Universidade de Lisboa

Faculdade de Ciências

Departamento de Informática

Portagem da aplicação ODS para Linux

Projecto realizado na

NAV Portugal E.P.E.

Por

Leonel Martins da Costa Duarte

(Versão Pública)

Mestrado em Engenharia Informática

2008

Universidade de Lisboa

Faculdade de Ciências

Departamento de Informática

Portagem da aplicação ODS para Linux

Projecto realizado na

NAV Portugal E.P.E.

Por

Leonel Martins da Costa Duarte

Projecto orientado por Prof. Dr. Hugo Miranda

e co-orientado por Eng. José dos Santos Mestre Vermelhudo

Mestrado em Engenharia Informática

2008

FACULDADE • DE • CIÊNCIAS UNIVERSIDADE • DE • LISBOA

Declaração

Leonel Martins da Costa Duarte, aluno nº25887 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 ”Portagem da aplicação ODS para Linux”, realizado no ano lectivo de 2007/2008 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 Novembro de 2008

José dos Santos Mestre Vermelhudo supervisor do projecto de Leonel Martins da Costa Duarte, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado ”Portagem da aplicação ODS para Linux”.

Lisboa, 12 de Novembro de 2008

Resumo

A aplicação ODS é parte integrante do sistema LISATM, utilizado para o controlo do tráfego na região de voo de Lisboa. Está implementada sobre o sistema operativo Solaris 8 e o objectivo deste projecto é migrar o código da aplicação de visualização por forma a que possa utilizar o sistema operativo Linux. A portagem envolve não só a migração do código da aplicação mas também assegurar que as ferramentas auxiliares que utiliza são aceites pela nova versão do compilador e envolve também a configuração do SO.

O ODS é desenhado com base na plataforma ODS Toolbox implementada sobre X Windows. A ODS Toolbox é um conjunto de ferramentas com o objectivo de desenhar sistemas e aplicações em que os objectos dinâmicos são visualizados num ambiente de interface gráfico.

É indispensável que a aplicação tenha exactamente o mesmo comportamento e aparência para os controladores de tráfego aéreo.

Agradecimentos

Ao Prof. Hugo Miranda pela sua disponibilidade, apoio e acompanhamento de todo o estágio.

Ao Eng. José Vermelhudo pela oportunidade, orientação e disponibilidade oferecidas.

Ao Eng. José Alexandrino pela orientação, conhecimentos partilhados e disponibilidade.

Ao Eng. Paulo Monteiro pelos conhecimentos partilhados e disponibilidade.

Ao Eng. Manuel Campaniço e Eng. Rui Pereira pela ajuda na integração na empresa.

Índice

Resumo vii
Agradecimentos ix
Índice xi
Índice de Imagens xiv
Índice de Anexos xv
1. Introdução 1
1.1. Âmbito 1
1.2. A Empresa 1
1.3. Acrónimos e Abreviaturas 5
1.4. Organização do documento 6
2. Solaris e Linux 7
2.1. Bibliotecas e Headers 7
2.2. Gestão de ficheiros 8
2.3. Comunicação 8
2.4. Threads 8
2.5. Gestão de Memória 9
2.6. Gestão de Processos 9
2.7. Timers 10
2.8. Sinais 10
3. O Projecto 11
3.1. O ODS 11
3.2. Objectivos 15
3.3. Ferramentas 16
3.3.1. ODS Toolbox 16
3.3.1.1. Componentes da ODS Toolbox 17
3.3.1.1.1. ODS Toolbox Interface Editor System 19
3.3.1.1.2. ODS Toolbox Kernel 19
3.3.1.1.3. Componentes de Serviço ODS Toolbox 19
3.3.1.1.3.1. ODS Window Manager 20

3.3.1.1.3.2. Interoperabilidade 20
3.3.1.1.3.3. ODS Toolbox Recording and Replay 21
3.3.1.1.3.4. Communication Converters 21
3.3.1.1.3.4.1. Communication Converter Asterix 21
3.3.1.1.3.4.2. Communication Converter ASN.1/BER 22
3.3.1.1.3.5. Color Server 23
3.3.2. LOKI 23
3.3.3. Cppunit 1.10.2 24
3.3.4. Snacc 1.3 24
3.3.5. Glib 1.2.10 24
3.3.6. MTL 2.1.2-21 24
3.3.7. ACE 5.5 24
3.3.8. CVS 1.11.20 24
3.3.9. EXPECT 5.40 24
3.3.10. TCL 8.4.9 24
3.3.11. TK 8.4.9 24
3.3.12. TCPDUMP 3.9.4 24
3.3.13. GDB 6.0 25
3.3.14. GCC 3.4.2 25
3.3.15. M4 1.4.2 25
3.3.16. RSYNC 2.6.8 25
3.3.17. PERL 5.8.7 25
3.3.18. MAKE 3.80 25
3.3.19. GAWK 3.1.4 25
3.3.20. Bugzilla 25
4. Trabalho Realizado 26
4.1. Integração 26
4.2. Documentação 26
4.2.1. Ponto de Situação 28

4.2.2. Interface Requirements Specification 28
4.3. Adaptação de código fonte e makefiles 29
4.4. Testes Unitários 31
4.4.1. Testes de Funcionalidade 31
4.5. Testes de Integração 31
4.6. Testes de Validação 31
4.7. Principais dificuldades 32
4.8. Trabalho Futuro 32
4.9. Planeamento Previsto 32
4.10. Planeamento Cumprido 33
5. Conhecimentos Adquiridos 34
6. Conclusão 35
7. Anexos 36

Índice de Imagens

Figura 1. DSTI 4
Figura 2. Sistema LISATM 11
Figura 3. Arquitectura do ODS 12
Figura 4. Diagrama de interfaces 14
Figura 5. Screenshot da aplicação ODS da posição de supervisão 15
Figura 6. Componentes da ODS Toolbox 18
Figura 7. Interfaces do atcwm 20
Figura 8. Communication Converter Asterix 21
Figura 9. Communication Converter ASN.1/BER 22
Figura 10. Procedimento operacional de desenvolvimento de sistemas em vigor na NAV 27

Índice de Anexos

Anexo 1. Planeamento previsto 37
Anexo 2. Planeamento cumprido 39

1. Introdução

1.1. Âmbito

Este documento descreve o trabalho realizado no âmbito da disciplina de Projecto em Engenharia Informática da Faculdade de Ciências Universidade de Lisboa.

Com o sistema NAV1 (anterior ao LISATM) a chegar ao seu limite, tanto em capacidade como em tempo de vida, a NAV encarregou uma empresa francesa de fornecer um novo sistema para a gestão e monitorização de tráfego aéreo. Essa empresa não conseguiu cumprir o contrato e coube à NAV procurar alternativas. O sistema LISATM foi desenvolvido tendo por base os contínuos desenvolvimentos realizados na modernização do sistema NAV1 (baseado em tecnologia P800), bem como da versão de Torre de Controlo, com garantias dadas. A aplicação ODS é parte integrante deste sistema e disponibiliza a interface com o utilizador CTA.

A aplicação ODS foi, originalmente, desenvolvida em Solaris 8. Com o suporte para este SO a chegar ao fim tornou-se necessário portar a aplicação para outro SO. As alternativas para a portagem da aplicação são Linux e Solaris 10. Linux permite a utilização de ferramentas de desenvolvimento que não estão disponíveis em Solaris 10. Por outro lado, Solaris 10 permite a monitorização da aplicação em termos de *performance* sem utilização de recursos do SO, algo não possível em Linux. Numa primeira fase a portagem será feita para Linux e posteriormente para Solaris 10. Este relatório restringe-se à portagem para Linux.

1.2. A Empresa

A NAV Portugal tem como missão prioritária a prestação de Serviços de Tráfego Aéreo nas Regiões de Informação de Voo sob a responsabilidade Portuguesa – Lisboa e Santa Maria, garantindo o cumprimento da regulamentação Nacional e Internacional nas melhores condições de Segurança, optimizando Capacidades, privilegiando a eficiência e sem descurar preocupações ambientais. A empresa exerce a sua actividade no Continente e nas regiões autónomas dos Açores e da Madeira. Junto ao Aeroporto de Lisboa está situada a Sede da empresa, o Centro de Controlo de Tráfego Aéreo de Lisboa e o Centro de Formação. Na região Autónoma dos Açores, concretamente na ilha de Santa Maria, está situado o Centro de Controlo Oceânico. Para além destes dois importantes Centros, a NAV Portugal tem ainda outras infra-estruturas com Serviços de Tráfego Aéreo a funcionar nas Torres de Controlo dos Aeroportos de Lisboa, Porto, Faro, Funchal, Porto Santo, Santa Maria, Ponta Delgada, Horta, Flores e no Aeródromo de Cascais. Para a plena concretização da sua missão de Controlo de Tráfego Aéreo, a NAV Portugal possui um vasto conjunto de equipamentos e instalações técnicas (estações radar, rádio-ajudas e comunicações) em vários pontos do Continente e Regiões Autónomas.

Nos serviços que a empresa fornece incluem-se:

- Promover o fluxo ordenado, seguro e expedito das aeronaves;
- Fornecer todas as informações e sugestões úteis à segurança dos voos;
- Evitar colisões entre aeronaves;
- Evitar colisões entre aeronaves e obstáculos no solo;
- Alertar os organismos apropriados sempre que uma aeronave se encontre numa situação de emergência e necessite dos Serviços de Busca e Salvamento e prestar a esses organismos toda a cooperação necessária;

A NAV Portugal presta o seu serviço nas várias fases do voo:

1. Piloto ou companhia submete um Plano de Voo

2. Plano de Voo é enviado aos órgãos de controle de tráfego aéreo envolvidos na condução do Voo

3. Aeronave recebe autorização para partir

4. Aeronave descola em contacto rádio com a Torre de Controle de Aeródromo

5. Aeronave em subida passa ao contacto rádio e radar com o Controle de Aproximação

6. Aeronave em subida para a altitude de cruzeiro passa ao contacto rádio e radar do Centro de Controle Regional

7. Aeronave atinge o nível de cruzeiro e prossegue em rota em contacto rádio e radar com um ou mais centros de Controle Regional

8. Aeronave inicia a descida em contacto rádio e radar com um centro de Controle Regional

9. Aeronave é transferida para o Controle de Aproximação, dando continuidade à descida e procedimento de aproximação

10. Aeronave efectua a última trajectória de aproximação e aterra em contacto rádio com a Torre de Controle de Aeródromo. Neste momento termina o serviço prestado pela NAV Portugal, passando para a esfera aeroportuária ou das companhias de aviação.

O estágio objecto deste relatório foi realizado na Divisão SISPRO da Direcção de Sistemas e Tecnologias de informação, cuja estrutura e breve descrição é ilustrada na figura seguinte.

