Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

APSE - APPLICATION PROGRAMMER SUPPORT ENVIRONMENT

Fernando Ricardo dos Santos Real

Mestrado em Engenharia Informática

2007

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

APSE - APPLICATION PROGRAMMER SUPPORT ENVIRONMENT

Fernando Ricardo dos Santos Real

Projecto orientado pela Prof. Dra. Maria da Graça Figueiredo Rodrigues Gaspar
e co-orientado pelo Eng. José Freitas

Mestrado em Engenharia Informática

2007

Declaração

Fernando Real, aluno nº 26526 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 "APSE - Application Programmer Support Environment", 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, 22 de Outubro de 2007

José Freitas, supervisor do projecto de *Fernando Real*, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado "APSE - Application Programmer Support Environment".

Lisboa, 22 de Outubro de 2007

Resumo

A componente de *software* tem assumido uma crescente importância no que diz respeito a assegurar o correcto funcionamento das aeronaves, bem como o respeitar dos procedimentos e o colmatar dos erros humanos. Este relatório apresenta o trabalho de desenvolvimento de uma ferramenta computacional destinada sobretudo a colmatar os erros humanos no pilotar de aeronaves, a ferramenta APSE - *Application Programmer Support Environment*.

O APSE é uma aplicação que serve de apoio a outra aplicação, o ICS (*Intelligent Crew Support*), e o seu objectivo é o auxílio à programação desta última. O ICS é uma componente de *software* desenvolvida com o objectivo de monitorizar as acções de uma tripulação e aconselhar a tripulação consoante a situação corrente. Contudo, o ICS necessita ser programado específicamente e é o APSE que presta esse auxílio. Ambas as aplicações foram desenvolvidas no âmbito de um projecto maior, fazendo parte integrante do mesmo: o projecto FLYSAFE. No entanto, a aplicação ICS foi desenvolvida por um parceiro de outro país.

O trabalho desenvolvido cobriu a totalidade do processo habitual de desenvolvimento de *software*, incluindo nomeadamente as fases de recolha e análise de requisitos, desenho, implementação e testes.

O projecto APSE foi desenvolvido nas instalações da Skysoft Portugal S.A., uma empresa portuguesa pertencente ao grupo espanhol GMV, cuja área de acção e especialização abrange principalmente as áreas de auronáutica e espaço.

PALAVRAS-CHAVE:
software, programação, tripulação, aeronáutica, inteligente

Abstract

Software components have been increasing their importance regarding the achievment of a correct aircraft functioning, as well as respecting the operational procedures and avoiding human errors. This report presents the work around the development of a computational tool with the purpose of supressing the human mistakes while piloting an aircraft, the APSE (Application Programmer Support Environment) tool.

The APSE is an application that supports another application, the ICS (Intelligent Crew Support), and its objective is to help programming the ICS. The ICS is a software component developed with the objective of monitoring the crew's actions and to offer advices regarding the current situation. Although, the ICS need specific programming and the APSE is the tool responsible for that. Both applications were developed in the scope of a much larger project, and both are included in it: the FLYSAFE project. The ICS is an application that was developed by another partner, in another country.

The work has covered the total software development process, including the requirements analysis, design, implementation and testing.

The APSE project was developed at Skysoft's headquarters. Skysoft Portugal S.A. is a portuguese company that belongs to the spanish group GMV. The specialization area of this group mostly covers aeronautics and space.

KEYWORDS:
software, programming, crew, aeronautics, intelligent

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Este documento relata o trabalho realizado no âmbito do Mestrado em Engenharia Informática (MEI), para a disciplina Projecto de Engenharia Informática. O trabalho foi desenvolvido na empresa Skysoft Portugal S.A., em ambiente profissional, enquadrado num projecto de dimensão mais ampla, o projecto FLYSAFE da Comissão Europeia.

A aplicação APSE (*Application Programmer Support Environment*) é o tema central deste relatório. É parte integrante do projecto FLYSAFE, um projecto complexo e de cariz abrangente, no âmbito da aeronáutica, que foca um conjunto de situações eventualmente problemáticas, que podem fazer perigar a segurança da tripulação e passageiros a bordo de uma aeronave, e que foi concebido e está ainda a ser desenvolvido por um conjunto de mais de três dezenas de empresas europeias, com o objectivo de tornar o espaço aéreo mundial um local seguro para que a aviação comercial possa prosperar.

Este projecto foi desenvolvido durante o ano lectivo de 2006/2007. Prevê-se que o projecto FLYSAFE, na sua totalidade, dure cerca de 4 anos. No entanto, o componente APSE não terá relevância nem será utilizado em todas as fases do projecto. Isto porque esta aplicação é utilizada como apoio à programação de um outro componente, o ICS (*Intelligent Crew Support*), que será descrito mais adiante, e por isso o APSE necessita estar desenvolvido completamente antes do ICS.

Dado o carácter internacional do projecto, muitos termos que poderiam ser traduzidos para português foram mantidos no inglês, de modo a manter a coerência entre o que está no relatório e os termos que foram utilizados no processo de desenvolvimento de *software*.

1.1 A Skysoft Portugal S.A.

Ao realizar este projecto na Skysoft Portugal, foi possível conciliar os percursos académico e profissional, visto já ser colaborador da empresa aquando da tomada

de decisão de efectuar o Mestrado. Assim sendo, foi possível conciliar os interesses da empresa com os requisitos do projecto de Engenharia Informática do MEI, e daí surgiu a possibilidade de incluir todo o trabalho que efectuaria este ano lectivo no MEI.

Na Skysoft, trabalhamos muitas vezes em projectos utilizando tecnologia de ponta, em diversificados meios como a aeronáutica, o espaço, transportes, mobilidade e telemática. Esses projectos desempenham muitas vezes um papel preponderante no avanço da tecnologia e é-nos oferecida a possibilidade de trabalhar no desenvolvimento dessas novas tecnologias, o que é de algum modo gratificante.

A Skysoft Portugal S.A. faz parte do enorme consórcio que se propôs desenvolver o projecto. Este consórcio é formado por 36 empresas de 14 diferentes países e constitui um poderoso grupo de desenvolvimento de *software*, dadas as diferentes áreas de especialização presentes. Este projecto enquadra-se, no que diz respeito à Skysoft Portugal S.A. no departamento de Aeroespacial, Segurança e Defesa, pois este é logicamente o departamento a que são atribuídos os projectos relacionados com aviação comercial, devido à sua já larga experiência e fortes competências no que refere a este tipo de problemas.

A empresa conta neste momento com cerca de 80 empregados e colaboradores, e pertence à multinacional espanhola GMV.

1.2 Organização do documento

Este documento está organizado da seguinte forma:

- Capítulo 2 - O projecto FLYSAFE e o APSE - Neste capítulo, é apresentado o projecto FLYSAFE e é feito o enquadramento da aplicação APSE no mesmo. É efectuada uma gradual apresentação dos módulos e sub-módulos em que a aplicação está inserida, de modo a dar a entender a importância da aplicação no âmbito do projecto. São também especificados aqueles que são os objectivos do APSE, assim como a metodologia seguida. É também apresentado o planeamento realizado aquando do relatório preliminar e o planeamento realizado posteriormente, após uma noção mais precisa do que seria necessário desenvolver, contendo um muito maior nível de detalhe. Finalmente, é feita a confrontação entre o planeamento inicial e o planeamento efectuado posteriormente, na altura de um maior conhecimento em relação ao projecto.

- Capítulo 3 - O APSE - Aqui é apresentada de forma detalhada a aplicação APSE. É também descrita a total colaboração da Skysoft Portugal no projecto FLYSAFE; é demonstrada a importância do APSE para a aplicação ICS, apresentando sucintamente essa mesma aplicação; são apresentadas as linguagens

ADL e PDL, as quais são parte fulcral do projecto, dado que um dos maiores objectivos do APSE é gerar eficientemente código referente a estas duas linguagens; é apresentado o desenvolvimento faseado do APSE, detalhando parcialmente as diferentes tarefas inerentes ao seu desenvolvimento; resumem-se as ferramentas e tecnologias utilizadas; e por último, é mostrado um exemplo da execução da aplicação.

- Capítulo 4 - Conclusão - A parte final do relatório está reservada à conclusão. Aqui é feito um pequeno resumo do que foi desenvolvido ao longo do projecto, bem como uma pequena análise crítica ao seu desenrolar durante o ano e ainda uma pequena introdução ao que poderá ser o trabalho futuro.

Capítulo 2

Contexto, objectivos, metodologias e planeamento

2.1 O projecto FLYSAFE

Estima-se que o tráfego aéreo triplicará nos próximos 20 anos. O equipamento actualmente existente a bordo e os sistemas de vigilância existentes no solo, se não forem adaptados às crescentes necessidades do tráfego aéreo, contribuirão para um aumento de incidentes com vítimas na mesma ou em maior proporção. Apesar do facto de hoje em dia os acidentes serem raros, este aumento é tido como inaceitável, do ponto de vista social e, por isso, novos sistemas e soluções devem ser encontrados de modo a que o número de acidentes se mantenha ao baixo nível corrente. Como a segurança dos voos depende largamente das acções dos membros da tripulação, é essencial que lhes seja frequentemente oferecida informação fiável. Além disso, é necessário certificar que os membros da tripulação agem conforme previamente estabelecido e que não executam acções inesperadas nem deixam de executar acções que deveriam ter executado. O projecto FLYSAFE é responsável por desenvolver os novos sistemas necessários para que a tripulação possa tomar a decisão correcta de modo a evitar conflitos causados por fenómenos meteorológicos, tráfego e terreno.

O FLYSAFE foca essencialmente as áreas identificadas como sendo as maiores causas de acidentes aéreos por todo o mundo: perda de controlo da aeronave, voo controlado e inadvertido contra o terreno e acidentes nas fases de voo de descolagem e aterragem. Existem 3 tipos de ameaças principais a ser analisadas neste projecto: condições meteorológicas adversas, ameaças relacionadas com tráfego e ameaças relacionadas com terreno. Para cada um desses tipos estão a ser desenvolvidos novos sistemas e funcionalidades, a saber: desenvolvimento de novas tecnologias de modo a oferecer um conhecimento da área/ambiente circundante ao avião mais fiável e completo, avisos de alerta emitidos com suficiente antecipação para facilitar a capacidade de resposta por parte da tripulação, prioritização eficiente de alertas e uma interface homem-máquina inovadora.

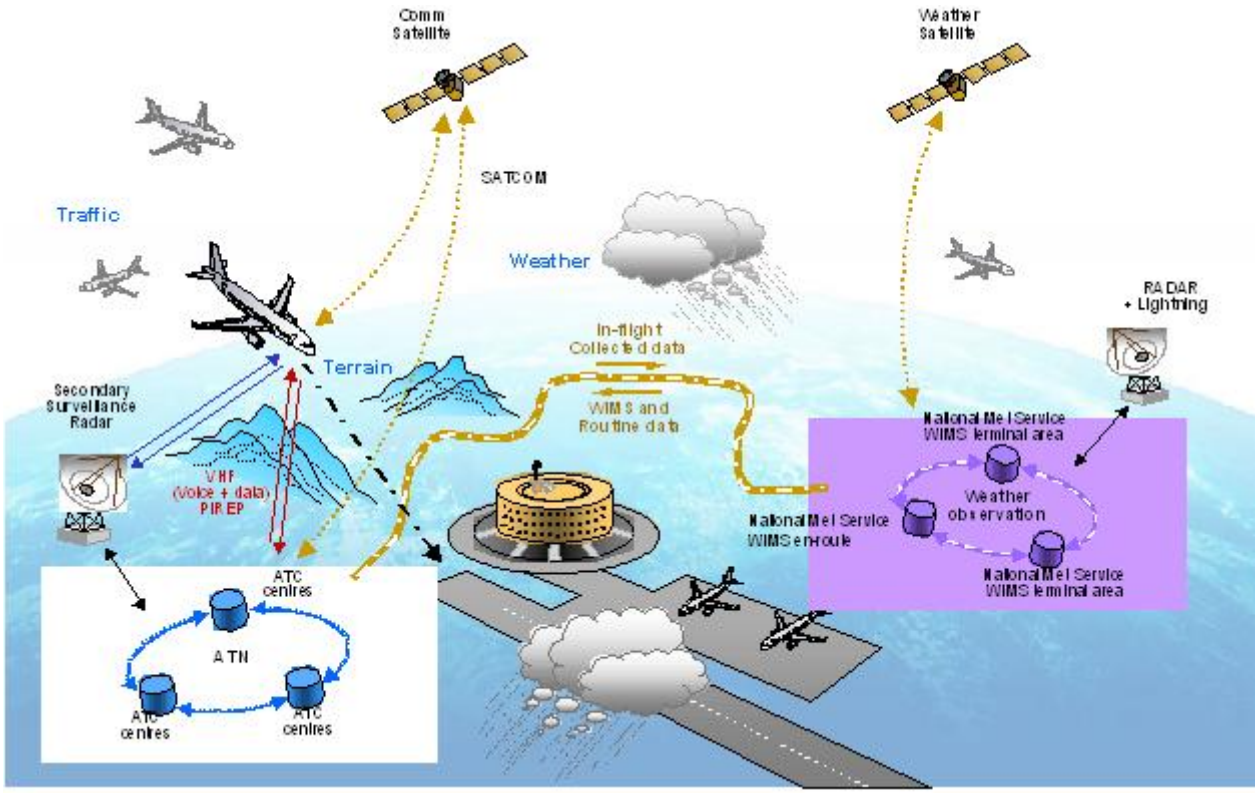

Figura 2.1: O conceito FLYSAFE

O projecto começou com uma revisão dos resultados de investigações passadas e decorrentes sobre acidentes e incidentes, com a identificação das causas prováveis e com a definição de modos de detectar as ameaças responsáveis por eles. O resultado desta análise foi utilizado na elaboração de requisitos de alto nível e de cenários operacionais que ofereçam as devidas condições de teste que serão utilizadas para testar as novas tecnologias e compará-las com as que são utilizadas correntemente. Como foi dito anteriormente, existem 3 tipos principais de ameaças para a aviação: condições meteorológicas adversas, tráfego e terreno. Estas diversificadas ameaças levaram à criação de 3 ramos distintos no projecto, havendo ainda um quarto ramo dedicado ao desenvolvimento do próprio *Next Generation Integrated Surveillance System* onde convergem os 3 outros ramos:

- "*Atmospheric Hazards*" - aos colaboradores responsáveis por este módulo cabe-lhes a tarefa de desenvolver meios para que a tripulação esteja mais ciente do ambiente que a rodeia, do ponto de vista meteorológico. Isto inclui o desenvolvimento de novas tecnologias com capacidades superiores às existentes hoje em dia que ajudam a aumentar a exactidão e a precisão do equipamento de vigilância meteorológica a bordo do avião. Estes novos equipamentos são desenvolvidos para oferecer uma maior fiabilidade na detecção e evasão em

relação às maiores e mais perigosas ameaças atmosféricas (rastro aerodinâmico, cisalhamento, turbulência em céu limpo, formação de gelo e tempestades; para helicópteros existe também um módulo em que se pretende melhorar a informação disponível em situações de má visibilidade: fumo, nevoeiro, etc.).

