FACULDADES INTEGRADAS DE CARATINGA

FACULDADE DE CIÊNCIA DA COMPUTAÇÃO

GESTÃO DE TEMPO: UM ESTUDO SOBRE A PRODUTIVIDADE DOS DESENVOLVEDORES DE SOFTWARE

FABIANO LOPES VIANA

Caratinga
2010

Fabiano Lopes Viana

GESTÃO DE TEMPO: UM ESTUDO SOBRE A PRODUTIVIDADE DOS DESENVOLVEDORES DE SOFTWARE

Monografia apresentada à banca examinadora da Faculdade de Ciência da Computação das Faculdades Integradas de Caratinga, como requisito parcial para obtenção do título de bacharel em Ciência da Computação, sob orientação da professora MSc. Fabrícia Pires Souza Tiola.

Caratinga

2010

Fabiano Lopes Viana

GESTÃO DE TEMPO: UM ESTUDO SOBRE A PRODUTIVIDADE DOS DESENVOLVEDORES DE SOFTWARE

Monografia submetida à Comissão examinadora designada pelo Curso de Graduação em Ciência da Computação como requisito para obtenção do grau de Bacharel.

_____
Prof. Msc. Fabrícia Pires Souza Tiola
Faculdades Integradas de Caratinga

_____
Prof. Filipe Fernandes
Faculdades Integradas de Caratinga

_____
Prof. Paulo Eustáquio
Faculdades Integradas de Caratinga

Caratinga, 14/12 /2010

AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer a Deus, pois sem Ele nada em nossas vidas seria concretizado; à meus pais que me apoiaram deste o início desta luta e nunca me deixaram desistir; à minha noiva que apareceu em minha vida na conclusão do curso e está me dando forças e à minha orientadora que está contribuindo muito para a concretização deste trabalho.

RESUMO

O fator tempo de desenvolvimento nos projetos de software tem sido uma grande preocupação das empresas desenvolvedoras e dos clientes que contratam tais serviços porque, na maioria das vezes este fator, tempo de desenvolvimento, não é cumprido como determinado.

Tem-se como objetivo deste trabalho, além da apresentação de algumas técnicas e práticas da engenharia de software relacionadas com este fator tempo, a exposição de uma pesquisa com profissionais da área de desenvolvimento de software com o intuito de entender o que estes profissionais pensam a respeito do tema na tentativa de se descobrir a causa deste problema em questão.

Para a realização da pesquisa foi disponibilizado um questionário na forma de formulário com acesso on-line. Este questionário foi divulgado entre os profissionais que se enquadram dentro do universo da pesquisa e com isso foi devidamente respondido. Os resultados da pesquisa apresentam que, dentro do universo pesquisado, o nível de conhecimento dos profissionais não se difere muito com relação ao cargo, porém, percebe-se que isto não é refletido na prática, dentro das empresas, por escopo de atribuições do cargo e pelas políticas adotas pelas empresas na qual os respondentes trabalham.

Como conclusão do trabalho é defendido que com planejamento, organização e políticas de incentivos à produtividade mais claras dentro das empresas, além da adoção de uma boa ferramenta *CASE*, que significa Engenharia de Software Auxiliada por Computador, mesclada com a utilização de metodologias disponibilizadas pela engenharia de software essas empresas iriam ter um ganho significativo quanto à produtividade e qualidade mas, principalmente iriam minimizar um dos maiores problemas por elas enfrentadas que é o fator tempo que, como já foi dito, na maioria dos projetos não é cumprido como determinado ocasionando uma reação em cadeia que prejudica essas empresas.

Palavras-chave: crise do software, desenvolvedor de software, tempo de desenvolvimento, medida, produtividade, CASE

ABSTRACT

The time factor in the development of software projects has been a major concern of developing companies and clients who hire such services because, in most cases, this factor, development time is not met as specified.

It has the objective of this work, and presents some techniques and practices of software engineering related to this time factor, the exposure of a survey of professionals in software development in order to understand what these professionals think the regarding the subject in an attempt to discover the cause of this problem.

To conduct the study was provided in the form of a questionnaire form with online access. This questionnaire was circulated among the professionals who fall within the realm of research and this was duly answered. The survey results show that within the group studied, the level of professional knowledge is not much different with respect to the position, however, one realizes that this is not reflected in practice, within companies, by scope of duties of the position and policies adopted by companies in which respondents work.

As a conclusion it is argued that with planning, organization and policies clearer incentives within firms, in addition to the adoption of a good CASE tool, which stands for Software Engineering Computer-Aided, mixed with the use of methodologies available for engineering software those companies would have a significant gain in terms of productivity and quality, but mainly would minimize one of the biggest problems faced by them which is the time factor, as has been said, most projects are not completed as causing a certain reaction chain that affect these companies.

Keywords: software crisis, software developer, development time, measure productivity, CASE

LISTA DE SIGLAS

CASE - Computer-Aided Software Engineering
COM - Critical Path Method
DER - Diagramas de Entidade-Relacionamento
DFD - Diagramas de Fluxo de Dados
EJB - Enterprise Java Beans
FPs - Pontos-por-Função
IA - Inteligência Artificial
JAD - Joint Application Design
LOC - Linhas de Código
MPS-Br - Melhoria do Processo do Software Brasileiro
OO - Orientado a Objetos
PERT - Program Evaluation and Review Technique
PRO - Prototipação
RAD - Rapid Application Development
SA - Análise Estruturada
SCM - Software Configuration Management
SD – Projeto Estruturado
SIM – Simulação
TI – Tecnologia da Informação

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O mercado de software está em constante evolução desde o final da década de 60 até os dias atuais, porém juntamente com esta evolução o que está sempre presente neste contexto é o fator tempo, que tem sido motivo de muita preocupação tanto para gerentes de projeto quanto para os clientes. Preocupação porque na maioria dos projetos de software o tempo de desenvolvimento não é cumprido como determinado, ocasionando uma reação em cadeia que leva ao aumento do orçamento pré-determinado.

De acordo com Ribeiro (2006) em 2004 foi realizada uma pesquisa, também citada no referencial teórico deste trabalho, obtendo como resultado que o fator tempo está entre os cinco maiores fatores de insucesso dos projetos, sendo responsável por dez por cento do universo pesquisado.

Uma das motivações para o desenvolvimento deste trabalho é a crescente demanda por softwares no mercado. Sendo que o tempo de desenvolvimento é um fator relevante para a escolha de qual empresa irá desenvolver o software. A empresa que tiver em seu histórico uma maior porcentagem de entrega de softwares no tempo pré-determinado irá se sobressair àquela que tiver uma menor porcentagem.

O objetivo deste trabalho consiste em apresentar algumas técnicas de engenharia de software existentes que tem em seu foco principal a redução do tempo de desenvolvimento dos projetos de software, assim como será apresentado também o que pensam os profissionais desta área a respeito deste tema objetivando ser uma tentativa de descobrir a causa de tal problema, que seria o não cumprimento dos prazos na maioria dos projetos de software.

Existem vários trabalhos e iniciativas que tratam deste tema. Ribeiro (2006) desenvolveu, em sua dissertação para obtenção do título de Mestre em Engenharia da Escola Politécnica da Universidade de São Paulo, um roteiro para a redução do tempo no desenvolvimento de projetos de software onde reuniu e organizou as técnicas e as práticas de engenharia para tal finalidade que seria a minimização do problema relacionado ao fator tempo.

Neste trabalho o referencial teórico traz um aparato da literatura a respeito deste tema, apresentando algumas técnicas e iniciativas para a redução do tempo. Na metodologia é mostrado como os dados foram coletados e uma breve explicação de como os dados foram analisados, a forma adotada para tais coletas de dados foi através de formulário anônimo disponibilizado on-line. E no final do trabalho é apresentado os resultados obtidos com a coleta dos dados, reunindo de uma forma genérica as respostas para o questionário.

2 REFERENCIAL TEÓRICO

São apresentadas nas próximas seções, até o capítulo 3, textos referentes ao tema do trabalho em questão. São textos referenciados de algumas das literaturas existentes, começando pela “crise do software” e logo em seguida trazendo informações sobre o desenvolvedor de software, o tempo de desenvolvimento juntamente com algumas metodologias relacionadas com este fator assim como iniciativas para a redução do tempo, depois são apresentadas algumas técnicas de medição de software e em seguida é abordado a produtividade no desenvolvimento e sua ligação com o tempo, terminando o referencial teórico tem-se informações a respeito das ferramentas *CASE*.

2.1 A CRISE DO SOFTWARE

Este termo “crise do software” ilustra muito bem o passado e o presente no que se diz respeito ao processo de desenvolvimento de software. Porém, como essa crise emergiu no final dos anos 60 este termo foi muito utilizado durante a década de setenta.

Para ajudar no entendimento da origem da crise do software pode-se tomar como base o histórico de outros tipos de indústrias da Idade Média. Pensando na fabricação de sapatos, antes da revolução industrial os mesmos eram confeccionados de forma individual, ou seja, os sapateiros faziam cada par de sapatos de forma única para cada cliente pegando desde a matéria prima até o produto final que era o sapato. E é justamente neste ponto em que a indústria do software se assemelha. Pode-se então pensar que cada software era desenvolvido exclusivamente para um determinado cliente, primeiramente porque as grandes quantidades de técnicas de desenvolvimento de software que surgiram nas últimas décadas não estão totalmente maduras e consolidadas. Consequentemente existe uma tendência muito forte do desenvolvedor em não aproveitar o material já produzido; e piorando essa situação além de entregar o produto (software) mal

documentado a maior parte do conhecimento envolvido no desenvolvimento existe apenas com os desenvolvedores.

Sob um outro ponto de vista, Pressman (2006) relata que a palavra “crise” não seria o termo ideal para demonstrar a real situação, o que tem-se realmente pode ser algo bem diferente.

> A palavra “crise” é definida no (*Webster’s Dictionary*)[1] como “um ponto decisivo no curso de algo; momento, etapa ou evento decisivo ou crucial”. Contudo, para o software, não tem havido nenhum “ponto decisivo”, nenhum “momento, etapa ou evento decisivo ou crucial”; somente uma mudança lenta e evolucionária. Na indústria do software, temos tido uma “crise” que nos acompanha há quase 30 anos, e essa é uma contradição, em termos. (Pressman, 2006)

Para Pressman (2006) o que realmente ocorre é uma aflição crônica. A definição de aflição descrita no *Webster’s Dictionary* está como “algo que causa dor ou sofrimento”, porém o que realmente importa nesse termo é a definição do adjetivo crônica, “que dura um longo tempo ou retorna frequentemente; que continua indefinidamente”. Segundo este autor esse termo, sim, expressa melhor o que realmente a computação está enfrentando ao longo desses quase 30 anos na indústria de desenvolvimento de software, para ele seria uma aflição crônica e não uma crise.

Foram realizadas muitas pesquisas em âmbito internacional apresentando como resultado que, mais da metade dos projetos de desenvolvimento de software resultaram em fracasso, apresentando problemas com relação ao escopo, prazo, orçamento e qualidade do software. Em uma pesquisa realizada pelo (*Standish Group*)[2], a questão do tempo foi classificada como um dos cinco maiores fatores de insucesso dos projetos, sendo responsável por dez por cento dos problemas no universo pesquisado, de acordo com Ribeiro (2006).

Os problemas que originaram essa crise tinham relacionamento direto com a forma de trabalho das equipes. Eram problemas que não se limitavam a "sistemas que não funcionam corretamente", mas envolviam também dúvidas sobre como desenvolver e manter um volume crescente de software e ainda estar preparado para as futuras demandas. Essencialmente, eram sintomas provenientes do pouco entendimento dos requisitos por parte dos desenvolvedores, somados às técnicas e

[1] Um dos primeiros dicionários desenvolvidos nos Estados Unidos.
[2] Realiza investigações e sondagens do mercado tecnológico para utilizadores e fornecedores na área de OLTP (On-Line Transaction Processing) e ambientes cliente/servidor.

medidas pobres aplicadas sobre o processo e o produto, além dos poucos critérios de qualidade estabelecidos até então, conforme argumentado por Pressman (2006).

São muitos os problemas que giram em torno desse fato que alguns chamam de crise do software, porém os gerentes responsáveis pelo desenvolvimento de um projeto de software concentram seus esforços em três questões principais chamada de “primeiro plano”:

1. as estimativas de prazo e de custo frequentemente são imprecisas;
2. a produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços;
3. a qualidade de software às vezes é menos que adequada.

Observando essas três questões o primeiro ponto a se analisar é o porquê isso acontece, porque que a indústria de desenvolvimento de softwares enfrenta problemas tão graves e bastante complexos de se resolverem. Porém ao se observar e estudar as causas percebe-se que não é difícil de entender o porquê desses problemas, são simplesmente a manifestação mais visível de outras dificuldades do software. De acordo com Pressman (2006), entre as considerações destaca-se:

- Não se dedica tempo necessário para coletar dados suficientes a respeito do processo de desenvolvimento de software. Consequentemente com poucos dados históricos como guia, as estimativas têm sido “a olho”, com resultados previsivelmente ruins. Sem nenhuma indicação sólida de produtividade, não podemos avaliar com precisão a eficácia de novas ferramentas, métodos ou padrões.
- Um projeto considerado “concluído” para os desenvolvedores muito frequentemente causam insatisfação por parte do cliente. Isso porque a comunicação entre o cliente e os desenvolvedores, que deveria ser sólida e constante, na maioria das vezes é muito fraca, fazendo com que o projeto de desenvolvimento seja levado apenas com um vago indicio das exigências do cliente, consequentemente nunca atingindo o produto desejado e pensado pelo cliente.
- Piorando a situação dos desenvolvedores, a qualidade do software muitas vezes é suspeita. Só recentemente é que se passou a entender a verdadeira importância dos testes de software sistemáticos e tecnicamente completos. Somente agora estão começando a surgir conceitos quantitativos sólidos de confiabilidade e garantia de qualidade de software.

- Depois de concluído o projeto de desenvolvimento, o software existente poderá ser difícil para manter. Na maioria das vezes a tarefa de manutenção do software aumenta o orçamento a ele destinado. Isso porque a capacidade de manutenção do software não foi enfatizada como um critério importante para a aceitação e desenvolvimento do suposto software.

Parece que os erros estão por toda parte, como uma epidemia que não conseguem controlar depois de décadas de muito trabalho e muita pesquisa. Porém cada um dos problemas descritos anteriormente podem ser corrigidos. A chave para essa suposta correção seria uma abordagem de engenharia ao desenvolvimento de software aliada a uma contínua melhoria de técnicas e ferramentas. Porém na informática as dificuldades começam durante as etapas iniciais do projeto, delimitar o escopo está longe de ser uma tarefa trivial. Uma mudança nas necessidades declaradas por um usuário pode repercutir em vários elementos da estrutura do programa. Embora os desenvolvedores tenham projetado a solução, no início do projeto raramente terão conhecimento dos algoritmos que efetivamente resolvem o problema.

Para ajudar no entendimento, apresenta-se uma comparação entre o projeto de uma ponte e de um software.

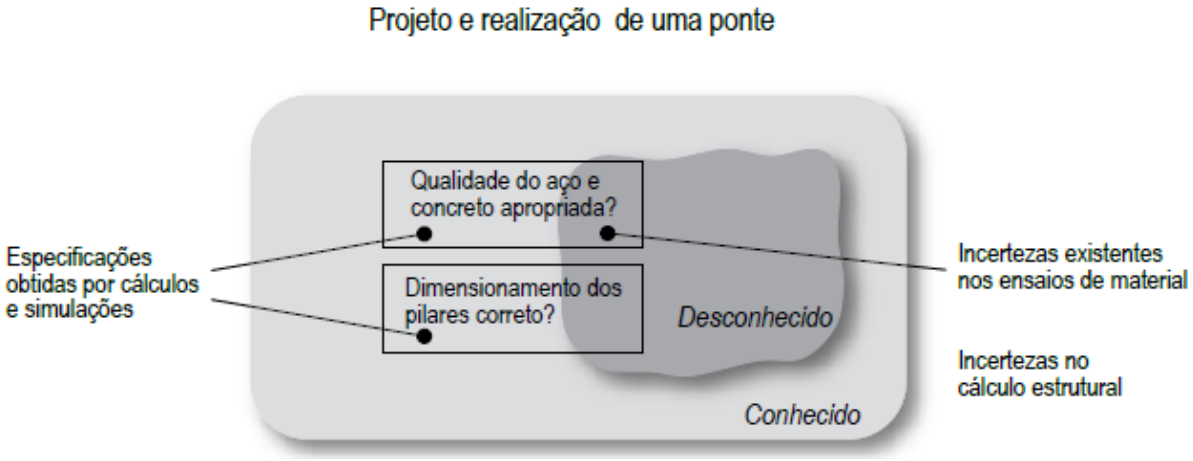

FIGURA 1 – Zonas de sombra em um projeto de engenharia
Fonte: Koscianski (2007) PÁG. 23

Na FIG. 1, observa-se que, para se construir uma ponte os processos de cálculo de estruturas, correção de inclinações, preparação do terreno e pavimentação são conhecidos. Podem ocorrer algumas surpresas como descobrir que o solo de um determinado ponto da estrada não é como se planejou, mas, em geral todas as etapas são cumpridas da mesma maneira.

É durante o projeto da ponte que existem maiores questões em aberto e soluções em engenharia devem ser desenvolvidas. Por exemplo, antes do projeto fica difícil saber qual traçado será mais confortável, seguro e econômico para os futuros usuários. Logo, terminado o projeto, já existem respostas para maioria das questões que dizem respeito à realização. Por exemplo, os carros que irão trafegar não mudarão de largura e nem comprimento repentinamente; a quantidade de veículos e, consequentemente, o peso máximo a ser suportado são calculados previamente e através de software de simulação. Alguns elementos, como por exemplo a força do vento, podem ser superdimensionados para garantir uma maior margem de segurança. Raramente um pilar deverá ser deslocado cinco centímetros depois que a construção já tenha sido iniciada, conforme argumentado por Koscianski (2007).