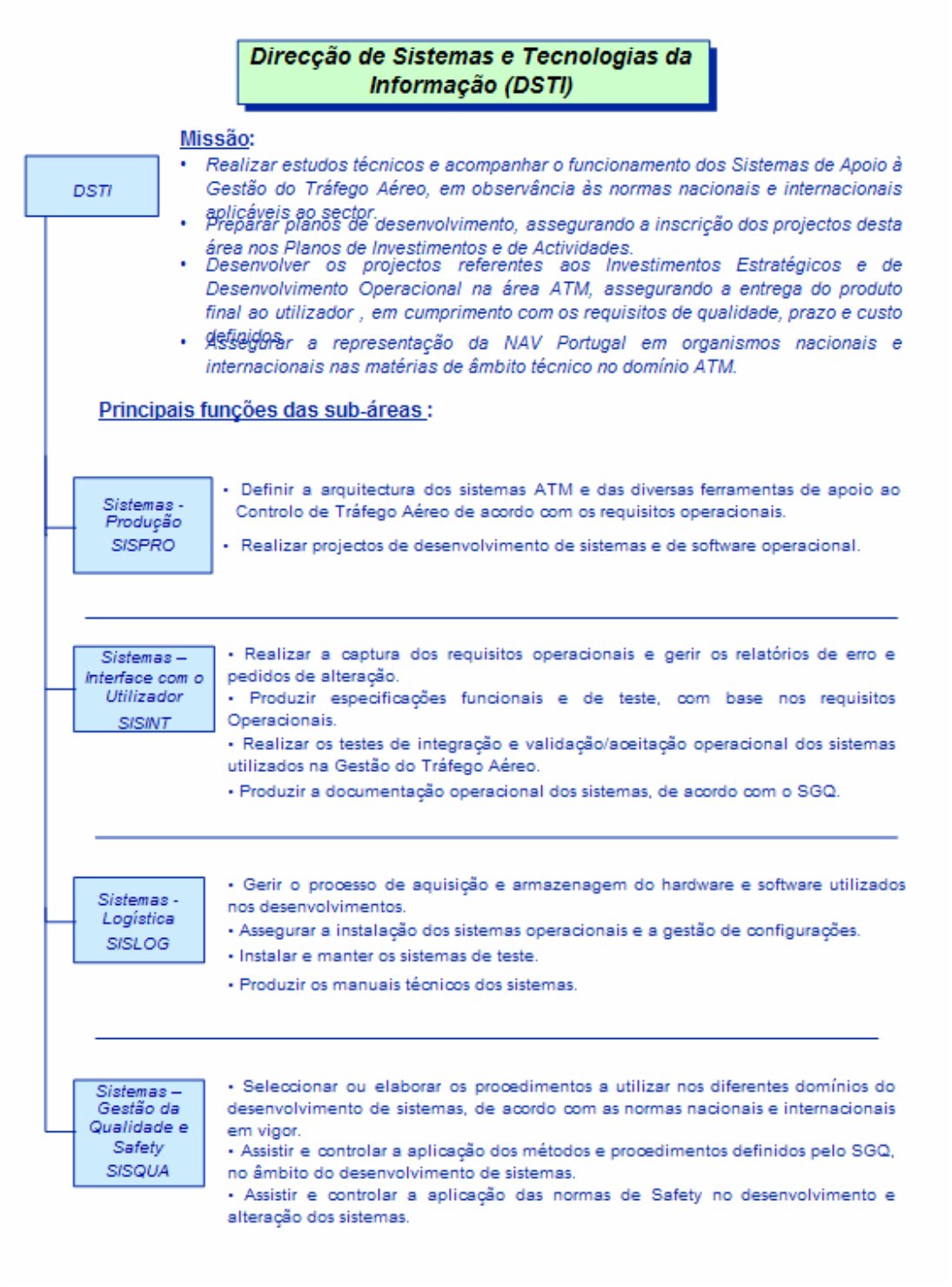


Figura 1. DSTI

1.3. Acrónimos e Abreviaturas

ACE	Adaptive Communication Environment
API	Application Programming Interface
ASTERIX	All purpose STructured Eurocontrol suRveillance Information eXchange format
ATC	Air Traffic Controller
ATD	Acceptance Tests Document
ATO	Actual Time Over route point
ATR	Acceptance Tests Results
ATSS	ATO TFC and Safety net System
CCA	Communication Converter ASN.1
CCX	Communication Converter Asterix
CEX	Communication Editor Asterix
CO	Control Object
COPS	Community Oriented Policing Services program
CPDLC	Control Pilot Data Link Communication
CTA	Controlador de Tráfego Aereo
CVS	Concurrent Versions System
DLPI	Data Link Provider Interface
DSTI	Direcção de Sistemas e Tecnologias de Informação
CSCI	Computer Software Configuration Item
CWP	Controller Working Position
ENV	ENVironment
FDPS	Flight Data Processing System
FIR	Flight Information Region
FPL	Flight PLan
FPP	Flight Plan data Processing
IES	Interface Editor System
IRS	Interface Requirements Specification
LISATM	LISbon Air Traffic Management
NPTL	Native POSIX Thread Library
ODS	Operator Display System
OLDI	Online Link Data Exchange
OTR	ODS Toolbox Recording and Replay
QNH	Queens Nautical Height
RDPS	Radar Data Processing System
RIV	Regiões de Informação de Voo
SCM	System Control and Monitoring
SCTL	Solaris Compatible Thread Library
SISPRO	SIStemas e PROdução de software
SO	Sistema Operativo
STIN	Short Term conflict alert Inhibition
TFC	Track Flight Correlation
TMP	Test Management Plan
UIMS	User Interface Management System

1.4. Organização do documento

O documento encontra-se organizado da seguinte forma:

- Solaris e Linux – uma comparação dos dois sistemas operativos com enfoque no desenvolvimento, que abrange diferenças ao nível das bibliotecas e *header files*, gestão de ficheiros, comunicação, *threads*, gestão de memória, gestão de processos, *timers* e sinais.

- O Projecto – descrição da aplicação ODS e das ferramentas que utiliza com atenção particular para a ODS Toolbox sobre a qual o ODS é desenhado.

- Trabalho Realizado – inclui a documentação elaborada e a descrição das diferentes tarefas que fazem parte do projecto, as principais dificuldades encontradas, o trabalho a realizar no futuro e aborda o planeamento previsto e o planeamento cumprido.

- Conhecimentos adquiridos – a realização deste projecto permitiu adquirir novos conhecimentos que são enumerados nesta secção

- Conclusão

2. Solaris e Linux

Antes de começar a portagem propriamente dita, aprofundei a questão dos diferentes sistemas operativos para descobrir o que me esperava, com especial atenção à gestão de memória, comunicação e gestão de processos. Ainda sem entrar em concreto nos subsistemas uma ideia sobressai no que respeita à portagem: apesar de ambos os sistemas operativos partilharem standards POSIX e uma vez corrigidos os problemas de compilação, a aplicação pode ter um comportamento totalmente diferente do desejado. A conclusão que se tira é que numa portagem é muito importante testar todos os componentes nos mais diversos cenários e a forma como interage com interfaces externas.

2.1. Bibliotecas e Headers

As bibliotecas de sistema fornecem funcionalidade semelhante em ambos os sistemas operativos. Aplicações desenvolvidas nos standards POSIX são portáveis apesar de existirem diferenças. Uma tarefa importante é identificar APIs e bibliotecas utilizadas fora do standard. Por exemplo, Solaris utiliza duas bibliotecas de *threads*. No entanto uma biblioteca é anterior ao standard POSIX. Uma portagem de uma aplicação que faça uso dessa biblioteca vai ter problemas nesse caso específico.

De uma forma geral os problemas na portagem podem ser classificados na seguinte forma:

- APIs em bibliotecas diferentes

- *Headers* diferentes
  Algumas APIs existem em ambos os SO mas necessitam de *headers* diferentes

- APIs e *headers* não existem
  Algumas APIs existem em Solaris mas não existem em Linux. Este problema acontece principalmente em APIs que não seguem o standard POSIX

- Protótipo e *headers* diferentes
  Alguns protótipos de API e declarações de *headers* existem em ambos os SO mas são diferentes

- Tipo de *return* diferente
  Alguns protótipos de APIs especificam tipos de *return* diferentes nos SO

- Diferenças em *errno*
  Ambos os SO partilham APIs equivalentes quando terminam com sucesso mas podem retornar valores de *errno* diferentes em caso de erro se o valor de *errno* não é especificado pelo POSIX

- Diferenças semânticas

2.2. Gestão de ficheiros

Tanto o Solaris como o Linux cumprem o standard POSIX nesta área. Estas APIs são implementadas como chamadas a sistema em ambos os SO. De uma forma geral o Solaris oferece melhor suporte para *clustering* e utiliza *flags* que não existem em Linux. As APIs que não existem em Linux têm equivalentes e uma outra API difere apenas nos argumentos e *flags*.

2.3. Comunicação

Ao nível da comunicação o código deve simplesmente ser recompilado em Linux e deve funcionar. Existem, no entanto, algumas diferenças. Linux suporta mais domínios ao nível da rotina *socket()* e aceita *raw packets* de forma imediata. Em Solaris esta funcionalidade é conseguida através de DLPI.

2.4. Threads

O Solaris suporta duas *packages* distintas de *threads*:

- Threads Solaris – biblioteca de *threads* não-POSIX proprietária da Sun. Como alternativa em Linux existe uma biblioteca compatível em *open-source*.
- Threads POSIX – facilmente portáveis para outras plataformas

Em Linux existem várias *packages* de *threads* distintas incluindo:

- LinuxThreads – disponível a partir do *kernel* 2.0.0
- Native POSIX Thread Library – disponível a partir do *kernel* 2.6

Existe ainda a Solaris-Compatible Threads Library desenvolvida pela HP para assistir na migração de aplicações de Solaris para Linux.

Abordando a portagem, muitas APIs da *package* Threads Solaris têm uma correspondência com APIs da *package* NPTL em Linux. Outras alternativas possíveis são alterar a aplicação para utilizar Threads POSIX ou utilizar SCTL para suportar o código com poucas alterações em Linux.

Ambos os SO implementam o standard POSIX 1003.1c. Desta forma o código que faça uso de APIs da *package* Threads POSIX deve portar sem problemas. Existem, no entanto, funções implementadas em Solaris que não estão disponíveis em Linux. Se a aplicação utiliza alguma dessas funções não existe alternativa que não seja recodificar.

De um ponto de vista mais funcional pode-se dizer que:

- Threads POSIX são portáveis
- *Threads* POSIX estabelecem características para cada *thread* de acordo com atributos de objecto configuráveis
- *Threads* POSIX implementam cancelamento de *threads*
- *Threads* POSIX implementam algoritmos de *scheduling*
- *Threads* POSIX permitem *clean-up handlers* para chamadas a *fork*
- *Threads* Solaris podem ser suspensas e continuadas
- *Threads* Solaris implementam *locks* a *mutexes* entre processos
- *Threads* Solaris implementam *threads daemon*, para as quais o processo não espera a terminação

2.5. Gestão de Memória

Sem entrar numa descrição exaustiva dos mecanismos de gestão de memória de Solaris e Linux é possível dizer que são bastante semelhantes, embora os nomes das estruturas de dados sejam completamente diferentes. Por exemplo, em Solaris o espaço de endereçamento de um processo é designado segmento. Em Linux é chamado de área de memória.

Áreas de memória e segmentos são delimitados por:

- Endereço virtual do inicio da área
- A sua localização dentro do objecto ou ficheiro que a área de memória / segmento mapeia
- Permissões
- Tamanho da projecção

Ao nível do desenvolvimento existem várias bibliotecas de alocação de memória e a maior parte estão disponíveis para ambos os SO.

2.6. Gestão de Processos

Um processo, em ambos os SO, é uma instância de um programa em execução. Um processo engloba um espaço de endereçamento e uma ou mais *threads*. Cada processo no sistema tem um *process ID* que permanece único até algum tempo depois do processo morrer. Os processos são criados utilizando *fork()* e variantes. Em Linux processos e *threads* também podem ser criados utilizando *clone()*. Solaris para além do *fork()* tem também *forkall()* que replica todas as *threads* do processo original.

Ambos os sistemas operativos suportam a noção de *binding* que liga um processo ou *thread* a um processador. Em Linux essa ligação é não exclusiva ao contrário de Solaris que é exclusiva. Esta noção é designada *fencing*. Linux não possui um mecanismo para *fencing* por defeito mas é possível implementar.

2.7. Timers

Ao nível da aplicação ambos os SO oferecem rotinas POSIX de *timers*. No entanto, o Solaris tem um *timer* adicional com uma precisão ao nanosegundo. Para obter essa precisão em Linux é necessário instalar um *patch* ao *kernel*.

2.8.Sinais

Solaris e Linux tratam os sinais de forma semelhante, embora alguns sinais existam em Solaris e não existam em Linux e vice-versa. Em Linux o tratamento de sinais difere do standard. O standard POSIX diz que um sinal entregue de forma assíncrona (um sinal de origem exterior a um processo) é tratado por qualquer *thread* que não tenha o sinal bloqueado. Em Linux podem ser enviados sinais assíncronos a *threads* específicas. Solaris, por outro lado, implementa o standard. Não existe forma de enviar um sinal a uma *thread* específica externa ao processo. É possível enviar um sinal ao processo, mas não a uma *thread* específica dentro do mesmo.

Em ambos os SO os sinais são tratados quando um sinal não bloqueado e não ignorado é encontrado em espera por uma *thread* que volta do modo *kernel* para modo utilizador. Em ambos os SO SIGKILL e SIGSTOP têm prioridade sobre todos os outros sinais. Em Solaris os sinais são tratados pelo número do sinal (mais baixo primeiro). Em Linux são tratados pela ordem em que são recebidos.

