PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DE COMPUTAÇÃO

FTSPROC - UM PROCESSO PARA MINIMIZAR AS DIFICULDADES DE PROJETOS QUE ADOTAM A ESTRATÉGIA *FOLLOW-THE-SUN*

Por

ESTEVÃO RICARDO HESS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Jorge Luis Nicolas Audy

Porto Alegre

2012

H586F Hess, Estevão Ricardo
FTSProc – um processo para minimizar as dificuldades de projetos que adotam a estratégia Follow-the-Sun / Estevão Ricardo Hess. – Porto Alegre, 2012.
136 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Jorge Luis Nicolas Audy.

1. Informática. 2. Engenharia de Software. 3. Sistemas Distribuídos. I. Audy, Jorge Luis Nicolas. II. Título.

CDD 005.1

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

AGRADECIMENTOS

Inicialmente, agradeço a minha noiva Adriana, que teve paciência e me deu a força necessária nesses dois últimos anos;

Aos meus pais, que são os exemplos que eu levo sempre comigo, e meus irmãos Cassiano e Bia. Todos, mesmo longe, sempre estiveram comigo quando eu precisei;

Aos profissionais do TECNOPUC e alunos do PPGCC que participaram do experimento para avaliação do processo, pela disponibilidade e seriedade com que trataram este experimento;

Ao convênio Dell/PUCRS, pelo auxílio com a bolsa de pesquisa tão necessária para a realização deste trabalho;

À PUCRS pela oportunidade e pela infraestrutura a mim oferecida;

Aos colegas de mestrado, em especial ao Rafael Audy e Josiane Kroll, e aos Professores Dr. Rafael Prikladnicki e Dr. Sabrina Marczak pelas importantes revisões e contribuições neste trabalho;

Ao Prof. Dr. Ricardo Bastos, pelo tempo despendido e por suas contribuições dadas em todas as avaliações intermediárias desta pesquisa;

À Prof. Dra. Elisa Huzita, pela sua disponibilidade e contribuições durante a banca;

Ao meu orientador, Prof. Dr. Jorge L. N. Audy, pela oportunidade, pela exigência, pelo constante cuidado e atenção ao meu trabalho e pelas críticas e sugestões que tanto ajudaram no decorrer da pesquisa.

A todos, muito obrigado.

FTSProc - Um Processo para Minimizar as Dificuldades de Projetos que Adotam a Estratégia *Follow-the-Sun*

RESUMO

Em busca de vantagens competitivas, tais como redução de custo e ganho de produtividade, cada vez mais as organizações optam por distribuir seus processos de desenvolvimento de software em países com custo de produção mais acessível. Os projetos estão também cada vez mais sendo desenvolvidos em ambientes geograficamente distribuídos, caracterizando o desenvolvimento distribuído de software. Entretanto, os desafios inerentes a este ambiente de desenvolvimento de software são significativos. Dentre estas adversidades está a diferença de fuso horário, a qual pode ser também encarada como uma vantagem, através da aplicação da estratégia *Follow-the-Sun*. Entretanto, a estratégia *Follow-the-Sun* apresenta alguns desafios, principalmente durante a transferência de trabalho de um centro de desenvolvimento para outro. Portanto, o foco desta pesquisa é apresentar um processo para amenizar estas dificuldades inerentes aos projetos que utilizam a estratégia *Follow-the-Sun*, focando na fase de desenvolvimento do ciclo de vida de software. Foi também realizado um experimento para avaliar a eficiência do processo proposto. Os resultados encontrados mostram indícios que o processo criado realmente ameniza as dificuldades encontradas na aplicação da estratégia *Follow-the-Sun*.

Palavras Chave: Desenvolvimento distribuído de software, Desenvolvimento global de software, Engenharia de software, Follow-the-sun, Round the clock, Time to market.

FTSProc - Um Processo para Minimizar as Dificuldades de Projetos que Adotam a Estratégia *Follow-the-Sun*

ABSTRACT

Searching for competitive advantages as low cost and productivity gains, organizations choose to distribute their software development process to other countries with more affordable production costs. Increasingly, projects are being developed in geographically distributed environments, featuring the distributed software development. However, the challenges inherent in this software development environment are significant. Among these challenges is the time zone difference, which can also be tackled as an advantage, through the use of the Follow-the-Sun development. However, the Follow-the-Sun strategy presents some challenges, mainly alongside the hand-offs. Therefore, this research focuses to present a development process to alleviate the challenges found in project that uses the Follow-the-Sun strategy, focusing in the development phase from the software development life cycle. Yet, it performs an experiment to evaluate the created process' efficiency. The findings from the experimental process shows evidences the created process actually alleviate the challenges found in the Follow-the-Sun strategy.

Keywords: Distributed software development, Global software development, Software engineer, Follow-the-sun, Round the clock, Time to market.

LISTA DE TABELAS

Tabela 1. Comparação entre as definições apresentadas. 39

Tabela 2. Comparação entre os experimentos apresentados. 48

Tabela 3. Comparação entre os trabalhos relacionados. 52

Tabela 4. Conhecimento dos Participantes. 82

Tabela 5. Distribuição do fator sobre os tratamentos. 83

Tabela 6. Instrumentação do experimento realizado. 83

Tabela 7. Validade do experimento realizado. 84

Tabela 8. Tempos para realização de cada *shift*. 87

Tabela 9. Tempos do projeto *FTSProc*. 87

Tabela 10. Tempos do projeto *ad hoc*. 87

Tabela 11. Requisitos implementados. 89

Tabela 12. Trabalho ocorreu de forma adequada? 90

Tabela 13. Percebe como o trabalho deve ser continuado? 90

Tabela 14. Acredita que a tranferência acarretou um *overhead*? 91

LISTA DE FIGURAS

Figura 1. Configurações de DDS, com destaque para *Offshore*, adaptado de [AUD07]. 24

Figura 2. Relacionamento entre empresas, adaptado de [AUD07]. 28

Figura 3. Nível de confiança durante projeto [AUD07]. 32

Figura 4. Disposição inicial das equipes, adaptado de [SET07]. 41

Figura 5. Duração do experimento, adaptado de [CAR09]..... 44

Figura 6. Desenho de pesquisa. 55

Figura 7. Diagrama de atividades do processo proposto. 61

Figura 8. *FTSProc*: Processo Proposto. 63

Figura 9. Detalhes do estado 1 do *FTSProc*..... 63

Figura 10. Detalhes do estado 2 do *FTSProc*..... 65

Figura 11. Detalhes do estado 5 do *FTSProc*..... 67

Figura 12. Casos de Uso da aplicação desenvolvida. 70

Figura 13. Tela para iniciar um dia de trabalho (shift). 74

Figura 14. Tela para finalizar um dia de trabalho (*shift*). 75

Figura 15. Relatório de quantidade de testes cobertos em cada shift. 76

Figura 16. Relatório da duração de cada shift. 77

Figura 17. Etapas do processo experimental, adaptado de [WOH00]. 78

Figura 18. Distribuição das equipes. 86

LISTA DE QUADROS

Quadro 1. Especificação do caso de uso “Consultar relatório de hand-off”.....71

Quadro 2. Especificação do caso de uso “Iniciar shift”.....72

Quadro 3. Especificação do caso de uso “Preencher formulário de hand-off”.....73

LISTA DE ABREVIATURAS E SIGLAS

AS	Arquitetura de Software
DDS	Desenvolvimento Distribuído de Software
ES	Engenharia de Software
FTS	*Follow-the-sun*
GSD	*Global Software Development*
SDLC	*Software Development Life Cicle*
SOA	*Software Oriented Architecture*
TDD	*Test-driven Development*
VPN	*Virtual Private Network*

SUMÁRIO

RESUMO vii

ABSTRACT ix

1 INTRODUÇÃO 21

1.1 Justificativa 22

1.2 Objetivos 23

1.3 Contexto 23

1.4 Estrutura 25

2 BASE TEÓRICA 26

2.1 Desenvolvimento DiStribuído de Software 26

2.2 Desenvolvimento Follow-the-Sun 35

2.2.1 Follow-the-Sun: Conceitos 36

2.2.2 Estudos em Follow-the-Sun 40

2.3 Trabalhos Relacionados 49

2.3.1 Análise comparativa 52

3 METODOLOGIA DE PESQUISA 55

3.1 Desenho de Pesquisa 55

3.2 Método de Pesquisa 56

4 *FTSProc*: PROCESSO PROPOSTO 59

4.1 Ferramenta de apoio ao *FTSProc* 68

5 ESTUDO EXPERIMENTAL 78

5.1 Definição 78

5.2 Planejamento 80

5.3 Operação 85

5.4 Análise e Interpretação dos resultados 88

5.4.1 Análise quantitativa 88
5.4.2 Análise qualitativa 90
5.4.3 Lições aprendidas 94
6 ANÁLISE CRÍTICA 98
7 CONSIDERAÇÕES FINAIS 101
7.1 Contribuições 102
7.2 Limitações do Trabalho 103
7.3 Estudos Futuros 104
7.4 Publicações 105
REFERÊNCIAS BIBLIOGRÁFICAS 107
APÊNDICE A – Artefato 1: Formulário de *hand-off* do *FTSProc* 110
APÊNDICE B – Artefato 2: Relatório de testes unitários do *FTSProc* 111
APÊNDICE C – Documento de requisitos utilizados no processo experimental 112
APÊNDICE D – Apresentação para a equipe que utilizou o *FTSProc* 118
APÊNDICE E – Apresentação para a equipe *ad hoc* 124
APÊNDICE F – Questionário Inicial 127
APÊNDICE G – Questionário Final 128
APÊNDICE H – Termo de Consentimento 130
APÊNDICE I – Manual da ferramenta de apoio 131

1 INTRODUÇÃO

Atualmente, o processo de globalização está se destacando gerando grande desafio para a área de Engenharia de Software (ES). Hoje em dia, mais projetos estão sendo desenvolvidos em ambientes geograficamente distribuídos, caracterizando o Desenvolvimento Distribuído de Software (DDS). O DDS é caracterizado sempre que um ou mais recursos humanos envolvidos no projeto estiver fisicamente distantes dos demais [AUD07]. Pode-se dizer também que uma equipe global de desenvolvimento de software está tipicamente em países diferentes, porém colaborando em um mesmo projeto de qualquer natureza (criação ou manutenção de software) [LAN08].

Durante a implementação do DDS, surgem diversos desafios de gerenciamento. Dentre estes desafios, diversos autores mostram que a diferença de fuso horário pode ser um fator de extrema relevância [HOL06, HER01, CAR10, TRE06]. Para amenizar este desafio, alguns estudos indicam o uso da estratégia *Follow-the-Sun* (FTS) [CAR09, HOL06, LIN07, SET07, SOL10, KNO07, TRE06]. Contudo, recentes trabalhos mostram que são poucos os casos de sucesso na indústria utilizando esta estratégia [SOL10, CAR11]. Os principais problemas apontados pela literatura estão relacionados às dificuldades de coordenação, sincronização e comunicação, principalmente durante a transferência de trabalho de um centro de desenvolvimento para outro [SET07, SOL10, CAR11].

Dentro desta área de estudo, a literatura mostra que o uso da estratégia FTS apresenta diversos desafios. Entretanto, não apresenta formas para amenizar estes desafios durante as transferências de trabalho. Neste sentido, o foco desta pesquisa concentra-se na transferência de trabalho entre centros de desenvolvimento, durante a fase de desenvolvimento de software em projetos que utilizam a estratégia FTS.

1.1 JUSTIFICATIVA

Conforme a literatura nos mostra, a utilização da estratégia FTS pode apresentar diversos desafios. Entre estes, os principais listados pela literatura são os problemas de comunicação e sincronização de tarefas entre as equipes distribuídas e problemas de coordenação entre os centros de desenvolvimento distribuídos [SET07, SOL10, CAR09]. Grande parte dos trabalhos apresentados na literatura ligados a esta temática visam comparar o uso da estratégia FTS com projetos realizados da forma tradicional, ou seja, projetos co-localizados. Estes trabalhos, geralmente apresentam os problemas que a utilização da estratégia FTS pode apresentar. Entretanto, trabalhos com o objetivo de amenizar estas dificuldades ainda são escassos na literatura, tornando esta pesquisa relevante para esta temática.

Apesar dos diversos desafios para a aplicação da estratégia FTS, esta forma de desenvolver software desperta o interesse da indústria, pois através do uso desta estratégia é possível diminuir o *time-to-market*, aumentando assim a sua produtividade. A estratégia FTS pode ser utilizada em todas as fases do desenvolvimento de software. Entretanto, a utilização da estratégia FTS de uma mesma maneira em todas as fases do ciclo de vida do desenvolvimento do software pode-se apresentar muito complexa e, em alguns casos, até inviabilizar a sua utilização [CAR11]. Segundo [CAR11], os processos para utilizar o FTS em cada fase devem ser diferentes. A utilização do FTS dentro de uma fase particular de forma distinta das demais fases é mais adequada para a esta estratégia, pois as suas características específicas permitem uma estrutura mais controlada para as transferências de trabalho (*hand-offs*) [CAR10, CAR09]. Portanto, esta pesquisa torna-se relevante, pois a criação de um processo para a transferência de trabalho na fase de desenvolvimento pode facilitar o uso desta estratégia nos projetos de software e assim, facilitar a sua utilização em projetos distribuídos de software. Além disto, para a teoria da área, esta pesquisa torna-se importante, pois, devido à escassez de trabalhos publicados neste campo de estudo, a literatura não apresenta a definição de um processo para a transferência de

trabalho em projetos que utilizam esta estratégia, focados na fase de desenvolvimento do ciclo de vida.

Sintetizando, a questão de pesquisa que norteia este estudo é: como transferir trabalho durante a fase de desenvolvimento do ciclo de vida de software em um ambiente de DDS, utilizando estratégia FTS?

1.2 OBJETIVOS

Para responder esta questão de pesquisa, emergem os seguintes objetivos:

- Objetivo Geral:

O objetivo geral deste trabalho é propor um processo de transferência de trabalho (*hand-off*) para a fase de desenvolvimento do ciclo de vida, para projetos que utilizam a estratégia FTS. Para alcançar estes objetivos, emergem os seguintes objetivos específicos:

1. Complementar os estudos da base teórica;
2. Propor um processo preliminar de transferência de trabalho para a fase de desenvolvimento do ciclo de vida;
3. Desenvolver uma ferramenta de apoio ao processo de transferência de trabalho;
4. Escrever artigos científicos decorrentes da pesquisa.

1.3 CONTEXTO

As formas de distribuição das equipes em ambientes de DDS podem variar entre as companhias. Conforme a Figura 1, temos: *Outsourcing* e *Insourcing*

representando as diferentes formas de relacionamento entre empresas e, *Inshore* e *Offshore* representando as distribuições geográficas. A escolha da distribuição a ser utilizada pode variar com base no projeto a ser desenvolvido. Com o objetivo de contextualizar esta pesquisa, abaixo é apresentado uma breve descrição sobre os conceitos relacionados às diferentes distribuições geográficas. O profundo detalhamento destes conceitos está descritos na seção 2.1.

Onshore: este termo representa uma distribuição geográfica mais próxima entre a empresa contratante e a empresa contratada (ou matriz e subsidiária). Esta distribuição acontece em um mesmo país, como por exemplo, podemos ter a matriz na cidade de São Paulo, e a sua subsidiária ou contratada na cidade de Belo Horizonte.

Offshore: este termo representa uma distribuição mais distante. Uma distribuição *offshore* deve obrigatoriamente acontecer entre países distintos. Um exemplo para esta distribuição poderia ser o caso onde temos a matriz na cidade de Londres (Inglaterra), e a sua subsidiária ou contratada na cidade de Bangalore (Índia).

O detalhamento destas diferentes formas de distribuições em ambientes de DDS está descrito na seção 2.1.

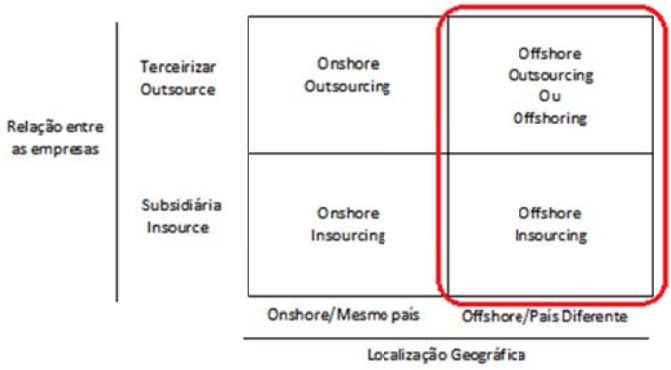


Figura 1. Configurações de DDS, com destaque para *Offshore*, adaptado de [AUD07].

Após esta breve descrição das diferentes formas de distribuição geográfica, o enfoque desta pesquisa está em equipes que utilizam a distribuição do tipo *offshore*, ou seja, companhias localizadas em países distintos. Este fato está relacionado à diferença de fuso horário, necessária para a utilização da estratégia FTS. A Figura 1 apresenta as diferentes formas de distribuições entre as

empresas em um ambiente de DDS e, em destaque, distribuições *offshore*, as quais representam o enfoque desta pesquisa.

1.4 ESTRUTURA

Para alcançar os objetivos citados, este trabalho está estruturado da seguinte forma: a seção dois é destinada a apresentar a *Base Teórica* envolvendo o Desenvolvimento Distribuído de Software (DDS) e a estratégia *Follow-the-Sun (FTS)*; a seção três aborda a metodologia de pesquisa utilizada; a seção quatro apresenta o processo proposto, o qual foi chamado de *FTSProc,* juntamente com a ferramenta de apoio ao processo desenvolvido; a seção cinco descreve o estudo experimental realizado; a seção seis apresenta uma análise crítica dos resultados encontrados; e finalmente, a seção sete expõe as considerações finais, onde são destacados os resultados obtidos nesta pesquisa juntamente com as principais contribuições, limitações e os estudos futuros.

2 BASE TEÓRICA

O Desenvolvimento Distribuído de Software (DDS) tornou-se uma tendência na indústria de software [DAM06, HER01], pois apresenta diversos fatores motivadores que levam as empresas a utilizar cada vez mais este conceito em seus negócios. Porém, o DDS adiciona inúmeros desafios ao processo de desenvolvimento de software. Dentre estes desafios, podemos citar a diferença de fuso horário entre as equipes distribuídas. Entretanto, o fuso horário pode ser encarado como uma vantagem para o projeto. Esta diferença de fuso horário pode ser utilizada para realizar o uso do desenvolvimento FTS [HOL06, LIN07]. Sendo assim, na seção 2.1 é apresentado o desenvolvimento distribuído de software (DDS), juntamente com suas vantagens e os seus desafios. Na seção 2.2 será mostrado como surge o desenvolvimento FTS em ambientes de DDS, suas definições e conceitos encontrados na literatura, juntamente com alguns estudos realizados nesta temática. Finalmente, na sessão 2.3 são apresentados os trabalhos relacionados à temática desta pesquisa.

2.1 DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

No ambiente atual da indústria de software, o processo de globalização está se destacando, gerando um grande desafio para a área da engenharia de software (ES). Na busca de vantagens econômicas na construção de software, empresas estão tornando o desenvolvimento cada vez mais distribuído e global [PRI09]. Hoje em dia, mais projetos estão sendo desenvolvidos em ambientes geograficamente distribuídos, caracterizando o Desenvolvimento Distribuído de Software (DDS), o qual está tornando-se uma tendência na indústria de software [DAM06**,** HER01].

As empresas de software são levadas ao uso de DDS, pois a construção de software torna-se cada vez mais cara e a competitividade da empresa diminui em um ambiente co-localizado [PRI09]. Desta maneira, empresas utilizam o DDS

com o objetivo de reduzir seus custos de produção de software e, portanto, aumentar a sua competitividade no mercado [PRI08].

O DDS não é um conceito novo. Este conceito surgiu nos anos 90, onde as empresas começaram a desenvolver software com equipes distribuídas [LAN08]. O DDS é caracterizado sempre que um ou mais recursos envolvidos no projeto estiverem fisicamente distante dos demais [AUD07]. Quando a distância física entre os elementos das equipes do projeto abrange mais de um país, caracteriza-se o Desenvolvimento Global de Software (GSD, do inglês *Global Software Development*) [LAN08].

As formas de distribuição das equipes em DDS podem variar entre as companhias. A escolha da forma de utilização do DDS deve ser realizada com base no tipo de projeto a ser desenvolvido. Com isso, torna-se relevante a definição de alguns conceitos utilizados para realizar esta caracterização: *Outsourcing* e *Insourcing* (formas de relacionamento entre empresas); *Inshore* e *Offshore* (distribuições geográficas).

Outsourcing: também conhecido como terceirização é a contratação de uma empresa para realizar um serviço. É encontrado no momento em que uma empresa designa uma tarefa a ser realizada por uma empresa contratada [PRI09]. Alguns autores defendem que esta é a forma mais fácil e rápida de implementação do DDS [AUD07, BIN07].

Insourcing: conceito que surgiu em oposição ao *outsourcing* [PIL06]. Este conceito é caracterizado quando as empresas criam os seus próprios centros de desenvolvimento. Dentre os motivos que levam a utilização do *insourcing* está o maior controle sobre os negócios da empresa, a maior flexibilidade e o menor custo em um longo prazo [AUD07]. Este modelo é considerado o que possui maior complexidade e que demanda mais tempo para implementação.

Offshore: consiste em enviar serviços ou projetos para uma companhia localizada em um país diferente da localização da matriz [PIL06]. Pode ser uma empresa contratada ou até mesmo uma subsidiária da matriz [AUD07]. Países componentes do grupo econômico BRIC, estão cada vez mais tornado-se uma alternativa atraente para este tipo de serviço, pois apresentam baixo custo de produção quando comparados com Estados Unidos e Europa. A motivação para

a utilização deste modelo está na redução de custos, capacidade de ampliar e reduzir a equipe conforme a demanda e o acesso a profissionais com habilidades específicas [PIL06]. Esta forma de distribuição é mais indicada para projetos que possuem plano de projeto bem definido e com requisitos de sistemas completamente entendidos [AUD07].

Onshore: ocorre quando a matriz e o cliente estão localizados no mesmo país. Duas situações distintas podem ocorrer neste modelo. Todo o desenvolvimento é realizado na empresa contratada em um local fisicamente distinto da empresa contratante, o que caracteriza o *offsite*. Outra situação é quando o desenvolvimento é realizado fisicamente no mesmo local onde o cliente está, e assim caracterizando o *onsite* [PRI09].

A Figura 2 mostra a combinação dos modelos de negócio mais adotados pelas empresas em relação ao modelo de trabalho entre as empresas e as suas distribuições geográficas.

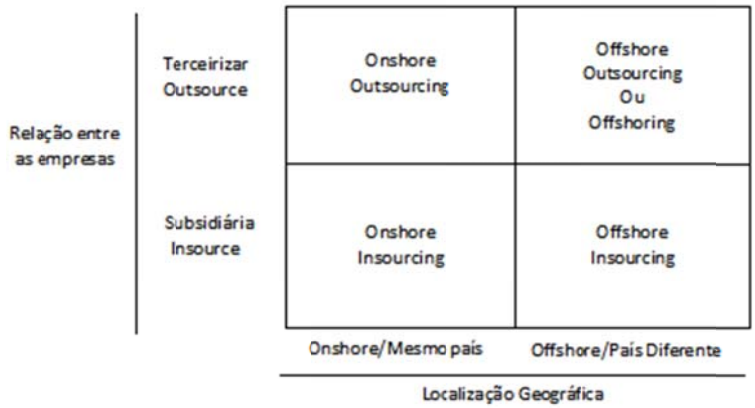


Figura 2. Relacionamento entre empresas, adaptado de [AUD07].

A Figura 2 nos remete a quatro principais modelos de desenvolvimento distribuído, os quais são caracterizados a seguir:

Onshore insourcing: este é o modelo mais simples e conhecido. Ocorre quando uma demanda interna da empresa, onde a partir da necessidade da construção de um software, um departamento dentro da própria empresa, ou até mesmo uma subsidiária é responsável pelo seu desenvolvimento. Como temos o termo *Onshore*, vale destacar que este departamento ou subsidiária deve estar localizado geograficamente no mesmo país.

Onshore outsourcing: é um modelo semelhante ao *onshore insourcing*, a principal diferença é que ao invés de um departamento interno da empresa ou uma subsidiária desenvolver o projeto, neste modelo, o desenvolvimento é realizado por uma empresa terceirizada. Como, novamente, temos o termo *onshore*, deve ficar claro que a companhia que desenvolverá o projeto deve estar localizada fisicamente no mesmo país que a companhia contratante.

Offshore insourcing: este modelo implica na criação de um centro de desenvolvimento da própria empresa. Devido ao uso do termo *offshore*, está implícito que esta subsidiária criada deve estar obrigatoriamente em um país distinto da matriz. Esta subsidiaria é responsável por executar serviços de desenvolvimento de software para a matriz (*insourcing*).

Offshore outsourcing: o termo *outsourcing* nos leva a definir que uma empresa terceira será contratada e *offshore* remete a diferente localização geográfica da contratante. Então, este modelo representa a contratação de uma empresa terceira, localizada em um país diferente para prover serviços de desenvolvimento de software.

O DDS apresenta muitos fatores motivadores que levam as empresas a utilizarem cada vez mais este conceito em seus negócios. A seguir são apresentados alguns destes fatores. Optou-se por apresentar as vantagens mais citadas por diferentes autores.

- Redução de custos [LAN08, DAM06, PRI08, AUD07, MAR09]: as organizações procuram alternativas para que o custo de produção seja cada vez menor. Desta maneira, elas contratam mão de obra de outro país ou localidade onde o custo de produção é mais baixo [KNO07] e assim conseguem reduzir custos;

- Ganho de proximidade com o cliente [LAN08]: como a demanda de software cresce em diversos países [KNO07], uma empresa que esteja expandindo seus negócios para uma nova região poderia iniciar uma operação de DDS nesta localidade e assim, estaria se aproximando dos seus clientes.

- Redução do tempo de projeto [LAN08, DAM06, PRI08]: também chamado de *time-to-market*, incluir um produto no mercado antes do concorrente pode ser vital para o sucesso de uma empresa. Sendo assim, com mais recursos

trabalhando nos projetos, o tempo de desenvolvimento pode ser reduzido. Ainda é possível citar a vantagem que é alcançada através da diferença de fuso horário entre os diferentes centros de desenvolvimento de uma empresa. Esta diferença torna possível a aplicação da estratégia de desenvolvimento de software *Follow-the-Sun (FTS)*, onde a qualquer momento do dia haverá uma equipe trabalhando na construção do software [KNO07].

- Recursos especializados [LAN08] e globais [DAM06, PRI08, AUD07, MAR09]: com o DDS é possível obter a disponibilidade da mão de obra tão especializada quanto em projetos tradicionais, porém, com a vantagem de manter o baixo custo.

Apesar de todas as vantagens que o DDS disponibiliza para as organizações, o processo de desenvolvimento de software continua sendo uma atividade complexa. Utilizando o DDS, adiciona-se ao processo inúmeros problemas, como a distância física, diferenças de fusos horários e diferenças culturais [PRI09, AUD07, DAM06], os quais tornam este tipo de projeto extremamente complexo de ser gerenciado. Portanto, diversas dificuldades são adicionadas ao processo de desenvolvimento de software, e essas novas adversidades devem ser conhecidas e gerenciadas para alcançar o sucesso no DDS. Segundo alguns autores, as adversidades encontradas no DDS podem ser divididas em algumas categorias [AUD07, PRI09, KNO07, PIL06], as quais afetam determinadas áreas do processo. Abaixo estão listadas algumas das dificuldades encontradas no DDS mais citadas por diferentes autores:

Desafios relacionados à *Projetos* [PRI08, HER01] também identificados como problemas técnicos [PIL06, KNO07], contemplam dificuldades relacionadas à forma como o projeto será desenvolvido, incluindo o uso de ferramentas e métodos [PRI09]. Dizem respeito ainda às adversidades relacionadas com a gerência de requisitos, integração, gerência de configuração, dificuldades no acompanhamento do projeto e na utilização de ferramentas [PIL06, KNO07]. Exemplos destas adversidades são [PRI09, LOP04, PIL06, BOS10, HER99]:

- Arquitetura de Software:

Um dos fatores decisivos para o sucesso de um projeto de DDS é a escolha da arquitetura de software (AS) [BOS10]. Uma AS apropriada para este

tipo de projeto deve ser uma arquitetura modular, pois desta forma é possível alocar tarefas complexas de forma distribuída [AUD07, PRI09, HER99].

- Processos de desenvolvimento:

A utilização de um processo de desenvolvimento único no ambiente DDS é fundamental [AUD07]. Com a distribuição de tarefas, a falta de sincronização pode tornar-se um fator decisivo. Para contornar este problema, a utilização de uma metodologia de desenvolvimento de software impõe rigor à equipe. Neste sentido, é possível verificar que cada vez mais companhias buscam excelência na qualidade do desenvolvimento de software. Para comprovar esta busca pela melhoria de processos, submetem-se a diversas avaliações tais como, ISO e CMMI [AUD07, PRI09].

- Telecomunicações:

A coordenação entre equipes distribuídas envolve alto custo. Neste sentido, é fundamental a existência de uma boa infraestrutura de comunicação. É essencial a existência de conexões confiáveis e de alta velocidade para que todas as formas de comunicações possam ser utilizadas. Atualmente a maior parte das localidades já possui esta infraestrutura. A segurança nesta comunicação também é um fator importante. Hoje em dia, existe grande diversidade de opções tecnológicas que garantem comunicações seguras. Ultimamente tem crescido a opção pelo uso de *VPN* (do Inglês, *virtual private network*) devido ao baixo custo, confiabilidade e alta disponibilidade [AUD07].

- Gerência de configuração:

O gerenciamento de configuração de software também apresenta desafios. O controle das modificações dos artefatos em localidades distintas, juntamente com o gerenciamento dos processos de desenvolvimento do sistema pode ser complexo. Para diminuir esta dificuldade é recomendada a utilização de um repositório central, para que o controle seja realizado de forma única [MAR09].

- Gerenciamento de projetos:

De forma a diminuir os impactos inerentes ao DDS, o gerenciamento de projeto deve sofrer adaptações em relação ao modelo tradicional utilizados em

times co-localizados. É recomendada a utilização de técnicas mais formais de gerenciamento [PRI09].

Desafios relacionados à *Pessoas* [PRI09], também identificado como problemas sociais [PIL06, KNO07] dizem respeito aos problemas que afetam diretamente os recursos envolvidos no projeto [PRI09]. Alguns exemplos são [PRI09, PIL06, KNO07, HER01, DAM06, BIN07]:

- Confiança:

Sendo o processo de desenvolvimento de software uma tarefa que geralmente depende da cooperação de vários membros dentro da equipe, a confiança torna-se fundamental. Esta relação é essencial para o bom funcionamento da equipe, resultando em diversos benefícios, como melhora no desempenho, redução de custo e maior sociabilidade entre os membros da equipe [AUD07, PRI09]. Algumas formas de interação podem facilitar o aumento da confiança entre os elementos dos times distribuídos, dentre elas, podemos citar: reuniões de *kick-off* e encontros em determinados *milestones* do projeto [OSH07]. O gráfico da Figura 3 demonstra como o nível de confiança tende a cair com o passar tempo. Por esta razão, torna-se importante encontros no decorrer do projeto.

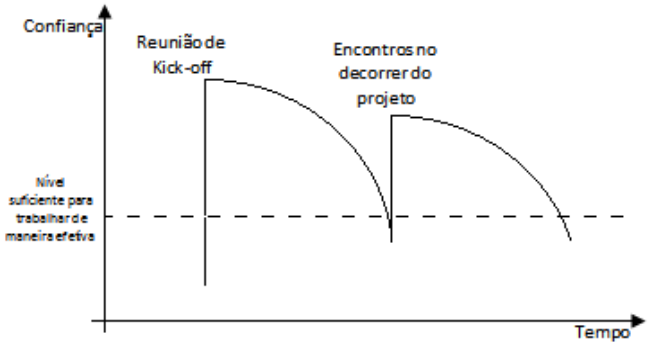


Figura 3. Nível de confiança durante projeto [AUD07].

- Conflitos:

Como em qualquer projeto de desenvolvimento de software, conflitos podem acontecer. Em alguns casos, eles podem ser benéficos para o projeto e podem ser resolvidos no próprio processo de desenvolvimento. Porém, podem acontecer problemas maiores, o que poderia acarretar na perda da produtividade. Neste caso, é interessante que seja definido a figura de um líder no início do

projeto. Este teria o papel de sanar os conflitos, tomando a decisão que julgar mais adequada. Deve ser a autoridade central do projeto [AUD07, PRI09].

