REVISTA DE CIÊNCIAS EXATAS E TECNOLOGIA

DESENVOLVIMENTO DE ABORDAGENS DE ENGENHARIA DE SOFTWARE PARA SISTEMAS NO CONTEXTO DE ENGENHARIA ELÉTRICA

Georgea Rita Burck Duarte – *Faculdade Anhanguera de Pelotas*
Cecilia Sosa Arias Peixoto – *Faculdade Anhanguera de Piracicaba*
Luiz Teruo Kawamoto Junior – *Faculdade Anhanguera de Campinas*
André Ribeiro Lins Albuquerque – *Anhanguera Educacional*
Cesar Candido Xavier – *Anhanguera Educacional*

RESUMO: Para o aluno de Engenharia, as metodologias de elicitação de requisitos e especificação de interfaces homem-computador são nebulosas. Com a finalidade de integrar a Engenharia Elétrica e a Computação, este trabalho aborda os conceitos e metodologias de Engenharia de Software em dois estudos de casos. Tendo como principiais resultados a escolha de padrões de elicitação de requisitos, modelagem de tarefas do software e modelagem dos usuários que vão interagir com o software de ambos os estudos de caso. No estudo de caso do software para controle aplicado a aparelhos de destilação, tem-se desenvolvido uma solução de controle preditivo baseado em modelos. As principais informações do projeto e dos dados operacionais das principais variáveis das colunas A(destilação) e B(retificação) do aparelho em estudo já foram documentadas. O segundo estudo de caso é dedicado à plataforma CUDA desenvolvida pela NVIDIA, pois permite utilizar o elevado poder computacional das placas gráficas que possuem, tipicamente, centenas de núcleos. A plataforma foi testada em dois sistemas operacionais: Windows e Linux.

PALAVRAS-CHAVE:
Ensino Multidisciplinar, Engenharia de Controle, Desenvolvimento de Software, Engenharia Elétrica, Ciência da Computação.

ABSTRACT: For engineering students, methods of requirements elicitation and specification of human-computer interfaces are fuzzy. In order to integrate the Electrical Engineering and Computer Science, this paper discusses the concepts and methodologies of software engineering in two cases. The main results are the choice of standards for requirements elicitation, modeling of software tasks, and modeling of the users who will interact with the software of both case studies. In the case of the software used to control the distillation apparatus, it was developed a model-based predictive control solution. The main project information and operating data of the main variables in columns A (distillation) and B (rectification) of the device under study have been documented. The second case study is dedicated to CUDA platform developed by NVIDIA as it allows to use the high computational power of graphics cards that typically have hundreds of cores. The platform was tested on two operating systems: Windows and Linux.

KEYWORDS:
Multidiscipline Education, Control Engineering, Software Development, Electric Engineering, Computation Science.

Artigo Original
Recebido em: 21/12/2012
Avaliado em: 30/05/2013
Publicado em: 23/05/2014

Publicação
Anhanguera Educacional Ltda.

Coordenação
Instituto de Pesquisas Aplicadas e Desenvolvimento Educacional - IPADE

Correspondência
Sistema Anhanguera de Revistas Eletrônicas - SARE
rc.ipade@anhanguera.com

1. INTRODUÇÃO

No cenário educacional, a cada dia novos recursos e ideias são lançadas em uma busca constante pelo aprimoramento e superação da qualidade do ensino. Atualmente existe um movimento tanto na comunidade de Engenharia Elétrica como na de Computação de integrar cada vez mais, de forma multidisciplinar, ambas as áreas.

Os alunos formados em Engenharia Elétrica e em Engenharia de Controle e Automação aproveitam conceitos oriundos da área de Computação, como desenvolvimento de software, para seu futuro profissional, enquanto os alunos de Computação aplicam suas competências na resolução de problemas complexos como os vindos da área de Engenharia Elétrica. Considerando o número crescente de estudantes dos cursos de Engenharia que se especializam em matérias técnicas relacionadas com desenvolvimento de software, e a necessidade dos alunos de cursos de graduação em Ciência da Computação integrar todos os conceitos de gerência de desenvolvimento de software, este projeto de pesquisa tem por objetivo a aplicação, adaptação e expansão de conhecimentos da Engenharia de Software para o desenvolvimento de sistemas na área de Engenharia Elétrica.

A Engenharia de Software está segmentada em várias subáreas, cobrindo as seguintes fases do ciclo de vida do software: requisitos, análise, projeto, codificação, testes e manutenção (PRESSMAN, 2006). Esta disciplina, nos programas de graduação, é fundamental para trabalhar junto aos alunos os conceitos (requisitos, usabilidade) e as técnicas (prototipagem, avaliação, etc.) necessárias na construção de sistemas de software. Contudo, o número de horas destinadas ao ensino destes conceitos raramente são suficientes. Nestes casos, por causa do tempo, são apresentados exemplos mais reduzidos de sistemas de software e raramente a documentação completa de todas as fases, principalmente no que se refere às subáreas de Especificação de Requisitos (KOTONYA; SUMMERVILLE, 1998) e Interfaces Homem Computador (IHC) (NIELSEN, 1993).

Para De Grande e Martins (2006), o processo de Engenharia de Requisitos objetiva, principalmente, buscar os conhecimentos das regras de negócio e verificar as necessidades do cliente, obtendo uma especificação não ambígua e completa dos requisitos de software, com o intuito de minimizar os erros, as inadequações e as falhas no produto final, a ser entregue ao cliente. Segundo Fred Brooks *apud* Pressman (2006):

> "A parte individual mais difícil da construção de um sistema de software é decidir o que construir [...] Nenhuma outra parte do trabalho prejudica tanto o sistema resultante se feita errada. Nenhuma outra parte é mais difícil de retificar depois".

Um problema clássico quando se faz a elicitação de requisitos de um sistema, é a dificuldade em delimitar a abrangência do sistema a ser desenvolvido, bem como identificar as responsabilidades a serem atribuídas para tal sistema (MARTINS, 2001). Para os softwares provindos da área de engenharia elétrica, esse problema torna-se ainda mais desafiador, pois inexistem técnicas e ferramentas eficazes que ajudem os projetistas a modelar e

raciocinar sobre a integração do ambiente físico com os dispositivos de interface do sistema e o comportamento dos usuários envolvidos.

Por outro lado, o conceito atual de Interfaces Homem Computador mudou significativamente se comparado a quando começou a surgir na década de 50. Hoje, uma interface corresponde a um componente coerentemente projetado que diminui a carga de aprendizado sobre o usuário, além do meio físico que separa a máquina e o humano (SHNEIDERMAN, 1998). Shneiderman e Plasaint (2009) determinam que para o desenvolvimento de um projeto de sucesso é necessário basear o mesmo em quatro pilares: elicitação de requisitos de Interfaces Homem Computador, *guidelines* de projeto de interfaces, testes de usabilidade e ferramentas de projeto de interfaces. Lamentavelmente é muito comum durante o desenvolvimento das interfaces que estas recomendações não sejam seguidas e até conhecidas pelos alunos.