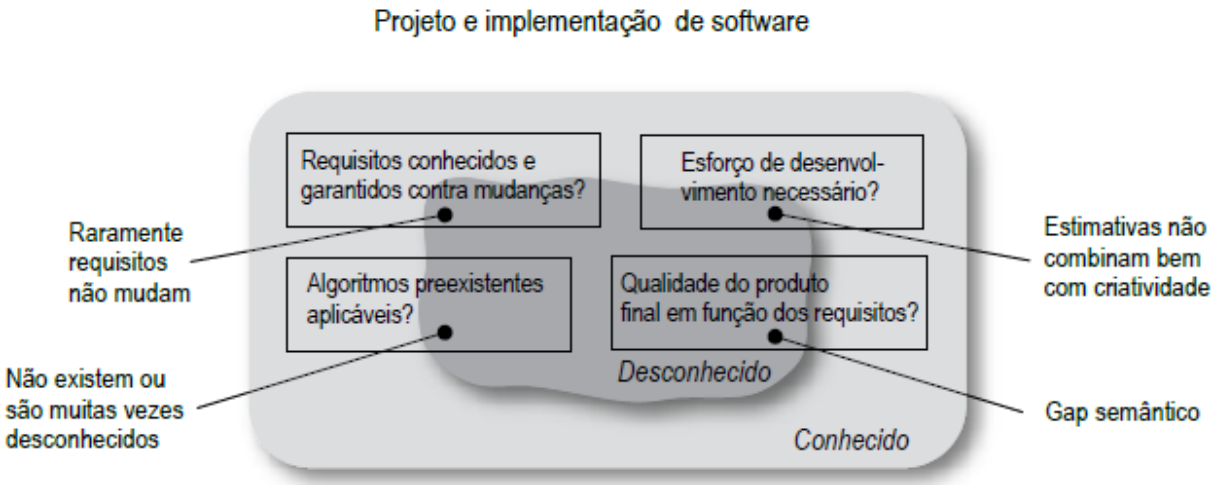

FIGURA 2 – Zonas de sombra em um projeto de software
Fonte: Koscianski (2007) PÁG. 23

Com a FIG.2, faz-se uma analogia ao primeiro exemplo, que traz uma análise no projeto de desenvolvimento de software onde pode-se dizer que essa zona de sombra que envolve os fatores desconhecidos é bem mais abrangente. O termo “Gap semântico” representa a dificuldade recorrente no processo de modelar o mundo real. De acordo com Koscianski (2007) os desenvolvedores de software sabem muito bem como é comum ter que resolver problemas semelhantes a encurtar ou alongar uma ponte “apenas alguns centímetros” alguns dias antes de sua inauguração.

Visto que embora não bastassem as dificuldades inerentes ao caráter mutável do software, ainda existem as pessoas que trabalham com ele, que devido ao

trabalho intelectual envolvido no desenvolvimento podem criar obstáculos dificultando assim o andamento do projeto.

> Todos nós resistimos à mudança. Entretanto, é realmente irônico que, enquanto o potencial de computação (hardware) experimenta enormes mudanças, as pessoas da área de software responsáveis pelo aproveitamento desse potencial muitas vezes se oponham à mudança quando ela é discutida e resistam à mudança quando ela é introduzida. Talvez esta seja a causa real para alguns de nossos problemas com software. (Pressman, 2006)

Talvez um dos maiores desafios enfrentados seja a tentativa de conciliação entre a necessidade de disciplina, que é fundamental para garantir uma certa previsibilidade de resultados, e o caráter relativamente aleatório da criação de soluções. Por esse motivo, na tentativa insensata de se conseguir minimizar essa busca por “inventar soluções”, várias metodologias têm sido desenvolvidas, testadas e adaptadas há décadas.

Em harmonia com as metodologias, um outro aspecto que vem tentando dar sua contribuição para amenizar o problema é o desenvolvimento de tecnologias e ferramentas. Para Koscianski (2007) várias tarefas podem ser automatizadas, garantindo assim uma certa uniformidade de execução de tais tarefas e ao mesmo tempo com essa automatização de tarefas a carga de trabalho exercida sobre as pessoas será amenizada, servindo até mesmo de motivação, quando uma tarefa é executada por uma ferramenta há menos chance de o resultado ser diferente.

Uma das alternativas mais utilizadas pelas empresas para eliminar os efeitos destas dificuldades é a aplicação de normas e modelos de qualidade de software que descrevem diretrizes, processos e guias de avaliação, visando a aplicação de técnicas e práticas de engenharia para a obtenção de melhoria em seus processos.

“A única relação entre a Engenharia de software e a noção de valor significa que essa profissão trata o software como um produto”, de acordo com Spinola (2007).

2.2 O DESENVOLVEDOR DE SOFTWARE E SUAS CARACTERÍSTICAS

O conhecimento relacionado à programação de computadores não é nato. Como referenciado por Peters (2001), foram identificados dois estilos de aprendizagem por parte dos novatos na arte de aprender a programação de computadores, que seria a aprendizagem por compreensão e a aprendizagem operacional.

Na primeira tem-se que o aprendiz entende perfeitamente o arranjo geral do programa, ou seja ele entende exatamente como será o programa, o que se está pedindo para ser desenvolvido, porém ele não tem a menor idéia de como fazer, de como desenvolver, ou seja ele não sabe quais as técnicas necessárias para o desenvolvimento propriamente dito. Pode-se dizer que basicamente esse tipo de aprendiz está interessado em compreender e não em fazer.

No segundo tipo de aprendizagem, que seria a aprendizagem operacional, pode-se dizer que é o inverso da aprendizagem por compreensão; ou seja, o aprendiz sabe as técnicas necessárias para o desenvolvimento mas não entendeu o contexto geral do programa, então esse tipo de aprendiz estaria mais interessado em fazer programação do que necessariamente compreendê-la.

Como abordado por Peters (2001) é importante ressaltar que os dois tipos de aprendizes são necessários e úteis dentro da organização das equipes de desenvolvimento de software; uma vez que os aprendizes por compreensão teriam um bom desempenho na especificação de requisitos, resolução de problemas e engenharia reversa durante a manutenção de software, e não do projeto; já os aprendizes operacionais poderiam ter um melhor desempenho no projeto de software propriamente dito, assumindo a implementação de requisitos, além disso, foi descoberto que esse tipo de aprendiz está mais apto a aprender uma linguagem de programação.

Saindo do contexto de aprendiz e assumindo ser um desenvolvedor de software este profissional assume a partir de então o perfil que melhor se adapta ao projeto ou a organização, perfil de aprendiz por compreensão ou operacional.

Existem muitos fatores que atrapalham o bom desempenho dos programadores, um fator muito interessante e bastante agravante com relação a este desempenho seria o fator humano.

Esses fatores humanos podem ser divididos entre fatores individuais e organizacionais. Como discutido por Sandhof (2004), vários fatores individuais e organizacionais relacionados aos seres humanos são influenciadores de suas atividades rotineiras. Como forma de identificar os fatores condicionantes presentes do dia-a-dia de trabalho de profissionais da área de software, alguns fatores foram objeto de estudo e designados como relevantes para o processo de desenvolvimento de software. Como exemplo de algum dos fatores tem-se:

- Organizacionais: comunicação entre os profissionais, o relacionamento, o controle sobre os processos do desenvolvimento do projeto, etc.
- Individuais: dificuldade do profissional no desempenho de tarefas passadas por seus superiores, problemas pessoais (financeiros, profissionais, saúde), carga de trabalho que poderá estar um pouco acima do limite do profissional, etc.

Algumas características, ou dicas, podem ajudar um programador a ser um melhor profissional. Como por exemplo; tentar ser autodidata, ou seja, tentar exercitar sua capacidade de aprender por si só; saber buscar informação, um autodidata que sabe buscar informação terá seu desempenho como programador de ótimo a excelente; em conjunto com isso mais algumas características que podem ajudar são:

- assumir responsabilidades e procurar corrigir os erros é muito importante;
- aceitar opiniões pode ajudar mas procurar por sua própria solução aprimora e evolui os conhecimentos;
- focar na solução, em vez de focar nos problemas, para tentar evitar que novos problemas surjam;
- e por ultimo, documentar e escrever códigos de maneira que os outros profissionais possam entender.

O desenvolvedor deve se preocupar e argumentar, antes que o desenvolvimento do projeto seja iniciado, a respeito do tempo de desenvolvimento que na maioria das vezes é causa certeira de insatisfação por parte dos clientes de software.

2.3 O TEMPO NO DESENVOLVIMENTO DE SOFTWARE

Por serem vários os problemas enfrentados no desenvolvimento de um projeto de software este trabalho será direcionado a um dos fatores considerado um dos mais relevantes, pode-se dizer que o fator tempo é o primeiro grão da neve que se transformou em uma enorme bola.

Todo o desenvolvimento de software segue uma determinada programação de atividades para sua realização, mas, obviamente nem todos os programas são criados igualmente. Pressman (2006) aborda algumas questões que levam a uma reflexão ajudando no entendimento desses tais problemas que cercam o desenvolvimento.

> A programação se desenvolve por si mesma ou foi planejada anteriormente? O trabalho foi feito sem critérios ou um conjunto de tarefas bem definidas foi identificado? Os gerentes se concentraram somente na data final de entrega ou um curso bem definido foi identificado e monitorado para garantir que o prazo final seja cumprido? O progresso foi medido por "Nós já o fizemos?" ou um conjunto de marcos de referência uniformemente espacejados foi estabelecido? (Pressman, 2006)

Por tudo isso é que se faz importante a realização do planejamento do desenvolvimento do projeto de software. É garantido que com um bom planejamento o tempo de desenvolvimento será reduzido, ou pelo menos não passará do limite de prazo pré-determinado. Para o gerenciamento do tempo de desenvolvimento a determinação de um cronograma poderá ser de muita importância.

Como referenciado por Silva (2008) estudos realizados pelo *Standish Group* apontam que 88% dos projetos apresentam atrasos no cronograma, sendo que a média do atraso em relação ao cronograma inicial é 222%. De acordo com Silva (2008) o problema com o prazo de execução é que está essencialmente ligado a uma previsão feita. Estas previsões são feitas durante a fase de planejamento do projeto, ou seja, antes da execução efetiva do projeto. Sabe-se que as previsões estão sujeitas a não ocorrerem 100% da forma esperada e que o acerto total é pouco provável de ocorrer. Deve-se procurar fazer previsões com altas probabilidades de ocorrer, visando obter os maiores índices de acertos possíveis. Uma situação é estimar quinze dias para execução de uma atividade e esta vir a ser

realizada em um, dois ou três dias a mais; bem diferente seria levar, por exemplo, 25 a 30 dias ou mais para executá-las.

Para Silva (2008) quanto mais audaciosos forem os prazos estimados em um projeto maior será a possibilidade de haverem atrasos. Portando deve-se evitar considerar no planejamento prazos excessivamente justos. Um equívoco bastante comum quando se estima os prazos é quanto à disponibilidade considerada para os recursos do projeto. Na prática, verifica-se que o tempo real de produção, em um dia normal de trabalho, é em torno de 60% a 75%, de acordo com Silva (2008). Diversas atividades pessoais, fisiológicas e profissionais (reuniões, treinamentos, etc.), dentre outras, impedem uma dedicação integral das pessoas nos projetos.

A utilização de metodologias eficientes e boas práticas são imprescindíveis para se atingir um índice satisfatório no cumprimento de prazos nos projetos.

2.3.1 Cronograma

Uma vez coletada todas as informações relacionadas as atividades a serem desenvolvidas, e dos profissionais alocados para o projeto, deve-se então definir o cronograma. Primeiramente as prioridades das atividades do desenvolvimento devem ser revisadas, e aprovadas pelo cliente. Consequentemente definidas as prioridades, deve-se relacionar as atividades com os profissionais alocados e capacitados para o desenvolvimento da atividade. Esta relação, que é o cronograma, deve ser divulgada para toda a equipe de desenvolvimento, pois existem atividades que são dependentes de outras atividades. De acordo com Haufe (2001) um cronograma pode ser apresentado por meio de uma tabela, relacionando a atividade com o profissional e o período no qual ele estará alocado para o seu desenvolvimento.

Segundo Pressman (2006) a determinação de um cronograma para projetos de desenvolvimento de software pode ser vista a partir de duas perspectivas bem diferentes. Na primeira perspectiva uma data final para a entrega de um sistema baseado em computador já foi estabelecida e é irrevogável. Então todo o esforço deve ser distribuído dentro daquele prazo estabelecido e irrevogável. Na segunda presume-se que limites cronológicos aproximados tenham sido discutidos, porém a

organização de engenharia de software é que irá estabelecer a data final, fazendo com que o esforço seja distribuído de maneira que se possa tirar o melhor proveito dos recursos; após uma cuidadosa análise é que a data final é definida. De acordo com Pressman (2006), infelizmente, a primeira perspectiva é encontrada bem mais frequentemente do que a segunda.

Faz-se relevante frisar que tão importante quanto elaborar o cronograma é o acompanhamento de sua execução, de seu andamento, a fim de se detectar, evitar, desvios e no caso de não conseguir essa detecção será necessário a ativação de ações corretivas para que o projeto volte ao cronograma planejado, como argumentado por Barbosa (2006).

2.3.2 Rede de atividades e diagrama de barras

De acordo com Sommerville (2003) a programação de um projeto é normalmente representada como um conjunto de diagramas, mostrando a estrutura analítica do trabalho, dependências da atividades e alocação de pessoal. O modelo de cronograma citado por Sommerville (2003) é o diagrama de barras e as redes de atividades, que não passam de notações gráficas para ilustrar a programação do projeto. Os diagramas de barras mostram quem é responsável por cada atividade e para quando está programado o início e o término da atividade. As redes de atividades mostram a dependência entre as diferentes atividades que constituem o projeto.

Na rede de atividades primeiramente cria-se uma tabela com um conjunto de atividades mostrando suas durações e suas interdependências. Logo em seguida é criado um diagrama, utilizando uma ferramenta de gerenciamento de projeto, representando a rede que mostrará a seqüência das atividades considerando as dependências e as durações estimadas. Essa rede mostra quais atividades podem ser realizadas em paralelo e quais devem ser executadas em seqüência, devido à dependência em relação a uma atividade anterior.

O diagrama de barras, também chamado de diagrama de Gantt em homenagem ao seu inventor, é uma maneira alternativa de representar as informações sobre a programação do projeto. Ele mostra a amplitude de um possível

atraso na programação com uma barra sombreada, mostrando que existe alguma flexibilidade na data de conclusão dessas atividades. Então se uma atividade não for concluída a tempo, o caminho principal não será afetado até o fim do período marcado pela barra sombreada. A FIG. 3, mostra em detalhes este diagrama gerado pela ferramenta *CASE – Computer-Aided Software Engineering, Gemetrics*.

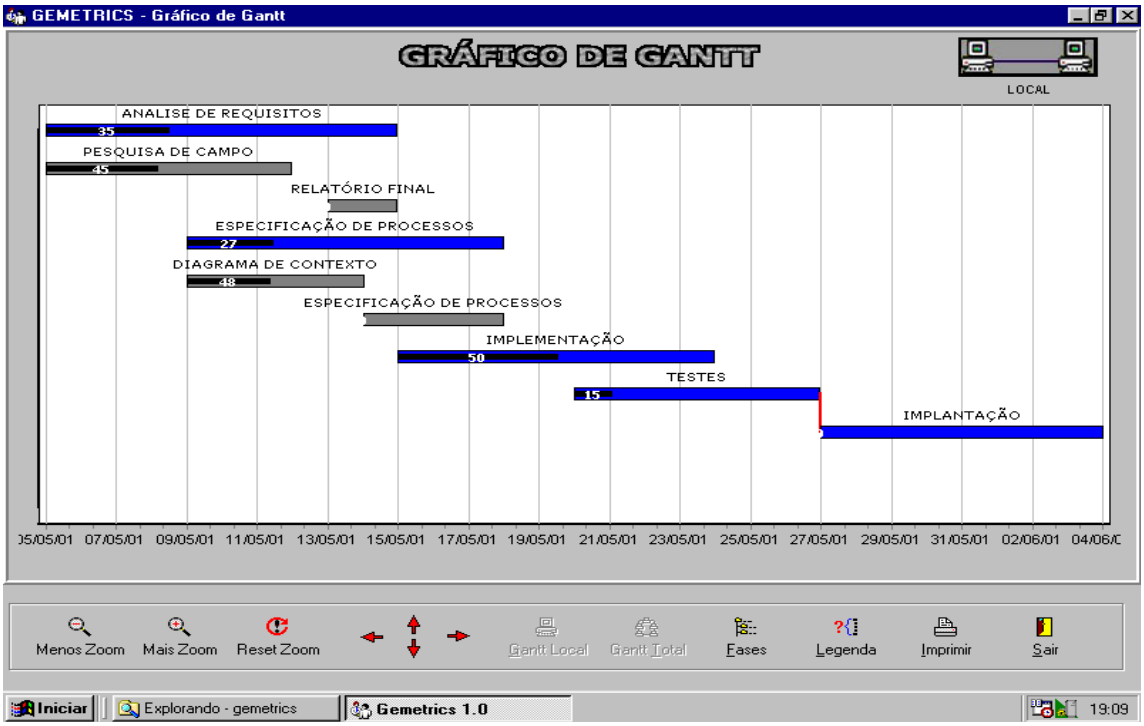

FIGURA 3 – Diagrama de Gantt
Fonte: Vavassori (2001)

Neste diagrama, de acordo com Vavassori (2001), tem-se que, as barras em azul correspondem as atividades do projeto, as barras pretas (em cada atividade) ao percentual de conclusão da atividade. As barras em cinza correspondem as sub-atividades. As linhas vermelhas correspondem a vínculos entre tarefas.

Outras considerações importantes devem ser feitas, como por exemplo, a alocação de recursos e de pessoal para a realização das atividades do projeto. Considerando a alocação de pessoal, um diagrama de barras também dever ser produzido confrontando com o diagrama de tempo mostrando assim os períodos nos quais o pessoal está designado para o projeto. A equipe não precisa estar designada para um projeto o tempo todo. Em alguns intervalos de tempo as pessoas podem estar de férias, trabalhando em outros projetos, assistindo a cursos de treinamento ou realizando alguma outra atividade; e tudo isso deve ser levado em consideração na hora de elaborar o cronograma.

2.3.3 PERT e CPM

Como a determinação de um cronograma para um projeto de software não difere significativamente da fixação de prazos para qualquer esforço de desenvolvimento de multitarefas, ferramentas e técnicas para determinação de um cronograma de projeto podem ser aplicadas no software com poucas modificações.

Conforme citado por Pressman (2006), dois métodos de determinação de cronogramas que podem ser aplicados no desenvolvimento de software é o PERT – *Program Evaluation and Review Technique* (método de avaliação e revisão de programa) e o CPM – *Critical Path Method* (método do caminho crítico). As duas técnicas desenvolvem uma descrição da rede de tarefas de um projeto, que é definida ao se desenvolver uma lista de todas as tarefas associadas a um projeto específico e uma lista de disposições que indica em que ordem as tarefas devem ser executadas. Além disso ambas as técnicas proporcionam ferramentas quantitativas que permitem ao planejador de software: determinar o caminho critico, que seria a cadeia de tarefas que determina a duração do projeto; estabelecer as estimativas de tempo mais prováveis para tarefas individuais ao aplicar modelos estatísticos; e calcular limites de tempo que definam uma “janela” de tempo para uma tarefa em particular.