- Diferenças culturais:

Diferenças culturais fazem parte do cotidiano no DDS. Com equipes distribuídas ao redor do globo, os problemas relacionados à cultura emergem rapidamente. Diferentes culturas fazem com que pessoas pensem, comuniquem-se e agem de formas distintas [BIN07]. Em um país, algo que possa ser trivial e corriqueiro, em outro local pode ser algo completamente incomum. Neste sentido, é recomendado que as equipes compreendam cada cultura e se adéquem as suas expectativas em relação as diferenças existentes. Algumas companhias utilizam o conceito de *liaison*, que consiste em ter um membro da equipe que age como uma ponte entre culturas diferentes [AUD07, HER99, HOL06, LAN08, LIN07]. Normalmente esta pessoa já conviveu com essa cultura e por isso pode fazer este papel. Por exemplo, um brasileiro trabalhando nos Estados Unidos auxiliando o lado brasileiro da equipe distribuída a compreender comportamentos da equipe americana.

- Liderança:

Questões relacionadas à liderança são tratadas de forma diferente dependendo da cultura. Algumas culturas tratam a liderança com menos rigor do que em outras, ou seja, encorajam a participação do time na tomada de decisões. Enquanto que outras culturas utilizam uma estrutura mais hierárquica e tomam as decisões de maneira autônoma, sem permitir que seus comandados expressem suas opiniões.

Quanto aos problemas relacionados à *Organização* [PRI09, HER01, PIL06, KNO07], mostram dificuldades que afetam as definições estratégicas de processos e métodos de gestão de uma organização [PRI09, PIL06, KNO07]. Alguns exemplos são [PIL06, LOP04, PRI09]:

- Legislação:

Algumas questões importantes dizem respeito às diferenças legais entre os locais envolvidos (principalmente em casos de distribuição global). Essas diferenças implicam em maior complexidade no estabelecimento de contratos,

bem como a necessidade de cautela com questões de sigilo e propriedade intelectual [KAR98].

- Gerenciamento de portfólio de projeto:

Este tipo de gerenciamento tem surgido como um importante aliado no gerenciamento de projetos. O principal objetivo deste tipo de gerenciamento não é executar projetos de desenvolvimento de software corretamente. Seu principal objetivo é realizar os projetos corretos, nos momentos adequados, baseado nas estratégias da empresa. Os desafios desta área estão em determinar, através de diversas análises, quais os projetos de DDS serão desenvolvidos, definir como os projetos serão realizados e para quais localidades será realizada a distribuição dos mesmos [PRI09, PIL06].

- Modelos de negócio:

Conforme pode ser visto na Figura 2 (página 28), as organizações podem optar por diversos modelos de negócio. Não existe somente um modelo correto, mas existe o modelo mais adequado ao problema a ser resolvido. Para tomar a decisão sobre qual modelo será utilizado, as empresas devem realizar uma análise, identificando suas reais necessidades com a utilização do DDS e quais os tipos de projetos que pretendem distribuir geograficamente. A prática demonstra que um bom planejamento tem determinado o sucesso no modelo escolhido [PRI09].

- Diferentes fusos horários:

As diferenças de fusos horários podem afetar os projetos de DDS de diferentes formas. Equipes separadas por uma pequena diferença de horário podem passar a maior parte do dia trabalhando ao mesmo tempo. Neste modo, a comunicação fica mais fácil, pois é possível usar a comunicação síncrona, como ferramentas de mensagens instantâneas, ligações telefônicas, vídeo conferências, etc. Como exemplo, podemos citar uma equipe localizada na cidade de Nova Iorque e outra em São Paulo. Para este caso a diferença seria de apenas uma hora na maior parte do ano.

Os problemas relacionados ao fuso horário começam a se tornar mais evidentes na medida em que a diferença de horário aumenta. Por exemplo, quando essa diferença atinge algo em torno de cinco horas, a possibilidade de

trabalhar juntos na maior parte do tempo já não é mais válida. Então, para este caso, a comunicação síncrona tende a diminuir e a comunicação assíncrona aumenta. Como por exemplo, pode-se citar uma equipe localizada em Nova Iorque e outra em Londres. Para este caso a diferença seria de cinco horas na maior parte do ano.

Finalmente, tem-se o caso em que as equipes distribuídas não trabalham no mesmo horário. Neste caso, quando uma equipe está terminando o seu dia de trabalho a outra está começando. Portanto, além da falta de comunicação síncrona, torna-se vital que a comunicação assíncrona seja realizada de forma clara e efetiva. Qualquer dependência de uma resposta de algum membro do outro time pode demorar no mínimo um dia. Com esta diferença de fuso horário, alguns autores defendem que seja possível utilizar a estratégia *Follow-the-Sun*, ou seja, desenvolvimento 24 horas por dia, sete dias por semana [HOL06, LIN07]. Para este exemplo, pode-se citar uma equipe localizada em São Paulo e outra em Tóquio, onde a diferença será de doze horas na maior parte do ano.

Nota-se que os problemas relacionados ao fuso horário tendem a ficar mais evidentes na medida em que as diferenças de horário aumentam. Para que esta diferença possa ser utilizada como uma vantagem, alguns autores defendem a utilização da estratégia FTS. Porém, devido as suas dificuldades, atualmente este conceito é pouco explorado na indústria [HOL06, LIN07, TRE06].

2.2 DESENVOLVIMENTO FOLLOW-THE-SUN

Devido aos desafios impostos pelo DDS relacionado às diferenças de fusos horários, surge o conceito chamado de desenvolvimento *Follow-the-Sun* (FTS). O desenvolvimento FTS é um subconjunto do desenvolvimento global de software, o qual está principalmente focado na diminuição do tempo de desenvolvimento de um projeto [CAR09, GOR96, GUP09b]. O FTS utiliza a diferença de fuso horário entre as equipes distribuídas como uma vantagem [CAR09, HOL06, LIN07, SET07, SOL10, KNO07, TRE06] para o desenvolvimento do projeto durante as 24 horas do dia. Porém, devido ao fato do FTS ser uma área nova de estudo na engenharia de software, pouco sobre esta temática foi publicado [TRE06].

Conforme o trabalho [SOL10], casos de sucesso na indústria utilizando o FTS ainda são escassos. Em consonância a esta afirmação, o trabalho apresentado com Carmel *et al.* afirma que não há casos de sucesso na indústria [CAR09].

Esta forma de trabalho, em outros ramos da indústria não é uma novidade. Dependendo do tipo de produto a ser criado, existe a necessidade de se trabalhar em um único local, durante as 24 horas do dia, implicando em trabalhos fora do horário convencional. Um exemplo é a indústria automobilística, onde uma equipe termina o seu turno e a próxima assume logo após, para iniciar a sua jornada, mantendo a produção 24 horas por dia, porém, no mesmo local [GOR96]. Este fato poderia afetar a qualidade do produto, pois pessoas ao redor do mundo estão acostumadas a trabalhar durante o dia e descansar durante a noite [CAR09, JAL04].

Na indústria do desenvolvimento de software, a primeira tentativa documentada do uso do desenvolvimento FTS foi protagonizada pela IBM [CAR99]. Em 1997 [SOO08], a IBM decidiu desenvolver um projeto utilizando o desenvolvimento FTS [CAR09]. Para isto, criou cinco equipes distribuídas em cinco centros de desenvolvimento distintos em cinco países diferentes [SOO08]. Durante o projeto, muitas dificuldades de coordenação foram encontradas, principalmente durante as transferências diárias de trabalho [SET07]. Portanto, como o FTS não estava funcionando como planejado, não trazendo o ganho esperado pelos gestores, os responsáveis pelo projeto desistiram de utilizar o FTS para acelerar o processo de desenvolvimento, mantendo apenas o desenvolvimento global de software [CAR09].

Para melhor entender do desenvolvimento FTS, nas próximas seções este tema será abordado em maior profundidade. Para isto, na seção 2.2.1 serão mostrados os conceitos encontrados na literatura, na seção 2.2.2 será exposta uma breve descrição sobre os estudos publicados relacionados à estratégia FTS.

2.2.1 Follow-the-Sun: Conceitos

Devido ao fato do FTS ser um assunto recente na literatura e na indústria do desenvolvimento de software, existem poucos trabalhos publicados neste

campo de pesquisa [TRE06]. Neste sentido, é possível observar a existência de diversas formas de definir o desenvolvimento FTS, também chamado como desenvolvimento 24-horas [GUP09a, GOR96], desenvolvimento *round-the-clock* [CAR09], desenvolvimento *around-the-clock* [YAP05] e *software shift work* [GOR96]. Diversos autores possuem suas próprias definições para este tema, sendo assim, na literatura desta área, não há um consenso na forma de definir o desenvolvimento FTS. A seguir são apresentadas definições de diferentes autores e ao final desta seção, é apresentada uma reflexão sobre estas definições, com o objetivo de oferecer uma definição onde sintetizaremos as principais ideias encontradas na literatura.

Segundo *Visser* [VIS09], o desenvolvimento FTS pode ser determinado ao ter equipes de desenvolvimento de *software* distribuídas por múltiplos fusos horários, onde ao final do dia de trabalho, uma equipe deve entregar as informações relevantes do trabalho realizado até o momento, juntamente com o código fonte do produto para a próxima equipe, a qual está iniciando a sua jornada. O trabalho deve ser continuado do ponto onde a equipe anterior parou. Desta forma, o projeto estará sendo desenvolvido 24 horas por dia e não apenas durante oito horas, como normalmente acontece.

Outra forma de identificar esta maneira de desenvolver software é apresentada por *Gorton* e *Motwani* [GOR96], onde é proposta uma nova denominação: *software shift work*. Os autores mostram que a indústria tem a necessidade de criar seus produtos em um tempo cada vez menor, porém, dependendo do tipo de produto a ser criado, existe a necessidade de trabalhar em um único local, durante as 24 horas do dia (implicando no desenvolvimento de tarefas fora do horário convencional de trabalho), como exemplo, o autor cita a indústria automobilística. Porém, a criação de software difere radicalmente, pois o produto final é uma coleção de arquivos binários, documentação, código-fonte e executáveis, onde utilizando redes modernas e rápidas de comunicação de dados como as que existem atualmente na grande parte dos países onde o desenvolvimento global de software é utilizado, estes artefatos podem rapidamente ser transferidos para qualquer lugar do mundo. Portanto, o desenvolvimento 24 horas de software pode ser alcançado apenas utilizando a diferença de fuso horário entre os locais onde os times estão alocados. Cada time

trabalha no seu horário convencional e, ao final do seu turno (um dia de trabalho), o próximo time, o qual está iniciando o seu turno assume o trabalho de onde a equipe anterior parou.

Já para *Gupta, Mattarelli, Seshasai* e *Broschak* [GUP09b] o desenvolvimento FTS pode ser estendido para diferentes tarefas, além do desenvolvimento. Os autores mostram o conceito de *fábrica de conhecimento*, onde as tarefas a serem desenvolvidas em um projeto podem ser voltadas ao conhecimento, o qual é passado entre os times globalmente distribuídos ao final de cada dia. Para exemplificar este conceito de desenvolvimento contínuo durante 24 horas por dia, é apresentado o ciclo de testes de um software, onde o conhecimento é o próprio software em desenvolvimento, e o conhecimento está onde o software atende os requisitos de forma satisfatória ou não. Este conhecimento é passado entre os times de desenvolvimento e de testes, os quais estão distribuídos em locais distintos. Desta forma, a fase de testes pode ser finalizada de forma mais rápida, o que segundo os autores, é um dos principais benefícios ao distribuir equipes de desenvolvimento em fusos horários distintos e utilizar o desenvolvimento FTS.

Treinen e *Miller-Frost* [TRE06] definem o FTS de uma forma mais sucinta. Para estes autores, a diferença de fuso horário deve ser vista como uma vantagem para assim, distribuir equipes com o objetivo de criar um ambiente de desenvolvimento de software onde as equipes trabalham apenas durante as suas horas normais de trabalho, e ao final do dia, apenas redistribuem as suas tarefas ao time que está iniciando a sua jornada, criando assim, efetivamente o desenvolvimento 24 horas.

Carmel, Dubinsky e *Espinosa* [CAR09], propõem uma definição para o *follow-the-sun* aonde quatro ideias básicas devem ser seguidas:

i. Equipes de desenvolvimento devem estar localizadas a diversos fusos-horário de diferença;
ii. Um dos principais objetivos é reduzir o tempo de desenvolvimento/ *time-to-market*;
iii. A cada momento existe somente uma equipe que possui a posse do produto;

iv. A transferência de tarefas deve ser realizada diariamente, onde esta é definida por um *check-in* de uma unidade de trabalho da qual a próxima equipe depende para dar continuidade ao trabalho.

Após apresentar estas quatro ideias básicas, o trabalho de [CAR09] apresenta a seguinte caracterização para o desenvolvimento FTS: um tipo de fluxo de trabalho de conhecimento global, criado com o objetivo de reduzir a duração de um projeto, onde o produto é de propriedade de um centro de desenvolvimento até ser entregue ao próximo. Esta entrega deve ser diária, e a próxima equipe deve estar localizada diversos fusos horários a frente para dar continuidade ao trabalho.

Após a apresentação das diferentes definições por diferentes autores, a Tabela 1 mostra uma análise comparativa entre estas diferentes abordagens. Para cada critério (colunas), procurou-se saber se a definição apresentada, de alguma forma, indicou importância para este fator.

Os fatores utilizados para a análise comparativa apresentada na Tabela 1 surgiram de uma avaliação prévia dos conceitos apresentados pelos autores. Procurou-se identificar os principais fatores de cada conceito encontrado na literatura, os quais poderiam ser comparados entre as definições.

Tabela 1. Comparação entre as definições apresentadas.

Autor	Objetivo principal do FTS			Transferências diárias de tarefas		Considera relevante	
	Aumentar velocidade de desenvolvimento do projeto	Diminuir *time to market*	Aumentar tempo diário de desenvolvimento	Código fonte (apenas codificação)	Múltiplas tarefas	Trabalhar somente nos horários convencionais	Diversos fusos horários de diferença
Visser [VIS09].			X	X	X		X
Gorton e Motwani [GOR96].	X	X		X	X	X	
Gupta, Mattarelli, Seshasai e Broschak [GUP09b].	X		X		X		
Treinen e Miller-Frost [TRE06].				X	X	X	
Carmel, Dubinsky e Espinosa [CAR09].		X		X		X	X

Como podemos perceber, não há uma convergência para apenas uma definição para o desenvolvimento FTS. Podemos identificar que apenas um autor

define o FTS somente para a codificação, enquanto que os autores restantes defendem a utilização para qualquer tarefa dentro do projeto. Outra característica nas definições está relacionada ao objetivo principal da utilização do FTS. Apesar de expresso de maneira distinta pelos autores, podemos perceber que o objetivo principal do FTS é a redução do tempo necessário para desenvolver um projeto. A diferença mínima necessária de fusos horários entre as equipes distribuídas foi pouco citada. Também existem poucos trabalhos que consideram relevante a importância de ter equipes distintas trabalhando no seu horário habitual, evitando horas extras durante a noite. Este fator poderia ser mais considerado, pois a maior parte das pessoas ao redor do mundo está acostumada a trabalhar durante o dia e descansar durante a noite [CAR09].

Portanto, após esta análise comparativa das diferentes definições sugeridas pelos autores, foram identificados pontos importantes em todas elas. Assim, através destas informações coletadas, propõe-se uma definição para o desenvolvimento FTS, a qual sintetiza as principais ideias e pode ser descrita da seguinte forma:

O *Follow-the-Sun (FTS)* é um tipo de desenvolvimento global de software onde o principal objetivo é a diminuição do *time-to-market,* acelerando a construção do produto final desde a concepção até a sua distribuição. Este ambiente opera com equipes distribuídas em fusos horários e países distintos, onde cada equipe detém o trabalho por determinado período, até que o mesmo seja transmitido para a próxima equipe que inicia a sua jornada. A transferência pode ser para qualquer tipo de tarefa relacionada com o desenvolvimento do projeto de software. Esta transferência deve acontecer diariamente e de forma padronizada.

Durante o desenvolvimento desta pesquisa, esta caracterização foi tema de um artigo aceito para o *International Conference on Global Software Engeneering (ICGSE), 2011.*

2.2.2 Estudos em Follow-the-Sun

A literatura apresenta poucos trabalhos relacionados ao desenvolvimento FTS [TRE06]. Após uma extensa pesquisa, encontrou-se um número reduzido de

artigos que realizam estudos teóricos nesta área. Esta seção apresenta uma compilação dos principais estudos relacionados à esta temática. Os estudos apresentados versam sobre diferentes óticas para avaliar as vantagens da utilização do desenvolvimento FTS em comparação à forma tradicional de desenvolvimento de software. Logo após, na seção 2.3, são apresentados os trabalhos relacionados ao tema desta pesquisa.

2.2.2.1 Estudo apresentado por Setamanit, Wakeland e Raffo (2007)

Apresentado por *Setamanit, Wakeland* e *Raffo* [SET07], o estudo mostra as vantagens ao utilizar o FTS em relação a um projeto realizado em um único local. O objetivo é identificar quando há uma vantagem ao utilizar o FTS, e quais são os requisitos para alcançar uma vantagem que seja interessante para um projeto desenvolvido desta forma.

O estudo inicia descrevendo a forma como as equipes devem realizar o desenvolvimento do projeto. Para esta fase, duas equipes distintas foram criadas. Uma delas opera em apenas uma localidade, sem fazer o uso do desenvolvimento FTS. A outra equipe é dividida em duas localidades com fuso horário distinto, e faz o uso do desenvolvimento FTS. A Figura 4 ilustra o formato das equipes.

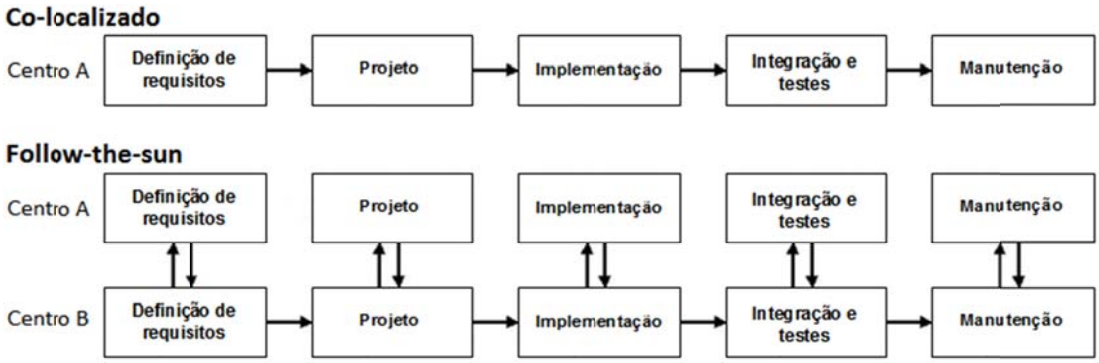


Figura 4. Disposição inicial das equipes, adaptado de [SET07].

Neste primeiro cenário, após realizar 30 testes para cada configuração (co-localizado e FTS), os autores identificaram que a utilização do desenvolvimento FTS não gerou o ganho esperado. Os resultados mostraram um aumento de 50% no tempo necessário para o desenvolvimento do projeto, foi notado também um

aumento de 70% no esforço, e a qualidade foi pior, pois a quantidade de defeitos foi maior que o dobro.

Após adquirir estes dados iniciais sobre o desempenho das equipes, os autores buscaram identificar fatores que teriam um forte impacto para a melhoria destas medidas. Estes fatores, segundo os autores, podem direcionar os projetos para a utilização de práticas onde os benefícios podem ser alcançados com maior facilidade. Após a identificação destes fatores, os autores mostram que os principais resultados foram:

- Diminuir o esforço: aumentando o tempo de trabalho simultâneo (também conhecido como *overlap*) entre equipes distribuídas resulta em um menor esforço necessário. Como na primeira configuração, as equipes não tinham a possibilidade de trabalhar ao mesmo tempo, a comunicação síncrona não existia. Ao introduzir a possibilidade deste tipo de comunicação, foi possível notar um aumento na produtividade. Entre outras vantagens, este tipo de comunicação melhora a confiança entre os membros da equipe, resultando, assim na diminuição do esforço.

- Diminuir duração do projeto: aumentar o *overlap* entre as equipes distribuídas pode aumentar a duração do projeto, pois o tempo de desenvolvimento diário diminui. Porém, ao notar que, além de ter um período maior de trabalho simultâneo entre equipes distribuídas, outros fatores já citados (produtividade, confiança), ajudam a diminuir o esforço, resultando assim em um menor tempo de desenvolvimento.

- Diminuição do número de defeitos: tendo um aumento na efetividade da comunicação síncrona entre as equipes através do trabalho simultâneo, diminuíram os problemas de entendimento das comunicações assíncronas. Isto resultou em um número menor de defeitos encontrados.

Após identificar os fatores que poderiam melhorar o desempenho do projeto, realizou-se o experimento novamente, porém, desta vez com um *overlap* de horas de trabalho entre as equipes distribuídas. O resultado encontrado apresentou uma pequena melhora em relação à primeira execução, porém, o desempenho do FTS ainda mostrou-se pior que o desenvolvimento em um único local.

Na tentativa de alcançar uma melhora significativa, os autores incluíram mais um centro de desenvolvimento, obtendo assim, a distribuição em três locais distintos para o desenvolvimento. Neste modelo, o tempo de duração foi menor (11,1%). Entretanto o esforço foi significativamente maior (45,4%). Porém, segundo os autores, este fator não se mostra relevante, devido ao fato de os centros de desenvolvimento distribuídos normalmente estarem em países que possuem baixo custo. A qualidade continuou inferior neste modelo, devido à dificuldade de comunicação e coordenação entre os 3 centros de desenvolvimento.

Finalmente, os autores afirmam que o FTS pode ser uma boa estratégia se dentre as necessidades do projeto está à diminuição do tempo de desenvolvimento. Porém, salientam que se esta decisão for tomada, deve-se utilizar a abordagem com no mínimo três centros distribuídos de forma a obter o desenvolvimento 24 horas por dia, e tendo um período de trabalho simultâneo para comunicação síncrona entre as equipes distribuídas.

2.2.2.2 Estudo apresentado por Carmel, Dubinsky e Espinosa (2009)

O recente estudo apresentado por *Carmel, Dubinsky* e *Espinosa* [CAR09] relata um *quasi-experimento* comparando duas formas distintas de desenvolver um projeto de software. O objetivo deste estudo foi medir o ganho no tempo de desenvolvimento de um projeto de software, o qual, para este tipo de projeto, teoricamente deveria ser de 50%.

O experimento controlado inicia com a definição de duas equipes de desenvolvimento de software, uma delas denominada *controle* (CO), composta por 7 integrantes e a outra equipe, denominada de FTS, composta por 8 participantes. As duas equipes deveriam implementar os mesmos requisitos de um sistema. Todos os participantes do estudo eram estudantes de Ciência da Computação ou Engenharia Elétrica de uma grande universidade. O tempo de desenvolvimento por recurso era o mesmo para ambos os times. Definiu-se que o tempo de desenvolvimento do projeto seria 5 semanas. Porém, como a equipe FTS faria o uso do desenvolvimento FTS, teoricamente, o tempo necessário para

o projeto deveria ser de duas semanas e meia. A Figura 5 ilustra o tempo de projeto definido para cada equipe.

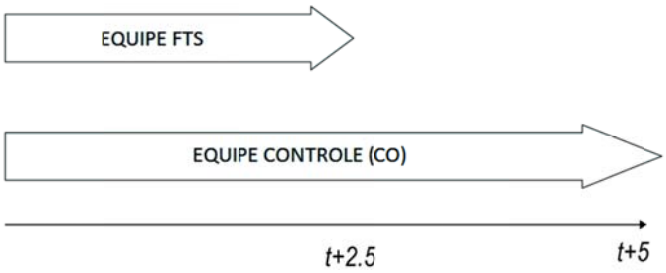


Figura 5. Duração do experimento, adaptado de [CAR09]

Para a equipe CO, nenhuma restrição foi imposta, ou seja, o software foi desenvolvido da forma tradicional, com todos os recursos da equipe trabalhando no mesmo local. Para a equipe FTS, a primeira etapa foi dividir a equipe em duas, *SubTeamA*, composta por 3 recursos, e *SubTeamB*, integrada por 5 recursos. Cada uma destas duas equipes, oriundas do grupo FTS, deveria trabalhar em horários diferentes, para simular a diferença de fuso horário: *SubTeamA* entre 21:30 e 11:00 e *SubTeamB* entre 11:30 e 21:00, mantendo 30 minutos para alguma hora extra, quando necessário. Para garantir que nenhuma comunicação síncrona entre as equipes *SubTeamA* e *SubTeamB* ocorressem, mecanismos como mensagens *SMS*, comunicador instantâneo, e qualquer outro tipo de comunicação síncrona foi proibido. Para controlar os horários de trabalho, cada uma das duas equipes do FTS só poderia alterar códigos no repositório nos horários determinados. Este controle foi realizado de forma automática pelo sistema de repositório de código utilizado no projeto.

Ao final do estudo, os resultados apresentados pelos autores ficaram muito aquém do esperado no início do estudo. Houve um ganho de apenas 10% no tempo de duração do projeto, ao invés dos 50% que teoricamente deveria ser alcançado. Analisando as estatísticas geradas pelo sistema de repositório de código utilizado, uma constatação foi importante para justificar a pouca redução do tempo de desenvolvimento deste experimento. A equipe FTS realmente terminou o projeto num tempo menor que a equipe CO. Porém, devido a falta de uma diretriz sobre o momento de parar a codificação, mesmo tendo terminado a codificação das funcionalidades propostas, continuou-se apenas aprimorando-as. Foi possível identificar que na última semana do projeto a equipe FTS fez *check-*

ins apenas sobre arquivos que já estavam no repositório, ou seja, atualizaram arquivos, melhorando funcionalidades prontas. Enquanto isso, a equipe CO realizava a criação das funcionalidades. Cabe lembrar, que segundo os autores, ambas as equipes concluíram a codificação dos requisitos de forma satisfatória. Os autores argumentam que apesar de terem identificado um ganho de apenas 10%, este número poderia ser maior, se tivessem definido o momento para terminar a fase de codificação para a equipe FTS. Este ponto foi identificado como uma limitação do estudo.

2.2.2.3 Estudo apresentado por Solingen e Valkema (2010)

No estudo apresentado por *Solingen* e *Valkema* [SOL10], os autores apresentaram algumas hipóteses e através de um experimento controlado, apontaram a validade ou não das hipóteses propostas. O estudo foca na investigação do impacto do aumento do número de centros distribuídos na velocidade de desenvolvimento de projetos que utilizam o FTS. Procura-se saber qual o número ideal de equipes para se trabalhar em um projeto FTS.

O experimento inicia definindo a configuração das equipes, as quais foram definidas da seguinte forma: cada execução do experimento seria realizada utilizando um número de centros de desenvolvimento diferente entre 2 e 4. Estes centros foram assim configurados: sempre existe um centro responsável por prover os requisitos e os *feedbacks* do trabalho realizado, então, na configuração com duas equipes, existirá uma equipe de desenvolvimento e uma de requisitos, para três equipes, da mesma forma, haverá uma de requisitos enquanto as outras duas para o desenvolvimento e assim sucessivamente.

Devido ao fato de ser um experimento controlado, as diferenças de fusos horários foram simuladas, criando horários de trabalho pré-definidos entre as diferentes equipes. Cada ciclo diário consiste de 2 a 4 *shifts* trabalhados (um para cada equipe distinta), dependendo do número de centros adotados. Os dias de trabalhos são contínuos, quando um dia termina (para um determinado centro) outro está começando, simulando assim fusos horários distintos. Ao final de um dia, o trabalho é repassado ao próximo time que está iniciando a sua jornada.

O experimento foi realizado com estudantes de Ciência da Computação. A cada dia, três execuções foram realizadas, utilizando 4, 3 e 2 centros de desenvolvimentos. Ao final de cada execução, retirou-se uma equipe aleatoriamente. Ao final de cada dia, a primeira equipe (responsável por requisitos e *feedbacks)* relatava para o primeiro time de desenvolvimento as informações necessárias para iniciar o próximo ciclo. As tarefas a serem realizados eram extremamente simples, sendo possível assim, simular um dia de trabalho em apenas quatro minutos. Com esta configuração, segundo os autores, foi possível analisar o efeito do número de centros distribuídos em um projeto FTS relacionado com o tempo de desenvolvimento.

O estudo discute os resultados obtidos da seguinte forma: apresenta-se a hipótese, e a validade ou não da mesma. A seguir estão relacionadas às hipóteses propostas pelos autores, juntamente com o resultado obtido em relação às mesmas.

- Quantidade de trabalho entregue:

Htotal: total4 > total3 > total2

Com os dados coletados, foi possível identificar que a quantidade de trabalho entregue realmente aumenta à medida que a quantidade de centros de desenvolvimento aumenta, concluindo assim, favoravelmente a esta hipótese.

- Velocidade média de trabalho por centro:

Havgspd: avgspd4 < avgspd3 < avgspd2

Os dados mostram que a adição de centros de desenvolvimento aumenta a velocidade de trabalho por volta de 20% na média por centro de desenvolvimento. Portanto, concluindo assim favoravelmente a está hipótese.

- Exatidão do trabalho realizado:

Haccur: accur4 < accur3 < accur2

Devido a pequena diferença encontrada nas três variações, nada se pôde afirmar sobre está hipótese.

- Percepção de velocidade:

Hperspd: perspd4 < perspd3 < perspd2

Segundo os dados obtidos, houve apenas uma diminuição desta percepção na configuração com 4 centros. Portanto, segundo a conclusão dos autores, esta hipótese não foi confirmada.

- Percepção de exatidão do trabalho:

Hperacc: peracc4< peracc3<peracc2

Mantendo 2 centros, a percepção de exatidão mostra-se normal, porém, quando esta configuração possui mais de 2 centros distintos trabalhando, esta percepção reduz drasticamente. Nesta situação, os participantes encontram mais dificuldade para medir a exatidão do trabalho em desenvolvimento entre todos os centros, confirmando assim a hipótese.

Ao final do estudo, os autores discutem os resultados obtidos e mostram que, apesar das dificuldades de coordenação e comunicação que o desenvolvimento FTS impõe, dependendo do que se espera de um projeto de software, a sua utilização pode ser benéfica. A principal vantagem que o FTS pode alcançar, segundo os autores, está na diminuição do tempo de projeto, o que está em consonância com os outros estudos apresentados no capítulo 2.2.2.

2.2.2.4 Análise comparativa

Após a apresentação dos estudos analisados, a Tabela 2 mostra uma análise comparativa entre os diferentes trabalhos. Para cada critério (colunas), procurou-se saber se o experimento relacionado, de alguma forma indicou importância para este fator.

Os fatores utilizados para a análise comparativa apresentada na Tabela 2 surgiram de uma avaliação prévia dos estudos apresentados. Procurou-se identificar os principais fatores de cada estudo, os quais poderiam ser utilizados para critério de comparação entre os estudos.

Tabela 2. Comparação entre os experimentos apresentados.

Experimentos Apresentados	Mede		Atinge		Sugere	
	Diminuição do tempo	Vantagem do número de centros	Algum Ganho	Ganho Esperado	Número mínimo de centros	Utilizar FTS
Setamanit, Wakeland e *Raffo* [SET07]	X	X	X		X	X
Carmel, Dubinsky e *Espinosa* [CAR09]	X		X			X
Solingen e *Valkema* [SOL10]	X	X	X		X	X

A primeira constatação que podemos perceber em relação aos estudos apresentados está relacionada às métricas utilizadas. Percebe-se que todos os estudos medem a diminuição do tempo de projeto através do uso do FTS. [CAR09] e [SET07] identificam esta métrica de uma maneira semelhante, comparando um projeto executado de forma tradicional, co-localizada em um único centro, com um projeto executado de forma distribuída, utilizando o FTS. [CAR09] executa o experimento uma única vez, alcançando um resultado inferior ao esperado, enquanto [SET07], ao obter um resultado aquém do esperado, investiga os problemas enfrentados e refaz o experimento, para tentar melhorar o ganho no tempo de projeto, alcançando um pequeno ganho. Já [SOL10] faz o experimento utilizando diversas distribuições com 2, 3 e 4 equipes, buscando avaliar se há uma melhora no ganho do tempo de projeto ao adicionar novos times.

A métrica relacionada ao número de centros necessários para obter vantagem na utilização do FTS foi utilizada por [SET07] e [SOL10]. Estes estudos fazem uma comparação do ganho no tempo de desenvolvimento adicionando mais de dois centros de desenvolvimento, obtendo resultados satisfatórios. Consequentemente, estes trabalhos sugerem um número mínimo de centros de desenvolvimento para aumentar as chances de sucesso na utilização do desenvolvimento FTS. Para [SET07], a vantagem no tempo de desenvolvimento só é atingida quando existem, pelo menos, três centros distribuídos. Enquanto que [SOL10] defende que o ganho aumenta conforme o número de centros

distribuídos aumenta (este estudo utilizou distribuição entre dois e quatro centros para chegar a esta conclusão).

