

**CENTRO PAULA SOUZA  
FACULDADE DE TECNOLOGIA DE JAHU  
TECNÓLOGO EM GESTÃO DA TECNOLOGIA DA INFORMAÇÃO**

**IGUES DE LACERDA  
LEONARDO CRISTIANO DE ALICE**

**COMBINANDO METODOLOGIAS ÁGEIS E FERRAMENTAS  
CASE PARA AUMENTAR A PRODUTIVIDADE NO  
DESENVOLVIMENTO DE SOFTWARE**

**Jaú, SP  
2º Semestre/2013**

**IGUES DE LACERDA  
LEONARDO CRISTIANO DE ALICE**

**COMBINANDO METODOLOGIAS ÁGEIS E FERRAMENTAS CASE PARA  
AUMENTAR A PRODUTIVIDADE NO DESENVOLVIMENTO DE SOFTWARE**

Trabalho de Conclusão de Curso  
apresentado à Faculdade de  
Tecnologia de Jahu, como parte  
dos requisitos para a obtenção do  
título de Tecnólogo em Gestão da  
Tecnologia da Informação.

Orientador: Prof. Ms. **Wdson de Oliveira**

**Jaú, SP  
2º Semestre/2013**

## DEDICATÓRIA

Dedicamos esse trabalho aos  
nossos familiares e companheiros,  
pessoas sem as quais nada disso  
teria sentido.

## **AGRADECIMENTOS**

Agradecemos a Deus pelo dom da vida.

Aos professores da Fatec Jahu que durante seis meses incansavelmente se doaram para nos transmitir um pouco de conhecimento, em especial, agradecemos ao professor e mestre Wdson de Oliveira, por nos orientar nessa caminhada árdua rumo à conclusão de curso.

Agradecemos também, todos os funcionários da Fatec Jahu por cada atitude colaborativa para nós alunos.

Agradecemos o apoio de nossos familiares e companheiras, os quais sempre estiveram juntos de nós, dando força para que pudéssemos seguir firmes.

Os nossos sinceros agradecimentos as empresas que colaboraram com esse trabalho, representadas pelas pessoas entrevistadas ao longo desse semestre.

Por fim, agradecemos todas as pessoas que não foram citadas aqui, mas que contribuíram para que esse trabalho fosse concluído.

Gostaríamos de deixar um pensamento de Chico Xavier: “Agradeço todas as dificuldades que enfrentei; não fosse por elas, eu não teria saído do lugar. As facilidades nos impedem de caminhar. Mesmo as críticas nos auxiliam muito.”

## SUMÁRIO

<b>1 – INTRODUÇÃO .....</b>	<b>9</b>
1.1 – SITUAÇÃO ATUAL DO MERCADO DE TIC .....	9
1.2 – A PROJEÇÃO DO MERCADO DE TIC.....	10
1.3 – O PROBLEMA .....	11
1.4 – O OBJETIVO.....	11
<b>2 – REVISÃO BIBLIOGRÁFICA.....</b>	<b>12</b>
2.1 – AS METODOLOGIAS ÁGEIS .....	12
2.1.1 – O manifesto para o desenvolvimento ágil de software .....	14
2.1.2 – O framework Scrum.....	15
2.1.2.1 – Visão geral do Scrum.....	16
2.1.2.2 – Scrum e seus fundamentos.....	17
2.1.2.3 – O Time Scrum .....	18
2.1.2.4 – A Sprint.....	21
2.1.2.5 – Artefatos do Scrum .....	25
2.2 – FERRAMENTAS CASE .....	28
2.2.1 – Introdução .....	28
2.2.2 – O que é uma ferramenta CASE .....	29
2.2.3 – Taxonomia das ferramentas CASE .....	30
2.2.4 – Vantagens e desvantagens sobre ferramentas CASE.....	33
<b>3 – METODOLOGIA .....</b>	<b>35</b>
3.1 – ANÁLISE DE RESULTADOS.....	36
3.1.1 – Análise de uso das metodologias .....	36
3.1.2 – Análise de uso das ferramentas CASE .....	37
3.1.3 – Análise da combinação de metodologias e ferramentas CASE .....	38
3.2 – EXEMPLO DE UTILIZAÇÃO DE UMA FERRAMENTA CASE.....	39
<b>4 – CONCLUSÃO .....</b>	<b>47</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS.....</b>	<b>48</b>
<b>APÊNDICE A – Questionário para coleta de dados.....</b>	<b>51</b>

## RESUMO

Neste trabalho estão reunidas informações que indicam a atual situação e tendências no mercado de TI em relação à demanda de mão de obra. Tais dados darão base à análise sobre se essa demanda é atendida e se a mão de obra disponível é qualificada para exercer as atividades de desenvolvimento de software no mercado de Tecnologia da Informação. A problemática apresentada por essa análise é o embasamento para a proposta e objetivo desse trabalho: metodologias ágeis de desenvolvimento combinado com softwares que são ferramentas de apoio. Na proposta, que consiste em dois conteúdos combinados, é apresentada a metodologia ágil Scrum e toda sua fundamentação baseada em materiais de fontes diretas, e em seguida definições dos softwares que são sugeridos como as ferramentas de apoio para as atividades relacionadas à metodologia. Para validar a proposta temos o formulário de questões que abordam como os entrevistados adotaram a metodologia ágil, quais foram as dificuldades e os ganhos no processo, e como exploram as ferramentas de apoio, se são acomodadas às rotinas do processo. O formulário de questões orienta uma pesquisa qualitativa que resulta em uma análise de como os conteúdos teóricos apresentados no trabalho ocorrem no ambiente real.

Palavras chave: Metodologia Ágil; Scrum; Ferramentas CASE; Desenvolvimento de Software.

## ABSTRACT

*In this work are a collection of data that shows the status and the trends of the IT market when it comes to work force demand. Such data is used to analyze if this demand is well filled and if these available work force aware of the development activities on the technology of information market. The problem shown in this research brings the proposal, which is the objective of this work: agile development methodologies combined to support tool softwares. On the proposal which is built of two combined contents, is presented the Scrum agile methodology system with its full foundation based on data of original sources, followed by the the definitions of softwares that are suggested as tools to support the activities related to the methodology. We have the questions form to validade the proposal. this form concerns on how did the interviewed adopted the agile methodology, which was its difficulties and gains gotten on the process and how do they explores the support tools, if it fits the process routines. The question form drives a qualitative research that results on an analisys about how the theoric content shown in this work happens in a real environment.*

Keywords: Agile Methodologies; Scrum; CASE Tools; Software Development.

## LISTA DE FIGURAS

<b>Figura 1 – Histórico das metodologias de desenvolvimento de software .....</b>	<b>14</b>
<b>Figura 2 – Visão geral do Scrum.....</b>	<b>18</b>
<b>Figura 3 – O ciclo da Sprint.....</b>	<b>23</b>
<b>Figura 4 – As três variáveis de cada solicitação do Backlog .....</b>	<b>24</b>
<b>Figura 5 – Quadro de tarefas Scrum.....</b>	<b>26</b>
<b>Figura 6 – Criação da tabela “produto” no banco de dados PostgreSQL.....</b>	<b>43</b>
<b>Figura 7 – Tela inicial do Maker .....</b>	<b>44</b>
<b>Figura 8 – Assistente de criação de projetos Maker .....</b>	<b>45</b>
<b>Figura 9 – Tela principal do projeto Maker.....</b>	<b>46</b>
<b>Figura 10 – Assistente de criação de formulário Maker .....</b>	<b>47</b>
<b>Figura 11 – Editor de formulário Maker .....</b>	<b>47</b>
<b>Figura 12 – Criação de menu Maker .....</b>	<b>48</b>
<b>Figura 13 – Serviço Webrun Maker .....</b>	<b>48</b>
<b>Figura 14 – Login no projeto Maker .....</b>	<b>49</b>
<b>Figura 15 – Tela inicial do programa resultado do projeto Maker .....</b>	<b>49</b>
<b>Figura 16 – Tela de cadastro criada no Maker .....</b>	<b>50</b>
<b>Figura 17 – Tela de filtro do cadastro criado no Maker .....</b>	<b>50</b>



## 1 – INTRODUÇÃO

### 1.1 – SITUAÇÃO ATUAL DO MERCADO DE TIC

Não raramente é possível ler manchetes como “Déficit de profissionais de tecnologia se aprofunda no País”, “Falta de profissionais de TI se agravará no Brasil, diz IDC”, “O déficit de mão de obra em TI”, “País tem déficit de 115 mil especialistas em TI”, entre outras derivações; todas elas retratam a escassez de Recursos Humanos para o Mercado de Tecnologia da Informação e Comunicação (TIC).

Giuseppe Marrara (2013), diretor de Relações Governamentais da Cisco do Brasil, afirma:

"As oportunidades na área de tecnologia da informação e comunicação no Brasil estão aumentando significativamente com a preparação do País para sediar grandes eventos, como Copa do Mundo e Olimpíadas. A falta de mão de obra qualificada ainda é um fator preocupante para atender a esta demanda e ainda para que o Brasil possa competir mais efetivamente no mercado mundial".

Para agravar a situação, nosso país não tem uma Educação sólida, o que é comprovado ao analisar o Ranking do PISA (*Program for International Student Assessment - Programa para Avaliação Internacional de Estudantes*), no qual em 2009 o Brasil ficou na 57ª posição em Matemática, um Conhecimento essencial para os fundamentos das TIC (REDAÇÃO, 2010).

De acordo com o IDC (Insights Community), existe atualmente no Brasil uma carência de cerca de 39,9 mil profissionais de tecnologia. Segundo a pesquisa, as principais razões para esse déficit de mão de obra qualificada são a rápida expansão das empresas de infraestrutura e tecnologia no país, a adoção acelerada de serviços de TI pelas iniciativas públicas e privadas e a ocorrência, no Brasil, da Copa do Mundo de 2014 e das Olimpíadas de 2016, no Rio de Janeiro. Para chegar a estas conclusões, a consultoria IDC realizou 767 entrevistas com órgãos como governos, empresas de educação, saúde, telecomunicações e serviços financeiros em companhias com mais de 100 empregados (VENCESLAU, 2013).

De acordo com Laskoski (2010), a falta de mão de obra qualificada é um problema crônico e que está se agravando ao decorrer do tempo. A economia do

Brasil está crescendo e isso exige com que as empresas busquem profissionais qualificados. Laskoski, baseado em que especialistas apontam, considera que a causa mais evidente da escassez de profissionais qualificados é a discrepância entre a formação acadêmica proposta por universidades de todo o país e a realidade mercadológica

## 1.2 – A PROJEÇÃO DO MERCADO DE TIC

Como se já não bastasse todo esse cenário caótico que vive o setor de TI, as perspectivas mostram um cenário ainda pior, uma vez que a tendência é que o setor de TI cresça muito mais.

Estimativas é que até 2014 o déficit de mão de obra chegue a 800 mil profissionais na área de TI, sendo que 86% desse total serão da área de Desenvolvimento e Análise de Sistemas. Ou seja, o que já é ruim, pode ainda ficar pior (LASKOSKI, 2010).

O Estudo da Softex (Associação para Promoção da Excelência do Software Brasileiro), apresentado no dia 10 de Julho de 2012, prevê que o Setor de Software e Serviços poderá perder R\$ 115,4 bilhões de receita líquida até 2020, caso não haja investimentos para capacitação de mão de obra voltada para TICs (BERBERT, 2012).

Segundo Berbert (2012), em 2012, o Setor teve uma receita líquida de R\$ 71,6 bilhões, esse montante representa 1,8% do PIB (Produto Interno Bruto) do Brasil. Mas a grande questão é que essas empresas de Software são as grandes responsáveis, indiretamente, do sucesso de muitos outros segmentos, para não dizer todos, pois no mundo em que estamos nenhuma empresa sobrevive sem um Sistema de Gestão. Daí o maior argumento para que esse problema se torne gravíssimo.

