Estimativas realísticas em projetos de software – Uma aplicação prática para empresas

Cleomar A. Gonçalves, André Luiz Alves.

Coordenação de Pós-graduação Latu Sensu – PROPE / PUC – GOIAS
Especialização em Qualidade e Gestão de Software
Avenida Universitária, 1440 Setor Universitário CEP: 74605-010 Goiânia – GO – Brasil.

cleomartech@gmail.com, andre.luiz@pucgoias.edu.br

Abstract. *In 1986, The Standish Group published the CHAOS Report, in which the software construction was compared to bridge construction. Usually bridges are delivered on time, on budget and do not fall. Software is typically delivered after the deadline, over budget and contains defects. According to the report, the reasons for the successful delivery of bridges reside in extremely detailed design, closed scope and little or no possibility of change. We intend in this article to extend the comparison, and demonstrate the applicability of techniques of architecture in building successful software on time and under the budget planned, delivering the scope originally proposed.*

Resumo. *Em 1986, The Standish Group publicou o CHAOS Report, no qual a construção de software foi comparada à construção de pontes. Normalmente pontes são entregues no prazo, no orçamento e não caem. Software, geralmente, é entregue fora do prazo, acima do orçamento e contém defeitos. Segundo o relatório, os motivos de sucesso na entrega de pontes residem no projeto extremamente detalhado, no escopo fechado e em pouca ou nenhuma possibilidade de mudanças. Pretendemos neste artigo estender a comparação, e demonstrar a aplicabilidade de técnicas da arquitetura na construção bem sucedida de software no prazo e orçamento planejados, entregando o escopo proposto inicialmente.*

1. Introdução

Esboçar é uma palavra de origem italiana, *sbozzare*, que transmite a idéia de criar um rascunho para efeito de estudo preliminar antes da execução de um projeto. Antes de fazer um desenho, por exemplo, podemos fazer um esboço, que nada mais é do que um modelo em traços simples do resultado final. Além de facilitar a realização de projetos ou idéias, um esboço pode ser útil para conhecermos onde serão necessárias modificações ou adaptações. As diversas áreas do conhecimento humano têm feito a utilização de esboços como passo inicial da realização de projetos. A pintura e o desenho usam rascunhos, o cinema utiliza os "*storyboards*", a arquitetura utiliza o projeto arquitetônico, para citar apenas alguns exemplos.

A despeito de outras áreas, fazer uma estimativa inicial tem sido um dos maiores desafios para empresas desenvolvedoras de software. A maioria dos processos e modelos de processo de desenvolvimento não soluciona a necessidade desta área de também poder esboçar o escopo e estabelecer uma estimativa real de custos e tempo para a realização de um projeto.

Em 1986, o presidente da Transarc Corporation, Alfred Spector, coautor de um artigo [1] que comparou a construção de pontes ao desenvolvimento de software, definiu que as pontes, geralmente, são construídas no prazo e orçamento planejados e não caem. Já os projetos de software, além de não satisfazerem essas premissas, contêm defeitos e simplesmente param de funcionar. Os motivos apontados para a entrega bem sucedida de pontes são as especificações de projeto (*design*) bem detalhadas, o "congelamento" e a inflexibilidade do escopo do projeto. No referido artigo, o *Standish Group* demonstrou que 31.1% dos projetos de software foram cancelados antes de seu término. Projetos com sucesso parcial, ou seja, que terminaram acima do orçamento, extrapolando prazos e com menos funções entregues que o planejado inicialmente, totalizaram 52.7%. A previsão de estouro de orçamento destes projetos chegou a 189%. O custo de oportunidades perdidas não foi mensurado, mas poderia facilmente chegar a trilhões de dólares. Os demais 16,2% dos projetos foram bem sucedidos. A Tabela 1 apresenta os números da pesquisa:

Tabela 1. Chaos Report 1986

Tipo	Descrição	%
Projetos de Sucesso	Projeto completado no prazo, dentro do orçamento e com todas as funções inicialmente especificadas.	16,2
Projetos com Sucesso parcial	Projeto completado acima do prazo, acima do orçamento e com menos funções que o especificado inicialmente.	52,7
Projetos Cancelados	Projeto cancelado em algum ponto durante o ciclo de desenvolvimento.	31,1
	Total	**100**

Em sua revisão de 2009 [2], o CHAOS Report traz os seguintes números:

Tabela 2. Chaos Report 2009

Tipo	Descrição	%
Projetos de Sucesso	Projeto completado no prazo, dentro do orçamento e com todas as funções inicialmente especificadas.	32
Projetos com Sucesso parcial	Projeto completado acima do prazo, acima do orçamento e com menos funções que o especificado inicialmente.	44
Projetos Cancelados	Projeto cancelado em algum ponto durante o ciclo de desenvolvimento.	24
	Total	**100**