Motivados por estas considerações, neste projeto, está sendo realizada uma pesquisa sobre as metodologias mais adequadas para especificação, modelagem, projeto e implementação de sistemas em dois estudos de casos específicos. O projeto de pesquisa busca atender as especificidades dos estudos de caso e contribuir na academia com justificativas das escolhas e modelos desenvolvidos. Requisitos funcionais, não funcionais, modelagem dinâmica, modelagem de processos, modelagem da interação do sistema, especificação das interfaces homem-computador, especificação de requisitos de integração entre hardware e software, modelagem de componentes são os conceitos foram trabalhados durante os estudos de caso de dois estudos de casos específicos: 1) Sistemas lineares esparsos e, 2) Sistema de Controle Avançado aplicado a Aparelhos de Destilação de Etanol.

A pesquisa tem como fio condutor a investigação e escolha de metodologias para serem aplicadas em cada uma das fases de Engenharia de Software, principalmente na seleção das mais adequadas para elicitação de requisitos e interfaces homem-computador. Estas metodologias foram aplicadas no desenvolvimento de um software, utilizando processamento paralelo, para a resolução de sistemas lineares esparsos baseados nos métodos do gradiente conjugado e de decomposição LU. Documentação elaborada pedagogicamente foram gerados neste primeiro estudo de caso. No segundo estudo de caso foi desenvolvido um sistema de controle avançado aplicado a aparelhos de Destilação de Etanol. Neste estudo de caso também foram gerados relatórios e documentação elaborada pedagogicamente para servir como base de estudo aos alunos de graduação.

2. METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE

A Engenharia de Software é, segundo Sommerville (2003,p. 5), a disciplina que se ocupa de todos os aspectos do desenvolvimento do software, desde seus estágios iniciais de especificação até a manutenção. Diversos ciclos de vida têm sido propostos com fases,

atividades e resultados que geram um produto de software. Entre eles podemos citar Espiral (BOEHM, 1988), Processo Unificado (BOOCH et al, 2000), e as novas abordagens Ágeis de Software (DIAS et.al., 2005), (LARMAN, 2004). Se bem os alunos de Computação conhecem estes ciclos de vida, eles são novos para os alunos de EE. Contudo, a perspectiva por trás das abordagens Ágeis, permeia realidades conhecidas pelos alunos de EE, como o *Lean Software Development* (LSD) (LEAN INSTITUTE BRASIL, 2012). Segundo, Poppendieck (POPPENDIECK, 2007), o LSD é a aplicação dos princípios da Toyota *Product Development System* para o desenvolvimento de software. Esta é umas das razões pelas quais foi escolhida uma metodologia Ágil para elaboração do projeto do estudo de caso e pelas diversas vantagens citadas pelos autores em relação aos ciclos de vida tradicionais (BONOW, 2008);(SILVA;SILVA, 2009);(DIAS et. al., 2005).

Aplicar as metodologias Ágeis é desenvolver um plano adaptativo promovendo entrega evolutiva, iterativa e resposta rápida a mudanças (LARMAN, 2004). Umas das características é a participação ativa do cliente durante o ciclo de vida e a documentação somente do que é essencial ao processo.

Neste projeto foi escolhida a metodologia Ágil Scrum (SCHWABER;BEEDLE, 2002), uma das mais populares e cada vez mais utilizadas em diversos projetos, ela aborda flexibilidade, adaptabilidade e produtividade (BASSI FILHO, 2008). Scrum é de fácil entendimento e aplicação por isso foi escolhida neste projeto.

2.1. SCRUM

A metodologia Scrum, dá ênfase ao gerenciamento e controle do desenvolvimento de software. O ciclo de vida em Scrum é composto por três fases: pré planejamento ou *pre-game phase*, desenvolvimento ou *game phase*, pós planejamento ou *post-game phase* (ABRAHAMSSON et al., 2002) ; (HIGHSMITH, 2002).

A etapa pré é composta do planejamento e a arquitetura. Na etapa de planejamento os requisitos do sistema são informados pelo cliente e registrados em um documento denominado lista de trabalho do produto (*Product Backlog*). Na fase de arquitetura, se elicitam detalhes dos requisitos, padrões, convenções, tecnologias e recursos necessários para o novo sistema..

Na fase de desenvolvimento, os requisitos são divididos em iterações denominadas *Sprints*. Cada iteração tem duração flexível de até trinta dias. Toda a equipe trabalha sobre os requisitos definidos no início de cada iteração entre o *Product Owner* (cliente ou financiador) e o *Scrum Master* ou líder da equipe. Cada iteração implementa uma lista de trabalho da iteração (*Sprint Backlog*). Essa lista contém os requisitos selecionados para a iteração, os quais são extraídos do P*roduct Backlog*. O desenvolvimento de cada iteração ocorre da forma tradicional onde a equipe irá trabalhar unida para atingir o objetivo de entregar parte do

software no intervalo de tempo predefinido.

A etapa de pós planejamento está composta por reuniões sobre o projeto e a versão atual do software é apresentada ao cliente. Nessa fase são feitos os testes na versão final e a documentação necessária é elaborada.

As fases do *Sprint* são (Figura 1):

- 1º Planejamento do Sprint (*Sprint Planning 1*). Nessa etapa, os requisitos do sistema são informados pelo cliente ou *Product Owner* e registrados no documento lista de trabalho do produto (*Product Backlog*). Exemplo do *Product Backlog* do projeto de Controle Avançado Aplicado a Aparelhos de Destilação (Estudo de Caso) é apresentado na Tabela 1, onde é mostrado o Id, título, prioridade, esforço e número de *Sprint*.

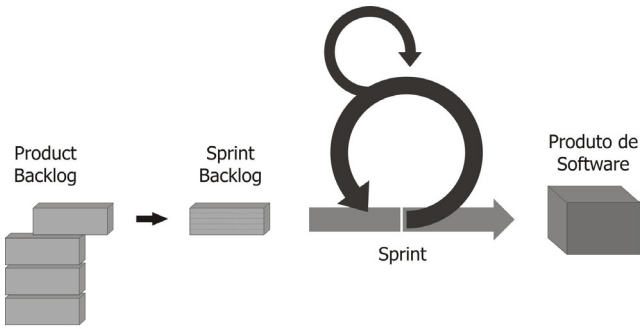

Figura 1– Ciclo de Vida do Sprint utilizado no Projeto

- 2º Planejamento do *Sprint* (*Sprint Planning* 2). Este planejamento consiste em priorizar os requisitos definidos na etapa anterior e realizar estimativas do esforço de implementação de cada requisito. Nesta etapa se adicionam todas as funcionalidades na lista de trabalho da iteração ou *Sprint Backlog*. Um Exemplo do *Sprint Backlog* do Projeto de Controle Avançado Aplicado a Aparelhos de Destilação, é apresentado na Tabela 2. A definição da equipe, ferramentas a serem utilizadas no projeto, treinamento etc. também são abordadas.
- *Sprint*: Com base no *Sprint Backlog*, esta fase consiste na implementação. A equipe irá trabalhar para atingir o objetivo de entregar parte do software em um intervalo de tempo fechado (*Time-Boxing*). Para controlar a execução do *Sprint Backlog*, temos duas opções de ferramentas, o *TaskBoard* (YOSHIMA,2007) e o Kanban (RIES, 2011). O Kanban, é uma prática do movimento Ágil, que apresenta as funcionalidades a serem desenvolvidas por meio de quadros sinalizadores ao igual que *TaskBoard*, mas ela é mais sucinta. No Kanban, são utilizados indicadores para mostrar o estágio de execução de cada funcionalidade. Ele apresenta as tarefas a fazer ("to do"), as sendo feitas "doing" e as completadas "done". Exemplo do Kanban do Projeto de Controle Avançado Aplicado a Aparelhos de Destilação é apresentado na Figura 2 (mês de julho).

