversações realizadas via MSN.

Durante a entrevista, o engenheiro de usabilidade Walter Cybis relatou alguns pontos importantes do processo de integração da usabilidade com processos ágeis o qual vem fazendo parte na empresa Interage. Segundo Cybis, não foi utilizado nenhum modelo de integração já proposto, tudo iniciou com base na sua experiência e conhecimento em usabilidade que permitiram a adaptação de práticas de usabilidade ao modelo de desenvolvimento ágil utilizado na empresa. Esse modelo é executado em ciclos denominados *Sprints* normalmente completados a cada duas semanas.

Cybis informou que o processo implantado na empresa interage em três pontos principais: programação em pares, avaliações heurísticas durante os *Sprints* e testes com usuários ao completar a primeira versão do sistema.

- Programação em pares: a programação em pares foi denominada assim, pois o desenvolvimento do sistema foi realizado pelo programador com o acompanhamento do especialista em usabilidade, isso ocorreu principalmente em função da ausência da modelagem conceitual e da prototipagem que conforme já foi dito, não puderam ser construídas no início do projeto.
- Avaliações heurísticas: as avaliações heurísticas eram realizadas a cada *Sprint* completado, e os problemas de usabilidade encontrados durante as avaliações eram listados e encaixados no próximo *Sprint* para as devidas correções.
- Testes com usuários: ao completar a primeira versão do sistema foram realizados alguns testes com usuários reais a fim de identificar os problemas de usabilidade na visão dos usuários. Os testes foram monitorados por meio da ferramenta *Camtasia* que faz a captura dos movimentos executados na tela do computador. Os problemas identificados nos testes alimentaram o último *Sprint* do ciclo de desenvolvimento que foi montado especificamente para corrigir problemas de usabilidade.

Após a conclusão do último *Sprint* do ciclo, o desenvolvimento prosseguiu com foco na performance do sistema, já que este também era um fator crítico e determinante para o sucesso do mesmo, em consequência disso os aspectos de usabilidade deixaram de ser levados em consideração nos *Sprints* seguintes.

Após a implantação, identificou-se por meio da experiência prática dos usuários na manipulação do sistema e também por meio da verificação de logs que mapeiam as ações executadas pelo usuário, que a performance estava apresentando problemas, especialmente no momento do login, pois todas as configurações eram carregadas a partir desse momento. Com base nisso o sistema sofreu uma alteração emergencial para liberação de uma versão 1.1. As alterações realizadas implicaram diretamente no aspecto de usabilidade do sistema, pois a interface inicial passou a não priorizar as regras de utilização do MSN, já que os logs ajudaram a identificar que o maior interesse dos usuários

era consultar conversas gravadas, portanto esse passou a ser o foco principal na interface inicial.

Atualmente, o sistema Trevio está sendo trabalhado para liberação da versão 2.0 com o objetivo de incluir uma série de novas funcionalidades, visando o melhor atendimento aos usuários. A nova versão segue o mesmo princípio da versão 1.1 em que o foco da interface principal do sistema é a visualização das conversas gravadas.

O novo projeto iniciou a pouco tempo já com a participação do especialista em usabilidade, que baseou-se nas pesquisas realizadas sobre padrões de interface para construir no primeiro *Sprint* de uma semana, o mapa conceitual do sistema, definindo as funções principais para a interface. O mapa conceitual representado pela figura 11 a seguir, foi construído com o auxílio da ferramenta *CmapTools*.

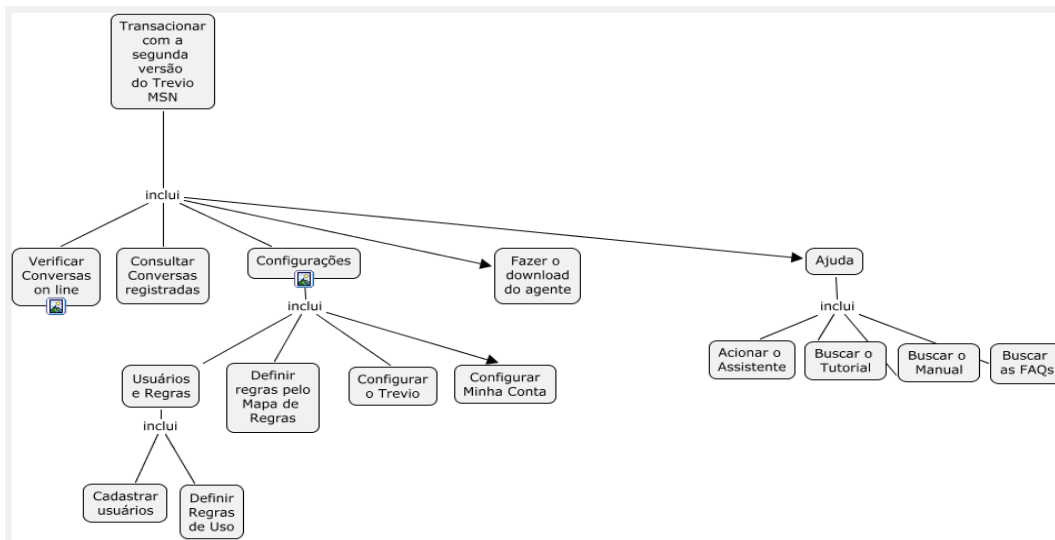


Figura 11: Modelo conceitual de mais alto nível do Trevio

No *Sprint* seguinte com duração de duas semanas, foi feita a maquetagem do sistema com base no mapa conceitual, desenhando as telas e organizando-as em uma seqüência que favorece a navegação e o aspecto dinâmico. A ferramenta utilizada foi a *Axure* que permite a construção de maquetes que oferecem interação com os usuários.

Após a conclusão das maquetes, no *Sprint* seguinte, os usuários farão testes a fim de encontrar problemas de usabilidade. O resultado da avaliação será utilizado para revisão e ajuste das maquetes do *Sprint* seguinte.