- "*Traffic Hazards*" - neste outro módulo, são desenvolvidos meios para aumentar o nível de atenção e a consciencialização da tripulação em relação ao ambiente circundante, no que diz respeito a outro tráfego aéreo e são também desenvolvidos métodos de detecção de eventuais conflitos com o seu plano de voo a médio e longo prazo.

- "*Terrain Information Management*" - à semelhança do módulo anterior, são desenvolvidos métodos para que a tripulação esteja totalmente a par das eventuais ameaças de terreno (montanhas, etc.) e de outros obstáculos (construções humanas), assim como metodologias que permitem a detecção com maior antecedência de eventuais conflitos com este tipo de ameaças.

2.1.1 O *Next Generation Integrated Surveillance System* (NG-ISS)

O projecto oferece a possibilidade de que seja desenhado, desenvolvido, implementado, testado e validado um completo sistema denominado *Next Generation Integrated Surveillance System* (NG-ISS), que constituirá um passo decisivo para além daquilo que é oferecido pelos sistemas de segurança integrados que presentemente estão a surgir. O NG-ISS é um dos componentes fulcrais do projecto pois nele são integradas as colaborações dos segmentos meteorológico, de tráfego e de terreno. Fazendo parte do NG-ISS, são desenvolvidos sistemas com funções inovadoras:

- "*Strategic Data Consolidation*" para antecipar qualquer ameaça a nível estratégico relacionada com fenómenos atmosféricos, tráfego ou terreno, que esteja em conflito com o plano de voo da aeronave. Esta funcionalidade reduz substancialmente o número de alertas tácticos gerados no *cockpit* através de uma antecipação apurada dessas ameaças e aconselhando a tripulação quando um replaneamento do plano de voo é necessário. Um conflito estratégico entende-se como sendo um conflito que pode ser detectado a médio ou longo prazo; ao invés, um conflito táctico é um conflicto que é detectado a curto prazo.

- "*Tactical Alert Management*" para ajudar a tripulação a gerir todos os alertas gerados pelas funções de último recurso, como o ACAS (*Airborne Collision Avoidance System*), sistema responsável por garantir que não existirão colisões

aéreas entre aviões, o TAWS (*Terrain Avoidance and Warning System*), responsável por garantir que não existirão colisões entre a aeronave e o terreno e a função de reacção ao cisalhamento, onde uma resposta imediata é imperativa de modo a evitar a catástrofe.

- "*Intelligent Crew Support*" para providenciar apoio à tripulação no caso de um eventual erro que possa ser cometido devido ao excesso de factores que é necessário monitorizar em certas fases do voo, fadiga, ansiedade, etc., através da monitorização da fase de voo, do ambiente envolvente e das acções tomadas pela tripulação.

Com a implementação deste sistema, pretende-se que seja obtida uma nova autonomia por parte de cada aeronave em relação a controladores e a outras entidades, de modo a garantir uma maior segurança a todos os níveis do voo, mas de uma forma totalmente independente, ao contrário do que ocorre hoje em dia, em que grande parte da segurança e das garantias de uma separação segura entre aeronaves é posta nas mãos de controladores aéreos.

2.1.2 O *Intelligent Crew Support* (ICS)

A percentagem de acidentes ou incidentes de aviação cuja origem é atribuída a um erro da tripulação (quer encarado como um factor apenas, quer faça parte de uma combinação de circunstâncias não previstas nos procedimentos habituais de emergência) é significativamente maior do que a atribuída a qualquer outro factor. Apesar dos melhoramentos a nível da fiabilidade e dos automatismos dos sistemas aviónicos terem reduzido o número de acidentes no geral, a proporção de incidentes ocorridos resultantes de erro humano (como principal causa) tem-se mantido constante, com valores a rondar os 70%.

Existem vários factores que podem contribuir para a ocorrência de erros por parte da tripulação:

- Situações de stress.
- Situações em que a carga de trabalho é substancialmente alta, nomeadamente nas fases de descolagem e aterragem, em que existe uma infinidade de parâmetros e variáveis que é necessário monitorizar.
- Nível inadequado de atenção por parte da tripulação.
- Tripulação com treino insuficiente ou inexperiente.
- Limitações ao nível da elaboração de interfaces pessoa-máquina para sistemas aviónicos.

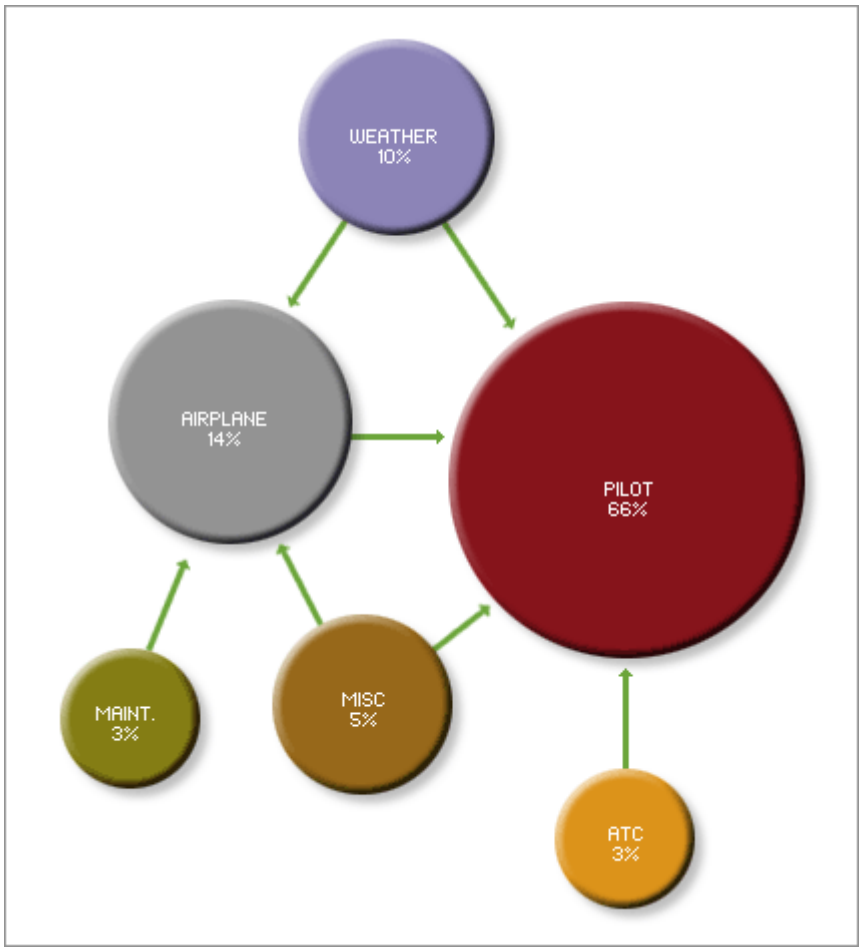

Figura 2.2: Acidentes aéreos e os seus principais responsáveis

- Condições ambientais adversas.
- Mudanças não observáveis de estado nos sistemas aviónicos.

Estes factores podem estar relacionados uns com os outros. A situação de alto nível de stress por parte da tripulação pode ocorrer, por exemplo, aquando de uma falha do sistema na presença de condições meteorológicas adversas. Pode igualmente ocorrer em pilotos de aviões militares em situações de combate. A falta de atenção da tripulação pode resultar de grandes períodos de pouca actividade por parte da tripulação, como no caso de um voo de longo curso. Condições ambientais adversas podem surgir no *cockpit* sob a forma de elevados níveis sonoros, o que pode levar a dificuldades em escutar as mensagens enviadas pelos controladores aéreos, ou sob a forma de elevadas temperaturas.

Todos estes factores podem contribuir para uma má compreensão da informação fornecida pelos instrumentos de bordo e para uma redução da percepção do piloto em relação à situação geral da sua aeronave, aumentando assim a probabilidade de erros serem cometidos. As operações para voar um avião regem-se por rigorosos pro-

cedimentos e o desrespeito por esses procedimentos pode levar a potenciais situações perigosas, mesmo que esse desrespeito não seja ele próprio directamente perigoso, e isso é totalmente inaceitável.

Na busca para limitar as ocorrências e o impacto dos erros da tripulação, o núcleo do sistema ICS, designado por *kernel* do ICS, providencia uma vasta gama de monitorizações programáveis e de geração de ajudas à tripulação. O *kernel* do ICS foi desenhado para ser genérico de modo a poder ser utilizado por uma grande variedade de aplicações. Mais precisamente, os seus objectivos gerais são:

1. Monitorizar as características e as atitudes tomadas pela tripulação e o ambiente existente no *cockpit*, de modo a antecipar eventuais erros que possam levar a situações perigosas.

2. Monitorizar o estado do sistema de modo a prever se um estado eventualmente perigoso estará prestes a ser atingido.

3. Utilizar as suas capacidades para reconhecimento de planos de voo de modo a fornecer atempadamente avisos pertinentes no caso de um evento provocado por uma acção errónea da tripulação ser identificado.

4. Optimizar dinamicamente a interface homem-máquina de acordo com o estado corrente do ambiente e com a performance da tripulação.

5. Comparar as acções efectuadas pela tripulação com uma biblioteca predefinida de sequências de acções permitidas de modo a detectar acções omissas ou inapropriadas consoante o que é requerido em determinada situação.

6. Fornecimento de capacidades multi-modais de *input/output* de modo a ser possível integrar meios alternativos de controlar o sistema.

A configuração do *kernel* do ICS é bastante flexível, de modo a que apenas as funções realmente necessárias a uma situação ou aplicação específicas sejam instanciadas. Uma aplicação foi desenvolvida, o APSE (*Application Programmer Support Environment*) de modo a assegurar que a programação do ICS é feita de modo eficiente e que são gerados atempadamente os alertas pertinentes em cada situação e a devida informação a ser apresentada à tripulação.

2.2 Objectivos do APSE

Como já foi dito anteriormente, o *kernel* do ICS é programado recorrendo ao uso da aplicação APSE, que facilita em muito a programação e a inserção de novas condições de modo a tornar mais eficiente o apoio que pode ser dado à tripulação pelo dito ICS.

Isto ajuda a gerar os alertas e as informações necessários, o que leva à eliminação de situações onde um comportamento errado é de todo indesejado. Um projecto desenvolvido no APSE consiste num conjunto de objectos contendo um diagrama de fluxo de dados principal, diagramas de estados, casos de uso e definições de acções, que todos juntos irão formar o requerido suporte ao ICS. O APSE oferecerá uma interface gráfica de fácil utilização, que permite a definição detalhada de objectivos e de actividades que servem como suporte à programação do ICS tendo como base os casos de uso existentes, definidos pelo utilizador. O típico utilizador do APSE é alguém encarregado de programar o ICS. Uma grande variedade de ajudas gráficas foi desenvolvida de modo a facilitar a introdução de informação pelo utilizador e de modo a que a verificação desta seja feita de forma eficiente para que a sua inclusão no ICS seja feita de modo a evitar qualquer procedimento erróneo. O código do ICS será constituído pelo conjunto do seu código do seu *kernel* e do código específico gerado pelo APSE. A partir da informação introduzida pelo utilizador, os diagramas relativos a essa informação são gerados e podem ser manipulados de modo a criar relações entre estados, casos de uso e acções. Dessa informação introduzida e modificada nos diagramas por parte do utilizador, é gerado código PDL (*Plan Definition Language*) e ADL (*Application Description Language*), que serve para ser integrado directamente no *kernel* do ICS, consoante os requisitos necessários a determinada fase do voo.

2.3 Metodologia

O APSE faz parte integrante de um projecto abrangente, com uma enorme equipa a trabalhar em conjunto, quer dentro da Skysoft, mas sobretudo tendo em conta os outros parceiros que constituem o consórcio.

A figura 2.3 mostra a distribuição dos *work packages* do projecto FLYSAFE. Nela são apresentados os *Work Packages*de nível 1 e 2. Pode observar-se a organização hierárquica dos módulos do projecto e notar que o projecto obedece a uma aproximação *top-down*, em que primeiro foram sendo definidos os *Work Packages* de nível superior, para posteriormente serem também eles divididos e refinados. Pode observar-se a organização hierárquica do projecto, existindo *Work Packages* de nível 1 até 4. O APSE faz parte integrante do *work package* 5.3 (WP de nível 2), que corresponde ao *Intelligent Crew Support* (ICS). Este, por sua vez, integra o *work package* 5 (WP de nível 1), referente ao *Next Generation Integrated Surveillance System* (NG-ISS). O WP de nível 3 (5.3.n) corresponde às diversas fases de desenvolvimento do ICS e o WP de nível 4 (5.3.x.y) corresponde às diversas fases de desenvolvimento do APSE.

