UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Karine Teixeira Porto, Leandro Antunes da Silva

DIFICULDADES NA IMPLANTAÇÃO DO *SCRUM*: ESTUDO DE CASOS MÚLTIPLOS

Alfenas, 04 de Fevereiro de 2014.

UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DIFICULDADES NA IMPLANTAÇÃO DO *SCRUM*: ESTUDO DE CASOS MÚLTIPLOS

Karine Teixeira Porto, Leandro Antunes da Silva

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Alfenas como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Eduardo Gomes Salgado

Alfenas, 04 de Fevereiro de 2014.

Karine Teixeira Porto, Leandro Antunes da Silva

DIFICULDADES NA IMPLANTAÇÃO DO *SCRUM*: ESTUDO DE CASOS MÚLTIPLOS

A Banca examinadora abaixo-assinada aprova a monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Alfenas.

Prof. Rodrigo Martins Pagliares

Universidade Federal de Alfenas

Prof. João Antônio Silva

Universidade Federal de Alfenas

Prof. Eduardo Gomes Salgado (Orientador)

Universidade Federal de Alfenas

Alfenas, 04 de Fevereiro de 2014.

AGRADECIMENTO

Primeiramente gostaria de agradecer a Deus, por ser presença constante na minha vida, por ter me dado familiares, amigos e todas as pessoas que já passaram pela minha vida. Por me fazer acreditar que posso seguir adiante e ter fé.

Gostaria de agradecer aos meus pais João e Mersia, por todos os ensinamentos, compreensão, apoio, oportunidades e principalmente por todo o amor. Sem vocês nada disso seria possível, ou faria sentido. Obrigada a minha família de uma forma geral e a família do Le que me acolheu tão bem.

A todos os amigos que estiveram comigo nessa caminhada e que continuarão, muito obrigada!

Ao Eduardo, nosso orientador, que foi um orientador de verdade, se fez presente, nos ajudou e puxou a orelha quando preciso. Muito obrigada mesmo!

Ao pessoal do NTI por todo o aprendizado e oportunidade de estagiar por quase dois anos no melhor lugar para se trabalhar. E especialmente a Bia, pela força e conversas e ao Cléber, pelos ensinamentos e situações vividas.

A todos os que participaram respondendo nosso questionário, sem vocês não seria possível a conclusão deste trabalho.

Aos professores, amigos e colegas da UNIFAL-MG muito obrigada pelos ensinamentos, momentos de superação e alegrias.

E por fim, mas não menos importante, ao Le, meu amigo, amor, companheiro, parceiro de todas as horas e a pessoa que eu escolhi para a minha vida. Obrigada por tudo, sempre!

Karine

AGRADECIMENTO

Aos meus pais, Odair e Edna, por todos os ensinamentos dados, que me moldaram naquilo que hoje sou, por acreditarem em mim e incentivarem todas as minhas decisões e por todo amor a mim dedicado.

À minha amiga-irmã, Ludmila, que mesmo com toda a distância, sempre se fez e faz presente em minha vida, me apoiando em todos os momentos.

À minha amiga e companheira Ka, por todos as etapas que me ajudou a superar no curso e que me ajudará em todas as outras que estão por vir. Também pelo esforço conjunto que tivemos, pois sem ela, este trabalho provavelmente ainda estaria em fase de desenvolvimento. Gostaria de agradecer também a família da Ka por ter me recebido tão bem e sempre estarem ao nosso lado quando precisamos.

Aos meus grandes amigos que sempre estiveram ao meu lado nos bons e maus momentos, e cujas amizades sempre foram essenciais em minha jornada.

Ao Eduardo, nosso orientador, que tornou possível a realização deste trabalho. Sempre nos apoiou e também deu seus puxões de orelha quando precisávamos.

Aos professores e colegas de trabalhos por onde passei, pelo conhecimento comigo compartilhado, que serão um diferencial em minha experiência profissional.

A todos aqueles que nos ajudaram a concluir este trabalho respondendo ao nosso questionário.

Leandro

"Valeu a pena? Tudo vale a pena se a alma não é pequena."
Fernando Pessoa (1934)

RESUMO

O processo de desenvolvimento de *software* vem evoluindo e se aperfeiçoando com o passar dos anos. Antigamente era visto como um processo teórico, mas a prática comprovou que não é essa a realidade, e assim, muitos projetos que utilizavam essa abordagem tradicional falharam e ainda falham. Na tentativa de conseguir melhorias, surgiram várias formas de controlar o processo, dentre elas o *Scrum*, que será o método abordado neste trabalho. Muitas organizações estão adotando métodos ágeis em seus processos de desenvolvimento de *software*, mas cada organização encontra dificuldades específicas na migração de métodos tradicionais para métodos ágeis. Neste intuito, o presente trabalho tem como principal contribuição, verificar as principais dificuldades de implantação do *Scrum* em empresas de três origens distintas: uma empresa privada, uma empresa pública e uma empresa que inicialmente foi incubada em uma universidade. Neste estudo de casos múltiplos, serão levantadas as principais dificuldades que cada empresa encontrou no Planejamento, Especificação, Implantação e Entrega de Produtos definidos pelo *Scrum*. Espera-se que os resultados deste trabalho possam auxiliar empresas que estejam implantando métodos ágeis em seus processos, ajudando-as a prever e combater os possíveis problemas que serão encontrados ao longo do processo de migração.

Palavras-Chave: [*Scrum,* Métodos Ágeis, Estudo de Caso]

ABSTRACT

The process of software development is evolving and has been improving over the years. Formerly it was seen as an theoretical process, but the practice proved that this is not reality, and so many projects that used this traditional approach failed and still fail. In an attempt to achieve improvements, there are several ways to control the process, among them *Scrum*, which is the method discussed in this paper. Many organizations are adopting agile processes in software development, but each organization has specific difficulties in migrating from traditional methods to agile methods. To this end, this paper's main contribution is to check the main difficulties of implementation of *Scrum* in companies of three different sources: a private company, a public company and a company that was initially incubated at a university. In this multiple case study, the main difficulties found in every company in Planning, Specification, Implementation and Delivery of products defined by *Scrum* will be raised. It is hoped that the results of this work can help companies that are implementing agile in their processes, by anticipating and dealing with the possible problems that are encountered along the migration process.

Keywords: Scrum, Agile Methods, Study Case.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIAÇÕES

PIB	Produto Interno Bruto
XP	Extreme Programming
TI	Tecnologia da Informação
DSDM	Método de Desenvolvimento de Sistemas Dinâmicos
TDD	Desenvolvimento Dirigido por Teste

SUMÁRIO

1
Introdução

Este capítulo apresenta alguns detalhes sobre a caracterização do problema, bem como objetivos, justificativa e estrutura do trabalho.

1.1 Caracterização do Problema

Um projeto de implantação de um sistema de software necessita do equilíbrio de três elementos: pessoas, processos e tecnologia. Esta é a tríade de sustentação, execução e entrega das estratégias nas empresas.

A história do triângulo tem sido implacável no sentido de que um avanço na tecnologia necessariamente exige uma mudança nas pessoas, que fatalmente acarretará impactos em processos (YOSHIMA, 2007). Com isso, os métodos ágeis ganharam maior espaço pois valorizam mais as pessoas e a tecnologia, tirando de foco a prática de processos que utilizam de documentação pesada e baixa produção. Sendo assim, a criação de novos *softwares* cresce, aumentando a competitividade por mercado entre as empresas e consequentemente, diminuição dos preços.

Segundo pesquisa realizada pela Associação Brasileira das Empresas de *Software* o ano de 2010 foi um ano de recuperação para o setor de TI no Brasil, que mostrou um crescimento da ordem de 21,3%. Especificamente os setores de *software* e serviços cresceram quase 24%, um pouco menos que o segmento de hardware. Entretanto, considerando-se que o mercado mundial de *software* e serviços apresentou um aumento discreto, da ordem de 0,5% em 2010, o Brasil terminou o ano em uma situação de destaque neste cenário, alcançando a 11ª posição no *ranking* mundial, tendo movimentado 19,04 bilhões de dólares, equivalente a 1,0% do PIB brasileiro daquele ano. Deste total, foram movimentados 5,51 bilhões de

dólares em *software*, o que representou perto de 2,2% do mercado mundial, e 13,53 bilhões de dólares em serviços relacionados.

Os sistemas de *software* em si estão passando a ser a ferramenta mais básica e fundamental dos processos econômicos e sociais. Atualmente, é improvável que exista alguma atividade econômica que não sofra nenhum efeito de um sistema de informação (CARVALHO, 2009). Logo a busca por rapidez, eficiência e qualidade é cada vez maior, o que nos leva novamente a utilização de métodos ágeis no desenvolvimento destes *softwares*, como por exemplo o uso do *Scrum*.

Sendo a ênfase do *Scrum* a comunicação, o trabalho em equipe, a flexibilidade e sempre fornecer *software* funcional e de forma incremental, há uma melhoria na qualidade do produto produzido (BARTON e CAMPBELL, 2007).

Uma vez que o *Scrum* propõe melhorias para o desenvolvimento de softwares e que suas diretrizes podem ser de grande utilidade quando bem empregadas, chegamos ao seguintes questionamentos: quais as maiores dificuldades encontradas por profissionais da área ao utilizar tal metodologia? Essas dificuldades são as mesmas em diferentes tipos de empresas?

1.2 Objetivos

1.2.1 Gerais

O objetivo geral deste trabalho é realizar um estudo sobre o método ágil Scrum, sua estrutura, principais características e principalmente as dificuldades encontradas durante sua implantação em empresas na área de Tecnologia da Informação.

1.2.2 Específicos

Outros exemplos de metodologias ágeis serão apresentados visando a um comparativo e até mesmo à demonstração do embasamento que este método possui. Será realizada uma pesquisa de campo em três empresas de Tecnologia da

Informação, com intuito de concretizar os estudos e obter resultados por meio de pesquisas sobre as dificuldades de implantação do *Scrum*. Cada uma delas possui origens diferentes: a primeira é uma empresa júnior, a outra é uma empresa de iniciativa pública, e a última partiu de iniciativa privada.

1.3 Justificativa

Métodos tradicionais consideram o processo de desenvolvimento como algo que pode ser completamente planejado e estimado, o que se mostrou errado na prática. O *Scrum* define o processo de desenvolvimento como um processo imprevisível e complicado que só pode ser avaliado em um progresso geral (SCHWABER, 1995).

O *Scrum* é uma metodologia direcionada para o planejamento e acompanhamento do projeto, sem entrar nos detalhes de como é feita a engenharia de software (SCHWABER, 2004). Atualmente muitas empresas utilizam o *Scrum* para outras áreas de desenvolvimento de produtos que não são baseadas somente em software.

1.4 Estrutura do Trabalho

Este trabalho está estruturado em seis capítulos. O primeiro capítulo é a introdução já apresentada, com a caracterização do problema, justificativa, objetivos e estruturação do trabalho. O próximo capítulo apresenta o método de pesquisa utilizado. Em seguida, é apresentado o conceito e alguns exemplos de métodos ágeis. O Capítulo 4 apresenta o conceito e as características do *Scrum*. E no Capítulo 5 será discutido o Estudo de Caso e no Capítulo 6 teremos a conclusão.

2
Métodos de Pesquisa

Este capítulo trará a definição e classificação da pesquisa. A sessão 2.2 apresentará a definição de Estudo de Caso e a sessão 2.3 o Estudo de Casos Múltiplos.

2.1 Considerações Iniciais

Para Miguel (2007), é importante definir e selecionar de forma adequada o método de pesquisa do trabalho. Isto pode ser justificado pela busca da melhor abordagem para endereçar as questões da pesquisa. Assim, há a necessidade de embasamento científico adequado sobre o método de pesquisa a ser adotado.

2.2 Estudo de Caso

Existem diversas estratégias de pesquisa, as quais apresentam vantagens e desvantagens, dependendo basicamente de três condições: (a) tipo de questão de pesquisa proposto; (b) extensão de controle que o pesquisador tem sobre eventos comportamentais efetivos; (c) grau de enfoque em acontecimentos históricos em oposição a acontecimentos contemporâneos. (YIN, 2001)

O estudo de caso representa a estratégia preferida quando as fronteiras entre o fenômeno contemporâneo e o contexto em que ele se insere não estão bem definidas, isto dentro de um contexto da vida real, ou então, tendo o pesquisador pouco controle sobre os eventos. (YIN, 2001)

O que diferencia o estudo de caso dos outros métodos de pesquisa é a sua capacidade de lidar com uma variedade de evidências muito grande. E suas questões de pesquisa são do tipo "como" e "por que". Porém, em algumas situações,

as questões "como" e "por que" podem não apontar para aquilo que o pesquisador deveria estudar. Nesses casos, estabelecer algumas proposições de estudo pode ajudar a conduzir a pesquisa para a direção certa. Cada proposição destina a atenção a alguma coisa que deveria ser examinada dentro do escopo do estudo (YIN, 2001).

Dentre os tipos de pesquisa, existem três classificações com base em objetivos gerais: exploratórias, descritivas e explanatórias.

O estudo de caso exploratório é considerado um tipo de estudo piloto que pode ser feito para testar as perguntas que guiam o projeto, hipóteses, e principalmente os instrumentos e procedimentos. Ao término do estudo exploratório, provavelmente algumas perguntas serão modificadas, retiradas ou acrescentadas, instrumentos que serão refinados, ou hipóteses que serão reformuladas, baseando-se no que teve êxito ou não. Mesmo sendo exploratório, haverá um planejamento cuidadoso, o mais detalhado possível, para que não haja desperdício de tempo, nem do pesquisador nem das pessoas envolvidas.

O estudo de caso descritivo tem por objetivo apresentar uma realidade desconhecida. Não procura estabelecer relações de causa e efeito, mas apenas mostrar a realidade como ela é, embora os resultados possam ser usados posteriormente para a formulação de hipóteses de causa e efeito. Pode mostrar, por exemplo, um professor fazendo uso inadequado da Internet, levando os alunos para o laboratório de informática para acessar uma página de texto sem links, numa atividade de leitura que poderia ser feita com menos desperdício de tempo com uma folha impressa na sala de aula. O estudo, no entanto, apenas descreveria o evento, sem preocupação de generalizar, sugerindo que seja um exemplo típico e que todos os professores fazem assim, nem de apontar relações de causa e efeito, sugerindo que o mau uso da tecnologia possa ser improdutivo.

O estudo de caso explanatório pode ser considerado o mais audacioso dos três, já que tem por objetivo não apenas descrever uma determinada realidade, mas também explicá-la em termos de causa e efeito. No exemplo acima, em vez de usar o caso de um único professor, pode mostrar dois, comparando um exemplo de mau uso da tecnologia com um exemplo adequado e tentar ver o impacto que isso pode ter na aprendizagem dos alunos. O estudo de caso explanatório pode também ter

como objetivo a confirmação ou generalização de determinadas proposições teóricas.

O método de estudo de caso pode ser implementado seguindo as atividades descritas na seguinte figura:

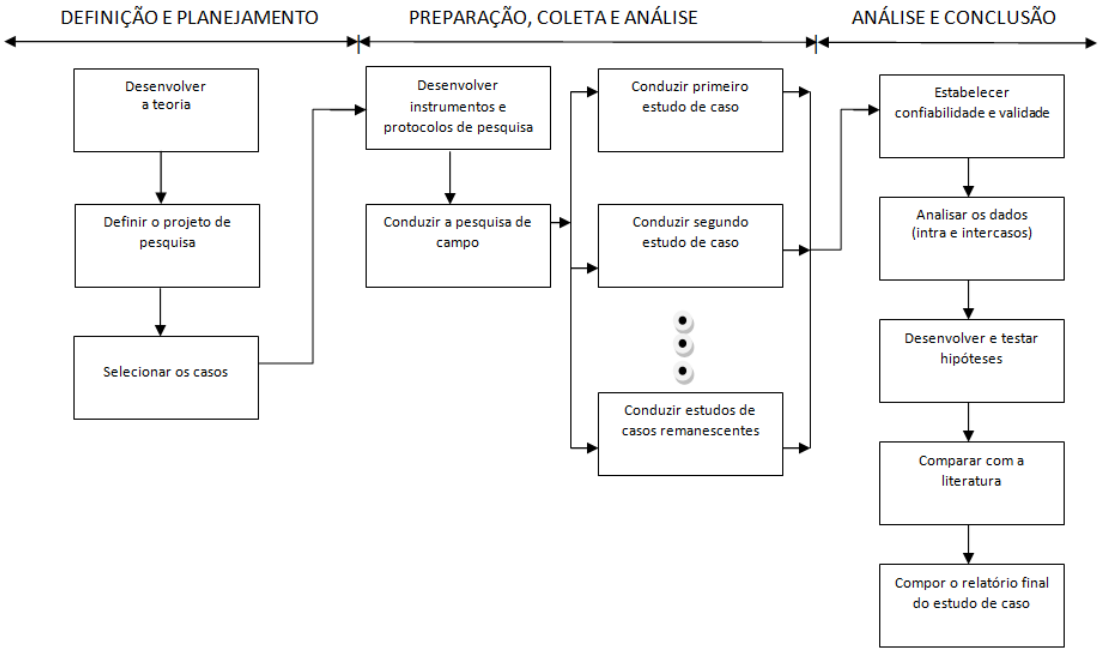


Figura 1 - Atividades do Método de Estudo de Caso

Fonte: Adaptada de Yin (2001)

Voss, Tsikriktsis e Frohlich (2002) consideram que o ponto de partida para o estudo de caso é a estrutura da pesquisa, a qual Yin (2001) denomina de projeto de pesquisa.