A partir dessa fase inicial do projeto, o desenvolvimento do sistema deve continuar conforme ocorreu nas versões anteriores conciliando em paralelo o desenvolvimento ágil e a usabilidade divididos em *Sprints* até a conclusão da versão 2.0.

Com a experiência obtida no projeto desde a implantação do processo de integração entre usabilidade e desenvolvimento ágil, Cybis afirmou durante a entrevista que os resultados foram muito satisfatórios e superaram as expectativas. A grande vantagem foi o fato de inserir conceitos de usabilidade em um contexto de desenvolvimento iterativo que favorece a realização de testes a cada iteração. A possibilidade de entregar uma versão funcionando do sistema a cada *Sprint* facilitou a identificação precoce de problemas de usabilidade, permitindo a rápida correção e o refinamento do sistema. A atenção dada para a usabilidade permitiu também uma objetividade muito maior, em função da convergência rápida na solução dos problemas, isso porque em um contexto normal de desenvolvimento que envolva uma equipe de programadores e uma equipe de testadores, geralmente ocorrem longas discussões devido à divergência de opiniões a respeito da melhor solução para cada problema. No contexto de integração com a usabilidade, os problemas foram capturados pelos próprios usuários que apontavam diretamente para a melhor solução.

ESTUDO DE CASO

1. Gennera: missão e visão da empresa

Com mais de dez anos no mercado, a Gennera dispõe de soluções integradas para gestão de Instituições de Ensino. Esta gestão compreende serviços relativos à administração acadêmica, como: matrículas, emissão de documentos oficiais (boletins, históricos, fichas individuais), registros de ocorrências, etc.; serviços financeiros, a exemplo: controle de títulos a receber e a pagar, cobrança; serviços de integração, a exemplo, integração contábil, entre outros. Atualmente, centenas de instituições são atendidas pelos serviços oferecidos, além de milhares de usuários, incluindo alunos, pais, professores e diretores.

O objetivo da empresa é fazer com que as equipes (funcionários das Instituições) gastem menos tempo e recursos na execução de tarefas administrativas, deixando sua preocupação principal para o trabalho de educação.

A missão da empresa é desenvolver soluções para facilitar a gestão de instituições de ensino, utilizando tecnologias inovadoras e seguras, estimulando o conhecimento e o desenvolvimento das pessoas de forma responsável, junto aos mercados que atuam e à sociedade. Já a visão da Gennera é ser a melhor e mais lucrativa empresa brasileira de soluções em gestão para instituições de ensino.

2. Sistema ASP Gestão Educacional Online

O nome ASP surgiu em função da utilização da tecnologia ASP – *Application Service Provider*. Esta solução fácil, simples e segura, permitiu a Gennera disponibilizar de forma pioneira as instituições parceiras o acesso aos seus dados pela Web 24 horas por dia, sem a necessidade de gastar tempo e dinheiro com infraestrutura tecnológica. Por ser totalmente online, as instituições não precisam investir na compra de equipamentos ou softwares

específicos, tampouco na instalação e administração de servidores, banco de dados ou backup, isto tudo é gerenciado e administrado pela Gennera.

O ASP Gestão Escolar / Educacional Online é uma solução que permite as instituições efetuar toda a sua gestão acadêmica e financeira, contando com serviços que oferecem segurança, facilidade de uso e acesso de qualquer lugar e a qualquer hora. Os professores também têm as tarefas facilitadas e podem interagir mais livremente com os alunos.

Com informações registradas em tempo real, os pais ou responsáveis podem ser mais pró-ativos no acompanhamento escolar dos seus filhos. O aluno tem acesso ao boletim online e diversos documentos disponibilizados pelo professor, além de um melhor canal de comunicação com ele, com outros setores da instituição e com outros estudantes.

3. Problema a ser estudado

3.1. Definição da *release*

Para iniciar a aplicação prática dos conhecimentos obtidos nas pesquisas realizadas, optou-se, dentre as metodologias estudadas, a metodologia U-SCRUM, em função da já utilização de algumas práticas de Scrum no processo de desenvolvimento da empresa Gennera.

Nossa contribuição para a empresa por meio deste trabalho iniciou, com base na pré-análise existente de cada solicitação, com o levantamento de casos que estavam causando impacto no trabalho dos utilizadores do sistema. A partir disso, foi realizada a reunião de planejamento da *release* no dia 01/04/2010. Nesta reunião foram definidas quais tarefas seriam desenvolvidas no período de 60 dias úteis, considerando a alocação de 4 horas diárias, já que a empresa não poderia liberar recursos dedicados em função das demais atividades executadas em paralelo pela mesma.

A reunião resultou na seleção de três melhorias para o sistema que além de alterações em funcionalidades, envolviam alterações de interface com o

usuário. Essas três melhorias foram incluídas no artefato *Product backlog* com a descrição, estimativa e prioridade de cada uma, conforme consta na figura abaixo. Considerando o período de 60 dias úteis, o cliente optou por dividir as entregas em três Sprints, conforme ilustra a figura 12 abaixo.

Product Backlog	
Data Início:	05/04/2010
Data Fim:	25/06/2010

Tarefas		
Descrição	Estimativa	Prioridade
Otimizar o registro de anotações de cobrança	X	1
Alterar os registros de Controle de Entrega	Y	2
Unificar dados para emissão de Bloquetos de Adotadores	Z	3

Figura 12: Product Backlog

3.2. Sprint 1: Registro de anotações de cobrança

3.2.1. Descrição do problema

O sistema possui recursos para a administração financeira da Instituição. Por meio de telas específicas, é possível efetuar o registro de valores recebidos relativos aos mais diversos tipos de receitas, que internamente são denominados Eventos Financeiros. São exemplos de eventos financeiros: Matrícula, Mensalidade, Material Didático, Alimentação, Passeios, etc. Esses eventos financeiros são vinculados no sistema aos alunos por meio do seu registro de matrícula.

