Faculdade de Engenharia da Universidade do Porto

Modelação e Simulação de um Sistema de Sinalização Ferroviária

Cláudio José da Silva Sagres

Versão 1

Relatório de Projecto

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Prof. João Pascoal Faria

Julho de 2008

Modelação e Simulação de um Sistema de Sinalização Ferroviária

Cláudio José da Silva Sagres

Relatório de Projecto realizado no âmbito do Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Rosaldo José Fernandes Rossetti (Professor Auxiliar)

_____

Arguente: Alberto Manuel Rodrigues da Silva (Professor)

Vogal: João Carlos Pascoal Faria (Professor Auxiliar)

31 de julho de 2008

Resumo

Hoje em dia existe a necessidade de informatizar cada vez mais o controlo do tráfego das linhas ferroviárias. Tratando-se de sistemas críticos e vitais, existem normas que devem ser respeitadas no desenvolvimento das soluções de controlo da sinalização ferroviária, de forma a evitar situações catastróficas. Com o intuito de uniformizar os sistemas europeus foram definidos conceitos para a protecção dos comboios, sendo que, por enquanto, a sua aplicação resume-se apenas às linhas de alta velocidade. Estes conceitos são o ETCS (*European Train Control System*) que é uma componente do outro, chamado ERTMS (*European Rail Traffic Management System*). As normas relativas a sistemas ferroviários definem, entre outros assuntos, directrizes para o desenvolvimento destes mesmos sistemas. Nelas pode-se retirar que deve haver uma rastreabilidade entre todos os documentos desenvolvidos durante o ciclo de vida do sistema. Para a verificação dos requisitos é corrente, para não dizer obrigatório, utilizar a modelação de requisitos. Para este fim existem várias técnicas disponíveis, umas mais utilizadas do que outras, sendo que algumas são gerais como UML (*Unified Modeling Language*) ou B e outras são proprietárias como SCADE. Cada vez mais é utilizado o UML, nomeadamente desde que os sistemas passaram a ser desenvolvidos em linguagens orientadas a objectos. Existem, para esta técnica, várias ferramentas no mercado como Artisan Studio e Rhapsody. Estas ferramentas permitem também testar ou simular os modelos, podendo assim verificar o cumprimento dos requisitos.

Esta dissertação retrata a modelação de um sistema de sinalização para uma linha de baixo tráfego, utilizando a técnica de modelação UML. Os modelos realizados foram desenvolvidos de forma a permitir uma posterior simulação de forma a verificar os requisitos definidos previamente, isto é o bom funcionamento do sistema especificado, assim como a sua segurança em caso de avarias.

Abstract

Nowadays the railway systems are growing up in complexity and exigency. The way to deal with such complexity is to make those systems based in computer methods. Restrictive procedures must to be taken into account to avoid catastrophic situations. All railway signalling solutions must follow Standards that oblige suppliers building safety products. The European Standards for railways were created mostly for train protection. The idea is to create a fully interoperability between European countries.

At the moment the high-speed rail are already implemented and soon all European rail-transports can be done using the same system, no matter for which country we are travelling. One concept is the ETCS (European Train Control System) which is a signalling, control and train protection system designed to replace the incompatible safety systems currently used by European railways, especially on high-speed lines. Another concept is the ERTMS (European Rail Traffic Management System).

The Standards for railway systems are forcing suppliers to make an exigent trace of all requirements along the product lifecycle. In railway, it is strongly recommended the usage of a modelling system to evaluate the consistency of the requirements. There are different languages that permit modelling such as UML, B and SCADE.

The UML is one technique largely applicable, mainly because this is an object-oriented modelling language. There are many tools which are using this technique such us Artisan Studio and Rhapsody. Those tools also allow to test and simulate the models helping to verify if the requirement are fulfilled.

This dissertation reports the development of a modelling interlocking system for Low Traffic Line Signalling, based on UML. All models developed permit a subsequent simulation that verifies the requirements defined previously mainly in a critical situation.