Segundo Yin (2001), o projeto de pesquisa é a sequência lógica que conecta os dados empíricos às questões de pesquisa iniciais do estudo e, em última análise, às suas conclusões. O projeto de pesquisa trata de, pelo menos, quatro problemas: quais questões estudar, quais dados são relevantes, quais dados coletar e como analisar os resultados. Seu propósito principal é ajudar a evitar a situação em que as evidências obtidas não remetam às questões iniciais da pesquisa.

Para Yin (2001) os projetos de pesquisa para o estudo de caso apresentam cinco componentes principais: as questões de estudo (ou da pesquisa), suas

proposições (se houver), suas unidades de análise, a lógica que une os dados às proposições e os critérios para se interpretar as descobertas.

Segundo Voss, Tsikriktsis e Frohlich (2002), o estudo de caso é usado para testar as hipóteses e para o desenvolvimento de teorias. Na maioria das pesquisas com estudos de caso existem algumas hipóteses iniciais que podem ser diretamente testadas usando-se os dados do caso. Entretanto, em outros estudos de caso o foco pode ser também o desenvolvimento de teorias e o desenvolvimento ou ajuste de novas hipóteses a partir dos dados coletados, assim como testar as hipóteses iniciais.

A pesquisa utilizando-se estudo de caso pode incluir tanto estudos de caso único quanto de casos múltiplos, e a decisão sobre qual método utilizar deve ser tomada antes da coleta de dados. Estes estudos podem ser baseados em qualquer mescla de provas quantitativas e qualitativas.

Segundo Yin (2001), o estudo de caso único é um projeto apropriado nas circunstâncias onde ele representa:

- um caso decisivo ao testar uma teoria bem formulada: pode ser utilizado para determinar se as proposições de uma teoria são corretas ou se algum outro conjunto alternativo de explanações pode ser mais relevante;
- um caso raro ou extremo;
- um caso revelador: quando o pesquisador tem a oportunidade de observar e analisar um fenômeno previamente inacessível à investigação científica.

Mesmo com a vantagem de observações mais profundas em uma pesquisa utilizando estudo de caso único, afirma-se que o mesmo tem suas limitações como cita Voss, Tsikriktsis e Frohlich (2002). A principal é o limite para a generalização das conclusões, modelos ou teorias desenvolvidos a partir do mesmo. Isso inclui o risco do mau julgamento de um único evento e na facilidade de se exagerar com os dados disponíveis. Os casos múltiplos podem reduzir a profundidade do estudo quando os recursos são restritos, mas podem aumentar a validade externa e auxiliar a evitar a tendenciosidade dos observadores. Yin (2001) acrescenta que o

caso único pode, mais tarde, acabar se revelando como não sendo o caso que se pensava que fosse no princípio.

Nossa pesquisa utilizará o estudo de casos múltiplos exploratórios, os quais serão apresentados na sessão seguinte.

2.3 Estudo de Casos Múltiplos

Cada caso deve servir a um propósito específico dentro do escopo global da investigação. Devem-se considerar os casos múltiplos como se considerariam experimentos múltiplos, ou seja, seguir a lógica da replicação. Segundo Yin (2001), cada caso deve ser cuidadosamente selecionado de forma a:

- prever resultados semelhantes (uma replicação literal);
- produzir resultados contrastantes apenas por razões previsíveis (uma replicação teórica).

Para Eisenhardt (1989), dada a limitação do número de casos que podem ser estudados, é mais sensato selecionar casos que apresentem situações extremas ou do tipo polar, no qual o processo de interesse é transparentemente observável.

Três tipos de situações para se selecionar os casos são descritas por Voss, Tsikriktsis e Frohlich (2002):

- quando podem ser encontrar casos típicos ou representativos;
- casos que neguem ou anulem uma proposição;
- ou casos do tipo polar, que apresentem características nitidamente contrastantes que irão destacar as diferenças que estão sendo estudadas.

Uma questão importante em um projeto de casos múltiplos é a respeito do número de casos supostamente necessários ou suficientes para o estudo. Além disso, não se deve empregar a lógica da amostragem, mas sim pensar nessa decisão como um reflexo do número de replicações de caso, literais e teóricas, que o pesquisador gostaria de ter no seu estudo, como afirma Yin (2001).

Segundo o mesmo, para o número de replicações literais, a seleção do número de replicações depende da certeza que você quer ter sobre os resultados obtidos dos casos múltiplos. Por exemplo, podem se estabelecer duas ou três replicações literais quando as teorias concorrentes forem completamente diferentes e o tema ao alcance exigir um grau excessivo de certeza. Entretanto, se as teorias concorrentes possuírem diferenças sutis ou se é desejável obter um alto grau de certeza, poder-se-iam solicitar cinco, seis ou até mais replicações. Para o número de replicações teóricas, quando não se tem certeza de que as condições externas produzirão resultados diferentes de estudo de caso, podem-se articular essas condições relevantes de uma forma mais explícita no princípio do estudo e identificar um número maior de casos que devem ser incluídos nele. Em contraste, quando não se acredita que as condições externas produzam muita variação no fenômeno que está sendo estudado, é necessário um número menor de replicações teóricas.

Para Voss, Tsikriktsis e Frohlich (2002), os casos múltiplos podem reduzir a profundidade do estudo quando os recursos são escassos, mas podem aumentar a validade externa e auxiliar a imparcialidade dos observadores.

Essa afirmativa vai de encontro ao proposto por Yin (2001) que afirma que a utilização dos estudos de casos múltiplos possibilita um maior grau de generalização dos resultados, mas consumem-se mais recursos, o que poderia prejudicar a profundidade do estudo do caso. Além disso, os resultados de múltiplos casos são considerados mais convincentes e o estudo é visto como sendo mais robusto. Para Miguel (2007), a partir da seleção dos casos devem-se determinar os procedimentos e técnicas para coleta e análise de dados.

O presente trabalho será do tipo polar, pois situações contrastantes serão apresentadas, sendo a análise dos dados feita de forma intracasos e intercasos.

Segundo Eisenhardt (1989), a análise intracaso é conduzida devido a uma das realidades da pesquisa por estudos de caso: o estonteante volume de dados.

A ideia geral é tornar-se intimamente familiar com cada caso como uma entidade única. Este processo permite que padrões únicos de cada caso surjam antes que os investigadores busquem generalizar esses padrões na análise cruzada dos casos. Voss, Tsikriktsis e Frohlich (2002) sugerem que o ponto de partida para a análise intracaso seja a construção de uma ordenação dos dados e com casos

longitudinais construir uma análise da sequência de eventos. A partir disso, o pesquisador poderia começar a procurar por explicações e causalidades.

A busca sistemática pelos padrões na análise intercasos é uma etapa chave na pesquisa por estudos de casos. Também é essencial para aumentar o poder de generalização das conclusões extraídas dos casos, segundo Voss, Tsikriktsis e Frohlich (2002). Eisenhardt (1989) considera que a chave para uma boa comparação intercasos é ver os dados de divergentes formas.

Ainda de acordo com a mesma autora, a ideia geral por trás das táticas de análises intercasos é forçar os pesquisadores a ir além das impressões iniciais, especialmente pelo uso de métodos estruturados para uso dos dados.

3
Métodos Ágeis

Este capítulo apresenta uma descrição sobre Métodos Ágeis, sua criação, princípios e valores. A sessão 3.2 fala sobre Modelo Teórico versus Empíricos.

3.1 Considerações Iniciais

Em fevereiro de 2001 nos Estados Unidos, um grupo de 17 pessoas se reuniu para discutir, com base em suas experiências, as melhores práticas que utilizavam para o desenvolvimento de *software* (HIGHSMITH, 2001).

O objetivo deste encontro não foi o de somente criar um marco na história do desenvolvimento de *software*, mas reunir um grande número de mestres com intuito de oferecer a todas as pessoas uma alternativa às metodologias pesadas e altamente dirigidas por documentação. (HIGHSMITH, 2001).

Os criadores foram: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn,Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland e Dave Thomas.(MANIFESTO ÁGIL, 2001).

De acordo com o Manifesto Ágil (2001), os 12 princípios dos métodos ágeis são:

- Satisfazer o cliente, com a entrega adiantada e contínua de software de valor.
- Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
- Entregar *software* funcionando com frequência, na escala de semanas até meses, com preferência por períodos mais curtos.

- Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
- Construir projetos ao redor de indivíduos motivados, dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
- 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.
- *Software* funcional é a medida primária de progresso.
- Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter, indefinidamente, passos constantes.
- Contínua atenção à excelência técnica e bom *design*, aumenta a agilidade.
- Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
- As melhores arquiteturas, requisitos e *designs* emergem de times auto-organizáveis.
- Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

No Quadro 1, abaixo, são apresentados os valores dos Métodos Ágeis. Os valores que se encontram do lado esquerdo da tabela são mais importantes do que os do lado direito.

Quadro 1 - Valores dos Métodos Ágeis

Indivíduos e interações entre eles	Processos e ferramentas
Software em funcionamento	Documentação abrangente
Colaboração com o cliente	Negociação de contratos
Responder a mudanças	Seguir o planejamento

Fonte: Adaptado do Manifesto Ágil

No Quadro 2, abaixo, são listados os principais Métodos Ágeis e algumas referências importantes sobre esses métodos:

Quadro 2 - Principais Métodos Ágeis e Referências

Métodos Ágeis	Referências
Métodos Ágeis no Geral	Ambler, 2002; Mundim *et al.*, 2002; Manifesto Ágil, 2001;
XP	Beck e Fowler, 2000; Jefrries, Anderson e Hendrickson,2000;
Scrum	Paasivaara, Durasiewicz e Lassenius, 2008; Schwaber e Beedle,2002;
Lean	Poppendieck,2002; Levine,2009;
Crystal	Cockburn,2004;
DSDM	Stapleton e Constable,1997; Tudor e Tudor,2010;
TDD	Beck,2002; Erdogmus, Morizo e Torchiano, 2005;

Fonte: Elaborado pelos autores

O XP (*eXtreme Programming)* é um bom exemplo de metodologia ágil e algumas de suas características são encontradas também no *Scrum*, as quais aumentam a qualidade de software gerado. Suas principais práticas e valores são: simplicidades, feedback, comunicação, respeito e coragem.(JEFFRIES, 2001)

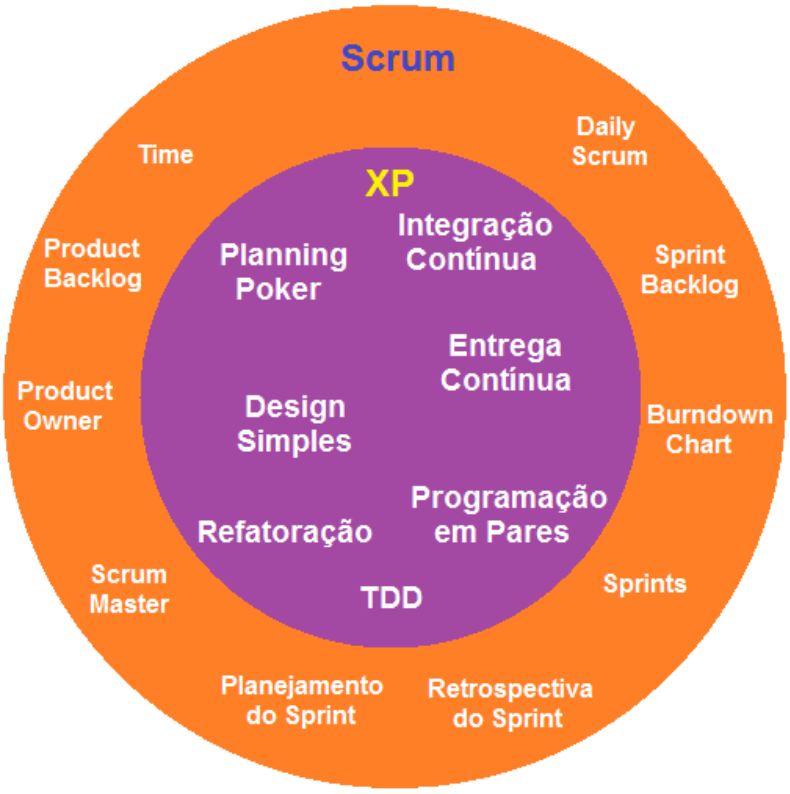


Figura 2 - Métodos Ágeis - XP e *Scrum*

Fonte: Adaptada de Crisp, 2014

Já o método *Lean* tem origem no Sistema de Produção Toyota pois utiliza de sua filosofia e ferramentas, e tem como finalidade conseguir um modelo produtivo ágil, flexível, eficiente e capaz de produzir ao ritmo de demanda, sem gerar custos adicionais e ineficientes. Existem quatro princípios básicos do pensamento *Lean* que são mais relevantes para o desenvolvimento de *software*: acrescentar apenas o que é de valor (eliminar resíduos), centralizar nas pessoas que agregam valor, estabelecer fluxo contínuo e otimizar todas as organizações. (POPPENDIECK,2002).

Crystal foi descrita por Cockburn (2004) e é considerada um exemplo de método ágil e leve. Suas metodologias focam na eficiência e habilidade nos componentes de segurança do projeto. E como no *Scrum*, se concentra na pessoas e não nos processos ou artefatos. Segue as seguintes propriedades: entrega frequente de código utilizável para usuários, melhoria reflexiva, osmótica de comunicação, segurança pessoal, foco, acesso fácil para usuários experientes, ambiente técnico com testes automatizados, gerenciamento de configuração e frequente integração.

O Método de Desenvolvimento de Sistemas Dinâmicos (DSDM) é um processo de senso comum focado em fornecer soluções de negócio de forma rápida e eficiente semelhante ao *Scrum* e XP, porém tem seus melhores usos quando o requisito de tempo é fixo. O método é descrito em nove princípios, sendo os quatro primeiros responsáveis por definir as bases sobre as quais o DSDM é construído: o envolvimento do usuário é imperativo, a equipe deve ser habilitada para tomar decisões, foco em frequente entrega de produtos, aptidão para fins comerciais é o critério essencial para a aceitação das entregas; os outros cinco fornecem os princípios que têm norteado a sua estrutura. Todos esses princípios encontrados se fazem necessários se a qualidade do sistema fornecido deve se manter em uma escala de tempo fornecida pelo negócio. (STAPLETON e CONSTABLE,1997)

O Desenvolvimento Dirigido por Teste (TDD) está relacionado diretamente aos conceitos expostos no método XP. Foi desenvolvido por Beck (2002) e determina que a escrita de novo código deve ser feita se primeiro ocorreu uma falha em teste automatizado ou para eliminar trechos códigos duplicados. O TDD pode ser utilizado por qualquer método ágil para agilizar a codificação e garantir a consistência do produto.

3.2 Modelo Teórico *versus* Modelo Empírico

Modelos de Processos Teóricos são derivados dos primeiros princípios, utilizando materiais e energia balanceada e leis fundamentais para determinar o modelo. Para um sistema de desenvolvimento ser classificado como Teórico ele deve estar de acordo com esta definição.

Modelos de Processos Empíricos são categorizados observando suas entradas e saídas, e definindo controles que fazem com que ocorram dentro dos limites prescritos. Não há necessidade de nenhum conhecimento a priori sobre o sistema, sendo o mesmo tratado como uma "caixa preta". (SCHWABER, 1995)

O *Scrum* é um método empírico e para maior esclarecimento serão apresentadas, no Quadro 3 abaixo, as principais diferenças entre os métodos supra citados:

Quadro 3: Modelagem Teórica *versus* Empírica

Modelo Teórico	Modelo Empírico
Geralmente envolve menos medidas, exige experimentação apenas para estimativas dos parâmetros do modelo desconhecido.	Exige medidas amplas, porque ele depende inteiramente de experimentação para o desenvolvimento do modelo.
Fornece informações sobre os estados internos do processo.	Apenas fornece informações sobre que parte do processo pode ser influenciada pela ação de controle.
Promove a compreensão fundamental do funcionamento interno do processo.	Trata o processo como uma "caixa preta".
Requer conhecimento preciso e completo sobre o processo.	Não requer nenhum conhecimento detalhado, apenas que os dados de saída devem ser obtidos em resposta às mudanças de entrada.
Não é completamente útil para processos mal definidos e/ou complexos.	Muitas vezes prova ser a melhor alternativa para modelagem de processos mal definidos e/ou complexos.
Produz naturalmente processos lineares e não lineares.	Requer métodos especiais para produzir modelos não lineares.

Fonte: Adaptado de Schwaber (1995)

4
Scrum

Este capítulo apresenta a definição de Scrum. A sessão 4.2 apresentará os papéis, a sessão 4.3 o planejamento, a sessão 4.4 trará as estimativas e por fim, a sessão 4.5 trará alguns problemas.

4.1 Considerações Iniciais

O nome "*Scrum*" tem origem no esporte coletivo inglês *Rugby*, e foi primeiramente referenciado na literatura em um artigo de Takeuchi e Nonaka no ano de 1986, em uma adaptação, rápida e auto organizável do processo de desenvolvimento de produto, originário no Japão (SCHWABER e BEEDLE, 2002). Por meio de uma pesquisa realizada na base de dados do site *Web of Knowledge*, buscando todos os artigos já publicados até o ano de 2014 cujo título contenha a palavra "*Scrum*" e cuja área seja "Ciência da Computação", fica evidente que essa metodologia ainda é relativamente nova e conforme mostra a Figura 3, poucas publicações sobre este tema foram feitas até o ano de 2014.