Nos últimos 25 anos, houve uma grande evolução no desenvolvimento de software, a qual resultou em um conjunto de boas práticas. Passamos pela febre das metodologias próprias de desenvolvimento; pela adoção da orientação a objetos em todas as etapas de desenvolvimento; pelos padrões de projeto (*design patterns*); pelo surgimento da arquitetura de software; pela popularização do gerenciamento de projeto através do PMI e PMBOK [9]; pelos modelos de maturidade tais como o CMMI [3]; pelas técnicas de medição de software como a APF [4]; pelos ciclos de vida de projeto interativos / incrementais, mais comumente usados no início da década passada pelos praticantes de RUP [11] e atualmente através de alguns dos métodos ágeis. Embora neste período a taxa de projetos de sucesso tenha aumentado de 16% para 32%, ainda estamos distantes de um percentual aceitável.

Acreditamos que as técnicas, métodos e processos citados anteriormente tenham contribuído para esta melhoria, entretanto, a produção de software não tem atendido às premissas básicas: efetuar a entrega respeitando o prazo e orçamento estimados inicialmente, atendendo ao escopo requerido pelo adquirente e com a qualidade inegociável encontrada nas pontes. Voltando à comparação inicial, não é plausível se aprovar a construção de uma ponte que foi o resultado de uma ou duas reuniões entre adquirente e fornecedor, o qual em duas semanas de trabalho confeccionou a proposta técnica e comercial. Mas, infelizmente isso é o que ocorre na maioria dos projetos de construção de software. Os adquirentes, devido a um planejamento estratégico de TI deficiente ou inexistente ou mesmo por desconhecimento da importância deste, tem a expectativa de que quanto mais cedo a construção se iniciar, mais rápido será terminada. Isto cria uma ilusão de que o fornecedor que promete entregar logo é o mais "ágil", sendo assim o melhor para o projeto. Por outro lado, os fornecedores pressionados por um mercado competitivo, pela necessidade de fechamento de novos contratos e geração de lucro aos acionistas ou mesmo por desconhecimento de algumas das boas práticas citadas anteriormente, acabam por fazer um levantamento de informações deficitário, afetando as estimativas de prazos e custo dos projetos. Muitos fornecedores preferem ter o contrato formalizado com o adquirente já sabendo que este, em vista de suas necessidades, vai acabar cedendo à renegociação contratual.

Conforme a definição do CHAOS *Report* [1], percebemos que a entrega de projetos bem sucedidos envolve outros interesses decorrentes do relacionamento entre adquirentes e fornecedores e de seus objetivos e necessidades individuais. Entretanto com este artigo pretendemos estabelecer uma base, comparando a dinâmica do projeto arquitetônico da arquitetura com a engenharia de software, para que os adquirentes possam aprimorar seu planejamento estratégico de TI e realizar o que chamaremos daqui em diante de **anteprojeto de software**. Através das informações obtidas pelos anteprojetos de software de uma organização, será possível priorizar os futuros projetos ou embasar a escolha entre compra e desenvolvimento de soluções. Aos fornecedores intencionamos delinear o processo para utilização na orçamentação de projetos de software, o qual deverá ser apresentado a adquirentes ainda não conscientizados de sua importância. Além da eficiência no resultado, o anteprojeto de software aqui proposto integrará o portfólio de serviços prestados pelo fornecedor.

O restante do artigo está organizado da seguinte maneira: A seção 2 apresenta alguns conceitos básicos sobre a arquitetura para facilitar nossa correlação com a engenharia de software; A seção 3 apresenta os detalhes sobre o anteprojeto de software, enquanto que a seção 4 conclui o trabalho.

2. Conceitos básicos

As pontes, conforme as conhecemos, são atribuídas aos romanos devido a sua necessidade de expandir o império e ligá-lo à capital. A primeira ponte romana teria sido construída no rio Tibre por volta do ano 621 a.c. e foi chamada de *Pons Sublicius* ("ponte das Estacas") [12]. Assim, embasados em 2500 anos de lições aprendidas, vamos aplica-las à construção de software, conforme adiante expendido.

O conhecimento acumulado na arquitetura culminou na especificação de normas internacionais de construção e em sua localização em muitos países para atender necessidades locais. No Brasil podemos citar duas normas publicadas pela ABNT – Associação Brasileira de Normas Técnicas – as quais fornecerão embasamento para as atividades inerentes deste artigo: NBR 13531 [5] e NBR 13532 [6]. Ambas as normas fixam e detalham as condições exigíveis para a elaboração de projetos de arquitetura para a construção de edificações, complementando-se mutuamente.