Um projecto desta dimensão tem de obedecer a mecanismos apertados de con-

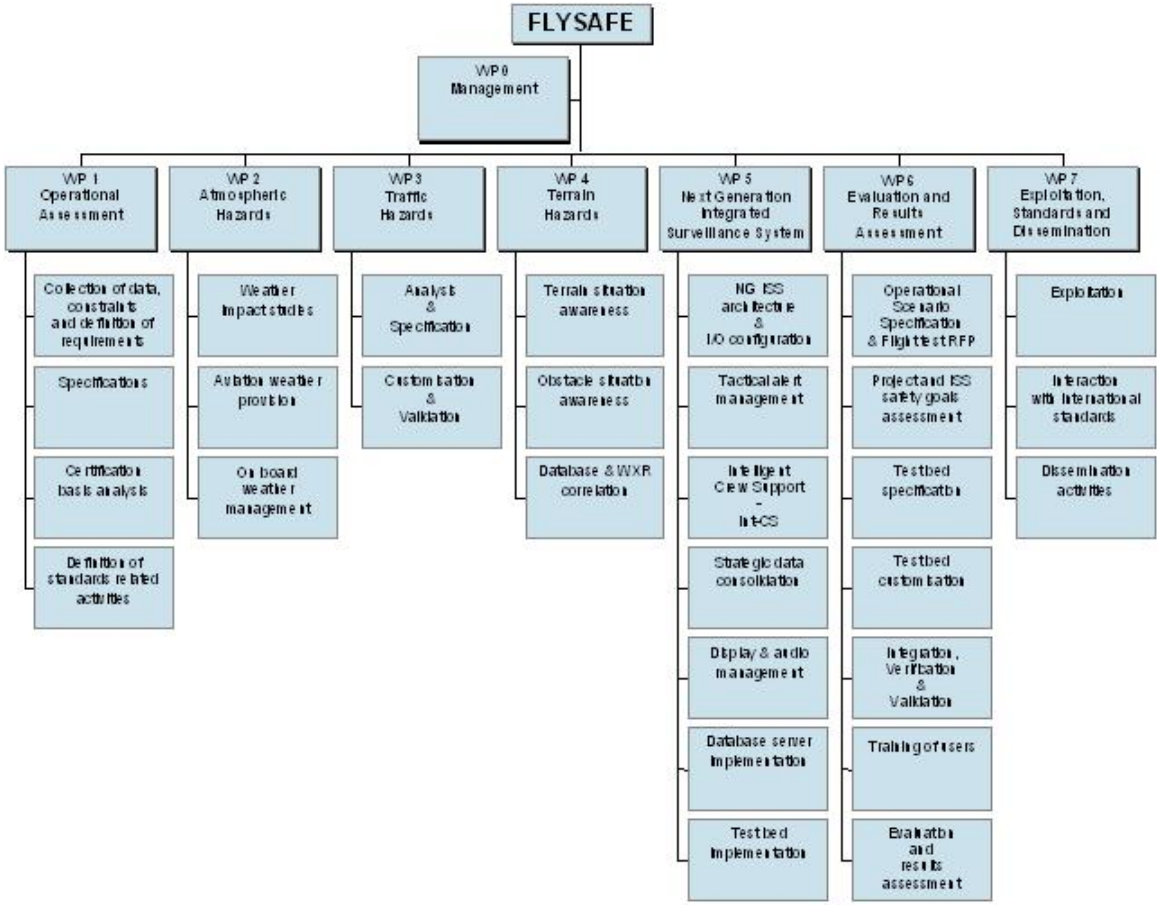

Figura 2.3: Divisão hierárquica dos *work packages* do projecto FLYSAFE

trolo de gestão e documentação. A interacção entre a Skysoft e o parceiro encarregado de desenvolver o ICS, a empresa ingles BAE Systems, não foi excepção.

O desenvolvimento do APSE seguiu um processo iterativo, com estreita comunicação entre as equipas portuguesa e inglesa. A troca de ideias entre as duas equipas foi constante e não raras foram as vezes em que o trabalho teve parcialmente de ser refeito ou modificado de modo a obedecer aos requisitos propostos no início do projecto e aos novos que iam surgindo à medida que o ICS era desenvolvido. Houve por vezes a necessidade de introduzir novas funcionalidades, o que levou a alterações da especificação e do desenho durante a fase de desenvolvimento de código (implementação) propriamente dito. Assim sendo, pode-se afirmar que o processo utilizado, teoricamente falando, foi o modelo incremental apresentado na figura 2.4, em que várias iterações são efectuadas até obter o produto final.

Deste modo, obedecendo a esse processo iterativo, o APSE foi sendo desenvolvido e várias entregas faseadas foram efectuadas:

- "*ICS APSE Specification*" - este documento foi desenvolvido em conjunto com a empresa inglesa, de modo iterativo também, e constitui a especificação e a

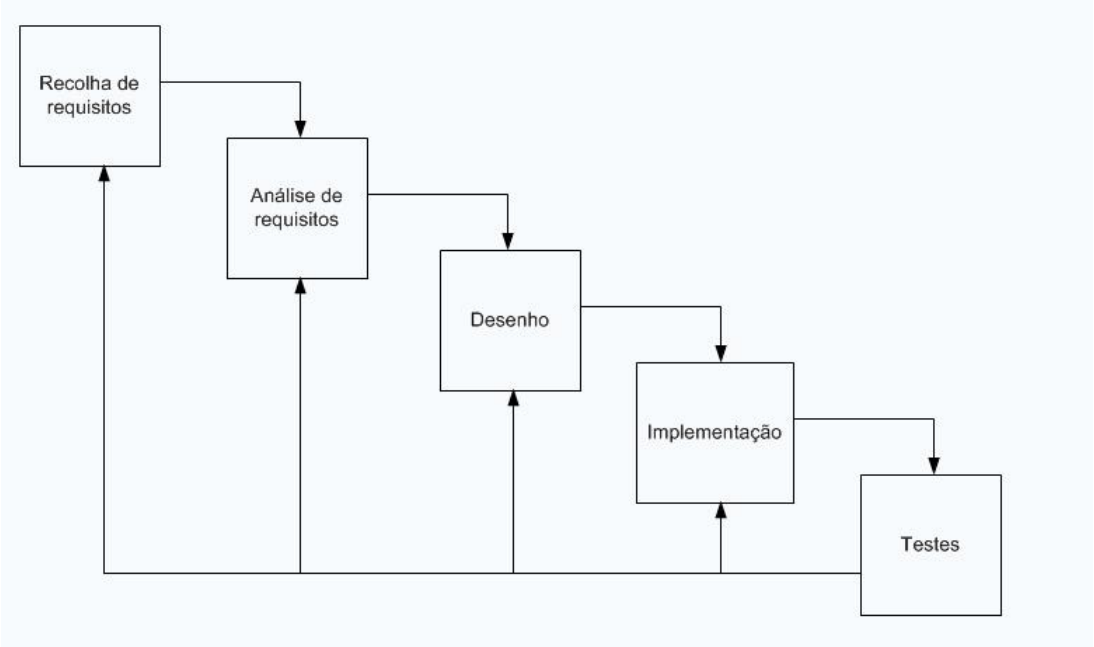

Figura 2.4: Processo incremental de desenvolvimento de software

análise de requisitos correspondente ao APSE. Nele se inclui a descrição de todas as funcionalidades que o APSE teria de oferecer.

- "*APSE software*" - aqui se inclui a entrega do *software* propriamente dito. Foi dada alguma liberdade no que diz respeito à implementação, e assim sendo, mais uma vez, as entregas foram feitas de modo iterativo: a entrega era efectuada, os proprietários do ICS testavam-na devidamente de modo a verificar se tudo funcionava devidamente, fornecendo depois alguma informação de avaliação, contendo sugestões ou pequenos relatórios de erros. Após este fluxo de informação, o processo repetia-se até a entrega satisfazer totalmente o cliente.

- "*APSE User Guide*" - como o próprio nome do documento indica, este contém o manual de instruções necessário para operar o ICS. Aqui se encontra a descrição de todas as capacidades que o *software* apresenta descritas de forma concisa, de modo a que um operador estranho ao desenvolvimento do APSE o consiga operar com facilidade.

Toda a documentação produzida no âmbito do projecto foi alvo da revisão e aprovação por parte do departamento da qualidade da Skysoft Portugal S.A., tendo sido posteriormente revista e aprovada também pelos membros do consórcio.

2.4 Planeamento

Na altura da da definição do primeiro plano, aquando do início deste trabalho, em Outubro de 2006, muitas decisões não estavam ainda tomadas e o projecto estava ainda numa fase muito embrionária, razão pela qual o planeamento efectuado na altura não estava devidamente detalhado. Havia ainda muitas questões em aberto, algumas que fugiam ao âmbito do responsável pelo desenvolvimento do *software*, relativamente a questões políticas entre as duas empresas e de carácter de gestão, que tiveram que ser devidamente discutidas pelos responsáveis das duas empresas. Deste modo, apenas após essas discussões foi possível começar a definir um plano mais detalhado, com a colaboração dos nossos colegas ingleses.

Seguidamente, é apresentado o plano inicial e o plano efectuado aquando de uma maior definição do trabalho que iria ser efectuado.

Como é de notar, o planeamento na altura da entrega do relatório preliminar, em Novembro de 2006, era ainda vago, muito generalizado. Apenas continha as tradicionais fases do processo de desenvolvimento de *software*, pois na altura não havia ainda informação mais concreta daquilo que seriam as tarefas a realizar no âmbito do projecto. Assim sendo, é bastante notória a diferença entre o planeamento inicial e o planeamento aqui apresentado após esse, tendo em conta o nível de detalhe. O segundo planeamento foi elaborado poucas semanas após a entrega do relatório preliminar e foi esse planeamento que guiou o desenvolvimento do projecto.

Tendo em conta o planeamento temporal e o cumprimento de datas, nota-se alguma disparidade e houve realmente alguma dificuldade em conseguir seguir à risca o planeamento. Isto devido, sobretudo, ao facto de o processo ter sido tão iterativo e de a aplicação ICS também estar a ser desenvolvida paralelamente ao APSE. Ao depender tão proximamente do ICS, é natural que o APSE só pudesse ser desenvolvido em pleno quando o ICS se encontrasse num estado de relativa maturidade, e enquanto isso não acontecesse, todo o desenvolvimento do APSE poderia sofrer um grande revés se a especificação e desenvolvimento do ICS sofresse ela também uma ou mais significativas alterações.

ID	Tarefa	Data de início	Data de fim	Duração
1	Análise de requisitos	18-10-2006	30-11-2006	32d
2	Desenho	01-12-2006	15-01-2007	32d
3	Implementação	16-01-2007	15-05-2007	86d
4	Testes	16-05-2007	29-06-2007	33d

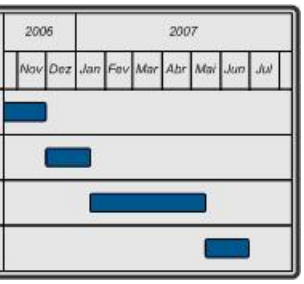

Figura 2.5: Planeamento efectuado para o relatório preliminar

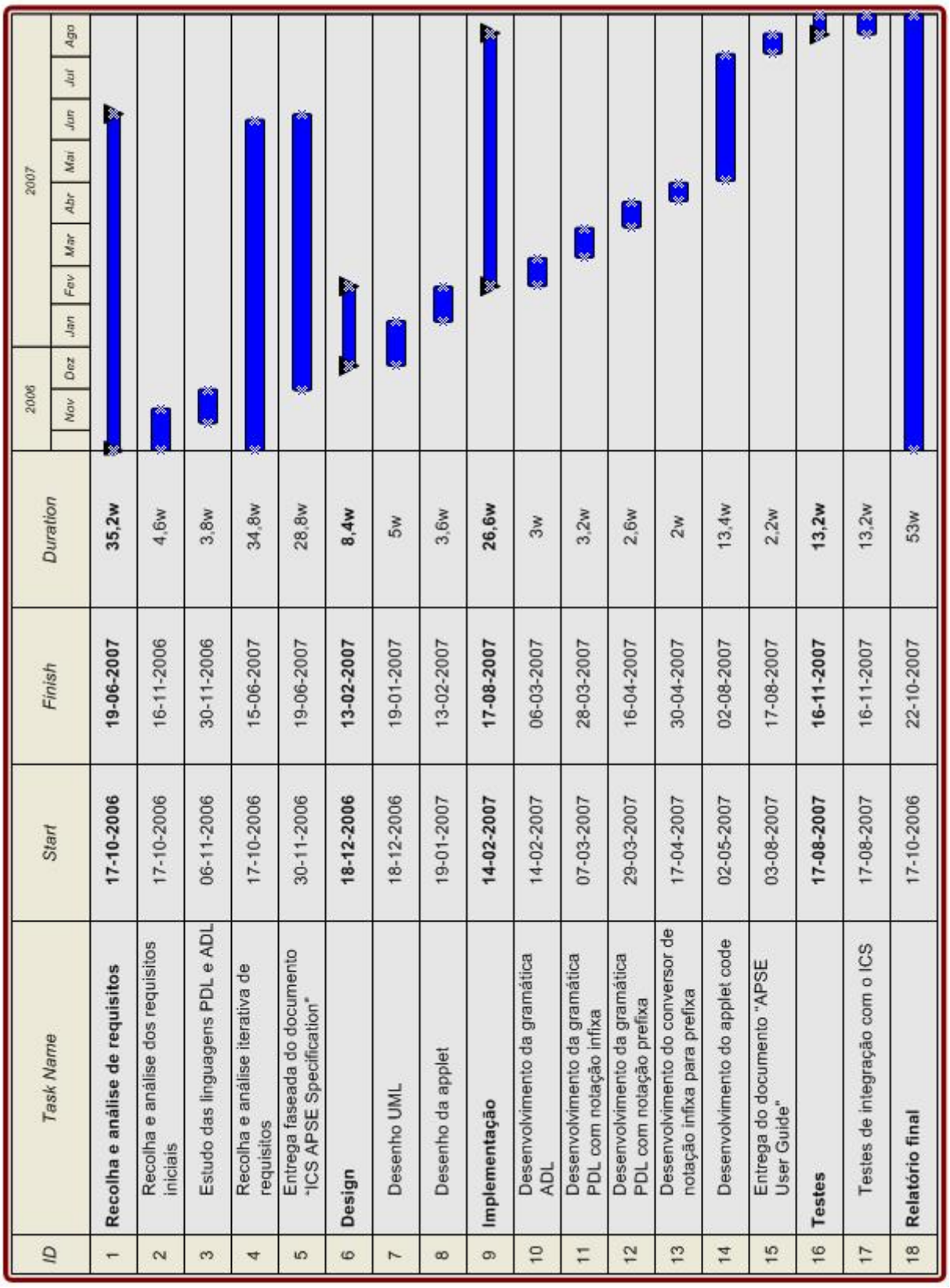

Figura 2.6: Planeamento

Capítulo 3

Trabalho realizado

3.1 O FLYSAFE na Skysoft Portugal

Como já foi dado a entender anteriormente, o APSE está englobado num projecto de grande dimensão, em que uma grande diversidade de *software* foi desenvolvido, quer pela nossa equipa, quer pelas outras equipas de outras empresas.