Agradecimentos

Agradeço a todos os meus colegas de trabalho, nomeadamente aos engenheiros Isaac Moreira, Pedro Tavares, José Veiga e Raul Esteves, com quem participei activamente no desenvolvimento (ainda a decorrer) do Sistema de Sinalização ferroviário para linhas de baixo tráfego.

Agradeço igualmente o meu Chefe e Supervisor, o Engenheiro Luís Roborêdo e Castro pelo apoio e pela disponibilidade que mostrou durante a fase do projecto.

Agradeço também ao meu orientador, o Engenheiro João Pascoal Faria pelos conselhos que me deu relativamente ao projecto e dissertação.

Por fim agradeço aos meus pais que me deram a oportunidade e condições para estudar no ensino superior.

Conteúdo

Lista de figuras

Capítulo 1

Introdução

Este primeiro capítulo tem por objectivo apresentar o enquadramento do projecto, e identificar e definir os problemas abordados no mesmo.

1.1 Enquadramento

O projecto relativo a esta dissertação enquadra-se no âmbito de um projecto de desenvolvimento de um sistema de sinalização ferroviária para uma linha de baixo tráfego, que está a ser realizado pelo sector de Sinalização Ferroviária da Efacec Sistemas de Electrónica, S.A.. A investigação e desenvolvimento de soluções para sistemas de sinalização ferroviária, é uma aposta recente deste sector da Efacec SE. Sendo um sistema de segurança crítica e vital, o sistema deve responder a certas normas, tanto na sua concepção como na sua implementação e instalação. Segundo estas normas, o sistema deve responder a um determinado ciclo de vida que vai do seu conceito até ao seu desmantelamento. Dentro deste ciclo, nas fases de desenvolvimento, insere-se a modelação do sistema, que é o assunto desta dissertação.

Este projecto exige portanto bastantes formalidades e conhecimento de normas, nomeadamente daquelas relativas ao domínio ferroviário. Todas as fases do ciclo de vida do sistema devem ser devidamente documentadas. São projectos de longa duração, principalmente se a empresa está a dar os primeiros passos no sector.

1.2 Projecto

O projecto visa portanto modelar o sistema de sinalização ferroviário para uma linha de baixo tráfego. O primeiro passo foi a escolha da técnica de modelação e a empresa optou pelo UML (*Unified Modeling Language*). Para efectuar a modelação foi necessário ter à disposição os requisitos de software e participar nas reuniões de análise HAZOP, que é um estudo sobre as situações potencialmente perigosas que podem afectar o sistema. Esta análise permitiu definir com mais afinco funcionalidades do sistema de forma a garantir a segurança do sistema. Aliás a escolha do UML acabou por ser interessante porque permitiu realizar uma especificação funcional do sistema utilizando o diagrama de casos de utilização, em que cada caso de utilização serviu para a realização da análise de situações potencialmente perigosas.

No entanto o principal objectivo é uma modelação formal e mais extensa portanto era necessário escolher uma ferramenta para realizá-la. Este foi o segundo passo. Depois de uma pesquisa de várias ferramentas para modelação em UML para tempo real, optou-se pelo Rhapsody da Telelogic.

1.3 Motivação e objectivos

Neste tipo de sistemas é primordial efectuar todos os passos de forma rigorosa, isto é, deve-se respeitar todas as fases do ciclo de vida do sistema, como definidas nas normas. Sendo que a modelação do sistema é uma delas, acaba por ser um passo obrigatório. O objectivo desta modelação é de desenhar o sistema de forma a garantir o cumprimento dos requisitos, principalmente aqueles relativos à segurança. A partir dos modelos poder-se-á realizar uma simulação, permitida graças a ferramenta, através do código por ela gerada. Esta simulação permitirá confirmar que o sistema cumpre aquilo que foi especificado.