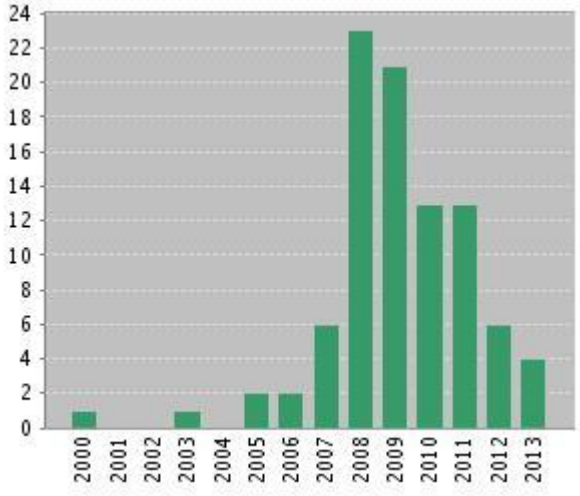


Figura 3 - Publicações sobre *Scrum* por ano

Fonte: *Web of Knowledge*, 2014

Como visto na figura acima, não são frequentes novas publicações sobre o tema, mas a quantidade de pesquisas relacionadas ao tema começou a crescer após 2005, o que fica evidente analisando a Figura 4, que mostra a quantidade de citações sobre Scrum até o ano de 2014.

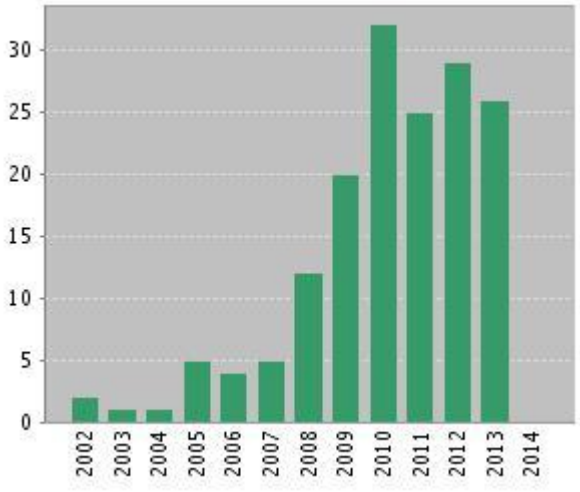


Figura 4 - Citações sobre *Scrum* por ano

Fonte: *Web of Knowledge*, 2014

O *Scrum* é um processo de desenvolvimento de software incremental em ambientes complexos, que propõe controles empíricos que permitem que o desenvolvimento ocorra o mais próximo possível do caos que a empresa pode aceitar (RISING e JANOFF, 2000).

Este método foi concebido em 1993 por Jeff Sutherland, Mike Beedle e Ken Schwaber e ganhou força com a criação do Manifesto Ágil no ano de 2001, pois segue os valores e princípios do mesmo.

Conforme Schwaber e Beedle (2002) temos como são baseadas as liberações de software seguindo o *Scrum:*

- Requisitos do consumidor: como o sistema atual precisa ser melhorado.
- Tempo necessário: qual o tempo necessário para ganhar uma vantagem competitiva.
- Competitividade: o que a concorrência faz, e o que é necessário para melhorar isso.
- Qualidade: qual a qualidade requerida, dadas as variáveis acima.
- Visão: quais mudanças são necessárias para concluir a visão do sistema.
- Recursos: quais equipes e fundos estão disponíveis.

Essas são as variáveis iniciais para a melhoria de projeto de software, mas elas irão mudar durante o projeto. Um processo de desenvolvimento bem sucedido deve levar em conta essas variáveis e a evolução natural das mesmas.

4.2 Papéis do *Scrum*

O *Scrum* é baseado em papéis e responsabilidades específicas para alcançar os objetivos de forma eficiente. Abaixo são apresentados os conceitos conforme são apresentados por Kniberg (2007), Schwaber (1995), Schwaber e Beedle (2002) e Takeuchi e Nonaka (1986).

4.2.1 Product Owner

É quem conhece todos os detalhes do produto, estando sempre disponível para tirar dúvidas do time sobre seu funcionamento em cada situação específica ou sobre as necessidades dos s*takeholders*. O *Product Owner* é quem define os requisitos prioritários, de acordo com os critérios informados pelos *stakeholders*, garantindo que o time trabalhe sempre de acordo com as prioridades.

4.2.2 *Scrum Master*

O *Scrum Master* é o responsável por garantir que: a reunião diária comece e termine na hora certa; os itens do *Backlog* sejam manipulados de forma a manter o *Sprint* no prazo e comunicando o *Product Owner* a respeito dessas mudanças; que o *Sprint Backlog* seja sempre atualizado pelo time; e o mais importante, que todos os problemas ou impedimentos sejam resolvidos ou repassados para o *Product Owner*.

No final de cada *Sprint*, o *Scrum Master* deve convocar todo o time de desenvolvimento, o *Product Owner* e os *stakeholders* que tenham interesse em participar da Retrospectiva do *Sprint*, no qual os pontos positivos e negativos são discutidos para alcançar a produtividade máxima nos próximos *Sprints*.

4.2.3 Time de Desenvolvimento

O time de desenvolvimento é uma equipe que trabalha em um ambiente conjunto, o que faz com que a comunicação entre os membros seja constante. Todos os membros cooperam entre si para implementar os requisitos definidos para o *Sprint* corrente e devem notificar ao *Scrum Master* sempre que algum problema ou impedimento ocorrer. Os times são auto gerenciáveis, cada membro assume responsabilidades e se compromete a cumpri-las, e são multifuncionais, pois cada membro pode desempenhar várias funções dentro do time. No *Scrum* a prática de horas extras não é aceitável.

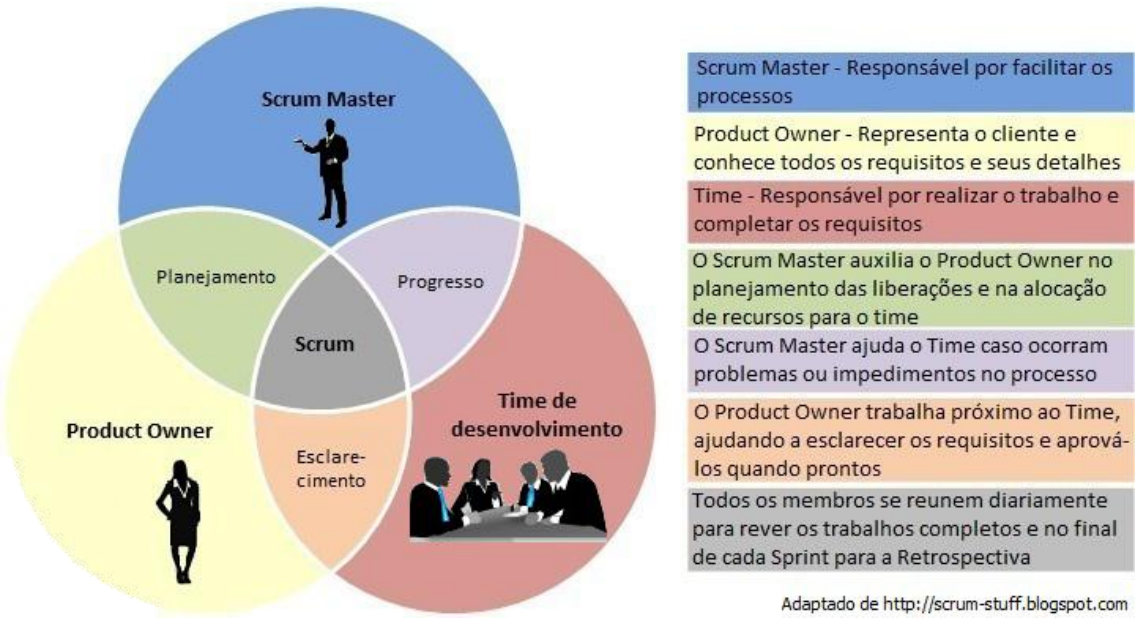


Figura 5 - Papéis do *Scrum*

Fonte: Adaptada de http://scrum-stuff.blogspot.com

4.2.4 *Stakeholders*

Todos os clientes ou representantes do cliente envolvidos no projeto são chamados de *stakeholders* e definem, no começo do projeto, quais os requisitos e quais as características desejadas. Podem também participar da Retrospectiva do *Sprint* para ter conhecimento dos requisitos que foram implementados, mas não opinam nas ações do time e nem interferem no decorrer do *Sprint*. Caso novos requisitos surjam ao longo do projeto, os *stakeholders* entram em contato com o *Product Owner*, que repassa estes ao time no planejamento do próximo *Sprint*.

4.3 Planejamento do *Scrum*

O *Scrum* utiliza conceitos importantes quanto à aplicação dos processos em um período de tempo do projeto. Na prática, estes conceitos são importantes para permitir a constante alteração/adaptação do projeto, garantindo uma excelente visibilidade do progresso dos trabalhos, além de oferecer uma grande previsibilidade das ações futuras. A Figura 6 a seguir, mostra todo o planejamento do *Scrum*.

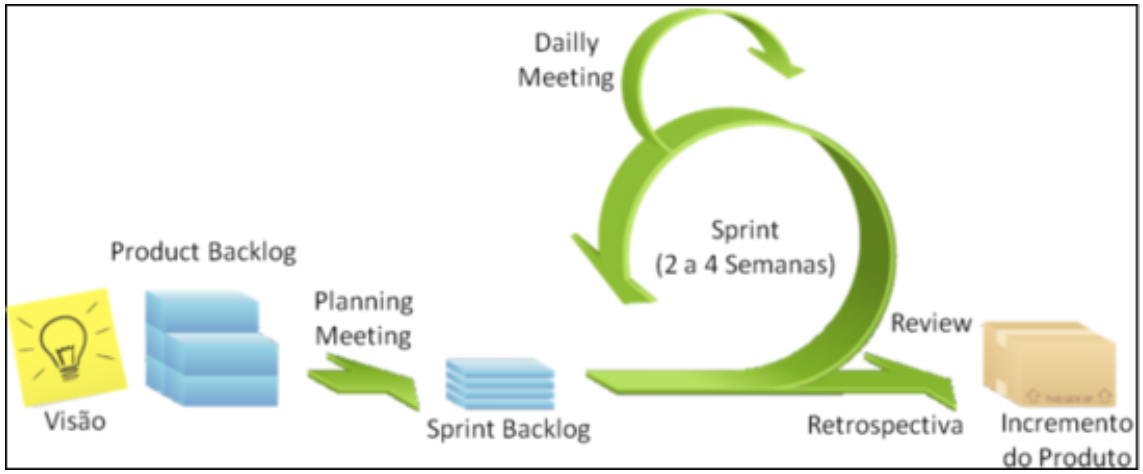


Figura 6 - Planejamento do *Scrum*

Fonte: Adaptada de Schwaber; Beedle (2002)

4.3.1 *Product Backlog*

O *Product Backlog* é um artefato do *Scrum* que se assemelha a um documento ou planilha contendo todas as funcionalidades de alto nível do sistema, ordenadas por prioridades, solicitadas pelos *stakeholders* (DEEMER, 2010).

É importante que o *Product Backlog* esteja sempre organizado e atualizado, pois é com ele que o *Product Owner* decide quais itens agregam mais valor ao produto, garantindo que os itens prioritários sejam alocados nos primeiros *Sprint*s. Os itens presentes no *Product Backlog* podem ser qualquer funcionalidade que seja do interesse dos *stakeholders* ou do próprio *Product Owner*.

Um conceito chave do *Scrum* é a "funcionalidade potencialmente implantável", que significa que quando um item do *Product Backlog* estiver finalizado, ele será completamente funcional e resolverá algum problema de negócio apresentado pelos *stakeholders*.

4.3.2 *Sprint*

No *Scrum* uma iteração é chamada de *Sprint*, normalmente com duração de 30 dias. Dentro deste período, o Time trabalha nos objetivos determinados para o *Sprint* (SCHWABER e BEEDLE, 2002).

Um projeto *Scrum* é composto por vários *Sprints* do mesmo tamanho. Não é aconselhável que o tamanho dos *Sprints* varie, pois um período de tempo fixo é a melhor maneira para observar resultados, principalmente para avaliar a produtividade da equipe (LÉVESQUE e KTATA, 2010).

4.3.3 *Sprint Backlog*

Um conceito importante para entender o *Sprint Backlog* é a definição de "Pronto", que é o consenso que a equipe juntamente com os *stakeholders* chega para definir o que deverá ser entregue no final de um *Sprint*.

O *Sprint Backlog* é um artefato do *Scrum* que contém o item potencialmente implantável que será atacado no *Sprint* corrente dividido em tarefas menores, de no máximo, 16 horas. Geralmente é utilizado um quadro com 4 colunas e tópicos em "*post-it*" para organizar o *Sprint Backlog*. Para um melhor entendimento do mesmo, segue abaixo a Figura 6. A cada *Sprint* é criado um novo *Sprint Backlog*.

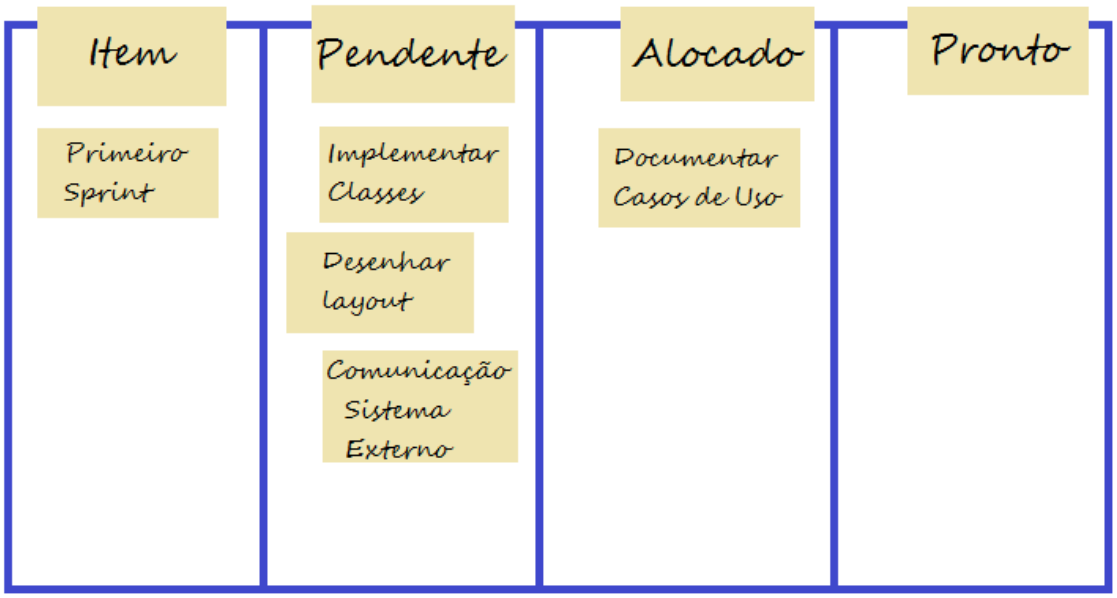


Figura 7 - *Sprint Backlog*

Fonte: Elaborada pelos autores

Para garantir um detalhamento ideal das tarefas, a equipe deve consultar o Scrum Master e o Product Owner. Um detalhamento ideal deixa bem claro tudo que é necessário para tornar um item em algo pronto de acordo com a definição previamente estipulada.

4.3.4 *Daily Meeting*

Durante todo o *Sprint* ocorre a *Daily Meeting* (ou *Daily Scrum*) que é uma reunião feita com o *Scrum Master*, todo o time e qualquer pessoa interessada no projeto. As reuniões diárias não devem ultrapassar o tempo máximo de 15 minutos (SCHWABER, 2004). É importante não extrapolar esse limite para que a reunião foque apenas nos assuntos pertinentes ao dia de trabalho corrente e não dure várias horas, atrapalhando a produtividade do time.

Para que a reunião seja proveitosa, cada membro do Time dá suas impressões a respeito do projeto respondendo a três perguntas fundamentais (SCHWABER, 2004):

1. O que eu fiz desde a última reunião diária?

2. O que eu pretendo fazer até amanhã?

3. Têm alguma coisa impedindo o meu trabalho?

Nesta reunião o Scrum Master fica por dentro do andamento dos trabalhos do time, dos impedimentos que surgem, resolvendo-os o quanto antes, garantindo assim a continuidade do processo.

4.3.5 *Sprint Review*

É o momento em que o Time juntamente com o *Scrum Master* demonstram as funcionalidades potencialmente implantáveis desenvolvidas durante o Sprint para o *Product Owner* e *stakeholders* (SCHWABER, 2004). Esta reunião ocorre no último dia do *Sprint* (LÉVESQUE e KTATA, 2010).

Essa reunião é de extrema importância, pois é o momento do *Product Owner* e *stakeholders* confirmarem se o resultado do *Sprint* atende à definição de "pronto" e se um novo item pode ser atacado, ou se é necessário retrabalhar alguma tarefa do *Sprint* que passou.

4.3.6 *Sprint Retrospective*

É a reunião entre o *Scrum Master* e o time, que ocorre após a reunião de revisão do *Sprint* (*Sprint Review*), na qual são feitas duas perguntas para tentar melhorar o decorrer do *Sprint* futuro (SCHWABER, 2004):

1. O que foi bom durante o *Sprint*?

2. O que pode ser melhorado?

O *Scrum Master* deve avaliar os pontos apresentados e prover recursos necessários para que as propostas de melhorias sejam efetivadas. Esta reunião tem um grande valor para o *Scrum* pois garante uma melhoria contínua do ambiente de trabalho.

4.4 Estimativas do *Scrum*

Projetos *Scrum* podem ser estimados utilizando a função padrão de estimativa de pontos. No entanto, isto é aconselhável para estimar a produtividade em cerca de duas vezes a métrica atual. Esta estimativa é somente para o início do projeto, e as observações feitas levaram à conclusão de que projetos *Scrum* têm tanto velocidade como aceleração.(SCHWABER,1995)

As características da estimativa do *Scrum* são:

- A velocidade inicial e aceleração são baixas conforme o ambiente é construído ou adaptado;
- Conforme as funcionalidades básicas são implementadas, a aceleração aumenta;
- A aceleração vai diminuindo conforme a velocidade vai estabilizando em um ponto alto;

Quando são iniciados os trabalhos, todos os membros do time participam das estimativas e os mesmos têm liberdade para estimar e não sofrem pressões externas para alterá-las.

