UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DE ESTÁGIOS DO DEE

RELATÓRIO DE ESTÁGIO INTEGRADO

Título do Trabalho: Curso Seqüencial de Formação Complementar em Análise de Testes

Trabalho Apresentado por: Natasha Benjamim Barbosa

Empresa: Motorola e CIn – Centro de Informática da UFPE

Orientador: Francisco Marcos de Assis

Campina Grande – Paraíba
Junho / 2003

Biblioteca Setorial do CDSA. Fevereiro de 2021.

Sumé - PB

Aluna: Natasha Benjamim Barbosa
Matrícula: 29811105

ESTÁGIO INTEGRADO

Julgado em ____/____/____

Nota: _____(_____)

BANCA EXAMINADORA:

_____
Orientador

_____
Convidado

CAMPINA GRANDE – PARAÍBA

Universidade Federal de Campina Grande
Centro de Ciências e Tecnologia
Coordenação de Estágios do DEE

AVALIAÇÃO DE ESTÁGIO

ALUNO: Natasha Benjamim Barbosa, Mat. 29811105
EMPRESA: Motorola e Cin – Centro de Informática da UFPE
ÁREA DE ESTÁGIO: **Telecomunicações**
PERÍODO DE ESTÁGIO: **11/11/02 à 12/06/03**

A Coordenação de Estágios do DEE, adotando os procedimentos de praxe para a avaliação da disciplina ESTÁGIO, apresenta seu parecer final:

COMISSÃO EXAMINADORA	**RELATÓRIO**	**DEFESA**
_____ PROF. ORIENTADOR	_____	_____
_____ PROF. CONVIDADO	_____	_____

NOTA FINAL: (_____) _____

Tendo em vista o conceito atribuído ao aluno pelo estágio realizado conforme autorização deste Departamento de Engenharia Elétrica e tendo em vista as normas que regulam o Estágio Integrado, o aluno tem direito a _____ créditos.

Campina Grande, ____ de junho de 2003

Centro de Informática U·F·P·E

Recife, 16 de Junho de 2003

DECLARAÇÂO

Declaramos para os devidos fins que a Srta. Natasha Benjamim Barbosa foi estagiária do Centro de Informática da Universidade Federal de Pernambuco, pelo período de 11 de Novembro de 2002 a 10 de Junho de 2003, com jornada diária de 8 horas, perfazendo um total de 1.120hrs.

Declaramos, ainda, que a mesma foi aluna regularmente matriculada no Curso Seqüencial de Formação Complementar em Análise de Testes do mesmo Centro de Informática, pelo período acima relacionado.

Augusto César Alves Sampaio
Coordenador

Caixa Postal 7851
50732-970 Recife PE
Fone: +55 (81) 32718430
Fax: +55 (81) 32718438
postmaster@cin.ufpe.br

AGRADECIMENTOS

Inicialmente, o agradeço ao professor Camilo Gondim, coordenador do PCT, que informou sobre a existência desse programa.

Ao Centro de Informática (CIn) da UFPE que idealizou o programa e à Motorola que ofereceu os subsídios necessários para a criação e manutenção do mesmo. Além dos grandes parceiros que foram a Qualiti e o CESAR.

Aos professores do CIn, que ensinaram bastante.

Aos colegas de estágio que foram muito importantes e tornaram muito agradável esse período, pois foram verdadeiros companheiros.

Toda a equipe de testes de TDMA, pela ajuda e paciência que tiveram, pessoas como o Team Leader Edson Fontes e Waleska Dantas, os quais não há como esquecê-los. Além da secretária Sara Normando.

Além de todos esses, os agradecimentos merecidos ao orientador Professor Francisco Marcos e ao Coordenador de Estágios Prof. Talvanes. Às secretárias de Engenharia Elétrica da UFCG, Adail e Rosilda, que sempre fazem o que podem pelos alunos do curso.

Um agradecimento especial às colegas do Centro Acadêmico de Engenharia Elétrica (CAEE) pela Coordenação de Estágios, que está trabalhando muito bem, e ao Prof. Benedito Aguiar, diretor do CCT, o qual está sempre disposto a ajudar os alunos de Engenharia Elétrica, contribuindo bastante com o Prof. Talvanes, graças à pedidos do CAEE.

Os últimos agradecimentos são a Deus e à família, sem o apoio dos quais nada seria possível.

SUMÁRIO

ABREVIATURAS

API	Application Program Interface
AWT	Abstract Window Toolkit
CESAR	Centro de Estudos e Sistemas Avançados do Recife
CIn	Centro de Informática
CLDC	Connected Limited Device Configuration
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integrated
CR	Change Request
CSD	Circuit Switch Data
CVS	Concurrent Versions System
GC	Gerência de Configuração
GSM	Global System Mobile
GUI	Interface Gráfica para Usuários
HW	Hardware
ISO	International Organization for Standardization
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JSDK	Java Software Development Kit
MIDP	Mobile Information Device Profile
MTP	Manual Test Procedure
MTS	Mobile Test System
OO	Orientação a Objetos
PTF	Phone Test Framework
RUP	Rational Unified Process
SEI	Software Engineering Institute
SW	Software
TDMA	Time Division Multiple Access
UFPE	Universidade Federal de Pernambuco
UML	Unified Modeling Language

RESUMO

O presente relatório descreve todo o desenvolvimento desse trabalho até o momento. O estágio foi dividido em duas etapas: inicialmente os estagiários receberam um curso preparatório na área de Engenharia de Software com ênfase em Análise de Testes, e, a partir do segundo mês, em paralelo ao curso o estagiário passou a realizar, também, atividades práticas, no ambiente de trabalho montado, sob a orientação direta da Motorola.

O curso foi descrito detalhadamente nas próximas seções do relatório, mas as atividades práticas foram apenas citadas, pois como o estagiário deve assinar um termo de sigilo, não é permitido relatar os detalhes das atividades realizadas. Em resumo, pode-se dizer que estão sendo realizados trabalhos com testes de softwares para celulares Motorola que ainda não foram lançados no mercado. Sendo assim, a questão do sigilo é tão importante para evitar espionagem industrial.

O relatório a seguir começa com uma breve introdução explicando o programa que deu origem a esse estágio. A seguir, serão apresentados alguns conceitos básicos que auxiliarão na compreensão das disciplinas que foram oferecidas no curso preparatório.

Os cronogramas do estágio serão mostrados: o da primeira etapa de curso preparatório e o da segunda etapa de atividades práticas. Na seção seguinte será detalhado todo o curso preparatório que foi dividido em dois módulos: o primeiro contendo disciplinas básicas e o segundo contendo as avançadas.

Finalmente serão apresentados algumas das atividades práticas desenvolvidas e os resultados obtidos. O relatório é concluído mostrando a importância que o estágio em questão teve para a formação profissional do estagiário, além das contribuições para os trabalhos realizados pela próxima turma de estagiários.

1. Introdução

Neste relatório será apresentado um estágio diferente dos usuais, por fazer parte de mais um programa resultante de uma parceria entre uma empresa da iniciativa privada e uma universidade. Este programa denominado "Curso Seqüencial de Formação Complementar", nasceu de um convênio entre a Motorola e o CIn – Centro de Informática da UFPE, e aborda, principalmente, conhecimentos de Engenharia de Software com ênfase em Análise de Testes.

O objetivo do programa é essencialmente "Fomentar a formação de recursos humanos com alto grau de especialização na área de software embutido para testes de aplicações em computação móvel, de forma a viabilizar a atração de empresas e a criação de um centro de excelência em software básico no Recife, usando os incentivos e benefícios previstos na Lei 10.176".

Os estagiários foram selecionados, mediante análise curricular, sendo um total de 30 bolsistas. Para se inscrever para a seleção, o candidato deveria estar formalmente matriculado ou ter concluído algum curso de Ciência da Computação, Engenharia da Computação, Engenharia Elétrica ou Engenharia Eletrônica de universidades do Nordeste.

Os coordenadores do curso são: o Prof. Augusto Sampaio e o Prof. Sérgio Cavalcante. O Gerente de Projeto da Motorola é o Sr. Luís Cláudio Figueiredo.

O curso foi organizado em duas etapas: uma etapa de treinamento e outra de desenvolvimento de projetos. O treinamento composto por dois módulos, sendo o primeiro de 188 horas de cursos básicos o qual proporcionou ao estagiário uma visão geral das diversas fases do processo de desenvolvimento de software. O segundo módulo tinha como objetivo capacitar os estudantes na especificação e modelagem de testes de software, sendo composto por cinco cursos, com 124 horas de carga horária. O desenvolvimento de projetos tem duração de 240h.

