INTERFACES CIENTÍFICAS

EXATAS E TECNOLÓGICAS

ISSN IMPRESSO - 2359-4934
ISSN ELETRÔNICO - 2359-4942
DOI - 10.17564/2359-4934.2015v1n3p43-54

A ENGENHARIA DE REQUISITOS: UM CASO DE IMPLEMENTAÇÃO DE UM SISTEMA PARA ENGENHARIA DE REQUISITOS

Pablo Marques Menezes[1]
Thiago de Almeida Correia[2]
Thiciane Couto[3]
Larissa Santana[4]

RESUMO

A implementação de softwares que atenda as expectativas dos usuários, em relação às funcionalidades desejadas, custos e prazos ainda é um processo complexo. O alto índice de falhas na construção de sistemas representa perda de qualidade e elevação de custos aos desenvolvedores, comprometendo os resultados perante o cliente-usuário. Como ferramenta para evitar esse cenário, esta investigação exploratória, utilizando a metodologia de apresentação de produto, trata sobre a implementação de um software de engenharia de requisitos, denominado SER, demonstrando as fases de construção e testes diante dos pressupostos teóricos que embasam as etapas a serem cumpridas na identificação e acompanhamento de aspectos essenciais à adequada elaboração de projetos de softwares. Confirmou-se a aplicabilidade do produto elaborado, o qual representa uma solução adequada ao atendimento das necessidades do projetista/analista de ferramentas de apoio para a etapa de projeto e levantamento de requisitos, pois cumpre os aspectos imprescindíveis recomendados pelo International Requirements Engineering Board (IREB).

PALAVRAS-CHAVES

Desenvolvimento de software. Engenharia de requisitos. Projetos de software.

ABSTRACT

The implementation of software that meets users' expectations in relation to the desired functionality, cost and time is still a complex process. The high rate of failures in building systems is loss of quality and increased costs for developers, committing the results to the client-user. As a tool to prevent this scenario, this exploratory research, using the product presentation methodology deals with the implementation of a requirements engineering software, called SER, showing the phases of construction and testing before the theoretical assumptions underlying the steps be complied with in identifying and monitoring key aspects to the appropriate elaboration of software projects. Confirmed the applicability of the developed product, which is an appropriate solution to meet the needs of the designer / analyst support tools for the design stage and requirements gathering, as meets the essential aspects recommended by the International Requirements Engineering Board (IREB).

KEYWORDS

Software Development. Engineering Requirements. Software Projects.

RESUMEN

La aplicación de software que atienda a las expectativas de los usuarios, en relación con las funcionalidades deseadas, costos y tiempo, todavía es un proceso complejo. La gran tasa de fallos en la construcción de sistemas representa la pérdida de calidad y aumento de los costos para los desarrolladores, que comprometen los resultados frente al cliente usuario. Como una herramienta para evitar este escenario, esta investigación exploratoria, utilizando la metodología de presentación de producto, trata sobre la implementación de un software de ingeniería de requisitos, llamado SER, que muestra las fases de construcción y pruebas delante de los presupuestos teóricos que defienden las etapas que deben cumplirse en la identificación y el seguimiento de los aspectos clave para la elaboración apropiada de proyectos de software. Se ha confirmado la aplicabilidad del producto desarrollado, el cual es una solución adecuada para satisfacer las necesidades del diseñador/analista de las herramientas de soporte para la etapa de proyecto y levantamiento de requisitos, ya que cumple con los aspectos esenciales recomendados por el **International Requirements Engineering Board** (IREB).

PALABRAS CLAVE

Desarrollo de software. Ingeniería de requisitos. Proyectos de software.

1 INTRODUÇÃO

A indústria de software, desde sua origem, foi marcada por dificuldade devido à abstração do produto. O seu desenvolvimento envolve abstração e complexidade em fornecer um parâmetro de qualidade antecipada. A implementação de software que atenda as expectativas dos usuários, em relação às funcionalidades desejadas, custos e prazos ainda é um processo complexo. 18% dos projetos de softwares são cancelados, 43% sofrem modificações de requisitos e apenas 39% dos projetos terminam com sucesso, ainda 69% dos projetos possuem problemas de requisitos (IREB, s.d).