Outra informação interessante, é que a expectativa de crescimento do Mercado de Software Brasileiro é de 400% até 2022; quem afirmou isso foi Rafael Moreira, coordenador geral de software e serviços de TI do Ministério da Ciência, Tecnologia e Inovação. Para Rafael, o crescimento será consequência do uso

intensivo de software em setores onde o País tem grande liderança ou grandes desafios socioeconômicos para enfrentar. O Ministério identificou quatro grandes áreas que serão estimuladas nos próximos anos: Desenvolvimento Social e Econômico; Posicionamento Internacional; Inovação e Empreendedorismo; e Competitividade. Esse tipo de iniciativa por parte do Governo é de grande valia para que o Brasil esteja preparado para suportar toda essa expectativa (MOURA, 2012).

### 1.3 – O PROBLEMA

Diante da situação atual do mercado de TIC e de sua projeção para o futuro, as empresas precisam se preparar para suportar as necessidades decorrentes desse novo cenário.

O Maior problema, e que continua se agravando, é a escassez de profissionais com qualificação, profissionais que assumam responsabilidades importantes, tais como o desenvolvimento de um projeto, e as consigam cumprir.

Em suma, a problemática desse trabalho gira em torno da crescente demanda de software no Brasil, contrastada pela escassez de profissionais qualificados no processo de desenvolvimento de software.

### 1.4 – O OBJETIVO

O objetivo desse trabalho é estudar a Metodologia Ágil de Desenvolvimento de Software Scrum e as Ferramentas CASE (do inglês *Computer-Aided Software Engineering*), para averiguar se a combinação de ambas, se bem aplicada, pode amenizar o iminente problema exposto no subitem 1.3, possibilitando que uma equipe aperfeiçoe suas práticas e seja mais produtiva.

## 2 – REVISÃO BIBLIOGRÁFICA

Nesse capítulo serão abordados dois assuntos, as Metodologias Ágeis no subitem 2.1, com um foco no Framework<sup>1</sup> Scrum, e as ferramentas CASE no subitem 2.2.

Baseado em algumas literaturas, os temas serão abordados de maneira objetiva, com o intuito de proporcionar uma visão geral, o suficiente para dar o embasamento necessário para a análise feita no capítulo 3; o qual irá demonstrar, na prática, o resultado da aplicação dos conceitos abordados nesse trabalho.

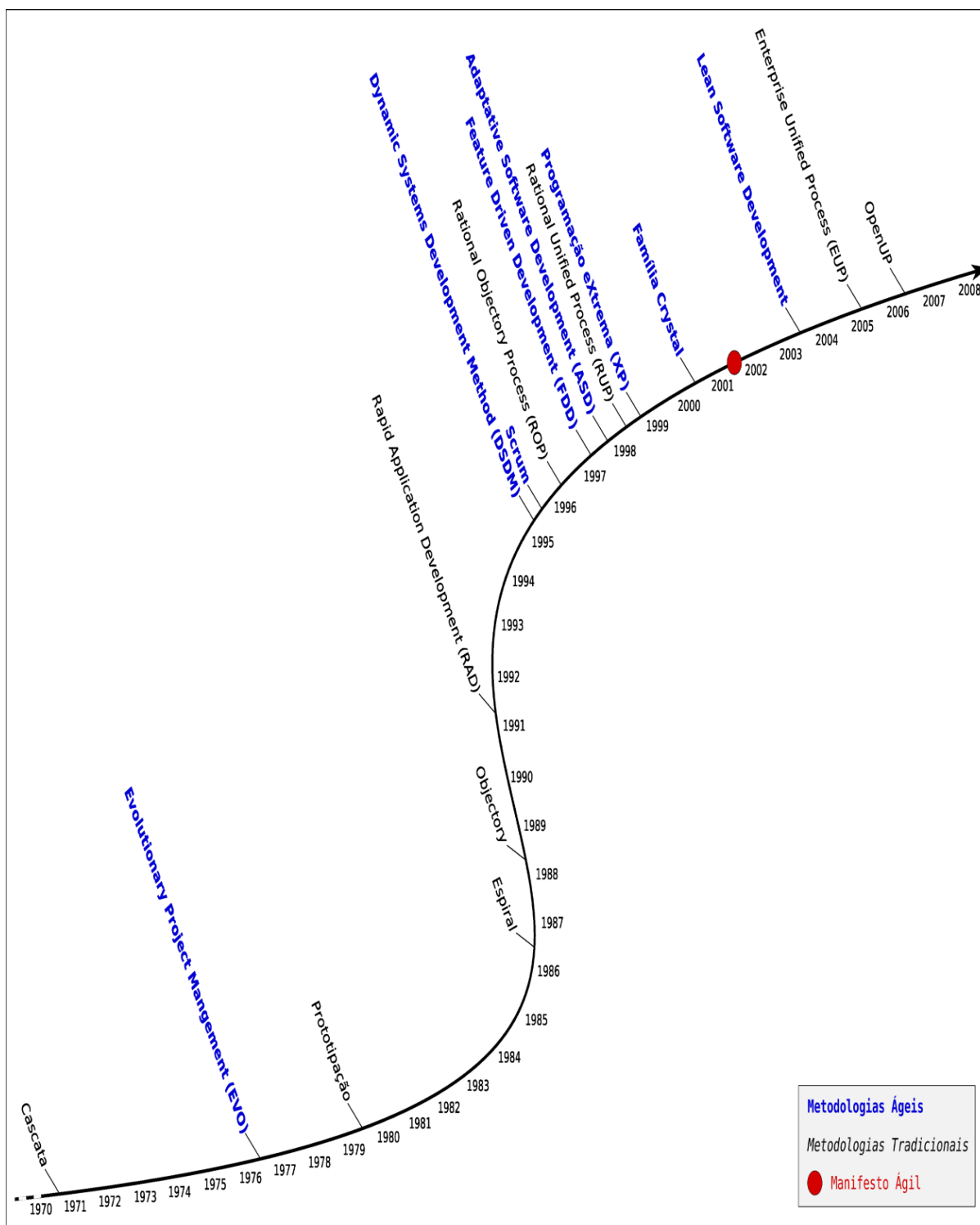
### 2.1 – AS METODOLOGIAS ÁGEIS

A Figura 1 é uma linha do tempo que mostra que as Metodologias Ágeis emergem em meados dos anos 90. Segundo Filho (2008), alguns líderes, ao perceberem os fracassos nas tentativas de desenvolver software, resolveram inovar seus processos, opondo-se às práticas comuns da época. A partir daí começaram a surgir as Metodologias Ágeis.

---

<sup>1</sup> Segundo Mattsson (1996, 2000), um framework é uma arquitetura desenvolvida com o objetivo de atingir a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização.

**Figura 1 – Histórico das metodologias de desenvolvimento de software**



**Fonte:** FILHO (2008, p. 10).

### 2.1.1 – O manifesto para o desenvolvimento ágil de software

Dos líderes que começaram a inovar seus processos de Software, 17 notaram a eficácia dessa inovação e a semelhança de uns com os outros. Sendo assim, resolveram se reunir em 2001, durante um final de semana, para discutir sobre suas formas de trabalho e para chegar a uma nova metodologia comum, substituindo os modelos tradicionais de desenvolvimento (FILHO, 2008).

A conclusão que chegaram é que desenvolver software é algo complexo demais para ser definido por um único processo, uma vez que existem muitas variáveis e principalmente por ser uma tarefa inerentemente realizada por pessoas em praticamente todas as etapas do processo. Mas essa reunião teve um fruto, e o resultado deste encontro foi o Manifesto Ágil (FILHO, 2008).

Essas 17 pessoas passaram a descobrir maneiras melhores de desenvolver software e seus valores sofreram mudanças (BECK; *et al.*, 2001).

Eles passaram a valorizar:

1. **Indivíduos e interação entre eles** mais que processos e ferramentas;
2. **Software em funcionamento** mais que documentação abrangente;
3. **Colaboração com o cliente** mais que negociação de contratos;
4. **Responder a mudanças** mais que seguir um plano.

Assim como valores sólidos, a reunião que resultou no Manifesto Ágil, também elencou doze princípios:

1. Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor;
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adaptam 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 através 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.

É a partir desse momento que as Metodologias Ágeis começaram a ganhar espaço no Mercado de Software.

### 2.1.2 – O framework Scrum

Entre os autores do Manifesto Ágil, estavam **Ken Schwaber** e **Jeff Sutherland**, os quais desenvolveram o Scrum, algo que eles classificaram como um Framework. Esses dois autores criaram um documento chamado Scrum Guide, o qual é periodicamente atualizado por eles mesmo, sendo que a última atualização é de julho de 2013 (SCHWABER; SUTHERLAND, 2013).

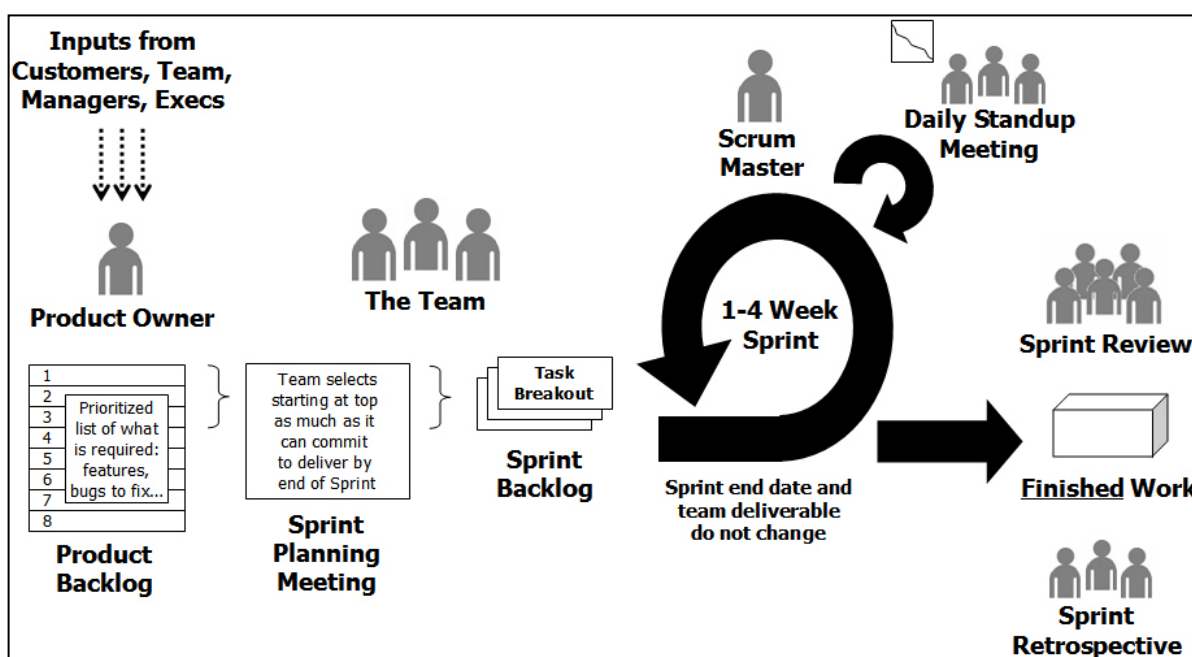
Abaixo está a definição de Scrum segundo seus criadores:

“Scrum (subs): Um framework dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível. Scrum é: Leve, Simples de entender e Extremamente difícil de dominar” (SCHWABER & SUTHERLAND, 2013, P. 3).

### 2.1.2.1 – Visão geral do Scrum

A Figura 2 apresenta uma noção de como o Scrum funciona de fato, desde a solicitação do cliente até a finalização do trabalho. As partes envolvidas nessa imagem serão descritas ao decorrer desse capítulo.

Figura 2 – Visão geral do Scrum



Fonte: SUTHERLAND (2007, p. 22).

Tudo começa com as *Solicitações*, sejam elas de *Clientes*, da *Equipe*, dos *Gerentes*, essas solicitações ficam concentradas nas mãos do *Product Owner*. O *Product Owner* irá elaborar uma lista com todas as solicitações, ordenando-as por prioridade e descrevendo cada solicitação, essa lista se chama *Product Backlog*.

Com o *Product Backlog* em mãos, *Product Owner*, *Scrum Master* e *the Team* se reúnem para planejar a *Sprint*, que é extrair do *Product Backlog*, formando o *Sprint Backlog*, um número de solicitações que será feito durante um determinado tempo, geralmente de uma a quatro semanas.