No treinamento dos alunos, durante os primeiros quatro meses, os principais conceitos a serem desenvolvidos no período do curso foram apresentados em salas de aula e vivenciados em laboratório. As aulas foram realizadas à noite, no horário das 18 às 22h. Os bolsistas, paralelamente ao período de treinamento, realizaram um estágio complementar de 4h. Este estágio complementar não teve início logo no primeiro mês de curso, pois a infra-estrutura física de trabalho ainda não estava pronta.

Na fase de desenvolvimento de projetos, os estagiários têm como meta o desenvolvimento de um projeto prático relacionado à formação básica adquirida na fase anterior. Cada aluno está sendo acompanhado por um orientador que dá suporte à consolidação de sua formação, garantindo que os conceitos teóricos estudados sejam aplicados na prática. O horário desta atividade está sendo flexibilizado de acordo com a disponibilidade dos alunos.

A avaliação dos estagiários em cada disciplina foi realizada através da aplicação de um exame no qual a nota obtida deveria ser maior ou igual a 7,0 (sete). Os projetos práticos desenvolvidos são também acompanhados pelos instrutores que medem desempenho dos alunos durante todo o programa, auferindo ao final seu grau de eficiência. No final do curso, todos os alunos deverão participar de um workshop, onde os resultados serão apresentados e discutidos.

Qualquer reprovação ou desistência de um aluno de cursar disciplinas implicaria na exclusão do mesmo do Curso Seqüencial. Os resultados obtidos no curso também serão analisados e compilados em um relatório produzido no workshop de avaliação.

Este relatório irá descrever, resumidamente, como decorreu este programa até então, os resultados que já foram obtidos e os conhecimentos adquiridos. Inicialmente serão apresentados alguns dos conceitos básicos necessários para o entendimento das disciplinas e atividades. Em seguida, será mostrada uma visão geral do curso de treinamento e do que foi desenvolvido no estágio.

2. Conceitos Básicos

RUP (Rational Unified Process): conjunto das melhores práticas em engenharia de software.

Caso de Uso: forma específica de uso do sistema pela execução de algumas de suas funcionalidades, mostra o que o sistema faz, e não como.

Projeto Iterativo e Incremental: maneira na qual o sistema é construído, de forma incremental, mediante o uso de vários mini-projetos que se repetem (iterações).

Ator: algo externo que possui interface com o sistema.

Arquitetura de um Sistema de SW: forma a espinha dorsal para se construir softwares. É composta de estruturas do sistema que incluem componentes de software, suas propriedades e seus relacionamentos.

UML: é uma linguagem usada para especificar, modelar e documentar os artefatos de um sistema. Tornou-se o padrão empresarial para modelagem OO.

Orientação a Objetos: é uma abordagem adotada em programação, para permitir o reuso de código, além de facilitar o trabalho do programador, pois aproxima os códigos de modelos reais, organizando assim o programa de maneira mais clara e permitindo ainda que o software tenha muitas outras boas características, como: manutenibilidade, portabilidade, robustez, entre outras.

Objeto: modelo de um objeto real, entidade física, conceitual ou de software.

Abstração: construção de um modelo para representação de uma realidade.

Encapsulamento: elimina dependência de implementação escondendo-a do cliente, usando, para isso interfaces.

Modularidade: decomposição de um problema em pequenos pedaços, para gerenciar complexidade.

Herança: criação de hierarquias de abstração, permite reuso de código.

Classe: descrições de objetos com propriedades e comportamentos comuns.

Atributos: são as propriedades de um objeto, que armazena o seu estado.

Métodos: são serviços que os objetos oferecem a outros objetos, eles modelam o comportamento da classe.

Interface: definem um tipo, especificando apenas a assinatura de seus métodos, elas não possuem atributos e seus métodos não têm corpo.

Componente: parte não trivial, quase independente e substituível de um sistema que faz a realização de interfaces. Ex.: códigos fontes, componentes de tempo de execução, executável.

Pacote: organiza elementos em grupos, para facilitar o entendimento do sistema e favorecer o reuso.

Relacionamentos: possíveis tipos de relações entre classes, como: associação, dependência, generalização e realização.

Cápsula: representa um *thread* do sistema, que é um fluxo de controle independente no sistema e identifica concorrência.

Protocolo: identificam o "contrato" entre cápsulas, definindo um conjunto de sinais usados na comunicação entre diferentes *threads* e a seqüência válida de troca de sinais entre cápsulas.

Subsistema: parte independente do sistema que pode representar uma abstração de produtos ou sistemas externos a serem incorporados na implementação.

Fachada: representa os serviços oferecidos pelo sistema, gerencia transações e centraliza as instâncias das coleções de negócio e/ ou controladores.

ThreadGroup: pode ser entendido como a simulação de um conjunto de usuários que acessarão o sistema. Cada *thread* representa um usuário.

Controller: responsável pelo controle do fluxo de passos de teste e pela criação dos próprios passos de teste.

Timer: é um elemento que representa o intervalo de atraso entre cada solicitação de um *thread* para o sistema naquele teste específico.

Listener: recebem informações sobre dados de respostas de cada solicitação enviada pelo JMeter para o sistema e habilitam recursos para a visualização dos resultados dos testes.

Configuração: conjunto de itens produzidos por um projeto de desenvolvimento de software, como: programas, documentação e dados.

Baseline: uma especificação ou produto que foi formalmente revisado e aceito, servindo como base para passos posteriores do desenvolvimento.

Repositório: local onde os itens de um sistema são guardados.

Lock: é a chave que garante que apenas um usuário pode alterar o arquivo, aquele que detém o lock.

Check-out: ação de recuperar a última versão de um item de configuração guardada no repositório.

Check-in: ação de inserir ou atualizar um item de configuração no repositório.

Tags: rótulos que são associados a conjuntos de arquivos.

Branch: criação de um fluxo alternativo para atualização de versões de itens de configuração.

Merge: unificação de diferentes versões de um mesmo item de configuração.

Build: uma versão ainda incompleta do sistema em desenvolvimento, mas com certa estabilidade.

Release: versão do sistema validada após os diversos tipos de teste.

Padrões: formas de representação de conhecimento, organizada de forma estruturada.

3. Curso Seqüencial de Formação Complementar em Análise de Testes

3.1. Cronograma do Curso Preparatório:

ID	Disciplina	Carga Horária	Início	Término	Professor
1	*Curso Seqüencial de Formação Complementar*	*69 dias*	*11/11/02*	*12/06/03*	
	Módulo 1	**23,5 dias**	**11/11/02**	**27/01/03**	
2	A) GER_01 – Fundamentos de um Processo Iterativo e Incremental	12 hs	11/11/02	13/11/02	Alexandre Vasconcelos
3	B) QSW_01 – Qualidade de *Software*	20 hs	14/11/02	21/11/02	Alexandre Vasconcelos
4	C) PGP_01 – Planejamento e Gerenciamento de Projetos	20 hs	22/11/02	28/11/02	Hermano Perrelli
5	D) RAP_01 – Engenharia de Requisitos e Análise e Projeto de *Software* com UML	56 hs	29/11/02	18/12/02	Augusto Sampaio
6	E) LIN_01 – Orientação a Objetos com Java	40 hs	19/12/02	13/01/03	Paulo Borba
7	F) TST_01 – Testes de *Software*	20 hs	14/01/03	20/01/03	Alexandre Vasconcelos
8	G) SCM_01 – Gerência de Configuração	20 hs	21/01/03	27/01/03	André Santos
	Módulo 2	**45,5 dias**	**28/01/03**	**12/06/03**	
9	H) LIN_02 – J2ME	40 hs	28/01/03	10/02/03	Jorge Fernandes
10	I) ASW_01 – Arquitetura de *Software* para Celulares	40 hs	11/02/03	24/02/03	Sérgio Cavalcante
11	J) TST_02 – Ferramentas de Testes	24 hs	25/02/03	07/03/03	Alexandre Vasconcelos
12	K) TST_03 – Ferramentas para Testes de Aplicações J2ME	20 hs	10/03/03	14/03/03	Jorge Fernandes
13	L) TCC_01 – Trabalho de Conclusão de Curso	240 hs	17/03/03	12/06/03	Sérgio Cavalcante

3.2. Cronograma do Estágio:

Novembro 2002						
S	T	Q	Q	S	S	D
-	-	-	-	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	-

Janeiro 2003						
S	T	Q	Q	S	S	D
-	-	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	-	-

Março 2003						
S	T	Q	Q	S	S	D
-	-	-	-	-	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	-	-	-	-	-	-

Dezembro 2002						
S	T	Q	Q	S	S	D
-	-	-	-	-	-	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	-	-	-	-	-

Fevereiro 2003						
S	T	Q	Q	S	S	D
-	-	-	-	-	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	-	-

Abril 2003						
S	T	Q	Q	S	S	D
-	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	-	-	-	-

Legenda:

Curso – 4hs / dia
Estágio – 4hs / dia
Curso e Estágio – 8hs / dia
Estágio – 8h s/ dia

Cálculo da Carga Horária Total
18 dias x 4 hs / dia = 72hs
3 dias x 4 hs / dia = 12hs
64 dias x 8 hs / dia = 512hs
28 dias x 8 hs / dia = 224hs
Total: 820 hs

4. Detalhamento do Curso

4.1. Módulo 1

A) GER_01 – Fundamentos de um Processo Iterativo e Incremental

Objetivos:

- Discutir problemas atuais em engenharia de software;
- Apresentar modelos de ciclo de vida;
- Apresentar os conceitos básicos do RUP;
- Introduzir seis boas práticas de desenvolvimento recomendadas.