Parte da contribuição da Skysoft neste projecto insere-se em factores relacionados com tráfego aéreo, visto a empresa ter já uma larga experiência neste ramo, adquirida com a participação em outros projectos em que o tráfego aéreo foi debatido e sujeito a diversos estudos e com a especialização dos seus funcionários no dito ramo quer através de acções de formação com controladores aéros, pilotos e outras autoridades credenciadas, quer através de fortes apostas na formação académica, contando com uma vasta equipa de engenheiros aeronáuticos, aeroespaciais e pilotos. Outra das importantes contribuições da Skysoft será o desenvolvimento dum estudo sobre uma eventual implementação de um *datalink* entre os segmentos meteorológicos situados no solo e o equipamento presente a bordo. Para além desse trabalho relacionado com o segmento de tráfego e com o segmento meteorológico, a Skysoft irá também desenvolver variados sub-componentes de software do NG-ISS, como o motor de gestão de alertas tácticos, a aplicação de detecção de conflitos estratégicos e o APSE.

Para além do total desenvolvimento da aplicação APSE, como membro da equipa responsável por trabalhar no projecto FLYSAFE, a minha função englobou também a colaboração noutros módulos do projecto, embora de um modo bastante mais secundário. Assim sendo, aqui ficam discriminados os módulos desenvolvidos pela nossa equipa, explicitando também a minha contribuição em cada um deles, de um modo sucinto, visto não serem elas o elemento central deste documento:

- No segmento de tráfego:
  - Fusão de dados de tráfego: recolha de informação dos diversos sensores responsáveis por gerar informação relativa a tráfego aéreo, desenvolver

algoritmos de modo a conseguir a consolidação e uma maior fiabilidade no que diz respeito a essa informação.

Contribuição: Parcial colaboração na análise de requisitos e parcial codificação

- Desenvolvimento de funções de último recurso: estas funções serão o último recurso responsável por garantir que não existirão colisões em pleno voo, quando todos os outros métodos de garantir uma separação segura não resultaram.

  Contribuição: Parcial colaboração na análise de requisitos

- SMGCS - *Surface Movement Guidance Control System*: Uma ferramenta utilizada pelo controlador aéreo que o auxilia e detecta automaticamente potenciais conflitos a ocorrer no segmento terrestre. É um produto já propriedade da Skysoft, para aumentar a segurança ao nível dos aeroportos, que irá ser adaptado de acordo com os novos requisitos do projecto FLYSAFE.

  Contribuição: Ainda não determinada. Até este momento ainda não está totalmente definida a colaboração da Skysoft no que diz respeito a este módulo. Assim sendo, a colaboração até esta altura é inexistente.

- No segmento meteorológico:

  - Estudo sobre os métodos de ligação e transferência de dados, entre o segmento meteorológico em terra e o equipamento a bordo. Compreensão das necessidades dos sensores meteorológicos de modo a determinar a taxa de envio e recepção de dados, a quantidade de dados necessária para que esses mesmos sensores possam funcionar na plenitude das suas funções, determinar e implementar os requisitos de segurança para essa transferência de dados e desenvolver algoritmos para aumentar a fiabilidade desta comunicação, nomeadamente a possível implementação de um protocolo de comunicação que a garanta. Adaptação e teste em voo real de um produto proprietário, um método de ligação e transferência de dados VHF por satélite desenvolvido em conjunto com a ESA, utilizando a tecnologia VDL modo 2.

    Contribuição: - Nenhuma.

- No NG-ISS:

  - Gestão de alertas tácticos: este módulo será responsável por recolher todos os alertas tácticos (a curto prazo) dos segmentos de tráfego, terreno e

meteorológico, e proceder à respectiva prioritização, de modo a apresentar apenas um alerta à tripulação, e assim não correr o risco desta ficar confundida com o aparecimento de vários alertas e ter que escolher qual deles tentar evitar.

Contribuição: Parcial colaboração na análise de requisitos e parcial codificação

- Detecção de conflitos de tráfego a nível estratégico: detecção de eventuais conflitos que possam existir a longo prazo entre aeronaves. Por longo prazo entende-se um limite de tempo em que a manobra evasiva possa ser efectuado sem brusquidão, de modo a não afectar o conforto dos passageiros.

  Contribuição: - Nenhuma.

- APSE

  Contribuição: Todo o processo de desenvolvimento de software. Este módulo foi totalmente desenvolvido apenas por mim, sem qualquer tipo de colaboração de outros colegas.

- Tarefas de disseminação de resultados em feiras, *media* e comunidade científica, dentro da área específica mas também em áreas de conhecimento mais generalizado

  Contribuição: - Nenhuma.

3.2 A importância do APSE para o ICS

O APSE, como já foi dito, é uma aplicação de apoio ao ICS, que é utilizada para programar o *kernel* deste componente. Funciona como uma aplicação isolada (*standalone*), dado que não interage directamente com nenhum outro módulo do projecto e não depende funcionalmente também de nenhum outro. No entanto a sua dependência lógica em relação ao ICS encontra-se patente, embora que implicitamente. Ou seja, não faria nunca sentido desenvolver o APSE se não existisse o módulo do ICS. No entanto, não existe comunicação directa entre eles, o ICS apenas utiliza como *input* os ficheiros que o APSE gera.

O ICS é um *software* que é propriedade de uma empresa inglesa de renome no mundo da aeronáutica. Assim sendo, a informação que foi fornecida acerca do mesmo cinge-se ao mínimo indispensável para a aplicação APSE ser desenvolvida. Naturalmente a dita empresa tenta proteger a sua propriedade intelectual e não divulga na totalidade detalhes da implementação do mesmo.

3.2.1 O *kernel* do ICS

O *kernel* do ICS denomina-se ARCHIE ("*A Reliable Computer - Human Interaction Environment*"). Este *kernel* é composto por diversos módulos, cada um deles com um papel e uma funcionalidade bem definidos. As relações entre esses componentes são representadas no seguinte diagrama:

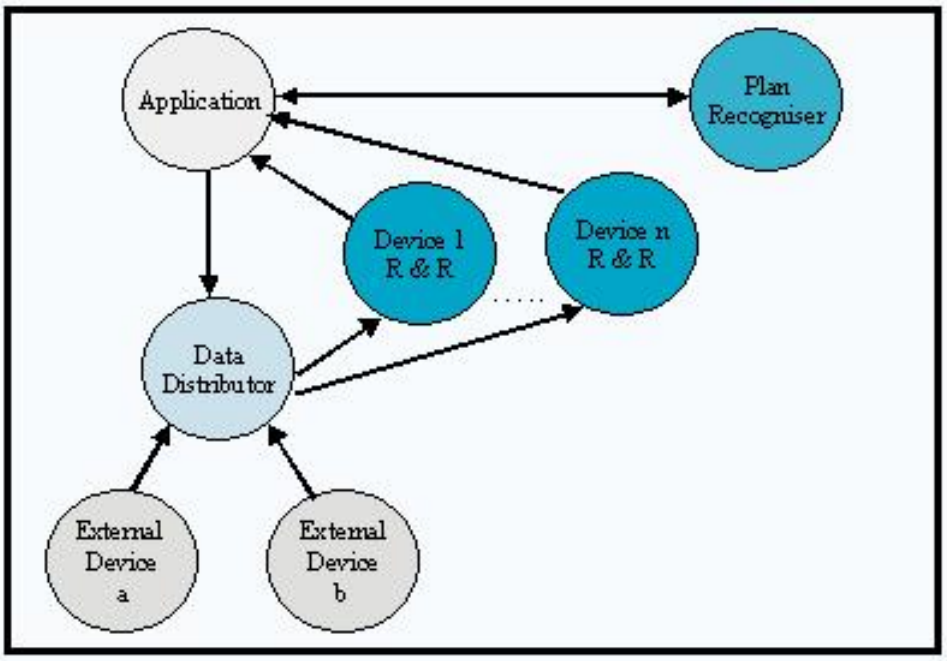

Figura 3.1: O *kernel* ARCHIE

Todos os módulos apresentados na figura, com a excepção do módulo *Application* e dos módulos *External Device*, são componentes do *kernel* ARCHIE:

- *Plan Recogniser* –O componente *Plan Recogniser* (PR) oferece a capacidade de monitorizar as acções do operador, gerando avisos ou recomendações quando uma ou mais acções erradas são detectadas. O PR contem uma biblioteca de planos (procedimentos operacionais), que é utilizada para identificar eventuais erros cometidos pelo operador. As acções do operador são recebidas a partir do componente *Application*. Estas acções são verificadas, de modo a concluir se são consistentes com os planos guardados na biblioteca do PR. Se houver desvios significativos aquando da comparação entre as acções do operador e os planos guardados, um aviso é gerado.

O módulo de PR é instanciado pelo compilador PDL. A linguagem PDL deve ser usada caso se deseje utilizar o componente PR do *kernel*, que permite ao módulo *Application* receber recomendações assim como a predição da intenção do operador da aplicação.

- *Devices R&R (Reports and Recomendations* -A monitorização do desempenho da tripulação e do ambiente envolvente é definido em termos de ADL (*Application Description Language*). O ADL pode ser utilizado de 3 modos:
  - Para permitir que a monitorização de uma aplicação (a aeronave, os seus sistemas ou o seu sistema operativo) seja efectuada com um grau de subtileza ainda não existente nos *Central Warning Systems*. Por exemplo, o *kernel* pode ser configurado de modo a monitorizar uma série de parâmetros relacionados com a aeronave com o fim de providenciar alertas atempadamente no caso de as barreiras superior ou inferior do limite operacional sejam infringidas.
  - O ADL pode também ser utilizado para configurar o *kernel* de modo a monitorizar e reagir a outros aspectos do ambiente ou do estado fisiológico da tripulação, por exemplo, o aumento da sonolência da tripulação no caso de a temperatura no *cockpit* estar entre determinados valores. Neste caso, o *kernel* pode ser programado para monitorizar a tripulação de modo a detectar provas de falta de atenção (como por exemplo, monitorizando a taxa a que o indivíduo pisca os olhos e para que direcção está ele a olhar) e recomendar uma pequena redução na temperatura no *cockpit*.
  - O ADL tem a capacidade de escolher o modo mais apropriado de alertar a tripulação. O *kernel* pode seleccionar entre canais visuais, aurais ou tácteis, escolhendo o mais adequado tendo em conta as condições ambientais.

- *Data Distributor* -O *Data Distributor* é o principal interface entre o *kernel* do ICS e o ambiente envolvente. Os comandos de controlo são recebidos através do módulo *Application* de modo a iniciar ou parar o *Data Distributor*. Do mesmo modo, os dados relativos à monitorização do ambiente são recebidos dos *External Devices*.

- *Application* -O componente *Application* é o responsável pela introdução dos dados de controle no ICS, controlando o estado do ICS: passando de *idle* a *running*, ou ordenando que pare de processar, utilizando o comando *stop*. Este componente é também responsável por receber os dados relativos ao operador (temperatura corporal, frequência do pestanejar, etc.).

- *External Device* -Cada um destes componentes é responsável por introduzir no ICS informação relativa aos dispositivos da aeronave e ao estado actual da mesma. Aqui se incluem uma variedade imensa de parâmetros, que pode ir desde a temperatura no *cockpit* até à velocidade da aeronave.

O *kernel* ARCHIE, sendo o núcleo do sistema ICS oferece a capacidade de gerar uma enorme gama de conselhos para a tripulação, para além de ser possível programar eficientemente o sistema de monitorização das acções do operador. Quando se fala do sistema ICS, o operador é a entidade responsável pela eficiente e segura operação de uma *workstation* complexa. No caso de ser uma aplicação a ser instalada a bordo, o operador pode ser um ou mais membros da tripulação e a *workstation* é o próprio avião que está a ser operado, com todas as suas funcionalidades implicitamente automatizadas. O ICS poderia eventualmente ser instalado num automóvel, sendo o operador, neste caso, o condutor do veículo e a *w*orkstation seria o próprio automóvel. Ao falar de funcionalidades implicitamente automatizadas, quer isto dizer que o ICS assume que a *workstation*, a aeronave, é um sistema devidamente automatizado e eficiente, e que apenas irá considerar o aspecto humano da operação e não o aspecto mecânico. Por outras palavras, o ICS não é um mecanismo para colmatar erros mecânicos ou de qualquer outro tipo em relação à aeronave e cinge-se apenas a monitorizar e emitir aconselhamentos em relação às atitudes tomadas pelo operador ou operadores.

O *kernel* do ARCHIE foi desenhado de modo a ser totalmente genérico e deve poder ser instanciado por uma gama variada de aplicações. A sua aplicação aos princípios aviónicos é apenas uma das múltiplas aplicações para as quais o ICS se revela uma ferramenta de auxílio bastante importante. O ICS pode ser igualmente aplicado a qualquer tipo de operação para a qual exista um operador humano responsável por ela.