Cada matrícula possui um responsável acadêmico, que administra as questões educacionais do aluno, e um responsável financeiro, que cuida da parte financeira do aluno perante a instituição de ensino. O responsável acadêmico e /ou financeiro é normalmente o pai ou a mãe do aluno, mas também pode ser um terceiro, como tia ou avó; um adotador, como uma Empresa ou ainda o próprio aluno.

Para analisar a situação financeira da instituição de ensino, o setor financeiro pode emitir relatórios de previsão e arrecadação, e também

relatórios de adimplência e inadimplência. Em se tratando de inadimplência, além da emissão de relatórios, cartas e e-mails, o financeiro também pode registrar os contatos efetuados com os responsáveis financeiros para a cobrança dos valores pendentes de pagamento ou ainda, agendar futuro contato por meio da adição de lembretes.

Todavia, todas estas atividades são executadas em telas diferentes do sistema, e isto aliado ao fato de os relatórios serem gerados somente em formato de impressão, estava tornando a tarefa dos funcionários do setor bastante trabalhosa, pelos seguintes motivos:

a) Ao iniciar o trabalho de cobrança, o usuário necessitava sempre imprimir todas as páginas do relatório;

b) O relatório impresso era utilizado para controlar ‘manualmente’ os contatos efetuados;

c) Caso o usuário, a partir dos lembretes registrados, quisesse selecionar os alunos no sistema para registrar novo contato de cobrança, precisava clicar uma vez sobre o lembrete para selecionar o registro em questão e abrir a tela de cobrança, e após, na tela aberta, informar novamente os dados do aluno, para efetuar a pesquisa e carregar os dados do mesmo em tela;

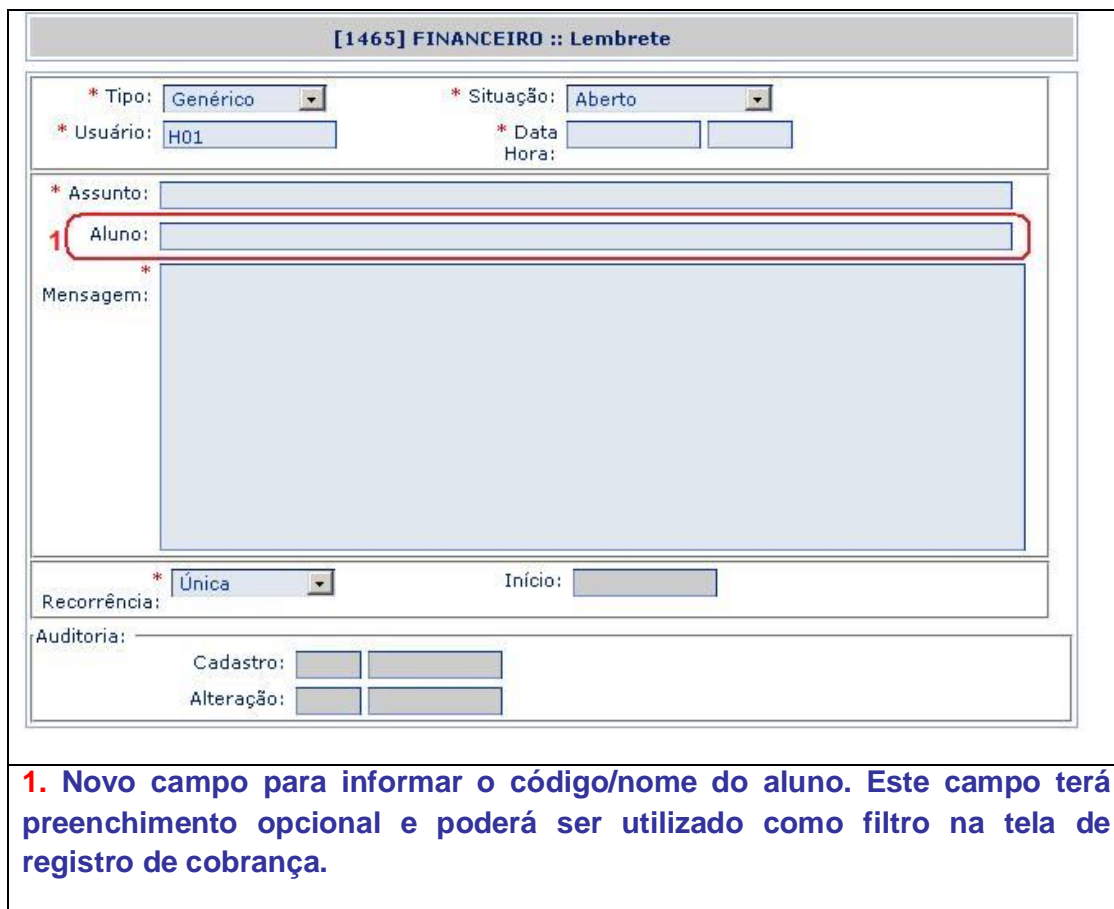
Se esta tarefa fosse executada esporadicamente, talvez esta situação não fosse tão incômoda, todavia, como é executada inúmeras vezes no mesmo dia e pelo mesmo usuário, acaba por gerar estresse, insatisfação e ainda queda de desempenho.

3.2.2. Definição da Sprint

Respeitando a ordem de prioridade definida no *product backlog*, a primeira *Sprint* tratou exclusivamente do problema relatado anteriormente: “Otimizar o Registro de anotações de cobrança”. O início desta *sprint* ocorreu com a reunião de planejamento, onde participaram o *product owner*, o *product owner* de usabilidade, o *scrum master* e o time.

Nesta reunião guiada pelo *Scrum Master*, o *product owner* juntamente com o *product owner* de usabilidade apresentaram suas necessidades de melhoria para o sistema, considerando tanto os aspectos da funcionalidade quanto os aspectos de usabilidade.

