UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
ESPECIALIZAÇÃO EM ENGENHARIA DE *SOFTWARE*

VALTER RODRIGO EKERT

IDENTIFICAÇÃO DOS PROCESSOS ÁGEIS PARA DESENVOLVIMENTO DE *SOFTWARE* EM MICROEMPRESAS NO MUNICÍPIO DE MEDIANEIRA/PR

MONOGRAFIA DE ESPECIALIZAÇÃO

MEDIANEIRA
2011

VALTER RODRIGO EKERT

IDENTIFICAÇÃO DOS PROCESSOS ÁGEIS PARA DESENVOLVIMENTO DE *SOFTWARE* EM MICROEMPRESAS NO MUNICÍPIO DE MEDIANEIRA/PR

Monografia apresentada como requisito parcial à obtenção do título de Especialista em Pós Graduação em Engenharia de *Software*, da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Medianeira.

Orientador: Prof. M.Eng Juliano Rodrigo Lamb.

MEDIANEIRA

2011

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Especialização em Engenharia de *Software*

TERMO DE APROVAÇÃO

IDENTIFICAÇÃO DOS PROCESSOS ÁGEIS PARA DESENVOLVIMENTO DE *SOFTWARE* EM MICROEMPRESAS NO MUNICÍPIO DE MEDIANEIRA/PR

Por

Valter Rodrigo Ekert

Esta monografia foi apresentada às 13:00 h do dia 15 de dezembro de 2011 como requisito parcial para a obtenção do título de Especialista no curso de Especialização em Engenharia de *Software*, da Universidade Tecnológica Federal do Paraná, Câmpus Medianeira. Os acadêmicos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Juliano Rodrigo Lamb, M.Eng
UTFPR – Câmpus Medianeira
(orientador)

_____
Prof. Claudio Leones Bazzi, M.Eng
UTFPR – Câmpus Medianeira

_____
Prof. Everton Coimbra de Araújo, M.Sc
UTFPR – Câmpus Medianeira

A folha de aprovação assinada encontra-se na coordenação do curso.

Dedico este trabalho a minha família que são os pilares da minha vida.

AGRADECIMENTOS

À Deus pelo dom da vida, pela fé e perseverança para vencer os obstáculos. Sendo a inspiração de todos os dias.

Aos meus pais, pela orientação, dedicação e incentivo nessa fase do curso de pós-graduação e durante toda minha vida.

Ao meu professor orientador Juliano Rodrigo Lamb, pela sua disponibilidade, interesse e receptividade com que me recebeu e pela prestabilidade com que me ajudou.

À Débora Cavalli que sempre esteve junto a mim nos momentos em que precisei.

À minha equipe de trabalho da Serviceweb, e especialmente a Isabella Gonçalves que me ajudou nas pesquisas e elaboração deste trabalho.

Agradeço aos pesquisadores e professores do curso de Especialização em Engenharia de *Software* – Administração e Desenvolvimento, professores da UTFPR, Câmpus Medianeira.

Enfim, sou grato a todos que contribuíram de forma direta ou indireta para realização desta monografia.

“O que não dá prazer não dá proveito.

Em resumo, senhor, estude apenas

o que lhe agradar”.

(WILLIAM SHAKESPEARE)

RESUMO

EKERT, Valter Rodrigo. Identificação dos processos ágeis para desenvolvimento de *software* em microempresas na cidade de Medianeira/PR. 2011. 59 páginas. Monografia (Especialização em Engenharia de *Software*). Universidade Tecnológica Federal do Paraná, Medianeira, 2011.

Este trabalho tem por objetivo realizar um estudo sobre processos que podem ser adotados para desenvolvimento de *software*s em microempresas, que em sua essência, contam com um número pequeno de colaboradores. Foi aplicado um questionário nas microempresas do município de Medianeira, para obter informações sobre qual metodologia ágil esta sendo usada pela microempresa, ou até mesmo qual a metodologia que pretende usar futuramente. Verificou-se que mesmo àquelas que afirmam utilizar uma metodologia, não existe totalidade na adoção das práticas ágeis, restrição essa, muitas vezes face o quadro reservado de colaboradores.

Palavras-chave: metodologia, gerenciamento de projetos, desenvolvimento de *software*.

ABSTRACT

EKERT, Valter Rodrigo. Process Management *Software* for small companies. 2011. 59 paginas. Monografia (Especialização em Engenharia de *Software*). Universidade Tecnológica Federal do Paraná, Medianeira, 2011.

This study it's to conduct a study of certain processes, which can be adopted to support tool in developing *software* for micro-enterprises, which has a small number of employees. Where a questionnaire was administered in micro enterprises in the municipality of Medianeira, for information about what Agile is being used by small business, or even what you want to use the methodology in the future. It can be Scrum or XP.

Keywords: *methodology*, *project management*, *software development*.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CMM	Capability Maturity Model.
CMMI	Capability Maturity Model Integration.
IEC	International Electrotechnical Commission.
ISO	International Organization for Standardization.
MPS-BR	Melhoria de Processos do Software Brasileiro.
SWEBOK	Software Engineering Body of Knowledge.
TI	Tecnologia da informação.
XP	Extreme Programming.

SUMÁRIO

1 INTRODUÇÃO

Segundo ANACLETO e WANGENHEIM (2002), no fim da década de sessenta aconteceu à chamada “crise do *software*” fortemente marcada pela desestruturação de seus desenvolvedores que não seguiam padrões e nem regras para o desenvolvimento dos *software*s. Esse panorama era ainda grave no setor de *hardware*, que se encontrava em desenvolvimento, e necessitava de *software*s melhores para operá-lo. Mas o setor de *software* não conseguia alcançar a velocidade de evolução do *hardware*. Foi este cenário que marcou o inicio da engenharia de *software*. Era preciso estabelecer padrões técnicos organizacionais e até mesmo de implementação para tentar suprir os problemas enfrentados no desenvolvimento dos *software*s.

Ainda de acordo com os mesmos autores, até hoje projetos de *software*s são atrasados e têm problemas de custo, por falta de um planejamento adequado e controle sistemático. Esse problema acarreta muitas vezes o fracasso do projeto.

Com isso as grandes organizações procuram a todo o momento a melhoria nos processos para o desenvolvimento de seus *software*s. Para tanto essas empresas podem utilizar alguns conceitos de gerenciamento de *software*, buscando sempre implementá-los e aperfeiçoá-los ao longo do tempo.

Além disso, com a grande competitividade e a exigência das empresas em estar desenvolvendo produtos cada vez com mais qualidade e em menor espaço de tempo, surge à necessidade de estar adaptando seus processos de uma maneira em que possa estar atendendo suas necessidades da melhor maneira possível.

Tais processos se tornam ainda mais desafiador quando se trata de pequenas empresas, com grandes limitações de recursos, sejam eles humanos, financeiros ou até mesmo de sua infraestrutura, sendo que essas pequenas empresas precisam de uma grande adaptação dos processos para que possam lhe atender.

A proposta desse trabalho é analisar quais os princípios de desenvolvimento de *software* que podem ser observados em microempresas e verificar se podem ser encaixados em uma metodologia única, ou se pode ser afirmado que microempresas não utilizam uma única metodologia e sim, uma coleção de boas práticas de diferentes modelos existentes na literatura.

1.1 OBJETIVO GERAL

Identificar características *ágeis* nos processos de gerenciamento de *software*s em microempresas.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho consistem em:

- Estudar modelos de processo para gerenciamento de *software* voltado a microempresas;
- Fazer um levantamento das metodologias utilizadas pelas microempresas;
- Aplicar um questionário nas microempresas para obter informações sobre as tecnologias utilizadas.

1.3 JUSTIFICATIVA

As microempresas de *software* fazem parte de um mercado crescente no setor de informática atual. Apesar disso, elas ainda mostram um perfil jovem nos processos de desenvolvimento utilizados e na qualidade de seus *software*s. Além de sofrerem com a falta de recurso pessoal, estas empresas na maioria não utilizam técnicas de desenvolvimento, nem têm padrões de processos bem definidos.

O resultado são produtos entregues fora do prazo, *software* com grande margem de erros, baixa usabilidade e consequentemente clientes descontentes.

De acordo com a Pesquisa Nacional de Qualidade e Produtividade no Setor de *Software* Brasileiro, apenas 30% a 35% das microempresas pesquisadas possuem atividades de gerência (ANACLETO, ALESSANDRA; WANGENHEIM, CHRISTIANE, 2002).

Segundo NETO e BOCOLI (2003), MARTINS (2003), PMI (2000), SANDEEP (2002) citados por (TORREÃO, 2006), o gerenciamento de *software* é:

> "... uma profissão relativamente nova e emergente. Isto se deve ao fato de várias organizações, públicas e privadas, instituições de pesquisa e ensino, entre outras, estarem buscando cada vez mais estudar, conhecer, difundir,

> capacitar, implementar e evoluir o conhecimento, as metodologias, as práticas e as ferramentas empregadas nesta área e profissão."

Segundo FREDDO, PARAISO e CAMPAGNOLO (2009), como os participantes das equipes pequenas realizam diversas atividades e desempenham varias funções, estão frequentemente sobrecarregados de trabalho. A consequência é que possuem pouco tempo para documentar suas atividades e, muitas vezes, simplesmente não têm tempo para fazê-lo. Além do mais, a análise do *software* é considerada muitas vezes como sendo de segundo plano. Desta forma, equipes pequenas apresentam necessidades específicas relativas à documentação do desenvolvimento de *software*.

No entanto, o problema da documentação é ainda mais importante e crítico quando pequenas equipes de desenvolvimento de *software* precisam obter certificação de qualidade ou passarem por processos de avaliação de maturidade. Os modelos e normas de qualidade existentes, tais como CMM, CMMI, ISO/IEC, SWEBOK, ISO 9001:2000 e o MPS.BR, têm fortes requisitos em documentação, registros e rastreabilidade. O atendimento a estes requisitos gera muito trabalho extra para as equipes (FREDDO; PARAISO; RAMOS, 2009).

Portanto a necessidade de padronizar os processos é extremamente importante, para o desenvolvimento de *software*, sendo necessário que cada organização se adapte a um modelo de processo que se encaixe na empresa, e que consiga trazer os melhores resultados diminuindo os custos e também o tempo de desenvolvimento.

1.4 ESTRUTURA DO TRABALHO

Esta monografia está estruturada em 6 capítulos, a saber:

- **O capítulo 1** apresenta uma introdução sobre os assuntos abordados no trabalho.
- **O capitulo 2** apresenta uma revisão teórica sobre algumas das metodologias já existentes no mercado.

- **O capitulo 3** apresenta os procedimentos adotados para a realização para o levantamento das informações necessárias para a elaboração deste trabalho.
- **O capitulo 4** apresenta os resultados obtidos através das informações levantadas.
- **O capitulo 5** apresenta as considerações finais.
- **O capitulo 6** apresenta as referencias bibliográficas que foram utilizadas para o desenvolvimento deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Cada vez mais os métodos ágeis têm despertado o interesse da comunidade de Engenharia de *Software* como uma alternativa para o processo de desenvolvimento de sistemas de uma maneira mais rápida, eficiente e que atenda as reais necessidades dos clientes (FAGUNDES; DETERS, 2010).

Dentre várias metodologias existentes no mercado, as que mais se destacam são *XP* e *Scrum*, conforme será visto neste capítulo.