Pressman (2011) afirma que entender os problemas de requisitos de software é uma das tarefas mais difíceis que um engenheiro de software enfrenta. Desde o início do processo de desenvolvimento de software um dos maiores empecilhos nos projetos é o levantamento e entendimento dos requisitos do software, pois o sucesso do software depende do entendimento correto dos problemas (CHENG, 2002; ENGHOLM JR, 2013). Sendo uma das principais fases do processo de qualidade do software o levantamento de requisitos deve ser realizado com base em técnicas tradicionais

de levantamento de requisitos tendo como objetivo o entendimento e retratação dos objetivos do software (CHENG, 2002; ENGHOLM JR, 2013; BARTIÉ, 2002).

Desta forma, o levantamento ideal das necessidades e operacionalidades de um software passa a ser um elo entre o projeto e a construção do software. Assim, surge o conceito de Engenharia de Requisitos (ER) que representa um conjunto de ferramentas e procedimentos, que visam identificar as reais necessidades de um projeto, estabelecendo as características que o software deve apresentar para a aceitação por parte do cliente. Se as necessidades do futuro usuário forem mal interpretadas, o produto será mal desenvolvido, resultando em um software de baixa qualidade, implicando em mudanças no decorrer do projeto. Isto pode transformar o produto em um mosaico irregular. Assim, o levantamento de requisitos impõe processos que permitam gerenciá-los para ajustes no sistema a ser construído. Sobre isso Sommerville (2007, p. 129) menciona:

> Você precisa manter o acompanhamento dos requisitos individuais e manter as ligações entre os requisitos dependentes, de modo que seja possível avaliar o impacto das mudanças de requisitos. Você precisa estabelecer um processo formal de fazer propostas de mudanças e ligá-las aos requisitos de sistema. O processo de gerenciamento de requisitos deve se iniciar assim que uma versão inicial do documento de requisitos esteja disponível, mas você deve iniciar o planejamento das mudanças de requisitos durante o processo de elicitação de requisitos.

Um projeto com problemas de requisitos gera um produto com má qualidade, e quanto antes forem detectadas eventuais falhas, menor será o custo de sua correção. Assim, uma falha encontrada na fase de requisitos tem um custo menor do que uma identificada na fase de manutenção. Um exemplo do que tal situação representa monetariamente, é descrito por Pressman (2011, p. 367), em seu estudo sobre a empresa Cigital:

> De acordo com os dados médios do setor, o custo para descobrir e corrigir defeitos durante a fase de codificação é de US$ 977 por defeito. Portanto, o custo total para corrigir os 200 defeitos “críticos” durante essa fase (200 x US$ 977) é de aproximadamente US$ 195.400.Os dados médios do setor mostram que o custo para descobrir e corrigir defeitos durante a fase de testes é de US$ 7.136 por defeito. Nesse caso, supondo que a fase de testes do sistema tenha revelado aproximadamente 50 defeitos críticos (ou apenas 25% daqueles encontrados pela Cigital na fase de codificação), o custo para descobrir e corrigir esses defeitos (50 x US$ 7.136) teria sido de aproximadamente US$ 356.800.

O exemplo de Pressman (2011) é corroborado por Sommerville (2007) ao mencionar que o maior problema enfrentado durante o desenvolvimento de um software é a ER. A evolução do custo de correção de um software é progressiva diante do avanço das fases em que vai se completando, conforme exposto na Figura 1, a seguir.

Figura 1 – Custo relativo para correção de erros e defeitos

Fonte: Pressman, 2011

O desenvolvimento de software com qualidade necessita de compreensão dos problemas, o que ocorre na fase de requisitos. Assim, infere-se o número de problemas por meio da adoção de um processo formal de ER, o qual permita uma real compreensão das necessidades para o sistema, com a delimitação correta dos processos a serem implementados e a validação das necessidades do cliente (PRESSMAN, 2011). A ER é um processo que visa fornecer informações adequa-

das sobre as necessidades de negócio e do cliente. Segundo o **International Requirements Engineering Board** – IREB (2012) é possível defini-las em cinco fases ordenadas e sequenciais, sendo elas:

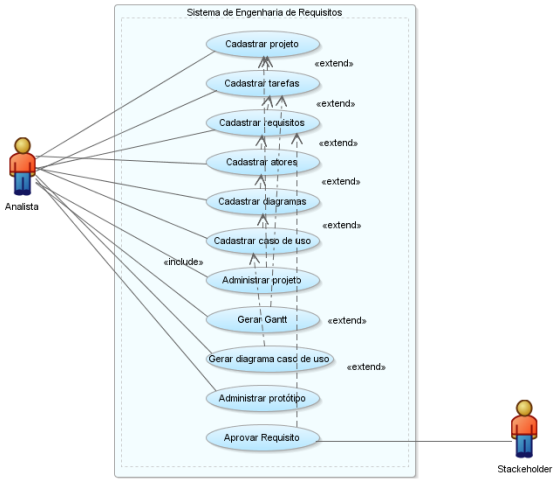
a) Delimitação do sistema;
b) Elicitação de requisitos;
c) Documentação de requisitos;
d) Validação e negociação de requisitos;
e) Gerência de requisitos.

Assim, adotando-se os referenciais de Pressman (2011) e Sommerville (2007), associados às fases definidas pelo IREB (2012), têm-se como objetivo desta pesquisa a implementação de um software, denominado Sistema de Engenharia de Requisitos (SER) para a redução ou eliminação de ruídos na comunicação entre cliente, analista e programador, por meio do cumprimento das etapas de requisitos para a construção do produto solicitado pelo usuário.

2 METODOLOGIA

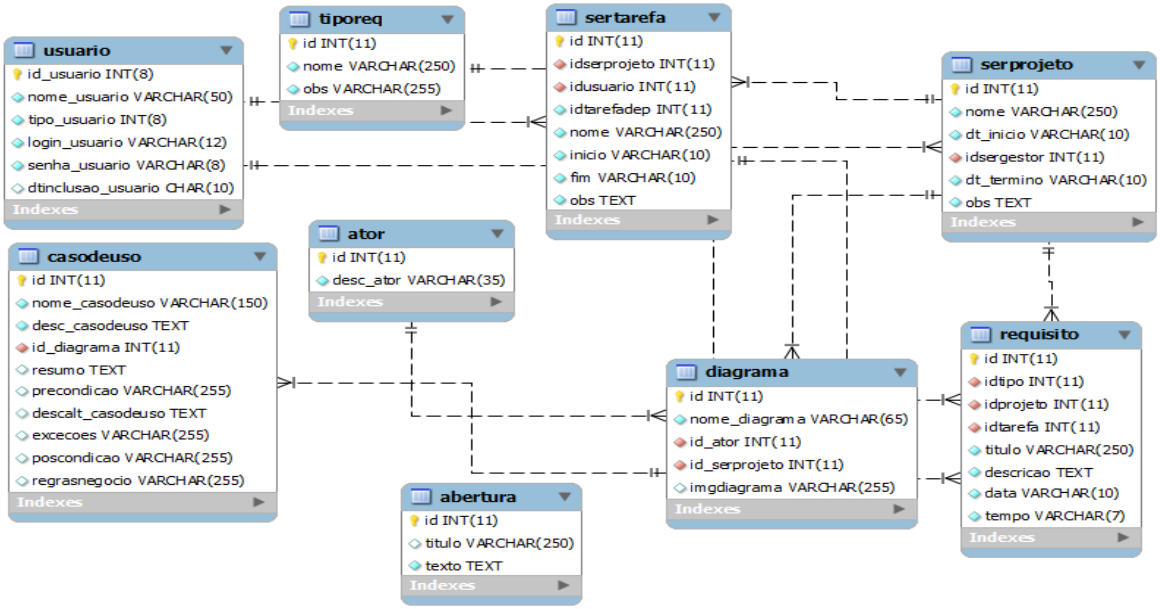
A pesquisa utilizará a metodologia de apresentação de um produto, que conforme define Wazlawick (2009) é apropriada a investigações exploratórias. Para o desenvolvimento do sistema foram definidas três etapas, iniciando-se pela análise de funcionalidades, com base nas fases do IREB, gerando o diagrama de caso de uso, exposto na Figura 2, a seguir.