Para facilitar a compreensão de todos a respeito da necessidade da melhoria em questão, o *product owner* de usabilidade apresentou *wireframes* com a nova disposição de campos nas telas envolvidas na solução, conforme representado pelas figuras 13 e 14 a seguir:



The wireframe shows a form titled "[1465] FINANCEIRO :: Lembrete". It contains several input fields and dropdown menus. A red box highlights a new field labeled "Aluno:" with a red "1" next to it. Below the form, there is a legend for the red "1" indicating a new optional field for student information.

* Tipo: <input type="text" value="Genérico"/>		* Situação: <input type="text" value="Aberto"/>	
* Usuário: <input type="text" value="H01"/>	* Data		<input type="text"/>
		Hora: <input type="text"/>	
* Assunto: <input type="text"/>			
1 Aluno: <input type="text"/>			
* Mensagem: <input type="text"/>			
* Recorrência: <input type="text" value="Única"/>		Início: <input type="text"/>	
Auditoria:			
Cadastro: <input type="text"/>		<input type="text"/>	
Alteração: <input type="text"/>		<input type="text"/>	

1. Novo campo para informar o código/nome do aluno. Este campo terá preenchimento opcional e poderá ser utilizado como filtro na tela de registro de cobrança.

Figura 13: Wireframe da tela de Lembrete

[1452] FINANCEIRO :: Registro de Cobrança

Filtro de pesquisa

Período:
Até:
Nº. dias em atraso:

Aluno/Matrícula:

CPF:

1
Lembretes
2
Inadimplentes

3 Lembretes: [x]
4 Inadimplentes: [x]

Data/Hora	Assunto	Atraso	Aluno	Série	Turma	Aluno	Lembrete	Valor
20/04/2010	Retornar ligação	10 dias	Paulo Cesar Mattos	3º Ano	AM	Adriane Valério	5	1.100,00
21/04/2010	Ficou de ligar	10 dias	Ana Paula Silva	2º Ano	BM	Adriano Souza Júnior	2	300,00
25/04/2010	Ligar	8 dias	Karina M. de Mello	Infantil II	CM	Ailton Teixeira		200,00
27/04/2010	Retornar ligação	5 dias	Ana Julia Medeiros	9º Ano	AT	Alana Cordeiro	3	350,00
02/05/2010	Retornar ligação	3 dias	João Carlos Santos	8º Ano	BT	Alexandre da Silva		150,00
04/05/2010	Retornar ligação	0 dias	Camila Batista	7º Ano	CT	Aline Zanatta	1	400,00
10/05/2010	Retornar ligação	0 dias	Pedro Simões	3º Ano	NA	Allison Bazani	1	650,00

1. Novo botão para acessar a janela flutuante (3) com a lista de lembretes.

2. Novo botão para acessar a janela flutuante (4) com a lista de inadimplentes.

3. Janela flutuante com a lista de lembretes. Ao clicar sobre uma linha da lista, esta deve ficar com cor diferenciada e os dados do item selecionado devem ser preenchidos automaticamente na tela respectiva.

4. Janela flutuante com a lista de inadimplentes. Ao clicar sobre uma linha da lista, esta deve ficar com cor diferenciada e os dados do item selecionado devem ser preenchidos automaticamente na tela respectiva.

Figura 14: Wireframe da tela de Registro de Cobrança

A partir da apresentação dos *product owners*, o time identificou como implementar a solicitação dos mesmos, incluindo no *sprint backlog* quais tarefas seriam desenvolvidas para compor a entrega da funcionalidade requerida para essa *sprint*, conforme ilustra a figura 15 a seguir:

Sprint Backlog	
Data Início:	05/04/2010
Data Fim:	30/04/2010

Tarefas		
Descrição	Estimativa	Prioridade
Inserir novo campo na tela de Lembrete		
Inserir dois novos botões na tela de Registro de cobrança		
Criar duas janelas flutuantes vinculadas aos dois novos botões na tela de Registro de cobrança		
Implementar as funcionalidades das duas novas janelas flutuantes.		

Figura 15: Sprint Backlog

Com as alterações propostas pelos *product owners*, serão resolvidos os três problemas apontados pelos usuários, isto é:

a) Não haverá mais necessidade de imprimir os relatórios de cobrança, já que a relação aparecerá em tela;

b) O controle dos contatos será efetuado de forma visual, por meio da cor que será diferente para os links já acessados, e;

c) Os dados serão carregados em tela automaticamente quando da seleção dos mesmos, com isso, o usuário não terá que informar os dados para execução da pesquisa, o que melhorará o tempo de execução da tarefa e também diminuirá o número de cliques na tela.

3.2.3. Desenvolvimento

A partir do dia 06/04/2010, data em que iniciou o desenvolvimento, o time trabalhou com dedicação de 4 (quatro) horas diárias na implementação das alterações durante o tempo de duração da sprint (20 dias). Por tratar-se de um escopo reduzido de tarefas, tempo e recursos, a reunião diária proposta pelo Scrum foi realizada apenas uma vez durante o desenvolvimento, pois o time era composto apenas por um programador e um analista de qualidade, tornando a socialização das tarefas desnecessária.

A reunião citada foi realizada em função de um problema encontrado nos requisitos da alteração, ou seja, durante a reunião de definição da *sprint* foi definido que um novo campo seria acrescentado à tela de registro de lembrete, para cadastro do código e nome do aluno, conforme ilustrou a figura 13. Entretanto, durante o desenvolvimento da alteração, foi constatado que a tela de registro de cobrança já possuía em uma aba interna, opção para registro de lembrete, conforme ilustra a figura 16 a seguir, e este lembrete, por ser registrado nesta aba interna, já armazenava a referência do aluno, não sendo necessário, portanto, implementar a alteração proposta na figura 13.