- Reunião diária (*Daily Scrum Meeting*): reunião com todos os integrantes da equipe com o objetivo de sincronizar e alinhar o trabalho. O principal objetivo é eliminar quaisquer impedimentos dos membros da equipe ao desenvolver determinada tarefa atribuída e aumentar o desempenho. Geralmente a duração da reunião é curta e em pé (15 minutos).
- Reunião de Revisão (*Sprint Review*): é um importante ponto de inspeção no Scrum. Ela ocorre no último dia do *Sprint* e representa o momento que a equipe e o *Scrum Master* demostram as funcionalidades para o *Product Owner*.
- Retrospectiva (*Sprint Retrospective*): para melhorar o desempenho da equipe de desenvolvimento, após cada entrega, é realizada uma retrospectiva para avaliar e identificar oportunidades evitando assim, possíveis erros nos próximos *Sprints* (BASSI FILHO, 2008). Esta reunião ocorre entre a equipe e o *Scrum Master* buscando uma melhoria continua na produtividade e qualidade.

Tabela 1– Product Backlog do Projeto de Controle Avançado Aplicado a Aparelhos de Destilação

Id	Título	Prioridade	Esforço (dias)	Sprint
1	Pesquisa de Dados	1	30	1
2	Análise dos Dados	1	30	2
3	Interface Homem-Máquina	1	25	3
4	Construção dos modelos	1	30	4
5	Desenvolvimento do especificação da Rede Neural	1	30	5
6	Comunicação	2	30	6
5	Coletor de Dados e Historiador	3	30	7
6	Predição e Ação de Controle	4	30	8
7	Ação de Controle e Comunicação	4	30	10

Tabela 2 – Planejamento do Sprint número 4 do Projeto de Controle: Construção dos Modelos

Id História	Historia/Tarefa	Esforço (h)
1	Especificação de Requisitos	60
2	Projeto	30
3	Desenvolvimento	30
4	Testes	20

Figura 2–Kanban do Projeto de Controle Avançado Aplicado a Aparelhos de Destilação

Dentro do acompanhamento de desempenho das atividades realizadas pela equipe de desenvolvimento, o Scrum sugere além do uso do quadro visual, o *Burndown Chart*. Este último, é um complemento visual do *Product Backlog* que reporta o andamento do projeto, apresentando no eixo X o tempo e no eixo Y as funcionalidades que deverão estar prontas na data estipulada. O *Burndown Chart* apresenta a linha azul (Figura 3) que é a estimativa de esforço calculado, e a linha vermelha, representa o esforço real que a equipe teve durante o *Sprint*. Se a linha vermelha está abaixo da linha azul, a velocidade do Sprint está acima do planejado, caso contrario, o Sprint pode não chegar a cumprir o prazo estabelecido. O *Burndown Chart* apresentado na Figura 3 corresponde ao planejamento de quarto *Sprint* do Projeto de Controle Avançado Aplicado a Aparelhos de Destilação (Figura 4). O *Burndown Chart,* facilita a comunicação com os membros da alta administração da organização (YOSHIMA, 2007).

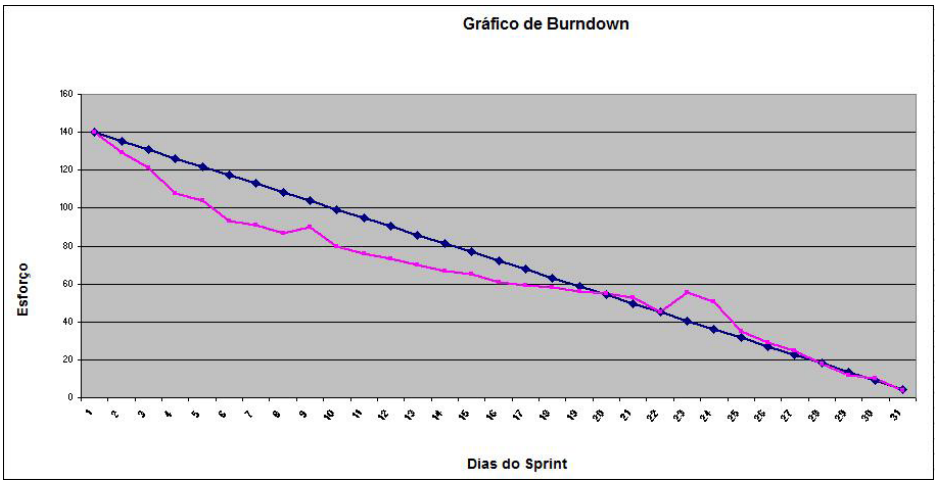

Figura 3– Exemplo de *Burndown Chart* do quarto *Sprint* (Construção dos Modelos) do Projeto de Controle Avançado Aplicado a Aparelhos de Destilação

ID da História	História / Desenvolvimento dos Modelos	Dias do Sprint / Esforço Restante																												
		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
		140	135	131	126	122	117	113	108	104	99	95	90	86	81	77	72	68	63	59	54	50	45	41	36	32	27	23	18	14	9	5
		140	129	121	108	104	93	91	87	90	80	76	73	70	67	65	61	59	58	56	55	53	45	56	51	35	29	25	18	12	10	4
1	Especificação de Requisitos																															
	Entrevistas com Engenheiros de Controle	20	18	15	10	8	6	6	6	5	5	5	5	5	5	5	4	4	4	4	4	4	2	2	2	2	0	0	0	0	0	0
	Definição da Arquitetura do Sistema	14	10	8	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Requisitos Funcionais	8	8	4	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Requisitos não Funcionais	8	3	4	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TOTAL	50	39	31	18	14	6	6	6	5	5	5	5	5	5	5	4	4	4	4	4	4	2	2	2	2	0	0	0	0	0	1
2	Projeto																															
	Diagramas de Classe	10	10	10	10	10	9	9	7	6	5	4	3	2	1	0	0	0	0	0	0	0	0	3	3	0	0	0	0	0	0	0
	Diagramas de Sequencia	8	8	8	8	8	6	6	6	6	5	3	2	1	0	0	0	0	0	0	0	0	0	6	5	0	0	0	0	0	0	0
	Estrutura de Classe	12	12	12	12	12	12	10	8	7	5	4	3	2	1	0	0	0	0	0	0	0	0	6	5	0	0	0	0	0	0	0
	TOTAL	30	30	30	30	30	27	25	21	19	15	11	8	5	2	0	0	0	0	0	0	0	0	15	13	0	0	0	0	0	0	0
3	Desenvolvimento																															
	Implementação das Classes	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	22	20	20	18	17	15	10	7	4	2	1	0	0	0	0	0
	Documentação	5	5	5	5	5	5	5	5	11	5	5	5	5	5	5	5	5	4	4	4	4	3	2	2	1	0	0	0	0	0	0
	TOTAL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Testes	30	30	30	30	30	30	30	30	36	30	30	30	30	30	30	27	25	24	22	21	19	13	9	6	3	1	0	0	0	0	0
	Especificação de Testes	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	9	8	5	3	3	0	
	Testes no Ambiente de Simulação	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	9	7	5	3	3	0	
	Testes em Campo	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	8	6	4	2	
	TOTAL	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	28	25	18	12	10	2

