UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA – UESB
COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO

ALAN MOITINHO SANTIAGO

DESENVOLVIMENTO DE SOFTWARE: APLICABILIDADE DE METODOLOGIAS ÁGEIS NO MERCADO DE SOFTWARE EM VITÓRIA DA CONQUISTA - BA

Vitória da Conquista – BA,
Setembro/2012

ALAN MOITINHO SANTIAGO

DESENVOLVIMENTO DE SOFTWARE: APLICABILIDADE DE METODOLOGIAS ÁGEIS NO MERCADO DE SOFTWARE EM VITÓRIA DA CONQUISTA - BA

Monografia apresentada à Universidade Estadual do Sudoeste da Bahia – UESB/Campus de Vitória da Conquista, como requisito parcial para conclusão do Curso de Ciência da Computação.

Orientador: Prof. Esp. Gidevaldo Novais dos Santos.

Vitória da Conquista – BA,
Setembro/2012

UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA (UESB)

CURSO DE CIÊNCIA DA COMPUTAÇÃO

DECLARAÇÃO DE APROVAÇÃO

Título: DESENVOLVIMENTO DE SOFTWARE: APLICABILIDADE DE METODOLOGIAS ÁGEIS NO MERCADO DE SOFTWARE EM VITÓRIA DA CONQUISTA - BA

Autor: Alan Moitinho Santiago

_____
Prof. Esp. Gidevaldo Novais dos Santos. – UESB
Orientador

_____
Prof. Msc. Adilson de Lila Pereira – UESB
Parecerista

_____
Prof. Esp. Clésio Rubens de Matos - FTC.
Parecerista

Data de realização: 14/09/2012

AGRADECIMENTOS

Agradeço a Deus por ter permitido a realização desse sonho. Aos meus pais Dilson e Marilene e meus avós Jorim e Maria do Carmo pela educação, carinho e atenção que me deram sempre ensinando a ser essa pessoa que hoje sou. A minha irmã Laiane e minha tia Silvana pela companhia e apoio dado. A minha namorada Thaylana e aos meus tios Jorim, Maysa, Patrícia e Arquimedes que nessa reta final estiveram ao meu lado me incentivando e ajudando a superar as dificuldades. Aos colegas de curso Juliano Castro, Anderson Marques, Douglésia Thiália e Mariana Rocha que juntos vencemos uma longa caminhada. Ao meu orientador Gidevaldo Novais que me instruiu com todo o seu conhecimento e entendimento. A Celina uma pessoa especial que através do seu competente trabalho auxiliou com muita atenção e carinho. As empresas que contribuíram para a realização da pesquisa e término do trabalho de conclusão. Enfim a todos que de alguma forma contribuíram para que esse objetivo antes sonho se tornasse hoje realidade, o meu muito obrigado.

Esta monografia é dedicada com todo o meu carinho e respeito a toda a minha família.

“Nós somos o que fazemos repetidamente, a excelência não é um feito, e sim, um hábito”

Aristóteles

RESUMO

O uso de metodologias ágeis para o desenvolvimento de software tem se mostrado mais interessante e oferecido diversas vantagens. Partindo desses pressupostos, o presente estudo tem como objetivo geral: analisar como as empresas do mercado conquistense fazem uso de metodologias ágeis para o desenvolvimento de software. E consequentemente identificar o perfil do quadro de pessoal das empresas conquistenses que fazem uso de metodologias ágeis; verificar quais os programas de software utilizados pelas empresas estudadas; identificar a percepção dos gestores, os principais pontos fortes e fracos do processo de desenvolvimento de software baseado em metodologias ágeis. A metodologia utilizada para sua realização é considerada descritiva e exploratória de caráter quantitativa. Foram utilizados como instrumento de coleta de dados um questionário aplicado junto as empresas de Vitória da Conquista – BA. Os resultados apontam que as empresas em sua maioria fazem uso de metodologias ágeis para o desenvolvimento de software em geral são utilizados o são o *Scrum* e o XP, com representação significativa. Com relação a percepção dos gestores com os pontos fortes e fracos no processo de desenvolvimento de software baseado em metodologias ágeis; se constatou que são mais eficientes devido a agilidade e a flexibilidade das mesmas. Concluindo-se assim, que a mesma tem se mostrado eficiente e promissora para o fim a que se destina.

Palavras chave: Aplicabilidade. Desenvolvimento. Metodologias ágeis. Software.

ABSTRACT

The use of agile methodologies to software development has proven to be more interesting and offer several advantages. Based on these assumptions, the present study has the general objective: to analyze how companies market in Vitória da Conquista make use of agile methods for software development. And consequently identify the profile of staff companies of this city that use agile methodologies; verify which software programs are used by companies, to identify the perceptions of managers, the main strengths and weaknesses of the process of software development based on agile. The methodology used for its realization is considered exploratory and descriptive quantitative character. The instrument used for data collection was a questionnaire addressed to the companies in Vitória da Conquista - BA. The results show that firms mostly make use of agile methodologies to software development in general, they use the Scrum and XP, with significant representation. Regarding the perception of managers with the strengths and weaknesses in the process of software development based on agile methodologies; been found to be more efficient due to the same agility and flexibility. It was concluded therefore that the same has proven effective and promising for the intended purpose.

Keywords: Applicability. Development. Agile methodologies. Software.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE SIGLAS

ASD - *Adaptive Software Development*

DSDM - Systems Development Method

FDD - Feature Driven Development

XP- *Extreme Programming*

VV&T Verificação, Validação e Teste

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, pode-se perceber a evolução do processo de desenvolvimento de *software* mundial. É de suma importância que um país como o Brasil acompanhe as mudanças constantes que ocorrem nesse processo. Muitas empresas ainda não utilizam métodos no processo de desenvolvimento. Com a evolução do mercado de *software*, é necessário que as empresas do mercado brasileiro também evoluam utilizando-se de ferramentas e métodos modernos na prática da engenharia de software.

Com o aumento em grande escala da produção de *software* no Brasil as metodologias ágeis surgem como uma alternativa para o processo de desenvolvimento tradicional. As metodologias tradicionais valorizam a importância de projetar antes de construir, tendo como resultado muita produção de documentos, sendo ainda muito utilizada. Em contrapartida têm-se as metodologias ágeis que são caracterizadas pelo desenvolvimento incremental e iterativo, pela valorização da contribuição do cliente no processo, e pelas constantes mudanças no código que são apresentadas periodicamente ao cliente.

As metodologias ágeis começaram a ser praticadas a partir de 2001 quando profissionais da área de desenvolvimento de *software* resolveram debater o processo de melhorar a qualidade do software. A partir daí deu-se então o início ao “Manifesto Ágil” passando a valorizar princípios aos quais tivessem a maior interação do cliente com o processo de desenvolvimento do *software*.

As primeiras metodologias ágeis a serem praticadas foram a *Extreme Programming* conhecida como XP, e a *Scrum*. Foram essas o ponto de partida para a criação do Manifesto Ágil. Logo após vieram outras que são bastante conhecidas e utilizadas: *Feature Driven Development* (FDD) – Desenvolvimento Guiado por Funcionalidades,*Crystal*, *Dynamic Systems Development Method* (DSDM) – Método Dinâmico de Desenvolvimento de Sistemas, *Adaptive Software Development* (ASD) – Desenvolvimento Adaptável de Software estão entre algumas utilizadas no mercado nos dias de hoje.

Atualmente, percebe-se que os projetos de desenvolvimento de *software* dificilmente são entregues nos prazos corretos e de forma desejada pelo cliente. Por isso, é necessário analisar a utilização de metodologias ágeis no processo de

desenvolvimento do *software* visando sempre à qualidade daquilo que o cliente busca.

A procura por desenvolvimento de software tem aumentado significativamente. É importante a prática da engenharia de software para que seja obtido um resultado positivo tanto para o cliente quanto para a equipe de desenvolvimento ao término do projeto, como ganho de produtividade e ganho de custo de manutenção.

O uso das metodologias ágeis tem aumentado nos últimos anos por criar alternativas para o modelo de desenvolvimento tradicional com relação à melhoria na qualidade do software. As grandes empresas têm utilizado tal metodologia cada vez mais, para que projetos obtenham sucesso e sejam entregues com prazos e custos desejados.

Com base neste contexto, o projeto em questão servirá de base para ampliar as discussões sobre o uso efetivo de metodologias ágeis no processo de desenvolvimento de *software* na cidade de Vitória da Conquista, bem como mostrar a comunidade acadêmica o quanto é importante o uso de tais metodologias.

O tema abordado nesse trabalho está relacionado ao uso de metodologias ágeis no desenvolvimento de software por empresas em Vitória da Conquista – BA.

Gil (1991) defende que toda pesquisa é iniciada a partir de um problema. Para o pesquisador, formulá-lo não constitui tarefa fácil, entretanto, para que seja solucionado ele deve ser apresentado de forma clara e precisa.

De acordo com Luna (2009), a formulação do problema de pesquisa necessita ser claro e é considerada uma etapa fundamental no processo de pesquisa. “As decisões a serem tomadas [...] dependerão da formulação do problema e, portanto serão mais adequadas quanto maior for a clareza em relação a ele” (LUNA, 2009, p.28).

Segundo Deutech, o desenvolvimento de *software* inclui muitas atividades em que as possibilidades de inserção de falhas humanas são enormes (1979, apud PRESSMAN, 1995, p. 786). Diversas técnicas podem ser empregadas para reduzir o número de falhas nos produtos e aumentar a qualidade dos mesmos. Dentre as atividades de VV&T (Verificação, Validação e Teste), que devem acompanhar todo o ciclo de desenvolvimento de *software*, os testes são as mais empregadas.

Muitas vezes, os projetistas convivem com atrasos no projeto devido a incoerências internas do sistema, que poderiam ser diminuídas enormemente casos

testes fossem adotados mais cedo no desenvolvimento. Os testes evitam que erros passem despercebidos e apareçam tardiamente. Existem diversos tipos detestes que podem ser utilizados durante o processo de desenvolvimento de *software* com o objetivo de encontrar erros, e consequentemente reduzir a quantidade de defeitos existentes.

A grande relevância econômica da pesquisa é apresentada na medida em que esta propõe redução de custos com manutenção, através da sugestão de implantação da atividade de testes sistemática nas empresas de desenvolvimento de *software* na cidade, ou propor melhorias ao processo de desenvolvimento de *software*,contribuindo para existir uma gestão formal da atividade de testes nas empresas. Este trabalho também serve de base para análise de profissionais que desejam compreender melhor teste de *software* e sua aplicabilidade.

Sabendo da importância de uma boa formulação da questão problema e o quanto ao uso de metodologias ágeis para o desenvolvimento de software por empresas em Vitória da Conquista - BA. O presente trabalho almeja responder o seguinte problema: como as empresas do mercado conquistense fazem uso de metodologias ágeis para o desenvolvimento de software?

Este estudo tem como objetivo geral: analisar como as empresas do mercado conquistense fazem uso de metodologias ágeis para o desenvolvimento de software. E consequentemente identificar o perfil do quadro de pessoal das empresas conquistenses que fazem uso de metodologias ágeis; verificar quais os programas de software utilizados pelas empresas estudadas; identificar a percepção dos gestores, os principais pontos fortes e fracos do processo de desenvolvimento de software baseado em metodologias ágeis;

O presente trabalho conta com cinco capítulos, sendo que o primeiro capítulo faz uma apresentação geral do estudo, relatando seus objetivos, problema de pesquisa e justificativa. O capítulo dois apresenta o referencial teórico onde são abordadas as metodologias ágeis e sua relevância para o desenvolvimento de software. O terceiro capítulo descreve a metodologia para a sua realização, o quarto capítulo mostra os resultados dos dados coletados e suas respectivas análises. Por fim, o quinto e último capítulo apresenta as considerações finais da pesquisa, as limitações do estudo e as sugestões de melhorias para um melhor desempenho de serviços e ações de segurança e higiene no trabalho.

2 ENGENHARIA DE SOFTWARE

Neste capítulo serão apresentadas informações de grande relevância relacionadas a metodologias ágeis para o desenvolvimento de software. Inicialmente será abordado de forma breve a respeito da engenharia de *software*. Logo após será abordado os conceitos relacionados às metodologias tradicionais. E finalizando, serão abordadas informações das metodologias ágeis.

Segundo Pressman (2006 p. 17-18) a engenharia de *software* é uma tecnologia em camadas. A base em que se apóia é o *foco na qualidade*. O alicerce é a camada de *processo*, pois mantém unidas as camadas de tecnologia e permite o desenvolvimento racional e oportuno de *softwares* de computador. Os *métodos* fornecem a técnica de “como fazer” para construir. E complementando o topo, as *ferramentas*, que fornecem apoio automatizado ou semiautomatizado para o processo e para os métodos.

A engenharia de software segue os mesmo passos de qualquer engenharia: análise do ambiente que será trabalhado, planejamento daquilo que será necessário, projeto do que foi observado, construção do que foi projetado e validação daquilo que lhe foi solicitado.

Para (MARTINS, 2007, p.5) independente da abordagem utilizada para gerenciar um projeto de desenvolvimento de *software*, todo projeto passa pelas seguintes etapas: análise do negócio, análise dos requisitos, projeto técnico, construção e validação. Algumas metodologias incorporam processos formais para tratar destes aspectos, enquanto outras os abordam de forma menos formal, deixando que a equipe de projeto escolha a melhor maneira para lidar com estes assuntos.

2.1 METODOLOGIAS TRADICIONAIS DE DESENVOLVIMENTO DE SOFTWARE

As metodologias tradicionais são aquelas em que existem processos definidos. É indicada para utilização quando se conhece previamente o minimundo que será trabalhado e se tem processos imutáveis. Cada fase é especificada, e somente depois disso inicia-se o seu desenvolvimento. É necessário que todas as fases sejam executadas de forma seqüencial, para que assim não comprometa o resultado final do projeto.

Estas metodologias surgiram em um cenário de construção de software muito diferente do atual. A seqüência de tarefas para desenvolvimento do software era realizada em terminais burros e mainframes, onde o custo para correção de erros ou qualquer outra alteração era muito elevado, pois o acesso aos computadores era muito limitado e não havia ferramentas de apoio à construção de software (PRESSMAN, 2002).

Então, a maneira de minimizar os problemas durante o desenvolvimento de um *software* foi planejar e documentar muito bem este antes do início de seu desenvolvimento. Esse foi um dos fatores primordiais para a existência de um processo de gerenciamento de software baseado em documentação detalhada, planejamento extenso e etapas bem delimitadas (PRESSMAN, 2002).

Nas metodologias tradicionais, ou pesadas, são assim chamadas devido ao seu tamanho e à dificuldade de serem implementadas na íntegra (OLIVEIRA, 2004). Historicamente, elas predominaram e ainda são muito utilizadas nos projetos de desenvolvimento de software (OLIVEIRA, 2004). O que melhor caracteriza estas metodologias é a separação bem rígida das fases de projeto, que consistem em: Levantamento de Requisitos, Análise, Desenho, Implementação, Testes e Implantação. Cada fase tem suas especificidades e possuem entre si interdependência, isto é, a próxima fase só começa quando a anterior estiver pronta. Isto significa que o processo é sequencial e linear, no qual cada fase deve ser concluída antes de passar para a próxima etapa.

Estas metodologias seguem o modelo tradicional da engenharia como a Engenharia Civil ou Elétrica, onde o desenvolvimento do sistema é dividido em duas fases distintas: o sistema é primeiramente planejado e, depois do planejamento pronto, o sistema é construído. A fase de concepção do projeto ou planejamento é realizada por uma equipe de analistas que, mediante reuniões e discussões com os clientes ou usuários, estrutura como o sistema vai ser, quais serão suas funcionalidades e características. Nesta fase é necessário grande trabalho de documentação, pois este será o alicerce do projeto; com isso muito tempo é gasto para que haja um levantamento de requisitos completo e claro, a fim de evitar o aparecimento de novos requisitos ou funcionalidades durante a fase de desenvolvimento do projeto.

O planejamento é detalhadamente descrito, extenso por sua vez, e enfatiza a criação e o cumprimento de cronogramas de atividades e procedimentos que

auxiliem na construção do produto e na coordenação do processo. Este tipo de documento ainda é utilizado na mensuração do progresso durante a fase de execução do projeto, podendo sofrer constantes mudanças conforme a evolução do processo (VIEIRA, 2003).

Por este motivo é recomendável que os analistas de sistema tenham como característica de personalidade a facilidade de comunicação. “É indispensável ao trabalho dos analistas de sistemas uma boa aptidão para o diálogo, para a expressão, comunicação verbal, escuta e para a interação, respeitosa e profunda, com pessoas que vivem e trabalham no meio ambiente onde a fábrica de informação vai ser instalada” (CARVALHO, 1988, p. 143).

2.1.1 Modelo em cascata

O modelo clássico ou cascata, que também é conhecido por abordagem “top-down”, foi proposto por Royce em 1970. Esse modelo foi derivado de modelos de atividade de engenharia com a finalidade de estabelecer ordem no desenvolvimento de grandes produtos de software.