O resultado mais importante apresentado nos estudos avaliados está relacionado ao ganho que o FTS pode prover. Percebemos que houve realmente um ganho em relação ao tempo de desenvolvimento em todos os trabalhos apresentados. Porém, este ganho ficou muito aquém do ganho teórico esperado pelo desenvolvimento FTS. Segundo [CAR09], este ganho ficou em torno de 10%, já para [SET07], o ganho foi de 11,1%, porém, apenas quando um terceiro centro de desenvolvimento foi adicionado ao projeto, pois utilizando apenas dois, o tempo para desenvolver o projeto, foi 50% maior. [SOL10] mostra apenas que a quantidade de trabalho entregue é maior se aumentarmos o número de centros de desenvolvimento distribuídos, obtendo melhor resultado com quatro centros distribuídos (número máximo utilizado).

Portanto, segundo os resultados apresentados, o desenvolvimento FTS ainda não produz o ganho teórico esperado, principalmente devido as dificuldades impostas, porém pode-se obter algum ganho no tempo de projeto. Percebe-se que todos os estudos indicam a utilização do desenvolvimento FTS. Porém, devido às dificuldades de coordenação, comunicação e sincronização de tarefas neste tipo de projeto, principalmente durante a transição de tarefas, a utilização desta forma de desenvolvimento deve ser utilizada, segundo os estudos, apenas quando há a necessidade de desenvolver o produto em um tempo menor, ou seja, diminuir o *time-to-market* [SET07, SOL10, CAR09]. As conclusões destes estudos vão ao encontro com a literatura desta área, a qual afirma que a principal vantagem do uso do desenvolvimento FTS está na diminuição do tempo de desenvolvimento de um projeto [CAR09, GUP09b, GOR96, VIS09].

2.3 TRABALHOS RELACIONADOS

Como o objetivo deste trabalho é propor um processo de transferência de trabalho (*hand-off*) para a fase de desenvolvimento do ciclo de vida, buscou-se na literatura por trabalhos com este foco. A literatura ainda carece de trabalhos relacionados à estratégia FTS. As publicações que apresentam formas para a

utilização da estratégia FTS ainda são escassas. Entretanto, alguns trabalhos que versam com uma temática semelhante a este estudo, ou seja, formas para atenuar adversidades durante transferência de trabalho estão descritos a seguir.

O trabalho proposto por Fadel *et al.* [FAD00], mostra formas para acelerar o tempo de desenvolvimento de um projeto. Para alcançar tal objetivo, os autores distribuíram equipes através de diferentes fusos horários, e fizeram o uso da estratégia FTS. Criou-se um processo de transferência de trabalho de um *site* para outro. Este processo consistiu em alocar trinta minutos (da equipe terminando o turno de trabalho e a que está começando) para preparar as informações a serem utilizadas durante o *hand-off*. Neste momento de trabalho simultâneo, todos os artefatos criados são entregues, assim como qualquer informação relevante para dar continuidade ao trabalho. Esta comunicação é realizada de forma síncrona, utilizando recursos de telefonia. Logo após, a equipe que iniciou o dia de trabalho, realiza um *brainstorm*, onde o estado atual do trabalho é discutido. Baseado no trabalho que ainda deve ser realizado, as tarefas são alocadas para todos os recursos dentro da equipe. Chegando ao final do dia, este processo é repetido. Este ciclo encerra-se no momento em que os requisitos estão todos alcançados.

O trabalho publicado por Taweel *et al.* [TAW02] apresenta os resultados de um experimento para avaliar a viabilidade o uso de um processo sequencial e colaborativo de engenharia de software para ambientes distribuídos em diferentes fusos horários. Neste experimento foi desenvolvida uma calculadora com funções simples. O projeto foi dividido em 3 fases: *set-up*, onde foram apresentados todos os requisitos para todas as equipes, juntamente com a distribuição do trabalho e o prazo para conclusão; *execução*, onde ocorreu a implementação usando os times distribuídos; *finalização*, onde os dados do experimento foram coletados. O processo colaborativo avaliado estava baseado no envio de *e-mails* entre as equipes com o *status* atual do projeto, contendo todas as informações relativas ao trabalho realizado. O trabalho mostra que, apesar de tratar apenas de tarefas simples, os resultados demonstram a viabilidade deste tipo de processo.

O trabalho apresentado por Denny *et al.* [DEN08] apresenta o conceito de *Composite Personae* (CP). Este conceito mostra como equipes distribuídas podem trabalhar como um único time virtual. Para que isto aconteça, é importante

ter uma equipe coesa, espalhada por diferentes fusos horários. Desta forma, o trabalho pode ser passado de um *site* para outro, e o mesmo é continuado. Todo o trabalho está baseado em *hand-off*, onde uma equipe termina o seu dia de trabalho e a outra inicia. Porém, alguns problemas podem ocorrer nesta transferência. Para tanto, este trabalho mostra uma forma simples de transição de trabalho. Esta transição está baseada em reuniões de *stand-up*, oriundas do *Scrum*. Ao se aproximar do final de um dia de trabalho, os desenvolvedores devem adicionar os seus resultados no repositório de código, e preencher um formulário automatizado, chamado de ferramenta de *hand-off*. Neste formulário deve-se responder às três perguntas básicas de uma reunião *stand-up*:

i. Quais tarefas foram realizadas desde a última reunião?

ii. O que está planejando realizar até a próxima reunião?

iii. Existe algum problema impedindo você de realizar seu objetivo?

Após preencher estas informações, o trabalho é considerado entregue para a equipe seguinte. O próximo *site* inicia o dia de trabalho coletando as informações disponibilizadas pelo *site* anterior e definindo o que deve ser realizado, utilizando como principal referência, as respostas para as perguntas *i, ii* e *iii.* Este trabalho destaca a importância de ter equipes equivalentes em todos os *sites* distribuídos. Esta equivalência não está relacionada ao número de recursos em cada *site*, mas em capacidade de entrega e de solução de problemas.

Denny et al. [DEN09] apresentam um processo de transferência de conhecimento, criado especialmente para o uso do conceito de fábrica de conhecimento em ambientes distribuídos. Este processo foi criado com base no *Personal Software Process* (PSP) [HUM95]. Este processo é desenvolvido para facilitar a transferência de conhecimento de uma equipe para outra ao final de cada dia (*hand-off*). Esta pesquisa apresenta ainda algumas formas de facilitar o entendimento do trabalho entre as equipes distribuídas. Uma destas formas é através da técnica de *Test-Diven Development* (TDD). Segundo os autores, TDD indica a utilização de testes unitários automatizados para redução de defeitos e controle de qualidade. Nesta técnica, os casos de teste são escritos de forma a validar se todos os requisitos estão implementados de forma correta. Os casos de

teste tornam-se um registro documentado da compreensão do requisito e da solução encontrada para atender o mesmo [DEN09].

2.3.1 Análise comparativa

Após a apresentação dos trabalhos relacionados, a Tabela 3 expõe uma análise comparativa entre os mesmos. Para cada critério (colunas), procurou-se saber se o trabalho relacionado, de alguma forma indicou importância para este fator.

Os fatores utilizados para a análise comparativa apresentada na Tabela 3 surgiram de uma análise prévia dos trabalhos apresentados. Procurou-se identificar os principais fatores de cada estudo, os quais poderiam ser utilizados para critério de comparação entre os estudos e como base para o *FTSProc*.

Tabela 3. Comparação entre os trabalhos relacionados.

Trabalho Relacionado	Transfere o trabalho		Planejamento e distribuição de tarefas		Transfere as informações			Conclui o *Hand-off*	
	Finalizado	Inacabado	Inicio do Projeto	Diário	Telefone	*e-mail*	Formulário padronizado	*Check-in*	Informações (form./e-mail/tel.)
Fadel [FAD00]		X		X	X			X	X
Taweel [TAW02]	X		X			X			X
Denny [DEN08]		X		X			X	X	X
Denny [DEN09]		X		X			X	X	X

Através da comparação apresentada na Tabela 3, podemos identificar que não há uma convergência para a forma como um processo deste tipo deve ser. Entretanto, podemos identificar características que podem ser utilizadas para um processo que está voltado exclusivamente para a fase de desenvolvido.

O primeiro ponto que podemos identificar através da comparação apresentada na Tabela 3 é que o planejamento diário é um ponto importante, presente na maioria dos trabalhos. Este fato é relevante, pois no início de cada dia, antes de iniciar um dia de trabalho uma equipe não sabe o ponto onde o

trabalho parou. Desta forma, o planejamento diário torna-se importante para continuar o trabalho do ponto em que parou. Outro fator que mostra a importância do planejamento diário é que a maioria dos trabalhos faz a transferência de trabalho inacabado, ou seja, o trabalho continua do ponto em que parou pelo próximo centro de desenvolvimento. Apenas o trabalho apresentado por [TAW02] não faz esta transferência, pois as tarefas são realizadas de forma paralela, ou seja, cada centro de desenvolvimento trabalha em uma funcionalidade distinta, sendo assim, não é necessária a transferência de trabalho inacabado.

Nenhum dos trabalhos relacionados aponta o uso de um processo em uma fase específica do ciclo de desenvolvimento de software. Entretanto, de acordo com [CAR11], não é possível utilizar a estratégia FTS de uma única forma em todas as fases do ciclo de desenvolvimento de software. Por esta razão, o processo proposto nesta pesquisa foca especificamente na fase de desenvolvimento.

A maior parte dos trabalhos relacionados aponta que a transferência está concluída no momento em que o *check-in* da unidade de trabalho é realizado, juntamente com as informações relevantes ao *shift* que está terminando. Estas informações são passadas através de um formulário padronizado, *e-mails* ou telefone, no caso de comunicação síncrona entre as equipes. Devido ao fato desta pesquisa estar focada em projetos distribuídos de forma global, para fazer o uso da diferença de fuso horário, a utilização de comunicação síncrona não pode ser utilizada, pois não há um período de trabalho simultâneo. Desta forma, os trabalhos relacionados nos mostram que a utilização de um formulário padronizado, onde cada recurso sabe a informação que deve ser adicionada, pode facilitar o uso do processo. Neste sentido, o trabalho proposto por [DEN08] mostra que um formulário de *hand-off* padronizado, utilizando as 3 questões das reuniões de *stand-up* (oriundas da metodologia *Scrum*) podem ser úteis, pois são simples, rápidas de serem preenchidas e contém todas as informações necessárias. Finalmente, outra sugestão, proposta pelo trabalho de [DEN09] mostra que a utilização de *Test-driven development* (TDD) também pode ser efetiva neste tipo de processo, pois facilita no entendimento das funcionalidades e nas validações de suas implementações.

A principal diferença entre os trabalhos relacionados e o trabalho desenvolvido está no momento e na forma como a transferência de trabalho deve ser realizada. O processo proposto está focado exclusivamente na fase de desenvolvimento do ciclo de vida de desenvolvimento do software, pois de acordo com [CAR11], não é possível utilizar a estratégia FTS de uma única forma em todas as fases do ciclo de desenvolvimento de software. Enquanto que os trabalhos relacionados não estão focados em uma única fase.

Outra diferença importante é com o trabalho proposto por [TAW02] onde as tarefas que cada centro distribuído irá desenvolver são definidas *a priori*, ao invés de tratar o time inteiro como um único time virtual. Desta forma, o centro de desenvolvimento que inicia o trabalho não continua o trabalho do ponto onde o centro anterior parou, mas apenas desenvolvem diferentes funcionalidades de forma paralela. Após esta análise da base teórica, o capítulo 4 apresenta o processo criado, denominado *FTSProc*.

3 METODOLOGIA DE PESQUISA

Uma pesquisa do tipo exploratória deve ser utilizada quando o objetivo é examinar um tema de pesquisa pouco estudado ou que não tenha sido estudado sob uma mesma abordagem, anteriormente, na literatura. Após um aprofundamento da base teórica, descrito no capítulo 2, nota-se que ainda há uma escassez de trabalhos abordando esta mesma temática sob a perspectiva adotada nesta pesquisa: um processo focado especificamente na fase de desenvolvimento do *Software Development Life Cycle (*SDLC). Portanto, esta pesquisa pode ser classificada como exploratória.

A seguir, a seção 3.1 apresenta o desenho de pesquisa utilizado durante este trabalho. Logo após, na seção 3.2 é exposto o principal método de pesquisa utilizado, juntamente com as razões para esta escolha.

3.1 DESENHO DE PESQUISA

Para alcançar os objetivos desta pesquisa, a Figura 6 apresenta o desenho da pesquisa desenvolvida no decorrer deste trabalho.

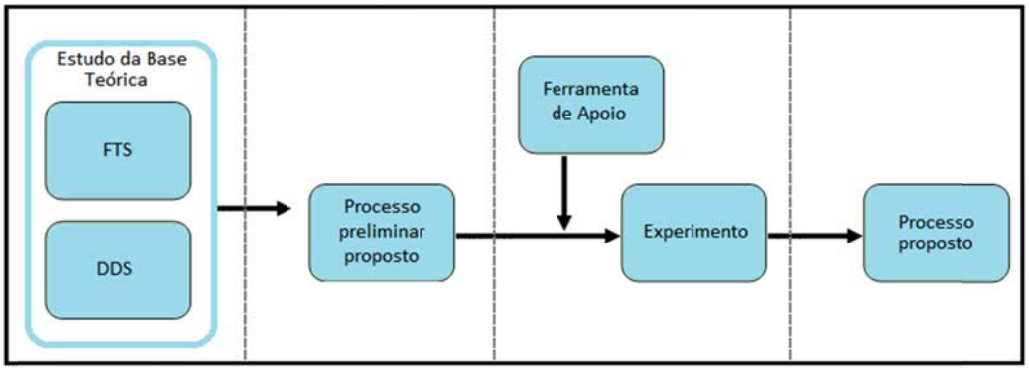


Figura 6. Desenho de pesquisa.

Os objetivos das quatro etapas principais que constituíram este trabalho são descritas a seguir:

Estudo da base teórica: o objetivo desta etapa foi aprofundar os conhecimentos obtidos durante os trabalhos de Introdução à Pesquisa I e

Introdução à Pesquisa II relacionados às duas temáticas principais desta pesquisa: DDS e FTS. Os resultados desta fase foram os insumos para a criação do processo. Os resultados desta fase estão descritos no capítulo 2.

Processo preliminar: consistiu em criar um processo preliminar de transferência de tarefas, baseado nos dados oriundos do aprofundamento dos estudos da base teórica. O processo criado foi chamado de *FTSProc* e está descrito no capítulo 4.

Experimento e ferramenta de apoio: esta fase foi responsável pela criação de uma ferramenta de apoio, a qual controla todo o processo proposto. Após o desenvolvimento desta ferramenta, a mesma foi utilizada durante o experimento. A especificação e os detalhes da implementação desta ferramenta de apoio estão descritos na seção 4.1.

Esta etapa também foi responsável pela realização de um experimento para avaliar os benefícios do processo preliminar criado e identificar pontos de melhorias para este processo. Durante o planejamento e condução do experimento realizado, utilizou-se como guia os estudos de [WOH00] e [TRA02], os quais estruturam experimentos nas seguintes etapas: definição, planejamento, operação, análise e interpretação dos resultados e apresentação. Os detalhes sobre o desenvolvimento de cada uma destas etapas estão descritos no capítulo 5.

Processo proposto: baseado nos dados provenientes do experimento, nesta etapa seriam realizadas as melhorias necessárias, e uma versão final do processo de transferência de tarefas seria elaborada. Entretanto, conforme exposto no capítulo 6, devido aos resultados obtidos no experimento não houve alterações no processo preliminar criado.

3.2 MÉTODO DE PESQUISA

Esta pesquisa utilizou o experimento controlado como principal método de pesquisa, pois é crucial a avaliação desta nova proposta, sendo o experimento uma forma de realizar esta avaliação [WOH00]. A principal ideia deste método de

pesquisa é obter conclusões a partir do teste de hipóteses, o qual estabelece uma relação de causa-efeito.

Antigamente, a investigação científica estava focada na observação do universo, sem que nada pudesse ser alterado. Atualmente, uma das formas de investigação é baseada na experimentação, ou seja, na observação dos fenômenos provocados para fins de investigação e na medição das variáveis envolvidas no fenômeno.

Atualmente, na ES, experimentos são atividades caracterizadas pela manipulação de algumas variáveis e a observação de outras, em ambientes controlados. Experimentos também envolvem controle, onde o pesquisador pode decidir que grupo de pessoas realizará cada atividade dentro do experimento. Isto difere de pesquisas observacionais, onde o pesquisador não tem controle sobre o grupo de pessoas. Normalmente, experimentos são realizados em laboratório, pois permite alto grau de controle. Um dos usos mais comuns de experimentos na ES é para testar teorias [JUR01]. Visto que esta pesquisa visa investigar a aplicação de um novo processo de transferência de trabalho em ambientes distribuídos que utilizam a estratégia FTS e, devido às características deste método, optou-se pela realização de um experimento. Esta escolha deve-se ao fato deste método possibilitar a avaliação de novas propostas, permitir o controle do ambiente de execução e proporcionar a obtenção de conclusões a partir de uma hipótese [WOH00].

Existem duas abordagens [JUR01] para realizar estudos empíricos: pesquisa quantitativa e pesquisa qualitativa. Pesquisa quantitativa tem como objetivo obter números provenientes do objeto de estudo, como por exemplo, o percentual de aumento de produtividade no desenvolvimento de um software em um ambiente distribuído que utiliza a estratégia FTS, ao fazer o uso de um novo processo de transferência de trabalho em comparação ao mesmo software sendo desenvolvido sem este processo. Nesse caso, o percentual obtido fornece o que a pesquisa quantitativa se propõe a fazer, ou seja, fatos em forma de números para evidenciar o que antes era uma apenas uma ideia.

A pesquisa qualitativa não foca somente em números, mas também em explicar fatores de qualidade associados ao uso de alguma técnica, ferramenta ou

processo. Um exemplo para esta abordagem poderia ser demonstrar por que os desenvolvedores tem mais produtividade ao utilizar um processo de transferência de trabalho. Nesse caso, sabe-se, *a priori*, que os desenvolvedores são mais produtivos utilizando o processo de transferência de trabalho. Como resultado, a pesquisa qualitativa apresenta explicações em forma de texto, gráficos, imagens, etc. [JUR01].

Como *a priori* não havia evidências que a utilização do processo proposto resultaria em um ganho de produtividade, o principal método de pesquisa utilizado foi o experimento com abordagem quantitativa. Entretanto, devido ao número de participantes não permitir o uso de uma análise estatística, complementou-se o estudo com uma análise qualitativa para alcançar os resultados do experimento. O capítulo 5 apresenta a estrutura utilizada durante a execução do processo experimental.

4 *FTSProc*: PROCESSO PROPOSTO

O aprofundamento da base teórica, juntamente com uma análise dos trabalhos relacionados, ambos apresentados no capítulo 2, serviram como insumos para a criação do processo de transferência de trabalho. Este processo proposto visa atenuar os desafios de coordenação, sincronização e comunicação durante a transferência de trabalho na fase de desenvolvimento do ciclo de vida de software. Neste sentido, os principais objetivos do processo são:

a. Ao iniciar um dia de trabalho (*shift*), uma equipe deve, de forma simples, ter a percepção do que deve ser desenvolvido e o trabalho já realizado pelo centro de desenvolvimento anterior.
b. Evitar a necessidade de comunicação síncrona entre as equipes distribuídas.
c. Garantir que a transferência de trabalho de um centro de desenvolvimento para outro ocorra sem problemas e que o trabalho possa ser continuado do ponto onde o centro anterior parou.

Os processos desta natureza ainda são insuficientes na literatura. Entretanto, alguns trabalhos mostram que este tipo de processo deve ser “leve” [DEN09, TAW02], ou seja, não pode acarretar um grande aumento na carga de trabalho (*overhead*) em um dia típico de trabalho.

O processo proposto foi chamado de *FTSProc*. Este processo utiliza como base os conceitos chamados de *Composite Persona* (CP) [DEN08] e *24hr Design and Development* [FAD00]. Estes trabalhos estão detalhadamente descritos na seção 2.3 juntamente com os trabalhos relacionados. Ainda, o processo criado utiliza algumas técnicas provenientes das metodologias ágeis, como por exemplo, *Test-driven development* (TDD), integração contínua e as perguntas que guiam as reuniões de *stand-up*, oriundas de *Scrum*.

Para a adequada utilização do *FTSProc*, alguns pré-requisitos são necessários. Estes requisitos iniciam com a necessidade de uma boa infraestrutura de comunicação entre todos os centros de desenvolvimentos envolvidos no projeto. Uma rede de comunicação de alta velocidade é essencial para que grandes volumes de dados sejam transferidos entre os centros de

desenvolvimentos. Além desta infraestrutura, é importante que o processo seja utilizado em projetos que possuam centros de desenvolvimento com uma grande diferença de fuso horário entre eles. Esta diferença deve ser o suficiente para que não haja um período de trabalho simultâneo. Normalmente oito horas de diferença são suficientes. Finalmente, o projeto deve ser realizado utilizando a metodologia cascata, também chamado de ciclo de vida clássico [PRE10] ou, do inglês, *waterfall.* Apresar de utilizar diversas práticas oriundas das metodologias ágeis, este processo deve ser utilizado em projetos que utilizam a metodologia cascata, pois o *FTSProc* é utilizado apenas na fase de desenvolvimento.

Como descrito anteriormente, o *FTSProc* atua apenas durante a fase de desenvolvimento do SDLC, pois segundo [CAR10], a utilização dentro de uma fase específica é mais adequada para esta estratégia, pois as suas características específicas permitem uma estrutura mais controlada para os ciclos de transferência de trabalho (*hand-offs*). Entretanto, conforme descrito nos dois itens abaixo, algumas pré-condições das fases anteriores (definição de requisitos e projeto) do SDLC auxiliam para o funcionamento do processo:

1. Fase de definição dos requisitos: esta fase tem como artefato de saída a documentação contendo os requisitos do sistema a serem desenvolvidos. Para o bom funcionamento do processo é importante que os requisitos sejam definidos da forma mais específica possível [GUP09a], preferencialmente utilizando o conceito de *User Stories* [HAU06], as quais dividem os mesmos em pequenas funcionalidades para diminuir a complexidade das tarefas [FAD00, DEN08], que normalmente são desenvolvidas em um único dia de trabalho. É importante que as *User Stories* tenham os critérios de aceitação bem definidos. Isto facilita o entendimento dos requisitos e, conforme apresentado no trabalho de [TAW02], neste tipo de projeto é essencial que toda a equipe de desenvolvimento tenha o total entendimento sobre os requisitos que serão desenvolvidos durante o projeto.

2. Fase de projeto: esta fase, também chamada de *design* ou análise, produzirá os artefatos que estão relacionados diretamente com a maneira

como as funcionalidades serão implementadas. Os diagramas necessários para o entendimento do sistema, como os diagramas de classes e de atividades são alguns exemplos dos resultados desta fase. Além disto, baseado nos critérios de aceitação de cada requisito, oriundos da fase anterior, testes unitários devem ser criados, para fazer o uso da técnica de *Test-driven development* (TDD). Portanto, cada requisito a ser desenvolvido, ao final desta fase, irá gerar diversos testes unitários. A utilização de TDD ainda está relacionada ao fato de manter um registro documentado da compreensão do requisito e da solução encontrada para atender o mesmo [DEN09, GUP09a]. Ainda, segundo [MEZ06], antes de iniciar a implementação, o TDD pode atuar como parte da especificação e, depois de construída a aplicação, o TDD torna-se o conhecimento sobre como a aplicação foi desenvolvida.

Conforme podemos identificar no diagrama de atividades apresentado a seguir, na Figura 7, a execução do processo inicia somente quando temos a definição dos requisitos e dos testes de aceitação finalizados. Enquanto estas atividades não estiverem concluídas, o processo não está apto a iniciar. Identificando a conclusão destas etapas, é iniciada a fase de desenvolvimento, sobre a qual o processo será utilizado.

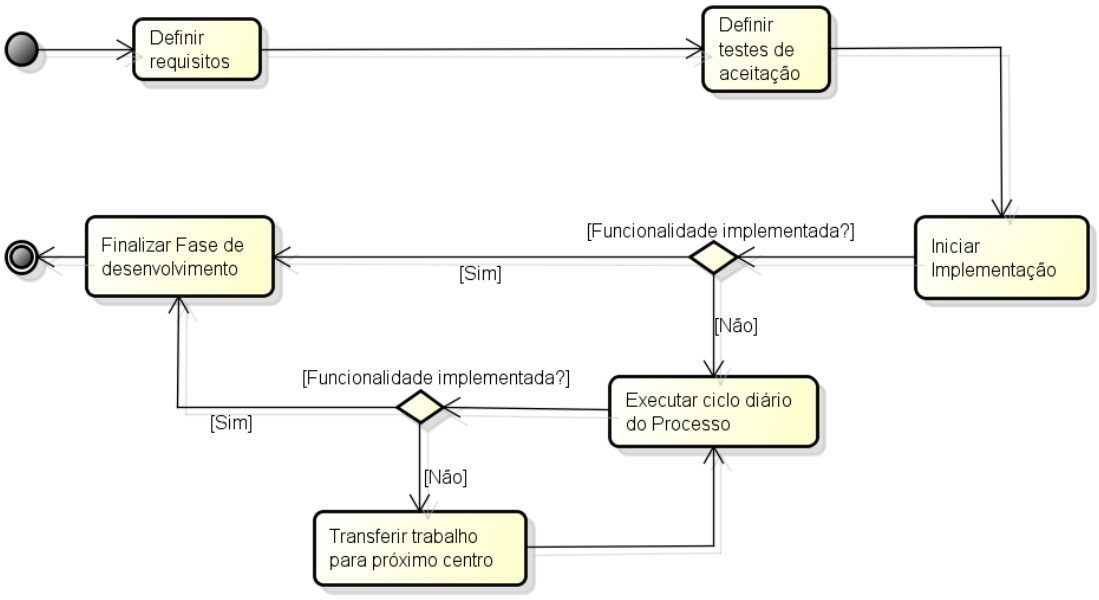


Figura 7. Diagrama de atividades do processo proposto.

De acordo com a Figura 7, o processo possui um ciclo, o qual é executado a cada dia de trabalho de cada centro de desenvolvimento, sempre verificando se a funcionalidade está totalmente implementada (requisitos definidos). Ao final de cada dia, se ainda houver trabalho a ser realizado, este é transferido para o próximo centro de desenvolvimento.

Além destas pré-condições, durante cada ciclo diário serão utilizados os seguintes artefatos, desenvolvidos para o *FTSProc*:

- Artefato 1 - Formulário de *hand-off* (Apêndice A):

  Representa o estado atual do trabalho e deve ser preenchido com informações sobre o trabalho desenvolvido. Todas as informações necessárias estão contidas neste artefato. Estas informações são necessárias para que o trabalho do próximo *shift* possa ser iniciado do ponto onde a equipe anterior parou. Um exemplo deste artefato está no Apêndice A.

- Artefato 2 - Relatório de testes unitários (Apêndice B):

  Todos os testes unitários que não estão cobertos, ou seja, os quais ainda não têm os requisitos atendidos, devem estar neste relatório. A importância deste artefato está em auxiliar o planejamento de um dia de trabalho (*shift*), conforme pode ser visto no passo 2 do *FTSProc*. Um exemplo deste artefato está no Apêndice B.

Conforme ilustrado na Figura 8, a fase de desenvolvimento inicia quando temos a fase de definição de requisitos e a fase de projeto concluídas. Portanto, os passos 1, 2, 3, 4 e 5, representam um único dia de trabalho de uma equipe de desenvolvimento, ou seja, um ciclo diário do processo. Estes passos representam o detalhamento do estado denominado “Executar ciclo diário do processo” apresentado anteriormente, na Figura 7.

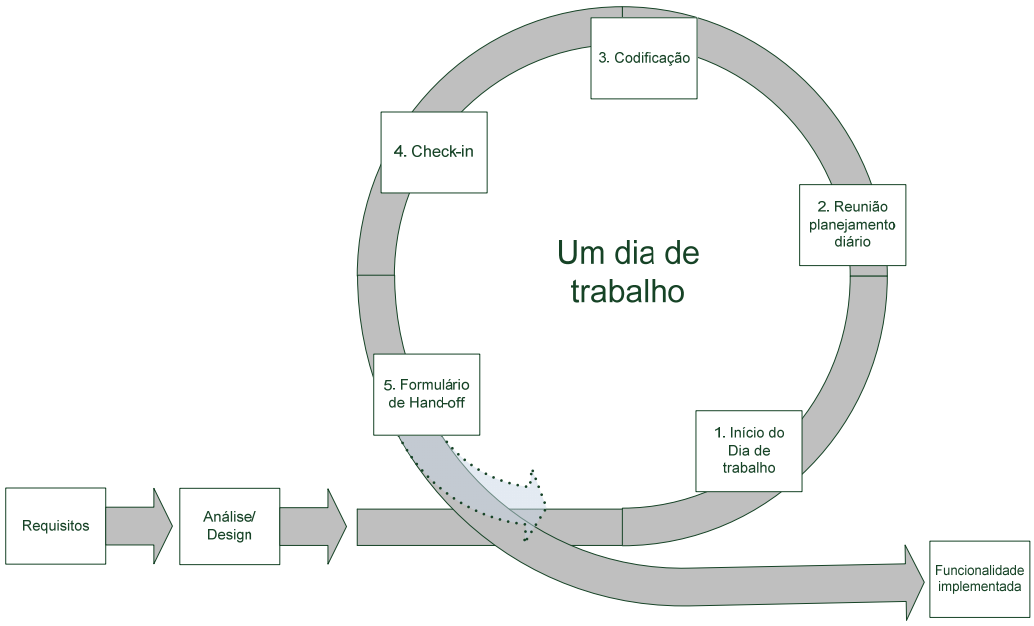


Figura 8. *FTSProc*: Processo Proposto.

O *FTSProc* é um processo iterativo e estes cinco estados serão repetidos a cada dia de trabalho, para cada time de desenvolvimento distribuído. Os dados da saída de cada estado são utilizados para iniciar o estado imediatamente seguinte. Estes cinco estados que compreendem um ciclo diário de trabalho estão detalhadamente descritos a seguir:

1. Início do dia de trabalho:

Este estado marca o início de um dia de trabalho (*shift*) de uma equipe. O diagrama apresentado na Figura 9 apresenta as principais etapas que compõem este estado:

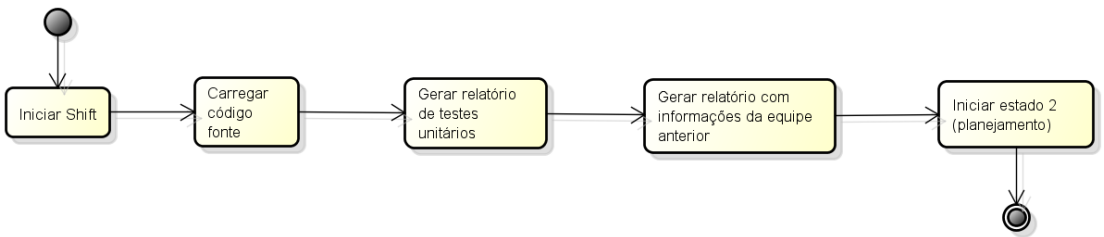


Figura 9. Detalhes do estado 1 do *FTSProc*.

Conforme apresentado na Figura 9, é possível observar que este estado do processo é composto por três etapas principais, conforme descrito a seguir:

i. Carregar código fonte: o centro de desenvolvimento que está iniciando o seu *shift* deve carregar a versão mais recente do código-fonte disponível no repositório. Desta forma, é garantido que o trabalho será iniciado sobre o código-fonte mais recente disponibilizado pela equipe anterior.

ii. Gerar relatório de testes unitários: a equipe que está iniciando um dia de trabalho deve gerar um relatório com os testes que já estão e os que ainda não estão aceitos. Ou seja, se o teste unitário está "passando", significa que aquele critério de aceitação já está coberto e não é necessário trabalhar no mesmo. Caso contrário, ainda será necessário aplicar esforço neste requisito. Este relatório é representado pelo Artefato 2 do *FTSProc*.

iii. Gerar relatório com informações da equipe anterior: além do relatório de testes unitários, cada equipe que está iniciando um dia de trabalho deve gerar um relatório com todas as informações disponibilizadas pela equipe que trabalhou no *shift* anterior. Este relatório está baseado no formato das reuniões de *stand-up*, oriundas da metodologia *Scrum* [DEN08, GUP09a]. Conforme [SCH04], as reuniões de *stand-up* acontecem a cada dia, e o seu objetivo principal é identificar o andamento do projeto. Durante esta reunião, cada membro da equipe responde a três perguntas:

- O que você fez desde a última reunião?
- O que você está planejando fazer até a próxima reunião?
- Existe algo impedindo você de continuar o seu trabalho?

Cada um dos desenvolvedores que trabalhou no *shift* anterior deve preencher este formulário (Artefato A), respondendo perguntas baseadas neste formato (estado 5 do *FTSProc*). Portanto, este relatório é composto pelo conjunto das informações passadas por todos os desenvolvedores.

Após estas três etapas principais serem concluídas, tem-se todas as informações necessárias para iniciar o planejamento diário para o *shift* que está iniciando. Deste modo, o fluxo de trabalho pode ser continuado no próximo estado do *FTSPRoc,* o qual está descrito a seguir.