A NBR 13531 [5] no item 2.4 “Etapas das atividades técnicas do projeto de edificação e de seus elementos, instalações e componentes” descreve, dentre outras, as seguintes atividades que consideraremos para o nosso estudo:

a) Levantamento:

   Etapa destinada à coleta das informações de referência que representem as condições preexistentes, de interesse para instruir a elaboração do projeto,...

b) Programa de necessidades:

   Etapa destinada à determinação das exigências de caráter prescritivo ou de desempenho (necessidades e expectativas dos usuários) a serem satisfeitas pela edificação a ser concebida.

c) Estudo de Viabilidade:

Etapa destinada à elaboração de análise e avaliações para seleção e recomendação de alternativas para a concepção da edificação e de seus elementos, instalações e componentes.

d) Estudo Preliminar:

Etapa destinada à concepção e à representação do conjunto de informações técnicas iniciais e aproximadas, necessários à compreensão da configuração da edificação, podendo incluir soluções alternativas.

e) Anteprojeto:

Etapa destinada à concepção e à representação das informações técnicas provisórias de detalhamento da edificação e de seus elementos, instalações e componentes, necessárias ao inter-relacionamento das atividades técnicas de projeto e suficientes à elaboração de estimativas aproximadas de custos e de prazos dos serviços de obra implicados;

O professor Irajá Gouvêa faz os seguintes comentários sobre algumas das etapas citadas acima em seu site “Arquitetando” [7]:

a) Levantamento de dados:

O cliente demonstra seus objetivos e necessidades. As características do terreno são estudadas (dimensões, solo, escritura, ventos, etc.).

b) Estudo preliminar:

Através das informações obtidas no "Levantamento de dados", o arquiteto já tem condições para fazer um esboço inicial do projeto. Deve ser acompanhado de perto pelo cliente, já que trata-se do início da elaboração da planta; antes de continuar o projeto, o arquiteto nesta fase deve receber a aprovação do cliente.

c) Anteprojeto:

Nesta etapa, as dimensões e características da obra serão definidas. Será desenvolvido o projeto com a elaboração da planta-baixa de cada pavimento, contendo informações de cada ambiente, pilares, cálculo das áreas e etc. A volumetria, estrutura, planta de cobertura e instalações gerais serão definidas. O cliente deve aprovar o anteprojeto, para que o arquiteto passe para a próxima etapa.

3. Anteprojeto de software

Em um anteprojeto de software usaremos algumas das disciplinas da engenharia de software, tais como: gerenciamento de projetos, engenharia de requisitos e medição de software. No entanto, a forma de utilizar cada uma dessas disciplinas difere de um projeto de desenvolvimento, principalmente porque objetivo de um anteprojeto não é entregar software e sim um orçamento ou proposta, que vai embasar a tomada de decisão de iniciar um projeto de construção de software. Desta forma um anteprojeto de software se preocupa em responder as perguntas: O quê? Quanto? E Quando?