3. O Projecto

3.1. O ODS

A aplicação ODS faz parte do sistema LISATM (figura 2). O sistema LISATM é um Sistema de Informação de arquitectura aberta, que permite a sua constante melhoria, para o Controlo de Tráfego Aéreo da FIR de Lisboa.

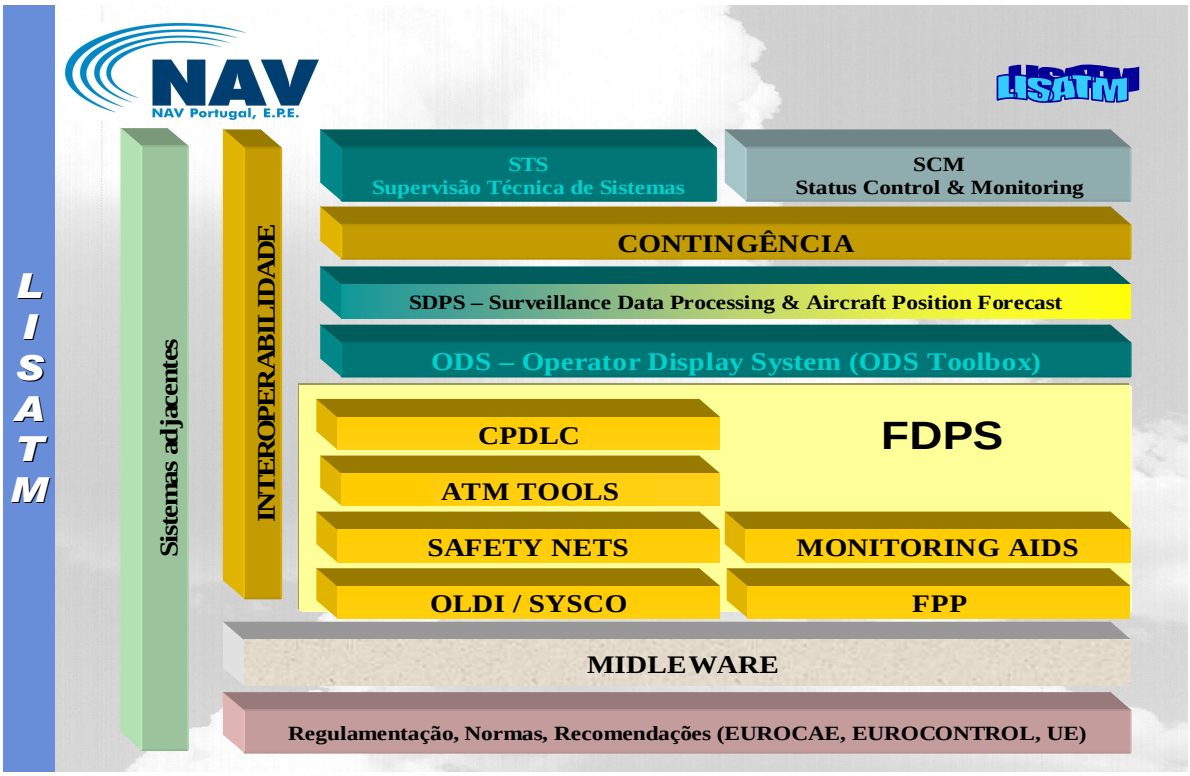


Figura 2. Sistema LISATM

A principal função da aplicação ODS é disponibilizar a visualização da informação radar e permitir transacções com outros subsistemas aos operacionais ATC. Neste momento existem 30 posições para controladores de tráfego aéreo.

A figura 3 mostra a arquitectura da aplicação e a tabela 1 descreve as ligações encontradas na arquitectura. A figura 4 ilustra as trocas de mensagens entre as interfaces e a aplicação ODS.

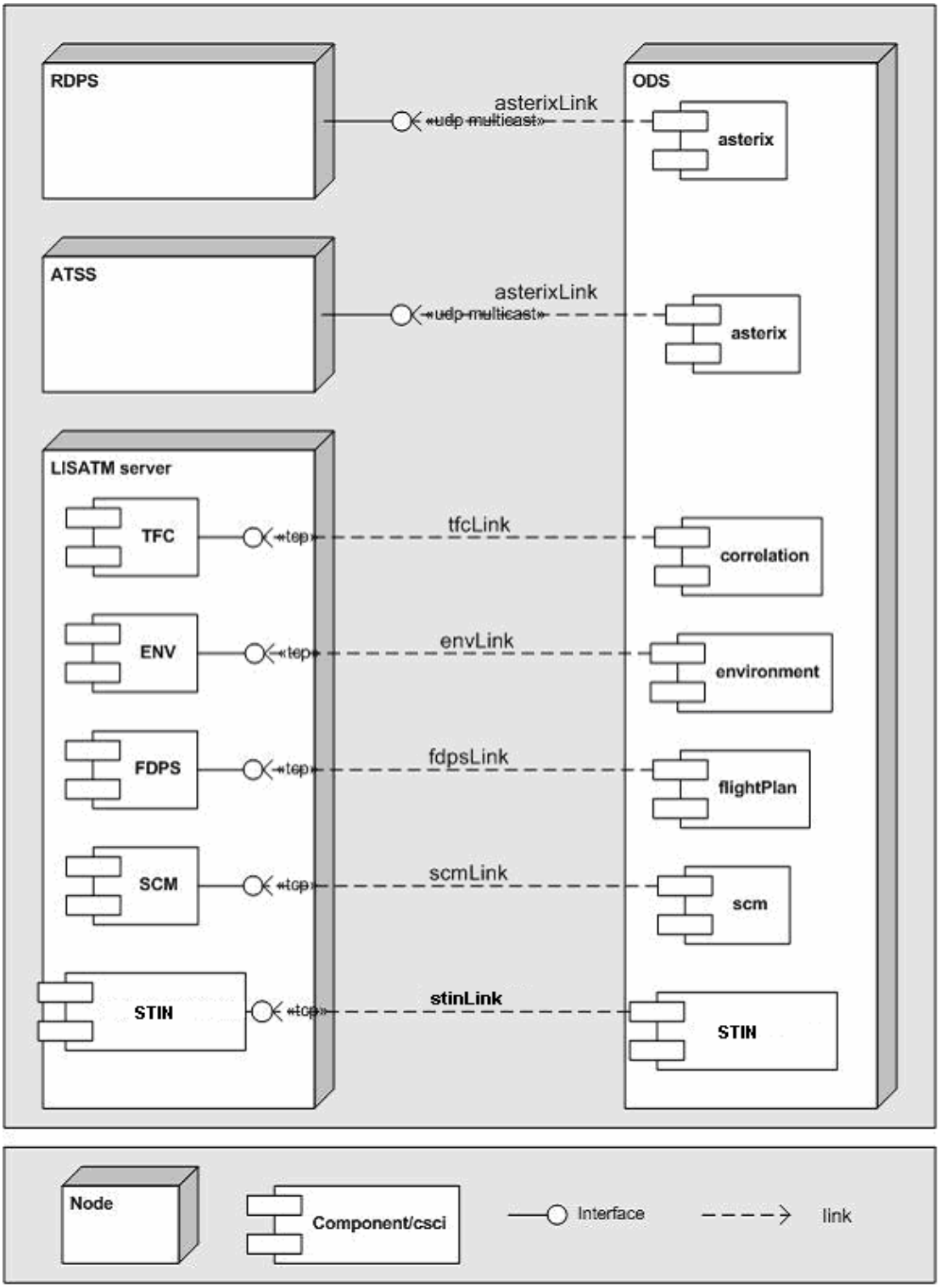


Figura 3. Arquitectura do ODS

Interface ID	Descrição e Objectivo
asterixLink	O ODS interage com o: • RDPS para obter dados radar • ATSS para obter dados radar e planos de voo limitados.
tfcLink	O ODS interage com o TFC para requisitar correlação e descorrelação manual *Track-Flight*.
envLink	O ODS interage com o ENV para obter dados do ambiente relacionados com a pista e volumes.
fdpsLink	O ODS interage com o FDPS para receber planos de vôo e para pedir *updates* destes dados.
scmLink	O ODS interage com o SCM para obter informação de monitorização sobre outros CSCIs e para relatar ao SCM o estado da monitorização do ODS CSCI.
stinLink	O ODS interage com o STIN para obter informação sobre áreas de informação inibidas, *callsigns* e *ssrcodes* do componente STCA.