2. Reunião de planejamento diário – *brainstorm*:

O diagrama apresentado na Figura 10 apresenta as etapas que compõe este estado:

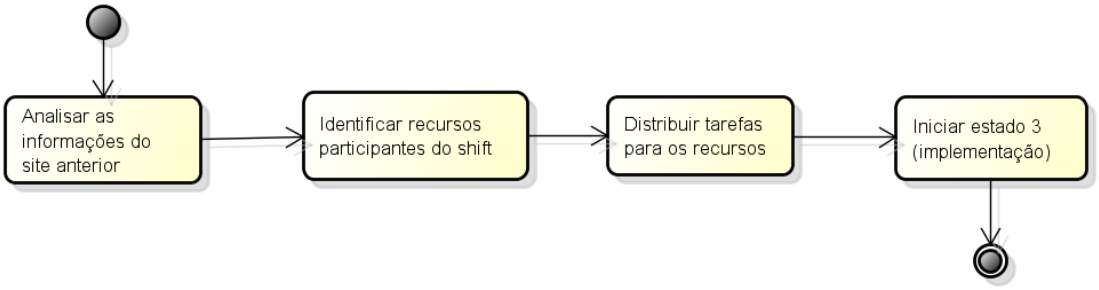


Figura 10. Detalhes do estado 2 do *FTSProc*.

Este é iniciado imediatamente ao final do estado 1 (início do dia de trabalho). Os resultados obtidos no estado 1 (relatórios) serão utilizados neste estado. Conforme apresentado na Figura 10, temos três passos principais para que este estado seja concluído. Estes passos são:

i. Analisar as informações do site anterior: ao final do estado 1 (Início do dia de trabalho) todas as informações fornecidas pelo *shift* anterior estão disponíveis através de relatórios. Portanto, o primeiro passo deste estado é analisar estas informações que o site anterior disponibilizou. Estas informações são compostas pelo relatório com as informações relativas ao *shift* anterior (Artefato 1), assim como o resultado dos testes unitários fornecidos (Artefato 2) [DEN08, FAD00, DEN09]. Um modelo do relatório de testes unitários é apresentado no Apêndice B.

ii. Identificar os recursos participantes do *shift*: a equipe que está iniciando um *shift* deve identificar todos os recursos que participarão deste *shift* para que o planejamento diário possa ser realizado.

iii. Distribuir tarefas para os recursos: utilizando as informações disponibilizadas pelo site anterior a equipe que inicia o seu dia de trabalho deve reunir-se e fazer a distribuição das tarefas para os recursos indicados que trabalharão no *shift* que está sendo iniciado (planejamento diário).

Depois de concluído este planejamento, este estado do *FTSProc* é finalizado. Neste ponto, todos os desenvolvedores envolvidos no *shift* que está iniciando possuem o conhecimento do ponto onde a equipe anterior parou e como

o trabalho deve ser continuado. Portanto, neste ponto os desenvolvedores podem iniciar o trabalho de implementação das funcionalidades, continuando um trabalho que estava sendo realizado no *shift* anterior, ou iniciando o desenvolvimento de uma nova funcionalidade. Com o trabalho de implementação pronto para ser iniciado, este estado está finalizado, e o próximo estado do *FTSProc* pode ser executado, conforme descrito a seguir.

3. Início da implementação:

Esta etapa apenas marca a implementação dos requisitos, seguindo as definições acordadas durante o planejamento (estado 2). Nesta etapa é iniciado ou continuado o trabalho de implementação das funcionalidades. Cada desenvolvedor é individualmente responsável por esta etapa, ou seja, o término deste estado não depende de todos os envolvidos. É o estado do ciclo diário do processo mais longo, pois é onde o desenvolvimento do projeto é realizado. Na perspectiva do *FTSProc*, o principal objetivo deste estado é fazer o controle do fluxo de trabalho, ou seja, com este estado, é possível controlar em qual etapa do projeto cada desenvolvedor está em um determinado momento. Quando cada desenvolvedor chegar ao final deste estado do processo, significa que o seu dia de trabalho está acabando e deve ser realizado o *check-in* do código-fonte, para isto, cada desenvolvedor deve iniciar o próximo estado, o qual está descrito a seguir.

4. *Check-in*:

Após finalizar a implementação, cada membro da equipe deve completar o *check-in* do trabalho realizado durante o dia, disponibilizando todas as informações necessárias para a próxima equipe continuar o trabalho do ponto onde parou. O controle de *check-in* é importante, pois evita que um recurso termine o seu dia de trabalho sem disponibilizar o código mais recente no repositório. Desta forma, evita-se iniciar um dia de trabalho sem o código mais recente disponível no repositório. Por esta razão, o *FTSProc* faz este controle. Após realizar o *check-in* e garantir que o código fonte mais recente está no repositório, este estado está finalizado. Neste ponto, cada recurso pode adicionar as informações relacionadas ao trabalho desenvolvido durante o *shift* além do código-fonte. Estas informações serão disponibilizadas no último passo do processo, conforme descrito a seguir.

5. Formulário de *hand-off*:

Chegando ao final do dia de trabalho, cada membro da equipe deve reservar um tempo para preencher o formulário de *hand-off* (Artefato 1 do *FTSProc*, apresentado no Apêndice A), com todas as informações necessárias para o próximo site. Este formulário foi adaptado do formato de reuniões *stand-up*, oriundas do *Scrum* e será utilizado para formalizar a transferência de trabalho. As seguintes questões abaixo devem ser respondidas por cada desenvolvedor envolvido no *shift*, as quais foram adaptadas do trabalho de [DEN08]:

i. O que foi realizado durante este período de trabalho?
ii. O que deve ser continuado no próximo período de trabalho?
iii. Existe algo bloqueando a equipe?
iv. Quais testes unitários foram finalizados durante este *shift*?

Este estado marca o final de um dia de trabalho. Após todos os recursos envolvidos no *shift* finalizarem esta etapa, novos critérios de aceitação estão cobertos pelo trabalho desenvolvido durante o *shift* que está terminando. O código-fonte mais recente está no repositório e a documentação necessária para a próxima equipe que iniciará o trabalho está disponível. Uma vez realizada todas as atividades indicadas no processo, a transferência de trabalho está concluída [CAR09, CAR10, TAW02, FAD00]. O diagrama de atividades representado na Figura 11 ilustra as etapas que compõem este último estado.

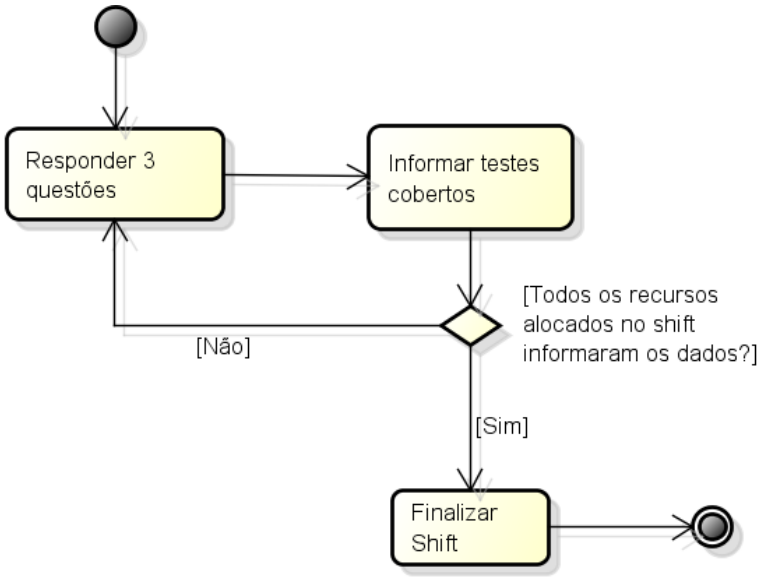


Figura 11. Detalhes do estado 5 do *FTSProc*.

Com o *shift* finalizado, o próximo centro de desenvolvimento pode iniciar o seu trabalho. Para isto, este irá repetir as cinco etapas do processo. Estas cinco etapas são repetidas em todos os dias de trabalhos por cada centro de desenvolvimento até que todos os critérios de aceitação estejam atendidos, ou seja, todos os testes criados durante a fase de projeto (TDD) para cada requisito estão cobertos, pois cada requisito gera diversos testes unitários. Após atender todos os critérios de aceitação encerra-se a fase de desenvolvimento e temos a funcionalidade implementada [FAD00, GUP09a].

4.1 FERRAMENTA DE APOIO AO *FTSPROC*

Em função do processo proposto ser inédito, foi necessário o desenvolvimento de uma ferramenta de apoio. A ferramenta desenvolvida realiza todo o controle necessário para a execução do *FTSProc*. Este controle inicia com a verificação dos recursos participantes do projeto. Estes recursos preenchem os seguintes papéis na ferramenta:

Administrador: responsável pela manutenção dos usuários da ferramenta e controle (adição, atualização e remoção) dos centros de desenvolvimentos do projeto.

Gerente global: responsável global do projeto. Suas responsabilidades estão voltadas ao acompanhamento do projeto. Para isto, a ferramenta disponibiliza diversos relatórios que apresentam o andamento do projeto. Dentre estes relatórios temos: quantidade de testes coberto por cada centro de desenvolvimento, quantidade de recursos em cada centro de desenvolvimento, informações passadas em cada *hand-off*, etc.

Gerente de Projeto: em cada centro existe um gerente de projeto. Este é responsável pela inicialização dos *shifts.* Durante esta inicialização, o gerente de projeto pode designar as tarefas que cada um dos desenvolvedores irá trabalhar. Após iniciado o *shift*, os desenvolvedores podem iniciar o seu trabalho. O gerente de Projeto também pode ser considerado um desenvolvedor.

Desenvolvedor: responsável pelo desenvolvimento do projeto. Durante o seu dia de trabalho, deve informar a ferramenta em qual passo do *FTSProc* se encontra. Desta forma, a ferramenta garante que todos os passos definidos no processo são efetivamente realizados, ou seja, controla o fluxo de trabalho.

É importante ressaltar que o *FTSProc* não define papéis. Entretanto, a ferramenta criada fez este uso para facilitar qual tipo de informação cada usuário pode ter acesso. Outro ponto a ser destacado é o fato que a ferramenta exige apenas um Administrador e um gerente de Projeto para realizar um projeto. O Administrador é necessário apenas para configurar a ferramenta, e o Gerente de Projeto é necessário para iniciar o *shift*, e posteriormente, realizar o desenvolvimento, já que um Gerente de Projeto também é um Desenvolvedor.

Além do controle de usuários, a ferramenta faz o controle do fluxo de trabalho de cada um dos desenvolvedores que atuam em cada centro de desenvolvimento durante os *shifts*. A ferramenta faz ainda, o controle dos centros de desenvolvimento, verificando qual será o próximo a iniciar o *shift.* Para atender todos estes requisitos, a Figura 12 apresenta o diagrama de casos de uso da ferramenta desenvolvida, seguida de uma descrição detalhada dos principais casos de uso. Finalmente uma breve descrição dos outros casos de uso é mostrada, juntamente com algumas imagens do funcionamento da ferramenta.

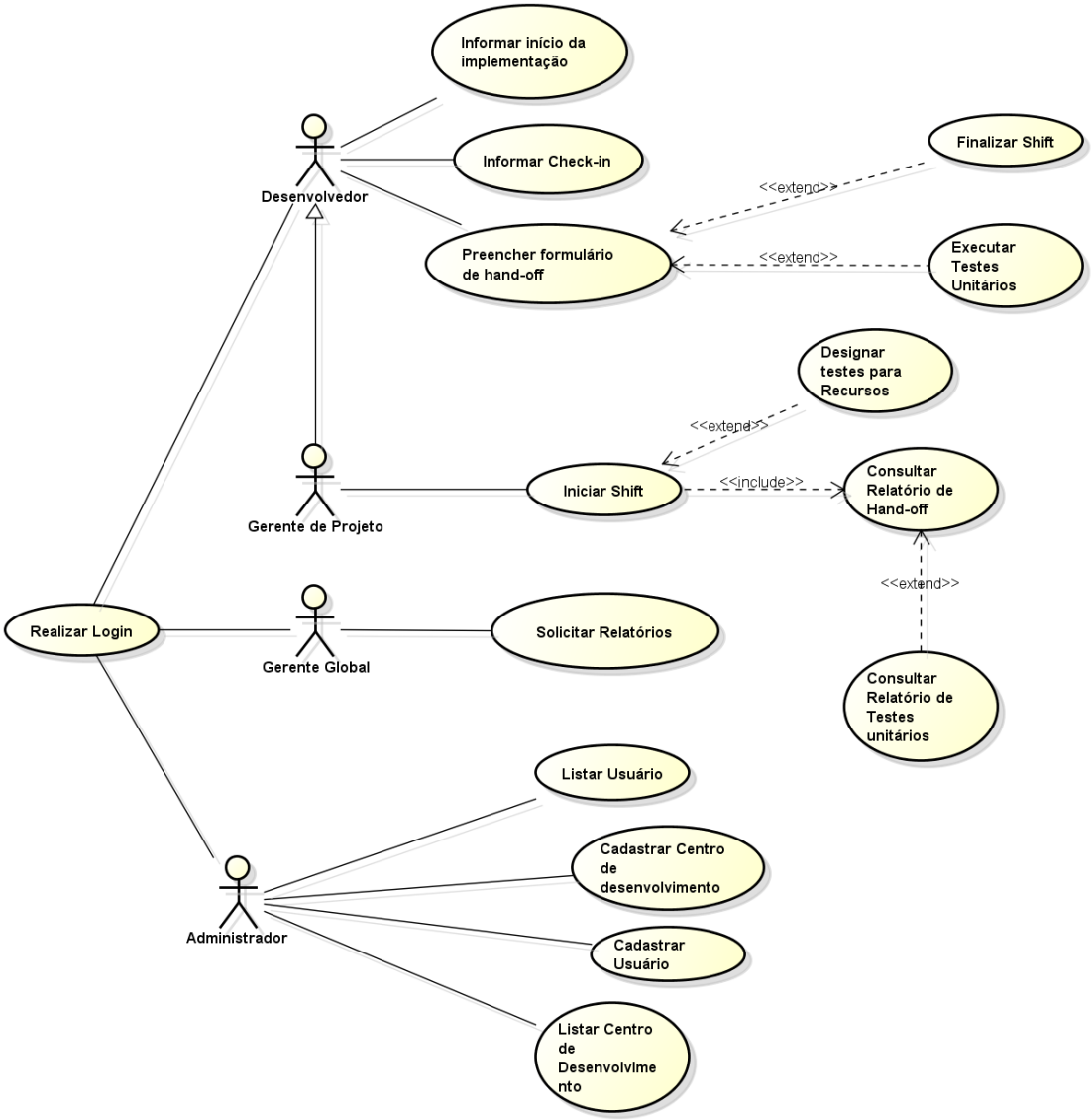


Figura 12. Casos de Uso da aplicação desenvolvida.

O Quadro 1 apresenta a especificação do caso de uso “Consultar relatório de *hand-off*”. Este caso de uso é executado a cada ciclo do processo, no início de cada dia de trabalho. Este é responsável por buscar as informações fornecidas pela equipe que trabalhou no *shift* imediatamente anterior. De forma automática, este caso de uso identifica qual equipe está iniciando o *shift*, e com esta informação gera um relatório com todas as informações necessárias, conforme definido no *FTSProc*, apresentando dados do *site* anterior, juntamente com os testes unitários ainda não cobertos.

Quadro 1. Especificação do caso de uso "Consultar relatório de *hand-off*"

UC: Consultar relatório de *hand-off*		
Ator	Gerente de Projeto	
Objetivo	Permite ao usuário exibir as informações fornecidas pelo site imediatamente anterior. Estas são as informações relativas ao último *shift* finalizado, e essenciais para iniciar o próximo *shift*.	
Fluxo Básico		
1.	Usuário seleciona a opção para visualizar o relatório.	
2.	Sistema identifica a qual site o usuário pertence, baseado nestas informações apresenta os seguintes dados:	
	2.1	Respostas para as três perguntas definidas no processo.
	2.2	Todos os testes alocados para cada recurso, mostrando se este foi coberto ou não;
Fluxo Alternativo		
A.	Caso usuário selecionar a opção para verificar o relatório de testes unitários:	
	1.	UC "Consultar Relatório de Testes Unitários" é acionado
Pós-condições:		
1. Sistema habilita a opção para que o início do *shift* seja realizado (UC "Iniciar *Shift*").		

O Quadro 2 apresenta o caso de uso "Iniciar *Shift*". A cada dia de trabalho, este caso de uso será executado. Sempre após o caso de uso especificado no Quadro 1. Este caso de uso é responsável por informar a ferramenta, dados sobre o *shift* que está sendo iniciado, para que a ferramenta consiga identificar o momento em que o *shift* é finalizado e a próxima equipe possa iniciar o desenvolvimento. Durante a execução deste caso de uso, deve-se informar a ferramenta quais os recursos que irão participar do *shift*, ou seja, os desenvolvedores que irão trabalhar neste dia. Esta informação é imprescindível, pois ao finalizar um *shift*, a ferramenta deve garantir que todos os recursos envolvidos no *shift* adicionaram as informações relativas ao *shift*. Este caso de uso ainda é responsável por designar recursos para trabalhar em determinadas tarefas. Com base nos testes que ainda devem ser cobertos, a ferramenta disponibiliza a opção para indicar qual recurso trabalhará em casa tarefa. Este passo é opcional, mas fortemente recomendado, pois facilita o controle das atividades de desenvolvimento.

Quadro 2. Especificação do caso de uso "Iniciar *shift*"

UC: Iniciar *Shift*		
Ator	Gerente de Projeto	
Objetivo	Permite ao usuário iniciar um novo *shift*. Para isto, é necessário executar o UC "Consultar relatório de *hand-off*".	
Pré-condições:		
1. UC "Consultar relatório de *hand-off*" já executado		
Fluxo Básico		
1.	Usuário seleciona a opção para iniciar o *shift*.	
2.	Sistema identifica a qual site o usuário pertence, baseado nestas informações solicita os seguintes dados para o usuário:	
	2.1	Desenvolvedores pertencentes a este site, juntamente com a opção para confirmar a sua participação no *shift* que está iniciando;
	2.2	Testes que ainda precisam ser trabalhados, juntamente com uma opção para designar um desenvolvedor para este teste;
	2.3	Campo de comentário para o *shift*, para adicionar outra informação relevante;
3.	Usuário preenche todas as informações, e confirma o início do *shift*;	
Pós-condições:		
1. Sistema armazena o novo *shift* como iniciado.		
2. Sistema habilita a opção "Iniciar Implementação" para todos os usuários confirmados para o *shift* que está iniciando.		

Da mesma forma que os casos de uso apresentados nos quadros 1 e 2, o caso de uso especificado no Quadro 3 também é executado a cada dia de trabalho, entretanto, ao final de cada dia. Este caso de uso é responsável por garantir que todas as informações necessárias para que o próximo centro de desenvolvimento possa iniciar o seu trabalho seja adicionado na ferramenta. Ao chegar ao final de um dia de trabalho, cada recurso que trabalhou no *shift* é responsável por adicionar as informações na ferramenta de apoio. Quando cada recurso adiciona as suas informações, a ferramenta verifica se todos os recursos envolvidos neste *shift* já adicionaram os dados. Quando for identificado que todos os recursos realizaram esta etapa, este caso de uso é responsável por executar outros dois casos de usos: *Executar testes unitários* e *Finalizar Shift.* Neste ponto, o *shift* está finalizado e um novo *shift* pode ser iniciado.

Quadro 3. Especificação do caso de uso "Preencher formulário de hand-off"

UC: Preencher formulário de *hand-off*		
Ator	Desenvolvedor	
Objetivo	Informar ao sistema todos os dados relacionados ao trabalho durante o *shift*.	
Fluxo Básico		
1.	Usuário informa ao sistema os seguintes dados:	
	1.1	O que foi realizado durante este período de trabalho?
	1.2	O que deve ser continuado no próximo período de trabalho?
	1.3	Existe algo bloqueando a equipe?
2.	Usuário informa quais os testes designados a ele que estão concluídos.	
3.	Sistema altera o status do usuário para que o mesmo não possa mais interferir neste *shift.*	
4.	Sistema armazena estas informações e faz a seguinte validação:	
	4.1	Fluxo Alternativo *A*
Fluxo Alternativo		
A.	Caso o usuário seja o último recurso do *shift* a preencher o relatório:	
	1.	UC "Executar testes unitários" é acionado
	2.	UC "Finalizar *Shift"* é acionado
Pós-condições:		
Após todos os desenvolvedores do *shift* submeterem os dados, o sistema armazena as informações e o *shift* é finalizado. O sistema habilita a opção para iniciar um novo *shift* para o próximo centro de desenvolvimento.		

Os casos de uso "Informar Início da Implementação" e "Informar *Check-in*" são utilizados apenas para a ferramenta controlar o fluxo de trabalho de cada recurso durante o projeto.

O caso de uso "Executar Testes Unitários" é responsável pela execução remota de testes. Configura-se a localização dos testes, e a ferramenta executa estes testes, coletando métricas como: número de testes executados, quais os testes estão cobertos e detalhes dos testes ainda não cobertos.

O caso de uso "Consultar Relatório de Testes Unitários" apresenta as informações relacionadas à situação atual dos testes unitários.

O caso de uso "Designar Testes para Recursos" apenas faz a associação dos testes e dos desenvolvedores para que possa ser realizado um controle sobre o trabalho de cada um dos desenvolvedores. A ferramenta não exige que esta associação seja realizada, e é apenas mais uma forma de realizar o controle.

O perfil de usuário "Gerente Global" tem a possibilidade de acessar o caso de uso "Solicitar Relatórios". Neste caso de uso, existe a opção para escolher

diversos relatórios, para fazer o acompanhamento do projeto. Alguns exemplos de relatórios são:

- Percentual de recursos alocados em cada centro de desenvolvimento (*site*).
- Dados detalhados sobre todos os *shifts* já executados.
- Quantidade de testes cobertos por cada *shift*.
- Tempo de duração de cada *shift*.

A ferramenta proposta foi implementada e utilizada durante a execução do experimento. Optou-se por desenvolver uma aplicação *Web*, para que a mesma pudesse ser utilizada de uma forma centralizada por todas as equipes. Utilizou-se a linguagem *Java* para a criação da mesma. Desta forma, é facilitada a instalação da ferramenta em qualquer plataforma (Servidor Web). Ainda, o banco de dados utilizado foi *MySQL* e, em todas as consultas, utilizou-se a linguagem *SQL* padrão. Desta forma também fica facilitada a alteração do banco de dados, alterando apenas configurações de conexão. Abaixo, algumas imagens da ferramenta em funcionamento:

1. A Figura 13 apresenta a tela utilizada na inicialização de um dia de trabalho (*shift*):

Figura 13. Tela para iniciar um dia de trabalho (shift).

Na figura acima, com o identificador *A*, temos uma lista com todos os possíveis participantes do *shift*, ou seja, todos os recursos alocados para o centro que está iniciando o *shift*. Deve-se, obrigatoriamente, selecionar os participantes (por padrão, todos são selecionados). Destacado com o identificador *B* está a lista de testes unitários gerada pela ferramenta. Com base nesta lista é possível associar testes para os participantes dos *shifts*, para que estes fiquem responsáveis por trabalhar nas atividades relacionadas ao teste. Este passo não é mandatório, mas recomendável, pois facilita no controle das atividades do projeto. Esta tela mostra ainda um campo de texto onde é possível adicionar qualquer informação que o Gerente de Projeto julgar relevante. Após informar todos os dados, deve-se clicar no botão "*Iniciar Shift*".

2. A Figura 14 apresenta a tela utilizada para adicionar informações sobre cada *shift*:

Figura 14. Tela para finalizar um dia de trabalho (*shift*).

Nesta tela, na área destacado com o identificador *A*, o usuário deve indicar, respondendo essas três perguntas, todas as informações relevantes para que o próximo centro inicie o trabalho do ponto onde parou. Ainda, na área destacado

com o identificador *B*, estão listados os testes designados ao usuário logado. Nesta área, o usuário pode indicar os testes que foram trabalhados. Esta informação é validada de forma automatizada pela ferramenta, se os testes foram realmente cobertos, o teste é considerado concluído, e não é mais apresentado no relatório de testes unitários, caso contrário, o *shift* seguinte identificará o teste como não coberto no relatório de testes unitários. O *shift* é considerado concluído somente quando todos os usuários alocados para este *shift* informarem os dados necessários para concluir o *hand-off*. Ao identificar este caso, a ferramenta habilita automaticamente o próximo centro a iniciar o seu *shift*, caso contrário, o *shift* não pode ser iniciado.

3. A Figura 15 apresenta a Tela para o relatório de quantidade de testes cobertos em cada *shift*:

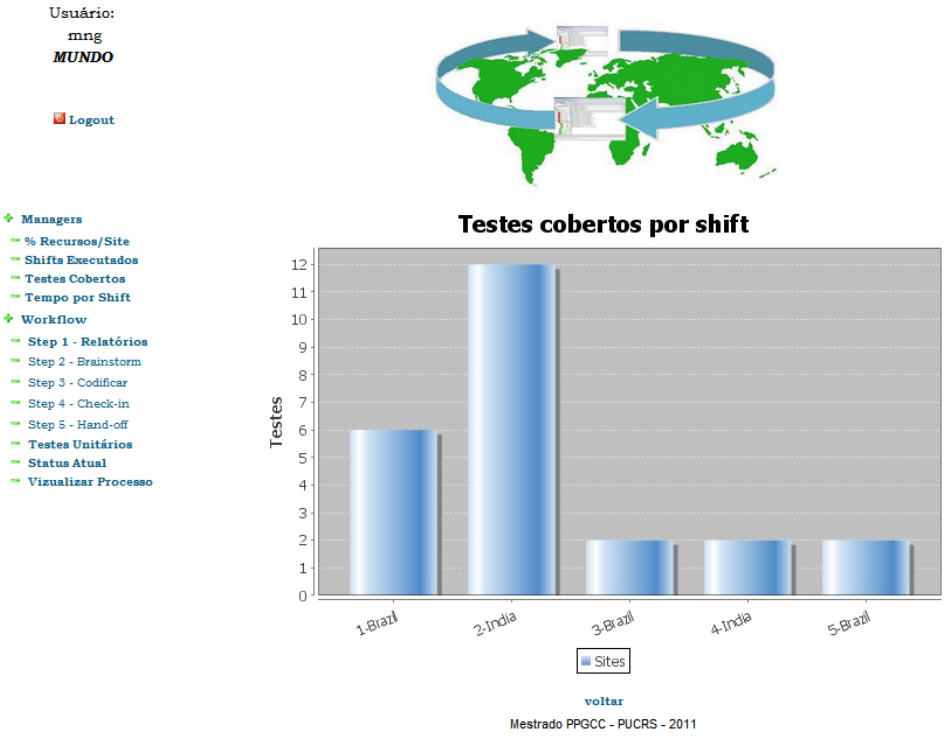


Figura 15. Relatório de quantidade de testes cobertos em cada shift.

Com esta informação é possível acompanhar o progresso do projeto, e identificar quais os *shifts* foram mais ou menos produtivos. Assim é possível tomar ações, como por exemplo, a adição de recursos a determinado centro de desenvolvimento.

4. A Figura 16 apresenta a tela para o relatório de duração de cada *shift*:

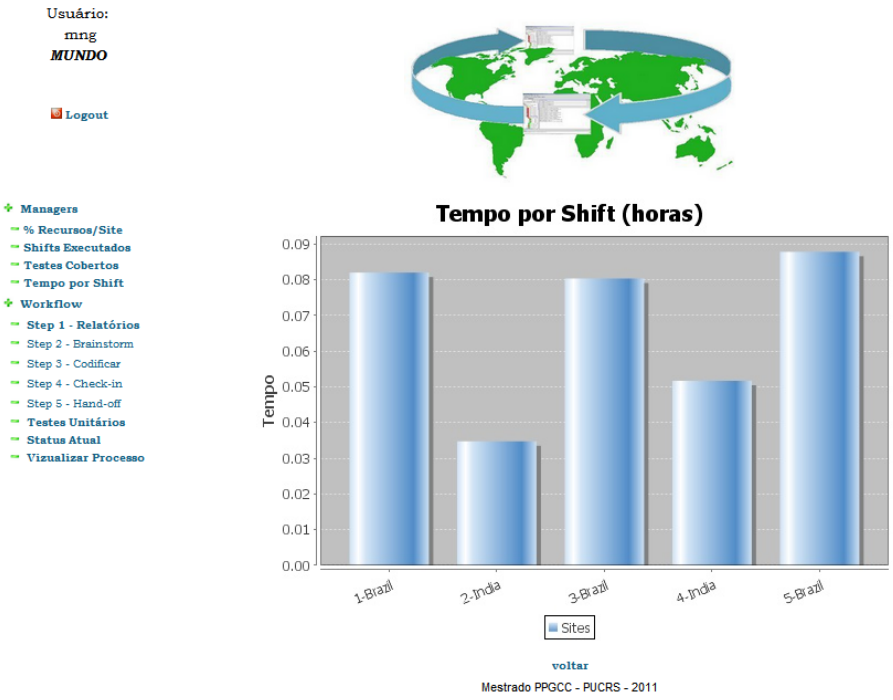


Figura 16. Relatório da duração de cada shift.

Este relatório mostra o tempo transcorrido desde o início do *shift*, até o momento em que o último desenvolvedor alocado no *shift* finalizou o seu *hand-off*. Esta informação é relevante, por exemplo, para verificar se está sendo necessária a utilização de horas-extras para a conclusão do processo.

5 ESTUDO EXPERIMENTAL

Conforme apresentado no capítulo 3, o principal método de pesquisa utilizado neste trabalho foi o experimento controlado. Um dos usos mais comuns de experimentos na ES é para testar teorias [JUR01]. Visto que esta pesquisa visa investigar a aplicação de um novo processo de transferência de trabalho em ambientes distribuídos que utilizam a estratégia FTS e, devido às características deste método, optou-se pela realização de um experimento. Esta escolha deve-se ao fato deste método possibilitar a avaliação de novas propostas, permitir o controle do ambiente de execução e proporcionar a obtenção de conclusões a partir de uma hipótese [WOH00]. Este processo experimental será estruturado conforme os estudos de [WOH00] e [TRA02]. A Figura 17 a seguir apresenta esta estrutura.

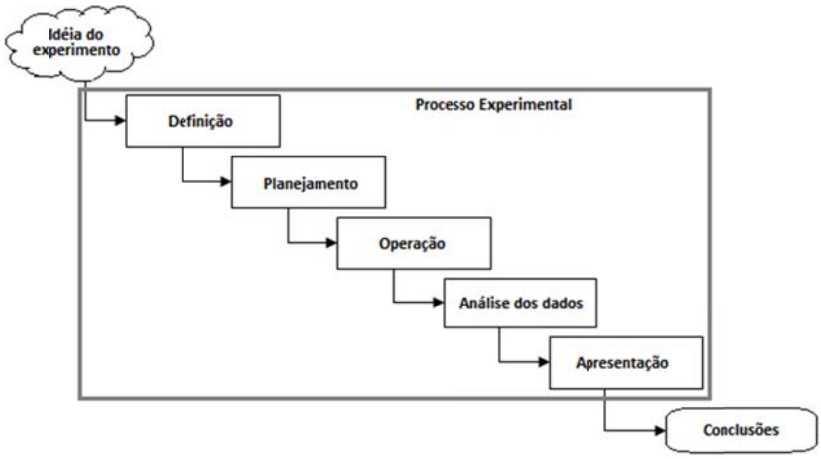


Figura 17. Etapas do processo experimental, adaptado de [WOH00].

As seções a seguir detalham as etapas do processo experimental realizado.

5.1 DEFINIÇÃO

Nesta etapa foi utilizada a abordagem *Goal Question Metric* (GQM) [BAS94] que define os objetivos (nível conceitual) para estabelecer questões

(nível operacional) e então identificar métricas (nível quantitativo). No contexto de experimentos esta abordagem auxilia a etapa de definição de objetivos [WOH00].

O objetivo global deste experimento consiste em investigar qual abordagem é mais eficiente em projetos que utilizam a abordagem FTS: utilizando o *FTSProc* ou sem o uso de tal processo (chamado de *ad hoc*), possibilitando, assim, identificar qual abordagem permite a entrega do maior número de requisitos implementados em um determinado intervalo de tempo.

Para alcançar o objetivo deste estudo buscou-se responder a seguinte questão: “Projetos que utilizam o FTSProc têm a mesma eficiência que projetos distribuídos realizados de forma ad hoc?”. A métrica associada a esta questão corresponde à eficiência do método, calculada pelo somatório dos requisitos corretamente implementados pelos participantes em cada uma das duas abordagens. Neste trabalho definiu-se como requisitos corretos aqueles que possuíam as seguintes características:

- Requisitos implementados em linguagem *Java*, de acordo com a descrição dos requisitos do sistema, entregue aos participantes (Apêndice C);
- Cada requisito possui uma série de critérios de aceitação onde, para os critérios de cálculos temos:
    - Todos os critérios de aceitação cobertos, o requisito está implementado corretamente;
    - Algum critério de aceitação não coberto, o requisito está parcialmente implementado.

