UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS PATO BRANCO
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MOISÉS MEIRELLES FILHO

IMPLEMENTAÇÃO DE UM SISTEMA WEB PARA GERAÇÃO DE ESTIMATIVAS DE ESFORÇO EM PROJETOS DE SOFTWARE

PATO BRANCO
2014

MOISÉS MEIRELLES FILHO

IMPLEMENTAÇÃO DE UM SISTEMA WEB PARA GERAÇÃO DE ESTIMATIVAS DE ESFORÇO EM PROJETOS DE SOFTWARE

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Profa. Eliane De Bortoli Fávero

PATO BRANCO
2014

ATA Nº: **259**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **MOISES MEIRELLES DOS SANTOS FILHO.**

Às 13:30 hrs do dia 19 de dezembro de 2014, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Eliane Maria de Bortoli Fávero (Orientadora), Beatriz Terezinha Borsoi (Convidada) e Lucilia Yoshie Araki (Convidada), para avaliar o Trabalho de Diplomação do aluno Moises Meirelles Dos Santos Filho, matrícula 1376640, sob o título **Implementação de um Sistema Web para Geração de Estimativas de Esforço em Projetos de Software** ; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 14:05 hrs foi encerrada a sessão.

Profa. Eliane Maria de Bortoli Fávero, M.Sc.
Orientadora

Profa. Beatriz Terezinha Borsoi, Dr.
Convidada

Profa. Lucilia Yoshie Araki, M.Sc.
Convidada

Profa. Eliane Maria de Bortoli Fávero, M.Sc
Coordenadora do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

MEIRELLES FILHO, Moisés. Implementação de um sistema web para geração de estimativa de esforço em projetos de software. 2014. 64f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2014.

Esforço em projetos de software está relacionado ao tempo necessário para a realização de cada atividade do projeto. A geração de estimativas de esforço para um projeto normalmente tem sido realizada com base na experiência do gerente de projetos e da equipe, o que muitas vezes torna-a bastante imprecisa, pois falta histórico de projetos anteriores como base para essas estimativas. Sendo assim, esse trabalho apresenta a modelagem e implementação de uma aplicação Web para a geração de estimativas e controle de histórico de software baseada em esforço por atividade. O sistema possibilita aos gerentes de projeto o cadastro de classificações padrão para projetos, para as quais estão vinculados esforços padrão para fatores definidores de prazo, conforme modelo proposto pelo Grupo de Estudos e Pesquisas em Tecnologias de Informação e comunicação (GETIC) da Universidade Tecnológica do Paraná (UTFPR). A implementação desse modelo ainda permite a estimativa de novos projetos de software com base no esforço padrão armazenado e na influência de fatores modificadores de prazo, gerando o esforço ajustado para o projeto. Fatores modificadores de prazo são aqueles relativos ao ambiente de desenvolvimento, aspectos tecnológicos e humanos, os quais influenciam nos resultados de esforço estimado para um projeto, gerando um esforço ajustado mais próximo da realidade de trabalho exercida. Considerando o conceito de esforço em projetos de software, o sistema proporciona também ao desenvolvedor um ambiente para controle do tempo efetivo na realização das atividades através de uma tela para o seu registro e contabilização (*tracker*). A partir dos registros do *tracker* é realizada a contabilização para ser utilizada na comparação de esforço estimado com o realizado, a fim de atualizar o cadastro de esforço padrão e auxiliar em estimativas mais precisas de futuros projetos, além de identificar possíveis melhorias no processo de desenvolvimento. Todas as interfaces e funcionalidades foram implementadas utilizando a linguagem Java, as tecnologias *Web Java Server Faces e Primefaces* e ainda para a estruturação de dados o SGDB *MySql*.

Palavras-chave: Estimativa de esforço. Métricas de software. MPS.BR. *Java Server Faces. PrimeFaces.*

ABSTRACT

MEIRELLES FILHO, Moisés. Implementation of a web system for generation of effort estimation in software projects. 2014. 64f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2014..

Effort in software projects is related to the time required for completion of each project activity. The generation of effort estimates for a project has typically been made based on the project manager's experience and team, which often makes it very inaccurate because historic lack of previous projects as the basis for these estimates. Therefore, this work presents the modeling and implementation of a web application to generate estimates and control software based on historic effort by activity. The system enables the standard ratings registration project managers for projects for which are bound standard efforts to defining the term factors, according to the model proposed by the Group of Studies and Research in Information and Communication Technologies (GETIC) of the University of Technology Paraná (UTFPR). The implementation of this model also allows the estimation of new software projects based on the standard effort stored and influence of term modifying factors, generating a concerted effort to the project. Term modifying factors are those related to the development environment, technological and human aspects, which influence the results of estimated effort for a project, creating a closer concerted effort exerted the work reality. Whereas the concept of effort in software projects, the system also provides the developer an environment for effective control of time in carrying out activities through a screen for registration and accounting (tracker). From the tracker records, the accounting is held for use in the estimated effort compared to that achieved in order to update the standard registration effort, assist in more accurate estimates of future projects, and identify possible improvements in the development process. All interfaces and features were implemented using the Java language, the Java Server Faces Web technologies and Primefaces and for the structuring of the data DBMS MySql.

Palavras-chave: Effort estimation. Software metrics. MPS.BR. *Java Server Faces. PrimeFaces.*

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

APF	Análise por Pontos de Função
AJAX	*Asynchronous Javascript and XML*
CMMI	*Capability Maturity Model Integration*
COCOMO	*Constructive Cost Model*
CRUD	*Create, Read, Update and Delete*
CSS	*Cascade Style Sheet*
GETIC	Grupo de Estudos e Pesquisas em Tecnologias de Informação e Comunicação
HTML	*Hypertext mark-up language*
IEC	*International Electrotechnical Commission*
IFPUG	*International Function Point Users Group*
ISO	*International Organization for Standardization*
JSF	Java Server Faces
LOC	*Lines of Code*
MA-MPS	Método de avaliação para melhoria do processo de software
MN-MPS	Modelo de Negócio para Melhoria do Processo de Software
MPS.BR	Melhoria do Processo de Software Brasileiro
MR-MPS	Modelo de Referência – Melhoria do Processo de Software Brasileiro
MVC	*Model View Control*
NBR	Norma da Associação Brasileira de Normas Técnicas
NESMA	*Netherlands Software Metrics Users Association*
PCU	Pontos de Casos de Uso
PSP	*Personal Software Process*
SEI	*Software Engineering Institute*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*
UTFPR	Universidade Tecnológica Federal do Paraná
XML	*Extensible Markup Language*
XHTML	*eXtensible Hypertext Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais do trabalho, os seus objetivos e a justificativa. Por fim está a organização do texto por meio da apresentação dos seus capítulos.

1.1 CONSIDERAÇÕES INICIAIS

A importância da qualidade em qualquer tipo de organização atualmente é indiscutível. Quando se fala em qualidade é esperado que um produto ou serviço que atenda às necessidades do seu público. Em se tratando do desenvolvimento de software não é diferente. Mas para que se tenha um produto de qualidade é necessário ter um processo de qualidade. Considerando que a primeira atividade do processo de desenvolvimento de software consiste no planejamento do projeto, pode-se dizer que a qualidade no planejamento consiste em estimar escopo, prazo e custo com a maior precisão possível, a fim de não gerar custos adicionais ao projeto.

Esforço em projetos de software está relacionado ao tempo necessário para a realização de cada atividade do projeto. Para gerar estimativas de esforço são necessários pelos menos dois aspectos: experiência (ex. gerente de projeto, equipe de desenvolvimento) e acesso a boas informações históricas sobre realização das atividades do processo de desenvolvimento, ou seja, métricas de tamanho e esforço.

Segundo Sommerville (2007), uma métrica de software é qualquer tipo de medição que se refira a um sistema de software, processo ou documentação relacionada. Sendo assim, esse trabalho visa a modelagem de um sistema para a coleta de métricas de duração das atividades de um projeto de software e geração de estimativas de esforço para futuros projetos visando otimização do controle de produção e sua aplicação no contexto das fábricas de software.

Desta forma, este trabalho pretende abordar conceitos acerca de métricas de software, métricas e sua relação com a qualidade, além de técnicas de estimativa de esforço para projetos de software, visando implementar um sistema *Web* que permita controlar o histórico de desenvolvimento de um projeto de software e gerar

estimativas de esforço para as atividades, o qual será usado na disciplina de Projeto e Desenvolvimento de Software.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

- Implementar um sistema *Web* que permita controlar o histórico de desenvolvimento de um projeto de software e gerar estimativas de esforço para as atividades, o qual será usado na disciplina de Projeto e Desenvolvimento de Software.

1.2.2 Objetivos Específicos

- Permitir o controle de esforço por atividade de um processo de desenvolvimento de software.
- Facilitar o registro da duração efetiva de atividades realizadas no processo de desenvolvimento de software.
- Permitir a geração de estimativa de esforço com base em uma técnica formal.
- Facilitar a coleta de dados históricos sobre o esforço por atividades e por projeto pela gerência de projetos.
- Contribuir para com a disciplina de Oficina de Projeto e Desenvolvimento de Software, dispondo de um mecanismo de auxílio na geração de estimativas dos seus projetos de software.
- Auxiliar na profissionalização e busca pela qualidade de empresas de software, especialmente as de menor porte, por meio do uso de uma ferramenta para estimativa de esforço desenvolvida como resultado de pesquisa.

1.3 JUSTIFICATIVA

Este trabalho visa o registro de esforços padrão para controle de estimativas realizado em um processo de software, de modo que um gerente de projetos poderá usufruir do conhecimento adquirido, para um dos aspectos mais importantes e mais difíceis de ser realizado, o processo de estimativa de esforço, o qual é realizado para a geração do cronograma em um projeto de software.

O sistema apresentado possui soluções ideais ao gerente de projetos na otimização do controle de estimativas e histórico de conhecimento de outros projetos, auxiliando em resultados positivos através da administração do tempo de execução de um projeto com ferramentas adequadas de análise e gerenciamento para projetos e estimativas. Assim o sistema procura trazer melhoria no processo de desenvolvimento com estimativas cada vez mais precisas e uma visão amplificada de projetos em andamento.

Utilizando de tecnologias atuais para o desenvolvimento do sistema, foram utilizados *Java Server Faces (*JSF) associada a implementação do *framework PrimeFaces* as quais facilitam o desenvolvimento de sistemas *Web* a partir de interfaces ricas, conciliando as principais vantagens da linguagem desktop em aplicações *online.*

1.4 ESTRUTURA DO TRABALHO

O trabalho está estruturado em capítulos. No Capítulo 1, o atual, estão contidas as considerações iniciais para o sistema, objetivos e justificativa. O Capítulo 2 contém o referencial teórico que fundamenta o sistema proposto. No Capítulo 3 são apresentados os materiais e método utilizados no desenvolvimento do trabalho. O Capítulo 4 apresenta o escopo, os requisitos e a modelagem para o sistema, além da implementação da interface e funcionalidades para o sistema. No Capítulo 5 são apresentadas as conclusões do trabalho.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico do trabalho, o qual possui foco em métricas de software, qualidade de software e estimativa de esforço em projetos de software. Por fim, apresenta algumas ferramentas destinadas a obtenção de métricas de produto e processos de software.

2.1 MÉTRICAS DE SOFTWARE

Entregar o produto dentro do prazo e desenvolver o projeto com os recursos planejados sempre foi o desafio das empresas de desenvolvimento de software. E diante das constantes mudanças tecnológicas do mercado é necessário possuir controle sobre os resultados obtidos ao longo de cada processo de desenvolvimento, permitindo gerenciá-lo com agilidade. A missão dos gerentes é obter desses dados um melhor planejamento para o avanço necessário em busca da melhoria dos processos da sua empresa.

Na Engenharia de Software encontram-se alguns recursos de auxílio à gerência de projetos, dentre elas estão as métricas de software. A partir desse princípio, é possível definir metodologias e padrões a serem adotados pelas empresas de Tecnologia de informação (TI) para controle de histórico das atividades realizadas, do esforço investido e dos custos, entre outros aspectos do processo de desenvolvimento de um produto de software.

Para definir o termo métrica, torna-se necessário definir o que são indicadores e medidas. Segundo Galorath e Evans (2006):

- Medida: Valor quantitativo da extensão, quantidade, dimensões, capacidade ou tamanho de algum atributo do processo ou produto de software. Exemplos: quantidade de classes-chave de um software ou de linhas de código.
- Medição (indicador): Ato de determinar uma medida. Exemplo: investigação do número de erros em um módulo.

- Métrica: Medida quantitativa do grau de posse de um atributo dado por parte de um sistema, componente ou processo. Exemplo: média de erros detectados na revisão e número de erros encontrados por pessoa.

Para Park (1996), existem quatro razões para medir o software: caracterizar, avaliar, prever e melhorar. Conquistando, assim, a capacidade de avaliar o estado em relação ao plano, prever metas alcançáveis e caracterizar recursos, ambientes e processos a fim de estabelecer linhas base para a melhoria de projetos futuros. Sendo assim, é possível dizer que um produto de software e o processo pelo qual é gerado é medido por muitas razões, dentre elas: indicar a qualidade do produto, avaliar a produtividade dos membros de uma equipe e da equipe como um todo, avaliar possíveis mudanças na metodologia de trabalho de uso de novos recursos, formar uma linha de base para estimativas, dentre outros.