[1452] FINANCEIRO :: Registro de Cobrança

Filtro de pesquisa

Período: Até: Nº. dias em atraso:

Aluno/Matricula: 0000 0000

CPF:

Editar	Aluno	Fone 1	Fone 2	Pai	Fone 1	Fone 2	Mãe
<input checked="" type="checkbox"/>							

Renegociação | Caixa (fechado) | Consulta Genérica

Dívida Registro Histórico

Divida Total: R\$ 77,45 Parcelas Selecionadas: R\$ 77,45

Cobrança:

* Data: *

* Descrição:

Lembrete:

Data: Assunto: Descrição:

Figura 16: Aba Interna - Tela de Registro de Cobrança

Resolvida esta questão, partiu-se para o desenvolvimento das demais alterações tratadas pela *sprint*, e todas estas ocorreram conforme previsto inicialmente. Assim, a cada tarefa completada do *sprint backlog* o analista de qualidade realizava os testes funcionais para validação do que foi implementado.

Após a conclusão de todas as tarefas, o teste da funcionalidade completa foi realizado por representantes dos usuários conforme sugere a metodologia U-SCRUM a fim de avaliar e validar o resultado. Estes testes foram positivos e de acordo com informações repassadas, as alterações

efetuadas foram satisfatórias, e não geraram nenhuma sugestão para ser incluída nas próximas *sprints*.

3.2.4. Entrega da Sprint

Ao concluir o desenvolvimento da sprint, no dia 29/04/2010, foi realizada a reunião de Revisão da *sprint* com o objetivo de apresentar aos *product owners* o resultado do desenvolvimento, para que os mesmos pudessem avaliar se estava de acordo com o solicitado e se contemplava todas as solicitações.

Assim, salvo o impasse relatado acima, o resultado obtido ficou dentro do esperado. Todos os itens da *sprint* foram desenvolvidos e a entrega da mesma foi entregue dentro do cronograma previsto.

3.3. Sprint 2: Controle de Entrega

3.3.1. Descrição do problema

Anualmente as instituições de ensino necessitam entregar aos alunos uma série de itens, a exemplo: material didático e carteira estudantil. Por questões econômicas e também de segurança, a entrega desses itens deve ser controlada e registrada de forma a não permitir a entrega de mais de um item para o mesmo aluno.

Atualmente o sistema já dispõe de uma solução que atende este quesito, todavia, surgiu uma nova necessidade, que se refere justamente ao contrário do que foi mencionado no item anterior, ou seja, algumas instituições necessitam entregar o mesmo item, mais de uma vez a um mesmo aluno, e em alguns casos, mais de uma vez em um mesmo dia, como por exemplo, entrega de lanches. Neste caso específico de entrega de lanches, algumas instituições públicas, participam de projetos em que o governo faz o ressarcimento do valor gasto pela instituição na compra dos lanches, e este controle é repassado a Instituição com base no número de lanches entregues aos alunos. Este quesito

tornou-se então uma necessidade para o cliente e em função disto, uma nova *sprint* foi definida, a qual é apresentada na próxima seção.

3.3.2. Definição da Sprint

Após a conclusão da primeira *sprint*, o desenvolvimento da *release* seguiu com a segunda para atender à próxima melhoria do sistema listada no *product backlog*: “Alteração no Controle de Entrega”.

A definição do *sprint backlog* ocorreu na reunião de planejamento da *sprint*, onde os dois *product owners* do projeto apresentaram quais as melhorias necessárias para solucionar o problema comentado acima.

A solução sugerida foi incluir na tela Tipo de Entrega um novo campo chamado ‘Tipo’, contendo as opções:

- a) Entrega Única: para tipos de entregas efetuadas somente uma vez;
- b) Diária (Única): para tipos de entregas efetuadas uma vez ao dia;
- c) Diária (Várias): para tipos de entregas efetuadas várias vezes ao dia.

A inclusão do novo campo implicou na alteração de outra tela do sistema chamada “Controle de Entrega - automático”, conforme ilustrado a seguir na figura 17 por meio de *wireframes*:

CADASTRO :: Tipo de Entrega :: Edição

Código: 16

* Nome: Lanche Manhã

Evento Financeiro: [dropdown]

* Controlar Saldo: Não [dropdown]

* Qtde Saldo Atual: 50,000

Código de Barras: [input]

1 * Tipo: Única [dropdown]
Diária (Única)
Diária (Várias)

1. Novo campo para indicação do tipo de entrega.

Figura 17: *Wireframe* com alteração da tela Tipo de Entrega

demais campos, o campo Data Entrega com a data atual como valor padrão, pois como a entrega é diária, não há necessidade de preencher individualmente esse valor para cada aluno, como mostra a figura 19 abaixo:

ESCOLAR :: Controle de Entrega - Automático

* Órgão Regulador: ASP Escola
* Regional: 01 - Florianópolis
* Instituição: Escola Teste
* Curso: Curso Homero
* Série: 220 1º Ano
* Ano Exercício: 2008 Turma:
Aluno/Matrícula:
* Tipo de Entrega: Lanche Manhã
1 * Data Entrega: 07/04/2010
* Tipo : Todos Entregues Não Entregues

<input checked="" type="checkbox"/>	Código	Nome
<input type="checkbox"/>	2962	Alunoteste
<input type="checkbox"/>	2899	Cesar Souza jr.
<input type="checkbox"/>	2897	Gilca Amancio

1. Campo para filtrar a data da entrega a ser realizada.

Figura 19: Wireframe tela Controle de Entrega Diária

No caso de já haver entregas, poderá também ser efetuada a re-entrega do item, sendo que, neste caso a tela habilitará os campos Re-entrega, Data de Re-entrega e Motivo, conforme ilustrado acima na figura 19.

c) Várias Entregas por dia: Quando o Tipo de Entrega selecionado na tela estiver cadastrado como 'Diária (Várias)', para fins de controle serão exibidos além dos demais campos, os campos Data Entrega e Nova Entrega. O campo Nova Entrega servirá como filtro para indicar se o cadastro é uma nova entrega ou não.

Quando o campo Nova Entrega estiver selecionado com a opção 'Não', serão exibidos todos os alunos que já possuem registro de entrega cadastrado, permitindo o cadastro de re-entrega, conforme ilustra a figura 20 .