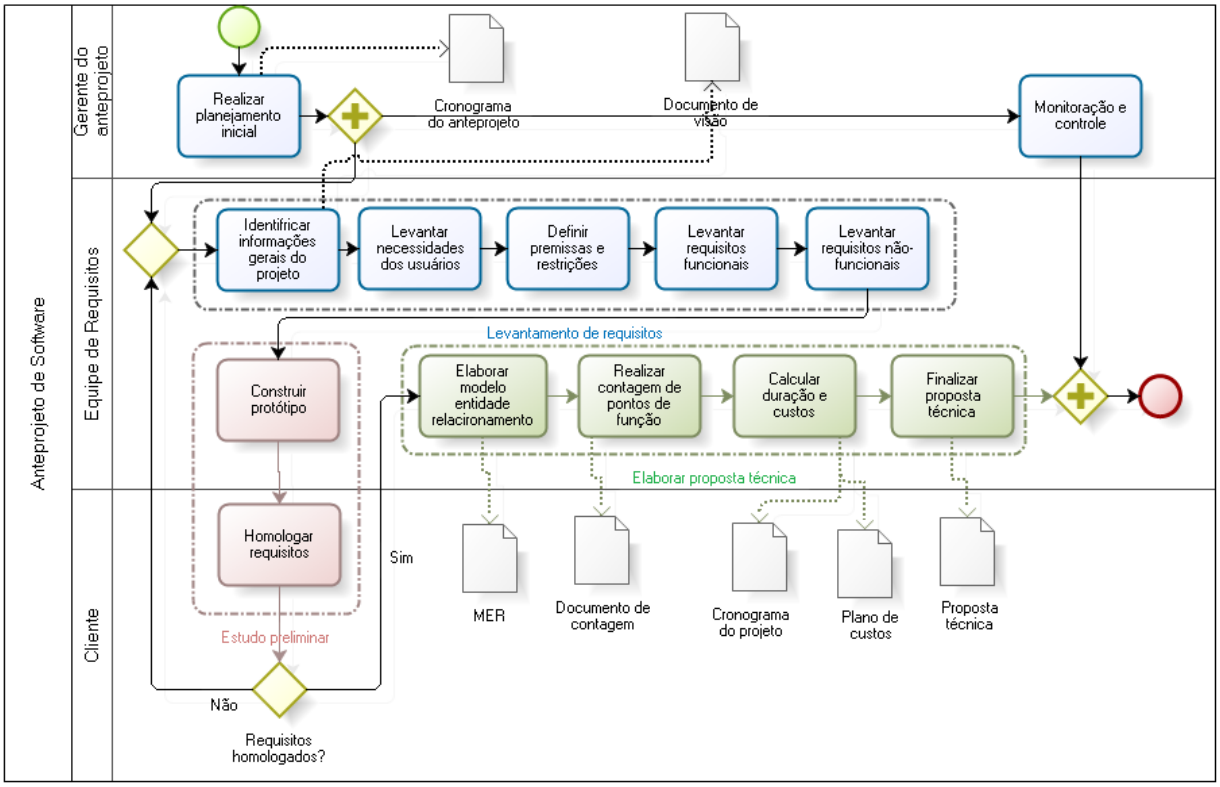

Figura 1. Exemplo de processo de anteprojeto de software

Ao realizar o anteprojeto também precisamos saber quanto este custará e quando terminaremos. É neste contexto que o gerenciamento de projetos nos apoiará com o planejamento do tempo, através de um cronograma de execução e acompanhamento dos custos mediante os apontamentos de atividades da equipe. O objetivo do anteprojeto é investigar as necessidades do cliente para definir a proposta. Mas quanto tempo durará este trabalho? Nos últimos dez anos, tivemos a oportunidade de gerenciar diversos anteprojetos de software. Pela nossa experiência prática a duração de um anteprojeto irá variar de 30 a 60 dias úteis. Este é o período necessário para se estimar um software de até 3000 pontos por função de complexidade média a alta. Outro fator que influenciará nos custos do anteprojeto é a equipe alocada. A equipe mínima é composta de um gerente de projeto e dois analistas de requisitos. Mais analistas de requisitos podem ser acrescentados conforme a necessidade.

Desenvolver os trabalhos com o envolvimento do adquirente é de suma importância. O CHAOS Report [2] aponta como principal fator de sucesso, em projetos bem sucedidos, o envolvimento do usuário, com 15,9%. O método JAD – *Joint Application Design* [8] favorece fortemente o desenvolvimento em conjunto com os usuários e a sua participação ativa desperta um vínculo forte de propriedade do trabalho sendo desenvolvido. Isto facilita a obtenção correta de informações, a priorização de objetivos, agiliza a tomada de decisão e vence a resistência natural do ser humano a mudanças.

O método JAD [8] tem em seu cerne a realização de reuniões de levantamento de informações. Sugerimos que na definição dos requisitos, sejam realizadas no mínimo três reuniões semanais e no máximo quatro. Nestas o gerente do anteprojeto será o

“maestro” dos trabalhos, um dos integrantes da equipe será o documentador, responsável pela ata de reunião, e os demais analistas registrarão requisitos através de anotações, protótipos e outras técnicas de registro de requisitos.

É importante que as anotações de toda a equipe se complementem ao final da reunião, a ata não deve ser a única fonte de informações. As atas das reuniões de anteprojeto são basicamente atas que registram requisitos, por isso devem ser detalhadas e conter o máximo de informações discutidas durante as reuniões, algo bem diferente da prática cotidiana onde as atas registram apenas as conclusões tomadas durante a reunião. Se os usuários consentirem e a equipe achar mais prático as reuniões poderão ser gravadas, o que dispensaria o uso de uma ata detalhada.

Nunca realize reuniões no ambiente de trabalho do usuário, pois haverá muitas interrupções. O ideal é que estas sejam realizadas em uma sala com quadro, projetor, “Flip Chart” e espaço nas paredes para fixar as definições feitas pela equipe. O acompanhamento de algumas atividades da rotina de trabalho, etnografia, poderá ser realizado após o levantamento, para enriquecer o entendimento das necessidades do usuário.

As reuniões devem ser realizadas preferencialmente pela manhã e as atividades de construção dos artefatos à tarde. Isto permite que a equipe do anteprojeto possa trabalhar na construção dos artefatos logo após as definições com os usuários. Esta é uma lição aprendida importante, a realização de reuniões o dia todo, para posterior construção de artefatos, é improdutiva, pois muitas informações são esquecidas quando se decorre muito tempo entre a realização das reuniões e as atividades de construção.

Uma premissa muito importante para o sucesso do anteprojeto é a sua aplicação para estimar software onde os processos de negócios estão definidos. Não é objetivo de um anteprojeto definir os processos de negócio. Para empresas onde seja necessária tal definição, sugerimos que primeiro seja executado um projeto de modelagem de processos de negócio.

Ao analisarmos as atividades do Anteprojeto de Software poderemos correlacioná-las as atividades descritas na NBR 13531 com a ajuda da Tabela 3.

Tabela 3. Correspondência entre atividades da NBR 13531 [5] e o Anteprojeto de Software

NBR 13531	Anteprojeto de Software
- Levantamento; - Programa de necessidades.	- Levantamento de requisitos.
- Estudo de Viabilidade; - Estudo Preliminar.	- Estudo Preliminar.
- Anteprojeto.	- Elaboração da Proposta.