As métricas determinam a realização de um importante processo dentro da gerência de projetos, o planejamento. A partir do planejamento, é feito o levantamento de recursos, esforço, custos, bem como as atividades propriamente ditas, que serão afetadas diretamente pelo controle de histórico da empresa, que resulta nas medidas finais do software. As métricas são recurso importante na geração de estimativas das variáveis contidas no planejamento de um projeto. O que pode ser alcançado a partir da acumulação de experiência e de dados, levando ao refinamento dessas métricas. Essas métricas são comparadas aos dados de projetos a serem desenvolvidos, de forma a avaliar a produtividade e otimizar o desenvolvimento. Uma etapa importante na implantação de um processo de medição é a escolha das medidas e métricas do software. Dentro do contexto de gerenciamento de projetos de software, as principais métricas referem-se à produtividade e qualidade.

Existem diversos tipos de métricas sob o ponto de vista de aplicação, as quais podem ser divididas em duas categorias principais (MARQUES, 2014).

- Métricas de produtividade: permitem avaliar a produtividade do processo, ou seja, avaliar a eficiência do processo em relação a custos, prazos, recursos, entre outros.

- Métricas de qualidade: permitem determinar o quão bem o software atende aos requisitos do usuário em conformidade com os padrões da organização e da equipe de desenvolvimento, avaliando assim a sua qualidade. São

exemplos: taxa de erros, facilidade de manutenção, cumprimento de prazo, produtividade e custos aceitáveis à equipe e organização.

Dentro do contexto de desenvolvimento de produtos de software, ainda é possível classificar as métricas em duas categorias (MARQUES, 2014):

- Métricas diretas: que são realizadas em termos de atributos observáveis, como por exemplo, esforço, tamanho, custo, o total de defeitos registrados durante um determinado período de tempo, entre outros.
- Métricas indiretas ou derivadas, que podem ser obtidas através de outras métricas, como por exemplo, complexidade, confiabilidade, e facilidade de manutenção, as quais são mais difíceis de serem avaliadas e, portanto só é possível obtê-las de forma indireta.

A medição de software auxilia a tomada de decisão, pois através de dados quantitativos é capaz de informar que aspectos do produto atendem ou não ao padrão de qualidade especificado; além de permitir a avaliação dos benefícios de novos métodos e ferramentas de engenharia de software, o entendimento e aperfeiçoamento do processo de produção, a avaliação do retorno do investimento e tornar o gerenciamento de projetos baseado em fatos e não na intuição (ABREU, MOTA e ARAÚJO, 2014). Desta forma, com essas métricas é possível fazer o controle dos processos de planejamento e execução a fim de melhorar o processo e consequentemente a qualidade do produto obtido desse processo.

2.2 MÉTRICAS E QUALIDADE DE SOFTWARE

Métricas em geral possuem um fim em comum, o de garantir a qualidade do software gerado e do processo. Segundo Brooks (1987), qualidade define-se por conformidade aos requisitos. A partir dessa premissa, é essencial determinar o que é a conformidade e como são especificados os requisitos. A qualidade de software é um importante conceito do processo, considerado como parte essencial de todas as outras áreas de conhecimento envolvidas no projeto. Com isso conclui-se que a qualidade de software é um aspecto estático, ou seja, ele independe da execução do

software, diferente de processos como o de teste, o qual está inclusa na qualidade de software.

Segundo a norma da ISO 9000 (INTERNATIONAL, 2000), qualidade é o grau em que um conjunto de características inerentes a um produto, processo ou sistema cumpre os requisitos inicialmente estipulados para estes. A partir disso conclui-se que no desenvolvimento de software, a qualidade do produto está diretamente relacionada à qualidade do processo de desenvolvimento. Assim, para a obtenção de um software com melhor qualidade, é preciso aplicar a melhoria ao processo de desenvolvimento do mesmo.

A realidade das empresas de desenvolvimento, na maioria dos casos, tem sido demandar mais tempo corrigindo seus produtos, do que desenvolvendo melhorias ou inovação à sua demanda. E isso é decorrente da grande porcentagem de retrabalho exercida e falta do controle da qualidade de seus produtos, segundo Kalinowski e Spinola (2008) é uma área preocupante de estudos, a qual necessita de soluções para reduzir o esforço.

Em conformidade com o conceito de Bartié (2002), sobre qualidade de software, é dito que é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos.

Dentre as normas e os modelos de qualidade utilizados na área de software estão: NBR ISO/IEC 12207 (INTERNATIONAL..., 2008), ISO/IEC 15504 (INTERNATIONAL..., 1999), CMMI (SOFTWARE..., 2006) e ainda o modelo brasileiro MPS.BR (SOFTEX, 2011).

A norma *Capability Maturity Model Integration* (CMMI) é um modelo de melhoria de processos que possui práticas necessárias para guiar a melhoria em projetos, processos ou mesmo organizações inteiras. Desenvolvida pela *Software Engineering Institute* (SEI) da Universidade de Carnegie Mellon, é uma evolução do modelo CMM que aborda o processo corporativo integrando diferentes modelos e disciplinas, uma vez que sua premissa é: “a qualidade é influenciada pelo processo”. Foi desenvolvida em 1980 como modelo de avaliação de risco pelo departamento de defesa dos Estados Unidos, que desejava ser capaz de avaliar os níveis de qualidade e previsão das empresas concorrentes a cargos, era de suma importância

conhecer seus processos de desenvolvimento, custos e prazos nos projetos contratados, dando origem a CMM, a qual gerou a CMMI.

O modelo visa ajudar as organizações envolvidas com o desenvolvimento de produtos, prestação de serviço e aquisição para melhorar a capacidade de seus processos por meio de um caminho evolucionário, constituído de três dimensões principais: pessoas, ferramentas e procedimentos. O processo serve para unir essas dimensões.

A ISO/IEC 12207 tem por objetivo fornecer uma estrutura única e comum para os processos de ciclo de vida com o intuito de que o adquirente, fornecedor, desenvolvedor, mantenedor, operador, gerentes e técnicos envolvidos utilizem uma linguagem única e processos bem definidos. A estrutura da norma foi concebida de maneira a ser flexível, modular e adaptável às necessidades de quem a utiliza. A ISO/IEC 12207 aborda os métodos e procedimentos usados nas atividades de desenvolvimento de software, definindo as ferramentas usadas nesse processo, buscando a melhor maneira de guiar o trabalho. Os processos são formados pelas atividades e cada atividade é formada por várias tarefas. São definidas três categorias de processos: de apoio, processos fundamentais e organizacionais.

O modelo ISO/IEC 15504, também conhecido como SPICE, estabelece os princípios, requisitos e metodologias que serão usadas na avaliação do nível de capacidade e maturidade das empresas relacionados ao modelo de processos definido pela norma ISO-12207 (SQS, 2009). Semelhantemente ao modelo CMMI, possui níveis de capacidade para cada processo envolvido na criação de um software. Foi desenvolvido como um *framework* para avaliação de processos de engenharia de software, organização do projeto e negócio.

O padrão de processo denominado Melhoria de processos do Software Brasileiro (MPS.BR), o qual não é somente um modelo de qualidade, mas representa um movimento para melhoria da qualidade, voltado para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software. O modelo ainda é inspirado nos modelos ISO/IEC 12207 e ISO/IEC 15504, ainda compatível com o CMMI. No Brasil, o projeto é desenvolvido e coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX). Suas vantagens consistem no reduzido custo de certificação em relação às normas estrangeiras, sendo uma opção viável para micro, pequenas e médias empresas.

O modelo é dividido em 3 partes: MR-MPS, MA-MPS, MN-MPS. O modelo de Referência para Melhoria do Processo de Software (MR-MPS) é composto de 7 níveis de maturidade, que são (SOFTEX, 2011):

A – Em Otimização,

B – Gerenciado quantitativamente,

C – Definido,

D – Largamente Definido,

E – Parcialmente Definido,

F – Gerenciado,

G – Parcialmente Gerenciado.

Cada nível de maturidade possui cinco processos fundamentais do ciclo de vida os quais são descritos abaixo (SOFTEX, 2011):

- Aquisição: o propósito do processo é obter um produto e/ou serviço que satisfaça a necessidade expressa pelo cliente. O processo inicia com a identificação da necessidade do cliente e termina com a aceitação do produto.
- Fornecimento: é a sustentação para a execução dos processos de desenvolvimento, manutenção e/ou operação do produto ou serviço.
- Desenvolvimento: contém as atividades e tarefas para o desenvolvimento do software, dentre elas: análise de requisitos, projeto, construção, integração, testes e instalação.
- Operação: Possui as tarefas e atividades operacionais aos usuários. O operar o produto de software no seu ambiente e fornece suporte aos clientes.
- Manutenção: O propósito é modificar o produto de software para corrigir falhas, melhorar o desempenho ou ainda adaptá-lo a mudanças do ambiente.

Subdivididos da seguinte forma (SOFTEX, 2011):

- Processos fundamentais: aquisição, gerência de requisitos, desenvolvimento de requisitos, solução técnica, integração do produto, instalação do produto, liberação do produto.

- Processos organizacionais: gerência de projeto, adaptação do processo para gerência de projeto, análise de decisão e resolução, gerência de riscos, avaliação e melhoria do processo organizacional, definição do processo organizacional, gerência quantitativa do projeto, análise e resolução de causas, inovação e implantação na organização.
- Processos de apoio: garantia de qualidade, gerência de configuração, validação, medição, verificação, treinamento.

Cada procedimento do MPS representa um avanço, em seu movimento de padronização e projeção do potencial brasileiro ao mercado de software. Tudo está relacionado à qualidade de software, uma vez que o modelo garante segurança nas instituições, garantindo que a conformidade com o modelo proporciona confiança que os projetos contratados serão entregues no prazo, com os custos combinados com seus clientes e com a qualidade esperada por eles.

O processo de Gerência de Projetos (GPR), definido no nível G do MPS.BR tem o propósito de estabelecer e manter planos que definem as atividades, recursos e responsabilidades do projeto, bem como prover informações sobre o andamento do projeto que permitam a realização de correções quando houver desvios significativos no desempenho do projeto. O propósito deste processo evolui à medida que a organização cresce em maturidade (SOFTEX, 2011). Dentre os resultados esperados por esse processo, está a GPR, que define que o esforço e o custo para a execução das tarefas e dos produtos de trabalho são estimados com base em dados históricos ou referências técnicas.

As estimativas de esforço e custo são, normalmente, baseadas nos resultados de análises utilizando modelos e/ou dados históricos aplicados ao tamanho, atividades e outros parâmetros de planejamento. Destaca-se que dados históricos incluem os dados de custo, esforço e tempo de projetos executados anteriormente, além de dados apropriados de escala para equilibrar as diferenças de tamanho e complexidade.

2.3 TÉCNICAS DE ESTIMATIVA DE ESFORÇO PARA PROJETOS DE SOFTWARE

As métricas de tamanho de software surgiram com o objetivo de estimar o esforço (ex. em pessoa-hora) e o prazo associados ao desenvolvimento de produtos de software. Segundo Macoratti (2014), para saber o custo de um projeto de software é necessário saber o esforço que será necessário para desenvolvê-lo e para determinar o esforço é necessário saber o tamanho do projeto de software. Desta forma, determinar o tamanho de um projeto de software é uma das primeiras e principais atividades relacionadas às estimativas a serem efetuadas durante o ciclo de vida do projeto.

As estimativas de custo e esforço em projetos de software são, comumente, baseadas na predição do tamanho do sistema que será desenvolvido (ABRAHAO; INSFRAN, 2008). A atividade de estimar o tamanho do software é uma das mais críticas dentro do ciclo de vida de desenvolvimento, pois será a partir dela que o projeto será contratualmente regido.

A Figura 1 apresenta um processo de estimativa de prazos e custos e destaca o papel das estimativas de tamanho na geração de estimativas de esforço, assim como as estimativas de produtividade são a base para estimativas de prazos e custos.

Figura 1 - Processo básico de estimativa de software
Fonte: Leite (2007).

Contudo, o desempenho das técnicas aplicadas nesse propósito, por muitas vezes, é bastante incipiente (HAUGEN, 2006). Ainda, segundo Haugen (2006), a

pouca precisão das estimativas causam estouro no orçamento, atrasos, perda de contratos e levam a baixa qualidade do software.

De acordo com Demarco (1991) as três principais maneiras de estimar o tamanho de um projeto de software são:

1. Por analogia – As estimativas de tamanho do projeto atual são baseadas em estimativas já realizadas em projetos similares.
2. Algorítmicas – Visa realizar medições das características do produto e usando uma metodologia algorítmica para converter a medição em uma estimativa de tamanho.
3. Opinião Especializada: especialistas utilizam sua experiência e intuição para estimar (ex. método Delphi, Estimativa de três pontos).