Figura 4– Gestão do Projeto do quarto Sprint do Projeto de Controle Avançado Aplicado a Aparelhos de Destilação

Durante o *Sprint*, o *Scrum Master* irá anotar as atividades feitas, escolhidas pelos programadores e quanto tempo durou, organizado por dia. À medida que as atividades forem concluindo, o esforço diminui e o controle do projeto é realizado no *Burndown Chart.*

Durante a o desenvolvimento de um item do *Product Backlog* são colocadas em prática as fases de Engenharia de Software relacionadas com a etapa de elaboração, onde é efetuado o levantamento de requisitos solidificando-os em modelos que permitam aos avaliadores do sistema o entendimento do fluxo da aplicação, a fase de construção do software e a fase de transição, onde são realizados os testes (REED, 2000). Para poder iniciar a fase de elaboração temos algumas metodologias que auxiliam esta descrição e que serão apresentadas nas próximas seções.

2.2. Engenharia de Requisitos

A Engenharia de Requisitos consiste no conjunto de técnicas empregadas nos processos envolvidos no desenvolvimento dos requisitos do sistema, ou seja, levantamento, detalhamento, documentação e validação (KOTONYA; SUMMERVILLE, 1998). Segundo Somerville (2007) se os requisitos não forem bem levantados, podem ocorrer diagnósticos errados sobre o problema, identificação dos sintomas e não das causas, soluções pobres, e omissão de processos fundamentais. Uma forma de iniciar a elicitação é através da modelagem do domínio do problema. Trabalhos como o de Nasr *et al.* (2002), ainda detalham esta fase como composta de análise do domínio, que consiste em capturar, analisar e organizar as informações que serão utilizadas para o desenvolvimento do sistema, e a fase de modelagem do domínio, na qual se representa a análise por meio de desenhos ou cenários. Para detalhar o domínio, neste trabalho de pesquisa, foi escolhida FBS (HIGHSMITH, 2004). O FBS (*Feature Breakdown Structure)* apresenta de maneira gráfica a hierarquia da composição de funcionalidades em suas respectivas atividades de negócio. Ele representa o domínio seguindo a escala: área de negócio (em azul), atividade de negócio (em verde) e funcionalidade (em amarelo). Na Tabela 3, se apresenta o FBS de um dos estudos de caso.

Tabela 3 – FBS do Projeto de Controle Avançado Aplicado a Aparelhos de Destilação

1. Empresa Santa Cruz
1.1 Operação
1.1.1 Automação
1.1.1.1 Pesquisa e Levantamento de Dados
1.1.1.2 Modelagem e Construção dos Modelos
1.1.1.3 Desenvolvimento da Rede Neural
1.1.1.4 Treinamento da Rede Neural
1.1.1.5 Desenvolvimento do Coletor de dados de Controle e Historiador
1.1.1.6 Executar a predição de controle
1.1.1.7 Efetivar a ação de controle

Se bem FBS contribui para descrição das funcionalidades, normalmente um sistema de tempo real é bem mais complexo e necessita a especificação dos componentes de hardware (HW). Neste sentido, se fez necessário a escolha de algum *template* de especificação de requisitos da área de sistemas de tempo real. Na próxima seção, será apresentada a pesquisa nesse sentido.

2.3. Templates para Elicitação de Requisitos

No âmbito desta pesquisa, foi necessário observar a utilização de diversos padrões na elicitação de requisitos para sistemas de tempo real. Entre estes padrões podemos citar o IEEE padrão 830830-1998, o qual fornece recomendações para uma especificação de requisitos de software, o *template* Volere (VOLERE, 2008), e *template* GERSE (OSSADA; MARTINS, 2010), além de outros modelos mais tradicionais como Redes de Petri, UML e seus diagramas. O importante destes padrões é que permitem satisfazer as visões de diferentes pessoas e compreender o objetivo da construção de um sistema de software. Os *templates* foram selecionados devido a natureza do estudo de caso, que revela uma combinação de hardware (sensores e atuadores) e software (SW) (*firmware*). Nestes projetos normalmente é de fundamental importância a especificação dos requisitos não-funcionais[1] como consumo de energia, tamanho e peso devem ser especificados adequadamente assim como os requisitos funcionais[2]. Nos projetos de sistemas de tempo real, existe um número maior de especialistas em Engenharia, com visões de HW e com pouco conhecimento de metodologias de desenvolvimento de software. O uso de um *template* permite a execução de um conjunto de atividades de elicitação de requisitos, ordenadas numa seqüência lógica que facilita a realização destas atividades (OSSADA; MARTINS, 2010). Os *templates* podem ser usados junto com GEM (MULLER,1997), um método de elicitação de requisitos em grupo. Ele permite que os participantes promovam novas ideias e busquem o consenso durante a especificação. No final da sessão de GEM, recomendamos nesta pesquisa, que se utilize em vez de um relatório, como o proposto por GEM, o preenchimento do *template*.

O *template* VOLERE consiste de um padrão de documentação para especificação de requisitos mais utilizados. Ele consta de várias seções, a saber: diretrizes de projeto, restrições de projeto, requisitos funcionais, requisitos não funcionais e assuntos de projeto, como pode ser visto na Tabela 4. Esta Tabela apresenta a especificação do Projeto de Controle Avançado Aplicado a Aparelhos de Destilação.

1 Descrevem um aspecto não funcional que o sistema deverá atender, como, por exemplo, desempenho, robustez, integração com redes, tempo, etc.

2 São declarações de funções que o sistema deve fornecer, como o sistema deve reagir as entradas específicas e como deve se comportar em determinadas situações (SOMMERVILLE, 2003).

Tabela 4 – Especificação do Projeto de Controle Avançado Aplicado a Aparelhos de Destilação