Para Pressman (2006) os cálculos de limite de tempo podem ser muito úteis na determinação de um cronograma para projetos de software. Por exemplo, o atraso no prazo de projeto de uma função pode retardar ainda mais o desenvolvimento de outras funções, fazendo com que isso se transforme em uma grande bola de neve quase que incontrolável, resultando diretamente em atraso na data de finalização de todo o projeto. Estes cálculos do tempo-limite levam à determinação do caminho crítico e proporcionam ao gerente um método quantitativo para avaliar o progresso à medida que as tarefas são executadas e concluídas.

2.3.4 Iniciativa para redução do tempo

De acordo com Ribeiro (2006), a primeira iniciativa para redução de tempo no desenvolvimento de software foi o RAD – *Rapid Application Development* (Desenvolvimento Rápido de Aplicação), desenvolvido por James Martin em 1991, tendo como principais objetivos a entrega mais rápida dos produtos e o aumento da qualidade enquanto diminui os gastos. Este modelo é baseado nas ferramentas *CASE*, e constituído de cinco pontos principais:

- Equipes reduzidas – Melhora a comunicação dentro do projeto;
- Prototipação evolutiva – Auxilia na identificação de requisitos na fase de entendimento do sistema;
- Uso de ferramentas *CASE* – O apoio de ferramentas *CASE* aumenta a produtividade através da ênfase na automação;
- JAD – *Joint Application Design* – Reuniões com todos os envolvidos no projeto com o objetivo de esclarecer e validar os requisitos.
- Rígidos limites de tempo no desenvolvimento (*timebox*) – Controle do escopo e limitação dos requisitos atendidos.

Conforme citado por Ribeiro (2006) os benefícios obtidos com a aplicação do RAD são reconhecidos pela maioria das organizações. Porém, ainda não existem comprovações sobre sua aplicação em projetos de maior porte, efetividade na manutenção da qualidade e o uso excessivo de ferramentas de automação para aumentar a produtividade.

Após a RAD em 1991, Arthur em 1992 desenvolveu o *Rapid Evolutionary Development* (Desenvolvimento Rápido Evolutivo), que é um modelo de ciclo de vida com o objetivo de reduzir o tempo de desenvolvimento de software baseado no desenvolvimento evolutivo tradicional aliado ao forte uso da prototipação, significando maior interação com o cliente e acomodação das alterações de requisitos durante as fases iniciais do projeto, para Ribeiro (2006).

Como citado por Ribeiro (2006) Arthur em 1992 enumerou três principais características do modelo por ele criado:

- Prototipação – Facilita a captura e o entendimento dos requisitos;

- Uso da Lei de Pareto – Chamada de regra 80/20. Sugere que 20% dos produtos possuem 80% do valor esperado. O foco desse modelo é desenvolver os requisitos que se enquadram nos 20% dessa regra;
- Interação com o cliente – Permite a validação constante dos requisitos e produtos durante o projeto.

A partir dessas características este modelo visa reduzir os riscos, o retrabalho e a complexidade dos produtos e dos processos durante o desenvolvimento de software.

Para Ribeiro (2006) entre 1996 e 1999 um estudo relacionado à redução de tempo realizado por Blackburn, Scudder e Wassenhove envolvendo 145 empresas de desenvolvimento de software nos Estados Unidos, Europa e Japão identificou onze fatores que auxiliam da redução de tempo no desenvolvimento de software. Esses fatores, apresentados na FIG. 4 receberam classificação de um a cinco quanto ao grau de importância de cada fator, onde zero é o menor valor e quatro o maior em uma escala de 0,5.

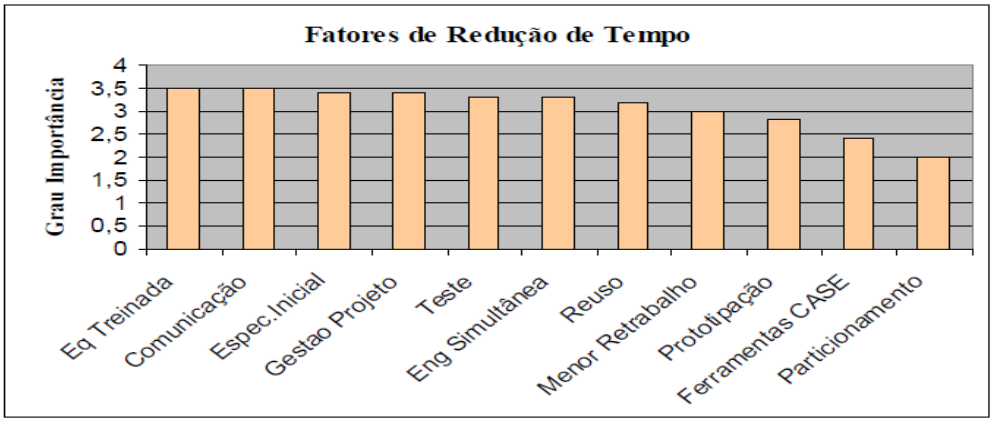

FIGURA 4 – Fatores de redução do tempo
Fonte: Ribeiro (2006) PAG. 59

Com esse estudo foi possível descrever algumas observações:

- A aplicação da engenharia simultânea no ciclo de desenvolvimento melhora a produtividade das esquipes de desenvolvimento e reduz substancialmente o tempo de desenvolvimento;
- Equipes reduzidas tendem a ser mais produtivas e reduzem os problemas de comunicação;
- A utilização de ferramentas *CASE* recebeu baixa importância em termos de redução de tempo de desenvolvimento;

- É possível reduzir o tempo de desenvolvimento sem aumentar a produtividade.

Foi possível também realizar outra avaliação tendo este estudo como base, que foi a verificação do relacionamento entre a velocidade de desenvolvimento e a alocação do tempo em cada fase do projeto. A FIG. 5 apresenta o resultado dessa avaliação, onde é possível observar que as empresas consideradas mais ágeis consumiram mais tempo nas fases iniciais que nas fases seguintes, exceto na fase de codificação.

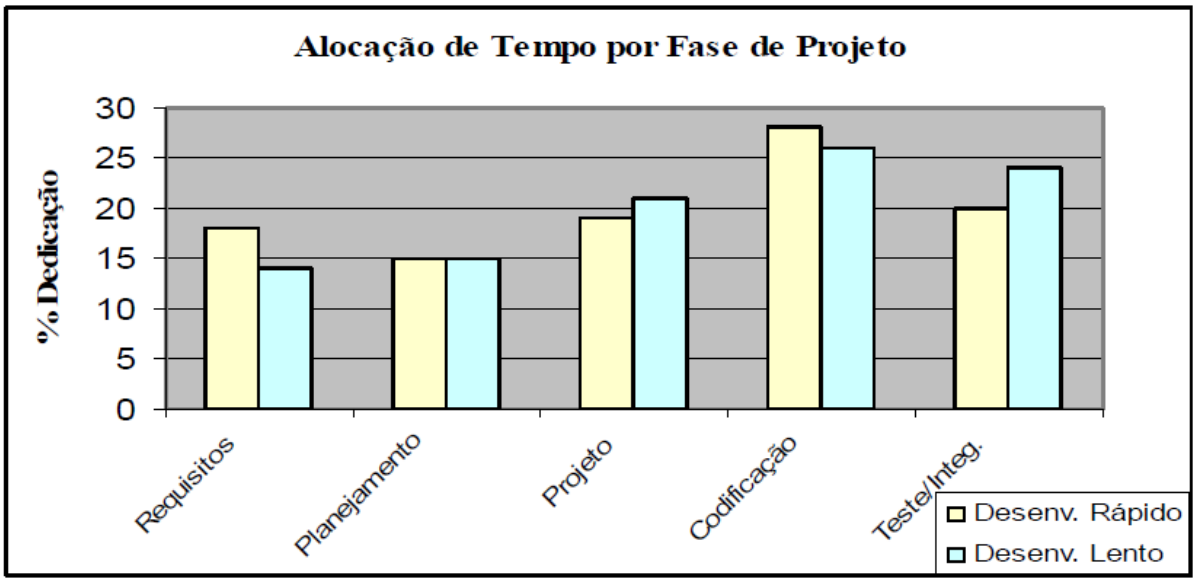

FIGURA 5 – Alocação de tempo por fase
Fonte: Ribeiro (2006) PAG. 60

Conforme citado por Ribeiro (2006), esta conclusão mostra que para reduzir o tempo no ciclo de desenvolvimento é necessário ser mais lento nas fases iniciais para se atingir o objetivo. Para aumentar a velocidade e a produtividade deve-se investir mais tempo em entender as necessidades, produzir especificações sem ambiguidades e elaborar planejamento adequado ao projeto.

2.4 MEDIDA DE SOFTWARE

Quando se trata de engenharia, a medição é um fator totalmente comum. Porem, infelizmente, a engenharia de software está longe de ter uma medição padrão amplamente aceita e com resultados sem nenhum fator subjetivo. Ainda

existem muitas dificuldades para a concordância sobre o que medir e como avaliar os resultados dessa medição.

De acordo com Vavassori (2001) um consenso que existe relacionado à medição de software é que as medições e métricas permitem um melhor entendimento do processo utilizado para desenvolver um produto, assim como uma melhor avaliação do próprio produto. Portando essas medições, no caso do processo de desenvolvimento, podem permitir melhorias no processo aumentando sua produtividade; e no caso do produto elas podem proporcionar informações a respeito de sua qualidade. Ambos os casos podem ser considerados metas para um gerente de projeto.

Para Pressman (2006) a medição do software é realizada por muitas razões, como pode-se relacionar: indicar a qualidade do produto; avaliar a produtividade das pessoas que produzem o produto; avaliar os benefícios (em termos de produtividade e qualidade) derivados de novos métodos e ferramentas de software; formar uma linha básica para estimativas; ajudar a justificar os pedidos de novas ferramentas ou treinamento adicional.

Conforme Borges (2003) no contexto específico do desenvolvimento de software a medição pode ser vista como o processo de definir, coletar, analisar e agir sobre medidas que possam potencialmente melhorar tanto a qualidade do software desenvolvido por uma organização quanto o próprio processo de desenvolvimento utilizado.

Tanto as medições do mundo físico quanto as métricas de software podem ser divididas em duas categorias, focando no projeto de software tem-se:

- Medidas diretas – pode-se considerar o esforço e o custo aplicados no desenvolvimento e manutenção do software, além da quantidade de linhas de código produzidas e o total de defeitos registrados durante um determinado período de tempo;
- Medidas indiretas – pode-se considerar qualidade, funcionalidade, complexidade, confiabilidade, eficiência do software ou a sua capacidade de manutenção, por serem mais difíceis de serem avaliadas.

O domínio das métricas de software ainda podem, minuciosamente, serem divididas em mais algumas categorias:

- Métricas da produtividade – se concentra, principalmente, na saída do processo de engenharia de software;

- Métricas da qualidade – indica quanto o software atende as requisitos implícitos e explícitos definidos pelo cliente;
- Métricas técnicas – se concentra na característica do software e não no processo por meio do qual o software foi desenvolvido.

Consequentemente uma segunda divisão em categorias também pode ser desenvolvida, segundo Pressman (2006).

- Métricas orientadas ao tamanho – usadas para compilar as medições diretas da saída e da qualidade da engenharia de software, ou seja, são medidas diretas do software e do processo por meio do qual ele é desenvolvido. A medida-chave para essa métrica é o uso da contagem de LOC (linhas de código), portanto KLOC seria mil linhas de código, e embora possa parecer simples existe muita discordância em torno dessa métrica, a ponto de não ser aceita universalmente como a melhor maneira de se medir o processo de desenvolvimento de software; e isso se dá pelo fato de que esses opositores afirmam que as medidas LOC são dependentes da linguagem de programação, que não podem acomodar facilmente linguagens não-procedimentais e que, além disso, para a maioria dos pesquisadores, essa medida não deveria contar linhas de comentários e linhas em branco, uma vez que estas linhas servem para o documentação interna do programa e não afeta a sua funcionalidade. Agravando a situação tem-se que o planejador deve estimar as LOC a serem produzidas muito antes que a análise e o projeto tenham sido concluídas.

Em Maldonado (2002) tem-se um modelo de como se pode desenvolver um conjunto de métricas de qualidade e produtividade usando esta técnica, mostrada logo a seguir.

TABELA 1 – Métricas orientadas ao tamanho

Produtividade	**Qualidade**	**Custo**	**Documentação**
KLOC/pessoa-mês	Defeitos/KLOC	$/LOC	Páginas/KLOC

A Tabela 1 traz o modelo onde na coluna produtividade coloca-se a estimativa de quantidade de mil linhas de código que o profissional consegue implementar; na coluna qualidade enumera-se a quantidade de defeitos ou erros existente por cada mil linhas de códigos; na coluna custo coloca-se a média do custo, na moeda desejada, de uma linha de código e na coluna documentação coloca-se a quantidade de páginas documentadas por cada mil linhas de código.

- Métricas orientadas à função – medidas indiretas do software e do processo por meio do qual ele é desenvolvido. A base desta medida se dá no fato de que em vez de contar as LOC (linhas de código) ela se concentra na “funcionalidade” ou “utilidade” do programa. De acordo com Pressman (2006), a pedido de um grupo de usuários, esta métrica foi proposta no início da década de 70 por pesquisadores da IBM, que tinham como objetivo a identificação das variáveis críticas que determinavam a produtividade da programação; descobriu-se que poderiam basear a avaliação de um software medindo o valor das funções executadas pelos programas, em vez de utilizar como base o volume ou a complexidade dos códigos. Dando continuidade a essas pesquisas, Allan Albrecht em 1979 sugeriu uma técnica de avaliação da produtividade conhecida, e até hoje chamada, de ponto-por-função.

Os pontos-por-função (FPs) está baseada na visão externa do usuário, permitindo calcular o esforço de programação e auxiliando o usuário final a melhorar a avaliação do projeto. Pressman (2006) fornece uma maneira de se computar os pontos por-função. Primeiramente completa-se a Tabela 2 apresentada:

TABELA 2 – Computando à métrica ponto-por-função

		Fator de ponderação			
Parâmetros de medida	**Contagem**		**Simples**	**Médio**	**Complexo**	
Número de entradas do usuário		X	3	4	6	=
Número de saídas do usuário		X	4	5	7	=
Número de consultas do usuário		X	3	4	6	=
Número de arquivos		X	7	10	15	=
Número de interfaces externas		X	5	7	10	=
Contagem - total						

A Tabela 2 reúne dados para serem computados. Estes dado são reunidos da seguinte forma: conta-se o número de entradas ao software pelo usuário, pondera-se essas entradas como sendo simples, média ou complexa e de acordo com a ponderação multiplica-se a quantidade de entradas pelo fator correspondente da ponderação obtendo-se o resultado da multiplicação; este mesmo processo é realizado com o número de saídas do software para o usuário, número de consultas

do usuário, número de arquivos do software e o número de interfaces externas do software; no final soma-se todos os resultados obtidos e tem-se a contagem total.

Uma vez reunidos esses dados, um valor de complexidade é associado a cada contagem. Demonstrada por Pressman (2006), a seguinte relação é usada para computar os pontos-por-função:

$FP = \text{contagem total} \times [0{,}65 + 0{,}01 \times SOMA(F_i)]$ (1)

onde contagem total é a soma de todas as entradas de FP obtidas a partir da Tabela 2; e F_i(i = 1 a 14) são "valores de ajuste da complexidade" baseados nas respostas de um questionário demonstrado logo abaixo.

TABELA 3 – Computando os pontos-por-função

Pontue cada fator numa escala de 0 a 5:					
0	**1**	**2**	**3**	**4**	**5**
Sem influência	Incidental	Moderado	Médio	Significativo	Essencial
F_i:					
1.	O sistema requer backup e recuperação confiáveis?				
2.	São exigidas comunicações de dados?				
3.	Há funções de processamento distribuídas?				
4.	O desempenho é crítico?				
5.	O sistema funcionará num ambiente operacional existente, intensivamente utilizado?				
6.	O sistema requer entrada de dados on-line?				
7.	A entrada de dados on-line exige que a transação de entrada seja elaborada em múltiplas telas ou operações?				
8.	Os arquivos-mestres são atualizados on-line?				
9.	A entrada, saída, arquivos ou consultas são complexos?				
10.	O processo interno é complexo?				
11.	O código foi projetado de forma a ser reusável?				
12.	A conversação e a instalação estão incluídas no projeto?				
13.	O sistema é projetado para múltiplas instalações em diferentes organizações?				
14.	A aplicação é projetada de forma a facilitar mudanças e o uso pelo usuário?				

A Tabela 3 é preenchida da seguinte forma: em uma escala de zero a cinco, onde zero é o menor valor significando que a questão não tem nenhuma influência e cinco é o maior valor significando que a questão é essencial; a tabela é composta de quatorze questões onde para cada questão deve ser atribuído apenas um valor dentro da escala mencionada.

Assim que calculados, os FPs serão usados como às LOC, servindo de parâmetro de medida de produtividade, qualidade e outros atributos do software. Esta métrica assim como a LOC é controversa por ser independente de linguagem de programação e se basear em dados empíricos e que são coletados logo no

começo da evolução do projeto, tornando-se mais atraente como abordagem de estimativa.

- Métricas orientadas às pessoas – de acordo com Pressman (2006), compilam informações sobre a maneira que as pessoas envolvidas no projeto desenvolvem software de computador, e também estuda as percepções humanas sobre a efetividade das ferramentas e métodos.

Fugindo um pouco do padrão dos tipos de medidas apresentadas, Borges (2003) apresenta alguns tipos de medidas de software. Medidas de software podem ser classificadas por diferentes aspectos: a natureza do atributo que está sendo medido, medidas de produto e de processo; o relacionamento entre a medida e o atributo medido, medidas básicas e derivadas; a objetividade, medidas objetivas e subjetivas; e o momento da mensuração, medidas preditivas e explanatórias. Entre as classificações Borges (2003) destaca:

- Medidas de produto e de processo – Medidas de produto são aquelas obtidas a partir de características de um produto ou artefato em qualquer estágio de desenvolvimento, como o código-fonte ou uma especificação de requisitos. Medidas de processo são obtidas a partir das atividades envolvidas no desenvolvimento dos produtos.

- Medidas básicas e derivadas – Medidas básicas são aquelas que podem ser mensuradas a partir de observação direta dos atributos envolvidos. Medidas derivadas são aquelas que não podem ser mensuradas diretamente a partir da observação de um atributo, mas são calculados a partir de combinações de outras medidas.

- Medidas objetivas e subjetivas – Uma medida objetiva consiste na contagem absoluta de atributos do produto ou processo, sendo idealmente independente do autor da mensuração, então a mesma medição, realizada por duas pessoas diferentes, gera resultados idênticos. Medidas subjetivas envolvem a classificação ou a qualificação de um aspecto do produto ou processo, baseada em julgamento humano; por essa característica é que observadores diferentes podem medir valores diferentes para um mesmo atributo, caso possuam opiniões divergentes.