2.1 MODELOS DE PROCESSO DE *SOFTWARE*

Um modelo de processo de desenvolvimento de *software* pode ser visto como uma representação, ou abstração dos objetos e as atividades envolvidas no processo de *software*. Assim sendo, os modelos são mais genéricos do que os processos. Porque além de descreverem atividades de uma forma mais superficial, os modelos normalmente abrangem projetos de diferentes tipos, já os processos são mais detalhados e específicos (OLIVEIRA e PAULA, 2009).

Nos modelos de processo de *software* é dada uma atenção especial à representação abstrata dos elementos do processo e sua dinâmica, não estabelecendo métodos de desenvolvimento, pois este trabalha com um nível mais alto de abstração do que os modelos de ciclo de vida (MACORATTI, 2005).

2.1.1 O modelo Cascata

Modelo criado por Royce em 1970, que também era conhecido como abordagem *top-down*, tem como sua principal característica a sequência de atividades onde cada fase transcorre completamente. Sendo que seus produtos são vistos como inicio de uma nova fase. Este modelo já sofreu várias melhorias sendo muito utilizado nos dias de hoje (MACORATTI, 2005).

Segundo SARTI (2010), a principal ideia deste modelo é que as diferentes etapas de desenvolvimento sigam uma sequência, ou seja, a saída da primeira etapa se encaixa na segunda etapa e a saída da segunda etapa se encaixa na

terceira e assim por diante. As atividades executadas são agrupadas em tarefas, e após são executadas em sequência, de forma que uma tarefa só poderá ter início quando a anterior tiver terminado conforme a FIGURA 1.

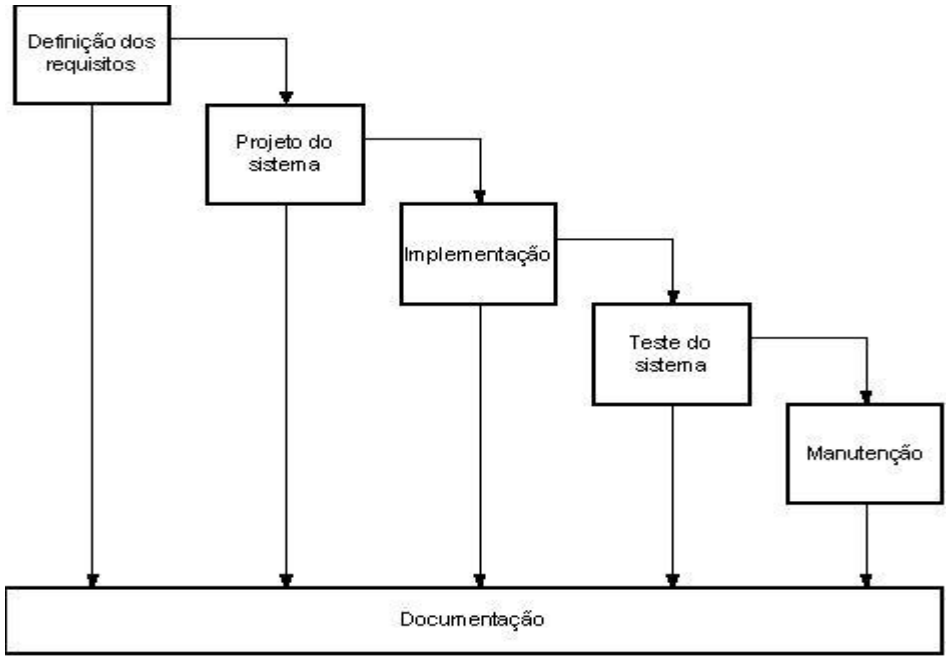


FIGURA 1 - Modelo Cascata
Fonte: DONATO (2011)

O modelo pressupõe que o cliente tenha uma participação ativa no desenvolvimento do projeto. Este modelo diminui o impacto da compreensão adquirida no decorrer projeto, uma vez que um processo não pode voltar atrás para alterar os modelos e as conclusões das tarefas anteriores, é comum que as novas ideias sobre o sistema não sejam aproveitadas.

Em uma tentativa de resolver este tipo de problema foi definido um novo processo baseado em cascata, cuja principal diferença consiste em prever a possibilidade de que a partir de qualquer parte do ciclo pode-se regressar para uma tarefa anterior de forma a contemplar alterações funcionais ou técnicas que, possam ter surgido, em virtude de um maior conhecimento que se tenha obtido.

O risco desta abordagem é que, quando não se tem um processo de gestão do projeto, e de controle das alterações bem definido, corre o risco de nuca ser alcançado o objetivo final do projeto, ou seja, nunca disponibilizar o sistema realmente funcionando (MACORATTI, 2005).

Além disso, o modelo Cascata não leva em consideração as questões mais importantes do desenvolvimento, que é prototipação[1], aquisição de *software*[2] e alterações contínuas nos requisitos[3] (BECK, 2000).

2.1.2 O Modelo Espiral

Segundo OLIVEIRA (2008), o objetivo do modelo espiral é prover um meta-modelo que pode acomodar diversos processos específicos, se tornando um processo de desenvolvimento de *software* que encaixa os elementos de projeto prototipação em suas etapas, em um esforço para combinar as vantagens dos conceitos de *top-down* e *bottom-up*, conforme a FIGURA 2.

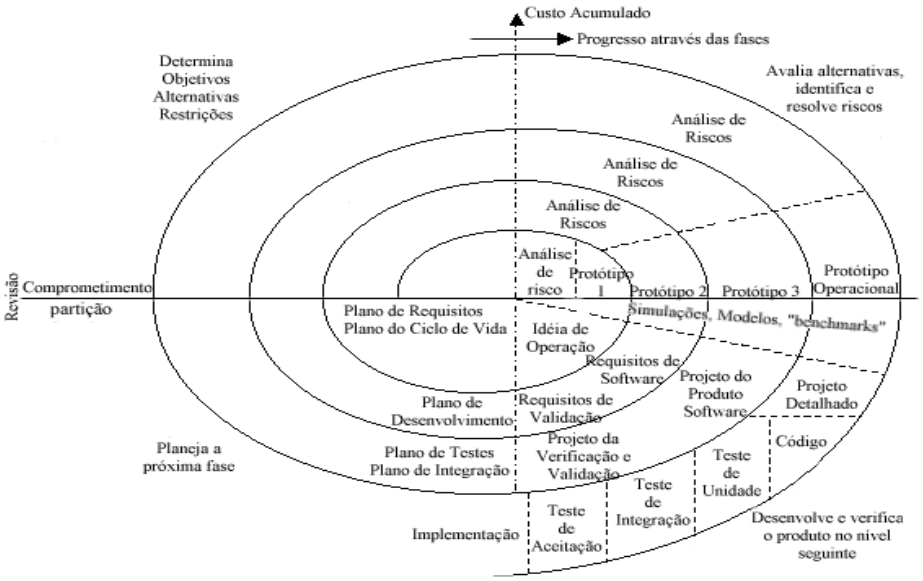


FIGURA 2 - Modelo Espiral
Fonte: DONATO (2011)

[1] Prototipação: Processo interativo de geração de modelos de *software* que faz parte da análise do ciclo de vida do desenvolvimento de sistemas.
[2] *Software*: Seqüência de instruções a serem executadas, na manipulação, redirecionamento ou modificação de uma informação ou acontecimento.
[3] Requisitos: Objetivos ou restrições estabelecidas por clientes ou usuários

Sua principal característica é conduzir o processo de desenvolvimento gerado a partir de um meta-modelo baseado em análise de riscos e planejamento, que são realizados durante toda a evolução do desenvolvimento.

Riscos são casos que podem surgir durante o desenvolvimento de *software* impedindo o processo ou diminuindo a qualidade final do produto. São exemplos de riscos: pessoas que abandonam a equipe de desenvolvimento; ferramentas que não podem ser utilizadas; falha em equipamentos usados no desenvolvimento dentre outros (OLIVEIRA, 2008).

A identificação e o gerenciamento dos riscos são atividades muito importantes no desenvolvimento de *softwares*, porque existe muita imaturidade[4] na área, e também falta de conhecimento técnico assim como ferramentas adequadas.

O modelo espiral demonstra um fluxo de atividades cíclicas e evolutivas constituídas de quatro estágios:

No **primeiro estágio** devem ser determinados objetivos, soluções alternativas e restrições.

No **segundo estágio** devem ser analisados os riscos das decisões do estágio anterior. Durante este estágio podem ser criados protótipos ou pode-se realizar simulações do *software*.

O **terceiro estágio** consiste nas atividades da fase de desenvolvimento, incluindo *design*, especificação, codificação e verificação. A principal característica é especificação de requisitos, do *software*, da arquitetura, da interface com usuário e dos algoritmos e dados que devem ser feita a verificação de maneira apropriada.

O **quarto estágio** compreende a revisão das etapas anteriores e o planejamento da próxima fase. Neste planejamento, dependendo dos resultados nos estágios anteriores, pode-se optar por seguir o desenvolvimento num modelo em Cascata (linear), Evolutivo ou Transformação. Por exemplo, se já no primeiro ciclo, os requisitos forem completamente especificados e também validados pode ser optado por seguir o modelo Cascata. Também, pode-se optar pela criação de novos protótipos, incrementando-o, avaliando novos riscos e replanejando o processo (LEITE, 2007).

[4] Imaturidade: Agir antes de pensar, tomar decisões precipitadas sem saber as conseqüências.

2.1.3 Modelo do processo de desenvolvimento iterativo e incremental

Este modelo é uma expansão do modelo espiral sendo, mais formal. O desenvolvimento de um *software* comercial é uma tarefa que pode ser estendida por muito tempo, possivelmente um ano ou mais. Por isso, é mais prático dividir o trabalho em partes menores ou iterações. Cada iteração resultará num incremento (MACORATTI, 2005).

Iterações são passos em fluxo de trabalho e incrementos são crescimentos do produto, conforme a FIGURA 3.

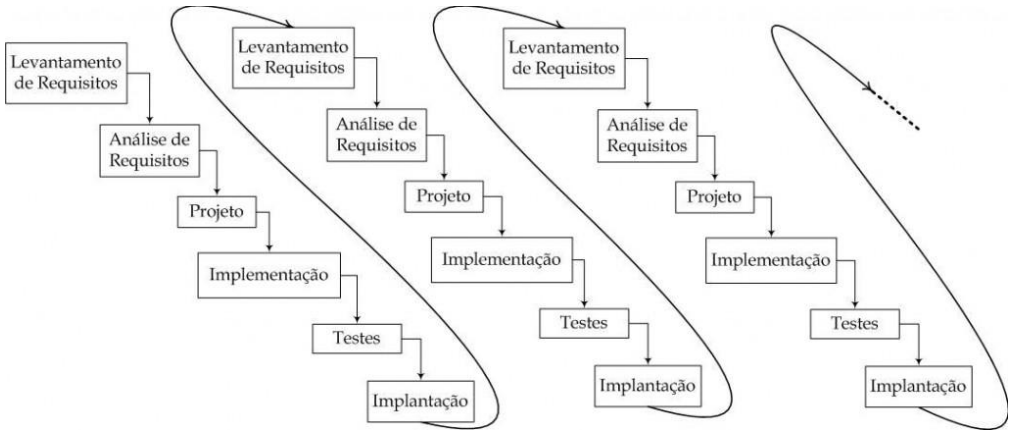


FIGURA 3 - Modelo Iterativo e Incremental
Fonte: DONATO (2011)

Segundo MACHADO (2011), o processo incremental e iterativo faz com que a equipe envolvida possa melhorar a qualidade, do sistema envolvido.