Seção	Conteúdo
DIRETRIZES DE PROJETO	
1 - Propósito do Produto	Controlar de Forma de Eficiente Variáveis Críticas no Processo de Destilação de Etanol
2 - Comprador, Cliente e Interessados (stakeholders)	Gerentes Industriais
3 - Usuários do Produto	Engenheiros de Controle e Automação
RESTRIÇÕES DE PROJETO	
4 - Restrições Necessárias	Os sensores devem estar calibrados. A estrutura de comunicação entre o sistema de controle e os transmissores e atuadores deve ser garantida pela infra-estrutura do cliente.
5 - Convenções para Nomes e Definições	MPC, Suporvisório, Sistema de Controle
6 - Fatos e Suposições Relevantes	influências exteriores que fazem alguma diferença para o produto
REQUISITOS FUNCIONAIS	
7 - O escopo do Trabalho	Colunas de Destilação e Retificação
8 - O escopo do Produto	Aplicação de controle preditivo nas malhas de controle de pressão da Coluna A (destilação) e da temperatura da coluna B (retificação).
9 - Requisitos Funcionais e de Dados	Coleta e armazenamento dos dados relativos ao controle (variáveis: de controle, manipuladas e os distúrbios); Construção e Atualização dos Modelos Preditivos; Execução dos algoritmos de controle; Envio à ação de controle.
REQUISITOS NÃO-FUNCIONAIS	
10 - Requisitos Sensoriais	Protótipo
11 - Requisitos de Usabilidade	Ainda não foram estipulados para este projeto
12 - Requisitos de Desempenho	Tempo de Execução total 200 ms
13 - Requisitos Operacionais	Robusto, possuir redundância.
14 - Requisitos de Manutenibilidade e Portabilidade	Não se aplica.
15 - Requisitos de Segurança	Somente usuários com privilégio poderão utilizar o Software.
16 - Requisitos de Cultura e Política	Não se aplica.
17 - Requisitos Legais	Não se aplica.
ASSUNTOS DE PROJETO	
18 - Questões em Aberto	Interface Final do Software.
19 - Soluções Prontas	Algoritmo de Controle e Modelos Preditivos Validados.
20 - Novos Problemas	Novas malhas de Controle a serem estudadas.
21 – Tarefas	Pesquisa e Levantamento de Dados. Seleção dos Algoritmos de Controle. Seleção das Varaiáveis de Controle. Escolha dos Modelos.
22 – Portes	tarefas de conversão a partir de sistemas existentes
23 – Riscos	Não disponibilidade de hardware, Coleta de Dados não cofiáveis na planta industrial.
24 – Custos	Aproximadamente 20 dias de Atividade Técnica em Campo
25 - Documentação do Usuário e Treinamento	plano para construir as instruções e documentação para usuários
26 - Sala de Espera	Implementação de Lógica Fuzzy, Aperfeiçoamento da Interface gráfica (usabilidade).
27 - Ideias para Soluções	Vide item anterior.

O template GERSE, também utilizado neste trabalho, é um guia para elicitação de requisitos, através de um conjunto de sete categorias. O guia está dividido em duas fases (pré fase e fase principal). Na pré-fase, as atividades foram classificadas em três categorias: organização de contexto, definição de stakeholders e requisitos de alto nível. A fase principal consta de quatro categorias: hardware, software, definição de métricas de qualidade e definição de requisitos de produção. O template GERSE é utilizado para especificar e documentar os requisitos, atividades fundamentais no desenvolvimento de projetos de software. Conforme Ossada e Martins (2010), ao finalizar as atividades da pré-fase e fase é possível analisar os requisitos de alto nível e descrever os requisitos técnicos. Na Tabela 5 é apresentada cada categoria do template GERSE e a especificação do estudo de caso de Solução de Sistemas de Equações Lineares utilizando o template GERSE para sua especificação.

Tabela 5 – Especificação do Estudo de Caso de Solução de Sistemas de Equações Lineares utilizando GERSE

Seção	Conteúdo
1 – CONTEXTO	
1.1 Obter o propósito e as metas organizacionais do produto frente ao mercado	estabelecer um procedimento de instalação do ambiente CUDA 5.0 nas plataformas Windows e Linux; identificar a plataforma que apresenta melhor desempenho em transferência de dados entre a CPU – GPU – CPU; identificar a plataforma cujo o desempenho foi superior na solução de sistemas lineares esparsos.
1.2 Definir as características gerais do produto	- capaz de resolver sistemas lineares esparsos armazenas no formato Matrix Market (MM) na GPU.
1.3 Definir os impactos organizacionais com o desenvolvimento do produto	- aumento de produtividade pela redução de tempo na solução de sistemas lineares esparsos.
1.4 Definir os impactos negativos com o não desenvolvimento do produto	- manutenção do atual nível de produtividade.
1.5 Definir as expectativas de tempo total de desenvolvimento do produto.	- oito meses.
1.6 Obter o público a ser atingido	- projetistas, engenheiros, físicos e matemáticos.
1.7 Recuperar projetos de sistemas legados	- Biblioteca CUSPARSE.
2 – STAKEHOLDERS	
2.1 Definir principais stakeholders	- projetistas, engenheiros, físicos e matemáticos.
2.2 Definir stakeholder especialista de domínio	- engenheiro que trabalha no projeto
2.3 Definir stakeholders contrários ao projeto	- N/A.
2.4 Definir o perfil do usuário	- projetistas, engenheiros, físicos e matemáticos.

3 - REQUISITOS DE ALTO NÍVEL	
3.1 Definir as funções do produto	- resolver sistemas lineares esparsos armazenados no formato MM por meio da GPU. Este produto terá como entrada um sistema linear esparso armazenado na formato MM, um vetor de termos independentes e um vetor para armazenamento da solução. A saída é um vetor com a solução do sistema linear esparso.
3.2 Definir as restrições do produto	a dimensão do sistema linear esparso está limitada à quantidade de memória disponível pela placa gráfica. O uso da GPU permite acelerar a solução de sistemas lineares esparsos.
3.3 Definir as restrições físicas do ambiente	- N/A.
3.4 Definir as características de consumo de potência	- computador pessoal com placa gráfica NVIDIA.
3.5 Definir as características físicas e mecânicas	- N/A.
3.6 Definir a interface	- N/A.
3.7 Definir as situações críticas	Tentar resolver um sistema linear esparsos cuja necessidade de memória ultrapassa a ofertada pela placa gráfica.
3.8 Definir o grau de confiabilidade	- Confiável.
3.9 Definir solução encontrada	- Por meio da biblioteca CUDA utilizar a capacidade de processamento das placas gráficas para resolver sistemas lineares esparsos.
3.10 Definir a estimativa de custo	- aproximadamente R$ 5.000,00
4 - ATIVIDADES DE IDENTIFICAÇÃO DE HARDWARE	
4.1 Definir Sensores	- N/A.
4.2 Definir Atuadores	- N/A.
4.3 Definir interação com o usuário	- N/A.
4.4 Definir Interrupções de Hardware	- N/A.
4.5 Definir Botões	- N/A.
4.6 Definir memórias	- N/A.
4.7 Definir portas de comunicação externa	- N/A.
4.8 Definir requisitos de componentes	- N/A.
4.9 Definir requisitos de layout da placa controladora	- N/A.
4.10 Definir parâmetros de HW legados	- N/A.
4.11 Definir parâmetros de COTS especiais	- N/A.
4.12 Lista de microcontroladores compatíveis	- N/A.
5- ATIVIDADES DE ELICITAÇÃO DE FUNÇÕES DE SOFTWARE	
5.1 Definir variáveis de ambiente	- N/A.
5.2 Definir funções de software	- N/A.
5.3 Definir exceções	- N/A.
5.4 Definir as funções de interrupções	- N/A.
5.5 Definir requisitos de idiomas	- N/A.
5.6 Definir interface de comunicação	- N/A.
5.7 Definir funções de monitoramento	- N/A.
5.8 Definir as funções de armazenamento de dados	- são criadas funções de comunicação com a GPU e, tipicamente, os dados a serem armazenados são do tipo float ou double.