As operações típicas efectuadas dentro de um *cockpit* de um avião comercial fornecem um bom exemplo de uma aplicação onde a tripulação trabalha sob uma enorme variedade de situações de stress. Ao mesmo tempo, há procedimentos altamente importantes e que exigem da tripulação um esforço mental e atenção substanciais, dado que qualquer pequeno erro pode revelar-se fatal.

3.3 As linguagens PDL e ADL

O *kernel* do ARCHIE é instanciado recorrendo à utilização de duas linguagens: ADL (*Application Description Language*) e PDL (*Plan Definition Language*). O programador do ARCHIE pode decidir se é ou não usada a plenitude das funcionalidades do sistema ARCHIE utilizando ambas ou apenas uma das duas linguagens.

Esta secção descreverá sucintamente as linguagens ADL e PDL. Estas linguagens foram desenvolvidas e são propriedade intelectual de uma empresa inglesa, à semelhança do ICS. Mais uma vez, apenas foi facultada a informação suficiente de modo a poder ser desenvolvido o *software* APSE. Sendo o APSE um módulo de *software* em que o seu objectivo principal é fornecer as funcionalidades e os mecanismos de modo a poder ser gerado quer código ADL, quer código PDL, uma pequena introdução ao que são ambas as linguagens é oferecida seguidamente. O objectivo das secções descritivas da linguagem não é fornecer uma explicação ou relato das capacidades das linguagens. É somente dar a conhecer sucintamente as ditas linguagens e tentar enquadrar as linguagens no âmbito do ICS e a sua utilização no mesmo.

Não existe qualquer relação entre as duas linguagens. O ADL é utilizado para instanciar os módulos *Device R&R* enquanto que o PDL é utilizado para instanciar o módulo PR.

3.3.1 A linguagem ADL

A linguagem ADL é uma linguagem de especificação, destinada a instanciar os módulos *Device R&R* do *kernel* do APSE. Tem características de linguagem imperativa, com uma sintaxe próxima da linguagem C, mas muito mais limitada ao nível dos comandos de controlo de fluxo. Não existe qualquer tipo de suporte para elaboração de procedimentos ou funções em ADL.

Variáveis, definições e comandos em ADL

Um programa em ADL é definido em termos de variáveis, expressões, fluxos de controlo de dados e expressões especiais em ADL. O APSE gera, a partir do comando do utilizador, um ficheiro ADL correspondendo ao projecto de actividades APSE, totalmente de acordo com as regras semânticas e sintácticas da linguagem ADL, pronto a ser utilizado pelo *kernel*.

Ao escrever um programa ADL, os tipos das variáveis não precisam de ser definidos como em outras linguagens de programação como o C ou o Pascal. Um nome de uma variável é simplesmente utilizado numa expressão, o que faz com que seja implicitamente definido, sendo que fica associada com um *Device R&R* previamente definido (de notar que o nome do *Device R&R* deve aparecer algures previamente, utilizando a expressão "define" em ADL). Não existe qualquer limite para o número de *Devices R&R* para cada programa ADL.

Existem duas classes de variáveis em ADL (variáveis normais ou variáveis de estado da aplicação). As variáveis de estado da aplicação são aquelas responsáveis por controlar o estado global do ICS, determinando se o programa está a funcionar, em *idle* ou parado. As variáveis de estado da aplicação possuem dados que reflectem

o estado da aplicação, o que neste caso corresponde aos dados de *input* recebidos via *ethernet*.

As variáveis normais são utilizadas como variáveis auxiliares para serem utilizadas nas expressões ADL. As variáveis normais são utilizadas para auxiliar a construção de fluxos de controlo e expressões.

Ambas as variáveis são definidas aquando da sua primeira utilização. As variáveis obedecem à seguinte convenção:

- O nome das variáveis normais deve começar por uma letra, seguida de uma combinação de zero ou mais letras, dígitos, *underscores* ou pontos.
- O nome das variáveis de estado da aplicação obedecem à mesma convenção que as variáveis normais, com a excepção de terem de começar com a sequência de caracteres "Appstate_".

Todas as variáveis escalares são representados internamente como sendo valores de vírgula flutuante em C e não existe nenhum requisito especial para variáveis definidas pelo utilizador. O tipo da variável é determinado consoante a sua atribuição inicial.

As expressões ADL são representadas e avaliadas como um programa feito em linguagem C. No entanto apenas um subconjunto dos operadores existentes no C é que existem na linguagem ADL.

A selecção é a única forma de controle de fluxo em ADL, utilizando expressões do tipo "if", representadas e avaliadas como as expressões em C. Não existe qualquer tipo de construção para efectuar ciclos.

As expressões "define" são obrigatórias no início do programa para:

- O componente *Data Distributor* do *kernel* saber que outros componentes (*Devices R&R*) estarão presentes e em que máquinas esses módulos supostamente irão correr.
- Definir os *output channels* que estão disponíveis para a geração de recomendações

Um excerto de código em ADL pode ser visto na figura 3.2.

3.3.2 A linguagem PDL

A linguagem PDL pode definir-se como uma extensão da linguagem LISP. Assim sendo, desenvolver um programa em PDL é bastante semelhante ao desenvolvimento de um programa em LISP, com o adicionar de alguns elementos extra. Essa extensão consiste em 4 macros: *defaction*, *defstate*, *defplan* e *def-initstate*, e ainda um conjunto de operadores. O uso destas macros e de alguns dos operadores é ilustrado no seguinte exemplo de uma especificação PDL.

```
define host application rcu090-0;
define device appstate rcu090-0 rcu090-0;

define mode screen      "message_normal";
define mode screen      "message_warning";
define mode screen      "message_emergency";

/*****************************************************************************/
/* Use Case Name: InitOutChan                                                */
/*                                                                           */
/* Description:                                                              */
/* Initialise output channel                                                 */
/*****************************************************************************/

//set log off
if ( first == 0 )
{
        recommend mode screen "message_normal";
        first = 1;
}

/*****************************************************************************/
/* "Declare" local variables                                                 */
/*****************************************************************************/

first = first + 0 * appstate;


/*****************************************************************************/
/* "Declare" application state variables                                     */
/*****************************************************************************/

AppState_Airspeed = AppState_Airspeed + 0 * appstate;
AppState_VerticalSpeed = AppState_VerticalSpeed + 0 * appstate;
AppState_Height = AppState_Height + 0 * appstate;
```

Figura 3.2: Excerto de código ADL

Esta especificação consiste apenas num único e simplificado plano, que especifica as acções que o piloto deve tomar para colocar o motor em andamento. De acordo com este plano, as acções do piloto dependem do facto de o tanque de combustível se encontrar ou não cheio. Se o tanque se encontrar cheio, o piloto pode ligar o motor sem realizar qualquer outra acção. Caso contrário, terá de enchê-lo. O plano trata também da situação em que o piloto tenta ligar o motor quando o tanque não se encontra cheio. Nesta situação, uma mensagem de aviso será despoletada.

Planos, planeamento e reconhecimento de planos

Um plano é uma especificação do que um operador deve fazer para conseguir realizar uma determinada tarefa. Isto envolve definir as acções a executar assim como as contingências aplicáveis à execução eficiente das ditas acções. Estas contingências influenciam a sequência e o escalonamento das acções, assim como influenciam também o estado da aplicação em que essas acções são executadas.

Os planos são utilizados para fazer planeamentos e reconhecimento de planos. Um planeamento é o processo de encontrar e executar um plano de forma a executar uma determinada tarefa. Isto envolve o refinamento de um ou mais planos de modo a definir uma sequência de acções que posteriormente serão executadas. O reconhecimento de planos envolve o inverso, em termos do que é oferecido e deve

```
( defaction FillUpTank StartMotor )
( defstate Tank string )
( defplan StartPlane ()
   ( sequence
      ( if ( not ( equal Tank "full" ) )
         ( sequence
            ( repeat
               ( sequence
                  ( output "You need to fill up the tank" )
                  ( action StartMotor )
               )
            )
            ( action FillUpTank )
         )
   )
   ( action Start Motor )
  )
)
  ( def-initstate init ( call StartPlane ) tank "empty" )
```

Figura 3.3: Excerto de código PDL

ser computado. É o processo de inferir quais os planos que estão a ser executados a partir da ocorrência de acções. Os planos inferidos representam hipóteses para explicar as acções.

A relação entre planos e sequências de acções é de muitos para muitos. Um plano pode definir diferentes sequêcias de acções de modo a ser cumprida uma determinada tarefa e podem existir diferentes planos associados a uma mesma sequência de acções.

Conceitos básicos de PDL

Em PDL, os planos são definidos em termos de acções, variáveis e expressões de planos. Estas noções são brevemente descritas em seguida:

- Acções - são as primitivas, a base da construção de planos. As acções são de 2 tipos: acções de entrada (*input*) e directivas. Acções de entrada são acções que são inseridas e sujeitas ao reconhecimento de planos. Estas acções podem incluir parâmetros. As directivas são acções que são executadas pelo motor de inferências durante o reconhecimento de planos. Elas permitem controlar o processo de reconhecimento de planos e os *outputs* dos planos durante o reconhecimento de planos.

- Variáveis - são utilizadas para guardar valores utilizados durante o reconhecimento de planos. A linguagem PDL oferece mecanismos para declarar variáveis, para atribuir valores a variáveis e para aceder aos valores das variáveis. As variáveis assumem a forma de parâmetros de acções e planos, variáveis de estado, variáveis auxiliares introduzidas utilizando o bloco LET e duas variáveis especiais, `|action|`e `|time|`. A variável `|action|`contem o nome

da última *action* introduzida no PR e a variável `|time|`contem a data dessa *action*.

- Expressões de planos - definem o modo como os planos são compostos por acções, restrições e outros planos.

3.4 O desenvolvimento do APSE

Esta secção descreve em detalhe o trabalho realizado no âmbito do desenvolvimento do APSE. Aqui são relatadas as opções tomadas, as razões para as tomar, no fundo, a descrição de todo o trabalho.

Ao ser responsabilizado por esta parte do projecto, foi-me dada alguma liberdade para o seu desenvolvimento. Tecnologias a usar, linguagens, metodologias, tudo isso foi escolhido por mim, tendo em conta as necessidades da empresa inglesa e aquilo que nos tinhamos proposto fazer.

3.4.1 Recolha e análise de requisitos

Assim sendo, a primeira fase do projecto consistiu na recolha e análise de requisitos. Durante este período, uma grande quantidade de *mails* foram trocados com os parceiros ingleses. No fundo, eles aqui podem ser vistos como nossos clientes, por isso, como em qualquer fase de desenvolvimento de *software* comercial, a necessidade de uma recolha apurada de requisitos impera. Ao obter uma recolha detalhada de requisitos, a necessidade de eventuais modificações é minimizada, e ao serem desde já esclarecidos alguns pontos, não será necessário voltar a discutir certos aspectos.

Tendo isto em conta, a análise de requisitos foi dividida em 3 fases distintas: as fases de análise de requisitos de alto nível, de médio nível e de baixo nível.

Requisitos funcionais do APSE

Esta secção descreve os requisitos funcionais do APSE. O conceito fundamental por trás do APSE é o *Operator Support Use Case*, que descreve:

- Uma sequência de actividades
- Potenciais erros que o operador (neste caso, o piloto) pode efectuar e que possam ser aplicáveis nessa sequência de actividades
- Aconselhamento preventivo e reactivo e outros *outputs* que podem ser gerados pelo ICS

O APSE oferece suporte no que diz respeito à definição de casos de uso e suporta também as actividades relativas à programação do ICS:

- Provisão de um interface gráfico de uso amigável para o utilizador, fácil de utilizar, que permite a definição de objectivos de tarefas principais e a decomposição das mesmas em tarefas de baixo nível, de modo a ser possível detalhar o requerido apoio ao ICS. Isto inclui:
  - Uma base de dados interna com ligações de referência entre todos os objectos de um projecto de actividades APSE, querendo isto dizer que uma alteração efectuada em um objecto qualquer pertencente ao projecto se propagará para os restantes elementos do mesmo projecto.
  - Inclusão de opções para visualizar e filtrar informação, de modo a facilitar a definição de casos de uso.
  - Funcionalidades de gestão de toda a informação relativa a um projecto de actividades APSE (adição, edição e remoção), com total *error-checking*.
  - Funcionalidades para carregar e salvar informação de/para dispositivos de armazenamento, de modo a poder reutilizar tarefas em diferentes projectos.
  - Inclusão de um mecanismo de *logging*, de modo a manter um historial de todas acções efectuadas pelo programador do ICS, assim como a provisão de avisos e erros quando uma acção errada é efectuada.
- Mecanismos para identificar riscos e para suporte à tripulação, incluindo a forma como esse suporte é feito, tipos de alerta e mensagens de alerta, de modo a sucessivamente atingir os objectivos da tarefa. Isto inclui:
  - Decisão de que tipo de suporte o *kernel* requere (ADL ou PDL) para um caso de uso em particular, baseado na informação fornecida pelo utilizador, particularmente a associação entre casos de uso e estados.
  - Janelas de diálogo com opções de modo a permitir ao utilizador a criação de fluxos de dados descrevendo reconhecimento de planos para a totalidade do projecto de actividades APSE, assim como para um estado particular da aplicação, encapsulado sob a sintaxe da linguagem PDL.
  - Verificação sintáctica da correcção do código ADL e PDL editado pelo utilizador e suporte para um determinado caso de uso.
  - Inclusão de mecanismos para visualizar e filtrar informação, de modo a facilitar a definição de riscos e suporte para um determinado caso de uso.
- Fornecimento de mecanismos para gerar código a ser carregado pelo *kernel*. Isto inclui:

- Geração de um ficheiro de código ADL referente a toda a actividade, totalmente de acordo com as regras sintácticas da linguagem e pronto a ser utilizado pelo *kernel* (após compilado).
- Geração de um ficheiro de código PDL referente a toda a actividade, totalmente de acordo com as regras sintácticas da linguagem e pronto a ser utilizado pelo *kernel* (após compilado).
- Geração de um ficheiro de parâmetros externos, com o mapeamento entre as variáveis do ICS e as variáveis da aplicação (neste caso, variáveis dos ficheiro .bse –ficheiros que descrevem as interfaces entre os diversos módulos do projecto), de modo a serem utilizados no ficheiro de interfaces do *kernel*.
- Geração de um ficheiro com a definição das acções, para ser utilizado no ficheiro de interfaces do *kernel*.
- Mecanismos de gestão de configuração, de modo a guardar diferentes versões dos ficheiros gerados e permitir que seja possível reverter para a informação antiga quando necessário.

