Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

Gestão do Processamento de Erros e Configuração de Software

projecto realizado na

Nokia Siemens Networks

por

Rúben Bruno de Araújo Mendes

Mestrado em Engenharia Informática
2007

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

Relatório de Projecto

Sobre

Gestão do Processamento de Erros e Configuração de Software

realizado na

Nokia Siemens Networks

por

Rúben Bruno de Araújo Mendes

Responsável pela FCUL: Prof. José Manuel de Sousa de Matos Rufino
Responsável pela NSN: Dr. Pedro Miguel Vieira da Costa Sequeira Borges

Lisboa, Julho de 2007

DEPARTAMENTO DE INFORMÁTICA
Faculdade de Ciências - Universidade de Lisboa
Bloco C6 - Piso 3 - Campo Grande, 1749-016 Lisboa
Tel
& Fax: 351.217500084

Declaração

Rúben Bruno de Araújo Mendes, aluno nº 25576 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o seu Relatório de Projecto em Engenharia Informática, intitulado "Gestão do Processamento de Erros e Configuração de Software", realizado no ano lectivo de 2006/2007 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

FCUL, 12 de Setembro de 2007

_____

Pedro Miguel Vieira da Costa Sequeira Borges, supervisor do projecto de Rúben Bruno de Araújo Mendes, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado "Gestão do Processamento de Erros e Configuração de Software".

Alfragide, 12 de Setembro de 2007

_____

Resumo

A Nokia Siemens Networks é um dos líderes de mercado em electrónica e computadores. É uma empresa de carácter internacional com um vasto número de colaboradores em mais de 50 países. Sendo Portugal o seu maior centro de desenvolvimento de *software* a nível europeu, alberga equipas que a nível cultural são distintas mas em termos de organização de trabalho deverão ser homogéneas. Para que se estabeleça um processo de trabalho padrão com vista a atingir os objectivos da empresa (qualidade, robustez do produto bem como a inteira satisfação dos clientes) o gestor do processamento de erros e configuração de *software* tem um papel muito importante desempenhando funções com vista a atingir a eficiência das equipas, definindo os processos de trabalho mais adequados, como é o caso de assegurar o controlo dos erros, melhoramento e redesenho dos processos de desenvolvimento, bem como o controlo das várias versões e integração do trabalho produzido.

Palavras-chave: Coordenação de erros, Gestão do processamento de erros, Gestão da configuração de software, Gestão e controlo de versões.

Abstract

Nowadays companies such Nokia Siemens Networks are trying to achieve standard work processes in order to reach the quality, product robustness and the entire customer's satisfaction as well.
Error Coordination and the Software configuration management plays a very important role defining and adjusting work processes, assuring the errors control, improving and redesigning new development processes, and controlling all versions and integrations of software.

Keywords: Error Coordination, Software Configuration Management, SCM, Error Process control Management.

Agradecimentos

Carlos Cardoso, Duarte Freitas, João Cardoso, José Reis, Martin Rau, Pedro Borges, Rui Pereira Leal, Susana Cabaço, Susana Mira, Thiago Santos.

Índice

Lista de Figuras

Lista de Tabelas

Abreviatura e Símbolos

A

ADM System Administrator
API Application Programming Interface

B

BD Base de dados
BI Business Intelligence
BPR Business Process Reengineering

C

CCB Coordination Control Board
CO Change Order
CR Change Request

D

DM Data Mining
DSL Digital Subscriber Line
DSS Decision Support Systems

E

EC Error Coordinator
GPE Gestão do Processamento de Erros
EJB Enterprise JavaBeans
EPCM Error Process Control Management ou Gestão do Processamento de Erros (GPE)

F

FDA Function Development Authority
FCB Full Control Board

G

GUI Graphical User Interface

H

HES Home Entertainment Solution
HDTV High Definition Television

I

IP Internet Protocol

J

jBPM Java Business Process Management
JDBC Java Database Connectivity
JSF Java Server Faces
JSP Java Server Pages
JVM Java Virtual Machine
jBPM Java Business Process Management

K

KPN Koninklijke PTT Nederland

L

M

MR Modification Request

N

NTSC National Television System(s) Committee

O

P

PAL Phase Alternating Line
POJO Plain Old Java Object
PL Project Leader

Q

R

RMI Remote Method Invocation

S

SCM Software Configuration Manager ou Software Configuration Management, em português GCS (Gestor(ão) da Configuração de Software).
SCP Secure Copy
SGBD Sistemas de Gestão e Base de Dados
SI Sistemas de Informação
SQL Structured Query Language
SSH Secure Shell
STB Set-top-box

T

TI Tecnologia de Informação
TL Team Leader
TR Trouble Report

U

UML Unified Model Language

V

VPN Virtual Private Network

W

WEB World Wide Web

X

XML eXtensible Markup Language

Y

Z

Glossário

A

Application Programming Interface — Interface que um sistema informático, biblioteca ou aplicação fornece para permitir a utilização de serviços por outro programa informático, ou para permitir a troca de dados entre eles.

B

Bug — Erro, falha, engano ou falta no programa informático que o impede de funcionar como pretendido, ou produz um resultado incorrecto.

Business Intelligence — Conceito explicado na secção de Business Intelligence.

C

CCB — Reunião entre o Error Coordinator e os vários coordenadores de áreas envolvidas nos projectos relacionados com o MR em questão.

Change List — Conjunto de ficheiros que irão ser submetidos no servidor central sendo atribuída uma dada revisão. É um identificador numérico unívoco que permite facilmente encontrar alterações nos ficheiros, quem as submeteu e quais os ficheiros alterados.

Change Order — Um Change Request (CR) ou Trouble Report (TR) originam normalmente um ou mais Change Order (CO's) direccionados aos developers.

Change Request — Pedido de alteração ao documento da especificação funcional.

Class — O mesmo que classe. Em programação orientada a objectos, as classes são usadas para agrupar variáveis e funções relacionadas. Uma classe descreve uma colecção de instâncias encapsuladas de variáveis e métodos, possivelmente com a implementação de ambos com uma função de construtor que pode ser utilizada para criar objectos da classe.

ClearCase — (Rational ClearCase) é uma ferramenta de software para controlo e revisão de código, desenvolvida pela Rational Software uma empresa da IBM.

CM — (Configuration Manager) o gestor de software e configurações que coordena e assegura o controlo das alterações no hardware, software, firmware, documentação, e testes no ciclo de vida de um projecto. O mesmo que SCM.

Compilable — É um estado que garante que o texto escrito numa linguagem para

	computadores (o código fonte) de um programa (ou grupo de programas) chamado por um compilador, é processado com sucesso com o objectivo de criar um programa executável.

D

Data Access Object	Componente que fornece uma interface comum entre a aplicação e um ou mais dispositivos de armazenamento de dados, tais como bases de dados ou ficheiros.
DB	(Database) é um conjunto ou colecção de registos, ou peças de informação, ou conhecimento.
Demilitarized Zone	(DMZ) ou rede de perímetro, é uma área na rede que está entre a rede interna da empresa e a rede externa, normalmente é usada por razões de segurança.
Depot	Estrutura de directórios que contém ficheiros com versões ou revisões no servidor e do qual num determinado estado de tempo todas as workspaces ou views clientes são imagem.
DSL	É uma família de tecnologias que fornecem um meio de transmissão digital de dados, aproveitando a própria rede de telefónica que chega à maioria das residências. As velocidades típicas de download de uma linha DSL variam de 128 kilobits por segundo (kbps) até 24 Mbits/s.
Developer	Alguém que programa computadores ou projecta um sistema consoante os pedidos de uma analista de sistemas.

E

Enterprise ReSource Planning	Método de Business Intelligence (BI) que tenta integrar todos os dados e processos de uma organização num único sistema.

F

FCB	Reunião de vários responsáveis das equipas de desenvolvimento, testes, Error Coordinator, Chief Engineering, e developers envolvidos no processo de erros.
FDA	Responsável pelo desenvolvimento de uma certa funcionalidade.
Firewall	Dispositivo de segurança que é configurado para permitir, recusar ligações determinadas nas configurações de segurança da empresa. As firewalls podem ser hardware ou software. A sua função básica é controlar o tráfego entre a rede de computadores e diferentes zonas de confiança.

Framework Estrutura de suporte definida em que um outro projecto de software pode ser organizado e desenvolvido. Normalmente, uma framework inclui programas de apoio, bibliotecas de código, linguagens de Scripts e outro software para ajudar a desenvolver e juntar diferentes componentes do projecto.

G

GUI (Graphical user Interface) é um caso particular dos casos de uso de interfaces para os utilizadores interagirem com os computadores, aplicam-se imagens gráficas, ícones e juntamente com texto para representar a informação e acções disponíveis ao utilizador. Normalmente as acções são realizadas pela manipulação de elementos gráficos.

H

HES Home Entertainment Solution.

HDTV É um sistema de transmissão televisiva com uma resolução no ecrã significativamente superior ao dos formatos tradicionais (NTSC, PAL).

Himalaya Nome de código usado pata designar o projecto HES para as versões 2.2.x.

I

IP (Internet Protocol) é um identificador de endereço único que é usado para a comunicação com outros computadores numa rede.

J

Java É uma linguagem orientada a objectos desenvolvida pela Sun Microsystems nos anos 90. As aplicações Java são compiladas em bytecodes, o que a torna compilável para outras máquinas nativas em tempo de execução.

JavaBean Componentes de software escrito em Java. A especificação da Sun Microsystems define-os como sendo "componentes de software que podem ser manipulados visualmente numa ferramenta de construção". Apesar de muitas semelhanças, JavaBeans não devem ser confundidas com Enterprise JavaBeans (EJB), uma tecnologia de componentes serverside integrada na Java EE.

K

KPN (Koninklijke PTT Nederland) é uma companhia de IPTV

Holandesa.

L

M

MDS (Metadata Server) Servidor no projecto HES responsável pela gestão pela configuração dos vários serviços, usados pela aplicação cliente Myrioi.

MMDDF Multicast Messaging Data Distribution Framework.

Modification Request Engloba Trouble reports (TR's), Change Request (CR's) e Change Order (CO's). É um tipo de relatório com o objectivo de ajudar no debugging.

MR-Tool Ferramenta utilizada pelos developers para consulta e edição de Modification Request (MR's).

N

NTSC É o sistema de televisão analógico em uso actualmente nos Estados Unidos e em muitos outros países, incluindo a maioria dos países americanos.

O

Open Source Conjunto de práticas que permite que as especificações ou código, programas e funcionalidades sejam abertas ao público.

P

Package Mecanismo Java para organizar classes. Arquivos que pertencem à mesma categoria ou que tenham funcionalidades semelhantes são guardados no mesmo package. Packages podem ser guardados em arquivos comprimidos. O mesmo que pacotes.

PAL (Phase Alternating Line) cuja tradução seria Linha de Fase Alternante, é uma forma de codificação da cor usada nos sistemas de transmissão televisiva criado na Alemanha, usado na maior parte do mundo.

Perforce (P4) é um controlador de versões como é o caso do Rational ClearCase, mas baseado numa arquitectura cliente-servidor com o servidor a gerir uma ou mais colecções de código em um ou mais depots.

PL (Production label or ProdLabel) marca que é aplicada a um determinado número de ficheiros que irão ser compilados para se produzir uma build.

Project Leader O elemento que é líder de um projecto e coordenada várias equipas diferentes. Normalmente envolve contacto com os Team Leaders das equipas.

Proxy É um computador que oferece um serviço de rede de forma a permitir clientes a ligarem-se indirectamente a rede. O cliente liga-se ao proxy server e depois pede uma ligação, ficheiro, ou um recurso disponível a outro servidor diferente. A proxy possibilita o recurso ou serviço ligando-se ao servidor específico servindo o cliente pela sua cache.

Q

Query (Interrogação) Pedido concreto de informação, normalmente palavras-chave combinadas com operadores booleanos e outros modificadores, no campo da recuperação de informação.

R

RSI (RSystems) é uma empresa de software indiana que pertence à Nokia Siemens Networks.

Router É um computador que envia pacotes de dados ao longo da rede para o seu destino, através de um processo conhecido como routing (encaminhamento). Um router age como uma junção entre duas ou mais redes para transferir os pacotes entre elas.

Runnable O mesmo que executável (.exe ou .bin), é um ficheiro cujo conteúdo é interpretado, ou reconhecido pelo computador.

S

Sa (Scientific Atlanta) é uma empresa norte americana que produz software e é responsável por algumas set-top-boxes da Siemens.

Servlet Objecto que recebe pedidos (ServletRequest) e gera uma resposta (ServletResponse) baseada no pedido.

SSH (Secure Shell) é um conjunto de standards e protocolos de redes associados que permitem estabelecer um canal seguro entre uma máquina local e uma máquina remota.

STB (Set-top-box) aparelho que permite que o sinal digital recebido seja convertido em sinal analógico de forma a ser visualizado por uma

TV convencional.

T

Team Leader O elemento que é o líder de uma equipa de desenvolvimento.

TM (Total Manage) é um servidor aplicacional responsável por serviços como a validação de utilizadores e implementação de políticas de segurança.

Trouble report Relatório de erros, tipo de MR.

U

V

VPN (Virtual Private Network) é uma rede privada de comunicação regularmente usada entre empresas de forma a comunicarem confidencialmente em redes não privadas.

W

X

XML (Extensible Markup Language), usada para facilitar a partilha de dados entre diferentes sistemas de informação, em particular em sistemas ligados via Internet

Z

1 Introdução

Ao longo da licenciatura em engenharia informática um aluno adquire competências consideradas fundamentais e importantes para que possa exercer a sua actividade profissional da forma mais notável possível.

Após deixar a faculdade é importante que os seus conhecimentos sejam postos em prática para que os seus esforços sejam também um contributo para a sociedade em que vivemos.

É importante saber como fazer, conhecer os problemas, desencadear acções e chegar a resultados expressos em conclusões, que poderão ser mais tarde equacionadas como óptimas ou suficientes para resolver os problemas iniciais.

A integração no mundo do trabalho e a integração em projectos reais representam assim, passos importantes e decisivos.

Neste capítulo inicial pretende-se definir a motivação, o contexto do projecto e ao mesmo tempo fazer um enquadramento do estágio durante os nove meses na empresa acolhedora.

1.1 A empresa

Fundada em 1847 em Berlim na Alemanha, a Siemens é uma das maiores e mais antigas empresas de electricidade, electrónica e telecomunicações do mundo. Embora tenha tido os primeiros clientes em Portugal em 1876, fornecendo fornos para a Indústria vidreira da Marinha Grande, a representação portuguesa nasce apenas em 1905, e até hoje mantém a sua actividade.

Em Portugal, emprega mais de 3000 trabalhadores nas suas diferentes unidades. Actualmente as suas maiores áreas de acção são as Comunicações, a Energia e os Transportes. O investimento em Investigação e Desenvolvimento é um dos pontos fortes da presença portuguesa da Siemens, existindo nas instalações de Alfragide as instalações da COM ou Siemens Networks.

Em 2005 a Siemens adquiriu a Myrio Corporation uma empresa americana de rápida expansão no mercado do *Home Entertainment* por IPTV (*Internet Protocol Television*). Com esta operação a Siemens encontra-se em posição de se expandir no mercado de IPTV em todo o mundo, tendo para isso estendido o *middleware* da Myrio com outras tecnologias como o *hardware* DSL (*Digital Subscriber Line*) (DSLAMs, DSL *gateways*, etc.) e agregando outras plataformas Ethernet de forma a fornecer produtos inovadores no mercado.

No dia 1 de Abril, a empresa Finlandesa, líder de telecomunicações, a Nokia, fundiu-se com a Siemens Networks. O resultado desta fusão é a jovem empresa Nokia Siemens Networks, passando a ser uma das empresas líderes globais das comunicações com um forte posicionamento em segmentos chave de crescimento de infra-estruturas e serviços de rede *fix*a e móvel.

A Siemens Networks, actual Nokia Siemens Networks (NSN), ficou classificada no topo do ranking das empresas com a melhor solução IPTV. Já com soluções

implantadas em operadoras europeias, como na Bélgica (Belgacom), na Croácia, na Holanda (KPN) em África, como é o caso de Cabo Verde (Telecom), nos Estado Unidos, Argentina e Austrália.

A plataforma de IPTV, é constituída por componentes de *software* de criptografia e protecção de conteúdos, servidores de vídeo "*On Demand*", TV *head-end* e *set-top-boxes*. Todos estes elementos foram reunidos numa única solução *end-to-end*, com o objectivo de assegurar uma interacção transparente de todos os componentes e minimizar os custos, esforços de implementação e riscos para a operadora.

1.2 Descrição do Projecto

1.2.1 Apresentação das equipas

O presente relatório insere-se no projecto de engenharia informática da Faculdade de Ciências da Universidade de Lisboa, e tem como orientador na Faculdade de Ciências o Prof. José Rufino. O planeamento e objectivos do mesmo ficaram a cargo do Eng.º Duarte Freitas para a parte de Gestão do Processamento de Erros e pelo Dr. Pedro Borges na parte de SCM, representado este último a empresa acolhedora, Nokia Siemens Networks como orientador responsável.

O projecto desenvolvido integra-se em duas equipas diferentes uma vez que a natureza dos projectos é inteiramente distinta. A primeira equipa diz respeito ao projecto de gestão do processamento de erros. Esta equipa é formada pelo Eng.º Duarte Freitas e o Thiago Santos (estagiário).

A segunda equipa, equipa de *Software Configuration Management* insere-se no Projecto de *Home Entertainment Solution* sendo esta constituída inicialmente pelo Dr. Pedro Borges e pelo autor deste trabalho permanecendo apenas com estes dois elementos até quase ao final do período de estágio. Actualmente a equipa é composta por cinco pessoas.

1.2.2 Estrutura do Relatório

O presente relatório organiza-se em 8 capítulos, que são:

1. Introdução - É o capítulo do qual esta é a ultima secção e pretende dar uma visão geral do projecto e do seu enquadramento, apresentando a empresa e a equipas que acolheram o aluno.

2. Definição de Objectivos – Apresenta os objectivos que se pretendiam ver atingidos quando se iniciou o projecto.

3. Metodologias e Calendarização – Mostra qual a metodologia de desenvolvimento seguida e apresenta a calendarização dos trabalhos realizados.

4. Gestão do Processamento de Erros – É o capítulo onde é descrita a matéria teórica necessária ao desenvolvimento do projecto de EPCM (*Error Process Control Management*). Pretende introduzir todos os conceitos necessários e criar uma motivação para o desenvolvimento.

5. Concretização do Plano para Gestão do Processamento de Erros – Neste capítulo estão descritas as tarefas realizadas na primeira fase do projecto, de forma a solucionar os problemas de GPE apontados no capítulo anterior.

6. HES – da Configuração de Software - Tal como no capítulo teórico da primeira fase do estágio, este capítulo descreve os conceitos teóricos necessários para o desenvolvimento de projecto de *Software Configuration Management*.

7. Concretização do Plano para gestão e configuração de software – Capítulo onde estão apresentadas as tarefas levadas a cabo na segunda fase do projecto relacionado com SCM (*Software Configuration Manager*).

8. Conclusões – Para além de um sumário do trabalho realizado, este capítulo também contêm um comentário crítico ao plano e tarefas realizadas, bem como as possibilidades de trabalho futuro onde é referido o que poderá ser melhorado.

2 Definição de Objectivos

2.1 Objectivos a atingir no projecto

Este relatório, para além de documentar as tarefas realizadas no âmbito da disciplina do Projecto de Engenharia Informática, pretende também apresentar as linhas gerais do projecto bem como as metas a atingir.

O importante será definir exactamente os objectivos que se pretendem ver atingidos de forma a dotar o aluno de competências técnicas necessárias para que no decorrer do tempo de estágio o seu espírito crítico, aptidões e maturidade profissional sejam uma mais valia para o mesmo e para os grupos de trabalho onde o próprio se insere.

Um objectivo geral para todos os alunos na Nokia Siemens Networks é o de adquirirem formação sobre normas da empresa e a área de negócio, formação sobre métodos de trabalho da empresa, apresentação e formação das tecnologias e ferramentas utilizadas. Especificamente este objectivo geral pode ser particularizado de acordo com a natureza de cada projecto.

Este projecto está dividido em duas partes com objectivos distintos devido as natureza dos dois projectos que compõem o estágio.

Numa primeira fase o objectivo está inteiramente relacionado com o projecto de GPE. Neste projecto procura-se que o aluno seja capaz de manipular as ferramentas já existentes como é o caso do *MR-Tool* e o *Eagle*, e compreender toda a problemática do processo de coordenação de erros.

Tendo como base a documentação e as ferramentas já existentes, pretendeu-se acrescentar novas funcionalidades (métricas, rácios, etc.). Para tal foram elaborados testes exploratórios com objectivo de se tirarem ilações sobre a receptividade das novas funcionalidades por parte dos utilizadores.

Como consequência do objectivo anterior, inicia-se então a fase de manutenção e melhoria das actividades realizadas até ao momento, através da implementação de novas funcionalidades para o *Eagle*.

Na segunda fase do projecto, os objectivos estão relacionados com *Software Configuration Management* para o *Home Entertainment Solution*. Um dos primeiros objectivos nesta fase foi o de criar um manual de *Software Configuration Management*, onde estariam estabelecidas as linhas mestras das políticas de integração de código a seguir para o novo protótipo HES 3.0.PLUS, bem como novas *labels* para as *builds* do projecto, e informações sobre a forma de utilização das ferramentas de controlo de versões, e ainda testes e métricas de forma a estudarem uma futura migração de ferramentas.

Ao longo desta segunda fase foram realizadas tarefas de forma regular inerentes a actividade do SCM tendo como objectivo garantir competências ao aluno nesta área. As tarefas realizadas estão organizadas nos seguintes temas distintos:

1. Gestão de repositórios e componentes – Devido a dimensão do projecto, existe a necessidade de guardar diferentes componentes de *software* diverso, de forma segura, contendo todas as versões ao longo do tempo. Este tópico inclui a gestão de versões e gestão de *software* crítico e complexo.

2. Auxiliar os engenheiros de *software* nas suas actividades usuais – Os engenheiros aplicam ferramentas aos seus objectos (ficheiros). Um sistema de SCM bem implementado tenta providenciar aos engenheiros, os objectos certos, na localização certa. Isto na verdade, refere-se ao controlo das *workspace*s. A compilação e criação de *builds* representam também um forte aspecto desta área.