6. DEFINIÇÃO DE MÉTRICAS DE QUALIDADE	
6.1 Definir grau de segurança	- N/A.
6.2 Definir desempenho	- No mínimo superior à 1.1 frente à CPU na solução de sistemas lineares esparsos.
6.3 Definir métricas de manutenção	- N/A.
7. DEFINIÇÃO DE MÉTRICAS DE PRODUÇÃO	
7.1 Definir restrições de linha de produção	- N/A.
7.2 Definir características da embalagem	- N/A.

Podemos destacar no *template* GERSE, a definição dos *Stakeholders*, do especialista do domínio e da definição do perfil. Esta última descrição permite estabelecer uma âncora com o projeto de Interfaces Homem-Computador, como será apresentado na próxima seção.

2.4. Especificação das Interfaces Externas ou Interfaces Homem Computador

Durante a fase de especificação dos requisitos são apontadas as interfaces externas, isto é, como o software sendo especificado interage com as pessoas. Neste contexto, surge um novo âmbito de estudo, o de Interfaces Homem Computador. Esta área, também multidisciplinar, consta com ciclos de vida, recomendações e diagramas próprios para o desenvolvimento das interfaces com o usuário. Podemos destacar como fases importantes dentro da análise de requisitos das interfaces, as subfases de análise do perfil do usuário, análise do contexto da tarefa, análise das possibilidades e restrições da plataforma e análise de princípios gerais para o projeto de interfaces (MAYHEW,1999).

Na fase de análise do perfil do usuário são elicitadas informações tais como atributos pessoais, habilidades e competências dos usuários. Neste projeto foi escolhida a metodologia Personas (CASAS, et al., 2008), para descrição do perfil. Na fase de análise de tarefas se descrevem as tarefas envolvidas, objetivos, resultados e estrutura. Neste projeto foi escolhida a metodologia COMM (JOURDE, et. al. 2010), extensão da metodologia CTT - *Concur Task Tree* (Paternò, 2000) muito utilizada para descrição das tarefas. A metodologia Personas consiste em criar uma pessoa fictícia que irá interagir com o sistema, em outras palavras, "personas" representam utilizadores do sistema e são definidas no conhecimento de usuários reais. É uma metodologia para identificar o perfil do usuário baseado em estereótipos de usuários elicitados para o sistema. Durante a elicitação dos perfis dos usuários, informações como nomes, personalidades, motivações e até fotos são levantadas, como apresentado na Figura 5.

Figura 5 – Elicitação do Modelo do Usuário com a Metodologia “Personas” para o Projeto de Controle.

Em relação à elicitação das tarefas, o Modelo CTT ou Árvore de Tarefas Concorrentes é comumente usado para especificar as atividades dos usuários com um determinado software. O CTT possui uma notação específica, em torno da idéia de que o objetivo do usuário ao realizar uma tarefa, pode ser traduzido como uma modificação do estado do sistema ou uma consulta a um recurso do sistema (WINCKLER; PIMENTA, 2004). Neste sentido, as tarefas são classificadas em quatro tipos, a saber, (BARBOSA; SILVA, 2010): tarefas do usuário, realizadas fora do sistema; tarefas do sistema, em que o sistema realiza um processamento sem interagir com o usuário; tarefas interativas, entre usuário e sistema e tarefas abstratas, que permitem a representação de uma composição de tarefas que auxilie a modelagem. Em CTT também são utilizados operados temporais que permitem definir o relacionamento de tarefas em um mesmo nível. Entre os operadores podemos citar (PATERNÓ, 1997): escolha, independência na ordem, concorrência independente, concorrência com troca de informação, desativação, suspende/continua, habilita e habilita com passagem de informações. Além destes operadores existem os unários (aplicáveis a uma tarefa individualmente) como iteração (*), opcionalidade ([T]) ou tarefa com conexão entre outros modelos (PATERNO,1997). Uma extensão de CTT é COMM (*Notation for Specifying Collaborative and MultiModal Interactive Systems*), onde se inclui o conceito de interação e tarefa multi-modal (diferentes modos de iteração). COMM foi desenvolvida tendo em mente identificar as atividades colaborativas (perfil) e especificar a interface envolvendo várias modalidades de interação (tarefa modal). Um sistema interativo é multimodal se os usuários podem interagir com o sistema usando a pelo menos dois (entrada ou saída) modalidades de interação, quer operando em paralelo ou não. Neste contexto, podemos classificar a tarefa segundo o estilo modal: teclado, voz, toque ou mouse. O COMM possui uma ferramenta de edição gratuita, o e-COMM (JOURDE et.

al, 2010), que possibilita a especificação de tarefas segundo a metodologia, as tarefas modais tem uma tarja verde adicionada, desta maneira elas são facilmente visíveis na modelagem.

Uma vez estabelecida a tarefa modal, que denota uma ação realizada em um dispositivo físico, em segundo lugar, se devem refinar os operadores CTT (temporais) usando as relações entre modalidades definidas em Vernier e Nigay (2000) e baseadas no trabalho de Allen (1983). A Figura 6, apresenta um exemplo com a notação COMM e fazendo uso do editor eCOMM. A Tabela 6 mostra a notação empregada no editor eCOMM.

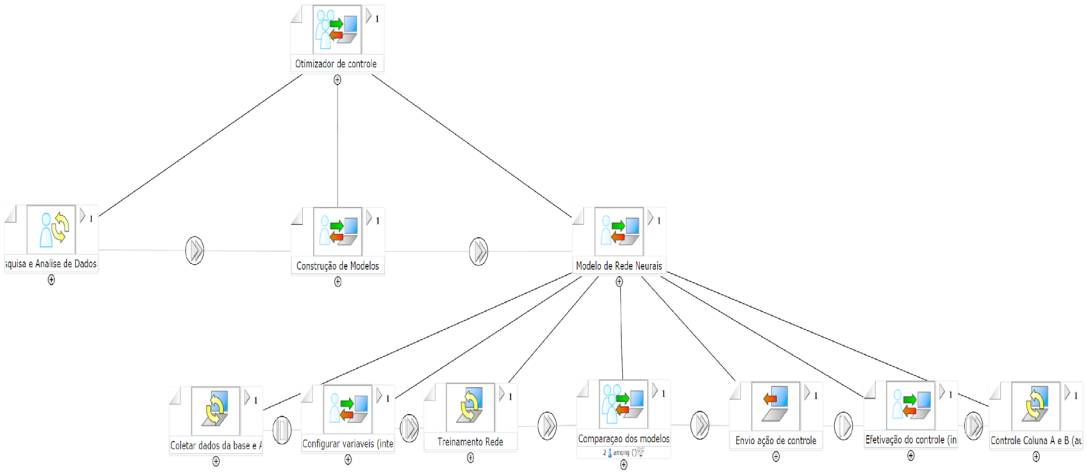

Figura 6 – Exemplo no eCOMM Editor (JOURDE et al.,2010).