Existem várias técnicas de estimativas de tamanho de software e a seguir são apresentadas, de forma resumida, as mais importantes (MACORATTI, 2014):

- COCOMO (*Constructive Cost Model*) – Modelo desenvolvido para estimar o esforço de desenvolvimento, prazos e tamanho da equipe para projetos de software. Utiliza equações desenvolvidas por Boehm (BARRY, 1981) para prever o número de programadores-mês e o tempo de desenvolvimento. Devem ser realizados ajustes nas equações a fim de representar as influências sobre os atributos, hardware e software durante o ciclo de vida do projeto. Uma desvantagem desta técnica é que os coeficientes da métrica não são aplicáveis a tamanho, ou seja, a produtividade é diferente, o que torna difícil realizar comparações.
- Linhas de Código – (LOC) – A técnica de mensuração por linhas de código é uma das mais antigas medidas de tamanho de projeto de desenvolvimento de software. Ela consiste na contagem da quantidade de número de linhas de código de um programa de software. Além de ser muito simples é também muito fácil automatizar sua implementação, mas, apresenta algumas desvantagens como: a dependência da linguagem de software e do desenvolvedor (PRESSMAN, 1995); ausência de padrão de contagem e o fato de somente poder ser aplicada na fase de codificação.
- Delphi – É uma técnica que se resume à consulta de especialistas de determinada área, em determinada linguagem e/ou determinado assunto para que, usando sua experiência e entendimento do projeto proposto, façam

estimativas devidas. Devem ser feitas várias estimativas do mesmo projeto, pois é comum que elas carreguem influências e tendências dos especialistas. É um método empírico, baseado em experiências profissionais que podem ser subjetivas (BOEHM,1981).

- PSP – *Personal Software Process* – (HUMPHREY, 1995) – É uma técnica derivada do SEI-CMM (*Software Engineering Institute – Capability Matutiry Model*) que foi desenvolvida com a função de capacitar, melhorar e otimizar o processo individual de trabalho. A técnica divide-se em sete etapas, sendo que nas etapas PSP0, PSP0.1 e PSP1 estima-se o tamanho e o tempo necessário para o desenvolvimento do produto.
- PCU – Pontos por Caso de Uso – Técnica criada por Gustav Karner em 1993 como uma adaptação específica dos Pontos de Função para medir o tamanho de projetos de software orientados a objeto. Explora o modelo e descrição do caso de uso, substituindo algumas características técnicas proposta pelos Pontos de Função. É um método simples e de fácil utilização, mas ainda está em fase de pesquisas e não existem regras de contagem padronizadas. Tem sido estudada a aplicação em conjunto da PCU e APF tentando explorar a relação existente entre elas (EDMÉIA, 2004).
- Análise por Pontos de Função (ALBRECHT, 1983) – Busca medir a complexidade do produto pela quantificação de funcionalidade expressa pela visão que o usuário tem do mesmo. O modelo mede o que é o sistema, o seu tamanho funcional e não como este será, além de medir a relação do sistema com usuários e outro sistemas. Pode ser utilizada nas fases iniciais do projeto para estimar seus custos e prazos, seu objetivo é ser independente das tecnologias utilizadas, de modo a estimar o que o software faz e não como foi feito (VAZQUEZ, 2005). Uma característica da Análise de Pontos por Função (APF) é que ela visualiza o sistema a partir da ótica do usuário, visualizando suas telas, relatórios e outros dados obtidos a partir do sistema, considerando não apenas o ser humano usuário, mas qualquer ator que irá interagir com o sistema.
- Contagem de Pontos de Função segundo o NESMA – *Netherlands Function Point Users Group* – (NESMA, 2005). Além do IFPUG, o *NESMA* também promove o uso de pontos de função e publica o seu próprio manual de

contagem complacente com o manual do IFPUG. O manual da NESMA apresenta três tipos de contagens por pontos de função: a contagem indicativa de ponto de função, a contagem estimada de ponto de função e a contagem detalhada de pontos de função. A contagem indicativa é muito usada, nela são identificados os grupamentos de dados relativos à natureza do negócio, conforme a visão do usuário. Estes grupamentos são classificados como internos (mantidos pela aplicação) e externos (referenciados ou consultados pela aplicação).

Também é possível definir estimativas a partir de analogias. Estas são baseadas em histórico de projetos já realizados, dos quais é possível avaliar aspectos como a duração para cada atividade, dependendo das características da pessoa que a realizou e da tecnologia utilizada, por exemplo. Também pode-se tomar como base estimativas já utilizadas anteriormente. É possível conciliar essa estimativa a outras métricas, aumentando ainda mais a precisão das estimativas de tempo dos projetos.

3 MATERIAIS E MÉTODOS

A ênfase deste capítulo está em reportar o que e como será feito para alcançar o objetivo do estágio. Este capítulo pode ser subdividido, inicialmente, em duas seções, sendo uma para os materiais e outra para o método.

3.1 MATERIAIS

O Quadro 1 apresenta as ferramentas e as tecnologias que foram utilizadas para modelar e implementar o sistema.

Ferramenta / Tecnologia	Versão	Referência	Finalidade
Visual Paradigm Community	11.1	http://www.visual-paradigm.com/solution/freeumltool/	Ferramenta para desenvolvimento de UML, diagramas, etc.
Netbeans	8.0	https://netbeans.org/	Ambiente de desenvolvimento
Java EE	7	http://www.oracle.com/technetwork/java/javaee/overview/index.html	Especificação Java para desenvolvimento de aplicações web
Java ServerFace	2.2	http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html https://javaserverfaces.java.net/	*Framework* para desenvolvimento da interface
PrimeFaces	5.0	http://primefaces.org/	Biblioteca de componentes para desenvolvimento da interface
Glassfish	4.0	https://glassfish.java.net/	Servidor *web* de aplicações
MySQL Cluster	5.6	http://www.mysql.com/	Sistema gerenciador de Banco de dados

Quadro 1 - Ferramentas e tecnologias utilizadas

3.2 MÉTODO

O processo de desenvolvimento adotado seguiu um modelo iterativo e incremental, uma vez que a necessidade de modificar e/ou adicionar requisitos ou funcionalidades era encontrada durante o desenvolvimento. Este processo divide o produto em ciclos e em cada ciclo, podem ser identificadas as fases de análise, projeto e implementação (BEZERRA, 2002). Assim, o desenvolvimento acontece de

uma maneira versionada, ou seja, com incrementos de funcionalidades a cada iteração.

O Quadro 2 apresenta as etapas do processo de desenvolvimento do trabalho em questão e as iterações realizadas.

Iterações / Etapas do processo	1ª iteração	2ª iteração	3ª iteração	4ª iteração
Levantamento do referencial teórico	Elaboração de referencial teórico envolvendo aspectos de métricas de software e técnicas de estimativa à elaboração do trabalho			
Levantamento de requisitos	Estudo do modelo adaptado da técnica de Análise de Pontos de Função proposto pelo GETIC	Identificação e classificação dos requisitos		
Análise e projeto		Modelagem de casos de uso e expansão Definição das entidades básicas para o sistema	Adaptação e complementação do DER existente Desenvolvimento de padrões de telas (*tracker*)	Revisões e complementações nos requisitos e modelagem
Implementação			Do *tracker* que fará a contagem de esforço e demais telas padrão.	Das demais funcionalidades do sistema.
Testes			Testes unitários e de sistema do tracker realizados pelo acadêmico	Testes unitários e de sistema realizados pelo acadêmico e orientador

Quadro 2 - Iterações definidas

A seguir estão descritas as etapas apresentadas no Quadro 2 definidas para o desenvolvimento desse trabalho e as principais atividades realizadas em cada etapa:

a) **Levantamento do referencial teórico:** O referencial teórico abrangeu o conceito geral de métricas de software, qualidade e técnicas de estimativa de esforço. A intenção no levantamento do referencial teórico foi compreender a estimativa feita em gerência de projetos e desenvolver uma solução aplicável ao modelo proposto.

b) **Levantamento de requisitos:** Primeiramente foi realizado um estudo sobre o modelo de solução a ser adotado, o qual foi proposto inicialmente pelo Grupo de Estudos e Pesquisas em Tecnologias de Informação e Comunicação (GETIC) da Universidade Tecnológica Federal do Paraná (UTFPR), na subárea de Engenharia de Software. A partir desse modelo e considerando sua utilização no contexto de uma fábrica de software para ser implementada na disciplina de Oficina de Projeto e Desenvolvimento de Software, foi definido o escopo do projeto. A partir da definição do escopo do projeto, foram levantadas as necessidades do aplicativo a ser desenvolvido. Destaca-se que o projeto aqui proposto será integrado a um módulo de Gestão de Projetos já existente, o qual também foi desenvolvido considerando o contexto de uma fábrica de software, conforme citado anteriormente.
c) **Análise e projeto do sistema:** A partir da definição dos requisitos, esses foram classificados, os casos de uso definidos e expandidos, elaborado o modelo conceitual e definidas e mapeadas as entidades envolvidas, bem como as tabelas necessárias.
d) **Implementação:** A partir de uma cópia do sistema, a qual está armazenada em um servidor *cloud* (Dropbox), com a definição das entidades e relacionamentos do sistema, foram criados os mapeamentos de cada entidade utilizando *JPA Hibernate*. Ainda foram desenvolvidas as funcionalidades apresentadas nos casos de uso deste documento, utilizando as estruturas de organização predefinidas do projeto para codificação de regras de negócios em classes *Bean.* Todo o sistema foi desenvolvido utilizando a IDE *NetBeans* e todos os *layouts* seguiram o padrão provido pela biblioteca *PrimeFaces*, sem a necessidade de utilização de estilos por meio de *Cascading Style Sheet* (CSS).
e) **Testes:** Foram realizados testes unitários do sistema ao desenvolvimento da cada cadastro, criando-se cadastros simulados para um projeto no sistema. Nos cadastros de fases, grupos de fatores e fatores definidores de prazo foram realizados os cadastros dos valores padrões definidos pelo GETIC. Para os testes de integração de sistema, cada módulo foi desenvolvido em paralelo à estrutura do sistema já definida,

acompanhando a integração e desempenho do sistema de estimativas com o sistema para gestão de fábricas de software.

4 RESULTADO

Esse capítulo tem como objetivo apresentar o modelo proposto, os resultados das etapas de levantamento de requisitos e análise e projeto do sistema, bem como o sistema desenvolvido. Inicialmente será apresentado o escopo definido para o sistema. O escopo define as funcionalidades que o sistema deverá conter, bem como suas características principais. Posteriormente é apresentada a análise e projeto para o sistema a ser implementado, contendo a especificação dos requisitos funcionais e não funcionais identificados assim como a modelagem a partir de diagramas da UML. Por fim, é apresentado o sistema implementado visando atender os objetivos propostos para esse trabalho.

4.1 MODELO PROPOSTO

A seguir é apresentado o modelo proposto para a implementação de um sistema para o controle de histórico e geração de estimativa de esforço em projetos de software. Esse modelo foi proposto pelo GETIC.

Foram realizados estudos acerca das diversas técnicas de estimativa de esforço em projetos de software, sendo, a partir desse estudo, selecionada a técnica de pontos de função como base para a definição do modelo. A escolha ocorreu pelo maior suporte teórico encontrado para a mesma, pelas opiniões favoráveis encontradas em publicações e pela abrangência fornecida em termos do sistema considerado. Além disso, entendeu-se que para uma empresa de pequeno porte pode ser mais difícil estimar, por exemplo, as linhas de código necessárias e que métricas baseadas em caso de uso requerem um estágio mais avançado do projeto para poderem ser implementadas.

Com base no referencial teórico sobre pontos de função e na experiência dos membros do grupo de estudos do GETIC, verificou-se como necessário uma elaboração própria para os grupos de funções. Foram tomadas como base as duas

categorias que fundamentam a métrica de pontos de função, que são dados e transações. Contudo, a classificação associada aos pontos de função com arquivos lógicos internos e de interface externa, consultas, entradas e saídas externas foi revista, visando facilitar a categorização pelo usuário e abranger as funcionalidades do sistema.

Com base na literatura, destacando-se os agrupamentos definidos para o cálculo de pontos de função (fases do processo, grupo de fatores, fatores definidores de prazo), conforme o Quadro 3.

Verificou-se que outros aspectos também influenciavam na definição de prazo. Autores como Roetzheim (2000) exemplificam esses fatores. Assim, foram definidos fatores ou requisitos modificadores da estimativa. Pela metodologia proposta, a estimativa é obtida com base nos requisitos e esses outros fatores servem para ajustá-la.

4.1.1 Variáveis validadas por especialistas

Nessa atividade a lista de variáveis definidas foi verificada por profissionais da área de análise, projeto e desenvolvimento de software. Essa verificação teve como objetivo validar as entradas do ponto de vista do especialista no assunto. Foram consultados profissionais que realizam estimativas de projeto de software e que atuam na área de projeto e desenvolvimento de software de várias empresas.

O resultado dessa verificação auxiliou a complementar os grupos definidos. Não houve grandes mudanças, os grupos principais permaneceram, havendo alterações apenas nos subgrupos ou fatores considerados na contagem.

4.1.2 Fatores definidores de prazo

A partir das observações dos especialistas e das discussões, com base no referencial teórico e conhecimento da equipe, os grupos de funcionalidades foram detalhados visando ampliar a sua especificação e facilitar o uso.