Módulos do Curso:

Introdução, fundamentos e metodologias:

1. Introdução
2. Casos de Uso
3. Arquitetura de *Software*
4. Modelos de Ciclo de Vida
5. Visão Geral da Metodologia

Seis boas práticas do RUP:

6. Arquitetura Baseada em Componentes
7. Gerência de Requisitos
8. Desenvolvimento Iterativo
9. Modelagem Visual
10. Verificação de Qualidade
11. Controle de Mudanças

Fases do RUP:

12. Concepção
13. Elaboração
14. Construção
15. Transição

Resumo:

Esta disciplina apresentou alguns dos principais problemas de engenharia de software, que justificam a criação do RUP (Rational Unified Process). RUP é um conjunto de processos que reúne as melhores práticas em engenharia de software, aquelas que são recomendadas para garantir que um produto de software tenha qualidade. O RUP não define uma metodologia, é uma plataforma de processos que fornece atividades, artefatos e guias ligados às ferramentas Rational e à linguagem UML.

Foram apresentados ainda os modelos de ciclo de vida de um software: força bruta, cascata, espiral e o iterativo e incremental. Este último foi mostrado mais detalhadamente, já que é o recomendado pelo RUP. Além do modelo iterativo e incremental, o RUP recomenda: a construção de uma arquitetura baseada em componentes, a gerência de requisitos, a modelagem visual, o controle de qualidade e o gerenciamento de mudanças. Todas essas práticas foram apresentadas em detalhes e também foram abordadas em outras disciplinas.

Por fim, as fases em que se divide um ciclo de vida de um software foram mostradas. Inicialmente, o software passa pela fase de concepção, em seguida começa a fase de elaboração para depois entrar na fase de construção e por último a de transição, como é mostrado na figura a seguir.

Fases de desenvolvimento de software

B) QSW_01 – Qualidade de Software

Objetivos:

- Discutir os principais problemas que desencadearam a busca por qualidade, tanto dos produtos de SW, como no processo;
- Apresentar o que é exigido de um produto de SW de qualidade;
- Introduzir os principais modelos de qualidade para a produção de software;
- Detalhar a norma ISO 9000-3, o CMM e o CMMI, traçando um paralelo entre eles.

Módulos do Curso:

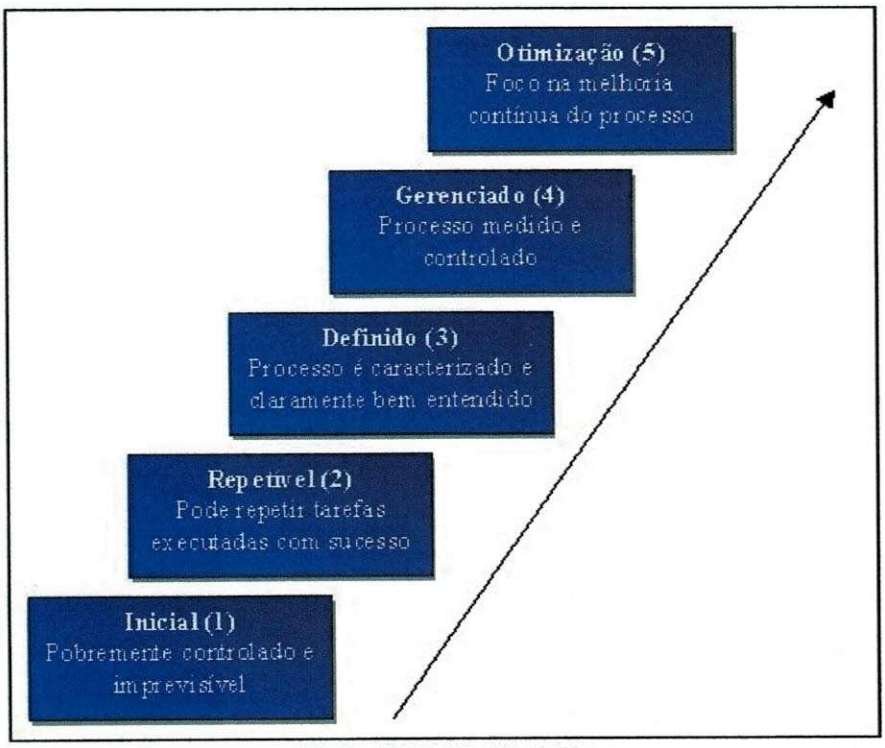
1. Qualidade de software
2. Qualidade de produto de software
3. Modelos ISO
4. Norma ISO 9000-3
5. Modelo SEI/SW-CMM
6. Modelo SEI/SW-CMM Nível 2
7. Modelo SEI/SW-CMM Nível 3
8. CMMI
9. Aspectos Práticos

Resumo:

Esta disciplina mostrou porque a qualidade se mostrou tão necessária nos últimos anos. Além disso, foram mostradas toda a evolução que houve e toda a política de custos que envolvem um processo de qualidade. E também a documentação e as normas exigidas pelos atuais modelos de qualidade.

Foram detalhadas as características de um produto de SW de qualidade como: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade,.

E toda a evolução, do processo de criação de normas e certificação para os produtos e empresas de SW, foi apresentada. Começou pela família ISO 9000 cuja norma ISO 9000-3 é específica para Software, passando por todos os níveis de maturidade do CMM, mostrados na figura a seguir, e chegando até o CMMI. Além disso, foram feitas comparações entra cada um desses modelos de qualidade, mostrando as vantagens oferecidas por cada um, e também os pontos negativos, ressaltando sempre a importância de se esforçar para alcançar tais padrões.



Níveis de Maturidade do CMM

C) PGP_01 – Planejamento e Gerenciamento de Projetos

Objetivos:

- Discutir uma metodologia de desenvolvimento iterativa e incremental;
- Apresentar as atividades de planejamento de projetos de software na óptica de um processo iterativo e incremental;
- Elucidar as dúvidas mais comuns relacionadas ao planejamento de projetos iterativos e incrementais;
- Trocar experiências.

Módulos do Curso:

1. Introdução
2. Modelos de Ciclo de Vida
1. Estimativas de Custo
2. Casos de Uso
3. Visão Geral da Metodologia
4. Fluxo de Atividades
5. Riscos
6. Planejamento
7. Organização de Equipes
8. Boas Práticas
9. Implementando o Processo

Resumo:

Essa disciplina pode ser considerada um treinamento, pois ela não só apresentou as formas de planejamento e gerenciamento de um projeto, mas também teve toda uma parte prática. Houve a simulação de um projeto: a turma foi dividida em equipes e cada uma recebeu uma lista de requisitos exigida pelo cliente para o produto. Então, realizou-se todo um planejamento: analisando os requisitos, seus riscos e prioridades, escolhendo quantos e que tipo de profissionais deveriam ser contratados, os treinamentos que seriam necessários. Foi dividido o tempo para cada etapa do projeto e escolhida a quantidade de iterações que iriam ocorrer, entre outros detalhes. A partir de então, os custos do projeto foram calculados. Tudo isso de acordo com o que estava sendo apresentado no curso. Para finalizar, realizou-se uma reunião com todas as equipes, para cada uma apresentar o seu projeto e tentar vendê-lo ao cliente. A solução real que foi apresentada para a turma, ao final.

D) RAP_01 – Engenharia de Requisitos e Análise e Projeto de Software com UML

Parte 1: Engenharia de Requisitos

Objetivos:

- Apresentar os conceitos básicos do fluxo de requisitos;
- Descrever e exercitar técnicas para levantar e especificar requisitos com casos de uso;
- Descrever e exercitar técnicas para documentação de requisitos não funcionais.