ESCOLAR :: Controle de Entrega - Automático

* Órgão Regulador: ASP Escola

* Regional: 01 - Florianópolis

* Instituição: Escola Teste

* Curso: Curso Homero

* Série: 220 1º Ano

* Ano Exercício: 2008 Turma: A

Aluno/Matrícula: [] [] [] []

* Tipo de Entrega: Lanche Manhã

* Data Entrega: 07/04/2010 **1** * Nova Entrega: Sim Não

* Tipo : Todos Entregues Não Entregues

<input checked="" type="checkbox"/>	Código	Nome	Turma	Reent	Dt. Reent.	Motivo Reent.
<input checked="" type="checkbox"/>	1 2962	Alunoteste	A	<input checked="" type="checkbox"/>	07/04/2010	
<input type="checkbox"/>	1 2899	Cesar Souza jr	A	<input type="checkbox"/>		
<input type="checkbox"/>	1 2897	Gilda Amancio	A	<input type="checkbox"/>		

1. Novo campo para filtrar se o cadastro será uma nova entrega ou não. Quando for Não será possível cadastrar re-entrega.

Figura 20: Wireframe tela Controle de Entrega – Várias

Quando o campo Nova Entrega estiver selecionado com a opção Sim, serão exibidos todos os alunos (com ou sem entregas já cadastradas) para cadastro de novas entregas. A tela deverá ser disponibilizada conforme ilustra a figura 21.

3.4. Sprint 3: Emissão de bloqu岸os de Adotadores

3.4.1. Descrição do problema

Conforme citado no item 3.2.1 da *Sprint 1*, cada matrícula de aluno possui um responsável acadêmico e também um responsável financeiro, e estes responsáveis podem ser: aluno, pai, mãe, terceiro ou um adotador.

No caso de ser aluno, pai, mãe ou terceiro, a relação com o aluno é relativamente clara; em relação ao adotador são necessários alguns esclarecimentos para compreensão do vínculo, já que esta figura é um tanto quanto nova nas instituições de ensino.

Esta figura surgiu em função de um programa de bolsas de estudo, criado para propiciar a crianças de baixa renda a oportunidade de estudarem em instituições de ensino particulares. Este programa teve início com a colaboração de instituições privadas, empresários e pessoas que se dispuseram individualmente a "adotar" um ou mais alunos, subvencionando parte ou a totalidade de suas despesas escolares.

Ao adotar um aluno, o adotador pode por meio do portal do aluno, consultar seu desempenho acadêmico, sua frequência, seu boletim, histórico, entre outros. A consulta é importante uma vez que pode influenciar na decisão do mesmo de continuar ou não com o adote.

Por meio do portal, o adotador também pode consultar seu extrato financeiro, que contém relação e demais dados de todas as parcelas em que o mesmo é responsável, e ainda, caso deseje, pode emitir mensalmente o bloqu岸o para pagamento de suas parcelas.

Caso o adotador "adote" mais de um aluno, a consulta aos seus dados acadêmicos pode ser efetuada por meio de um único *login*, todavia, a emissão de bloqu岸os não pode ser efetuada desta forma, ou seja, deve ser gerado, mensalmente, um bloqu岸o individual para cada aluno adotado.

Para os casos em que o adotador adota um ou dois alunos, isto não é tão problemático, todavia, caso o adotador seja responsável financeiramente

por dez alunos ou mais, o fato de ter que gerar dez ou mais bloquetes de pagamento, mesmo sendo a mesma instituição de ensino, tem gerado insatisfação.

Como o adote é uma espécie de caridade, fatores que desagradam os adotadores pode levá-los a desistir ou cancelar a “adoção”, o que é um fator crítico para as instituições de ensino, pois pode representar a perda do recebimento de parcelas de um ou mais alunos.

3.4.2. Definição da Sprint

A terceira sprint estava programada para iniciar em 01/06/2010, todavia não foi possível iniciá-la a tempo, em função do período de início e término da mesma, com isso a definição da solução para o problema listado acima, não será apresentada neste trabalho.

3.5. Resultados

3.5.1. Desenvolvimento sem processo

É fato conhecido que muitas empresas desenvolvem software sem utilizar nenhum processo, uma vez que, muitos dos processos tradicionais normalmente não são adequados a realidade da mesma, ou mesmo por não possuir recursos suficientes para adotar/implantar estes processos.

O caso da Gennera é semelhante, ou seja, nenhum processo formal é utilizado, e mesmo com a utilização de um processo interno (espécie de fluxo de trabalho), durante o acompanhamento efetuado, foi possível perceber que a falta de sistematização na produção/desenvolvimento das customizações implicou na redução da qualidade final do produto, e ainda dificultou a entrega nos prazos e custos previamente definidos.

3.5.2. Desenvolvimento utilizando scrum

O grande ganho na utilização desta metodologia é o enfoque nas pessoas ao invés de processos ou algoritmos especificamente, além de agregar a preocupação de gastar mais tempo com a implementação do que com a documentação.

Outra característica importante desta metodologia é a adaptabilidade, ou seja, ao invés de procurar analisar tudo o que pode acontecer no decorrer do desenvolvimento, ela se adapta às mudanças, e para aplicações web, como é o caso da Gennera, isto é muito importante em função de o ambiente ser bastante dinâmico.

Outro ponto bastante favorável é a questão de entregas constantes de partes operacionais, que no caso desta *release*, foi dividida em três partes, assim o cliente não precisará esperar muito para ver o resultado das suas solicitações.

E por fim, tem-se ainda a questão dos testes contínuos, que também possibilitam de maneira significativa a melhora na qualidade do software.

3.5.3. Desenvolvimento com técnicas de Usabilidade

A utilização de testes de usabilidade pode diminuir os custos do desenvolvimento do sistema, por meio da identificação e resolução de questões problemáticas e da adição das necessidades e desejos dos utilizadores antes da liberação para o cliente. Também pode reduzir os custos de manutenção e suporte, uma vez que, produtos com bom nível de usabilidade, são mais fáceis de usar e de aprender, implicando, portanto, em economia de custos com ensino e suporte.