3.1. Levantamento de requisitos

A alocação das pessoas para a reunião de levantamento de requisitos é parte importante do planejamento. É preciso convocar um número ideal de participantes que conheçam sobre as necessidades que serão atendidas pelo software desejado. Alguns desses participantes devem ter poder de decisão para endossar as definições do grupo. Conforme comentamos anteriormente é importante ter uma quantidade de analistas de requisitos suficiente para registrar as informações do levantamento, pelo menos dois, e discutir a diversidade de pontos de vista. Entretanto ter muitas pessoas não significa eficiência e sim ter as pessoas certas.

O principal documento a ser produzido, como resultado das diversas reuniões de levantamento, é o documento de visão. O RUP [11] define este documento da seguinte forma:

> O documento de Visão fornece uma base de alto nível - algumas vezes contratual - para os requisitos técnicos mais detalhados. Também pode conter uma especificação de requisitos formal. O documento de Visão captura restrições de design e requisitos de nível muito elevado para que o leitor possa compreender o sistema a ser desenvolvido. Ele fornece informações para o processo de aprovação do projeto e, portanto, está intrinsecamente relacionado ao Caso de Negócio. Ele comunica os principais questionamentos relacionados ao projeto e funciona como um regulador com base no qual todas as decisões futuras deverão ser validadas. (RUP, Artefato: Visão)

Assim o modelo de documento de visão fornecido pelo RUP [11] é um bom exemplo a ser utilizado como base para a criação de seu próprio modelo.

Podemos agrupar as informações a serem registradas no documento de visão nos seguintes tópicos:

a) Informações gerais do projeto;
b) Necessidades dos usuários;
c) Premissas e restrições;
d) Requisitos Funcionais;
e) Requisitos Não-Funcionais.

3.1.1. Informações gerais do projeto

O objetivo deste tópico é fornecer à equipe de anteprojeto uma visão comum dos problemas enfrentados pelo cliente, os envolvidos e o ambiente de trabalho.

Ao buscar uma justificativa para a realização do projeto de desenvolvimento de software as principais questões a serem respondidas são:

- Qual ou quais os problemas pretendemos resolver com o software a ser desenvolvido?
- Quem são as pessoas afetadas pelo problema e respectivamente pelo software a ser desenvolvido?

- A convivência com o problema causa que impactos na organização? Insatisfação dos clientes? Percepção de falta de qualidade? Insatisfação dos funcionários? Perda de receitas?
- Qual seria uma solução bem sucedida para resolver os problemas enfrentados?

É preciso deixar claro tanto os objetivos do projeto quanto os não-objetivos, ou seja, aquilo que o projeto não se propõe a fazer. É muito comum alguns clientes argumentarem que determinada necessidade ou objetivo era óbvio. Desde o primeiro dia do anteprojeto é preciso haver um acordo com o cliente de que não existem obviedades, e sim os requisitos que estão sendo construídos em conjunto com o cliente.

Identifique não só os usuários, mas também todos os demais envolvidos ou afetados pelo software proposto. O quanto antes identificarmos os possíveis focos de resistência ou aqueles dispostos a apoiar o projeto maior será a probabilidade de termos um projeto bem sucedido. Isto possibilitará um planejamento estratégico para diminuir a resistência e aumentar a quantidade de apoiadores.

O entendimento sobre o ambiente no qual o software proposto funcionará é primordial. É de suma importância entendermos quais plataformas são utilizadas atualmente na empresa, quais outros aplicativos estão em uso para resolver o problema e se será necessário que o software proposto interaja com alguns dos aplicativos atuais. A infraestrutura atual do cliente precisa ser avaliada também, pois pode ser que não consiga suportar um novo software. Se há necessidade de expansão da infraestrutura é melhor que seja identificada no princípio do projeto e não na entrega.

3.1.2. Necessidades dos usuários

Levante os principais problemas enfrentados e a melhor solução para estes de acordo com o ponto de vista dos usuários ou demais envolvidos. É importante esclarecer quais são as causas de cada problema, como ele é resolvido atualmente e como os usuários e envolvidos gostariam que fosse solucionado. Neste trabalho vale a premissa de que precisamos fazer distinção entre necessidades e desejos. E para isso nada melhor que a priorização de atendimento das necessidades. Podem ser usadas escalas simples como prioridade alta, média e baixa, por exemplo.

3.1.3. Premissas e restrições

São os mesmos conceitos expostos na literatura de gerenciamento de projeto, onde as premissas devem ser condições que precisam ser satisfeitas para que o projeto tenha sucesso e restrições são imposições com as quais teremos que conviver durante a execução do projeto.