Módulos do Curso:

1. Introdução
2. Casos de Uso
3. Fluxo de Requisitos
4. Levantamento de Casos de Uso
5. Especificação Detalhada dos Casos de Uso
6. Diagrama de Atividades
7. Requisitos Não-Funcionais

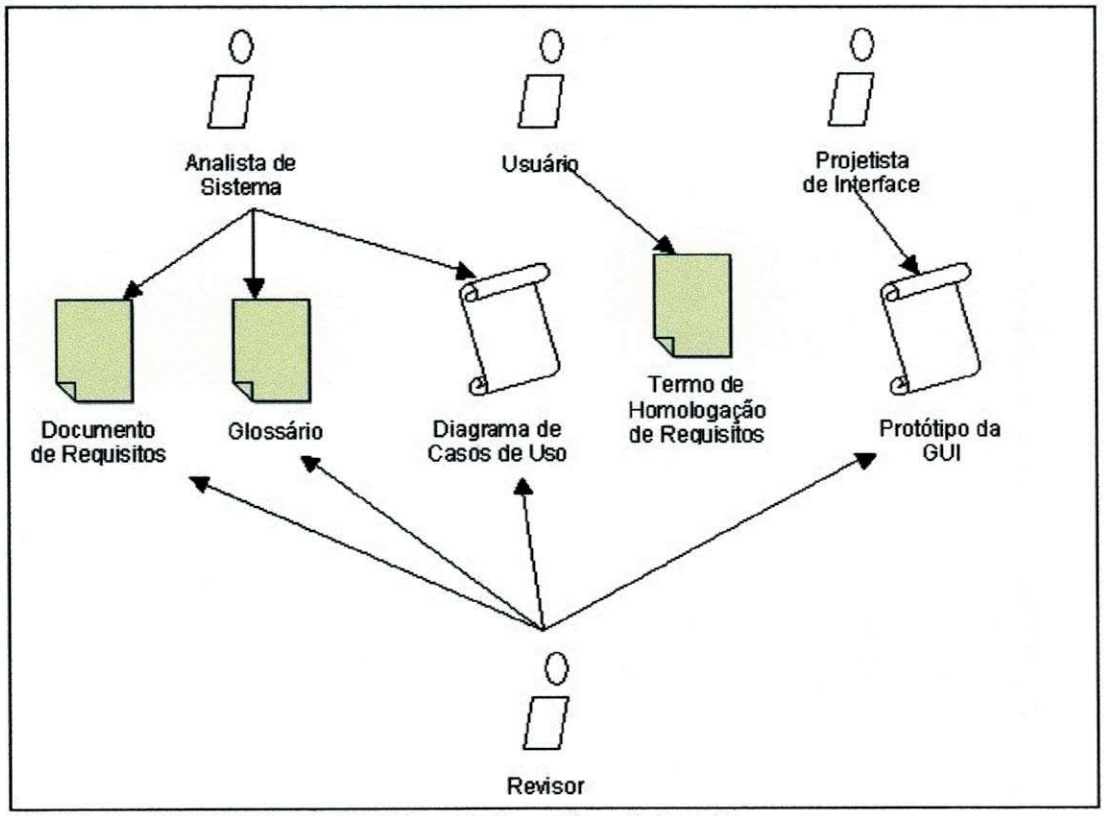
8. Estrutura do Modelo de Casos de Uso
9. Qualificação do Modelo de Casos de Uso

Resumo:

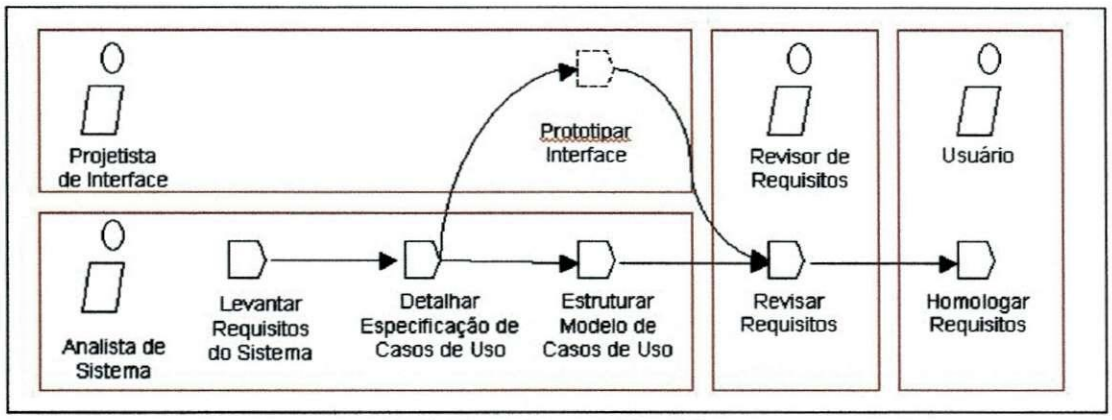
Nessa primeira etapa da disciplina foram apresentados alguns conceitos básicos que seriam necessários à segunda etapa, como os conceitos de: casos de usos, fluxo de requisitos, atores do sistema, as representações gráficas dos diagramas.

Além disso, detalhes do fluxo de requisitos foram mostrados, a saber: as atividades, os principais artefatos produzidos e seus responsáveis, que podem ser visualizados nas figuras a seguir.

Outro conceito importante também foi apresentado, o de requisitos não-funcionais os quais englobam requisitos relacionados a: segurança, usabilidade, confiabilidade, desempenho, distribuição, adequação a padrões e restrições de hardware e software.



Artefatos criados no Fluxo de Requisitos



Atividades do Fluxo de Requisitos

Parte 2: Análise de Projeto OO com UML

Objetivos:

- Apresentar os conceitos básicos do fluxo de análise e projeto;
- Descrever e exercitar técnicas para modelar um sistema orientado a objetos;
- Conhecer os principais diagramas de UML;
- Apresentar o padrão de arquitetura em camadas da Qualiti-CESAR.

Módulos do Curso:

1. Orientação a Objetos com UML
2. Fluxo de Análise e Projeto
3. Analisar Caso de Uso
4. Projetar Arquitetura
5. Projetar Distribuição
6. Projetar Caso de Uso
7. Projetar Subsistema
8. Projetar Cápsulas
9. Projetar Classes
10. Projetar Base de Dados
11. Considerações Finais

Resumo:

Na segunda etapa foi feita uma breve introdução sobre Orientação a Objetos com UML, onde foram mostrados muitos conceitos básicos. Depois, foram mostrados as atividades, artefatos e responsáveis do fluxo de Análise e Projeto.

Em seguida, partiu-se para um estudo mais detalhado sobre casos de uso, incluindo análise e construção de diagramas de casos de uso, de classes, de seqüência e de colaboração. A ferramenta utilizada para a construção desses diagramas foi o Rose Real Time da Rational.

Foi realizado um estudo sobre projeto de arquitetura ressaltando os conceitos de herança, cápsulas e concorrência, protocolos, subsistemas e fachadas. A figura a seguir apresenta um dos modelos de Arquitetura de SW. Foram analisadas oportunidades de reuso, a estruturação em camadas e o agrupamento das classes em pacotes.

Foram detalhados ainda os projetos de distribuição, de casos de uso, de classes, de subsistemas, de fachadas, de cápsulas, entre outros.

Arquitetura de SW - Modelo de Visões

E) LIN_01 – Orientação a Objetos com Java

Parte 1: Fluxo de Implementação

Objetivos:

- Obter uma visão geral dos aspectos básicos de implementação: conceitos básicos, responsáveis e documentos, fluxo de atividades e guias;
- Implementar as classes do modelo de projeto em termos de componentes (código fonte ou executável, arquivo de documentação, etc);
- Definir a organização do código utilizando subsistemas de implementação;
- Testar os componentes como unidades;
- Integrar os componentes desenvolvidos em uma versão executável do sistema.

Parte 2: Orientação a Objetos com Java

Módulos do Curso:

1. Introdução
2. JSDK
3. Sintaxe básica
4. Classes, objetos, atributos e métodos
5. Criação e remoção de objetos
6. Tipos de referência, strings e *arrays*
7. Herança, polimorfismo e ligação dinâmica
8. Modelagem em camadas
9. Classes abstratas e interfaces
10. Exceções

11. Pacotes
12. Componentes AWT

Resumo:

Inicialmente, houve um pequeno curso introdutório de fluxo de implementação que, principalmente, serviu para ensinar boas práticas de codificação, mostrar a atividade de codificação dentro de um projeto de software, como ela se encaixava nas fases do projeto, quem era responsável por essa atividade, os artefatos produzidos, entre outros conceitos básicos.