2.2 Competências Técnicas

É espectável que no decorrer do estágio sejam adquiridas várias competências. Tendo em conta que o mesmo se encontra dividido em duas partes distintas é de esperar que as competências técnicas que o aluno deverá possuir no final sejam elas também diferentes de acordo com a natureza de papel desempenhado. No que se refere à primeira parte do estágio, nomeadamente a parte de GPE no final o aluno deverá ser capaz de:

Recolher toda a informação de erros no correspondente sistema – inclui manipular o *MR-Tool* bem como o *Eagle*, reconhecer as ligações entre os diferentes relatórios de erro, filtrar a informação (falsos erros e duplicação);

Coordenar as alterações e correcções durante o processo de "*bug fix*ing" - havendo procedimentos de erros definidos, garantir que os mesmo são seguidos com vista a resolução de "*bugs*";

Definir estados prioritários bem como intervalos de tempos para correcção de erros;

Iniciar estados de testes após a correcção e verificar os seus resultados (se necessário solicitar novas correcções);

Definir novas métricas, rácios, bem como aplicações de forma a fornecer informação sobre o processo de erros a toda a equipa de gestão;

1. Aprofundamento dos conhecimentos da linguagem Java;

2. Domínio da plataforma de desenvolvimento Eclipse;

3. Aprendizagem das funcionalidades do *Java 2 Enterprise Edition*;

4. Aprendizagem do Servidor *Aplicacional Jboss*;

5. Aprendizagem do sistema de mapeamento Objecto-*Relacional Hibernate*;

Relativamente à segunda parte do estágio, a parte de *Software Configuration Management* são expectáveis as seguintes competências técnicas:

1. Domínio de ferramentas de controlo de versões, *Perforce*;

2. Aprendizagem de *Rational Clear Case*;
3. Domínio de *Shell Script*
4. Domínio de sistema operativo *UNIX* e *Microsoft Windows Server*;
5. Domínio da ferramenta colaborativa *SourceForge*;
6. Aprendizagem dos métodos de qualidade internos da empresa.

3 Metodologias e Calendarização

3.1 Metodologia

3.1.1 Modelo de Processos Internos Siemens Networks

Inicialmente foi dada formação sobre este modelo de processos que é utilizado em toda a empresa, havendo ajustes consoante o tipo de projecto. Este processo foi seguido na fase de Gestão do Processamento de Erros e é também utilizado por todas as equipas de desenvolvimento do projecto HES. Actualmente a nova empresa Nokia Siemens Networks já segue um modelo um pouco diferente.

O diagrama da Figura 3-1, ilustra de um modo geral o processo de desenvolvimento de produtos, onde se podem ver os quatro grandes processos e as principais metas a atingir.

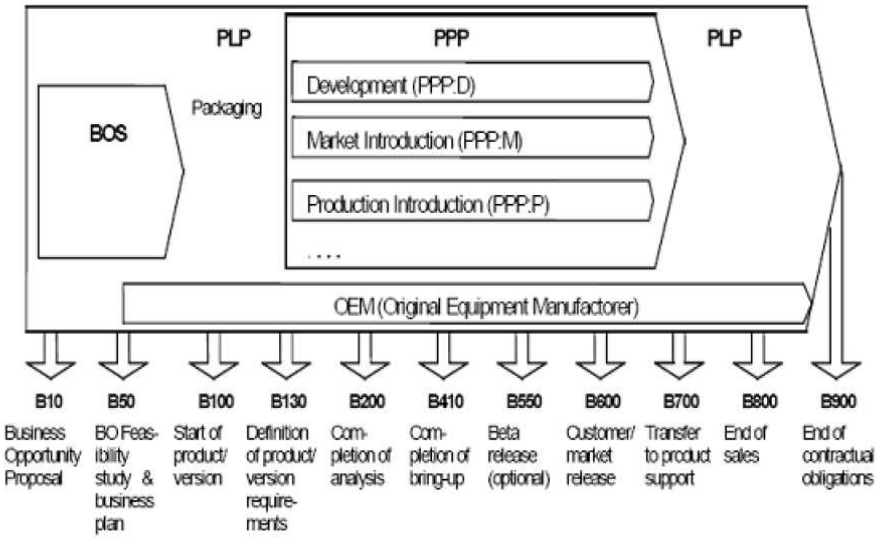


Figura 3-1 - Modelo de desenvolvimento de produtos

Sempre que é identificada uma nova oportunidade de negócio (*Business Opportunity*) é lançado um novo PLP (*Product Line Process*). Este processo de uma linha de produto é transversal e contém mais três processos nucleares. O BOS (*Business Opportunity Scanning Process*), é o processo onde uma unidade de negócio é analisada de modo a verificar a sua exequibilidade e se traça um plano de negócio para essa oportunidade e produto ou linha de produtos que lhe corresponderá. O processo PPP (*Product Provisioning Process*) é o processo relativo ao desenvolvimento do produto. Dentro do PPP são identificados vários "sub-processos".

O PPP: D (desenvolvimento) é o "sub-processo" relativo às actividades R&D (*Research and Development*) seguidas no trabalho realizado no âmbito deste projecto.

3.1.2 Product Provisioning Process Development (PPP: D) 4.0

O modelo de desenvolvimento, corresponde a um metodologia em cascata, com a particularidade de que o inicio de uma nova etapa não implica a conclusão da etapa antecessora, podendo arrancar assim que exista matéria com qualidade suficiente para prosseguir.

A Figura 3-2 seguinte, mostra uma representação mais detalhada do PPP: D versão 4.0.

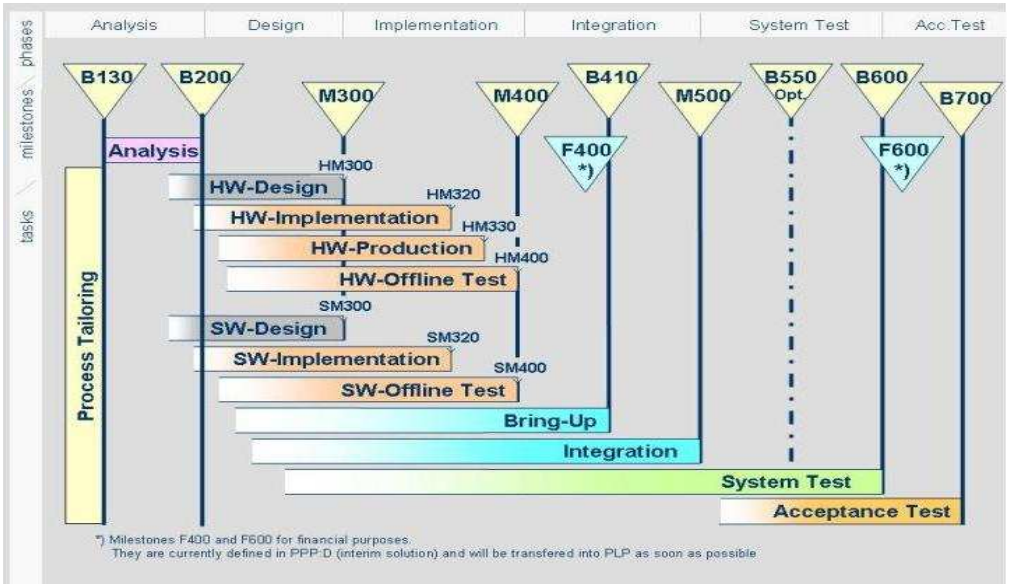


Figura 3-2 - PPD versão 4.0

O desenvolvimento passa por várias etapas:

- *Analysis* (Análise do sistema);
- *Design* (desenho da arquitectura do sistema);
- *Implementatio*n (implementação do sistema);
- *Offline-test* (testes feitos pela equipa de testes de *software*);
- *Bring-up* (onde entra uma primeira equipa de testes) e
- Integração.

Ainda dentro do desenvolvimento existe a fase *System test* (na qual são feitos os testes *end-to-end* e simuladas as condições do cliente).

A última fase é a fase de *Accepance test* (na qual os testes são realizados pelo cliente).
As etiquetas com acrónimo B (como por exemplo B130), são designadas de *Baselines*, *Milestones* comuns aos restantes processos internos da empresa. As etiquetas com acrónimo M (exemplo, M300), são *Milestones* referentes ao processo de Desenvolvimento. Para atingir uma determinada *Milestone*, é necessário cumprir os respectivos requisitos e objectivos previamente definidos.
O início do Desenvolvimento é representado na *Milestone* B130. Marcado pela finalização da configuração da arquitectura do projecto e pela costumização (*tailoring*) do mesmo. É definida a organização e o conteúdo do projecto (orçamento, recursos,

calendarização e qualidade). Nesta altura os requisitos e as funcionalidades já estão completamente definidos.

B200, representa o fim da fase de Análise. Resultando a especificação de funcionalidades e arquitectura do projecto. O planeamento do projecto fica concluído.
Como já foi referido as *Milestone*s marcam a conclusão de uma fase ou de um processo ou sub-processo. Por exemplo a *Milestone* M300, marca a conclusão da fase de Desenho, dentro da qual se considera a HM300, a conclusão do desenho de *Hardware* e a SM300, a conclusão de desenho do *software*.

É de salientar, as *Milestone*s situadas entre a M300 e a M400 (Fim da implementação).
A *Milestone* SM320, marca a conclusão da implementação de *software*, e a *Milestone* SM400 marca a conclusão dos testes por parte da equipa de testes de *software*.

B410, significa que as funcionalidades básicas do sistema estão a funcionar, isto é, o arranque do sistema implementado está estável.

B550, representa a conclusão de testes para uma "Beta *Release*", isto é, quando é testada com sucesso e entregue, uma determinada funcionalidade do projecto destinada a um cliente específico. Na qual todas as restrições ao planeamento de qualidade B600 são identificadas e aceites pelo cliente.

B600, significa que se concluiu a fase de testes do sistema. O conteúdo do projecto foi testado com sucesso e entregue. Esta pronto a ser instalado no cliente.

A *Milestone* B700, marca a conclusão dos testes de aceitação, isto é, o produto foi testado com sucesso pelo cliente e ficou pronto para entrar em fase de manutenção.

3.2 Calendarização

3.2.1 Plano de Trabalho para Gestão do processamento de Erros

Inicialmente o plano do projecto encontrava-se apenas vocacionado para as tarefas de *Error Coordinator*, tendo o seguinte planeamento:

Até 29 de Outubro de 2006: Integração na equipa, leitura da documentação, apresentação das funcionalidades dos equipamentos, formação nas ferramentas pertinentes.
Até 26 de Fevereiro de 2007: Análise e desenho das funcionalidades. Especificação de testes funcionais.

Até 30 de Junho de 2007: Implementação e testes de funcionalidades. Revisão e actualização da documentação previamente produzida.

No entanto a participação neste projecto fez-se apenas até 13 de Novembro de 2006. A partir desse momento iniciou-se uma fase de divisão de tarefas entre o projecto de *Error Process Control Manangement* e para *Software Configuration Management*.

3.2.2 Plano de Trabalho para Hes – Gestão e Configuração de Software

Este plano teve como objectivo introduzir o aluno dentro dos conceitos do HES bem como das tarefas de SCM, através da elaboração de um manual para SCM com vista a se definirem directrizes para o projecto HES3.0. Após a conclusão deste manual e já com um conhecimento de causa, então desempenhar tarefas de forma a auxiliar as várias equipas de desenvolvimento e testes do projecto. O mapa mais detalhado está apresentado na Tabela 3-1:

Tabela 3-1 - Calendário para SCM

ID	Task name	Duration	Start	Finish
1	**Project 1**	**75,5 days**	**Mon 13-11-06**	**Thu 18-01-07**
2	**Ramp-up**	**5 days**	**Mon 13-11-06**	**Fri 17-11-06**
3	Ramp-up new resource	5 days	Mon 13-11-06	Fri 17-11-06
4	**Configuratio Management Policy**	**12 days**	**Mon 20-11-06**	**Wed 06-12-06**
5	Branching strategy	3 days	Mon 20-11-06	wed 22-11-06
6	Client configuration (clientspec)	3 days	Thu 23-11-06	Mon 27-11-06
7	view management	3 days	Tue 28-11-06	Thu 30-11-06
8	Production porcedure/labelling	3 days	Mon 04-12-06	Wed 06-12-06
9	**Functional testing of the proposed policy**	**6 days**	**Thu 07-12-06**	**Thu 14-12-06**
10	Branching strategy	1,5 days	Thu 07-12-06	Mon 11-12-06
11	Client configuration (clientspec)	1,5 days	Mon 11-12-06	Tue 12-12-06
12	view management	1,5 days	Wed 13-12-06	Thu 14-12-06
13	Production porcedure/labelling	1,5 days	Thu 07-12-06	Mon 11-12-06
14	**Configuratio Management Policy - Review**	**4 days**	**Thu 14-12-06**	**Wed 20-12-06**
15	Configuratio Management Policy - Review	2 days	Thu 14-12-06	Mon 18-12-06
16	Configuratio Management Policy - doc	2 days	Mon 18-12-06	Wed 20-12-06
17	**Update of the build process**	**7 days**	Thu 14-12-06	Tue 26-12-06
18	Adapt production scripts	3 days	Thu 14-12-06	Tue 19-12-06
19	Test	4 days	Tue 19-12-06	Tue 26-12-06
20	**User Guide to Configuration Management**	**11,5 days**	**Wed 20-12-06**	**Thu 28-12-06**
21	Production	3 days	Wed 20-12-06	Tue 26-12-06
22	Developer	3 days	Wed 20-12-06	Tue 26-12-06
23	Test and validation	2,5 days	Tue 26-12-06	Thu 28-12-06
24	Presentation slide set	3 days	Fri 29-12-06	Wed 03-01-07
25	**Process migration**	**15 days**	**Thu 04-01-07**	**Tue 09-01-07**
26	Prepare migration	1 day	Thu 04-01-07	Thu 04-01-07
27	Execute migration	4 days	Thu 04-01-07	Tue 09-01-07
28	Post migration support	10 days	Fri 05-01-07	Thu 18-01-07
29	**Project 2**	**65 days**	**Mon 13-11-06**	**Thu 15-02-07**
30	Build Environment Analysis	15 days	Mon 13-11-06	Mon 04-12-06
31	Build process improvements - doc	5 days	Tue 05-12-06	Tue 12-12-06
32	Build process improvements - review	5 days	Wed 13-12-06	Tue 19-12-06
33	Implement improvements	40 days	Wed 20-12-06	Thu 15-02-07
34	**Project 3**	**162 days**	**Thu 15-02-07**	**Fri 01-06-07**
35	Pratical in field	162 days	Thu 15-02-07	Fri 01-06-07

4 Gestão do Processamento de Erros

No passado as alterações ocorriam lentamente para que cada geração de gestores não caía em desuso. Agora, os avanços tecnológicos reciclam mais rapidamente do que a gestão. As organizações agora têm de se prevenir contra processos de gestão obsoletos e os baixos valores de eficiência.

Nesta era de informação massiva e automática a mesma não deixa de ser indispensável, no entanto é necessário filtrar e simplificar a mesma, pois os processos de trabalho dentro das estruturas organizacionais não deixam de ser complexos.

Surgem assim plataformas que facilmente se adaptam aos processos de trabalho e fluxo de informação nas empresas permitindo também a configuração e redefinição dos processos após análises, medidas correctivas e ajustes dos mesmos.

Em empresas de grande infra-estrutura em que os projectos atingem uma dimensão internacional, há a necessidade de se ter alguém que coordene todos os fluxos dos processos, avalie as situações, elabore medidas correctivas de forma ajustar os processos tendo em vista cumprimento dos objectivos, prazos e qualidade do *software* produzido e a gestão racional dos recursos pelas várias equipas.

Inicialmente a Siemens construiu o *MR-Tool* uma ferramenta colaborativa para a gestão do processo de controlo de erros permitindo gerir os erros através de estados ao longo do processo de desenvolvimento. Tendo em vista o BPR (*Business Process Reengineering*) decidiu fazer uma nova aplicação (*Eagle*) que permitisse fornecer de forma prática e dinâmica informação aos intervenientes interessados.

4.1 O MODELO JBoss jBPM

O JBoss jBPM (*Java Business Process Management*) é um modelo de gestão de fluxo de trabalho WFMS (*WorkFlow Management System*) que tem como entrada uma descrição formal do processo de negócio e mantém o estado de execução dos processos delegando actividades pelas pessoas e aplicações. Se a parte de *software* do processo de negócio não é flexível a mudanças, as organizações tendem a despender um enorme esforço na análise do processo de negócio e vêem-se encurraladas na esperança de acertar à primeira.

O WFMS torna fácil a actualização de novas versões do processo de negócio. Como consequência o uso de um WFMS permite mais eficiência, e redução de riscos porque o processo de negócio pode ser desenvolvido de forma interactiva.

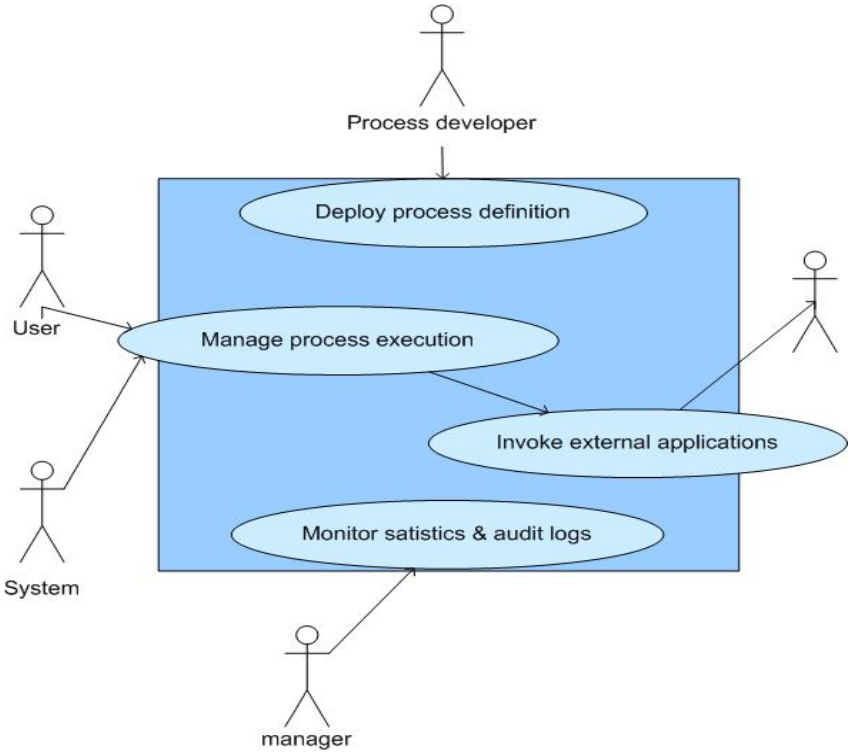


Figura 4-1 - Interface de um WorkFlow Management System

Antes de se automatizar o processo de negócio, a mais árdua mas compensatória tarefa é a de analisar e criar uma descrição formal do processo existente. Os seguintes benefícios podem ser alcançados após a análise:

Melhoria da eficiência – a automatização de muitos processos resulta da eliminação de muitos passos desnecessários.
Melhoria do controlo dos processos – melhoria da gestão dos processos alcançado pela normalização do método de trabalho e auditorias.
Melhoria dos serviços aos clientes – fiabilidade e consistência do processo origina uma maior previsibilidade dos níveis de satisfação dos clientes.
Flexibilidade – controlo de *software* ao longo do processo permite o seu redesenho à medida que as necessidades de negócio mudam.
Melhoria do processo de negócio – foco no processo de negócio conduz à simplificação e aperfeiçoamento.

A introdução de ferramentas colaborativas nas organizações origina benefícios no desenvolvimento de *software* bem com ao nível da área de negócios.

4.2 Management support system

Este tópico apresenta a primeira fase do projecto BPR, que foca o sistema de suporte de gestão assíncrono para as equipas e *project leader*s.

O *Eagle* como ferramenta de BPR inclui um menu de login numa página web de forma a permitir o acesso em tempo real a informação como são o caso das estatísticas dos vários projectos e equipas. Esta informação inclui TRs e CRs (*Trouble report*s e *Change Report*s) fornecidos pela base de dados do *MR-Tool*.

O sistema está dividido em três módulos: O ERP, o jBoss jBPM e o modulo DSS. O ERP consiste na ferramenta *MR-Tool* que actualmente é usada para a gestão de erros e é onde todos os MR e actualizações dos mesmos são feitos.

O módulo jBPM contacta a base de dados *MR-Tool* e analisa a informação dos MRs. Nenhuma alteração é feita à base de dados pelo módulo jBPM. O acesso às bases de dados do *MR-Tool* está restringido a interrogações e protegido a qualquer acção de leitura.

O módulo jBPM é composto por quatro camadas. A *State layer* do processo de negócio que é onde estão definidos os vários estados e o fluxo de controlo dos processos. A *Context Layer* que consiste na associação de variáveis às instâncias do processo de desenvolvimento de forma a guardar informação durante o ciclo de vida do mesmo. A *Programming Logic layer* que combina todas estas peças de *software* com a informação que especifica em que eventos as peças de *software* deverão ser executadas. Exemplos da *Programming Logic Layer* são o envio de mails, o envio de mensagens pelo *broker* de mensagens, carregar informação do ERP, e actualização da base de dados. A *Interface Layer* é usada para permitir aos vários intervenientes (Actor/utilizador) alterarem o estado de um MR, habitualmente este é o evento onde a informação encaixa nas variáveis do processo. É também usado para relacionar os resultados da informação do utilizador bem como os estados do *Workflow* do processo.

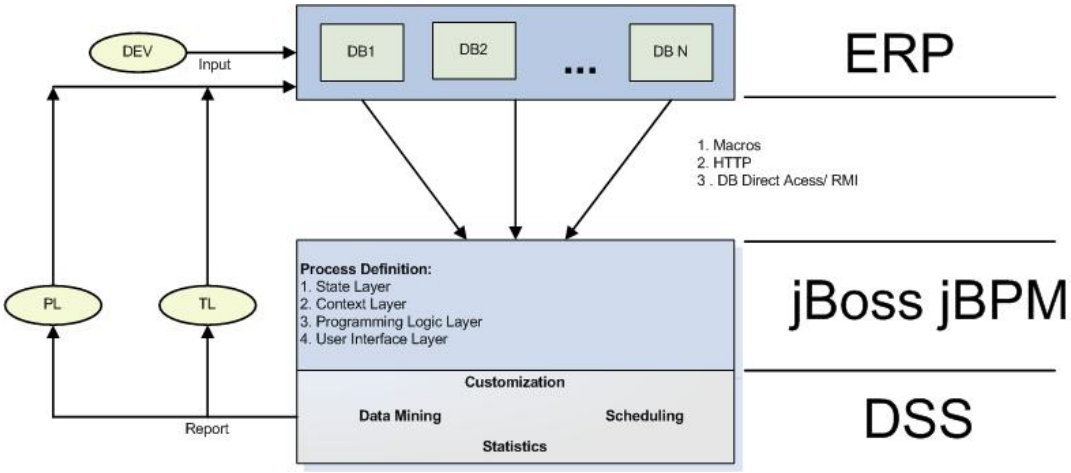


Figura 4-2 - Management Support System

O módulo jBPM está preparado para se adaptar a diferentes tipos de bases de dados. Neste projecto é usado o MySQL. Esta BD foi escolhida devido a usabilidade, capacidade de armazenamento e custo.