Vale ressaltar que na análise dos resultados do experimento focou-se para identificar a quantidade de requisitos implementados corretamente. A seguir, apresenta-se a fórmula para o cálculo da métrica:

- Eficiência do projeto *FTSProc* = ∑ reqImpleFTSProc
- Eficiência do projeto *FTSProc* = ∑ reqImpleAdHoc.

Onde “*reqImpleFTSProc*” representam os requisitos implementados de forma correta utilizando o *FTSProc* e “*reqImpleAdHoc*” representam os requisitos implementados de forma correta sem a utilização do *FTSProc* (projeto *ad hoc*).

5.2 PLANEJAMENTO

Para este experimento, utilizou-se a seguinte abordagem de contexto:

- O experimento ocorreu num ambiente controlado (*in-vitro*) em um dado instante de tempo (*off-line*);
- O grupo de participantes era composto por alunos do Programa de Pós Graduação em Ciência da Computação da PUCRS (PPGCC) e profissionais experientes, oriundos de empresas do Parque Tecnológico da PUC/RS (TECNOPUC);
- A realidade do experimento é considerada modelada, visto que os requisitos a serem implementados foram desenvolvidos pelo Pesquisador;
- A generalização do experimento é considerada específica.

Este contexto foi adotado por motivos de complexidade e viabilidade para a realização do experimento. Desta maneira, a execução do experimento ocorreu num ambiente controlado durante um momento previamente estabelecido, com uma amostra definida por conveniência (participantes estudantes de mestrado e profissionais da área) e com um problema fictício desenvolvido pelo Pesquisador.

Definiu-se as seguintes hipóteses para o experimento:

- Hipótese Nula (H_0): A eficiência de um projeto que utiliza o *FTSProc* é igual ao de um projeto distribuído desenvolvido de forma *ad hoc*.
    - H_0: ∑reqImpleFTSProc = ∑reqImpleAdHoc
- Hipótese Alternativa (H_1): A eficiência de um projeto que utiliza o *FTSProc* é maior que a de um projeto distribuído desenvolvido de forma *ad hoc*.
    - H_1: ∑reqImpleFTSProc > ∑reqImpleAdHoc
- Hipótese Alternativa (H_2): A eficiência de um projeto que utiliza o *FTSProc* é menor que a de um projeto distribuído desenvolvido de forma *ad hoc*.
    - H_2: : ∑reqImpleFTSProc < ∑reqImpleAdHoc

As duas abordagens, com a utilização do processo e sem a utilização do mesmo (*ad hoc*), são consideradas as variáveis independentes do experimento e a eficiência é a variável dependente. Os sujeitos do experimento incluíram 8 pessoas, as quais formaram dois grupos de quatro componentes. Cada grupo executou o mesmo projeto onde em um deles foi utilizado o *FTSProc* e o outro foi executado de forma *ad hoc*, ou seja, sem um processo definido.

Antes de iniciar o experimento cada participante preencheu um questionário com informações sobre os conhecimentos das áreas da pesquisa, as quais incluíram DDS e FTS, juntamente com outros conhecimentos técnicos necessários para a realização do experimento (Apêndice F). Este instrumento auxiliou de duas formas: na preparação dos treinamentos necessários e na divisão das duas equipes da forma mais balanceada possível. A Tabela 4 expõe os conhecimentos e o tempo de experiência dos participantes nas áreas pertinentes ao experimento.

A amostragem do experimento é considerada por conveniência e não probabilística, optando-se por esta opção por questões de viabilidade. Para minimizar a possível obstrução causada pelas diferenças no nível de experiência e conhecimento dos participantes nos temas relacionados ao experimento (DDS e FTS) foram selecionados estudantes de mestrado e profissionais que trabalham em projetos que utilizam DDS (pressupondo obter homogeneidade em relação a experiências e conhecimentos) e foram fornecidos treinamentos sobre estes temas. Foi adotado o princípio de balanceamento para que cada abordagem fosse utilizada pela mesma quantidade de participantes.

A Tabela 4 apresenta os dados dos participantes obtidos através de um questionário aplicado previamente ao experimento (Apêndice F). Estes dados representam o nível de conhecimento dos participantes nas áreas necessárias para a execução do experimento, as quais foram usadas para fazer o balanceamento das equipes. As informações contidas em cada coluna são:

- ID: identificação do participante;
- DDS: nível de conhecimento do participante em DDS;
- DDS Tempo: anos de experiência profissional do participante em DDS;
- FTS: nível de conhecimento do participante em FTS;

- FTS Tempo: anos de experiência profissional do participante em FTS;
- Exp. Java: anos de experiência profissional em desenvolvimento na linguagem Java;
- *JUnit*: nível de conhecimento do participante em na ferramenta *JUnit*;
- TDD: nível de conhecimento do participante na técnica de TDD;

Tabela 4. Conhecimento dos Participantes.

ID	DDS	DDS Tempo (anos)	FTS	FTS Tempo (anos)	Exp. Java	JUnit	TDD
S1	Avançado	6	Intermediário	0	>5	Avançado	Intermediário
S2	Básico	2	Nenhum	0	3-5	Básico	Intermediário
S3	Avançado	5	Básico	0	3-5	Intermediário	Básico
S4	Intermediário	5	Nenhum	0	>5	Intermediário	Intermediário
S5	Avançado	8	Básico	2	>5	Avançado	Intermediário
S6	Intermediário	4	Nenhum	0	3-5	Intermediário	Intermediário
S7	Intermediário	4	Nenhum	0	3-5	Intermediário	Básico
S8	Intermediário	4	Básico	0	3-5	Avançado	Básico

A Tabela 5 apresenta a distribuição dos participantes no experimento. Utilizou-se o mesmo identificador da Tabela 4. Pode-se observar através da Tabela 4 que se procurou fazer o balanceamento dos participantes entre as duas abordagens: com a utilização do *FTSProc* e sem a utilização do processo (projeto *ad hoc*). Para a realização do experimento os conhecimentos mais importantes necessários para os participantes estavam voltados ao desenvolvimento de sistemas, ou seja, experiência em Java e conhecimentos em *JUnit* e *TDD*. Portanto, para o balanceamento na distribuição dos participantes, procurou-se avaliar as suas capacidades focando nestas áreas, como por exemplo, o tempo de experiência em *Java*, conhecimentos em *JUnit* e TDD. Esta tabela apresenta ainda, a distribuição dos participantes entre os diferentes centros de desenvolvimento (*site*).

Tabela 5. Distribuição do fator sobre os tratamentos.

ID Participante	Site	*FTSProc*	*Ad hoc*
S1	Site 1	**X**	
S2	Site 1	**X**	
S3	Site 2	**X**	
S4	Site 2	**X**	
S5	Site 1		**X**
S6	Site 1		**X**
S7	Site 2		**X**
S8	Site 2		**X**

A Tabela 6 apresenta todos os instrumentos utilizados durante o processo experimental realizado, incluindo o tipo de objeto e a descrição detalhada de cada um, como por exemplo: ferramentas utilizadas e questionários aplicados.

Tabela 6. Instrumentação do experimento realizado.

Tipo	Descrição
Objeto	Ferramentas: IDE Eclipse *Indigo* para o desenvolvimento da aplicação, *Tortoise* SVN para a sincronização de arquivos (*check-in* e *check-out*) com o repositório e navegador *Chrome* para utilização da ferramenta de apoio.
	Ferramenta de apoio desenvolvida pelo Pesquisador para controle do processo *FTSProc* e coleta das métricas durante a realização do experimento para o projeto FTS e, para a coleta de métricas no projeto *ad hoc.*
	Descrição do sistema a ser desenvolvido, diagramas de casos de uso e diagramas de classes necessárias para o desenvolvimento da aplicação (Apêndice C).
	Código fonte sobre o qual o desenvolvimento foi realizado. Todas as classes e arquivos necessários foram disponibilizados pelo Pesquisador.
Guia	Apresentação para a equipe que utilizou o *FSTProc*, contendo informações sobre DDS, FTS, processo *FTSProc*, ferramenta de apoio e a dinâmica do experimento (Apêndice D).
	Apresentação para a equipe que realizou o projeto de forma *ad hoc*, contendo informações sobre DDS, FTS e a dinâmica do experimento (Apêndice E).
Métrica	Questionário enviado aos participantes alguns dias antes da execução do experimento, para a coleta de dados demográficos e sobre o conhecimento dos mesmos em relação à DDS e FTS (Apêndice F).
	Questionário entregue no final da execução do experimento, para coletar as percepções dos participantes sobre o experimento e suas sugestões para o método que utilizou (Apêndice G).

A Tabela 7 apresenta as considerações acerca da validade do processo experimental. Alguns dados que esta tabela apresenta são: validade interna, como dados históricos e seleção dos participantes; validade externa como a possibilidade de generalização; validade de construção como a explicação aos participantes sobre a forma como os dados seriam extraídos; e dados sobre a

validade de conclusão do experimento mostrando, por exemplo, a falta de poder estatístico devido ao número reduzido de participantes.

Tabela 7. Validade do experimento realizado.

Validade interna	
Histórico	A data de aplicação do experimento foi definida evitando períodos em que os participantes poderiam sofrer influências externas (choque de horários com compromissos). Devido à dificuldade de encontrar uma data comum onde todos os participantes tivessem a disponibilidade, o experimento foi realizado em dois dias distintos, um para a equipe *FTSProc* e outro para a equipe *ad hoc*.
Maturação	Buscou-se motivar os participantes durante a execução do experimento indicando a importância da realização do mesmo.
Seleção	Os participantes participaram voluntariamente do experimento.
Difusão ou imitação de tratamentos	Durante a execução do experimento, não foi permitido qualquer tipo de interação entre os participantes que representavam *sites* diferentes, simulando times distribuídos em locais e fusos horários distintos.
Validade externa	
Interação de seleção e tratamento	Os participantes possuíam conhecimento prévio sobre os assuntos relacionados a pesquisa.
Interação do ambiente e tratamento	Foram utilizadas ferramentas atuais e amplamente conhecidas. Todas as ferramentas foram previamente configuradas pelo Pesquisador.
Interação entre histórico e tratamento	A execução do experimento ocorreu em um momento em que os participantes não sofreram influências externas.
Possibilidade de generalização	Devido ao fato do experimento ser *in-vitro* e *off-line*, a generalização do experimento é considerada específica.
Validade de construção	
Inadequada explicação pré-operacional	Buscou-se explicar detalhadamente questões operacionais do experimento (como ocorreria a extração dos dados, uso de ferramentas, etc.)
Adivinhação de hipóteses	Manteve-se o foco no objetivo planejado, não divulgando a métrica do experimento.
Apreensão sobre a avaliação	Foi declarado que se manteria o anonimato dos participantes e que eles não estavam sendo “avaliados”.
Validade de conclusão	
Poder estatístico	O pequeno tamanho da amostra (8 participantes) resultou na impossibilidade da utilização de métodos estatísticos para o teste de hipóteses, por isto optou-se por uma interpretação analítica de base qualitativa dos resultados, conforme apresentado seção 5.4 (Análise e interpretação de resultados).
Confiabilidade das medidas	Utilizou-se medidas objetivas no experimento.
Configurações do ambiente do experimento	O experimento foi conduzido em laboratório totalmente controlado.
Heterogeneidade do ambiente do experimento	Foram escolhidos participantes alunos do PPGCC e profissionais da área.

5.3 OPERAÇÃO

Nesta etapa ocorreu a preparação, execução e validação inicial dos resultados do experimento realizado. Este processo experimental realizado para avaliar a eficiência do *FTSProc* foi realizado em um ambiente totalmente controlado, atendendo todos os pré-requisitos necessários para a utilização do FTSProc.

Durante a preparação foi fornecido o embasamento teórico necessário para a participação dos sujeitos deixando claro quais eram os objetivos do experimento e como ele ocorreria. Foi pedido que os participantes assinassem um termo de consentimento no qual foram descritos os objetivos da pesquisa e os direitos dos participantes (Apêndice H). Para evitar influências nos resultados, foi adotada uma postura de anonimato dos participantes na descrição do experimento.

A execução do experimento foi estruturada em quatro fases sequenciais, apresentadas a seguir:

Fase 1 – Coleta de dados iniciais: Inicialmente o Pesquisador foi responsável por enviar aos participantes, alguns dias antes da execução do experimento, um questionário para coletar informações demográficas e informações relativas ao conhecimento e experiência sobre os assuntos envolvidos no experimento: DDS e FTS (Apêndice F). Analisando estes questionários o Pesquisador construiu apresentações (Fase 2 do experimento), contendo informações sobre estes assuntos (Apêndice D e Apêndice E). Logo após, foram definidas as duas equipes, cada uma composta de quatro integrantes, mantendo um balanceamento entre elas conforme apresentado na A Tabela 5 apresenta a distribuição dos participantes no experimento. Utilizou-se o mesmo identificador da Tabela 4. Pode-se observar através da Tabela 4 que se procurou fazer o balanceamento dos participantes entre as duas abordagens: com a utilização do FTSProc e sem a utilização do processo (projeto ad hoc). Para a realização do experimento os conhecimentos mais importantes necessários para os participantes estavam voltados ao desenvolvimento de sistemas, ou seja, experiência em Java e conhecimentos em JUnit e TDD. Portanto, para o balanceamento na distribuição dos participantes, procurou-se avaliar as suas

capacidades focando nestas áreas, como por exemplo, o tempo de experiência em Java, conhecimentos em JUnit e TDD. Esta tabela apresenta ainda, a distribuição dos participantes entre os diferentes centros de desenvolvimento (site). Uma equipe utilizou o FTSProc e a outra não fez uso do processo, realizando o projeto de forma ad hoc. A Figura 18 apresenta a distribuição das equipes.

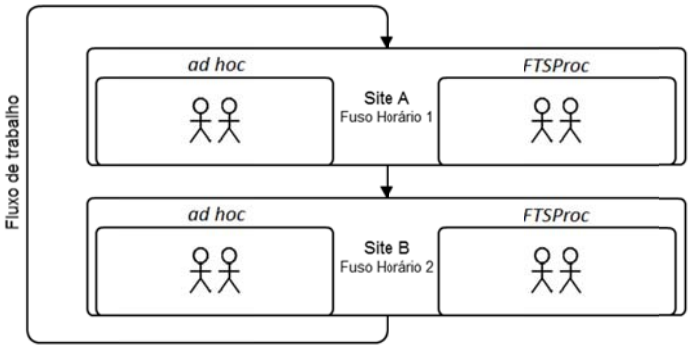


Figura 18. Distribuição das equipes.

Fase 2 – Apresentações: Nesta fase, foram realizadas as apresentações, as quais incluíram 2 conjuntos de slides, assim divididos:

- Equipe *FTSProc*: informações sobre DDS, FTS, *FTSProc*, ferramenta de apoio e a dinâmica do experimento (Apêndice D);
- Equipe *ad hoc*: informações sobre DDS, FTS e a dinâmica do experimento (Apêndice E).

Após estas apresentações, as dúvidas dos participantes foram solucionadas. Nesta fase não ocorreu nenhum imprevisto e tudo ocorreu conforme o planejamento. O tempo total desta fase foi de:

- Projeto *FTSProc*: 25 minutos.
- Projeto *ad hoc*: 10 minutos.

Fase 3 – Execução: Nesta fase, o Pesquisador divulgou os requisitos que seriam implementados (Apêndice C). Vale ressaltar que os requisitos a serem desenvolvidos eram de um sistema matemático simples. Este domínio foi escolhido devido a sua facilidade e provável conhecimento de todos os participantes. Além disto, foram disponibilizadas as apresentações realizadas na Fase 2 para todos os participantes. De posse destes materiais, as equipes analisaram os requisitos por 5 minutos e iniciaram a implementação dos mesmos.

Os tempos disponibilizados para cada projeto, *FTSProc* e *ad hoc*, eram os mesmos e estão apresentados na tabela a seguir:

Tabela 8. Tempos para realização de cada *shift*.

Shift	*Site*	Tempo
1	Site 1	20 Minutos
2	Site 2	20 Minutos
3	Site 1	20 Minutos
4	Site 2	20 Minutos

Ao final desta fase, as equipes disponibilizaram todo o código fonte desenvolvido durante a execução do experimento, juntamente com os dados de cada *hand-off* coletados através da ferramenta de apoio. Nesta fase, nenhum imprevisto ocorreu e a Tabela 9 e Tabela 10 apresentam os tempos necessários para concluir esta etapa em cada abordagem:

- Projeto *FTSProc*

Tabela 9. Tempos do projeto *FTSProc*.

Atividade	Hora Inicial	Hora Final	Tempo Total
Análise dos requisitos	18:05	18:10	5 Minutos
Shift 1	18:12	18:32	20 Minutos
Shift 2	18:40	19:00	20 Minutos
Shift 3	19:06	19:26	20 Minutos
Shift 4	19:27	19:34	7 Minutos

- Projeto *Ad hoc*

Tabela 10. Tempos do projeto *ad hoc*.

Atividade	Hora Inicial	Hora Final	Tempo Total
Análise dos requisitos	17:52	17:57	5 Minutos
Shift 1	18:02	18:22	20 Minutos
Shift 2	18:22	18:42	20 Minutos
Shift 3	18:45	19:05	20 Minutos
Shift 4	19:06	19:26	20 Minutos

Fase 4 – Coleta de dados finais: Na fase final, o Pesquisador aplicou um questionário (Apêndice G) para coletar as percepções dos participantes sobre o experimento e sobre a utilização do *FTSProc*. Nesta fase não ocorreu nenhum desvio ou imprevisto, tudo ocorreu conforme o planejamento. O tempo total desta fase em cada abordagem foi:

- Projeto *FTSProc*: 5 minutos.
- Projeto *ad hoc*: 8 minutos.

5.4 ANÁLISE E INTERPRETAÇÃO DOS RESULTADOS

Esta etapa compreende a análise e interpretação dos resultados com o intuito de obter conclusões sobre as hipóteses do experimento. Conforme descrito anteriormente, o tamanho da amostra do experimento foi de 8 participantes. Este baixo número foi obtido devido aos conhecimentos necessários para cada participante e a viabilidade para a realização do experimento, incluindo:

- Conhecimentos básicos em Orientação a Objetos;
- Conhecimentos em linguagem *Java* (J2SE);
- Conhecimentos da *IDE Eclipse*;
- Conhecimentos em *Test-driven development* (TDD);
- Conhecimento em testes unitários (*JUnit*);
- A restrição de tempo para que o experimento ocorresse no cronograma e prazo planejados para a conclusão desta pesquisa.

Todos estes fatores restringiram a possibilidade de obtenção de uma amostra maior. Desta maneira, devido a este baixo número da amostra não se obteve dados suficientes para a utilização de métodos estatísticos no teste das hipóteses, optando-se então, por uma interpretação analítica de base qualitativa utilizando uma estatística simples para analisar os resultados obtidos.

5.4.1 Análise quantitativa

O resultado mais importante oriundo do experimento está relacionado diretamente à quantidade de requisitos desenvolvidos pelas equipes em cada

abordagem. Este resultado será utilizado para a verificação da hipótese proposta em relação à eficiência do método *FTSProc* proposto. A Tabela 11 apresenta os resultados obtidos em relação ao número de requisitos implementados corretamente e parcialmente em cada projeto.

Tabela 11. Requisitos implementados.

Requisitos implementados	*FTSProc*	*Ad hoc*
Corretamente	12	4
Parcialmente	0	8
Não Implementado	0	0
Total do Projeto	12	12

Analisando os resultados da Tabela 11, podemos verificar que a eficiência da equipe que utilizou o *FTSProc* é maior do que a equipe que realizou o projeto de forma *ad hoc*. A equipe que utilizou o *FTSProc* implementou uma quantidade maior de requisitos corretos do que a equipe do projeto *ad hoc.* Além disto, a equipe que utilizou o *FTSProc* obteve uma maior taxa de aproveitamento de trabalho (quantidade de requisitos corretos *100/quantidade de requisitos definidos):

- A equipe que utilizou o *FTSProc* obteve uma percentagem de aproveitamento de 100% (12) dos requisitos corretamente implementados em relação ao total de requisitos que elas trabalharam;
- A equipe que executou o projeto de forma *ad hoc* obteve uma percentagem de aproveitamento de 33,3% (4) dos requisitos corretamente implementados e 66,6% (8) dos requisitos parcialmente implementados, em relação ao total de requisitos que elas trabalharam (12).

É importante salientar que, conforme podemos ver na Tabela 9, a equipe que utilizou o *FTSProc* precisou somente de 3 *shifts* e 7 minutos do quarto *shift*, para garantir que todos os requisitos fossem implementados corretamente. Enquanto isso, a outra equipe utilizou todos os 4 *shifts* integralmente, para terminar apenas 4 requisitos corretamente.

Com estes resultados percebemos que a quantidade de requisitos corretos e a taxa de aproveitamento é maior quando utilizamos o *FTSProc* para a

execução de projetos distribuídos utilizando a estratégia FTS. Estes dados fornecem indícios para a aceitação da hipótese alternativa H_1 ("A eficiência de um projeto que utiliza o *FTSProc* é maior que a de um projeto distribuído desenvolvido de forma *ad hoc*").

5.4.2 Análise qualitativa

Na fase 4 do experimento (Coleta de dados finais), o Pesquisador aplicou um questionário (Apêndice G) para que os participantes respondessem sobre suas percepções acerca do método que eles utilizaram: *FTSProc* ou *ad hoc*. Para cada pergunta aplicada para os dois grupos, procurou-se comparar as percepções dos participantes nas duas abordagens. Os resultados obtidos nesta etapa estão representados nas tabelas a seguir, onde, para cada pergunta, apresenta-se a contagem de respostas positivas e negativas para cada uma das equipes. Logo após, é apresentada uma análise destes resultados.

- A transferência de trabalho de um centro ao outro ocorreu da forma adequada?

Tabela 12. Trabalho ocorreu de forma adequada?

	FTSProc	*Ad hoc*
Sim	4	3
Não	0	1
Total	4	4

- No início de cada *shift*, você percebia de forma direta como o trabalho deveria ser continuado?

Tabela 13. Percebe como o trabalho deve ser continuado?

	FTSProc	*Ad hoc*
Sim	4	0
Não	0	4
Total	4	4

- Você acredita que a transferência de trabalho de um centro de desenvolvimento para o outro acarretou um *overhead* significativo no trabalho?

Tabela 14. Acredita que a tranferência acarretou um *overhead*?

	FTSProc	*Ad hoc*
Sim	0	1
Não	4	3
Total	4	4

Nota-se que, na percepção dos participantes, a transferência de trabalho de um centro para o outro aconteceu de uma forma adequada em ambas as abordagens. Apenas um participante no projeto *ad hoc* discorda. Entretanto, podemos observar claramente que a identificação do ponto em que o trabalho deveria ser continuado não foi direto na equipe *ad hoc*. Já na equipe que utilizou o *FTSProc*, esta identificação foi facilitada. Este resultado vai ao encontro de um dos objetivos do processo proposto, o qual é facilitar a identificação do ponto onde o trabalho deve ser continuado. Finalmente, podemos identificar que, na percepção dos participantes, o *overhead* causado pelo uso do *FTSProc* não foi significativo. Os resultados encontrados estão em consonância com a literatura, a qual mostra que este tipo de processo deve ser "*leve*", ou seja, não pode acarretar um grande aumento na carga de trabalho (*overhead*) em um dia típico de trabalho [DEN09, TAW02].

Adicionalmente, outros questionamentos foram aplicados aos participantes, conforme apresentados no Apêndice G, visando a identificação de pontos positivos, pontos negativos e oportunidades de melhorias das duas abordagens (*FTSProc* ou *ad hoc*). Os resultados obtidos de cada participante estão transcritos a seguir, divididos em duas partes, uma para a equipe *ad hoc* e outra para a equipe *FTSProc*. Logo após é apresentado uma breve análise sobre estes dados.

- *Ad hoc*
    - Pontos Positivos
        - Baixa complexidade das tarefas;
        - Comentários no código fonte e no repositório ajudaram na continuação das tarefas.
    - Pontos Negativos
        - Não houve uma didática para a transferência de trabalho;
        - Não tinha como saber em que ponto do trabalho a equipe anterior parou, sem olhar diretamente todo o código fonte;

        - Em todos os *shifts*, antes de iniciar o trabalho era preciso verificar o que tinha sido feito pela outra equipe e então continuar;
        - Não é muito intuitivo se as equipes não tiverem um mecanismo de sincronização;
        - Falta de visibilidade do progresso do trabalho realizado;
        - Código implementado de forma parcial, bem como testes unitários dificultaram o trabalho;
        - Dificuldade para saber em que ponto começar.
    - Sugestões para melhorar o processo
        - Com o auxilio de *kanban* ou outras técnicas de gerenciamento;
        - Utilização de padrões de código;
        - Utilização de *Unit Tests* (TDD) e comentários no código fonte e durante os *check-ins*;
        - Definição de um sistema de sincronização de tarefas.
- *FTSProc*
    - Pontos Positivos
        - Utilização de TDD;
        - Requisitos diretos, com critérios definidos de forma clara;
        - 3 Perguntas utilizadas no processo.
    - Pontos Negativos
        - Dependente da ferramenta;
        - Todos tem que usar um mesmo “vocabulário” para que a comunicação seja efetiva.

    - Sugestão de melhoria ao *FTSProc*
        - Utilizar algum mecanismo de controle automático para *check-in*. Desta forma, evita-se que o próximo *shift* inicie sem o código mais recente no repositório.

Os pontos positivos em relação ao método *ad hoc* citados pelos participantes do experimento incluem a baixa complexidade das tarefas, pois os requisitos do experimento foram criados com este objetivo. Outro ponto positivo

apontado foi a forma como os participantes procuraram mostrar à outra equipe o estado atual do trabalho. Para isto, os participantes utilizaram comentários no código fonte e no repositório de código a cada *check-in*.

Já os pontos negativos citados em relação ao método *ad hoc* nos mostram, principalmente, os problemas que a falta de um processo pode ocasionar. Dentre estes problemas, a falta da percepção do ponto onde o trabalho havia parado no *shift* anterior e como este deveria ser continuado foram os mais citados pelos participantes. Cabe ressaltar que estes pontos estão em consonância com literatura. Os principais problemas apontados pela literatura estão relacionados as dificuldades de coordenação, sincronização e comunicação, principalmente durante a transferência de trabalho de um centro de desenvolvimento para outro [SET07, SOL10, CAR09]. O *FTSProc* procura amenizar estes problemas.

As sugestões de melhorias listadas pelos participantes que utilizam o método *ad hoc* mostram a necessidade da utilização de uma forma padrão para a transferência de trabalho, tal qual o *FTSProc*. Ainda, a utilização da técnica de TDD foi citada como um possível facilitador para a sincronização entre as equipes.

Os pontos positivos citados pelos participantes que utilizaram o *FTSProc* estão relacionados diretamente com a forma como o processo foi criado, ou seja, a utilização de requisitos diretos, utilização de TDD e as 3 perguntas utilizadas como base para a transferência de trabalho.

Como era esperado no início do experimento, os pontos negativos apontados pelos participantes do projeto *FTSProc* eram poucos, se comparados com o projeto *ad hoc* e não estão relacionados a sincronização ou coordenação de trabalho. Um ponto levantado versa acerca do vocabulário utilizado. Como as 3 perguntas do processo eram respondidas com texto podem haver problemas de entendimento e interpretação entre os diferentes centros de desenvolvimento. Apenas este ponto negativo foi citado pelos participantes do experimento nesta abordagem.

Apenas uma sugestão de melhoria para o *FTSProc* foi citada e não está relacionada diretamente ao processo, mas às ferramentas utilizadas. Esta sugestão refere-se à utilização de uma forma automática de *check-in* do trabalho

realizado. Desta forma, caso algum dos participantes do *shift* anterior não faça o *check-in*, evita-se que o próximo *shift* seja iniciado sem o código fonte mais recente no repositório.

Após esta breve apresentação dos resultados qualitativos e quantitativos oriundos do processo experimental, o capítulo 6 apresenta uma análise crítica sobre estes achados, juntamente com as conclusões sobre o processo experimental.

5.4.3 Lições aprendidas

Nesta seção são apresentadas as lições aprendidas com a realização do experimento, as quais poderão futuramente auxiliar pesquisadores no planejamento e execução deste método de pesquisa.

- Lição aprendida 1: escolha dos participantes

A escolha dos participantes do experimento, também chamados de sujeitos experimentais, é vital para o sucesso do processo experimental. Para a realização do experimento desta pesquisa, foi necessária a utilização de participantes que tivessem sólidos conhecimentos em desenvolvimento de sistema. Este requisito se fez necessário para evitar problemas voltados unicamente ao desenvolvimento, podendo assim, ter o foco total na avaliação do processo proposto. Devido a esta necessidade, a quantidade de possíveis participantes tornou-se menor. Ainda, a necessidade de todos os participantes estarem presentes em um mesmo momento para a execução do experimento tornou o número de sujeitos ainda menor. A ideia inicial era realizar este experimento com 12 participantes, simulando 3 centros distribuídos. Entretanto, reunir 12 pessoas em um mesmo momento não foi possível. Para contornar este problema, a solução encontrada foi utilizar 8 pessoas e dividir em 2 momentos separados. Para isto, foi realizado o experimento em 2 etapas, sendo uma com a equipe *FTSProc* e a outra com a equipe *ad hoc*. Salientamos que esta abordagem pode aumentar o risco de sucesso no experimento, pois havendo algum problema no segundo grupo, todo o experimento é invalidado e deve-se iniciar novamente,

selecionando outro problema e outros participantes. Para amenizar este risco, realizou-se a primeira etapa com a equipe *FTSProc*, pois envolvia o uso da ferramenta de apoio, a qual poderia ocasionar problemas.

- Lição aprendida 2: experiência dos participantes

Outro ponto positivo na escolha dos participantes foi a experiência dos participantes. Os alunos e profissionais escolhidos para o experimento possuíam sólidas experiências em desenvolvimento de software, bem como em projetos de DDS. Isto facilitou o inicio da execução do experimento e o tempo necessário para os treinamentos dos participantes foi reduzido. Se estes participantes fossem pessoas com pouca experiência, isto poderia impactar os resultados obtidos. O ponto negativo no uso de 8 participantes foi o fato que isto limitou o uso de cálculos estatísticos para a validação das hipóteses. Por esta razão, é interessante que a quantidade de participantes do experimento seja maior.

- Lição aprendida 3: infraestrutura necessária

A infraestrutura necessária para este tipo de experimento deve ser bem planejada. Este ponto é importante para a execução deste tipo de processo experimental. Este planejamento inicia pelos recursos computacionais necessários. Nesta etapa, atentou-se para computadores atualizados, onde fosse possível a utilização de ferramentas atuais para o desenvolvimento das atividades. Com estas informações, buscou-se por laboratórios que pudessem ser utilizados dentro da universidade. Finalmente, avaliou-se as necessidades de redes para comunicações. Ao final do planejamento da infraestrutura, optou-se por utilizar uma sala de aula, a qual foi reservada para os dois dias, conforme disponibilidade dos participantes. Nesta sala, foram instalados três computadores, previamente configurados e testados pelo Pesquisador. Destes computadores, um deles foi utilizado como servidor de arquivos (repositório SVN), e servidor web e de banco de dados da ferramenta de apoio. Os outros dois estavam configurados com as ferramentas necessárias para o desenvolvimento das atividades dos participantes. A conexão entre estes computadores foi realizada através de uma rede local, a qual estava conectada apenas estes três computadores. Desta forma, tinha-se maior controle e qualquer problema na rede da universidade não afetaria o andamento do experimento.

- Lição aprendida 4: ferramenta de apoio

Para evitar qualquer tipo de problema relacionado à ferramenta de apoio, durante o desenvolvimento da mesma, executaram-se diversos testes, incluindo a utilização por mais de um usuário simultaneamente, como ocorreria durante o experimento. Assim, foi possível identificar problemas e corrigi-los antes da execução do experimento evitando imprevistos. Esta etapa foi fundamental, já que qualquer problema identificado durante o experimento poderia invalidar o processo experimental. Um ponto negativo sobre a ferramenta está no fato da mesma ter sido desenvolvida como um protótipo e testada para a utilização do experimento. Para futuros experimentos, utilizando um número maior de participantes, ou até mesmo em empresas, seria necessário rever aspectos de usabilidade, desempenho e confiabilidade da ferramenta.

- Lição aprendida 5: requisitos do sistema fictício

Para a elaboração dos requisitos a serem utilizados neste tipo de experimento buscou-se na literatura qual seria um domínio propício. Encontrou-se em alguns trabalhos o uso de um sistema matemático, devido a sua facilidade e provável conhecimento de todos os participantes [TAW02]. Neste sentido, os requisitos foram criados dentro deste domínio (Apêndice C). Após a criação destes requisitos os mesmos foram implementados e validados pelo Pesquisador. Ainda, durante esta etapa, foram criados os testes para serem utilizados para o critério de cálculo dos requisitos implementados. O esforço empregado nesta etapa de elaboração dos requisitos foi um fator positivo, já que não se encontrou nenhum tipo de problema durante a execução do experimento, relacionado aos requisitos do sistema fictício.