A segunda parte dessa disciplina foi muito interessante, pois à medida que, cada módulo era ensinado, havia uma atividade prática a ser executada. Como cada módulo era um complemento do todo, ao final do curso um projeto completo havia sido desenvolvido. É essa a forma ideal de ser ensinada uma linguagem de programação, pois só se apreende a programar praticando.

Inicialmente, foi mostrada um pouco da história sobre Java e Orientação a Objetos, além de suas vantagens. Depois foi apresentado o JSDK – Java Software Development Kit, que é o conjunto de ferramentas, bibliotecas e exemplos para desenvolvimento em Java. Foi ensinado ainda como compilar e executar arquivos Java.

Na seqüência, foi abordado: a sintaxe básica Java, suas classes, objetos, atributos e métodos, referências, strings, arrays, conceitos de herança e polimorfismo, classes abstratas e interfaces, exceções, pacotes e componentes AWT.

F) TST_01 – Testes de Software

Objetivos:

- Apresentar e discutir os conceitos básicos sobre o processo de testes;
- Entender como criar e utilizar os artefatos gerados ao longo deste processo;
- Exercitar os conceitos apresentados;
- Apresentar os diferentes tipos de ferramentas que podem ser utilizadas no processo de testes.

Motivação:

- Existe grande possibilidade de injeção de falhas humanas no processo de desenvolvimento de software;
- O processo de testes faz parte da garantia de qualidade de software (SQA);
- Os custos associados às falhas de software justificam um processo de testes cuidadoso e bem planejado.

Finalidades dos testes:

- Verificar se todos os requisitos do sistema foram corretamente implementados;
- Identificar defeitos e erros não identificados;
- Assegurar, na medida do possível, a qualidade e a corretude do software produzido;

- Reduzir custos de manutenção corretiva e retrabalho;
- Assegurar a satisfação do cliente com o produto desenvolvido;
- Produzir casos de teste que têm elevada probabilidade de revelar um erro ainda não descoberto com uma quantidade mínima de tempo e esforço;
- Comparar o resultado dos testes com os resultados esperados a fim de produzir uma indicação da qualidade e da confiabilidade do software;
- Verificar a correta integração entre todos os componentes de software.

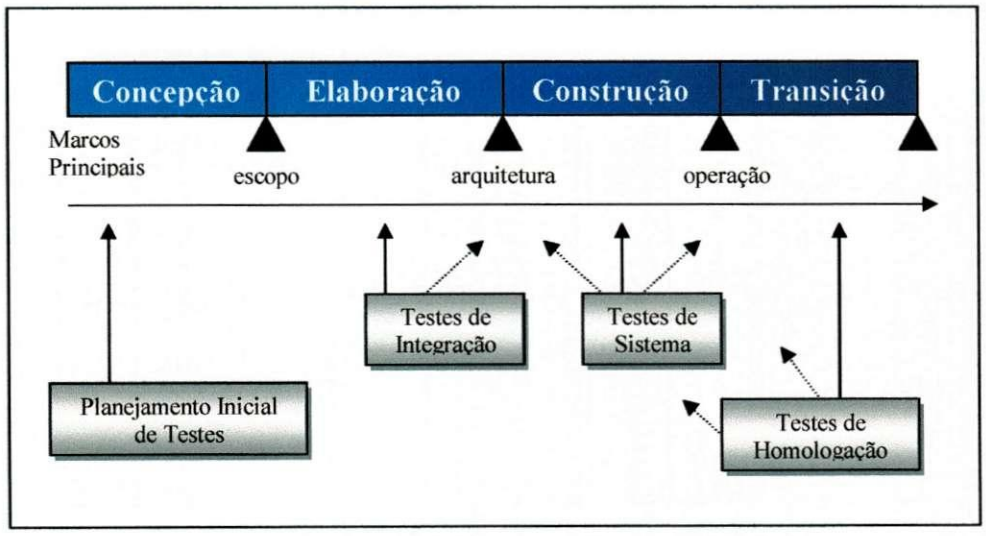
Módulos do Curso:

1. Conceitos chaves
2. Relação com fases e outros fluxos
3. Fluxo de testes
4. Ferramentas de testes

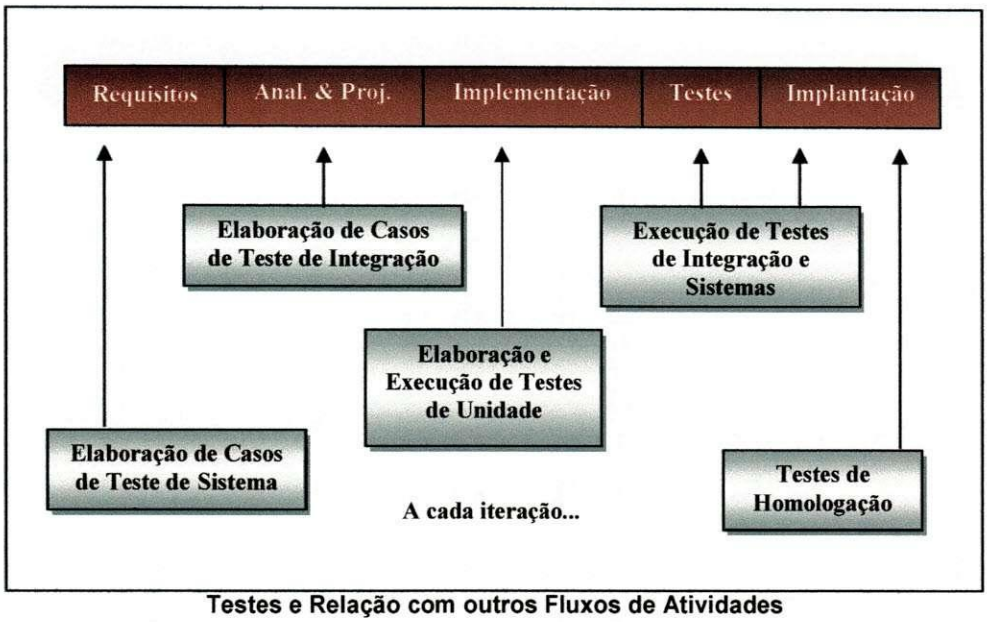
Resumo:

Inicialmente, foi observado o processo de teste que incorpora as seguintes etapas: planejamento, projeto, implementação, execução e avaliação dos testes. Em seguida foram analisados os dois tipos de abordagem de testes, a funcional ("caixa preta") e a estrutural ("caixa branca"). Outro conceito chave discutido foi o de estágios de testes, que são: os testes de unidade, de integração, de sistema e de aceitação (homologação). Cada um desses estágios foi detalhado, e foram apresentados os principais tipos de testes realizados em cada estágio.

O curso teve continuidade, com a relação entre os testes e as fases de desenvolvimento de um SW e com os fluxos de atividades. Essa relação fica mais clara ao observar as figuras a seguir.



Testes e Relação com as Fases de Desenvolvimento



Testes e Relação com outros Fluxos de Atividades

Foi discutido, então, as atividades, artefatos e responsabilidades do fluxo de testes, onde foi visto que existem os seguintes papéis:

- projetista de teste: responsável pela produção do plano de testes, pelo projeto de testes, pela planilha de testes e pelo relatório de avaliação;
- desenvolvedor: aquele que constrói os componentes de teste;
- testador: preenche a planilha de testes e quando necessário, elabora o registro de solicitação de mudanças.

Por fim, algumas das ferramentas utilizadas para testes foram introduzidas, estas, serão abordadas mais detalhadamente, nas duas outras disciplinas de ferramentas de testes, a TST_02 e a TST_03.

G) SCM_01 – Gerência de Configuração

Objetivos:

- Fornecer os principais conceitos relacionados à gerência de configuração;
- Criar uma visão geral de como a gerência de configuração pode ser aplicada ao seu projeto.

Módulos do Curso:

1. Introdução
2. Adoção gradual
3. Padrões
4. CVS
5. *Builds* com ant
6. Bugzilla
7. Gerência de mudanças

8. Fluxo de gerência de configuração de ambiente

Resumo:

Antes de tudo, foram discutidos com a turma, todos os problemas que vinham ocorrendo antes da criação da gerência de configuração. Problemas como: quebra de comunicação, dados compartilhados, manutenção múltipla e atualização simultânea.

A gerência de configuração, ou ainda o controle de mudanças ou versões, como ela também é conhecida, é o processo de identificar, organizar e controlar modificações ao software que está sendo construído, maximizando a produtividade ao minimizar os enganos.

Foram apresentados, então, alguns conceitos básicos de gerência de configuração, como: configuração, baseline, repositório, lock, check-out, check-in, tags, branch, merge, build e release.