3.4.2 Design e implementação

Após a análise de requisitos, seguiu-se a fase de desenho. Aqui começaram a ser tomadas as decisões acerca das ferramentas a usar e do modo como seria estruturada a aplicação. A estrutura da aplicação é apresentada na figura 3.4

O código da aplicação APSE é composto por diversos pacotes principais, cada um deles contendo um conjunto bem definido de funcionalidades. Esses pacotes são:

- *Package* Core – este *package* contem o núcleo do APSE, os principais elementos necessários para que a aplicação preencha os seus requisitos. Contem as principais classes necessárias à implementação do APSE e a função de gerar código ADL e PDL assim como a geração dos outros ficheiros necessários.
- *Package* GUI – este *package* contem todo o código relativo ao interface gráfico com o utilizador. Isto inclui todas as janelas de diálogo que são utilizadas no APSE. Inclui também o código relativo à elaboração dos diagramas de fluxo de dados e as classes necessárias para gerar os objectos utilizados nesses diagramas. Contem também os mecanismos para impressão de código e diagramas.
- *Package* XML –este *package* é responsável por fazer a leitura e o parsing dos ficheiros XML de configuração aquando da inicialização do APSE. Recolha toda a informação desses ficheiros e transforma parte dela em código Java.

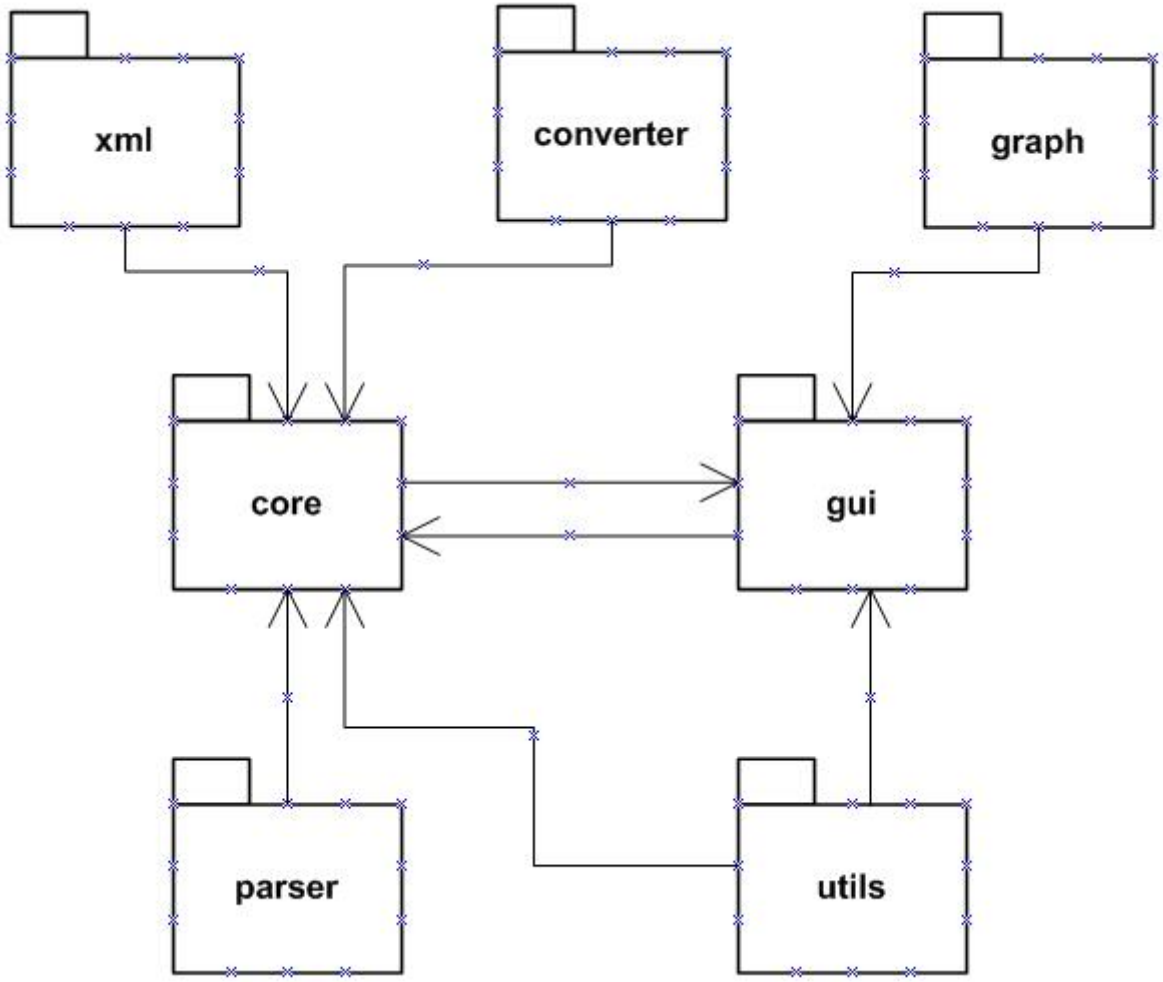

Figura 3.4: Estrutura do APSE

- *Package* Parser -este módulo contem todo o código responsável pela geração de código ADL e PDL a partir dos diagramas de fluxo de dados e os *parsers* de ambas as linguagens.
- *Package* Graph -este módulo contem todo o código responsável pela manipulação dos diagramas de fluxo de dados.
- *Package* Converter -este módulo é responsável por transformar código PDL infixo em código PDL prefixo.
- *Package* Utils -este módulo contem diversas utilidades, não específicas de qualquer um dos outros módulos.

Nas seguintes subsecções são apresentados com mais detalhe os pacotes principais do APSE.

Package Core

Este *p*ackage nuclear do APSE está dividido em *code elements*, geradores de ficheiros de código e o *activity project*. A figura 3.5 representa o modelo UML deste pacote.

- *Code elements* - todos os componentes seguintes são indispensáveis ao desenvolvimento de um completo projecto APSE. São verdadeiramente eles os elementos, as noções mais importantes que necessitam estar presentes aquando da utilização do APSE. Um projecto APSE é basicamente um conjunto dos seguintes itens, ligados uns aos outros de um modo lógico:

  - *Collaborating elements* - um *collaborating element* refere-se a um ficheiro XML contendo informação sobre diversas variáveis. Cada *collaborating element* está relacionado com um ficheiro BSE. Um ficheiro BSE é um ficheiro que descreve uma interface entre dois módulos do projecto. Assim sendo, cada *collaborating element* referencia uma das interfaces entre os módulos do projecto. Isto serve para que ao ICS possa ser oferecida informação relativa a um determinado interface. Cada interface possui um conjunto de variáveis que são preenchidas e é esta informação que é transferida de um módulo para outro. O ICS tem, portanto, a capacidade de utilizar variáveis destas interfaces - as *external variables*.
  - *User variables*
    - *External variables* - todos os módulos do projecto, aparte do APSE, comunicam entre si através de uma rede *ethernet*. É utilizada uma biblioteca de comunicação de um dos parceiros, que utiliza o protocolo UDP e envia mensagens em *broadcast*. Essas mensagens, podem ser recolhidas por qualquer módulo que esteja à escuta e necessite de informação contida nessas mensagens. As *external variables* referem-se exactamente às variáveis que são recolhidas da rede, que são geradas e controladas por outros módulos e que servem de *input* para o ICS proceder ao seu funcionamento em pleno.
    - *Internal variables* - estas variáveis referem-se às variáveis internas ao APSE, ou seja, variáveis que são declaradas e utilizadas apenas no âmbito do APSE e não são partilhadas nem geradas por nenhuma outra aplicação que não o ICS. Podem ser vistas quase como as variáveis auxiliares de uma função, enquanto que as variáveis externas podem ser comparadas às variáveis globais de um programa.
  - *States* - um *state* é, como o nome sugere, um estado para o qual o ICS pode transitar. Por exemplo, em termos de aviação, um estado pode ser visto como uma das fases de voo: descolagem, cruzeiro, descolagem, etc.

As decisões tomadas pelo ICS, as recomendações à tripulação são sempre tomadas tendo em conta a fase de voo presente.

  - *Use cases* - um *use case*, neste contexto, refere-se a uma situação ou acção para a qual o ICS oferece suporte. Este tipo de situações pode ser tão diversificado como por exemplo monitorizar a descolagem ou verificar se a velocidade que o avião possui se adequa à fase de voo em que se encontra ou ainda verificar se os *flaps* se encontram na posição que seria desejável para a situação presente.
  - *Actions* - uma *action* consiste numa acção ou procedimento efectuado pela tripulação ou pelo avião. Uma *a*ction é algo que é efectuado sob um determinado *state*. Esta acção pode ser, por exemplo, o piloto configurar os *flaps* consoante a fase de voo, ou o avião ter entrado na área de intercepção ILS.
  - *Output channels* - um *output channel* refere-se ao canal ou meio através do qual a tripulação é notificada que um aviso do ICS foi emitido. Esse canal ou meio pode ser um aviso aural, letras vermelhas no PFD (*Primary Flight Display*) ou ND (*Navigation Display*), por exemplo, ou uma combinação de vários canais.

- *Activity project* - esta classe em particular contem toda a informação relativa a um projecto APSE. Contem o conjunto completo de toda a informação relativa a um projecto: *user variables, states, use cases, actions, collaborating elements* e *output channels*. Contem também a informação relativa aos diagramas associados a um determinado projecto. Basicamente, existe apenas um objecto desta classe por cada projecto APSE, na qual estarão incluídos todos os detalhes relativos ao mesmo. Pode ser visto como a base de dados de um projecto em particular.

- Geradores de ficheiros de código - são responsáveis por gerar:

  - *ADL code file* - ficheiro contendo todo o código ADL de um projecto APSE. Esta geração de código ADL é efectuada recorrendo à utilização de funções do *package parser* e é fornecido um eficiente mecanismo de controlo de versões. Este mecanismo renomeia um ficheiro já existente quando uma nova geração desse ficheiro é efectuada. O novo nome contem a data da última modificação do ficheiro antigo.
  - *PDL code file* - ficheiro contendo todo o código PDL de um projecto APSE. Esta geração de código PDL é efectuada recorrendo à utilização de funções do *package parser* e é fornecido um eficiente mecanismo de controle de versões. Este mecanismo renomeia um ficheiro já existente

quando uma nova geração desse ficheiro é efectuada. O novo nome contem a data da última modificação do ficheiro antigo.

- *Action file* - este ficheiro contem informação sobre todas as acções de um determinado projecto. É gerado sob a forma de um ficheiro pronto a compilar em linguagem C, com a devida sintaxe e formatação. O mecanismo de controlo de versões também aqui foi implmentado.
- *Use case files* - um ficheiro para cada caso de uso de cada projecto, de modo a que os casos de uso utilizados num determinado projecto possam ser facilmente exportados para outros projectos APSE.
- *Interface file* - este é o ficheiro que é utilizado para salvar e carregar um projecto na totalidade. Consiste basicamente no carregamento de toda a base de dados do projecto para um ficheiro. O mecanismo de controlo de versões foi também implementado para este ficheiro.

***Package* GUI**

Este *p*ackage é composto pelos seguintes elementos:

- *APSE main form* - o *main form* do APSE refere-se ao *main container* da *applet* APSE, ou seja, a janela principal. Este é o *container* externo da *applet*, contendo os menus da barra de ferramentas. Ver figura 3.6.
- *Project details* - esta classe refere-se ao conteúdo do *main form*. Contem toda a informação contida no *main form*, excluindo os menus da barra de ferramentas, ou seja, contem toda a informação relativa ao próprio projecto APSE. Para ele são carregadas todas as informações de um *activity project*.
- *Dialog windows* - todas as outras janelas que são mostradas de cada vez que um botão é pressionado ou que alguma acção é requerida.
- Diagramas de fluxo de dados - é importante distinguir estes ficheiros dos outros ficheiros pertencentes à *applet*. Estes são também ficheiros pertencentes à *applet* mas são responsáveis pelo desenho dos diagramas de fluxo de dados e pelas ferramentas responsáveis por alterar ou manipular esses fluxos.
  - Diagrama de estados - este é o diagrama principal de um projecto APSE. Nele está representado o fluxo completo de dados de um projecto APSE. Aqui todos os *states* e *actions* estão presentes e são ligados da forma pretendida pelo operador.