Durante a aplicação deste projeto, ainda nas fases iniciais do ciclo de vida de desenvolvimento foi possível incluir alguns critérios de usabilidade, e após a entrega de parte do projeto ao cliente (*Sprint 1*), percebeu-se os seguintes benefícios:

a) Condução - Agrupamento/distinção de itens: Foram inseridos novos campos em algumas telas. Para facilitar o trabalho do usuário, os mesmos são exibidos de acordo com o tipo de cadastro que será efetuado, como por exemplo, o campo 'Data de Re-entrega', que é exibido somente nos casos em que já houve o cadastro de um registro de entrega.

b) Carga de Trabalho: No caso de 'Entregas Diárias', a data de entrega é informada na parte superior da tela como parâmetro, e esta data é válida para todos os registros; assim, não é necessário que o usuário a informe para todos os alunos.

c) Gestão de Erros: No caso do 'Controle de Entregas', o usuário poderá remover o registro incorreto de uma entrega cadastrada, simplesmente desmarcando o campo '*Chek box*' da tela. Por segurança, também será disponibilizado um relatório de log, para registrar estas movimentações.

CONCLUSÃO

A competitividade do mercado e a facilidade de acesso às informações por parte do cliente fazem com que a permanência da empresa e seu sucesso estejam diretamente ligados a possibilidade de atração e retenção dos clientes. Essa retenção é realizada por meio de um bom atendimento, excelência em produtos e serviços, alto valor agregado para o cliente e velocidade na resposta às demandas do mercado.

Com este trabalho buscou-se destacar a relevância da incorporação de técnicas de usabilidade nos ciclos do processo de uma metodologia de desenvolvimento ágil, com o objetivo de aumentar a qualidade dos produtos, mantendo a produtividade.

Como o produto final está diretamente ligado ao processo utilizado em seu desenvolvimento e manutenção, a aplicação de conceitos de melhoria da qualidade permite aumentar as condições de sucesso dos projetos, minorando os riscos e facilitando o seu monitoramento.

Para a execução do processo sugerido neste trabalho deve existir o apoio e investimento da alta gerência da empresa, motivando os envolvidos para a realização correta das atividades, visto que, além das técnicas para verificação de usabilidade, também é necessário seguir uma metodologia de desenvolvimento.

Com a experiência obtida através da execução do estudo de caso, comprovou-se que com a alteração do processo de desenvolvimento, foi possível potencializar o trabalho realizado, e a inclusão de técnicas de usabilidade não implicou no aumento do prazo para desenvolvimento, pelo contrário, a visão voltada para a usabilidade ajudou a direcionar a melhor solução para os problemas, e o fato de a empresa necessitar de um profissional exclusivo para tratar de aspectos de usabilidade certamente será compensada com o ganho em produtividade e qualidade.

A maioria das empresas ainda investe pouco em usabilidade, entretanto, entende-se que essas atividades podem evoluir de forma gradativa,

acumulativa e paralela ao processo de desenvolvimento. E este é um grande desafio, visto que, em alguns momentos este processo competirá dentro da empresa com atividades que visam meramente a produtividade. Nestes casos, mesmo que o resultado final seja positivo, o fato de não seguir nenhum processo, pode implicar a afetar diretamente na qualidade do produto.

Procurou-se ressaltar também com este trabalho, a importância da incorporação das técnicas de usabilidade ainda no início dos projetos da empresa, pois isto pode evitar re-trabalho através da identificação precoce de possíveis problemas, o que implica na redução de custos e prazos. Mediante a execução das atividades seguindo os padrões e metodologia sugerida, foi possível identificar também a existência de requisitos incompletos e faltantes, o que possibilitou intensificar a confiabilidade do projeto, com posterior aceitação e satisfação do cliente.

A experiência prática realizada nos permitiu concluir, de acordo com os objetivos gerais e específicos deste projeto, que a realização de cada item do projeto, dentro do processo de desenvolvimento proposto, diminui os defeitos do produto, uma vez que este é realizado com a interação de todos os envolvidos. O detalhamento das evidências encontradas amadureceu o entendimento do processo definido, contribuindo para um maior conhecimento e possibilitando uma futura melhora na qualidade do sistema. A realização das atividades relacionadas à usabilidade desde o início do processo de desenvolvimento contribuiu para que não-conformidades fossem identificadas logo nas suas ocorrências, diminuindo o re-trabalho e os custos necessários para a sua correção.

O planejamento e incorporação de técnicas de usabilidade, durante as fases de desenvolvimento do sistema, também podem proporcionar mudança de conceitos e paradigmas na empresa, uma vez que todos os envolvidos no processo passam a se preocupar com a qualidade do produto.

Também com as reuniões efetuadas ao final de cada projeto, em que são analisadas as evidências coletadas durante o desenvolvimento de cada produto, possibilita a identificação de melhorias no processo, que poderão ser

utilizadas no desenvolvimento dos próximos produtos, reutilizando os itens de sucesso e a experiência adquirida.

SUGESTÕES DE TRABALHOS FUTUROS

Como trabalho futuro, a sugestão seria aplicar métodos de padronização em todas as etapas de desenvolvimento do produto, que englobam o Planejamento, Desenvolvimento, Entrega e Manutenção. Este assunto surgiu, visto que além de boas técnicas de desenvolvimento, é necessário obter a participação e o comprometimento dos colaboradores.

Como segunda opção, propõe-se a aplicação de técnicas e critérios de usabilidade também no desenvolvimento, mas no sentido de verificação e organização da codificação do sistema. Neste contexto, as heurísticas de Nielsen [25] poderiam ser adaptadas, conforme descrito a seguir:

Critério 1. Visibilidade do estado do sistema – Ao se ler um código fonte, é importante saber o que está acontecendo.

Critério 2. Consistência e padrões – Padronização de código (nome de métodos, etc.) e arquitetura.

Critério 3. Prevenção de erros – Se existiu um erro, o mesmo deve ser comunicado, assim como a sua solução.