Depois foram mostrados alguns padrões usados em gerência de configuração e o CVS (Concurret Versions System). Foi também mostrado o Ant, um software para a construção automática de builds, e o Bugzilla, uma ferramenta de controle de defeitos e melhorias em sistemas e que pode ser usada para gerência de mudanças.

O curso foi finalizado com uma discussão sobre o planejamento de configurações de mudanças, os passos que são recomendados, o ambiente que tem que ser organizado, as atividades e responsáveis, além das ferramentas mais utilizadas.

4.2. Módulo 2

H) LIN_02 – J2ME (Java 2 Micro Edition)

Objetivos Gerais:

- Capacitação inicial para desenvolvimento de software e sistemas móveis com tecnologia J2ME.

Objetivos Específicos:

- Apreensão de conceitos básicos e intermediários sobre sistemas de informação baseados em dispositivos móveis;
- Apreensão de conceitos básicos e intermediários sobre desenvolvimento de software usando J2ME, com ênfase em CLDC e MIDP;
- Apreensão de conceitos básicos e intermediários sobre desenvolvimento de sistemas distribuídos usando o protocolo HTTP;
- Apreensão de conceitos básicos e intermediários sobre desenvolvimento de software usando métodos e processos sistemáticos;
- Desenvolvimento de projetos de software, de natureza didática:
- Técnicas de modelagem de sistemas móveis
- Métodos de processos sistemáticos
- Ferramentas de apoio ao desenvolvimento
- Uso de emuladores de dispositivos móveis

Conteúdo:

Conceitos teóricos:

- Desafios à construção de software e sistema Wireless
- Visão geral da tecnologia Java
- Organização da J2ME
- Detalhes técnicos CLDC
- Detalhes técnicos MIDP
- API MIDP
- API CLDC
- Técnicas para desenvolvimento de MIDLets

Atividades práticas:

- Ambiente para desenvolvimento MIDP
- MIDLets com interface de alto nível
- MIDLets com persistência
- MIDLets com interface de baixo nível
- MIDLets conectados utilizando o protocolo HTTP

Resumo:

Essa disciplina também foi muito importante, as atividades práticas ocorreram no decorrer de toda a disciplina e ao fim, a avaliação dela foi realizada mediante um projeto utilizando alguns dos conceitos ensinados.

A disciplina, que é continuação da LIN_01, aborda a linguagem de programação J2ME, que é a linguagem Java para softwares embarcados, a Micro Edition, linguagem usada pelos softwares de sistemas móveis como os celulares e PDAs.

Inicialmente, foi apresentado um breve histórico, além de alguns conceitos básicos para J2ME, como: CDC, CLDC e MIDLets.

A parte prática teve início com a instalação de todo o ambiente para desenvolvimento e teste de MIDLets usando emuladores. A figura a seguir, apresenta alguns dos tipos de emuladores:

Em seguida, começou-se a utilização do J2ME Wireless Toolkit e foi ensinado como criar um MIDLet HelloWorld, o primeiro. Foi ensinado ainda como compilar e executar o primeiro MIDLet.

Foi apresentado, a seguir, vários elementos visuais da LCDUI de alto nível, como: Displayable, Screen (List, Textbox, Alerts, Form), Ticker. E alguns MIDLets foram desenvolvidos na parte prática.

Foram abordadas a modelagem e a implementação de MIDLets com persistência de registros e componentes da LCDUI de baixo nível, como o Canvas, Image, Graphics e Font.

Por fim, ensinou-se a publicar MIDLets com servidores HTTP e a construir protocolos de comunicação usando servlets.

I) ASW_01 – Arquitetura de Software para Celulares

Módulos do Curso:

- Introdução aos sistemas embarcados
- Arquiteturas de software para celulares
- A arquitetura Plataforma 2000 da Motorola
- Gerenciamento de memória limitada

Resumo:

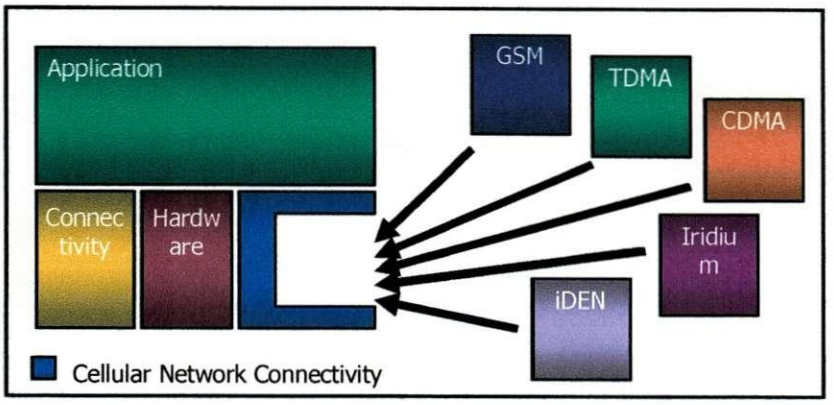
Este curso também foi dividido em duas partes, inicialmente foi apresentada uma introdução sobre sistemas embarcados, e em seguida a arquitetura de software da Motorola foi detalhada.

Na primeira etapa, foi explicado o conceito de softwares embarcados, foram mostradas suas características principais, áreas de aplicações. Discutiu-se ainda sobre os desafios de projeto e os principais processadores, suas vantagens e desvantagens, de acordo com as especificações de projeto foram mostradas. Os principais processadores estudados foram:

- Processadores de uso geral (software);
- Processadores de aplicação específica (ASIPs);
- Processadores de propósito único (hardware).

Além dos processadores foram apresentados os principais dispositivos de entrada/saída e as linguagens empregadas. E finalizando essa etapa, foram mostrados alguns tópicos importantes sobre metodologia de projeto.

Na segunda etapa, a arquitetura Plataforma 2000, ou P2K, da Motorola foi apresentada. Essa plataforma P2K, é a arquitetura padrão para quase todos os produtos da Motorola, as diferenças de software para cada produto específico é acrescentada à P2K, como mostra a figura a seguir.



Arquitetura P2K da Motorola

J) TST_02 – Ferramentas de Testes:

Ferramentas de Teste:

- Automatizam atividades do processo de testes.
- Tipos de ferramentas descritas no curso:
  - Planejamento e projeto de testes: Rational TestManager.
  - Implementação de componentes de teste
    - Teste de GUI (capture/ replay; criação de scripts): Rational Robot.
    - Testes de unidade: JUnit e Rational Test RealTime.
    - Testes de performance: Canoo e JMeter.

Módulos do Curso:

1. Rational TestManager
2. JUnit
3. Rational Test RealTime
4. Rational Robot
5. JMeter
6. Canoo

Resumo:

Neste curso algumas ferramentas utilizadas para testes de software foram estudadas em laboratório, para uma maior familiarização da turma.

Discutiu-se sobre as ferramentas citadas acima, abordando seus detalhes nos módulos do curso, incluindo atividades práticas como: criação de um projeto de testes, implementação, execução e avaliação dos resultados de alguns casos de testes, utilizando o TestManager; criação de um plano de testes, configuração de Controllers, do Timer e dos Listeners de um ThreadGroup, utilizando o JMeter, com isso foi possível até avaliar os resultados por gráficos gerados pela ferramenta.

O software Ant, uma ferramenta em Java utilizada para automatizar processos (javadoc, criação de arquivos jars e testes), também foi apresentada neste curso.

K) TST_03 – Ferramentas para Testes de Aplicações J2ME

Ferramentas Usadas:

- Rational RequisitePro

- Rational TestManager
- Rational ManualTest

Módulos do Curso:

1. Conceitos de teste e certificação de aplicações J2ME
2. Testes funcionais de aplicações J2ME
3. Testes unitários de aplicações J2ME
4. Automação de testes unitários
5. Teste de carga em aplicações J2ME Cliente/ Servidor

Resumo:

Este último curso na área de ferramentas de testes foi bastante interessante, pois, foi quase todo prático e poucos assuntos novos foram apresentados. Como este estava fazendo parte de uma seqüência de duas outras disciplinas anteriores, havia bastante assunto para colocar em prática. Esta disciplina pode ser praticamente considerada um laboratório das outras duas.

A turma foi dividida em equipes, cada uma encarregada de alguns casos de testes de um projeto preparado pelo professor. Foi necessário criar um projeto de testes utilizando o Rational Administrator, depois um projeto de requisitos, utilizando o RequisitePro, em seguida foi necessário elaborar o plano de testes no TestManager e, seguindo o plano de testes, executaram-se os testes manualmente, e foram gerados relatórios com os resultados.