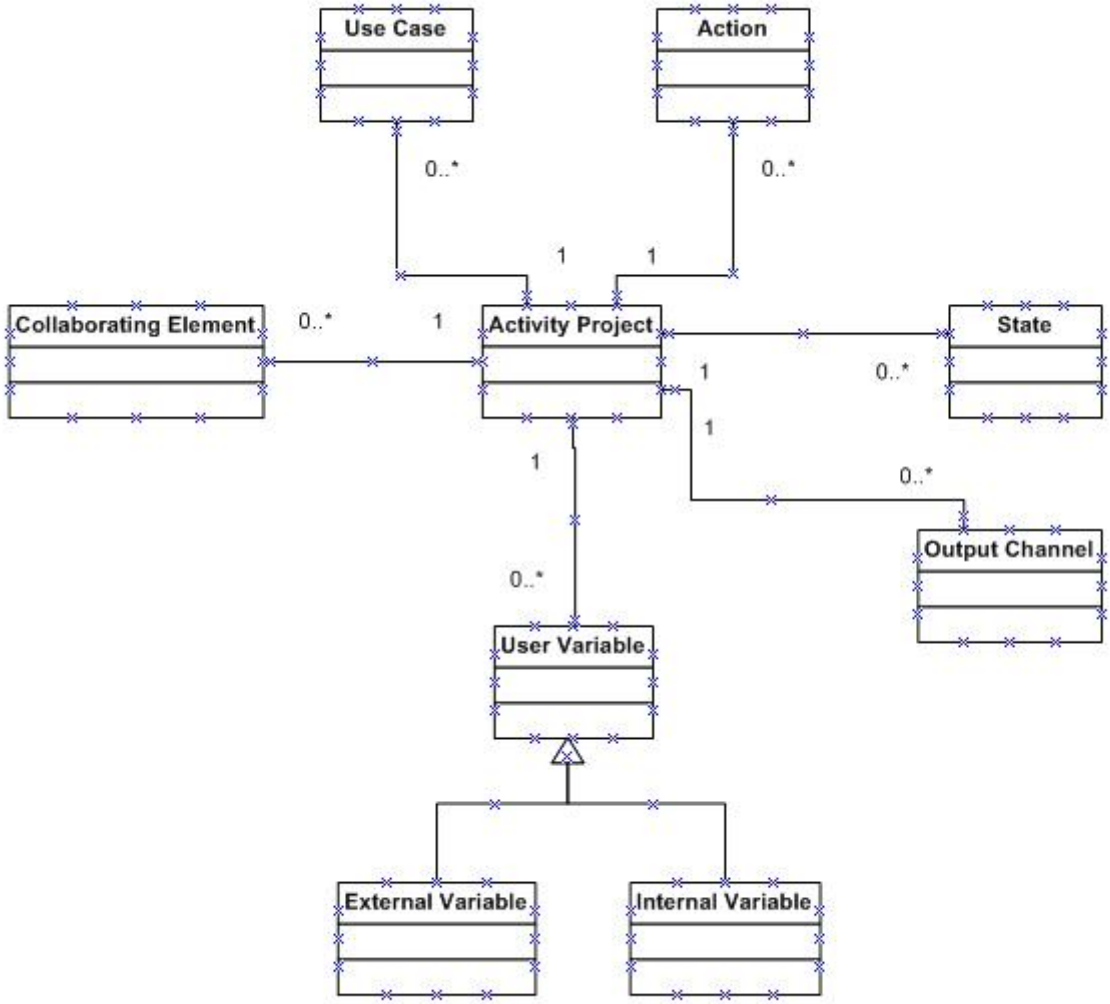

Figura 3.5: Parte do modelo de classes UML do referente ao *package* core

  - Diagrama de casos de uso - para cada estado do projecto, pode ser descrito o que acontece no interior desse estado, sob a forma de *use cases*, *actions* e até de outros *states*.

- Graphical Objects - estas são as classes que serão responsáveis pela geração e manipulação de objectos necessários à representação de fluxos. Cada um deles tem um símbolo e uma cor correspondente e simbolizam *states*, *actions* e *use cases*, para além de um estado inicial.

  - Rectângulo de cantos arredondados - esta figura simboliza um *use case*. Possui a cor azul clara e tem a forma de um rectângulo com cantos arredondados.
  - Elipse - esta figura representa um *state*. Contem a cor branca e tem a forma de uma elipse.
  - Rectângulo -esta figura simboliza uma *action*. Possui a cor verde e tem a forma de um rectângulo.

***Package* Parser**

Este componente do APSE contem os ficheiros das 3 gramáticas utilizadas e o código dos correspondentes analisadores sintácticos, gerado automáticamente por um gerador de compiladores - o Javacc, explicado mais adiante.

- As gramáticas - dada a necessidade de reconhecimento de linguagens e também de geração das mesmas, foi necessário desenvolver as gramáticas relativas às linguagens utilizadas, ADL e PDL, sendo que a linguagem PDL, devido a requisitos do cliente, teve de ser desenvolvida utilizando quer notação infixa, quer notação prefixa. Estas gramáticas servem sobretudo para verificar se um determinado pedaço de código pode ou não corresponder à linguagem em causa. No entanto, foram também auxiliares para o desenvolvimento de código, para ambas as linguagens.

  - Gramática de PDL com notação prefixa
  - Gramática de PDL com notação infixa
  - Gramática de ADL

***Package* XML**

Existem 2 elementos principais deste *package*:

- *Elements Scanner* -esta classe é responsável pela leitura dos ficheiros XML que contêm os *collaborating elements* predefinidos. Além de fazer a leitura do

ficheiro XML, ainda mapeia o que é lido desse ficheiro com as *external variables* do projecto.

- Options Scanner –esta classe é responsável por ler o ficheiro XML que contem as principais opções do APSE. Ao inicializar o APSE, este ficheiro é lido e contem as principais opções de configuração do APSE, como as directorias de destino dos ficheiros gerados ou as directorias onde encontrar os *collaborating elements* correspondentes ao projecto.

***Package* Graph**

Este módulo contem todo o código responsável por verificar se há ciclos nos grafos e se todos os nós possuem uma ligação ou se não existe nenhum tipo de inconsistências ou incongruências nos grafos. Para além disso, os grafos têm um papel importante na geração de código, pois consoante o tipo de ligação existente entre 2 componentes do grafo, o código será gerado de modo diferente. Aqui estão contidos, entre outras funcionalidades, algoritmos que verificam se o mesmo elemento está contido repetidamente no mesmo grafo (o que não é permitido), algoritmos para verificar se existem estados inatingíveis, etc.

***Package* Converter**

este módulo é o único desenvolvido em linguagem C, ao contrário de todos os outros módulos, desenvolvidos em Java. Consiste em um conversor de linguagem PDL, que recolhe a informação introduzida pelo utilizador sob a forma de notação infixa e transforma-a em notação prefixa, de modo a poder ser inserida directamente no *kernel*. A razão principal que levou a utilizar a linguagem C deveu-se ao facto de a Skysoft já ter desenvolvida uma ferramenta com uma função semelhante à requerida nessa linguagem e foi possível aproveitar parte da lógica subjacente.

3.4.3 Testes

Aquando da realização deste relatório, a fase de testes ainda decorre, por isso é impossível divulgar qualquer tipo de informação acerca deste assunto.

3.5 Ferramentas e tecnologias utilizadas

Nesta secção são apresentadas as tecnologias necessárias e as ferramentas utilizadas para o desenvolvimento do APSE. Também são apresentadas as principais razões que levaram a optar por essas mesmas tecnologias e ferramentas.

- Linguagens utilizadas:

  - Linguagem de modelação UML.
  - Linguagem de programação Java.
  - Linguagem de programação C.
  - Linguagem XML.
  - Linguagem do gerador de *parsers* javacc - semelhante ao Java, mas com o adicionar de algumas macros e algumas alterações sintácticas.
  - Linguagens de *output* de código - ADL e PDL

- Ferramentas utilizadas:

  - Microsoft Visio - para desenvolvimento do modelo UML
  - Microsoft Visual Studio .NET
  - Netbeans - para desenvolvimento do código da *applet*.
  - Eclipse - para desenvolvimento do restante código
  - CVS - sistema de controle de versões
  - O javacc - Java Compiler Compiler - um gerador de *parsers* para ser utilizado com aplicações Java. Ou seja, é uma ferramenta que lê a especificação presente numa gramática e converte-a num programa Java que pode reconhecer e fazer corresponder elementos aceites na gramática.

3.6 Um exemplo da execução do APSE

Nesta secção é demonstrada parcialmente a aplicação APSE em execução. As figuras são meramente exemplificativas, somente para dar a conhecer parcialmente a aplicação. O intuito desta secção não é descrever detalhadamente a execução da aplicação, apenas fornecer uma visão global das funções mais importantes do APSE.

Na figura 3.6 é apresentada a janela principal da aplicação. Esta é composta por:

- Barra de menus –aqui são apresentados as opções do APSE. Os menus incluem as tradicionais opções para carregar e salvar projectos, no menu *File*; incluem igualmente as opções para geração de ficheiros, no menu textitProject; o menu *Tools* contem opções como ver o *Log* da execução do APSE ou alterar as opções da aplicação, como as directorias onde os projectos serão guardados, etc. O menu *Print* contem as opções referentes à impressão de código gerado pelo APSE ou de diagramas desenvolvidos no mesmo; e finalmente, o menu *Help*, onde são apresentadas as opções de ajuda relativas à utilização da aplicação.

- Detalhes do projecto - correspondem ao restante espaço da janela principal. Assim é possível obter uma visão de alto nível daquilo que é o projecto correntemente carregado.

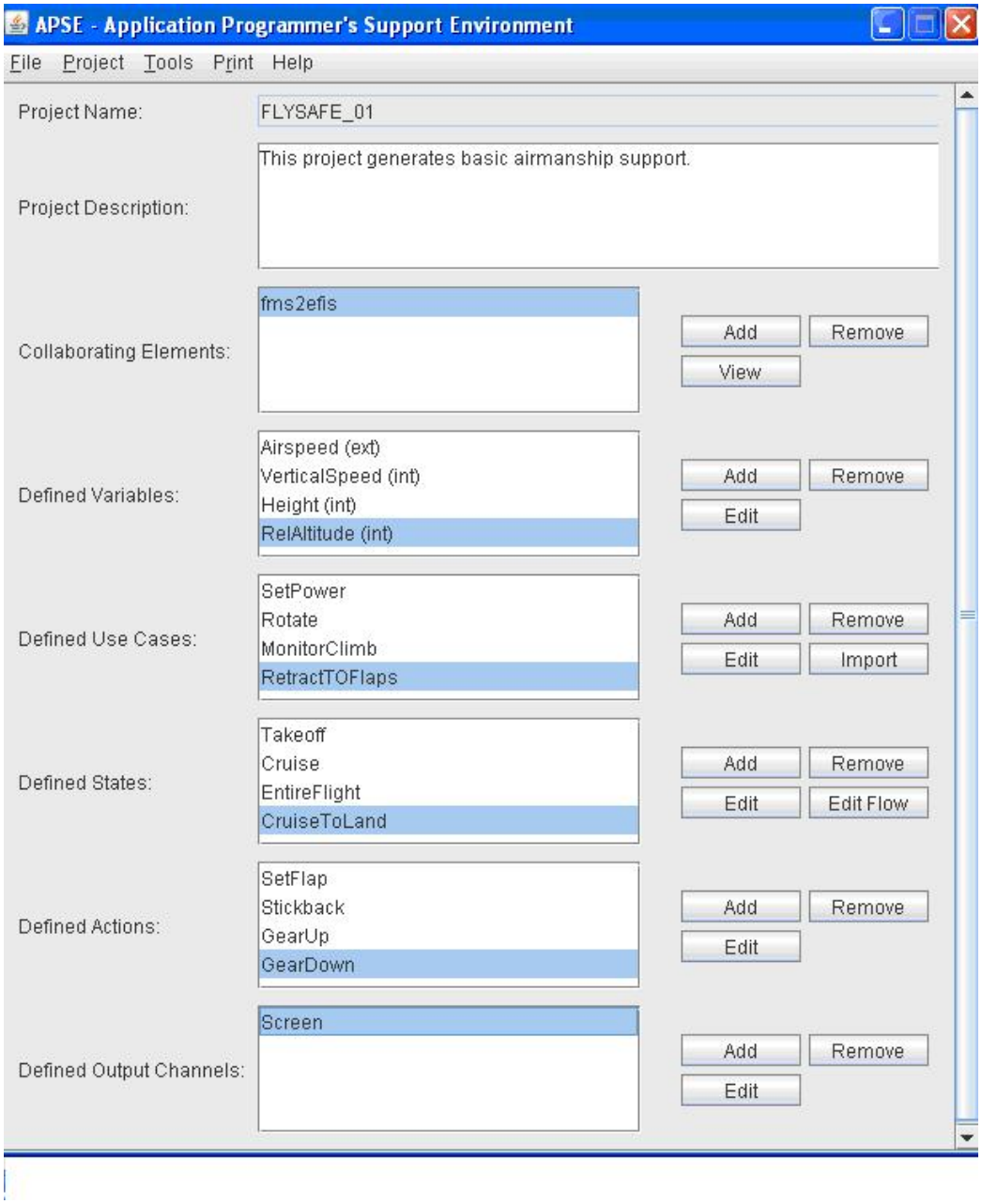

Figura 3.6: Main form do APSE

A figura 3.7 demonstra um dos modos de introduzir código PDL oferecida pelo APSE. Este modo de introdução de código permite a inserção manual do mesmo ou a inserção recorrendo ao auxílio dos diversos botões. Estes botões permitem a inserção de excertos de código PDL. Os botões do lado direito, em cima, têm a capacidade de gerar diversos tipos de expressões válidas na linguagem PDL. Os botões situados

por baixo da caixa de texto são os operadores permitidos em PDL. A caixa *variables* permite que seja feita a selecção de variáveis e que as mesmas sejam introduzidas no código PDL. Os botões do lado direito, em baixo, permitem avaliar a validade do código introduzido, isto é, se respeita ou não as regras PDL; permitem ainda guardar este código ou cancelar a introdução. Esta introdução é feita utilizando a notação infixa, apesar do ICS necessitar que o código PDL seja fornecido de forma prefixa. Esta conversão é feita recorrendo ao módulo *converter* do APSE.