Por exemplo, numa primeira iteração deve-se verificar o sistema como um todo e determinar a viabilidade econômica do mesmo, onde dever ser feito a maior parte da análise e um pouco de implementação;

Numa segunda iteração, deve-se concluir a análise, e fazer uma parte significativa da implementação;

Numa terceira iteração, deve-se fazer uma parte substancial da implementação, testar e integrar. Ou seja, a principal consequência da aproximação iterativa é que os produtos finais de todo o processo vão sendo amadurecidos e completados ao com o tempo, mas cada iteração produz sempre um conjunto de produtos finais. A cada iteração são realizadas as seguintes tarefas:

- **Análise**: Refinamento de requisitos, refinamento do modelo conceitual;

- **Projeto:** Refinamento do projeto arquitetural, projeto de baixo nível;
- **Implementação:** Codificação e testes;
- **Transição para produto:** Documentação e instalação;

2.1.3.1 Vantagens do processo incremental e iterativo.

Segundo MACORATTI (2005) as vantagens do processo iterativo e incremental são:

- Possibilita a avaliação dos riscos e dos pontos críticos do projeto com antecedência, podendo identificar métodos para eliminar ou controlar os mesmos;
- Reduz os riscos envolvendo custos em uma única vez. Se a equipe precisar repetir a iteração, a empresa perde somente o esforço mal direcionado de uma iteração, e não o valor de todo o projeto;
- Define uma arquitetura que possa ajudar melhor no desenvolvimento;
- Disponibiliza logo de inicio um conjunto de regras para controlar melhor os pedidos de alterações futuras;
- O usuário pode se entusiasmar com a primeira versão do sistema e pensar que tal versão já corresponde ao sistema como um todo.

2.1.4 Modelo de Prototipagem

Segundo VARGAS (2011), protótipos são construídos para simular a aparência e a funcionalidade de um projeto de *software*. Poderia ser identificado como a primeira versão do sistema de *software* possibilitando a pessoa que esta desenvolvendo conhecer melhor os problemas e também as possíveis soluções. Algumas vezes o protótipo pode ser usado para definir a aparência externa do sistema, conforme apresenta na FIGURA 4.

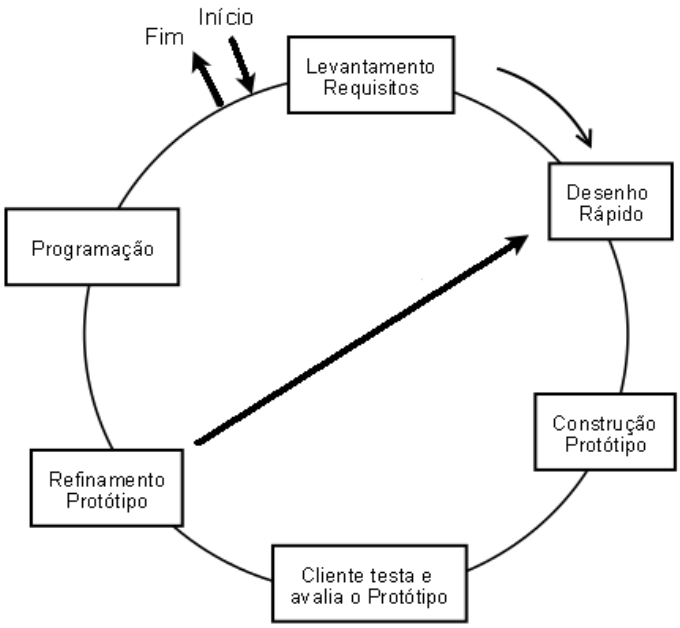


FIGURA 4 - Modelo de Prototipagem

As etapas do modelo são:

- **Levantamento dos requisitos:** o desenvolvedor e o cliente definem as funcionalidades do *software*, identificando quais os requisitos são necessários inicialmente.
- **Desenho rápido:** é entregue ao cliente uma versão resumida do projeto, este poderia ser até um *layout*. Nessa etapa o cliente vai poder ver como o *software* ira funcionar e quando ficará pronto.
- **Construção do protótipo:** após ser feito um projeto rápido e concluídas todas as modificações necessárias no projeto, passa-se para uma rápida construção de um protótipo que poderá ser a versão final do produto.
- **Cliente testa e avalia o protótipo:** e feito à entrega do produto ao cliente que verifica se é isso mesmo que ele precisa.
- **Refinamento do protótipo:** aplicam-se as melhorias necessárias para a conclusão do projeto, sendo feito as modificações solicitadas pelo cliente. Importante para que as necessidades do cliente sejam satisfeitas.

- **Programação:** identificam-se os requisitos podendo ser descartado o protótipo, logo após e a versão de produção deve ser construída (podendo até ser baseada no protótipo) sendo considerados todos os critérios de qualidade.

Segundo PFLEEGER (2004), o desenvolvimento do sistema começa com alguns requisitos que são fornecidos pelos clientes e usuários, E então, as alternativas são exploradas. Verificam-se as telas, tabelas, relatórios e também outras saídas do sistema, diretamente utilizadas pelos clientes e usuários. Assim que os usuários e clientes decidem o que querem, realmente do sistema os requisitos são novamente revisados.

Uma vez que haja um acordo de como devem ser os requisitos, o desenvolvedor se volta para as atividades, a fim de reconsiderar e alterar a especificação. Finalmente, o sistema é codificado, e as alternativas, discutidas, de novo com uma possível iteração entre requisitos e projeto (VARGAS, 2011).

2.2 METODOLOGIAS ÁGEIS

O desenvolvimento ágil começou a partir da década de 90, onde foram criadas um conjunto de metodologias que auxiliam no desenvolvimento dos projetos.

Segundo LUDVIG; REINERT (2007) Os métodos sugerem uma abordagem de desenvolvimento ágil, em que os processos adotados tentam se adaptar às mudanças, apoiando a equipe de desenvolvimento em seu trabalho.

2.2.1 Manifesto ágil

Segundo SABBAGH (2010), em fevereiro de 2001, um grupo de dezessete representantes de metodologias de desenvolvimento de *software* se reuniu por dois dias em uma estação de esqui para verificar os pontos em comum que existiam entre suas metodologias. A partir então dessa reunião, é que foi criado o Manifesto Ágil, onde são definidos os valores Ágeis, que são:

a) **Indivíduos e interações** mais que processos e ferramentas;

b) ***Software* em funcionamento** mais que documentação abrangente;

c) **Colaboração com o cliente** mais que negociação de contratos;

d) **Responder a mudanças** mais que seguir um plano;

O manifesto reconhece que determinados conceitos como processos, ferramentas, documentação, contratos e planos podem ser muito importantes no desenvolvimento de *software*, mas que ainda existem conceitos ainda mais importantes que devem ser valorizados. A seguir serão listados os doze princípios *ágeis* (BECK, BEEDLE, *et al.*, 2001):

- Prioridade em satisfazer o cliente por meio de entregas contínuas do *software* com valores agregado.
- Mudanças nos requisitos são sempre bem-vindas, mesmo com o processo do desenvolvimento mais adiantado. Processos ágeis dão vantagem e competitividade para o cliente.
- Entregar sempre o *software* funcionando, em uma curta escala de tempo.
- Toda a equipe deve trabalhar diariamente em conjunto no desenvolvimento de todo o projeto.
- Construir projetos com indivíduos motivados, dando a eles ambiente e suporte necessário, e confiando neles para fazer o trabalho.
- O método mais eficiente e eficaz de transmitir informações para a equipe e conversando diretamente.
- O *software* deve sempre estar funcionando.
- Os processos ágeis promovem desenvolvimento sustentável. A equipe que esta no projeto deve manter o ritmo constante indefinidamente.
- Atenção com a excelência e a técnica e um bom design aumenta a agilidade.
- Simplicidade a arte de maximizar a quantidade de trabalho não realizado é essencial.
- As melhores arquiteturas, requisitos e designs precisam de equipes que se auto-organizam.
- Em intervalos regulares, a equipe deve refletir sobre como se tornar mais eficaz e refinando e ajustando seu comportamento de acordo.

2.2.2 Desenvolvimento ágil

Segundo TEIXEIRA; PIRES e PINTO (2005), a maioria do desenvolvimento de *software* é feita de forma desordenada, assim caracterizada pelo método programar e corrigir. O código é feito sem um planejamento aprofundado do problema, sendo feito conforme vão surgindo os problemas, isso pode até se tornar bem eficaz para pequenas aplicações, mas com o aumento do sistema construído, torna-se muito mais difícil adicionar as novas funcionalidades ao programa ou alterar outras que já existem.

Este estilo de desenvolvimento de *software* foi feito durante muito tempo, tendo como mais forte concorrente o desenvolvimento ordenado, que impõe um processo de desenvolvimento mais disciplinado, com o objetivo de tornar o desenvolvimento de *software* mais previsível e eficiente, conseguido através do desenvolvimento de documentação detalhada, e dando um grande destaque ao planejamento inspirado em outras áreas da engenharia de *software*.

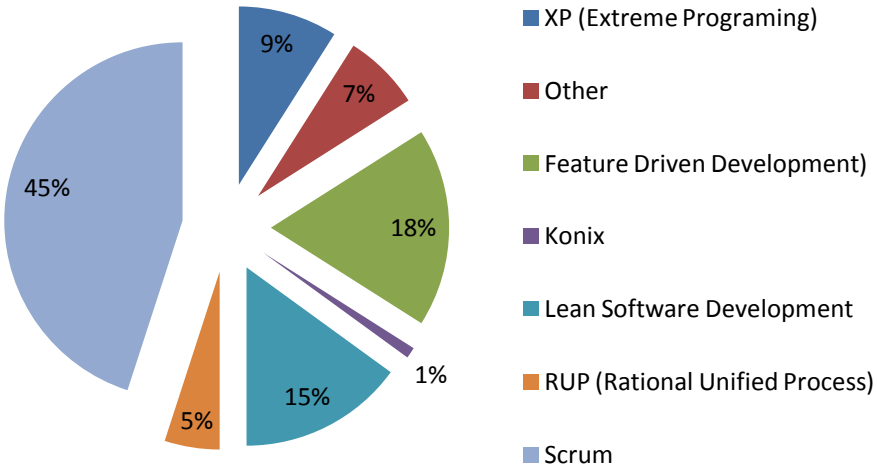


FIGURA 5 - Metodologias Ágeis
Fonte: CHAPMAN (2010)

Proveniente desta metodologia surgiu um novo grupo nomeado de Metodologias Ágeis (FIGURA 5). Ela apresenta ainda quais às metodologias *ágeis* mais utilizadas (DANIEL, FERNANDO, *et al.*, 2005). Os métodos ágeis são métodos

que tentam encontrar um ponto de equilíbrio entre o “pouco trabalho” e o “trabalho excessivo”. Como resultado, as metodologias ágeis têm sido adotadas para o desenvolvimento de diversas aplicações, e alterado expressivamente as metodologias de engenharia de *software*. A principal diferença pode ser observada pela desnecessidade de gerar muita documentação para fazer as mesmas tarefas que eram feitas antes. Estas metodologias são mais orientadas ao código propriamente dito.