Figura 2 – Diagrama de caso de uso com funcionalidades do projeto SER



Fonte: Elaborado pelos autores

Figura 3 – Diagrama de banco de dados



Fonte: Elaborado pelos autores

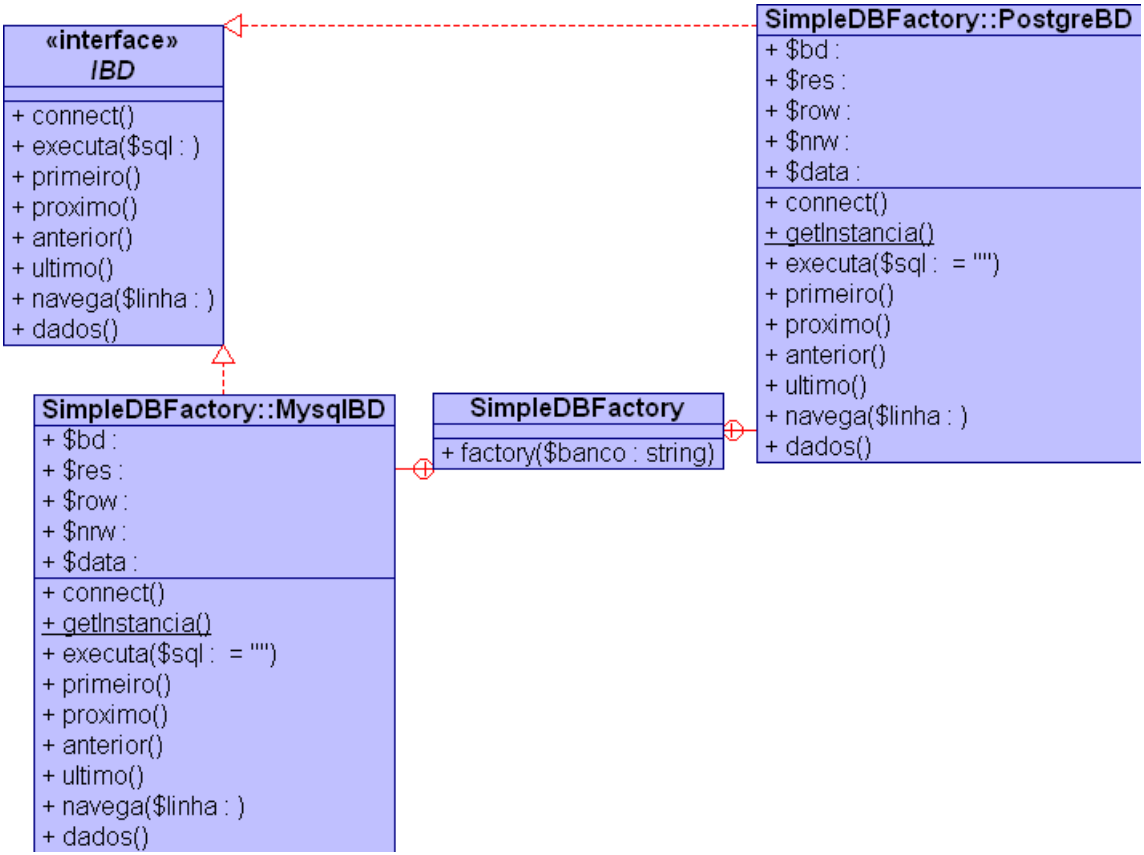
Um sistema de informação de requisitos integrado, como qualquer outro, consiste em dois componentes: hardware e software, contendo diversos módulos ou subsistemas, dentro dos seguintes grupos funcionais: elicitação, validação, gerenciamento de requisitos, modelagem da aplicação, conforme a especificação do caso de uso, exposto na Figura 2. Para atender os grupos funcionais levantados, seguiu-se para a análise de dados, sendo desenvolvido o diagrama de banco de dados (Figura 3), contemplando as necessidades do projeto.