- Medidas preditivas e explanatórias – Medidas preditivas consistem em estimativas geradas com o objetivo de prever certos aspectos do desenvolvimento

com antecedência. Medidas explanatórias são produzidas a partir das ocorrências dos eventos, com o intuito de caracterizá-lo objetivamente.

São várias, e bastante divergentes, as maneiras de “medir um software”, um estudo prévio de cada uma deve ser realizada pelo gerente do projeto para que ele possa entender como medir e como adaptar a melhor maneira de se fazer isso com seu projeto atual.

2.5 PRODUTIVIDADE NO DESENVOLVIMENTO DE SOFTWARE

A produtividade é considerada uma das mais importantes armas de competição das indústrias. É com o aumento da produtividade que as indústrias terão melhor competitividade, os produtos serão melhores e mais baratos, os serviços serão mais bem prestados, com isso pode ser que os salários melhorarão possibilitando uma melhoria de vida para os funcionários e um possível desenvolvimento econômico do país.

O que muito acontece, e acaba sendo um erro, é tratar a produção do software como se fosse semelhante a outras produções, onde o aumento da velocidade da linha de montagem ou o aumento de pessoal incrementam a produção. Acaba se pensando na produtividade como a produção de um conjunto de componentes obtidos num certo período de tempo, do qual espera-se maximizar o número de componentes produzidos para uma dada duração. Porém, a produtividade de um programador individual (medida em quantidade de linhas de código por pessoa/mês) acaba diminuindo quando adiciona-se mais pessoas ao projeto e para piorar o pessoal adicionado pode afetar diretamente na qualidade do software e na maioria das vezes negativamente.

Para Filho (2000) a produtividade dos desenvolvedores de software pode ser estimada de acordo com projetos anteriores da organização, porém é afetada por muitas variações, que dependem de pessoas, processos e tecnologias, além disso, riscos previstos e não previstos podem vir a se materializar. De acordo com este autor, um número assustador indica que, pelo menos, 50% dos projetos são executados com níveis de produtividade abaixo do normal.

2.5.1 Problemas da produtividade

Os problemas enfrentados pela produtividade no desenvolvimento de software, são basicamente, ou tem como base, os mesmos problemas já abordados neste trabalho relacionados ao termo “crise do software”.

Peters (2001) conceitua tais problemas de duas maneiras um pouco diferenciadas. De acordo com este autor foi identificado: o problema conceitual e o problema representativo.

O problema conceitual seria especificar, projetar e testar a construção conceitual implícita em um sistema de software. Este problema é considerado difícil devido ao fato de que a essência de uma entidade de software é uma construção de conceitos inter-relacionados, sendo que esses conceitos podem ser encontrados nos conjuntos de dados, nas relações entre os itens de dados, nos algoritmos e nas chamadas de funções dentro de um programa.

O problema representativo seria representar o software e testar a fidelidade de uma representação. Este problema, contrastando o problema conceitual, já é considerado mais fácil, pois está ligado a recursos acidentais do software.

De acordo com Peters (2001) um recurso é considerado acidental se a existência de algo não depender do recurso; exemplos de recursos acidentais de software são a linguagem no qual o software foi desenvolvido (alto nível ou nível de máquina), a representação gráfica deste software ou composição (modular ou não); se esses recursos podem ser modificados sem que a essência do software seja alterada.

Em contrapartida existem os recursos essenciais. “Um recurso é considerado essencial se algo não puder ser mantido sem ele” (Peters, 2001). Exemplos de tais dificuldades essenciais das entidades de software seria a complexidade e a abstração. A complexidade pode ser medida: em relação ao número de predicados condicionais de um programa; com base em tipos de instruções, contagem de operadores, níveis de aninhamento, fluxo de informações e contagem de instruções; pela determinação dos requisitos para os recursos, espaço e tempo. Considerada outra dificuldade essencial dos requisitos de software seria a abstração. Em particular, a lógica implícita em um modelo de processo de desenvolvimento de software é uma coisa abstrata.

Dicas, ou inovações na engenharia de software, apresentadas por Peters (2001) podem ajudar os engenheiros de software a minimizar tais problemas apresentados. Formas para combater as dificuldades essenciais: compre, não faça; refine os requisitos de forma iterativa e interativa com o cliente usando protótipos; desenvolva o software de forma incremental; contrate projetistas talentosos; utilize frameworks básicos; modele sistemas de software; análise sistemas de software. Para as dificuldades acidentais o autor recomenda: linguagens de programação de alto nível; programação orientada a objeto – OO; inteligência artificial - IA; sistemas especialistas; programação automática; programação visual; verificação de programas; aprimoramentos de hardware.

Sob uma abordagem bastante abrangente, Filho (2000) apresenta alguns problemas, ou poderiam ser chamados de erros, relacionados à produtividade ou ao desenvolvimento de softwares:

- Os erros clássicos: geralmente são cometidos por organizações imaturas, na maioria das vezes esses erros são sempre repetidos apesar de terem soluções conhecidas e publicadas há muito tempo;
- Os erros relativos ao produto: decorrem de defeitos da definição do próprio produto. Estes erros são decorrentes desde a origem da definição; assim como podem também vir à tona ao decorrer do projeto. Os mais comuns são: a introdução de características interessantes, mas dispensáveis; o inchaço causado por adições descontroladas de novos requisitos; o desenvolvimento orientado para a pesquisa, e não para a realização de um produto;
- Os erros relativos a processos: são comuns em organizações que utilizam processos informais, mas também podem ocorrer naquelas que adotam processos oficiais rígidos e burocráticos, que não são realmente seguidos pelos desenvolvedores. Os erros mais comuns são: tempo desperdiçado antes do inicio do projeto; pressões causadas por prazos excessivamente otimistas; planejamento insuficiente dos projetos; falta de controles gerenciais adequados para acompanhar os projetos, e fazer cumprir o planejado; abandono dos planos, sob pressão para o cumprimento de prazos impossíveis; codificação desenfreada, baseada em desenho insuficiente ou inexistente; falha de subcontratados, comum em casos de terceirização sem controle competente de compromissos; entrega prematura do produto.

Para Sommerville (2003) o processo é um conjunto de atividades e resultados associados que produzem um produto de software. As fases ou atividades que compõem este processo de software são: a especificação, o projeto, a implementação, a validação, a manutenção e evolução.

- Os erros relativos às pessoas: erros que poderiam ser evitados com técnicas consagradas da área de desenvolvimento humano, recursos humanos ou gestão de pessoas, mas o problema é que a maioria dos gerentes de projeto não tiveram nenhuma formação relacionada a esta questão e muito menos se interessam em adquirir tal formação. Os erros mais comuns são: falta de patrocínio eficaz para o projeto; falta de participação das partes interessadas em um produto, principalmente na engenharia de requisitos; atritos entre desenvolvedores e usuários ou clientes, causados por expectativas irreais destes e otimismos sem fundamentos daqueles; defeitos de formação de staff[3] do projeto; escritórios apinhados e barulhentos; falta de motivação dos desenvolvedores, que geralmente surge em conseqüência dos outros problemas;
- Os erros relativos à tecnologia: é de conhecimento que a engenharia de software trabalha com tecnologia de ponta, porém a confiança nas soluções tecnológicas também é causa de alguns erros: a crença nas “balas de prata” fecha os olhos de muitos desenvolvedores e gerentes de projeto às limitações da tecnologia; mudança de ferramentas no meio do projeto; em poucos casos, a falta de automação de algumas atividades.

Em meio a tantos problemas enfrentados pelos gerentes de software relacionados à produtividade de software existem algumas ferramentas que auxiliam neste processo de controle de tais problemas, conhecidas com ferramentas *CASE*, que será abordada na seção 2.6.

2.5.2 A produtividade e o tempo de desenvolvimento

Como cita Ribeiro (2006), o cumprimento de prazos no desenvolvimento de softwares é tão critico que, além do objetivo de execução do projeto dentro do prazo, o próprio controle de atrasos no ciclo de produção é um fator a ser considerado na análise de redução do tempo de desenvolvimento. A complexidade do ambiente de

[3] Quadro dos dirigentes de uma empresa, de uma organização.

software, a competitividade de mercado, as mudanças de requisitos constantes durante o projeto e o tempo disponível cada vez mais restrito aumentam a chance de insucesso quando analisado o indicador de tempo na produção de software. A agilidade e rapidez com que os ambientes de produção evoluem, atualmente tornam o fator de redução do tempo no ciclo de vida do desenvolvimento um diferencial para as empresas de desenvolvimento de sistemas.

Uma iniciativa existente, apresentada por Carriel (2007) é a programação sendo realizada em par, ou seja, conjuntamente por dois programadores. Esse tipo de iniciativa tem sido apontada como uma boa alternativa para a melhoria da produtividade em desenvolvimento de software. O objetivo do trabalho realizado por Carriel (2007) é confirmar ou não que a programação em par produz software de melhor qualidade, com menos defeitos e com a produtividade superior. Entende-se que programação em par seria aquela em que dois programadores trabalhem efetivamente no mesmo problema, na mesma máquina no mesmo momento e a programação tradicional seria aquela em que o programador trabalha só em seu computador. Como resultado, não muito satisfatório para esta iniciativa, Carriel obteve que os programadores trabalhando em par demoraram 37,5% a mais do tempo que os programadores que trabalharam em máquinas separadas. Além disso foi observado que na dupla que programava em máquinas separadas houve uma diferença na experiência dos programadores com relação as tecnologias utilizadas, sendo que um deles desenvolveu 2/3 da carga do trabalho enquanto que o outro desenvolveu o 1/3 restante.

Machado (2003) traz alguns dados levantados pela pesquisa realizada pelo *Standish Group* relacionado com projetos de software.

Percentual de sucesso:

- 9% em grandes empresas;
- 16% em médias empresas;
- 28% em pequenas empresas.

Dentre os fracassos:

- 31% foram cancelados;
- 53% custaram mais que 189% de sua estimativa.

Cenário nacional:

- 12,8% dos gerentes desconhecem o custo associado;
- 41,43% atrasaram, mas foram considerados sucesso;
- 50,68% utilizaram métricas para planejamento.

De acordo com este levantamento, pequenas empresas terminam seus projetos com sucesso com uma porcentagem bem superior às grandes empresas, mas, vale ressaltar que 28% de projetos finalizados com sucesso está muito abaixo do ideal para acabar com a crise do software. Um projeto que no final custa em média 189% a mais do que estimado tem que ser considerado um projeto fracassado e 53% dos projetos se enquadram nesta categoria. Métricas para planejamento é essencial para o sucesso do projeto, porém de acordo com esta pesquisa apenas 50,68% utilizaram alguma métrica com esta finalidade.

Por causa da existência de diferentes mudanças tecnológicas apresentadas no desenvolvimento de software, tem sido muito difícil para as indústrias de software, desenvolver uma técnica adequada para medir sua produtividade.

2.6 FERRAMENTAS CASE

Na engenharia de software, com sua a constante evolução, os engenheiros de software, nome que se dá ao profissional projetista do software, contam com uma grande ajuda que é a engenharia de software auxiliada por computador (*computer-aided software engineering – CASE*). As tecnologias *CASE* compreendem uma ampla variedade de tópicos que abrangem métodos de engenharia de software e procedimentos de gerenciamento de projetos.

As ferramentas *CASE* tinham como objetivo automatizar atividades manuais pré-codificação, como DER - diagramas de entidade-relacionamento e DFD – diagramas de fluxo de dados. Assim, com isso, os engenheiros de software, ou analistas de sistemas, utilizam softwares para elaborar tais diagramas, facilitando suas correções e a documentação do sistema.

De acordo com Pressman (2006), hoje as ferramentas *CASE* somam-se a caixa de ferramentas do engenheiro de software, pois, proporciona ao engenheiro a capacidade de automatizar atividades manuais e de melhorar a informação de

engenharia; porém, para se tornar “o mais importante avanço tecnológico”, o *CASE* precisa evoluir. Ele deve formar os blocos de construção de uma oficina de desenvolvimento de software. Oficina de engenharia de software é denominada como um ambiente integrado de suporte a projetos.

Frisando o surgimento, a evolução e o conceito de ferramentas *CASE*, segue uma parte de um texto apresentado na dissertação de Kido (2009).

> O aumento das complexidades dos sistemas criou a necessidade da existência de uma sistemática de desenvolvimento de software que garantisse qualidade e produtividade para o desenvolvimento de sistemas. Desde o final da década de 1960 a engenharia de software evolui em diversos sentidos, estruturando técnicas, teorias, métodos e ferramentas necessárias para garantir qualidade e produtividade no processo de desenvolvimento. Nesse cenário as ferramentas de apoio ao processo de desenvolvimento de software, ou ferramentas *CASE*, possuem um papel importante na evolução da engenharia de software, automatizando diversas tarefas e auxiliando em diversas partes do ciclo de desenvolvimento de software. Embora a existência de diversas ferramentas *CASE* possa significar um grande aumento no apoio à melhoria do processo é necessário ter bastante atenção ao adotar uma ferramenta.

São várias as maneiras de se classificar as ferramentas, e são muitos os modelos que fazem esse papel, por isso é complicado encontrar um padrão de classificação para as ferramentas *CASE* nas literaturas. Conforme citado por Kido (2009), as ferramentas *CASE* podem ser classificadas utilizando diversos fatores, como por exemplo: por tipo de interatividade; por tipo de atuação; por etapas no processo de desenvolvimento; por função. Uma outra forma, também citado por Kido (2009), seria a classificação das ferramentas em interativas por natureza, chamada de ferramentas *CASE*; e aquelas que não são interativas, chamadas de ferramentas de desenvolvimento.

Seguindo ainda as citações de Kido (2009), outra maneira de se classificar as ferramentas seria de acordo com a atuação dentro do projeto: vertical, atuação pontual e especifica; horizontal, mais genérica abrangendo diversas fases do desenvolvimento. Finalizando, outra maneira, seria a separação das ferramentas que atuam nas primeiras fases do ciclo de desenvolvimento de um projeto de software daquelas que são utilizadas nas últimas fases do ciclo; tem-se então a seguinte classificação: *front end ou upper CASE*, apoiam as etapas iniciais de criação dos sistemas, ou seja, as fases de planejamento, análise e projeto do programa; *back end ou lower CASE*, dão apoio a parte física, isto é, a codificação,

testes e manutenção da aplicação; *I-CASE* ou *integrated CASE*, classifica os produtos que cobrem todo o ciclo de vida do software, desde os requisitos do sistema até o controle final da qualidade.

Para Pressman (2006) categorizar as ferramentas *CASE* exige muito cuidado, pois, é inerente que existe uma série de riscos, por isso ele defende que é necessário criar uma taxonomia de ferramentas *CASE*, para que melhor se entenda a amplitude do *CASE* e melhor a apreciasse onde tais ferramentas podem ser aplicadas no processo de engenharia de software.

A taxonomia por Pressman (2006) apresentada utiliza a classificação por função como critério primordial e seria, basicamente, da seguinte forma: O engenheiro de software, ou analista de sistemas, teria um banco de dados *CASE* que conteria as seguintes ferramentas para serem utilizadas basicamente na ordem em que serão descritas:

- Ferramentas de planejamento de sistemas comerciais: constituem um “meta-modelo” a partir do qual sistemas de informação específicos são derivados; tem como principal objetivo ajudar a melhorar a compreensão de como a informação flui entre as várias unidades organizacionais.
- Ferramentas de gerenciamento de projetos: atualmente concentram-se num elemento especifico do gerenciamento de projetos, em vez de oferecer um suporte abrangente à atividade de gerenciamento. Um conjunto de ferramentas *CASE* que poderia ser dividida da seguinte maneira: ferramentas de planejamento de projetos; ferramentas de rastreamento de requisitos; ferramentas de métricas e gerenciamento.
- Ferramentas de apoio: reúne ferramentas de aplicação e de sistemas que complementam o processo de engenharia de software. Essas ferramentas abrangem as atividades de “guarda-chuva” que são aplicáveis em todo o processo de engenharia de software. Entre elas incluem: ferramentas de documentação; ferramentas de rede e software básico; ferramentas de garantia de qualidade; ferramentas de gerenciamento de bancos de dados e SCM – Software Configuration Management (gerenciamento de configuração de software).
- Ferramentas de análise e projeto: possibilitam ao engenheiro de software criar um modelo do sistema que será construído e auxiliam na avaliação da qualidade do modelo. Sua subdivisão seria: ferramentas SA – análise estruturada e SD – projeto estruturado; ferramentas PRO – prototipação e SIM – simulação;

ferramentas de projeto e desenvolvimento de interfaces; núcleos de análise e projeto.

- Ferramentas de programação: abrange compiladores, editores e depuradores que se encontram à disposição para apoiar a maioria das linguagens de programação convencionais. Sua subdivisão seria: ferramentas de codificação convencionais; ferramentas de codificação de quarta geração; ferramentas de programação orientadas a objeto.
- Ferramentas de integração e teste: as seguintes categorias de ferramentas de teste poderia ser definida: aquisição de dados; medição estática; medição dinâmica; gerenciamento de teste; ferramentas transfuncionais.
- Ferramentas de prototipação: como a prototipação é um paradigma da engenharia de software amplamente usado qualquer ferramenta que o suporte pode ser legitimamente chamada de ferramenta de prototipação, por isso muitas das ferramentas já apresentadas também podem ser incluídas nesta categoria.
- Ferramentas de manutenção: voltam-se a uma atividade que absorve aproximadamente 70% de todos os esforços relacionados a software, a manutenção. Sua subdivisão seria: ferramentas de engenharia reversa para especificação; ferramentas de análise e reestruturação de código; ferramentas de reengenharia de sistemas on-line.
- Ferramentas de estrutura: exibem componentes funcionais que suportam dados, interfaces e integração de ferramentas.

Sommerville (2003) apresenta um termo muito interessante a respeito das ferramentas *CASE*; o *workbenches CASE*. Seria um conjunto de ferramentas que são utilizadas em determinada fase do processo de software, como a fase de projeto, implementação ou teste. A vantagem é que as ferramentas podem trabalhar em conjunto para oferecer um apoio mais abrangente. Serviços comuns podem ser implementados e solicitados por todas as outras ferramentas. Elas podem ser interligadas por meio de arquivos compartilhados, de um repositório compartilhado ou de estruturas de dados compartilhadas.