Tabela 6 – e-COMM Editor

	Tarefa de interação multi-usuário (pelo menos 2 usuários)
	Tarefa de computação de sistema (0 usuários)
	Tarefa de apresentação do sistema (0 usuários)
	Tarefa mental do usuário (1 usuário)
	Ação do usuário (1 usuário)
	Tarefa interativa (1 usuário)
	Tarefa de coordenação multi-usuário (pelo menos 2 usuários)
	Ação de grupo (pelo menos 2 usuários)

Segundo Sinnig et. al. (2010), o modelo de tarefas pode ser usado junto com outro método de especificação bem conhecido na área de Computação, os casos de uso (BOOCH

et. al., 2000). Ambos modelos são baseados em cenários, permitindo assim a captura de cenários de uso do sistema em consideração. É possível usar a especificação da tarefa com CTT ou COMM, como mostrado na Figura 6, e descrever o caso de uso com a funcionalidade do sistema por meio de um cenário de sucesso principal e extensões. Na documentação dos estudos de caso, foi realizada esta ligação entre os casos de uso e o modelo das tarefas.

Finalmente, as metodologias selecionadas foram resumidas e apresentadas afim de poder documentar as diversas fases onde elas são usadas. A Figura 7 mostra a documentação (resumida) das metodologias de Engenharia de Software que foram adicionadas neste trabalho de pesquisa.

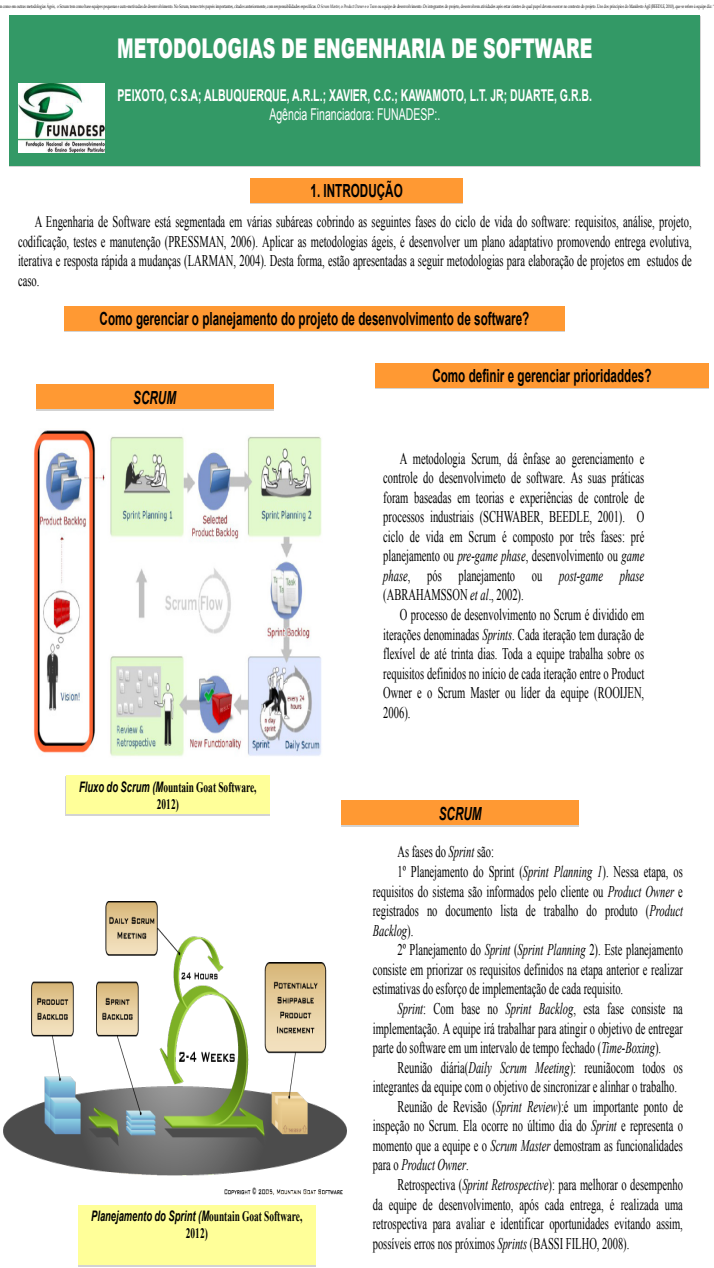

Figura 7– Metodologias de Engenharia de Software.

Sobre os detalhes das metodologias, foram feitos *banners* para apresentação das metodologias de uma maneira mais lúdica para os alunos de EE e Computação.

3. CONSIDERAÇÕES FINAIS

Neste trabalho de pesquisa, se teve como objetivo integrar as diversas metodologias, indicando sua fase de utilização e ampliando a escolha para o âmbito de Engenharia Elétrica e Computação. Nestes meses do projeto, foram realizadas as pesquisas bibliográficas do estado da arte na Engenharia de Requisitos, Metodologia Ágeis de gerenciamento, Modelagem de Tarefas e Modelagem de Usuários. Em relação ao gerenciamento dos projetos foi utilizado Scrum, que possibilitou o controle das atividades de desenvolvimento, contudo, foi necessário incluir atividades de pesquisa no *Product Backlog*, nessa situação nenhum software foi desenvolvido durante o *Sprint*, mas estudo, pesquisa e simulações foram feitas a fim de determinar a melhor escolha. Neste sentido usamos Scrum tanto para gerenciar tarefas de desenvolvimento como tarefas de pesquisa, o que se apresenta como um detalhe inovador.

Outra contribuição do projeto está relacionada com o projeto das interfaces homem-computador. Autores como Sinnig et. al. (2010), afirmam que uns dos inconvenientes dos métodos de design de interface do usuário é que são mal integrados nos processos de desenvolvimento de software, quando não esquecidos. Dentre os problemas citados quando as metodologias não são seguidas ou integradas num processo de gerencia de projetos, existem as inconsistências e redundâncias produto das especificações trabalhadas de forma independente (SINNIG et al., 2010). Segundo Paternò (2001), notações especializadas devem ser utilizadas de uma forma complementar e integrada no processo comum e fácil de usar por Engenheiros e *Designers* de interface.

Nesse sentido podemos destacar a escolha de três formatos de *templates* para fundamentar a documentação dos estudos de caso. Os mesmos já foram documentados nestas fases seguindo a proposta apresentada neste relatório. Para a modelagem das tarefas foi escolhida a metodologia COMM, extensão de CTT com o editor disponível por seus autores em: http://giove.cnuce.cnr.it/ctte.html. Para a modelagem dos usuários foi escolhida a metodologia Personas, apresentada neste relatório. Para a elicitação de requisitos foram escolhidos dois modelos que auxiliaram os engenheiros na especificação evitando erros comuns de falta de documentação, clareza ou confusão entre requisitos funcionais, não funcionais juntamente com detalhes da interface do usuário. O uso dos *templates*, facilita a manutenção das especificações, e detalhes sobre as interfaces são feitas no início do processo como recomendado por autores como Cockburn (2001) e Sinning et al. (2010).

Nesta pesquisa, se sentiu a necessidade de relacionar a especificação de Personas, com recomendações especificas para o projeto de interfaces. Isto poderia ser realizado utilizando um sistema especialista desenvolvido para tal fim, o GuideExpert (CINTO; PEIXOTO, 2010). Assim, a especificação junto com a utilização da ferramenta permitiria definir a persona e estabelecer as recomendações que darão norte ao projeto das interfaces. Realizar estas fases

juntas permite refinar as recomendações conforme o perfil de persona sendo modelado, sem necessidade de ter que voltar para a definição, como seria na fase do ciclo de vida de Mayhew (1999). O uso do sistema especialista, ajuda na rapidez dessa seleção de recomendações e permite também selecionar questões que serão posteriormente avaliadas nos testes de usabilidade do projeto. A utilização desta ferramenta (GuideExpert) será proposta como continuação deste trabalho de pesquisa.