O módulo *Statistics* incorpora também o DSS que actualiza alguma informação das estatísticas. O DSS gera informação que preenche os dados para as várias equipas e projectos através do *Data Mining*1, análise e outras técnicas, de acordo com a perspectiva status *report*s do PL/TL. Este módulo ajuda não só a nível operacional, mas

[1]*Data Mining* - É o processo de exploração de grandes quantidades de dados na procura de padrões consistentes, como regras de associação ou sequências temporais, para detectar relacionamentos sistemáticos entre variáveis, detectando assim novos subconjuntos de dados.

também a nível estratégico, ajudando a definir novas metodologias para um trabalho futuro e a organização de recursos.

4.3 Eagle - Error coordination tool

O *Eagle* usa a BD através da entrada de informação das várias bases de dados clientes. Este procedimento tem como objectivo manter actualizada a base de dados MySQL onde se encontra todo os histórico dos MRs de todos os projectos.

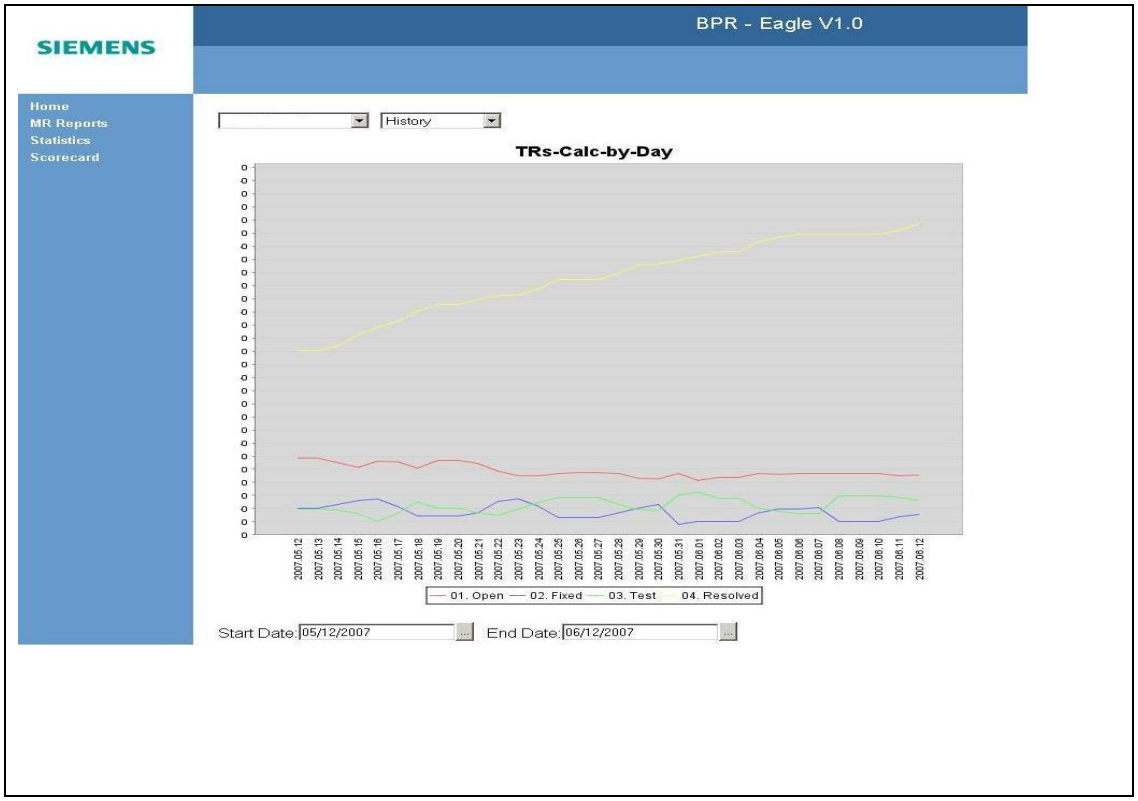


Figura 4-3 - TRs vistos no *Eagle* (intervalo de 1 mês)

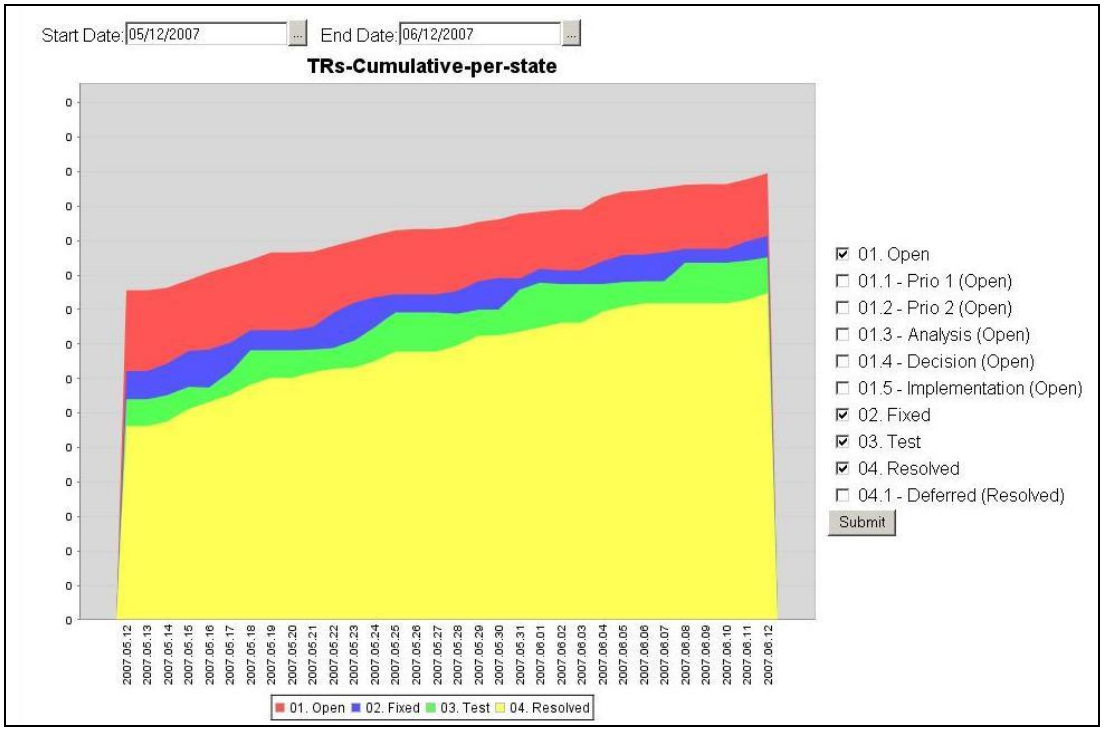


Figura 4-4 - MRs acumulados durante 1 mês

As Bases de dados são sincronizadas com a ferramenta BPR de uma das seguintes formas:

- *Report files* (relatórios) – O *MR-Tool* produz relatórios que contêm toda a informação necessária para actualizar a base de dados central. Implica uma baixa taxa de sincronização (1/2 vezes por dia).
- Via protocolo Http – Procedimento mais flexível mas implica maior esforço de análise. A taxa de sincronização pode aumentar.
- Acesso directo à BD – Esta é a solução óptima garantindo que vamos obter toda a informação necessária, para este tipo de acesso é necessário utilizar um protocolo RMI (Remote Method Invocation).

Na altura da criação da ferramenta, definiu-se que deveria suportar e gerir os fluxos de trabalho das várias equipas, a inclusão dos CRs foi adiada para futuras funcionalidades mas representam uma grande fatia do esforço dispendido pelos membros das equipas de desenvolvimento.

Com a inclusão dos CRs no processo de *error coordination workflow,* os TL podem ver todas as tarefas de um dado membro da sua equipa. Consequentemente, a gestão do desempenho das equipas torna-se mais fácil, eficiente e mais bem organizada.

Os relatórios são gerados para os TL, PM e EC de acordo com algumas especificações na interface, como exemplo temos a frequência dos relatórios, o tipo de informação (interrogações especializadas) os tipos de relatórios (e-mail ou *login*). As mesmas são configuráveis dependendo da equipa ou projecto e preferências do próprio utilizador.

4.3.1 Utilizadores e funcionalidades

Existem três tipos de utilizadores que intervêm directamente com a aplicação originando três tipos de interfaces.

- PL user (*Software Project Leader*)
- TL user (*Software Team Leader*)
- EC user (*Error Coordinator*)

Cada interface é actualizada de acordo com cada papel. Cada utilizador registado após um login é submetido a uma das três categorias.
A página web da aplicação tem um menu de funções de suporte como "*report* customization" e "*Change* pwd", etc. que é comum aos vários utilizadores.

O utilizador PL tem disponível uma visão dos Projectos em termos de MRs e equipas desenvolvimento. É possível configurar alertas e avisos de acordo com as necessidades e restrições dos projectos. Alguns alertas comuns são por exemplo a sobrecarga de trabalho para um dado recurso (*team member*) ou o perigo de não se conseguir corrigir um dado MR no tempo esperado. O PL deve também manter actualizados os MRs de acordo com a BD do *MR-Tool*.

O utilizador TL tem de monitorizar as tarefas dos membros das equipas pelos vários projectos. Este tem visibilidade sobre as tarefas e prioridades de cada membro da equipa de forma a planear e a atribuir recursos limitados de forma optimizada e eficiente. A vista do TL também possui funções similares do PL, mas com uma perspectiva relativa à equipa. O TL deve também manter actualizados os MRs de acordo com a BD do *MR-Tool*.

Para o EC o mais importante é garantir que os procedimentos de Workflow dos MRs estão a ser seguidos e que não existem incoerências resultantes de erro humano. Interrogações predefinidas, verificações automáticas e vários tipos de alertas consistem nas funções principais da *view* deste interveniente. Acima de tudo o EC deverá manter um controlo associado a cada MR visível a todos os utilizadores. Esta tarefa permite prevenir e precaver fluxos erróneos ou comportamentos inesperados do processo.

4.4 Error Coordinator

Ao longo de todo o processo de controlo de erros as funções do EC são evidenciáveis. O EC constitui um dos maiores intervenientes no processo de controlo de erros, sem o mesmo ocorreriam falhas no processo que se tornariam uma verdadeira catástrofe, faltaria o sincronismo e a orientação objectiva. Podemos fazer a seguinte analogia e ver o EC fundamentalmente como o Maestro de uma orquestra ou o óleo na lubrificação de um motor.

Como interveniente no processo de controlo de erros este recolhe toda a informação do impacto de um erro em todo o sistema e subsistemas, coordena as alterações e *bug fix*ing, seguindo as decisões do CCB (*Coordination Control Board*) depois do código congelado (*freeze*). Faz a coordenação de alterações/correcções que serão incluídas na próxima *release* de uma dada aplicação. O seu principal objectivo é garantir que a

correcção irá melhorar o produto e que a estabilidade do mesmo será afectada o menos possível.

Quando requerido, o EC pode organizar uma equipa (*Error Correction Team*) que inclui:

- *Error Coordinator*;
- Sub coordenadores para tarefas específicas;
- FDA (*Function Development Authority* - PL e os vários TL responsáveis no projecto).

4.4.1 Tarefas mais comuns

Como é óbvio uma das tarefas mais comuns do EC é naturalmente analisar os vários MRs, incluindo o reconhecimento de relações entre diferentes error *report*s, fazer a filtragem em busca de duplicados e falsos erros.

O EC é também um coordenador técnico para as correcções (pois busca soluções), especialmente nos casos que envolvem alterações em várias áreas, equipas ou módulos. Esta representa uma tarefa central dos vários sub coordenadores de cada área.

Fixa a *Software release* na qual deverão vir integradas as correcções/alterações propostas pelos MRs.

Estabelece o intervalo temporal e o grau de prioridade para a correcção de um erro. Não obstante, após a correcção do mesmo inicia o período de testes e verifica os seus resultados (se necessário volta a pedir novas correcções).

O EC é um participante assíduo das equipas de gestão de *software,* pois fornece pareceres técnicos sobre o estado de evolução dos projectos enquadrados com as *Milestone*s determinadas pela Siemens. Colabora em cooperação com os sub coordenadores e *System Engineerings* na análise de riscos de partes específicas dos projectos.

Em todas estas tarefas o EC está auxiliado por ferramentas como é o caso do *MR-Tool* ou o *Eagle*.

4.5 MR-Tool

Os TR e os CR e toda a informação descrita nos mesmos durante o seu ciclo de vida num processo são todos geridos por uma ferramenta da Siemens, que é o *MR-Tool* usando as BD.

Cada CR e TR têm um identificador unívoco consistindo em duas letras “versão-pre*fix*o especifico” seguido de seis dígitos, exemplo ‘TP000123’. A correspondente *Change* Order tem o mesmo identificador que os CR/TR mais um su*fix*o individual. Estes identificadores são gerados automaticamente pelo *MR-Tool*. Se um CR/TR for exportado de uma BD para outra, o seu nome e o pre*fix*o mantêm-se, isto é, podem ser CR/TR na mesma BD com diferentes pre*fix*os.

O MR-*Manager* é responsável pela configuração do *MR-Tool*, criação de novos colaboradores, exportar e importar entre as várias BD, e realizar convites para CCB /FCB (Full Control Board).

4.5.1 *Trouble report*s (TR)

Um *Trouble report* (TR) é um documento que descreve uma avaria ou erro relevante no desenvolvimento de uma funcionalidade, de uma especificação funcional ou num conceito operacional. Todos os problemas têm de ser documentados num TR a partir da Siemens *Milestone* B410.

Um colaborador é uma pessoa que está autorizada a trabalhar com o *MR-Tool*, estando na lista de login do servidor de forma a poder trabalhar com a BD.

Durante o processo de desenvolvimento e fase de testes são feitos actualizações com soluções de *software* (*fixes*) dos TR distribuídos. Estas actualizações são chamadas de *builds*. A *build* é uma versão completa do *software* do projecto com o seu próprio nome e é apresentada com a *Build Release Note* (BRN). Qualquer solução entregue adicionalmente à *build* é chamada de Patch.

Uma propriedade é uma característica de um MR. Cada propriedade tem definida um valor e ajuda a distinguir os MRs usando-as nos filtros do *MR-Tool*.
Destas podem-se destacar as propriedades:

- State – os vários estados pelo qual um TR pode passar ao longo do processo de controlo de erros (new, sent, wait, prepared, test, active, implemented, retest, observation, closed, exported, not concerned, not *fix*ed, fcb, rejected, duplicated, unsolved);
- Priority – a prioridade reflecte a necessidade de se obter a correcção de um erro;
- Severity – reflecte os requisitos do cliente ou seja o impacto de um potencial erro no cliente.

4.5.1.1 Fluxo dos TRs

O diagrama Figura 4-5 dá uma visão da forma como evoluem os TRs ao longo do processo de correcção de erros:

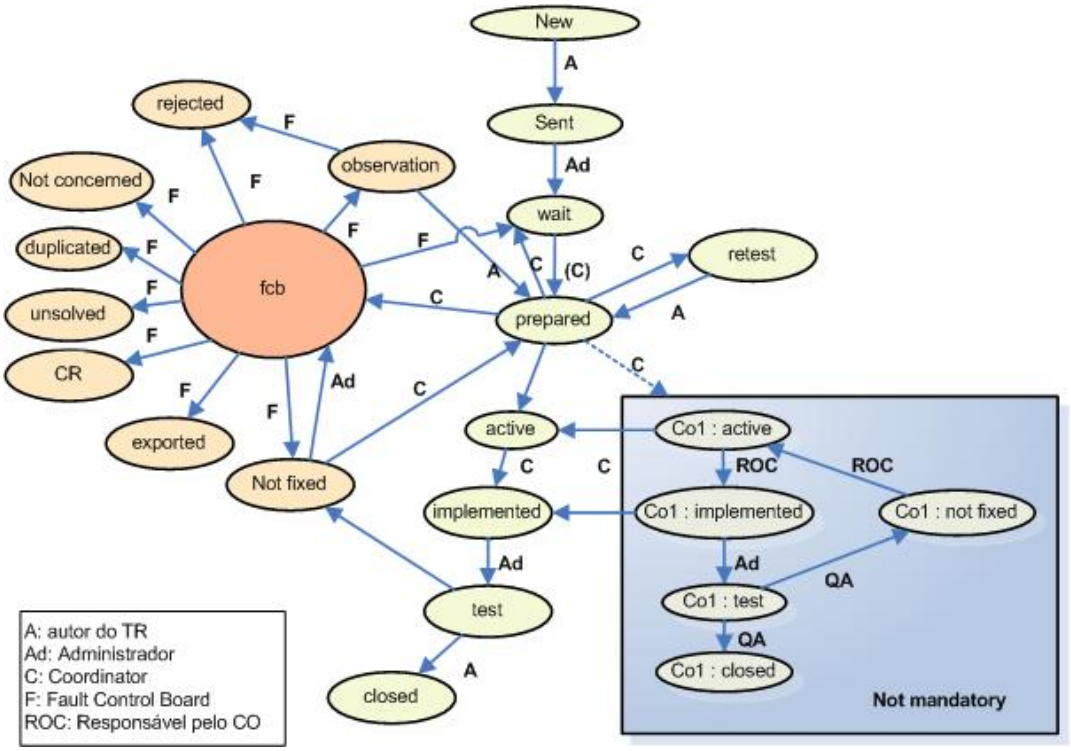


Figura 4-5 - TRs Siemens workflow

A transição de estados reflecte um processo criterioso fundamentado pela análise do TR para que toda a informação necessária para a resolução do mesmo seja alcançada. Os vários intervenientes permitem a resolução do problema, mas é da responsabilidade do EC verificar o seguimento do processo para que não incorram erros ou atrasos não desejáveis, há que ter em mente que o facto de se tratar de um TR existe sempre a possibilidade do impacto ser prejudicial ao desenvolvimento existente, bem como (de uma forma mais catastrófica) a qualidade do *software* e imagem junto do cliente.

A Tabela 4-1 mostra os critérios e procedimentos a seguir pelos vários intervenientes no processo de controlo para os TRs:

Tabela 4-1 - Metodologia dos TRs na Siemens

Do estado	Para o estado	Comentário	Por decisão	Acção requerida	final
new	sent	O autor escreve um TR e envia-o.	Autor		Não
sent	wait	Admin faz uma verificação rápida	Autor	Verificar	Não
wait	prepared	A pessoa questionada dá uma resposta	*Coordinator*	*Coordinator* responde	Não
prepared	retest	*Developer* precisa mais informações (traces e informações sobre o retest).	*Coordinator*	Escreve a reposta, com descrição detalhada do retest (trace levels, etc.)	Não
prepared	fcb	O *coordinator* quer discutir o TR na FCB. Talvez queira rejeitar, passar a duplicate ou mudar para CR. Será possível	*Coordinator* MR-*Manager*	Resposta escrita	Não

		alterar também o subsistema depois da FCB.			
fcb	Duplicate	O mesmo TR já existe.	FCB	A correlação tem de ser indicada entre os dois TRs	Sim
fcb	Rejected	TR é: erro do *developer*, comportamento normal, na funcionalidade, não reproduzível	FCB	Tem de ser dada uma resposta com a razão	Sim
fcb	wait	FCB decide: investigação deve prosseguir, o coordenador talvez altere o sistema	FCB MR-*Manager*		Não
Test	closed	Os testes concluíram com sucesso, o problema está resolvido, o TR pode ser fechado	Autor/*Tester*	Os *teste*rs tem de dar uma resposta, incluir a *build* usada após B600, O MR é documentado pelo MR *Manager* na *Release Note* do SP correspondente	Sim
Test	Not *fix*ed	Testes falharam, TR tem de ser investigado novamente. Este estado é importante para as estatísticas.	Autor/*Tester*	*tester* tem de dar uma resposta, incluindo a *build* usada	Não
Fcb	observation	Erro não é reproduzível. Se o mesmo não ocorrer num tempo limite (1 ou 2 meses), o TR passa a rejected – se nada foi entregue - ou closed – se um *bug fix* foi entregue.	FCB	*Tester* têm de escrever uma resposta com as razões. FCB define um *deadline* para observação. *Teste*rs deverão indicar o que fazer se o erro ocorrer novamente.	Não
Retest	FCB	Erro não é reproduzível	FCB	*Tester* têm de escrever uma resposta com as razões.	Não
observation	prepared	Erro não reproduzível mas o problema voltou a ocorrer	Autor	Mais informação deverá estar disponível	Não
observation	reject	O erro não reproduzível Atingiu o tempo limite e não ocorreu, não tendo sido nada entregue com correcções.	FCB		Sim
fcb	not concerned	TR não é relevante. Por exemplo o TR foi indevidamente importado de outra BD.	FCB, MR-*Manager*		Sim
fcb	exported	TR foi exportado de outra BD.	FCB	MR-*Manager* tem de exportar o TR para a BD correspondente. Deverá submeter uma nova resposta com um novo *trouble ticket* copiado do original *trouble ticket*.	Não
fcb	Unsolved	TR não será resolvido	FCB	Referenciado na RN	Sim
Not *fix*ed	Fcb	Decidir na FCB, que componente tem de ser corrigida novamente se não for óbvio	Admin		Não
Not *fix*ed	prepared	É necessária uma segunda correcção	*Coordinator*		Não

Prepared	Active	Com CO	Tool	Pelo menos um CO foi escrito e tem o estado active. O campo *fix*ed in *build* tem de ser actualizado no *MR-Tool* pelo *Coordinator* depois de todos os ROCs decidirem que a *build* integrará a correcção.	Não
Prepared	active	sem CO	*Coordinator*	Razão pelo qual o TR foi identificado. Se apenas uma componente foi afectada, não é necessário criar um CO. O número da *build* com a correcção tem de ser inserido no campo *fix*ed in *build*	Não
Active	Implemented	sem CO	*Coordinator*	A correcção está numa *build* especial que tem de ser inserida no campo *fix*ed in.	Não
Active	Implemented	Com COs	*Coordinator*	Todos os COs alcançaram o estado implemented	Não
Implemented	Test	*Build* que é mencionada no campo *fix*ed in field do *MR-Tool*	Admin	*Build* com a correcção disponível para testes	Não
Retest	Prepared		Autor	Retest feito, informações adicionais escritas nas respostas	Não
CO: sent	CO: Active		*coordinator*	A mesma *build* para correcção foi definida por todos os ROCs. O número da *build* tem de ser inserido na *release build* pelo *coordinator*. Tem de ser submetida pelo responsável deste CO e inserida no campo pelo responsável.	Não
CO: active	CO: implemented		ROC	Numero da *build* inserido no campo *fix*ed in. A correcção é entregue nesta *build*.	Não

CO: implemented	CO: test		Admin	*Build* com a correcção disponível para ser testada.	Não
CO: test	CO:closed		QA ou Admin	Se o CO pode ser testado sozinho é fechado pelo QA. Se o TR apenas pode ser testado como um todo, os COs serão fechados pelo Admin, depois do TR ser fechado pelo autor do TR.	sim
CO:test	CO: not *fix*ed		QA	CO não corrigido	Não

4.5.2 Change Request (CR)

Um CR (*Change* Request) pode ser escrito por qualquer membro do projecto incluindo um gestor de produto, técnico de vendas e serviços. Os CRs vêm de fora do projecto, como companhias associadas, e a sua entrada é submetida pelo gestor de produto ou o técnico de vendas de forma a manter-se uma distinção entre o planeamento de uma nova versão do *software* no futuro e as alterações necessárias na versão a decorrer.