Segundo SOARES (2004), as Metodologias Ágeis são bem aceitas pela indústria de *software* e também por pesquisadores da Engenharia de *Software*. Apesar de não existir uma grande base de comparações os primeiros resultados são satisfatórios.

O desafio futuro das Metodologias Ágeis é encontrar formas de eliminar alguns dos seus pontos fracos, como a falta de análise de riscos, tudo isso sem torná-las metodologias pesadas. Outro desafio é como usar essas metodologias em grandes corporações e em grandes equipes, uma vez que normalmente são baseadas em equipes pequenas. Neste caso, pelo menos é necessário que sejam resolvidos os problemas de comunicação internos na equipe, sendo que é muito comum nas grandes empresas os colaboradores estarem em ambientes separados. Apesar do interesse nas metodologias ágeis, falta ainda casos de sucesso de seu uso em projetos grandes e críticos.

2.3 EXTREME PROGRAMMING

O *Extreme Programming* (XP) é uma metodologia ágil que pode ser trabalhado em equipes pequenas e médias que desenvolvem *software* baseado em requisitos e que se modificam rapidamente (REIS, 2008).

Dentre as principais diferenças do XP em relação às outras metodologias estão:

- *Feedback* constante
- Abordagem incremental
- A comunicação entre as pessoas é encorajada.

Na maioria das vezes as regras do XP não fazem sentido quando aplicadas isoladamente. É a junção de todo seu conjunto que sustenta o sucesso do XP,

encabeçando uma verdadeira revolução para o desenvolvimento de *software*. O XP tem como característica o desenvolvimento rápido do projeto e visando garantir a satisfação do cliente, além de sempre favorecer o cumprimento das estimativas. As regras, práticas assim como os valores da XP proporcionam um ótimo ambiente para o desenvolvimento de *software*, que são conduzidos por quatro valores, sendo eles: comunicação, simplicidade e *feedback*, conforme a FIGURA 6 (REIS, 2008).

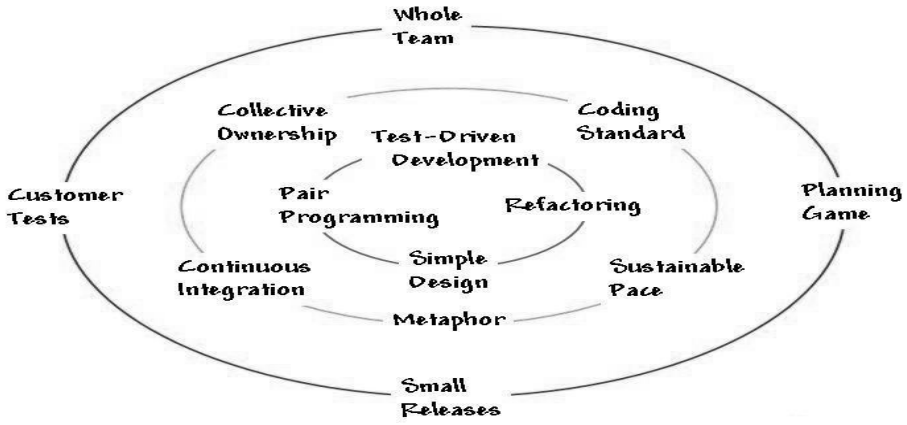


FIGURA 6 - Ciclo de vida do XP
Fonte: FERNANDES (2010)

O *XP* baseia-se nas 12 práticas que são: planejamento, metáfora, entregas frequentes, projetos simples, testes, programação em pares, refatoração, propriedade coletiva, integração continua, 40 horas de trabalho semanal, cliente presente e código padrão, listados a seguir (SOARES, 2004).

Planejamento: verifica o que necessita ser feito e o que pode ser adiado no projeto. O XP baseia-se nos requisitos atuais para o desenvolvimento de *software*, e não nos requisitos futuros. Além disso, com a utilização do XP, é possível evitar os problemas de relacionamento entre a área de negócios e a área do desenvolvimento. As duas áreas devem trabalhar juntas para o sucesso do projeto e cada uma deve focar em suas partes específicas para o desenvolvimento do mesmo. Desta forma, enquanto a área de negócios deve decidir sobre qual será o escopo, a composição das versões e também as datas de entrega, a equipe de desenvolvedores devem decidir sobre as estimativas de prazo, sobre o processo de desenvolvimento e o cronograma detalhado. Tudo isso para que o *software* seja entregue nas datas estabelecidas.

Metáfora: é a descrição do *software* sem a utilização de termos técnicos, tendo a característica de guiar o desenvolvimento do *software*.

Entregas frequentes: visa construir *software*s mais simples, e atualizar conforme o surgimento dos requisitos. Para cada versão entregue o tamanho deve ser o menor possível, contendo apenas os requisitos de maior valor para o negócio. O ideal é que seja entregue versões a cada mês, ou ainda no máximo a cada dois meses, aumentando assim a possibilidade de um *feedback* rápido por parte do cliente. Isto evita algumas surpresas caso o *software* seja entregue somente após um longo tempo, e melhora também as avaliações do cliente, aumentando assim a probabilidade da versão final estar de acordo com os requisitos que foram repassados pelo cliente no inicio do projeto.

Projeto simples: o *software* desenvolvido utilizando o XP deve ser o mais simples possível, sem a preocupação de requisitos futuros. Eventuais requisitos futuros devem ser adicionados quando necessário.

Testes: o XP focaliza a validação do projeto durante todo o processo de desenvolvimento. Os programadores desenvolvem o *software* criando primeiramente os testes.

Programação em pares: é feita implementação em dupla, dois desenvolvedores trabalham em um único computador. O código é implementado pelo desenvolvedor que está com o controle do teclado e do mouse, enquanto o outro desenvolvedor observa o trabalho que está sendo feito, procurando identificar erros de sintaxe e de semântica, pensando em como melhorar o código que está sendo implementado. Esses papéis devem ser alterados continuamente. Uma das vantagens da programação em dupla é possibilitar os desenvolvedores estarem sempre aprendendo um com o outro.

Refatoração: tem como foco o aperfeiçoamento do projeto do *software* estando presente em todo o desenvolvimento. Devendo ser feita apenas quando é necessário, ou seja, quando se percebe que é possível simplificar o módulo atual sem perder nenhuma funcionalidade.

Propriedade coletiva: o código do projeto pertence a todos os colaboradores da equipe. Sendo assim qualquer pessoa pode adicionar valor a um código, mesmo que ele não seja o que tenha desenvolvido, porem o colaborador deve fazer os testes necessários. Isto só é possível porque no XP todos são responsáveis pelo *software*. Uma das vantagens é que, caso um colaborador deixe o

projeto antes do fim, a equipe consegue continuar o projeto sem muitas dificuldades, pois todos já conhecem todas as partes do *software*, mesmo não sendo de forma detalhada.

Integração contínua: é interagir e construir o sistema de *software* durante várias vezes no dia, mantendo os programadores sempre em sintonia, possibilitando processos rápidos. Integrar apenas um conjunto de modificações por vez é uma prática que funciona bem porque fica claro que deve ser feito as correções quando existem falhas nos testes. Esta prática é facilitada com o uso de uma máquina apenas para integração, que deve ter o livre acesso para todos os membros da equipe.

40 horas de trabalho semanal: com característica de não fazer hora extra. Caso seja necessário trabalhar mais do que as 40 horas previstas o problema deve ser resolvido não com aumento de horas trabalhadas diariamente, mas com melhor planejamento. Esta prática procura focar nas pessoas e não nos processos e planejamentos. Caso necessário, os planos devem ser alterados, ao invés de sobrecarregar os colaboradores.

Cliente presente: é importante a participação do cliente durante todo o processo de desenvolvimento do projeto. O cliente deve estar sempre disponível para tirar as dúvidas referentes aos requisitos, evitando atrasos e também construções erradas. O que pode ser feito é manter o cliente como parte integrante da equipe, para o desenvolvimento.

Código padrão: padronizar a arquitetura do código, para que este possa ser compartilhado com todos os programadores.

2.4 SCRUM

O *Scrum* é uma metodologia cujas práticas são aplicadas em um processo iterativo e incremental. Assume-se que os projetos no qual o *Scrum* se insere são complexos e imprevisíveis, não sendo possível prever tudo que irá acontecer. Por esta razão, ele oferece um conjunto de práticas que torna tudo isso visível (SCHWABER, 2004).

Seu objetivo é fornecer um processo conveniente para projeto e desenvolvimento orientado a objeto (SANTOS e LUZ, 2003).

O *Scrum* mostra uma abordagem empírica que coloca em pratica algumas ideias de controle de processos industriais para o desenvolvimento de *software*s, reintroduzindo as ideias de flexibilidade, adaptabilidade e produtividade. O foco da metodologia é trabalhar de forma flexível em um ambiente em constante mudança.

A ideia principal do *Scrum* é que o desenvolvimento de *software*s envolva muitas variáveis técnicas, como requisitos, recursos e tecnologias, que podem mudar durante o processo. Isto torna o processo de desenvolvimento mais imprevisível e complexo, necessitando mais flexibilidade para acompanhar as mudanças. O resultado do processo deve ser um *software* útil para o cliente (FAGUNDES; DETERS, 2010).

A metodologia é baseada em princípios semelhantes aos do XP: com equipes pequenas, requisitos pouco estáveis ou desconhecidos, iterações curtas para promover visibilidade no desenvolvimento. A FIGURA 7 mostra o modelo do ciclo de vida do Scrum.

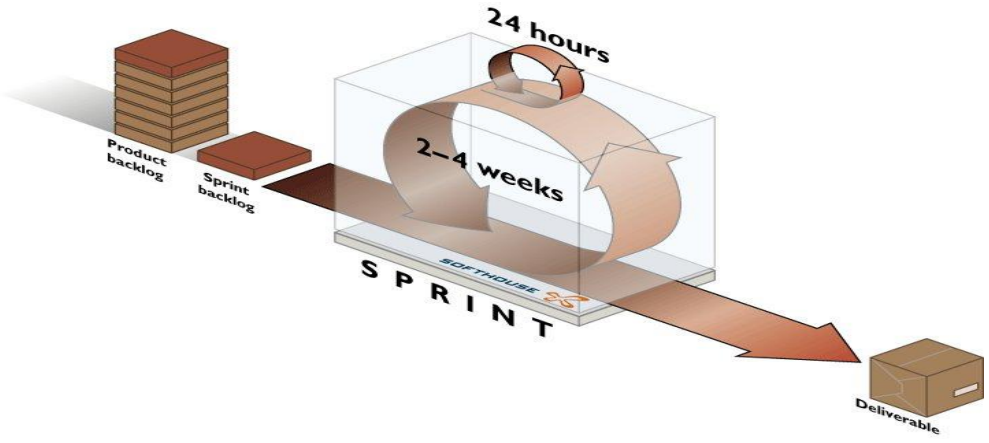


FIGURA 7 - Ciclo de vida do Scrum
Fonte: NASCIMENTO (2010)

O ciclo de vida da *Scrum* é baseado nas três fases principais, divididas em subfases (SOARES, 2004).