REFERÊNCIAS

ABRAHAMSSON, P., SALO, O., RONKAINEN, J.; WARSTA, J. Agile software development methods Review and Analysis. Finlândia: VIT Publications, 2002.

ALLEN, J. Maintaining Knowledge about Temporal Intervals. Comm. of the ACM, 1983, 26(11), ACM Press. 1983. p. 832-843.

BASSI FILHO, D. Experiências com desenvolvimento ágil. São Paulo: USP, 2008. 170 p.

BARBOSA, S. D. J. E SILVA, B. S. Interação humano-computador. Novatec editora, 2010.

BOEHM, B. A Spiral Model of Software Development and Enhancement. IEEE Computer, vol. 25, 5 May 1988, pp 61-77.

BONOW, R. Sugestão e Modelagem de Práticas do Desenvolvimento Ágil para Aplicação em Desenvolvimento Distribuído onshore insourcing. Monografia Curso de Bacharelado em Ciência da Computação do Instituto de Física e Matemática da Universidade Federal de Pelotas, RS. 2008. 91p.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I., UML: Guia do Usuário, 4. ed. Rio de Janeiro, Campus, 2000. p. 472

CASAS, R. et. al. User Modelling in Ambient Intelligence for Elderly and Disabled Peoples, Proceedings ICCHP, Heidelberg: Springer-Verlag Berlin, 2008. p. 114-122.

DIAS, M., MAXIMIANO, A. Um novo enfoque para o gerenciamento de projetos de desenvolvimento de software. Dissertação de Mestrado. São Paulo. USP. 2005.

HIGHSMITH, J. Agile Project Management: Creating Innovative Products. AddisonWesley, 2004

JOURDE, F., LAURILLAU, Y., MORÁN, A., AND NIGAY, L., Towards Specifying Multimodal Collaborative User Interfaces: A Comparison of Collaboration Notations, in Proceedings of DSV-IS'08 (Kingston, Canana, July 2008), Springer, 281-286. 2008.

JOURDE, F., LAURILLAU, Y. AND NIGAY, L. COMM Notation for Specifying Collaborative and multimodal Interactive Systems. EICS'10, Copyright 2010 ACM 978-1-4503-0083-4/10/06. June 19–23, 2010, Berlin, Germany.

KOTONYA, G.; SOMMERVILLE, I.; Requirements Engineering: Processes and Techniques, John Wiley & Son, Chichester, Englad, 1998.

LARMAN, C. Agile and Iterative Development: A Manager's Guide. Boston: Addison-Wesley, 2004. 342 p.

LEAN INSTITUTE BRASIL. Disponível em <http://www.lean.org.br>. Acesso em Novembro de 2012

MARTINS, L., Uma Metodologia de Elicitação de Requisitos de Software Baseada na Teoria da Atividade; Tese de Doutorado (UNICAMP), Campinas, SP, agosto de 2001.

MAYHEW, D., Principles and Guidelines in Software User Interface Design. New Jersey: Prentice Hall, 1992.

MULLER, M. The Group Elicitation Method for Participatory Design and Usability Testing. G u y a . B o y Methods & Tools. Instituto Europeu de Ciências Cognitivas e Engenharia (EURISCO).

Interactions. March-April, 1997

NIELSEN, J., Usability Engineering. Boston, Academic Press, 1993.

NIELSEN, J., Agile Development Projects and Usability. Disponível em: <http://www.useit.com/alertbox/agile-methods.html>. Acesso em: 4 dez. 2011.

NIGAY, L., COUTAZ, J., A Generic Platform for Addressing the Multimodal Challenge, in Proceedings of CHI'95 (Denver CO, May 1995), ACM Press, 1995, p.98- 105.

NASR E.; MCDERMID J.; BERNAT G. Eliciting And Specifying Requirements With Use Cases For Embedded Systems., Proceedings at 7th International Workshop on Object-Oriented Real-Time dependable systems (WORDS 2002), 2002.

OSSADA, J., MARTINS, L., GERSE: Guia de Elicitação de Requisitos para Sistemas Embarcados de Pequeno e médio porte, Universidade Metodista de Piracicaba- Mestrado em Ciência da Computação. UNIMEP: Piracicaba.SP. 2010.

PATERNO, M. M. ConcurTaskTrees: a Diagrammatic Notations for Specifying Task Models, Proceedinds of INTERACT 97, Sydney, Chapman&Hall. 1997. p. 362-369.

PATERNÓ, F. Task models in interactive software systems. In Handbook of Software Engineering and Knowledge Engineering, S. K. Chang, Ed. World Scientific Publishing Co., 2001.

PRESSMAN, R. Engenharia de Software, 6. ed., Makron Books, 2006.

POPPENDIECK, M. Lean Software Development. ICSE COMPANION '07 Companion to the proceedings of the 29th International Conference on Software Engineering. 2007. p. 165-166.

RIES, E. The Lean Startup. São Paulo: Lua de Papel, 2011.

ROBERTSON J.; ROBERTSON S. Mastering the requirements Process. 2. ed. Addison Wesley, 2006.

SCHWABER K.; BEEDLE, M. Agile Software Development with SCRUM, Prentice-Hall. 2002.

SHNEIDERMAN, B., Designing the User Interface: Strategies for Effective Human-computer Interaction. 3. ed. Boston: Addison Wesley Longman, Inc., 1998.

SHNEIDERMAN, B.; PLAISANT, C., Designing the User Interface: Strategies for Effective Human-computer Interaction. 4. ed. Boston: Addison Wesley Longman, Inc., 2009.

SILVA, F., SILVA, V. Estudo Comparativo dos processos de desenvolvimento de software ágil e tradicional, sob a ótica do guia PMBOK. Monografia. Curso de Pós Graduação em Produção e Sistemas. Instituto Federal de Educação, Ciências e Tecnologia Fluminense- Campus Campos-Centro. Campos do Goytacazes. RJ. 2009. 63p

SINNIG, D.; MIZOUNI, R.; KHENDEK, F. Bridging the Gap: Empowering Use Cases with Task Models. EICS'10. ACM 978-1-4503-0083-4/10/06. June 19–23, 2010, Berlin, Germany.2010.

SOMMERVILLE, I. Engenharia de Software, 8. Ed. São Paulo: Pearson Education do Brasil, 2007.

VERNIER, F., NIGAY, L., A Framework for the Combination and Characterization of Output Modalities, in: Proceedings of DSV-IS'00 (Limerick, Ireland, June 2000), Springer, 2000. p. 32-48.

VOLERE. Requirements Specification Template. Disponível em: <http://www.volere.co.uk>. Acesso em 7 jun. 2012.

WINCKLER, M. A. A.; PIMENTA, M. S. Análise e Modelagem de Tarefas. VI Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais. IHC 2004. Curitiba. Brasil. 17-20 Outubro. 2004

YOSHIMA, R. Gerenciamento de Projetos com Scrum. ASPERCOM. 2007.