4.4.1 *Planning Poker*

A ideia por trás do *Planning Poker* é simples, onde histórias individuais são apresentadas para a estimativa. Após um período de discussão, cada membro do time, a partir de seu cartão numerado - numeração esta que segue a sequência de Fibonacci (1,2,3,5,8) - pontua a complexidade de cada item do *Product Backlog*. Cada um mostra sua carta ao mesmo tempo, para que nenhuma opinião seja influenciada pelos outros membros do time. A partir daí, são iniciadas discussões para se chegar a um consenso sobre o melhor valor, se opiniões extremamente diferentes surgirem. Na Figura 8 é apresentada uma rodada do *Planning Poker*.

Os *stakeholders* e *Product Owner* não participam ativamente deste processo, apenas são solicitados quando existe dúvida acerca de algum item do *Product*

Backlog, visando com isso, a não influência sobre os membros do time para que optem por números baixos nas suas estimativas.

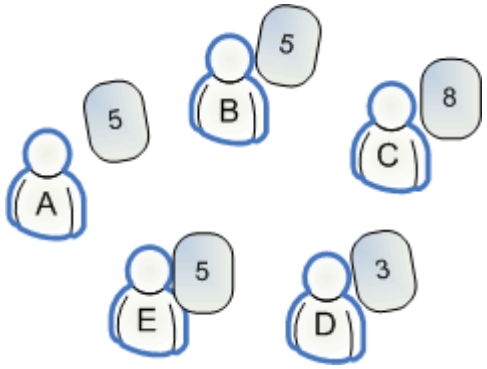


Figura 8 - *Planning Poker*

Fonte: Crisp, 2014

4.5 Problemas na adoção do *Scrum*

O método *Scrum* possui forte aceitação na indústria de software, porém, como qualquer método, sofre críticas e é questionado quanto ao seu domínio de aplicação. Conforme Gregório *et al.* (2007), o principal ponto fraco é a falta de escalabilidade para equipes grandes e geograficamente dispersas, além da necessidade de mudanças na cultura da organização.

A primeira crítica mostrou-se inválida por Paasivaara *et al.* (2008), que mostrou empiricamente que o *Scrum* foi usado com sucesso em vários projetos de grandes dimensões, com times distribuídos em diversas plantas empresarias. Já a segunda crítica é válida, uma vez que uma das maiores barreiras para se implantar uma metodologia diferente da atual é a necessidade de se mudar a cultura na gestão de projetos.

Outras dificuldades que podem ser evidenciadas com a utilização do *Scrum* estão listadas no Quadro 4, abaixo:

Quadro 4 - Problemas do *Scrum* e suas consequências

Problema	Consequência
Product Owner pouco presente	Documento Visão inexistente/mal formulado Plano de Liberações inexistente *Product Backlog* inexistente/desatualizado
Product Backlog não mantido	Faltam estimativas, priorizações e acompanhamento
Reuniões não acontecem	Faltam planejamento, comprometimento do time e itens que não estão prontos podem ser aceitos pela falta de validação
Retrospectiva não realizada	Falta feedback do time para melhorar o ambiente Mesmos erros se repetindo em todos os *Sprint*s Impedimentos não são removidos por falta de comunicação com o *Scrum Master*

Fonte: Elaborado pelos autores

5
Estudo de Casos Múltiplos

Este capítulo apresenta as análises intracasos e intercasos das diferentes iniciativas.

5.1 Considerações Iniciais

Foram analisados resultados obtidos de entrevistados lotados em órgãos de: iniciativa privada denominada de "Empresa A", empresa júnior denominada de "Empresa B" e de iniciativa pública denominada de "Empresa C". Por questões de confidencialidade o nome das empresas não foi divulgado.

As perguntas feitas aos entrevistados foram baseadas no estudo de Carvalho (2009) que apresenta os pontos-chave que foram considerados no momento de validar a implantação correta do *Scrum* na empresa de seu estudo.

No total foram 18 entrevistados, sendo 6 da iniciativa privada, 6 da iniciativa pública e 6 de empresa júnior. Com relação aos papéis no *Scrum*, foram entrevistados *Scrum Master's*, *Product Owner's* e integrantes do time de desenvolvimento.

5.2 Análise Intracasos

5.2.1 Empresa A

Quanto ao *Product Owner*, ele está disponível e entende o produto e as necessidades do cliente, porém não é sempre que sabe quais as funcionalidades prioritárias, talvez falte comunicação entre ele e os *stakeholders* para definição destes requisitos.

Porém ele faz com que o time trabalhe nessas funcionalidades, quando identificadas.

Quanto ao time, ficou evidente que eles trabalham lado a lado e colaboram entre si mutuamente, desempenham várias funções e pedem ajuda ao *Scrum Master* quando algum problema é encontrado. Compartilham a mesma linguagem e se comprometem com as responsabilidades assumidas, porém a primeira dificuldade apresentada pela Empresa A foi quanto ao time fazer hora extra. Já foi explicado que isto não é uma boa prática dentro do *Scrum,* e mesmo sendo feito com o objetivo de manter o prazo, isso pode ser prejudicial.

O *Scrum Master* é o papel melhor definido dentre as respostas obtidas; ele trabalha sempre ao lado do time e está presente quando solicitado, além de resolver os impedimentos encontrados de forma prioritária.

O *Product Backlog* geralmente é dinâmico e está acessível e visível. Na maioria das vezes é atualizado antes de cada *Sprint*, contém somente as funcionalidades e estas geralmente contêm uma definição de si.

Os problemas são quanto ao cálculo das estimativas porque não é sempre que todos os times participam quando estas estão sendo realizadas. Em um caso isolado a velocidade de desenvolvimento não é mensurada e nem utilizada para ações corretivas e melhorias futuras.

Todos os membros do time participam das Reuniões de Planejamento do *Sprint*, e como resultado da reunião, tem-se o plano do *Sprint* e o *Sprint Backlog*. Na maioria das vezes há consenso entre os membros do time, e eles se comprometem com o planejado.

Em relação ao *Sprint,* nota-se que em alguns casos o time não age corretivamente quando está atrasado, logo, detectamos que não é sempre que o time entrega um produto funcionando ao final do mesmo. Como forma corretiva, a duração do *Sprint* acaba não sendo a mesma em todos os *Sprints* e algumas funcionalidades são terminadas em *Sprints* futuros. Estas duas colocações são erradas, pois se o *Sprint* aumenta, a definição do que será construído pode sofrer outras alterações, a complexidade pode aumentar e o risco pode crescer.

Quanto ao *Daily Meeting* (ou *Daily Scrum*), ela acontece sempre no mesmo lugar e horário todos os dias e na maioria das vezes é pontual e sem interrupções,

porém não é sempre que os membros da equipe respondem as três perguntas-chave. Às vezes o *Product Owner* não está presente durante essas reuniões, o que impede que dúvidas sejam tiradas, se elas existirem. O time deveria usar esta reunião para inspecionar o progresso em direção ao objetivo do *Sprint* e para inspecionar se o progresso tende para completar o trabalho do *Sprint Backlog*. O *Daily Scrum* aumenta a probabilidade de o time de desenvolvimento atingir o objetivo do *Sprint. (Scrum Guide*,2013). Na maioria das vezes os membros da equipe buscam e decidem as tarefas e geralmente se cobram em relação à realização das mesmas.

A Reunião Retrospectiva acontece ao final de cada *Sprint,* sendo que na maioria das vezes todos participam e como resultado tem-se sugestões concretas de melhorias que são de fato implementadas.

Como temos um caso em que a estimativa não é calculada, a velocidade também não é mensurada. Com isso, seu resultado não é utilizado como ação corretiva para melhorar futuras estimativas. Porém, na maioria dos casos a velocidade é mensurada e utilizada da melhor forma possível.

Quanto ao *Sprint Backlog,* o time possui um e é visível por toda a equipe. Na maioria das vezes é atualizado diariamente por todos, sendo as funcionalidades e tarefas facilmente distinguíveis encontrando-se de forma clara quais tarefas foram originadas por cada funcionalidade.

Segundo a visão dos entrevistados, optou-se pela utilização do *Scrum* pela troca de informações e experiências entre desenvolvedores e clientes, facilitando o entendimento do que foi proposto e à medida que o processo de desenvolvimento avançava, ter um *feedback* do que já foi feito e do que podia ser melhorado. Além disso, acreditavam que fechar um projeto do início ao fim na sua idealização não era viável, além de que os constantes ajustes impediam que o desenvolvedor ficasse focado na funcionalidade em desenvolvimento.

Quanto às melhorias de desempenho que o *Scrum* trouxe, há um consenso de que as principais são quanto à comunicação, troca de conhecimento e cooperação entre os membros do time. Aumentou-se a quantidade de produto entregue e facilitou-se a gestão dos times.

Quanto às maiores dificuldades, acreditam que estão relacionadas ao *Product Owner*, ao tempo gasto durante as reuniões e principalmente à cultura da empresa e de seus gestores.

Em Levesque e Ktata (2010), temos os conceitos de que o *Sprint* não deve variar, e nos baseamos nisso para comentarmos sobre a irregularidade do tamanho do *Sprint.* Quanto ao calculo de estimativas, temos em Schwaber (1995), que uma estimativa inicial deve ser feita, e com suas observações ficaria claro que projetos *Scrum* possuem velocidade e aceleração. Ainda de acordo com o mesmo autor, temos que a retrospectiva deve ser feita e isso encontramos na empresa A. E por se tratar de iniciativa privada, prezam pelo aumento do retorno de investimentos em projeto de novos produtos, como fica claro em Sulaiman *et. al* (2006) e Sutherland (2005).

De acordo com a análise feita acima, apresentamos de forma simplificada, na Figura 9, os problemas encontrados.

Scrum	Classificação
Product Owner	
Time	
Scrum Master	
Product Backlog	
Estimativas	
Reuniões de Planejamento do *Sprint*	
Sprint	
Daily Scrum	
Reunião Retrospectiva	
Velocidade	
Sprint Backlog	

Figura 9 - Análise Intracaso Empresa A

Fonte: Elaborada pelos autores

5.2.2 Empresa B

Na Empresa B optou-se por utilizar o *Scrum* principalmente como forma de aprendizado de um método ágil, além do intuito de realizar um estudo aplicado de suas regras para um melhor controle e melhor desempenho durante o desenvolvimento do projeto.

O *Product Owner* não está sempre presente, mas geralmente tem conhecimento do domínio do produto e das necessidades do cliente. Além disso, na maioria das vezes, sabe priorizar as funcionalidades e garante que o time trabalhe de forma a cumprir essas prioridades.

O *Scrum Master* está sempre presente e quase sempre trabalha lado a lado com o time. Na maioria das vezes resolve os impedimentos para não prejudicar o desempenho do time.

O Time trabalha lado a lado, ajudando-se mutuamente, garantindo assim um trabalho colaborativo. Quase sempre utilizam a mesma linguagem e desempenham mais de uma função. Na maioria das vezes aceitam as responsabilidades a eles atribuídas e se comprometem com elas. Geralmente sabem onde buscar informações necessárias para implementar funcionalidades e pedem ajuda ao *Scrum Master* quando problemas são detectados. Por fim, todos têm conhecimento sobre os *Sprints* e não têm o costume de fazer horas extras.

Na maioria das vezes todos os membros do time participam do planejamento do *Sprint*, no qual geralmente é atualizado o *Product Backlog*. Durante o planejamento há consenso entre os membros do time, que se comprometem com as decisões tomadas durante o mesmo.

O *Product Backlog* não é totalmente dinâmico e atualizado, além de não estar sempre visível ou acessível para os envolvidos. Desta forma, faltam estimativas para o *Product Owner* e a elaboração do *Sprint Backlog* fica prejudicada.

O *Sprint Backlog* é utilizado e as funcionalidades e tarefas são facilmente distinguíveis, porém não é sempre atualizado e nem sempre está visível para todos os envolvidos, atrapalhando assim as estimativas de velocidade do time. Além disso, a definição de "Pronto" não é compartilhada por todos os envolvidos.

Os maiores problemas são em relação às estimativas, pois os *Sprints* não duram sempre o mesmo período de tempo, as funcionalidades não são entregues no mesmo *Sprint* em que começam e não há entrega de produtos funcionais ao fim de todos os *Sprints*. Tais dificuldades estão relacionadas à falta de acompanhamento da velocidade do time de desenvolvimento, o que também dificulta a tomada de ações corretivas quando os prazos não são cumpridos.

O *Daily Meeting* nem sempre ocorre no mesmo local e horário. Desta forma, nem sempre todos os membros envolvidos estão presentes nas reuniões e o *Product Owner* geralmente não está presente. Além disso, as três perguntas-chaves do evento não são respondidas, prejudicando a tomada de ações corretivas para os erros cometidos anteriormente.

Assim como no *Daily Meeting*, o *Retrospective Backlog* nem sempre acontece e, quando acontece, não conta com a participação de todos os envolvidos. Desta forma os erros cometidos no *Sprint* encerrado não são discutidos a fim de melhorar o desempenho do time.

Os principais pontos positivos apresentados pelos entrevistados são a maior motivação e a capacidade de auto-organização da equipe, tornando-a mais dinâmica, e a maior clareza nas especificações do produto.

As principais dificuldades apresentadas pelos entrevistados são o *Daily Scrum*, *Product Owner* e Estimativas de Velocidade/Planejamento, respectivamente.

Ainda de acordo com os entrevistados, as principais vantagens que o *Scrum* trouxe para o projeto são: maior motivação e dinamicidade da equipe, que se tornou mais independente; maior entrosamento da equipe e melhorias no desempenho com a programação em pares; maior clareza nas especificações do produto.

De acordo com Schwaber (2004), fica claro que o *Dailly Scrum* não deve ultrapassar 15 minutos para que o foco da reunião não se perca, e quanto às três perguntas-chave seguimos Rising e Janoff (2000), os quais apresentam a importância e os ganhos que se tem com esta reunião. Ainda de acordo com estes autores, seguimos considerações sobre o que acontece ao final de cada *Sprint* para analisar a empresa B. Em Kniberg (2007) nos orientamos sobre a implantação no *Scrum* de uma forma geral.

De acordo com a análise feita acima, apresentamos de forma simplificada, na Figura 10, os problemas encontrados.

Scrum	Classificação
Product Owner	
Time	
Scrum Master	
Product Backlog	
Estimativas	
Reuniões de Planejamento do *Sprint*	
Sprint	
Daily Scrum	
Reunião Retrospectiva	
Velocidade	
Sprint Backlog	

Figura 10 - Análise Intracaso Empresa B

Fonte: Elaborada pelos autores

5.2.3 Empresa C

Na empresa C, o *Product Owner* geralmente está disponível e entende o produto e as necessidades do cliente, porém não se faz presente na maioria das *Daily Scrums*, o que é uma falha da sua parte. Na maioria das vezes sabe quais as funcionalidades prioritárias e faz o time trabalhar nestas, porém muitas são as requisições externas, o que atrapalha no desenvolvimento das atividades planejadas para o *Sprint*.

O time trabalha lado a lado, constantemente colaborando entre si e desempenhando várias funções. Pedem ajuda ao *Scrum Master* quando empecilhos são encontrados, admitem problemas e ajudam-se mutuamente. Em geral, aceitam responsabilidades e se comprometem em finalizá-las. Quanto a fazer hora extra, as respostas foram divergentes, e mais uma vez deixamos claro que esta não é uma

boa prática. Os participantes do time compartilham a mesma linguagem e a definição de "Pronto".

O *Scrum Master* é presente e efetivamente trabalha ao lado de seu time, resolvendo os impedimentos.

O *Product Backlog* está acessível e visível, mas nem sempre é dinâmico, sendo atualizado efetivamente antes de cada *Sprint.* Contém somente funcionalidades, e estas quase sempre possuem definições.

Metade dos entrevistados decididamente afirma que o *Product Owner* recebe da equipe estimativas de quantidade de trabalho de cada funcionalidade, porém as outras opiniões foram discordantes. Da mesma forma, quando questionados se todos do time participam das estimativas houve discrepâncias. Mesmo assim, afirmam que têm liberdade de estimar, quando estão presentes nas reuniões.

Nas Reuniões de Planejamento do *Sprint,* todos os membros do time regularmente participam, e ela resulta no plano do *Sprint* e *Sprint Backlog*. Quase sempre há consenso entre os membros do time e frequentemente eles se comprometem com o planejado.

Quanto ao *Sprint*, podemos observar potenciais falhas, tais como: a entrega de um produto funcional ao final do *Sprint* ocorrer ocasionalmente; o time não seguir rigorosamente as prioridades do *Sprint Backlog;* raramente o time agir de forma corretiva quando está atrasado e principalmente, as funcionalidades raramente serem finalizadas no *Sprint* que começam. Às vezes o time não sabe onde estão as informações para implementar as funcionalidades. Quase sempre a duração do *Sprint* é a mesma em todos os *Sprints* e o intervalo entre eles é de um dia.

O *Daily Scrum* acontece no mesmo lugar e horário todos os dias, começa e termina pontualmente e todos respondem as três perguntas-chave. Raramente ocorrem interrupções e quase todos os membros do time estão presentes. Estes buscam e decidem tarefas e frequentemente se cobram quanto às suas realizações. Como dito anteriormente, raramente o *Product Owner* está presente nesta reunião.

A Reunião Retrospectiva acontece ao final do *Sprint* e quase sempre conta com a presença de todos. às vezes resulta em sugestões concretas, as quais esporadicamente são implantadas de fato.