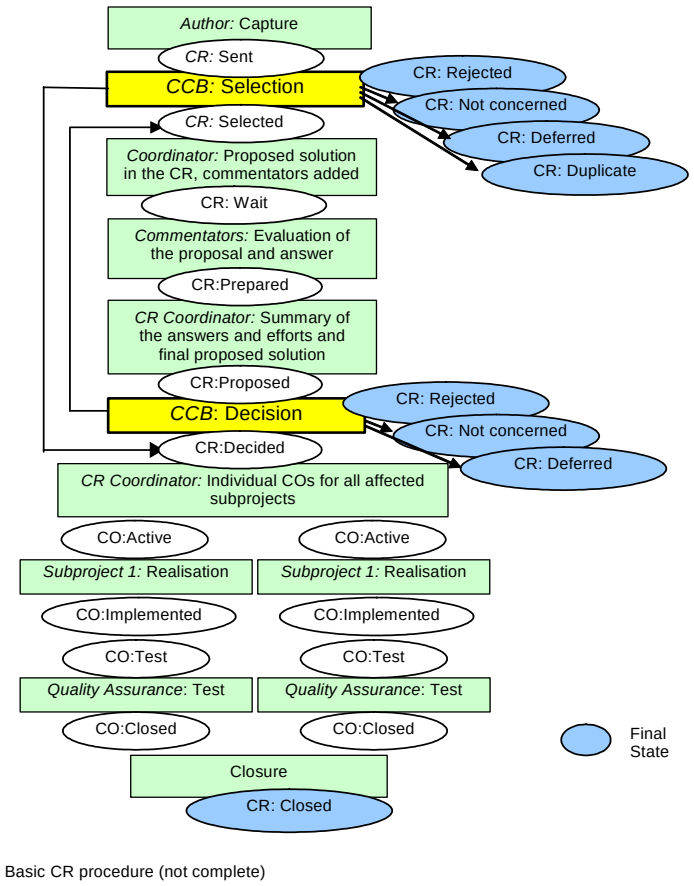


Figura 4-6 - CRs Siemens Workflow

4.5.2.1 Captura

O CR é capturado pelo *MR-Tool* através da entrada de um MR (*Modification Request*) do tipo *Change* Request. Enquanto o autor não terminar o CR, esta mantém-se no estado "New" e não é visível. Assim que o autor o enviar, este torna-se publico e é automaticamente passado para o estado "Sent"
O autor deve propor um coordenador para o CR, que é informado automaticamente pelo *MR-Tool*, assim que o CR é enviado o coordenador pode preparar-se para a selecção. O coordenador pode ser alterado numa CCB se o CR for seleccionado.

4.5.2.2 Selecção

Antes qualquer análise começar o CR, é apresentado à CCB para selecção, que quer dizer que a CCB decidirá o que fazer com o mesmo:

- "Selected" – O CR é proposto para mais análise e um coordenador é nomeado ficando com sub projectos afectos a si. O CR deverá ser submetido para decisão, em conjunto com uma proposta de solução.

- “Rejected”, “Deferred”, “Duplicated” ou “Not Concerned”: O CR não é considerado para esta versão do projecto.
- “Decided”: (Excepto em casos urgentes) se a solução para o CR já está prevista, neste caso o CR não tem de ser submetido para a CCB.

4.5.2.3 Análise

Depois do CR ser seleccionado, o coordenador é responsável pela análise e especificação da solução técnica. O primeiro caso é fazer uma proposta da descrição técnica de como resolver o problema do CR. Esta proposta dá entrada no *MR-Tool* como “Proposed solution”, o coordenador verifica se todos os subsistemas afectos foram identificados e muda o estado do CR para “Wait”.
O estado “Wait” implica que as pessoas na lista do campo “Answer from” serão automaticamente notificadas para darem um parecer sobre o CR. Os sub projectos afectos analisam as soluções propostas e dão um parecer sobre no *MR-Tool* sobre a praticabilidade, esforço estimado e possíveis repercussões nas futuras versões e funcionalidades. Como a solução proposta pode ser estendida durante o processo de análise, é importante sempre referenciar o nome e a data na resposta. Quando todos *sub coordinators* dos sub projectos tiverem dado o seu *input*, o estado do CR é automaticamente alterado para “Prepared”.

Depois de obter todas as respostas dos *sub coordinators* dos vários subsistemas, o coordenador analisa os resultados e estabelece uma solução final, alterando o estado do CR para “Proposed”. Isto quer dizer que o CR está pronto para uma decisão da CCB.

É da responsabilidade do coordenador preparar completamente o CR e permitir uma decisão eficiente da CCB, isto é, se as respostas dos vários subsistemas contiverem objecções, contradições, alternativas, ou questões em aberto, o coordenador deverá encontrar uma solução fiável e consistente antes do CR poder ser decidido. Isto quer dizer evidentemente, que poderá ser necessário ter de repetir uma reunião técnica para se encontrar uma solução. Se novas ou adicionais propostas forem necessárias, o coordenador pode alterar o estado de volta para “Wait” e repetir esta parte do processo.
A solução proposta pode, como é claro, ainda ser rejeitada ou deferida. As razões para isto devem ser descritas numa proposta.

4.5.2.4 Decisão

O coordenador assiste à CCB onde o seu CR é discutido.
A CCB decide se a modificação proposta pode ser efectuada. A decisão poderá ser:

- “Decided”: O CR deve ser realizado.
- “Selected”: são necessários mais esclarecimentos.
- “Rejected”, “Deferred”, “Duplicated” ou “Not Concerned”: o CR já não é considerado para esta versão de projecto.

Para um CR decidido, o coordenador escreve um CO (*Change* Order) para cada subsistema afectado. Os COs devem ser postos a “Active” pelo coordenador. Neste CO, o coordenador também põe a versão a que diz respeito o CO, incluído se possível o

número da *build*, o esforço a ser despendido pelo sub projecto, e o responsável pela qualidade.

A CCB pode alterar o estado do CR para "Decided" e definir determinadas condições para a modificação. Neste caso, o coordenador considera este aspecto também para o CO final. Se ele encontrar grandes problemas durante esta fase, o CR tem de ser submetido a CCB novamente.
Se a decisão for a de cancelar a funcionalidade ou adiá-la, cabe ao PL (*Project Leader*) a responsabilidade de actualizar a informação na base de dados.

4.5.2.5 Modificação

Para cada sub projecto o coordenador responsável aplica a modificação de acordo com a *Change* Order. Para as alterações no *software*, o respectivo sub projecto faz a alteração do estado para "Test" e submete o número da *build* no campo "*Fix*ed in" depois de feita a modificação na respectiva *build*. No caso de alterações na especificação o estado deve ser alterado para "Test".
Para alterações na documentação do cliente, *Release Note*, ou na lista de Funcionalidades do produto, e também para o grupo de testes, a *Change* Order não vai para o estado de "Test" mas sim para "Closed".
No caso da CO não poder ser implementada como decidido, a *Change* Order deve ser novamente submetida à CCB. Até mais qualquer decisão o coordenador decide como proceder e coordena as acções necessárias.

4.5.2.6 Verificação e Fecho

O estado "Test" está directamente ligado com o responsável pela qualidade, que, agora verifica se o CR foi implementado correctamente. Durante a verificação são realizados testes. Se ocorrer algum problema estes são *report*ados como *Trouble report*s, mas com referência ao CR que o deu origem. Quanto mais cedo os testes tiverem sido realizados e os erros capturados como TRs, mais cedo o respectivo CO é fechado pelo *Tester*.
Depois de tudo isto o CR atinge o estado terminal "Closed".
Se o estado de um CR é "Deferred" e a próxima versão atingiu o B130, deverá haver uma decisão explícita da CCB para a próxima versão.
Se a próxima versão ainda não atingiu o B130, o CR é capturado como uma funcionalidade e ainda é implementada nessa mesma versão.

Tabela 4-2 - Metodologia dos CRs na Siemens

Do estado	Para o estado	Comentário	Por decisão	Acção requerida	final
new	sent	O autor escreve um CR e envia-o.	Autor		Não
sent	Selected/wait	CCB analisa o CR. O *Coordinator* tem de dar seguimento ao CR.	CCB	Verifica se necessário	Não
wait	prepared	São pedidas informações sobre o CR	*MR-Tool*		Não
Wait, selected, prepared	proposed	Proposta sobre o que se ira fazer.(rejeitar também é uma proposta)	*Coordinator*	A proposta tem de ser aceite por todos os responsáveis dos components afectados. São dadas estimativas de	Não

				esforço. A proposta tem de ser inserida no *MR-Tool*.	
proposed	Duplicate	O CR já existe, bem como a forma de o tratar no TR existente.	CCB	A relação tem de ser indicada em ambos os TRs.	Sim
proposed	Rejected	O CR não será seguido.	CCB	As razões têm de ser escritas	Sim
proposed	deferred	O CR não se realizará nesta versão.	CCB	As razões têm de ser escritas	Sim
Proposed	decided	Vai ser implementado!	CCB		Não
decided	Active	Se não é necessário nenhum CO a *build* com correspondente é inserida no campo *fix*ed in do *MR-Tool*.	*Coordinator*	Todos os COs que foram escritos e passaram a active com vista a serem integrados numa *build* específica.	Não
Test	Closed	Sem CO	Autor	O Autor testou os resultados. Se o teste falhar um TR tem de ser aberto.	Sim
Test	closed	Com COs	Admin/Autor	Se o CO pode ser testado sozinho é fechado pelo QA. Se o CR só pode ser testado como um todo, o CO será fechado pelo Admin depois do CR ser fechado pelo autor do CR. Se os testes falharem um TR tem de ser aberto.	Sim
Prepared	active	Com CO	Tool	Pelo menos um CO foi aberto e chegou ao estado active. *Note*-se que o campo *Fix*ed in do CR tem de ser alterado pelo *coordinator* depois de todos serem submetidos na *build.*	Não
Prepared	active	Sem CO	*Coordinator*	Se apenas um componente foi afectado, não é necessário criar um CO. A *build* com as alterações tem de ser inserida no campo *fix*ed in do *MR-Tool*.	Não
Active	Implemented	Sem CO	*Coordinator*	A correcção é incluída numa *build* especial, que foi inserida no campo *fix*ed in.	Não
Active	Implemented	Com COs	*Coordinator*	Todos os COs chegaram ao estado implemented. O campo *fix*ed in *build* tem de ser actualizado.	Não
Implemented	Test	A *Build* que é mencionado no campo *fix*ed in onde está disponível.	Admin	A *Build* com a correcção está disponível para ser testada.	Não
CO: sent	CO: Active		*coordinator*	A mesma *build* com a correcção foi acordada entre todos os ROCs. O	Não

				número é inserido na *release build* pelo *coordinator*. Tem de ser submetida pelo responsável pelo CO e inserida no campo *fix*ed in pelo responsável pelo CO.	
CO: active	CO: implemented		ROC	O número da *Build* tem de ser inserida no campo *fix*ed in. A correcção tem de ser entregue nesta *build*.	Não
CO: implemented	CO: test		Admin	A *build* com a correcção está disponível para ser testada.	Não
CO: test	CO:closed	CO passa a Closed mesmo que o teste falhe	QA ou Admin	Se o CO pode ser testado sozinho é fechado pelo QA. Se o CR só pode ser testado como um todos, o CO será fechado pelo Admin depois do CR ser fechado pelo autor do CR. Se os testes falharem um TR tem de ser aberto.	Sim

4.6 Problema

Com o trabalho realizado pretende-se dotar a ferramenta já existente (*Eagle*) com novas métricas e rácios bem como, implementar restrições temporais a nível de interface de forma a permitir os intervenientes escolherem intervalos de tempo para a análise de informação.

Por outro lado, fazer o acompanhamento dos vários processos desempenhados as tarefas mais comuns de *Error Coordinator*, analisando e filtrando os dados fornecidos pelo *MR-Tool*, e a criação de relatórios para as equipas de gestão de projectos.

5 Concretização do Plano para Gestão do Processamento de Erros

Esta secção descreve o trabalho realizado de forma a resolver o problema descrito no capítulo anterior.

Durante as primeiras semanas foram discutidos conceitos e objectivos a atingir, tanto a nível pessoal como no âmbito do projecto. Estando já o projecto numa fase bastante evoluída foi necessário analisar todo o material herdado, incluindo documentação e código desenvolvido, bem como adquirir conhecimentos nas plataformas e frameworks a serem utilizadas. Por fim, iniciou-se um ciclo de análise, implementação e testes com vista a permitir que as funcionalidades implementadas estivessem isentas de *bugs*.

5.1 Ferramentas e tecnologias utilizadas

Para além do *MR-Tool* e do *Eagle* foi necessário recorrer a uma vasta lista de tecnologias e ferramentas open *source*, sem as quais não seria possível desenvolver o projecto de forma eficiente.

5.1.1 Java

A tecnologia Java é simultaneamente uma linguagem de programação e uma plataforma.

Java é uma linguagem de programação de alto nível que permite o desenvolvimento de aplicações seguras de alto desempenho. Por isso e pelo facto de as aplicações desenvolvidas serem altamente robustas, foi escolhida a tecnologia Java para o projecto.

Esta simples linguagem de arquitectura neutral possibilita ainda *multithreading* e distribuição de componentes. Ainda pelo facto de ser orientada a objectos, é possível tirar partido das metodologias modernas de *software development* que melhor se adaptam em aplicações cliente-servidor.

O ambiente de *software* ou *hardware* em que um programa corre é designado por plataforma. As mais conhecidas são *Microsoft Windows*, Linux, Solaris OS e MacOS. A maior parte das plataformas são a combinação de um sistema operativo e o *hardware* subjacente. A plataforma Java difere das demais pelo facto de ser uma plataforma *software-only* que corre por cima de outras plataformas de *hardware*.

A Plataforma Java tem dois componentes:

- A *Java Virtual Machine* (JVM);
- A Java *Application Programming Interface* (API).

A API é uma colecção de componentes predefinidos que fornecem um variado leque de capacidades, tal como *graphical user interface* (GUI) *widgets*. Tudo isto está agrupado em bibliotecas de classes e interfaces relacionadas – *packages*.
A figura seguinte ilustra como a API e a JVM suportam o programa a partir do *hardware*.

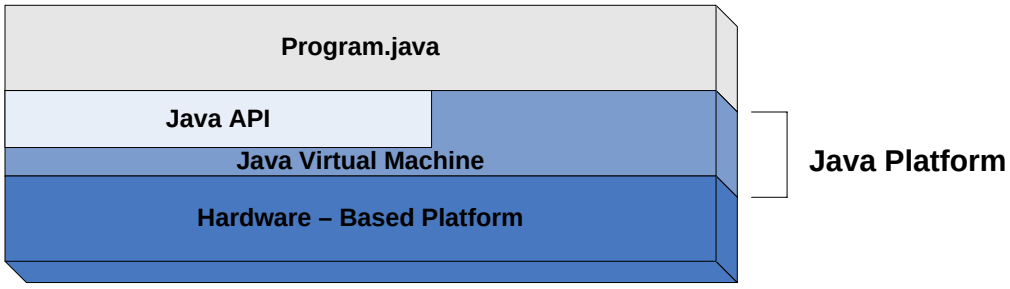


Figura 5-1 - Java Platform

Por ser um ambiente independente da plataforma, a plataforma Java pode ser algo lenta; no entanto os avanços no compilador e na JVM têm vindo a melhorar o desempenho geral.

5.1.2 Java EE 5 Platform

A plataforma Java escolhida para o projecto foi a versão 5 da *Enterprise Edition* – Java EE, antigamente referida como J2EE. As vantagens associadas à utilização desta plataforma no desenvolvimento de aplicações empresariais em relação às versões anteriores e outros tipos de plataforma são o aumento de desempenho e o tempo reduzido de desenvolvimento.
A plataforma Java EE 5 introduz a nova *Java Persistence* API que pode ser utilizada directamente por aplicações web e aplicações cliente. A introdução de EJB 3.0 simplifica a programação com *Enterprise JavaBeans* (EJB) através do uso de *Plain Old Java Objects* (POJO's). A Java EE 5 suporta tecnologias de ponta como Java*Server* Faces (JSF), JSTL ou AJAX, cruciais para o desenvolvimento de aplicações web 2.0

5.1.2.1 Java*Server* pages

Java*Server Pages* (JSP) é uma tecnologia Java que permite gerar dinamicamente HTML, XML ou outro tipo de documentos em resposta a pedidos de uma web client.
A sintaxe JSP adiciona XML tags, chamadas acções JSP, utilizadas para invocar funcionalidades internas. Esta tecnologia possibilita a criação de JSP *tags*, que funcionam como extensões às normas de HTML ou XML. As bibliotecas de tags proporcionam uma forma independente da plataforma de estender as capacidades de um servidor web.
OS JSP's são compilados em *Java Servlets* e devem estar publicados num servidor web específico, tal como o Tomcat.

5.1.2.2 Java*Server* Faces

Java*Server Faces* (JSF) é uma web *application framework* baseada em Java, que simplifica o desenvolvimento de user interfaces para aplicações Java EE. Os JSF's utilizam Java*Server Pages* para a sua tecnologia de *display*, podendo também acomodar outras tecnologias, tal como XML. A tecnologia JSF inclui ainda um conjunto de API's para controlar componentes de *user interface* (UI), um conjunto de componentes UI e duas bibliotecas JSP para incluir interfaces JSF dentro de uma página JSP.

5.1.3 Apache Ant

Apache *Ant* é uma ferramenta utilizada para automatizar a construção de *software* similar ao *make*, mas escrita em Java e desenvolvida inicialmente para ser utilizada em projectos desta linguagem.
A diferença mais evidente entre as ferramentas *Ant* e *make*, é que a primeira utiliza um arquivo no formato XML para descrever o processo de construção (*build*) e as suas dependências, enquanto o *make* possui o seu próprio formato de arquivo, o *Makefile*. Por omissão, este arquivo XML tem o nome *build*.xml.

5.1.4 Apache Tomcat

O Apache Tomcat é um web container *open source*, que implementa as especificações de *servlets* e JSP, providenciando um ambiente em que o código Java corre em conjunto com um servidor web. Inclui ferramentas de configuração e gestão, podendo ser configurado usando *tags* de XML. Pelo facto de incluir internamente o seu próprio servidor HTTP, pode também ser considerado um servidor web autónomo. O Tomcat costuma correr conjuntamente com outros servidores.
Os *developers* desenvolveram o Tomcat em Java, pelo que corre em qualquer sistema operativo que tenha uma JVM.

5.1.5 Unified Modeling Language

A *Unified Modeling Language* (UML) é uma linguagem de modelação não proprietária de terceira geração. A UML não é um método de desenvolvimento, o que significa que ela não diz o que fazer primeiro e de seguida ou como projectar o sistema; simplesmente ajuda a visualizar o design e a comunicação entre objectos.
Basicamente, a UML permite que os *developers* visualizem os produtos do seu trabalho em diagramas normalizados. Juntamente com uma notação gráfica, a UML também especifica significados, isto é, semântica. É uma notação independente dos processos, embora o *Rational Unified Process* (RUP) tenha sido especificamente desenvolvido com UML.
É importante distinguir entre um modelo UML e um diagrama (ou conjunto de diagramas) de UML – este último é uma representação gráfica da informação do primeiro, mas o primeiro pode existir independentemente.

5.1.6 Eclipse

O Eclipse é uma framework open *source* utilizada para desenvolver *Integrated Development Environments* (IDE's), tal como o Java IDE chamado *Java Development Toolkit* e o compilador que faz parte do Eclipse, que também é usado para desenvolver o próprio Eclipse.

5.2 Realização de testes piloto

A ferramenta *Eagle* V1.0 já permitia fazer relatórios estatísticos sobre as várias equipas tendo alguns gráficos disponíveis. Nesses gráficos já era possível fazer uma triagem dos MRs tendo em conta a prioridade ou estado dos mesmos.

Dotar a ferramenta de novos gráficos e métricas exigiu analisar os já existentes, procurar necessidades nos PL e TM, ver o grau de funcionalidade e utilidade dos mesmos, sendo para isso necessário fundamentar os resultados através de testes piloto (em várias iterações) com vista a obter um retorno positivo dos clientes da ferramenta.

Algumas das métricas e rácios pretendiam saber a evolução dos MRs ao longo do tempo.

5.2.1 Métricas

Foi decidido que as métricas seriam separadas por projectos e estes por sua vez estariam apresentados por ordem de prioridade crescente (General – valor de todas as prioridades, prioridade 1, prioridade 2,...).

Foram definidas as seguintes métricas:

- Numero Total de TRs de um projecto;
- Reabertura (um TR já fechado, mas que devido a novos testes foi considerado "Not *Fix*ed");
- Dev Time (Tempo de desenvolvimento Total – Tempo de teste)
- Tempo de teste (Tempo desde que um TR entra em estado de teste e depois este estado é alterado).

O principal objectivo era o de se poderem tirar ilações sobre onde é que as várias equipas de desenvolvimento perdem mais tempo. Estas métricas vêm auxiliar o EC nas suas tarefas de controlo uma vez que o facto de um MR ter um valor consideravelmente elevado em tempo de desenvolvimento poderá significar o facto de estar esquecido neste processo de desenvolvimento, quando já poderia ter passado para teste.

Por outro lado o facto de se calcular a média de todas as versões do mesmo projecto permite obter um valor médio e fazer uma avaliação, posicionando a versão de um dado projecto a acima ou abaixo desse valor.

A Figura 5-2 ilustra o resultado final destas métricas após a implementação na ferramenta de TI.