Tabela 1. Interfaces com a aplicação ODS

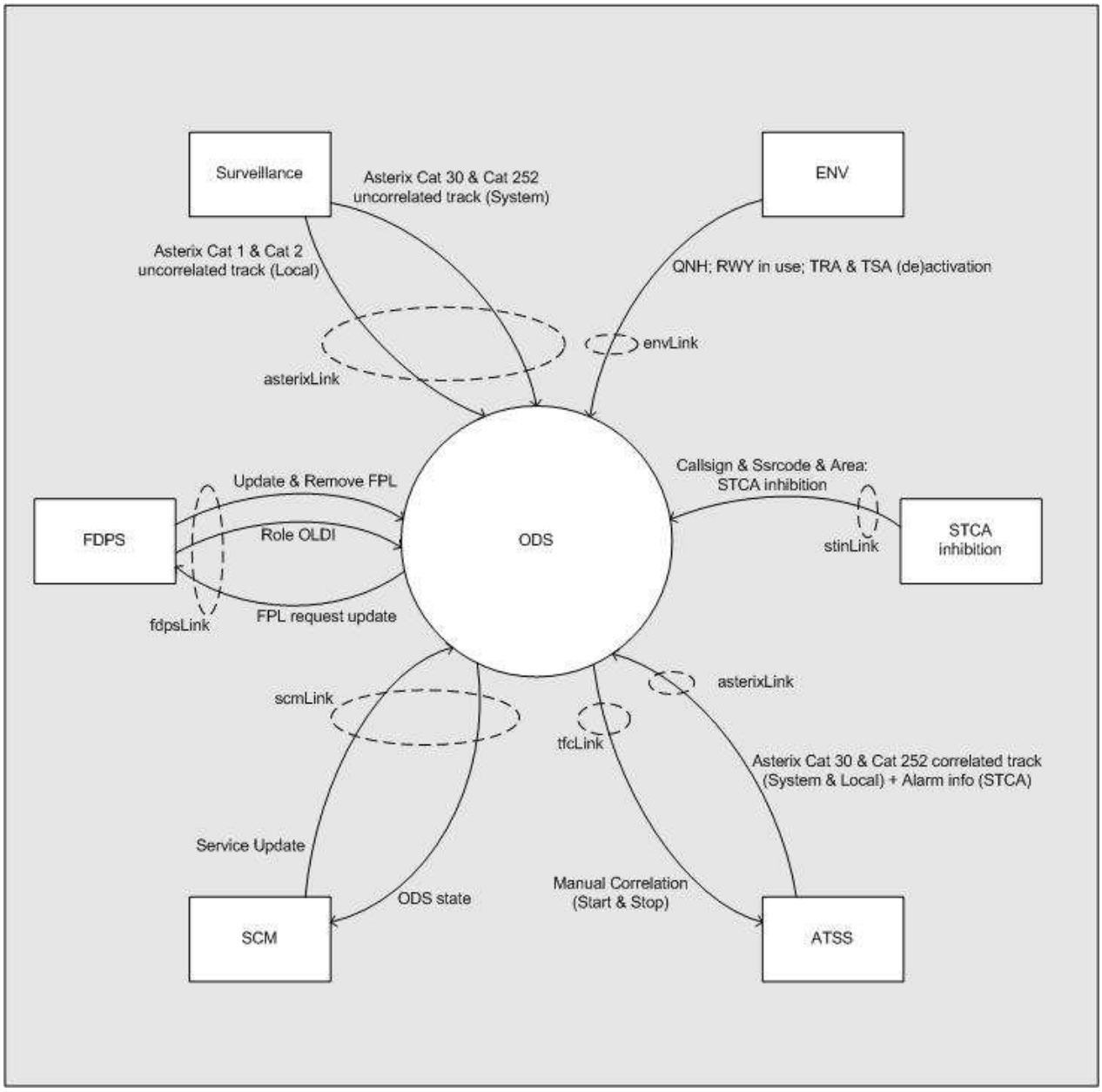


Figura 4. Diagrama de interfaces

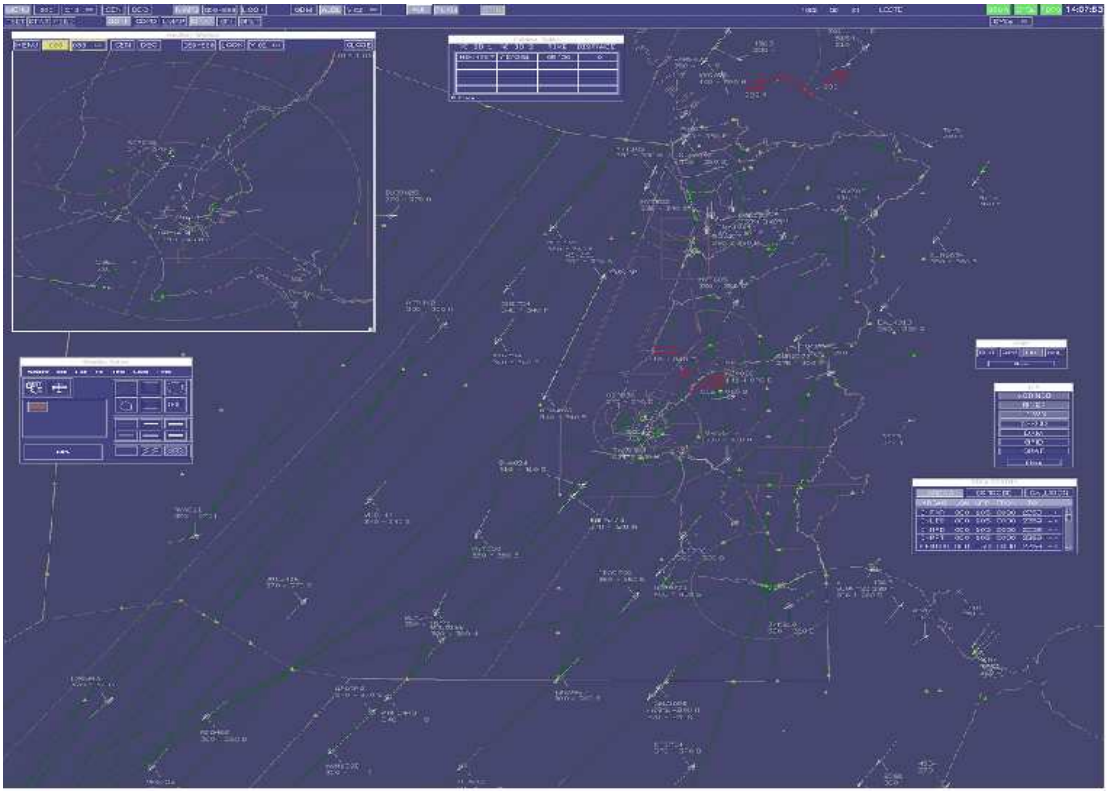


Figura 5. Screenshot da aplicação ODS da posição de supervisão

A figura 5 mostra uma *screenshot* da aplicação ODS. Uma vez concluída a portagem a aplicação deve ter o mesmo aspecto e comportamento da aplicação original.

3.2. Objectivos

Os controladores de tráfego aéreo recebem formação para trabalhar com a aplicação ODS. Sendo uma área sensível é muito importante que a aplicação portada tenha um aspecto gráfico e comportamento exactamente semelhante à original. Com o objectivo de concretizar a portagem foi-me disponibilizado acesso a repositórios, máquinas e recursos humanos.

Associado à portagem da aplicação deve ser mantido um registo de todas as alterações realizadas sobre o código original e configuração do sistema sobre o qual a aplicação irá ser executada. Esse registo é feito através do Bugzilla[1].

[1] http://www.bugzilla.org/

Para além da aplicação portada propriamente dita, é também necessário elaborar toda a documentação associada. Documentos como o IRS, TMP, ATD e ATR são importantes para a gestão da aplicação.

3.3.Ferramentas

A aplicação ODS é desenvolvida sobre a ODS Toolbox que permite o design gráfico e a definição de objectos dinâmicos. A aplicação utiliza outras ferramentas que auxiliam no desenvolvimento da aplicação e na depuração de erros.

Os pontos seguintes dão uma breve descrição das ferramentas presentes na aplicação com especial destaque para a ODS Toolbox.

3.3.1. ODS Toolbox

A ODS Toolbox foi desenhada para suportar o desenvolvimento e configuração de sistemas de visualização de objectos gráficos. Embora vocacionada para o ambiente de controlo de tráfego aéreo, é uma ferramenta que pode ser utilizada para uma grande variedade de aplicações onde objectos dinâmicos são visualizados dentro de uma interface gráfica e interactiva. Como exemplos típicos de aplicações deste género temos:

- Controlo tráfego aéreo
- Controlo processos
- Controlo de redes e sistemas
- Controlo de navios

O desenvolvimento da ODS Toolbox está intrinsecamente ligado à definição dos standards para sistemas ATC futuros como o COPS.

A ODS toolbox fornece um conjunto de editores incluindo um UIMS, bibliotecas *run-time* de alto desempenho para gráficos interactivos, comunicação, manipulação de dados, blocos pré-construídos para *displays* ATC standard e várias funções que suportam:

- *Recording* e *replay*
- Conectividade
- Ajuda *on-line*
- Interoperabilidade

3.3.1.1. Componentes da ODS Toolbox

Os componentes da ODS Toolbox podem ser subdivididos nas seguintes áreas:

- Interface Editor System – O IES contem um conjunto de editores que podem ser utilizados para definir interactivamente o aspecto e a interacção da aplicação. Os editores incluídos definem a disposição dos objectos, regras, mapas, gráficos, recursos e comunicação.

- Kernel – O *Kernel* inclui funcionalidade completa para controlar objectos de diálogo assim como objectos pré definidos. A disposição e comportamento dinâmico do interface do utilizador (*Controller Working Position*) são determinados pelo *Kernel* e dependem do *script* de diálogo. O *Kernel* também disponibiliza serviços que tratam da distribuição de dados de objectos conceptuais dentro da aplicação.

- Componentes de Serviço – Os componentes de serviço complementam o *Kernel* da seguinte forma:

    - *Recording* & *Replay* grava o *input* do utilizador entre workstations diferentes e repete-o com os eventos originados do exterior como *updates* dos dados do radar e assim reconstruindo o cenário exacto.

    - Objectos de diálogo ATC Toolkit que são construídos no topo de OSF/Motif para cumprir requisitos operacionais.

    - ODS Window Manager adiciona tarefas que não são disponibilizadas por *window managers* convencionais como gestão de prioridade de janelas, protecção de áreas e funções de emergência.

    - Color Server gere a locação de cores utilizadas.

    - *Communication Converters* providenciam uma fácil integração de dados de fontes externas.

    - *ODS Toolbox Models* para assistir o utilizador a desenvolver aplicações com a ODS Toolbox são fornecidos vários elementos pré configurados e amostras de aplicações. Um gerador de dados pode ser utilizado para simular dados de vôo.

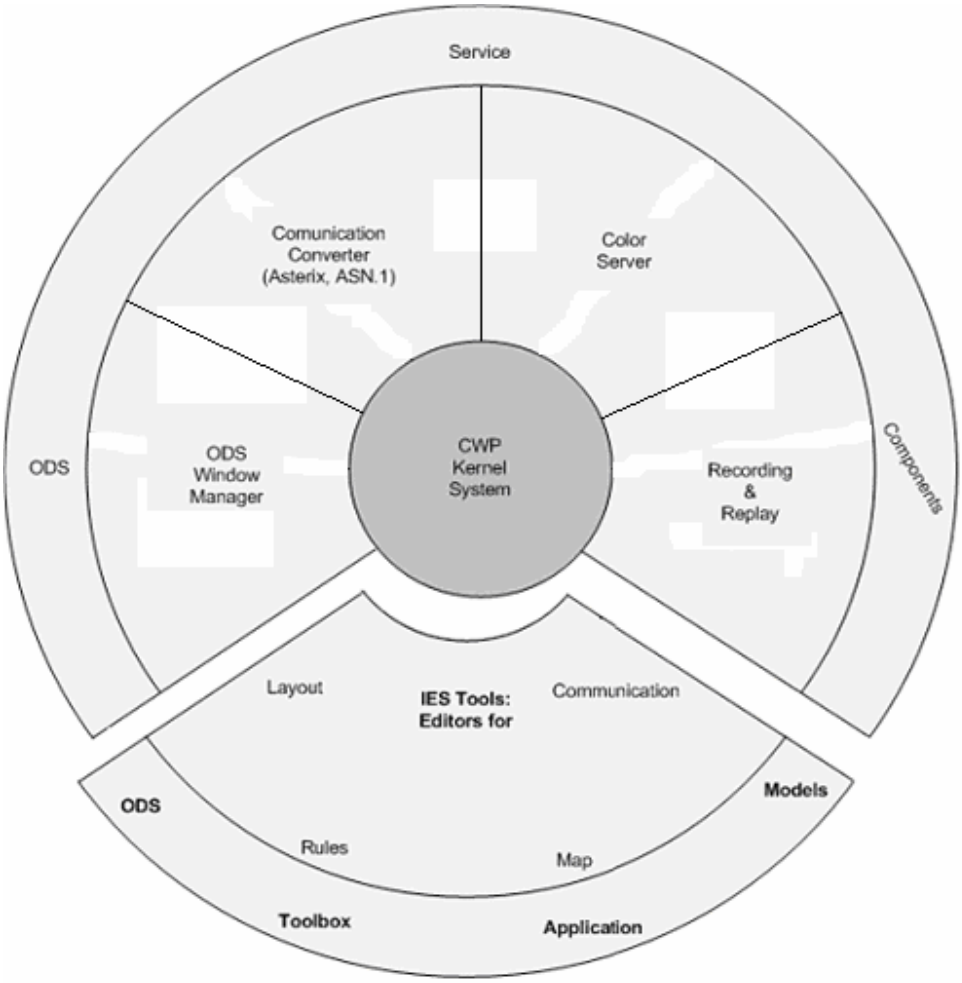


Figura 6. Componentes da ODS Toolbox

3.3.1.1.1. ODS Toolbox Interface Editor System

O IES é utilizado para gerar, adaptar e testar a interface gráfica. É construído no topo de camadas standard de software como o X Window System, o ATC Toolkit e os objectos conceptuais com as suas representações.

Dentro do IES vários editores combinam para tratar de diferentes requisitos. Esses editores facilitam a:

- definição de objectos de diálogo como janelas, botões, *listboxes*, etc.;
- definição de objectos relacionados com ATC;
- definição de mapas;
- definição de dados do *communication converter*;
- edição de regras que descrevem a reacção da interface gráfica ao input do utilizador e eventos do sistema;
- descrição de dependências entre os objectos de representação e os valores guardados nos objectos conceptuais;

3.3.1.1.2. ODS Toolbox Kernel

O ODS Toolbox Kernel executa a configuração definida pelos editores do IES. Consiste em três partes:

- Servidor de objectos – trata da administração e coerência de objectos de dados dentro do ODS Toolbox Kernel. Disponibiliza uma API para manipular objectos do exterior;
- Dialog runtime – trata da criação e comportamento dinâmico da interface do utilizador;
- Componentes pré-construídos – consiste num conjunto extensível de objectos conceptuais, num conjunto de elementos de diálogo interactivos e num conjunto de objectos de representação configuráveis.

3.3.1.1.3. Componentes de Serviço ODS Toolbox

Os componentes de serviço são todos os programas e entidades funcionais que não pertencem explicitamente ao ODS Toolbox Kernel ou que podem substituir um componente do mesmo.

3.3.1.1.3.1. ODS Window Manager

O ODS Window Manager (atcwm) é um *window manager* para o X Window System. Disponibiliza um conjunto de decorações de janela, funções interactivas para manipulação de janelas e funcionalidade estendida como prioridades de janelas e gestão de área protegida, funções de emergência e *interfaces* para a funcionalidade de *Recording & Replay*. A figura 7 ilustra a forma como o window manager interage com os outros componentes do sistema.