Após definir a *Sprint*, o trabalho se segue durante o tempo estipulado, sendo que diariamente *Scrum Master* e *the Team* se reúnem para identificar impedimentos e verificar como está o andamento da *Sprint*. Durante toda a *Sprint*, o *Scrum Master*



tem a responsabilidade de guiar *the Team*, fazendo com que os princípios do Scrum sejam aplicados.

Ao final do período, é feita a *Sprint Review*, na qual todas as novas funcionalidades são demonstradas, assim o trabalho é finalizado. Mas antes de iniciar uma nova *Sprint*, é feita a *Sprint Retrospective*, que tem o objetivo de identificar as melhorias, as quais poderão ser aplicadas posteriormente para um melhor aproveitamento dos trabalhos.

Sutherland (2007) quis mostrar com essa imagem que o Scrum é um ciclo que deve ser exaustivamente repetido, sendo que a cada repetição, novas melhorias são alcançadas, assim como novos obstáculos surgem no caminho. Isso faz lembrar uma frase, cujo autor é desconhecido, “A repetição é o caminho da perfeição”.

### 2.1.2.2 – Scrum e seus fundamentos

Segundo os próprios criadores de Scrum, a fundamentação do mesmo está nas teorias empíricas de controle de processo, ou seja, o Empirismo<sup>2</sup>. Assim sendo, a abordagem do Scrum é iterativa e incremental, procurando encurtar os laços entre os envolvidos, aperfeiçoando assim a capacidade de prever e de controlar os riscos que decorrem durante a relação (SCHWABER; SUTHERLAND, 2013).

Segundo Schwaber; Sutherland (2013), os controles de processo empírico apóiam-se em três pilares, sendo eles:

- **Transparência:** Tudo o que pode impactar o processo deve ser da ciência dos envolvidos pelo resultado, caso contrário os resultados poderão ser negativos;
- **Inspeção:** A inspeção pode garantir que tudo o que foi acordado seja cumprido, por isso frequentemente os envolvidos devem inspecionar o andamento do processo, a fim de identificar impedimentos à entrega dos resultados; mas essa inspeção não pode se tornar mais um impedimento,

---

<sup>2</sup> A teoria do **Empirismo** afirma que o conhecimento vem da experiência e de tomada de decisões baseadas no que é conhecido (SCHWABER; SUTHERLAND, 2013, p. 4)

portanto precisa ser de forma que não atrapalhe o desenvolvimento das atividades.

- **Adaptação:** Em caso de um desvio inaceitável do produto final, o Time Scrum precisava se adaptar para que os resultados possam ser o mais positivo possível.

Ainda sobre os pilares do Empirismo, Schwaber; Sutherland (2013) identificam quatro oportunidades formais para inspeção e adaptação:

- Reunião de planejamento da Sprint
- Reunião diária (Daily Scrum)
- Reunião de revisão da Sprint
- Retrospectiva da Sprint

### 2.1.2.3 – O Time Scrum

O Time Scrum compõe-se de três partes, sendo que a ausência de qualquer uma delas impossibilita a aplicação do Scrum. Essas partes são o **Product Owner**, a **Equipe de Desenvolvimento** e o **Scrum Master**. Cada uma delas será vista com um maior detalhamento no decorrer desse subitem (SCHWABER; SUTHERLAND, 2013).

**Product Owner**, como o próprio nome já diz, é o Dono do Produto, e como tal, a sua responsabilidade é zelar pelo produto, visando maximizar o valor do mesmo e o trabalho daqueles que agregam valor ao Produto, ou seja, a Equipe de Desenvolvimento (SCHWABER; SUTHERLAND, 2013).

O Product Owner é representado por apenas uma pessoa, podendo até representar um comitê, mas somente uma pessoa pode ser o Product Owner. A grande responsabilidade dessa parte é manter o Product Backlog, ou seja, a lista de solicitações dos clientes (SCHWABER; SUTHERLAND, 2013).

Segundo Kniberg (2007), o Product Owner precisa entender cada solicitação contida no Product Backlog, pois será ele o responsável por priorizá-las. Não é necessário um conhecimento profundo de cada solicitação, mas conhecimento o

suficiente para entender o porquê da solicitação e o que ela vai agregar ao produto, podendo assim estabelecer uma prioridade.

Kniberg (2007) também é enfático ao dizer que o Product Owner precisa manter o Product Backlog a nível de negócio. Caso contrário, o Scrum Master e/ou a Equipe precisa intervir.

“Quando vejo histórias com orientação técnica como ‘Adicionar índice na tabela de eventos’, normalmente eu pergunto ao Product Owner uma série de questões de ‘mas por quê’, até encontrar o objetivo subjacente. Então nós reformulamos a história nos termos do objetivo subjacente (‘acelerar o formulário de pesquisa de eventos no back office’). A descrição técnica original acaba virando uma nota (‘Indexar a tabela de eventos poderia resolver isso’).” (KNIBERG, 2007, p. 12)

O sucesso do Product Owner depende do respeito que os envolvidos terão perante suas decisões, pois é ele, e somente ele, que determinará a prioridade de cada solicitação, o implica diretamente na ordem em que cada atividade será realizada (SCHWABER; SUTHERLAND, 2013).

A **Equipe de Desenvolvimento** é responsável por agregar valor ao Produto, realizando o trabalho descrito em cada Sprint (SCHWABER; SUTHERLAND, 2013).

Segundo Schwaber; Sutherland (2013), as Equipes deverão ter a autonomia para organizar e gerenciar seu próprio trabalho, e tem as seguintes características:

- Elas são auto-organizadas. Ninguém (nem mesmo o Scrum Master) diz ao Time de Desenvolvimento como transformar o Backlog do Produto em incrementos de funcionalidades potencialmente utilizáveis;
- Times de Desenvolvimento são multifuncionais, possuindo todas as habilidades necessárias, enquanto equipe, para criar o incremento do Produto;
- O Scrum não reconhece títulos para os integrantes do Time de Desenvolvimento que não seja o Desenvolvedor, independentemente do trabalho que está sendo realizado pela pessoa; Não há exceções para esta regra;
- Individualmente os integrantes do Time de Desenvolvimento podem ter habilidades especializadas e área de especialização, mas a responsabilidade pertence ao Time de Desenvolvimento como um todo;
- Times de Desenvolvimento não contém sub-times dedicados a domínios específicos de conhecimento, tais como teste ou análise de negócios.

Quanto ao tamanho das Equipes, esse pode ser variado, mas deve-se levar em conta a agilidade dos processos, sendo que quanto mais integrantes, maior deve ser o esforço de coordenação da Equipe (SCHWABER; SUTHERLAND, 2013).

O **Scrum Master**, tal como sua definição de Mestre é aquele que domina um determinado assunto, mas mais do que dominar o Scrum, ele é responsável por difundir essa idéia fazendo com que todos os envolvidos entendam e apliquem cada parte do Scrum. Para o Time Scrum, o Scrum Master deve ser um servo-líder, conduzindo todos na direção do Scrum (SCHWABER; SUTHERLAND, 2013).

Mas a função dessa figura não se limita ao Time Scrum, pois ele também precisa fazer com os que estão de fora entendam a importância de cada interação com o Time Scrum (SCHWABER; SUTHERLAND, 2013).

Schwaber; Sutherland (2013) citam alguns exemplos do que o Scrum Master deve fazer.

O Scrum Master serve o Product Owner de várias maneiras, incluindo:

- Encontrando técnicas para o gerenciamento efetivo do Backlog do Produto;
- Claramente comunicar a visão, objetivo e itens do Backlog do Produto para o Time de Desenvolvimento;

O Scrum Master serve o Time de Desenvolvimento de várias maneiras, incluindo:

- Treinar o Time de Desenvolvimento em autogerenciamento e interdisciplinaridade;
- Ensinar e liderar o Time de Desenvolvimento na criação de produtos de alto valor;
- Remover impedimentos para o progresso do Time de Desenvolvimento;

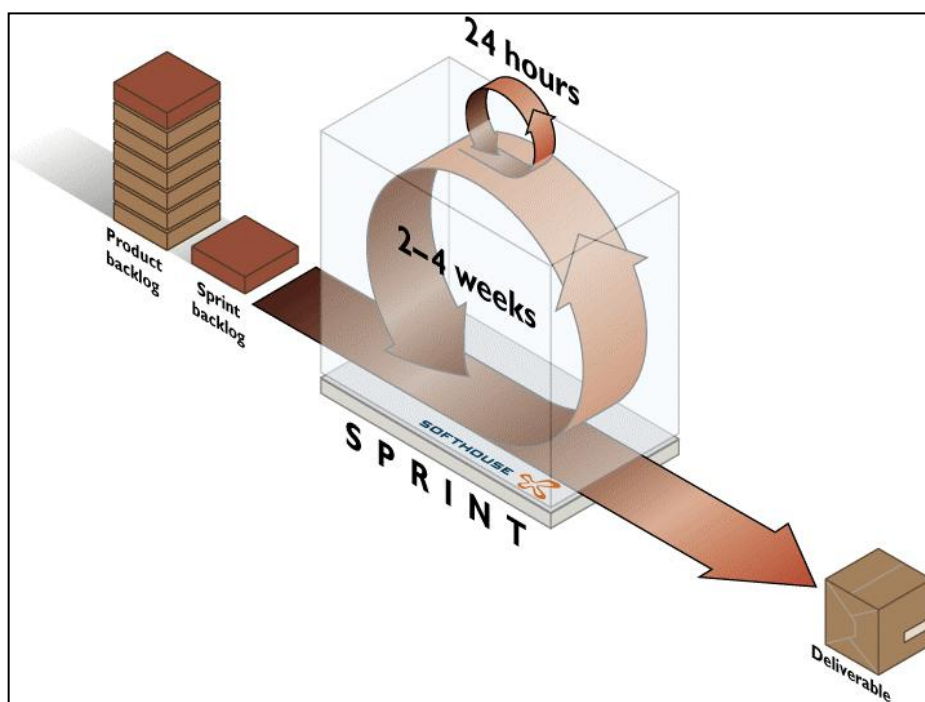
O Scrum Master serve a Organização de várias maneiras, incluindo:

- Liderando e treinando a organização na adoção do Scrum;
- Planejando implementações Scrum dentro da organização;
- Ajudando funcionários e partes interessadas a compreender e tornar aplicável o Scrum e o desenvolvimento de produto empírico;

### 2.1.2.4 – A Sprint

Na Figura 2 foi possível ter uma visão geral do Scrum, agora na Figura 3 será possível ver um pouco mais detalhado a Sprint.

**Figura 3 – O ciclo da Sprint**



**Fonte:** SUTHERLAND (2007, p. 15).

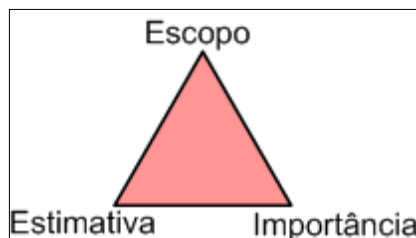
Segundo Schwaber; Sutherland (2013), a Sprint é considerada o coração do Scrum, e deve durar até um mês, sendo que ao final desse período o Time Scrum deverá entregar as solicitações selecionadas no Sprint Backlog. Ao decorrer da Sprint, quatro eventos ganham destaque, sendo a **Reunião de Planejamento**, as **Reuniões Diárias**, a **Revisão da Sprint** e a **Retrospectiva da Sprint**.

Uma Sprint se inicia com a **Reunião de Planejamento**, a qual normalmente tem uma duração de oito horas para uma Sprint de quatro semanas, e um tempo proporcional para Sprints com outro intervalo de duração (SCHWABER; SUTHERLAND, 2013).

Kniberg (2007) diz que a presença do Product Owner na reunião de planejamento é imprescindível, isso porque cada estória do Backlog do Produto, ou

seja, as solicitações dos clientes, contém três variáveis dependentes umas das outras, é o que mostra a figura 4.