Com base nas funcionalidades identificadas, foram analisados os requisitos não funcionais do projeto:

- Multiplataforma (capacidade de funcionar em diversas plataformas como Windows, Linux e Mac OS X);
- Multiusuário (Capacidade de suportar diversos usuários simultaneamente);
- Colaborativo (Capacidade de dois ou mais usuários trabalhar simultaneamente);
- On-line (capacidade de funcionar em ambiente de internet).
- Facilidade de adaptação;
- Facilidade de transporte do sistema;
- Instalação simplificada.

Com base nos requisitos não funcionais de facilidade de adaptação, transporte do sistema, instalação simplificada e de troca de base de dados, buscando a integração da aplicação com banco de dados, foi criada uma biblioteca com o objetivo de facilitar a migração da base de dados, utilizando-se o padrão fabrica, que define uma interface para a criação de objetos, deixando para as classes especialistas a sua manipulação Melo (2007) e, possibilitando agregar suporte a novos bancos de dados de forma simplificada, com isso foi criada uma biblioteca responsável por fabricar as conexões com o banco de dados, empregando **SimpleFactory**, conforme exposto na Figura 4, a seguir.

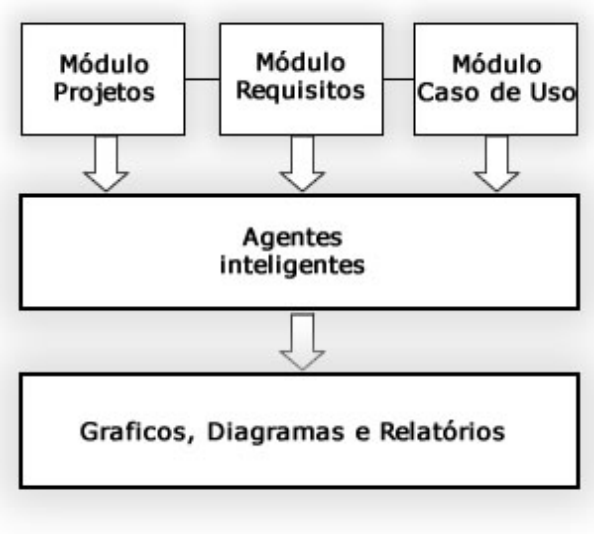
Figura 4 – Diagrama de classe biblioteca de persistência



Fonte: Elaborado pelos autores

A biblioteca possibilitou a adaptação do sistema para múltiplos bancos de dados, inicialmente, foi criado suporte para MySQL e PostgreeSQL, porém, devido ao formato fabrica, incluir novos bancos é um processo simples. A preocupação com a arquitetura, resultou em um sistema multi-módulos com divisão de responsabilidade e com agentes inteligentes para automatizar tarefas, este modelo foi utilizado, pois possibilita mudanças nas camadas sem impactos no sistema como um todo, minimizando os custos de manutenção no sistema Silveira (2012), conforme exposto na Figura 5. O sistema inicia-se pela tela de **login**, partindo para as liberações de módulos conforme o perfil do usuário.

Figura 5 – Projeto SER



Fonte: Elaborado pelos autores

Os módulos possuem agentes que auxiliam na automação de tarefas como a geração de gráficos. Segundo Russell e Norvig (2004) um agente é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores. Todos os módulos dependem do agente de projeto, ou seja, para iniciar um levantamento é necessário criar um projeto-base para os requisitos. O agente de projetos e responsável por gerar automaticamente o gráfico de Gantt, com base nas informações do cadastro de projeto. O agente de projeto e tarefas utiliza uma **Application Programming Interface** (API) que atua em conjunto com o **JQuery** como ferramenta atuadora para criar o gráfico.

O próximo agente é um dos mais importantes, o agente de caso de uso. Após ter preenchido os casos de uso no sistema, esse agente utiliza sensores que permite, de forma automática, utilizando uma API como atuadora, gerar o diagrama de caso de uso. Esse agente atualmente é dependente de Internet para a geração de diagramas. Os agentes proporcionam autonomia ao sistema, tornando-o mais inteligível, prático e ágil. Além disso, o desenvolvimento proporcionou boa performance ao sistema, resultando em sensores e atuadores ágeis e pouco dependentes, tornando o ambiente simples e prático.