Comparado com outros modelos de desenvolvimento de software, este é mais rígido e menos administrativo. Este modelo considera as atividades mostradas anteriormente e as representa como fases separadas do processo, como especificações de requisitos, projetos de software, implementação, testes e assim por diante. Após cada estágio ter sido definido, ele é aprovado e o desenvolvimento prossegue para o estágio seguinte (SOMERVILLE,2004).

A idéia principal do modelo é que as diferentes etapas de desenvolvimento seguem uma sequencia, ou seja, a saída da primeira etapa "fluí" para a segunda etapa, a saída da segunda etapa "flui" para a terceira e assim por diante, como se pode analisar na Figura 1. As atividades a executar são agrupadas em tarefas, realizadas sequencialmente, de forma que uma tarefa só poderá ter início quando a anterior tiver terminado. Uma das vantagens do modelo é que só avança para a tarefa seguinte quando o cliente valida e aceita os produtos finais da tarefa atual (MACORATTI,2007).

Há diversos problemas associados a essa aplicação desvirtuada do modelo em cascata, todos provenientes de sua característica principal: a sequencialidade das fases na construção do sistema como um todo que pode levar a ter problemas

na comunicação entre as fases; não suporta as modificações nos requisitos; não prevê a manutenção; não permite a reutilização; é excessivamente sincronizado e não permite total visualização antecipada do software, fazendo com o que o cliente tenha que esperar pelo resultado final.

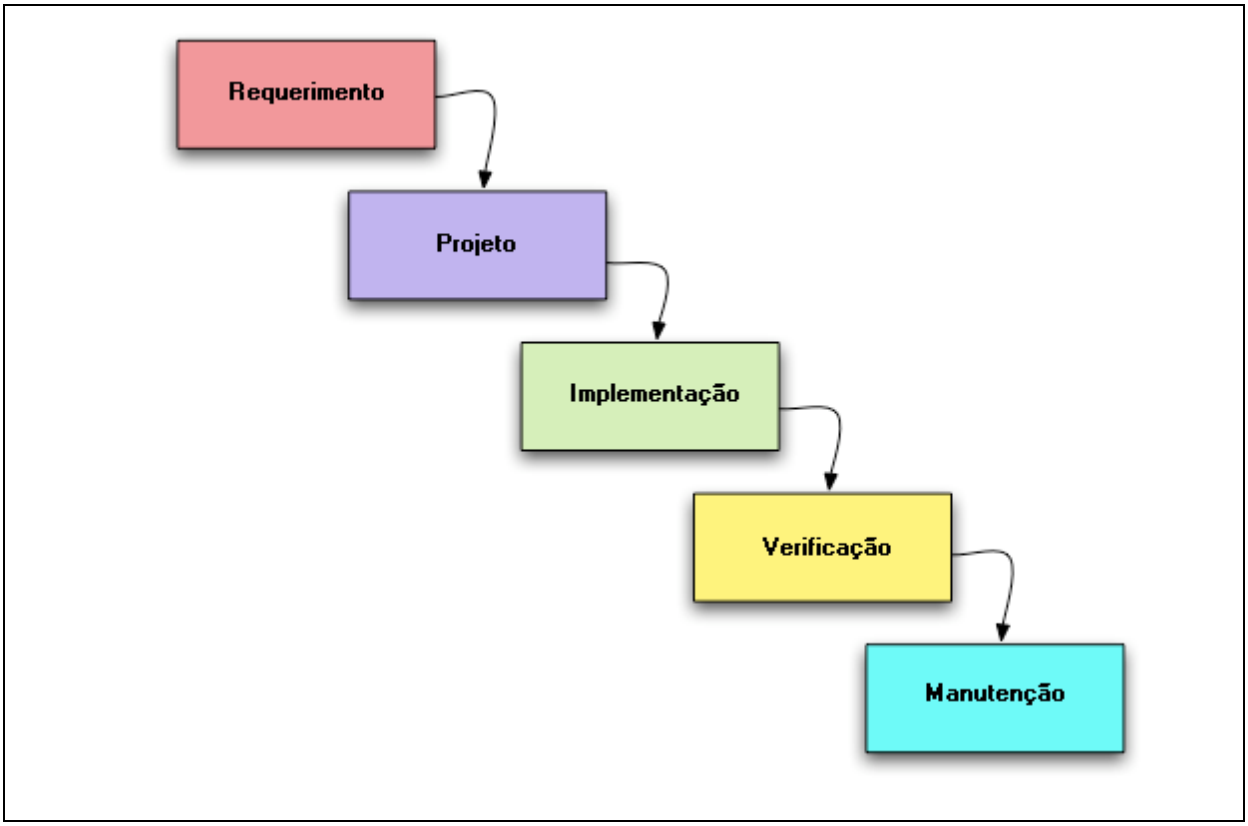

Figura 1 – Modelo Cascata
Fonte: WIKIPÉDIA 2001

O modelo cascata é um dos mais importantes modelos, e é referência para outros modelos, servindo de base para muitos projetos modernos. A versão original deste modelo foi melhorada e retocada ao longo do tempo, de modo que ele continua sendo muito utilizado hoje em dia. Grande parte do sucesso do modelo cascata está no fato dele tornar o processo de desenvolvimento, estruturado. No entanto, com o decorrer do tempo, a proposta inicial de Royce foi sendo deturpada, e a segunda característica (retro-alimentação) relegada ao desuso.

2.1.2 Modelo Incremental

É baseado em repetições de etapas, em que estas são apenas melhoradas e uma nova versão surge ao final de cada iteração, ou seja, ao invés de

disponibilizar o sistema como um todo, o desenvolvimento é dividido em incrementos, onde cada incremento entrega parte da funcionalidade requerida.

Este modelo é uma extensão do modelo espiral sendo, porém o desenvolvimento de um produto comercial de software é uma tarefa que pode ser estendida por vários meses, possivelmente um ano ou mais. Por isso, é mais prático dividir o trabalho em partes menores ou iterações. Cada iteração resultará num incremento, sendo que iterações são passos em fluxo de trabalho e incrementos é o crescimento do produto (MACORATTI,2007).

São vantagens:

• Possibilidade de avaliar mais cedo os riscos e pontos críticos do projeto, e identificar medidas para eliminá-los ou controlar;
• Redução dos riscos envolvendo custos a um único incremento. Se a equipa que desenvolve o software precisar repetir a iteração, a organização perde somente o esforço mal direcionado de uma iteração, não o valor de um produto inteiro;
• Definição de uma arquitetura que melhor possa orientar todo o desenvolvimento;
• Disponibilização naturalde um conjunto de regras para melhor controlar os inevitáveis pedidos de alterações futuras;
• Permite que os vários intervenientes possam trabalhar mais efetivamente pela interação e partilha de comunicação daí resultante; (MACORATTI,2007).

2.1.3 Modelo em espiral

O modelo em espiral foi proposto por Boehm em 1988 como formas de integrar os diversos modelos existentes à época. Baseado em uma sequência de fases que culminam em releases incrementais do software e eliminam suas dificuldades explorando seus pontos fortes.

Este modelo foi desenvolvido para abranger as melhores características tanto do ciclo de vida clássico, acrescentando, ao mesmo tempo, um novo elemento - a análise de riscos - que falta a esses paradigmas. Entretanto a integração não se da através da simples incorporação de características do modelo cascata. (PETERS, 2001).

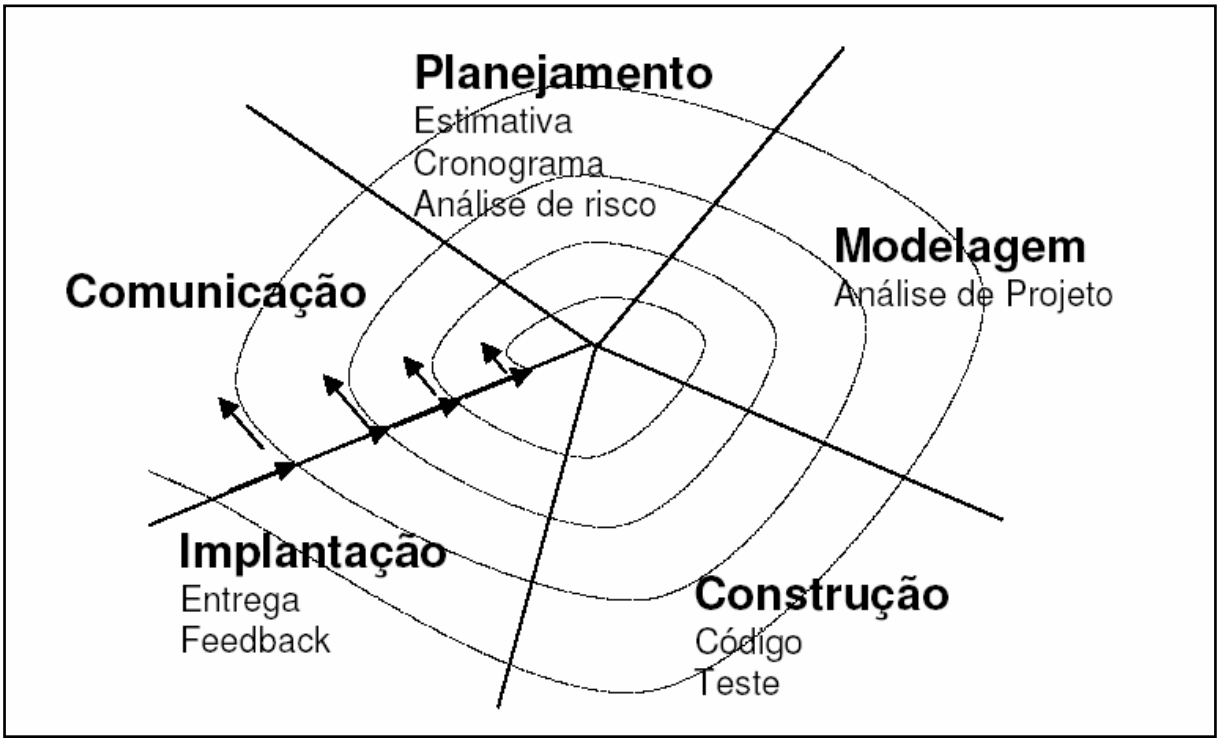

Figura 2: Modelo Espiral
Fonte: BEZERRA, 2002.

O modelo original em espiral organiza o desenvolvimento como um processo iterativo em que vários conjuntos de quatro fases se sucedem até se obter o sistema final. Um ciclo se inicia com a determinação de objetivos, alternativas e restrições (primeira tarefa) onde ocorre o comprometimento dos envolvidos e o estabelecimento de uma estratégia para alcançar os objetivos. Na segunda tarefa, análise e avaliação de alternativas, identificação e solução de riscos, executa-se uma análise de risco. (GUSTAVO,2007) Se o risco for considerado inaceitável, pode parar o projeto. Na terceira tarefa ocorre o desenvolvimento do produto. Neste quadrante pode-se considerar o modelo cascata. Na quarta tarefa o produto é avaliado e se prepara para iniciar um novo ciclo.

O Modelo Espiral, ainda assim, assume que existe alguma seqüência entre as fases: não há suporte para fases que ocorrem simultaneamente, ou que necessitam de intercomunicação contínua para operarem.

Vantagens deste modelo

• Modelo em espiral permite que ao longo de cada iteração se obtenham versões do sistema cada vez mais completas, recorrendo à prototipagem para reduzir os riscos.

• Este tipo de modelo permite a abordagem do refinamento seguido pelo modelo em cascata, mas que incorpora um enquadramento iterativo que reflete, de uma forma bastante realística, o processo de desenvolvimento.

Desvantagens

• Pode ser difícil convencer grandes clientes (particularmente em situações de contrato) de que a abordagem evolutiva é controlável.

• A abordagem deste tipo de modelo exige considerável experiência na avaliação dos riscos e baseia-se nessa experiência para o sucesso. Se um grande risco não for descoberto, poderão ocorrer problemas.

• Este tipo de modelo é relativamente novo e não tem sido amplamente usado.

• É importante levar em consideração que podem existir diferenças entre o protótipo e o sistema final. O protótipo pode não cumprir os requisitos de desempenho, pode ser incompleto, e pode refletir somente alguns aspectos do sistema a ser desenvolvido.

• O modelo em espiral pode levar ao desenvolvimento em paralelo de múltiplas partes do projeto, cada uma sendo abordada de modo diferenciado, por isso é necessário o uso de técnicas específicas para estimar e sincronizar cronogramas, bem como para determinar os indicadores de custo e progresso mais adequados (MACORATTI,2007).

2.1.4 Prototipação

A utilização deste modelo é recomendada quando há dificuldade de identificar os requisitos ou estes não estão claros. É feita uma especificação inicial dos requisitos, como se pode observar na Figura 3,dados de entrada e saída que o sistema deve ter, e a partir daí é feito um protótipo para o cliente ver se está de acordo com suas exigências.

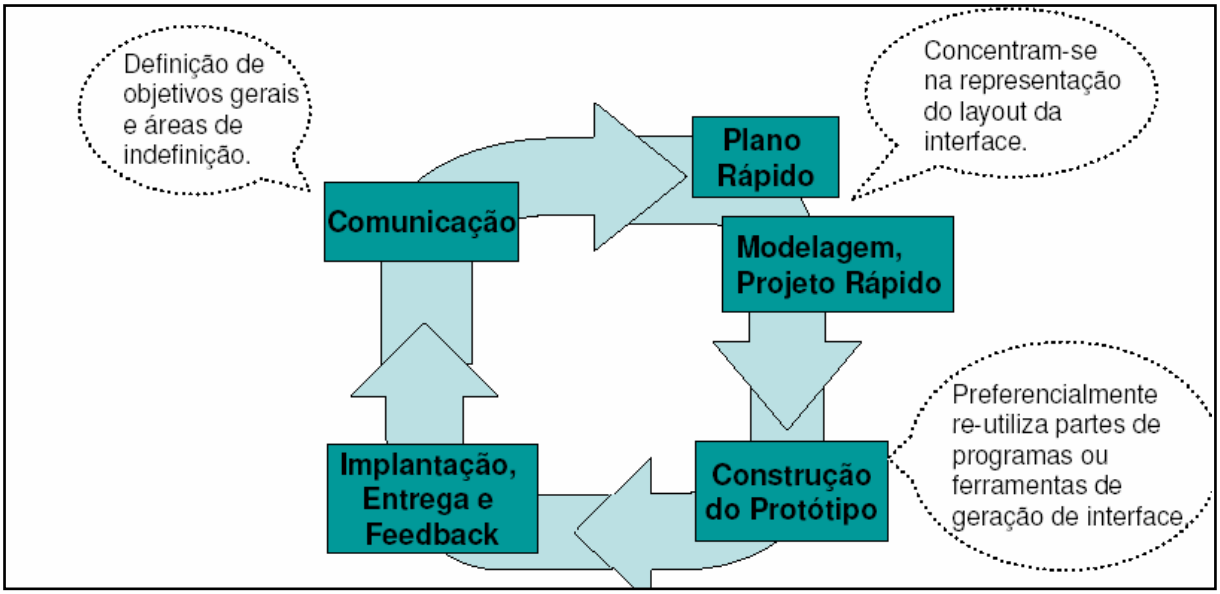

Figura 3: Modelo de prototipagem
Fonte: BEZERRA, 2002.

O modelo Prototipação objetiva a realização de diversas iterações em que um protótipo é desenvolvido a partir de um conjunto de requisitos, e em seguida é avaliado pelo cliente. Através dos resultados obtidos da avaliação, um novo protótipo, ou o sistema final, é construído (SOMMERVILLE,2004). Depois de o protótipo estar pronto o cliente o utiliza, e baseado na sua avaliação, repassa suas impressões do que precisa ser alterado, o que está faltando e o que não é necessário. O protótipo é então modificado incorporando as sugestões de mudança e o cliente usa o protótipo novamente repetindo o processo até que o mesmo seja válido em termos de custo e tempo. No final os requisitos iniciais são alterados para produzir a especificação final dos requisitos (MACORATTI,2007).

Este modelo pode trazer os seguintes benefícios:

• O modelo é interessante para alguns sistemas de grande porte nos quais representam certo grau de dificuldade para exprimir rigorosamente os requisitos;

• É possível obter uma visão do que será o sistema com um pequeno investimento inicial;

• A experiência de produzir o protótipo pode reduzir o custo das fases posteriores;

• A construção do protótipo pode demonstrar a viabilidade do sistema.