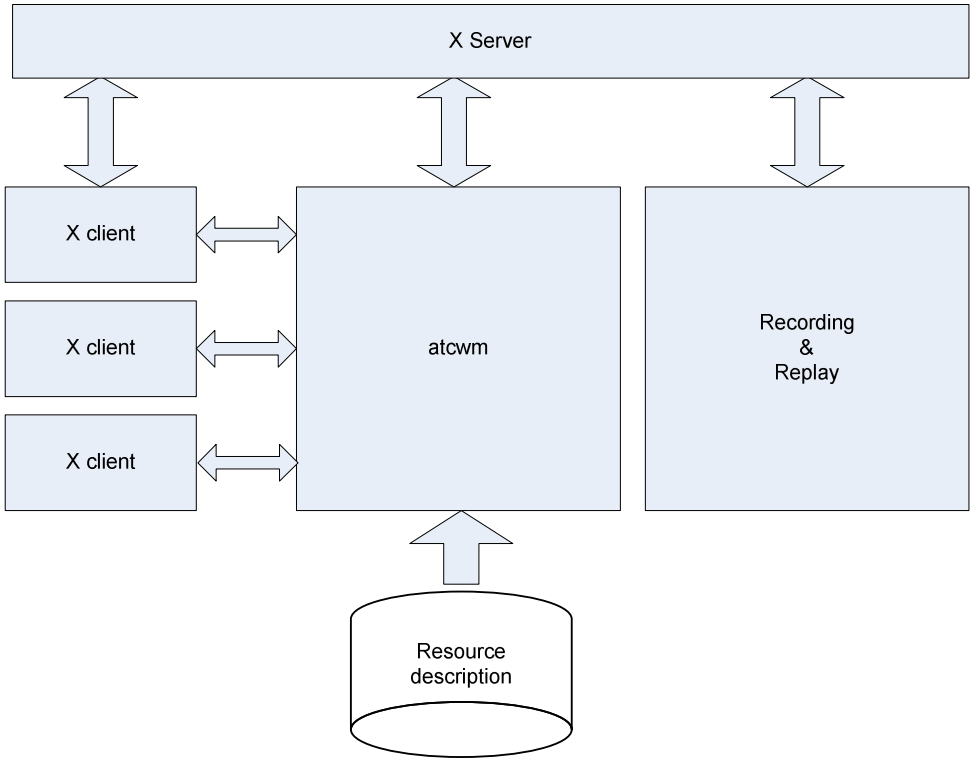


Figura 7. Interfaces do atcwm

3.3.1.1.3.2. Interoperabilidade

O pacote da interoperabilidade permite ao utilizador operar um *display* remoto. Para que seja possível é necessário uma interface de *dialog* e um método de interacção para mudar para o *display* remoto. Este módulo é:

- Independente do hardware;
- Baseado em software standard (X Window System);
- Portável;

3.3.1.1.3.3. ODS Toolbox Recording and Replay

Cada sistema baseado na ODS Toolbox possui uma interface de utilizador flexível que permite ao utilizador abrir várias janelas interactivas no ecrã. Algumas janelas contêm informação passiva, outras são actualizadas por mensagens da LAN, e outras permitem a entrada de novos dados ou manipulação directa dos objectos. Janelas de alarme aparecem sem interacção do utilizador.

Recording and Replay significa que uma determinada situação do ecrã pode ser restaurada e repetida mais tarde. O replay pode ser passivo, todas as acções são refeitas e o *display* reage de acordo, ou pode ser activo, os eventos de utilizador não são repetidos e o sistema processa apenas eventos de rede.

3.3.1.1.3.4. Communication Converters

A troca de dados com fontes externas é concretizada pelos *communication converters*. Dois *communication converters* importantes são o Communication Converter Asterix e o Communication Converter ASN.1.

3.3.1.1.3.4.1. Communication Converter Asterix

O CCX inicializa-se lendo toda a informação fornecida pelo CEX. Para o CCX estes dados são guardados em *Control Objects*, um para cada categoria de ASTERIX. O Communication Converter usa essa informação para ler e descodificar os dados que chegam, converter e guardar o resultado nos atributos de um objecto conceptual.

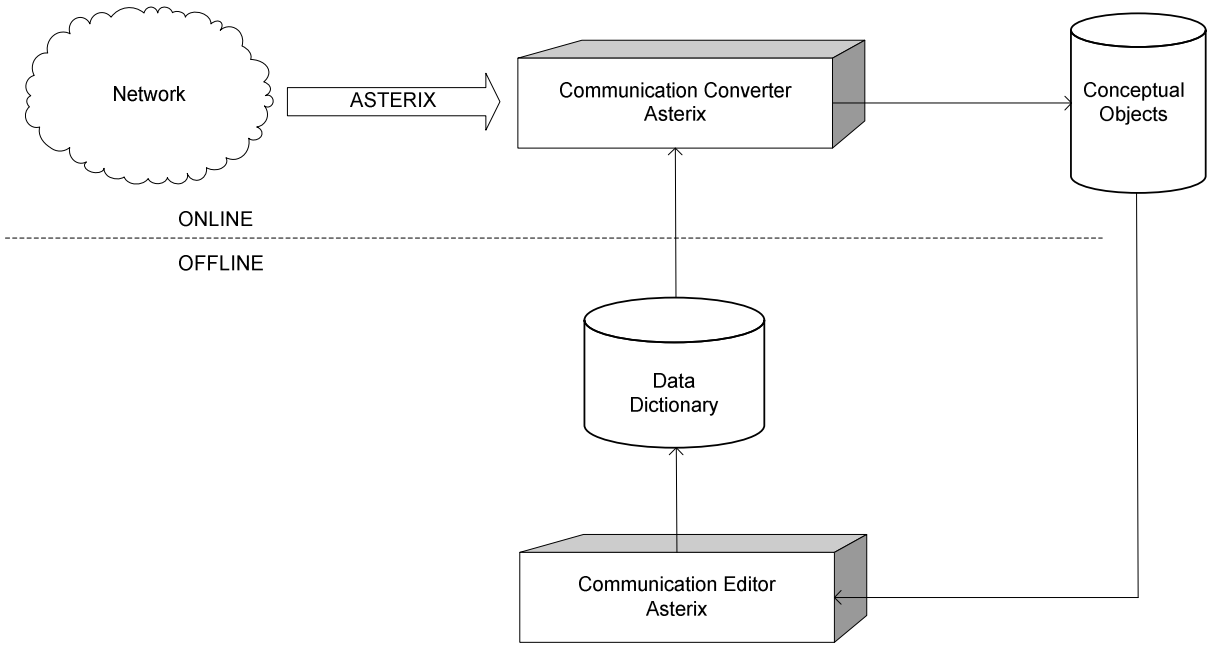


Figura 8. Communication Converter Asterix

3.3.1.1.3.4.2. Communication Converter ASN.1/BER

O próprio Communication Converter é responsável por lidar com a recuperação de erros de dados, como dados codificados inválidos. No entanto não é da sua responsabilidade obter datagramas da LAN. Depende de mecanismos externos não bloqueantes para a obtenção dos dados.

Assim que as mensagens ASN.1 são definidas, o compilador ASN.1 para C gera funções para codificar e descodificar. Porém, para enviar e receber dados, tem de ser fornecida informação de onde os dados estão ou para onde os dados devem ser guardados. Esta informação é denominada o mapeamento de uma mensagem ASN.1.

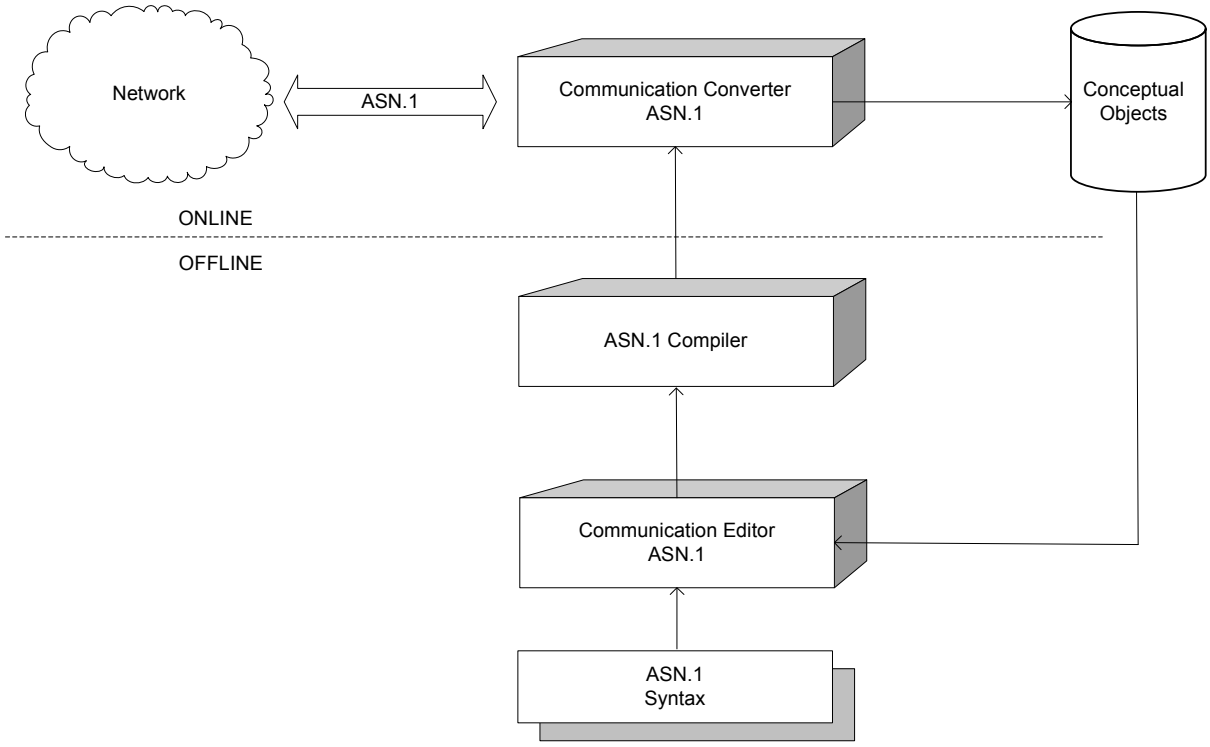


Figura 9. Communication Converter ASN.1/BER

Para cada campo de dados numa mensagem o utilizador tem de identificar o atributo de uma classe de CO.

Um campo de cada mensagem recebida tem de ser marcado como a chave da mensagem. O valor deste campo é utilizado para obter o CO no qual guardar os dados descodificados.

O utilizador pode especificar para cada mensagem a enviar um CO e um evento que força o envio da mensagem. Eventos são a criação, remoção e modificação de objectos.

A definição de mapeamento de informação inclui:

- O nome do atributo sob o qual os dados são guardados
- O nome da classe de CO à qual o atributo pertence. Pode ser fornecido por cada campo da mensagem ou pela mensagem completa.
- Factor de calibração a ser aplicado ao valor obtido da mensagem ASN.1 antes de ser guardado o valor
- Offset de calibração que é adicionado ao valor antes de guardar
- Uma marca de chave num campo particular da mensagem
- Uma marca de recepção que marca a mensagem a ser recebida
- Os nomes de três extensores de regra que são chamados durante a recepção para alterar o procedimento de recepção standard
- Uma marca de envio que marca a mensagem a ser enviada
- Os nomes de três extensores de regra que são chamados durante o envio para alterar o procedimento de envio standard
- O nome de uma classe de objecto conceptual e um evento que força o envio.

3.3.1.1.3.5. Color Server

O Color Server gere a atribuição de cores utilizadas pela ODS Toolbox incluindo o ODS Window Manager. Fá-lo da forma mais eficiente de forma a que não exista *overflow* da tabela de cores em tempo de execução. O objectivo principal do Color Server é centralizar o acesso ao X Window System e respectivo hardware gráfico.

O Color Server fornece a funcionalidade que está interligada ao suporte do hardware gráfico:

- Suporte gráfico para permitir o acesso a vários *buffers* de video através do X Window System
- Piscar é suportado através de software para um piscar suave ou através de hardware
- Cores transparentes

O Color Server está acessível através de uma API para que possa ser utilizado por aplicações que não baseadas na ODS Toolbox.