A velocidade é mensurada e quase sempre é utilizada para ações corretivas e para melhorar futuras estimativas. Porém os entrevistados relataram que uma das maiores dificuldades se encontra na sua determinação.

Visíveis para toda a equipe e atualizadas diariamente no *Sprint Backlog*, as funcionalidades e tarefas são facilmente distinguíveis e estão claras quais tarefas foram originadas por quais funcionalidades. No entanto, quando questionados se a estimativa de trabalho para as tarefas é atualizada diariamente, as respostas foram diversas, e raramente se aplica.

Optou-se pelo *Scrum* pela garantia de organização da demanda do cliente, a qual anteriormente era confusa e de comunicação falha; pela melhor distribuição da carga de trabalho, antes concentrada individualmente; e pela natureza constantemente mutável dos requisitos.

Quanto às melhorias no desempenho da equipe, houve um maior envolvimento e colaboração entre todos os membros, e a divisão das tarefas ficou mais clara. Para os entrevistados, o *Product Owner* passou a ser mais organizado, direto e conciso em suas requisições.

Os maiores empecilhos citados são quanto à velocidade e interrupções externas excessivas.

Quanto às interrupções externas, Kniberg (2007) apresenta ações típicas para solucionar este problema, tais como: a equipe deve pedir ao *Product Owner* que as perturbações sejam canalizadas; pedir ao time para reduzir seu fator de foco no próximo *Sprint,* para assim ter um plano mais realista; pedir ao time para que uma pessoa seja escolhida a responsável por resolver todas as interrupções. Seguimos o *Scrum Guide* (2013) no decorrer da análise e também Schwaber (1995). Também através da literatura, temos em Maurer *et. al*.(2007) e em Berczuck(2007) que com a utilização do *Scrum* ocorre a melhoria na comunicação e aumento da colaboração entre os envolvidos nos projetos, assim como aconteceu na empresa C.

De acordo com a análise feita acima, apresentamos de forma simplificada, na Figura 11, os problemas encontrados.

Scrum	Classificação
Product Owner	
Time	
Scrum Master	
Product Backlog	
Estimativas	
Reuniões de Planejamento do *Sprint*	
Sprint	
Daily Scrum	
Reunião Retrospectiva	
Velocidade	
Sprint Backlog	

Figura 11 - Análise Intracaso Empresa C

Fonte: Elaborada pelos autores

5.3 Análise Intercasos

De uma forma geral, temos que o *Product Owner* está quase sempre disponível e na maior parte do tempo entende o produto e as necessidades do cliente. Frequentemente sabe quais as funcionalidades prioritárias e faz o time trabalhar nestas funcionalidades. Porém quando as pessoas foram questionadas sobre onde se encontravam as maiores dificuldades ou empecilhos, o *Product Owner* ficou em primeiro lugar. Talvez a pessoa não tenha um conhecimento significativo sobre as necessidades e prioridades, não compareça às reuniões ou não saiba se comunicar com os membros do time.

Os membros do time se enquadram bastante nas regras do *Scrum*, seguindo da melhor forma conceitos de colaboração, responsabilidade e iniciativa. Eles trabalham lado a lado, colaboram entre si e se ajudam mutuamente, desempenham funções distintas, admitem problemas e pedem ajuda para o *Scrum Master*. Quase sempre compartilham a mesma linguagem e a definição de "Pronto". Porém,

quanto a aceitarem responsabilidades e se comprometerem, isso mostrou-se mais comum na iniciativa privada do que nas outras iniciativas. E em relação a fazer hora extra, muitas foram as respostas distintas, sendo algo comum mas que deveria ser evitado.

O *Scrum Master* está presente, trabalha sempre ao lado do time e na maioria das vezes resolve os impedimentos que as equipes encontram. Em geral, é um papel que o time considera como sendo bem desempenhado, ou seja, que encontram-se bem preparados, comunicativos, entendem das funcionalidades e tentam resolver os problemas. Entretanto, em relação as atribuições do mesmo foi possível perceber que essa não é a realidade, já que problemas no andamento do *Sprint,* nas *Daily Scrum's* e nas Reuniões de Planejamento do *Sprint* foram citados pelos entrevistados.

O *Product Backlog* ocasionalmente é dinâmico, mas encontra-se acessível e visível para todos. É atualizado antes de cada *Sprint*, com ressalvas na Empresa B. Contém somente funcionalidades e regularmente cada funcionalidade tem uma definição de si.

Quanto as estimativas, muitas foram as divergências, sendo que na Empresa A em um caso isolado ela não ocorre. Em todas as iniciativas o time até tem liberdade para estimar, mas não são todos que participam. Com isso, um dos maiores problemas encontrados é que o *Product Owner* geralmente não recebe da equipe estimativas de quantidade de trabalho de cada funcionalidade.

Nas Reuniões de Planejamento do *Sprint* todos participam e dela resultam o plano do *Sprint* e o *Sprint Backlog.* Quase sempre há consenso entre os membros do time, porém quanto ao comprometimento do time com o planejado temos que ele vai decaindo, começando pela Empresa A, Empresa B e Empresa C.

Sobre o *Sprint* temos que na Empresa A o time quase sempre entrega um produto funcionando no final do *Sprint,* o que ocorre só às vezes na Empresa B e C. Isso acontece porque não é sempre que as prioridades do *Sprint Backlog* são seguidas e o time só age efetivamente de forma corretiva nesta situação na Empresa A, sendo uma atitude rara nas outras iniciativas. Consequentemente, algumas funcionalidades não são terminadas no mesmo *Sprint* que começam. O time alerta o *Scrum Master* quando encontra problemas todavia não é sempre que

estes são resolvidos. Frequentemente o time sabe onde estão as informações para implementar as funcionalidades e todos sabem sobre o *Sprint.* Nas Empresas B e C quase sempre a duração do *Sprint* é a mesma e o intervalo entre dois *Sprint's* é de no máximo um dia, na Empresa A as respostas foram diversas.

O *Daily Scrum* de uma forma geral acontece todos os dias no mesmo lugar e horário, e regularmente começam e terminam pontualmente, salvo exceções na Empresa B. Quase sempre todos os membros do time estão presentes, porém não é sempre que as três perguntas-chave são respondidas. Raramente ocorrem interrupções, o que é bom e significa que o foco da reunião não é perdido. O *Product Owner* deveria ser uma presença mais frequente nesta reunião. Na maioria das vezes os membros da equipe buscam e decidem as tarefas e quase sempre se cobram as realizações das mesmas.

A Reunião Retrospectiva de forma geral acontece ao final de cada *Sprint* e quase sempre todos participam. Geralmente resulta em sugestões concretas de melhoria, as quais frequentemente são de fato implementadas.

Quanto a velocidade, ela é mensurada mas quanto a ser utilizada para ações corretivas temos grande divergência nas respostas. Porém, preponderantemente ela é utilizada para melhorar futuras estimativas.

O *Sprint Backlog* existe, é visível porém não necessariamente é atualizado todos os dias. No geral, a estimativa de trabalho para as tarefas às vezes é atualizada diariamente, e quase sempre funcionalidades e tarefas são facilmente distinguíveis ficando claro quais tarefas foram originadas por cada funcionalidade.

De acordo com a literatura, no que diz respeito aos pontos positivos encontrados na análise intercasos, temos um aumento da motivação da equipe de desenvolvimento tal como cita Kniberg & Farhang (2008) e Paasivarra *et. al.*(2008); a Reunião de Planejamento do *Sprint* e a Retrospectiva ocorrem corretamente, como temos em Kniberg (2007) e o *Product Backlog* segue características encontradas em Blom (2010). Temos que alguns aspectos propostos por Schwaber (1995) e Takeuchi e Nonaka (1986) são contrariados.

De acordo com a análise feita acima, apresentamos de forma simplificada, na Figura 12, os problemas encontrados.

Scrum	Classificação
Product Owner	
Time	
Scrum Master	
Product Backlog	
Estimativas	
Reuniões de Planejamento do *Sprint*	
Sprint	
Daily Scrum	
Reunião Retrospectiva	
Velocidade	
Sprint Backlog	

Figura 12 - Análise Intercasos

Fonte: Elaborada pelos autores

6
Conclusões

Este capítulo apresenta as conclusões desta monografia. É organizado da seguinte forma: Na Seção 6.1, são apresentadas as considerações gerais e na seção 6.2 temos as sugestões para trabalhos futuros.

6.1 Considerações Gerais

Este estudo de caso procurou mostrar as maiores dificuldades de implantação do *Scrum* de forma distinta e conjunta. A adequação de uma empresa utilizando metodologias ágeis representa uma mudança geral, principalmente na cultura de todas as pessoas, algo que nem sempre é fácil.

Em todas as iniciativas analisadas, ocorreram problemas com o *Product Owner*, seja por não estar sempre presente ou por não saber priorizar os itens do *Product Backlog*. Provavelmente falte um conhecimento sobre as atribuições relacionadas ao papel, o que prejudica todo o processo, já que o *Product Backlog* é um item fundamental no desenvolvimento do *Scrum*, pois serve como diretriz para as ações que serão realizadas em todos os *Sprints*.

Muitos foram os problemas quanto às atividades que acontecem diariamente, faltando comprometimento da equipe e talvez liderança do *Scrum Master* para manter o foco, não tornar as reuniões desinteressantes e minar interrupções externas.

Quando os envolvidos não entendem como funciona uma equipe de desenvolvimento, não respeitam as reuniões essenciais e não conseguem explicar corretamente as regras de negócio, dificilmente o *Scrum* funcionará.

Cada iniciativa possui problemas bem específicos, mas o cálculo da velocidade por exemplo é algo comum a todas. Mensurar a velocidade, faz com que se tenha uma ideia se será possível ou não terminar todos os itens que foram

determinado para o *Sprint*, e com isso entregar algo potencialmente pronto. Também faz com que a equipe interaja e decida conjuntamente.

Sendo o *Scrum* considerado por muitos um *framework*, ele pode ser adaptado para adequações nas empresas, porém não pode perder características de sua essência.

Com base nos problemas encontrados, sugerimos possíveis soluções:

- Treinamentos específicos para *Product Owner* e para *Scrum Master*, a fim de garantir que os mesmos saibam todas as suas atribuições e possam assim corrigir os pontos falhos e então tornar a experiência com o *Scrum* mais proveitosa;
- Combinar o *Scrum* com outras metodologias ágeis, como por exemplo o XP, que pode facilitar na estimativa de prazos e mensuração de velocidade do time;
- Repensar a forma de utilização do *Product Backlog* e do *Sprint Backlog* de modo que sejam facilmente atualizáveis e que estejam de fato sempre atualizados;
- Mais interação com o cliente, garantindo um maior *feedback* das ações desenvolvidas.

O objetivo geral do trabalho foi atingido, pois através do questionário aplicado e das informações fornecidas pelos entrevistados, foi possível identificar as principais dificuldades em cada iniciativa. Esta constatação responde as perguntas propostas como a caracterização do problema.

6.2 Sugestões para trabalhos futuros

De forma a abranger um maior número de pessoas, uma sugestão seria a abordagem de um levantamento tipo *survey* ou partir para uma pesquisa do tipo pesquisa-ação.

Como as principais dificuldades estão conhecidas, sugerimos aplicar-se as soluções para sanar os problemas, de forma prática e assertiva.

Em momento algum o cliente foi consultado para conhecimento de sua opinião sobre a velocidade de entrega de funcionalidades prontas e quanto a sua satisfação; talvez ele possa participar das entrevistas em momentos futuros.

Acrescentar ao questionário perguntas sobre treinamento das equipes visando inteirar-se sobre seus níveis de conhecimento, habilidades e competências.

7 Referências Bibliográficas

Ambler, S. *Agile Modeling*, Wiley Computer Publishing. New York, 2002.

Associação Brasileira das Empresas de *Software*, 2011. Disponível em: http://www.abes.org.br/templ3.aspx?id=306&sub=650. Acesso em: outubro/2011.

Barton, B. & Campell, E. *Implementing a Professional Services Organization Using Type C Scrum*. System Sciences, p. 275, 2007.

Beck, K. *Test Driven Development: By Example*. Addison-Wesley Professional. 1ª Edição. 18 Novembro 2002.

Beck, K. & Fowler, M. *Planning Extreme Programming*. Addison-Wesley Professional. 1ª Edição. 26 Outubro 2000.

Blom, M. *Is Scrum and XP suitable for CSE Development?*. International Conference on Computational Science, ICCS 2010

Berczuk, S. *Back to basics: The Role of Agile Principles in Success with an Distributed Scrum Team*. Agile Conference, p. 382-388, 2007.

Carvalho, B. V. Aplicação do Método Ágil *Scrum* no Desenvolvimento de Produtos de *Software* em uma Pequena Empresa de Base Tecnológica.Unifei.Itajubá MG. 2009.

Cockburn, A. *Crystal Clear: A Human-Powered Methodology for Small Teams*. Addison-Wesley Professional. 1ª Edição. 29 Outubro 2004.

CRISP, Disponível em: http://www.crisp.se/bocker-och-produkter/planning-poker, Acesso em: janeiro/2014

Deemer, P., Benefield, G., Larman, C. & Vodde B. *The SCRUM Primer*, version 1.2, 2010

Eisenhardt, K. M. *Building theories from case study research*. The Academy of Management Review, Vol.14, No. 4, Oct. 1989, p. 532-550.

Erdogmus, H.; Morisio, M. & Torchiano, M. *On the effectiveness of the test - First approach to programming.* IEEE Transactions on Software Engineering. 25 Abril 2005.

Highsmith, J. *The Agile Manifesto History*. 2001. Disponível em: http://www.agilemanifesto.org/history.html Acesso em: janeiro/2014

Jeffries, R. *What is Extreme Programming?*. 2001. Disponível em http://xprogramming.com/what-is-extreme-programming/ Acesso em: janeiro/2014

Jeffries, R.; Anderson, A. & Hendrickson, C. *Extreme Programming Installed*. Addison-Wesley Professional. 1ª Edição. 26 Outubro 2000.

Kniberg, H. *Scrum and XP from the Trenches – How We Do Scrum*. InfoQ, 2007.

Kniberg, H. & Farhang, R. *Bootstrapping Scrum and XP under Crisis*. Agile Conference, p. 436-444, 2008.

Lévesque, G. & Ktata, O. *Designing and Implementing a Measurement Program for SCRUM Teams: What do agile developers really need and want?*. Montréal, Quebec, Canada, p. 101-107, Maio. 2010.

Levine, M. *A Tale of Two Systems: Lean and Agile Software Development for Business Leaders*. Productivity Press. 1ª Edição. 24 Junho 2009.

Manifesto Ágil, 2001. Disponível em: http://www.manifestoagil.com.br/. Acesso em: dezembro/2013.

Mar, K.; Schwaber, K. *Scrum With XP*. 2001.

Maurer,F., Melnik,G. *Agile Methods: Crossing the Chasm*. 29th International Conference on Software Engineering, 2007.

Miguel, P. A. C. Estudo de caso na engenharia de produção: estruturação e recomendações para sua condução. Revista Produção, v. 17, n.1, p. 216-229, 2007.

Mundim, A. P. F., Rozenfeld, H.; Amaral, D. C., Silva, S. L., Guerreiro, V. & Horta, L. C. Aplicando o cenario de desenvolvimento de produtos em um caso pratico de capacitação profissional. Gestao & Producao. v.9, n.1, p.1-16, 2002.

Paasivaara, M., Durasiewicz, S. & Lassenius, C. *Distributed Agile Development: Using Scrum in a Large Project.* Global Software Engineering, p. 87-95, 2008.

Pessoa, F. Mensagem. Parceria Antônio Maria Pereira. 1934

Poppendieck, M. *Principles of Lean Thinking*. Poppendieck LLC, 2002.

Rising, L. & Janoff, N. S. *The Scrum Software Development Process for Small Teams*. IEEE Software, vol. 17, n. 4. 2000.

Schwaber, K. *SCRUM Development Process*. 1995.

Schwaber, K. Agile Project Management with SCRUM. 2004

Schwaber, K. & Beedle, M. *Agile Software Development with SCRUM*. Prentice Hall, 2002.

Scrum Guide, Disponível em: https://www.scrum.org/Scrum-Guide, Acesso em: janeiro/2014.

Stapleton, J. & Constable, P. *DSDM Dynamic Systems Development Method: The Method in Practice*. Addison-Wesley Professional. 1ª Edição. 18 Junho 1997.

Sulaiman, T.; Barton, B. & Blackburn, T. *AgileEVM - Earned Value Management in Scrum Projects*. Agile Conference, 2006.

Sutherland, J. *Future of Scrum Parallel Pipelining of Sprints in Complex Projects.* Agile Conference, p. 90-99, 2005.

Takeuchi, H. & Nonaka, I. The New New Product Development Game. Harvard Business *Review*, p. 137-146, Jan-Fev. 1986.

Tudor, D. J. & Tudor, I. J. *The DSDM Atern Student Workbook*. Galatea Training Services Ltd. 2ª Edição. 5 Fevereiro 2010.

Voss, C.; Tsikriktsis, N. & Frohlich, M. *Case research in operations management*. International Journal of Operations & Production Management, v.22, n. 2, p. 195-219, 2002.

Yin, R. Estudo de caso. Planejamento e métodos. 2a edição, Bookman, Porto Alegre/RS, 2001.

Yoshima, Rodrigo. Gerenciamento de Projetos com *Scrum*. Aspercom, 2007.

Web of Knowledgment, Disponível em: http://www.webofknowledge.com, Acesso em: janeiro/2014

8 Apêndices

8.1 Apêndice I

Questionário de Trabalho de Conclusão de Curso, o qual possui o seguinte tema: "DIFICULDADES NA IMPLANTAÇÃO DO SCRUM: ESTUDO DE CASOS MÚLTIPLOS".