- Lição aprendida 6: disponibilidade dos materiais

Finalmente, outro ponto positivo durante o experimento, foi o fato de utilizar o que foi chamado de *pacote experimental*. Este *pacote* era composto por todos os materiais necessários para cada um dos participantes durante toda a execução do experimento. Estes materiais foram impressos para facilitar a consulta por parte dos participantes. Estes *pacotes* eram compostos por:

- Termo de consentimento para ser assinado por cada participante (Apêndice H);

- Documento de requisitos (Apêndice C);

- Manual da Ferramenta de Apoio, somente para os participantes da equipe *FTSProc* (Apêndice I);

- Questionário pós-experimento (Apêndice G);

- Dados para acesso as ferramentas, contendo usuário e senha para cada participante (SVN e Ferramenta de apoio);

Este *pacote* entregue para cada um dos participantes foi apontado pelos mesmos como um facilitador durante a execução do experimento.

O próximo capítulo apresenta uma análise crítica sobre os resultados obtidos a partir do processo experimental realizado. Essa análise é uma complementação da análise preliminar realizada sobre os resultados qualitativos e quantitativos do experimento.

6 ANÁLISE CRÍTICA

Após apresentar os resultados obtidos e os indícios para a confirmação da hipótese H_1, a qual mostra que um projeto que utiliza o *FTSProc* é mais eficiente que um projeto executado de forma *ad hoc*, esta seção apresenta mais alguns fatores que corroboram para este resultado.

Analisando os resultados qualitativos obtidos, verificamos que parte da vantagem do *FTSProc* está no fato de que a equipe *FTSProc* percebe de forma clara e rápida como o trabalho deve ser continuado. Assim, o tempo destinado nesta identificação é menor do que a equipe *ad hoc*, resultando em um tempo de desenvolvimento de requisitos maior durante cada dia de trabalho (*shift*). Esta vantagem deve-se a dois fatores principais, relacionados ao *FTSProc*: utilização de TDD e às três perguntas principais que o processo propõe.

A utilização do TDD é eficaz, pois a equipe *ad hoc* implementou 8 requisitos parcialmente, ou seja, alguns critérios de aceitação não estavam cobertos. Em um projeto real, estes problemas seriam identificados posteriormente, apenas em uma fase de testes. Pelo fato de não utilizar TDD, esta equipe deveria além de implementar os requisitos, testá-los. Neste sentido, a falta desta técnica prejudica também no tempo necessário para a implementação. Enquanto a equipe *FTSProc* tinha os testes unitários para garantir a correta implementação dos requisitos, a equipe *ad hoc* investia parte do seu tempo criando e executando testes. Outro fator que o TDD auxilia é na percepção do progresso do trabalho, pois verificar os testes já cobertos e os que ainda devem ser trabalhados, facilita o entendimento sobre o progresso do trabalho, ou seja, o quanto ainda falta para finalizar todos os requisitos. Por esta razão, durante o experimento, em apenas sete minutos do quarto *shift*, a equipe *FTSProc* identificou que todo o trabalho havia sido realizado, e não existia mais trabalho a ser continuado.

As três perguntas principais que o processo propõe (*(i)* O que foi realizado durante este período de trabalho?, *(ii)* O que deve ser continuado no próximo período de trabalho?, *(iii)* Existe algo bloqueando a equipe?) também auxiliaram as equipes na identificação de onde o trabalho deveria continuar. Facilmente, os

participantes verificavam as respostas disponibilizadas pelo centro anterior e rapidamente sabiam o que havia sido implementado. Após esta leitura, o ponto para iniciar o desenvolvimento era confirmado com a utilização dos testes unitários e, a partir destes, as tarefas eram distribuídas na equipe. Desta forma, em pouco tempo, a equipe iniciava o trabalho de desenvolvimento.

Enquanto a equipe *FTSProc* de forma rápida e direta identificava como o trabalho deveria ser continuado, diferentemente, a equipe *ad hoc* não tinha esta percepção. Para identificar o ponto que o trabalho deveria ser continuado, a equipe *ad hoc* precisava analisar grande parte do código fonte criado ou modificado pela equipe anterior, o que despende muito tempo. Como não havia tempo de trabalho simultâneo para a transferência e não havia a ferramenta de apoio, a equipe *ad hoc* utilizou comentários no código fonte e nos *check-ins* do repositório para informar o que havia sido realizado. Entretanto, por não haver uma estrutura definida para passar esta informação e nem uma obrigação, não eram todos os participantes que utilizaram este recurso e, os que usavam, não seguiam um padrão definido. Ao final do experimento, estes comentários foram citados como pontos positivos. Mais uma vez, este fato vai ao encontro do que o *FTSProc* propõe para facilitar a transferência de trabalho, com a utilização das três perguntas.

Analisando os pontos negativos apontados pelas equipes (seção 5.4.2), notamos que o projeto realizado de forma *ad hoc* apontou diversos problemas. Dentre estes problemas, notamos que a maior parte deles foram originados pela falta de um padrão na transferência de trabalho de um centro ao outro. Notamos que diversos problemas apontados pela equipe *ad hoc* não foram identificados na equipe *FTSProc*. Este fato é mais um indício que o processo criado, de fato facilita a transferência de trabalho. Ainda, ao analisar as sugestões de melhorias para a equipe que não utilizou o *FTSProc* verificamos que há indicações para utilizar técnicas e práticas que o *FTSProc* já utiliza, tais como: testes unitários, *Test-driven development* (TDD), uma forma de informar o que foi realizado e a definição de um sistema de sincronização de tarefas.

Ao analisarmos os pontos negativos apontados na equipe *FTSProc* notamos que os poucos pontos citados, são relacionados não ao processo em si, mas às dificuldades encontradas em qualquer projeto, mesmo os que são

desenvolvidos de forma co-localizados. A dependência da ferramenta, apontada como um dos pontos negativos, está relacionada ao fato da ferramenta de apoio desenvolvida ser responsável pelo controle do fluxo de trabalho, então, qualquer problema na ferramenta pode prejudicar o andamento do projeto. Outro ponto levantado está no fato de não haver a definição de um “vocabulário” único. Neste caso, pode haver problemas de interpretação, já que as perguntas são respondidas através de texto livre. O último ponto levantado mostra que problemas gerados afetam o time como um todo. Neste sentido, notamos que este problema está relacionado ao fato do trabalho ser continuado *shift* após *shift,* ou seja, um defeito gerado e não concertado é transferido para o próximo *site*.

Esta análise qualitativa nos permitiu identificar que o *FTSProc* apresentou diversos pontos positivos, como a utilização das 3 perguntas base do processo e utilização de TDD. Ainda, a análise dos resultados do experimento e do questionário apresenta indícios da maior eficiência de projetos que utilizam o *FTSProc* em relação aos projetos executados de forma *ad hoc*.

Devido ao fato dos resultados do experimento realizado terem sido amplamente favoráveis ao processo proposto, optamos por não fazer modificações no processo proposto. Portanto, o *FTSProc* descrito no capítulo 4 é considerado a versão final do processo desenvolvido neste trabalho.

Entretanto, por este ser um estudo exploratório em uma nova temática de pesquisa, o mesmo não pode ser considerado um trabalho final nesta área. Portanto, apesar dos resultados favoráveis encontrados durante o desenvolvimento desta pesquisa foram identificadas algumas limitações, como por exemplo, a falta de um experimento com um número maior de participantes possibilitando assim, uma análise estatística dos resultados. Ainda, durante esta pesquisa foram identificados ainda alguns trabalhos futuros, como por exemplo, a necessidade de expandir este tipo de pesquisa para outras fases do desenvolvimento de software. O capítulo 7 apresenta detalhadamente estas informações nas sessões de limitações do trabalho e estudos futuros.

7 CONSIDERAÇÕES FINAIS

Ao final deste estudo, objetivos inicialmente propostos para este trabalho foram alcançados. Conforme abordado nos capítulos 4 e 5, o objetivo geral desta pesquisa, o qual sugeria criação de um processo de transferência de trabalho (*hand-off*), foi atingido. Este processo proposto, denominado *FTSProc*, é considerado a principal contribuição desta pesquisa.

O objetivo específico de complementar os estudos da base teórica, focando nas temáticas principais desta pesquisa (DDS e FTS) também foi atingido, conforme apresentado no capítulo 2. Este aprofundamento da base teórica foi utilizado para alcançar outro objetivo específico, o qual era propor um processo preliminar de transferência de trabalho durante a fase de desenvolvimento do ciclo de vida. Este processo foi denominado *FTSProc* e está detalhadamente apresentado no capítulo 4 portanto, este objetivo específico foi atingido.

Outro objetivo específico proposto no inicio desta pesquisa foi a criação de uma ferramenta de apoio ao processo criado. Após a criação do processo, foi desenvolvida uma ferramenta de apoio, descrita na seção 4.1, portanto, este objetivo específico foi atingido.

Após definir o processo e desenvolver a ferramenta de apoio, foi realizado um experimento para avaliar a eficiência do processo criado. Os resultados obtidos neste experimento foram favoráveis ao *FTSProc*. Conforme apresentado nos capítulos 5 e 6, os resultados oriundos do experimento fornecem fortes indícios de que projetos que utilizam o *FTSProc* tem maior eficiência. Através do experimento, verificou-se também que o processo alcança os seus objetivos iniciais, os quais facilitam a adoção da estratégia FTS em projetos de desenvolvimento de software.

7.1 CONTRIBUIÇÕES

As contribuições deste estudo situam-se em três dimensões: para a teoria, para o mercado e para o pesquisador:

Para a teoria da área, a principal contribuição desta pesquisa foi a criação de um processo para a transferência de trabalho, durante a fase de desenvolvimento. Conforme apresentado no capítulo 4, o processo foi proposto com dados oriundos da base teórica. Conforme podemos verificar na Tabela 1, o objetivo principal do FTS é a redução do tempo de desenvolvimento de um projeto. Já a Tabela 3 apresenta como os trabalhos relacionados buscam alcançar esta redução. Com base nestas comparações e nos resultados obtidos, podemos verificar que o *FTSProc* consegue reduzir o tempo de desenvolvimento, utilizando diversas contribuições de diferentes trabalhos relacionados. Após o experimento, descrito no capítulo 5, e os seus resultados descritos no capítulo 6, verificamos que o mesmo mostrou-se eficaz. Este ponto é outra contribuição importante para a teoria, pois enquanto a literatura ainda não apresenta um processo específico para a fase de desenvolvimento, neste estudo foi proposto um processo, realizado um experimento, e apontado indícios da eficácia do processo proposto. Além disto, como contribuição deste trabalho pode-se citar a avaliação do processo *FTSProc* através de um método experimental em um cenário de DDS, identificando os benefícios de sua utilização e ainda, a descrição detalhada, lições aprendidas e disponibilização da instrumentação do experimento realizado, permitindo que o mesmo seja replicado para novas avaliações do processo. Ainda, para a teoria, outra contribuição está no fato da estratégia FTS ser uma área recente de estudo na engenharia de software, havendo poucos estudos publicados sobre esta temática. Portanto, esta pesquisa é um avanço nesta área de estudo. Finalmente, outra contribuição desta pesquisa para a teoria foi a criação de uma definição para este conceito, o qual está apresentado na seção 2.2.1. Cabe salientar ainda, que esta definição foi tema de um artigo publicado no decorrer desta pesquisa [KRO11].

Para o mercado, notamos que atualmente, na busca de vantagens competitivas, como a diminuição de custo e o ganho de produtividade, as

indústrias estão realizando operações *offshore*. Neste sentido, este trabalho pode contribuir com o aumento do ganho de produtividade, já que o processo criado facilita o uso da estratégia FTS durante a fase de desenvolvimento, diminuindo assim, o tempo gasto durante esta fase do ciclo de vida. Portanto, acredita-se que o processo proposto seja um ponto de partida para que as organizações que trabalham de forma distribuída possam iniciar a utilização desta estratégia. Ainda, além do processo, considera-se também a ferramenta de apoio que, mesmo ainda sendo um protótipo, uma importante contribuição do ponto de vista prático, já que a mesma oferece um importante conjunto de funcionalidades.

Para o pesquisador, este trabalho proporcionou a oportunidade de pesquisar uma área de estudo até então desconhecida: a estratégia FTS. Por outro lado, possibilitou o aprofundamento nos conhecimentos na área de DDS. Com o conhecimento adquirido durante o desenvolvimento desta pesquisa será possível dar continuidade aos estudos na área acadêmica ou ainda, aplicar os conhecimentos obtidos durante esta pesquisa no aspecto profissional.

7.2 LIMITAÇÕES DO TRABALHO

A primeira limitação deste trabalho está no tempo disponível da pesquisa. Conforme o planejamento inicial, o tema desta pesquisa era a utilização de Arquitetura Orientada a Serviço (SOA) em ambientes de DDS. Entretanto, ao final do primeiro semestre, houve a troca do tema para a estratégia FTS. Desta forma, foi necessário refazer grande parte do estudo da base teórica principalmente na temática da estratégia FTS e por esta razão, foi perdido aproximadamente 25% do tempo disponível. Portanto, apesar dos objetivos terem sido atingidos de forma satisfatória, acredita-se que a pesquisa poderia ter avançado ainda mais.

Outras limitações deste trabalho estão diretamente relacionadas a avaliação utilizada do processo *FTSProc*. No experimento realizado obteve-se um baixo número de participantes, o que impossibilitou a utilização de métodos estatísticos para a comprovação das hipóteses, optando-se então, por uma interpretação de base qualitativa para analisar os resultados obtidos. Esta interpretação apresentou indícios da maior eficiência do projeto que utilizou o

FTSProc, porém não permitiu a obtenção de conclusões com um grau de confiança significativo, o que é alcançado através do uso de experimentos com análise estatística dos resultados.

Além disto, como restrições deste trabalho, têm-se a generalização do experimento considerada específica, devido ao fato do escopo do projeto ser fictício e criado pelo Pesquisador. Ainda, têm-se as questões pertinentes a aplicação de um método de pesquisa experimental, como a influência subjetiva do pesquisador ou dos participantes nos resultados.

A ferramenta de apoio criada para o *FTSProc* ainda é um protótipo e portanto, pode ser considerada uma das limitações desta pesquisa. Apesar de ter sido implementado todos os requisitos planejado, antes de utilizar a ferramenta em um ambiente real, seria necessário rever, aspectos de usabilidade, desempenho e confiabilidade da ferramenta.

7.3 ESTUDOS FUTUROS

A interpretação dos resultados do experimento apresentaram indícios favoráveis para projetos que utilizam o *FTSProc*, indicando que ele é mais eficiente para projetos distribuídos que utilizam a estratégia FTS do que projetos *ad hoc*, comumente utilizado nas empresas. Desta maneira, com o finalidade de comprovar os indícios apresentados pelo método experimental, sugere-se como estudos futuros a replicação do experimento com uma amostra maior para avaliar o *FTSProc*, a qual permita uma validação estatística significativa para a obtenção de conclusões sobre as hipóteses.

Ainda, é relevante a realização de um estudo de caso para avaliar a utilização do processo criado em um ambiente real, utilizando um projeto e uma equipe real em uma empresa que utiliza o desenvolvimento distribuído de software. Desta forma, será possível verificar o comportamento do processo neste tipo de ambiente. Assim pode-se comprovar se os resultados encontrados nesta pesquisa, através de um experimento controlado, são equivalentes em um ambiente real.

Finalmente, trabalhos com o objetivo de expandir este processo para outras fases dos projetos de software são relevantes. Desta forma, outras fases do SDLC poderiam ser contempladas com a criação de um processo para facilitar o uso da estratégia FTS. Focando em fases especificas de diferentes formas, ao final, pode-se criar um processo composto por diversos sub-processos, os quais contemplariam todas as fases do SDLC. Desta forma, todo o projeto de software poderia ser realizado utilizando a estratégia FTS e, assim, reduzindo o tempo de construção em todas as fases de um projeto.

7.4 PUBLICAÇÕES

Outro objetivo específico desta pesquisa inicialmente planejado foi a submissão de artigos científicos no decorrer do trabalho. Este objetivo foi atingido, pois no decorrer da pesquisa, foram produzidos os seguintes artigos:

- Artigo Publicado – Researching into Follow-the-Sun Software Development: Challenges and Opportunities
    - Local de publicação: *Global Software Engineering (ICGSE '11). IEEE Computer Society*, Finlândia.
    - Nesse artigo é descrito a necessidade da criação de um processo de transferência de trabalho para projetos que utilizam a estratégia FTS; Proposta de uma definição para o conceito FTS.

- Artigo Publicado – *Follow-the-Sun*: Um Processo para Minimizar as Dificuldades de Projetos que Adotam esta Estratégia
    - Local de publicação: V Workshop de Desenvolvimento Distribuído de Software – WDDS 2011 - CBSoft, São Paulo;
    - Nesse artigo é apresentado o processo preliminar proposto para a transferência de trabalho para projetos que utilizam a estratégia FTS, e demonstra a necessidade da realização de um experimento para avaliar a eficiência do processo.

- Artigo aceito como pôster – Challenges in the Follow-the-Sun strategy: How to alleviate them.
    - Local de Publicação: *AMCIS 2011 Proceedings - All Submissions,* Detroit, EUA.
    - Neste artigo está descrito algumas dificuldades que a estratégia FTS apresenta, e formas para amenizar estas dificuldades, incluindo a ideia principal do processo proposto.

REFERÊNCIAS BIBLIOGRÁFICAS

[AUD07] Audy, J.; Prikladnicki, R. “Desenvolvimento Distribuído de Software – Desenvolvimento de Software com Equipes Distribuídas”. Campus, 2007, 232p.

[BAS94] Basili, V. R.; Caldiera, G.; Rombach, H. D. “The Goal Question Metric Approach: Encyclopedia of Software Engineering”. Wiley-Interscience, 1994, 578p.

[BIN07] Bin, X. “A Service Oriented Model for Role Based Global Cooperative Software Development". In: International Conference Convergence Information Technology, 2007, pp. 376-381.

[BOS10] Bosch, J.; Bosch-Sijtsema, P. "Software Product Lines, Global Development and Ecosystems: Collaboration in Software Engineering". In: Collaborative Software Engineering, 2010, pp. 77-92.

[CAR09] Carmel, E.; Espinosa, A.; Dubinsky, Y. "Follow The Sun Software Development: New Perspectives, Conceptual Foundation, and Exploratory Field Study". In: 42nd Hawaii International Conference on System Sciences, 2009, pp. 1-9.

[CAR10] Carmel, E.; Espinosa, A.; Dubinsky, Y. "Follow the Sun: Workflow in Global Software Development: Conceptual Foundations". *Journal of Management Information Systems*, vol. 27-1, Julho 2010, pp. 17-38.

[CAR11] Carmel, E.; Espinosa, A. “I'm Working While They're Sleeping: Time Zone Separation Challenges and Solutions”. Nedder Stream Press, 2011, 188p.

[CAR99] Carmel, E. "Global Software Teams: collaborating across borders and time zones". Prentice Hall PTR, 1999, 269p.

[DAM06] Damian, D.; Moitra, D. "Guest Editors' Introduction: Global Software Development: How Far Have We Come?". *IEEE Software*, vol. 23-5, Set-Out 2006, pp. 17-19.

[DEN08] Denny, N.; Nadella, R. S.; Samdal, J.; Gupta, A. “Hybrid Offshoring Composite Personae and Evolving Collaboration Technologies”. *Information Resources Management Journal*, vol. 21-1, Fevereiro 2008, pp. 89-104.

[DEN09] Denny, N.; Crk, I.; Nadella, R. S.; Gupta, A. “Agile Software Processes for the 24-Hour Knowledge Factory Environment”. *Journal of Information Technology Research*, vol. 1-1, Setembro 2009, pp. 57-71.

[FAD00] Fadel, G. M.; Lindemann, U.; Anderl, R. “Multi-National Around the Clock Collaborative Senior Design Project”. capturado em: http://www.asme.org/education/enged/awards/cia00/fadel.pdf, Junho 2011.

[GOR96] Gorton, I.; Motwani, S. “Issues in co-operative software engineering using globally distributed teams”. *Journal of Information and Software Technology*, vol. 38-10, Agosto 1996, pp. 647-655.

[GUP09a] Gupta, A. “Deriving mutual benefits from offshore outsourcing”. *Communications of the ACM*, vol. 52-6, Junho 2009, pp. 122-126.

[GUP09b] Gupta, A.; Mattarelli, E.; Seshasai, S.; Broschak, J. “Use of collaborative technologies and knowledge sharing in co-located and distributed teams: Towards the 24-h knowledge factory”. *The Journal of Strategic Information Systems*, vol.18-3, Setembro 2009, pp. 147-161.

[HAU06] Haugen, N. C. "An empirical study of using planning poker for user story estimation". In: Agile Conference, 2006, pp. 9-34.

[HER01] Herbsleb, J. D.; Mockus, A.; Finholt, T. A.; Grinter, R. E. "An empirical study of global software development: distance and speed". In: International Conference on Software Engineering, 2001, pp. 81-90.

[HER99] Herbsleb, J. D.; Grinter, R. E. "Architectures, coordination, and distance: Conway's law and beyond". *IEEE Software*, vol. 16-5, Setembro 1999, pp. 63-70.

[HOL06] Holmstrom, H.; O'Conchuir, E.; Agerfalk, P. J.; Fitzgerald, B. "Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance". In: International Conference on Global Software Engineering, 2006, pp. 3-11.

[HUM95] Humphrey, W. S. "Introducing the personal software process". *Software Engineering*, vol. 1-1, Abril 1995, pp. 311-325.

[JAL04] Jalote, P.; Jain, G. "Assigning tasks in a 24-hour software development model". In: 11th Asia-Pacific Software Engineering Conference, 2004, pp. 904-911.

[JUR01] Juristo, N. "Basics of Software Engineering Experimentation". Kluwer Academic Publishers, 2001, 101p.

[KAR98] Karolak, D. W. "Global Software Development - Managing Virtual Teams and Environments". IEEE Computer Society, 1998, 159p.

[KNO07] Knob, F. F. "RiskFree4ppm: uma proposta de processo para o gerenciamento de portfólios de projetos distribuídos". Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2007, 185p.

[KRO11] Kroll, J.; Hess, E. R.; Audy, J. L. N.; Prikladnicki, R. "Researching into Follow-the-Sun Software Development: Challenges and Opportunities". In: International conference on Global Software Engineering, 2011, pp. 60-65.

[LAN08] Lane, M.; Agerfalk, P. "On the Suitability of Particular Software Development Roles to Global Software Development". In: International Congress on Global Software Engineering, 2008, pp. 3-12.

[LIN07] Lings, B.; Lundell, B.; Ågerfalk, P. J.; Fitzgerald B. "A reference model for successful Distributed Development of Software Systems". In: International Congress on Global Software Engineering, 2007, pp. 130-139.

[LOP04] Lopes, L. "Um Modelo de Processo de Engenharia de Requisitos para Ambientes de Desenvolvimento Distribuído de Software". Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2004, 142p.

[MAR09] Martignoni, R. "Global Sourcing of Software Development - A Review of Tools and Services". In: International Congress on Global Software Engineering, 2009, pp. 303-308.

[MEZ06] Meszaros, G. "XUnit Test Patterns: Refactoring Test Code". Prentice Hall PTR, 2006, 883p.

[OSH07] Oshri, I.; Kotlarsky, J.; Willcocks, L. P. "Global Software Development: Exploring Socialization in Distributed Strategic Projects". *Journal of Strategic Information Systems*, vol. 16-1, Outubro 2007, pp. 25-49.

[PIL06] Pilatti, L. S. M. "Estrutura e Características para Análise de Ambientes de Desenvolvimento Global de Software em Organizações Offshore Insourcing". Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2006, 136p.

[PRI08] Prikladnicki, R.; Damian, D.; Audy, J. "Patterns of Evolution in the Practice of Distributed Software Development in Wholly Owned Subsidiaries: A Preliminary Capability Model". In: International Congress on Global Software Engineering, 2008, pp. 99-108.

[PRI09] Prikladnicki, R. “Padrões de Evolução na Prática de Desenvolvimento de Software em Ambientes de Internal Offshoring: Um Modelo de Capacidade”. Tese de Doutorado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2009, 237p.

[SCH04] Schwaber, K. “Agile Project Management with Scrum”. Microsoft Press, 2004, 163p.

[SET07] Setamanit, S.O.; Wakeland, W.; Raffo, D. "Improving Global Software Development Project Performance Using Simulation". In: Management of Engineering and Technology, 2007, pp. 2458-2466.

[SOL10] van Solingen, R.; Valkema, M. "The Impact of Number of Sites in a Follow the Sun Setting on the Actual and Perceived Working Speed and Accuracy: A Controlled Experiment". In: International Congress on Global Software Engineering, 2010, pp. 165-174.

[SOO08] Sooraj, P.; Mohapatra, P. K. J. "Modeling the 24-h software development process". *Strategic Outsourcing: An International Journal*, vol. 1-2, Março 2008, pp. 122-141.

[TAW02] Taweel, A.; Brereton, P. “Developing Software Across Time Zones: An Exploratory Empirical Study”. *Journal of Information and Software Technology*, vol. 48-1, Agosto 2002, pp. 1-11.

[TRA02] Travassos, G.H.; Gurov, D.; Amaral, G. “Introdução a Engenharia de Software Experimental”. Relatório Técnico, Programa de Engenharia de Sistemas e Computação, UFRJ, 2002, 66p.

[TRE06] Treinen, J. J.; Miller-Frost, S. L. “Following the Sun: Case Studies in Global Software Development”. *IBM Systems Journal*, vol. 45-4, Outubro 2006, pp. 773-783.

[VIS09] Visser, C. "Connecting Time Zones and Languages in Follow-the-Sun: a routing model". Relatório Técnico, Department of Software Technology, EEMCS, 2009, 56p.

[WOH00] Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A. “Experimentation in Software Engineering: An introduction”. Kluwer Academic Publishers, 2000, 204p.

[YAP05] Yap, M. "Follow the sun: distributed extreme programming development". In: Agile Conference, 2005, pp. 218-224.

APÊNDICE A – Artefato 1: Formulário de *hand-off* do *FTSProc*

Site:	<identificador do site>		
Desenvolvedor:	<nome desenvolvedor>		
Data:	dd/mm/yyyy	Hora	hh:mm
Informe os seguintes dados para que a próxima equipe possa continuar o trabalho:			
O que foi realizado durante este período de trabalho?			
O que deve ser continuado no próximo período de trabalho?			
Existe algo bloqueando a equipe?			
Lista de Testes designados, que estão concluídos:			
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			

APÊNDICE B – Artefato 2: Relatório de testes unitários do *FTSProc*

Relatório de Testes Unitários – Gerado em: dd/mm/yyyy – hh:mm
Todos os Testes Cobertos: NÃO
Número Total de Testes: 33
Número Total de Testes não Cobertos: 6
Lista de Testes não Cobertos:
Id - <nomeTeste>(classe de teste)
1 - testMulPass(com.test1.calculos.SomasTest) 2 - testMulFail(com.test1.calculos.SomasTest) 3 - testDivPass(com.test1.calculos.SomasTest) 4 - testDivFail(com.test1.calculos.SomasTest) 5 - testMinusPass(com.test1.calculos.SomasTest) 6 - testMinusFail(com.test1.calculos.SomasTest)

APÊNDICE C – Documento de requisitos utilizados no processo experimental

Requisitos da Unidade Experimental

Programa de Pós Graduação em Ciência da Computação

Pesquisador: Estevão Ricardo Hess

Experimento Follow-the-Sun

Porto Alegre, 2011

Requisitos da Unidade Experimental

Conteúdo Deste Documento

1. Descrição das funcionalidades a serem desenvolvidas 3
2. Informações sobre os casos de usos (requisitos) deste módulo 4
a. Nome do Caso de Uso 4
3. Casos de Uso do Projeto 5
3.1. Diagrama de Casos de Uso da Aplicação 5
a. Somatório de Valores 6
b. Subtração de Valores 7
c. Multiplicação de Valores 8
d. Divisão de Valores 9
e. Resto da Divisão 11
f. Fatorial de um número 13
g. Raiz quadrada de um número 14
h. Potência 15
i. Área de círculo 16
j. Volume de esfera 17
k. Área de retângulo 18
l. Volume de um cubo 19
4. Design da aplicação 20
4.1 Diagrama de Classes do Sistema 20
5. Mapeamento Requisitos X Casses/Métodos 21
6. Classes utilitárias e suas funções 22

1. Descrição das funcionalidades a serem desenvolvidas

- Dados do projeto em desenvolvimento
  - Nome do projeto: Calculadora
  - Módulo a ser desenvolvido: *BackEnd Layer*
  - Objetivos do Módulo
    - i. Criar métodos background para serem utilizado por aplicações externas;
    - ii. Este módulo é responsável apenas por realizar cálculos matemáticos;
  - O que está no escopo do módulo
    - i. Apenas a implementação das operações descritas nos requisitos
    - ii. Nada mais deve ser implementado.
  - O que está fora do escopo do módulo
    - i. Criação de interface com o usuário
      1. Telas
      2. Linhas de comando
  - Linguagem de programação a ser utilizada: Java 1.6

2. Informações sobre os casos de usos (requisitos) deste módulo

a) A implementação deve seguir o padrão de classes e métodos definidos no design da aplicação – Item 4.
b) Todos os atores envolvidos nos casos de uso são usuários externos. Portanto, não serão explicitado em cada caso de uso.
c) A descrição dos requisitos seguirá o seguinte padrão:

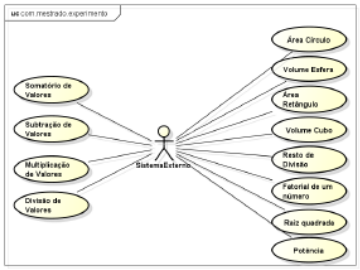
a. Nome do Caso de Uso

ID: XX	Título do caso de uso
Observações	Informação adicional necessária para a implementação do requisito.
Descrição	Breve texto descrevendo o caso de uso. Breve texto descrevendo o caso de uso. Breve texto descrevendo o caso de uso. Breve texto descrevendo o caso de uso. Breve texto descrevendo o caso de uso. Breve texto descrevendo o caso de uso. Breve texto descrevendo o caso de uso.
Exemplos de entrada e saída	Exemplos para entrada e saída, conforme a entrada.
Critérios de aceitação	Além da descrição, este quadro mostra critérios extras de aceitação

3. Casos de Uso do Projeto

a. Somatório de Valores 6
b. Subtração de Valores 7
c. Multiplicação de Valores 8
d. Divisão de Valores 9
e. Resto da Divisão 11
f. Fatorial de um número 13
g. Raiz quadrada de um número 14
h. Potência 15
i. Área de círculo 16
j. Volume de esfera 17
k. Área de retângulo 18
l. Volume de um cubo 19

3.1. Diagrama de Casos de Uso da Aplicação



a. Somatório de Valores

ID: 01	GENERIC: Soma de Valores
Observações	Seguir o tipo de entrada e saída definidos no design.
Descrição	Recebe um conjunto de valores, e retorna o valor da soma de todos eles;
Exemplos de entrada e saída	List(X,Y) = X+Y List(Y,X) = Y+X List(Y,X,Z) = Y+X+Z
Critérios de aceitação	1. Se lista for vazia, gera um NotCorretParametersException("Lista Vazia."); 2. Se lista contiver apenas um valor, retorna este valor;

b. Subtração de Valores

ID: 02	GENERIC: Subtração de Valores
Observações	Seguir o tipo de entrada e saída definidos no design.
Descrição	Recebe um conjunto de valores, e retorna o valor da subtração de todos eles, do inicio para o final da lista;
Exemplos de entrada e saída	List(X,Y) = X-Y List(Y,X) = Y-X List(Y,X, Z) = Y-X-Z
Critérios de aceitação	1. Se lista for vazia, gera um NotCorretParametersException("Lista Vazia."); 2. Se lista contiver apenas um valor, retorna este valor;

c. Multiplicação de Valores

ID: 03	GENERIC: Multiplicação de Valores
Observações	Seguir o tipo de entrada e saída definidos no design.
Descrição	Recebe um conjunto de valores, e retorna o valor da multiplicação de todos eles;
Exemplos de entrada e saída	List(X,Y) = X*Y List(Y,X) = Y*X List(Y,X,Z) = Y*X*Z
Critérios de aceitação	1. Se lista for vazia, gera um NotCorretParametersException("Lista Vazia."); 2. Se lista contiver apenas um valor, retorna este valor;

d. Divisão de Valores

ID: 04	GENERIC: Divisão de Valores
Observações	Seguir o tipo de entrada e saída definidos no design.
Descrição	Recebe um conjunto com 2 valores, e retorna o valor da divisão do primeiro pelo segundo;
Exemplos de entrada e saída	List(X, Y) = X/Y List(Y,X) = Y/X
Critérios de aceitação	1. Se lista for vazia, gera um NotCorrectParametersException("Lista Vazia.");
	2. Caso um único operador seja passado, lançar a exceção NotCorretParametersException ("Número de parâmetros insuficientes. Esperado 2.");
	3. Caso mais de 2 operadores sejam passados, lançar a exceção NotCorretParametersException("Número de parâmetros maior que o esperado: 2.");
	4. Caso o segundo operador seja 0 (zero), lançar exceção NotCorrectParametersException("Segundo