Pré-planejamento (*Pre-game phase*): os requisitos são descritos em um documento que se chama *backlog*. Logo após eles são priorizados e são feitas as estimativas do esforço para o desenvolvimento de cada requisito. O planejamento também inclui a definição da equipe para o desenvolvimento, as ferramentas a

serem usadas, assim como os possíveis riscos do projeto e as suas necessidades de treinamento. Finalmente é proposta uma arquitetura de desenvolvimento.

Desenvolvimento (*game phase*): as variáveis técnicas e do ambiente são observadas, identificadas e controladas durante o processo de desenvolvimento. Ao invés de considerar essas variáveis somente no início do projeto, como nas metodologias tradicionais, no *Scrum* o controle é feito continuamente, aumentando a flexibilidade para acompanhar as mudanças. Nesta fase o desenvolvimento do *software* é feito em ciclos (*sprints*) onde novas funcionalidades podem ser adicionadas. Cada um desses ciclos são desenvolvidos de forma tradicional, onde, primeiramente é feita a análise, em seguida o projeto, implementação e os testes. Também cada um desses ciclos são planejados para durar de uma semana até um mês.

Pós-planejamento (*post-game phase*): após a fase de desenvolvimento, são feitas as reuniões para analisar o progresso do projeto e demonstrar como esta o *software* atual para o cliente. Nesta fase são feitas as etapas de integração, e testes finais juntamente com a documentação (SOARES, 2004).

2.4.1 Artefatos

Dentre os principais artefatos produzidos pelo SCRUM estão:

Product Backlog: Uma lista de funcionalidades definidas pelo Cliente. Lista que pode sofrer alterações durante o decorrer do projeto dependendo das necessidades. Das funcionalidades definidas o cliente deve também definir qual a prioridade que terá cada item da lista.

Sprint Backlog: Lista de tarefas do *Sprint*[5]. Estas tarefas são atividades que foram priorizadas pelo cliente e que tiveram suas estimativas definidas pela equipe que se compromete a entregar no período do *Sprint*. Caso alguma das tarefas definidas para o *Sprint* não seja atendida esta devem ser priorizada no próximo *Sprint Backlog*.

Burndown chart: É o principal artefato para realizar o acompanhamento do projeto. Este é um gráfico que é gerado a partir da lista de tarefas do *Sprint*, e

[5] Sprint: Representa um tempo dentro do qual um conjunto de atividades deve ser executado.

mostra o número de horas restantes para a conclusão do projeto. Por este artefato é possível visualizar a velocidade da equipe no *Sprint*.

2.4.2 Papeis

O *Scrum* tem os seguintes papeis (principais):

- *Product Owner*;
- *Scrum Master*;
- *Scrum Team*;

Conforme (MATSUDO, 2009), pode-se descrever cada um dos papeis:

- **Product Owner**: Refere-se ao dono do produto, basicamente o cliente, responsável por definir o que será o produto, quais suas características, e quais serão suas funcionalidades, prioridades e podendo aprovar ou não o resultado do trabalho desenvolvido. Define a data de entrega e quando necessário redefine as prioridades e as características do produto.
- **Scrum Master:** Trabalha próximo ao *Product Owner*, tem a responsabilidade de garantir que a equipe seja funcional e produtiva e deve acompanhar tudo o que está sendo realizado. Deve ajudar a equipe removendo os obstáculos que possa ocorrer no desenvolvimento das *Sprints*, e também deve proteger a equipe de riscos e interferências externas assim como o excesso de otimismo.
- **Scrum Team:** Também chamada de Equipe, são pessoas responsáveis por desenvolver e entregar as *Sprints* realizadas. Deve ter como características: disciplina e auto gerência. Geralmente são formados em pequenos grupos.

2.5 MICROEMPRESAS

São consideradas microempresas aquelas cuja receita bruta anual é igual ou inferior a R$ 433.755,14 (quatrocentos e trinta e três mil, setecentos e cinquenta e

cinco reais e quatorze centavos), e que tenha de 0 a 9 funcionários em caso de prestadora de serviço, ou de 0 a 19 no caso de indústrias (LEITE, 2006).

2.5.1 Desenvolvimento de *software*s em microempresas

Na maioria das microempresas o desenvolvimento de *software* é feito sem um planejamento adequado, onde normalmente não são feitas a analise do *software* a ser desenvolvido, e nem mesmo são feitos testes rigorosos para verificar se o *software* esta realmente atendendo as necessidades do cliente. No entanto existem alguns agravantes que fazem com que estas etapas não são desenvolvidas, devido às limitações que existem nas microempresas.

2.5.2 Limitações

As dificuldades encontradas durante a implantação de práticas para melhoria de processo têm sido em fator marcante nas microempresas. Segundo (GUERRA, 2010) as principais limitações são:

- Equipes muito pequenas com falta de recursos financeiros para contratar novos colaboradores. E também a falta de experiência na contratação de novos integrantes para a equipe;
- Equipes das empresas sem os conhecimentos básicos de Engenharia de *Software*;
- Pessoas com dificuldade em conciliar os papéis desempenhados com as práticas que devem ser realizadas;
- Falta de disciplina interna para realizar as atividades de maneira equilibrada;
- Falta de compromisso, por parte dos responsáveis pela empresa. E falta de compromisso dos dirigentes para que se tenha desempenho, e um processo definido, por parte dos elementos da empresa, de uma forma homogênea e também disciplinada;
- Realização de trabalhos paralelos, comprometendo o trabalho que estava sendo realizado;
- Falta de coerência quanto aos objetivos dos responsáveis e da equipe operacional. Os gerentes de projetos e desenvolvedores sentem a

necessidade de adotar melhores práticas, enquanto o objetivo da diretoria é meramente comercial.

3 PROCEDIMENTOS METODOLÓGICOS DA PESQUISA

Na realização dessa pesquisa, foram definidos itens como população e amostragem, os instrumentos de pesquisa utilizados, bem como os seus procedimentos de utilização, aplicação ou análise. No decorrer deste capítulo encontrar-se-ão essas informações devidamente especificadas.

3.1 POPULAÇÃO E AMOSTRAGEM

Compreende a população desse projeto as microempresas de desenvolvimento de *software*, cujo numero de colaboradores é menor que 10 pessoas. E como amostragem as 4 microempresas existentes no município de Medianeira.

3.2 O MUNICÍPIO

Medianeira é um município que esta localizada na região Oeste paranaense tendo o setor agroindustrial como a principal base da economia do município.

Atualmente conta com uma ampla área industrial e também com um setor de tecnologia da informação em plena expansão, contando com mais de 7 (sete) empresas no ramo de desenvolvimento de sistemas, e mais de 10 (dez) no ramo de concerto e manutenção de computadores.

3.3 INSTRUMENTOS DE PESQUISA

A pesquisa foi feita através de um questionário (APÊNDICE A) que foi aplicado em todas as 4 microempresas do município de Medianeira durante o período de 25 de Outubro de 2011 à 05 de Novembro de 2011.

3.4 PROCEDIMENTOS DA PESQUISA

O questionário foi elaborado com 20 perguntas seguindo os conceitos das metodologias ágeis, onde também foram utilizados alguns conceitos específicos do *Scrum* e do *XP*. As perguntas foram elaboradas através do sistema de alternativas, sendo: **Sim**, **Não**, **Gostaria** e **Inviável**, sendo questões fechadas e podendo apenas ser escolhido uma das alternativas.

Para a aplicação do questionário foi utilizado à ferramenta do *Google Docs*., conforme a FIGURA 8 que mostra o modelo do questionário.

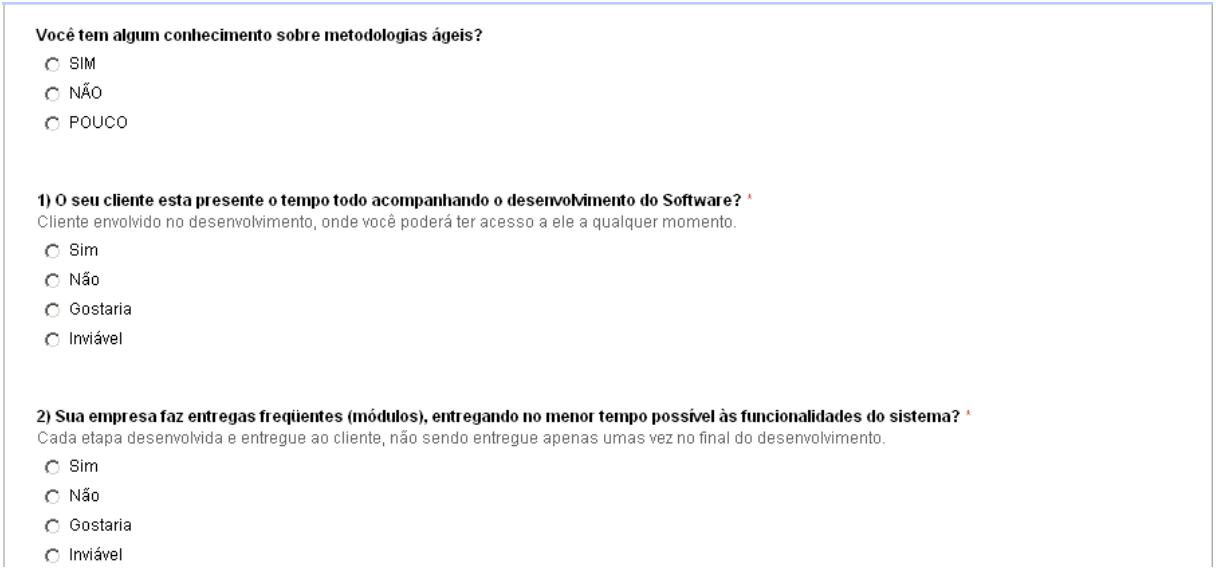


FIGURA 8 - Modelo de questionário

3.5 ANÁLISE DOS DADOS

Após coletados os dados através do questionário os mesmos foram analisados através de gráficos gerados sobre as informações obtidas, sendo feita uma análise quantitativa da porcentagem de empresas que utilizam:

- Metodologias ágeis;
- Conceitos de *Scrum*;
- Conceitos de XP;
- Pretendem utilizar *Scrum*;
- Pretendem utilizar XP;

4 RESULTADOS E DISCUSSÃO

Neste capitulo são apresentados os resultados obtidos através da análise dos resultados, da pesquisa realizada nas microempresas de desenvolvimento do município de Medianeira.

É demonstrado através de gráficos quanto que cada microempresa utiliza dos princípios das metodologias ágeis. Também será demonstrado uma visão geral de todas as microempresas. Os resultados detalhados são apresentados no APÊNDICE B, sendo aqui apresentado o resumo destas informações.

4.1 METODOLOGIAS ÁGEIS NAS MICROEMPRESAS

Nesta seção é apresentada a porcentagem de microempresas que tem conhecimento sobre metodologias ágeis conforme o gráfico da FIGURA 9.