**Figura 4 – As três variáveis de cada solicitação do Backlog**



**Fonte:** KNIBERG (2007, p. 16).

“Escopo e importância são definidos pelo Product Owner. Estimativa é definida pela Equipe. Durante uma reunião de planejamento da Sprint, estas três variáveis são refinadas continuamente por diálogo cara-a-cara entre Equipe e Product Owner.” (KNIBERG, 2007, p. 12)

Segundo Schwaber; Sutherland (2013), a reunião é dividida em duas partes que visam responder duas questões, sendo respectivamente:

- O que será entregue como resultado de incremento da próxima Sprint?
- Como o trabalho necessário para entregar o incremento será realizado?

Para iniciar a primeira parte da reunião, todos tomam ciência de quatro coisas, o Backlog do Product, onde estão todas as solicitações ordenadas por prioridade; o resultado da última Sprint; a capacidade projetada da Equipe de Desenvolvimento, ou seja, o foco da Equipe nas atividades relacionadas à Sprint; e o desempenho passado da Equipe de Desenvolvimento. Esses quatro elementos ajudarão a nortear as escolhas para definir o Backlog da Sprint. Os dois últimos são de suma importância para a equipe saber o quanto se comprometerá com a Sprint que está sendo montada. A partir daí, a Equipe de Desenvolvimento começa a estimar as solicitações e montar o Backlog da Sprint de acordo com o que é conversado com o Product Owner (SCHWABER; SUTHERLAND, 2013).

Segundo Schwaber; Sutherland (2013), algo importante dessa primeira parte, e que é responsável por encerrá-la, é a definição do objetivo da Sprint; basicamente o Time Scrum define o porquê de desenvolver as atividades propostas. E na segunda parte, a Equipe de Desenvolvimento define a forma como as solicitações serão desenvolvidas, ou seja, como o Backlog da Sprint será convertido em um incremento utilizável do Produto.

A **Reunião Diária**, que é outro componente de uma Sprint, tem uma duração de 15 minutos e serve para inspecionar o trabalho realizado desde a última reunião como também planejar o que será feito nas próximas 24 horas. Ela deve ser mantida no mesmo local e horário (SCHWABER; SUTHERLAND, 2013).

Segundo Schwaber; Sutherland (2013), na Reunião Diária, também chamada Daily Scrum, cada membro da Equipe precisa responder três questionamentos:

- O que foi completado desde a última reunião?
- O que será feito até a próxima reunião?
- Quais os obstáculos que estão no caminho?

Uma dica de Kniberg (2007) para manter a Reunião Diária dentro dos 15 minutos estipulados é permanecer em pé durante esse evento. Outra prática comum de Kniberg (2007) é atualizar o quadro de tarefas.

A Figura 5 é um exemplo de quadro de tarefas e gráfico burndown.

**Figura 5 – Quadro de tarefas Scrum**



Fonte: KNIBERG (2007).

Schwaber; Sutherland (2013) acreditam que:

“Reuniões Diárias melhoram as comunicações, eliminam outras reuniões, identificam e removem impedimentos para o desenvolvimento, destacam e promovem rápidas tomadas de decisão, e melhoram o nível de conhecimento do Time de Desenvolvimento. Esta é uma reunião chave para inspeção e adaptação.”

Ao serem concluídas as atividades da Sprint, deve-se executar a **Revisão da Sprint**, com o intuito de inspecionar o que será incrementado ao Produto e de adaptar o Backlog do Produto caso haja necessidade. O tempo de duração deve ser de quatro horas para uma Sprint de quatro semanas, e proporcionalmente para diferentes tamanhos de Sprint (SCHWABER; SUTHERLAND, 2013).

Segundo Schwaber; Sutherland (2013), a Revisão da Sprint inclui:

- Os participantes incluem o Time Scrum e os Stakeholders<sup>3</sup> convidados pelo Product Owner;
- O Product Owner esclarece quais itens do Backlog do Produto foram “Prontos” e quais não foram “Prontos”;
- O Time de Desenvolvimento discute o que foi bem durante a Sprint, quais problemas ocorreram dentro da Sprint, e como estes problemas foram resolvidos;
- O Time de Desenvolvimento demonstra o trabalho que está “Pronto” e responde as questões sobre o incremento;
- O Product Owner discute o Backlog do Produto tal como está. Ele (ou ela) projeta as prováveis datas de conclusão baseado no progresso até a data (se necessário);
- O grupo todo colabora sobre o que fazer a seguir, e é assim que a Reunião de Revisão da Sprint fornece valiosas entradas para a Reunião de Planejamento da próxima Sprint;
- Análise de como o mercado ou o uso potencial do produto pode ter mudado e o que é a coisa mais importante a se fazer a seguir; e,
- Análise da linha do tempo, orçamento, potenciais capacidades, e mercado para a próxima versão esperada do produto.

---

<sup>3</sup> **Stakeholder** significa **público estratégico**. Em inglês stake significa interesse, participação, risco. Holder significa aquele que possui. Assim, stakeholder também significa **parte interessada** ou **interveniente**. É uma palavra em inglês muito utilizada nas áreas de comunicação, administração e tecnologia da informação cujo objetivo é designar as pessoas e grupos mais importantes para um planejamento estratégico ou plano de negócios, ou seja, as partes interessadas. (Disponível em: <<http://www.significados.com.br/stakeholder/>>. Acesso em: 03 nov. 2013.)



Como resultado da Revisão da Sprint, tem-se um Backlog do Produto atualizado que provavelmente será o utilizado para na próxima Reunião de planejamento da Sprint (SCHWABER; SUTHERLAND, 2013).

Conforme falado no subitem 2.1.2.2, Scrum se baseia no Empirismo, ou seja, toma decisões baseado no conhecimento adquirido. Assim sendo, a **Retrospectiva da Sprint** é a oportunidade de olhar o passado para melhorar no futuro (SCHWABER; SUTHERLAND, 2013).

Segundo Schwaber; Sutherland (2013), o propósito da Retrospectiva da Sprint é:

- Inspecionar como a última Sprint foi em relação às pessoas, aos relacionamentos, aos processos e às ferramentas;
- Identificar e ordenar os principais itens que foram bem e as potenciais melhorias;
- Criar um plano para implementar melhorias no modo que o Time Scrum faz seu trabalho;

Em complemento ao propósito da Retrospectiva mostrado acima, Kniberg (2007) exemplifica uma reunião de retrospectiva com três colunas:

- Bom: se pudéssemos refazer a mesma Sprint novamente, faríamos as coisas do mesmo modo;
- Poderia ter sido melhor: se pudéssemos refazer a mesma Sprint novamente, faríamos as coisas de uma maneira diferente;
- Melhorias: ideias concretas sobre como podemos melhorar no futuro.

Schwaber; Sutherland (2013) concluem que:

“Ao final da Retrospectiva da Sprint, o Time Scrum deverá ter identificado melhorias que serão implementadas na próxima Sprint. A implementação destas melhorias na próxima Sprint é a forma de adaptação à inspeção que o Time Scrum faz a si próprio. A Retrospectiva da Sprint fornece um evento dedicado e focado na inspeção e adaptação, no entanto, as melhorias podem ser adotadas a qualquer momento.”

#### 2.1.2.5 – Artefatos do Scrum

No Scrum existem três artefatos, sendo o **Backlog do Produto**, o **Backlog da Sprint** e o **Incremento**.

O **Backlog do Produto** é um artefato vivo, uma vez que sofre alterações constantemente. Isso se deve ao fato de o Produto estar sendo usado, o que gera um retorno e mais necessidades, aumentando a lista do Backlog do Produto (SCHWABER; SUTHERLAND, 2013).

Schwaber; Sutherland (2013) diz que:

“O Backlog do Produto é uma lista ordenada de tudo que deve ser necessário no produto, e é uma origem única dos requisitos para qualquer mudança a ser feita no produto. O Product Owner é responsável pelo Backlog do Produto, incluindo seu conteúdo, disponibilidade e ordenação.”

Para Kniberg (2007), as histórias precisam ter as seguintes informações:

- **ID** – Uma identificação única, apenas um número com autoincremento. Isso é para evitar que percamos o controle sobre as histórias quando nós mudamos seus nomes.
- **Nome** – Um nome curto e descritivo para a história. Por exemplo, “Ver o histórico de transações”. Suficientemente claro para que os desenvolvedores e o product owner entendam mais ou menos sobre o que estamos falando, e claro o bastante para distingui-la das demais histórias. Normalmente de 2 a 10 palavras.
- **Importância** – A pontuação de importância dessa história para o Product Owner. Por exemplo 10. Ou 150. Mais pontos = mais importante.
- **Estimativa inicial** – As estimativas iniciais da equipe sobre quanto tempo é necessário para implementar aquela história, se comparada a outras histórias. A unidade é pontos por história e geralmente corresponde mais ou menos a “relação homem/dias” ideal.
  1. Pergunte à equipe “se vocês puderem ter o número ideal de pessoas para esta história (nem muitas, nem poucas, normalmente duas), e se trancarem em uma sala cheia de comida e trabalharem sem distúrbio algum, após quantos dias vocês apresentarão uma implementação pronta, demonstrável e testada?” Se a resposta for “com 3 pessoas trancados em uma sala levará aproximadamente 4 dias” então a estimativa inicial é de 12 pontos por história.
  2. O importante não é ter estimativas absolutamente precisas (por exemplo, dizer que uma história com 2 pontos deverá

gastar 2 dias), mas sim obter estimativas relativas corretas (por exemplo, dizer que uma estória com 2 pontos gastará cerca da metade de uma estória com 4 pontos).

- **Como demonstrar** – Uma descrição em alto nível de como a estória será demonstrada na apresentação do sprint. Isso é simplesmente uma simples especificação de teste. “Faça isso, então faça aquilo e então isso deverá acontecer.”
- **Notas** – Quaisquer outras informações, esclarecimentos, referências a outras fontes de informação, etc. Normalmente ágil bem breve.

Sobre a informação prioridade, Kniberg (2007) tem a seguinte opinião:

“Eu tento evitar o termo “prioridade” já que prioridade 1 é tipicamente interpretado como “prioridade mais alta”, o que fica feio se mais tarde você decidir que algo é ainda mais importante. Qual pontuação de prioridade esse item deveria receber? Prioridade 0? Prioridade -1?”

Conforme foi dito anteriormente no subitem 2.1.2.4, o **Backlog da Sprint** é montado na Reunião de Planejamento da Sprint. Nada mais é do que a seleção de um número de solicitações do Backlog do Produto, que provavelmente será concluído durante a Sprint que está sendo planejada.

Segundo Schwaber; Sutherland (2013):

“Sempre que um novo trabalho é necessário, o Time de Desenvolvimento adiciona este ao Backlog da Sprint. Conforme o trabalho é realizado ou completado, a estimativa do trabalho restante é atualizada. Quando elementos do plano são considerados desnecessários, eles são removidos. Somente o Time de Desenvolvimento pode alterar o Backlog da Sprint durante a Sprint. O Backlog da Sprint é altamente visível, uma imagem em tempo real do trabalho que o Time de Desenvolvimento planeja completar durante a Sprint, e pertence exclusivamente ao Time de Desenvolvimento.”

A definição de **Incremento** segundo Schwaber; Sutherland (2013):

“O incremento é a soma de todos os itens do Backlog do Produto completados durante a Sprint e o valor dos incrementos de todas as Sprints anteriores. Ao final da Sprint um novo incremento deve estar “Pronto”, o que significa que deve estar na condição utilizável e atender a definição de “Pronto” do Time Scrum. Este deve estar na condição utilizável independente do Product Owner decidir por liberá-lo realmente ou não.”

## 2.2 – FERRAMENTAS CASE

### 2.2.1 – Introdução

Os desafios em desenvolvimento de softwares já eram vividos na década de 60, quando surgiu a chamada “Crise do Software”, onde foi conhecido problemas ocorrentes até hoje, tal como: Previsão incorreta, softwares de baixa qualidade, manutenção de alto custo e esforços duplicados (GIBBS, 1994). Já podemos imaginar a quanto tempo há a necessidade de se utilizar ferramentas para auxílio ao desenvolvimento de software.

De acordo com Schach (2008, p. 89), após duas décadas de promessas mal cumpridas sobre produtividade e qualidade de retornos ao experimentar novas metodologias e tecnologias de softwares, as indústrias e organizações têm dado conta de que o problema fundamental é a inabilidade de gerenciar os processos de software.