3.1.4. Requisitos Funcionais

Trata-se das funcionalidades do software das quais se esperam a geração de resultados aos usuários e benefícios ao negócio do cliente. A grande diferença com relação a um projeto de desenvolvimento é o grau de abstração que aplicaremos no levantamento. Como esses requisitos estarão registrados no Documento de Visão, sendo este revisado por uma ampla variedade de pessoas envolvidas, o nível de detalhamento precisa ser

genérico o bastante para que todos possam compreendê-lo. No entanto, é preciso ter detalhes suficientes para fornecer à equipe as informações necessárias para a criação do protótipo funcional, do qual falaremos mais adiante. Note que nosso objetivo não é construir casos de uso ou registrar detalhadas de regras de negócio. Precisamos nos concentrar no porque (e não em como) os requisitos serão desenvolvidos. Um código, nome e uma descrição para cada requisito é o suficiente para as necessidades do anteprojeto. Documentos, planilhas e as telas usadas nas aplicações atuais podem servir de complemento ao registro de requisitos funcionais. Associar os requisitos com as necessidades levantadas anteriormente podem nos apoiar na priorização dos requisitos, visto que as necessidades também foram priorizadas. Se desejar, insira outros atributos como estabilidade, benefício, esforço e risco, por exemplo.

Um dos fatores de sucesso atribuídos à construção de pontes é a imutabilidade dos requisitos, algo impossível de acontecer no desenvolvimento de software. Caso o anteprojeto seja aprovado, os requisitos serão refinados através do ciclo normal da engenharia de requisitos [13], durante o projeto de desenvolvimento do software. A grande vantagem é que teremos uma linha de base, resultado da medição com APF, sobre a qual falaremos mais adiante, que apoiará no gerenciamento de mudanças caso o detalhamento de requisitos resulte em alterações. Outro benefício desse escopo preliminar é que o planejamento da engenharia de requisitos no projeto de desenvolvimento poderá ser feito de uma forma bem detalhada, o que aumenta a assertividade no cumprimento dos prazos do projeto.

3.1.4. Requisitos Não-Funcionais

Num anteprojeto é importante entendermos os requisitos Não-Funcionais, pois através deles teremos informações suficientes para calibrar os fatores de ajuste da contagem detalhada de pontos de função.

3.2. Estudo Preliminar

A técnica de prototipação tem ajudado clientes e desenvolvedores entender os requisitos de software reduzindo riscos por identificar os problemas previamente. Dentre os benefícios do uso da técnica podemos citar a redução de equívocos de entendimento sobre o que é “óbvio”, requisitos esquecidos podem ser detectados, requisitos confusos podem ser esclarecidos e antes do início do projeto de desenvolvimento o usuário saberá exatamente como será o software adquirido. Por mais que possamos explicar através de requisitos ou casos de uso, o cliente compreende plenamente aquilo que ele pode ver e experimentar. Vale neste caso o dito popular: “Uma imagem vale mais que mil palavras”.

Nesta etapa teremos informações suficientes para iniciarmos a construção de um protótipo de interface com o usuário, navegável, sem validação de qualquer regra de negócio e descartável.

O protótipo deve ser navegável para que o usuário simule a experiência de utilizar o sistema e fomentar a melhora da usabilidade do software.

Quando dizemos que o protótipo é descartável, estamos dizendo que seu código não será utilizado para construção da aplicação final. Entretanto este será usado para compreensão e homologação dos requisitos; para a medição do software; no projeto de

desenvolvimento como insumo para o detalhamento dos requisitos e criação dos casos de uso. Existem diversas ferramentas no mercado que apoiam a construção deste tipo de protótipo de forma não vinculada às linguagens de programação atualmente no mercado. Isto evita que o usuário final pressione a equipe achando que o protótipo já é o software desejado ou que possa ser convertido neste.

Um ponto de grande ajuda é aprovar previamente com o cliente, um padrão de interface com o usuário. A construção do protótipo ficará mais objetiva e rápida.

A prototipação deve ser realizada junto com o usuário porque, conforme citamos anteriormente, este se sentirá integrado à equipe e dono do produto.

Ao final da prototipação, o documento de visão e o protótipo deverão ser homologados com os usuários e patrocinadores do anteprojeto. Após a homologação poderemos passar para a parte final, a elaboração da proposta técnica.

3.3. Elaboração da Proposta Técnica

Ao chegarmos neste ponto temos quase todos os elementos necessários para medirmos o tamanho funcional do software proposto e a partir dai derivar esforço, duração e custo. Utilizaremos como técnica de medição a análise de pontos de função conforme descrita no Manual de Práticas de Contagem de Pontos de Função do IFPUG – *International Function Points Users Group* [4]. Embora a técnica APF não exija a construção de um protótipo para ser aplicada, vimos que a utilização do mesmo facilita o entendimento do usuário e reduz os possíveis riscos. Além disso, o protótipo será muito útil para a realização da medição em pontos de função. A APF possui como um de seus elementos de contagem o que ela chama de funções de dados, e se refere a um grupo de dados logicamente relacionados e reconhecidos pelo usuário. Podemos contar as funções de dados simplesmente usando as descrições existentes no Documento de Visão, entretanto a construção de um MER – Modelo de Entidade Relacionamento - facilitará o entendimento da equipe de técnicos do projeto durante a medição do software.