A fundamentação teórica para definir os fatores para a contagem de tempo a partir de pontos de função baseou-se no modelo intermediário COCOMO e em APF de Albrecht. De acordo com APF, as funções são divididas em manipulação de dados (arquivos lógicos internos e arquivos de interface externa) e transações (entrada externa, saída externa e consulta externa). A APF é utilizada como base para definir as funcionalidades do sistema das quais é obtido o prazo estimado para implementar o sistema. Contudo, não são usados somente e exatamente os mesmos agrupamentos sugeridos pela APF.

Os atributos que definirão as entradas do sistema como os diretamente vinculados ao produto, são os requisitos funcionais e não funcionais e que determinam os atributos que efetivamente definem o prazo de projeto de software. Se o sistema fosse desenvolvido em condições ideais (recursos, conhecimento e experiência da equipe, requisitos definidos de forma a não haver necessidade de alterações durante o desenvolvimento do projeto, entre outros) o prazo utilizado seria o estimado a partir dos requisitos. Esses requisitos são denominados neste trabalho como os atributos definidores do prazo. Isso porque eles definem o prazo necessário para implementar o sistema.

Esses atributos incluem as funcionalidades do sistema (requisitos funcionais) e requisitos de qualidade (requisitos não funcionais) e estão apresentados no Quadro 3. Eles estão categorizados em grupos que definem categorias de requisitos de um sistema de software. Esses grupos são compostos por fatores que são considerados, ou seja, pontuados na contagem. A coluna Quantidade indica a quantidade de elementos que compõem uma unidade de fator. Tempo é a quantidade de tempo prevista como necessário para implementar a respectiva quantidade de requisitos.

Ressalta-se que um mesmo requisito do sistema pode ser considerado (contado) em mais de um grupo (fator), desde que sejam consideradas funcionalidades distintas para cada item contado, isto é, sem sobreposição de contagem.

Para que a contagem possa gerar um somatório homogêneo é necessário definir a unidade de tempo de entrada, por exemplo, minutos ou horas. Essa unidade deve ser utilizada para indicar os tempos de todos os itens pontuados.

Fase do processo (Tipo)	Grupos de fatores (Subtipo)	Fatores definidores de prazo (Subsubtipo)	Quantidade de atributos	Esforço Padrão (horas)	Quantidade	Esforço por atividade (horas)
Requisitos	Identificação, classificação/organização e modelagem	1. Simples (inclusão, exclusão, alteração, consultas/relatórios)				
		2. Complexos (regras de negócio)				
Análise e projeto	Expansão de casos de uso, modelagem estática e dinâmica e casos de teste	1. Expansão de casos de uso				
		2. Modelagem de interação				
		4. Modelagem de classes				
		5. Modelagem de banco de dados				
		6. Elaboração de casos de teste				
Implementação	a. Manutenção de dados	1. Simples (inclusão, exclusão, alteração, consulta simples em banco de dados)	. até 5 campos			
			. 6 a 15 campos			
			. + 15 campos			
		2. Complexa (com validação, referências cruzadas, campos calculados, buscas inclusive com filtros)	. até 5 campos			
			. 6 a 15 campos			
			. + 15 campos			
	b. Geração de relatório/consulta	1. Simples (listagem de cadastros, validação de dados)	. 1 tabela			
		2. Complexo (campos calculados, joins, union, subselect, filtros, agrupamentos, ordenação, referência cruzada, gráficos, *login* com criptografia)	. 1 tabela			
			. 2 a 5 tabelas			
			. + 5 tabelas			
		3. Com geração de arquivos externos para exportação de dados (ex. arquivos de remessa)	. 1 tabela			
			. 2 a 5 tabelas			
			. + 5 tabelas			
	c. Interação com	1. Acesso a sistemas	. 1 tabela			

	periféricos	externos (validação em banco de dados de outro sistema, interface/protocolo de comunicação)	. 2 a 5 tabelas			
			. + 5 tabelas			
		2. Interação com dispositivos como leitores de biometria, código de barras e etc. Obtenção de dados de dispositivos (sensores)	. 1 tabela			
			. 2 a 5 tabelas			
			. + 5 tabelas			
		3. Envio de comandos para periféricos (sensores, atuadores)	. 1 tabela			
			. 2 a 5 tabelas			
			. + 5 tabelas			
	d. Processamento (transações)	1. Cálculos com ou sem consulta a banco de dados envolvendo regras do negócio (ex. cálculo de impostos, fórmulas específicas em geral). Conversão/formatação de dados.	. 1 cálculo			
			. 2 a 5 cálculos			
			. + 5 cálculos			
		2. Cálculos para atender a exigências externas. (ex.legislação).	. 1 cálculo			
			. 2 a 5 cálculos			
			. + 5 cálculos			
		3. Processamento interno complexo, lógico e matemático extensivo.	. 1 transação			
			. 2 a 5 transações			
			. + 5 transações			
		4. Requisitos não funcionais: restrições de desempenho, restrições de memória, portabilidade, confiabilidade das operaçoes realizadas, integração com sistemas existentes níveis de acesso ao sistema, entre outros.	. 1 requisito			
			. 2 a 4 requistios			
			. mais de 5 a 8 requisitos			
			. mais que 8 requisitos			
		5. Bancos de dados distintos com peculiaridades no SQL para cada banco.	. 1 tabela			
			. 2 a 5 tabelas			
			. + 5 tabelas			
Testes	Execução dos casos de testes	Número de cenários				
				Esforço total para o projeto		

Quadro 3 - Funcionalidades do sistema (agrupamento de requisitos)

Para o cálculo da estimativa é feito o somatório dos prazos obtidos em cada subgrupo (por exemplo: Implementar uma manutenção de dados simples, com até 5 campos: 1 hora):

TempoNãoAjustado = ∑dos tempos dos fatores definidores de prazo

O objetivo é que a metodologia proposta possa ser utilizada na fase de requisitos, então, para que a mesma pudesse se tornar mais efetiva considerou-se necessário sugerir um modelo para o artefato de requisitos. Esse modelo visa que os requisitos do sistema sejam expressos de forma a abranger e conter as informações necessárias para as entradas dos fatores de contagem de prazo. O Quadro 4 apresenta esse modelo.

Identificação	Nome	Descrição	Implementar	Prioridade
RF001	*Nome do requisito funcional*	Explicação do requisito para complementar o seu nome	( )	
RNF001	*Nome do requisito não funcional*	Explicação do requisito para complementar o seu nome	( )	

Quadro 4 - Modelo do artefato de requisitos

No Quadro 4, a coluna implementar define se o requisito será implementado ou é um requisito proveniente de reuso; prioridade pode ser utilizada para a composição das iterações ou mesmo a ordem de implementação dos requisitos. No campo descrição podem ser acrescentadas explicações sobre cálculos necessários, requisitos de interface, validações e outros que possam auxiliar na melhor categorização dos requisitos.

4.1.3 Tempos padrão definidos por especialistas

A listagem apresentada no Quadro 3 foi encaminhada para diversos especialistas em estimativas e análise, projeto e desenvolvimento de sistemas do grupo GETIC. As respostas foram bastante distintas e verificou-se que a linguagem de programação utilizada influencia no prazo estimado. Isto porque há linguagens que permitem um desenvolvimento mais ágil que outras, mas isso também depende da forma de uso dos recursos da linguagem, da habilidade da equipe, dentre outros fatores. Inicialmente havia sido definido que a linguagem de programação seria um fator modificador de prazo. No sentido que o prazo seria estimado e posteriormente

seria considerada a linguagem utilizada. Contudo, a partir das respostas e considerações dos especialistas verificou-se que:

a) não seria efetivo definir tempos padrão a serem utilizados para estimar software implementados em linguagens distintas;

b) a linguagem a ser utilizada poderia ser escolhida como primeira entrada para o sistema, ou seja, a partir da linguagem seriam selecionados os tempos padrão utilizados pelo sistema;

c) o usuário poderia indicar os tempos padrão para que o sistema os utilizasse no cálculo da estimativa. Assim, seriam considerados aspectos como o conhecimento (habilidades) da equipe, o histórico de projetos da empresa e outros.

Desta forma, decidiu-se que os tempos poderiam ser indicados como sugestão ao usuário. Esses tempos poderiam ser definidos a partir de uma base histórica e considerando categorias de linguagens de programação ou outros fatores considerados pertinentes para a empresa, equipe ou projeto. A estimativa estaria baseada na escolha dos fatores e indicação do prazo necessário. Como fatores modificadores estariam o conhecimento e habilidade da equipe em utilizar a linguagem.

4.1.4 Fatores modificadores de prazo

Os fatores constantes no Quadro 3 foram definidos tendo como base os requisitos funcionais e não funcionais do sistema. O cálculo da estimativa de tempo tem como base unicamente esses requisitos. Contudo, há diversos fatores que podem influenciar no tempo efetivamente despendido para implementar um sistema. Assim, foram definidos outros fatores, denominados secundários, que influenciam nos fatores que representam requisitos funcionais e não funcionais. Dentre esses fatores estão, por exemplo, a experiência da equipe em análise e desenvolvimento e o conhecimento que a equipe possui das tecnologias que serão utilizadas. Alguns fatores secundários podem incrementar esse prazo e outros diminuí-lo. Assim, os atributos que definirão as entradas para a contagem de tempo foram agrupados em:

a) Atributos definidores de prazo – são os requisitos funcionais e não funcionais do sistema e por si só definiriam o prazo se as condições de

desenvolvimento fossem ideais (necessárias e suficientes). Esses atributos se referem ao produto em si e estão diretamente vinculados ao sistema a ser implementado.

b) Atributos modificadores de prazo – auxiliam a reduzir ou aumentar o prazo obtido com base nos atributos definidores de prazo. Pontuados no reuso de componentes e artefatos de software e na aquisição de partes já prontas eles reduzem o prazo obtido a partir dos atributos definidores. Aspectos como dificuldades da equipe no domínio do negócio e das tecnologias podem incrementar o prazo calculado a partir dos requisitos do sistema. Esses atributos definem fatores de ajuste do prazo obtido com a contagem a partir dos atributos definidores de prazo.

c) Atributos que auxiliam a definir a precisão da estimativa – o quanto a estimativa pode estar correta, baseada na precisão dos requisitos que serviram de entrada para o sistema.

O Quadro 5 apresenta os fatores que podem alterar o prazo obtido com os fatores definidores de prazo que são baseados nos requisitos do sistema. Esses fatores auxiliam a ajustar o prazo obtido em decorrência de fatores que não são requisitos do sistema, ou seja, o contexto de realização das atividades envolvidas no desenvolvimento do sistema. O contexto se refere às condições em que o sistema é implementado que podem influenciar na implementação, mas não estão diretamente relacionados ao sistema que será produzido e que é objeto da estimativa.

Grupo	**Número (subgrupo)**	**Descrição**	**Ocorrência**	**% de Influência**
Relacionados ao produto (o sistema a ser desenvolvido)	1	Componentes prontos (da própria empresa ou adquiridos de terceiros). Tem-se como base que todo o código será implementado, que não existem componentes para uso. Se existir reduz tempo estimado para desenvolver o projeto. O tempo para implementar o sistema é reduzido à proporção que componentes existentes são utilizados. Exemplos de componentes: cadastros, relatórios, validações de dados.	O tempo para implementar o sistema é reduzido à proporção que componentes existentes são utilizados.	
	2	Confiabilidade das operações realizadas pelo sistema. Considera-se como base de análise o grau de	O tempo para implementar o sistema aumenta se houver operações que exijam confiabilidade ou risco	

		confiabilidade padrão para os sistemas desenvolvidos pela empresa. Se for superior a isso aumenta o prazo de desenvolvimento. O quanto crítico são as operações realizadas pelo sistema. É a criticidade, o risco das operações realizadas pelo sistema.	envolvido.	
	3	Integração com sistemas existentes. Considera-se como parâmetro de base que não há integração.	O tempo para implementar o sistema aumenta se há necessidade de integração entre sistemas.	
Relacionados à equipe	4	Conhecimento e experiência da equipe em análise e projeto. Tem-se como base que o projeto será desenvolvido por uma equipe com conhecimento e experiência necessários para um desenvolvimento ideal.	Se a equipe não possui conhecimento e experiência em análise e projeto, o tempo aumenta.	
	5	Conhecimento e experiência da equipe nas tecnologias adotadas para desenvolvimento.	Se a equipe não possui conhecimento e experiência nas tecnologias adotadas, o tempo aumenta.	
	6	Mudanças que possam causar impacto na equipe.	Se previstas mudanças que possam causar impacto na equipe o tempo aumento.	
Relacionados ao domínio do negócio pela equipe	7	O conhecimento da equipe no processo de negócio sendo implementado. Tem-se como base que o domínio do negócio que o sistema implementará é completo. Se o sistema não é plenamente conhecido aumenta tempo ao projeto.	Se a equipe não possui conhecimento no domínio de negócio do sistema, o tempo aumenta.	
Relacionados ao uso de ferramentas, métodos e modelos	8	Uso de processos, incluindo qualidade, e métodos. Tem-se como base o uso e existência de processos bem definidos. Se não for dessa forma aumenta tempo ao projeto.	Se não há processos, procedimentos bem definidos e documentados o tempo aumenta.	
	9	Uso de artefatos (modelos para documentar os produtos gerados no ciclo de vida, como requisitos, análise, projeto, testes e a existência de padronização de codificação, de interface). Tem-se como base o uso e existência desses modelos de maneira ótima. Se não for dessa forma aumenta tempo ao projeto.	Se não são utilizados modelos de artefatos para documentação, o tempo aumenta.	