Portanto para realizar a modelação foi necessário ter os requisitos do software, uma especificação funcional de base, e a análise das situações potencialmente perigosas (SPP). A partir destes dados desenvolveu-se (durante este projecto de fim de curso) uma nova especificação funcional tendo em conta os resultados do estudo das SPP. E foi a partir deste documento que se efectuou a modelação, pois nele estavam descritos todos os passos a seguir para a realização dos casos de utilização, tanto em funcionamento normal, como em funcionamento defeituoso.

O objectivo do projecto consistiu portanto na elaboração de um modelo UML especificando o comportamento de um sistema de sinalização para uma linha ferroviária de baixo tráfego, e na validação do modelo através de simulações suportadas em ferramentas.

1.4 Estrutura da dissertação

Para além desta introdução, este documento contém mais 4 capítulos. No capítulo 2 descreve-se em pormenor o problema, definindo ao mesmo tempo alguns termos técnicos relativos à sinalização ferroviária. Nesse capítulo apresentam-se igualmente os requisitos do sistema, a especificação funcional e a sua arquitectura. No capítulo 3, descrevem-se de forma sucinta alguns aspectos das normas existentes na área da ferrovia, várias técnicas e ferramentas de modelação que podem ser utilizadas para modelar um sistema de sinalização ferroviário, e descreve-se, de uma forma geral, a ferramenta utilizada no projecto. O capítulo 4 refere-se ao desenho dos modelos expondo as razões que levaram às escolhas efectuadas, mostrando alguns dos diagramas mais relevantes e descrevendo alguns aspectos da ferramenta que foram utilizados para a realização das opções tomadas. O capítulo 5 conclui a dissertação, relatando se os objectivos foram cumpridos e os futuros desenvolvimentos.

Capítulo 2

Definição e Descrição do Problema

A primeira parte deste capítulo trata de definir detalhadamente o sistema de Encravamento, que não é nada mais do que o sistema de sinalização ferroviária a que este relatório se refere. A segunda parte proporciona uma descrição de como se pretende realizar o sistema de um ponto de vista físico e lógico.

2.1 Sistema de Encravamento

O projecto proposto visa modelar um sistema de sinalização que permite gerir o tráfego de uma linha ferroviária. Esta gestão dá pelo nome de encravamento e implica impedir a circulação de mais do que um comboio num mesmo troço de via única. O Encravamento realiza-se com o apoio de equipamentos de via tais como sinais luminosos de ferrovia, agulhas, circuitos de via e/ou pedais para detecção de comboios, balizas, etc. Estes equipamentos guiam o veículo e o maquinista para uma circulação segura na via. Todo o controlo destes equipamentos era efectuado mecânica e/ou electromecanicamente mas hoje em dia a exploração dos caminhos de ferro está cada vez mais apoiada na electrónica e informática. No que diz respeito à sinalização ferroviária existem autómatos programáveis que controlam os demais aparelhos de via citados anteriormente. Este é o tipo de sinalização lateral.

Com o aparecimento dos comboios de grande velocidade houve a necessidade de colocar toda a informação relativa à sinalização dentro dos próprios comboios, pois é considerado que a partir de uma velocidade superior à 160 km/h o tempo em que os sinais laterais se encontram no campo de visão do maquinista é julgado insuficiente para uma interpretação correcta e segura por parte deste. Para além da informação de sinalização há um mecanismo que controla a velocidade do comboio consoante esta informação. Este sistema pode ser utilizado em paralelo com um sistema de sinalização lateral.

Para facilitar uma circulação rápida dos comboios através das fronteiras na Europa, de forma a garantir segurança está a ser concebido / desenvolvido um sistema que será comum a vários países chamado ETCS (*European Train Control System*) que é uma componente do ERTMS (*European Rail Traffic Management System*). A sua implementação está prevista numa primeira fase para as linhas de grande velocidade e a longo termo nas linhas mais clássicas. Com este sistema visa-se tornar a sinalização mais inteligente e segura, reduzindo os custos de investimento e de manutenção dos equipamentos fixos como por exemplo os sinais, e também substituir os diversos sistemas nacionais de ATP (*Automatic Train Protection*) por um mais uniforme de forma a melhorar e aumentar a capacidade e velocidade das linhas europeias.