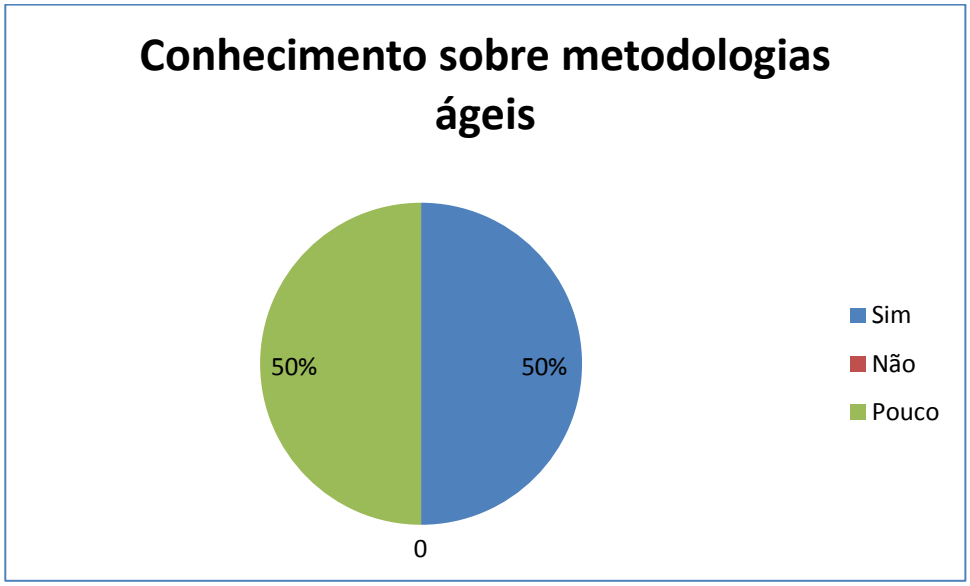


FIGURA 9 - Conhecimento sobre metodologias ágeis

De acordo com a FIGURA 9 é possível visualizar que 50% das empresas tem conhecimento sobre metodologias ágeis, enquanto outros 50% tem pouco conhecimento, este indicador mostra que alguns princípios de metodologias ágeis podem não estar sendo usados nas microempresas que tem pouco conhecimento,

sendo que nestas microempresas o desenvolvimento deve ser feito de forma artesanal.

4.2 CARACTERÍSTICAS DO XP NAS MICROEMPRESAS

Nesta seção são apresentadas as 4 empresas que utilizam as características do XP para o desenvolvimento dos seus *softwares*, conforme os gráficos das FIGURAS 10 e 11.

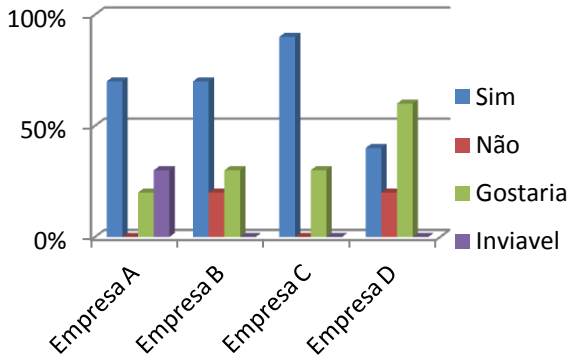


FIGURA 10 – Microempresas que utilizam características do XP

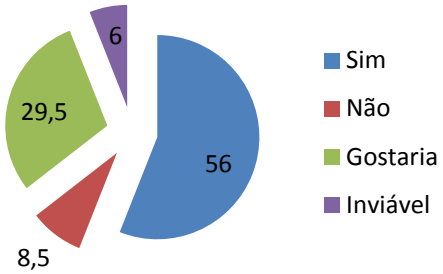


FIGURA 11 - Média geral do XP

De acordo com a FIGURA 10 é possível visualizar que as empresas **A**, **B** e **C** utilizam mais de 50% das características que compõem a metodologia do *XP*. Enquanto que a empresa **D** não chega a utilizar 40% dessas características, porém a empresa **D** gostaria de utilizar em quase 60%.

No comparativo com a FIGURA 11 que mostra a média geral. É possível perceber que, mesmo que as empresas não utilizam de todas as características do *XP* em breve as empresas tendem a utilizar, já que somando o percentual de que já é utilizado com o que gostaria de utilizar chega a quase 90%. Fazendo com que realmente o *XP* será uma metodologia muito usada nas microempresas da cidade de Medianeira, condicionado aos dados observados na pesquisa.

4.2.1 Pontos em comum do XP nas microempresas

Nesta seção serão exibidas quais práticas ágeis são comuns a todas as 4 empresas que utilizaram das tecnologias do XP, mostrando em que pontos que todas as empresas concordam com o mesmo objetivo, conforme a TABELA 1.

TABELA 1 - Pontos em comum do XP

Empresas	A	B	C	D
1) Sua empresa faz entregas frequentes (módulos), entregando no menor tempo possível às funcionalidades do sistema?	Sim	Sim	Sim	Sim
2) Existem padrões de codificação? Onde todos os programadores escreverão código respeitando as regras?	Sim	Sim	Sim	Sim

Pode ser verificado na TABELA 1 que todas as empresas trabalham com entregas frequentes, dividindo o sistema em partes e entregando cada parte no menor tempo possível para seus clientes. Além de que todas as empresas utilizam de algum padrão para o desenvolvimento de seus projetos e também para fazer a codificação, fazendo com que todos os seus colaboradores programem utilizando os mesmos conceitos.

4.3 CARACTERÍSTICAS DO SCRUM NAS MICROEMPRESAS

Nesta seção serão apresentadas as empresas que utilizam os conceitos o *Scrum*, conforme os gráficos representados pelas FIGURAS 12 e 13.

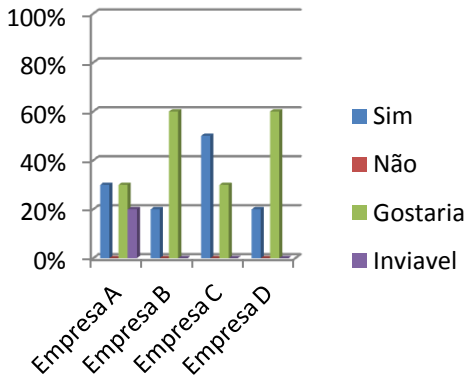


FIGURA 12 - Utiliza características do Scrum

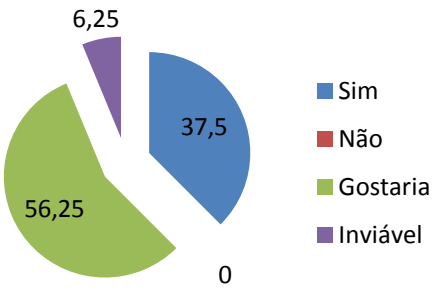


FIGURA 13 - Média geral do Scrum

De acordo com a FIGURA 12 é possível verificar que apenas a empresa **C** utiliza quase 50% das características do *Scrum.* As empresas **A**, **B** e **D** utilizam menos de 25%, porém a empresa **B** e **D** gostaria de utilizar as características em quase 60%, enquanto que a empresa **A** acha que quase 20% das características do *Scrum* são inviáveis para utilização na empresa.

No comparativo com a FIGURA 13, que mostra a média geral. É possível perceber que, apenas 37,5% das características do *Scrum* são utilizados pelas empresas, porém as empresas acharam convenientes e pretendem utilizar no futuro mais 60% das características referente á metodologia do *Scrum*. Fazendo com que o *Scrum* seja uma metodologia com tendência de ser bem utilizada no futuro pelas microempresas da cidade de Medianeira.

4.3.1 Pontos em comum do Scrum nas microempresas

Nesta seção serão exibidas quais práticas ágeis são comuns a todas as 4 empresas que utilizaram das tecnologias do *Scrum*, mostrando em que pontos que todas as empresas concordam com o mesmo objetivo, conforme a TABELA 2.

TABELA 2 - Pontos em comum do Scrum

Empresas	**A**	**B**	**C**	**D**
1) Frequentemente a equipe faz análise dos riscos envolvidos no desenvolvimento do *software*?	Gostaria	Gostaria	Gostaria	Gostaria
2) Existe uma pessoa da equipe responsável por remover os obstáculos para que o desenvolvimento possa fluir de maneira tranquila conforme o previsto?	Gostaria	Gostaria	Gostaria	Gostaria
3) Quando divididas as tarefas, as mesmas são sempre analisadas de forma que as de maior prioridade devem ser desenvolvidas primeiro?	Sim	Sim	Sim	Sim

Pode ser verificado na TABELA 2 que todas as empresas não estão fazendo análise de riscos, porém todas gostariam de fazer, assim como todas as empresas gostariam de ter uma pessoa que ficasse removendo os obstáculos para que os desenvolvedores não precisem perder tempo desenvolvendo funções que não são de suas responsabilidades. Também pode ser verificado que todas as empresas dividem suas tarefas em pequenas partes, e assim priorizam as tarefas que tem mais urgência para ser repassado ao cliente.

5 CONSIDERAÇÕES FINAIS

Este trabalho fez uma análise das metodologias ágeis que são utilizadas pelas microempresas de medianeira para o desenvolvimento dos projetos de *software*. O trabalho propôs a adoção das práticas e valores do *Scrum* e do *XP* como uma alternativa que poderia ser utilizados para o desenvolvimento dos projetos de *software*.

Através da revisão de literatura, procurou-se estabelecer bases teóricas e significativas para a adoção de cada uma das práticas e valores de cada metodologia (*Scrum* e *XP*). Foi mostrado que embora tais práticas não sejam de amplo conhecimento por parte das empresas, existem razões sólidas pelas quais as empresas utilizam dos conceitos dessas metodologias.

O trabalho apresentou uma forma de demonstrar como podem ser utilizadas características ágeis sem mesmo ter definido alguma das metodologias para o uso. Onde na maioria dos casos para que sejam atendidas às necessidades da empresa, são utilizadas mais do que uma metodologia, e mesmo assim se consegue ter ótimos resultados.

Em uma visão geral pode-se perceber que as características do *XP* estão mais presente no cotidiano das empresas, mesmo sem que algumas empresas tenham um conhecimento mais aprofundado sobre metodologias ágeis, enquanto o *Scrum* é apenas uma metodologia que é pouco utilizada nas empresas, porém a maioria das empresas pretende utilizá-la no futuro.

Nos pontos em comum pode ser verificado que todas as empresas focam em alguns objetivos específicos como, fazer entregas frequentes para que o cliente possa ir utilizando o sistema desde o inicio, não precisando deixar para utilizar o sistema como um todo apenas no final do seu desenvolvimento. Todas as empresas utilizam um padrão para o desenvolvimento de seus projetos, mesmo ainda sendo microempresas já buscam uma maneira de padronizar os seus processos. Todas as empresas dividem as tarefas em pequenas partes, para que possam desenvolver primeiro as tarefas que tenham uma prioridade maior. Além de que todas gostariam de desenvolver analises de riscos, assim como cada empresa gostaria de ter uma pessoa para remover os obstáculos para os desenvolvedores.

5.1 TRABALHOS FUTUROS/CONTINUAÇÃO DO TRABALHO

Aplicar o mesmo questionário sobre metodologias ágeis em empresas maiores, onde se tem um numero maior de colaboradores, podendo assim verificar se essas empresas trabalham com os mesmo princípios das microempresas.

6 REFERÊNCIAS

ANACLETO, ALESSANDRA; WANGENHEIM, CHRISTIANE. Aplicando mensuração em microempresas de *software* para suporte da gerência de projetos. **Anais do SBQS**, 2002.

BECK, K. et al. Manifesto para Desenvolvimento Ágil de *Software*. **http:** //www.agilemanifesto.org, 2001. Disponivel em: <http://www.agilemanifesto.org/iso/ptbr/principles.html>. Acesso em: 21 Novembro 2011.