Quadro 5 - Fatores modificadores de prazo

O percentual relativo aos fatores modificadores de prazo seria obtido a partir da equação a seguir:

$$\sum(F2+F3+F4+F5+F6+F7+F8+F9)-F1$$

4.1.5 Tempo ajustado

A partir do resultado do esforço não ajustado e os valores percentuais inseridos na tabela de fatores modificadores de prazo, será obtido um valor ajustado e adequado à realidade da empresa. Desse modo consegue-se estimar com mais precisão, considerando aspectos ambientais da organização.

Sendo assim, para obter o esforço ajustado estimado para o projeto deve-se aplicar uma regra de três entre o esforço não ajustado e o percentual obtido dos fatores modificadores.

Abaixo é apresentada a fórmula baseada no cálculo de tempo ajustado do método de APF e utilizada no sistema para retornar o valor ajustado em horas do projeto.

$$TempoAjustado = TempoNãoAjustado * 100 / \sum FatoresModificadores$$

4.2 ESCOPO DO SISTEMA

O sistema proposto deverá ser capaz de registrar a duração de atividades relacionadas ao desenvolvimento de um projeto, contabilizando e armazenando o esforço efetivo para cada atividade, o qual deve ser armazenado no histórico de atividades de cada projeto. Cada atividade deverá ser classificada em relação ao seu status, prioridade e a definição de prazo relacionada que condiz ao nível de complexidade. Cada projeto deverá ser caracterizado por meio de atributos como tipo de projeto, palavras-chave e nível de complexidade, de forma a permitir a reutilização desse histórico de esforço em projetos futuros.

Além disso, o sistema deverá gerar estimativas de esforço para o desenvolvimento do projeto a partir de uma técnica formal. Nesse caso, será utilizada uma adaptação da técnica de contagem de pontos de função, considerando

fatores definidores e modificadores de prazo. Dessa forma, o sistema gerará estimativas que poderão auxiliar o gerente de projetos na definição de prazos de projeto de software. O sistema não irá fazer controle dos recursos humanos ou horas trabalhadas, permitindo estimativas por perfil de usuário (membro de uma equipe de desenvolvimento).

4.3 MODELAGEM DO SISTEMA

A partir dos estudos realizados, do escopo proposto para o sistema e reuniões com professores da área de Engenharia de Software, a seguir são apresentados os requisitos funcionais e não funcionais para o sistema proposto, conforme os Quadros 6 a 13.

F1 Cadastrar tipos de projetos	**Oculto ( )**			
Descrição: O sistema deve permitir o cadastro de tipos de projetos (ex. Desktop, Web, Mobile)				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**

Quadro 6 - RF- Cadastrar tipos de projetos

F2 Cadastro de palavras-chave	**Oculto ( )**			
Descrição: O sistema deve permitir o cadastro de palavras-chave para a classificação de projetos (ex.gestão de estoque, produtos, fatura).				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**

Quadro 7 – RF - Cadastro de palavras-chave

F3 Cadastrar esforço padrão	**Oculto ( )**			
Descrição: O sistema deve permitir o cadastro de esforço padrão para o desenvolvimento de um determinado projeto, o qual deverá ser classificado com relação ao seu tipo, palavras-chave, nível de complexidade e nível de experiência da equipe de desenvolvimento.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**

Quadro 8 - RF - Classificar projetos

F4 Contabilizar esforço efetivo por atividade	**Oculto (X)**			
Descrição: O *tracker* do sistema controlará o tempo efetivo de execução de cada atividade de um projeto, e fazer o controle da quantidade de acessos a uma atividade, bem como a contabilização do tempo total para execução da atividade.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF2.1 Cálculo	O cálculo do tempo será feito internamente sempre que o	Regra de negócio	( )	(X)

	status da atividade for alterado para "concluída", e armazenada no campo duração efetiva da atividade em questão.			
NF2.2 Controle de Acesso	Programadores terão acesso para ativar a contagem do *tracker* e desativar. Apenas gerentes de projeto podem alterar o tempo já contado.	Segurança	( )	(X)
NF2.3 Número de instâncias do *tracker*	Cada atividade deverá ser realizada individualmente, sendo possível haver somente uma instância do *tracker* em um determinado momento.	Interface	( )	(X)

Quadro 9 - RF – Contabilizar esforço efetivo por atividade

F5 Gerar esforço padrão conforme a classificação do projeto	**Oculto ( )**			
Descrição: O sistema deverá permitir a geração de esforço padrão por tipo e subtipo e por número de atributos dependendo da classificação do projeto (tipo, palavras-chave, nível de complexidade, características da equipe)				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**

Quadro 10 - RF - Gerar esforço padrão conforme a classificação do projeto

F6 Gerar estimativa de esforço para um projeto	**Oculto ( )**			
Descrição: O sistema irá gerar uma tabela contendo o número de atividades por fase do projeto, tipo (grupo de fatores), subtipo (fatores modificadores) e por número de atributos (informados pelo gerente de projetos). Esse número deverá ser multiplicado pelo esforço padrão para cada tipo/subtipo de atividade (conforme histórico).				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF5.2 Carregar esforço padrão	Deve ser permitida a recuperação de um esforço padrão previamente cadastrado, o qual deverá ser filtrado com base em parâmetros como: tipo de projeto, palavras-chave, complexidade, nível de experiência da equipe de desenvolvimento.	Interface	( )	( )
NF5.1 Restrição de edição	O campo "esforço padrão" somente poderá ser editado pelo gerente de projetos.	Segurança	( )	(X)

Quadro 11 - RF - Gerar estimativa de esforço para um projeto

F7 Gerar totalizador para os fatores modificadores de prazo	**Oculto ( )**			
Descrição: O sistema irá cadastrar os fatores modificadores de um projeto e respectivos percentuais, os quais irão interferir na estimativa do projeto e o sistema deverá gerar o total de fatores modificadores de prazo.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**

Quadro 12 - RF - Gerar totalizador para os fatores modificadores de prazo

F8 Gerar esforço ajustado para um projeto	**Oculto ( )**
Descrição: O sistema deverá gerar o esforço ajustado para um projeto a partir do esforço não-ajustado e os percentuais dos fatores modificadores de prazo.	
Requisitos Não-Funcionais	
Nome **Restrição** **Categoria** **Desejável** **Permanente**	

Quadro 13 - RF - Gerar esforço ajustado para um projeto

A Figura 2 apresenta o modelo de casos de uso para o sistema proposto, o qual apresenta as funcionalidades completas, ou seja, aquelas que além de possuir algum tipo de interação com um ator, podem ser realizadas de forma isolada.

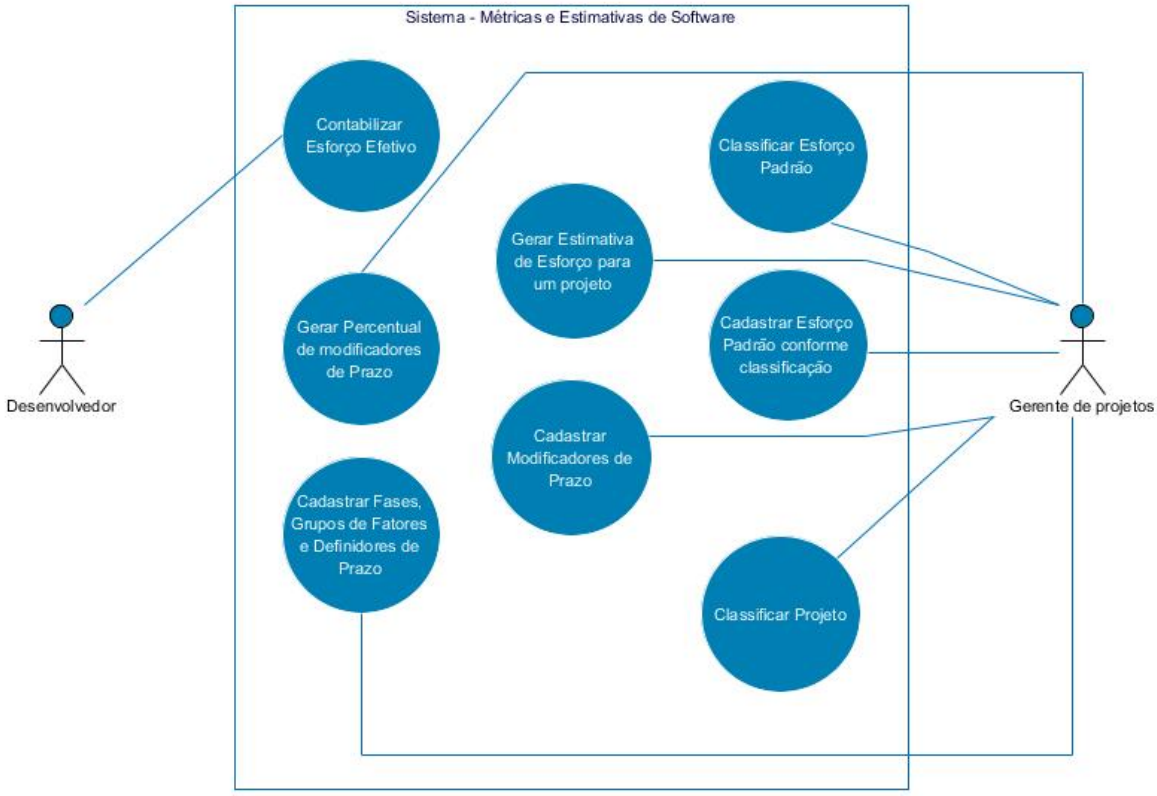

Figura 2 - Casos de Uso

A expansão de casos de uso tem como finalidade especificar os processos de um sistema de forma a facilitar a descoberta dos objetos envolvidos em cada processo como também servir de base para testes do sistema.

O quadro 14 apresenta a expansão do caso de uso Registrar tempo de execução de atividade, enquanto o quadro 15 apresenta a expansão do caso de uso Gerar estimativa de esforço para um projeto.

Caso de Uso: Registrar tempo de execução de atividade
Atores: Desenvolvedores, Gerente de projetos,
Precondições: O desenvolvedor deverá estar conectado no sistema (mesmo off-line). O desenvolvedor deverá ter a tarefa atribuída a si. O desenvolvedor não poderá ter outra tarefa em execução.
Pós-condições: O *log* do tempo deverá ser salvo no banco de dados, na tabela referente às atividades.
Sequência típica de eventos (Fluxo Principal): 1. [IN] Este caso de uso inicia quando o desenvolvedor acessa uma atividade a ele atribuída. 2. [IN] O desenvolvedor ativa o contador de tempo referente à atividade e marca o seu início. 3. O desenvolvedor realiza a atividade. 4. [OUT] O sistema registra o tempo de realização da atividade 5. [IN] O desenvolvedor ativa o contador de tempo e marca o fim da atividade 6. [OUT] O sistema contabiliza o tempo de realização da atividade 7. [IN] O desenvolvedor relata a execução do que foi feito. Variante 6a: Atividade totalmente concluída 7a.1.: [IN] O desenvolvedor muda o *status* da atividade para “finalizada”. 8. Variante 6b1: Atividade parcialmente concluída. 9. [IN] O desenvolvedor salva o registro do tempo.
Tratamento de Exceções e Variantes: **Exceção 1a:** Já existe uma tarefa em andamento 1.1 [OUT] O sistema retorna uma mensagem de erro. 1.2 [OUT] O caso de uso é finalizado. **Exceção 2a:** O desenvolvedor esquece o *tracker* ligado/desligado 2.1 [IN] O desenvolvedor deverá informar o gerente de projetos e o gerente de projetos irá alterar o tempo real 2.3 Vai para o passo 2. **Exceção 3a:** O desenvolvedor fecha o contador/ atualiza a página 3.1 [OUT] O sistema armazenará o último minuto contabilizado em memória cache do navegador. 3.2 [OUT] O sistema deve atualizar todos os campos com os últimos dados salvos.

Quadro 14 - Caso de uso Registrar tempo de execução de atividade

Caso de Uso: Gerar estimativa de esforço para um projeto
Atores: Gerente de projetos
Precondições: O projeto deverá ter tarefas cadastradas. O gerente de projetos deverá cadastrar estimativas de tempo para cada tarefa.
Pós-condições: O sistema irá gerar os valores estimados na tabela de estimativas de acordo com o projeto.
Sequência típica de eventos (Fluxo Principal): 1. [IN] Este caso de uso inicia quando o gerente de projetos seleciona um projeto para o qual deseja gerar uma estimativa 2. [IN] O gerente de projetos classifica o projeto por tipo de projeto, palavra-chave, complexidade e nível de experiência da equipe. 3. [IN] O gerente de projetos informa o número de atividades por fase do projeto, tipo, subtipo e número de atributos. 4. [IN] O gerente de projetos define o impacto dos fatores modificadores de prazo. 5. [OUT] O sistema calculará o esforço estimado para cada atividade.