Critério 4. Reconhecimento ao invés de lembrança – Intuitividade do código.

Critério 5. Flexibilidade e eficiência de uso – Utilização, por exemplo, de *frameworks* que otimizem a codificação de funções.

Critério 6. Estética e design minimalista – Codificação clara e padronizada.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] PRIBERAM, Dicionário Priberam da Língua Portuguesa. Disponível em: <<http://www.priberam.pt/DLPO/Default.aspx>>. Acesso em 22 de novembro de 2009.
- [2] ISO DIS 8402, *Quality Vocabulary*, 1994.
- [3] LOBO, Alfredo. Qualidade e Produtividade – Obra Intelectual. Disponível em: <http://www.inmetro.gov.br/producao intelectual/obras_intelectuais/109_obraIntelectual.pdf>. Acessado em: 11 de abril de 2010.
- [4] Pressman, R. Engenharia de Software. McGraw-Hill, 2001.
- [5] [6] KANO, N.; SERAKU, N.; TAKAHASHI, F.; TSUJI, S. – Attractive Quality and Must-Be. Vol. 14, pp.39-48, 1984.
- [6] KANO, N.; SERAKU, N.; TAKAHASHI, F.; TSUJI, S. – *Attractive Quality and Must-Be*. Acesso em: 13 de dezembro de 2009.
- [7] CYBIS, Walter; BETIOL, Adriana Holtz; FAUST, Richard. *Ergonomia e Usabilidade: Conhecimentos, Métodos e Aplicações*. São Paulo : Novatec Editora, 2007.
- [8] SOMMERVILLE, Ian. *Engenharia de Software*. 8 ed. São Paulo: Pearson Addison-Wesley, 2007.
- [9] MAYHEW, Deborah J. The usability engineering lifecycle: a practitioner's handbook for user interface design. San Francisco: Morgan Kaufmann, 1999.
- [10] ISO 13407 e ISO TR 18529. Disponível em: <<http://www.usabilitynet.org/tools/13407stds.htm>>. Acesso em: 13 de dezembro de 2009.
- [11] MANIFESTO for Agile Software Development. Disponível em <<http://www.agilemanifesto.org>>. Acesso em 10 de outubro de 2009.
- [12] SCHWABER, ken; SUTHERLAND, Jeff. *Scrum Guide*. 2009. Disponível em: <<http://www.scrum.org/storage/scrumguides/Scrum%20Guide.pdf#view=fit>>. Acesso em: 05 de dezembro de 2009.
- [13] Ciclo proposto pelo Scrum. Disponível em: <www.imporveit.com.br/br/scrums/index>. Acesso em: 05 de dezembro de 2009.
- [14] BECK, Kent; trad. MACHADO, Adriana P.S.; LOPES, Natália N.P.. *Programação Extrema (XP) explicada*. Acolha as mudanças. Porto Alegre: Bookman, 2004.
- [15] NIELSEN, Jakob. *Agile Development Projects and Usability*. Jakob Nielsen's Alertbox, 2008. Disponível em <<http://www.useit.com/alertbox/agile-methods.html>>. Acesso em 07 de novembro de 2009.
- [16] CONSTANTINE, Larry L. *Process Agility and Software Usability: Toward Lightweight Usage-Centered Design*. Constantine & Lockwood, Ltd. Disponível em <<http://www.foruse.com/articles/agiledesign.pdf>>. Acesso em 10 de janeiro de 2010.
- [17] Brainstorming. Disponível em <fabioap.files.wordpress.com/2007/11/brainstorming.doc>. Acesso em 21 de abril de 2010.

- [18] MEMMEL, Thomas; GUNDELSWEILER, Fredrik; REITERER, Harald. *CRUISER: A Cross-Discipline User Interface and Software Engineering Lifecycle*. University of Konstanz, Germany, 2007. Disponível em <http://hci.uni-konstanz.de/downloads/HCI_Int_2007_AgileUsability_Cruiser.pdf>. Acesso em 12 de janeiro de 2010.
- [19] Feedback. Disponível em <http://www.rh.com.br/Portal/Comunicacao/Blog_Alberto_Ruggiero/5809/voce-sabed-dar-feedback.html>. Acesso em 21 de abril de 2010.
- [20] VASCONCELOS, Cesar Rocha. *XPU – Um Modelo Para o Desenvolvimento de Sistemas Centrado no Usuário*. Dissertação de Mestrado, Centro de Ciência e Tecnologia, UFCG, Campina Grande, Paraíba, 2004.
- [21] SINGH, Mona. *U-SCRUM: An Agile Methodology for Promoting Usability*. In *Agile 2008 Conference*, p.555. IEEE Computer Society, 2008.
- [22] Workshop. Disponível em <<http://pt.wikipedia.org/wiki/Workshop>>. Acesso em 21 de abril de 2010.
- [23] Neves, Ana. Storyboard - O que é? Para que serve? Disponível em <<http://mestrediarario.blogspot.com/2007/05/guio-de-autor-o-que-para-que-serve.html>>. Acesso em 21 de abril de 2010.
- [24] Pereira, Ana Paula Sedrez de Souza. O que é Wireframe? Disponível em <<http://www.baixaki.com.br/info/976-o-que-e-wireframe-.htm>>. Acesso em 21 de abril de 2010.
- [25] Heurísticas de Nielsen. Disponível em <http://pt.wikipedia.org/wiki/Heur%C3%ADsticas_de_Nielsen>. Acesso em 21 de abril de 2010.
- [26] COTTMEYER, Mike. The Agile Project Manager. Disponível em <http://pm.versionone.com/whitepaper_AgilePM.html>. Acesso em: 20 de fevereiro de 2010.
- [27] CYBIS, Walter. Desenvolvimento ágil e engenharia de usabilidade. Disponível em <http://imasters.uol.com.br/artigo/12299/usabilidade/desenvolvimento_agil_e_engenharia_de_usabilidade/>. Acesso em: 13 de dezembro de 2009.