Todos os artefatos produzidos por cada equipe foram entregues ao professor, que fez uma troca entre as equipes, e cada equipe ficou responsável por executar os casos de testes de outra equipe, gerar novos resultados e depois disso avaliar os artefatos produzidos pela outra equipe.

Ao final de tudo, cada equipe teve que ir apresentar para toda a turma seus casos de testes e seus resultados. Para que o projeto inteiro fosse discutido pela turma e cada um pudesse aprender com os erros e acertos dos outros.

L) TCC_01 – Trabalho de Conclusão de Curso

Objetivos:

- Realizar uma atividade de estágio sob a orientação direta de um dos professores do curso;
- Verificar a capacidade que o estagiário tem de propor inovações para suas atividades do estágio, inovações que poderão vir até a ser adotadas pela Motorola;
- Avaliar o rendimento do estagiário, analisando assiduidade e produtividade, além de outros fatores;
- Ao fim, cada estagiário deve ter desenvolvido um TCC, que pode ser individual ou em grupo, e escrito uma monografia, que será avaliada junto com as métricas colhidas durante todo o período de avaliação;

Resumo:

Essa atividade está tendo início agora, toda a equipe de estagiários foi dividida sob a orientação de três professores do CIn. Estes professores estão se reunindo periodicamente com os estagiários, que estão ainda na fase de escolha do tema para o TCC.

Algumas métricas já estão sendo recolhidas, para isso estão sendo utilizadas algumas ferramentas computacionais que geram relatórios de atividades diárias, além das avaliações dos líderes de equipe e dos professores orientadores.

5. Estágio

5.1. Introdução

O estágio é resultado de uma parceria entre a Motorola e o Cin - Centro de Informática da UFPE. Ao todo foram selecionados 30 estagiários. Para a etapa de estágio, realizada sob a orientação da Motorola, na área de testes, os estagiários foram divididos em três equipes de trabalho, coordenadas pelos líderes de equipe. As equipes são: TDMA, KJava e GSM. A divisão ficou da seguinte forma:

- TDMA: 16 estagiários;
- KJava: 2;
- GSM: 12.

O presente relatório descreve atividades realizadas na equipe de TDMA, sobre a coordenação do Team Leader Sr. Edson Fontes. Esta equipe já existia no CESAR e trabalha ligada diretamente ao núcleo de testes da Motorola em New Jersey (EUA). Além do líder Edson, existem ainda na equipe mais cinco pessoas, que auxiliaram no treinamento dos novos estagiários, pois já tinham experiência nessa área.

Sabe-se que a tecnologia TDMA para celulares vem perdendo espaço para a tecnologia GSM e futuramente ela deixará de existir, mas como ainda há uma grande fatia de mercado que ainda usa aparelhos TDMA, a Motorola ainda investe nessa tecnologia. Além disso, ainda existem melhorias sendo realizadas e novos serviços sendo oferecidos para que esses usuários não fiquem insatisfeitos.

Inicialmente foi necessário estudar parte da documentação da Motorola sobre a área de testes em TDMA, para familiarização com a forma de trabalho e a organização. Depois, iniciou-se o estágio propriamente dito na área de testes de CSD (Circuit Switch Data). Em seguida houve uma mudança para a área de testes automáticos com a ferramenta PTF. Algumas atividades pequenas nas áreas de MTS e testes de regressão, também, foram realizadas. E, atualmente, está sendo realizado um trabalho com novas aplicações de aparelhos de lançamentos futuros da Motorola.

5.2. Testes de CSD

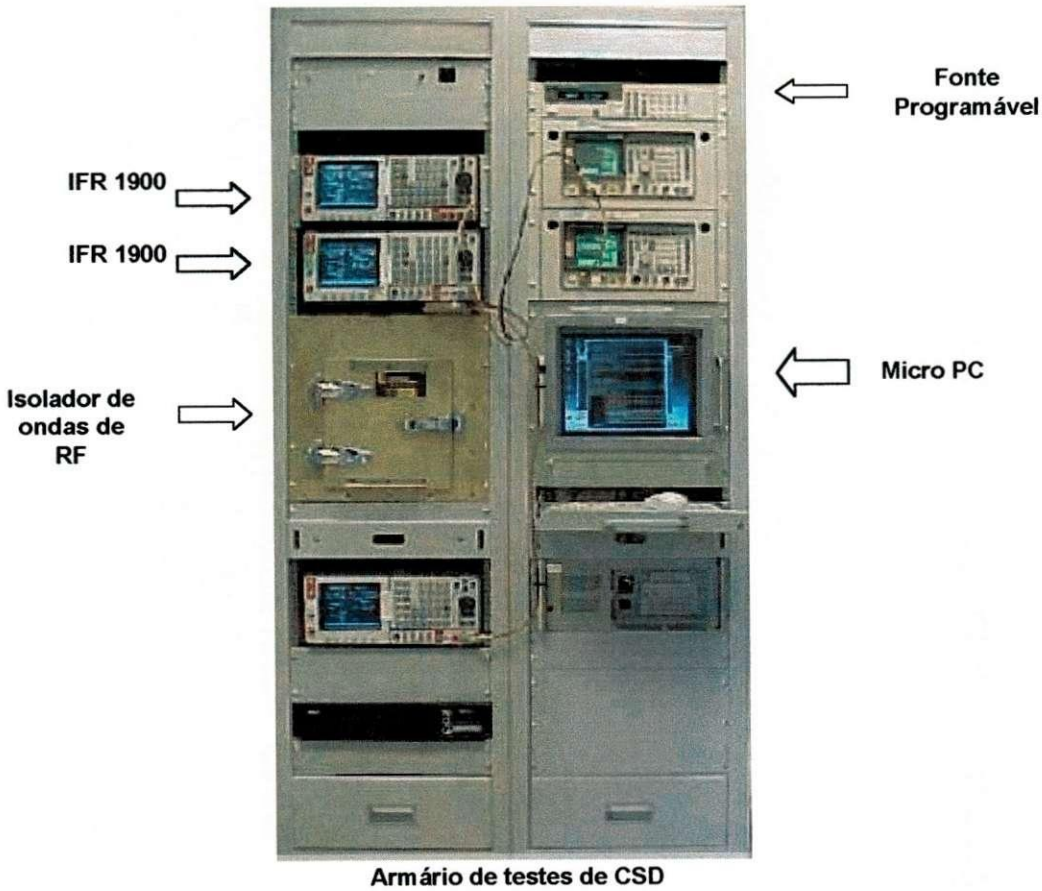
Os testes de CSD (Circuit Switch Data) simulam a comutação telefônica entre dois aparelhos celulares e duas estações rádio-base (ERB). Os procedimentos dos testes são automatizados e executados através de programas feitos em linguagem Java. Os testes utilizam equipamentos sofisticados que medem níveis de potência dos sinais e verificam a sinalização entre os equipamentos, por exemplo. Um exemplo de teste é o do comportamento do aparelho no deslocamento entre células adjacentes de um sistema celular.

Equipamentos utilizados:

- Uma fonte de alimentação programável.

- Dois IFR 1900, analisadores de banda de sistemas celulares, equipamento que simula uma ERB e analisa o comportamento da sinalização entre ele e a unidade de teste.
- Uma Júnior Box, interface entre o PC e os IFR.
- Um micro PC.
- Um aparelho celular (chamado unidade de teste);
- Um isolador de ondas de rádio-frequência (RF Box).
- Cabos de conexão.