O ETCS apresenta três níveis de implementação.

O primeiro permite ser instalado em paralelo com o sistema convencional tendo desta forma menores custos. Em vez de utilizar GSM-R (*Global System for Mobile Communications – Railways*) para comunicar a sua posição e receber instruções são colocados na via equipamentos para transmitir informações aos veículos chamados eurobalisas (radio), que transmitem os dados da sinalização lateral para o interior do comboio. Implica a colocação de aparelhos que permitem a detecção de comboios na via, tais como os circuitos de via.

Tal como o nível 1, o nível 2 pode ser utilizado em paralelo com o sistema convencional e necessita igualmente de aparelhos de detecção do comboio na via. Só que desta vez as informações relativas à sinalização não são emitidas pelas eurobalisas mas sim pela rede GSM-R, sistema de comunicação baseado no GSM e desenvolvido especificamente para aplicações e comunicações ferroviárias. Com este sistema o comboio comunica ao centro de controlo a sua posição e recebe deste as acções a efectuar. Não são utilizados os sinais laterais.

O nível 3 ainda se encontrava em fase de desenvolvimento em 2007. Neste nível não se necessita de dispositivos na via para detecção do comboio, e a ocupação do troço de via é determinada pelo próprio veículo. Continuará a haver eurobalisas para actualizar a odometria embarcada. A integridade do comboio é assegurada por este.

2.2 Objectivos e arquitectura do sistema

O projecto diz respeito à modelação de um sistema de encravamento para linha de baixo tráfego, com a arquitectura apresentada no anexo **XXX**. Neste sistema não se coloca a questão de melhorar e aumentar a capacidade e velocidade de circulação da linha. O que se pretende neste sistema é mais um apoio informático ao Regime de Exploração Simplificado (RES) que é o sistema actualmente em vigor – conjunto de regras de exploração que através de protocolo à voz, via telefone, entre o Chefe de Linha e o Chefe de Comboio, asseguram a circulação em segurança.

2.2.1 Descrição Geral

O que actualmente existe à nível informático é o Posto de Comando Centralizado (PCC), sistema considerado não seguro, que apresenta um diagrama sinóptico da linha ao operador que lhe permite obter uma visão de toda a circulação informando-o sobre os itinerários seleccionados e a serem percorridos assim como os horários reais e previstos. A decisão de ocupação ou não de um troço de via é tomada pelo operador de linha, que se encontra frente ao PCC, e a comunica por comunicação telefónica ao Chefe de Comboio em resposta a um requerimento deste último.

O Sistema de Encravamento a desenvolver tem como objectivo implementar a gestão da circulação dos comboios de uma forma segura controlando a disponibilidade dos vários blocos, isto é, os troços de via única entre estações, que compõem a linha. Pode-se chamar a esta gestão, a gestão dos itinerários. Um itinerário é o trajecto de um comboio de uma estação de origem até uma estação de destino. Para tal, deverá controlar o fecho e a abertura dos sinais de saída das estações em consonância com o deslocamento e localização dos comboios, não permitindo a ocupação de um bloco da via única entre estações por mais do que um comboio, evitando assim choques frontais e em cadeia. Desta forma o operador de linha limita-se a

enviar os pedidos de formação de itinerário, isto é, ocupação do bloco, e observar o desenrolar das operações através do sinóptico apresentado pelo PCC.

Com base neste objectivo o sistema apresentará três módulos de software:

- Um Módulo de Software de Condução de Processos (MSCP) que apresentará janelas de interface com o utilizador e lhe permitirá algumas funcionalidades tais como:
  - Gestão de Pedidos de Formação de itinerário
  - Condução de Processos
  - Sinóptico da Linha
  - Graficagem de Circulações
  - Gestão / Administração de Utilizadores