9

	operador não pode ser zero.");
	5. Caso o primeiro operador seja 0 (zero), retorne 0 (zero);

10

e. Resto da Divisão

ID: 05	GENERIC: Resto de uma divisão
Observações	Seguir o tipo de entrada e saída definidos no design.
Descrição	Recebe um conjunto com 2 valores, e retorna o valor do resto da divisão do primeiro pelo segundo;
Exemplos de entrada e saída	4%2 = 0 5%2 = 1 10%5 = 0 50%32 = 18 -10%4 = -2
Critérios de aceitação	1. Se lista for vazia, gera um NotCorrectParametersException("Lista Vazia.");
	2. Caso um único operador seja passado, lançar a exceção NotCorretParametersException ("Número de parâmetros insuficientes. Esperado 2.");
	3. Caso mais de 2 operadores sejam passados, lançar a exceção NotCorretParametersException("Número de parâmetros maior que o esperado: 2.");
	4. Caso o segundo operador seja 0 (zero),

11

	lançar exceção NotCorrectParametersException("Segundo operador não pode ser zero.");
	5. Caso o primeiro operador seja 0 (zero), retorne 0 (zero);

12

f. Fatorial de um número

ID: 06	COMPLEX: Fatorial de um Número
Observações	Seguir o tipo de entrada e saída definidos no design.
Descrição	Recebe um valor e retorna o fatorial deste número;
Exemplos de entrada e saída	3! = 3*2*1 = 6 5! = 5*4*3*2*1 = 120
Critérios de aceitação	1. Recebe um valor do tipo inteiro; 2. O retorno deve ser do tipo long, para evitar problemas de overflow; 3. Se valor for negativo, gera um NotCorretParametersException ("Parâmetro não pode ser negativo."); 4. Se valor informado for 0 (zero), o resultado deve ser 1;

13

g. Raiz quadrada de um número

ID: 07	COMPLEX: Raiz quadrada de um Número
Observações	Seguir o tipo de entrada e saída definidos no design.
Descrição	Recebe um valor, e retorna a raiz quadrada deste valor;
Exemplos de entrada e saída	Sqrt(4) = 2 Sqrt(25) = 5 Sqrt(10) = 3,16
Critérios de aceitação	1. Recebe um valor do tipo inteiro; 2. O retorno deve ser do tipo double; 3. Se valor for negativo, gera um NotCorrectParametersException("Parâmetro não pode ser negativo."); 4. Se valor informado for 0 (zero), o resultado deve ser 1;

14

h. Potência

ID: 08	COMPLEX: Potência
Observações	Seguir o tipo de entrada e saída definidos no design.
Descrição	Recebe um valor, e retorna a raiz quadrada deste valor;
Exemplos de entrada e saída	Power(3,4) = 3^4 = 3*3*3*3 = 81 Power(2,2) = 2^2 = 2*2 = 4 Power(5,4) = 5^4 = 5*5*5*5 = 625
Critérios de aceitação	1. Recebe 2 valores inteiros, e retorna o primeiro elemento elevado ao segundo. 2. O retorno deve ser do tipo Double; 3. Se valor do segundo parâmetro informado for 0 (zero), o resultado deve ser 1; 4. Se valor do primeiro parâmetro informado for 0 (zero), o resultado deve ser 0;

15

i. Área de círculo

ID: 09	GEOMETRIC: Área de um círculo
Observações	• Seguir o tipo de entrada e saída definidos no design. • Fórmula a ser utilizada **areaCirc** = πr^2 • Usar π=3,14
Descrição	Recebe um valor, representando o Raio de um círculo, e retorna o cálculo da área do círculo.
Exemplos de entrada e saída	areaCirc(raio); areaCirc(3) = 3^2*3,14 = 28,26 areaCirc(2) = 2^2*3,14 = 12,56 areaCirc(5) = 5^2*3,14 = 78,5
Critérios de aceitação	1. Valor do retorno deve ser do tipo Double; 2. Se valor for menor ou igual a 0, gera um NotCorrectParametersException("O parâmetro não pode ser negativo nem ZERO.");

16

j. Volume de esfera

ID: 10	GEOMETRIC: Volume de uma esfera
Observações	• Seguir o tipo de entrada e saída definidos no design. • Fórmula a ser utilizada: $volumeEsf = \frac{4}{3} * (\pi r^3)$ • Usar π=3,14
Descrição	Recebe um valor, representando o Raio de uma esfera, e retorna o cálculo do volume de uma esfera;
Exemplos de entrada e saída	volumeEsf (raio); volumeEsf (3) = (4/3)*(3,14*3³) = 113,04 volumeEsf (2) = (4/3)*(3,14*2³) = 33,49333 volumeEsf (5) = (4/3)*(3,14*5³) = 525,3333
Critérios de aceitação	1. Valor do retorno deve ser do tipo Double; 2. Se valor for menor ou igual a 0, gera um NotCorrectParametersException("O parêmetro não pode ser negativo nem ZERO.");

17

k. Área de retângulo

ID: 11	GEOMETRIC: Área de um retângulo
Observações	Seguir o tipo de entrada e saída definidos no design.
Descrição	Recebe 2 valores, representando base e altura, e retorna a multiplicação dos valores, representando a área.
Exemplos de entrada e saída	retArea(largura, comprimento); retArea (3,4) = 3*4 = 12 retArea (2,2) = 2*2 = 4 retArea (5,4) = 5*4 = 20
Critérios de aceitação	1. Retorno deve ser do tipo Double; 2. Se base ou altura for menor ou igual a ZERO, gera um NotCorrectParametersException("Nenhum parâmetro pode ser menor ou igual a 0.");

18

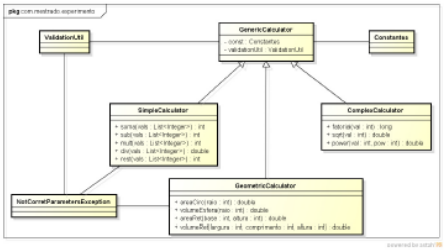
l. Volume de um cubo

ID: 12	GEOMETRIC: Volume de um cubo
Observações	• Seguir o tipo de entrada e saída definidos no design. • Use a seguinte fórmula: o largura*comprimento*altura
Descrição	Recebe um conjunto com 3 valores, largura, comprimento e altura, e retorna a multiplicação dos valores, representando o volume.
Exemplos de entrada e saída	volume(base, altura); volume(3,4,2) = 3*4*2 = 24 volume(2,2,2) = 2*2*2 = 8 volume(5,4,3) = 5*4*3 = 60
Critérios de aceitação	1. Retorno deve ser do tipo Double; 2. Se base ou altura ou comprimento for menor ou igual a ZERO, gera um NotCorrectParametersException("Nenhum parâmetro pode ser menor ou igual a 0.");

19

4. Design da aplicação

4.1 Diagrama de Classes do Sistema



20

5. Mapeamento Requisitos X Casses/Métodos

5.1. A tabela abaixo apresenta as classes que deve ser implementadas para a criação de cada um dos requisitos do sistema. Utiliza estas classes. Novas classes não devem ser criadas.

Requisito ID	Classe	Método
01 – Soma de Valores	SimpleCalculator	soma(...)
02 – Subtração de Valores	SimpleCalculator	sub(...)
03 – Multiplicação de Valores	SimpleCalculator	mult(...)
04 – Divisão de Valores	SimpleCalculator	div(...)
05 – Resto de uma Divisão	SimpleCalculator	rest(...)
06 – Fatorial de um Número	ComplexCalculation	fatorial(...)
07 – Raiz quadrada de um Número	ComplexCalculation	sqrt(...)
08 – Potência	ComplexCalculation	power(...)
09 – Área de um círculo	GeometricCalculator	areaCirc(...)
10 – Volume de uma esfera	GeometricCalculator	volumeEsfera(...)
11 – Área de um retângulo	GeometricCalculator	areaRet(...)
12 – Volume de um cubo	GeometricCalculator	volumeRet(...)

6. Classes utilitárias e suas funções

6.1. ValidationUtil.

- Utilizada para fazer as validações necessárias. Cada método desta classe possui a sua documentação.
- Para verificar qual a responsabilidade de cada método, veja o JavaDoc.

6.2. GenericCalculator

- Possui uma instância da Classe ValidationUtil, chamada de util. Utilize esta referencia para realizar as validações já implementadas na classe ValidationUtil quando for necessário.
- Possui uma instância da classe Constantes, chamada *cons*. Utilize esta referencia para utilizar as constantes do sistema quando for necessário.

6.3. Constantes

- Possui as constantes necessárias para a implementação da solução.

6.4. NotCorrectParametersException

- Todas as exceções que o sistema necessita (conforme os requisitos) serão deste tipo. Utilize esta classe sempre que uma exceção for necessária. No construtor, passe um parâmetro do tipo "String", que será a "cause" da exceção.

APÊNDICE D – Apresentação para a equipe que utilizou o *FTSProc*

ÁREAS DE PESQUISA E EXPERIMENTO

Informações básicas sobre as áreas envolvidas na pesquisa e o funcionamento do experimento

Estevão Ricardo Hess

Agenda

- Áreas de Pesquisa
  - Desenvolvimento Distribuído de Software
  - Follow-the-Sun
- Experimento
  - Definições do experimento
    - Simulação do Follow-the-Sun
  - Ferramenta utilizada
  - Requisitos

Desenvolvimento Distribuído de Software (DDS)

- Globalização
  - Desenvolvimento de software
- Terceirização de serviços
- DDS surgiu nos anos 90, onde as empresas começaram a desenvolver software com times distribuídos
- O DDS é caracterizado sempre que um ou mais recursos humanos envolvidos no projeto estiverem fisicamente distante dos demais

Desenvolvimento Distribuído de software

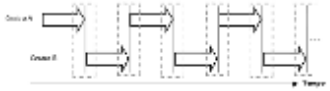
- Vantagens
  - Redução de custos
  - Ganho de proximidade com o cliente
  - Redução do tempo de projeto / time-to-market
  - Recursos especializados e globais
- Desafios
  - Legislação
  - Arquitetura de Software
  - Processos de desenvolvimento
  - Telecomunicações
  - Gerência de configuração
  - Gerenciamento de projetos
  - Confiança
  - Conflitos
  - Diferenças culturais
  - **Diferentes fusos horários**

Estratégia Follow-the-Sun

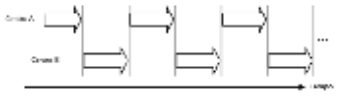
- O *follow-the-sun* é uma estratégia de desenvolvimento global de software;
- O objetivo é a diminuição do *time-to-market*, acelerando a construção do produto final;
- Este ambiente opera com equipes distribuídas em fusos horários e países distintos;
- Cada equipe detém o trabalho por determinado período;
- A transferência pode ser para qualquer tipo de tarefa;
- Esta transferência deve acontecer diariamente e de forma padronizada.

Estratégia Follow-the-Sun

- Trabalho é continuado em por cada centro distribuído
- Trabalho acontece em *Shifts*
- Com *Overlap* (Ex: Brasil e EUA):



- Sem *Overlap* (Brasil e Índia):



Estratégia Follow-the-Sun

- Principais vantagens
  - Utiliza a diferença de fuso horário como uma vantagem
  - Redução do time-to-market
  - Acelera o desenvolvimento
- Dificuldades
  - Coordenação
  - Sincronização
  - Comunicação
  - Transferência de trabalho

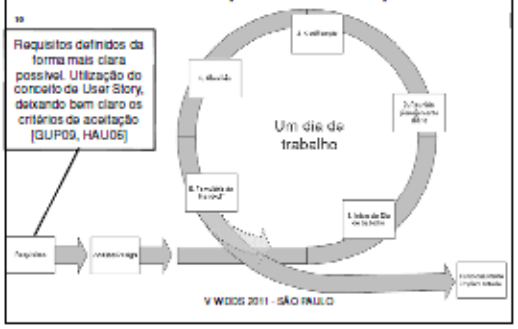
FTSPROC

Processo Proposto

V WDDS 2011 - SÃO PAULO

Processo Proposto - Requisitos

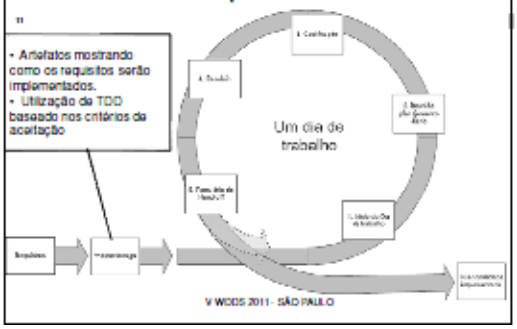
Requisitos definidos da forma mais clara possível. Utilização do conceito de User Story, deixando bem claro os critérios de aceitação [GUP09, HAU06]



V WDDS 2011 - SÃO PAULO

Processo Proposto - Análise

- Artefatos mostrando como os requisitos serão implementados.
- Utilização de TDD baseado nos critérios de aceitação

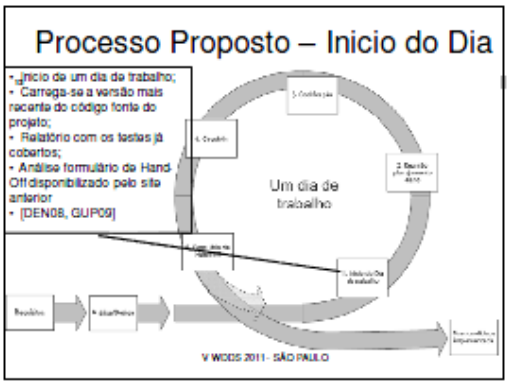


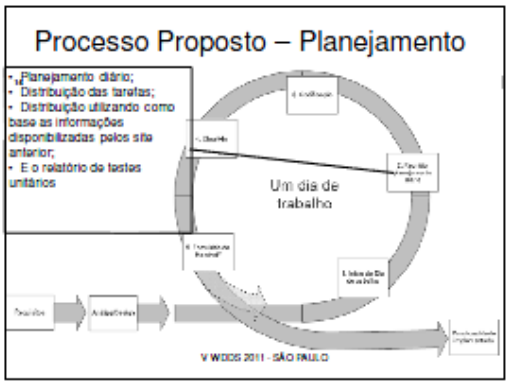
V WDDS 2011 - SÃO PAULO

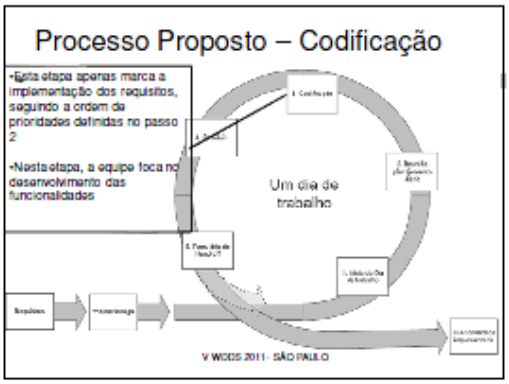
Processo Proposto

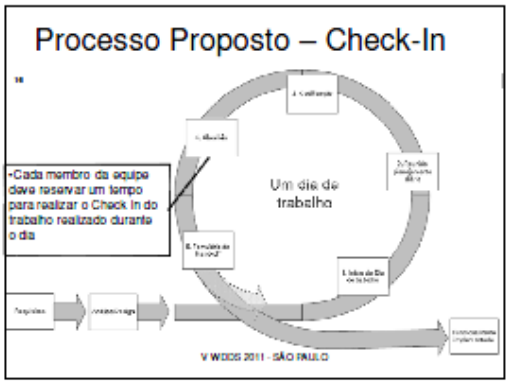
- Requisitos definidos de forma clara;
- Testes unitários criados, para utilizar a técnica de TDD.
- Fase de desenvolvimento inicia neste ponto.

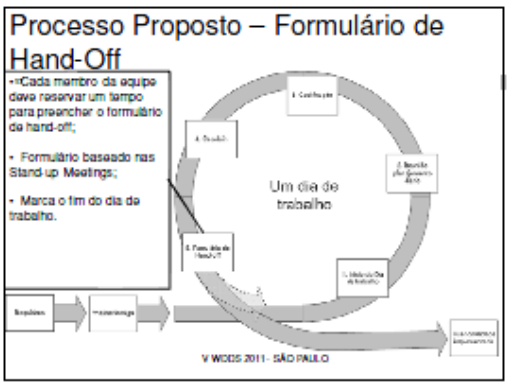
V WDDS 2011 - SÃO PAULO

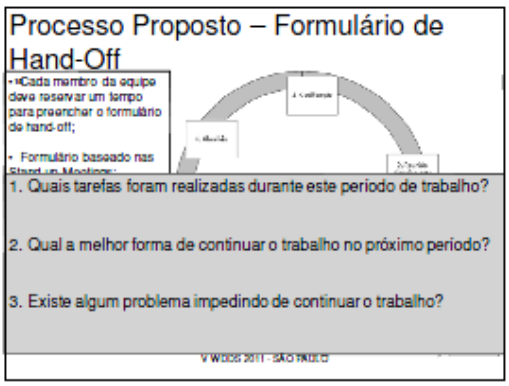

Processo Proposto – Inicio do Dia
• Início de um dia de trabalho;
• Carrega-se a versão mais recente do código fonte do projeto;
• Relatório com os testes já cobertos;
• Análise formulário de Hand Off disponibilizado pelo site anterior
• [DEN08, GUP09]
Um dia de trabalho
V WDDS 2011 - SÃO PAULO


Processo Proposto – Planejamento
• Planejamento diário;
• Distribuição das tarefas;
• Distribuição utilizando como base as informações disponibilizadas pelos site anterior;
• E o relatório de testes unitários
Um dia de trabalho
V WDDS 2011 - SÃO PAULO


Processo Proposto – Codificação
•Esta etapa apenas marca a implementação dos requisitos, seguindo a ordem de prioridades definidas no passo 2
•Nesta etapa, a equipe foca no desenvolvimento das funcionalidades
Um dia de trabalho
V WDDS 2011 - SÃO PAULO


Processo Proposto – Check-In
•Cada membro da equipe deve reservar um tempo para realizar o Check In do trabalho realizado durante o dia
Um dia de trabalho
V WDDS 2011 - SÃO PAULO


Processo Proposto – Formulário de Hand-Off
•Cada membro da equipe deve reservar um tempo para preencher o formulário de hand-off;
• Formulário baseado nas Stand up Meetings;
• Marca o fim do dia de trabalho.
Um dia de trabalho
V WDDS 2011 - SÃO PAULO


Processo Proposto – Formulário de Hand-Off
•Cada membro da equipe deve reservar um tempo para preencher o formulário de hand-off;
• Formulário baseado nas
1. Quais tarefas foram realizadas durante este período de trabalho?
2. Qual a melhor forma de continuar o trabalho no próximo período?
3. Existe algum problema impedindo de continuar o trabalho?
V WDDS 2011 - SÃO PAULO

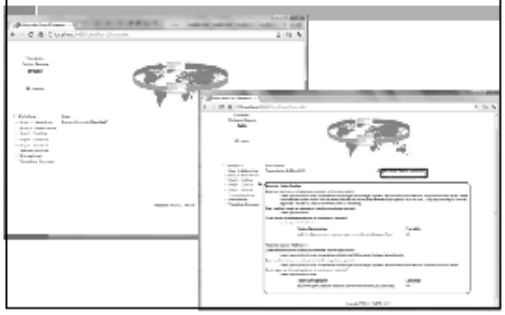
Processo Proposto – Funcionalidade Implementada

- Este processo é iterativo e estes cinco estados serão repetidos a cada dia de trabalho, para cada time de desenvolvimento distribuído (FADDO, GUP09)

FERRAMENTA DE APOIO

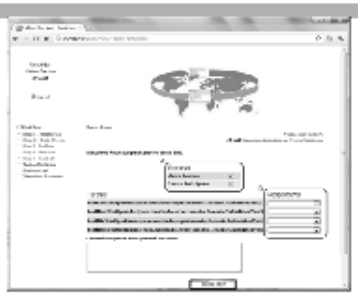
Tela Inicial – DEV/PM

Relatórios - PM



Relatórios Testes Unitários - PM

Iniciando Shift - PM



Durante Shift – DEV/PM

Finalizar Hand-Off – DEV/PM

DINÂMICA DO EXPERIMENTO

Objetivo

- Realizar a implementação de um sistema simples
- Utilizar a estratégia follow-the-sun
  - **Utilizar o FTSProc para a transferência de tarefas**
    - **Ferramenta de Apoio**

Fuso Horário

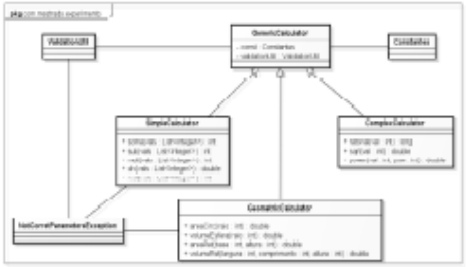
- Fuso horário será simulado
- Cada dia (shift) terá 20 minutos
- Ao final de cada shift, o trabalho deve continuar pelo próximo centro de desenvolvimento (site)

Requisitos

- Sistema matemático
- Implementação de métodos
- Não há interface gráfica
  - Utilização de Classes de Testes, já disponibilizadas
- Os critérios de aceitação serão validados para cálculo de requisitos implementados
- Todos os requisitos estão disponíveis no documento de requisitos

Casos de Uso

Design



Experimento - Ferramentas

- Desenvolvimento da aplicação – Java
  - Classes e arquivos a serem utilizados serão disponibilizados pelo pesquisador.
  - Não devem ser criadas novas classes.
- Utilizando IDE Eclipse
- Utilizando SVN como repositório
  - Dados para acesso serão fornecidos pelo pesquisador

Q&A

APÊNDICE E – Apresentação para a equipe *ad hoc*

ÁREAS DE PESQUISA E EXPERIMENTO

Informações básicas sobre as áreas envolvidas na pesquisa e o funcionamento do experimento

Estevão Ricardo Hess

AD

Agenda

- Áreas de Pesquisa
  - Desenvolvimento Distribuído de Software
  - Follow-the-Sun
- Experimento
  - Definições do experimento
    - Simulação do Follow-the-Sun
  - Requisitos

Desenvolvimento Distribuído de software

- Globalização
  - Desenvolvimento de software
- Terceirização de serviços
- DDS surgiu nos anos 90, onde as empresas começaram a desenvolver software com times distribuídos
- O DDS é caracterizado sempre que um ou mais recursos humanos envolvidos no projeto estiver fisicamente distante dos demais

Desenvolvimento Distribuído de software

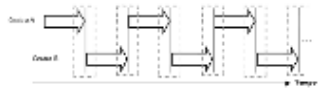
- Vantagens
  - Redução de custos
  - Ganho de proximidade com o cliente
  - Redução do tempo de projeto / time-to-market
  - Recursos especializados e globais
- Desafios
  - Legislação
  - Arquitetura de Software
  - Processos de desenvolvimento
  - Telecomunicações
  - Gerência de configuração
  - Gerenciamento de projetos
  - Confiança
  - Conflitos
  - Diferenças culturais
  - **Diferentes fusos horários**

Estratégia Follow-the-Sun

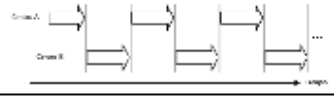
- O *follow-the-sun* é uma estratégia de desenvolvimento global de software;
- O objetivo é a diminuição do *time-to-market*, acelerando a construção do produto final;
- Este ambiente opera com equipes distribuídas em fusos horários e países distintos;
- Cada equipe detém o trabalho por determinado período;
- A transferência pode ser para qualquer tipo de tarefa;
- Esta transferência deve acontecer diariamente e de forma padronizada.

Estratégia Follow-the-Sun

- Trabalho é continuado em por cada centro distribuído
- Trabalho acontece em *Shifts*
- Com *Overlap* (Ex: Brasil e EUA):



- Sem *Overlap* (Brasil e Índia):



Estratégia Follow-the-Sun

- Principais vantagens
  - Utiliza a diferença de fuso horário como uma vantagem
  - Redução do time-to-market
  - Acelera o desenvolvimento
- Dificuldades
  - Coordenação
  - Sincronização
  - Comunicação
  - Transferência de trabalho

DINÂMICA DO EXPERIMENTO

Objetivo

- Realizar a implementação de um sistema simples
- Utilizar a estratégia follow-the-sun
  - **Transferência de tarefas como achar conveniente**

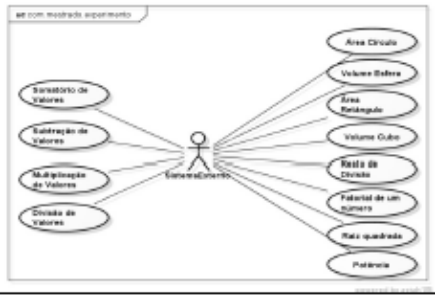
Fuso Horário

- Fuso horário será simulado
- Cada dia (shift) terá 20 minutos
- Ao final de cada shift, o trabalho deve continuar pelo próximo centro de desenvolvimento (site)

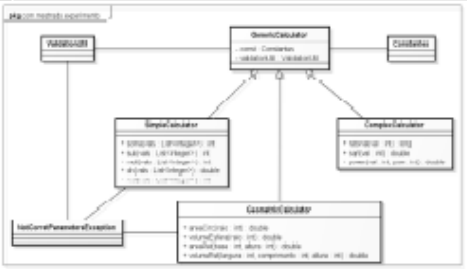
Requisitos

- Sistema matemático
- Implementação de métodos
- Não há interface gráfica
  - Utilização de Classes de Testes, já disponibilizadas
- Os critérios de aceitação serão validados para cálculo de requisitos implementados
- Todos os requisitos estão disponíveis no documento de requisitos
- Para critério de cálculo, cada requisito deve ser implementado inteiramente

Casos de Uso



Design

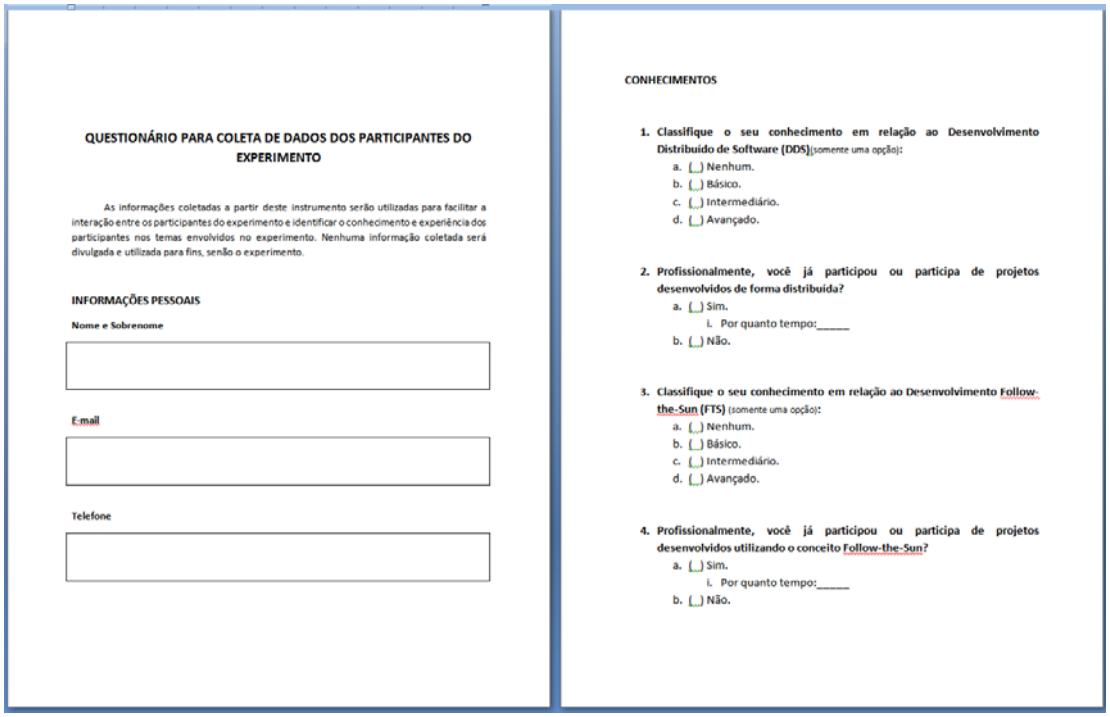


Experimento - Ferramentas

- Desenvolvimento da aplicação – Java
  - Classes e arquivos a serem utilizados serão disponibilizados pelo pesquisador.
  - Não devem ser criadas novas classes.
- Utilizando IDE Eclipse
- Utilizando SVN como repositório
  - Dados para acesso serão fornecidos pelo pesquisador

Q&A

APÊNDICE F – Questionário Inicial



QUESTIONÁRIO PARA COLETA DE DADOS DOS PARTICIPANTES DO EXPERIMENTO

As informações coletadas a partir deste instrumento serão utilizadas para facilitar a interação entre os participantes do experimento e identificar o conhecimento e experiência dos participantes nos temas envolvidos no experimento. Nenhuma informação coletada será divulgada e utilizada para fins, senão o experimento.

INFORMAÇÕES PESSOAIS

Nome e Sobrenome

E-mail

Telefone

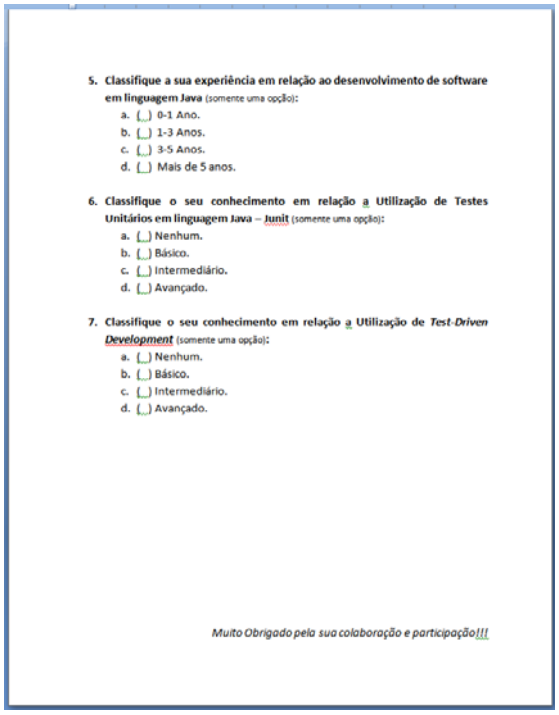
CONHECIMENTOS

1. **Classifique o seu conhecimento em relação ao Desenvolvimento Distribuído de Software (DDS)**(somente uma opção):
   a. (_) Nenhum.
   b. (_) Básico.
   c. (_) Intermediário.
   d. (_) Avançado.

2. **Profissionalmente, você já participou ou participa de projetos desenvolvidos de forma distribuída?**
   a. (_) Sim.
      i. Por quanto tempo:_____
   b. (_) Não.

3. **Classifique o seu conhecimento em relação ao Desenvolvimento Follow-the-Sun (FTS)** (somente uma opção):
   a. (_) Nenhum.
   b. (_) Básico.
   c. (_) Intermediário.
   d. (_) Avançado.

4. **Profissionalmente, você já participou ou participa de projetos desenvolvidos utilizando o conceito Follow-the-Sun?**
   a. (_) Sim.
      i. Por quanto tempo:_____
   b. (_) Não.



5. **Classifique a sua experiência em relação ao desenvolvimento de software em linguagem Java** (somente uma opção):
   a. (_) 0-1 Ano.
   b. (_) 1-3 Anos.
   c. (_) 3-5 Anos.
   d. (_) Mais de 5 anos.