A tradicional e conhecida expressão “em casa de ferreiro espeto é de pau” cabe bem à realidade dos informáticos, assim como o ferreiro não utiliza utensílios para facilitar seu trabalho, grande parte da atividade de desenvolvimento de software apresentam carência de ferramentas de apoio. (SILVA & VIDEIRA, 2001).

Por outro lado, hoje temos à disposição diversos recursos para facilitar e melhorar o trabalho nessa área de desenvolvimento de software, tal como ferramentas analíticas que auxiliam em processos de tomada de decisões e análises de custo-benefício, e ferramentas de software, que dão suporte à equipe de engenheiros de software no processo de desenvolvimento e manutenção de softwares (SCHACH, 2008).

Os tópicos seguintes abordarão a exploração de softwares que dão apoio às atividades de desenvolvimento e que são conhecidos como ferramentas CASE, será explicado o que são, e então demonstrar como utilizar algumas ferramentas conhecidas como geradores de código, figurando atividades básicas de desenvolvimento utilizando meios visuais de implementação para comparar aos meios mais utilizados de desenvolvimento de softwares comerciais que é a

construção de códigos fonte, propondo assim, uma solução ou apoio para problemas comuns conhecidos em projetos de desenvolvimento de software.

### **2.2.2 – O que é uma ferramenta CASE**

Durante o processo de desenvolvimento de um produto de software, deve ser considerado diversas etapas e atividades. Atividades típicas que incluem estimativa de demanda de recursos, desenho de documentos de requerimentos e especificações e montagem de manuais de usuário. Existem ferramentas para todas estas atividades, porém, nenhuma tem a capacidade de executar tarefas sozinhas, todas necessitam da intervenção humana.(SCHACH, 2008).

Apesar de tantas etapas e tanta atenção e esforço humano requerido para criar softwares que fazem a máquina calcular e executar uma infinidade operações, ironicamente até, o computador pode acompanhar e dar assistência a todas as etapas do processo de desenvolvimento em todos os aspectos, executando grande parte da carga de trabalho como preparando ambientes prontos e ideais para o desenvolvedor trabalhar apenas na lógica do negócio, utilizando seus esforços apenas onde interessa. Vale lembrar que a documentação é um processo muito importante que faz parte do desenvolvimento, porém geralmente não tem a devida atenção por parte dos envolvidos. A documentação é uma parte essencial para o desenvolvimento e manutenção de softwares pois ao manter os diagramas e descrições da regra de negócios torna-se muito útil ao passo que facilita possíveis alterações e implementações futuras.(SCHACH, 2008).

Neste ponto, compreendemos que existem estes softwares que são ferramentas para todas as atividades de desenvolvimento, que auxiliam e guiam os profissionais de tecnologia a desempenhar seus trabalhos de documentação, gerenciamento e desenvolvimento. Estes softwares são simples formas de ferramentas conhecidas pelo termo “CASE”.

Ferramentas CASE é uma classificação que abrange todas as ferramentas baseadas em computadores que auxiliam atividades de engenharia de software, desde análise de requisitos e modelagem até programação e testes. Podem ser

consideradas como ferramentas que tem como objetivo auxiliar o desenvolvedor de sistemas em uma ou várias etapas do ciclo de desenvolvimento de software. (WEINRICH, 1999).

Segundo Schach (2008, p. 128), o “A” do termo “CASE” como mencionado, diz respeito a *Aided* (Apoio) e não *Automated* (Automatizado) como pode ser encontrado em algumas definições, pois ainda nenhum computador têm a capacidade de substituir um humano quando diz respeito a desenvolvimento e manutenção de software, os recursos e capacidades podem evoluir muito no futuro, mas o computador sempre será uma ferramenta para o profissional de software.

Dentro da definição de ferramentas CASE estão softwares que permitem ao desenvolvedor gerar código fonte, scripts e até construir um sistema completo por meio de modelagem visual, baseado em vários conceitos como UML (do inglês Unified Modeling Language), por exemplo, e teoricamente, sem a necessidade do desenvolvedor ter de intervir com códigos de programação para executar implementações ou alterações.

Existem ferramentas CASE que “permitem o envolvimento de especialistas que não necessariamente dominam as linguagens envolvidas na construção de sistemas Web (\*HTML, JavaScript, Java...) ou Mobile (Objective-C, Java...). Como exemplo, um desenvolvedor especialista em Clipper ou Cobol pode ser inserido no processo de desenvolvimento de aplicações Web ou Mobile” (SOFTWELL, 2013).

### **2.2.3 – Taxonomia das ferramentas CASE**

Assim como apresentado anteriormente, por definição, uma ferramenta CASE pode ser qualquer programa que dê suporte para desenvolvimento de um trabalho ou que torne possível a execução de um trabalho tendo o computador como ferramenta. Neste caso, um editor de texto pode ser considerado uma ferramenta CASE, apesar deste conceito não ser bem aceito pelos informáticos, sendo que um simples editor de texto não desempenharia um papel tão nobre no desenvolvimento de um Projeto por exemplo, mas a partir do momento que lembrarmos que um editor de texto pode oferecer funções de auto completamento, correção, substituição de

palavras e que sem ele não seria possível a construção de códigos, podemos reconsiderar a inclui-los na definição de ferramentas CASE. (SILVA & VIDEIRA, 2001).

Esta definição por sua vez, incluiu uma grande diversidade de possíveis softwares a serem considerados tais ferramentas de apoio, portanto, a taxonomia das ferramentas CASE define grupos de categorias de aplicabilidade.

Iniciando com uma definição mais simples e generalista, segundo Schach (2008), pode se observar as ferramentas que auxiliam o desenvolvedor durante os primeiros fluxos de trabalho do processo (os fluxos de trabalho de levantamento de necessidades, de análise e de projeto) algumas vezes são chamadas de **upperCASE** ou ferramentas **front-end**, ao passo que aquelas que ajudam no fluxo de trabalho de implementação e na manutenção pós-entrega são denominadas **lowerCASE** ou ferramentas **back-end**.

Agora, baseando-se em Silva & Videira (2001), temos a taxonomia das ferramentas CASE por funções, para a melhor compreensão de onde cada ferramenta se encaixa no desenvolvimento de um projeto de software:

- **Modelação de processos de negócio:** São softwares baseados em BPM (*Business Process Modeling*) que permitem visualizar e desenhar processos para análise e melhoria de estratégias e objetivos do negócio por meio de notações e diagramas, possibilita melhor compreensão do funcionamento da empresa e visualização de novas oportunidades. Exemplos: SmartDraw ([www.smartdraw.com](http://www.smartdraw.com)), Logizian ([www.visual-paradigm.com/product/lz/](http://www.visual-paradigm.com/product/lz/)), Open ModelSphere ([www.modelsphere.com](http://www.modelsphere.com)).
- **Modelação de análise e desenho do sistema:** A maioria das ferramentas CASE estão nesta categoria, são ferramentas que permitem a construção da arquitetura de um software com base na modelagem de negócios, basicamente são os geradores de códigos, os quais possibilitam a programação de softwares através de linguagem visual, são capazes de gerar códigos de diversas linguagens de programação a partir de uma linguagem visual unificada. Geralmente são baseados nas técnicas de Linguagem de modelagem unificada (UML). Exemplos: Rational Rose ([www-03.ibm.com/software/products/br/pt/ratirosefami](http://www-03.ibm.com/software/products/br/pt/ratirosefami)), Maker ([www.softwell.com.br](http://www.softwell.com.br)), GeneXus ([www.genexus.com](http://www.genexus.com)).

- **Desenho de bases de dados:** Muitas vezes são integradas a categoria anterior, porém, sua finalidade é a modelagem de base de dados, podendo ser usado para versionamento de um banco de dados de um projeto, podem oferecer funções de engenharia reversa, geração de scripts, suportam a modelagem física e lógica da base de dados a partir de diagramas de relacionamento ou SQL. Exemplos: Erwin (erwin.com), DB-UML (sourceforge.net/projects/dbuml), Modelright ([www.modelright.com](http://www.modelright.com)), MySQL Workbench ([www.mysql.com/products/workbench](http://www.mysql.com/products/workbench)).
- **Programação de aplicações:** Aqui estão os ambientes de desenvolvimento (\*IDE) para as linguagens de programação (C, C++, C#, Java, Delphi, PHP, Visual Basic e diversas outras) e que são acompanhados de pacotes de programas integrados como interpretadores, compiladores, geradores de código, debuggers, editores de interfaces visuais e editores de código fonte. Todos com a finalidade de criação e edição de programas. Exemplos: Visual Studio ([www.microsoft.com](http://www.microsoft.com)), NetBeans (netbeans.org), Eclipse ([www.eclipse.org](http://www.eclipse.org)), RAD Studio ([www.embarcadero.com](http://www.embarcadero.com)).
- **Gestão de alterações no software:** São as ferramentas para controle de desenvolvimento de um projeto em equipe. Utilizam o conceito de *check-in* e *check-out* que consiste em bloquear arquivos para um usuário e liberá-los aplicando as alterações feitas, possuem gestão de versão do projeto e seus arquivos, podem ser acompanhados de ferramentas para comparação de conteúdo entre arquivos para gestão de conflitos. Exemplos: Team Foundation Server ([www.microsoft.com](http://www.microsoft.com)), StarTeam ([www.borland.com](http://www.borland.com)), Subversion (subversion.apache.org), Git (git-scm.com), Mercurial (mercurial.selenic.com).
- **Testes:** Estas ferramentas dão apoio às atividades de qualidade de software, permitindo a construção de scripts para automação de testes em softwares aplicando técnicas de stress e exaustão, fazem gestão e coleta de dados sobre erros (Bug-tracking) e performance. Exemplos: STAF (staf.sourceforge.net), SpiraTest ([www.inflectra.com/SpiraTest](http://www.inflectra.com/SpiraTest)), WinAutomation ([www.winautomation.com](http://www.winautomation.com)).



- **Orientado a gestão de projetos:** Várias dessas ferramentas são baseadas em metodologias ágeis de desenvolvimento tal como apresentado anteriormente neste documento, gerenciam alocação de recursos para desenvolvimento, atribuição de responsabilidades e atendimento de solicitações de suporte, podem contar com interfaces externas para serem usadas por usuários para solicitações. Também oferecem informações sobre estimativa de tempo, custos e recursos utilizados. Exemplos: Brazil SCRUM ([www.brazil.com.br](http://www.brazil.com.br)), Wrike ([www.wrike.com](http://www.wrike.com)), OpenProject ([www.openproject.org](http://www.openproject.org)), MS Project ([office.microsoft.com/project/](http://office.microsoft.com/project/)).

#### **2.2.4 – Vantagens e desvantagens sobre ferramentas CASE**

Até agora, mencionamos diversas forças das ferramentas CASE, porém não há apenas glórias e sucesso em sua história, até hoje pode ser considerado um desafio em alguns casos o processo de adoção de uma metodologia de trabalho unida com um conjunto de ferramentas. Segundo Silva & Videira (2001), no passado era comum os resultados obtidos não serem como o esperado, apesar do grande crédito colocado por conta do que era prometido fazer, este foi um fator que dificultou a entrada das ferramentas CASE no mercado. Outra dificuldade que ocorreu, foi o grande período de tempo demandado para treinamento e aprendizagem a fim de que os usuários fossem capazes de tirar todo o proveito das ferramentas. E esta quantidade de tempo não é compatível com os prazos de entrega com que as empresas têm que lidar.

Quando se cogita a possibilidade de adotar ferramentas CASE em um ambiente onde já existem metodologias consolidadas, há o receio, pois o choque cultural já é esperado nessas situações, e geralmente trata-se de um conjunto de ferramentas, pois todo o processo de software é abordado. Sabendo disso, é possível ter ideia da dimensão do impacto sobre as rotinas de desenvolvimento de software em uma empresa. Por este motivo, deve ser considerado um estudo detalhado de viabilidade. E se aprovado, existem todo um processo de implantação

para acomodar o uso das ferramentas, que envolve mudanças drásticas nas rotinas de desenvolvimento, talvez até remodelação dos processos, treinamento intensivo dos envolvidos e tudo deve acontecer com atenção e acompanhamento em como vão reagir os desenvolvedores, gerentes de projeto e os clientes (CASTELLANI, 1995).