Questões a serem consideradas quanto à utilização do modelo:

• A Prototipação deve ser utilizada apenas quando os usuários podem participar ativamente no projeto;

• Não descuidar de uma boa análise que deve ser conduzida durante todo o processo de prototipação;

• Esclarecer aos usuários que o desempenho apresentado pelo protótipo não necessariamente será o mesmo do sistema final

• Evitar que o sistema final seja um protótipo em que foram implementados todos os requisitos especificados, pois se corre o risco de ter-se um sistema mal implementado, uma vez que as técnicas utilizadas para desenvolver um protótipo são diferentes daquelas utilizadas na implementação de um sistema (relaxamento de regras de negócio, manipulação de exceções etc);

2.2 TESTES DE SOFTWARE

Segundo Molinari (2006), a atividade de teste de software é um elemento crítico da garantia de qualidade de software e representa a última revisão de especificação, projeto e codificação. Não é incomum que uma organização de software gaste 40% do esforço de projeto total em teste. Alguns casos dos quais dependam vidas humanas (por exemplo, controle de voo), pode custar de 3 a 5 vezes mais que todos os outros passos de engenharia de software juntos.

2.2.1 Tipos de testes

Segundo Pressman(1997), os teste podem ser categorizados:

• Unidade: Também conhecida como Teste Unitário. É a fase do processo de teste em que se testam as menores unidades de software desenvolvidas (pequenas partes ou unidades do sistema). O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenos trechos de código. Assim, o objetivo é o de encontrar falhas de funcionamento dentro de uma pequena parte do sistema funcionando independentemente do todo;

• Integração: Na fase de teste de integração o objetivo é encontrar falhas provenientes da integração interna dos componentes de um sistema. Geralmente os tipos de falhas encontradas são de envio e recebimento de dados. Por exemplo, um objeto A pode estará guardando o retorno de um valor X ao executar um método do

objeto B, porém este objeto B pode retornar um valor Y, desta forma gerando uma falha. Não faz parte do escopo dessa fase de teste o tratamento de interfaces com outros sistemas (integração entre sistemas). Essas interfaces são testadas na fase de teste de sistema, de apesar de que ao critério do gerente de projeto, estas interfaces poderem ser testadas mesmo antes do sistema estar plenamente construído;

• Sistemas: Na fase de Teste de Sistema o objetivo é executar o sistema sob ponto de vista de seu usuário final, varrendo as funcionalidades em busca de falhas. Os testes são executados em condições similares - de ambiente, interfaces sistêmicas e massas de dados - àquelas que um usuário utilizará no seu dia-a-dia de manipulação do sistema. De acordo com a política de uma organização podem ser utilizadas condições reais de ambiente, interfaces sistêmicas e massas de dados.

• Aceitação: Fase de Teste em que o teste é conduzido por usuários finais do sistema. Os testes são realizados, geralmente, por um grupo restrito de usuários finais do sistema. Esses simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado. Teste formal conduzido para determinar se um sistema satisfaz ou não seus critérios de aceitação e para permitir ao cliente determinar se aceita ou não o sistema. Validação de um software pelo comprador, pelo usuário ou por terceira parte, com o uso de dados ou cenários especificados ou reais.

2.2.2 Ferramentas de testes

Segundo Vincenzi (1998), a qualidade e produtividade da atividade de teste são dependentes do critério de teste utilizado e da existência de uma ferramenta que o suporte. Sem a existência de uma ferramenta, a aplicação de um critério torna-se uma atividade propensa a erros e limitada a programas muito simples. As disponibilidades de ferramentas de teste permitem a transferência de tecnologia para a indústria e contribuem para uma contínua evolução de tais ambientes, fatores indispensáveis para a produção de software de alta qualidade.

Além disso, a existência de ferramentas auxilia pesquisadores e alunos de Engenharia de Software a adquirirem os conceitos básicos e experiência na comparação, seleção e estabelecimento de estratégias de teste

Dentre algumas dessas ferramentas temos segundo Domingues, 2002:

• C++ Test é uma ferramenta de teste de unidade para códigos C/C++ que executa os seguintes tipos de teste: teste funcional; teste estrutural; e teste de regressão;

• JProbe Suite é um conjunto de três ferramentas composto por: JProbe Profiler and Memory Debugger que ajuda a eliminar gargalos de execução causados por algoritmos ineficientes em códigos. Java e aponta as causas de perdas de memória nessas aplicações rastreando quais objetos seguram referências para outros;

• Jtest é uma ferramenta de teste de classes para códigos Java que executa os seguintes tipos de teste: análise estática; teste funcional; teste estrutural; e teste de regressão. Esta ferramenta pode executar todos esses tipos de teste em uma simples classe ou em um conjunto de classes;

• PureCoverage é uma ferramenta de análise de cobertura para códigos C++ e Java que aponta as áreas do código que foram ou não exercitadas durante os testes. Ela expõe o código não testado em todas as partes da aplicação.

2.3 METODOLOGIAS ÁGEIS PARA DESENVOLVIMENTO DE SOFTWARE

Neste trabalho monográfico, o foco será dado às metodologias ágeis XP e *Scrum* por serem as mais utilizadas na atualidade. As metodologias ágeis têm despertado cada vez mais o mercado de desenvolvimento de *software* devido aos resultados obtidos com relação à eficácia na produção do mesmo. É utilizada em situações onde as variáveis que englobam o processo são mutáveis, podendo assim gerar resultados imprevisíveis. Esse tipo de metodologia é indicado quando o produto final a ser gerado é algo novo no mercado, onde não se conhece o resultado final.

Na visão de (JACOBSON, 1999) agilidade é explanada da seguinte forma: agilidade tornou-se atualmente uma palavra mágica quando se descreve um processo moderno de software. Tudo é ágil. Uma equipe ágil é uma equipe esperta, capaz de responder adequadamente a modificações. Modificação é aquilo para o qual o desenvolvimento de software está principalmente focado. Modificações no software que está sendo construído, modificações nos membros da equipe, modificações por causa de novas tecnologias, modificações de todas as espécies

que podem ter impacto no produto que eles constroem ou no projeto que cria o produto.

O apoio para modificações deveria ser incorporado em tudo que fazemos em software, algo que se adota porque está no coração e na alma do software. Uma equipe ágil reconhece que o software é desenvolvido por indivíduos trabalhando em equipes e que as especialidades dessas pessoas e sua capacidade de colaborar estão no âmago do sucesso do projeto.

Logo, se vem a discussão qual metodologia utilizar? Cada metodologia tem sua aplicação, tem seus pontos favoráveis e desfavoráveis. Deve-se analisar pela equipe que se tem, podendo ser praticada a abordagem ágil, a tradicional ou a combinação de ambas as práticas para o gerenciamento do projeto.

Uma premissa fundamental das metodologias ágeis é o reconhecimento da dificuldade do usuário saber de antemão as funcionalidades que gostaria que o sistema tivesse. Por isso, essas metodologias adotam a abordagem ascendente, isto é, criam condições favoráveis para as interações e as retro-alimentações entre usuários e o sistema durante todo o projeto, uma vez que são as necessidades reais dos usuários, e não o “conceito” do sistema ideal, o ponto chave do sucesso do projeto. As necessidades dos usuários são, por outro lado, sempre mutáveis, isto é, não são definidas (nem definíveis) a priori, mas vão-se desenhando ao longo do projeto. Donde a necessidade de uma abordagem que rompa com a temporalidade linear do projeto organizado em fases sequenciais como, (a metodologia do “objetos intermediários” (CAMPOS, 2002; VINCK, 1999), que aperfeiçoa as simulações e modelos convencionais de prototipagem).

> A percepção que os usuários têm de suas necessidades também evolui à medida que eles conhecem o sistema. É difícil compreender o valor de uma determinada funcionalidade até que ela seja efetivamente usada, principalmente porque não se pode requerer de um usuário comum a mesma capacidade de abstração que um desenvolvedor possui ao olhar um conjunto de requisitos (OLIVEIRA, 2003, p. 16).

As metodologias ágeis são estruturadas de modo a atender a natureza mutável e dinâmica do processo de concepção do sistema. Diferentemente das metodologias pesadas que possuem fases bem separadas e delimitadas, nas metodologias ágeis, as fases de concepção e desenvolvimento interagem durante

todo o projeto, possibilitando desse modo uma interação constante entre os usuários e os analistas, ou seja, entre os profissionais da concepção e da operação, como ocorre nos projetos na "produção enxuta": “as equipes de projeto japonesas permanecem no local até bem depois da implantação e operam contínuas mudanças, acumulando, assim, novos conhecimentos, que serão transferidos aos sistemas através das modificações que fazem” (LOJKINE, 1995, p. 248).

As metodologias ágeis propõem que os projetos devam ser conduzidos de forma adaptativa, isto é, feito através de desenvolvimento iterativo e interativo. A idéia central é trabalhar com iterações curtas. Cada iteração entrega ao seu final um produto completo e pronto para ser usado, que contém a implementação de um novo subconjunto de características. O uso de iterações curtas permite aos usuários e clientes fazerem uma avaliação do sistema logo que uma versão inicial é colocada em produção.

Neste momento, usuários, clientes e desenvolvedores decidem sobre quais características devem ser adicionadas, quais devem ser modificadas, e até, quais devem ser retiradas do sistema. O sistema é desenvolvido da forma mais iterativa possível.

2.3.1 Manifesto ágil

O manifesto ágil foi criado em 2001, com o objetivo de melhorar a produtividade de projetos de software. Em *Improve It* expõem os valores abordados pelo Manifesto Ágil da seguinte forma: está se descobrindo maneiras melhores de desenvolver software. Através desse trabalho passamos a valorizar:

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

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens da esquerda.

2.3.2 Extreme Programming – XP

Foi criada em meados dos anos 90 no projeto da Chrysler no desenvolvimento de um sistema chamado C3. É utilizada para equipes pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente (BECK, 1999). Recomenda-se o máximo de 12 pessoas envolvidas no processo de desenvolvimento.

Segundo Teles (2004), XP pode ser definido como um processo de desenvolvimento que busca assegurar que o cliente receba o máximo de valor de cada dia de trabalho da equipe de desenvolvimento. A XP enfatiza o desenvolvimento rápido do projeto e visa garantir a satisfação do cliente, além de favorecer o cumprimento das estimativas.

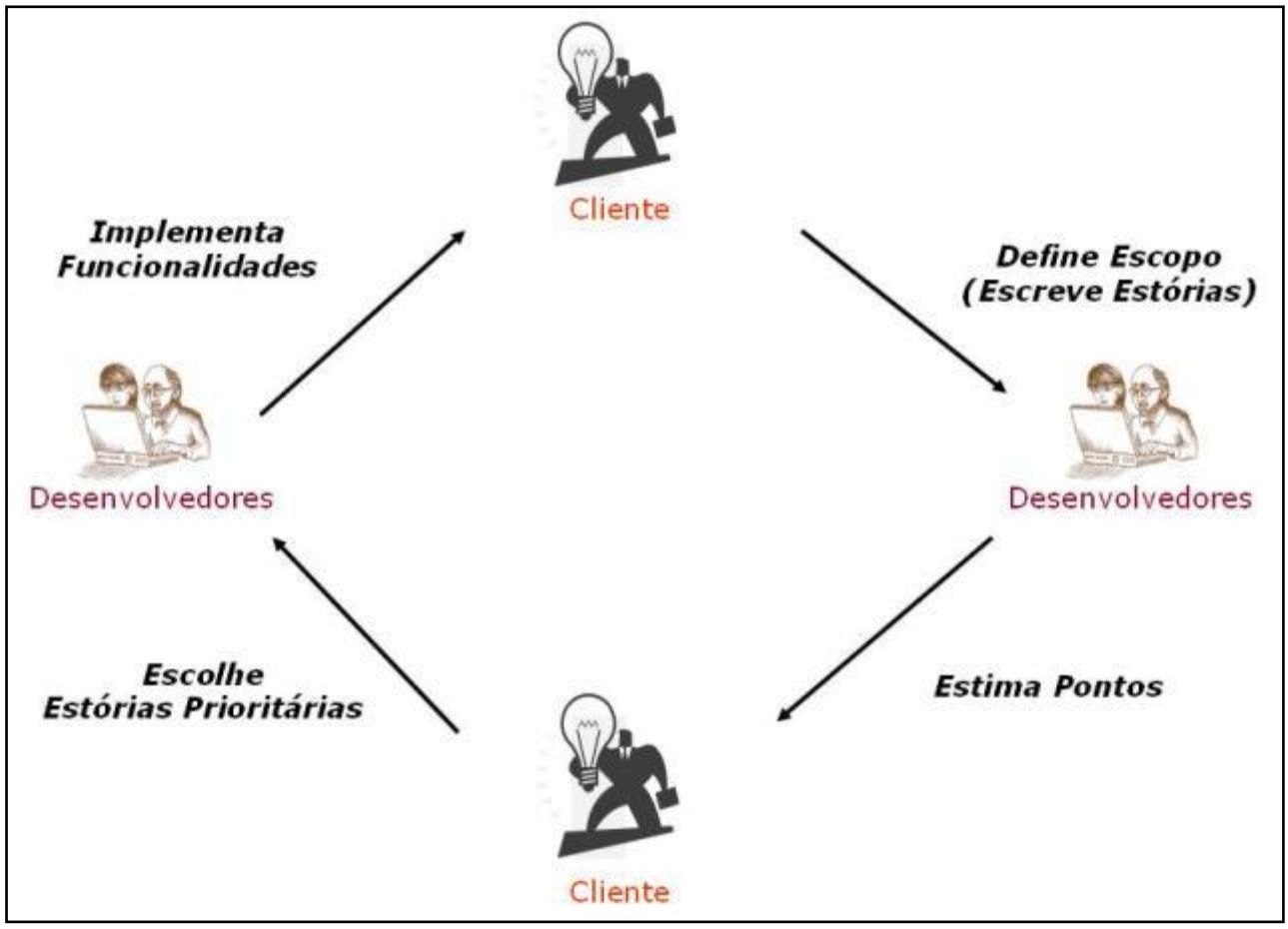

Figura 4- Ciclo de vida do projeto XP. Fonte: Mountain 2012.

A metodologia é baseada em algumas praticas, são elas:

-Feedback

A equipe de programação sempre está informada com assuntos que diz respeito ao código e ao cliente. O código é constantemente testado pela equipe de desenvolvimento, encontrando assim erros individuais ou integrados. O cliente a todo o momento participa do processo, avaliando o software parcialmente que lhe é entregue testando as funcionalidades a que foi solicitada.

O feedback é o mecanismo fundamental que permite que o cliente conduza o desenvolvimento diariamente e garanta que a equipe direcione as suas atenções para aquilo que irá gerar mais valor (TELES, 2004, p.22).

- Comunicação

A equipe de desenvolvimento está em constante comunicação para que não haja desentendimento com relação ao escopo do projeto. O cliente também faz parte dessa comunicação, para que assim possam ser atendidas todas as suas solicitações.

A comunicação entre o cliente e a equipe permite que todos os detalhes do projeto sejam tratados com a atenção e a agilidade que merecem. Dessa forma, busca aproximar todos os envolvidos do projeto de modo que todos possam se comunicar face a face ou de forma mais rica que for viável (TELES, 2004, p.22).

- Simplicidade

É o desenvolvimento de código simples, sem possuir módulos desnecessários fazendo com que o sobrecarregue. Código simples é aquele o qual realiza a mesma tarefa com o menor número de métodos e classes.

O valor da simplicidade, que nos ensina a implementar apenas aquilo que é suficiente para atender a cada necessidade do cliente. Devemos nos preocupar apenas com os problemas de hoje e deixar os problemas do futuro para o futuro (TELES, 2004, p.22).

- Coragem

É preciso acreditar naquilo que é desenvolvido a cada incremento, buscando soluções concretas e descartando outras que não trazem segurança para a equipe.

A equipe precisa ser corajosa e acreditar que, utilizando as práticas e valores do XP, será capaz de fazer o software evoluir com segurança e agilidade (TELES, 2004, p.23).

A partir desses valores, o XP reuniu um conjunto de 12 práticas as quais comprovam a agilidade no processo de desenvolvimento do software. Dentre elas pode-se citar:

- Cliente Presente

O contato direto do cliente com a equipe de desenvolvedores do projeto auxilia a compreensão daquilo que foi solicitado.

O cliente deve conduzir o desenvolvimento a partir do *feedback* que recebe do sistema. É essencial que o cliente participe ativamente do processo de desenvolvimento, para que assim torne o processo simples em diversos aspectos, especialmente na comunicação (TELES, 2004, p.24).

- Jogo do Planejamento

Consiste na tomada de decisões do que pode ser feito e do que se fará futuramente. É trabalhado em cima daquilo que o cliente necessita no momento. A cada reunião é discutido aquilo que será implementado.

O projeto no XP é dividido em *releases* e iterações. *Releases* são módulos do sistema que geram um valor bem definido para o cliente. Iterações são períodos de tempo de poucas semanas (em média, duas semanas) no qual a equipe implementa um conjunto de funcionalidade acordado com o cliente. No início de cada *release* e de cada iteração ocorre o jogo de planejamento (TELES, 2004, p.24).

- *Stand Up Meeting*

Tem seu real sentido em inglês significando “reunião em pé”, não sendo uma coisa que será demorada. A equipe de desenvolvimento se reúne a cada manhã para avaliar o trabalho que foi executado no dia anterior e priorizar aquilo que será implementado no dia que se inicia (TELES, 2004, p.24-25).