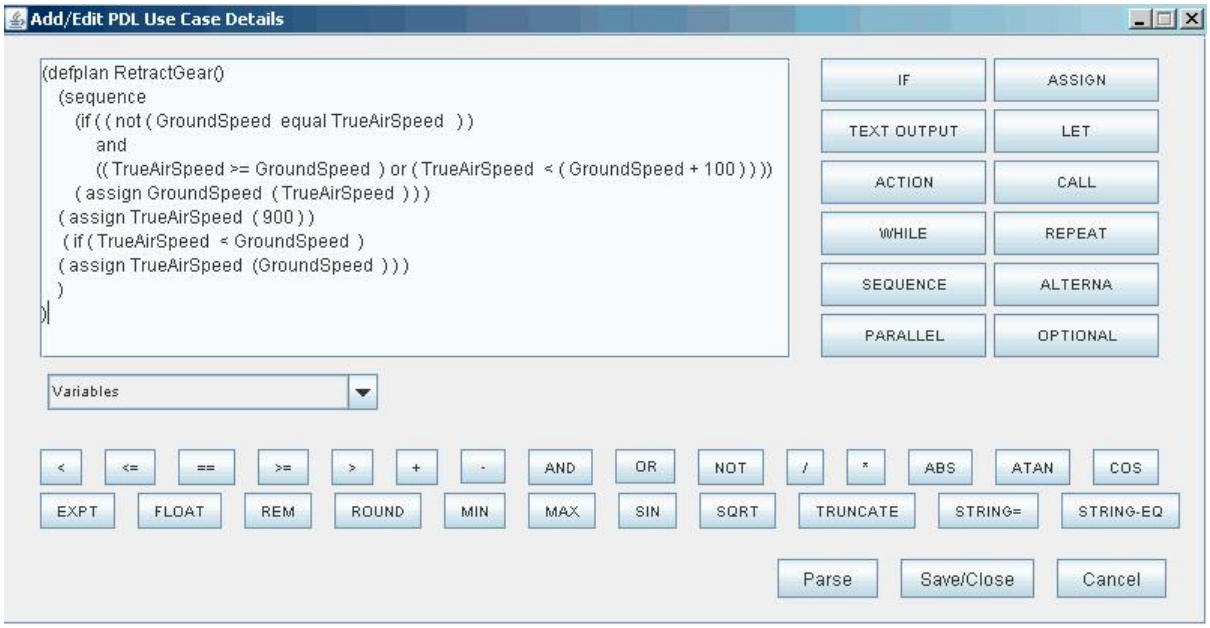

Figura 3.7: Introdução de código PDL

A figura 3.8 mostra o resultado da conversão do código PDL da figura 3.7 de notação infixa para notação prefixa. É um excerto do ficheiro de código PDL gerado pelo APSE.

```
;;;
;;; Use Case Name RetractGear
;;;
;;; Description
;;;

(defplan RetractGear()
        (sequence
                (if(and(not(equal GroundSpeed TrueAirSpeed))
                        (or(= TrueAirSpeed GroundSpeed)( TrueAirSpeed(+ GroundSpeed 100))))
                (assign GroundSpeed TrueAirSpeed)))
        (assign TrueAirSpeed 900))
        (if( TrueAirSpeed GroundSpeed)
                (assign TrueAirSpeed GroundSpeed)))
        )
)
```

Figura 3.8: Excerto de código PDL prefixo gerado a partir da figura 3.7

A figura 3.9 demonstra o fluxo principal, ou seja, o diagrama de estados, com

os estados presentes no diagrama expandidos. Como cada estado tem associado a si um fluxo de dados, cada estado presente no diagrama pode ser substituído pelo respectivo diagrama, de modo a oferecer uma visão mais global de todo o projecto APSE. Esta janela apresenta duas partes distintas, o diagrama em si e a parte inferior com os botões para realizar operações sobre o fluxo. Esses botões permitem efectuar operações como ligar 2 objectos, expandir estados, imprimir o diagrama, entre outros.

A figura 3.10 demonstra o código PDL gerado a partir dos diagramas da figura 3.9.

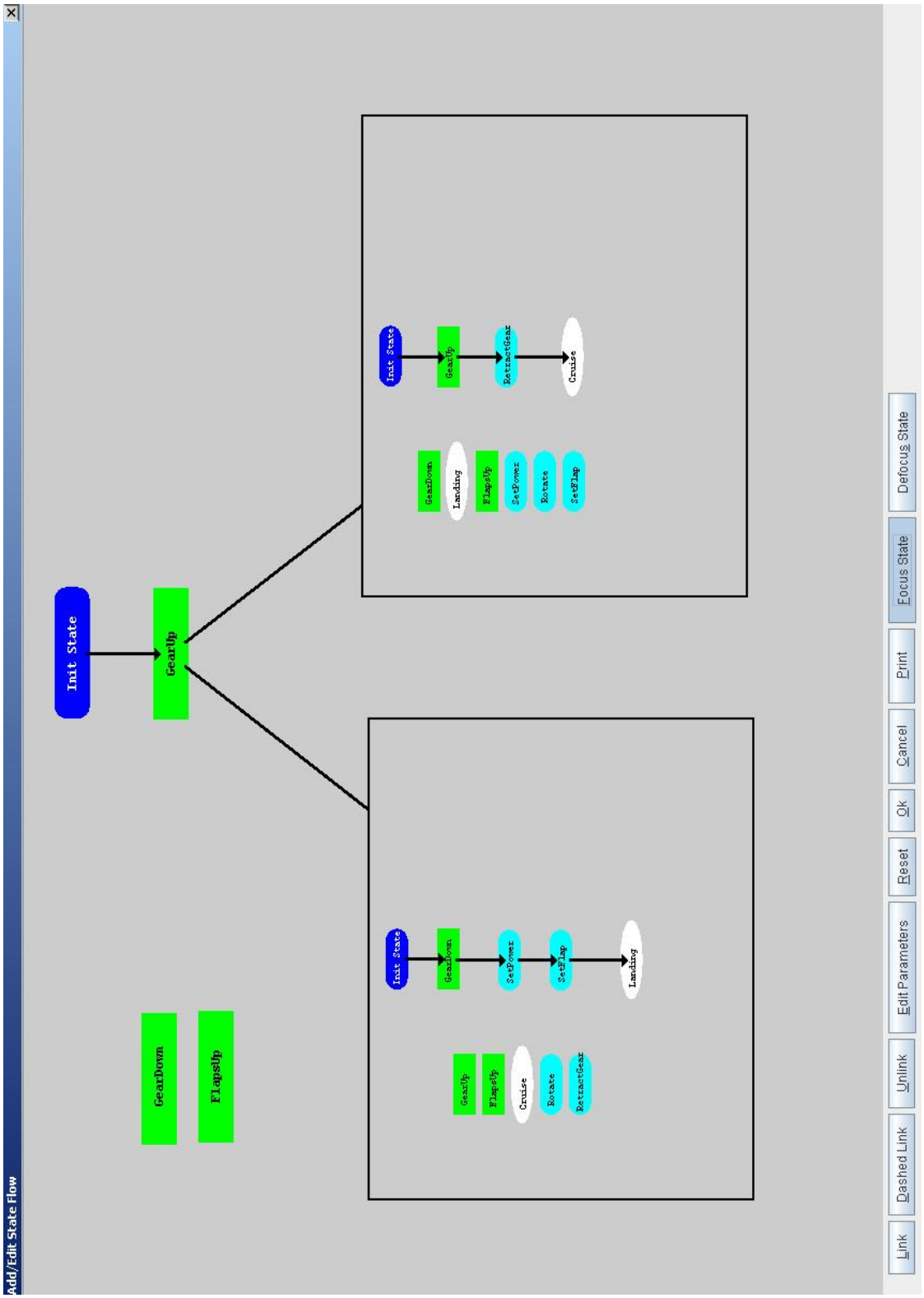

Figura 3.9: Fluxo de dados principal

```
;;;
;;; State Name: Cruise
;;;
;;; Description:
;;;

(defplan Cruise()
                (sequence
(action GearUp)
(call RetractGear)
(call Cruise)
)
)


;;;
;;; State Name: Landing
;;;
;;; Description:
;;;

(defplan Landing()
                (sequence
(action GearDown)
(call SetPower)
(call SetFlap)
(call Landing)
)
)


;;;
;;;       MAIN PLAN ;;;
;;;

(defplan main()
        (sequence
                (action GearUp)
(parallel
(call Cruise)
(call Landing)
)
                (block)
        )
)
```

Figura 3.10: Excerto de código PDL gerado pelo APSE a partir do diagrama de fluxo de dados principal

Capítulo 4

Conclusão

4.1 Sumário do trabalho realizado

Neste trabalho, foi desenvolvida a aplicação APSE, um módulo de apoio à programação do ICS, com o objectivo de facilitar a programação do mesmo, de modo fácil e intuitivo. Esta aplicação estava inserida num projecto da Comissão Europeia, o projecto FLYSAFE, de grande complexidade, e com um número bastante elevado de parceiros. O APSE consiste numa ferramenta que permite ao utilizador programar o kernel do ICS sem ter qualquer tipo de noção da sua linguagem e sem sequer alguma vez visualizar parte do código que é gerado automaticamente. Facilita grandemente a inserção de dados no ICS e faz com que o operador encarregado dessa inserção se possa abstrair totalmente do código que posteriormente é gerado, focando-se apenas no fluxo lógico de dados que pretende introduzir. Durante este ano, as principais fases do processo de desenvolvimento de software foram ultrapassadas e cumpridas: recolha e análise de requisitos, design, implementação e parte dos testes.

Para além de desenvolver o dito módulo, houve ainda outras pequenas contribuições que efectuei para o mesmo projecto, que me ocuparam uma parcela de tempo relativamente pequena, mas que vale a pena mencionar também. Essa colaboração envolveu pequenas cooperações em alguma documentação ou em determinadas situações em que para a codificação era necessário uma equipa maior.

4.2 Critica

Este projecto serviu para colocar muito do que aprendi durante o meu percurso académico em prática. Para além obviamente da parte técnica, fomentou também a minha capacidade de resposta a constantes novos desafios. Nunca no meu percurso académico tinha sido confrontado com um projecto de tal magnitude e complexidade como o FLYSAFE. De início muitas coisas pareceram confusas: o elevado número de

documentos de especificação e gestão, a quantidade enorme de parceiros externos e todo um mundo totalmente diferente e desconhecido como é o mundo da aeronáutica.

No entanto, tudo isso que parecia confuso acabou por se revelar totalmente necessário e interessante. A quantidade elevada de documentação sempre foi algo que nos foi incutido no percurso académico, mas muitas vezes, quase sempre por falta de tempo, era descurado. Ao ter um cliente exigente como a CE e sendo o projecto tão abrangente, a necessidade de haver uma rígida e complexa documentação, quer técnica, quer de gestão é totalmente imperativa. Sem esse nível de exigência de documentação, o projecto tornar-se-ia caótico e praticamente impossível de ser gerido. Extensivos planeamentos e constantes análises de risco levaram a uma produção massiva de documentos organizativos e planos de mitigação de riscos.

Em relação ao facto de haver um elevado número de parceiros, acabou por revelar-se uma experiência bastante enriquecedora. O contacto com diferentes formas de pensar, com culturas diferentes, com metodologias e práticas muito diferentes das apresentadas no nosso país, incutiu-me um novo modo de analisar os problemas e de tentar encontrar eventuais soluções para os mesmo.

Apesar de o módulo APSE não ter uma ligação directa com a aeronáutica, necessitei de me instruir bastante em relação a esse campo, de modo a compreender toda a nomenclatura e terminologia que envolvia o projecto. O mundo da aeronáutica, até então totalmente desconhecido, revelou-se um mundo muito interessante. Todavia, ao ser totalmente desconhecido, também ao início pareceu um tanto ou quanto confuso, com uma imensidão de termos, tecnologias e procedimentos até então completamente estranhos.

Nunca também tinha tido a necessidade de desenvolver um projecto recorrendo totalmente a uma língua que não a língua materna. Foi algo que trouxe algumas dificuldades inicias. Não por falta de domínio e compreensão do idioma inglês, mas a redacção de documentação técnica numa língua que não a materna, apresenta, de início, algumas dificuldades.

4.3 Trabalho futuro

Para uma aplicação com as características do APSE, é um pouco difícil falar de trabalho futuro. Seria mais fácil falar de eventuais inovações se houvesse uma maior noção do que é o funcionamento do ICS. Na teoria, o desenvolvimento de APSE teria sempre que acompanhar a evolução do ICS, visto estar tão dependente dos requisitos dessa aplicação.

No entanto, apesar de apenas poder especular acerca do funcionamento do ICS, existe a informação de que os ficheiros gerados pelo APSE são depois carregados pelo ICS. Uma sugestão seria, por exemplo, fazer o carregamento directo da informação

do APSE para o ICS, sem ter que recorrer a geração de ficheiros que mais tarde são carregados. Neste caso, ao gerar o código, este seria de imediato enviado para o ICS. Seria então necessário desenvolver o módulo de comunicação entre as duas aplicações, o que levaria a que o APSE e o ICS pudessem quase ser considerados uma só aplicação.

Com a experiência obtida após mais testes com o ICS poderão surgir mais ideias, e a Skysoft tenciona continuar a desenvolver as ferramentas referidas neste tipo projectos. A minha colaboração neles será então feita de modo continuado.

Acrónimos

ACAS Airborne Collision Avoidance System

ADL Application Description Language

ARCHIE A Reliable Computer - Human Interaction Environment

BSE dataBaSE

ESA European Space Agency

GUI Graphical User Interface

ICS Intelligent Crew Support

ILS Instrument Landing System

MEI Mestrado em Engenharia Informática

ND Navigation Display

NG-ISS New Generation Integrated Surveillance System

PDL Plan Definition Language

PFD Primary Flight Display

PR Plan Recogniser

RR Reports and Recomendations

SMGCS Surface Movement Guidance Control System

TAWS Terrain Avoidance Warning System

UDP User Datagram Protocol

UML Unified Modeling Language

VDL VHF Data Link

VHF Very High Frequency

WP Work Package

XML eXtensible Markup Language

Bibliografia

[1] ADL Language Summary, BAE Systems

[2] PDL Language Summary, BAE Systems

[3] APSE Software Requirements Specification, Skysoft Portugal

[4] ICS Description Document, BAE Systems

[5] APSE Software User Manual, Skysoft Portugal

[6] Java API Specification, http://java.sun.com/javase/6/docs/api/

[7] Javacc homepage, https://javacc.dev.java.net/

[8] UML Toolkit, Hans Erik Eriksson and Magnus Penker, John Wiley and Sons, 1998