3.3.2. LOKI

Loki é uma biblioteca de design desenvolvida em C++ que inclui implementações de padrões de desenho comuns. O meu contacto com esta biblioteca resume-se à alteração de código para ser aceite pelo compilador sem adulterar o resultado desejado.

3.3.3. Cppunit 1.10.2

CPPunit é uma *framework* de testes unitários para C++. O resultado dos testes pode ser em formato XML, ou em texto para testes automáticos ou em modo gráfico para testes acompanhados. Será utilizado nos testes de validação.

3.3.4. Snacc 1.3

Snacc é uma ferramenta que gera funções em C++ para codificar BER, descodificar e imprimir dada uma fonte em ASN.1. Em C, para além das funcionalidades descritas para C++, é possível libertar recursos. Esta ferramenta também sofreu alterações no código para ser aceite pelo compilador.

3.3.5. Glib 1.2.10

Ferramenta em C utilizada pela aplicação ODS na manipulação de datas. Foi também alterada para ser aceite pelo compilador.

3.3.6. MTL 2.1.2-21

Biblioteca que fornece funcionalidade de álgebra linear para diversos formatos de matrizes. Foi também alterada para ser aceite pelo compilador.

3.3.7. ACE 5.5

Framework em C++ que implementa padrões para comunicação concorrente como intercomunicação entre processos ou gestão de *threads.*

3.3.8. CVS 1.11.20

CVS é o sistema de controlo de versões utilizado pelo projecto.

3.3.9. EXPECT 5.40

Extensão tcl utilizada para simplificar a interacção entre programa e *script*. É utilizado pela aplicação para testes.

3.3.10. TCL 8.4.9

Tcl é uma linguagem de programação dinâmica que permite entre outros, o desenvolvimento de aplicações, testes e administração. No projecto em causa é utilizado principalmente para testes.

3.3.11. TK 8.4.9

Toolkit que permite a construção de interfaces gráficas.

3.3.12. TCPDUMP 3.9.4

Tcpdump é um *packet sniffer* utilizado para interceptar pacotes enviados ou recebidos que passam na rede.

3.3.13. GDB 6.0

Debugger que permite a monitorização e manipulação da aplicação em *runtime.*

3.3.14. GCC 3.4.2

A aplicação ODS originalmente desenvolvida fazia uso do compilador GCC 3.3.2. A ODS Toolbox é clara nas plataformas que a suportam e está definido que o SO Linux Fedora Core 3 suporta a toolbox com o compilador GCC 3.4.2. Uma das minhas tarefas foi alterar o código da aplicação ODS para ser aceite pelo novo compilador.

3.3.15. M4 1.4.2

M4 é um macroprocessador utilizado para o pré processamento de ficheiros em C. Permite a substituição de uma *string* por outra, aritmética, inclusão de ficheiros e expansão condicional.

3.3.16. RSYNC 2.6.8

O Rsync permite a transferência de ficheiros de forma incremental. É utilizado para obter os mapas que o ODS necessita.

3.3.17. PERL 5.8.7

Perl é uma linguagem optimizada para a manipulação de ficheiros de texto e obtenção de relatórios sobre a informação que analizou.

3.3.18. MAKE 3.80

Ferramenta que permite construir aplicações de forma automática.

3.3.19. GAWK 3.1.4

Ferramenta que permite detectar o aparecimento de determinado padrão em ficheiros de texto, alterá-los ou extrair dados.

3.3.20. Bugzilla

Ferramenta desenvolvida em Perl para fazer *tracking* das alterações realizadas no código.

4. Trabalho Realizado

4.1.Integração

O primeiro dia foi preenchido a conhecer pessoas de várias secções desde outros membros da Divisão SISPRO, na qual fui inserido, até pessoas encarregadas da manutenção das máquinas. Foi-me designada uma área de trabalho e os computadores atribuídos.

Os dias seguintes foram ocupados na instalação do sistema operativo (Linux Fedora Core 3), na configuração do mesmo e também na instalação de ferramentas no Windows XP.

Também nestes dias foi-me dada formação por parte do departamento de qualidade (SISQUA). Esta formação visa garantir que o desenvolvimento de sistemas segue determinadas normas, tanto ao nível do desenvolvimento do sistema em si, como ao nível da documentação associada ao mesmo.

A norma ISO 9001, na qual a NAV Portugal é certificada, é o referencial que estabelece os requisitos a serem cumpridos por uma organização com vista a implementar, manter e melhorar um Sistema de Gestão da Qualidade.

Com esses objectivos em mente, as actividades a realizar devem respeitar os procedimentos da Qualidade já definidos para os vários departamentos. No caso concreto da Divisão SISPRO as actividades em questão estão incluídas no processo designado por POP20, sobre o qual também recebi formação.

Em relação à ODS Toolbox a formação teve por base o estudo dos próprios manuais da aplicação e o apoio dos colegas para duvidas.

4.2. Documentação

Uma vez definido o sistema operativo passei para a fase de documentação. A empresa tem definidos os passos necessários para o desenvolvimento de sistemas, como mostra o procedimento operacional em vigor na NAV Portugal ilustrado na figura 10.

Restrições técnicas não me permitiram seguir à risca o procedimento operacional, de forma que não foi possível realizar todos os documentos indicados no procedimento.

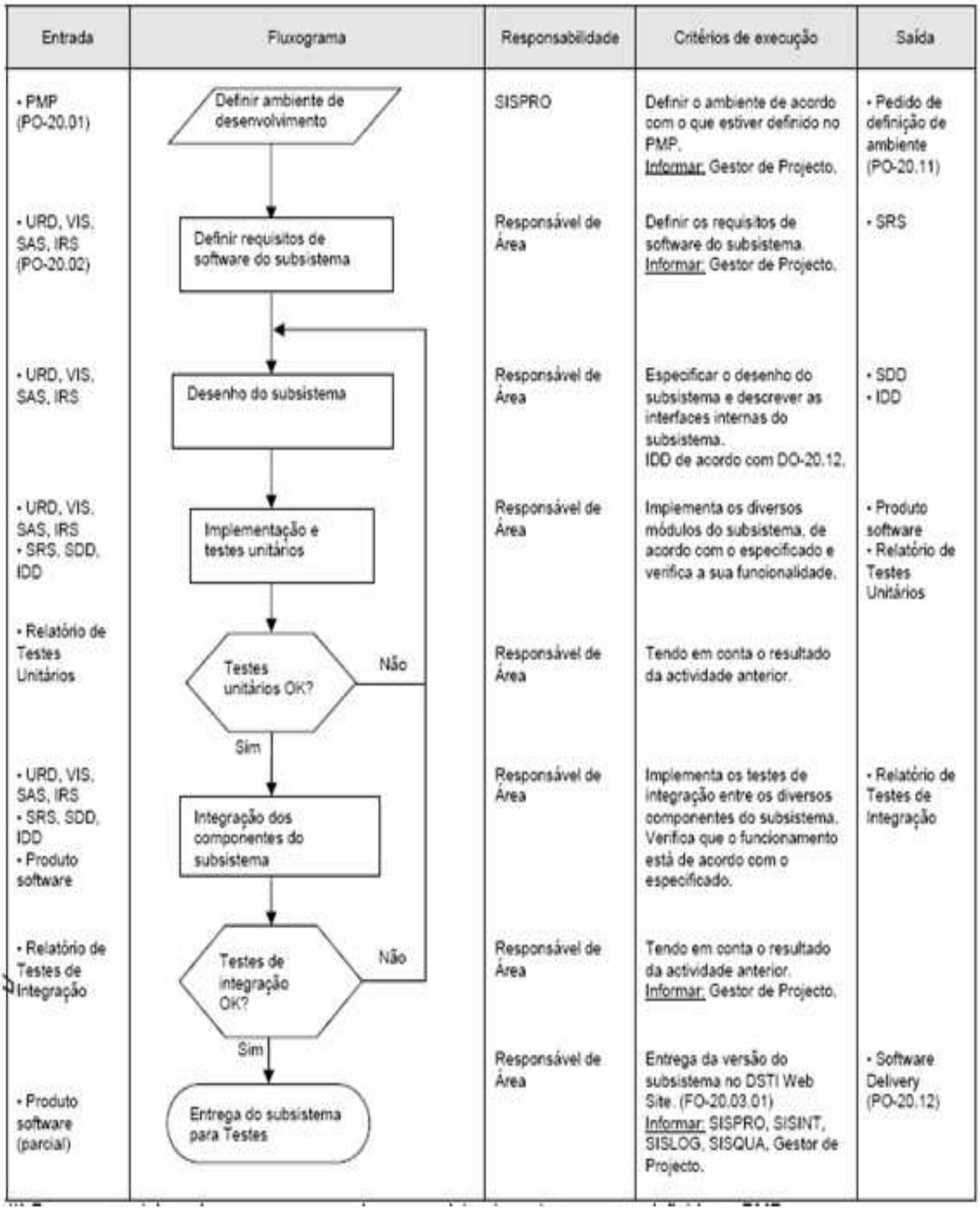


Figura 10. Procedimento operacional de desenvolvimento de sistemas em vigor na NAV

Na figura 10 é possível observar o procedimento operacional de desenvolvimento de sistemas em vigor na NAV Portugal. O procedimento especificado no fluxograma define os documentos que devem entrar em cada passo do desenvolvimento, assim como os documentos que esse passo deve gerar. Todos os documentos produzidos ao longo do processo estão sujeitos a aprovação em revisão formal.

A empresa disponibiliza *templates* para os vários tipos de documentos que devem ser utilizados e seguidos rigorosamente. Esses *templates* estão em rede para fácil acesso.

Os documentos são identificados univocamente com o número a que corresponde o projecto (por exemplo o ODS tem a identificação 9011), a sigla que designa o tipo de documento e um número de parte. É também incluída uma data de alteração e versão actual. Os documentos possuem informação adicional sobre quem é responsável pelas versões anteriores e actual bem como as secções alteradas e datas de criação. Finalmente todos os documentos sujeitos a aprovação formal possuem uma tabela que designa as entidades responsáveis e o seu representante. Este representante tem a função de rever o documento, deve submeter possíveis alterações ao responsável do documento e alertar para a necessidade de uma reunião caso assim o entenda.

4.2.1. Ponto de Situação

Para facilidade o controlo e monitorização das actividades em curso está instituído no SISPRO a realização de um ponto de situação mensal que ilustra as tarefas concluídas, activas e previstas para o mês seguinte, o tempo dedicado a cada tarefa e a data inicialmente prevista para a sua conclusão e a previsão actual. Cada tarefa tem ainda discriminado o seu estado.

O Ponto de Situação inclui também riscos para um eventual atraso na conclusão das tarefas e possível mitigação desse risco.

Os formulários em vigor são alvo de frequentes melhorias, e durante o estágio, o documento Ponto de Situação sofreu alterações.

4.2.2. Interface Requirements Specification

O IRS foi o primeiro documento que realizei para o projecto. Ainda sem conhecimentos ao nível do sistema preparei-me revendo documentos já criados sobre a aplicação ODS e exemplos de outros documentos IRS.

O documento foi criado segundo o *template* em vigor e foi determinado que o realizasse em inglês.

Quando iniciei o projecto existia já uma primeira versão do documento, a qual desenvolvi para reflectir uma correcta descrição das interfaces com o ODS com o auxilio de colegas de trabalho. Foram-me dados conhecimentos, em primeiro lugar, dos diversos componentes envolvidos e de seguida ao nível da sua comunicação, nomeadamente como e quem inicia a troca de mensagens e que informação é incluída nessas mensagens.

O documento descreve cada campo de cada mensagem enviada/recebida. Essa descrição inclui o tipo de variável, o tamanho, e o significado do campo no contexto em que é enviada/recebida. Para poder realizar esta tarefa foi-me fornecida a gramática que gera todas as mensagens envolvidas.

4.3. Adaptação de código fonte e makefiles

Terminado o documento IRS passei para a fase de adaptação de código fonte e de *makefiles*. As alterações nas *makefiles* passaram por corrigir as *paths* para algumas ferramentas e para a invocação de comandos.