Além da busca pela praticidade, o sistema encaminha-se para atender às necessidades da ER e dos projetos de software, possuindo em cada módulo o cadastro, a validação, a manutenção e o acompanhamento por meio de relatórios e informações. O sistema, por funcionar em ambiente Web, permite que sejam feitos validações e acompanhamentos por parte do cliente de forma dinâmica e remota, criando-se, assim, um trabalho colaborativo.

3 ESTRATÉGIA DE DESENVOLVIMENTO

O sistema foi desenvolvido adotando-se a seguinte estratégia: para a parte visual foi selecionada uma biblioteca que oferecesse facilidade ao desenvolvimento, praticidade ao ambiente e visual agradável ao usuário. Para isso, desenvolvimento do sistema utilizou-se a biblioteca **JQUery** na área visual, buscando facilitar a utilização e aprendizagem do ambiente. A biblioteca **JQuery** tem como características:

- livre, podendo ser utilizada e modificada;
- suporta as especificações do padrão do CSS3;
- multinavegador, facilitando a portabilidade;
- suporte a mobile.

Assim, a biblioteca além de melhorar a usabilidade do sistema, auxiliaria no processo de requisitos não funcionais como o suporte à multiplataforma.

Para a conectividade com o banco de dados, conforme apresentado na Figura 4, foi implementada uma biblioteca de persistência, em linguagem de **script** php. Tal biblioteca possibilita a troca de banco de dados de forma simples e prática, alterando-se apenas um arquivo de configuração. Os agentes foram implementados em linguagem de **script** php. Eles são os responsáveis pela percepção do sistema, ou seja, a

observação sobre o que está sendo feito, verificando se o usuário cadastrou uma nova tarefa, ou um novo caso de uso, permitindo ordens para os atuadores que executam funções específicas.

Os atuadores utilizam um conjunto de programação **script** php com XML e **javascript** em conjunto com API´s específicas para cada tarefa, gerando-se, assim, os resultados necessários como gráficos e diagramas. Além disso, os atuadores são responsáveis pelo armazenamento das informações em banco de dados ou em disco. Para que os agentes funcionem corretamente, foram construídos algoritmos de busca de informação que possibilitassem ao sensor verificar e validar o que o usuário realizou. Os algoritmos fazem uso extensivo de linguagem SQL para que a consulta seja eficiente e funcional.

Para completar o sistema foi utilizado a API **GuiDesigner** na construção do módulo de prototipagem. Essa biblioteca é prática e simples, facilitando a criação de telas de protótipos. Todo ambiente conta com validações e recursos que buscam a facilidade de desenvolvimento e manutenção. O sistema foi documentado, utilizando-se o **PHPDoc** e testado amplamente com o **SimpleTest**, em conjunto com o **JMeter**.

4 RESULTADOS

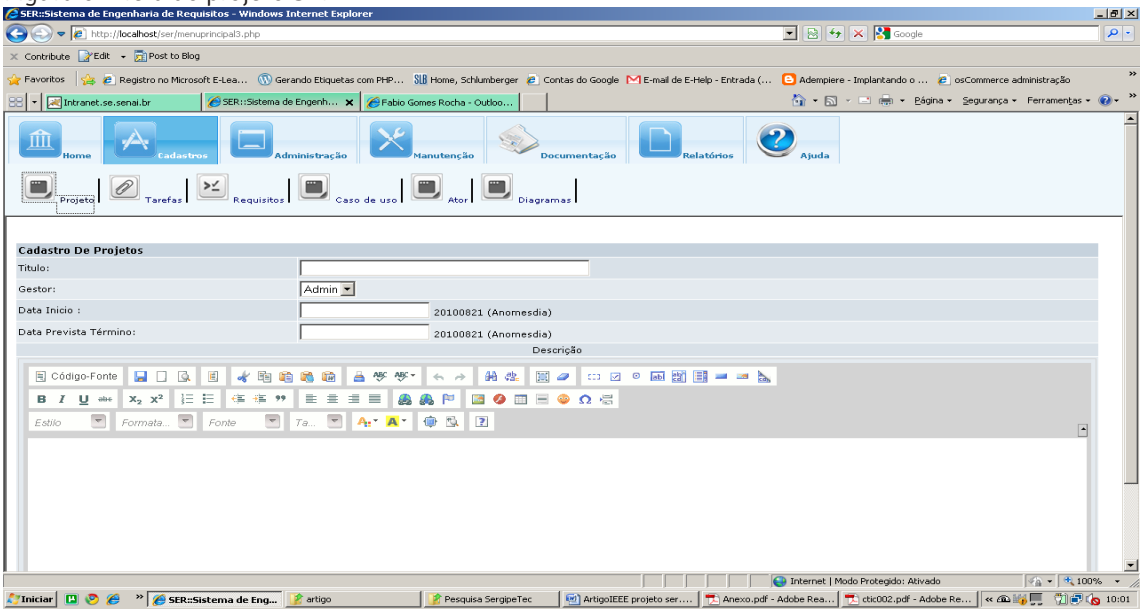
O resultado do desenvolvimento foi o Projeto SER, um software livre que permite que seus usuários utilizem seus recursos para elicitar, validar e gerenciar requisitos, bem como o gerenciamento de projetos, disponível sob licença GPL 2.0 no endereço http://sourceforge.net/projects/sersistemadeeng.