Por outro lado, baseando-se em Silva & Videira (2001), em um ambiente onde a implantação de ferramentas CASE for bem sucedida, a médio prazo, é possível perceber a facilidade de manutenção e desenvolvimento do produto por conta da padronização do processo de produção e das características dos produtos. A recursividade de elementos e reutilização de recursos dentro e entre projetos, causa ganho considerável em desempenho de produtividade e simplificação do código fonte, que facilita a interação de mais pessoas nos projetos, diminui o tempo de adaptação e treinamento para o desenvolvedor, que são outros fatores a favor do ganho de produtividade. Há a diminuição do esforço por parte do desenvolvedor em atividades redundantes, pois a ferramenta além de automatizar a geração de código fonte, automatiza também a documentação do produto, mantendo as informações atualizadas e sincronizadas com a situação do produto. A qualidade final do produto é superior, pois a padronização do processo de produção impõe maior rigor obrigando o desenvolvimento a tomar uma abordagem mais estruturada, em outras palavras, o desenvolvedor passa a ter etapas a cumprir para realizar determinados trabalhos, o que aumenta a detecção de erros e garante a funcionalidade do sistema.

Outras vantagens podem ser citadas em relação à padronização do processo, principalmente em relação ao controle de tempo, ganha-se a facilidade de mensuração e estimativas de tempo para a realização de trabalhos, maior previsão que por sua vez favorece ao atendimento e definição de prazos. Nota-se melhor mensuração de demanda de recurso, que possibilita a alocação de recursos mais precisa e assim, otimiza a definição de valores e redução de custos.

Todos estes fatores favoráveis ao controle de tempo e recurso, são forças que propiciam a melhoria da qualidade de gerenciamento do projeto. A agilidade e facilidade de desenvolvimento possibilitam a prototipagem, que refina ainda mais a qualidade do produto final.

### 3 – METODOLOGIA

Como já foi dito no subitem 1.3, a problemática desse trabalho gira em torno da crescente demanda de software no Brasil, contrastada pela escassez de profissionais qualificados de TI, sendo assim, o objetivo desse presente estudo é mostrar que as empresas de desenvolvimento de software podem aumentar a produtividade de suas equipes, uma vez que consigam combinar as Metodologias Ágeis e ferramentas CASE certas em seus processos, além de melhorar outras características, tais como a qualidade de seus produtos e a estimativa de prazos.

Para que fosse verificada na prática a aplicação dos conceitos abordados nesse trabalho, foi desenvolvido um questionário (Apêndice A) com o objetivo de colher informações que seriam analisadas posteriormente, com foco no uso de Metodologias Ágeis e ferramentas CASE nas empresas da região.

Para uma melhor organização das respostas, as questões foram esquematizadas em três grupos, as que começam com 1 são voltadas às Metodologias Ágeis, em especial o Scrum; as que começam com 2 relacionam-se com as ferramentas CASE; por fim, as questões que começam com 3 referem-se à combinação de Métodos Ágeis e ferramentas CASE.

A opção adotada foi uma análise qualitativa, visando o aprofundamento na experiência das empresas da região, e não apenas a geração de estatísticas de uso das metodologias e ferramentas abordadas.

Foram selecionadas seis empresas, sendo que cinco dessas são do ramo de software, e uma é do ramo financeiro, mas conta com uma equipe voltada para desenvolver e dar suporte à software.

O questionário foi desenvolvido através do Google Drive e sua ferramenta de formulário, que permite disponibilizar um link via internet e armazena as resposta na nuvem. Esse link foi enviado às empresas através de e-mail.

As pessoas que responderam o questionário atuam como coordenadores de desenvolvimento, e lideram equipes de tamanhos variáveis, ou já tiveram essa experiência.

### 3.1 – ANÁLISE DE RESULTADOS

Nesse item, será feita uma análise das informações colhidas com o questionário. Para uma melhor organização, separamos a análise em três partes, seguindo o mesmo raciocínio que foi proposto no questionário.

#### 3.1.1 – Análise de uso das metodologias

Todas as empresas que responderam o questionário utilizam metodologias ágeis, sendo o framework Scrum a opção mais aderida. A necessidade de uso de uma metodologia ágil surgiu de motivos como dificuldade de controlar o fluxo dos trabalhos, aumento de produtividade, contemplar prazos estabelecidos e organização, sendo que este foi o motivo mais citado.

Apesar de todas as empresas utilizarem Scrum, somente uma afirmou estar usando o Scrum em sua totalidade, essa empresa é nova e eles afirmam “estamos adequando a empresa ao SCRUM e não o SCRUM à empresa”. Esse ponto de vista é interessante, e gerou uma dúvida, será que o Scrum só pode ser adotado quando uma empresa está começando a iniciar seus trabalhos? O ponto comum em que se pode chegar, é que as empresas que já estão no mercado procuram adaptar o Scrum à empresa, para amenizar as “dores de cabeça” que as mudanças causam. Outra prática adotada é mesclar Scrum com outras metodologias, seja ágil ou tradicional, um exemplo é um misto de Scrum e Kanban<sup>4</sup> adotado por uma das empresas entrevistadas.

Percebeu-se que uma das maiores dificuldades é a de formar equipes auto-organizáveis como propõe o Scrum, em contrapartida a Reunião Diária é muito bem assimilada pelas empresas, tendo como maior fruto a melhoria na comunicação da Equipe, cumprindo sua função com maestria quando se trata de impedimentos.

---

<sup>4</sup> Kanban é um termo de origem japonesa e significa literalmente “cartão” ou “sinalização”. É um conceito relacionado com a utilização de cartões (post-it e outros) para indicar o andamento dos fluxos de produção em empresas de fabricação em série. Nesses cartões são colocadas indicações sobre uma determinada tarefa, por exemplo, “para executar”, “em andamento” ou “finalizado”. (Disponível em: <<http://www.significados.com.br/kanban/>>. Acesso em: 12 nov. 2013.)

O quadro de tarefas também foi citado como um dos grandes pontos positivos, pois tem a capacidade de tornar o andamento das atividades muito visível, o que gera um sentimento de comprometimento coletivo.

Em relação às melhorias que foram alcançadas, uma resposta em particular chamou a atenção, esta será exposta na íntegra, pois resume todas as outras respostas da questão específica sobre melhorias:

“Redução do tempo de resposta, organização no trabalho, redução de retrabalho e ganho na produtividade. Com organização o próprio time se sente mais a vontade para trabalhar com uma ‘visão de horizonte’. Quando se tem ideia de ‘onde se quer chegar’, o trabalho se torna prazeroso, sendo assim, muito mais produtivo e efetivo.”

Quando perguntado sobre as dificuldades enfrentadas ao adotar a metodologia, o maior problema é a assimilação dos conceitos e a mudança que se faz necessária para implantar os processos propostos. A cultura organizacional também se torna um grande empecilho. Mas todos disseram em uníssono que a relação custo-benefício é positiva para a empresa, ou seja, o custo que a adoção da metodologia trará será compensado por seus benefícios.

Por fim, todas as pessoas que responderam o questionário, concluíram que metodologias ágeis como o Scrum funcionam sim, mas é necessário observar particularidades que a empresa possui e moldar a teoria para o cenário da empresa. Para funcionar, é necessário um engajamento coletivo e esforço para entender o que cada processo significa dentro do macro-funcionamento da proposta.

### **3.1.2 – Análise de uso das ferramentas CASE**

As ferramentas CASE são, para todos os entrevistados, importantes para o controle e a aplicação das regras definidas pela equipe. Tais ferramentas permitem que a adoção das regras seja de forma fácil, pois permitem uma flexibilidade no modo como os envolvidos irão usar os conceitos aplicados.

As respostas recebidas foram focadas mais na parte gerencial do que técnica, as empresas seguiram dois caminhos diferentes, o primeiro foi adotar uma ferramenta de terceiro, o TFS (Team Foundation Server) da Microsoft, e o segundo

foi desenvolver um software próprio adaptado às necessidades da empresa e que faça o gerenciamento dos projetos desenvolvidos, juntamente com suas atividades. Inclusive, uma das respostas deixa claro que seria praticamente impossível gerenciar os projetos na ausência da ferramenta desenvolvida.

Sobre ferramentas que geram código fonte automaticamente, somente uma empresa utiliza e é o Visual Studio da Microsoft, mas o código passa por uma customização após a geração.

Sobre as melhorias trazidas pelas ferramentas, o destaque fica pela documentação que as mesmas geram vinculada às atividades desenvolvidas.

E sobre as dificuldades, se limitam ao domínio da ferramenta, sendo que quanto maior o conhecimento, menor a dificuldade enfrentada. Uma dica que pode ser considerada interessante é definir recursos dentro da Equipe que possam transmitir o conhecimento das ferramentas para os outros membros.

### **3.1.3 – Análise da combinação de metodologias e ferramentas CASE**

A combinação de uma Metodologia Ágil com ferramentas CASE é vista com bons olhos por todas as empresas participantes do questionário. Sem dúvida os resultados poderão ser positivos.

Após a análise dos dados coletados com o questionário, concluí-se que esta combinação pode ser extremamente positiva, pois é com esta combinação que se obtêm um ganho na agilidade, uma vez que as ferramentas de apoio concentram todo o gerenciamento que é feito de forma prática, rápida e Online, sendo possível criar visões como quantidade de reprovos e de bugs, cumprimento de prazos por desenvolvedor, entre outras análises.

Desta forma, a metodologia alinhada com uma boa ferramenta tem tudo para que a equipe de desenvolvimento possa ser mais eficiente, seja por que os integrantes da Equipe estarão focados no que deve ser feito, como por poder acompanhar em tempo real o que está indo para o lado errado e corrigir de forma preventiva.

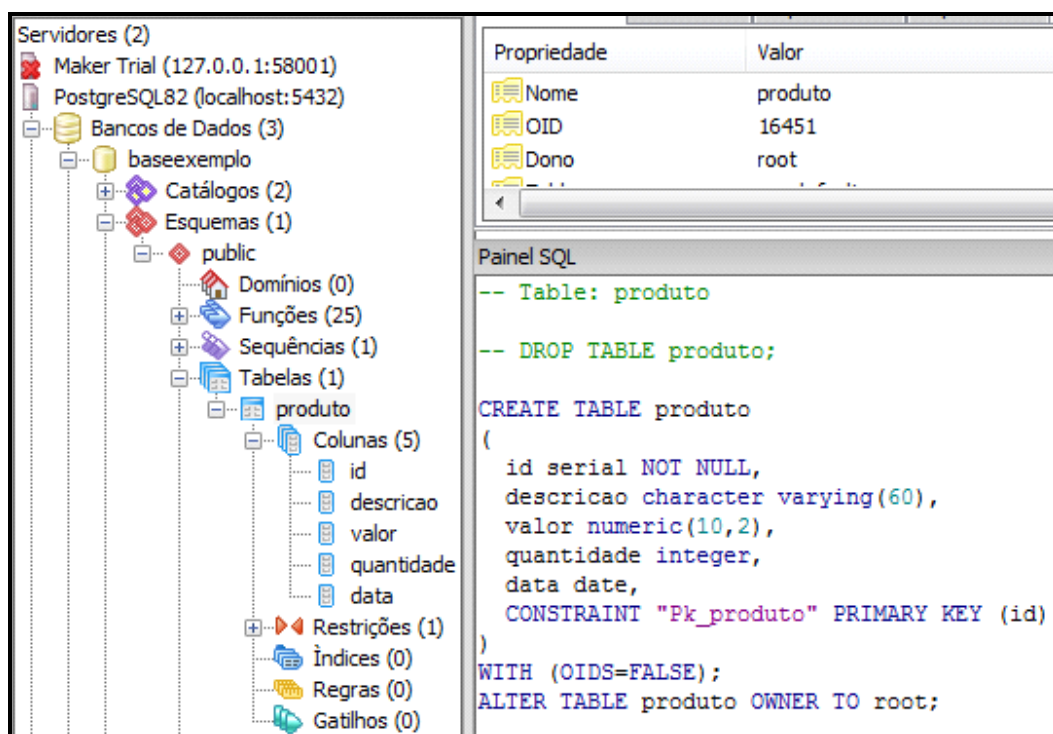
### 3.2 – EXEMPLO DE UTILIZAÇÃO DE UMA FERRAMENTA CASE

Para exemplificar um caso de uso de uma ferramenta CASE de desenvolvimento, foi escolhido o software Maker da empresa Softwell. A Softwell é uma empresa brasileira com mais de oito anos de atuação no mercado e com abrangência nacional e também em ambientes internacionais. O Maker All é um de seus produtos, desenvolvido com o intuito de agilizar o processo de produção dos recursos de automação das empresas, de modo que os esforços sejam focados na regra de negócios. Maker All pode gerar aplicativos para plataformas web e mobile.