As alterações ao código são obrigatoriamente registadas no Bugzilla para existir um *tracking* das mesmas.

Ao nível da compilação várias ferramentas tiveram de ser alteradas por não seguirem os standards actuais.

As linguagens com que trabalhei nesta fase foram o C e C++. A alteração de compilador foi a causa dos problemas encontrados nesta fase dado que o código quer da aplicação, quer de algumas ferramentas utilizadas não seguiam o standard aceite pelo GCC 3.4.2.

O estudo inicial sobre os sistemas operativos preparou-me para os outros problemas que encontrei nesta fase. Os mais simples de resolver resumiram-se a retirar *headers* que não eram necessárias quando o compilador indicava que não conseguia encontrar o ficheiro. A um nível de complexidade acima, o compilador não reconhecia determinada função. Neste problema o indicado, pela preparação que fiz, é encontrar onde se encontra esta função em Linux. Esta situação ocorreu um numero considerável de vezes. Numa fase inicial em que o código ainda me era bastante desconhecido levou a alguns erros da minha parte. Ao apagar código que já não era utilizado também acabei por apagar código que era de facto utilizado pela aplicação. Não seria um problema se o compilador indicasse erros imediatamente, mas como só o fez mais tarde após a adaptação de outras partes do código levou-me a pensar que se tratava de funções que fazem parte de Solaris e que teria de as realizar em Linux. O erro foi corrigido repondo o código removido. Esta situação de métodos exclusivos de Solaris nunca se verificou de forma que não precisei de concretizar funções completas.

A principal preocupação nesta fase foi certificar-me que o código era aceite pelo compilador. Modificações mais profundas no código só iria realizar se a aplicação revelasse um comportamento diferente do esperado. Comportamento que seria revelado pelos testes o que me permitiria localizar o problema.

Esta foi a primeira vez que entrei em contacto com a linguagem C++ o que levou a alguns problemas nomeadamente quanto à sintaxe. O GCC 3.4.2. tem algumas diferenças em relação à sintaxe aceite pelo GCC 3.3.2. e o erro que indica muitas vezes não é descritivo do problema real.

A aplicação ODS é constantemente alterada de forma a melhorar o serviço que disponibiliza. Acontece, por vezes, partes de código deixarem de ser utilizadas mas continuarem na aplicação. À medida que me deparava com métodos/funções já não utilizados aproveitei para fazer uma limpeza do código.

Segue-se uma tabela com exemplos de alterações que foi necessário efectuar. Não pretende ser uma tabela completa dessas alterações, apenas pretende ilustrar sob um ponto de vista prático alguns exemplos diversificados. Não foram incluídas alterações sobre o código da ODS Toolbox dado que essas ainda estão sujeitas a alterações.

SOLARIS	LINUX	Descrição
Gcc –o $@ -I. –Ilib $(VNC_objs) –L/usr/ucblib	Gcc –o $@ -I. –Ilib $(VNC_objs)	A biblioteca utilizada em Solaris não é necessária em Linux
strStkG	strStkG_my	A existência de bibliotecas em C e C++ que utilizam variáveis com o mesmo nome levou à alteração de uma dessas variáveis
#include <sys/filio.h>	#include <sys/iotcl.h>	Alteração do nome da *header*
	using namespace std;	Indica ao compilador para tratar os nomes na biblioteca standard como se estivessem definidos no próprio programa. A alteração de compilador levou ao aparecimento deste erro
register len	register int len	É necessário especificar o tipo.
#include <sys/systeminfo.h>		Removido. Não necessário.
#include <sys/siginfo.h>	#include <usr/include/bits/siginfo.h>	Localização diferente
const size_t Command3Test<Fact>::bufferSize=28	template<> const size_t Command3Test<Fact>::bufferSize=28	O novo compilador é mais rigoroso na sintaxe exigida. Necessitou de uma especificação parcial para aceitar a instrução.
Makefile	Makefile	Correcção das paths para as várias ferramentas, flags especificas para a nova versão, opções para o SO Linux
	LANG=en_US.ISO885915	Variável de ambiente que vai ser utilizada pela aplicação para determinar a linguagem. Na aplicação em questão corrigiu o problema de o sistema não encontrar determinados conjuntos de caracteres
	Section "Files" FontPath "/usr/lib/X11/fonts/100dpi"	No ficheiro xorg.conf foi necessário incluir *path*s para a localização de ficheiros de fontes. Várias localizações foram incluídas sendo a descrita à esquerda uma delas

Tabela 2. Exemplos de alterações ao código

4.4. Testes Unitários

Terminada a adaptação do código fonte passei para a realização dos testes unitários. Com a conclusão da fase anterior o código é aceite pelo compilador mas não é possível determinar como se irá comportar em Fedora Core 3.

Numa primeira fase o objectivo é corrigir as discrepâncias visuais entre a aplicação original e a aplicação portada. É um aspecto importante dado que um dos requisitos da portagem é de que a aplicação deve ter exactamente o mesmo aspecto gráfico. Problemas com as fontes e com o Window Manager que é responsável pelas janelas no ambiente foram o 1º desafio. Alterações nos ficheiros de configuração X Server para *paths* de fontes e alteração de variáveis de ambiente resolveram parte dos problemas. Para fontes externas ao sistema operativo tem de ser criado um índice na directoria onde se encontram utilizando o comando *mkfontdir*. Nesta parte foi útil a utilização do comando *xfontsel* que me permitiu certificar se as fontes pretendidas eram ou não reconhecidas pelo SO.

Outros problemas foram resolvidos com a configuração de um novo Window Manager, mais concretamente, o ATCWM da ODS Toolbox. Nesta fase foi-me disponibilizado uma máquina na sala de testes para poder visualizar em pormenor o aspecto gráfico e comportamento da aplicação ODS original.

O Color Server necessitou de alterações à configuração inicial do X Server. Para poder atribuir as cores correctamente precisa de ter o *display* configurado para *pseudocolor*s. Por defeito é utilizado *truecolor*. Uma cor *pseudocolor* é uma cor representada por um valor denominado *pixel value* (valor do pixel). A cor representada no ecrã pode ser alterada dinamicamente alterando a sua definição na tabela de cores (color lookup table). Uma cor *truecolor* é representada por um número que é a cor em vez de um índice na tabela de cores. O número tem normalmente 24 bits utilizando 8 bits para cada valor de vermelho, verde e azul. Para suportar *pseudocolors* o X Server é configurado para utilizar 256 cores ou é lançado com a opção *–depth 8* no próprio terminal.

4.4.1. Testes de Funcionalidade

Já na preparação para estes testes adaptei um *script*, originalmente desenvolvido para ser executado num ambiente de Windows para Linux. Este *script* tem o objectivo de injectar dados directamente na aplicação. Os dados inseridos referem-se a todas as interfaces externas da aplicação. Injectando os dados desta forma é possível testar a aplicação ODS de forma isolada dessas interfaces.

4.5.Testes de Integração

Até à data de entrega da dissertação não foi possível realizar os testes de integração. Estes testes são da responsabilidade do SISINT e acompanhados pelo SISPRO.

4.6. Testes de Validação

Os testes de validação também não foram realizados. Estes testes utilizam a *framework cppunit*.

4.7. Principais dificuldades

As principais dificuldades encontradas na fase de documentação do projecto surgiram principalmente pelo desconhecimento do sistema do qual a aplicação faz parte.

Na fase de adaptação do código fonte as dificuldades encontradas deveram-se ao desconhecimento do próprio código. Adaptar código de outras pessoas sem ter conhecimento do papel de cada função, variável ou algoritmo, especialmente num código tão extenso, levou a um cuidado acrescido na portagem.

Grande parte da aplicação está em C++. Este foi o primeiro contacto que tive com a linguagem e alguns problemas levaram a que perdesse mais tempo com problemas específicos de sintaxe do que inicialmente previsto.

Passada a fase de adaptação do código encontrei problemas ligados à ODS Toolbox. Também o primeiro contacto com esta ferramenta levantou algumas dificuldades tanto ao nível da configuração do sistema como do funcionamento da ferramenta. Os problemas da configuração de sistema levaram a que tivesse de alterar o ambiente do X Server especificamente variáveis de ambiente, *paths* para fontes e configuração do Window Manager da ODS Toolbox. Ao nível do funcionamento da ferramenta os problemas encontrados devem-se à falta de prática da utilização da mesma.

De assinalar que o co-orientador mostrou sempre total disponibilidade no acompanhamento do projecto, no auxilio na resolução de problemas ou a indicar a direcção correcta sempre que necessário.

4.8. Trabalho Futuro

O projecto não foi concluído até à data de entrega da dissertação. Está na fase de testes unitários o que significa que falta realizar a restante documentação (TMP, ATD, ATR) e os testes de integração e validação.

4.9.Planeamento Previsto

O planeamento previsto (Anexo 1) foi elaborado pelo co-orientador, Eng. José Vermelhudo. Contém as diferentes fases que inicialmente se pensava que iria realizar assim como a previsão do tempo que iria necessitar.

4.10. Planeamento Cumprido

O planeamento cumprido (Anexo 2) difere consideravelmente do planeamento previsto.

Tarefas como a elaboração do documento SRS não foram realizadas. Quando a tarefa foi aprofundada apercebeu-se que não existiam todos os requisitos para a elaboração do documento. Para que o documento pudesse ser elaborado era necessário um mapeamento de requisitos de sistema (expresso por exemplo com Functional Requirements Specification) para o subsistema ODS.

Por outro lado também se percebeu que o documento não era absolutamente necessário para a continuação do projecto. No entanto enquanto não se chegou a esta conclusão utilizei o tempo a preparar-me para o realizar lendo documentação sobre o sistema LISATM e outros SRS relacionados.

O tempo indicado na elaboração do IRS não é indicativo do tempo real na realização do documento. Indica principalmente o tempo que demorou deste o inicio do documento até ao IRS definitivo. Após a aprovação do documento, foram feitas melhorias sugeridas pelos revisores. A data de conclusão da tarefa reflecte isso, mais do que o tempo dedicado. No mesmo período, paralelamente, a tarefa de adaptação do código fonte prosseguia.

Ainda neste período de tempo o projecto sofreu outro atraso importante. Tive uma baixa prolongada por motivos de saúde que atrasou a conclusão e aprovação do IRS, adaptação do código e fez deslizar todas as datas previstas.

A adaptação do código fonte sofreu também atrasos devido à inexperiência de trabalhar com C++, nomeadamente ao nível da sintaxe. Foi também necessário fazer com que ferramentas auxiliares do ODS fossem aceites pelo novo compilador, tendo algumas demorado mais do que seria de prever. Por fim migrar código feito por outras pessoas, o que leva ao desconhecimento inicial do significado dos métodos, funções e variáveis presentes dificulta a tarefa. Foi necessário algum tempo para ambientação ao código.

Quanto aos testes unitários, a inexperiência na utilização da ODS Toolbox levou a atrasos. O ODS Window Manager e o Color Server deveriam ter sido utilizados desde o inicio. A configuração do X Server para a correcta atribuição de cores por parte do Color Server consumiu mais tempo do que se poderia prever.

5. Conhecimentos Adquiridos

Os conhecimentos adquiridos podem ser enumerados nos seguintes pontos:

- Diferenças e semelhanças entre os sistemas operativos ao nível do desenvolvimento de aplicações
- Formação recebida a nível de qualidade para assegurar planeamento, desenvolvimento e entrega de acordo com os standards da empresa
- Elaboração de documentos de acordo com o exigido pela NAV Portugal EPE
- Introdução a novas linguagens (C++), toolkits, ferramentas, e comandos do sistema operativo
- Configuração do X Server
- Experiência adquirida tanto a nível de programação como a nível de inclusão no ambiente empresarial
- Conceitos e procedimentos relacionados com o controlo de tráfego aéreo

6. Conclusão

Este documento descreveu o desenrolar do estágio na NAV Portugal EPE no projecto “Portagem da aplicação ODS de Solaris 8 para Linux Fedora Core 3”. Descreve a aplicação e como se integra no sistema do qual faz parte, e as ferramentas que utiliza para a sua concretização, com especial atenção para a ODS Toolbox.