Spinola (2004) apresenta que a tarefa de integração é por natureza complexa e árdua. Na área de desenvolvimento de software existem basicamente duas principais abordagens de integração: a baseada em modelo canônico e a fundamentada na integração de aplicações aos pares. Na primeira abordagem todas

as ferramentas devem estar projetadas para lidar com uma mesma tecnologia de armazenamento e ainda possuir interfaces para permitir a integração em níveis de apresentação, controle, processo, dado e conhecimento. Na segunda abordagem a integração é menos acoplada e, conceitualmente, foi projetada para facilitar o intercâmbio de informação entre aplicações independentes. Porém, corre-se o risco da manutenção do mapeamento entre as ferramentas exigir muito esforço, já que há a necessidade da criação de dois conversores para transformação dos artefatos para cada par de aplicações integradas.

Hoje no mercado existem inúmeras ferramentas *CASE* por isso não se tem a garantia de que foram enumeradas todas, será apresentado, suscintamente, duas iniciativas apresentadas no XV Simpósio Brasileiro de Engenharia de Software.

Prado (2001) apresenta a ferramenta MVCASE, que suporta o desenvolvimento de software orientados a objetos, baseados em componentes. Nesta ferramenta o engenheiro de software modela o sistema, seguindo o método especifico *Catalysis*, e gera seu código para uma plataforma distribuída, usando componentes distribuídos com a tecnologia EJB – *Enterprise Java Beans*. A ferramenta disponibiliza os componentes em um *browser*, que facilita seu reuso, através da herança ou instanciação de suas classes.

Vavassori (2001) apresenta a ferramenta Gemetrics, que tem como objetivo fornecer ao gerente de projeto úteis estimativas de esforço, custo e duração de um projeto de software, além de definir uma estrutura de divisão do trabalho, planejar uma programação viável de projeto e acompanhar projetos em base contínua. No mais o gerente pode usar a ferramenta para compilar métricas, que por fim oferecerão uma indicação da produtividade no desenvolvimento de software e da qualidade do produto. De acordo com Vavassori (2001) a ferramenta está sendo implementada em Delphi, sendo portanto, executada em plataforma Windows.

Os gerentes de software podem contar com várias iniciativas, uma delas, aqui citada, é um comparativo feito pela OAT SOLUTIONS (2005), uma empresa de São Paulo que atua na área de consultoria, treinamento e ferramentas de apoio.

PRODUTO >>>	System Architect	Rational Rose	Enterprise Architect	Erwin	Visible Suite	Power Designer
A. Características Gerais						
Características Operacionais						
Trabalham em plataformas Windows 9x/ NT/ ME / 2000 / XP	✓	✓	✓	✓	✓	✓
Possui dicionário (repositório) unificado de dados	✓	✓	✓	✓	✓	✓
Permite compartilhamento do repositório entre "n" analistas	✓	✓(através do Clear Case)	✓	✓	✓	✓
Customizável	✓		✓		✓	✓
Permite definir diferentes níveis de acesso a usuários ou grupos de usuários	✓	✓	✓	✓	✓	✓
Possui interface de usuário customizável	✓	✓	✓		✓	✓
É extensível, permitindo utilizar VBA, COM/DCOM e OLE	✓	✓	✓	✓	✓	✓
B. Suporte Metodológico						
Técnicas Estruturadas (Gane & Sarson, Ward & Mellor, entre outros)	✓				✓	✓ (alguns autores)
Análise Essencial	✓				✓	✓
Análise Orientada a Objetos	✓	✓	✓		✓	✓
Suporta técnicas para modelagem de processos de negócios	✓ (IDEF e UML)	✓ (UML)	✓ (UML)	✓ (via BPWin)	✓	

FIGURA 6 – Comparativo entre ferramentas CASE
Fonte: OAT SOLUTIONS (2005)

Agregando conhecimento intelectual a uma boa ferramenta *CASE*, a garantia de que um projeto de desenvolvimento de software se concretize, dentro das possíveis normalidades, será muito maior e melhor.

Uma vez que o referencial teórico já fora explícito, o próximo capítulo traz a metodologia utilizada na elaboração deste trabalho.

3 METODOLOGIA

O trabalho em questão consiste em um estudo a respeito da produtividade dos desenvolvedores de software com ênfase no tempo de desenvolvimento que já fora abordado como sendo um fator relevante e motivador da crise do software e na insatisfação do cliente. O estudo visou levantar informações a respeito de como os profissionais da área de desenvolvimento de software pensam a respeito desta questão, o que conhecem de ferramentas e como funcionam estes aspectos em organizações em que trabalham. Entender a realidade dos envolvidos na criação de sistemas pode colaborar com a engenharia de software uma vez que se conhece a realidade nacional.

3.1 CLASSIFICAÇÃO DA PESQUISA

Pelo motivo de existirem profissionais com perfis diferentes é que o meio adotado para coleta dos dados foi através da pesquisa bibliográfica e levantamento. A pesquisa bibliográfica contribuirá para: obter informações sobre a situação atual do tema ou problema pesquisado; conhecer publicações existentes sobre o tema e os aspectos que já foram abordados; verificar as opiniões similares e diferentes a respeito do tema ou ao problema de pesquisa.

O levantamento se faz necessário, pois como citado por Menezes (2001), levantamento é quando a pesquisa envolve a interrogação direta das pessoas cujo comportamento se deseja conhecer. O levantamento foi realizado pois deseja-se saber se as características relatadas em publicações se manifestam ainda no contexto da engenharia de software no dias de hoje.

3.2 POPULAÇÃO E AMOSTRA DA PESQUISA

O público alvo desta pesquisa abrange completamente todos os profissionais da área de TI, focando as pessoas diretamente ligadas ao desenvolvimento de sistemas. São eles os gerentes em geral, os analistas, os programadores e os projetistas. Geralmente os profissionais que estão na parte mais baixa desta hierarquia, que são os programadores, não se preocupam muito com esta questão teórica que é muito importante para um bom desempenho do desenvolvimento; esse tipo de profissional tem seu perfil mais voltado para a parte prática do desenvolvimento em questão que seria a implementação do código; mas é importante relevar que a participação do programador na fase do projeto e planejamento é essencial e imprescindível.

Gerentes e analistas já possuem em seu perfil um olhar um pouco mais voltado para esta questão, que é a fase inicial do projeto onde se montam os cronogramas, as redes de atividades etc., focando boa parte de sua preocupação no tempo de desenvolvimento do projeto.

O questionário foi respondido por 21 profissionais que se enquadram dentro dos cargos focados para a pesquisa, ou seja, gerentes, desenvolvedores, analistas e projetistas. A pesquisa, mais especificamente a coleta de dados, iniciou-se no dia 12/11/2010 e terminou no dia 22/11/2010.

3.3 INSTRUMENTO DE COLETA DE DADOS

Para se conseguir as informações relacionadas à opinião dos profissionais da área de desenvolvimento de softwares foi montado um questionário através dos recursos disponibilizados pelo sistema da *Google Docs*, que é um sistema de criação e manipulação de documentos de vários formatos disponibilizado pela empresa Google através da internet facilitando o acesso a esses arquivos de qualquer computador que tenha acesso a internet..

O questionário é um formulário com acesso on-line composto de 21 questões sendo 16 obrigatórias (em anexo). Vale ressaltar que o entrevistado não seria

identificado e tem a liberdade de não responder ao questionário sem ter que dar explicações a respeito de sua decisão.

O levantamento dos dados, ou seja, a divulgação do questionário, foi feito da seguinte forma. Foi criado uma rede de contatos com profissionais da área em questão, onde através de e-mail foi enviado o link para acesso ao questionário, uma breve explicação do foco da pesquisa, assim como uma breve explicação a respeito das regras do questionário e um pedido para a replicação de tais informações, ou seja um pedido para que este profissional que recebeu este e-mail passasse para seus colegas da mesma empresa ou de outra empresa na tentativa de se atingir o maior número de respondedores possível.

O questionário foi limitado quanto ao respondente, que obrigatoriamente deve estar trabalhando na área de TI com foco em desenvolvimento ou até mesmo que já tenha trabalhado, não se estende a alunos ou professores da área em questão mas que não atuam profissionalmente.

O questionário esteve aberto para ser respondido por aproximadamente 11 dias onde esteve na fase de coleta dos dados. Após este período esteve disponível apenas para o pesquisador que entraria na fase de análise dos dados, se por ventura acontecesse algum problema e o questionário pudesse ser respondido neste período após os 11 dias estas respostas seriam descartadas, não entrando assim na análise.

3.4 TRATAMENTO DOS DADOS

Os dados coletados foram reunidos e a partir deles foram elaborados primeiramente tabelas e depois gráficos para que se obtenha a relevância de incidência das respostas para cada pergunta de múltipla escolha. A partir dos gráficos tem como saber a porcentagem das respostas para cada pergunta, uma vez conhecida a porcentagem foram apontados os pontos relevantes a respeito do que os profissionais pensam refletindo a realidade dos profissionais desta área no geral.

Para as questões abertas foram analisadas as respostas de cada respondente e feita uma união destas respostas, desta união foi feito um resumo na tentativa de se obter respostas mais genéricas.

4 RESULTADOS

Os resultados são apresentados nas seções seguintes após a análise dos dados colhidos com o questionário. O universo da pesquisa abrange vinte e um respondentes, sendo todos profissionais da área de TI, mais especificamente aqueles que trabalham diretamente com desenvolvimento de softwares. Primeiramente foi feita uma análise de cada pergunta e depois seguirá uma análise das perguntas por cargo; ou seja, uma análise focando o que todos os profissionais de todos os cargos responderam a uma determinada questão e uma outra focando o que diferentes profissionais do mesmo cargo responderam as questões.

Segue as questões com suas devidas análises em relação as respostas obtidas. Objetivando ilustrar e ao mesmo tempo confrontar as respostas foram gerados gráficos com as respectivas porcentagens e estatísticas das questões.

4.1 ANÁLISE DE CADA PERGUNTA DO QUESTIONÁRIO

QUESTÃO 1 - Cargo ocupado:

TABELA 4 – Respostas da questão 1

Cargo	Quantidade	Porcentagem
Analista	7	33%
Desenvolvedor	10	48%
Gerente	2	10%
Projetista	1	5%
Outro	1	5%

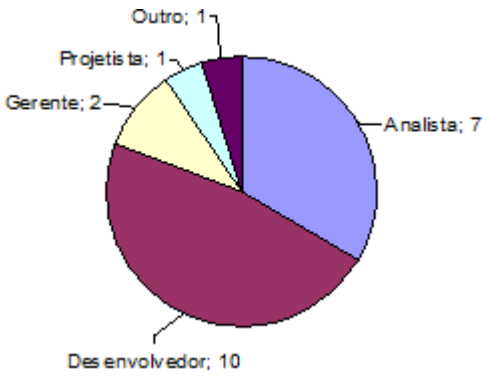

FIGURA 7 – Gráfico da questão número 1

Nesta primeira pergunta que tem como objetivo saber o cargo do respectivo respondente obtivemos que 33% são analistas, 48% são desenvolvedores, 10% são gerentes, 5% são projetistas e 5% ocupa outro cargo que foi descrito como sendo professor mas que tem desenvolvido projetos dentro da instituição. Portanto a maior parte dos pesquisados são desenvolvedores.

QUESTÃO 2 - Tempo que ocupa este cargo atual:

TABELA 5 – Respostas da questão 2

Tempo	**Quantidade**	**Porcentagem**
0 a 1 ano	3	14%
1 a 2 anos	5	24%
2 a 3 anos	5	24%
3 a 5 anos	7	33%
5 a 10 anos	1	5%
Mais de 10 anos	0	0%

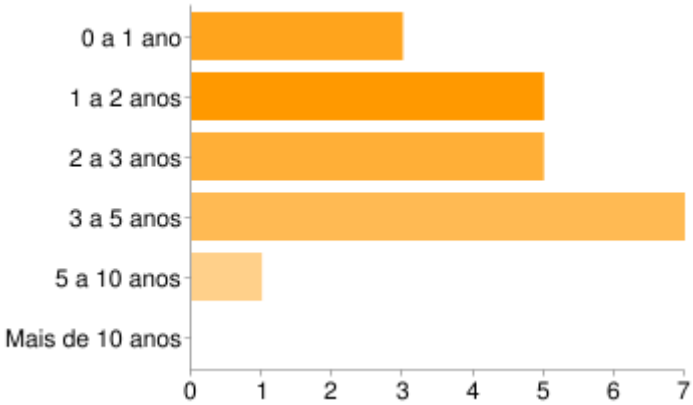

FIGURA 8 – Gráfico da questão número 2

Na FIG.8 tem-se o gráfico que representa a Tabela 5 mostrando que 14% dos profissionais pesquisados ocupam o cargo atual por menos de um ano, 24% ocupam entre um e dois anos, 24% de dois a três anos, 33% entre três e cinco anos e apenas 5% de cinco a dez anos. A maioria dos profissionais pesquisados ocupam o cargo atual de três a cinco anos. Isso implica que são experientes e suas respostas são ainda mais válidas para o questionário.

QUESTÃO 3 - A quanto tempo você trabalha com sistemas?

TABELA 6 – Respostas da questão 3

Tempo	Quantidade	Porcentagem
0 a 1 ano	0	0%
1 a 2 anos	1	5%
2 a 3 anos	7	33%
3 a 5 anos	6	29%
5 a 10 anos	7	33%
Mais de 10 anos	0	0%

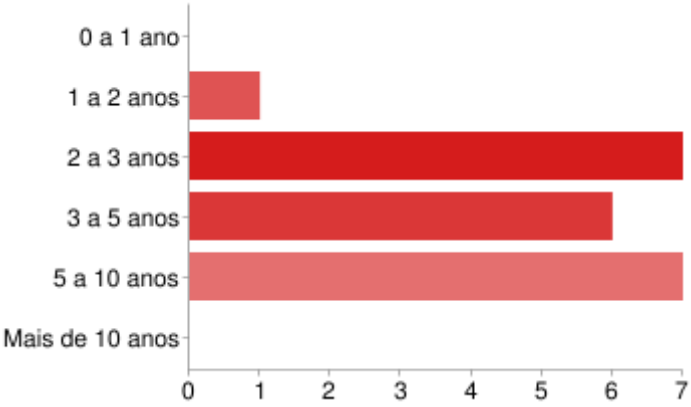

FIGURA 9 – Gráfico da questão número 3

Na FIG. 9 tem-se o gráfico que representa a Tabela 6 apresentando que 5% dos profissionais pesquisados trabalham com sistemas entre um e dois anos, 33% trabalham por volta de dois a três anos, 29% de três a cinco anos e 33% de cinco a dez anos. Conclui-se que a maioria do profissionais já trabalham com sistemas entre dois e três anos e de cinco a dez anos; ou seja um período muito bom para ser avaliado.

QUESTÃO 4 - Você tem conhecimento a respeito do termo “crise do software”?

TABELA 7 – Respostas da questão 4

Resposta	Quantidade	Porcentagem
Sim	11	52%
Não	10	48%

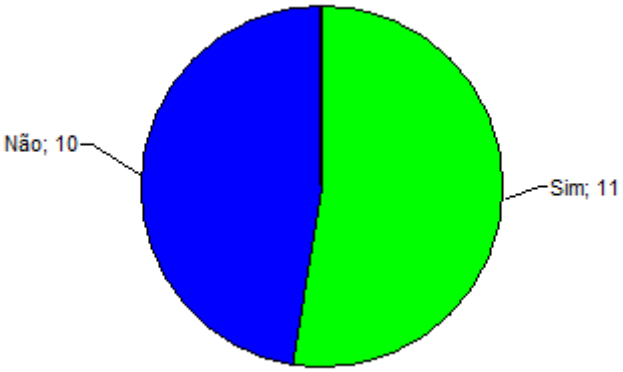

FIGURA 10 – Gráfico da questão número 4

Este gráfico que representa a Tabela 7 mostrando que 52% dos profissionais pesquisados tem conhecimento do termo “crise do software” e que os outros 48% não tem conhecimento deste termo. Nota-se que completamente a metade do universo pesquisado não tem conhecimento do termo “crise do software”. A porcentagem de profissionais que tem conhecimento deste termo, que é de 52%, deveria aumentar, refletindo que esses profissionais passaram a se preocupar um pouco mais com a teoria, logicamente que, ao mesmo tempo, evoluindo a prática.

QUESTÃO 5 - Os projetos de software na qual você participa são sempre entregues no tempo pré-determinado?

TABELA 8 – Respostas da questão 5

Resposta	**Quantidade**	**Porcentagem**
Sim	5	24%
Não	16	76%
Não sei responder	0	0%

FIGURA 11 – Gráfico da questão número 5

A FIG. 11 traz o gráfico que representa a Tabela 8 mostrando que, 24% dos profissionais responderam que os projetos na qual ele participa são entregues no tempo pré-estabelecido, os outros 76% disseram que o tempo de entrega do software ficou fora do escopo planejado. Isto reflete bastante a realidade em que a maioria dos projetos sofrem com esta questão de não cumprimento do prazo de entrega do projeto. Essa realidade condiz com um dos fracassos de projetos de softwares e reforça a teoria de que ainda estamos vivendo a crise do software.

QUESTÃO 5.1 - Se a resposta da pergunta número 5 for Sim, diga quais os métodos são utilizados pela empresa para que se consiga tal resultado.

Esta questão foi analisada da seguinte forma, obtive-se 24% das repostas para a pergunta número 5 como sendo Sim, portanto o universo de análise para a questão 5.1 será 24% do total dos profissionais pesquisados. De acordo com as respostas dos profissionais é necessário muita dedicação, organização e planejamento tanto das empresas quanto dos profissionais; uma alternativa para auxiliar neste planejamento é que seja utilizado o *SCRUM* (uma metodologia ágil para gestão e planejamento de projetos de software).

QUESTÃO 5.2 - Se a resposta da pergunta número 5 for Não, diga qual a sua idéia para minimizar este problema.

Como 76% dos profissionais pesquisados responderam Não para a pergunta número 5, após a análise a resposta para a questão 5.2 ficou da seguinte forma: As estimativas estão sendo feitas de forma errônea, melhora-se esta estimativa através de métricas relevantes, deve-se usar metodologias padronizadas como o *SCRUM* porém com a maturidade necessária para que a empresa possa adaptar os processos de acordo com as suas necessidades e não ao contrário; além do que se deve ter organização e qualificação do pessoal.

QUESTÃO 6 - A empresa que você atualmente trabalha planeja bem seus produtos e prestação de serviços?

TABELA 9 – Respostas da questão 6

Resposta	Quantidade	Porcentagem
Sim	5	24%
Não	9	43%
Não sei responder	7	33%

FIGURA 12 – Gráfico da questão número 6

A FIG. 12 traz o gráfico que representa a Tabela 9 mostrando que 24% dos profissionais pesquisados responderam que a empresa na qual ele trabalha planeja bem seus produtos e prestação de serviços, 43% responderam que a empresa não planeja bem e 33% não sabem responder. Mais uma vez a maioria refletindo a realidade que os profissionais da área enfrentam com o tema em estudo. Se a empresa não planeja bem seus produtos e serviços o resultado será um produto que não atenderá o cliente da maneira como precisa e um serviço mal prestado, contribuindo cada vez mais para a não solução dos problemas relacionados com a crise do software.