Com o tamanho funcional do software proposto em mãos podemos iniciar o cálculo do esforço. Podemos definir o esforço como a quantidade de horas necessárias para construir o software. Assim como o esforço de construção de uma ponte será influenciado pelos materiais utilizados (concreto, cabos, metal) a tecnologia influenciará no cálculo do esforço. Outro fator de influência é a experiência da equipe na tecnologia escolhida. Quando temos uma equipe experiente na tecnologia e que já tenha desenvolvido outros projetos na empresa podemos utilizar os dados históricos como base para a nossa estimativa de esforço. Se a equipe é nova na tecnologia ou não tivermos um histórico de projetos a única opção que temos neste primeiro projeto é usar parâmetros de mercado.

Podemos dizer que o esforço é o produto do tamanho funcional do software pela produtividade da equipe (ESFORÇO = TAMANHO * PRODUTIVIDADE). Digamos que determinada equipe tenha produtividade de dez horas em determinada tecnologia. Se esta equipe fosse desenvolver um projeto de 150 PF (pontos de função) o esforço total do projeto será de 1.500 horas.

A partir do esforço podemos calcular a duração e custo do projeto e naturalmente, somos tentados a fazer um cálculo linear da duração, principalmente se

usarmos uma equipe interna de desenvolvimento. Neste raciocínio podemos dizer que a duração do projeto em dias úteis é a razão do esforço pela capacidade produtiva da equipe (DURAÇÃO = ESFORÇO / CAPACIDADE PRODUTIVA). Por exemplo, em uma equipe com cinco pessoas, trabalhando oito horas diárias temos uma capacidade produtiva de quarenta horas por dia. Se esta equipe estivesse desenvolvendo o software hipotético citado anteriormente, ela o completaria em 37,5 dias úteis. O mesmo raciocínio valeria para o custo, pois temos o custo fixo de cada membro da equipe.

Entretanto, conforme comentamos, planejar a duração e custo de forma linear é uma verdadeira tentação. Não é possível que todos os membros da equipe iniciem desde o primeiro dia do projeto, pois temos perfis diferentes. Por mais que utilizemos um processo com ciclo de vida iterativo / incremental, as disciplinas da engenharia de software tem certa dependência sequencial. Por exemplo, a implementação depende do projeto (design), que depende da análise de requisitos. Ao planejar de forma linear corremos o risco de ter membros da equipe ociosos em certos momentos e sobrecarregados em outros; ambas as situações elevam o custo do projeto.

Para termos duração e custos coerentes, sugerimos que seja aplicada a técnica de WBS – *Work Breakdown Structure* [9][10] para decomposição das atividades de acordo com o processo de desenvolvimento que será utilizado e com as características do produto a ser construído. Sim, estamos falando de uma WBS hibrida onde as atividades a serem executadas refletem o processo de desenvolvimento usado e a organização dos pacotes de entrega está relacionada ao produto a ser construído. Desta forma teremos os elementos necessários para a construção do cronograma. Ao utilizarmos um software de planejamento como o Microsoft Project, poderemos cadastrar os membros da equipe do projeto com seu respectivo custo e em seguida atribuí-los as atividades. Quando informarmos o trabalho (esforço) de cada atividade, obteremos a informação de duração da atividade. O encadeamento das atividades nos dará a duração final do projeto como um todo. Desta forma teremos de uma forma confiável o prazo final de execução do projeto e seu respectivo custo.

Entretanto ficou uma questão em aberto: Como distribuir o esforço durante o planejamento das atividades do projeto? Medimos o software e sabemos exatamente quantos pontos de função tem cada funcionalidade, portanto sabemos o esforço em horas para construir cada uma delas. Na Tabela 4 temos a distribuição do esforço em fases, de acordo com o RUP [11], e uma correlação com as disciplinas da engenharia de software. Considerando uma funcionalidade com esforço total de 100 horas, os requisitos desta funcionalidade consumirão 11,10% deste tempo, ou seja, gastaremos cerca de onze horas para detalharmos seus requisitos. Esta é a sistemática para distribuirmos o esforço ao longo de todas as atividades do projeto. Logicamente a experiência e a opinião da equipe são balizadores indispensáveis para o planejamento e deverão ser ricamente utilizadas.