8.1.1 Product Owner

1. O time tem um e ele está disponível? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

2. Entende o produto e as necessidades do cliente? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

3. Sabe quais as funcionalidades prioritárias? *

- Sempre
- Quase sempre
- Às vezes
- Raramente

- Nunca
- Não se aplica

4. Faz o time trabalhar nas funcionalidades prioritárias? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.2 Time

1. Trabalham lado a lado? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

2. Colaboram entre si? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

3. Desempenham várias funções? *

- Sempre
- Quase sempre

- Às vezes
- Raramente
- Nunca
- Não se aplica

4. Admitem problemas e pedem ajuda ao *Scrum Master*? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

5. Ajudam-se mutuamente? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

6. Aceitam responsabilidades e se comprometem? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

7. Não fazem hora extra sistematicamente? *

- Sempre
- Quase sempre

- Às vezes
- Raramente
- Nunca
- Não se aplica

8. Os participantes compartilham a mesma linguagem? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

9. A definição de "Pronto" é compartilhada por aqueles que realizam o trabalho e por aqueles que aceitam o resultado do trabalho? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.3 Scrum Master

1. O time tem um? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

2. Trabalha sempre ao lado de seu time? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

3. Resolve os impedimentos? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.4 Product Backlog

1. É dinâmico? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

2. Está acessível e visível? *

- Sempre
- Quase sempre
- Às vezes

- Raramente
- Nunca
- Não se aplica

3. É atualizado antes de cada *Sprint*? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

4. Contém somente funcionalidades? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

5. Cada funcionalidade contém uma definição de si? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.5 Estimativas

1. O dono do produto recebe da equipe estimativas de quantidade de trabalho de cada funcionalidade? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

2. O time tem liberdade de estimar? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

3. Todos os times participam das estimativas? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.6 Reuniões de Planejamento do *Sprint*

1. Todos os membros do time participam? *

- Sempre
- Quase sempre
- Às vezes
- Raramente

- Nunca
- Não se aplica

2. Ela resulta em plano do *Sprint* e *Backlog* do *Sprint*? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

3. Há consenso entre os membros do time? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

4. O time se compromete com o planejado? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.7 Sprint

1. O time entrega um produto funcionando no final de cada *Sprint*? *

- Sempre

- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

2. O time segue rigorosamente as prioridades do *Backlog* do *Sprint*? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

3. O time age corretivamente quando está atrasado? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

4. O time alerta o *Scrum Master* quando há problemas? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

5.O time sabe onde estão informações para implementar funcionalidades? *

- Sempre
- Quase sempre

- Às vezes
- Raramente
- Nunca
- Não se aplica

6. Os problemas são resolvidos quando ocorrem? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

7. A duração do *Sprint* é a mesma em todos os *Sprints*? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8. O intervalo entre dois *Sprints* é de no máximo um dia? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

9. Todos os envolvidos sabem sobre o *Sprint*? *

- Sempre
- Quase sempre
- Às vezes

- Raramente
- Nunca
- Não se aplica

10. As funcionalidades são terminadas no mesmo *Sprint* que começam? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.8 Daily Scrum

1. Acontecem no mesmo lugar e horário todos os dias? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

2. Começam e terminam pontualmente? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

3. Todos os membros do time estão presentes? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

4. Todos respondem às três perguntas-chave? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

5. Ocorrem interrupções? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

6. O dono do produto a visita regularmente? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

7. Os membros da equipe buscam e decidem as tarefas? *

- Sempre

- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8. Os membros da equipe se cobram as realizações das tarefas? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.9 Reunião Retrospectiva

1. Ela existe ao final de cada *Sprint*? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

2. Todos participam? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca

- Não se aplica

3. Resulta em sugestões concretas de melhoria? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

4. Algumas dessas sugestões são implementadas de fato? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.10 Velocidade

1. A velocidade de desenvolvimento é mensurada? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

2. A velocidade é utilizada para ações corretivas? *

- Sempre
- Quase sempre

- Às vezes
- Raramente
- Nunca
- Não se aplica

3. Seu registro é utilizado para melhorar futuras estimativas? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.11 Sprint Backlog

1. O time tem um? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

2. É visível por toda a equipe? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

3. É atualizado diariamente (e facilmente) por toda a equipe? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

4. A estimativa de trabalho para as tarefas é atualizada diariamente? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

5. As funcionalidades e tarefas são facilmente distinguíveis? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

6. Estão claras quais tarefas foram originadas por cada funcionalidade? *

- Sempre
- Quase sempre
- Às vezes
- Raramente
- Nunca
- Não se aplica

8.1.12 Considerações Finais

1. Por que optou-se pela utilização do *Scrum*? *

2. Como o *Scrum* melhorou o desempenho da equipe? *

3. Ao seu ver, onde se encontram as maiores dificuldades/empecilhos? *

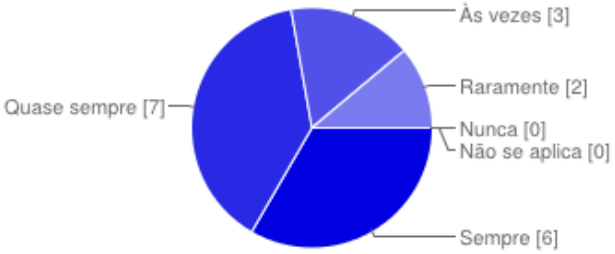
- ☐ Time
- ☐ Scrum Master
- ☐ Product Owner
- ☐ Product Backlog
- ☐ Reuniões de Planejamento do Sprint
- ☐ Sprint
- ☐ Reunião Diária
- ☐ Reunião Retrospectiva
- ☐ Velocidade
- ☐ Outras:

4. Comentários, sugestões e/ou críticas:

8.2 Respostas - Análise Intercasos

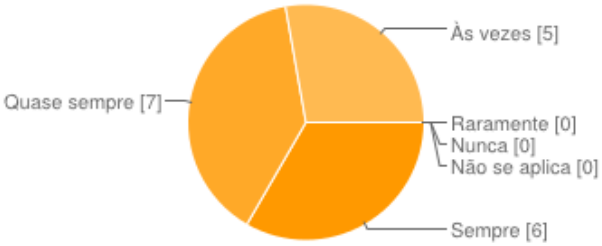
1 - Product Owner

1. O time tem um e ele está disponível?



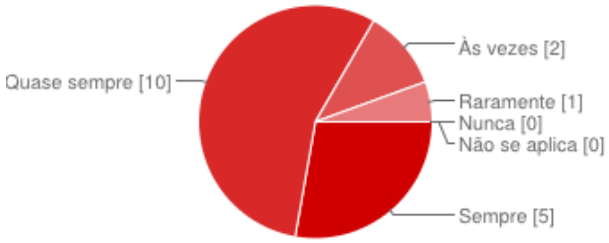
Sempre	**6**	33%
Quase sempre	**7**	39%
Às vezes	**3**	17%
Raramente	**2**	11%
Nunca	**0**	0%
Não se aplica	**0**	0%

2. Entende o produto e as necessidades do cliente?



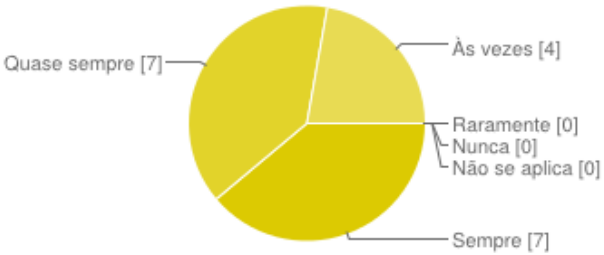
Sempre	**6**	33%
Quase sempre	**7**	39%
Às vezes	**5**	28%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

3. Sabe quais as funcionalidades prioritárias?



Sempre	**5**	28%
Quase sempre	**10**	56%
Às vezes	**2**	11%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**0**	0%

4. Faz o time trabalhar nas funcionalidades prioritárias?

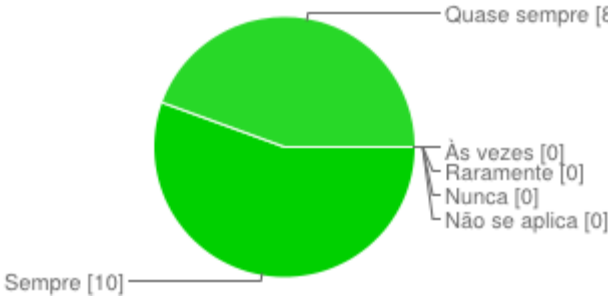


Sempre	**7**	39%
Quase sempre	**7**	39%
Às vezes	**4**	22%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

2 - Time

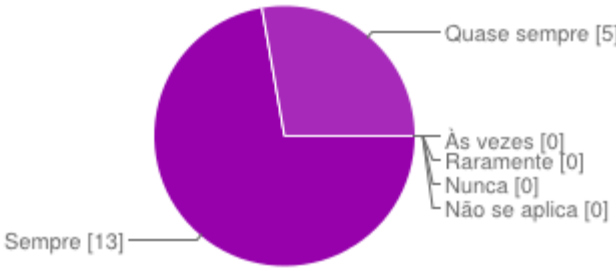
Sempre	**10**	56%
Quase sempre	**8**	44%

1. Trabalham lado a lado?



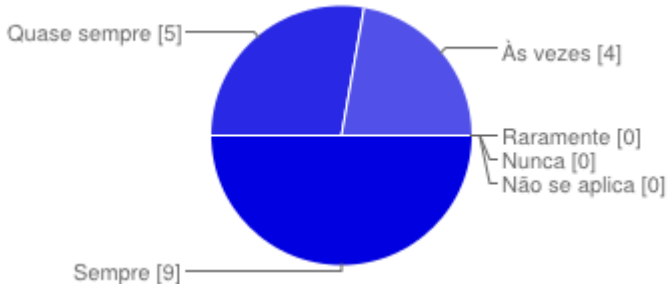
Às vezes	**0**	0%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

2. Colaboram entre si?



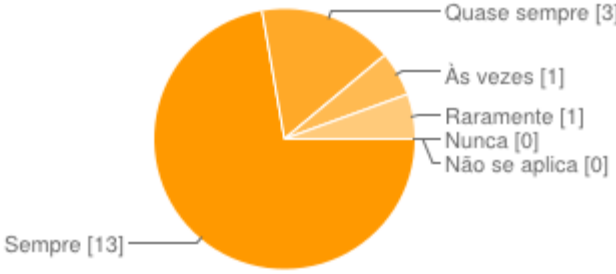
Sempre	**13**	72%
Quase sempre	**5**	28%
Às vezes	**0**	0%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

3. Desempenham várias funções?



Sempre	**9**	50%
Quase sempre	**5**	28%
Às vezes	**4**	22%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

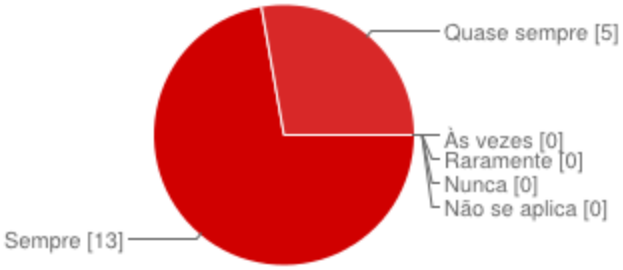
4. Admitem problemas e pedem ajuda ao Scrum Master?



Sempre	**13**	72%
Quase sempre	**3**	17%
Às vezes	**1**	6%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**0**	0%

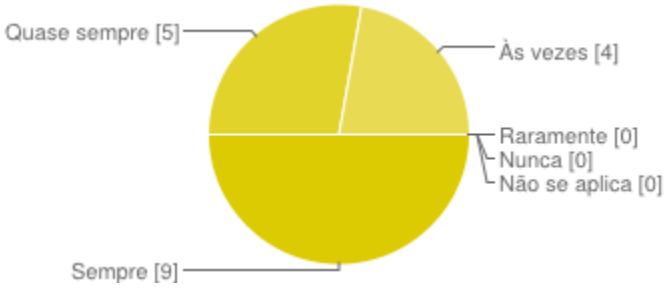
5. Ajudam-se mutuamente?

Sempre	**13**	72%



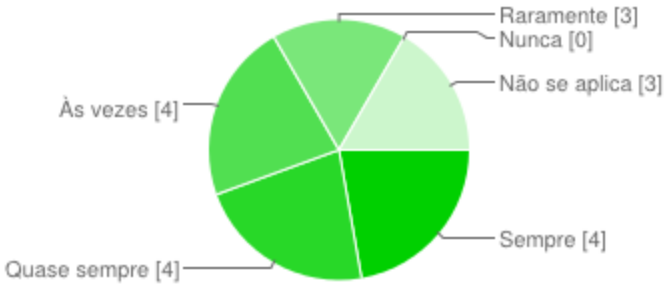
Quase sempre	**5**	28%
Às vezes	**0**	0%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

6. Aceitam responsabilidades e se comprometem?



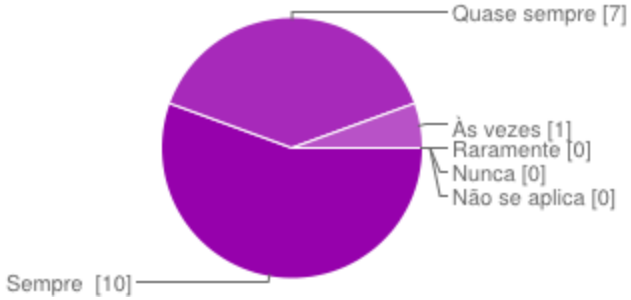
Sempre	**9**	50%
Quase sempre	**5**	28%
Às vezes	**4**	22%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

7. Não fazem hora extra sistematicamente?



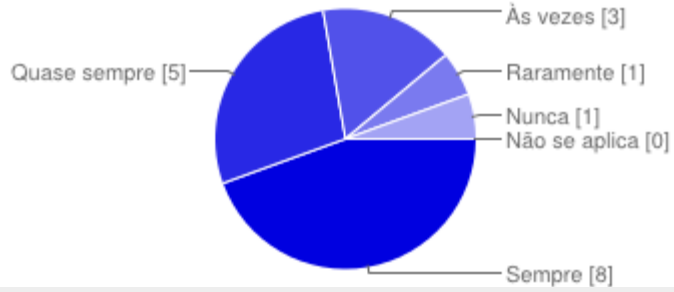
Sempre	**4**	22%
Quase sempre	**4**	22%
Às vezes	**4**	22%
Raramente	**3**	17%
Nunca	**0**	0%
Não se aplica	**3**	17%

8. Os participantes compartilham a mesma linguagem?



Sempre	**10**	56%
Quase sempre	**7**	39%
Às vezes	**1**	6%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

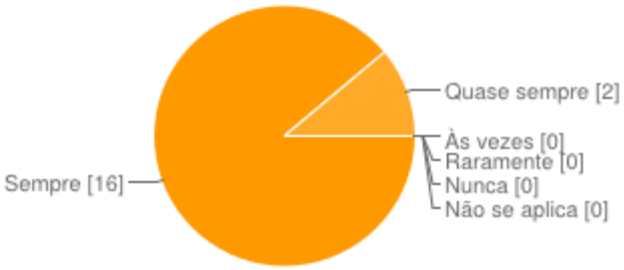
9. A definição de "Pronto" é compartilhada por aqueles que realizam o trabalho e por aqueles que aceitam o resultado do trabalho:



Sempre	**8**	44%
Quase sempre	**5**	28%
Às vezes	**3**	17%
Raramente	**1**	6%
Nunca	**1**	6%
Não se aplica	**0**	0%

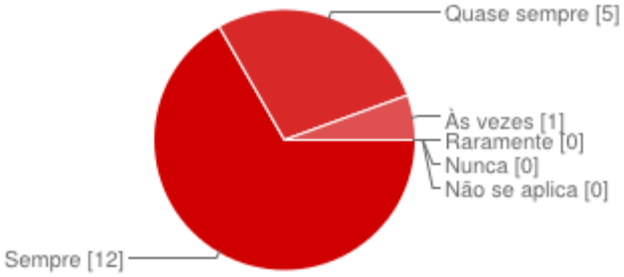
3 - Scrum Master

1. O time tem um?



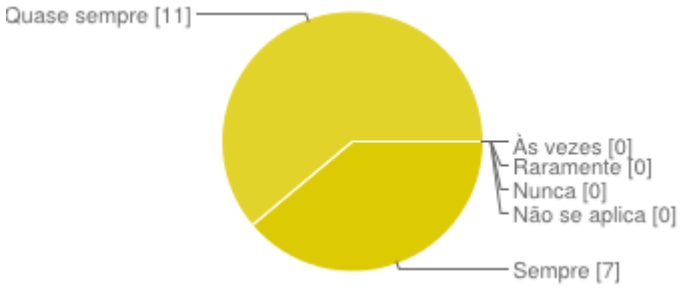
Sempre	**16**	89%
Quase sempre	**2**	11%
Às vezes	**0**	0%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

2. Trabalha sempre ao lado de seu time?



Sempre	**12**	67%
Quase sempre	**5**	28%
Às vezes	**1**	6%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

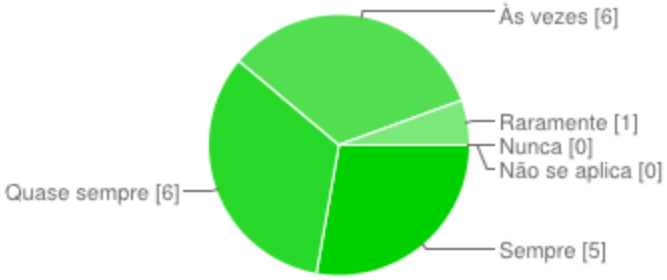
3. Resolve os impedimentos?