QUESTÃO 6.1 - Se a resposta da pergunta número 6 for Sim, diga como é feito este planejamento.

O universo de análise para esta questão será de 24% do total dos profissionais pesquisados, após esta análise foi obtido a seguinte resposta:

- É utilizado uma equipe de *quality assurance* (garantir a qualidade);
- o planejamento é realizado bem no início do projeto;
- utiliza-se a ferramenta Clockingit para auxiliar neste processo de planejamento;

- é utilizado um processo de desenvolvimento certificado nível G MPS-Br - Melhoria do Processo do Software Brasileiro.

QUESTÃO 7 - Quais destas metodologias relacionadas com métricas de tempo de desenvolvimento de Software você já ouviu falar?

TABELA 10 – Respostas da questão 7

Metodologia	**Quantidade**	**Porcentagem**
COCOMO	6	29%
Diagrama de Gantt	16	76%
Orientadas ao tamanho	3	14%
PERT/CPM	3	14%
Pontos-por-função	13	62%
Nenhum	3	14%
Não sei do que se trata essas metodologias	0	0%
Outro	2	10%

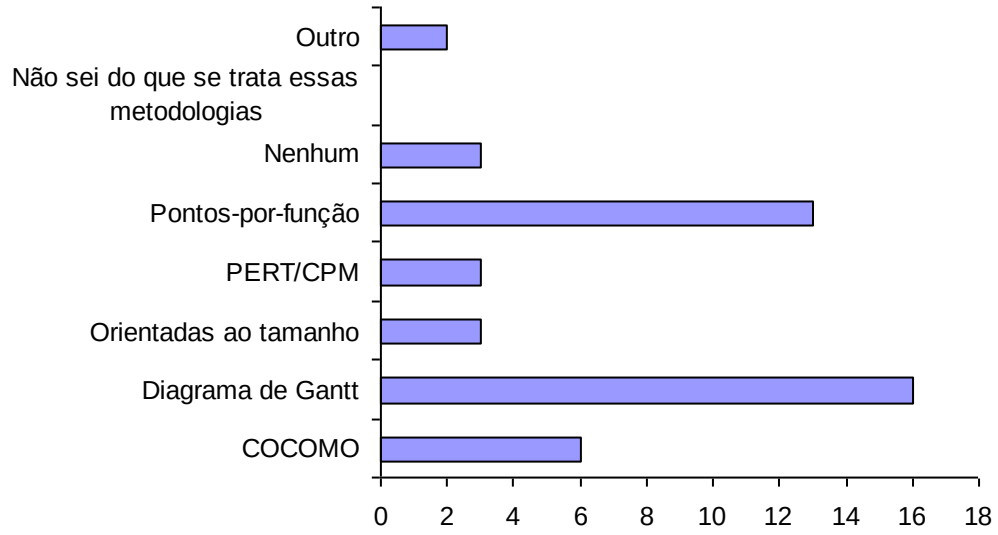

FIGURA 13 – Gráfico da questão número 7

As pessoas tinham a opção de assinalar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%. A FIG. 13 que representa a Tabela 10 mostrando que a metodologia COCOMO é conhecida por 29% dos profissionais pesquisados, diagrama de Gantt por 76%, orientadas ao tamanho por 14%, PERT/CPM por 14%, pontos-por-função por 62%, 14% alegaram não conhecer nenhuma das metodologias citadas e 10% conhecem outra metodologia sendo elas: pontos por caso de uso e todas dentro do Clockingit. Conclui-se que a metodologia mais conhecida dentre os entrevistados é o diagrama de Gantt, porém nesta

pergunta não tem como afirmar que é a mais utilizada. Percebe-se que muitos conhecem mais de uma metodologia.

QUESTÃO 8 - Você tem noção de quantas KLOC (mil linhas de código) você implementa por mês?

TABELA 11 – Respostas da questão 8

KLOC	Quantidade	Porcentagem
De 1 a 10 KLOC	4	19%
De 10 a 15 KLOC	2	10%
De 15 a 20 KLOC	0	0%
Mais de 20 KLOC	0	0%
Não sei responder	15	71%

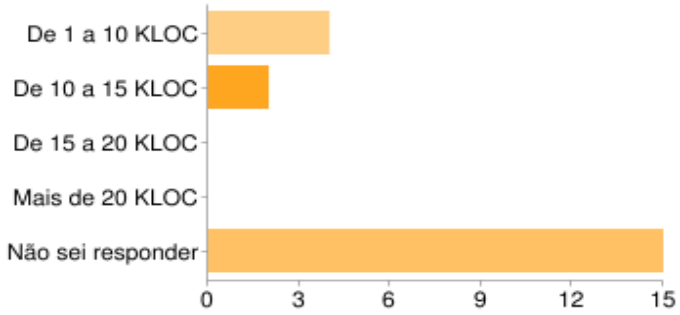

FIGURA 14 – Gráfico da questão número 8

Neste gráfico que representa a Tabela 11 mostrando que 19% dos profissionais pesquisados estimam que implementam de um a dez KLOC/mês, 10% de dez a quinze e 71% não sabem responder a esta questão. Pode-se concluir que a maioria dos entrevistados não fizeram nenhuma estimativa para descobrir quantas KLOC implementam por. Vale a pena promover maneiras de se mostrar aos envolvidos, independentemente do cargo que ocupa, a sua produtividade.

QUESTÃO 9 - Você conhece alguma ferramenta *CASE* para a gestão de tempo?

TABELA 12 – Respostas da questão 9

Resposta	Quantidade	Porcentagem
Sim	14	67%
Não	5	24%
Não sei responder	2	10%

FIGURA 15 – Gráfico da questão número 9

Na FIG. 15 tem-se o gráfico que representa a Tabela 12 mostrando que 67% dos profissionais pesquisados conhecem alguma ferramenta *CASE* para gestão de tempo, 24% não conhecem e 10% são sabem responder. Isto representa um dado relativamente bom pois a maioria tem conhecimento de alguma ferramenta o que poderá ser muito útil para toda a equipe.

QUESTÃO 10 - Destas ferramentas, quais você já ouviu falar?

TABELA 13 – Respostas da questão 10

Ferramenta	**Quantidade**	**Porcentagem**
Clockingit	9	43%
MS Project Standard	10	48%
OpenProj	7	33%
Redmine	5	24%
Workbench	5	24%
Nenhuma	3	14%
Outro	3	14%

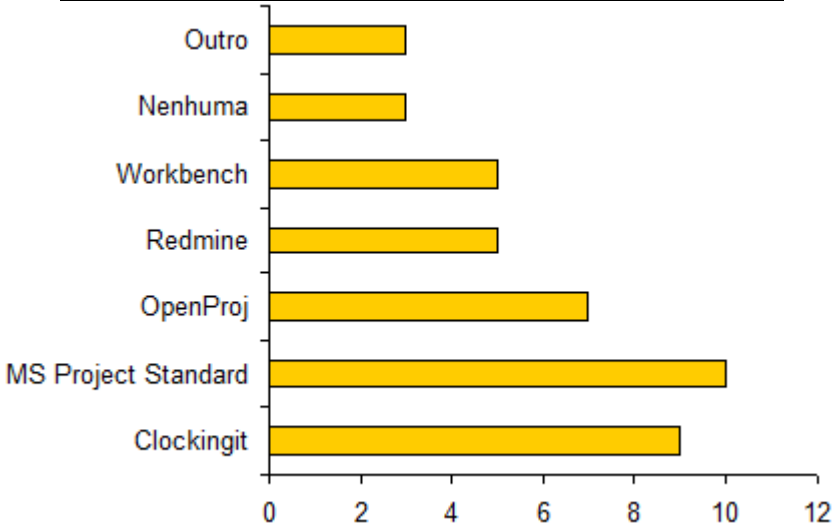

FIGURA 16 – Gráfico da questão número 10

As pessoas tinham a opção de assinalar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%. A FIG.16 traz o gráfico que representa a Tabela 13 mostrando que a ferramenta Clockingit é conhecida por 43% dos profissionais pesquisados, MS Project Standard por 48%, OpenProj por 33%, Redmine por 24%, Workbench por 24%, 14% alegaram não conhecer nenhuma das ferramentas citadas e 14% conhecem outra ferramentas sendo elas: Dot project, PSP dashboard e Agilo. Conclui-se que a ferramenta mais conhecida dentre os entrevistados é o MS Project Standard, porém nesta pergunta não tem como afirmar que é a mais utilizada. A minoria dos pesquisados não conhecia as ferramentas.

QUESTÃO 11 - A empresa na qual você trabalha utiliza alguma ferramenta de gestão de projetos?

TABELA 14 – Respostas da questão 11

Resposta	**Quantidade**	**Porcentagem**
Sim	12	57%
Não	6	29%
Não sei responder	3	14%

FIGURA 17 – Gráfico da questão número 11

Neste gráfico que representa a Tabela 14 mostrando que 57% dos profissionais pesquisados afirmaram que a empresa na qual ele trabalha utiliza alguma ferramenta de gestão de projetos, 29% responderam que a empresa não utiliza e 14% não sabem responder. Ter uma porcentagem de 57% dos entrevistados afirmando que a empresa utiliza alguma ferramenta de gestão de projetos é um dado bastante razoável diante da realidade que os profissionais enfrentam. Porém ter 29% dos pesquisados afirmando que a empresa não utiliza nenhuma ferramenta de gestão de projetos é alarmante, pois, sem uma ferramenta

para auxiliar, certamente o projeto sairá do controle dos profissionais nele envolvido podendo ocasionar até a não conclusão do projeto.

QUESTÃO 11.1 - Se a resposta da pergunta número 11 for Sim, diga qual.

Sabendo que o universo de respondentes para esta questão é de 57% do total dos profissionais pesquisados, após a análise foi obtido a seguinte resposta: As ferramentas para gestão de projetos utilizadas pela empresas dos profissionais pesquisados que fazem parte do universo de análise desta questão são: Centric, Workbench, Clockingit, Dot Project, Enterprise Project, MS Project, OpenProj, Trac com Agilo *plugin*.

QUESTÃO 12 - A empresa na qual você trabalha utiliza alguma ferramenta de gestão de tempo?

TABELA 15 – Respostas da questão 12

Resposta	**Quantidade**	**Porcentagem**
Sim	9	43%
Não	7	33%
Não sei responder	5	24%

FIGURA 18 – Gráfico da questão número 12

Neste gráfico que representa a Tabela 15 mostrando que 43% dos profissionais pesquisados afirmaram que a empresa na qual ele trabalha utiliza alguma ferramenta de gestão de tempo, 33% responderam que a empresa não utiliza e 24% não sabem responder. Diferentemente da ferramenta de gestão de projetos, a ferramenta específica para a gestão do tempo é um pouco menos utilizada, enquanto que a ferramenta de gestão de projetos obteve como resposta positiva de utilização por parte das empresas dos pesquisados uma porcentagem de 57% a ferramenta específica para gestão do tempo obteve apenas 43%.

QUESTÃO 12.1 - Se a resposta da pergunta número 12 for Sim, diga qual.

Sabendo que o universo de respondentes para esta questão é de 43% do total dos profissionais pesquisados, após a análise foi obtido a seguinte resposta: As ferramentas para gestão do tempo citadas pelos profissionais pesquisados que fazem parte do universo de análise desta questão são: Clockingit, Redmine, Dot Project, MS Project, Cronus e uma proprietária. Confrontando esta questão com a 11.1 observa-se que algumas ferramentas são utilizadas com as duas finalidades, tanto para a gestão do projeto quanto para, especificamente, a gestão do tempo de desenvolvimento do projeto.

QUESTÃO 13 - Assinale o grau de importância que você atribui ao gerenciamento de produtividade para desenvolvimento de softwares.

TABELA 16 – Respostas da questão 13

Escala	Quantidade	Porcentagem
0	0	0%
1	0	0%
2	0	0%
3	0	0%
4	0	0%
5	0	0%
6	0	0%
7	2	10%
8	7	33%
9	6	29%
10	6	29%

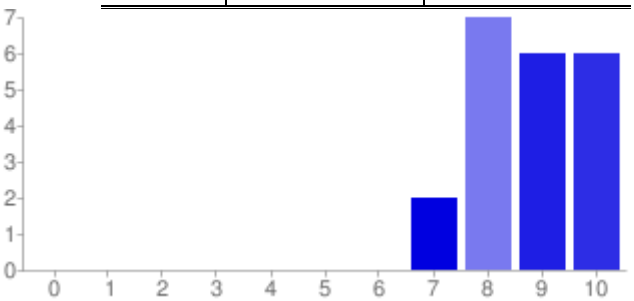

FIGURA 19 – Gráfico da questão número 13

A FIG. 19 traz o gráfico que representa a Tabela 16 sendo uma questão muito interessante, em uma escala de zero a dez onde zero representa menor importância e dez representa maior importância, foi obtido como resultado que 10% dos profissionais atribuíram nota sete, 33% atribuíram nota oito, 29% atribuíram nota

nove e 29% atribuíram nota dez. Observa-se que nenhum dos pesquisados atribuíram nota menor que sete para a importância do gerenciamento de produtividade mostrando que na teoria os profissionais são conscientes quanto a questão da produtividade.

QUESTÃO 14 - Em sua empresa existe política para incentivo a produtividade no desenvolvimento de softwares?

TABELA 17 – Respostas da questão 14

Resposta	**Quantidade**	**Porcentagem**
Sim	3	14%
Não	16	76%
Não sei responder	2	10%

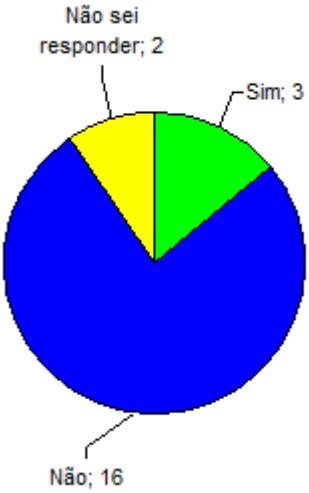

FIGURA 20 – Gráfico da questão número 14

Não menos interessante a FIG. 20 traz um gráfico que representa a Tabela 17 mostrando que apenas 14% dos pesquisados responderam existir políticas de incentivo quanto à produtividade na empresa em que trabalha, 76% responderam não existir esse incentivo e 10% não souberam responder a esta questão. A questão 15 irá complementar esta pergunta que obteve uma porcentagem relativamente alta quanto ao não incentivo à produtividade.

QUESTÃO 15 - Se existissem políticas para incentivo mais claras você acha que os desenvolvedores iriam ter maior produtividade no trabalho?

TABELA 18 – Respostas da questão 15

Resposta	Quantidade	Porcentagem
Sim	20	95%
Não	0	0%
Não sei responder	1	5%

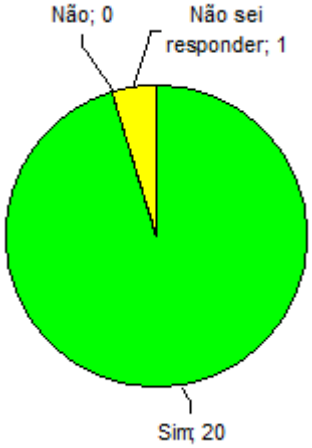

FIGURA 21 – Gráfico da questão número 15

Complementando a questão 14 a FIG. 21 traz o gráfico que representa a Tabela 18 mostrando que foi quase unânime a resposta positiva quanto ao questionamento a respeito do incentivo à produtividade, 95% dos profissionais pesquisados acham que se existissem políticas mais claras os desenvolvedores iram ter maior produtividade e 5% não souberam responder, nenhum dos entrevistados afirmaram que este incentivo não ajudariam os desenvolvedores a terem maior produtividade.

Isso aponta que faltam incentivos financeiros, educacionais, pessoais ou mesmo oportunidades profissionais dentro da organização, tanto que na questão anterior 76% dos pesquisados afirmaram não existir essa política de incentivo na empresa que trabalha. Então, de acordo com os profissionais pesquisados, com incentivos os profissionais iriam ter maior produtividade.

QUESTÃO 16 - Quais os erros, problemas, mais comuns que ocorrem na empresa na qual você trabalha?

TABELA 19 – Respostas da questão 16

Erros	**Quantidade**	**Porcentagem**
Erros relativos ao produto	7	33%
Erros relativos à processos	13	62%
Erros relativos à pessoas	10	48%
Erros relativos à tecnologia	4	19%
Nenhum	2	10%
Não tenho idéia do que se trata esses erros	0	0%
Outro	0	0%

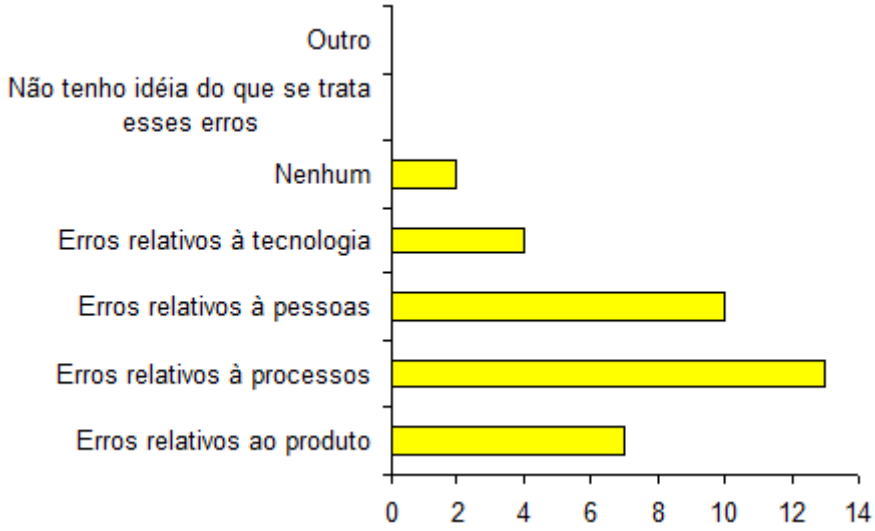

FIGURA 22 – Gráfico da questão número 16

As pessoas tinham a opção de assinalar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%. A FIG.22 traz o gráfico que representa a Tabela 19 apresentando que os erros relativos ao produto que acontece na empresa foi assinalado por 33% dos profissionais pesquisados; erros relativos à processos por 62%; erros relativos à pessoas por 48% dos pesquisados e erros relativos à tecnologia por 19%; 10% dos pesquisados responderam que a empresa na qual trabalham não sofrem com nenhum dos erros apontados na questão. Conclui-se que o erro mais comum das empresas dentre os pesquisados são os erros relativos à processos que pode ser considerado o epicentro dos problemas. Se uma organização erra muito nos processos que envolvem o desenvolvimento do projeto certamente sua produtividade será baixa e consequentemente o tempo deste projeto não será cumprido. Deve-se concentrar muitos esforços, por parte do profissionais, na tentativa de se controlar esses erros

relacionados ao processo, que como exemplo pode-se citar: estimativa de tempo excessivamente otimista, planejamento insuficiente, desenho inadequado e abandono do planejamento.