  Estas funcionalidades não apresentam aspectos críticos de segurança.
- Um Módulo de Software dos Autómatos dos controladores de Estação (MAEst) que tratará de:
  - Recolha, tratamento e validação da informação de transições do estado dos pedais (sensores) de detecção de comboios,
  - Comando, teste e controlo dos sinais de saída das estações,
  - Controlo dos sistemas de alimentação de energia e telecomunicações
- Um Módulo de Software do Autómato do Controlador Central de Encravamento (MAEnc) que implementará algoritmos permitindo:
  - Aquisição, a partir dos autómatos de estação, dos estados lógicos das transições das variáveis correspondentes à actuação dos sinais adquiridos pelos autómatos,
  - Interpretação lógica das sequências das suas actuações de modo a determinar o estado de ocupação dos troços de via nas estações e no bloco,
  - Encravamento lógico de modo a não permitir estados não seguros do sistema,
  - Detecção de estados não seguros e reposição do sistema em estado seguro,
  - Comando remoto dos sinais de saída das estações,
  - Controlo dos sistemas de alimentação de energia e telecomunicações,
  - Interface com MSCP para processamento dos pedidos de abertura dos sinais de estação, informação dos estados de ocupação de vias, anulação e destruição de itinerários e transmissão de alarmes emitidos pelos autómatos.

De referir que os módulos MAEst e MAEnc deverão ser de segurança crítica pois deverão garantir o acesso controlado dos comboios aos blocos entre estações de forma segura.

A nível de ambiente o MSCP será instalado no Posto de Comando Centralizado e o MAEst e MAEnc em autómatos. A integração com o MSCP e o MAEnc será assegurada via rede local Ethernet utilizando o protocolo do autómato. O MAEst e MAEnc comunicarão entre si em IP segundo um protocolo de segurança dos autómatos designado SafeEthernet.

Para além deste ambiente o Sistema de Encravamento terá interface com vários sistemas externos (REFER) tais como:

- Sistema ERCom (comunicações IP) – interface que permite comandar o sistema ERCom que assegura as comunicações de dados e voz em protocolo IP.
- SITRA – interface que permite ao encravamento comunicar o registo cronológico das entradas e saídas de estação ao SITRA que assegura o registo central de horários de circulação realizados e respectivos motivos de eventuais desvios.
- SATA (Sistema Automático de Tratamentos de Alarmes de Passagens de Níveis) – interface que permite enviar informação sobre estados de alarme ao sistema SATA que regista esses eventos.
- Sincronismo Horário assegurado através de protocolo FTP – solicita informação periódica ao servidor central da REFER, para acertar a data e hora do sistema de encravamento.
- Servidor de Horários assegurado através de protocolo FTP – permite ao sistema receber do Servidor de Horários.

2.2.2 Requisitos Funcionais

O Subsistema de Encravamento funciona somente com os comandos vindos do Posto de Comando Central (PCC). Os comandos de formação, anulação e destruição de itinerários são dados pelo operador de linha e as mensagens relacionadas com a informação de operações / alarmes são transmitidas pelo Autómato de Encravamento ao PCC. A exploração do sistema é efectuada a nível central exibindo um sinóptico da linha onde se encontram informação sobre o estado dos sinais, o estado dos bloco e estações assim como o estado dos itinerários (formado, anulado, destruído). São ainda exibidas no sinóptico uma janela de comandos e informações categorizadas somo alarme, aviso ou informação.

O Subsistema de Encravamento deve ser capaz de detectar falhas que coloquem em risco a realização das operações em condições de fiabilidade e de segurança aceitáveis. Para o efeito deve monitorizar e reportar as seguintes falhas:

- Alimentação
- Comunicação entre o PCC e o AEnc
- Comunicação entre o AEnc e algum dos AEst
- Interface com outros sistemas (SATA, micro-URR, etc)

Em caso de detecção de falha o Subsistema deve passar para um modo seguro caracterizado por:

- Todos os sinais passam para um estado restritivo (vermelho ou apagado), ou pelo menos aqueles que se encontram na estação que está envolvida na falha (no caso da falha ser de um AEst, ou equipamento de via da estação).
- Impedida formação, anulação ou destruição de itinerário (no caso de falha de um AEst é impedido apenas qualquer itinerário envolvendo a estação controlada por ele)