É uma forma organizada de iniciar o projeto a cada dia, deixando ciente todos os membros da equipe de desenvolvimento daquilo que será trabalhado ao longo do dia.

- Programação em Par

Duas pessoas trabalham em um único computador desenvolvendo o código. Um auxilia o outro com seus conhecimentos ao longo do desenvolvimento. Enquanto um trabalha no desenvolvimento bruto do código, digitando, o outro fica somente observando a semântica e a sintaxe do código.

- Desenvolvimento Guiado pelos Testes

Cada unidade é estritamente testada pelos programadores antes de serem codificadas. Utiliza-se a técnica computacional dividir para conquistar, pois testando cada unidade individualmente ao juntar as partes, todo o projeto estará funcionando perfeitamente.

Se preocupam com as interfaces externas dos métodos e classes antes de codificá-los, sabem até onde codificar cada funcionalidade e passam a contar com uma massa de testes que podem ser usada a qualquer momento para validar todo o sistema (TELES, 2004, p.25).

- *Refactoring*

A técnica de refatoração consiste no ato de melhorar o código sem que as funcionalidades do mesmo sejam alteradas. É utilizada com o objetivo de tornar mais claro para os desenvolvedores o código trabalhado.

É utilizado para tornar o software mais simples de ser manipulado e se utiliza fortemente dos testes descritos anteriormente para garantir que as modificações não interrompam o seu funcionamento (TELES, 2004, p.25).

- Código Coletivo

Auxilia na agilidade do desenvolvimento do projeto. Ao estar em coletivo pessoas interagem, aprendendo umas com as outras, tornando o processo mais proveitoso.

Os desenvolvedores têm acesso a todas as partes do código e podem alterar aquilo que julgarem importante sem a necessidade de pedir autorização de outra pessoa, pois o código é coletivo (TELES, 2004, p.26).

- Código Padronizado

Auxilia no entendimento de cada membro de diversas partes do projeto. Economiza tempo permitindo facilidade na manutenção do código.

Ajudam a equipe a utilizar a prática do código coletivo, bem como o código coletivo ajuda a equipe a manter os padrões de codificação. Dado que o código é manipulado por diferentes pessoas na equipe, caso alguém fuja do padrão, isso será identificado rapidamente (TELES, 2004, p. 149).

- Design Simples

O custo da manutenção é muito caro, visto que na maioria das vezes o tempo que se perde corrigindo erros é superior ao tempo que gastaria para realizar tal tarefa do ponto inicial com simplicidade.

Assumindo a simplicidade, a equipe evita o trabalho especulativo que ocorre quando ela não tem certeza sobre uma necessidade futura, mas implementa uma solução para esta necessidade, pois acredita que ela poderá surgir futuramente (TELES, 2004, p.154).

- Metáfora

É o software descrito sem a utilização de termos específicos da área, e tem como objetivo guiar toda a operação do projeto. Para facilitar a criação de um design simples, a equipe de desenvolvimento utiliza metáforas, já que elas têm o poder de transmitir idéias complexas de fora simples, através de uma linguagem comum que é estabelecida entre a equipe de desenvolvimento e o cliente (TELES, 2004, p.26).

- Ritmo Sustentável

Não se devem realizar horas extras. Recomenda-se que não ultrapasse a carga horária de 8 horas diárias completando assim 40 horas semanalmente. O foco dessa prática é na saúde dos membros envolvidos no projeto, para que o físico e o emocional do indivíduo possam estar em harmonia no momento do trabalho.

- Integração Contínua

Consiste na prática da construção do software diversas vezes ao dia, deixando a equipe de desenvolvedores em sintonia, possibilitando assim rápidos processos.

- *Releases* Curtos

Uma versão beta do sistema é colocada diariamente ao cliente para que possa ser testada sua funcionalidade e se está justamente da forma que foi solicitado. A vantagem é que o cliente pode começar desde já a se beneficiar do produto.

2.3.3 Scrum

É uma outra metodologia ágil bastante utilizada. Foi criada no início da década de 90 por Jeff Sutherland ao trabalhar na *Easel Corporation*. O *Scrum* é um processo bastante leve para gerenciar e controlar projetos de desenvolvimento de software e para criação de produtos. O *Scrum* é uma metodologia ágil que segue as filosofias iterativa e incremental. Ele se concentra no que é realmente importante:

gerenciar o projeto e criar um produto que acrescente valor para o negócio. O valor decorre da funcionalidade propriamente dita, do prazo em que ela é necessária, do custo e da qualidade (MARTINS, 2007, p.253).

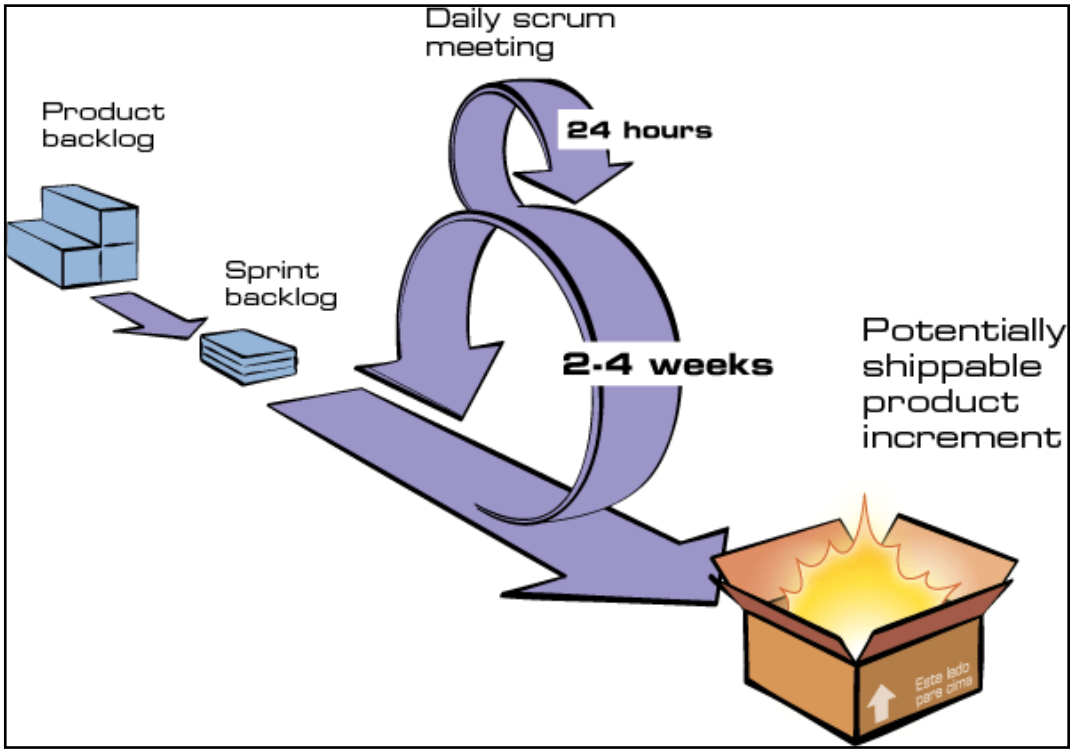

Figura 5- Visão geral do scrum
Fonte: Mountain 2012.

O *Scrum* pode ser utilizado sempre que um grupo de pessoas tenha a necessidade de trabalhar de forma conjunta para atingir um objetivo determinado previamente, desde o gerenciamento de projetos de *software* até tarefas do cotidiano, como organizar uma festa, sendo assim pode ser considerada uma metodologia ágil que valoriza o trabalho em equipe.

Difere um pouco da metodologia XP, recomenda-se equipes pequenas de até sete pessoas. O desenvolvimento é dividido em *Sprints* de 30 dias. As equipes trabalham em cima de cada módulo funcional do sistema os quais são definidos ao início de cada *Sprint.* Um *Sprint* significa o conjunto de tarefas que serão realizadas em um período de tempo pré-determinado.

Assim como o XP que é realizado pequenas reuniões diárias, o *Scrum* também utiliza dessa prática, com reuniões de 15 minutos quando é discutido o que

será feito no dia seguinte. Nessas reuniões também são discutidos os fatores que podem levar ao fracasso e os fatores que levarão ao êxito do projeto.

Segundo (MARTINS, 2007) os *Scrum* são divididos em 3 fases: pré game, game e pós game.

- Pré game

É dividida em duas partes: planejamento e arquitetura. No planejamento é definido um novo *release*, com base nas informações conhecidas até o momento, tabuladas num *Backlog*, junto com uma estimativa de prazo e custo. Esta fase consiste numa análise limitada, inicia-se no desenvolvimento de uma lista de *backlog* e vai até a etapa final que consiste na aprovação da direção para o projeto e para o orçamento. O *release* de um software é planejado com base nas seguintes variáveis: requisitos do cliente, prazo, concorrência, qualidade, visão e recursos (MARTINS, 2007, p.257).

Na arquitetura definem-se as bases para a construção do produto. Os itens do *backlog* são projetados com vistas à implementação. Esta fase inclui definições e modificações na arquitetura e *design* de alto nível (MARTINS, 2007, p.257).

- Game

É composta por um ou mais *Sprints* de desenvolvimento, que tem como objetivo o desenvolvimento das funcionalidades de um novo *release*, sempre respeitando as restrições de prazo, requisitos, qualidade, custos e características dos concorrentes (MARTINS, 2007, p.259).

- Pós game

Quando a gerência decide que as variáveis de tempo, concorrência de mercado, requisitos, custos e qualidade estão adequadas para distribuir o produto, ela declara que o *release* está fechado e a fase de pós-game começa. O produto final é preparado para distribuição, incluindo integração, testes adicionais, documentação de usuário, preparação de material para treinamento e de marketing, dentre outras coisas (MARTINS, 2007, p.259).

2.3.4 Feature Driven Development – FDD

Foi criada em 1999 por Jeff De Luca, Peter Coad e Stephen Palmer quando participavam de um projeto em um banco de Cingapura. A sigla FDD em português significa Desenvolvimento Guiado por Características. Cada iteração no FDD tem o prazo máximo de até duas semanas. Consiste em cinco processos principais: desenvolver o modelo global, construir uma lista de características, planejar por característica, projetar por característica e construir por característica.

Em (MARTINS, 2007, p.317) o FDD é definida como um processo ágil e adaptativo com as seguintes características:

- Iterativo;
- Enfatiza a qualidade;
- Entrega resultados tangíveis e frequentes;
- Provê relatórios de progresso precisos e significativos, requerendo pouca sobrecarga de trabalho por parte dos programadores;
- É apreciado pelos clientes, gerentes e desenvolvedores.

Coad (1999) e seus colegas sugerem o seguinte gabarito para definir uma característica:

<ação> o **<resultado><por|para|de|a>** um **<objeto>**

em que **<objeto>** é uma pessoa, lugar coisa. Um exemplo de característica para uma aplicação de comercio eletrônico poderia ser: exibe as especificações técnicas de um produto.

2.3.5 Crystal/Clear

Faz parte da família *Crystal* criada por Cockburn. É uma metodologia utilizada em pequenos projetos, com equipe de no máximo 6 desenvolvedores, onde cada membro da equipe é responsável por um módulo específico. A comunicação entre os desenvolvedores da equipe é de extrema relevância.

As iterações são realizadas no período de 1 mês onde uma versão Beta do sistema é entregue para que o cliente possa testar.

De acordo com Pressman (2006, p.71) a família Crystal é na verdade, um conjunto de processos ágeis que se mostram efetivos para diferentes tipos de

projetos. A intenção é permitir que equipes ágeis selecionem o membro da família Crystal mais apropriado para seu projeto e ambiente.

2.3.6 *Dynamic Systems Development Method* – DSDM

O DSDM é progenitor do XP, significa Método de Desenvolvimento Dinâmico de Sistemas. Assim como o XP suporta mudanças nos requisitos durante o ciclo de vida e utiliza-se pequenas equipe para o desenvolvimento de projetos. Possui características do processo RAD (Desenvolvimento Rápido de Aplicações).

De acordo com Pressman (2006, p.68) define cinco ciclos iterativos diferentes o ciclo de vida do DSDM, precedidos de duas atividades adicionais de ciclo de vida:

- estudo de viabilidade: estabelece os requisitos básicos e restrições do negócio associados à aplicação em construção e depois avalia se a aplicação é candidata viável ao processo DSDM.

- estudo do negócio: estabelece os requisitos funcionais e de informação que permitirão à aplicação fornecer valor de negócio.

- interação do modelo funcional: produz um conjunto de protótipos incrementais que demonstram a funcionalidade para o cliente.

- iteração de projeto e construção: revisita os protótipos construídos durante a iteração do modelo funcional para garantir que cada um tenha passado por engenharia, de modo que seja capaz de fornecer valor de negócio operacional para os usuários finais.

- implementação: coloca o último incremento de software no ambiente operacional.

O desenvolvimento é voltado a *Feature*, ou funcionalidade, que representa um requisito funcional do sistema. É definido como sendo mais apropriado a projetos iniciais, à atualização de código existente, à criação de uma segunda versão, ou ainda à substituição de um sistema inteiro em partes. Diferentemente de outros métodos ágeis, o FDD possui características específicas para desenvolver sistemas críticos (SILVA, 2006).

2.3.7 *Adaptive Software Development* - ASD

Metodologia proposta por Jim Highsmith tem como características: modelo iterativo e incremental, utilizada em sistemas grandes e complexos e cliente também sempre presente auxiliando no desenvolvimento do projeto.

Segundo (PRESSMAN, 2006, p.67) a ASD tem seu ciclo de vida definido em 3 fases:

- Especulação: o projeto é iniciado e o planejamento do ciclo adaptativo é conduzido. Planejamento do ciclo adaptativo usa informações de iniciação do projeto – a declaração de missão feita pelo cliente, as restrições do projeto – e requisitos básicos – para definir o conjunto de ciclos de versão que serão necessários para o projeto.

- Colaboração: pessoal motivado trabalha junto de um que multiplica seus talentos e resultados criativos além de seu número absoluto. Essa abordagem colaborativa é um tema recorrente em todos os métodos ágeis. Não é apenas uma comunicação, nem somente uma questão da equipe de trabalho, é uma questão de confiança.

- Aprendizado: a medida que os membros de uma equipe ASD começam a desenvolver os componentes que fazem partes do ciclo adaptativo, a ênfase está no aprendizado quanto no progresso em direção a um ciclo completo. As equipes ASD aprendem de 3 modos:

Foco nos grupos onde o cliente e/ou usuários finais fornecem feedback sobre os incrementos de software; *Revisões de técnicas formais* onde os membros da equipe revisam os componentes de software que são desenvolvidos; Pós-conclusão onde a equipe torna-se introspectiva, cuidando do seu próprio desempenho e processo.

2.4 COMPARATIVO ENTRE A METODOLOGIA ÁGIL E A METODOLOGIA TRADICIONAL

De acordo com as metodologias utilizadas, cada uma oferece uma vantagem sobre um aspecto e desvantagem sob outro, por isso o quadro abaixo traz um demonstrativo de comparação entre as duas metodologias.

Abordagem Tradicional	**Abordagem Ágil**
Preditivo: detalhar o que ainda não é bem conhecido	**Adaptativo**: conhecer problema e resolver o crítico primeiro
Rígido: seguir especificação predefinida, a qualquer custo	**Flexível**: adaptar-se a requisitos atuais, que podem mudar
Burocrático: controlar sempre, para alcançar objetivo planejado	**Simplista**: fazer algo simples de imediato e alterar se necessário
Orientado a processos: segui-los possibilita garantir a qualidade	**Orientado a pessoas**: motivadas comprometidas e produtivas
Documentação gera confiança	**Comunicação** gera confiança
Sucesso = **entregar o planejado**	Sucesso = **entregar o desejado**
Gerência = "**comando-e-controle**" voltado para trabalho em massa, ênfase: papel do gerente, com planejamento e disciplina fortes	Gerência = **liderança/orientação** trabalhadores do conhecimento, ênfase: criatividade, flexibilidade e atenção às pessoas
Desenvolvedor hábil (variedade)	**Desenvolvedor** ágil (colaborador)
Cliente pouco envolvido	**Cliente** comprometido (autonomia)
Requisitos conhecidos, estáveis	**Requisitos** emergentes, mutáveis
Retrabalho/reestruturação caro	**Retrabalho/reestruturação** barata
Planejamento direciona os resultados (incentiva controlar)	**Resultados** direcionam o planejamento (incentiva mudar)
Conjunto de processos, com metodologia definida	**Conjunto de valores**, com atitudes e princípios definidos
Premia a garantia da qualidade	**Premia** o valor rápido obtido
Foco: grandes projetos ou os com restrições de confiabilidade, planej. estratégico / priorização (exigem mais formalismo)	**Foco:** projetos de natureza exploratória e inovadores, com equipes pequenas/médias (exigem maior adaptação)
Objetivo: controlar, em busca de alcançar o objetivo planejado (tempo, orçamento, escopo)	**Objetivo:** simplificar o processo de desenvolvimento, minimizando e dinamizando tarefas e artefatos

Quadro 1 - Quadro comparativo adaptado por Nigrado 2012. Fonte: http://neigrando.wordpress.com.