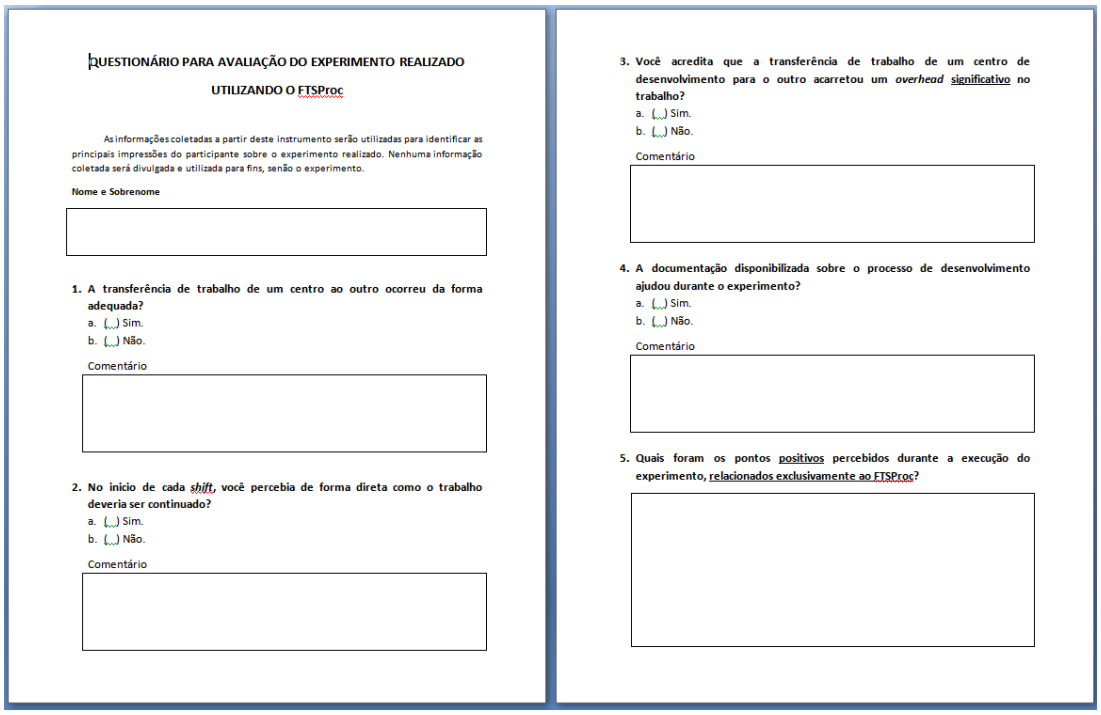
6. **Classifique o seu conhecimento em relação a Utilização de Testes Unitários em linguagem Java – Junit** (somente uma opção):
   a. (_) Nenhum.
   b. (_) Básico.
   c. (_) Intermediário.
   d. (_) Avançado.

7. **Classifique o seu conhecimento em relação a Utilização de *Test-Driven Development*** (somente uma opção):
   a. (_) Nenhum.
   b. (_) Básico.
   c. (_) Intermediário.
   d. (_) Avançado.

Muito Obrigado pela sua colaboração e participação!!!

APÊNDICE G – Questionário Final

1. Equipe *FTSProc*



QUESTIONÁRIO PARA AVALIAÇÃO DO EXPERIMENTO REALIZADO

UTILIZANDO O FTSProc

As informações coletadas a partir deste instrumento serão utilizadas para identificar as principais impressões do participante sobre o experimento realizado. Nenhuma informação coletada será divulgada e utilizada para fins, senão o experimento.

Nome e Sobrenome

1. A transferência de trabalho de um centro ao outro ocorreu da forma adequada?
a. (...) Sim.
b. (...) Não.

Comentário

2. No início de cada *shift*, você percebia de forma direta como o trabalho deveria ser continuado?
a. (...) Sim.
b. (...) Não.

Comentário

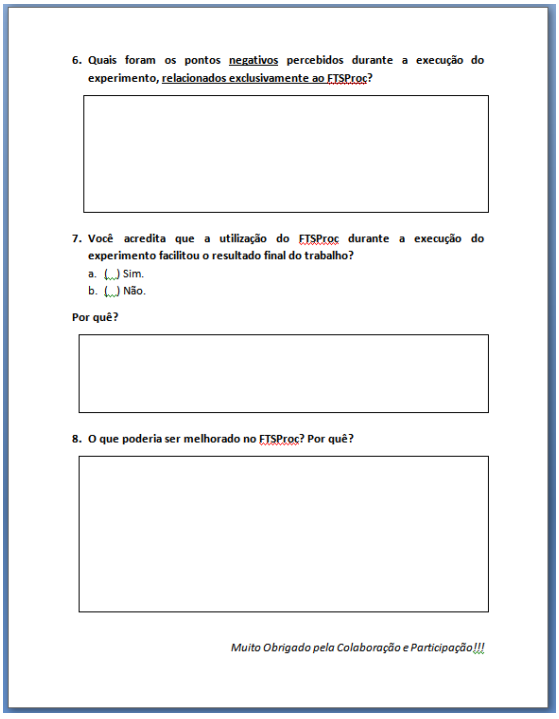
3. Você acredita que a transferência de trabalho de um centro de desenvolvimento para o outro acarretou um *overhead* significativo no trabalho?
a. (...) Sim.
b. (...) Não.

Comentário

4. A documentação disponibilizada sobre o processo de desenvolvimento ajudou durante o experimento?
a. (...) Sim.
b. (...) Não.

Comentário

5. Quais foram os pontos positivos percebidos durante a execução do experimento, relacionados exclusivamente ao FTSProc?



6. Quais foram os pontos negativos percebidos durante a execução do experimento, relacionados exclusivamente ao FTSProc?

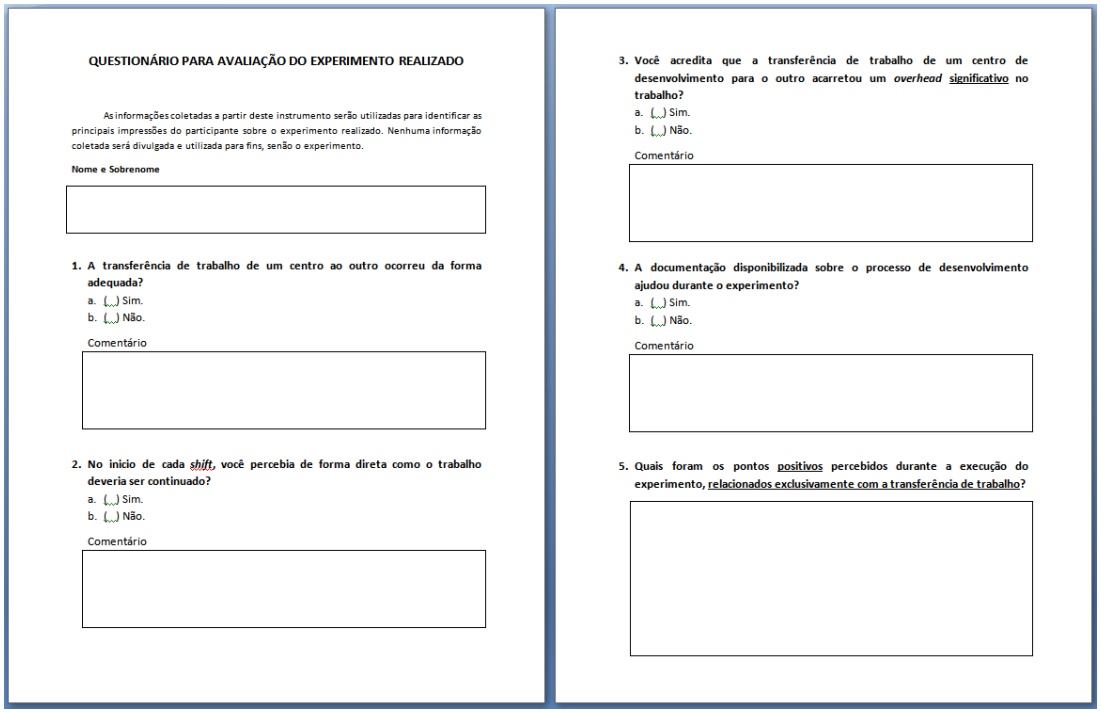
7. Você acredita que a utilização do FTSProc durante a execução do experimento facilitou o resultado final do trabalho?
a. (...) Sim.
b. (...) Não.

Por quê?

8. O que poderia ser melhorado no FTSProc? Por quê?

Muito Obrigado pela Colaboração e Participação!!!

2. Equipe *ad hoc*



QUESTIONÁRIO PARA AVALIAÇÃO DO EXPERIMENTO REALIZADO

As informações coletadas a partir deste instrumento serão utilizadas para identificar as principais impressões do participante sobre o experimento realizado. Nenhuma informação coletada será divulgada e utilizada para fins, senão o experimento.

Nome e Sobrenome

1. A transferência de trabalho de um centro ao outro ocorreu da forma adequada?
a. (...) Sim.
b. (...) Não.

Comentário

2. No inicio de cada *shift*, você percebia de forma direta como o trabalho deveria ser continuado?
a. (...) Sim.
b. (...) Não.

Comentário

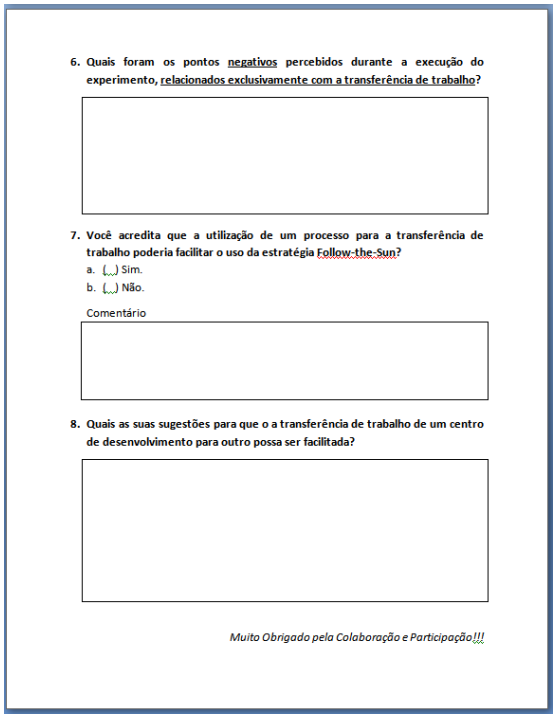
3. Você acredita que a transferência de trabalho de um centro de desenvolvimento para o outro acarretou um *overhead* significativo no trabalho?
a. (...) Sim.
b. (...) Não.

Comentário

4. A documentação disponibilizada sobre o processo de desenvolvimento ajudou durante o experimento?
a. (...) Sim.
b. (...) Não.

Comentário

5. Quais foram os pontos positivos percebidos durante a execução do experimento, relacionados exclusivamente com a transferência de trabalho?



6. Quais foram os pontos negativos percebidos durante a execução do experimento, relacionados exclusivamente com a transferência de trabalho?

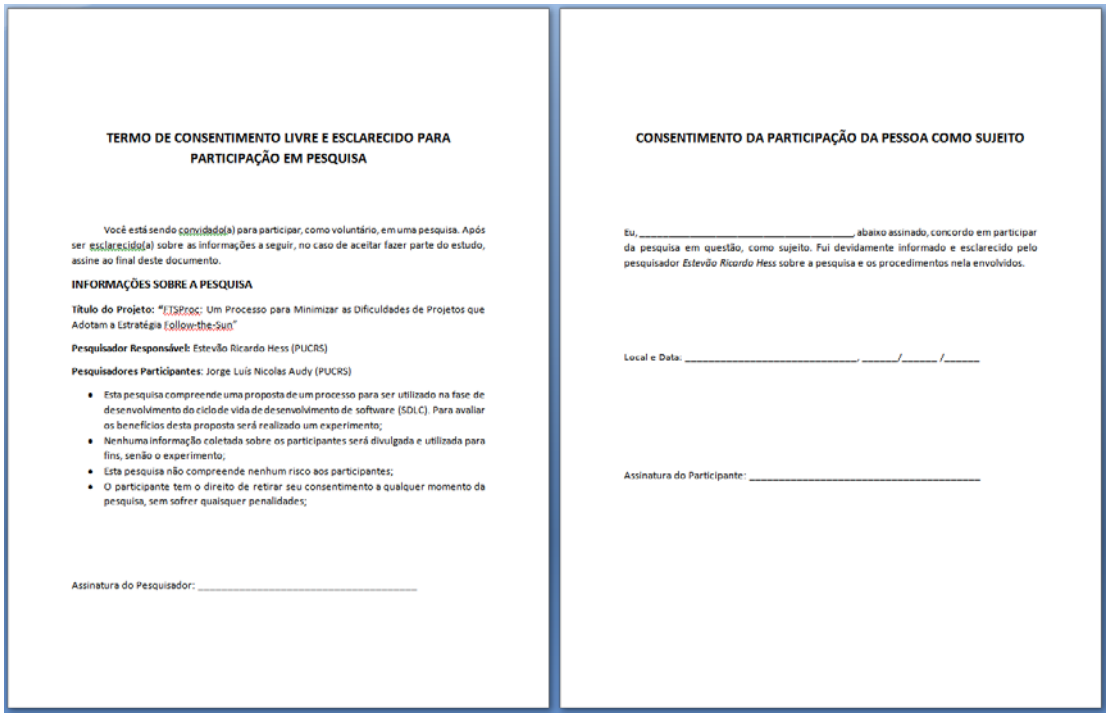
7. Você acredita que a utilização de um processo para a transferência de trabalho poderia facilitar o uso da estratégia Follow-the-Sun?
a. (...) Sim.
b. (...) Não.

Comentário

8. Quais as suas sugestões para que o a transferência de trabalho de um centro de desenvolvimento para outro possa ser facilitada?

Muito Obrigado pela Colaboração e Participação!!!

APÊNDICE H – Termo de Consentimento



TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO PARA PARTICIPAÇÃO EM PESQUISA

Você está sendo convidado(a) para participar, como voluntário, em uma pesquisa. Após ser esclarecido(a) sobre as informações a seguir, no caso de aceitar fazer parte do estudo, assine ao final deste documento.

INFORMAÇÕES SOBRE A PESQUISA

Título do Projeto: "FTSProc: Um Processo para Minimizar as Dificuldades de Projetos que Adotam a Estratégia Follow-the-Sun"

Pesquisador Responsável: Estevão Ricardo Hess (PUCRS)

Pesquisadores Participantes: Jorge Luís Nicolas Audy (PUCRS)

- Esta pesquisa compreende uma proposta de um processo para ser utilizado na fase de desenvolvimento do ciclo de vida de desenvolvimento de software (SDLC). Para avaliar os benefícios desta proposta será realizado um experimento;
- Nenhuma informação coletada sobre os participantes será divulgada e utilizada para fins, senão o experimento;
- Esta pesquisa não compreende nenhum risco aos participantes;
- O participante tem o direito de retirar seu consentimento a qualquer momento da pesquisa, sem sofrer quaisquer penalidades;

Assinatura do Pesquisador: _____

CONSENTIMENTO DA PARTICIPAÇÃO DA PESSOA COMO SUJEITO

Eu, _____, abaixo assinado, concordo em participar da pesquisa em questão, como sujeito. Fui devidamente informado e esclarecido pelo pesquisador *Estevão Ricardo Hess* sobre a pesquisa e os procedimentos nela envolvidos.

Local e Data: _____, _____/_____ /_____

Assinatura do Participante: _____

APÊNDICE I – Manual da ferramenta de apoio

Descrição Da Ferramenta
de Apoio ao FTSProc

Programa de Pós Graduação em
Ciência da Computação

Pesquisador: Estevão Ricardo Hess

Experimento Follow-the-Sun

Porto Alegre, 2011

Descrição Da Ferramenta
de Apoio ao FTSProc

Conteúdo Deste Documento

1.1 Configuração – Acesso de Administrador 4
1.2 Inicio de um *Shift* – Acesso de Gerente de Projeto 8
1.3 Inicio do trabalho – Acesso de Desenvolvedor 13
1.4 Inicio do próximo *Shift* – Acesso de Gerente de Projeto 15
1.5 Acompanhamento – Acesso de Gerente Global 16
1.6 Outras informações disponíveis 21

1 DESCRIÇÃO DA FERRAMENTA DE APOIO AO FTSProc

Em função de o processo proposto ser inédito, é necessário o desenvolvimento de uma ferramenta de apoio. A ferramenta desenvolvida realiza todo o controle necessário para a execução do *FTSProc*. Este controle inicia com a verificação dos recursos participantes do projeto. Estes recursos preenchem os seguintes papéis na ferramenta:

Administrador: responsável pela manutenção dos usuários da ferramenta e controle (adição, atualização e remoção) dos centros de desenvolvimentos do projeto.

Gerente global: responsável global do projeto. Suas responsabilidades estão voltadas ao acompanhamento do projeto. Para isto, a ferramenta disponibiliza diversos relatórios que apresentam o andamento do projeto.

Gerente de Projeto: em cada centro existe um gerente de projeto. Este é responsável pela inicialização dos *shifts*. Durante esta criação, o gerente de projeto pode designar as tarefas que cada um dos desenvolvedores irá trabalhar. Após iniciado o *shift*, os desenvolvedores podem iniciar o seu trabalho.

Desenvolvedor: responsável pelo desenvolvimento do projeto. Durante o seu dia de trabalho, deve informar a ferramenta em qual passo do FTSProc se encontra. Desta forma, a ferramenta garante que todos os passos definidos no processo são efetivamente realizados.

Dentre os quatro papéis apresentados, destaca-se a obrigatoriedade de apenas dois papéis: Administrador, para a configuração e o Gerente de Projeto, que atua também como um Desenvolvedor.

Além do controle de usuários, a ferramenta faz o controle do fluxo de trabalho de cada um dos desenvolvedores que atuam em cada centro de desenvolvimento durante os *shifts*. A ferramenta faz, ainda, o controle dos centros de desenvolvimento, verificando qual será o próximo a iniciar o *shift*.

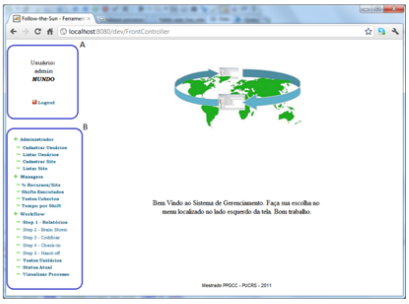
Para ilustrar o funcionamento da ferramenta de apoio, será simulado o seu funcionamento para um projeto distribuído em dois países distintos: Brasil e Índia. Cada um destes centros terão 2 desenvolvedores. A tabela a seguir mostra esta

distribuição, identificando cada componente e o seu papél, para facilitar o entendimento.

País	Brasil	Índia
Recursos	Jairo Santos (PM)	Nishant Kapoor (PM)
	Lucas Rodrigues (DEV)	Nikhil Divya (DEV)

1.1 Configuração – Acesso de Administrador

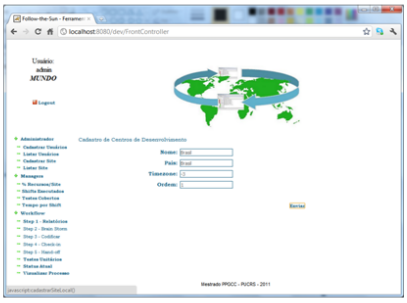
Inicialmente, devemos entrar no sistema utilizando uma conta de administrador para configurar a ferramenta. Ao acessar usando este perfil, a tela a seguir é apresentada.



Temos duas informações importantes nesta tela e que estará presente durante toda a utilização da ferramenta. Destacado com o identificador *A*, temos a

informação do usuário logado no sistema, e o país ao que ele pertence. Identificado com a letra *B*, temos um menu de contexto, onde todas as opções disponíveis para o usuário aparecem. As opções que estão temporariamente desativadas, apenas o nome da opção é exibida, mas não é possível clicar.

Agora, devemos cadastrar os centros de desenvolvimento distribuídos. Para este exemplo temos Brasil e Índia. Para este passo, usaremos a opção "Cadastrar Site", localizado no menu de contexto. A tela a seguir é apresentada.

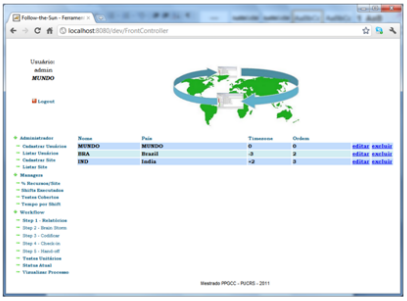


Devemos informar as seguintes informações e clicar em "Enviar":

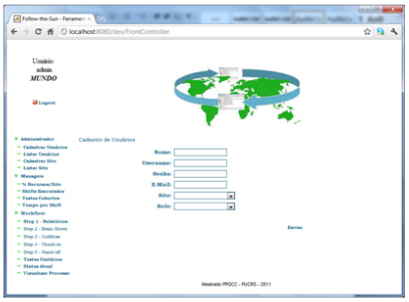
- Nome: apenas um identificador do centro;
- País: o país onde o centro está localizado;
- Timezone: qual o fuso do centro;
- Ordem: esta informação é usada para a organização dos shifts, ou seja, validar qual o centro deve iniciar o shift a cada fatia de tempo;

Ao confirmar o cadastro, o sistema nos leva automaticamente para a lista de centros distribuídos. Após a criação dos dois centro, é possível verificar as

informações dos mesmos clicando na opção "Listar Site"; a tela a seguir, contendo os dados de cada centro será exibida.



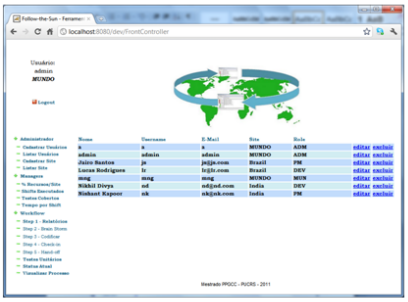
O próximo passo será o cadastro de usuários. Para cada, centro deve ser cadastrado pelo menos um gerente de projeto. Para iniciar o cadastro, basta entrar na opção "Cadastrar Usuários", e a seguinte tela será exibida:



Devemos informar as seguintes informações e clicar em "Enviar":

- Nome: representa o nome completo do usuário;
- Username: identificador para acessar o sistema;
- Senha: combinado com o username provêem acesso ao sistema;
- E-mail: e-mail para contato com o usuário;
- Site: representa a qual centro este usuário pertence;
- Role: qual o papel o usuário representa no projeto (Gerente de Projeto, Desenvolvedor, Gerente Global);

Ao confirmar o cadastro, o sistema nos leva automaticamente para a lista de usuários cadastrados. Após a criação dos quatro usuários do exemplo, é possível verificar as informações dos mesmos clicando na opção "Listar Usuários"; a tela a seguir, contendo estes dados será exibida.

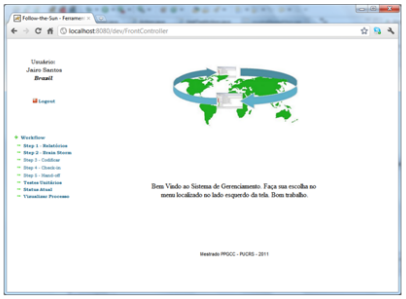


Neste ponto temos: todos os centros de desenvolvimento cadastrados, juntamente com os seus usuários. Portanto, a ferramenta está configurada para iniciar o uso. Entretanto, opcionalmente, pode-se cadastrar outros usuários com o papel de Gerente Global ou Administradores.

1.2 Início de um *Shift* – Acesso de Gerente de Projeto

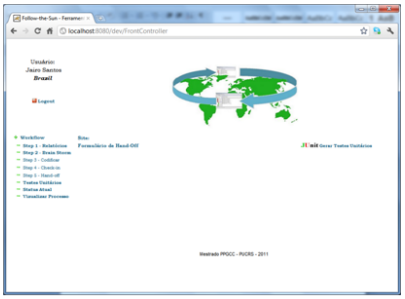
Com todos os cadastros e configurações finalizadas, é possível iniciar o uso da ferramenta. Para simular o uso desta ferramenta, usaremos a seguinte estrutura:

O centro localizado no Brasil vai iniciar o primeiro shift. Para isto, o Gerente de projeto deste centro irá acessar a aplicação. Ao fazer este acesso, a seguinte tela será exibida:



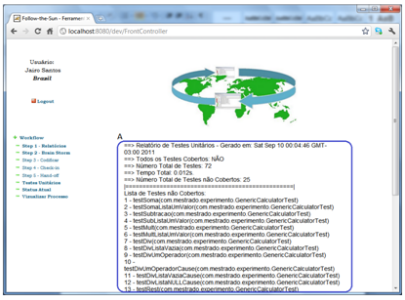
Notamos através da imagem, que devido ao controle de acessos dos usuários, o menu localizado na parte esquerda da tela já está alterado, apresentando apenas as informações permitidas para este perfil de usuário. Os passos listados a seguir são realizados pelo Gerente de Projeto para iniciar um shift:

i. Visualizar os relatórios de *Hand-off*: para isto, basta acessar a opção "Step 1 – Relatórios" do menu de opções. A seguinte tela será apresentada:



Como este é o primeiro shift sendo realizado, não há nenhuma informação relevante disponibilizada pelo centro de desenvolvimento anterior.

ii. Gerar relatório dos testes unitários: para isto, basta acessar a opção "Gerar Testes Unitários" localizado do lado superior direito. A seguinte tela será apresentada:



Destacado com o identificador *A* temos uma lista com todos os testes unitários que estão falhando no projeto em desenvolvimento. Estas informações serão úteis para identificar como iniciar o trabalho em cada shift.

Após gerar o relatório de testes unitários, é possível seguir para o segundo passo do processo, chamado de *Brainstorm*.

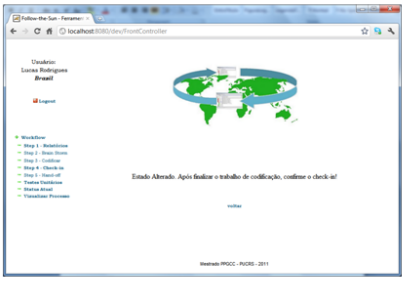
iii. Brainstorm: para isto, basta acessar a opção "Step 2 – Brainstorm". A seguinte tela é apresentada:

Destacado com o identificador *A* temos uma lista com todos os possíveis participantes do shift, ou seja, todos os recursos alocados para o centro que está iniciando o shift. Deve-se obrigatoriamente selecionar os participantes (por padrão, todos estão selecionados). Destacado com o identificador *B* está a lista de testes unitários geradas no passo anterior. Com base nesta lista é possível designar testes para os participantes dos shifts, para que estes fiquem responsáveis por trabalhar nas atividades relacionadas ao teste. Este passo não é mandatório, mas recomendável. Esta tela mostra ainda um campo de texto onde é possível adicionar qualquer informação que o Gerente de Projeto julgar relevante. Após informar todos os dados, deve-se clicar na opção "Iniciar Shift". Com o shift iniciado, todos os desenvolvedores (além do Gerente de Projeto) devem iniciar o trabalho na ferramenta.

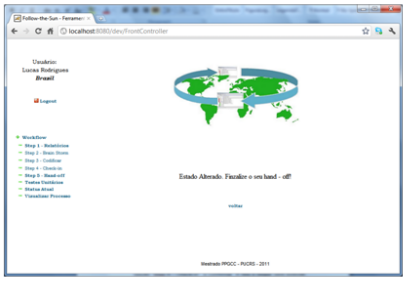
1.3 Inicio do trabalho – Acesso de Desenvolvedor

Com o shift iniciado, todos os recursos alocados neste shift devem utilizar a ferramenta para controlar o fluxo de trabalho exigido pelo processo. Estes passos serão ilustrados a seguir:

i. Iniciar o trabalho de implementação: informar ao sistema que iniciou o trabalho de implementação, para isto, basta clicar na opção "Step 3 – Codificar", e confirmar. A tela a seguir será exibida:



ii. Informar a realização do check-in do trabalho: a ferramenta exige que seja informado o check-in do trabalho realizado. Isto deve-se a este ser um passo obrigatório no processo. Para fazer esta operação, após finalizar o check-in do seu trabalho, basta clicar na opção "Step 4 – Check-in", e confirmar. A tela a seguir será exibida:



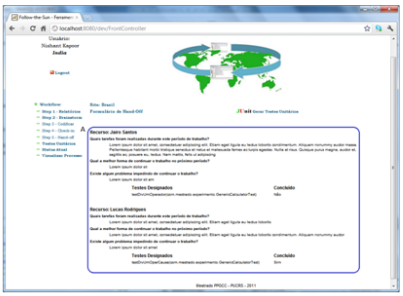
iii. Após finalizar este passo, a ferramenta habilita a opção para realizar o *hand-off*. Nesta opção o usuário irá relatar o que foi realizado durante o dia de trabalho, qual a melhor forma de continuar o trabalho e, finalmente, se há algum impedimento para a continuação do mesmo. Acesse a opção "Step 5 – Hand-off" para finalizar o hand-off. A seguinte tela será exibida:

Nesta tela, na área destacado com o identificador *A*, o usuário deve indicar, respondendo essas três perguntas, todas as informações relevantes para que o próximo centro inicie o trabalho do ponto onde você parou. Ainda, na área destacado com o identificador *B*, estão listados os testes designados ao usuário logado no início do shift. Nesta área, o usuário pode indicar os testes que foram trabalhados. Esta informação será validada de forma automatizada pela ferramenta, se os testes foram realmente cobertos, caso contrário, o shift seguinte identificará o teste como não coberto no relatório de testes unitários.

O shift é considerado concluído somente quando todos os usuários alocados no shift concluírem o hand-off. Ao identificar que todos os usuário finalizaram o hand-off, a ferramenta habilita automaticamente o próximo centro a iniciar o seu shift, caso contrário, o shift não pode ser iniciado.

1.4 Inicio do próximo *Shift* – Acesso de Gerente de Projeto

O Gerente de projeto do centro seguinte acessa a ferramenta para iniciar o shift. Como visto na seção 1.2 deve-se acessar a opção de relatórios através do menu "Step 1 – Relatórios". A seguinte tela é apresentada:



A única diferença para o primeiro shift, é que o time que está iniciando o segundo (ou mais) shift deve considerar as informações passadas pelo centro anterior para programar o seu dia de trabalho. Esta tela, diferentemente do que acontece no primeiro shift, contém estas informações, conforme podemos ver na área destacada com o identificador *A*. Após vizulizar o relatório, deve-se gerar o relatório de testes unitários, e seguir para o *Brainstorm*. Os outros passos seguem como mostrado no primeiro shift, na seção 1.2.

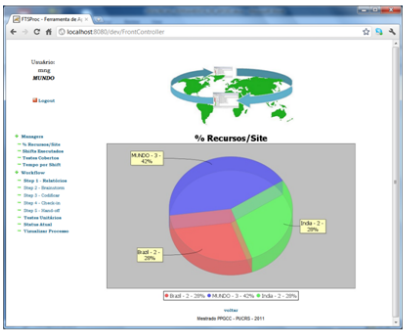
1.5 Acompanhamento – Acesso de Gerente Global

Ao acessar a aplicação como um Gerente Global, a seguinte tela é apresentada:

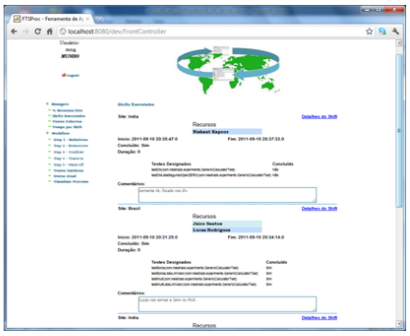
Da mesma forma, esta tela também apresenta opções para o usuário com perfil de Gerente Global, voltado principalmente para relatórios de acompanhamento do projeto.

Os seguintes relatórios podem ser gerados:

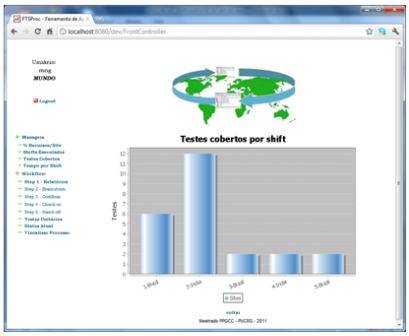
i. Percentual de recursos alocados por centro de desenvolvimento. A imagem abaixo mostra um exemplo deste relatório.



ii. Informações sobre os shifts executados: contém todos os dados informados por cada recurso em cada um dos shifts já executados. Esta tela ainda permite detalhar cada um dos shifts para obter mais informações. A imagem abaixo mostra um exemplo deste relatório.



iii. Quantidade de testes cobertos em cada shift: com esta informação é possível acompanhar o progresso do projeto, e identificar quais os shifts foram mais ou menos produtivos. Assim é possível tomar ações, como por exemplo, a adição de recursos a determinado centro de desenvolvimento. A imagem abaixo mostra um exemplo deste relatório.



iv. Tempo por shift: este relatório mostra o tempo transcorrido desde o inicio do shift pelo Gerente de Projeto, até o último desenvolvedor finalizar o seu hand-off. Esta informação é relevante para verificar se está sendo necessária a utilização de horas-extras para a conclusão do processo. A imagem abaixo mostra um exemplo deste relatório.

1.6 Outras informações disponíveis

As seguintes informações podem ser obtidas por qualquer perfil de usuário:

i. Testes Unitários: gera um relatório *ad hoc*. Ou seja, é possível verificar a qualquer momento quantos testes ainda precisão ser trabalhados. Nesta opção, os testes unitários são executados. Esta opção não substitui o processo de iniciar shift, onde a geração de um relatório similar é exigido. A imagem abaixo mostra um exemplo desta tela.

ii. Status Atual: mostra o andamento do projeto em um dado momento. Se neste momento existe um shift em andamento, informações sobre este serão exibidas. Caso contrário uma mensagem mostrando que não há shifts em andamento é mostrada. A tela abaixo mostra a tela exibida para esta opção quando há um shift em andamento.