A versão utilizada para a amostra é o Maker 2.7 trial. A versão trial possui limitações como por exemplo suporte apenas para o banco de dados PostgreSQL 8.0 a 8.2. Neste exemplo será demonstrado como gerar um formulário de cadastro a partir de uma tabela já existente no banco de dados.

Inicialmente, em um banco de dados PostgreSQL já devidamente instalado e funcionando, preparamos uma tabela com o nome “produto”, conforme Figura 6.

**Figura 6 – Criação da tabela “produto” no banco de dados PostgreSQL**



Fonte: O autor.

Ao executar o Maker, a tela inicial, Figura 7, é exibida oferecendo opções como iniciar um novo projeto ou abrir um já existente, selecione Novo Projeto.

**Figura 7 – Tela inicial do Maker**



**Fonte:** O autor

Para iniciar um novo projeto é necessário informar os dados para conexão de banco, nome do projeto e informações sobre a categoria e razão do projeto. Esse processo é guiado pelo assistente de criação de projetos, mostrado na Figura 8.



**Figura 8 – Assistente de criação de projetos Maker**

Assistente de criação de projetos

Cria um novo arquivo de projeto com as configurações de acesso ao banco de dados e as propriedades do projeto.

**1** Conectando ao Banco de Dados

Tipo de banco de dados  
PostgreSQL

Servidor  
localhost

Nome do banco de dados  
baseexemplo

Login do usuário  
postgres

Senha  
xxxxxx

[Configurações avançadas](#)

Conectar >>

**2** Projeto

☒ Criar novo projeto ☐ Configurar um projeto existente

Nome do projeto  
Projetoexemplo

Sigla  
PEX

Quais as categorias às quais pertencem o novo projeto?

☒ Uso Geral

Para que serve e como será utilizado o seu novo projeto?

Demonstração de criação de formulário de cadastro

OK Cancelar

**Fonte:** O autor

Em seguida, o assistente de criação de projetos irá perguntar onde salvar o arquivo que vai conter o projeto e abri-lo logo após. Por padrão, todos os projetos Maker possuem usuário e senha para ser acessado, o acesso padrão é usuário *master* e senha *1*.

Agora temos o projeto criado e pronto para ser montado. Na Figura 9 vemos a tela principal do projeto e a próxima opção a ser selecionada, a criação de formulário.

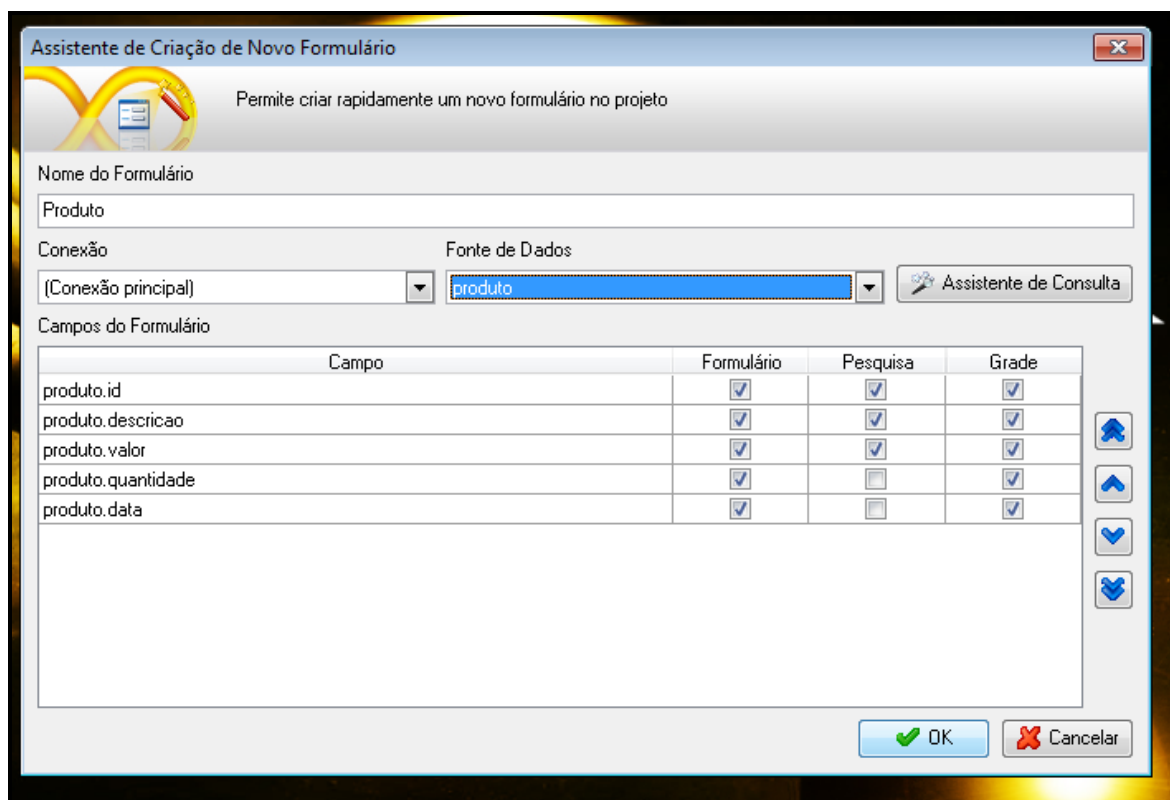
**Figura 9 – Tela principal do projeto Maker**



**Fonte:** O autor

Como vemos na Figura 10, o assistente de criação de formulário solicita o nome do formulário a ser criado e com base em qual tabela será utilizado. Ao informar a tabela, o assistente lista os campos automaticamente para serem selecionados e utilizados no formulário e na pesquisa.

**Figura 10 – Assistente de criação de formulário Maker**



Assistente de Criação de Novo Formulário

Permite criar rapidamente um novo formulário no projeto

Nome do Formulário: Produto

Conexão: (Conexão principal) Fonte de Dados: produto

Assistente de Consulta

Campos do Formulário

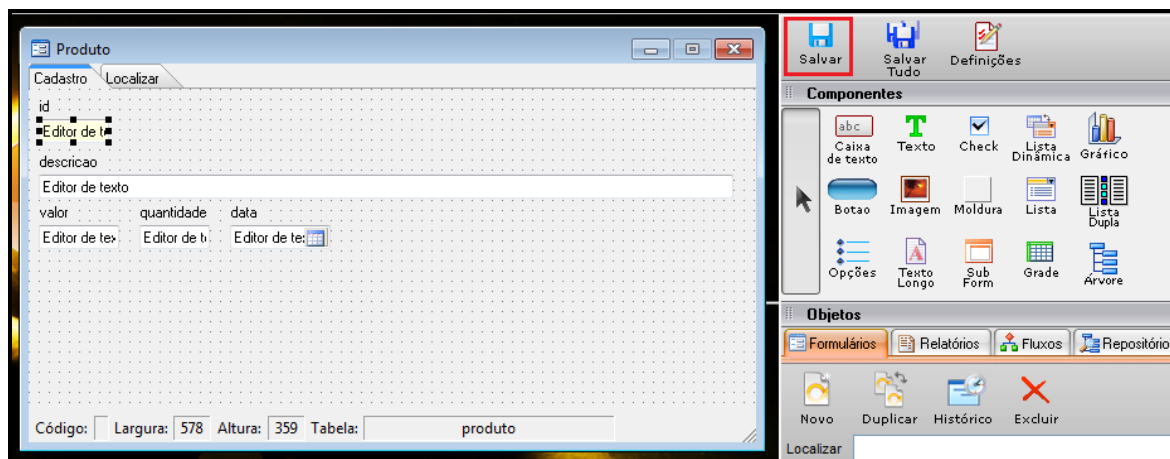
Campo	Formulário	Pesquisa	Grade
produto.id	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
produto.descricao	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
produto.valor	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
produto.quantidade	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
produto.data	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

OK Cancelar

Fonte: O autor

Ao confirmar, o formulário será criado e aberto no editor para que possa ser modelado conforme necessário, à direita há uma caixa de componentes disponíveis para serem adicionados no formulário aumentando sua funcionalidade. Para salvar as alterações, clique no botão Salvar, como demonstra a Figura 11.

**Figura 11 – Editor de formulário Maker**



Produto

Cadastro Localizar

id: Editor de texto

descricao: Editor de texto

valor: Editor de texto quantidade: Editor de texto data: Editor de texto

Código: Largura: 578 Altura: 359 Tabela: produto

Salvar Salvar Tudo Definições

Componentes

- Caixa de texto
- Texto
- Check
- Lista Dinâmica
- Gráfico
- Botao
- Imagem
- Moldura
- Lista
- Lista Dupla
- Opções
- Texto Longo
- Sub Form
- Grade
- Arvore

Objetos

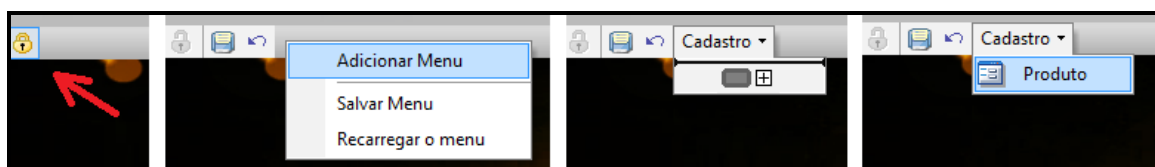
- Formulários
- Relatórios
- Fluxos
- Repositório
- Novo
- Duplicar
- Histórico
- Excluir

Localizar

Fonte: O autor

Neste momento já temos o formulário criado e pronto para usá-lo, porém, é necessário criar um acesso para a tela através do menu, seguindo os passos: desbloquear a edição do menu, criar item de menu e arrastar o formulário de cadastro para dentro do item de menu criado, conforme Figura 12.

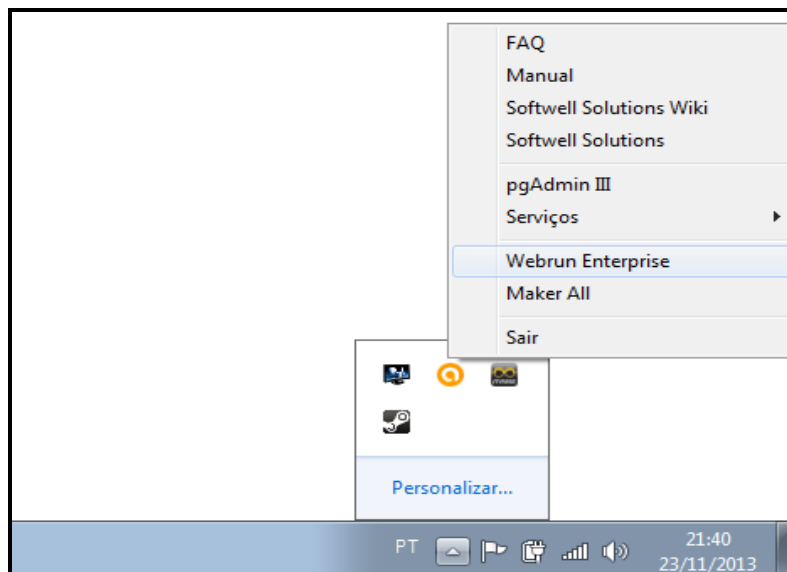
**Figura 12 – Criação de menu Maker**



**Fonte:** O autor

O projeto está pronto para ser testado, os projetos Maker rodam em um serviço servidor web chamado Webrun que pode ser acessado através do ícone do maker no system tray. Veja como acessar o serviço na Figura 13.