O que se observa é que as metodologias tradicionais, se preocupam mais com a relação de adiantamento a problemas, é uma metodologia proativa, se adianta, por isso é considerada ainda como sendo cansativa e causa desgaste aos colaboradores que com ela trabalham. Já as metodologias ágeis se concentram na adaptação do problema e não ao adiantamento destes, assim são realizados projetos mais flexíveis, e consequentemente mais produtiva e por sua vez mais proveitosa.

3 METODOLOGIA

Este capítulo tem todos os elementos que favorecem o entendimento de como foi realizada essa pesquisa o tipo de metodologia utilizada e a população estudada.

3.1 DELINEAMENTO DA PESQUISA

Para a realização desse estudo fez-se uso da metodologia descritiva, exploratória e de caráter qualitativo e quantitativo.

A pesquisa quantitativa segundo Polit e Hungler (1995, p.45), é aquela que “permite a contemplação entre palavras e números, as duas linhagens fundamentais da comunicação humana”. Na pesquisa qualitativa, “não se emprega procedimentos estatísticos como centro do processo de análise de um problema”. E encontrar a sua solução.

A partir dos conceitos supracitados e mediante o objetivo deste estudo, esta pesquisa é caracterizada de acordo com Gil (2004) como “exploratória” e “descritiva”. Para realizá-la, utilizou-se como apoio o estudo bibliográfico e o documental (utilização de conceitos e teorias publicadas em livros, periódicos, manuais, etc.); e a aplicação de questionário aos gestores e colaboradores de empresas que desenvolvem software na cidade de Vitória da Conquista – BA.

3.2 DEFINIÇÃO DA ÁREA OU POPULAÇÃO ALVO

A pesquisa foi realizada no universo de empresas de desenvolvimento de software na cidade de Vitória da Conquista - BA. Inicialmente, para localizar as empresas foi utilizado pesquisas em sites de busca, lista de telefones e informações de discentes que já haviam realizados pesquisas no mesmo universo, onde foi selecionado um total de 22 empresas que atuam no ramo de desenvolvimento de software na região.

Vale ressaltar que o universo apresentado não se refere ao total de empresas existentes na região. Durante a procura houve dificuldade em encontrar algumas empresas, pois algumas mudaram de endereço, outras se instalaram em

locais não conhecidos, ou então por estarem em locais não identificados com placas ou algo do tipo que identificasse a empresa no lugar citado.

3.3 PLANO DE COLETA DE DADOS

Para obtenção dos dados foram utilizados questionários por ser um meio de custo baixo, análise fácil e acesso fácil ao universo presente. Os questionários foram aplicados no período de 16/07/2012 até o dia 03/08/2012 quando foram entregues pessoalmente em cada empresa num total de 18 questionários e recolhido para análise um total de 15 empresas, pois as outras não entregaram o mesmo. Os questionários foram respondidos pelos profissionais responsáveis pelo setor de desenvolvimento de cada empresa.

3.4 ANÁLISE DOS DADOS DA PESQUISA

Os dados provenientes do questionário aplicado foram tabulados utilizando-se algumas ferramentas gráficas e estatísticas do *software* aplicativo *Microsoft Office Excel*.

4 ANÁLISE DOS DADOS DA PESQUISA

Este capítulo traz os dados coletados junto as empresas estudadas. O universo dessa pesquisa é de 22 empresas, sendo que apenas 15 se propuseram a entregar os questionários respondidos por completo, outros sete foram entregues sem estarem totalmente completo com as questões respondidas. Dessa forma os dados coletados representam a coleta de dados que foram recolhidos pelo pesquisador.

4.1 ANÁLISE DOS DADOS

De acordo com os dados coletados por meio de aplicação de questionários constatou-se que as empresas estão na área de atuação de *software* convencionais com um percentual de 70%, seguidos de software governamental com 20% de representação, e outros 10% de *software* industrial conforme pode ser visto abaixo no (Gráfico 1).

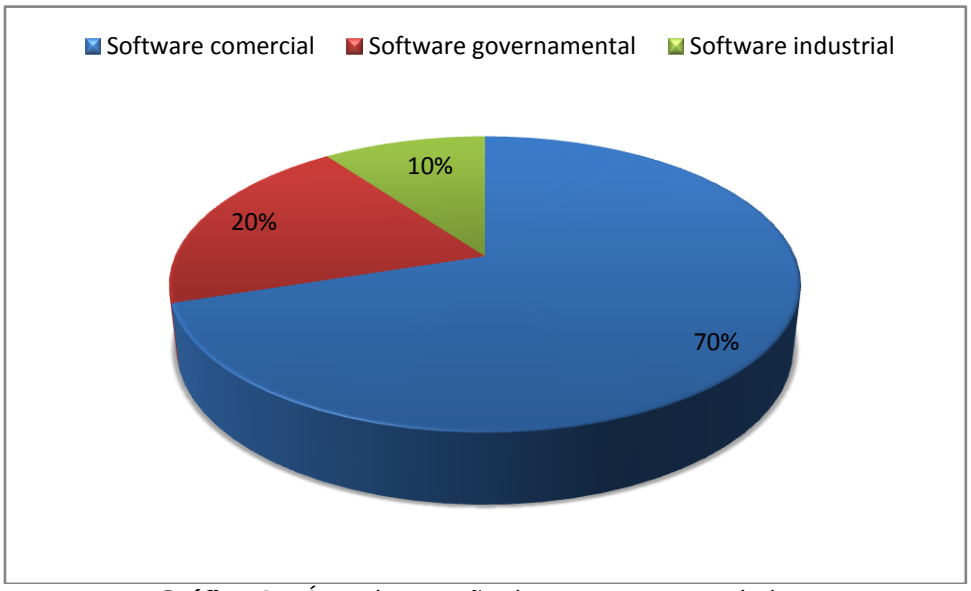

Gráfico 1 – Área de atuação das empresas estudadas.
Fonte: pesquisa de campo 2012.

A área de atuação das empresas implicam demonstrar quais são as suas atuações no desenvolvimento de *software*. Pressman (2002) defende que cada área desenvolve software para um determinado fim, com objetivos previamente definidos.

Com relação ao tempo de atuação das empresas no mercado de informática é de 6 a 10 anos para a maioria significativa das empresas estudadas, seguidos de 20% com tempo entre 11 e 20 anos, e apenas 10% entre 1 e 5 anos (gráfico 2).

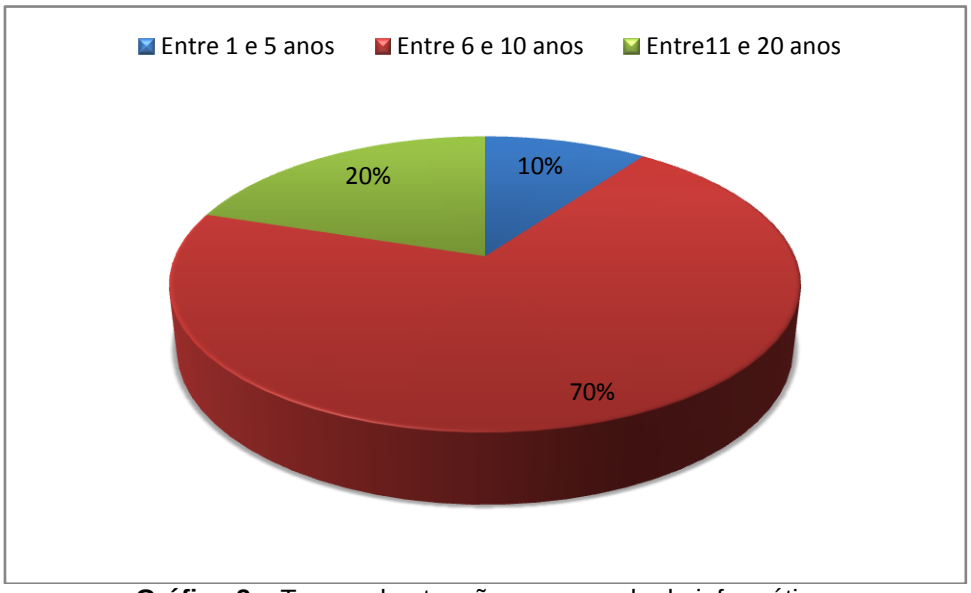

Gráfico 2 – Tempo de atuação no mercado de informática.
Fonte: pesquisa de campo 2012.

O desenvolvimento de *software* é considerado uma atividade de caos (MARTINS, 2007). Sendo asssim, é preciso que se tenha um tempo adequado para o entendimento das atividades a serem desenvolvidas, e o tempo de atuação das empresas e funcionários na área é um fator importante para se ter um entendimento claro das metodologias e a adequação para o seu uso.

O planejamento é um ponto a ser levado em consideração também na ciência da computação, assim como é nas demais engenharias que planejam antes de executar, suas ações e o tempo na área é considerado positivo para autores como (TELES, 2004) e (PRESSMAN 2006).

O público alvo das empresas estudadas são em maioria para empresas locais com representação de 60%, outros 30% para empresas regionais e apenas 10% para empresa nacionais (gráfico3).

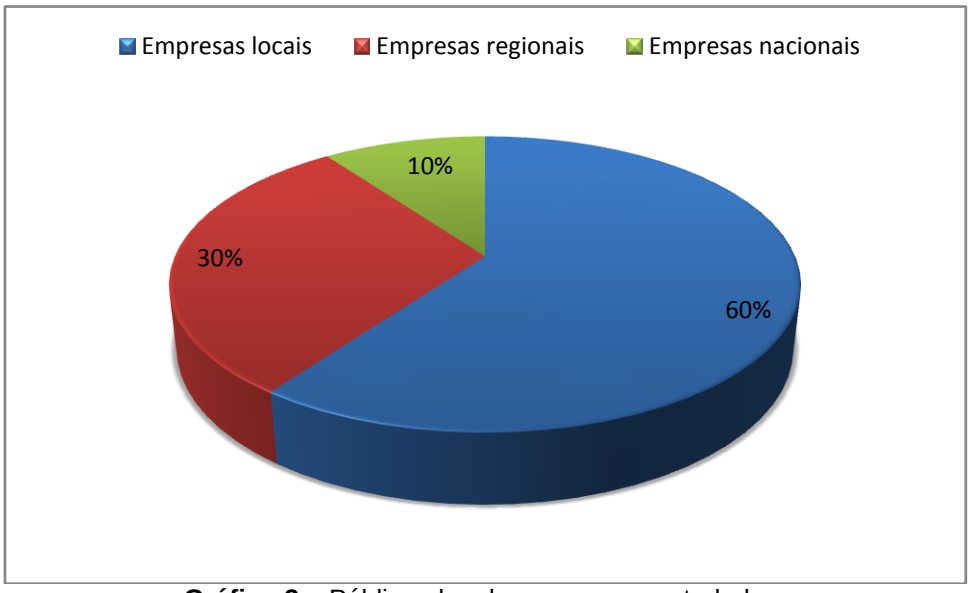

Gráfico 3 – Público alvo das empresas estudadas.
Fonte: pesquisa de campo 2012.

As empresas de desenvolvimento de software estão atendendo em média mais público local e regional, do que nacional. Isso se constata em relações estabelecidas pelo (SEBRAE, 2012), por ser um segmento ainda em ascensão e que primam pela qualidade dos produtos e serviços oferecidos. Essa qualidade não se limita apenas a ausência de defeitos, mas no aprimoramento de recursos técnicos para se manterem no mercado cada vez mais competitivo.

O tipo de empresa que mais são atendidas com frequência são de 40% para microempresas, seguidos de 30% para empresas de pequeno porte, outros 20% para empresas de médio porte e apenas 10% para empresas de grande porte (gráfico 4).

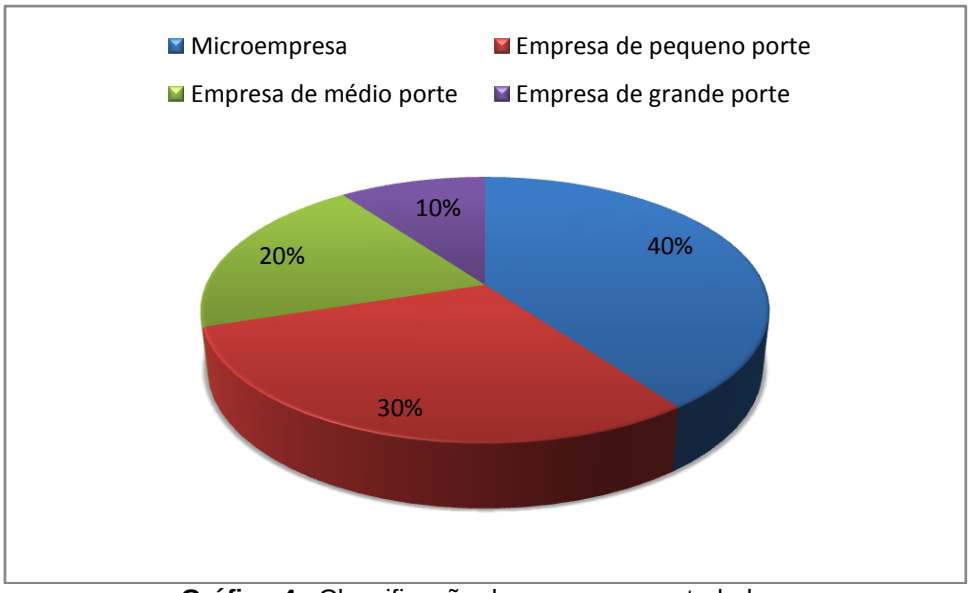

Gráfico 4 –Classificação das empresas estudadas.
Fonte: pesquisa de campo 2012.

O método de desenvolvimento de *software* para as empresas dependem do que elas precisam, as empresas estudadas são em maioria de pequeno porte. Essas empresas são classificadas como micro, pequeno porte, médio porte e grande porte. De acordo com a classificação do (SEBRAE, 2012). As micro empresas são aquelas com receita bruta anual igual ou inferior a R$ 433.755,14 (quatrocentos e trinta e três mil, setecentos e cinquenta e cinco reais e quatorze centavos). As de pequeno porte com receita bruta anual superior a R$ 433.755,14 e igual ou inferior a R$ 2.133.222,00 (dois milhões, cento e trinta e três mil, duzentos e vinte e dois reais).

Essas empresas representam um percentual significativo na economia do Brasil e têm se mostrado com grande desenvolvimento na área de informática nos últimos dez anos. Com os dados coletados também se constatou que as empresas estudadas atuam em sua maioria nas áreas de educação, de contabilidade e de jogos.

Com relação a composição da equipe os participantes do estudo disseram que são em maioria é de empresarios sem socios e capital próprio com 80%, seguidas de 20% com empresa mista, composta por capital próprio e de terceiros (gráfico 5).

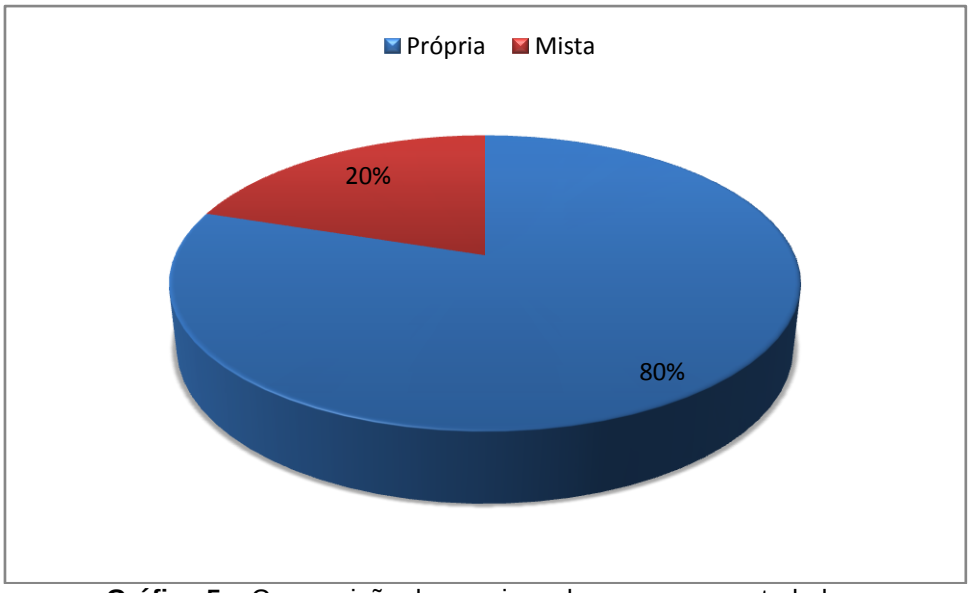

Gráfico 5 – Composição das equipes das empresas estudadas.
Fonte: pesquisa de campo 2012.

Uma equipe de trabalho é o ponto mais importante para qualquer empresa. Na pesquisa realizada há predominância de empresas com capital próprio, seguidos de mista, ou seja, com próprio e terceiros e não há empresa terceirizada.