6. [OUT] O sistema retornará o esforço total para o projeto.
Tratamento de Exceções e Variantes:

Quadro 15 - Gerar estimativa de esforço para um projeto

A Figura 3 mostra o Diagrama Entidade e Relacionamento (DER) que apresenta as tabelas e seus relacionamentos identificados, os quais representam o banco de dados da aplicação. De acordo com a legenda, é demonstrado a evolução do diagrama por parte dos outros desenvolvedores, sendo que as cores se tratam das tabelas desse sistema desenvolvido e as outras cores as tabelas já desenvolvidas.

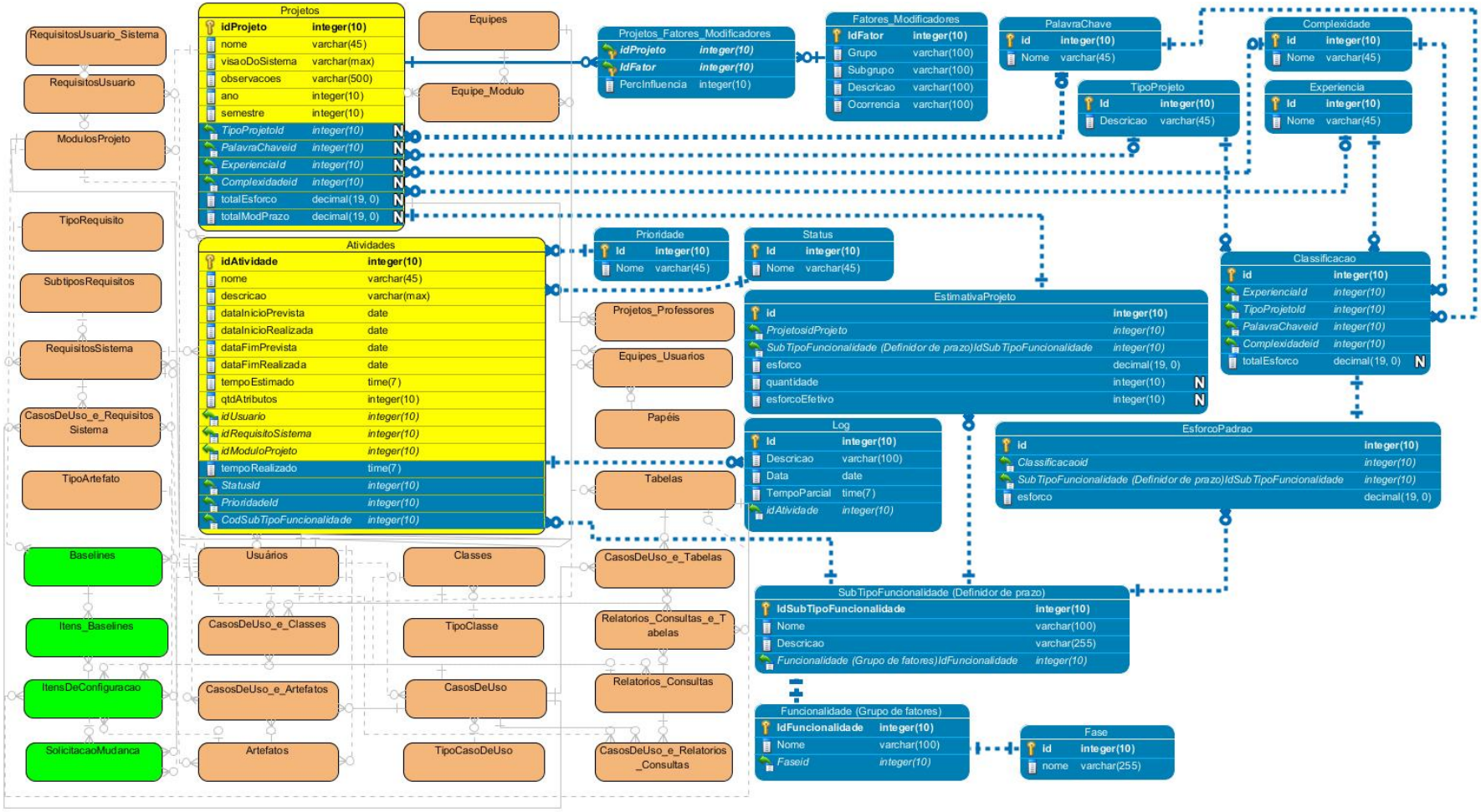

Figura 3 - Diagrama entidade e relacionamento

A tabela de *Log* (Quadro 16) armazenará os tempos parciais de execução de cada atividade iniciada por um usuário até que esta tenha o *status* “finalizada”.

Log			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
Descricao	100, obrigatório	Varchar	
Data	Datetime, obrigatório	Datetime	
TempoParcial	Time, obrigatório	Time	Armazena o tempo parcial de cada execução de uma atividade
idAtividade	Inteiro, obrigatório	Chave estrangeira	

Quadro 16 – Tabela Log

A tabela *Log* terá um código, descrição do que foi feito naquela atividade, data e hora da execução, o tempo parcial executado e o ID da atividade, que já possui o código do usuário ao qual a atividade está atribuída. No momento em que o *status* da atividade for alterado para “concluída”, o sistema contabilizará os *log*s daquela atividade.

A tabela de esforço padrão (Quadro 17) armazenará os valores totais em cada tipo de funcionalidade para uma determinada classificação de projeto no sistema, assim o gerente de projetos poderá selecionar estimativas armazenadas dentro de uma classificação adequada para atribuir a um projeto atual. A tabela armazenará um código de chave primária, uma chave de uma determinada classificação de projeto, um código de subtipo de funcionalidade (fator definidor de prazo) e ainda o esforço propriamente dito armazenado em horas.

EsforcoPadrao			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
idEsforco	Inteiro, obrigatório	Autoincremento primary key	
idClassificacao	Inteiro, obrigatório	Chave estrangeira	
idSubTipoFuncionalidade	Inteiro, obrigatório	Chave estrangeira	
esforco	decimal, obrigatório	decimal	

Quadro 17 – Tabela EsforcoPadrao

Os Quadros 18 a 27 representam valores constantes, porém com o intuito de viabilizar a futura implementação de seus respectivos cadastros no sistema.

Fatores_Modificadores			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
IdFator	Inteiro, obrigatório	Autoincremento	
Grupo	100, obrigatório	Varchar	
Subgrupo	100, obrigatório	Varchar	
Descricao	100, obrigatório	Varchar	
Ocorrencia	100, obrigatório	Varchar	

Quadro 18 - Fatores Modificadores

Projetos_Fatores_Modificadores			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
IdProjeto	Inteiro, obrigatório	Chave estrangeira	
IdFator	Inteiro, obrigatório	Chave estrangeira	
PercInfluencia	Inteiro, obrigatório	Integer	

Quadro 19 - Projetos_Fatores Modificadores

PalavraChave			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
Nome	45, obrigatório	Varchar	

Quadro 20 – Tabela Palava-Chave

Complexidade			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
Nome	45, obrigatório	Varchar	

Quadro 21 – Tabela Complexidade

TipoProjeto			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
Descricao	45, obrigatório	Varchar	

Quadro 22 – Tabela TipoProjeto

Fase			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
Nome	255, obrigatório	Varchar	

Quadro 23 – Tabela Fase

Funcionalidade			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
Nome	255, obrigatório	Varchar	
idFase	Inteiro, obrigatório	Chave estrangeira	

Quadro 24 – Tabela TipoFuncionalidade

SubTipoFuncionalidade			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
Nome	100, obrigatório	Varchar	
Descricao	255, obrigatório	Varchar	
idFuncionalidade	Inteiro, obrigatório	Chave estrangeira	

Quadro 25 – Tabela SubTipoFuncionalidade

Status			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
Nome	45, obrigatório	Varchar	

Quadro 26 – Tabela Status

Prioridade			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
Nome	45, obrigatório	Varchar	

Quadro 27 – Tabela Prioridade

Experiencia			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
Nome	45, obrigatório	Varchar	

Quadro 28 – Tabela Experiencia

Classificação			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento	
idExperiencia	inteiro, obrigatório	Chave estrangeira	
idTipoProjeto	Inteiro, obrigatório	Chave estrangeira	
idPalavraChave	Inteiro, obrigatório	Chave estrangeira	
idComplexidade	Inteiro, obrigatório	Chave estrangeira	
totalEsforco	decimal, não obrigatório	decimal	

Quadro 29 – Tabela Classificação

A tabela classificação armazena as características de critério de um projeto no sistema. Ela possui chave estrangeira da tabela experiência, da tabela de tipos de projeto, da tabela de palavra chave e ainda da tabela de complexidade. A chave primária encontrada nessa tabela é utilizada na tabela de esforço padrão. A tabela ainda possui um campo não nulo, no qual será atualizado o total de esforço do padrão correlacionado à respectiva classificação.

EstimativaProjeto			
Atributos	**Tamanho/restrições**	**Tipo**	**Observações**
Id	Inteiro, obrigatório	Autoincremento Primary key	
idProjeto	inteiro, obrigatório	Chave estrangeira	
idSubtipoFuncionalidade	Inteiro, obrigatório	Chave estrangeira	
quantidade	Inteiro, não obrigatório	inteiro	
esforco	decimal, obrigatório	decimal	
esforcoEfetivo	decimal, não obrigatório	decimal	

Quadro 30 – EstimativaProjeto

A tabela de estimativa de projeto, semelhantemente a tabela de esforço padrão, armazenará o esforço de respectivas funcionalidades do sistema. A

diferença desta tabela, é que a mesma possui como critério de classificação, um projeto já existente no sistema. A tabela armazenará um fator definidor de prazo (ex. manutenção de dados) associado a um valor de esforço. A tabela armazenará ainda a quantidade de elemento de acordo com esse fator dentro do sistema e ainda o valor de esforço efetivo realizado para aquele fator.

4.4 APRESENTAÇÃO DO SISTEMA

O trabalho aqui apresentado foi resultado de um estudo gerado pelo GETIC como solução para organização do processo de desenvolvimento de sistemas, abordando todo o processo de gerenciamento de um projeto - criação de um projeto de software, definição e atribuição de equipes e atividades do projeto e gerenciamento de artefatos gerados durante o processo de desenvolvimento do projeto - além de um módulo para controle de atividades de gestão de configuração . Desta forma, o sistema apresentado a seguir visa complementar esse sistema, dispondo de um módulo para a geração de estimativa de esforço de projetos de software a partir do registro de esforço padrão por tipo de atividades em cada fase de um projeto.

A estrutura do trabalho foi implementada utilizando a arquitetura *Model View Control* (MVC), como apresentado a seguir na Figura 4.

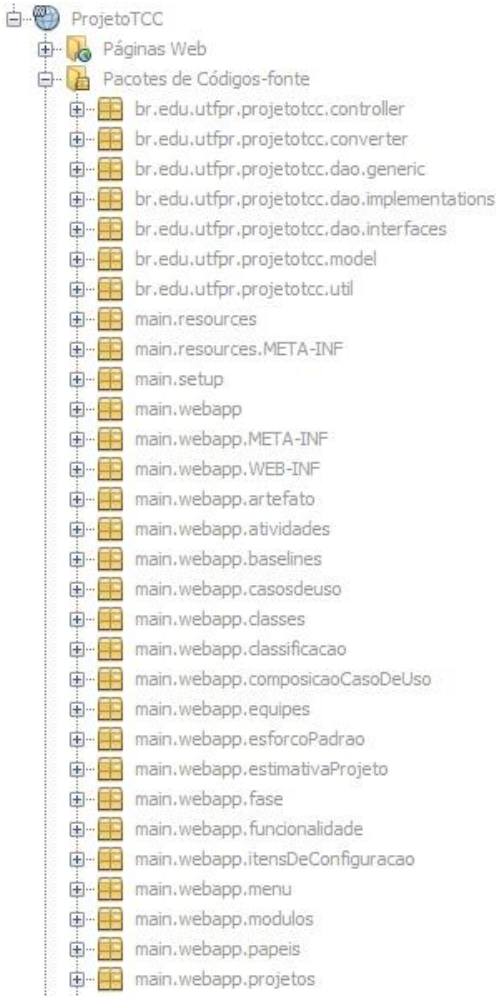

Figura 4 - Estrutura MVC

A partir dessa estrutura foi implementado cada módulo do sistema para as respectivas áreas de gerência, as quais podem ser acessadas por um menu lateral no sistema através de autenticação de usuário.

A Figura 5 mostra como está organizada as estruturas de menus, dando ênfase ao acesso a gerência de projetos e gerência de estimativas, referentes ao assunto do sistema apresentado.