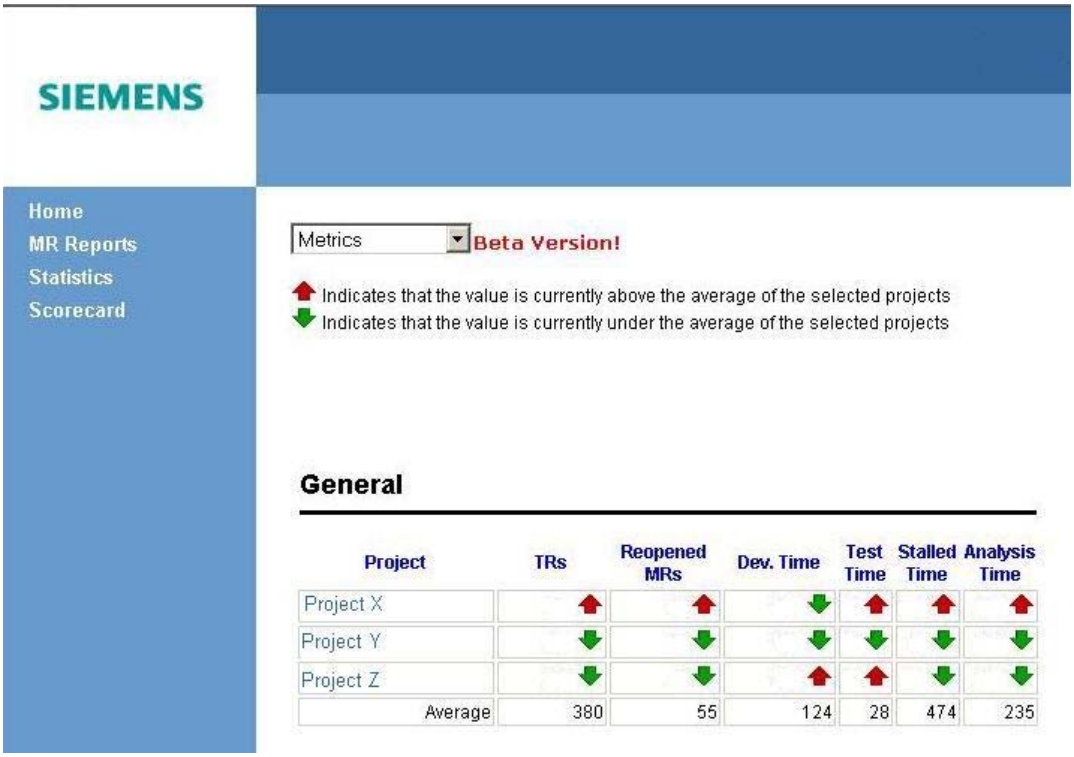


Figura 5-2 - Métricas no Eagle

5.2.2 Rácios

Apesar de ser uma informação bastante útil, às métricas faltava a possibilidade de comparar os projectos realmente e não apenas comparar versões. As métricas só comparavam as versões do mesmo projecto.

Os rácios definidos também apresentam o mesmo critério de apresentação de forma a manter uma visão padronizada na ferramenta. Foram definidos os seguintes rácios:

- Dev/ Total Time (Tempo de desenvolvimento / TempoTotal);
- Reopened / Total MRs

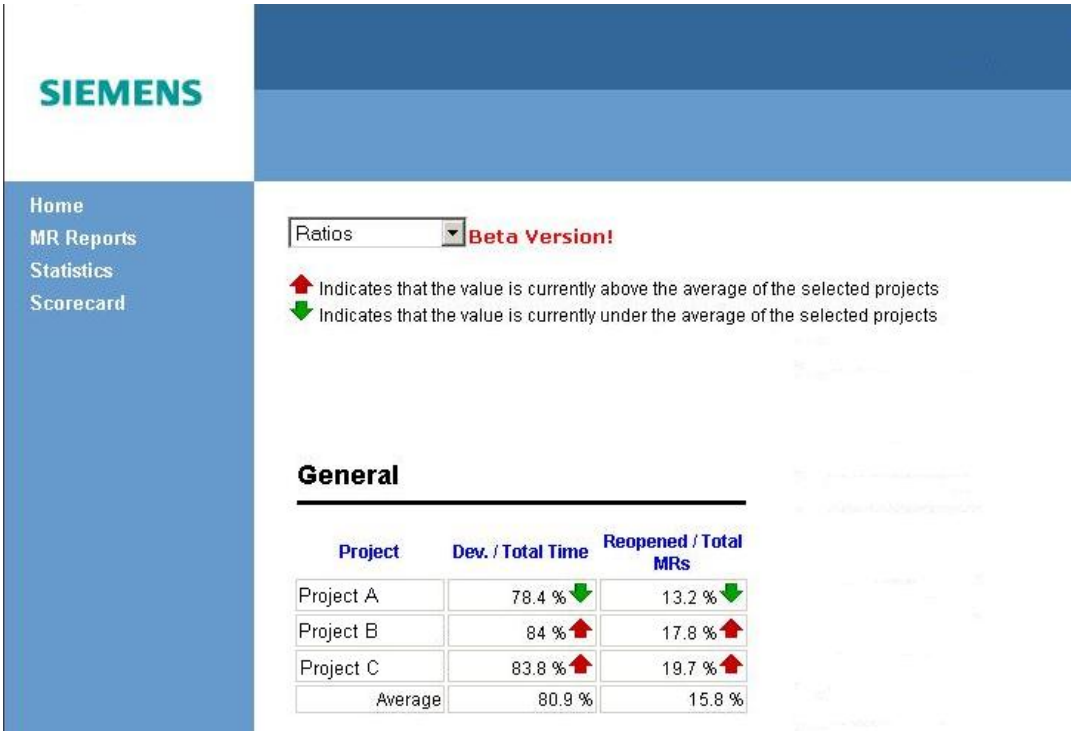


Figura 5-3 - Rácios no *Eagle*

A ideia é similar ao utilizado pelas métricas mas de forma a permitir comparar diferentes projectos.

O primeiro rácio permite perceber qual a percentagem de tempo gasto no desenvolvimento por um dado MR.

O segundo rácio torna mensurável a quantidade de MRs reabertos para correcção ou teste. Este é um problema que permite medir o grau de eficácia efectiva de uma equipa ou projecto.

5.3 Implementação das interfaces temporais

Um dos objectivos propostos dentro do projecto de GPE (Gestão do Processamento de Erros) foi o de fazer actualização das interfaces da ferramenta *Eagle*, à medida que novas funcionalidades iam sendo incluídas.

Esta fase de implementação precede a fase de desenho que acaba por ser omitida neste relatório.

Durante a implementação foi necessário determinar as classes necessárias para o desenvolvimento das novas funcionalidades bem como perceber a metodologia do jBoss.

A primeira preocupação foi a de criar uma hierarquia na VOB (*workspace* do rational clear case) para acomodar os diferentes tipos de ficheiros. Como esta aplicação é uma aplicação web esta hierarquia segue a estratégia da Sun para aplicações web. A estrutura do directório de desenvolvimento foi o seguinte:

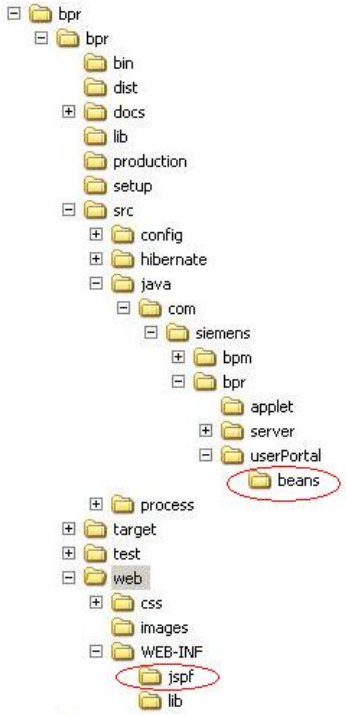


Figura 5-4 - Directório das classes implementadas

- Directórios que contêm o *Source code* java para o projecto.
- Directórios com os ficheiros que são copiados durante a compilação.
- E os jsp utilizados para a aplicação web.

A implementação de interfaces temporais permite escolher intervalos de tempo para que se determinem períodos de visualização da informação, por outras palavras permite ao utilizador escolher uma data de início e outra de fim para ver os dados nos gráficos.

Para este objectivo foram criados novos atributos e criados métodos em classes.

Por outro lado foi também necessário alterar o JSP responsável por fazer o render da interface na *webpage*. Como tal tiveram de ser chamados métodos em *javaScript* e passados os respectivos valores para os ficheiros JSP.

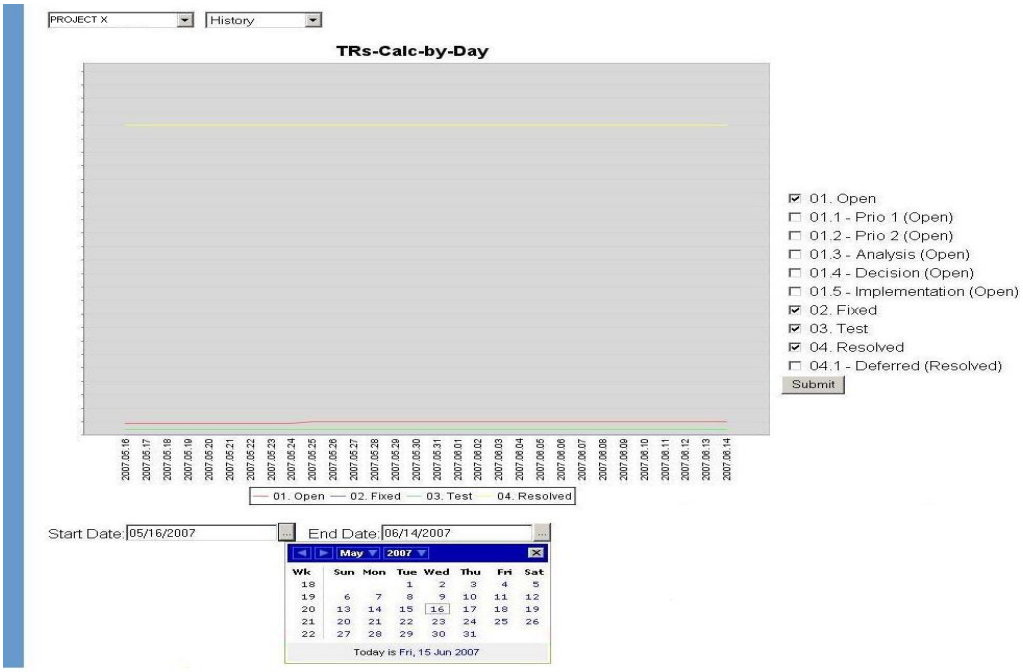


Figura 5-5 - Interface temporal implementada

5.4 Coordenação de Erros para o HES

Por outro lado foi solicitado à equipa de EC que realizasse testes piloto para o projecto Home Entertainment Solution (HES - Himalaya). Este projecto utiliza uma ferramenta de controlo de erros diferente do *MR-Tool*, o *Source Forge* uma ferramenta colaborativa open *source*. Embora a metodologia utilizada seja a mesma que no *MR-Tool*, existem algumas diferenças que a tornam mais poderosa (no caso das interfaces e usabilidade) mas por vezes também mais limitada.

Para este efeito dividimos o relatório para o HES da seguinte forma:

- Por *Target Release*s;
- Por Prioridades;
- Por Intervalos de tempo;
- Por equipas.

Os TRs são agrupados em *Open* e *Closed*, dentro do *Closed* ainda os subdividimos em *Resolved*. Os *Open* correspondem aos seguintes estados:

- *Open*
- *Pending*
- *More Information Req'd*
- *Postponed*
- *Work in Progress*

Closed:

- *Closed*
- *Closed – As Designed*
- *Closed – Do not Fix*
- *Closed – Duplicate*
- *Closed – External*
- *Closed – Not Reproducible*
- *Closed – Unable To Resolve*
- *Closed – User Error*
- *Closed – Verified*

Resolved:

Resolved – As Designed
Resolved – Duplicated
Resolved – External
Resolved – Fixed
Resolved – Not Reproducible
Resolved – User Error

A maior parte dos gráficos tem como base comparar o número de MRs abertos VS MRs fechados. A partir daqui poderemos analisar a desempenho de uma dada equipa, sabendo a priori que a prioridade de um dado MR pode influenciar o tempo de correcção.

5.4.1 Target Releases

As target *Release*s representam a altura em que todas as correcções ou alterações deverão ser submetidas, representando assim uma meta a alcançar para as equipas de desenvolvimento.

Pelo facto de uma equipa ou TR não entrar nenhuma Target *Release* programada, ajuda-nos a concluir que esse objectivo não foi atingido pela a equipa ou *developer* e teremos de nos servir de mais informações para perceber porque é que esta situação ocorreu. Uma causa plausível pode ser o facto de haver excesso de tarefas alocadas a um recurso.

5.4.1.1 Prioridades

Os TRs apresentam diferentes prioridades. A eficiência e eficácia de uma equipa podem ser medidos (com ajuda de outras rácios e métricas) através do número de TRs resolvidos de acordo com as prioridades. A questão das prioridades remete a questão do grau de dificuldade de resolução de um erro, tempos estimados para resolução mas também o impacto do mesmo no cliente. TRs de prioridade 1, deverão estar no topo da lista de resolução das equipas.

5.4.1.2 Intervalos de tempo

O intervalo de tempo considerado foi definido para relatórios semanais, e mensais.
Por outro lado o espaçamento de dois dias está inteiramente relacionado com o facto de se conseguirem obter diferenças mais significativas.

5.4.1.3 Equipas

Medir o nível de eficácia e eficiência das equipas permite prever os valores reais do projecto, tomar medidas correctivas atempadamente e redistribuir a alocação de recursos se for caso disso.

Os valores mais significativos são apresentados em tabelas que permitem ver as alterações, tendo em conta o *workload* das equipas e a distribuição dos MRs pelas mesmas.

5.4.1.4 Situações de Alarme/Atenção

Um outro tópico existente nos relatórios são as situações de Alarme/Atenção em que são deixadas notas aos PL e TM de forma a se evidenciarem as situações que requerem uma atenção prioritária podendo comprometer o desempenho da equipa ou equipas, bem como o projecto.

Após as várias iterações nestes testes piloto seguiu-se a fase de implementação[2] das funcionalidades no *Eagle*.

[2] A implementação das funcionalidades diz respeito apenas aos projectos controlados pelo MR-Tool, como o projecto HES utiliza outra ferramenta (*SourceForge*), esta fase não existiu para este projecto. No entanto, existe neste momento a intenção de se realizar uma aplicação em Java para este efeito.

5.4.2 Exemplo dos gráficos resultantes para o HES

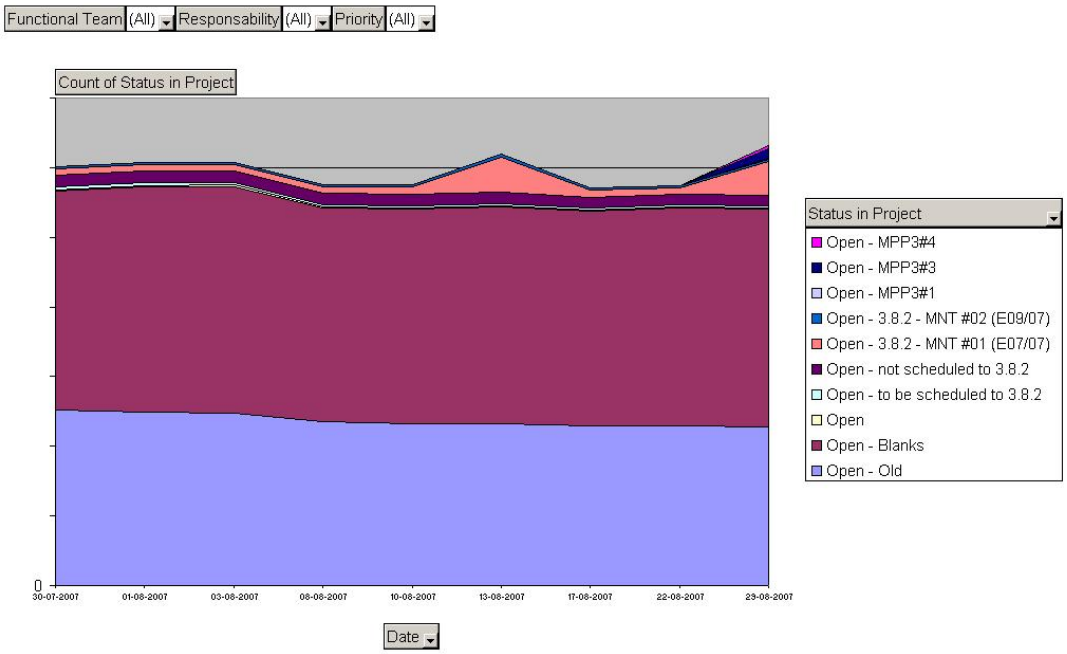


Figura 5-6 - TRs Open no HES

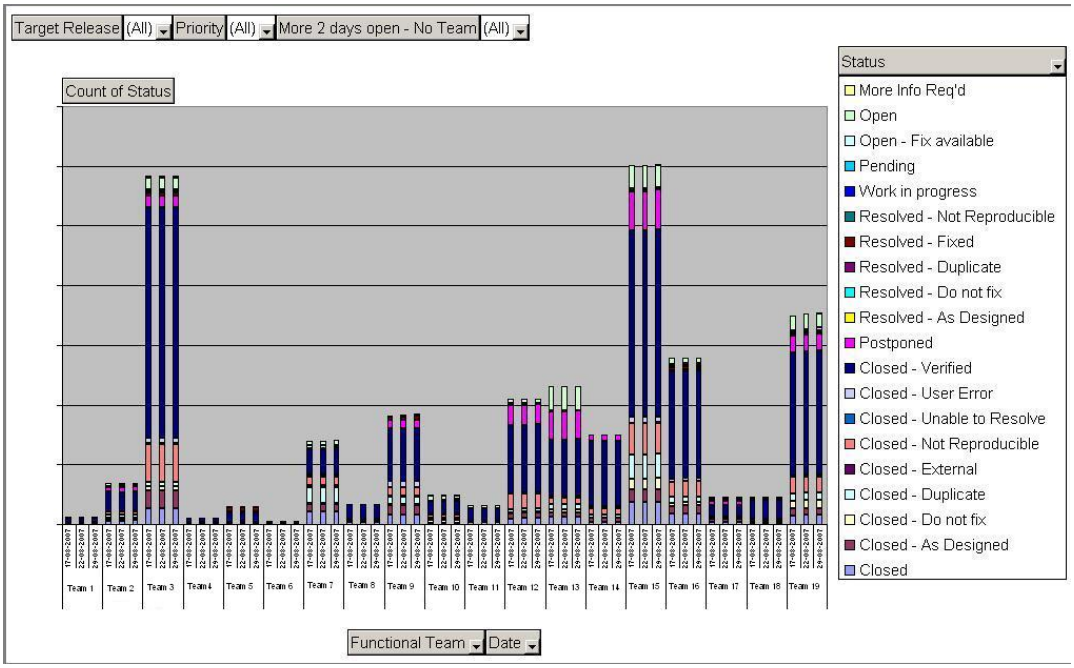


Figura 5-7 - TRs por equipas no HES

6 HES – Gestão e Configuração de Software

Todas as tarefas realizadas no âmbito do perfil de *Software Configuration Manager* (SCM), tiverem como palco o projecto *Home Entertainment Solution* (HES).

Na altura em que se iniciou o estágio neste projecto o mesmo já se encontrava separado em três produtos distintos:

- HES 2.2 - também denominado de Himalaya;
- KPN - Produto já em fase de manutenção presente no cliente;
- HES 3.0.plus – Novo Protótipo em fase de análise e levantamento de requisitos.

De uma forma geral os três produtos deste projecto tem particularidades únicas mas ao mesmo tempo funcionalidades homogéneas pois uns e outros representam a evolução quer de arquitecturas quer de novas funcionalidades, bem como melhorias ao longo do tempo graças à perícia e competência dos vários engenheiros envolvidos neste projecto.

Será necessário introduzir e contextualizar a equipa de SCM dentro do HES, bem como as suas tarefas que por vezes não são reconhecidas de extrema importância para o projecto.

6.1 Home Entertainment Solution -HES

O *Home Entertainment* é a solução que a Nokia Siemens Networks apresenta para televisão digital interactiva (IDTV), que oferece aplicações multimédia, como *time shift tv*, vídeos "*On Demand*", jogos *online*, telefone, e karaoke tudo num só produto (ver 6.1.1).

Esta solução é baseada no *middleware* Myrio IPTV, completada com os melhores componentes para encriptação/protecção de conteúdos, servidor de conteúdos, TV *head-end* e *set-top-box*.

A pré-integração destes componentes numa solução *end-to-end* garante uma boa interoperabilidade e reduzidos custos de distribuição e dá liberdade ao utilizador de escolher os componentes que melhor satisfaçam as suas necessidades.

6.1.1 Serviços fornecidos

Os serviços fornecidos por esta funcionalidade podem ser usados por qualquer utilizador que tenha um televisor PAL/NTSC ou HDTV ficando assim com uma ligação à internet, transformando o televisor num verdadeiro centro de entretenimento.

6.1.1.1 Serviços Difusão

Difusão da TV Digital (DTV) – permite ao utilizador ter acesso a canais digitais transmitidos por difusão pela rede em tempo real. A configuração dos pacotes dos canais é totalmente flexível, permitindo ao operador oferecer qualquer número e combinação dos pacotes de canais, básico e Premium, ou outro tipo de pacotes que podem ser definidos pelo operador. Um guia de programas iterativo (IPG) permite ao utilizador navegar e ver a lista de canais mais facilmente.

Musica/Rádio digital difusão **(DMX) -** Permite aos utilizadores, receberem estações de rádio em tempo real.

Pay Per View (PPV) - Funciona como um canal difusão de TV, mas neste caso o utilizador tem que alugar o canal ou o programa. Para este processo o utilizador tem que usar um PIN para ter acesso ao conteúdo PPV. Este tipo de canais também é mostrado no IPG.

6.1.1.2 Serviços On Demand

Video *On Demand* (VoD) e Audio *On Demand* (AoD) – Permite aos utilizador ter acesso a conteúdos de vídeo ou musica (ex. blockbuster), que queiram ou não alugar. O utilizador pode usar esta funcionalidade do tipo DVD vídeo clube permitindo alugar um filme a partir de uma lista milhares de filmes fornecidos pelos mais famosos estúdios cinematográficos. Permite realizar acções *trick Play* (*pause, play, stop, fast forward, fast rewind*), pode parar e retomar o *playback*.
Os conteúdos *OnDemand* são alugados de forma semelhante aos PPV, podendo o serviço estar disponível 24 horas, como num vídeo clube convencional.