A quantidade de funcionários das empresa estudadas é de 30% composta de 4 a 5 pessoas, seguidos de outros 30% com 6 a 10 pessoas, ficando 20% com até 3 pessoas, 10% com 11 a 30 pessoas e outros 10% restante com mais de 30 pessoas (gráfico 6).

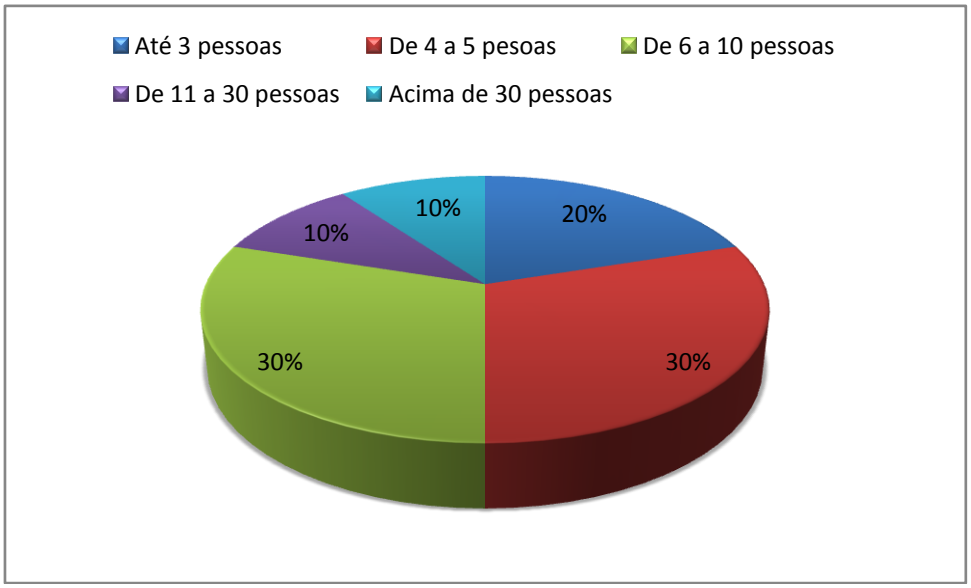

Gráfico 6 – Quantidade dos funcionários nas empresas.
Fonte: pesquisa de campo 2012.

A quantidade de funcionários é um ponto importante, pois equipes pequenas são mais fáceis de se adaptarem as mudanças e agilizar os processos. Equipes prósperas de desenvolvimento de software precisam constantemente trabalhar na extremidade do caos, eles precisam agir rapidamente sem perder o controle. Isto, às vezes, significa cometer algumas falhas, mas sem perder a coragem para se tomar as decisões certas, mesmo quando pressionado para fazer outra tarefa (HAYES, 2001).

O grau de formação das equipes que compõem as empresas estudadas são de 70% com especialização, seguidos de 20% com mestrado de até 3 pessoas e 10% com nível médio técnico até 3 pessoas (gráfico 7).

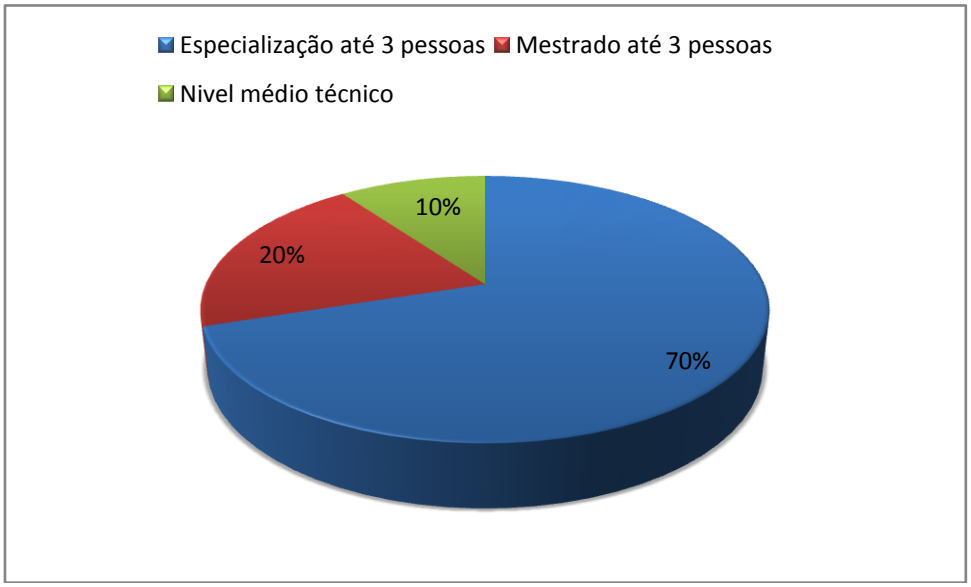

Gráfico 7 – Grau de formação dos funcionários das empresas.
Fonte: pesquisa de campo 2012.

Uma equipe qualificada para atender as necessidades da empresa é de relevância significativa para garantir a qualidade nas metas estabelecidas. (MARTINS, 2007, p. 32), defende que “uma vez que os desenvolvedores são profissionais capazes em suas próprias áreas de conhecimento, eles precisam trabalhar como iguais com os outros profissionais das outras áreas de conhecimento”.

Por isso, ter uma equipe diversificada que trabalham de forma conjunta é como um grande “quebra cabeças” que se juntam com um propósito e conseguem atingir os mesmos com mais eficiência.

Os cursos de graduação das pessoas que formam as equipes das empresas são em maioria significativa de 70% de ciência da computação, outros 20% com Sistemas de Informação, e apenas 10% com análise de sistemas (gráfico 8).

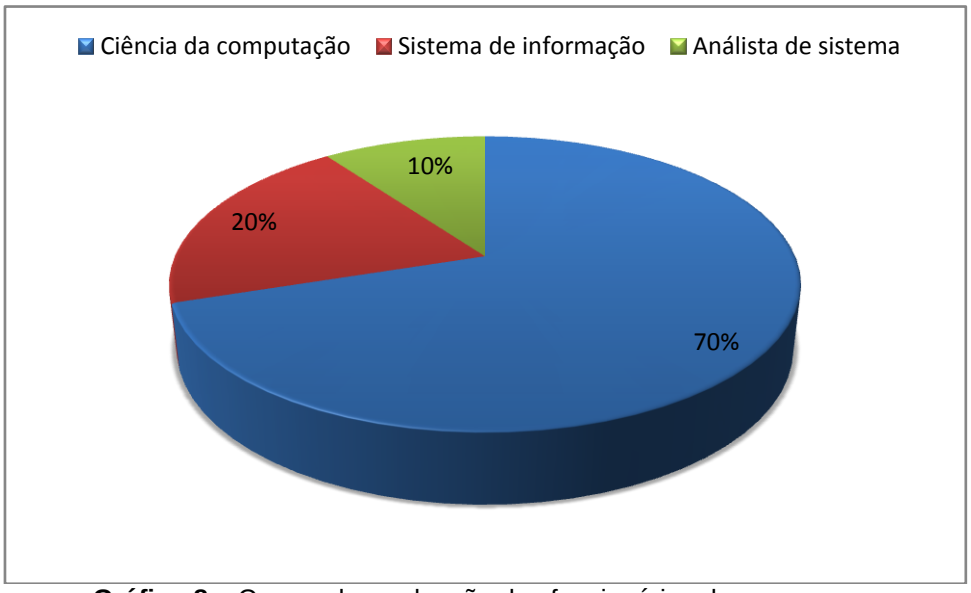

Gráfico 8 – Cursos de graduação dos funcionários das empresas.
Fonte: pesquisa de campo 2012.

Os cursos de graduação são importantes fontes de conhecimento para ter profissionais qualificados no desenvolvimento de novos projetos. Porém a formação continuada garante mais qualidade e efetivação dos projetos.

De acordo com os dados coletados as metodologias utilizadas pelas empresas são em sua maioria tradicional com 60%, os outros 40% com metodologia ágil (gráfico 9).

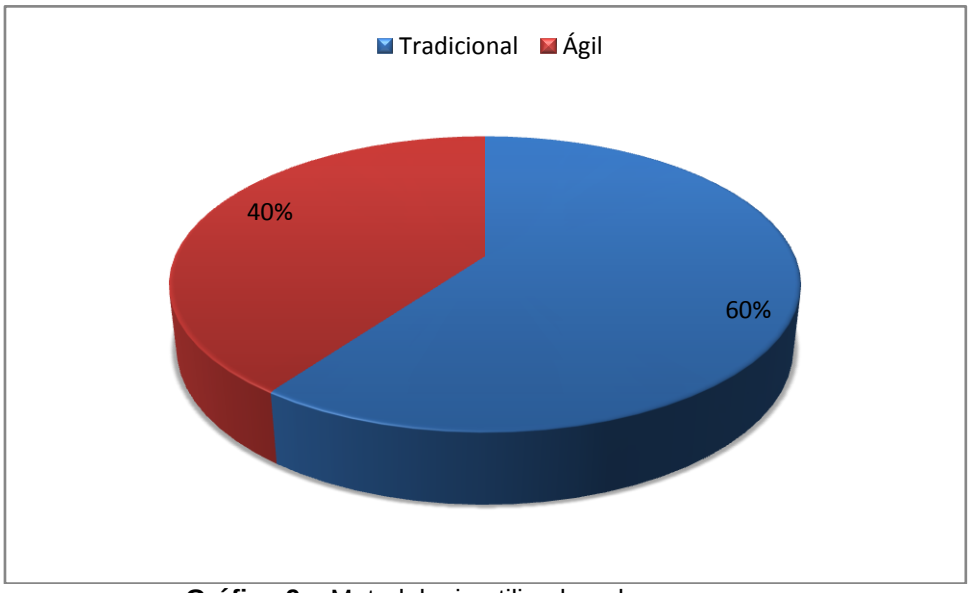

Gráfico 9 – Metodologia utilizada pelas empresas.
Fonte: pesquisa de campo 2012.

A metodologia utilizada pelas empresas pesquisadas é, em sua maioria, ainda a tradicional, esse tipo de metodologia segundo (MOREIRA; RIOS, 2006), é uma metodologia pesada, mas a sua grande vantagem reside na comparação e repetição com dados obtidos anteriormente devido a sua padronização, os processos podem ser identificados e solucionados por meio de mensuração do mesmo. Historicamente, elas predominaram e ainda são muito utilizadas nos projetos de desenvolvimento de software (OLIVEIRA, 2004).

As metodologias mais utilizadas são o Scrum adaptado e o XP, ambos com 90% de frequência de uso. O *Scrum* é um processo bastante leve para gerenciar e controlar projetos de desenvolvimento de software e para criação de produtos. O *Scrum* é uma metodologia ágil que segue as filosofias iterativa e incremental (MARTINS, 2007). Concordando com os dados coletados no presente estudo e a literatura vigente sobre o tema e atendendo ao objetivo desse estudo.

O tempo de uso da metodologia ágil pelas empresas que adotaram esse método é de 50% de 1 a 3 anos, seguidos de 30% com menos de 1 ano, outros 20% não souberam responder e apenas 10% de 3 a 7 anos (gráfico 10).

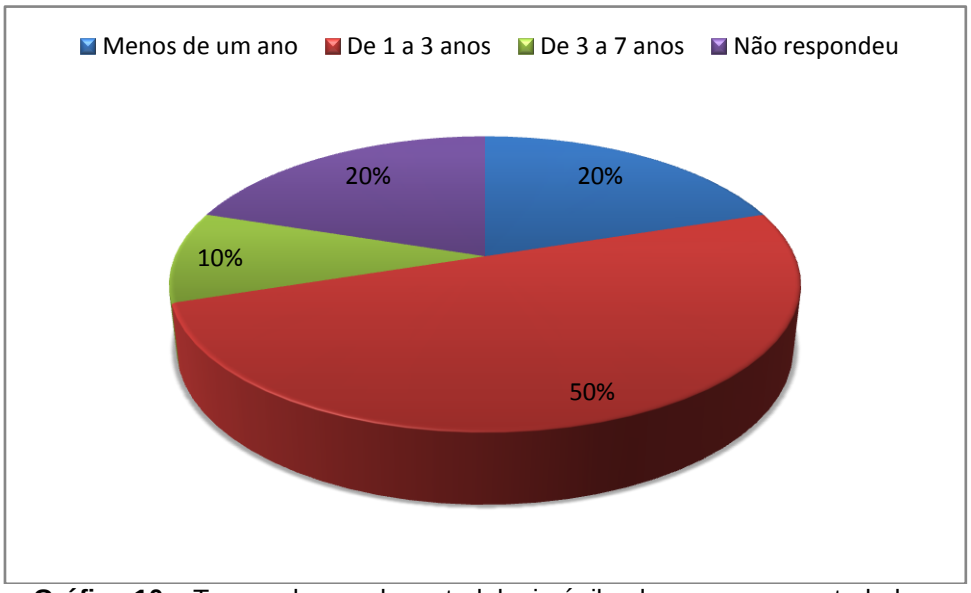

Gráfico 10 – Tempo de uso da metodologia ágil pelas empresas estudadas.
Fonte: pesquisa de campo 2012.

Os métodos ágeis são adaptativos e não preditivos, como é o caso das metodologias tradicionais. Eles se adaptam aos novos fatores que surgem durante o desenvolvimento do projeto ao invés de verificar antecipadamente todos os

impedimentos que podem acontecer durante a construção do produto. O problema em si não está nas mudanças, pois estas estão sempre sujeitas a ocorrer em quaisquer dos âmbitos. O problema está na maneira que cada uma das metodologias trata essas mudanças. Como recebem, avaliam e respondem às mesmas (COCKBURN, 2001).

A adaptação da equipe com relação às mudanças na metodologia utilizada foi considerada normal para 60% dos entrevistados, seguidos de 20% que consideraram fácil e outros 20% que afirmaram ser dificil (gráfico 11).

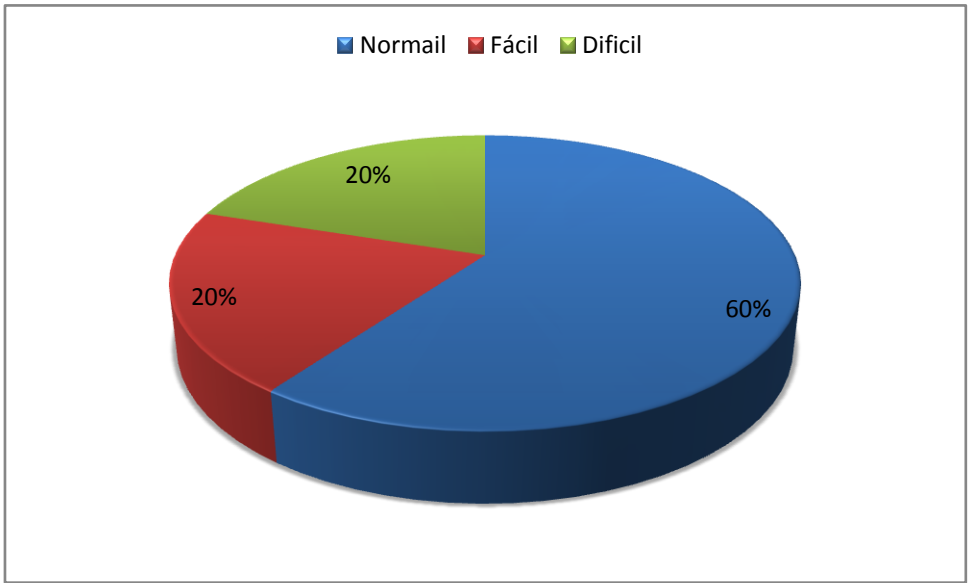

Gráfico 11 – Adaptação das equipes às mudanças necessárias.
Fonte: pesquisa de campo 2012.

A adaptação da equipe às metodologias utilizadas são também importantes pois, segundo Martins (2007), a equipe precisa ser efetiva tanto na qualidade de seus indivíduos, quanto em como essa equipe interage da forma de um time de verdade que buscam objetivos comuns. Existe também uma sinergia importante: não apenas a adaptatividade requer uma equipe mais forte, mas também a maioria dos bons desenvolvedores prefere um processo adaptativo.

Ainda de acordo com Martins (2007) na metodologia tradicional as pessoas não são consideradas importantes e sim os métodos e as suas funções, o contrário da metodologia ágil.

Os fatores que foram considerados importantes para o uso de metodologias ágeis pelos participantes do estudo foram o aumento da produtividade para 40%, seguidos de outros 40% que consideraram a redução de riscos, e apenas 20% restante consideraram que foi o fato da equipe ser flexìvel para atender as novas necessidades (gráfico 12).