4.2 ANÁLISE POR CARGO

Para a análise de algumas perguntas, tendo como parâmetro principal o cargo que o profissional pesquisado ocupa atualmente, será citado apenas os três cargos que obtiveram maior porcentagem, ou seja, os três cargos que com certeza são ocupados por pelo menos mais que uma pessoa, são eles o cargo de analista que representa 33% dos profissionais, desenvolvedor que representa 48% e gerente que representa 10%. Assim esta análise possibilitará perceber diferentes visões de uma mesma questão, visões que são criadas de acordo com o cargo que o profissional ocupa. Vale ressaltar que nesta análise não constará todas as questões, mas apenas aquelas se enquadram melhor no enriquecimento dos resultados obtidos com a pesquisa.

QUESTÃO 3 - A quanto tempo você trabalha com sistemas?

TABELA 20 – Análise por cargo da questão 3

Cargo	**Representam no universo pesquisado**	**Tempo que trabalham com sistemas**			
		1 a 2 anos	2 a 3 anos	3 a 5 anos	5 a 10 anos
Analista	33%	0%	43%	14%	43%
Desenvolvedor	48%	10%	40%	40%	10%
Gerente	10%	0%	0%	50%	50%

Observa-se que apenas 10% dos desenvolvedores tem menos experiência trabalhando com sistemas, no geral os profissionais são muito experientes o que dá maior credibilidade a pesquisa, nota-se que 43% dos analistas já trabalham com sistemas entre cinco e 10 anos, 40% dos desenvolvedores entre três e cinco anos e 100% dos gerentes já trabalham com sistemas por mais de três anos.

QUESTÃO 4 - Você tem conhecimento a respeito do termo “crise do software”?

TABELA 21 – Análise por cargo da questão 4

Cargo	**Representam no universo pesquisado**	**Conhece o termo crise do software**	
		Sim	Não
Analista	33%	57%	43%
Desenvolvedor	48%	40%	60%
Gerente	10%	50%	50%

Se os profissionais conhecessem com maior profundidade a “crise do software” este termo já não estaria tão evidente nos dias atuais. Dados alarmantes desta pesquisa mostra que 43% dos analistas, 60% dos desenvolvedores e 50% dos gerentes não tem conhecimento deste termo.

QUESTÃO 5 - Os projetos de software na qual você participa são sempre entregues no tempo pré-determinado?

TABELA 22 – Análise por cargo da questão 5

Cargo	**Representam no universo pesquisado**	**Os projetos de software são entregues no tempo pré-determinado**	
		Sim	Não
Analista	33%	43%	57%
Desenvolvedor	48%	20%	80%
Gerente	10%	0%	100%

Ter consciência da deficiência e do problema já é um grande passo para a resolução. Tem-se 57% dos analistas e 80% dos desenvolvedores afirmando que os projetos de software na qual participa não são entregues no tempo pré-estabelecido e com participação direta no planejamento do projeto 100% dos gerentes responderam que esses projetos não são entregues no tempo correto.

QUESTÃO 6 - A empresa que você atualmente trabalha planeja bem seus produtos e prestação de serviços?

TABELA 23 – Análise por cargo da questão 6

Cargo	**Representam no universo pesquisado**	**A empresa planeja bem os produtos e serviços**		
		Sim	Não	Não sei responder
Analista	33%	14%	57%	29%
Desenvolvedor	48%	30%	30%	40%
Gerente	10%	0%	100%	0%

Uma informação que reforça a tese da crise do software é o fato de as empresas não planejarem bem seus produtos e prestações de serviços. Observa-se que 100% dos gerentes, profissionais que estão à frente dos projetos de software, responderam que as empresas não estão fazendo um bom planejamento.

QUESTÃO 7 - Quais destas metodologias relacionadas com métricas de tempo de desenvolvimento de Software você já ouviu falar?

As pessoas tinham a opção de assinalar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%.

TABELA 24 – Análise por cargo da questão 7

Metodologias que já ouviu falar	**Cargo**		
	Analista	Desenvolvedor	Gerente
COCOMO	14%	20%	50%
Diagrama de Gantt	57%	80%	100%
Orientadas ao tamanho	0%	20%	0%
PERT/CPM	0%	10%	0%
Pontos-por-função	57%	60%	50%
Nenhum	0%	10%	0%
Outro	0%	10%	0%

A correta utilização do diagrama de Gantt é uma boa alternativa no auxílio a gerência dos projetos já que uma boa porcentagem de profissionais de variados cargos tem conhecimento desta metodologia, sendo 57% dos analistas, 80% dos desenvolvedores e 100% dos gerentes.

QUESTÃO 9 - Você conhece alguma ferramenta *CASE* para a gestão de tempo?

TABELA 25 – Análise por cargo da questão 9

Cargo	**Representam no universo pesquisado**	**Conhece alguma ferramenta CASE para gestão de tempo**		
		Sim	Não	Não sei responder
Analista	33%	57%	29%	14%
Desenvolvedor	48%	70%	20%	10%
Gerente	10%	50%	50%	0%

Lamentavelmente, 50% dos gerentes responderam não ter conhecimento de alguma ferramenta *CASE* para gestão de tempo o que é muito preocupante, pois, sem a utilização de uma ferramenta *CASE* a gerência do tempo de desenvolvimento poderá ser muito difícil e provavelmente será falha. Têm-se dados um pouco melhores com relação a analistas e desenvolvedores uma vez que 57% dos

analistas e 70% dos desenvolvedores responderam ter conhecimento de alguma ferramenta.

QUESTÃO 11 - A empresa na qual você trabalha utiliza alguma ferramenta de gestão de projetos?

TABELA 26 – Análise por cargo da questão 11

Cargo	**Representam no universo pesquisado**	**A empresa utiliza alguma ferramenta de gestão de projetos**		
		Sim	Não	Não sei responder
Analista	33%	29%	57%	14%
Desenvolvedor	48%	80%	0%	20%
Gerente	10%	50%	50%	0%

Mais uma vez sendo dados preocupantes, 50% dos gerentes afirmaram que a empresa não utiliza uma ferramenta para gestão de projetos e reforçando esta preocupação 57% dos analistas também fizeram a mesma afirmação. Primeiramente deve-se investir em qualificação profissional desses profissionais para que tenham conhecimento suficiente para a consequente adoção de uma ferramenta por parte da empresa.

QUESTÃO 12 - A empresa na qual você trabalha utiliza alguma ferramenta de gestão de tempo?

TABELA 27 – Análise por cargo da questão 12

Cargo	**Representam no universo pesquisado**	**A empresa utiliza alguma ferramenta de gestão de tempo**		
		Sim	Não	Não sei responder
Analista	33%	43%	0%	57%
Desenvolvedor	48%	40%	10%	50%
Gerente	10%	50%	50%	0%

A questão de utilização de uma ferramenta para gestão de tempo reforça as mesmas preocupações mencionadas quanto a utilização de uma ferramenta de gestão de projetos, porém nesta análise os desenvolvedores têm uma maior contribuição para este problema, pois 50% desses respondentes não souberam responder a este questionamento.

QUESTÃO 13 - Assinale o grau de importância que você atribui ao gerenciamento de produtividade para desenvolvimento de softwares.

Em uma escala de zero a dez onde zero representa menor importância e dez representa maior importância.

TABELA 28 – Análise por cargo da questão 13

Cargo	Representam no universo pesquisado	Escala representando o grau de importância do gerenciamento da produtividade				
		0 ... 6	7	8	9	10
Analista	33%	0%	14%	29%	29%	29%
Desenvolvedor	48%	0%	0%	30%	30%	40%
Gerente	10%	0%	50%	0%	50%	0%

Não que os resultados obtidos com esta questão foi ruim, mas o ideal para o grau de importância atribuída ao gerenciamento da produtividade deve ser a nota dez. Ruim é perceber que nenhum gerente atribui esta nota ideal a esta questão.

QUESTÃO 14 - Em sua empresa existe política para incentivo a produtividade no desenvolvimento de softwares?

TABELA 29 – Análise por cargo da questão 14

Cargo	Representam no universo pesquisado	Na empresa existe política para incentivo a produtividade		
		Sim	Não	Não sei responder
Analista	33%	14%	71%	14%
Desenvolvedor	48%	20%	70%	10%
Gerente	10%	0%	100%	0%

Esta questão levanta um dado muito importante para as empresas começarem a batalhar com mais rigor contra a crise do software, pois, 71% dos analistas, 70% dos desenvolvedores e 100% dos gerentes responderam que em sua empresa não existe nenhuma política de incentivo a produtividade.

QUESTÃO 15 - Se existissem políticas para incentivo mais claras você acha que os desenvolvedores iriam ter maior produtividade no trabalho?

TABELA 30 – Análise por cargo da questão 15

Cargo	Representam no universo pesquisado	Com políticas de incentivos mais claras os desenvolvedores iriam ter maior produtividade		
		Sim	Não	Não sei responder
Analista	33%	100%	0%	0%
Desenvolvedor	48%	90%	0%	10%
Gerente	10%	100%	0%	0%

Para que as pessoas que estão a frente dessas empresas comecem realmente a pensar a respeito deste incentivo a produtividade, esta questão traz que 100% dos analistas e gerentes e 90% dos desenvolvedores acham que se existissem políticas mais claras quanto a este incentivo, os desenvolvedores iram aumentar sua produtividade.

QUESTÃO 16 - Quais os erros, problemas, mais comuns que ocorrem na empresa na qual você trabalha?

As pessoas tinham a opção de assinalar mais de uma caixa de seleção, então a soma das percentagens pode ultrapassar 100%.

TABELA 31 – Análise por cargo da questão 16

Erros mais comuns que ocorrem na empresa	**Cargo**		
	Analista	Desenvolvedor	Gerente
Erros relativos ao produto	29%	50%	0%
Erros relativos à processos	71%	60%	50%
Erros relativos à pessoas	43%	60%	50%
Erros relativos à tecnologia	14%	10%	50%
Nenhum	14%	10%	0%

Um ponto crucial onde as empresas devem ficar atentas, para que se possa resolver boa parte dos problemas relacionados à produtividade e ao tempo de desenvolvimento, é a minimização dos erros relacionados ao processo, pois 71% dos analistas, 60% dos desenvolvedores e 50% dos gerentes afirmaram que na prática estes são os erros mais comuns que acontecem dentro da organização.

4.3 DISCUSSÃO DOS RESULTADOS

Foi obtida uma quantidade satisfatória de respondentes para as análises dos resultados, sendo um total de vinte e um; a maioria dos profissionais já trabalham com sistemas de dois e três anos e de três a cinco anos; ou seja um tempo também satisfatório para dar credibilidade aos profissionais e à pesquisa.

Percebe-se, considerando o universo pesquisado, que o nível de conhecimento teórico dos profissionais não está se diferenciando muito de acordo com o cargo.

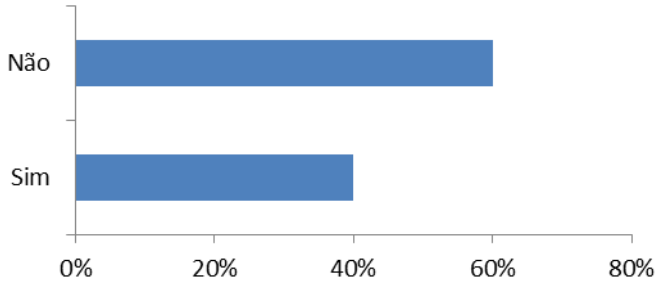

FIGURA 23 – Conhecem o termo “crise do software” (desenvolvedores)

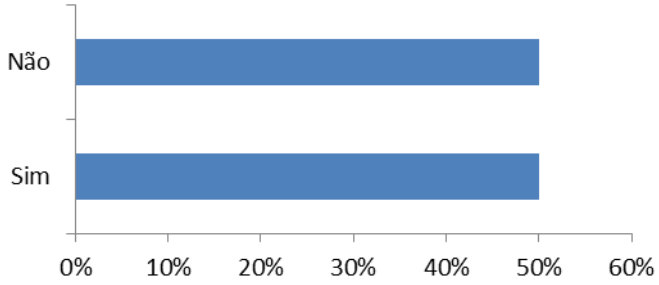

FIGURA 24 – Conhecem o termo “crise do software” (gerentes)

Na questão onde os pesquisados são questionados se conhecem o termo “crise do software”, como pode ser observado na FIG. 23 e FIG. 24, 60% dos desenvolvedores responderam que não e 50% dos gerentes também tiveram a mesma resposta, percebemos então que as porcentagens completamente se igualam.

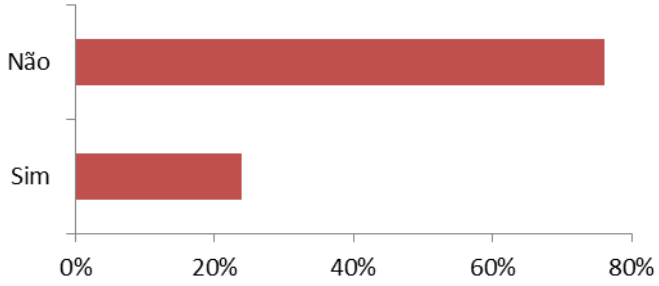

FIGURA 25 – Projetos entregues no tempo correto (universo total)

Quando questionados se os projetos de software na qual eles participam são entregues no tempo pré-determinado na FIG. 25 tem-se que 76% de todo o universo pesquisado responderam que não são entregues, ou seja, a maioria tem conhecimento deste fato e assumem existir uma falha.

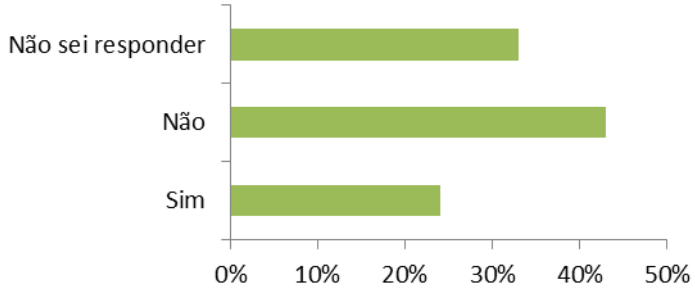

FIGURA 26 – Empresa planeja bem (universo total)

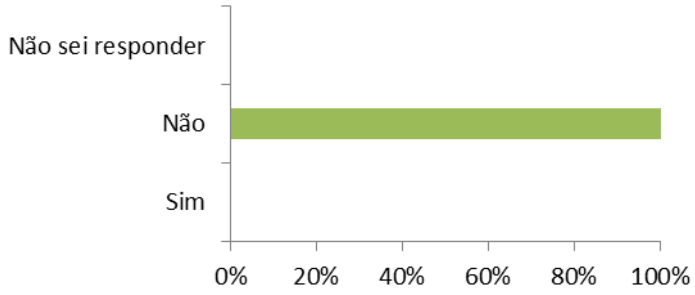

FIGURA 27 – Empresa planeja bem (gerentes)

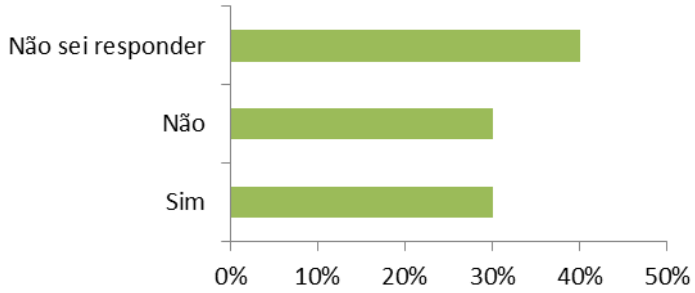

FIGURA 28 – Empresa planeja bem (desenvolvedores)

Complementando a questão na FIG. 26 observa-se que 43% responderam que a empresa não planeja bem seus produtos e prestação de serviços porém 33% não souberam responder, por se tratar na prática de uma questão que envolve mais os profissionais ligados a gerência dos projetos, tanto que a FIG. 27 mostra que 100% dos gerentes responderam que a empresa não planeja e a FIG. 28 mostra 40% dos desenvolvedores não souberam responder.

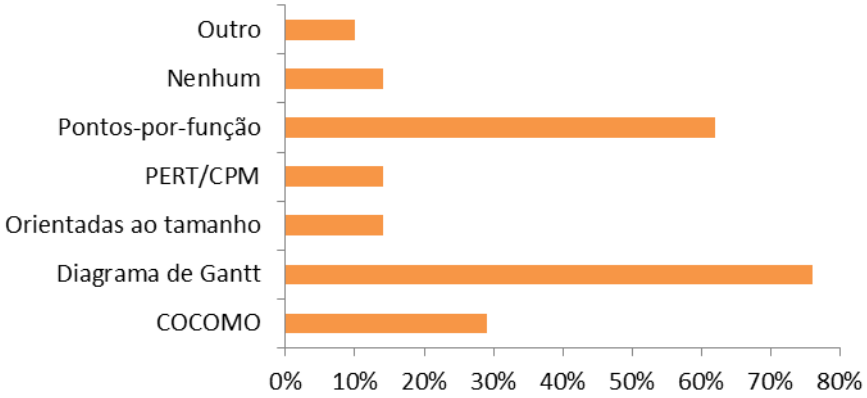

FIGURA 29 – Métricas conhecidas (universo total)

Com relação as métricas de softwares citadas neste trabalho obteve-se um bom resultado, boa parte das métricas são conhecidas por todos os profissionais destacando-se o Diagrama de Gantt que é conhecida por 76% dos pesquisados, como pode ser observado na FIG. 29.

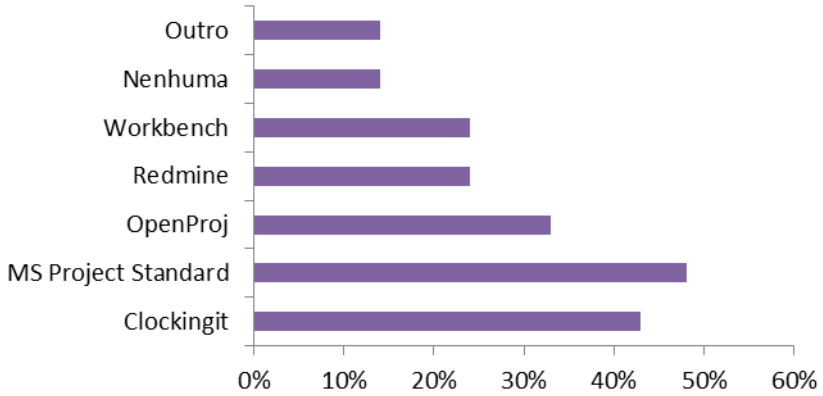

FIGURA 30 – Ferramentas conhecidas (universo total)

Ter conhecimento de alguma ferramenta *CASE* para gestão de tempo é também muito importante, na FIG. 30 tem-se que 67% dos pesquisados responderam ter conhecimento, das ferramentas citadas as mais conhecidas pelos pesquisados são a MS Project Standard que obteve 48% e a Clockingit que obteve 43%.