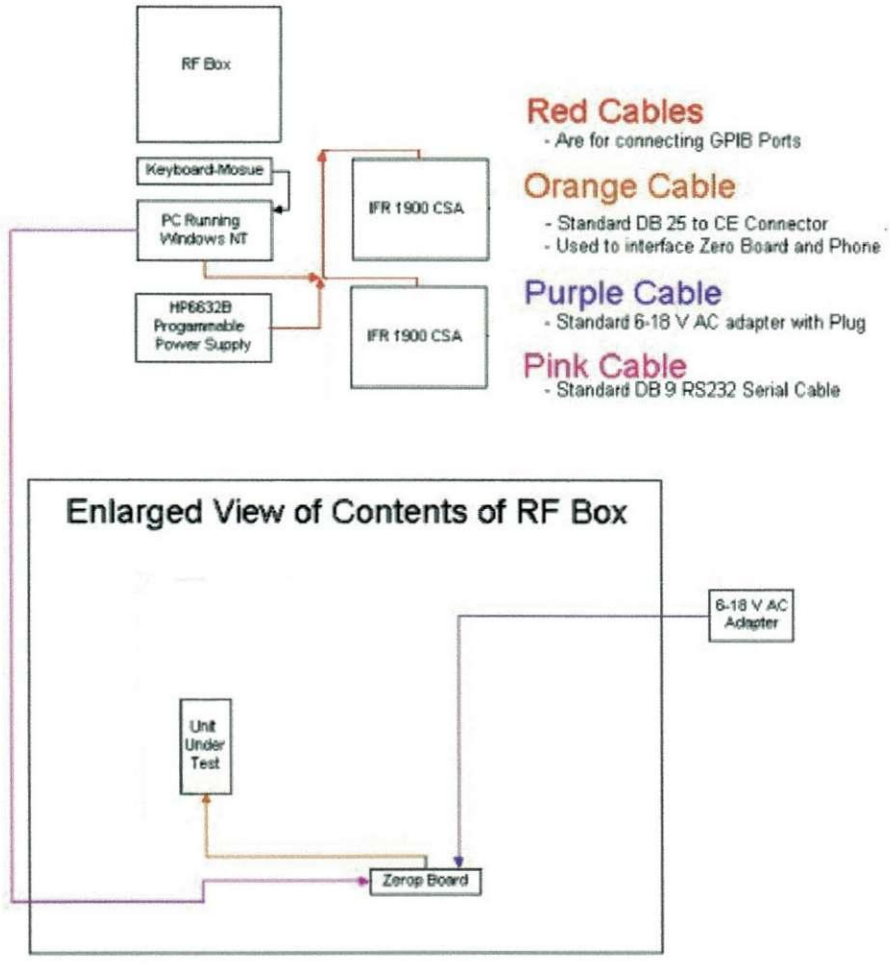
As figuras abaixo mostram alguns dos equipamentos utilizados:



Armário de testes de CSD

Unidade de teste

Vista frontal do IFR 1900



Ligação física entre os equipamentos

Resultados Obtidos:

Esses testes ainda não haviam sido realizados antes pela equipe de testes do CESAR, portanto essa ainda é uma atividade experimental, mas não foi possível conseguir

grandes progressos, pois uma das fontes programável queimou e o processo de reparo é bastante demorado. Sendo assim, os testes foram interrompidos e os estagiários que trabalhavam com CSD tiveram que ser re-alocados para outras atividades.

5.3. PTF

PTF (Phone Test Framework) é uma ferramenta de automatização de testes utilizada pela Motorola. É desenvolvida em Java e formada por um grupo de casos de testes que manipulam o telefone celular como se fosse um usuário. Estes casos de testes verificam o comportamento apropriado do aparelho, observando o que é apresentado no display, ou seja, os detalhes que um usuário veria pressionando teclas, escolhendo menus, fazendo e recebendo ligações, enviando mensagens de texto, etc.

O PTF é rodado num microcomputador com o Java instalado e se comunica com o aparelho celular através de uma interface serial (RS-232) ou de uma interface USB. Em alguns casos de testes são necessários dois aparelhos, usando duas interfaces de comunicação do microcomputador.

Inicialmente teve-se que aprender a instalar o PTF nos microcomputadores. Depois dessa fase foi ensinado como realizar testes automatizados pelo PTF, começando-se a rodar alguns desses para familiarização com a ferramenta e com os testes. Mas esses testes que foram rodados inicialmente não eram testes que a Motorola esperava oficialmente os resultados, já que ainda era fase de treinamento. Quando muitos desses testes já haviam sido realizados e já era possível detectar e até corrigir algumas falhas, passou-se a realizar também testes com prazos de entrega de resultados para a Motorola. Quando não se conseguia corrigir as falhas, era preciso relatar, para que fossem abertas requisições de mudanças (CRs),.

Depois outra etapa foi iniciada, a de desenvolvimento, utilizando o PTF. Como existem ainda alguns testes de TDMA, que não foram totalmente automatizados, testes semi-automáticos de outra ferramenta de teste, o MTS (Móbile Test System), é preciso automatizá-los, utilizando para isso o PTF. Tentou-se automatizar um caso de teste do SMS utilizando inicialmente a linguagem de baixo nível. Após, ter sido aprendido um pouco da linguagem, passou-se a estudar umas bibliotecas específicas para simplificar o trabalho, já que elas permitem o reuso. A ferramenta que contém essas bibliotecas é denominada Application Test Framework (ATF). Atualmente está tentando-se desenvolver casos de teste utilizando o PTF e o ATF. Porém, essa atividade não é prioridade. Assim, quando aparece outra com um prazo definido, é necessário deixá-la e passar para a outra e por isso os progressos têm sido um pouco limitados.

Resultados Obtidos:

- Conseguiu-se instalar já várias versões do PTF.
- Diversos tipos de teste foram realizados, na fase de treinamento, testes que necessitavam de configurações diferenciadas.

- Foi aprendido como identificar diferentes tipos de falhas, como corrigir e relatar falhas;
- Testes com prazos de entrega de resultados à Motorola foram realizados.
- Automatizou-se um caso de teste utilizando PTF e está tentando-se automatizar novos casos de teste utilizando para isso a biblioteca ATF.

5.4. Feature – Feature Interaction

Atualmente está sendo realizado um trabalho novo na área de análise e desenvolvimento de novos testes. Esse trabalho é feito com as novas aplicações (features) que estão sendo criadas para um telefone que ainda será lançado no mercado. Ele é chamado internamente pela equipe de Bonito, mas quando ele for lançado pela Motorola terá outro nome de produto.

Esses testes verificam o comportamento de duas aplicações interagindo entre si (feature interaction), procurando encontrar possíveis falhas que não foram previstas. Por exemplo, alguns casos de teste verificam o comportamento da interação entre a aplicação PhoneBook (agenda) e a de mensagens SMS, analisando o que acontece com a primeira aplicação quando chega uma nova mensagem de texto.

Nessa atividade, há um maior contato com o pessoal de New Jersey através de reuniões feitas com o software de conferência NetMeeting, além de contato por telefone e e-mail.

A atividades resumem-se a:

- elaboração de novos casos de teste e verificação de seus resultados, que devem ser relatados na língua inglesa.
- Envio dos casos de teste à equipe de New Jersey para avaliação.
- A equipe de New Jersey avalia, corrige erros, informa os resultados e escolhe os casos de teste que deverão ser implementados.
- Elabora-se o documento que relata os testes manuais.
- Finalmente é apresentado novamente o documento para o pessoal de New Jersey, que irá automatizar os casos de teste.

Provavelmente alguns desses casos de teste serão desenvolvidos utilizando o PTF.

Resultados Obtidos:

- Elaborou-se um conjunto de casos de teste, que foram apresentados para avaliação da Motorola.
- O resultado da avaliação foi recebido e a partir dele foi elaborado o documento que detalha a realização dos testes manuais.
- Este documento também já foi enviado para a avaliação da Motorola.

- Ele foi enviado de volta para serem feitas alterações.
- Entrou-se, então, numa segunda fase, foi passado um novo conjunto de casos de teste para serem elaborados.

CONCLUSÃO

O estágio em questão ainda não foi finalizado, todas as suas atividades deverão, realmente, terminar em meados de junho, por isso, os resultados obtidos até então podem ser considerados resultados parciais, existem muitas atividades que foram citadas, mas ainda estão em andamento.

Essa turma de estagiários teve as desvantagens de ser a primeira, já que este tipo de programa ainda não existia. Por isso, os resultados podem não ter sido os esperados, mas certamente muito servirá para a próxima turma que terá início logo após o término das atividades da primeira.

Pode-se considerar que o objetivo fundamental do programa foi alcançado, já que a formação de profissionais com especialização em Engenharia de Software foi bem sucedida. O curso que foi dado aos estagiários foi de excelente nível, vieram inclusive profissionais da Motorola dos Estados Unidos e de São Paulo para complementar essa formação.

Além disso, o bom trabalho que vem sendo realizado está começando a colher resultados, pois a Motorola vem passando mais atividades para o núcleo de Recife agora, e trabalhos que exigem uma maior responsabilidade. A próxima turma terá o privilégio de ver consolidado esses resultados e, com isso, terão acesso a atividades bem mais interessantes.

A experiência adquirida tanto com o treinamento, como com a atividade prática foram muito importantes. É muito interessante conhecer o funcionamento de uma grande empresa, ver como ela trabalha, como funcionam os seus processos, e tudo isso foi proporcionado pelo estágio em questão, sem que a formação profissional fosse deixada em segundo plano. Esse foi, com certeza o diferencial desse programa.

REFERÊNCIAS BIBLIOGRÁFICAS

1. Caderno de Apostilas do Curso Seqüencial de Formação Complementar em Análise de Testes, oferecido pelo Centro de Informática e pela Motorola, com o apoio da Qualiti (Qualiti Software Processes).