Devido a biblioteca de banco, a configuração de conexão é realizada com apenas um arquivo do tipo **ini**, facilitando a instalação e uso da aplicação, assim, além do SER, como resultado obteve-se uma biblioteca para a integração do sistema com diversos bancos de dados, intitulada **GLibSer**. O banco de dados padrão do sistema é o MySQL, porém a biblioteca permite a substituição por meio de um simples processo de edição de arquivo.

No tocante à linguagem de programação, foi adotada a PHP por ter se consolidado como uma das principais linguagens Web da atualidade e por atender as demandas do projeto.

O Projeto SER é prioritariamente baseado em tecnologia livre, daí as escolhas de bibliotecas livres, linguagens e banco de dados livre para melhor resultado. A implantação de testes do sistema foi feito em servidor Web Apache, mas pode ser utilizado qualquer servidor compatível com a Linguagem PHP. Com base no conhecimento tácito da realidade de analistas de sistemas, o sistema foi dividido em partes e os menus possuem facilidade de acesso, conforme exposto na Figura 6, a seguir:

Figura 6 – Tela do projeto SER



Fonte: Elaborado pelos autores

Todos os cadastros que demandam longos textos possuem um editor para facilitar essa tarefa, podendo-se utilizar recursos como negrito, itálico para marcar informações importantes durante o processo de elicitação, por exemplo, além disso, o sistema permite a comunicação instantânea entre Engenheiro de Requisitos, analistas, equipe de desenvolvimento e **stackholder**, possibilitando interação em tempo real e colaboração entre os membros do projeto, tal interação agrega conhecimento a equipe, sendo fruto de um processo ativo de relação entre os indivíduos e o meio, nesse caso o sistema de requisitos, pois possibilita a intra e inter-relação pessoal entre os membros. A interação ocorre por meio das ações entre os indivíduos mediadas pelo ambiente, o que gera influências simultâneas.

Outros resultados obtidos pelo projeto são os agentes inteligentes que, com base nos cadastros efetuados pelo analista, geram diagramas e gráficos automaticamente. O agente do sistema é onisciente, sabendo antecipadamente o resultado de suas ações com base nos processos executados. Atualmente, o agente que gera os diagramas UML depende da Internet para manipular as informações e gerar a imagem. Após isso, a imagem é copiada pelo sistema para o disco e cadastrada no banco de dados. Por fim, o sistema pode ser instalado diretamente em uma memória flash, tornando-o um ambiente prático para consultores, ou pode rodar na Internet, sendo acessado de qualquer lugar, já que possui uma camada de segurança que exige o **login** no sistema pelo analista ou o cliente.

5 CONSIDERAÇÕES FINAIS

Observando-se o alto índice de insucesso de projetos que não cumprem as etapas necessárias à identificação de fatores, aspectos e envolvimentos dos recursos, a ER emerge não só como uma ferramenta técnica, mas também como analítica no sentido de proporcionar a identificação de lacunas e

ajustes que, eventualmente, venham a comprometer os objetivos do projeto.