Para facilitar a navegação através dos conteúdos OnDemand, estes estão organizados por géneros (ou categorias) e podem ser pesquisados por título, artista, etc.

Sub*Script*ion VOD (SVoD) – VoD com subscrição. Permite ao operador fornecer conteúdos *OnDemand* ao utilizador, através de uma subscrição, não sendo desta forma necessário ao utilizador passar pelo processo de aluguer sempre que o conteúdo for acedido.

Free *On Demand* (FoD) – Permite ao operador oferecer conteúdos *OnDemand* livres de pagamento.

Karaoke *On Demand* (KoD) – Permite ao utilizador aceder a conteúdos karaoke. Funciona como o VoD tanto no processo de aluguer como na utilização dos comandos *trick Play*.

Play List – Permite ao utilizador organizar e sequenciar os seus conteúdos *OnDemand* (*clips* de áudio e karaokes) de curta duração previamente alugados pelo utilizador alugados ou grátis (*free*).

Time Shift TV (TSTV) – Permite ao utilizador fazer "*time shift*" do que esta a ver na TV. O TSTV, permite gravar imediatamente programas que estão a ser transmitidos em

directo, ou agendar a gravação de um programa no futuro ou visualizar qualquer canal, qualquer programa que tenha sido exigido na última semana.

Pause Live TV - Enquanto está a guardar um programa (difusão) ao vivo, o utilizador pode colocar na pausa o programa em questão e mais tarde voltar a fazer playback a partir da pausa, tendo ao seu dispor os comandos trick Play (passar para trás ou para a frente).

Jogos – O utilizador tem a possibilidade de jogar jogos através da STB, na sua televisão. Os jogos são descarregados temporariamente do operador, para a STB do utilizador. O utilizador pode jogar usando o comando (controlo remoto) da STB.

Internet na TV – O acesso à internet através da TV é feito através de um web browser integrado na solução. Recorrendo a um teclado opcional, o utilizador pode navegar na internet de forma semelhante a um PC. O Browser fornece as opções standard como o histórico e bookmarks. Também tem acesso a caixas de correio, chat e publicidade.

6.1.2 Arquitectura da Solução Myrio IPTV

A Solução Myrio IPTV é um sistema composto por duas componentes: Myrio Interactive e Myrio *Total manage.*

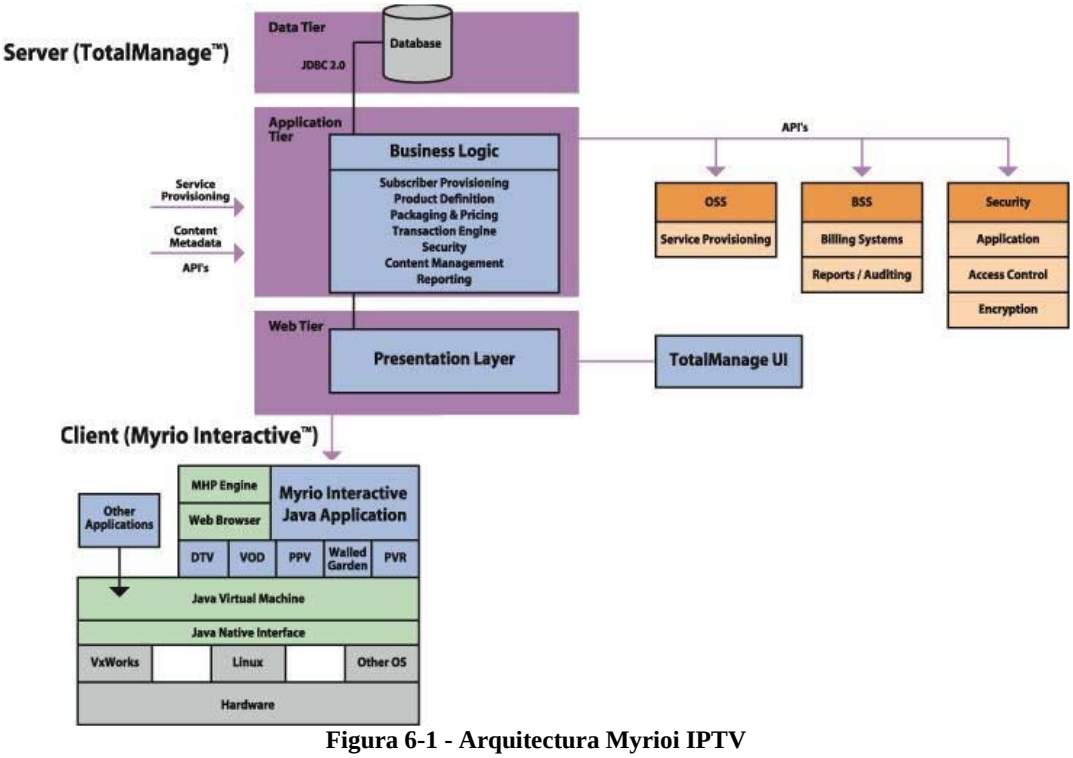


Figura 6-1 - Arquitectura Myrioi IPTV

O Myrio Interactive (Myrioi), corresponde ao *software* baseado na tecnologia JAVA, destinado às STB's (*set-top-box*s), que constitui a aplicação no cliente.

O utilizador só interage com o servidor de uma forma indirecta, em casos de, autenticação, autorização e aspectos relacionados com a contabilidade e configurações do serviço.

O Myrio *Total manage*, é um servidor de aplicações, responsável pela gestão dos módulos de serviços disponibilizados. Os módulos que constituem o *Total manage*, são responsáveis por gerir as configurações dos respectivos serviços.

Os módulos existentes são os seguintes:

- **Gestor de produtos:** Multicast de informação associada a cada canal de televisão.
- **Gestor de conteúdos:** Disponibiliza informação sobre conteúdos onDemand.
- **Gestor de pacotes de preços:** Configuração de pacotes de serviços e definições de preços.
- **Gestor do site:** Configuração dos parâmetros globais do sistema.
- **Gestor de Segurança:** Aspectos de segurança.
- **Gestor de transacções:** Controlo sobre as transacções.
- **Gestor de Subscritores (clientes):** Gestão da informação relativa aos utilizadores.

A solução *end–to–end* e os seus diversos componentes estão ilustrados de uma forma geral na Figura 6-2:

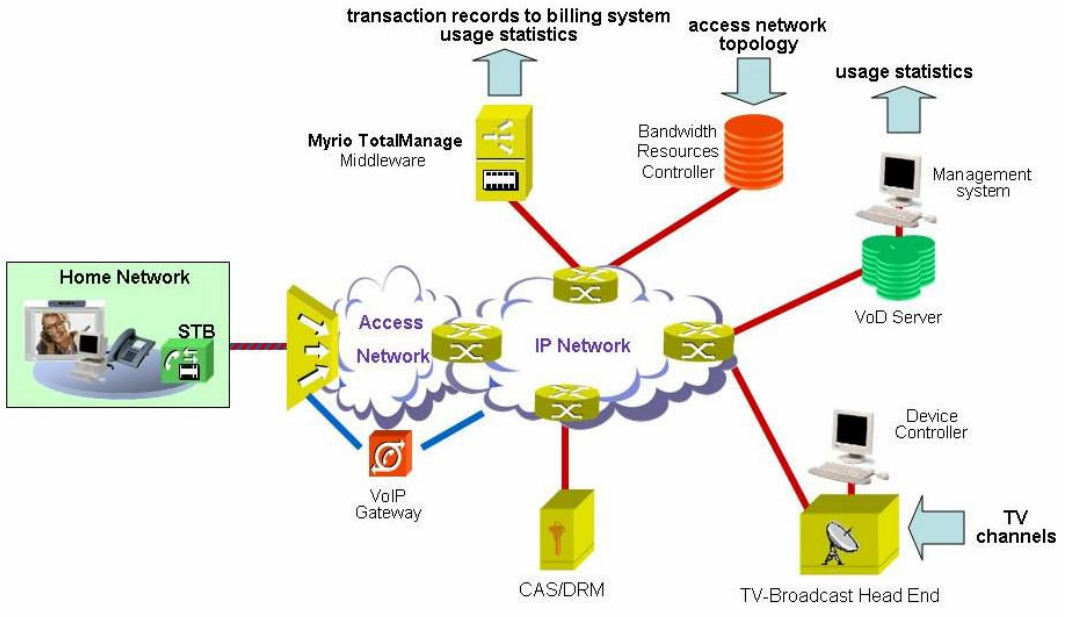


Figura 6-2 - Funcionamento do HES

6.1.3 Organização das equipas de desenvolvimento no HES

O desenvolvimento do projecto HES, está dividido em sete equipas. Para cada área funcional, existe o conceito de coordenador (*Feature Coordenator*), responsável por uma determinada funcionalidade no sistema.

A gestão do projecto Home Entertainment segue uma estrutura matricial, de modo a delegar autoridade aos diversos líderes de cada equipa para coordenarem recursos e tarefas.

O autor deste documento, integrou a equipa liderada neste momento pelo Dr. Pedro Borges, equipa que é responsável por gerir todos os *branches*, bem como gerir todo o processo de *builds* do projecto e ainda produzir toda a documentação relevante no que diz respeito a ferramentas utilizadas no processo de desenvolvimento.

6.1.4 A equipa de SCM

O desenvolvimento levado a cabo no projecto HES, atingiu uma dimensão de larga escala e com sistemas cada vez mais complexos, debaixo de restrições de tempo, requer que os *developers* trabalhem concorrentemente em vários projectos e em diferentes versões.

Por outro lado a diversidade de sistemas existente entre equipas necessita de ambientes de desenvolvimento específicos para as aplicações que desenvolvem mas ao mesmo tempo de processos coerentes de integração de forma a permitir que das partes se chegue ao todo e do todo se volte às partes.

A equipa de Software Configuration *Manager* deve ser capaz de:

- Gerir os processos de desenvolvimento;
- Configurar, identificar e controlar *software* de várias equipas, vários projectos em diferentes versões e diferentes estados de desenvolvimento;
- Realizar Auditorias e controlar código através de ferramentas automáticas;
- Realizar e divulgar as *release Notes* do *software* produzido.
- Responsável pela entrega SW, *builds*, *release*s e *release Note*s que são entregues a clientes reais, à equipa QA ou System Tests.
- Responsável pela criação dos cds de instalação das ferramentas em ambiente real de produção.
- Supervisionar as actividades e entregas de toda a equipa e *report*ar erros, atrasos nas entregas ou práticas erradas utilizadas.
- Responsável pela manutenção de todo o processo de *build* e pelo conjunto de servidores que são necessários para fazer uma *build*.
- Responsável pelas políticas de *branch* dos 3 projectos em Portugal com necessidades e políticas diferentes, bem como da manutenção de todo o código numa ferramenta de controlo de versões (Neste caso P4).
- Dar o suporte e formação à equipa sobre o uso, bem como as melhores práticas a realizar com o uso do P4.
- Responsável pela administração do *Source Forge*, bem como da elaboração de relatórios e gráficos sobre o desenvolvimento dos *bugs report*ados vs corrigidos, entre outro tipo de relatórios.

Na perspectiva da gestão o SCM controla a evolução e a integridade do produto identificando os seus elementos, gere e controla as alterações, verifica, grava e reporta

toda a informação relevante desde o início do processo de desenvolvimento, passando pelas configurações até à integração do código submetido para a produção de um cd de instalação.

Do ponto de vista do *developer*, o SCM facilita o desenvolvimento e as actividades de integração e alteração daquilo que se implementa.

A tabela seguinte reflecte as áreas de acção do SCM, dividindo esta em seis áreas fulcrais de acção:

Software Configuration Management		
Gestão do Processo de desenvolvimento	Configurar e identificar o software	Controlar
Contexto Organizacional do SCM Ciclo de vida do software, planos e calendários; Estratégias do projecto (concorrente, distribuida); Reutilização de software; Desenvolvimento e plataformas usadas; Ferramentas de desenvolvimento. **Restrições e condutas de SCM** **Planear para a gestão de software e configurações** Organização e responsabilidades; Recursos e calendarização; Selecção de ferramentas e implementação: Manual de SCM; Gestão do código e alterações; métricas e relatórios de integrações; Auditorias; Gestão da documentação; Criar Builds; Gerar release notes e distribui-las; Controlo de licenças; Controlo de Interfaces; **Plano de SCM**	**Identificação de itens para serem controlados** **Configuração de ambientes e ferramentas de controlo de versões** **Aquisição de software e Hardware**	**Pedir, avaliar e aprovar as alterações no software** Software configuration control board; Software Change Request Process. **Implementar as alterações de software** **Desvios e alarmes**
Gerir estados e erros dos processos	Auditorias e revisões	Gerir criar e distribuir Releases
Elaborar relatórios sobre os processos Envio de relatórios de erros aos feature coordinators	Reuniões de auditoria e inspecção Elaborar documentação e dar pareceres	Criar Builds e release notes Distribuição das mesmas

Tabela 6-1 – áreas de acção do SCM

6.1.4.1 Criação e Gestão de *branches*

Vamos poder ver mais a frente como os *branches* são utilizados e a sua grande importância nos projectos a nível de repositórios de código bem como a nível de segurança.

O HES usava já para os dois projectos inicias (Hes 2.2 e KPN) uma filosofia de *branches* que se traduz na utilização de dois *branches*, um para colocar o código como um repositório partilhado (*main branch*) e outro para identificar o código que será entregue na próxima *build* (*production branch*).

Existem dois conceitos que são muito importantes para que as integrações possam ter efeito em qualquer um dos dois *branches* descritos anteriormente:

- Ser **compilable**[3] (compilável) sem este requisito não poderá ser integrado no *main branch*;
- Ser **runnable**[4] (correr) sem este requisito e o anterior não poderá ser integrado no production *branch*.

6.1.4.2 *Branch*es – regras e politicas

Quando um projecto está em fase de arranque um main *branch* deverá ser criado para dar suporte à equipa de *Chief Engeneering* e posteriormente à equipa de desenvolvimento.

Um grupo de utilizadores para este *branch* é definido no *Perforce* (P4 group) de forma a mapear a equipa de desenvolvimento, com regras de acesso definidas usando o P4 protect.

As regras de segurança são geridas pelo SCM definindo as permissões de escrita e/ou leitura nos *branches*.

É possível criar um novo *branch*, se existe uma nova funcionalidade ou grupo de funcionalidades que precisem de estar isoladas do resto do código. Neste caso um novo *branch* deve ser criado para permitir a manutenção do código geral sem interferência do código das novas funcionalidades.

Todos os produtos estão mapeados num *branch* no *Perforce* e sempre que um novo produto deriva desse mesmo *branch*, um novo *branch* deve ser criado a partir do já existente tendo esta operação o nome de integração. Criar um novo *branch*, não só dá a possibilidade de manter um histórico do código desenvolvido, como a possibilidade de fazer a correcção (*bub fix*) de menor prioridade mais à frente no tempo.

As regras de segurança são geridas pelo SCM definindo as permissões de escrita e/ou leitura nos *branches*.

[3] Tornar computável tendo como requisito mínimo não ter erros.
[4] Se é compilável e computável então corre numa máquina de forma a chegar a uma última instrução que o termina.

É possível criar um novo *branch*, se existe uma nova funcionalidade ou grupo de funcionalidades que precisem de estar isoladas do resto do código. Neste caso um novo *branch* deve ser criado para permitir a manutenção do código geral sem interferência do código das novas funcionalidades.

Todos os produtos estão mapeados num *branch* no *Perforce* e sempre que um novo produto deriva desse mesmo *branch*, um novo *branch* deve ser criado a partir do já existente tendo esta operação o nome de integração. Criar um novo *branch*, não só dá a possibilidade de manter um histórico do código desenvolvido, como a possibilidade de fazer a correcção (*bug fix*) de menor prioridade mais à frente no tempo.

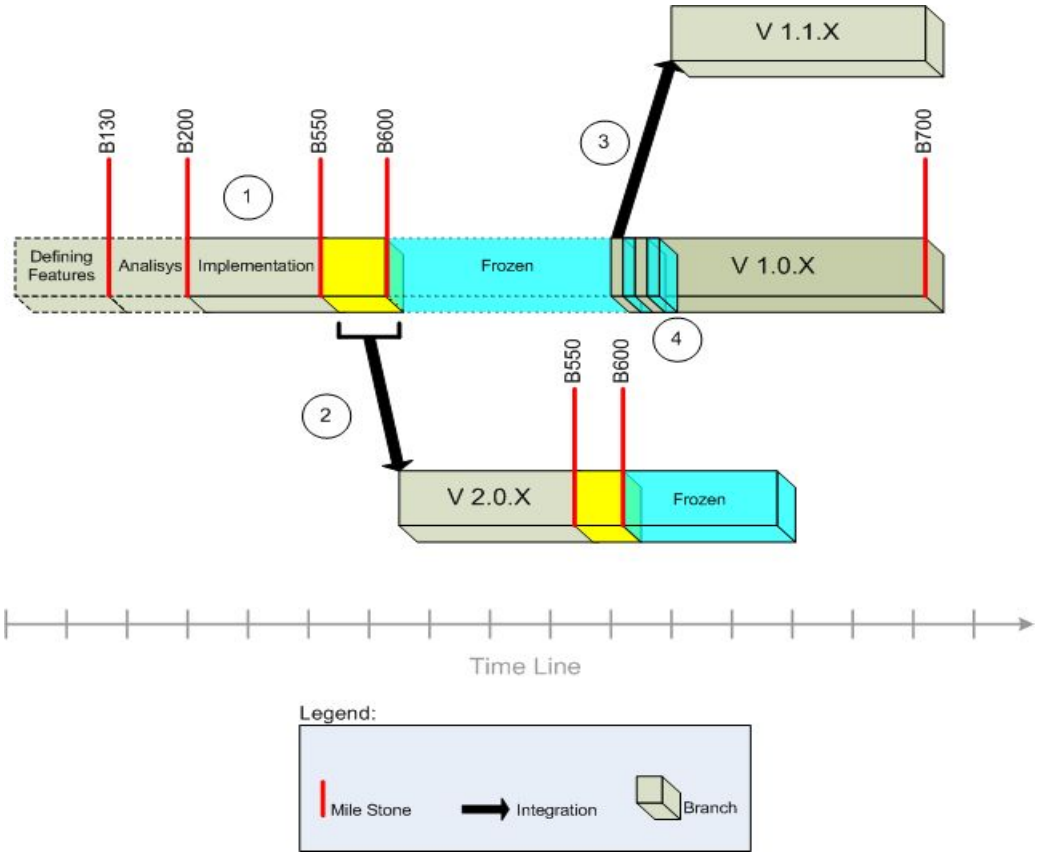


Figura 6-3 - ciclo de vida de um produto

Na Figura 6-3 é possível observar algumas fases e *Milestone*s (ver Product Provisioning Process Development (PPP: D) 4.0) do ciclo de vida de um projecto. Após o B200 o *main branch* já deverá estar criado. Durante a fase de análise todos os documentos e protótipos podem ser submetidos no P4.

1

- A equipa de SCM deve criar o *branch*

À medida que o tempo passa e o código vai sendo desenvolvido, o projecto alcança o B550. Nesta fase do projecto existe um conjunto de funcionalidades que estão prontas a ser testadas pela equipa de testes e o *branch* V 1.0.X torna-se estável com as correcções que serão feitas até o B600.

Após o B550 o seguinte cenário poderá ocorrer:

2

- Um novo cliente tem novos requisitos que devem ser especificamente implementados no projecto e um novo *branch* (V 2.0.X) é criado para o efeito. Como já foi referido anteriormente o resultado é um novo *branch*. Este novo *branch* assume as mesmas políticas de desenvolvimento que o anterior.

Após a declaração do B600 o *branch* é congelado (frozen), não se podem fazer mais alterações ou integrações a não ser que existam pedidos expressos da equipa de gestão ou SCM, como consequência de erros graves detectados no produto pelo cliente.

Assumindo que o novo cliente pretende um novo produto, com as últimas funcionalidades actualizadas:

3

- O novo cliente tem novos requisitos tendo como base V 1.0.X (pós B600).
- Um novo *branch* (V 1.1.X) é criado. O resultado é um novo produto. Este novo produto assume as mesmas políticas internas de desenvolvimento, iguais ao anterior.

Se a CCB decidir que um *bug* específico deverá ser corrigido, nessa altura, o *branch* é descongelado e a equipa de desenvolvimento ou alguns elementos dela podem trabalhar no *branch* V 1.0.X, de forma a corrigirem *bugs* de prioridade baixa ou implementarem outra funcionalidade necessária.

4

- O *branch* pode ser congelado ou descongelado de forma a permitir algumas últimas correcções.
- O *branch* V.1.0.X torna-se o mais estável e robusto produto.

Quando um projecto atinge o B700, a politica da Nokia Siemens Networks é a de fazer o acompanhamento e manutenção do produto com inputs vindos do cliente.

6.2 Problema

Criar um manual de SCM para o novo protótipo *HES 3.0.PLUS* tendo como base o melhoramento dos processos já existentes no HES 2.2 e KPN.

Com uma equipa de desenvolvimento muito grande, com muitos conflitos graves na submissão de código, principalmente nas partes comuns.

O acesso remoto ao servidor nos EUA é demasiado lento e tem tempos de resposta abaixo daquilo que é espectável para um projecto com esta dimensão.

O servidor de *builds* subcarregado com muitas *builds* de todos os países envolvidos no projecto (EUA, Bélgica, Croácia, República Checa, Índia, Áustria, Alemanha e Portugal).

7 Concretização do Plano para gestão e configuração de software

Inicialmente de forma a ser possível levar a cabo o plano estabelecido foi necessário adquirir conhecimentos com as ferramentas mais comummente usadas pela equipa de SCM. Todas a explicações que surgem adiante neste capítulo são fruto da investigação do aluno devido ao facto de ter de elaborado um manual de SCM que depois de revisto e aprovado por uma vasta equipa espalhada por todo o mundo Siemens foi posteriormente implementado no projecto.

Neste capítulo serão também dadas noções importantes para o desenvolvimento deste projecto que embora inteiramente relacionadas com o projecto HES são específicas das tarefas de SCM.

O projecto teve início com a instalação de *software* necessário, bem como a criação de uma *workspace* na máquina de trabalho de forma a dotar o estagiário de uma conta com privilégios de administrador no projecto (ver Ferramentas e tecnologias utilizadas).

Foi também necessário perceber toda a metodologia inerente ao projecto tendo feito o estagiário um reconhecimento da arquitectura da rede utilizada no projecto de forma a visualizar as máquinas envolvidas no projecto tendo em mente as tarefas a desempenhar.