Sempre	**7**	39%
Quase sempre	**11**	61%
Às vezes	**0**	0%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

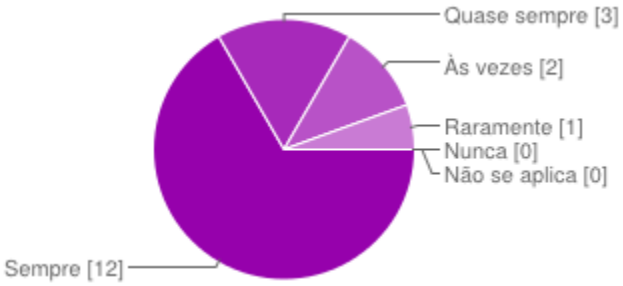
4 - Product Backlog

1. É dinâmico?



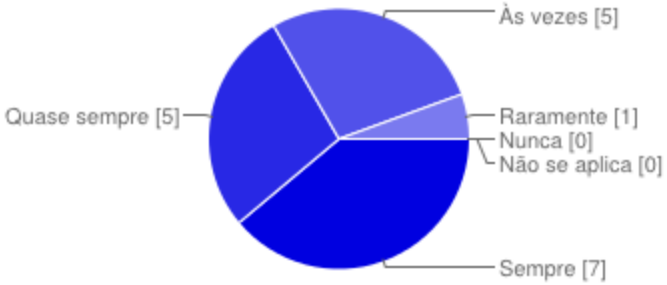
Sempre	**5**	28%
Quase sempre	**6**	33%
Às vezes	**6**	33%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**0**	0%

2. Está acessível e visível?



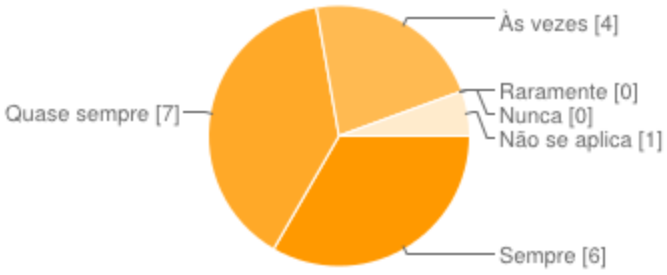
Sempre	**12**	67%
Quase sempre	**3**	17%
Às vezes	**2**	11%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**0**	0%

3. É atualizado antes de cada Sprint?



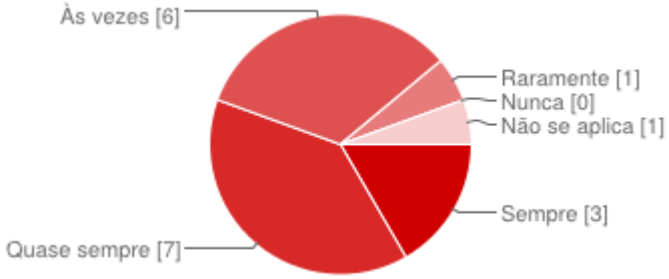
Sempre	**7**	39%
Quase sempre	**5**	28%
Às vezes	**5**	28%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**0**	0%

4. Contém somente funcionalidades?



Sempre	**6**	33%
Quase sempre	**7**	39%
Às vezes	**4**	22%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**1**	6%

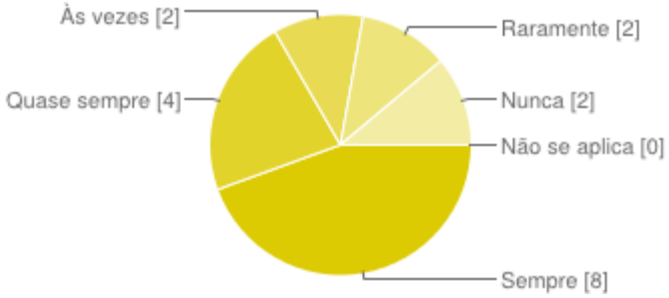
5. Cada funcionalidade contém uma definição de si?



Sempre	**3**	17%
Quase sempre	**7**	39%
Às vezes	**6**	33%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**1**	6%

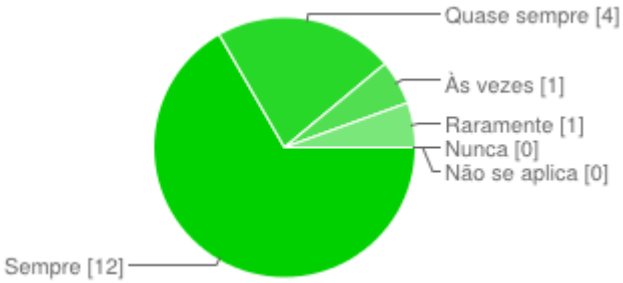
5 - Estimativas

1. O dono do produto recebe da equipe estimativas de quantidade de trabalho de cada funcionalidade?



Sempre	**8**	44%
Quase sempre	**4**	22%
Às vezes	**2**	11%
Raramente	**2**	11%
Nunca	**2**	11%
Não se aplica	**0**	0%

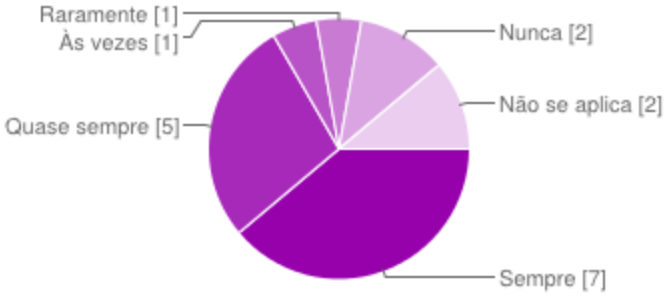
2. O time tem liberdade de estimar?



Sempre	**12**	67%
Quase sempre	**4**	22%
Às vezes	**1**	6%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**0**	0%

Sempre	**7**	39%
Quase sempre	**5**	28%
Às vezes	**1**	6%

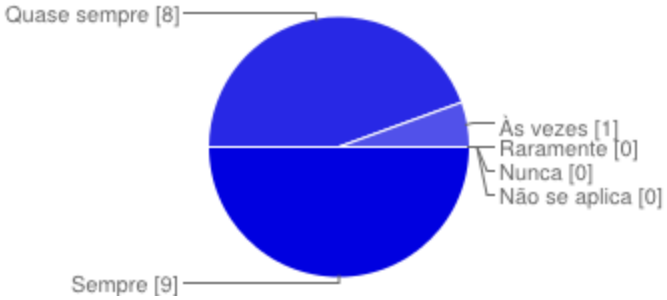
3. Todos os times participam das estimativas?



Raramente	**1**	6%
Nunca	**2**	11%
Não se aplica	**2**	11%

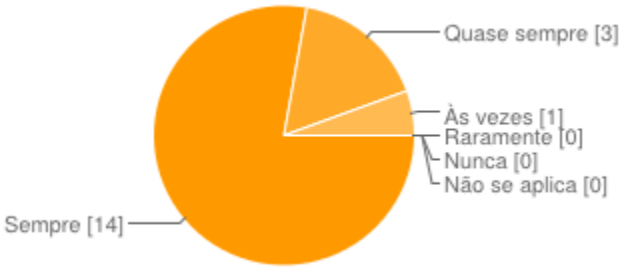
6 - Reuniões de Planejamento do Sprint

1. Todos os membros do time participam?



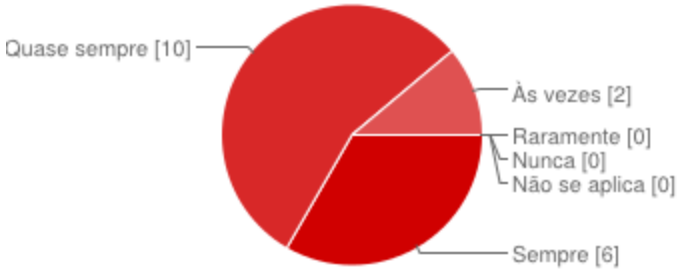
Sempre	**9**	50%
Quase sempre	**8**	44%
Às vezes	**1**	6%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

2. Ela resulta em plano do Sprint e Backlog do Sprint?



Sempre	**14**	78%
Quase sempre	**3**	17%
Às vezes	**1**	6%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

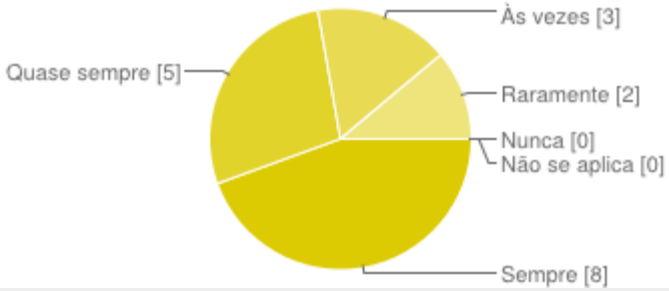
3. Há consenso entre os membros do time?



Sempre	**6**	33%
Quase sempre	**10**	56%
Às vezes	**2**	11%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

Sempre	**8**	44%
Quase sempre	**5**	28%

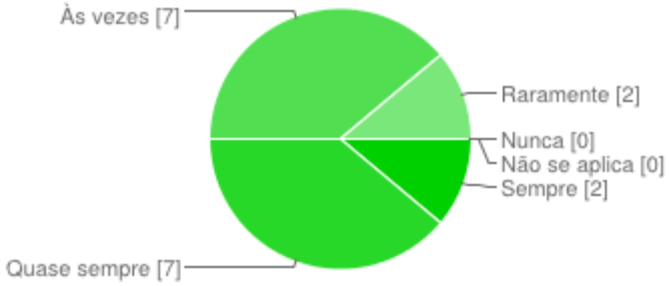
4. O time se compromete com o planejado?



Às vezes	**3**	17%
Raramente	**2**	11%
Nunca	**0**	0%
Não se aplica	**0**	0%

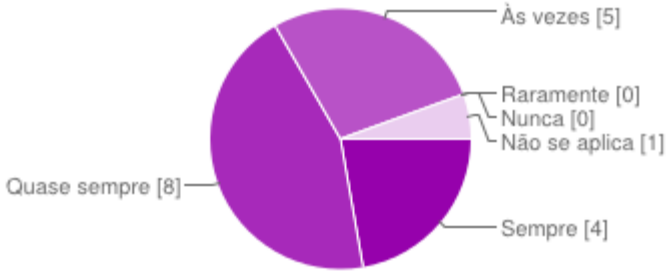
7 - Sprint

1. O time entrega um produto funcionando no final de cada Sprint?



Sempre	**2**	11%
Quase sempre	**7**	39%
Às vezes	**7**	39%
Raramente	**2**	11%
Nunca	**0**	0%
Não se aplica	**0**	0%

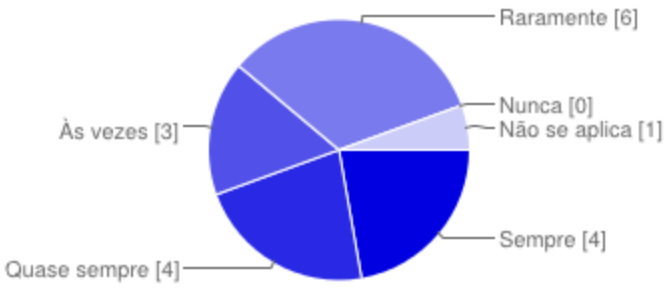
2. O time segue rigorosamente as prioridades do Backlog do Sprint?



Sempre	**4**	22%
Quase sempre	**8**	44%
Às vezes	**5**	28%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**1**	6%

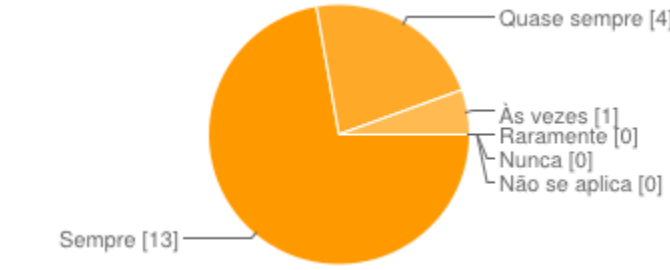
3. O time age corretivamente quando está atrasado?

Sempre	**4**	22%
Quase sempre	**4**	22%
Às vezes	**3**	17%
Raramente	**6**	33%
Nunca	**0**	0%



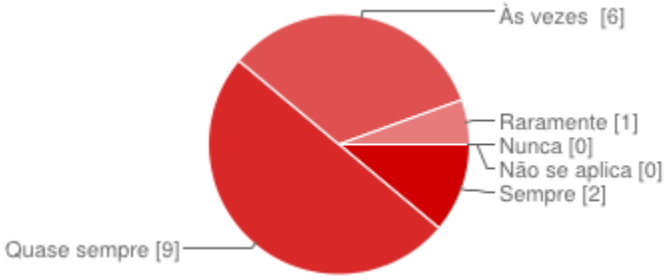
Não se aplica	**1**	6%

4. O time alerta o Scrum Master quando há problemas?



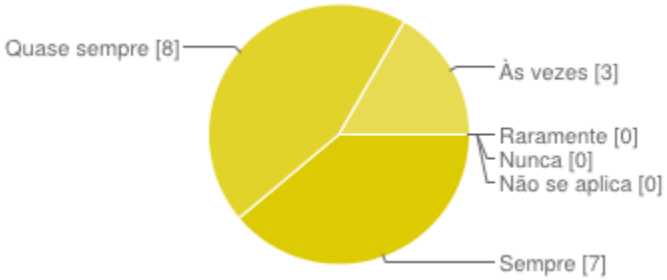
Sempre	**13**	72%
Quase sempre	**4**	22%
Às vezes	**1**	6%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

5. O time sabe onde estão informações para implementar funcionalidades?



Sempre	**2**	11%
Quase sempre	**9**	50%
Às vezes	**6**	33%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**0**	0%

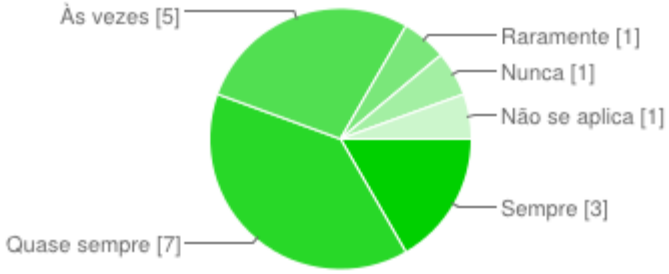
6. Os problemas são resolvidos quando ocorrem?



Sempre	**7**	39%
Quase sempre	**8**	44%
Às vezes	**3**	17%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

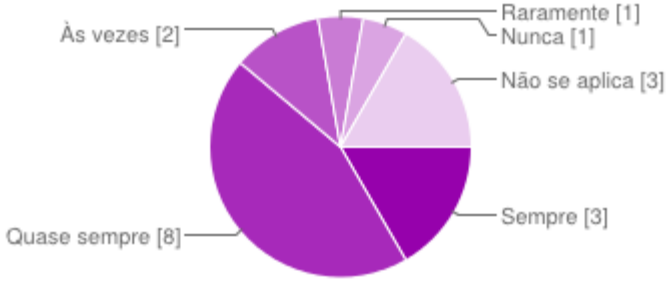
Sempre	**3**	17%

7. A duração do Sprint é o mesmo em todos os Sprints?



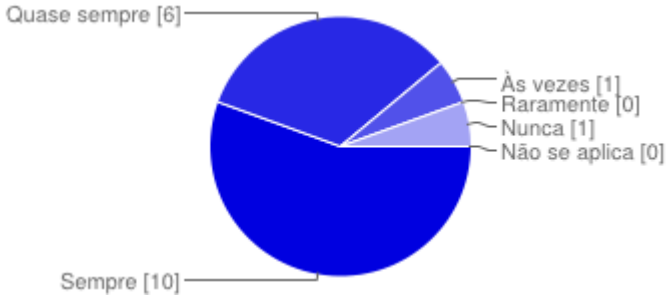
Quase sempre	**7**	39%
Às vezes	**5**	28%
Raramente	**1**	6%
Nunca	**1**	6%
Não se aplica	**1**	6%

8. O intervalo entre dois Sprints é de no máximo um dia?



Sempre	**3**	17%
Quase sempre	**8**	44%
Às vezes	**2**	11%
Raramente	**1**	6%
Nunca	**1**	6%
Não se aplica	**3**	17%

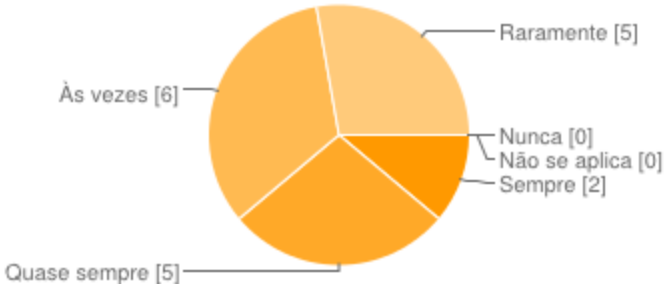
9. Todos os envolvidos sabem sobre o Sprint?