O projecto não foi concluído o que dificulta tirar conclusões sobre a portagem em si. No entanto é possível concluir que preparar a portagem informando-se sobre as tecnologias utilizadas, nomeadamente ao nível das bibliotecas utilizadas, permite uma resolução mais imediata dos problemas que se seguirão. É natural que surjam problemas numa actividade deste género pelo que se torna importante testar a aplicação em todos os cenários que seja possível fazê-lo.

Os projectos estão sujeitos a atrasos pelo que se torna importante uma identificação dos potenciais riscos e possíveis mitigações dos mesmos para a redução desses atrasos.

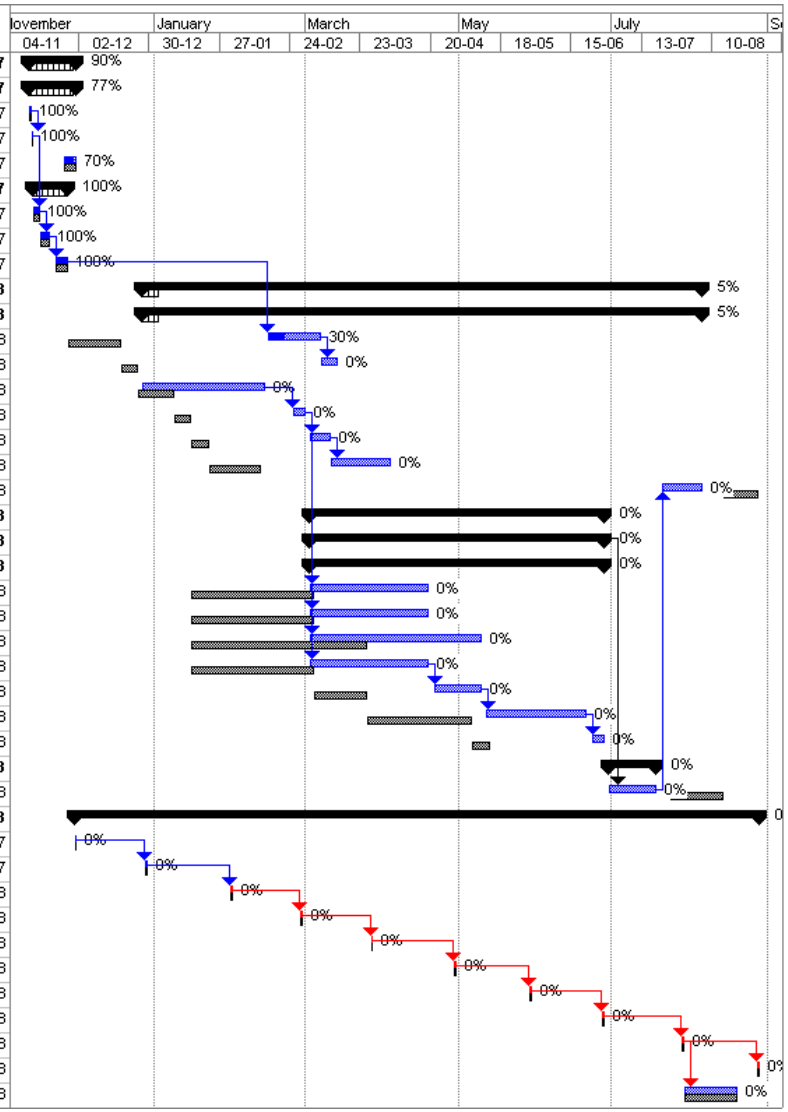
A faculdade fornece as bases necessárias para o desenvolvimento de projectos. No entanto são apenas as bases. Neste ramo é necessário um constante acompanhamento sobre as mais diversas vertentes deste ramo como novas linguagens, ferramentas, sistemas operativos, metodologias de trabalho, etc.

A nível empresarial, a NAV Portugal EPE proporcionou uma boa entrada no mundo de trabalho integrando-me num projecto de considerável complexidade pelo sistema do qual a aplicação faz parte e pelos conceitos que engloba.

7. Anexos

Anexo 1. Planeamento previsto

ID		Task Name	Duration	Start	Finish
1		**1ª Fase Estágio**	**15 days**	**Mon 12-11-07**	**Fri 30-11-07**
2		**Formação**	**15 days**	**Mon 12-11-07**	**Fri 30-11-07**
3	✓	Introdução Controlo Trafego Aereo	1 day	Mon 12-11-07	Mon 12-11-07
4	✓	Formação Qualidade - POP 20	0,5 days	Tue 13-11-07	Tue 13-11-07
5		Formação técnica ODS	5 days	Mon 26-11-07	Fri 30-11-07
6	✓	**Ambiente Trabalho**	**10 days**	**Tue 13-11-07**	**Tue 27-11-07**
7	✓	Preparar Computador	3 days	Tue 13-11-07	Fri 16-11-07
8	✓	Instalar Linux - RedHat Entreprise 4	2 days	Fri 16-11-07	Tue 20-11-07
9	✓	Instalar Ferramentas	3 days	Thu 22-11-07	Tue 27-11-07
10		**2ª Fase Estágio**	**159 days**	**Thu 27-12-07**	**Tue 05-08-08**
11		**Documentação**	**159 days**	**Thu 27-12-07**	**Tue 05-08-08**
12		Doc Esp. Software - SRS	15 days	Fri 15-02-08	Thu 06-03-08
13		Doc SRS - Approval	5 days	Fri 07-03-08	Thu 13-03-08
14		Doc Interface - IRS	35 days	Thu 27-12-07	Wed 13-02-08
15		DOC IRS - Approval	5 days	Mon 25-02-08	Fri 29-02-08
16		Doc Planeamento Teste - TMP	6 days	Mon 03-03-08	Mon 10-03-08
17		Doc Especificação Teste - ATD	18 days	Tue 11-03-08	Thu 03-04-08
18		Doc Resultado de Teste - ATR	12 days	Mon 21-07-08	Tue 05-08-08
19		**3ª Fase Estágio**	**85 days**	**Mon 03-03-08**	**Fri 27-06-08**
20		**Desenvolvimento Software**	**85 days**	**Mon 03-03-08**	**Fri 27-06-08**
21		**Preparação package em Linux**	**85 days**	**Mon 03-03-08**	**Fri 27-06-08**
22		Adaptação de código fonte	35 days	Mon 03-03-08	Fri 18-04-08
23		Testar bibliotecas (open source)	35 days	Mon 03-03-08	Fri 18-04-08
24		Adaptação da geração do sistema (make files)	50 days	Mon 03-03-08	Fri 09-05-08
25		Adaptação da instalação do sistema	35 days	Mon 03-03-08	Fri 18-04-08
26		Testes Unitários	15 days	Mon 21-04-08	Fri 09-05-08
27		Testes Integração	30 days	Mon 12-05-08	Fri 20-06-08
28		Entrega de ODS em Linux	5 days	Mon 23-06-08	Fri 27-06-08
29		**4ª Fase Estágio**	**15 days**	**Mon 30-06-08**	**Fri 18-07-08**
30		Testes Validação	15 days	Mon 30-06-08	Fri 18-07-08
31		**Realização de relatórios de Ponto de Situação**	**195 days**	**Fri 30-11-07**	**Thu 28-08-08**
32		RPS 1.0	1 day	Fri 30-11-07	Fri 30-11-07
33		RPS 2.0	1 day	Fri 28-12-07	Fri 28-12-07
34		RPS 3.0	1 day	Thu 31-01-08	Thu 31-01-08
35		RPS 4.0	1 day	Thu 28-02-08	Thu 28-02-08
36		RPS 5.0	1 day	Thu 27-03-08	Thu 27-03-08
37		RPS 6.0	1 day	Tue 29-04-08	Tue 29-04-08
38		RPS 7.0	1 day	Thu 29-05-08	Thu 29-05-08
39		RPS 8.0	1 day	Fri 27-06-08	Fri 27-06-08
40		RPS 9.0	1 day	Tue 29-07-08	Tue 29-07-08
41		RPS 10.0	1 day	Thu 28-08-08	Thu 28-08-08
42		Realização de relatório Final	15 days	Wed 30-07-08	Tue 19-08-08



Anexo 2. Planeamento cumprido

ID		Task Name	Duration	Start	Finish
1		**1ª Fase Estágio**	**27 days**	**Mon 12-11-07**	**Tue 18-12-07**
2		**Formação**	**27 days**	**Mon 12-11-07**	**Tue 18-12-07**
3	✓	Introdução Controlo Trafego Aereo	1 day	Mon 12-11-07	Mon 12-11-07
4	✓	Formação Qualidade - POP 20	1 day	Tue 13-11-07	Tue 13-11-07
5		Formação técnica ODS	17 days	Mon 26-11-07	Tue 18-12-07
6	✓	**Ambiente Trabalho**	**10 days**	**Tue 13-11-07**	**Tue 27-11-07**
7	✓	Preparar Computador	3 days	Tue 13-11-07	Fri 16-11-07
8	✓	Instalar Linux - Fedora Core 3	3 days	Fri 16-11-07	Wed 21-11-07
9	✓	Instalar Ferramentas	3 days	Thu 22-11-07	Tue 27-11-07
10		**2ª Fase Estágio**	**251 days**	**Wed 12-12-07**	**Wed 26-11-08**
11		**Documentação**	**251 days**	**Wed 12-12-07**	**Wed 26-11-08**
12		Doc Interface - IRS	142 days	Wed 12-12-07	Thu 26-06-08
13		DOC IRS - Approval	40 days	Fri 27-06-08	Thu 21-08-08
14		Doc Planeamento Teste - TMP	5 days	Tue 30-09-08	Mon 06-10-08
15		Doc Especificação Teste - ATD	10 days	Tue 07-10-08	Mon 20-10-08
16		Doc Resultado de Teste - ATR	5 days	Thu 20-11-08	Wed 26-11-08
17		**3ª Fase Estágio**	**171 days**	**Wed 19-03-08**	**Wed 12-11-08**
18		**Desenvolvimento Software**	**171 days**	**Wed 19-03-08**	**Wed 12-11-08**
19		**Preparação package em Linux**	**171 days**	**Wed 19-03-08**	**Wed 12-11-08**
20		Adaptação de código fonte	85 days	Wed 19-03-08	Tue 15-07-08
21		Testar bibliotecas (open source)	85 days	Wed 19-03-08	Tue 15-07-08
22		Adaptação da geração do sistema (make files)	85 days	Wed 26-03-08	Tue 22-07-08
23		Adaptação da instalação do sistema	85 days	Wed 26-03-08	Tue 22-07-08
24		Testes Unitários	30 days	Mon 25-08-08	Fri 03-10-08
25		Testes Integração	18 days	Mon 06-10-08	Wed 29-10-08
26		Entrega de ODS em Linux	10 days	Thu 30-10-08	Wed 12-11-08
27		**4ª Fase Estágio**	**5 days**	**Thu 13-11-08**	**Wed 19-11-08**
28		Testes Validação	5 days	Thu 13-11-08	Wed 19-11-08
29		**Realização de relatórios de Ponto de Situação**	**260 days**	**Fri 30-11-07**	**Thu 27-11-08**
30		RPS 1.0	1 day	Fri 30-11-07	Fri 30-11-07
31		RPS 2.0	1 day	Fri 28-12-07	Fri 28-12-07
32		RPS 3.0	1 day	Thu 31-01-08	Thu 31-01-08
33		RPS 4.0	1 day	Thu 28-02-08	Thu 28-02-08
34		RPS 5.0	1 day	Thu 27-03-08	Thu 27-03-08
35		RPS 6.0	1 day	Tue 29-04-08	Tue 29-04-08
36		RPS 7.0	1 day	Thu 29-05-08	Thu 29-05-08
37		RPS 8.0	1 day	Fri 27-06-08	Fri 27-06-08
38		RPS 9.0	1 day	Tue 29-07-08	Tue 29-07-08
39		RPS 10.0	1 day	Thu 28-08-08	Thu 28-08-08
40		RPS 11.0	1 day	Tue 30-09-08	Tue 30-09-08
41		RPS 12.0	1 day	Fri 31-10-08	Fri 31-10-08
42		RPS 13.0	1 day	Thu 27-11-08	Thu 27-11-08
43		Realização de relatório Final	15 days	Mon 08-09-08	Fri 26-09-08