CHAPMAN, R. Softletter. **http:** //www.softletter.com, 2010. Disponivel em: <http://www.softletter.com/Resources/RicksBlogonSaaS/tabid/99/PostID/13/Default.aspx>. Acesso em: 23 Novembro 2011.

DANIEL, T. et al. paginas.fe.up.pt. **FEUP Faculdade de Engenharia da Universidade do Porto**, Novembro 2005. Disponivel em: <http://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/es_final_14.pdf>. Acesso em: 28 Agosto 2011.

DONATO, R. Caverna do *Software*. **Caverna do *Software***, 2011. Disponivel em: <http://voat.com.br/rdal>. Acesso em: 10 Novembro 2011.

FAGUNDES; DETERS. COMPARAÇÃO ENTRE OS PROCESSOS DOS MÉTODOS ÁGEIS. **Senai**, p. 37-46, 2010.

FERNANDES, P. PHPaulo. **PHPaulo**, 2010. Disponivel em: <http://www.phpaulo.com.br/metodologia/extreme-programming-xp-em-2-minutos/>. Acesso em: 11 Novembro 2011.

FREDDO, CASAR; PARAISO, EMERSON; CAMPAGNOLO, MILTON P. Uma Arquitetura Multiagente para o Suporte ao Desenvolvimento Colaborativo de *Software* em Pequenas Equipes. **Simpósio Brasileiro de Sistemas Colaborativos**, p. 178 - 183, 2009.

GUERRA, A. C. **IMPLEMENTAÇÃO DE BOAS PRÁTICAS PARA O DESENVOLVIMENTO DE *SOFTWARE* EM MICRO E PEQUENAS EMPRESAS BRASILEIRAS**. Centro de Tecnologia da Informação Renato Archer - CTI. Campinas - SP. 2010.

LEITE, D. G. M. M. **Análise e Gestão de Riscos nas Micro e Pequenas Empresas de *Software*s**. UNIVERSIDADE METODISTA DE SÃO PAULO. São Bernardo do Campo, p. 18-21. 2006.

LEITE, J. C. engenhariade*software*.blogspot.com. **Engenharia de *Software***, 02 Março 2007. Disponivel em: <http://engenhariade*software*.blogspot.com/2007/03/o-modelo-espiral.html>. Acesso em: 01 Setembo 2011.

LUDVIG, D.; REINERT, J. D. **Estudo do uso de Metodologias Ágeis no Desenvolvimento de uma Aplicação de Governo Eletrônico**. Universidade Federal de Santa Catarina. Florianópolis. 2007.

MACHADO, F. P. **SISTEMAS WEB - Análise e desenvolvimento de sistema para promoção do comércio**. Faculdades Integradas de Cuiabá. Cuiabá, p. 11. 2011.

MACORATTI. macoratti.net/Default.aspx. **Macoratti.net**, 06 Janeiro 2005. Disponivel em: <http://www.macoratti.net/proc_sw1.htm>. Acesso em: 25 Agosto 2011.

NASCIMENTO, A. Um pouco sobre desenvolvimento de *software*, arquitetura, gerenciamento e minhas experiências. **Um pouco sobre desenvolvimento de *software*, arquitetura, gerenciamento e minhas experiências**, 2010. Disponivel em: <http://blog.anascimento.net/2010/02/>. Acesso em: 10 Novembro 2011.

OLIVEIRA, F. C. D.; PAULA, L. L. D. **Engenharia de *Software* baseada em componentes: uma abordagem prática em ambientes Web.** Universidade de Brasília. Brasilia, p. 13. 2009.

OLIVEIRA, R. B. D. **AUTOMAÇÃO DE FORÇA DE VENDAS VIA POCKET PC'S**. UTFPR. Cornélio Procópio, p. 17. 2008.

REIS, D. F. Devmedia. **http:** //www.devmedia.com.br, 2008. Disponivel em: <http://www.devmedia.com.br/post-10596-Conceitos-basicos-sobre-Metodologias-Ageis-para-Desenvolvimento-de-*Software*--Metodologias-Classicas-x-Extreme-Programming.html>. Acesso em: 22 Novembro 2011.

REIS, F. D. devmedia.com.br. **Devmedia**, 10 Outubro 2008. Disponivel em: <http://www.devmedia.com.br/articles/viewcomp_forprint.asp?comp=10596>. Acesso em: 28 Agosto 2011.

SABBAGH, R. scrumemacao.com.br. **Scrum em Ação**, 2010. Disponivel em: <http://scrumemacao.com.br/web/index.php>. Acesso em: 12 Outubro 2011.

SANTOS, R. G. D.; LUZ, G. D. **Princípios do Manifesto Agil**. Universidade do estado de São Paulo. São Paulo. 2003.

SARTI, B. **Projeto de um sistema gerencial sob medida para a administracão de uma micro empresa**. Universidade Federal do Rio Grande do Sul. Porto Alegre, p. 19. 2010.

SITE, P. www.medianeira.pr.gov.br. **Município de Medianeira**, 2011. Disponivel em: <http://www.medianeira.pr.gov.br/index.php?pagina=dados>. Acesso em: 10 Outubro 2011.

SOARES, M. D. S. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de *Software***. UNIPAC- Universidade Presidente Antônio Carlos. Conselheiro Lafaiete. 2004.

SOARES, M. S. Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de *Software*. **Open Journal Systems - Revista Eletrônica**, 2004.

TORREÃO, P. G. **Gerenciamento de Projetos**. [S.l.]. 2006.

VARGAS, F. pensoti.com.br. **Penso TI**, 18 Março 2011. Disponivel em: <http://www.pensoti.com.br/2011/engenharia-de-*software*/modelo-de-processos-prototipagem/>. Acesso em: 01 Setembro 2011.

APÊNDICES

APÊNDICE A – Questionário aplicado para obtenção das informações
* Este questionário foi aplicado utilizando a tecnologia Google Docs.

1) O seu cliente esta presente o tempo todo acompanhando o desenvolvimento do *Software*?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

2) Sua empresa faz entregas frequentes (módulos), entregando no menor tempo possível às funcionalidades do sistema?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

3) As funcionalidades do sistema são compartilhadas com todas as pessoas que estarão envolvidas para o desenvolvimento do mesmo?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

4) É feito um planejamento determinando o escopo da próxima versão do sistema?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

5) O desenvolvimento é feito de forma simples? Entregando apenas o que realmente o cliente precisa?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

6) A programação é feita em par onde duas pessoas estão pensando no mesmo problema?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

7) O Desenvolvimento é guiado em testes? Onde é escrito os testes antes da codificação?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

8) O código esta sempre sendo melhorado durante o desenvolvimento?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

9) Qualquer integrante da equipe pode alterar qualquer código a qualquer momento?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

10) Existe uma integração continua onde cada etapa do sistema desenvolvido é entregue ao cliente?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

11) As atividades repassados para o funcionário são feitas de maneira que exista um tempo hábil para realização das mesmas?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

12) Existem padrões de codificação? Onde todos os programadores escreverão código respeitando as regras?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

13) Os clientes se tornam parte da equipe de desenvolvimento, estando sempre presente para que seja tiradas as duvidas referente ao sistema em tempo integral?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

14) Todas as entregas feitas aos clientes (módulos), são feitas com suas funcionalidades 100% desenvolvidas?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

15) Frequentemente a equipe faz análise dos riscos envolvidos no desenvolvimento do *software*?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

16) São feitas reuniões rápidas e diárias com toda a equipe, verificando os seguintes fatores: O que fiz desde ontem? O que planejo fazer até amanha? Existe algo que possa impedir de eu atingir a minha meta?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

17) Existe transparência no planejamento e no desenvolvimento?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

18) Existe uma pessoa da equipe responsável por remover os obstáculos para que o desenvolvimento possa fluir de maneira tranquila conforme o previsto?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

19) O ambiente de trabalho e tranquilo fazendo com que as pessoas consigam desenvolver melhor suas atividades?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

20) Quando divididas as tarefas. As mesmas são sempre analisadas de forma que as de maior prioridade devem ser desenvolvidas primeiro?

SIM ( ) NÃO ( ) GOSTARIA ( ) INVIÁVEL ( )

Empresa:_____

Nome do participante:_____

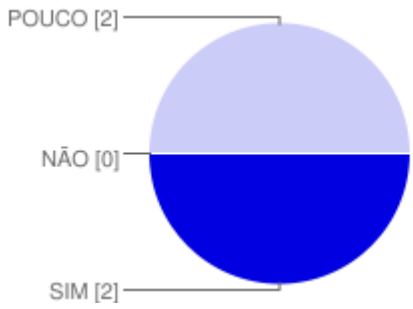
Quantidade de Funcionários:_____

Você tem algum conhecimento sobre metodologias ágeis?

SIM ( ) NÃO ( )

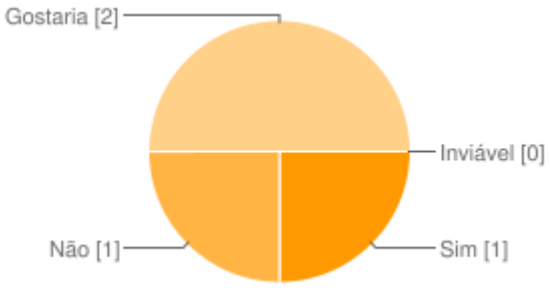
APÊNDICE B – Resultado completo do questionário aplicado para obtenção das informações

1. Você tem algum conhecimento sobre metodologias ágeis?



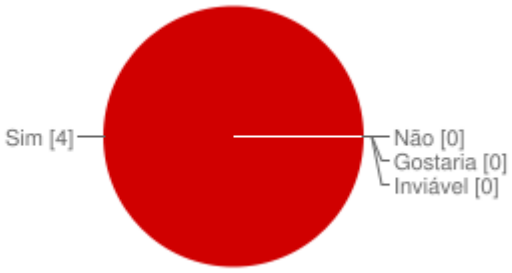
SIM	2	50%
NÃO	0	0%
POUCO	2	50%

2. O seu cliente esta presente o tempo todo acompanhando o desenvolvimento do *Software*?



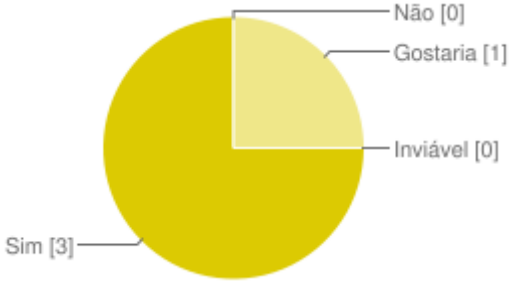
Sim	1	25%
Não	1	25%
Gostaria	2	50%
Inviável	0	0%

3. Sua empresa faz entregas frequentes (módulos), entregando no menor tempo possível às funcionalidades do sistema?



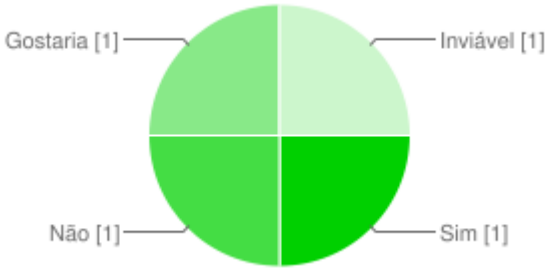
Sim	4	100%
Não	0	0%
Gostaria	0	0%
Inviável	0	0%

4. As funcionalidades do sistema são compartilhadas com todas as pessoas que estarão envolvidas para o desenvolvimento do mesmo?