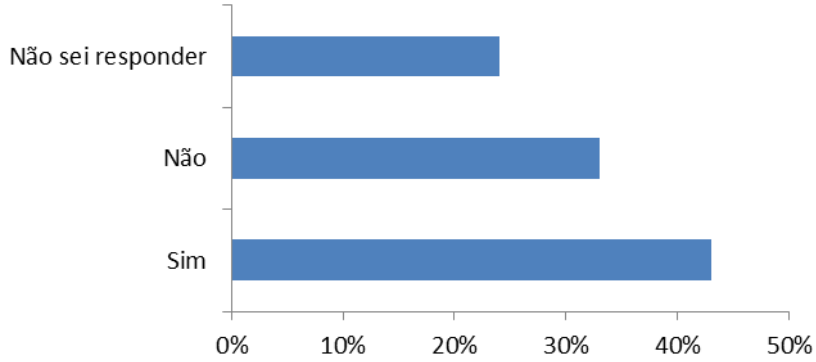

FIGURA 31 – Utiliza ferramenta CASE para gestão de tempo (universo total)

Quando questionados se a empresa na qual trabalham utilizam alguma ferramenta *CASE* para gestão de tempo 43% responderam que Sim como pode ser observado na FIG. 31, ou seja, a utilização teve uma porcentagem menor do que o conhecimento.

A maioria dos profissionais consideram o gerenciamento de produtividade um fator muito importante, em uma escala de zero a dez onde zero é o menor valor e dez o maior a maioria dos pesquisados assinalaram uma nota igual ou maior que oito, ou seja, sendo de muita importância este gerenciamento.

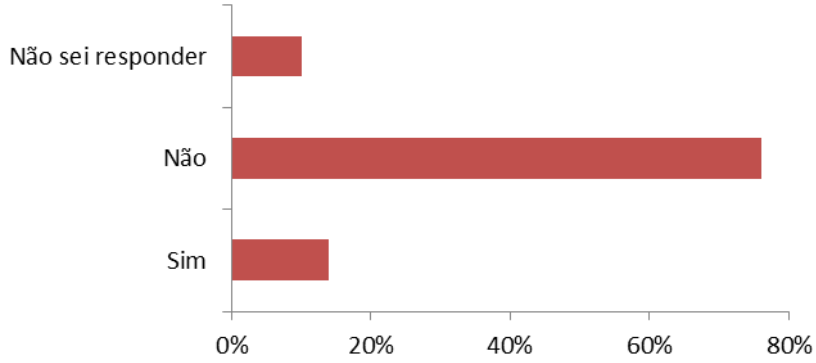

FIGURA 32 – Existência de política de incentivo na empresa (universo total)

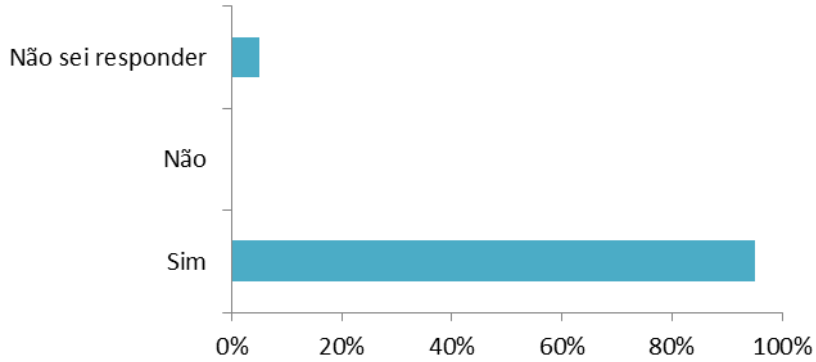

FIGURA 33 – Política de incentivo aumenta a produtividade (universo total)

Com relação ao incentivo quanto à produtividade 95% dos pesquisados acham que se existissem políticas mais claras quanto a isso a produtividade dos desenvolvedores iriam aumentar como pode ser observado na FIG. 32, reforçando esta questão tem-se a FIG. 33 mostrando que 76% dos pesquisados afirmaram que em sua empresa não existe este incentivo.

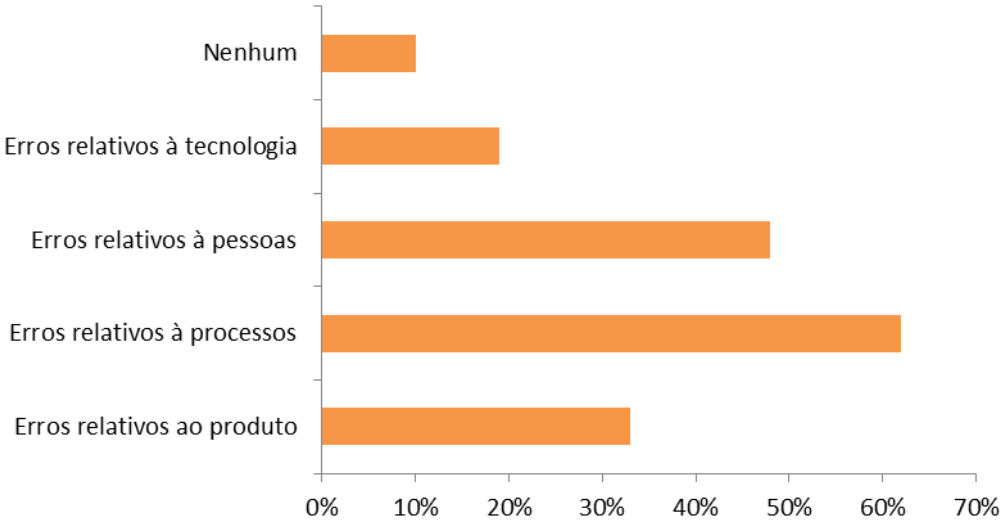

FIGURA 34 – Erros mais comuns das empresas (universo total)

Tem-se na FIG. 34 que os erros mais comuns das empresas na qual trabalham, de acordo com os profissionais pesquisados, são os erros relativos à processos, que são aqueles erros que mais ocorrem em organizações que utilizam processos informais mas que podem ocorrer também naquelas que utilizam processos burocráticos, pode-se citar como exemplo o desperdício de tempo antes do início do projeto, planejamento insuficientes dos projetos, pressões, abandono dos planos, etc.

Uma vez analisadas e planejadas todas as fases do processo, as chances do projeto não sofrer com os problemas de produtividade e tempo será muito maior.

5 CONCLUSÃO

Chegou-se a conclusão de que na teoria os níveis de conhecimento dos profissionais completamente se igualam, ou seja, o nível de conhecimento teórico a respeito do tema abordado nas questões de um desenvolvedor não está muito abaixo do nível de conhecimento de um gerente ou de um analista. Foi obtida uma quantidade satisfatória de respondentes com uma faixa de tempo trabalhando com sistemas suficiente para dar credibilidade a pesquisa.

De acordo com as respostas obtidas com a pesquisa conclui-se que devem existir mais planejamento e organização das empresas desenvolvedoras. Além do que elas devem adotar uma boa ferramenta *CASE* para auxiliar na gestão tanto do projeto quanto do tempo de desenvolvimento, combinada com a utilização de metodologias, como *SCRUM*, disponibilizadas pela engenharia de software; aplicar um esforço um pouco maior para inserção de todos os profissionais desde as fases iniciais dos projetos juntamente com a utilização de políticas mais claras de incentivos à produtividade.

Sendo tudo isto bem aplicado e utilizado, as empresas iriam ter um ganho significativo e um diferencial quanto às outras empresas nos quesitos produtividade e qualidade além de estarem minimizando o maior dos problemas enfrentados por estas empresas que é o tempo de desenvolvimento que na maioria dos projetos, é muito falho trazendo com si vários outros problemas que comprometem e muito a credibilidade do projeto em andamento reforçando a continuidade da crise do software.

6 TRABALHOS FUTUROS

Sugestão de trabalhos futuros poderia ser a aplicação das técnicas aqui citadas em uma empresa que não as utiliza, na tentativa de comprovar que elas realmente ajudam a reduzir o tempo de desenvolvimento minimizando o problema citado. Uma outra sugestão poderia ser a aplicação deste estudo tendo como universo de pesquisa profissionais atuantes fora do território nacional, na tentativa de fazer um parâmetro entre o que acontece dentro e fora de nosso país no que se diz respeito ao fator tempo de desenvolvimento de software e produtividade.

Interessante, também, seria a aplicação desta pesquisa em uma empresa específica de grande porte com o objetivo de se descobrir como é o comportamento dos profissionais desta empresa. Como sugestão, também, o estudo e aplicação de métodos únicos no processo de desenvolvimento; assim como métodos de desenvolvimento ágeis. A mudança no tipo de entrevista, como por exemplo, a entrevista grupal, serviria para fazer um parâmetro com esta pesquisa. Finalizando, desenvolver um questionário com foco na escolaridade do respondente seria útil para perceber se este fator, escolaridade, influencia na produtividade do profissional.

REFERÊNCIAS

BARBOSA, A., L. ***Análise comparativa de metodologias para o gerenciamento de projetos de desenvolvimento de software***. Dissertação de mestrado. (Engenharia elétrica), Faculdade de Engenharia Elétrica e de Computação (FEEC / UNICAMP), Campinas – SP, 2006.

BORGES, E., P. ***Um modelo de medição para processos de desenvolvimento de software***. Dissertação de mestrado. (Ciência da Computação), Universidade Federal de Minas Gerais (UFMG), Belo Horizonte – MG, 2003.

CARRIEL, G., N. ***Produtividade de desenvolvedores de software programando em par.*** Quinto congresso de Pós-graduação – UNIMEP, 2007.

FILHO, W., P., P. ***Engenharia de software: fundamentos, métodos e padrões.*** Editora LTC, 2000.

HAUFE, M., I. ***Estimativa da produtividade no desenvolvimento de software***. Dissertação de mestrado. (Ciência da Computação), Universidade Federal do Rio Grande do Sul, Porto Alegre – RS, 2001.

KIDO, E., Y. ***Modelo de identificação e análise de impactos da implantação de ferramentas CASE***. Dissertação de mestrado. (Departamento de Engenharia de Computação e Sistemas Digitais), Escola Politécnica da Universidade de São Paulo, São Paulo - SP, 2009.

KOSCIANSKI, André; SOARES, Michel dos S. ***Qualidade de software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software***. Segunda edição. Novatec, 2007.

MACHADO, P., D., L. ***Gerência de projetos: Gerência de tempo e gerência de riscos***. Slides em pdf da professora da DSC/UFCG, Patrícia D. L. Machado; acessado em 03/12/2010 às 19:00 horas; disponível em: http://www.dsc.ufcg.edu.br/~patricia/esii2003.1/aulas/gerenciadeprojetos.pdf. Ano: 2003.

MALDONADO, J., C,; NAKAGAWA E., Y. ***Gerência e planejamento de projeto***. Slides em pdf da professora, do Departamento de informática da UFPR, Silvia Regina Vergílio; acessado em 28/10/2010 às 15:00 horas; disponível em: http://www.inf.ufpr.br/silvia/ES/planejamento/planeja2.pdf. Ano: 2002.

MENEZES, Estera M; SILVA, Edna L. da. ***Metodologia da pesquisa e Elaboração de dissertação***. 3ª Edição. UFSC / PPGEP / LED, Florianópolis – SC, 2001.

OAT SOLUTIONS. ***Comparativo entre ferramenta CASE v3***. Encontrado em http://www.oatsolutions.com.br/artigos/Comparativo_CASES_v3.pdf, acessado em 28/09/2010 as 23:30 hrs. Ano: 2005.

PETERS, J., F.; PEDRYCZ, W. ***Engenharia de software – Teoria e prática***. Tradução de *An Engineering aproach*. Rio de Janeiro. Editora: Campus, 2001.

PRADO, A., F.; LUCRÉDIO, D. ***Ferramenta MVCASE – Estágio atual: especificação, projeto e construção de componentes***. Artigo apresentado no XV Simpósio Brasileiro de Engenharia de Software; 2001.

PRESSMAN, Roger S. ***Engenharia de Software***. 6ª ed. Editora: McGraw-Hill, 2006.

RIBEIRO, A., L., D. ***Um roteiro para a redução do tempo no desenvolvimento de projetos de software***. Dissertação de mestrado. (Departamento de Engenharia de Computação e Sistemas Digitais), Escola Politécnica da Universidade de São Paulo, São Paulo - SP, 2006.

SANDHOF Karen. ***Fatores humanos no processo de desenvolvimento de software: Um estudo visando qualidade***. Dissertação de mestrado. Escola Politécnica da Universidade de São Paulo, São Paulo - SP, 2004.

SILVA, L., M. da. ***A importância do gerenciamento do tempo nos projetos***. Artigo do acervo da IETEC – Instituto de Educação Tecnológica, Belo Horizonte – MG, 2008.

SPINOLA, Rodrigo O.; ARAUJO, Marco A. P.; SPINOLA, Eduardo O. ***Engenharia de software magazine - Qualidade de software: Entenda os principais conceitos sobre testes e inspeção de software****. DevMedia, 2007.*

SPINOLA, Rodrigo O.; KALINOWSKI, M.; TRAVASSOS, G., H. ***Uma infra-estrutura para integração de ferramentas CASE.*** Artigo apresentado no XVIII Simpósio Brasileiro de Engenharia de Software; 2004.

SOMMERVILLE, Ian. ***Engenharia de Software***. Tradução de Software Engineering, 6ª ed., São Paulo: Editora Pearson Education do Brasil, 2003.

VAVASSORI, F., B.; SOUZA, E., W.; FIAMONCINI, J., C. ***Ferramentas CASE para gerenciamento de projetos e métricas de software***. Artigo apresentado no XV Simpósio Brasileiro de Engenharia de Software; 2001.

ANEXO 1 – QUESTIONÁRIO

Informações para o(a) participante voluntário(a):

Você está convidado(a) a responder este questionário anônimo que faz parte da coleta de dados da pesquisa "GESTÃO DE TEMPO: UM ESTUDO SOBRE A PRODUTIVIDADE DOS DESENVOLVEDORES DE SOFTWARE", responsabilidade do pesquisador FABIANO LOPES VIANA sob orientação da professora Msc. FABRICIA PIRES SOUZA TIOLA. Caso você concorde em participar da pesquisa, leia com atenção os seguintes pontos: a) você é livre para, a qualquer momento, recusar-se a responder às perguntas que lhe ocasionarem constrangimento de qualquer natureza; b) você pode deixar de participar da pesquisa e não precisa apresentar justificativas para isso; c) não será pedido sua identidade, ou seja, será mantido em sigilo; d) você será informado de todos os resultados obtidos com a pesquisa.

1. Cargo ocupado: (marcar somente uma opção)

- Analista
- Desenvolvedor
- Gerente
- Projetista
- Outro :_____

2. Tempo que ocupa este cargo atual: (marcar somente uma opção)

- 0 a 1 ano
- 1 a 2 anos
- 2 a 3 anos
- 3 a 5 anos
- 5 a 10 anos
- Mais de 10 anos

3. A quanto tempo você trabalha com sistemas? (marcar somente uma opção)

- 0 a 1 ano
- 1 a 2 anos
- 2 a 3 anos
- 3 a 5 anos
- 5 a 10 anos
- Mais de 10 anos

4. Você tem conhecimento a respeito do termo “crise do software”? (marcar somente uma opção)

- Sim
- Não

5. Os projetos de software na qual você participa são sempre entregues no tempo pré-determinado? (marcar somente uma opção)

- Sim
- Não
- Não sei responder

5.1. Se a resposta da pergunta número 5 for Sim, diga quais os métodos são utilizados pela empresa para que se consiga tal resultado.

_____

5.2. Se a resposta da pergunta número 5 for Não, diga qual a sua idéia para minimizar este problema.

_____

6. A empresa que você atualmente trabalha planeja bem seus produtos e prestação de serviços? (marcar somente uma opção)

- Sim
- Não
- Não sei responder

6.1. Se a resposta da pergunta número 6 for Sim, diga como é feito este planejamento.

_____

7. Quais destas metodologias relacionadas com métricas de tempo de desenvolvimento de Software você já ouviu falar? (marcar uma ou várias opções)

- COCOMO
- Diagrama de Gantt
- Orientadas ao tamanho
- PERT/CPM
- Pontos-por-função
- Nenhum
- Não sei do que se trata essas metodologias
- Outro :_____

8. Você tem noção de quantas KLOC (mil linhas de código) você implementa por mês? (marcar somente uma opção)

- De 1 a 10 KLOC
- De 10 a 15 KLOC
- De 15 a 20 KLOC
- Mais de 20 KLOC
- Não sei responder

9. Você conhece alguma ferramenta *CASE* para a gestão de tempo? (marcar somente uma opção)

- Sim
- Não
- Não sei responder

10. Destas ferramentas, quais você já ouviu falar? (marcar uma ou várias opções)

- Clockingit
- MS Project Standard
- OpenProj
- Redmine
- Workbench
- Nenhuma
- Outro :_____

11. A empresa na qual você trabalha utiliza alguma ferramenta de gestão de projetos? (marcar somente uma opção)

- Sim
- Não
- Não sei responder

11.1. Se a resposta da pergunta número 11 for Sim, diga qual.

_____

12. A empresa na qual você trabalha utiliza alguma ferramenta de gestão de tempo? (marcar somente uma opção)

- Sim
- Não
- Não sei responder

12.1. Se a resposta da pergunta número 12 for Sim, diga qual.

_____

13. Assinale o grau de importância que você atribui ao gerenciamento de produtividade para desenvolvimento de softwares. (marcar somente uma opção)

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

14. Em sua empresa existe política para incentivo a produtividade no desenvolvimento de softwares? (marcar somente uma opção)

- Sim
- Não
- Não sei responder

15. Se existissem políticas para incentivo mais claras você acha que os desenvolvedores iriam ter maior produtividade no trabalho? (marcar somente uma opção)

- Sim
- Não
- Não sei responder

16. Quais os erros, problemas, mais comuns que ocorrem na empresa na qual você trabalha? (marcar uma ou várias opções)

- Erros relativos ao produto
- Erros relativos à processos
- Erros relativos à pessoas
- Erros relativos à tecnologia
- Nenhum
- Não tenho idéia do que se trata esses erros
- Outro :_____