Tabela 4. Fases x Disciplinas

	Fases				
Disciplina	**Concepção**	**Elaboração**	**Construção**	**Transição**	**Total**
Gerenciamento de Projeto	0,70%	2,40%	6,50%	1,40%	**11,00%**
Configuração e Mudanças	0,50%	1,60%	3,25%	0,50%	**5,85%**
Requisitos	1,90%	3,60%	5,20%	0,40%	**11,10%**
Análise e Projeto	0,95%	7,20%	10,40%	0,40%	**18,95%**
Implementação	0,40%	2,60%	22,10%	1,90%	**27,00%**
Testes	0,40%	2,00%	15,60%	2,40%	**20,40%**
Implantação	0,15%	0,60%	1,95%	3,00%	**5,70%**
Total	**5,00%**	**20,00%**	**65,00%**	**10,00%**	**100,00%**

4. Conclusão

Apresentamos neste artigo que é possível para o desenvolvimento de software, assim como em outras áreas do conhecimento humano, a aplicação da prática de esboçar.

Percebemos que em vinte e cinco anos de evolução nas técnicas, métodos e processos de desenvolvimento de software, dobramos nossa capacidade de entrega bem sucedida de software, entretanto conseguimos chegar ao ínfimo percentual de 32%, de acordo com o CHAOS Report [2].

Levantamos a questão, a ser estudada em um universo maior de projetos, de que o problema pode não estar no desenvolvimento de software e sim em estimativas iniciais subestimadas e que menosprezam um grande número de fatores de influência externos. Nos últimos dez anos, tivemos a oportunidade de acompanhar a evolução de oito projetos desenvolvidos por determinado cliente. Para apenas seis destes foi realizado o anteprojeto, sendo todos deste conjunto finalizados de forma bem sucedida, mesmo havendo mudanças de escopo ao longo do projeto que foram controladas através de técnicas de gerenciamento de mudanças. Os outros dois projetos, para os quais não se realizou o anteprojeto, foram entregues com o dobro do tempo planejado e com estouro de orçamento, pois não havia uma linha de base bem definida para o escopo. Em outro cliente, que já havia determinado um prazo e orçamento para o desenvolvimento de um projeto com escopo fracamente delimitado, obtivemos um aumento de quatrocentos por cento de prazo e trezentos por cento de custo após a realização do anteprojeto.

Alicerçados na engenharia de software, correlacionamos o projeto arquitetônico da arquitetura com um processo de anteprojeto de software. Embora o anteprojeto de software demore um pouco mais que a pré-venda tradicional acreditamos que traz mais segurança para ambas as partes, o adquirente e o fornecedor, sendo este um serviço prestado pelo fornecedor que poderá ser cobrado do adquirente. Ao adquirente apresentamos a opção de realizar o anteprojeto com um fornecedor e ter insumos para orçar o projeto com outros fornecedores com a garantia de que todos estarão precificando a mesma solução. Indo um pouco além, o adquirente pode desenvolver anteprojetos para todos os softwares constantes em seu planejamento estratégico e assim, com base em valores financeiros e de mercado, priorizar o desenvolvimento de determinado software em detrimento de outro.

5. Referências

[1] - THE STANDISH GROUP – **The CHAOS Report.** West Yarmouth, MA: Standish Group International Inc. 1986.

[2] - THE STANDISH GROUP – **The CHAOS Report.** West Yarmouth, MA: Standish Group International Inc. 2009.

[3] - SOFTWARE ENGINEERING INSTITUTE - SEI. ***CMMI® for Development, Version 1.2***, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, agosto, 2006.

[4] - IFPUG - **Function Point Counting Practices Manual: Versão 4.2.1**. International Function Point Users Group, jan. 2005.

[5] - ABNT – ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 13531 – Atividades técnicas – Elaboração de projetos de edificações.** Rio de Janeiro: ABNT, 1995.

[6] - ABNT – ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 13532 – Arquitetura – Elaboração de projetos de edificações.** Rio de Janeiro: ABNT, 1995.

[7] - GOUVEIA, IRAJÁ – **Etapas de um projeto Arquitetônico** - Acessado em 2012 Jul 10. Disponível em: http://www.arquitetando.xpg.com.br/etapas_de_um_projeto arquitetonico.htm

[8] - COSTA, OSVALDO W.D. **JAD - Joint Applications Design – Como Projetar Sistemas de Informações Mais Eficazes e Criativos**. IBPI Press, 1994.

[9] - PROJECT MANAGEMENT INSTITUTE – PMI. **A Guide to the Project Management Body of Knowledge - PMBOK™**, Syba: PMI Publishing Division, 2004.

[10] - VAZQUEZ, C. E.; SIMÕES, G. S; ALBERT, R. M. **Análise de Pontos de Função – Medição, Estimativas e Gerenciamento de Projetos de Software**. Editora Érica, São Paulo, 3.ed. 2005.

[11] – IBM. **– Rational Unified Process** – Acessado em 2012 Jul. 10. Disponível em http://www.wthreex.com/rup/smallprojects/index.htm

[12] – STRUCTURAE – **Pons Sublicius** – Acessado em 2012 Jul. 10. Disponível em http://en.structurae.de/structures/data/index.cfm?ID=s0001258

[13] - SOMERVILLE, IAN. **Software Engineering**, Addison-Wesley, 8th ed., 2007.