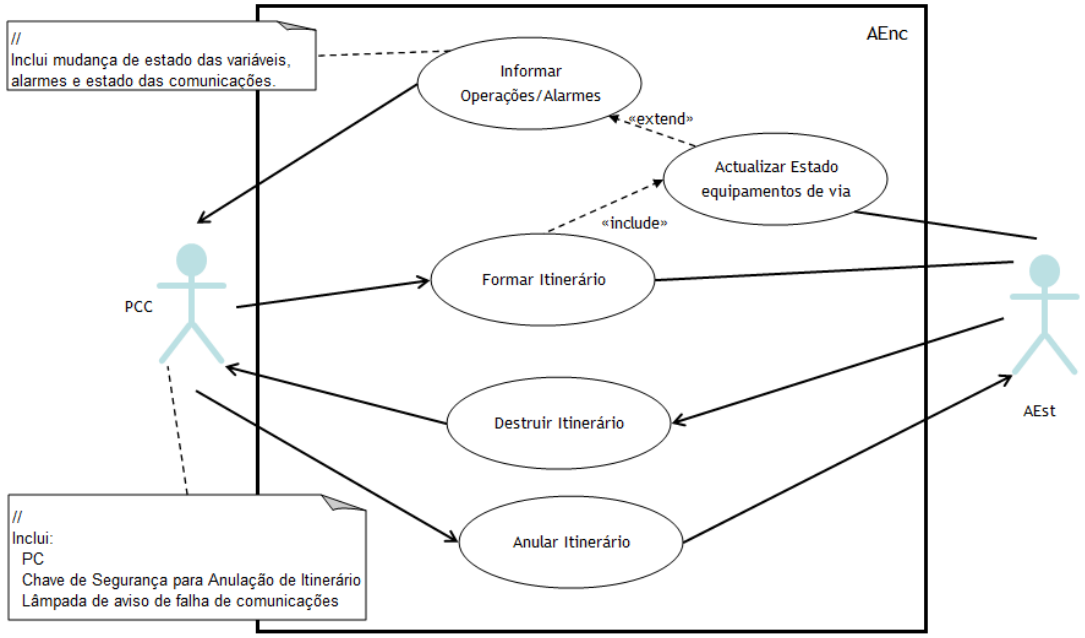
Os autómatos só poderão voltar a entrar em modo operacional normal após realização de procedimentos de reinicialização do sistema.

2.2.3 Especificação Funcional do Subsistema de Encravamento

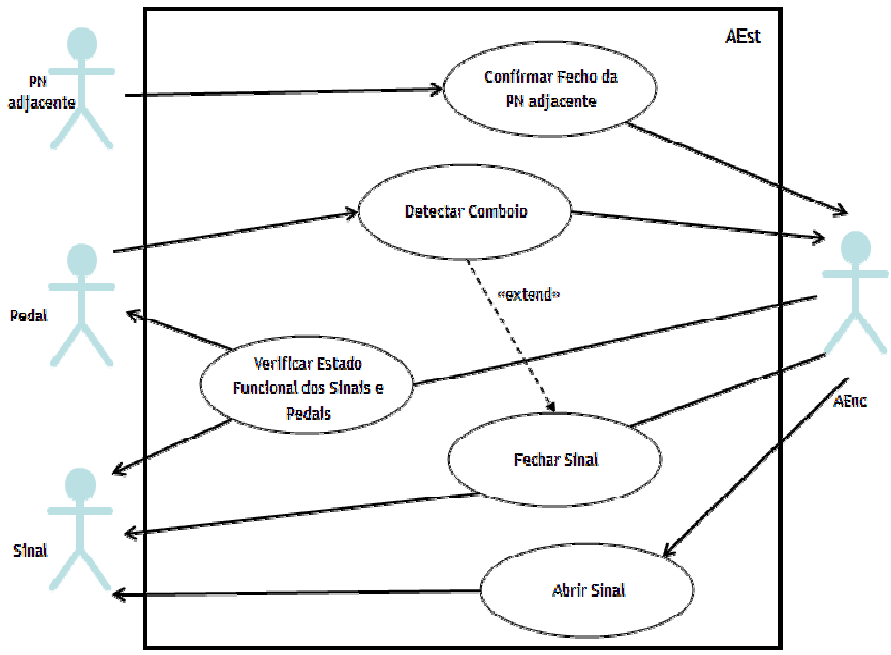
Este subcapítulo propõe uma especificação funcional do Subsistema de Encravamento, isto é, da parte lógica do encravamento, utilizando a linguagem semi-formal UML (*Unified Modeling Language*). Dos diversos diagramas propostos por esta metalinguagem, escolhe-se o diagrama de Casos de Utilização muito utilizado para definir requisitos estruturais ou por outras palavras aquilo que o sistema em causa disponibiliza. O diagrama está acompanhado de um texto no qual se descreve a interacção entre os actores e os casos de utilização mais relevantes.