**Figura 13 – Serviço Webrun Maker**



**Fonte:** O autor

O Webrun abrirá pelo browser padrão do computador e exibirá uma lista de projetos cadastrados, selecione o projeto "Projetoexemplo" e clique em acessar. O acesso é o mesmo solicitado pelo Maker ao abrir um projeto, usuário *master* e senha 1. Veja a Figura 14 para mais detalhes.

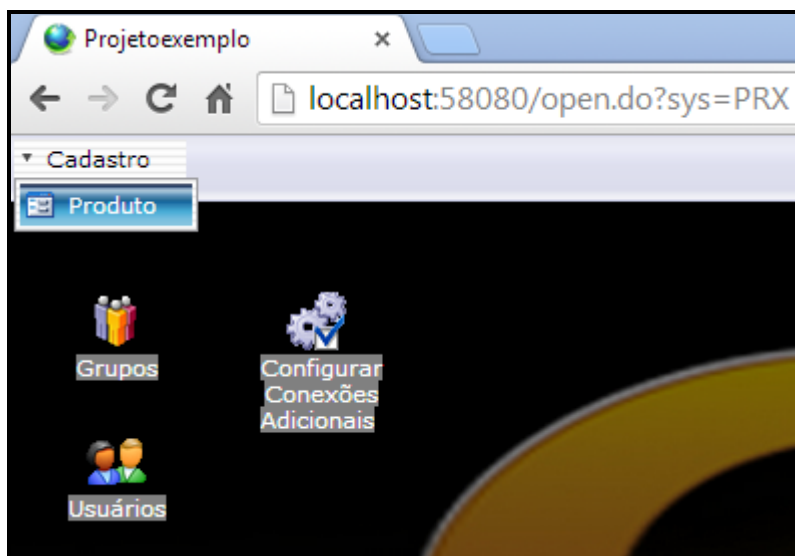
**Figura 14 – Login no projeto Maker**



**Fonte:** O autor

Na Figura 15 temos a tela inicial com o menu do formulário criado.

**Figura 15 – Tela inicial do programa resultado do projeto Maker**



**Fonte:** O autor

O resultado final desse exemplo de utilização do Maker pode ser visualizado ao acessar o menu produto, pois o formulário de cadastro resultante do projeto Maker é exibido conforme mostrado na Figura 16. Esse mesmo formulário tem uma opção de filtro que pode ser acessada pela aba “Localizar”, vemos a tela de filtro na Figura 17.

**Figura 16 – Tela de cadastro criada no Maker**

Produto - Google Chrome

localhost:58080/form.jsp?sys=PRX&action=openform&formID=3&align=0&r

Cadastro Localizar

id  
1

descricao  
Tenis esportivo

valor  
75

quantidade  
13

data  
22/11/2005

Fonte: O autor

**Figura 17 – Tela de filtro do cadastro criado no Maker**

Produto - Google Chrome

localhost:58080/form.jsp?sys=PRX&action=openform&formID=3&align=0&r

Filtro Localizar

Parâmetros para a busca

id  
=

descricao  
Iniciando com

valor  
=

data  
=

Resultados

id	descricao	valor	quantidade	data
1	Tenis esportivo	75	13	22/11/2005

Fonte: O autor

## 4 – CONCLUSÃO

Após várias consultas a diversas fontes de informações, constatamos o que já era evidente, a demanda de software, não somente de software, mas de TI em geral, não para de crescer; mas a oferta de profissionais para suprir essa demanda não está acompanhando-a.

Então o sentimento foi de que o caminho era realmente esse, era preciso estudar formas as quais tivessem a capacidade de amenizar os efeitos dessa defasagem profissional.

A nossa pesquisa teve seu foco definido, estudando a metodologia ágil Scrum e algumas ferramentas CASE que davam apoio à equipe de desenvolvimento de software, tanto na parte operacional quanto na parte gerencial.

Grande parte do conteúdo bibliográfico referente ao framework Scrum, foi retirado do Guia do Scrum, documento elaborado pelos próprios autores dessa metodologia, Ken Schwaber e Jeff Sutherland.

Sobre as ferramentas CASE, foi dada mais ênfase à teoria, uma vez que existe uma vasta gama dessas ferramentas no mercado.

Já na prática, o objetivo foi buscar uma experiência mais profunda com tudo aquilo que vimos na teoria, por isso elaboramos um questionário que nos permitiu ter contato com pessoas que usavam tais metodologias e ferramentas em seu dia-a-dia. Sem dúvida foi experiência muita gratificante.

Em suma, o trabalho atingiu seus objetivos, pois baseado nas respostas das seis empresas entrevistadas, pudemos confirmar que o uso combinado de metodologias ágeis e ferramentas CASE pode ser positivo para as equipes de desenvolvimento de software, levando-as a uma melhoria contínua de seus processos, como uma comunicação eficaz, prazos mais precisos, software de maior qualidade, contribuindo assim para que a equipe consiga alcançar seus objetivos.

Em um trabalho futuro, a pretensão é de vivenciar essa experiência de combinar Metodologias Ágeis e Ferramentas CASE, desenvolvendo um estudo de caso da aplicação dos conceitos abordados por este trabalho. Assim, teremos um acompanhamento de todas as mudanças geradas, sejam elas positivas ou negativas.

## REFERÊNCIAS BIBLIOGRÁFICAS

BECK, Kent; *et al.* **Manifesto para o desenvolvimento ágil de software**, 2001. Disponível em: <<http://manifestoagil.com.br/>>. Acesso em 29 out. 2013.

BERBERT, Lúcia. **Softex vê perdas de R\$ 115 bi até 2020 por falta de profissionais de TI**, 2012. Disponível em: <<http://telesintese.com.br/index.php/plantao/19895-softex-ve-perdas-de-r-115-bi-ate-2020-por-falta-de-profissionais-de-ti>>. Acesso em 24 out. 2013.

CASTELLANI, Márcia Reiff. **ASPECTOS CRÍTICOS NA IMPLEMENTAÇÃO DE FERRAMENTAS CASE**. Artigo de mestrado, Caderno de Pesquisas em Administração nº1, 1995.

FILHO, Dairton Luiz Bassi. **Experiências com desenvolvimento ágil**. Dissertação (Mestrado em Ciência da Computação), Universidade de São Paulo, SP, 2008.

GIBBS, W. **Software's Chronic Crisis**. Scientific American Magazine (Set. 1994).

KNIBERG, Henrik. **Scrum and XP from the Trenches**. Toronto: C4Media, 2007.

LASKOSKI, Jackson. **Os novos tempos e o velho problema da falta de mão de obra qualificada**, 2010. Disponível em: <<http://www.jack.eti.br/os-novos-tempos-e-o-velho-problema-da-falta-de-mao-de-obra-qualificada/>>. Acesso em: 24 out. 2013.

MARRARA, Giuseppe. **Estudo revela demanda crescente por profissionais de tecnologia no Brasil, mas há falta de mão de obra qualificada**, 2013. Disponível em: <<http://globalnewsroom.cisco.com/easyir/BR/pt/local/press-release/Estudo-revela-demanda-crescente-por-profissionais-de-tecnologia-no-Brasil-mas-ha-996920.html>>. Acesso em: 24 out. 2013.

MATTSSON, M. **Object-oriented Frameworks - A survey of methodological issues**, Licentiate Thesis, Department of Computer Science, Lund University,



CODEN: LUTEDX/(TECS-3066)/1-130/(1996), also as Technical Report, LU- CS-TR: 96-167, Department of Computer Science, Lund University, 1996.

MATTSSON. M. **Evolution and Composition Object-Oriented Frameworks**, PhD Thesis, University of Karlskrona/Ronneby, Department of Software Engineering and Computer Science, 2000.

MOURA, Viviane. **Mercado brasileiro de software vai crescer 400% em 10 anos**, 2012. Disponível em <<http://www.cepromat.mt.gov.br/index.php/mnu-noticias/424-mercado-brasileiro-de-software-vai-crescer-400-em-10-anos>>. Acesso em 24 out. 2013.

REDAÇÃO. **Brasil é o 53º em leitura e o 57º em matemática em ranking do Pisa 2009**, 2010. Disponível em: <<http://www.todospelaeducacao.org.br/comunicacao-e-midia/noticias/12239/brasil-e-o-53-em-leitura-e-o-57-em-matematica-em-ranking-do-pisa-2009/>>. Acesso em: 24 out. 2013.

SCHACH, Stephen R. **Object-oriented Software Engineering 1st ed.** McGraw-Hill (2008).

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do Scrum**, 2013. Disponível em: <<https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide-Portuguese-BR.pdf#zoom=100>>. Acesso em 25 out. 2013.

SILVA, Alberto Manoel Rodrigues; VIDEIRA, Carlos Alberto Escaleira. **UML, Metodologias e Ferramentas CASE**. Linguagem de Modelação UML, Metodologias e Ferramentas CASE na Concepção e Desenvolvimento de Software. (Abr. 2001).

SOFTWELL, **Softwell Maker All.** Disponível em: <<http://www.softwell.com.br/produto/softwell-maker/>>. Acessado em 07 Out. 2013.

SUTHERLAND, Jeff. **The Scrum Papers**, 2007. Disponível em: <<http://faculty.ksu.edu.sa/zohair/Documents/CSC541/Chap2-SWE%20processes/Scrum%20Papers.pdf>>. Acesso em 01 nov. 2013.

VENCESLAU, Marcelo. **Falta de profissionais de TI se agravará no Brasil, diz IDC**, 2013. Disponível em: <<http://exame.abril.com.br/carreira/noticias/falta-de-profissionais-de-ti-se-agravara-no-brasil-diz-idc>>. Acesso em: 24 out. 2013.

WEINRICH, Jair. **GRAHL Everaldo-Software de apoio a avaliação e seleção de ferramentas case baseado na norma ISO/IEC 14102**. Artigo SEMINCO FURB-Universidade Regional de Blumenau, 1999.

## **APÊNDICE A – Questionário para coleta de dados**

Este questionário visa dar embasamento ao presente trabalho, buscando identificar a aplicação prática dos conceitos expostos.

1.1 – A empresa utiliza Scrum ou alguma Metodologia Ágil? Se sim, explique como surgiu essa necessidade.

1.2 – Quais recursos da Metodologia você utiliza?

1.3 – Quais foram as melhorias trazidas pela Metodologia Ágil, e de que forma isso impactou no processo de desenvolvimento de software?

1.4 – Quais foram as dificuldades enfrentadas ao adotar a Metodologia? As limitações podem ser contornadas pelos envolvidos, ou comprometem o desenvolvimento de alguns processos? Em relação à custo-benefício, os problemas relevantes na mudança são compensados pelas vantagens e facilidades proporcionadas?

1.5 – Scrum, ou a Metodologia utilizada, funciona? Porquê? Dê uma nota de 0 a 10 para Scrum.

2.1 – A empresa utiliza ferramentas de apoio aos processos de gerenciamento e desenvolvimento de software? Qual a importância dessas ferramentas nas atividades internas da empresa?

2.2 – A empresa utiliza alguma ferramenta de apoio ao desenvolvimento de software com a capacidade de gerar código fonte automaticamente?

2.3 – Quais foram as melhorias trazidas pelas ferramentas de apoio, e de que forma isso impactou no processo de desenvolvimento de software?

2.4 – Quais foram as dificuldades enfrentadas ao adotar as ferramentas? As limitações podem ser contornadas pelos envolvidos, ou comprometem o desenvolvimento de alguns processos? Em relação à custo-benefício, os problemas relevantes na mudança são compensados pelas vantagens e facilidades proporcionadas?

3.1 – As ferramentas se adaptam aos processos ágeis de desenvolvimento da empresa? Qual a relação das ferramentas de apoio com o método ágil adotado pela empresa?

3.2 – Combinar Metodologias Ágeis com Ferramentas de Apoio pode dar resultados positivos? Conte-nos um pouco da sua experiência.