Sempre	**10**	56%
Quase sempre	**6**	33%
Às vezes	**1**	6%
Raramente	**0**	0%
Nunca	**1**	6%
Não se aplica	**0**	0%

10. As funcionalidades são terminadas no mesmo Sprint que começa?

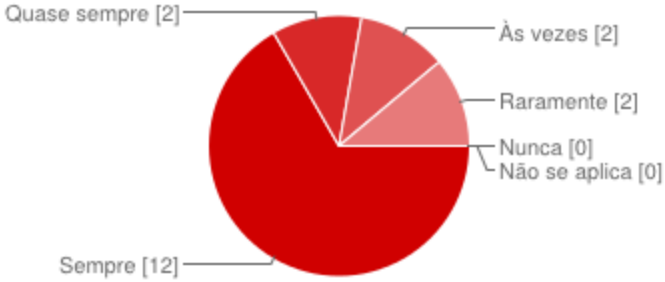
Sempre	**2**	11%
Quase sempre	**5**	28%
Às vezes	**6**	33%
Raramente	**5**	28%



Nunca	**0**	0%
Não se aplica	**0**	0%

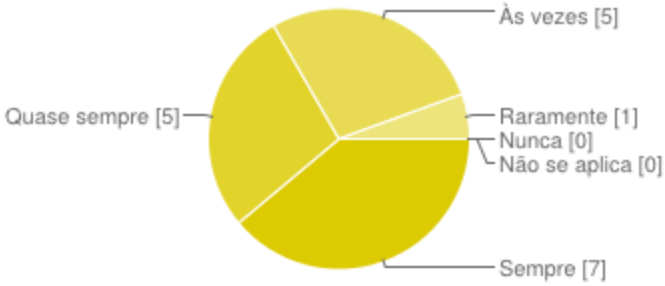
8 - Daily Scrum

1. Acontecem no mesmo lugar e horário todos os dias?



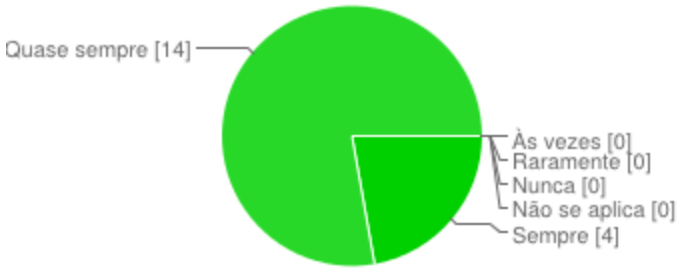
Sempre	**12**	67%
Quase sempre	**2**	11%
Às vezes	**2**	11%
Raramente	**2**	11%
Nunca	**0**	0%
Não se aplica	**0**	0%

2. Começam e terminam pontualmente?



Sempre	**7**	39%
Quase sempre	**5**	28%
Às vezes	**5**	28%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**0**	0%

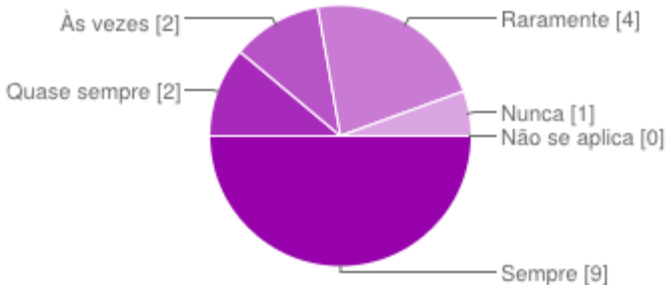
3. Todos os membros do time estão presentes?



Sempre	**4**	22%
Quase sempre	**14**	78%
Às vezes	**0**	0%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

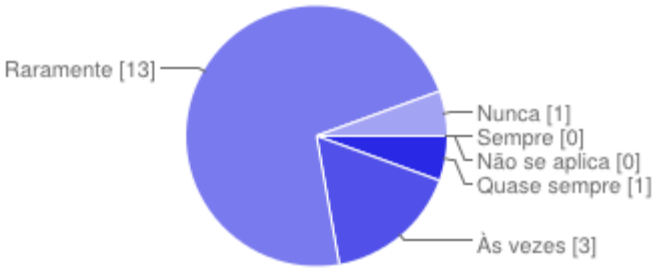
Sempre	**9**	50%

4. Todos respondem às três perguntas-chave?



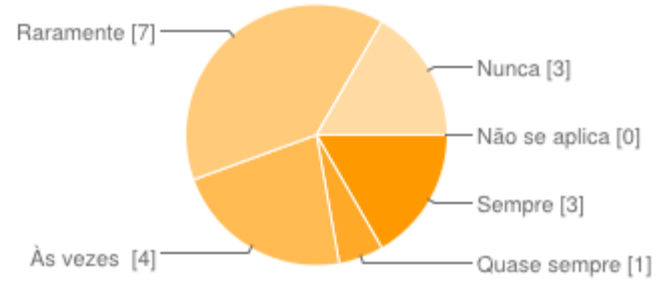
Quase sempre	**2**	11%
Às vezes	**2**	11%
Raramente	**4**	22%
Nunca	**1**	6%
Não se aplica	**0**	0%

5. Ocorrem interrupções?



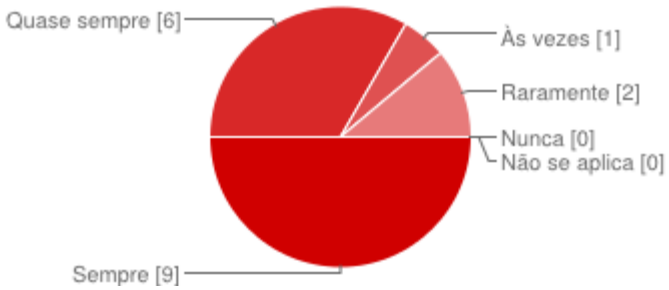
Sempre	**0**	0%
Quase sempre	**1**	6%
Às vezes	**3**	17%
Raramente	**13**	72%
Nunca	**1**	6%
Não se aplica	**0**	0%

6. O dono do produto a visita regularmente?



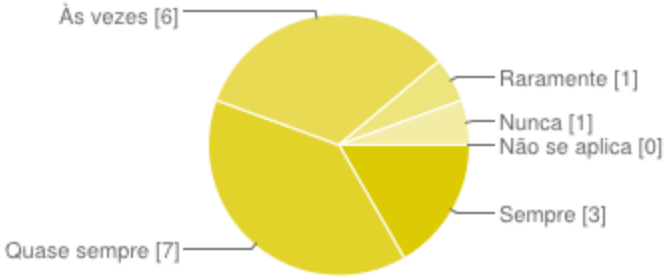
Sempre	**3**	17%
Quase sempre	**1**	6%
Às vezes	**4**	22%
Raramente	**7**	39%
Nunca	**3**	17%
Não se aplica	**0**	0%

7. Os membros da equipe buscam e decidem as tarefas?



Sempre	**9**	50%
Quase sempre	**6**	33%
Às vezes	**1**	6%
Raramente	**2**	11%
Nunca	**0**	0%
Não se aplica	**0**	0%

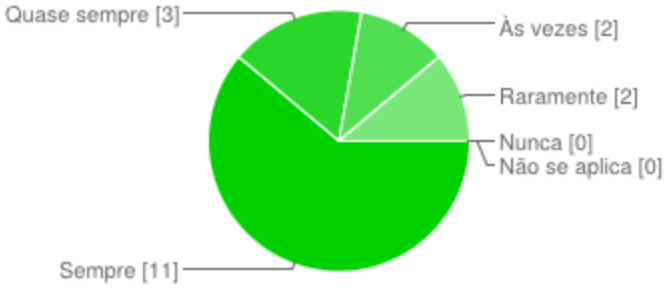
8. Os membros da equipe se cobram as realizações das tarefas?



Sempre	**3**	17%
Quase sempre	**7**	39%
Às vezes	**6**	33%
Raramente	**1**	6%
Nunca	**1**	6%
Não se aplica	**0**	0%

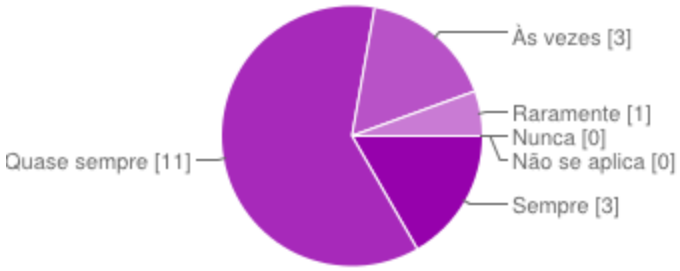
9 - Reunião Retrospectiva

1. Ela existe ao final de cada Sprint?



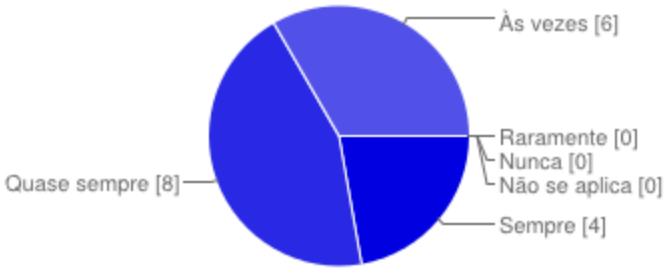
Sempre	**11**	61%
Quase sempre	**3**	17%
Às vezes	**2**	11%
Raramente	**2**	11%
Nunca	**0**	0%
Não se aplica	**0**	0%

2. Todos participam?



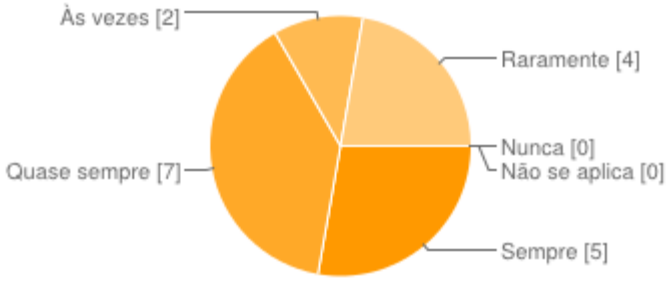
Sempre	**3**	17%
Quase sempre	**11**	61%
Às vezes	**3**	17%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**0**	0%

3. Resulta em sugestões concretas de melhoria?



Sempre	**4**	22%
Quase sempre	**8**	44%
Às vezes	**6**	33%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

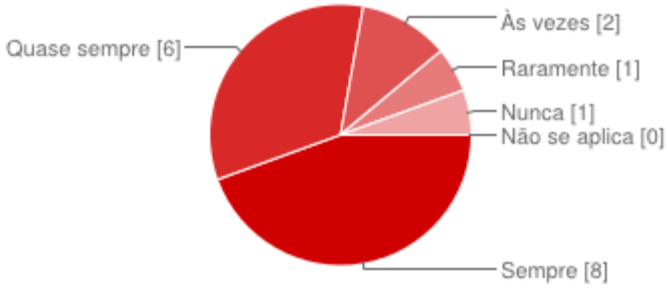
4. Algumas dessas sugestões são implementadas de fato?



Sempre	**5**	28%
Quase sempre	**7**	39%
Às vezes	**2**	11%
Raramente	**4**	22%
Nunca	**0**	0%
Não se aplica	**0**	0%

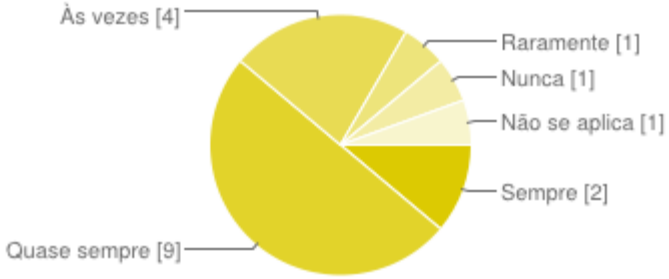
10 - Velocidade

1. A velocidade de desenvolvimento é mensurada?



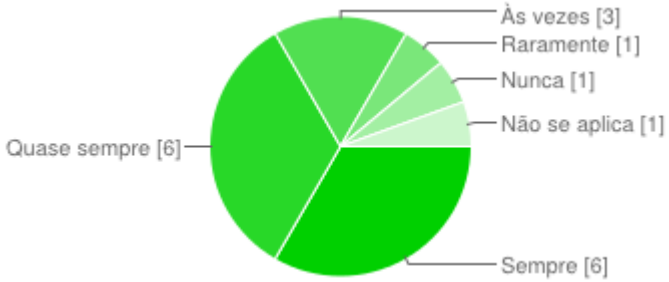
Sempre	**8**	44%
Quase sempre	**6**	33%
Às vezes	**2**	11%
Raramente	**1**	6%
Nunca	**1**	6%
Não se aplica	**0**	0%

2. A velocidade é utilizada para ações corretivas?



Sempre	**2**	11%
Quase sempre	**9**	50%
Às vezes	**4**	22%
Raramente	**1**	6%
Nunca	**1**	6%
Não se aplica	**1**	6%

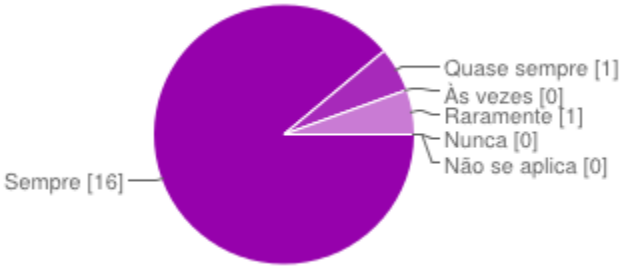
3. Seu registro é utilizado para melhorar futuras estimativas?



Sempre	**6**	33%
Quase sempre	**6**	33%
Às vezes	**3**	17%
Raramente	**1**	6%
Nunca	**1**	6%
Não se aplica	**1**	6%

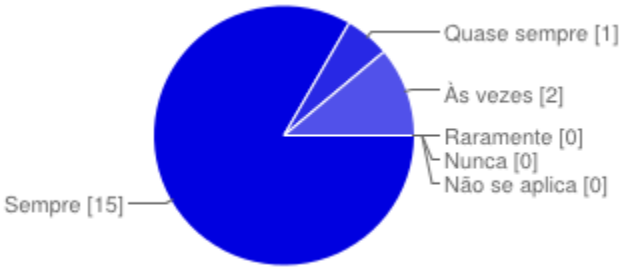
11 - Sprint Backlog

1. O time tem um?



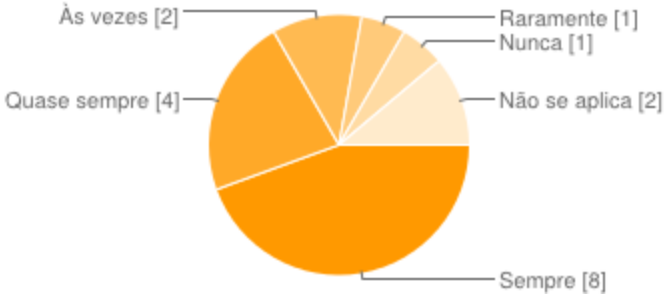
Sempre	**16**	89%
Quase sempre	**1**	6%
Às vezes	**0**	0%
Raramente	**1**	6%
Nunca	**0**	0%
Não se aplica	**0**	0%

2. É visível por toda a equipe?



Sempre	**15**	83%
Quase sempre	**1**	6%
Às vezes	**2**	11%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

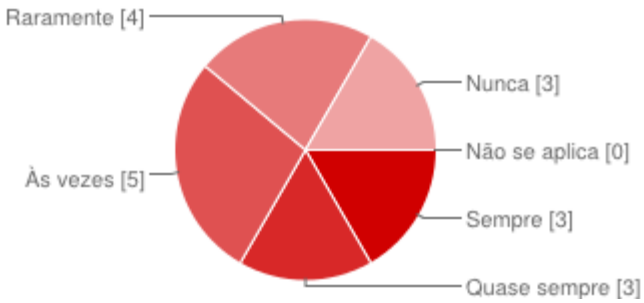
3. É atualizado diariamente (e facilmente) por toda a equipe?



Sempre	**8**	44%
Quase sempre	**4**	22%
Às vezes	**2**	11%
Raramente	**1**	6%
Nunca	**1**	6%
Não se aplica	**2**	11%

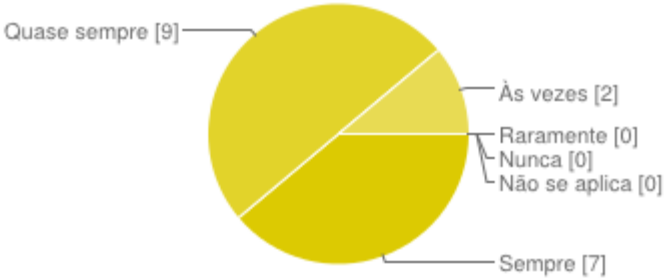
4. A estimativa de trabalho para as tarefas é atualizada diariamente?

Sempre	**3**	17%
Quase sempre	**3**	17%
Às vezes	**5**	28%
Raramente	**4**	22%



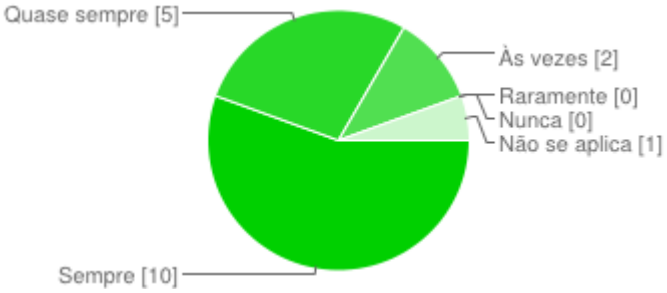
Nunca	**3**	17%
Não se aplica	**0**	0%

5. As funcionalidades e tarefas são facilmente distinguíveis?



Sempre	**7**	39%
Quase sempre	**9**	50%
Às vezes	**2**	11%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**0**	0%

6. Estão claras quais tarefas foram originadas por cada funcionalidade?



Sempre	**10**	56%
Quase sempre	**5**	28%
Às vezes	**2**	11%
Raramente	**0**	0%
Nunca	**0**	0%
Não se aplica	**1**	6%