7.1 Arquitectura de rede do HES

No diagrama da Figura 7-1 é possível perceber a rede utilizada na altura em que o aluno iniciou o projecto. Existiam dois servidores de *builds* um sendo o servidor oficial situado em Seattle e outro de testes em Lisboa. O servidor de *builds* em Seattle sendo o servidor oficial para o projecto aglomerava todos os projectos. O servidor de Lisboa sendo inicialmente um servidor de testes, cujo principal objectivo era o de realizar testes para a obtenção de métricas no que respeita a tempos de acesso, sincronização e submissão de código no servidor de P4 (*Perforce Server*), bem como determinar os tempos de realização de *builds* através de acesso remoto ao servidor de *builds* oficial.

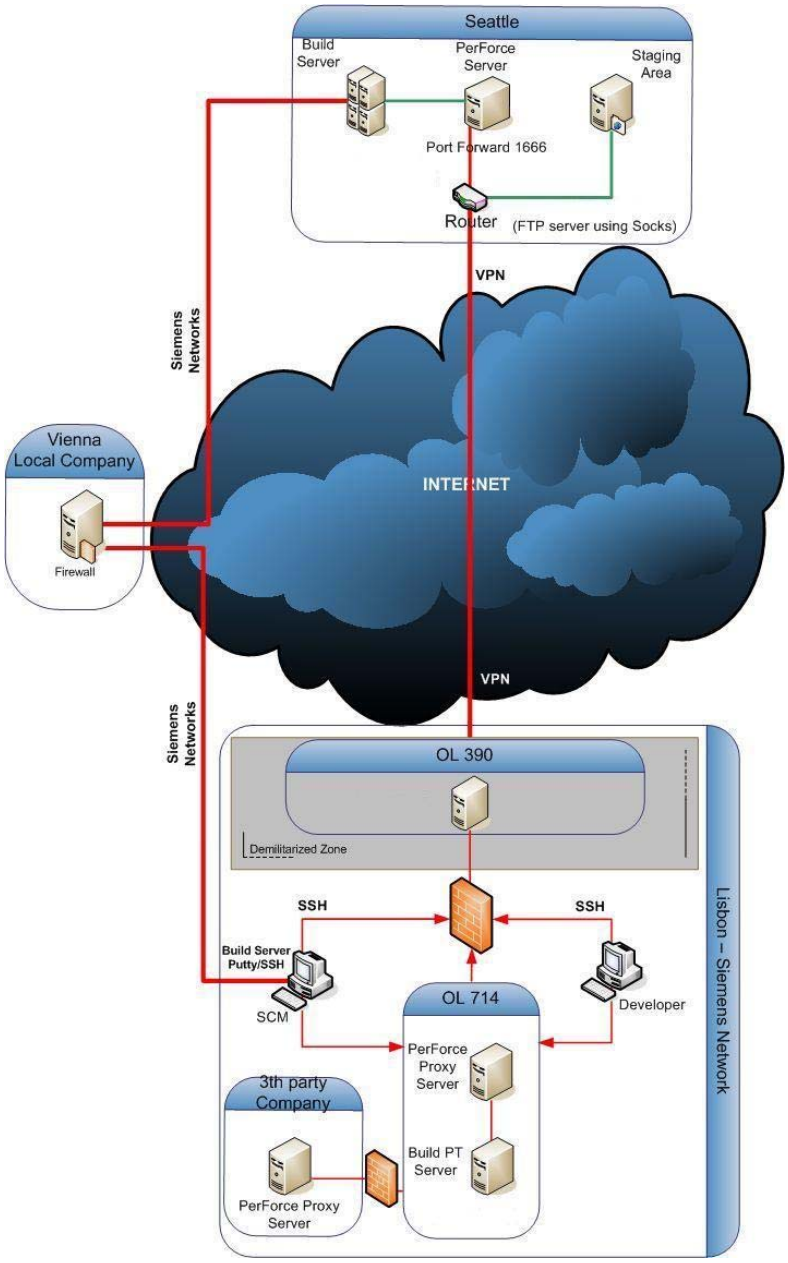


Figura 7-1 - Arquitectura de Rede do HES

7.1.1 Problemas identificados na rede

- A forma indirecta como o acesso é feito ao servidor de *builds* oficial está longe de ser perfeita uma vez que para se aceder ao mesmo, percorremos um conjunto de máquinas desde Portugal até os USA passando ainda por Viena de Áustria;

- A velocidade da rede em si apresenta uma limitação bem como a quantidade de tráfego na rede;

- Sendo Portugal o centro de desenvolvimento mundial, tendo as equipas mais numerosas e uma vez que produz a maior parte de código, faz todo o sentido ter

os recursos físicos mais próximos traduzindo-se isso num maior desempenho uma vez que a rede interna é mais estável, segura e rápida.

7.1.2 Resolução dos problemas da rede

A estrutura da rede foi sofrendo alterações à medida que se foram implementando as alterações previstas no manual de SCM com vista a ultrapassarem-se alguns problemas referidos no parágrafo anterior.

A equipa de SCM começou por instalar um *proxy server* em Lisboa deforma a diminuir o tempo de sincronização do código das *Workspaces* das várias equipas de desenvolvimento com o servidor de P4 em Seattle. Com esta medida os tempos de sincronização do código diminuíram para menos de metade, pois a informação pretendida encontra-se agora em *cache*, sendo assim mais rapidamente acedida.

Uma vez que a maior parte do desenvolvimento para o novo projecto HES 3.0.PLUS está em Lisboa. Decidiu-se instalar o servidor de *builds* oficial para este projecto em Portugal.
O mesmo está configurado numa máquina mais moderna, mais rápida, segura e fiável produzindo *builds* mais rapidamente e diminuindo os tempos de *downloads* e *uploads* das *builds* para a *Staging Area* ou repositório de *builds.*

A velocidade na rede passou de 40 kilobits/seg para 4 Mbits/seg o que o demonstra uma estrondosa melhoria. A rede interna é também menos utilizada pois conseguimos distribuir os recursos de forma mais eficiente em termos de tráfego e é mais rápida.

7.2 O processo de builds

O processo de *builds* constitui um dos processos mais importantes do processo de desenvolvimento, uma vez que o resultado de todo o trabalho feito pela equipa desenvolvimento só pode ser entregue à equipa de testes ou ao cliente após este processo.

As máquinas directamente implicadas neste processo encontram-se em Seattle nos U.S.A (apelidado de *build server*) e outra em Lisboa (servidor de teste). Ver Figura 7-1 - Arquitectura de Rede do HES.

Quando falamos de *builds* referimo-nos ao processo de efectuar imagens do sistema de ficheiros, incluído o sistema operativo para multiplataformas que são distintas de acordo com as STBs em causa, gerando assim *loads* diferentes para cada um dos respectivos *hardware*s.

7.2.1 O processo de *builds* antes do HES 3.0.PLUS

O processo de produção de *builds* é gerido por uma ferramenta de *builds* (*Script*s), que não é nada mais do que um programa em *Shell Script* que coordena e controla outros programas.

As *builds* são processadas por um *Script* denominado de autobuild.sh. O autobuild faz *builds* chamando algumas funções internas, algumas das quais responsáveis por criar e destruir directorias e chamar outros dois *Script*s de acordo com as seguintes regras:

- *Build*client.sh (no caso de ser detectado o client - Myrioi)
- Cm*build*.sh (caso se pretenda um TM (*Total manage*) *server* ou md*server* (metadata *server*)

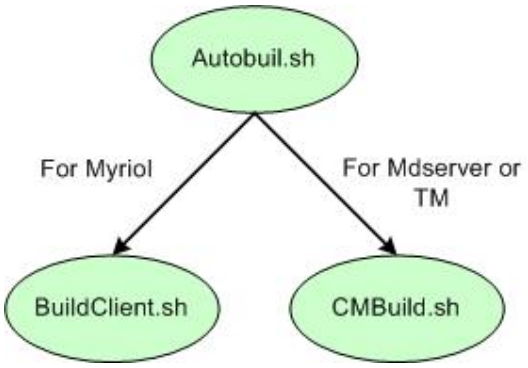


Figura 7-2 - Hierarquia de *Script*s

7.2.1.1 Builds manuais e builds automáticas

As *builds* inicialmente só poderiam ser produzidas de forma automática sendo escalonadas para as 00:00 GNT. Este processo percorre todos os projectos e produtos existentes no *build server* e caso verifique que existem CL (*Change Lists*) submetidas desde a última *build*, processa a *build* respectiva.

Para melhorar a desempenho das equipas surgiu a necessidade de se produzirem *builds* manuais após integrações, uma vez que de outro modo a verificação em termos oficiais das *builds* só poderiam sair no dia seguinte.

Todas as *builds* podem ser geradas a partir da chamada de *Script*s utilizados no processo de *builds* automáticas, bastando para isso chamar o buildqueue.sh como no caso da Figura 7-2 - Hierarquia de *Script*s.

Ao iniciarmos uma *build* manual com o buildqueue.sh basta escolher o *branch* e o produto respectivo através do número. Depois disto, o buildclient.sh ou o *cmbuild.sh* são chamados respectivamente de acordo com os *branches* e produtos que pretendemos. O *build* menu gerado pode ser visto na figura seguinte:

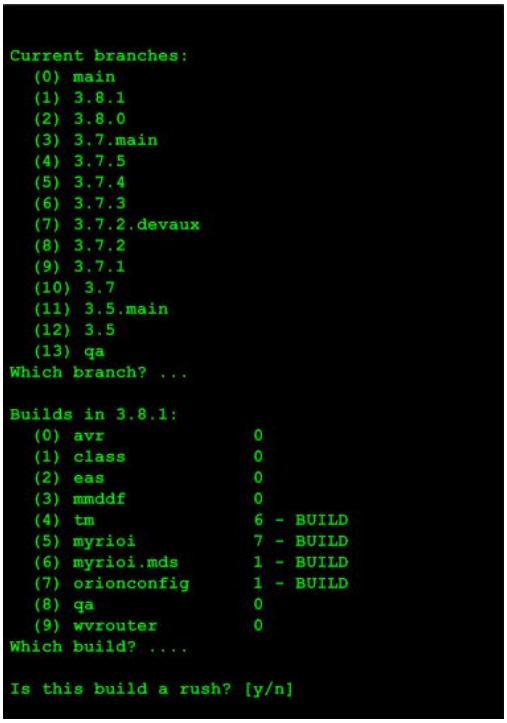


Figura 7-3 – *Build* manual menu

7.2.1.2 Compilação e marcação de Labels

É claro, que para se obter parte do processo é necessário compilar vários tipos de linguagens como Java, EFX. Após o processo de compilação pelo ANT é feita a marcação de *labels*.

O seu primeiro objectivo é o de marcar determinado código para produção que esta numa ou várias CLs para entrarem numa *build*, o segundo é o de garantir uma gravação numa dada altura no tempo dos estados dos ficheiros do servidor. Isto, porque a qualquer altura podemos querer restabelecer um dado estado e seguir a implementação a partir do mesmo.

7.2.1.3 Sincronização de Servidores

O autobuild.sh primeiro sincroniza o servidor de *builds* com o servidor de *Perforce* de forma a ter certeza de que obtém os ficheiros mais actualizados ou integrados para a produção.

- Sincroniza todos os outros *branches* com a última versão do código.

Este passo de sincronização é fundamental porque o processo de *builds* precisa de obter informação deveras importante do servidor de *Perforce*, como é o caso do *build* counter (o numero da última *build*), e ter a certeza de que todos os ficheiros estão checked in na submitted *change*list ou *change*lists que serão incluídas na *build*.

```
perforce_sync () {
  decho 'Syncing files...'
  SYNC_COUNT=`p4 sync ...@ProdLabel 2> $LOG_DIR/sync_error.log | wc -l`
  SYNC_ERRORS=`cat $LOG_DIR/sync_error.log | grep -v "... - file(s) up-to-date."`
  if [ "$SYNC_ERRORS" != "" ] ; then
    THIS_DIR=`pwd`
    mail_error "Error syncing files in $THIS_DIR. See sync_error.log for details."
    return 1
  else
    rm -f $LOG_DIR/sync_error.log
  fi
  decho "Files synced: $SYNC_COUNT"
  decho 'Syncing qa, tools and devexternal...'
  p4 sync //devexternal/... //tools/... //qa/... > /dev/null
  [ $? -gt 0 ] && mail_error "Error syncing qa, devexternal and tools!"
  return 0
}
```

Figura 7-4 - sincronização com o *Perforce*

7.2.1.4 Criação do cliente e IDK

Este passo representa o início da construção da *build* em si. É criado o cliente para as STBs que é comum a todas e uma camada de interface que difere de box para box a que damos o nome de IDK.

7.2.1.5 Criação de Plataformas para as STBs

Após a criação do cliente é criada a *load* para as STBs. Como existem vários representantes de STBs com os quais a NSN trabalha esta camada é distinta para cada STBs.

7.2.1.6 Publicação marcação de labels e actualização de contadores

Uma vez concluídos o cliente e o IDK bem como a *load* para a STBs específicas o processo de *builds* termina através de uma rotina que marca a *label* da *build* produzida, actualiza os contadores e pública a *build* na *Stagind Area* de Seattle. Num último passo para que a produção se torne oficial é enviado um e-mail para um conjunto de pessoas interessadas neste processo, que inclui a equipa de SCM, TL, PL e a equipa de testes.

7.2.2 O processo de *builds* após o HES 3.0.PLUS

Ao longo do estudo exaustivo do processo de *builds* e como o tempo de produção de *builds* era considerado excessivo a equipa de SCM implementou um conjunto de melhorias deforma a tornar o processo de *builds* mais rápido e eficiente para o projecto HES 3.0.PLUS.

Existia uma enorme dependência histórica em termos de *Script*s que levou ao aumento excessivo de *Script*s à medida que o tempo foi passando. Com o aparecimento de novos projectos, o processo tornou-se de tal forma ineficiente devido a estrutura de directórios criados e apagados durante o processo de *builds* para que fossem criadas as camadas de *software* muitas das vezes não utilizadas, mas como faziam parte do processo anterior eram desnecessariamente produzidas.

A alteração deste processo levou a redução de cerca de 80% dos ficheiros usados no processo de desenvolvimento, compilação e *Script*s. O resultado foi uma melhoria inigualável no tempo de produção de *builds* tendo uma *build* passado a ser produzida

não em 90 minutos mas agora em apenas 8 minutos no caso do cliente; e no caso do md*server* e TM de 30 minutos para 10 minutos.

Por outro foi também criado um novo repositório em Lisboa que guarda todas as *builds* do projecto HES 3.0.PLUS. Os tempos de *download* e *upload* das *builds* passaram de 30 minutos aproximadamente para apenas cerca de 10 segundos. Para que o repositório de *builds* em Seattle continua-se actualizado e sincronizado com o novo repositório em Lisboa, criámos um *Script* em *perl* que permite fazer uma actualização e sincronização destes dois repositórios através de um protocolo FTP.

7.2.2.1 Staging Area - Repositório de builds

A *Staging Area* é o repositório oficial para todo o projecto, esta máquina encontra-se localizada geograficamente em Seattle como se pode ver na Figura 7-1 - Arquitectura de Rede do HES. Os acessos à mesma são feitos por um utilizador comum via ftp, após a criação de um canal seguro de comunicação (vpn). Normalmente os *developers* usam o putty que é nada mais nada menus que um GUI que permite estabelecer ligações remotas.

A máquina usada para estabelecer o túnel é uma máquina *UNIX* de laboratório denominada com OL390 que está numa zona desmilitarizada. As ligações são possíveis por *port fowarding*.

Como administradores temos acesso indirecto através do *build server* utilizado acesso remoto através de comandos ssh.

Um *link* mapeia todos os ficheiros para todos os utilizadores autorizados, sendo apenas dadas permissões de leitura.

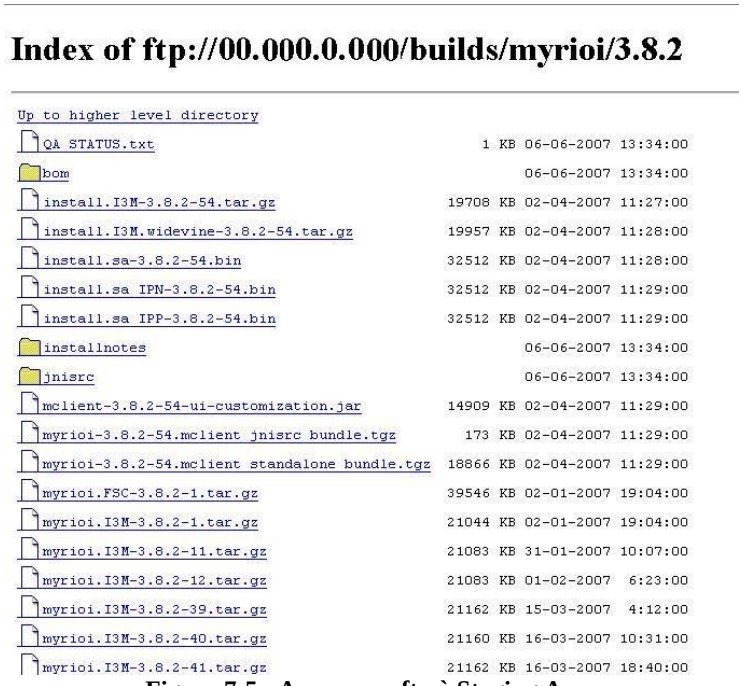


Figura 7-5 - Acesso por ftp à Staging Area

7.3 Ferramentas e tecnologias utilizadas

7.3.1 *Perforce* – controlador de versões

O *Perforce* é um *software* de SCM baseado numa arquitectura cliente servidor. Os utilizadores usam um programa cliente que se liga a um servidor de *Perforce* e partilham ficheiros entre si. O repositório do servidor tem o nome de *depot* e do cliente *workspace*. Pode haver mais que um *depot* por servidor.

Os *depot*s contêm todas a revisões/versões de cada ficheiro sobre controlo do *Perforce*.

Os ficheiros ficam organizados em árvores de directórios como num grande disco rígido. O servidor mantém todas as operações nos ficheiros numa base de dados, permissões dos utilizadores, e que utilizador tem que ficheiros *checkout* e em que altura.

Como qualquer ferramenta controladora de versões permite acessos concorrentes de forma a manter os dados consistentes quando alterações concorrentes são efectuadas no mesmo ficheiro (artefacto). Este processo permite que toda a equipa de desenvolvimento esteja como um todo a trabalhar, embora na realidade estejam em repositórios particulares ou de grupo tornado este processo transparente ao *developer* mas controlável pelo *Software Configuration Management*.

7.3.2 Source Forge

É um *software* de colaborativo para projectos de grande envergadura, permitindo a gestão, análise de estatísticas e troca de código, bem como fóruns. O sistema é administrado pela VA *Software*. Ele oferece uma interface para um diversificado serviço do ciclo de vida no desenvolvimento de *software*s e integra com um grande número de aplicações de código aberto.

No HES é utilizado como repositório de documentação e *bug tracker* para delegar *bugs* a *developers*, fazendo um histórico dos mesmos como a ferramenta *MR-Tool* em outros projectos da Nokia Siemens Networks.

7.3.3 Putty

O Putty é um cliente de SSH destinado a promover o acesso remoto a servidores via *Shell* Segura- SSH - e a construção de "túneis" criptográficos entre servidores.

7.3.4 Cygwin

Cygwin é um simulador de Linux em ambiente *Windows*. Consiste em duas partes:

- A DLL (cygwin1.dll) que actua como Linux API emulando as funções do Linux.
- Conjunto de ferramentas que se assemelham ao Linux.

8 Conclusões

O ambiente no mundo real de desenvolvimento de *software* é bastante diferente daquele encontrado no mundo académico. O ritmo é bastante mais acelerado e as pressões bastante maiores, mas também o trabalho realizado é mais significativo e enquadrado, sendo possível observar resultados reais do que é desenvolvido. A aprendizagem de novas ferramentas e o contacto com tecnologias ainda em desenvolvimento é muito gratificante, colocando o informático numa posição de destaque como criador da tecnologia de amanhã e torna o seu trabalho compensador e útil.

Embora tenha havido uma mudança no contexto inicial do estágio, uma vez que ao fim de cerca de dois meses houve alteração do plano do mesmo e do projecto, podemos afirmar que os objectivos propostos foram cumpridos no que respeita às tarefas de SCM, e em termos de objectivos gerais como o enquadramento do aluno no ambiente de trabalho.

Relativamente à ferramenta e às tarefas de Gestão do Processamento de Erros constituíram uma excelente aprendizagem dos métodos de trabalha da Nokia Siemens Networks lançando o aluno para uma base de conhecimento que ainda hoje é aplicada no projecto HES.

A ferramenta desenvolvida, o *Eagle* permitiu adquirir conhecimentos com a plataforma j2EE e o servidor aplicacional jboss que são ferramentas usuais no mundo de trabalho da informática, permitindo novas competências ao aluno e uma vantagem perante a concorrência existente no mercado.

O projecto HES é pioneiro nesta área a nível mundial apresentando algumas particularidades inigualáveis, pois é um projecto desenvolvido em grande escala, com pessoas diferentes, problemas diferentes e diferentes perspectivas de análise e resolução de problemas. No centro da resolução de muitos problemas está a equipa de SCM em que cada problema parece ser diferente do anterior, e à medida que ganhamos competências na área, vemos que todo o caminho percorrido embora por vezes com muito esforço é compensatório. “Trabalhar sob pressão e bem” tornando o que é difícil parecer fácil.

8.1 Trabalho futuro

Após a minha saída do projecto de gestão de processamento de erros, foram implementadas muitas mais funcionalidades que permitem a ferramenta estar hoje muito mais completa, uma visão já partilhada na altura em que o aluno se encontrava no projecto. No entanto, poderão ainda ocorrer mais melhorias relativamente a novos gráficos, métricas e rácios.

O HES 3.0.PLUS dá agora os primeiros passos como solução mais robusta no ramo do IPTV, muitas das sugestões apresentadas no manual de SCM foram seguidas, no entanto a migração dos servidores é ainda um problema chave a resolver em termos de desempenho.

De momento as políticas de integração estabelecidas estão numa fase embrionária para convergirem para as estabelecidas pelo manual. O manual é hoje uma referência para todos os *developers* que entram no projecto.
Relativamente às tarefas de *Scripts* já está prevista uma futura implementação de interfaces gráficas para os mesmos de forma a tornar estas ferramentas mais usáveis.

O processo de *builds*, encontra-se em fase de afinação mas já se preparam novas medidas de optimização para os projectos mais antigos.