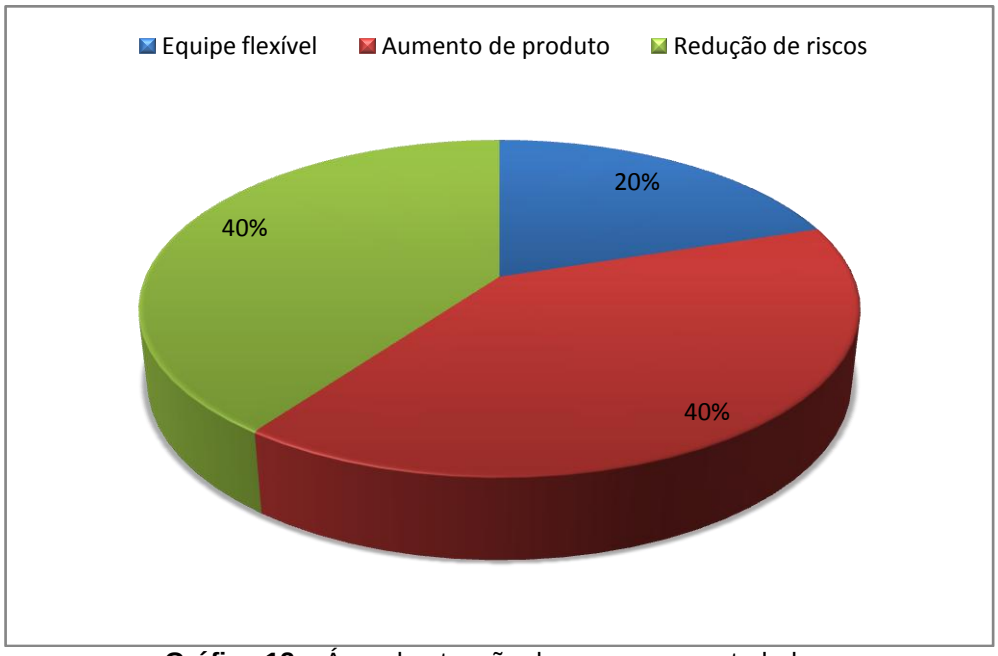

Gráfico 12 – Área de atuação das empresas estudadas.
Fonte: pesquisa de campo 2012.

O aumento da produtividade é um dos pontos mais relevantes da metodologia ágil pelas empresas. (MARTINS, 2007) acredita que uma das vantagens mais importantes é poder produzir mais em tempo menor e oferecer melhores condições para a empresa e a satisfação para o cliente. A redução dos riscos também é considerada uma ponto positivo para se trabalhar com a metodologia ágil.Também tem como característica a simplificar o processo de desenvolvimento, minimizando e dinamizando tarefas e artefatos.

As razões para o uso da metodologia ágil foi considerada para a maioria dos participantes do estudo como sendo a indicação da equipe para 34%, seguidos de 33% afirmando que foram palestras, outros 22% disseram que foi as pesquisas e somente 11% disseram que foi o enquadramento (gráfico 13).

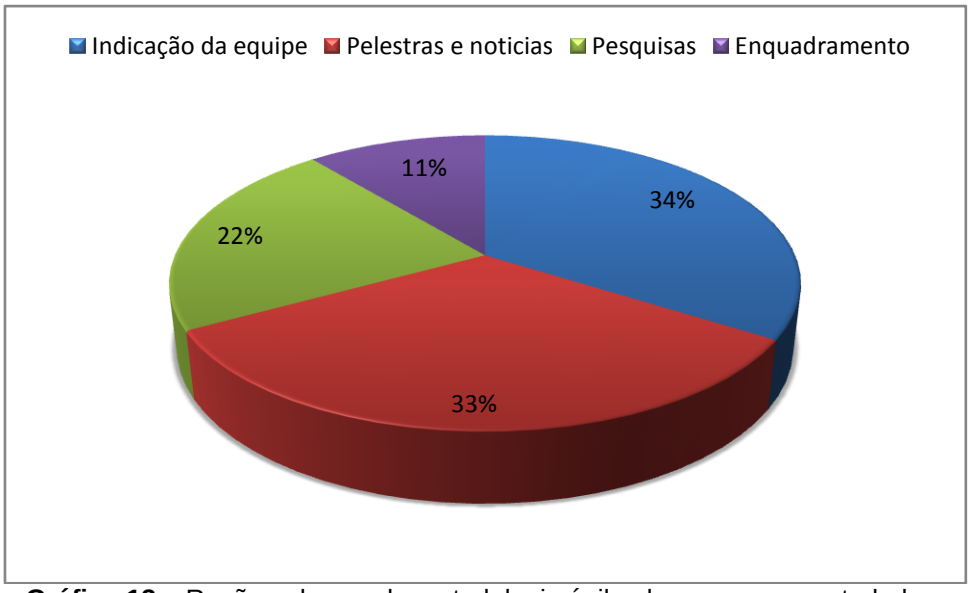

Gráfico 13 – Razões de uso da metodologia ágil pelas empresas estudadas.
Fonte: pesquisa de campo 2012.

O uso da metodologia ágil foi adotada depois de se ter assistido palestras sobre o tema e buscado informações para que a mesma pudesse atender as necessidades das emrpesas. Quanto mais impedimentos ocorrerem durante a execução do projeto, maior será o re-trabalho e o tempo de finalização do mesmo. Isso implica em mudanças estruturais complicadas e que têm impacto muito grande sobre o custo, o prazo e a qualidade do produto final (SOUZA, 2008).

Quando questionados se já houve testes na empresa os resultados apontam que 60% afirmaram que sim, outros 40% que não fizeram testes (gráfico 14).

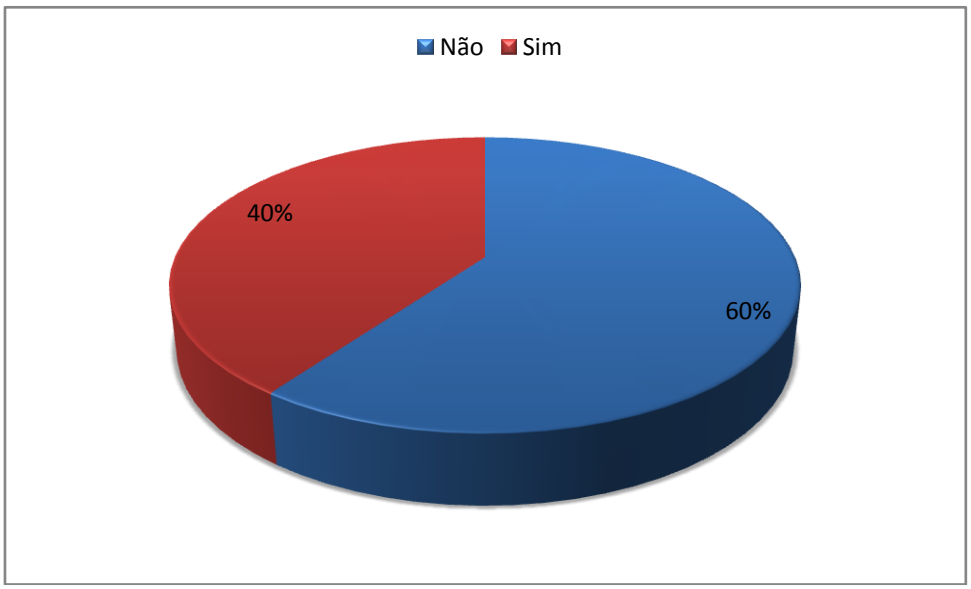

Gráfico 14 – Realização de testes pelas empresas estudadas.
Fonte: pesquisa de campo 2012.

Por muito tempo, os testes foram realizados de maneira muito arcaica. Eram realizados pelos próprios desenvolvedores através de simulações manuais ou de programas temporários criados com este fim. Existiam poucas ferramentas disponíveis, as quais eram difíceis de manipular e ofereciam poucos benefícios à realização da atividade de testes. Com a complexidade dos sistemas, surgiu a necessidade de testá-los e elaborar ferramentas mais eficientes que introduzissem maiores ganhos ao processo (MOREIRA; RIOS, 2006). À medida que os *softwares* aumentavam em complexidade, tornava-se mais difícil o gerenciamento e execução dos testes.

Para Pressman (2006), o teste de integração pode ser realizadousando-se uma estratégia baseada em uso. O teste baseado em usoconstrói o sistema em camadas, começando com aquelas classes que nãoutilizam classes servidoras. Os métodos de projeto de casos de teste deintegração podem também usar testes aleatórios e de partição. Além disso,o teste baseado em cenário e os testes derivados dos modelos decomportamento podem ser usados para testar uma classe e suascolaboradoras. Uma sequencia se testes rastreia o fluxo de operações aolongo das colaborações de classe.

As principais dificuldades encontradas pelas empresas foram para 30% a resistência da equipe em mudanças, outros 30% a colaboração do cliente, seguidos de outros 30% com a capacitação, e apenas 10% com a complexidade (gráfico15).

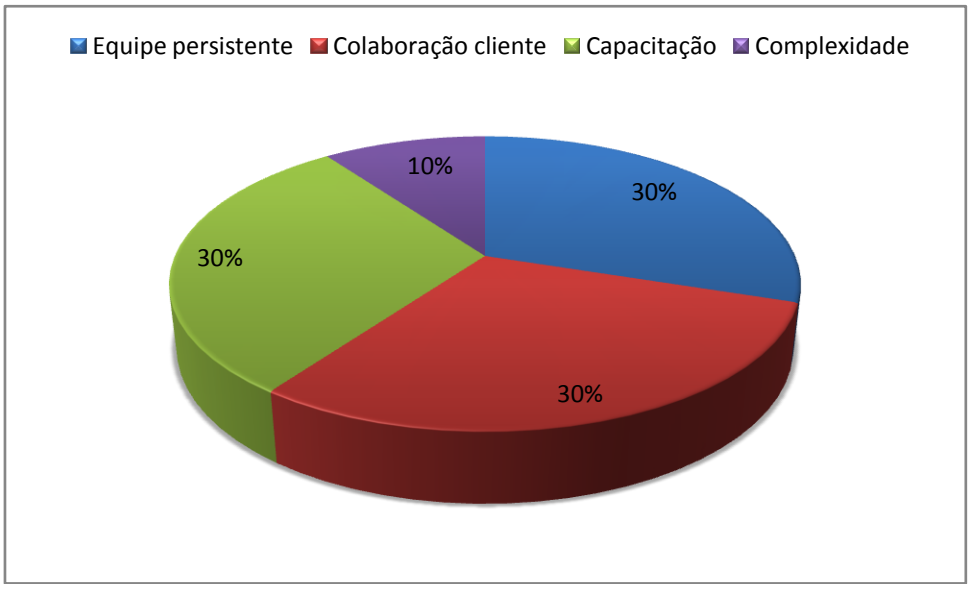

Gráfico 15 –Dificuldades encontradas pelas empresas estudadas.
Fonte: pesquisa de campo 2012.

As dificuldades relevantes são muitas e para Martins (2007) o cliente é uma das mais importantes, pois quando se contrata uma outra empresa para desenvolver um software, é comum os clientes preferirem um contrato fixo e ai o ônus do desenvolvimento será exclusivamente da empresa. Isso por que os contratos fixos requerem alguns requisitos estáveis e implica que se trabalhe com o custo fixo. O que dificulta a execução em tempo hábil e gera conflitos de interesse.

Os resultados dessa pesquisa estão de acordo com o pensamento de Martins (2007) e de Teles (2006).

As práticas utilizadas foram para 30% os códigos coletivos, outros 20% Stand up, seguidos de mais 20% com código padronizado e mais 20% com design simples e apenas 10% com interação continuada (gráfico 16).

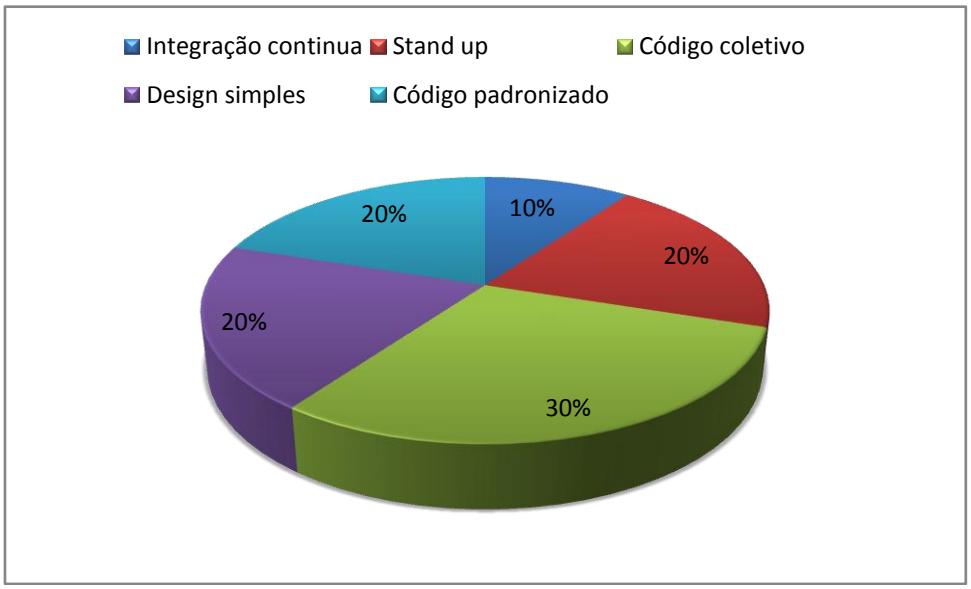

Gráfico 16 –Práticas utilizadas pelas empresas estudadas.
Fonte: pesquisa de campo 2012.

Para a construção e desenvolvimento de um software é necessário o planejamento adequado, Martins (2007) defende que uma vez que os desenhos especificam as partes e como elas devem ser encaixadas, eles agem como uma fundação para um plano de construção ainda mais detalhado. De tal plano pode ser derivado o que precisa ser feito e quais dependências existem entre as tarefas. Isso permite um cronograma e um orçamento razoavelmente previsíveis para a construção.

Também é preciso que esse planejamento seja realizado de forma conjunta, por isso, Teles (2006) acredita que as interações continuadas também são importantes para serem utilizados no desenvolvimento de softwares.

Neste contexto, Martins (2007) defende que são duas as atividades mais importantes e que são diferentes entre si, *Design* - que é difícil prever e requer pessoas caras e criativas, e *construção* - que é mais fácil de prever. Uma vez que se tem o *design*, então se pode planejar a construção.

A análise de risco é considerada como ponto forte para 40% dos participantes do estudo, outros 40% afirmaram ser o ponto fraco, e apenas 20% afirmaram que não há interferências (gráfico17).

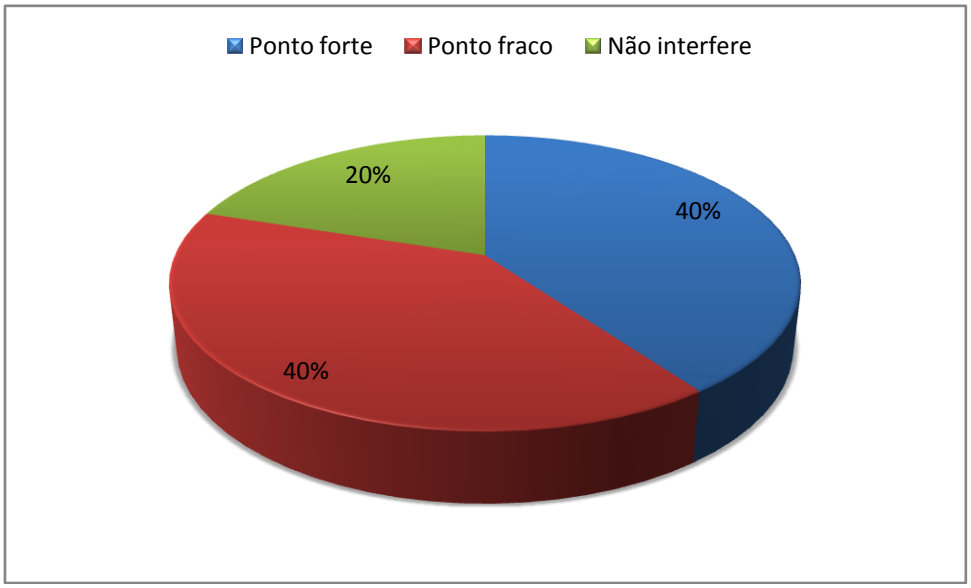

Gráfico 17 – Análise de risco das empresas estudadas.
Fonte: pesquisa de campo 2012.

Os métodos e técnicas da engenharia de *software* possibilitam produzir sistemas mais adequados às necessidades do cliente, contudo, defeitos no produto ainda podem ocorrer. Os erros podem afetar a usabilidade, confiabilidade, funcionalidade e segurança do sistema, impactando em prejuízos a ambas as partes: cliente e desenvolvedor (INTHURN, 2001).

O principal objetivo de gerir um processo de testes é descobrir e retirar erros com eficiência, melhorando a produtividade, qualidade e satisfação dos usuários. Sendo assim, os riscos devem ser tratados para que o processo transcorra de maneira planejada. Devem ser estabelecidas ações de contingência, (caso o risco aconteça) ou as ações preventivas (para evitar que ele aconteça) (MOREIRA e RIOS, 2006).