Figura 5 - Menu Lateral

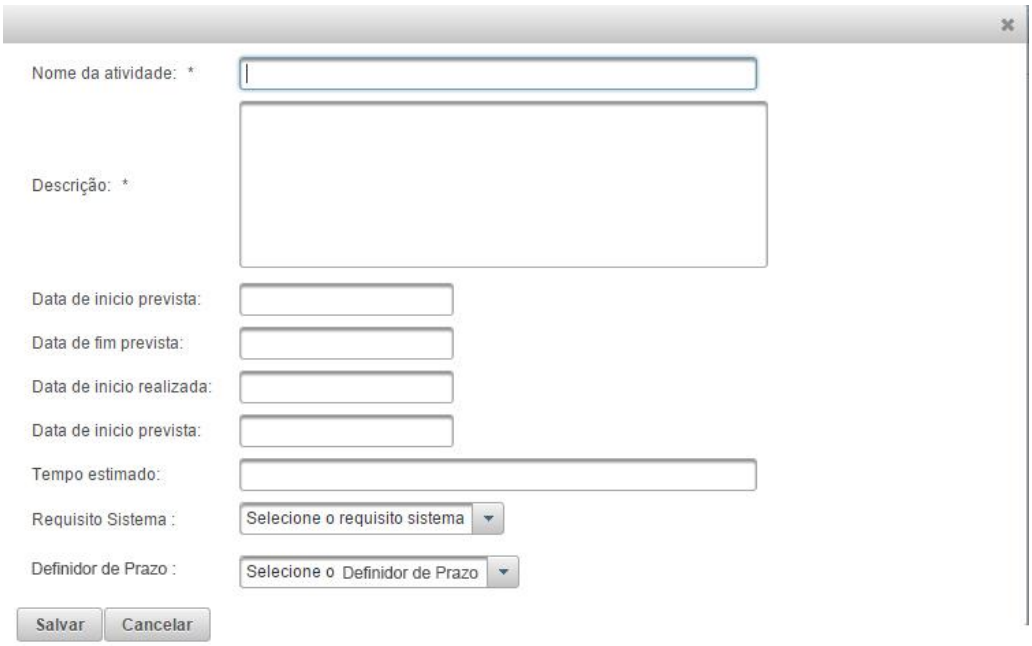

Figura 6 - Cadastro de Atividade

Com isso, é possível visualizar a Figura 6 que apresenta a tela de cadastro de atividade, onde é possível através do módulo de um projeto já cadastrado no sistema cadastrar atividades relacionadas ao desenvolvimento do projeto. Esse cadastro é utilizado para a contabilização do esforço efetivo para o comparativo posterior da estimativa gerada pelo sistema através da parametrização determinada pelos fatores definidores de prazo.

Posteriormente, já acessando o painel de gerência de estimativas, o usuário tem acesso à tela de listagem padrão do sistema. Através desta o usuário pode acessar todos os registros referentes àquele módulo bem como as ações

correlacionadas. A seguir um exemplo dessa listagem (Figura 7), apresentando a listagem de grupo de fatores. Essa tabela é construída com linhas selecionáveis, uma vez selecionadas, disponibilizam as ações de edição, exclusão e visualização dos dados no menu superior. A parte dessas ações específicas, a ação de criação de um novo grupo de fator está sempre disponível.

Figura 7 - Lista de Grupos de Fatores

Na Figura 8 é apresentada a tela de cadastro de fatores definidores de prazo, chamados internamente de subtipo de funcionalidade, onde seus dados são derivados dos cadastros da Fase do processo e Grupos de Fatores, também encontrados no módulo de gestão de estimativa do sistema.

O cadastro solicita o nome do grupo de fator, de inserção obrigatória, descrição desse definidor, a fase da qual ela é utilizada. Ao selecionar uma fase, o combo de grupo de fator é atualizado automaticamente via AJAX com os grupos de fatores referentes àquela fase apenas.

Figura 8 - Cadastro de Fatores Definidores de Prazo

A Figura 9 apresenta a listagem de classificações de projetos e esforço padrão do sistema. A partir dessa classificação é possível associar um padrão de estimativa a um projeto iniciado no sistema.

Figura 9 - Listagem de Classificações

Uma vez criadas combinações de classificações do sistema, é possível cadastrar um padrão de estimativa para a respectiva classificação. A Figura 10 mostra a tela de cadastro de esforço para fator definidor de prazo do padrão. A tela é composta pela especificação de uma classificação através do acesso do menu lateral do formulário, intitulada classificar padrão. Os esforços padrão são declarados no acesso aos definidores de prazo, mostrando uma tabela editável (*editable grid*) listando fatores definidores de prazo referentes à aba superior em que se localizam as fases. Uma vez armazenados os esforços, o sistema armazenará cada esforço relacionado a um fator definidor de prazo e o total da estimativa.

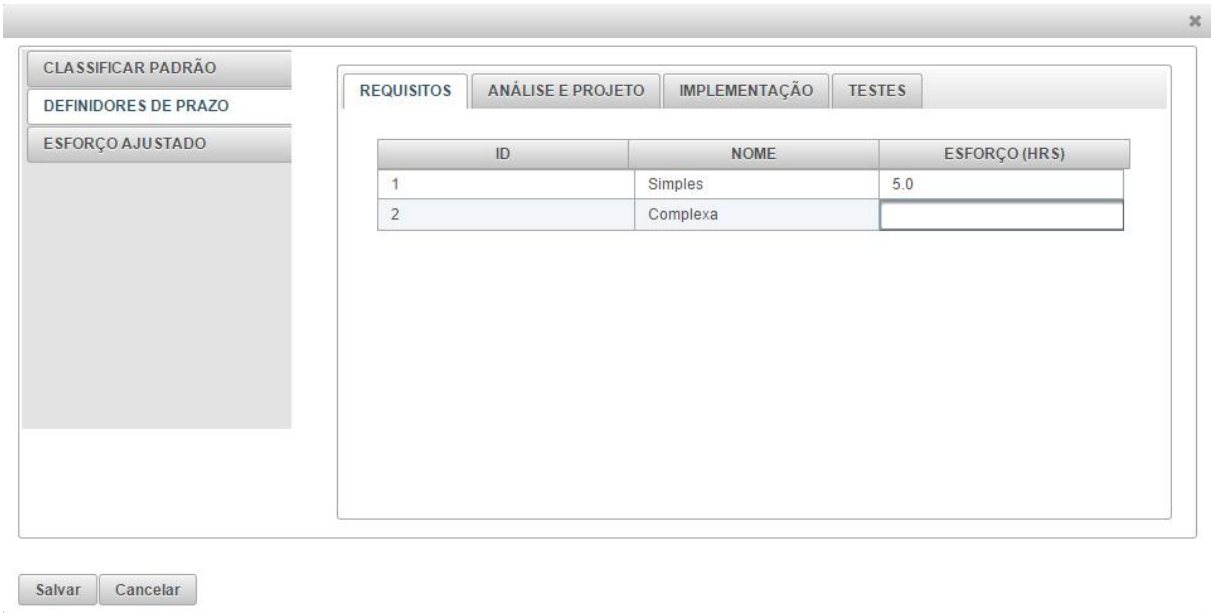

Figura 10 - Cadastro de estimativa padrão

Na Figura 11 está o modelo de apresentação do *tracker* do sistema, componente utilizado para contagem do esforço efetivo realizado em cada atividade de um projeto.

Ao acessar a listagem de atividades no módulo de gestão de projetos, o desenvolvedor poderá ativar o *tracker* e começar a contar o esforço efetivo para a respectiva atividade. O sistema permitirá apenas um *tracker* ativo para cada usuário *online* no sistema, assim como o tempo será associado à atividade atual que está sendo realizada pelo usuário.

Figura 11 – *Tracker*

Ao confirmar a conclusão da atividade, o usuário poderá salvar o esforço determinado para a atividade. A Figura 12 mostra a tela de cadastro, a qual virá com o nome selecionado e o respectivo tempo, sendo necessário o cadastro de uma descrição para o que foi feito na execução da atividade. O tempo salvo será armazenado no esforço efetivo do projeto para a determinada atividade, associado ao fator definidor de prazo.

Figura 12 - Cadastro de Esforço Efetivo

Cada esforço efetivo cadastrado na Figura 12 é automaticamente atualizado na tela de cadastro de estimativa para o projeto, a partir do seu registro em uma tabela de *log* de atividades. A Figura 13 mostra um exemplo da tela resultante da

geração de estimativa para um projeto, que possui o cadastro da classificação do projeto, cadastro de estimativa para fatores definidores de prazo bem como a geração da estimativa de esforço ajustada. A visualização do esforço efetivo só ocorrerá após a realização das atividades..

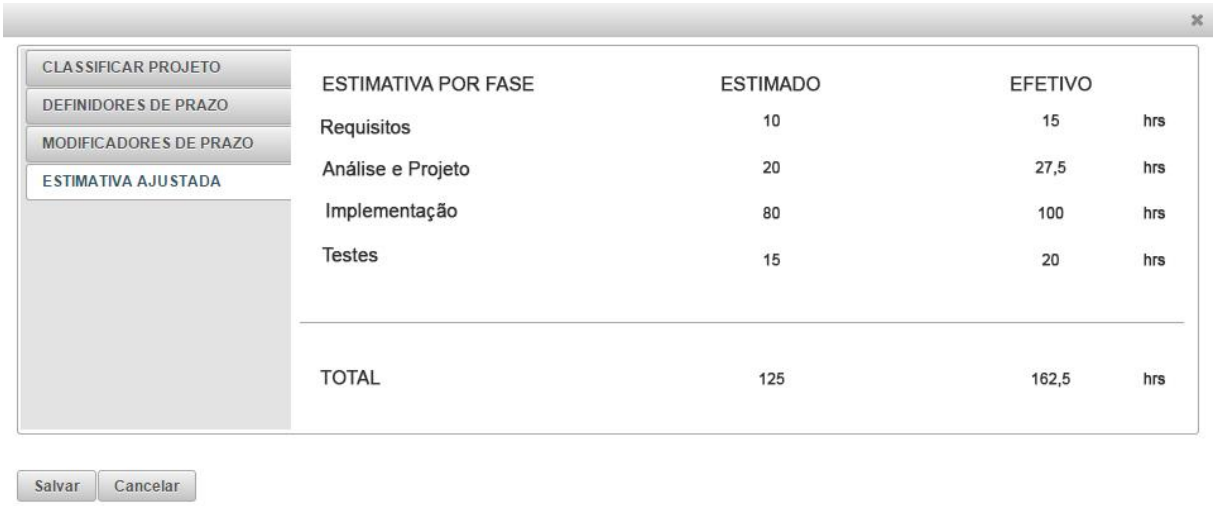

Figura 13 - Cadastro de estimativa para projeto

4.5 IMPLEMENTAÇÃO DO SISTEMA

De acordo com o modelo MVC, as telas acessadas do menu lateral chamam um arquivo XHTML denominado *index*, que faz as chamadas separadamente das operações de Inclusão, edição, leitura e exclusão. A Listagem 1 mostra um exemplo desse arquivo para o cadastro de estimativa padrão do sistema.

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
                xmlns:h="http://xmlns.jcp.org/jsf/html"
                xmlns:ui="http://java.sun.com/jsf/facelets">

               <ui:include src="/esforcoPadrao/list.xhtml"/>
               <ui:include src="/esforcoPadrao/view.xhtml"/>
               <ui:include src="/esforcoPadrao/edit.xhtml"/>
</ui:composition>
```

Listagem 1 - Página agrupadora de ações

O JSF implementa variáveis iterativas acessíveis através dos arquivos XHTML. Através dessa técnica é possível acessar os atributos mapeados de cada

entidade. A implementação de todas as listagens do sistema foram utilizadas dessa forma, bem como utilizando o mapeamento de outras entidades a partir de uma entidade pai.

A Listagem 2 mostra parte do código da listagem de fatores definidores de prazo. A tabela é montada sobre o método *listItems* da entidade *subtipoFuncionalidade*, equivalente a definidores de prazo no sistema. A partir dessa listagem a tabela é montada mostrando o nome da fase do fator, o grupo do fator, atributos nome e descrição.

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
                xmlns:h="http://java.sun.com/jsf/html"
                xmlns:f="http://java.sun.com/jsf/core"
                xmlns:p="http://primefaces.org/ui"
                xmlns:ui="http://java.sun.com/jsf/facelets"
                xmlns:c="http://java.sun.com/jsf/composite/comps">

    <h:form id="frmprincipal" prependId="false">
      <p:dataTable  var="subtipoFuncionalidade"
                    value="#{subtipoFuncionalidadeBean.listItems}"
                    id="tabela"
                    selection="#{subtipoFuncionalidadeBean.selectedItem}"
                    selectionMode="single"
                    rowKey="#{subtipoFuncionalidade.id}"
                    emptyMessage="Nenhum registro encontrado">

          <f:facet name="header">
            <div class="header-crud">
              <p:commandButton id="novoRegistro"
                               value="Novo definidor de prazo"
                               actionListener="#{subtipoFuncionalidadeBean.addItem()}"
                               update=":form:formDetalhes"
                               oncomplete="PF('dlgformDetalhes').show()"/>
              <p:commandButton id="cmdAlterar"
                               value="Alterar"
                               update=":form:formDetalhes"
                               oncomplete="PF('dlgformDetalhes').show()"
                               disabled="#{empty subtipoFuncionalidadeBean.selectedItem}"/>
              <p:commandButton id="cmdExcluir"
                               value="Excluir"
                               actionListener="#{subtipoFuncionalidadeBean.delete}"
                               update="tabela"
                               disabled="#{empty subtipoFuncionalidadeBean.selectedItem}">
                 <p:confirm header="Confirmação" message="Confirma a exclusão?" />
              </p:commandButton>
              <p:commandButton id="view"
                               value="Visualizar"
                               update=":SubtipoFuncionalidadeView"
                               oncomplete="PF('dlgsubtipoFuncionalidadeViewDlg').show()"
                               disabled="#{empty subtipoFuncionalidadeBean.selectedItem}"/>

              <p:commandLink id="atualizar"
                             value="Atualizar"
                             update="tabela"
                             style="float: right"/>
            </div>
          </f:facet>

          <f:facet name="header">Lista de definidores de prazo Cadastradas</f:facet>
```

```
        <p:column headerText="Fase do Processo (Tipo)" width="20%">
            <h:outputText value="#{subtipoFuncionalidade.idFuncionalidade.idFase.getNome()}" />
        </p:column>
        <p:column headerText="Grupos de fatores (Subtipo)" width="20%">
            <h:outputText value="#{subtipoFuncionalidade.idFuncionalidade.getNome()}" />
        </p:column>
        <p:column headerText="Fator definidor de prazo (Subsubtipo)" width="10%">
            <h:outputText value="#{subtipoFuncionalidade.nome}" />
        </p:column>
        <p:column headerText="Descrição" width="40%">
            <h:outputText value="#{subtipoFuncionalidade.descricao}" />
        </p:column>
    </p:dataTable>
  </h:form>
</ui:composition>
```

Listagem 2 - Listagem de fatores definidores de prazo

Como citado anteriormente, cada página destas faz referência a uma entidade mapeada por uma classe à qual é denominada *bean.* Cada uma dessas classes é responsável por trazer os métodos e mapeamentos de dados provenientes da entidade declarada, como ao exemplo mostrado anteriormente.