Bibliografia

[1] Cygwin. http://www.cygwin.com/

[2] *Perforce*. http://www.*Perforce*.com/

[3] Myrio. http://www.myrio.com/

[4] Sun Microsystems. http://java.sun.com/

[5] KPN. http://www.kpn.com/

[6] Carnegie Mellon University. http://www.sei.cmu.edu/

[7] Nokia Siemens Networks.https://sharenetims.inside.nokiasiemensnetworks.com/

[8] Nokia Siemens Networks. http://nokiasiemens.pt/

[9] Nokia Siemens Networks. PEPP Process Handbook PPP:D for BU. 2006

[10] Andrew S. Tanenbaum. Computer Networks; 3rd edition. Prentice Hall, 3rd edition, 1996

[11] Nadine M. Bounds, Susan A. Dart. Configuration Management (CM) Plans: The Beginning to your CM solution. Software Engineering Institute, Carnegie Mellon University Pittsburgh, Pennsylvania, July 1993.

[12] Susan A. Dart. Concepts in Configuration Management Systems. Software Engineering Institute, Carnegie Mellon University Pittsburgh, Pennsylvania,1993.

[13] IEEE Guide to Software Configuration Management.1987. IEEE/ANSI Standard 1042-1987.

[14] Whitgift, D.. Software Configuration Management: Methods and Tools. John Wiley and Sons, England, To be published June 1991.

[15] Whitgift, D.. Software Configuration Management: Methods and Tools. John Wiley and Sons, England, To be published June 1991

[16] Rúben Mendes, Pedro Borges. HES - Software Configuration Management. Nokia Siemens Networks, 2007.

Anexos

Definição de Estados dos TRs

Titulo	State	
Assunto	Estado actual de um MR	
Valores	**New**	new TR, apenas visível para o autor do TR
	Sent	TR enviado para a base de dados, passará a para wait após o MR-Admin fazer a o check TR.
	Wait	MR ainda não decidido, são necessárias respostas
	Prepared	Todas as respostas foram respondidas no *MR-Tool* Mas ainda sem correcções o decisão final. Primeira análise de desenvolvimento a decorrer. O *Coordinator* tem de identificar os componentes adicionais que possam estar envolvidos e tem de escrever pelo menos um CO.
	Test	O SW com a correcção chega ao departamento de teste. A correcção tem de ser testada. O estado muda de implemented para test (pelo *MR-Tool* admin) assim que a *build* chega ao grupo de testes.
	Active	Razão pela qual o TR foi aberto. Se apenas um componente foi afectado não é necessário criar um CO. Neste caso a *build* que terá a solução tem de ser inserida no campo '*fix*ed in' do *MR-Tool*. Se mais que um componente foi afectado, COs tem de ser escritos. Active indica que pelo menos um CO alcançou o estado "active".
	Implemented	Se a solução foi introduzida numa certa *build* o *coordinator* tem de alterar o estado. O campo '*fix*ed in' inclui a *build* para entrega. Se alguns COs foram escritos o estado tem de ser alterado se todos os COs são implementados.
	Retest	Um TR no estado de retest dever ser testado novamente para produzir rastos, por exemplo. A propriedade "*fix*ed in" indica a verão na qual devem ocorrer os (re)testes. Não há entrega de uma solução final se assim fosse o estado seria test. Se existirem razões pare crer que um TR está resolvido e não foi feita nenhuma correcção especial então terá de ser novamente testada, ficando com o estado retest. Uma resposta tem de ser dada ao *tester* de forma a que este possa fazer o retest da forma indicada, ou seja que tipo de (re)teste tem de ser feito (rastreio ou correcção).
	observation	Erro não reproduzível. Não há solução porque os rastreios não permitem detector o erro. Por isso terá de se aguardar por mais testes. O próximo estado será FCB. Uma data terá de ser definida, assim como um *deadline* para a observação. Dependendo das circunstâncias (tudo foi arranjado ou nada foi resolvido) o estado final será closed ou rejected.
	closed	Test (ou retest) terminou com sucesso. A *build* respectiva (um test com o patch não é suficiente para fechar um TR) deve estar contida no campo „*fix*ed in" – do *MR-Tool*. Uma resposta tem de ser escrita.
	exported	TR exportado para outra base de dados.
	not concerned	TR não é relevante. Por exemplo o TR veio enganado de outra base de dados.

Titulo	State	
	Not *fix*ed	OTR falhou durante os testes, não está resolvido. Tem de se actualizar o campo „*fix*ed in“ – do *MR-Tool*. Uma resposta tem de ser dada descrevendo o comportamento.
	Fcb	O *coordinator* quer discutir o TR na FCB. Talvez queira rejeitar, passar a duplicate ou mudar para CR. Será possível alterar também o subsistema depois da FCB
	rejected	O TR foi aberto erroneamente e não será investigado (erro do operador, comportamento normal, sem funcionalidade, etc) A razão desta rejeição tem de ser dada no *MR-Tool*. Este estado só pode ser atribuído após uma FCB.
	duplicate	O TR é duplicado de um já existente, e já tratado. Este é um estado final sem mais qualquer acção. Nota: mesmo TR significa: o mesmo problema E mesma forma de encontrar o erro.
	unsolved	Uma pista (se requerida) é dada como resposta e uma funcionalidade específica é inserida no campo funcionalidade pelo MR-Admin.

A Metodologia dos CRs

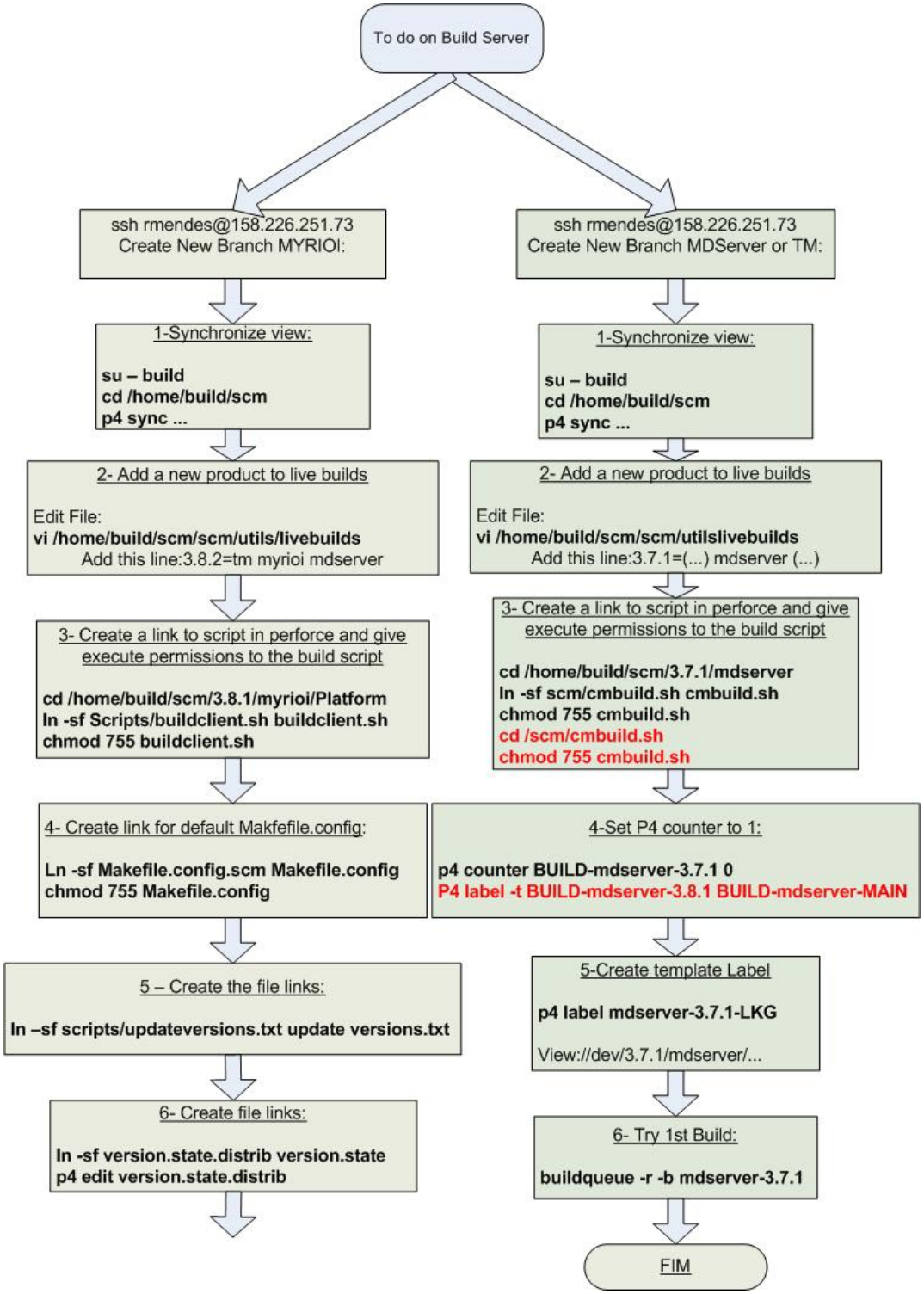
Estado	Significado	Usado para CR/CO	Determinado por
new	Novo, ainda não submetido, apenas visível para o autor do CR/CO.	CR, CO	*MR-Tool* quando o autor submete o CR/CO
sent	CR publicado na base de dados do projecto e visível a todos os colaboradores.	CR	*MR-Tool* quando o autor envia o CR
selected	CR seleccionado para análise, o coordenador prepara uma proposta para solução	CR	CCB
wait	Solução proposta, análise em andamento, são necessárias respostas.	CR	Coordenador
prepared	Todas as respostas necessárias no *MR-Tool*.	CR	*MR-Tool* quando todas as respostas foram dadas
proposed	CR está pronto para a decisão, a proposta sumariza todas as respostas relevantes	CR	*Coordinator*
decided	CR decidido, ÓS no entanto ainda não disponíveis.	CR	CCB
active	CR decidido, todas as ÓS disponíveis.	CR, CO	CO: Coordenador CR: *MR-Tool* quando todos os ÓS estão activos
test	CO a ser implementado, pronto para verificação.	CO	CO responsável
closed	CO implementação verificada, toda a documentação actualizada Estado Final.	CR, CO	CO: QA responsável CR: *MR-Tool* quando todos ÓS estão closed
rejected	CR não é mais reconhecido. Estado Final.	CR, CO	CCB
deferred	CR a ser implementado numa próxima versão. A versão é definida no campo '*fix*ed in'. Estado Final.	CR, CO	CCB
duplicate	CR já referido por outro CR ou funcionalidade.. Estado Final.	CR	CCB
not concerned	CR não relevante para esta a actual versão Estado Final	CR, CO	CCB

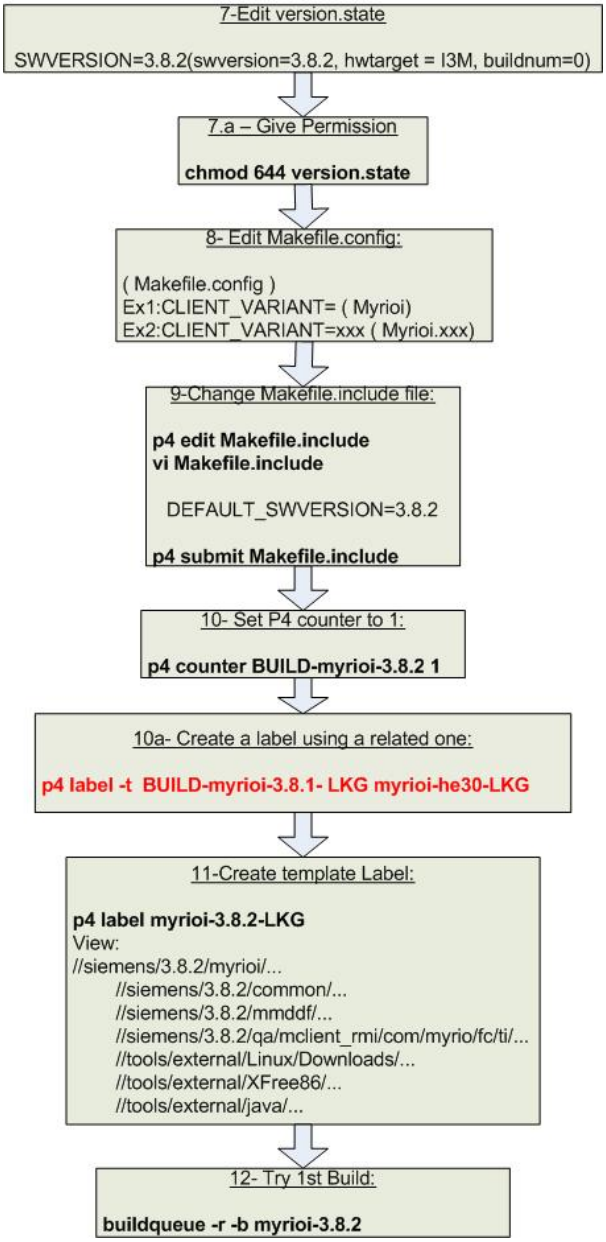
Algumas tarefas mais comuns

Configurar um brach para ser "*build*able"

Uma das necessidades crescentes deste projecto é o de criar *branches* para os novos produtos e funcionalidades que vão aparecendo. Um dos objectivos do plano de estágio era o de perceber o funcionamento do processo de *builds* e como consequência poder tornar os vários *branches* "*build*ables" em termos de *builds* à medida das necessidades das equipas.

Tendo em conta a especificidade de cada projecto, os vários serviços e plataformas existentes criámos uma check-list de forma a descrever os passos necessários para que um *branch* possa produzir *builds* (quer oficiais, quer não oficiais). Assim sendoem seguida são descritos os passos necessários para este efeito, que apesar de estarem demonstrados de forma resumida é um processo que demora cerca de duas horas a concluir.


To do on Build Server
ssh rmendes@158.226.251.73
Create New Branch MYRIOI:
ssh rmendes@158.226.251.73
Create New Branch MDServer or TM:
1-Synchronize view:
su – build
cd /home/build/scm
p4 sync ...
1-Synchronize view:
su – build
cd /home/build/scm
p4 sync ...
2- Add a new product to live builds
Edit File:
vi /home/build/scm/scm/utils/livebuilds
Add this line:3.8.2=tm myrioi mdserver
2- Add a new product to live builds
Edit File:
vi /home/build/scm/scm/utilslivebuilds
Add this line:3.7.1=(...) mdserver (...)
3- Create a link to script in perforce and give execute permissions to the build script
cd /home/build/scm/3.8.1/myrioi/Platform
ln -sf Scripts/buildclient.sh buildclient.sh
chmod 755 buildclient.sh
3- Create a link to script in perforce and give execute permissions to the build script
cd /home/build/scm/3.7.1/mdserver
ln -sf scm/cmbuild.sh cmbuild.sh
chmod 755 cmbuild.sh
cd /scm/cmbuild.sh
chmod 755 cmbuild.sh
4- Create link for default Makefile.config:
Ln -sf Makefile.config.scm Makefile.config
chmod 755 Makefile.config
4-Set P4 counter to 1:
p4 counter BUILD-mdserver-3.7.1 0
P4 label -t BUILD-mdserver-3.8.1 BUILD-mdserver-MAIN
5 – Create the file links:
ln –sf scripts/updateversions.txt update versions.txt
5-Create template Label
p4 label mdserver-3.7.1-LKG
View://dev/3.7.1/mdserver/...
6- Create file links:
ln -sf version.state.distrib version.state
p4 edit version.state.distrib
6- Try 1st Build:
buildqueue -r -b mdserver-3.7.1
FIM

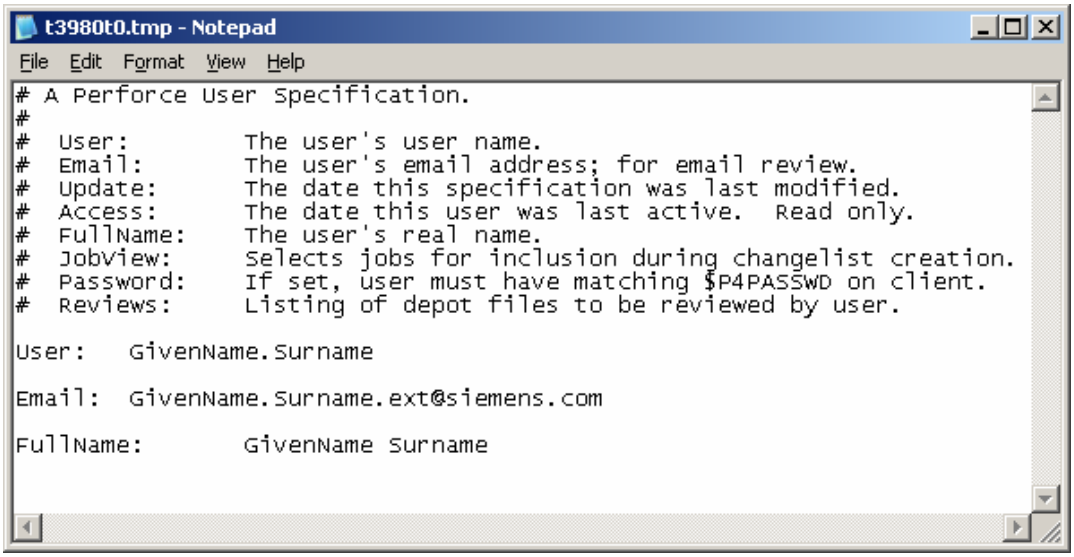

7-Edit version.state
SWVERSION=3.8.2(swversion=3.8.2, hwtarget = I3M, buildnum=0)
7.a – Give Permission
chmod 644 version.state
8- Edit Makefile.config:
(Makefile.config)
Ex1:CLIENT_VARIANT= (Myrioi)
Ex2:CLIENT_VARIANT=xxx (Myrioi.xxx)
9-Change Makefile.include file:
p4 edit Makefile.include
vi Makefile.include
DEFAULT_SWVERSION=3.8.2
p4 submit Makefile.include
10- Set P4 counter to 1:
p4 counter BUILD-myrioi-3.8.2 1
10a- Create a label using a related one:
p4 label -t BUILD-myrioi-3.8.1- LKG myrioi-he30-LKG
11-Create template Label:
p4 label myrioi-3.8.2-LKG
View:
//siemens/3.8.2/myrioi/...
//siemens/3.8.2/common/...
//siemens/3.8.2/mmddf/...
//siemens/3.8.2/qa/mclient_rmi/com/myrio/fc/ti/...
//tools/external/Linux/Downloads/...
//tools/external/XFree86/...
//tools/external/java/...
12- Try 1st Build:
buildqueue -r -b myrioi-3.8.2

Adicionar utilizadores
Por defeito o *Perforce* cria um utilizador na sua base de dados sempre que um comando é gerado por um utilizador que não existe. Os denominados super utilizadores podem criar novos utilizadores usando a flag –f (force) no comando user:

P4 user –f nome.apelido

Depois de ser submetido este comando é gerado um ficheiro temporário no qual se pode ver toda a informação sobre o novo utilizador:

São preenchidos os campos com a informação relevante.
Todas as alterações são submetidas.



Como forma de prevenir o *Perforce* da criação automática de utilizadores, todos deverão estar definidos numa tabela de protecções. A forma mais simples do o fazer é incluir todos os utilizadores num grupo e configurar os acessos apenas para os membros do grupo em questão.

Os privilégios de acesso para cada grupo é definido pela equipa de SCM no ficheiro P4 protect como poderemos verificar mais à frente.
Submeter um novo utilizador num grupo
O SCM deve impedir o servidor de *Perforce* de criar novos utilizadores. Tudo começa por criar grupos, como no caso do grupo users-siemens-LIS usado como exemplo:

P4 group users-siemens-LIS

Depois do comando ser dado será gerado o ficheiro temporário no qual se pode ver toda a informação:

São preenchidos os campos coma informação relevante.
Todas as alterações são submetidas.

Segurança no *Perforce*

Sempre que adicionamos um novo grupo ou um utilizador ele precisa de permissões para poder fazer alterações no código aos *branches* onde trabalha. Para isso é definido um ficheiro de políticas pelo qual o servidor de *Perforce* se rege, e que tem o nome de "**protect".** Este ficheiro é editável bastando para isso ter permissões de super utilizador e digitar o seguinte comando:

P4 protect

As próprias permissões de super utilizador também elas estão definidas neste ficheiro. A metodologia utilizada é a de dar permissões máximas e ir restringindo à mediada que vamos caminhando pela a árvore de directórios.

*View*s e clientspecs

A *view* de um utilizador representa aquilo que o mesmo vê na sua *workspace*, ou seja o mapeamento das directorias que estão no servidor no disco local. Esta pode ser uma forma de filtrar acessos a outros *branches* embora não muito segura pois a *view* pode ser editada pelos próprios utilizadores. No entanto, a verdadeira funcionalidade de uma *view* é a de mapear conteúdos, conteúdos esses que variam de projecto para projecto.

Ao conjunto de directorias que são mapeadas na aplicação client dos utilizadores ou na *workspace* damos o nome de clientspec. Nos Anexos estão as diferentes clientspecs de acordo com os vários projectos. Desde já, fica a indicação da existência de directorias comuns facilmente explicadas pela história de integrações ao longo do tempo.

*Script*ing de algumas tarefas

*Script*ing de algumas tarefas

Algumas tarefas pelo facto de serem tão usualmente requisitadas à equipa de SCM foram tornadas automáticas, nesse sentido decidimos utilizar um conjunto de *Script*s que realizam tarefas como:

Saber a lista de *change*list de uma dada *build* relativamente a outra

```
@REM EXAMPLE: diff labels.bat myrioi-3.8.1-54 myrioi-3.8.1-55
@echo off

p4 changes @%1 > dl_file1
p4 changes @%2 > dl_file2
c:\cygwin\bin\diff.exe dl_file1 dl_file2
del dl_file1
del dl_file2
```

Saber a lista de *change*lists submetidas na última *build* num certo *branch*

```
@echo off
REM List all CL since a given label until now
REM EXAMPLE: diff labelsV2 myrioi-3.8.1-54

p4 label -o %1  find "//" > dl_file0
FOR /F "tokens=1* delims=          " %%A IN ('type dl_file0') do call :do_job
%1      %%A
del dl_file0
goto :eof

:do_job
echo CL on "%2" :
p4 changes %2@%1 > dl_file1
p4 changes %2@now > dl_file2
c:\cygwin\bin\diff.exe dl_file1 dl_file2  find "Change"
del dl_file1
del dl_file2
echo.
```

Saber a última changelist que entrou num dado branch

```
@echo off
REM EXAMPLE: findlastbuild myrioi main
p4 counters  find "%1-%2"  find /v "devaux"  find /v "pvr"
```

Saber o que um *developer* submeteu, num determinado *branch*, filtrado por data

```
p4 changes –u ruben.mendes //devaux/he30/fb_client/...@2007/06/04,2007/06/08
```