5 CONSIDERAÇÕES FINAIS

Este capítulo contém as informações finais das quais se chegou com o término desse estudo, bem como as sugestões e conclusões que se pode chegar com a sua realização.

5.1 CONCLUSÕES

De acordo com os dados coletados com essa pesquisa, os objetivos foram atingidos satisfatoriamente, pois se verificou que as empresas do mercado conquistense que fazem uso de metodologias ágeis para o desenvolvimento de software em geral utilizam o *Scrum* e o XP, com representação significativa. Com relação a percepção dos gestores com os pontos fortes e fracos no processo de desenvolvimento de software baseado em metodologias ágeis; se constatou que são mais eficientes devido a agilidade e a flexibilidade das mesmas.

Em se tratando das ferramentas utilizadas são o código coletivo e o *design* simples, além de reuniões rápidas para ver o andamento do projeto como um todo. Este estudo teve como limitação o tempo para a realização da pesquisa, tendo em vista que alguns participantes do estudo se recusaram a responder os questionamentos, ou demoraram para a entrega dos mesmos.

Para as empresas pesquisadas as metodologias ágeis ajudam na melhoria do desempenho, na conquista da satisfação junto ao cliente e as vantagens em relação a tradicional são relevantes ao ponto das mesmas preferirem a metodologia ágil. As vantagens são ainda mais significativas quando se observa que as equipes devem e estão sempre bem informadas, os códigos simples facilitam o trabalho em equipe e o tempo para o desenvolvimento de software é menor em relação a tradicional que também é pesada e cansativa.

A sugestão que fica para os trabalhos futuros é que seja realizada uma pesquisa com duas empresas que fazem uso da mesma metodologia para que se possa fazer um comparativo ente ambas, devido as peculiaridades da pesquisa é necessário maior demanda de tempo a ser utilizado para a realização de outros estudos relacionados ao mesmo tema. Outra sugestão que fica é que seja realizada

uma pesquisa no território brasileiro que utilizam as metodologias ágeis e observar como são utilizadas as mesmas e fazer um comparativo com relação à cidade de Vitória da Conquista-BA.

Este estudo tem informações relevantes e que podem seguramente ser utilizadas para a composição de um trabalho futuro com o mesmo tema.

REFERÊNCIAS

BECK, K. **Extreme Programming Explained**: Embrace Change. New York. 2000.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. Campus, 2006.

CARVALHO, L.C. **Análise de sistemas**, Rio de Janeiro: Livros Técnicos e Científicos. 1988.

COCKBURN, A. e HIGHSMITH, J. **Desenvolvimento de software ágeis**: The Business of Innovation. IEEE Computer 2001.

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa.** 3. ed. São Paulo: Atlas, 2004.

GRANDO. N. **metodologias ágeis**. Disponível em: <http://neigrando.wordpress.com/>. Último acesso em: agosto de 2012.

GUSTAVO. SITE. **Engenharia de software**. Disponível: <http://www.di.ufpb.br/gustavo/engenharia_software/> - Acesso em : 7 agosto. 2012.

HAYES, S. **An Introduction to Extreme Programming**. Kathovartech (2001). Disponível em <http://www.khatovartech.com>. Acessado em setembro de 2012.

INTHURN, C. **Qualidade e teste de software**. Santa Catarina: Visual Books. 2001.

JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. *UML* – **Guia do Usuário***.* Rio de Janeiro: Ed. Campus, 1999.

LOJKINE, J. **A revolução informacional**, São Paulo: Cortez. 1996.

LUNA, Sérgio Vasconcelos de. **Planejamento de Pesquisa:** uma introdução. 2.ed. EDUC Editora da PUC-SP. São Paulo, 2009.

MACORATTI, José C.. Modelos de processo de software. Disponível em:<http://www.macoratti.net/proc_sw1.htm>. Acesso em: 14 abr 2012.

MARTINS, José Carlos C. **Gerenciando projetos de desenvolvimento de software comPMI, RUP e UML**.3 Ed. Rio de Janeiro: Brasport. 2006.

MOLINARI, Leonardo. **Testes de software: produzindo sistemas melhores e mais confiáveis.** 3.ed. Cidade: Érica, 2006.

MOUNTAIN. Mountain Goat Software. 2011. Disponível em: <http://www.mountaingoatsoftware.com/scrum>. Acesso em: 28 mar. 2012.

MOREIRA, T.; RIOS, E. **Teste de software**. Rio de Janeiro: Alta Books, 2006.

OLIVEIRA, E. S. **Uso de Metodologias Ágeis no Desenvolvimento de Software**, Monografia apresentada no Programa de Pós-Graduação em Engenharia de Software da UFMG. 2003.

POLIT, D. F.; HUNGLER, B. P. **Fundamentos de pesquisa em Enfermagem**. 3. ed. Porto Alegre: Artes Médicas, 1995.

PRESSMAN, R. S. **Engenharia de software**. 3. ed. São Paulo: Makron Books, 1995.

PRESSMAN, R. S. **Software engineering a practitioner's approach.** 4th Edition New York: McGraw Hill, 1997.

_____. **Engenharia de Software**. Rio de Janeiro. McGraw-Hill, 2002.

_____. **Engenharia de Software**. 6.ed., 2006, McGraw-Hill, São Paulo. 2006.

RIBEIRO, Leonardo Rodrigues. **Um Processo de Desenvolvimento de Software:**Estudo de Caso Empresa Creativer. Trabalho de Conclusão de Curso, UESB. Disponível em: < http://www.uesb.br/computacao/ocurso/tccs/tccsengenharia/creativer.pdf > Acesso em: 21 de Abril de 2012.

SEBRAE. **Classificação das empresas no Brasil**. Disponível em: http://www.sebrae.com.br/uf/goias/indicadores-das-mpe/classificacao-empresarial. Acessado em 2 de setembro de 2012.

SILVA, Ana Eliza Pedroso, *et al.***Metodologias de Desenvolvimento de Sistemas – Métodos Ágeis**. In: UNIVERSIDADE FEDERAL DE SÃO CARLOS. São Carlos, SP – Brasil, 2006.

SOARES, Michel dos Santos. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software.** Artigo, UNIPAC.Disponível em: < http://wiki.dcc.ufba.br/pub/Aside/ProjetoBitecIsaac/Met._Ageis.pdf > Acesso em: 20 de Abril de 2012.

SOARES, Michel dos Santos. **Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software***.* Artigo, UNIPAC.Disponível em: < http://www.facecla.com.br/revistas/resi/edicoes/ed4artigo06.pdf > Acesso em: 21 de Abril de 2012.

SOMMERVILLE, Ian. **Software engineering**. 6. ed. Reading: Addison- Wesley, 2004.

SOUZA, C. B. **Autoria de Artefatos de Software**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, Brasil. 2008.

TELES, Vinícius Manhães. **Extreme Programming**. Novatec Editora Ltda, São Paulo. 2004.

TELES, Vinícius Manhães. **Extreme Programming**. Aprenda como encantar seus usuários desenvolvendo *software* com agilidade e alta qualidade. Ed. Novatec Editora Ltda: São Paulo, 2004.

VIEIRA, M. F. **Gerenciamento de Projetos de Tecnologia da Informação**. Rio de Janeiros. Campus 2003.

VINCENZI, A. M . R . **Subsídios para o estabelecimento de estratégias de teste baseadas na técnica de mutação.** Dissertação Mestrado em Ciências da Computação - ICMC-USP, São Carlos, São Paulo, 1998.

WIKIPEDIA. **A Enciclopédia livre**. Disponível em: <http://pt. wikipedia.org/wiki/> Acesso em: 18 agosto. 2012.

APÊNDICE –A

QUESTIONÁRIO

Estadual do Sudoeste da Bahia – UESB, requisito necessário para obtenção do título de Bacharel em Ciência da Computação.

TEMA: Desenvolvimento de Software: Aplicabilidade de metodologias ágeis no mercado de software de Vitória da Conquista-Ba.
Discente: Alan Moitinho Santiago
Professor: Gidevaldo Novais dos Santos

1. Nome da empresa: _____

2. Nome do pesquisado: _____

3. Área de atuação da empresa:
( ) Software comercial ( ) Software governamental ( ) Software industrial
( ) Outro(s) _____

4. Tempo de atuação no mercado de informática:
( ) Menos de 1 ano ( ) Entre 1 e 2 anos ( ) Entre 3 e 5 anos
( ) Entre 6 e 10 anos ( ) Entre 11 e 20 anos ( ) Acima de 20 anos

5. Principais públicos-alvo:
( ) Empresas locais ( ) Empresas regionais
( ) Empresas nacionais ( ) Empresas internacionais

6. Qual(is) tipo(s) de software é(são) desenvolvido(s) pela empresa?

Tipos	Sim	Não
Comércio eletrônico	( )	( )
Educação	( )	( )
Contábil	( )	( )
Páginas web	( )	( )
Jogos	( )	( )
Administração de saúde	( )	( )
Automação de motores	( )	( )
Automação bancária	( )	( )
Automação industrial	( )	( )
Automação comercial	( )	( )
Geoprocessamento	( )	( )
Cartográfico	( )	( )
Dispositivos Móveis	( )	( )
Estocástico	( )	( )
Outro: _____	( )	( )

7. Trabalham em maior frequência com que tipo de empresas?
( ) Microempresas
( ) Empresas de pequeno porte
( ) Empresas de médio porte
( ) Empresas de grande porte

8. Qual tipo de software é mais comercializado:

Tipos	Sim	Não
Comércio eletrônico	( )	( )
Educação	( )	( )
Contábil	( )	( )
Páginas web	( )	( )
Jogos	( )	( )
Administração de saúde	( )	( )
Automação de motores	( )	( )
Automação bancária	( )	( )
Automação industrial	( )	( )
Automação comercial	( )	( )
Geoprocessamento	( )	( )
Cartográfico	( )	( )
Dispositivos Móveis	( )	( )
Outro: _____	( )	( )

9. Composição da equipe de desenvolvimento:

( ) Própria ( ) Tercerizada ( ) Ambas

10. Quantidade de funcionários da empresa?

( ) Até 3 ( ) De 4 a 5 ( ) De 6 a 10
( ) De 11 a 30 ( ) Acima de 30

11. Grau de formação da equipe:

( ) Nível médio/técnico até 3 pessoas ( ) Nível médio/técnico acima de 3 pessoas
( ) Nível Superior até 3 pessoas ( ) Nível Superior acima de 3 pessoas
( ) Especialização até 3 pessoas ( ) Especialização acima de 3 pessoas
( ) Mestrado até 3 pessoas ( ) Mestrado acima de 3 pessoas
( ) Doutorado até 3 pessoas ( ) Doutorado acima de 3 pessoas

12. Quais cursos de graduação são ou já foram cursados pelos membros da equipe?

( ) Ciência da Computação ( ) Engenharia de Computação
( ) Sistemas de Informação ( ) Análises de Sistemas
( ) Outros: _____

13. Utiliza qual metodologia para desenvolvimento de software?

( ) Metodologia ágil ( ) Metodologia tradicional (maior produção de documentos)

As perguntas abaixo só serão preenchidas para aqueles que utilizam metodologias ágeis.

14. Qual metodologia é utilizada no desenvolvimento do software:

Tipos	Não utiliza	Pouco	Muito
XP - eXtreme Programming	( )	( )	( )
FDD - Feature Driven Development	( )	( )	( )
DSDM - Dinamic Systems Development Method	( )	( )	( )
ASD - Adaptative Software Development	( )	( )	( )
Scrum	( )	( )	( )
Crystal/Clear	( )	( )	( )

YP - easY Process	( )	( )	( )
Outra: _____	( )	( )	( )

15. Há quanto tempo utiliza metodologia ágil na empresa?
( ) Menos de 1 ano ( ) 1 a 3 anos ( ) 3 a 5 anos
( ) 5 a 7 anos ( ) 7 a 10 anos

16. Como foi a adaptação da equipe de desenvolvimento com a migração da metodologia tradicional para uma metodologia ágil?
( ) Fácil ( ) Normal ( ) Difícil

17. Quantidade de sistemas desenvolvidos com metodologias ágeis?
( ) Até 3 ( ) De 4 a 5 ( ) De 6 a 10
( ) De 11 a 30 ( ) Acima de 30

18. Quantidade de pessoas que atuam no desenvolvimento ágil da empresa?
( ) Até 3 ( ) De 4 a 5 ( ) De 6 a 10
( ) De 11 a 30 ( ) Acima de 30

19. O que levou a empresa a utilizar metodologias ágeis?
( ) Indicação de algum membro da equipe
( ) Palestras ou notícias relacionados ao tema
( ) A partir de alguma pesquisa a qual comprovou melhoria no desenvolvimento
( ) A partir de testes de várias metodologias para perceber qual melhor se enquadrava no perfil

20. Quais os fatores mais importantes para a utilização de metodologias ágeis na empresa (assinale a(as) alternativa(s) que se enquadra(m))?
( ) Redução do controle gerencial
( ) Ausência de planejamento antecipado
( ) Redução de riscos
( ) Equipe flexível a mudanças
( ) Eliminação de documentação
( ) Redução de custos operacionais
( ) Melhoria na manutenção do software
( ) Aumento de produtividade
( ) Troca de conhecimento entre o membros da equipe

21. Já testou mais de uma metodologia ágil na empresa?
( ) Sim ( ) Não
21.1 Se sim, quantidade: _____

21.2 Se sim, quais as metodologias testadas?
_____

22. Qual(is) as dificuldades para a implantação de metodologias ágeis em sua empresa (assinale a(as) alternativa(s) que se enquadra(m))?
( ) Equipe resistente à mudanças
( ) Colaboração com o cliente
( ) Capacitação insuficiente
( ) Complexidade do projeto
( ) Orçamento curto
( ) Outra: _____

23. Qual(is) as práticas de agilidade a empresa utiliza durante o processo de desenvolvimento do software (assinale a(as) alternativa(s) que se enquadra(m))?
( ) Refatoração ( ) Desenvolvimento guiado por testes
( ) Integração contínua ( ) Código coletivo
( ) Cliente presente ( ) Código padronizado
( ) Jogo do planejamento ( ) Design simples

( ) Programação em pares ( ) Metáfora
( ) Stand Up Meeting(reunião rápida a cada manhã) ( ) Ritmo sustentável
( ) *Releases* curtos

24. Qualidade de software desenvolvidos com metodologias ágeis:
( ) Péssima ( ) Ruim ( ) Regular ( ) Boa ( ) Ótima

25. Como se dá a comunicação entre os membros da equipe com a utilização de metodologias ágeis?
( ) Baixa interatividade ( ) Mediana ( ) Alta interatividade

26. Comparando-se com a metodologia tradicional como está sendo o prazo de entrega depois da utilização de metodologias ágeis:
( ) Melhor do que a metodologia tradicional
( ) Pior do que a metodologia tradicional
( ) Continua igual

27. Com relação aos custos, as metodologias ágeis:
( ) Reduziu custos ( ) Manteve o mesmo ( ) Aumentou custos

28. No que diz respeito aos padrões de qualidade do software:
( ) Melhorou a qualidade ( ) Manteve o padrão ( ) Diminuiu a qualidade

29. A empresa realiza inspeção para avaliação da qualidade do software?
( ) Sim ( ) Não

30. A empresa utiliza quais tipos de teste no desenvolvimento ágil (assinale uma ou mais opções)?
( ) Teste unitário
( ) Teste de sistema
( ) Teste de integração
( ) Teste de auditoria
( ) Teste de aceitação

31. Com relação à confiança da equipe de desenvolvimento e do cliente:
( ) Aumentou a confiança da equipe e do cliente
( ) Aumentou a confiança da equipe e diminuiu a confiança do cliente
( ) Diminuiu a confiança da equipe e aumentou a confiança do cliente
( ) Diminuiu a confiança da equipe e do cliente
( ) Manteve da mesma forma a confiança da equipe
( ) Manteve da mesma forma a confiança do cliente

32. Como o cliente vê o processo de informalidade no levantamento de requisitos?
( ) Demonstra amadorismo
( ) Demonstra insegurança
( ) Demonstra segurança
( ) Demonstra agilidade
( ) Demonstra satisfação
( ) Outro: _____

33. Como se dá a programação em pares na empresa:
( ) Há facilidade na interação entre os membros mais experientes com os menos experientes
( ) Não interação entre os membros mais experientes com os menos experientes
( ) Há dificuldade de interação no inicio do processo, havendo adaptação ao longo do processo
( ) Outro: _____

34. Como é visto por parte da equipe de desenvolvimento a questão da análise de riscos?
( ) Ponto fraco ( ) Ponto forte ( ) Não interfere no processo

35. Como é visto por parte da equipe de desenvolvimento a questão da análise de planejamento?

( ) Ponto fraco ( ) Ponto forte ( ) Não interfere no processo

36. Com relação a refatoração de códigos, como é interpretada pela equipe no momento da apresentação do mesmo?
( ) Demonstra amadorismo
( ) Demonstra insegurança
( ) Demonstra segurança
( ) Demonstra agilidade
( ) Demonstra satisfação
( ) Outro: _____