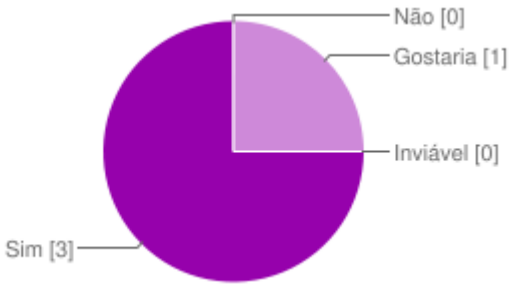
Sim	3	75%
Não	0	0%
Gostaria	1	25%
Inviável	0	0%

5. É feito um planejamento determinando o escopo da próxima versão do sistema?



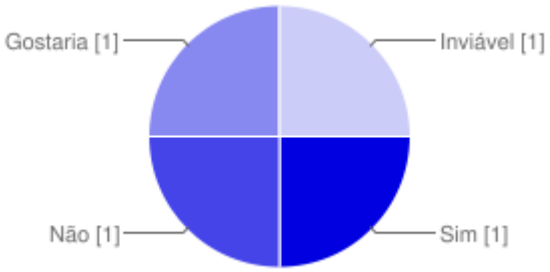
Sim	1	25%
Não	1	25%
Gostaria	1	25%
Inviável	1	25%

6. O desenvolvimento é feito de forma simples? Entregando apenas o que realmente o cliente precisa?



Sim	3	75%
Não	0	0%
Gostaria	1	25%
Inviável	0	0%

7. A programação é feita em par onde duas pessoas estão pensando no mesmo problema?

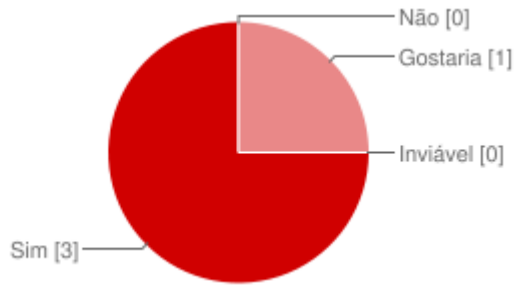


Sim	1	25%
Não	1	25%
Gostaria	1	25%
Inviável	1	25%

8. O Desenvolvimento é guiado em testes? Onde é escrito os testes antes da codificação?

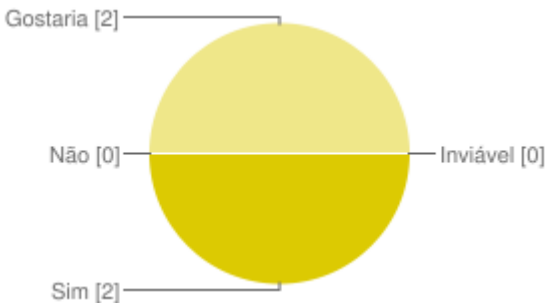
Sim	1	25%
Não	1	25%
Gostaria	1	25%
Inviável	1	25%

9. O código esta sempre sendo melhorado durante o desenvolvimento?



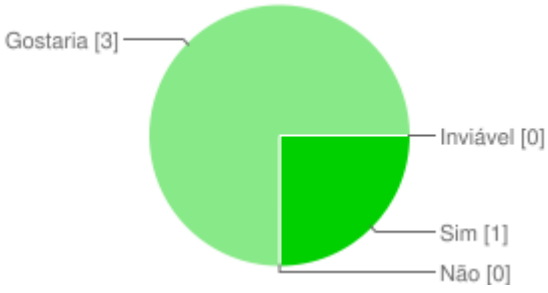
Sim	3	75%
Não	0	0%
Gostaria	1	25%
Inviável	0	0%

10. Qualquer integrante da equipe pode alterar qualquer código a qualquer momento?



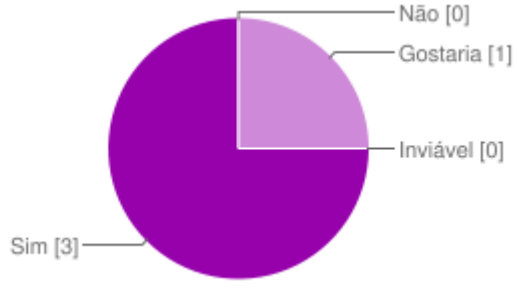
Sim	2	50%
Não	0	0%
Gostaria	2	50%
Inviável	0	0%

11. Existe uma integração continua onde cada etapa do sistema desenvolvido é entregue ao cliente?



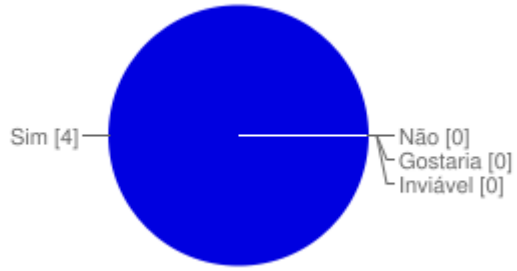
Sim	1	25%
Não	0	0%
Gostaria	3	75%
Inviável	0	0%

12. As atividades repassados para o funcionário são feitas de maneira que exista um tempo hábil para realização das mesmas?



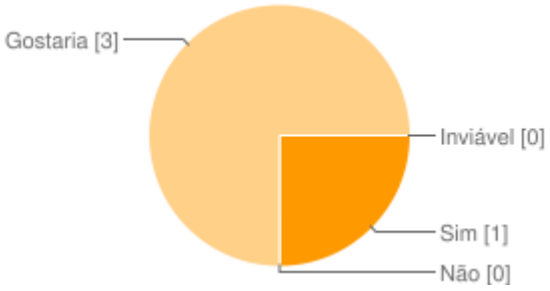
Sim	3	75%
Não	0	0%
Gostaria	1	25%
Inviável	0	0%

13. Existem padrões de codificação? Onde todos os programadores escreverão código respeitando as regras?



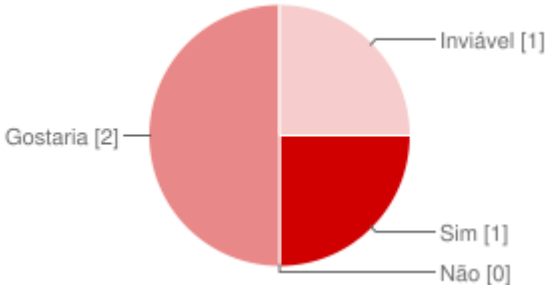
Sim	4	100%
Não	0	0%
Gostaria	0	0%
Inviável	0	0%

14. Os clientes se tornam parte da equipe de desenvolvimento, estando sempre presente para que seja tiradas as duvidas referente ao sistema em tempo integral?



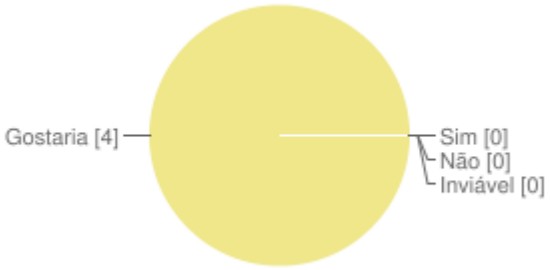
Sim	1	25%
Não	0	0%
Gostaria	3	75%
Inviável	0	0%

15. Todas as entregas feitas aos clientes (módulos), são feitas com suas funcionalidades 100% desenvolvidas?



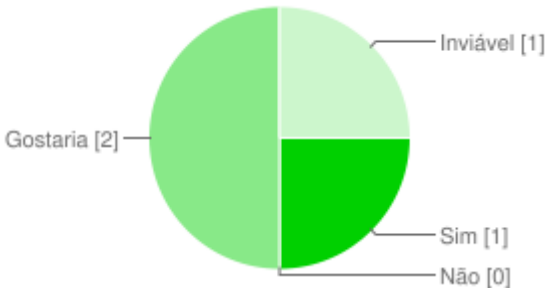
Sim	1	25%
Não	0	0%
Gostaria	2	50%
Inviável	1	25%

16. Frequentemente a equipe faz análise dos riscos envolvidos no desenvolvimento do *software*?



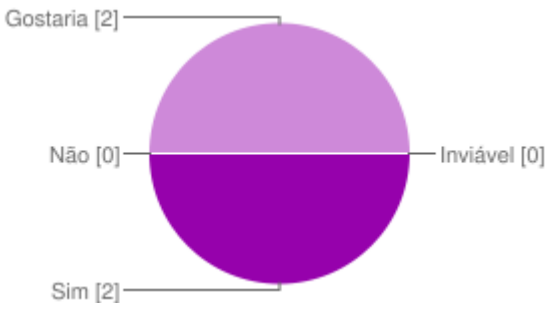
Sim	0	0%
Não	0	0%
Gostaria	4	100%
Inviável	0	0%

17. São feitas reuniões rápidas e diárias com toda a equipe?



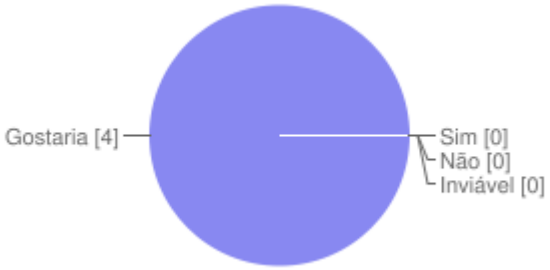
Sim	1	25%
Não	0	0%
Gostaria	2	50%
Inviável	1	25%

18. Existe transparência no planejamento e no desenvolvimento?



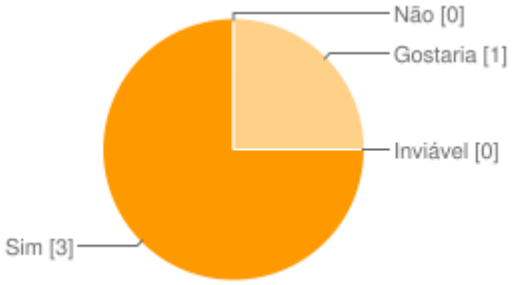
Sim	2	50%
Não	0	0%
Gostaria	2	50%
Inviável	0	0%

19. Existe uma pessoa da equipe responsável por remover os obstáculos para que o desenvolvimento possa fluir de maneira tranqüila conforme o previsto?



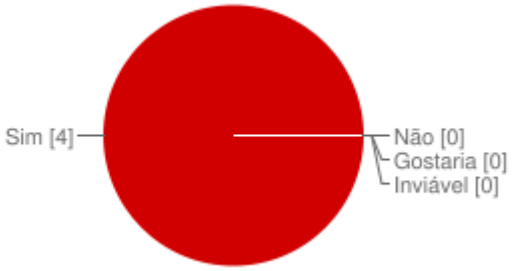
Sim	0	0%
Não	0	0%
Gostaria	4	100%
Inviável	0	0%

20. O ambiente de trabalho e tranquilo fazendo com que as pessoas consigam desenvolver melhor suas atividades?



Sim	3	75%
Não	0	0%
Gostaria	1	25%
Inviável	0	0%

21. Quando divididas as tarefas. As mesmas são sempre analisadas de forma que as de maior prioridade devem ser desenvolvidas primeiro?



Sim	4	100%
Não	0	0%
Gostaria	0	0%
Inviável	0	0%