A vantagem de se utilizar essa arquitetura é a facilidade de separar funcionalidades e regras de negócio do sistema, desse modo a entidade fica responsável apenas pelos dados, enquanto a classe *bean* trabalha conduzindo e gerenciando esses dados para a *view*. Ao próprio exemplo anterior, o método *listItems* é implementado dentro da *bean* de SubtipoFuncionalidadeBean bem como os métodos *Save* e *Delete*.

A seguir, na Listagem, 3 é apresentado um modelo de código utilizado na classe EsforcoPadraoBean, referente ao mapeamento da entidade Esforço Padrão.

```
@Named
@ViewScoped
public class EsforcoPadraoBean extends GenericBean<EsforcoPadrao>{

   private static final long serialVersionUID = 1L;
   List<SubtipoFuncionalidade> listaDefinidores;

   public EsforcoPadraoBean() {
   }

   @Inject
   public EsforcoPadraoBean(EsforcoPadraoDao dao) {
      super(dao);
   }

   @Override
   public void addItem() {
      listaDefinidores();
      super.addItem(EsforcoPadrao.class);
   }

   @Override
   public void save() {
      for (SubtipoFuncionalidade sub : listaDefinidores) {
         sub.getEsforcoPadrao().setSubtipoFuncionalidade(sub);
      }
      super.save();
   }
```

Listagem 3 - *Bean* controladora da entidade EsforcoPadrao

As operações da *Bean*, as quais são mapeadas e implementadas nas classes chamadas *Data Acces Object* (DAO), foram separadas em interfaces e implementação, a fim de garantir o modelo MVC. Essa metodologia apresenta suas qualidades em relação a herança do Java, tornando o sistema altamente escalável, experiência encontrada ao desenvolver esse projeto, o que possibilita maior agilidade no desenvolvimento, pois facilita a criação de classes simples.

```
public interface EsforcoPadraoDao extends GenericDAO<EsforcoPadrao>{
}

package br.edu.utfpr.projetotcc.dao.generic;

import java.util.List;
```

```
public interface GenericDAO<T>{

   void beginTransaction();

   void commitTransaction();

   void save(T object);

   void delete(T object);

   List<T> listAll();

   T findById(Object id);

}
```

Listagem 4 - Interface de Dao de EsforcoPadrao

Um método implementado para retornar os dados é diretamente ligado à entidade mapeada com a *JPA Hibernate*. Desse modo pode-se criar um método na DAO e esta estar associada a uma *Query* específica da entidade. A Listagem 5 mostra a *Query* genérica associada ao método *listAll()*, da classe DAO.

```
@Entity
@Table(name="esforcopadrao")
@NamedQueries({
   @NamedQuery(name = "EsforcoPadrao.findAll", query = "SELECT e FROM EsforcoPadrao e")})
public class EsforcoPadrao implements Serializable {
  ...
}
```

Listagem 5 - Mapeamento e Declaração da entidade EsforcoPadrao

Por fim, cada dado fornecido ao sistema por meio de um cadastro será inserido diretamente à tabela mapeada pela entidade de maneira automática via *JPA Hibernate*, como mostra a Listagem 6, no exemplo da tela de cadastro de esforço padrão.

```
<c:formulario id="form"
                formName="formDetalhes"
                width="960"
                tabelaAtualizar=":frmprincipal:tabela"
                salvar="#{esforcoPadraoBean.save()}"
                cancelar="#{esforcoPadraoBean.cancel}">

              <p:tabView  id="esforcoPadraoTabView"
                      orientation="left" >

                 <p:tab title="Classificar Padrão">

                    <h:panelGrid columns="2" cellpadding="10">
                       <p:outputLabel for="selectClassifica" value="Classificação: " />
                       <p:selectOneMenu id="selectClassifica"
```

```
                converter="classificacaoConverter"
                value="#{esforcoPadraoBean.selectedItem.classificacao}"
                required="true"
                requiredMessage="Selecione um tipo">
            <f:selectItem itemLabel="Selecione classificação"/>
            <f:selectItems value="#{classificacaoBean.listItems}"
                var="classi"
                itemLabel="#{classi.id}"
                itemValue="#{classi}"/>
        </p:selectOneMenu>
    </h:panelGrid>
</p:tab>
```

Listagem 6 - Cadastro de Classificação de Esforço Padrão

5 CONCLUSÃO

O trabalho aqui apresentado atingiu seus objetivos, de implementar uma solução para o controle de métricas de esforço e estimativa de esforço de software a partir das técnicas propostas desenvolvidas especificamente para esse projeto pelo GETIC.

O sistema se propunha parametrizar os atributos de um sistema, a fim de padronizar o registro e estimativa de esforço, determinando, assim, uma métrica para a estimativa de todos os projetos desenvolvidos pela.

A partir de experiências do acadêmico de convivências em dois ambientes de desenvolvimento divergentes em organização dos requisitos e processos de desenvolvimento, uma consistindo de uma organização muito bem implementada, enquanto outra estava no andamento da implementação de um método organizacional. Durante o processo de implementação desse trabalho de conclusão, foi possível concluir que elementos como número de pessoas necessárias para atuar em determinado projeto ou o tempo de conclusão de um projeto com determinadas especificações são influenciados diretamente e de forma relevante pelos fatores de padronização e histórico de desenvolvimento da própria empresa.

A partir dessa padronização de fatores relativos a um projeto, é possível armazenar padrões de esforço no histórico da empresa, propiciando um desenvolvimento sustentável e uma gestão do conhecimento da empresa, tornando a estimativa de esforço e custos cada vez mais ajustados à realidade. Essa capacidade do sistema traz retornos visíveis ao processo de desenvolvimento de projetos em empresas de tecnologia.

As tecnologias utilizadas mostram-se eficazes para solucionar e suprir as necessidades de implementação. Foram utilizados a linguagem Java bem como os frameworks *JSF* e ainda *Primefaces* que consiste em uma biblioteca visual de componentes compatíveis com *JSF*.

Uma vez determinados esses parâmetros e junto dos padrões de estimativas, as empresas poderão agora utilizar o sistema de controle de esforço, para uma melhor organização dos processos, criação de projetos classificados e estimados.

Ao final do trabalho tem-se uma ferramenta que poderá auxiliar nas dificuldades encontradas na realidade de empresas e profissionais da área, além de

destinar-se a ser utilizado na disciplina de Oficina de Projeto e Desenvolvimento de Software.

Este trabalho ainda requer algumas complementações futuras a fim de atender de forma mais completa e facilitada às necessidades do processo de estimativas pelos gerentes de projeto. Dentre essas complementações estão fazer uma classificação mais detalhada de projetos cadastrados, utilização do esforço efetivo registrado para cada atividade de um projeto com relação à sua classificação para a geração de estimativas de futuros projetos. É importante ressaltar também a necessidade de aperfeiçoamento do sistema em aspectos como organização e diagramação do *layout* dos cadastros. Ainda relevante citar a necessidade das validações com o *tracker* para as situações adversas identificadas durante esse trabalho e que podem ocorrer durante a sua execução, como, por exemplo, de quedas do servidor e a ainda possível implementação de módulos compatíveis com outras plataformas para uma possível solução dessas eventualidades, como por exemplo, um módulo *desktop*.

REFERÊNCIAS

ABRAHAO, S.; INSFRAN, E. **A Metamodeling approach to estimate software size from requirements specifications.** Software Engineering and Advanced Applications, 2008, p. 465-475. Washington (DC): IEEE Computer Society

ABREU, Thamine Chaves; MOTA, Leonardo da Silva; ARAÚJO, Marco Antônio Pereira. Métricas de software: somo utilizá-las no gerenciamento de projetos de software. **Engenharia de Software Magazine - Métricas de Software**, p. 50-55. 12 mai. 2014.

ALBRECHT, A. J., & Gaffney, J. E. (1983). **Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation.** IEEE Transactions on Software Engineering , VI (9), pp. 639-648.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 12207**: **tecnologia de informação - processos de ciclo de vida de software.** Rio de Janeiro: 1998.

BARTIÉ, Alexandre. **Garantia da qualidade de software: adquirindo maturidade organizacional.** Rio de Janeiro, Elservier, 2002.

BEZERRA, E. **Princípio de análise e projeto de sistemas com UML.** 3ª Tiragem. Rio de Janeiro: Campus, 2002.

BROOKS, F. P. **No Silver Bullet: Essence and accidents of software engineering.** Computer, v. 20, n. 4, p. 10–19. April 1987.

CMMI Product Development Team. CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.1 Staged Representation (CMU/SEI-2002-TR-012, ESC-TR-2002-012). Software Engineering Institute, Carnegie Mellon University, 2002

DEMARCO, TOM. **Controle de projetos de software.** 9. ed. Rio de Janeiro: Editora Campus, 1991.

GALORATH, Daniel D.; EVANS, Michael W. **Software sizing estimation, and risk management.** Auerbach Publications – New York 2006.

HAUGEN, N. C. An empirical study of using planning poker for user story estimation. **Proceedings of the Conference on AGILE 2006 (pp. 23-34).** Washington (DC): IEEE Computer Society, 2006.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 9000:2000, Sistemas de gestão da qualidade – Fundamentos e vocabulário**, 2000.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 15504 International Standard Organization. ISO/IEC TR 15504 - Software process**. Montreal: ISO/IEC JTC1 SC7, 1999.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION AND THE INTERNATIONAL ELECTROTECHNICAL COMISSION. **ISO/IEC 12207:2008 Systems and software engineering — Software life cycle processes**, Geneve: ISO, 2008.

KALINOWSKI, M.; SPINOLA, R. O. .Introdução à Inspeção de Software. Engenharia de Software Magazine, p. 68- 74, 01 mar. 2008.

LEITE, Jair C., **Engenharia de software**. 2007. Disponível em: <http://www.dimap.ufrn.br/~jair/ES/slides/EstimativasMetricas.pdf>. Acesso em: 5 maio de 2014.

MACORATTI, **Análise de pontos por função - O processo de contagem.** Disponível em: <http://www.macoratti.net/apf_pcta.htm>. Acesso em: 25 mai. 2014.

MACORATTI, José Carlos. **Estimativas de tamanho de software e APF.** Disponível em: <http://www.macoratti.net/net_est1.htm>. Acesso em: 20 mai. 2014.

MARQUES, Daniela. **Métricas de software. Techno*log*y Leadership Council Brazil.** Disponível em: <https://www.ibm.com/developerworks/community/b*log*s/tlcbr/entry/metricas_de_software?lang=en>. Acesso: 20 mai. 2014.

PARK, Robert E.; GOETHERT, Wolfhart B.; FLORAC, William A. **Goal-driven software measurement** - A Guidebook, 1996.

ROETZHEIM, William. **Project cost adjustments.** SD Magazine, November/2000. Disponível em: <http://carlosproal.com/itpm/files/cocomo/SDArticle2.pdf>. Acesso: set. 2014.

SOFTEX. **MPS.BR - Melhoria de Processo do Software Brasileiro. Guia de Implementação – Parte 2: Fundamentação para Implementação do Nível F do MR-MPS**, Brasília: Sociedade Softex, 2011.

SOFTEX. **Guia de Implementação Parte 9: Implementação do MR-MPS em Organizações do tipo Fábrica de Software.** Novembro 2011.

SOMMERVILLE, Ian. **Engenharia de software,** 8. ed. São Paulo: Pearson, 2007.

SQS Portugal. **ISO/IEC 15504 - Avaliação de Maturidade e Capacidade de Processos.** Disponível em: <http://www.sqs.pt/portfolio/servicos/services_15504.htm>. Acesso em: 23 mai.

2014. WEB SOCKET. **What is Web Socket?.** Disponível em: <http://www.websocket.org/>. Acesso em: 8 set. 2014.