Além disso, o sistema atende as necessidades do IREB, premissa inicial do projeto. Outro ponto importante é a colaboração, item identificado como importante nos requisitos não funcionais, que foi contemplado e testado, sendo totalmente funcional, para próximas versões e testes, deseja-se que o sistema possua um sistema automático de controle de versão documental, para rastrear mudanças de requisitos.

Outro fator importante é a simplicidade de uso, a ferramenta foi construída sob a premissa de facilidade de uso, empregando componentes visuais que facilitassem a utilização, nos testes iniciais realizados pela equipe de desenvolvimento, o sistema apresentou simplicidade para novos usuários, porém, adaptações em relação aos gráficos foram feitas para atender as demandas dos projetos.

Um ponto importante é que os requisitos, no sistema, são levantados pelo Engenheiro de Requisitos, acompanhado pela equipe, mas aprovado e validado apenas pelo cliente, de forma a garantir a confiabilidade do requisito.

Diante do exposto, conclui-se que o software SER representa uma solução adequada ao atendimento das necessidades do projetista/analista de ferramentas de apoio para a etapa de projeto e levantamento de requisitos, pois cumpre aos aspectos imprescindíveis recomendados pelo IREB, além das bases: levantamento (elicitação), análise, especificação, validação e gestão de requisitos. Porém, apesar do sistema estar concluído, já ter sido feito mais de 1000 downloads desde seu lançamento, requer experimentos controlados entre equipes para validar a colaboração e melhorias no sistema, o que está em andamento para a conclusão da pesquisa.

O estudo demonstra-se, assim, contribuinte para futuras investigações que busquem a ampliação do software por meio da construção de módulos complementares, como aplicáveis a celulares e **tablets**, a integração com WebService, a migração para linguagem Java e, por fim, de novos diagramas UML.

REFERENCIAS

BARTIÉ, Alexandre. **Garantia de qualidade de software**. Rio de Janeiro: Elsevier, 2002.

CHENG, Betty H.C.; ATLEE. **Joanne M. Research directions in requirements engineering**. Future of Software Engineering, IEEE, 2007.

ENGHOLM JR, Hélio. **Análise e design orientados a objetos**. São Paulo: Novatec, 2013.

IREB. **Syllabus IREB-CPRE-FL v.2.1**. Disponível em: <http://www.abramti.org.br/sites/default/files/arquivos/IREB_CPRE-FL_Syllabus_pt_v2.1.pdf>. Acesso em: 2 set. 2013.

MELO, Alexandre A.; NASCIMENTO, Mauricio G.F. **PHP profissional**. São Paulo: Novatec, 2007.

PRESSMAN, Roger S. **Engenharia de software**: uma abordagem profissional. 7.ed. Porto Alegre: Bookman, 2011.

SILVEIRA, Paulo; SILVEIRA, Guilherme; LOPER, Sérgio; MOREIRA, Guilherme; STEPPAT, Nico; KUNG, Fabio. **Introdução à arquitetura e design de software**. Rio de Janeiro: Elsevier, 2012.

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

STANDISH GROUPINTERNATIONAL. **Chao manifesto 2013**. Disponível em: <http://versionone.com/assets/img/files/ChaosManifesto2013.pdf>. Acesso em: 2 set. 2013.

STUART, Russel; NORVIG, Peter. **Inteligência artificial**. 2.ed. Rio de Janeiro: Elsevier, 2004.

WAZLAWICK, Raul S. **Metodologia de pesquisa para ciência da computação**. Rio de Janeiro: Elsevier, 2009.

1. Mestrando Universidade Federal de Sergipe, Especialista em Redes de Computadores, ESAB, Professor Visitante. Universidade Federal de Sergipe, Campus São Cristóvão. E-mail: menezes.pmm@gmail.com
2. Especialista em Redes de Computadores, Professor Faculdade Impacta Tecnologia. E-mail: thiago@thiagocorreia.com.br
3. Graduanda em Ciências da Computação Universidade Tiradentes. E-mail: thicianecouto@gmail.com
4. Graduanda em Ciências da Computação Universidade Tiradentes E-mail: lariissasantana_n@hotmail.com

Recebido em: 12 de Março 2015
Avaliado em: 18 de Abril 2015
Aceito em: 4 de Maio de 2015