Sendo que toda a lógica de encravamento é efectuada pelos módulos de software MAEnc e MAEst presentes nos Autómatos de Encravamento e de Estação (AEnc e AEst), define-se dois diagramas cada um representando um módulo.

AUTÓMATO DE ENCRAVAMENTO:



AUTÓMATO DE ESTAÇÃO:



As seguintes tabelas contêm o texto descrevendo os sistemas, actores e alguns casos de utilização, nomeadamente a interacção entre os actores e estes casos de utilização.

SISTEMAS

AEnc	Autómato de Encravamento Central, responsável por gerir o sistema de encravamento de toda a linha
AEst	Autómato de Estação, responsável por controlar e gerir os diversos equipamentos que interagem com o comboio

ACTORES

PCC	O Posto de Comando Central inclui: Funcionalidade Interface Homem Máquina (Terminal RESI) com o Autómato de Encravamento (AEnc) Chave de Segurança de Anulação de Itinerário utilizada para confirmação de um pedido de anulação de itinerário. Lâmpada de aviso de falha de comunicação que se desliga aquando da falha
Sinal	Informa o comboio da autorização ou restrição ao avanço
Pedal	Actor associado à detecção do comboio
PN Adjacente	Actor que interage com o sistema como condição de segurança
AEnc	Representa o sistema AEnc interagindo com o Autómato de Estação
AEst	Representa o sistema AEst interagindo com o Autómato de Encravamento

CASOS DE UTILIZAÇÃO

FORMAR ITINERÁRIO

Identificação	
Objectivo	Operação do Subsistema de Encravamento que consiste em preparar (verificar disponibilidade) e formar um itinerário
Actores	PCC AEst
Sistema	AEnc
Descrição de funcionamento	
Pré-condições	Comunicação do PCC com AEnc operacional Comunicação do AEnc com AEst da estação de origem e AEst da estação de destino operacionais

Sequência	Comando: PCC envia pedido de avanço ao AEnc AEnc recebe o pedido Preparação: Verificar estado (funcional) dos sinais e pedais Verificar existência de encravamento do bloco Formação: Encravar bloco (encravar sentido do bloco, isto é, impedir que itinerários em sentido contrário possam ser formados + encravar trânsito do bloco, isto é, impedir que um segundo itinerário incompatível, no mesmo sentido possa ser formado) AEnc envia comando ao PCC para fechar a(s) PN(s) adjacente(s) (no caso de existir(em)) Executar ***Temporizador geral de PN*** para abertura do sinal da estação de origem (garante que o comboio não alcança a(s) PN(s) adjacente(s) antes do tempo regulamentado no caso de não receber a confirmação de fecho da(s) PN(s)) Receber do AEst da estação de origem a confirmação de que a PN se encontra fechada Executar ***Temporizador de PN*** para abertura do sinal da estação de origem (garante que o comboio não alcança a(s) PN(s) adjacente(s) antes do tempo regulamentado) Enviar comando de abertura do sinal ao AEst da estação de origem, após tempo definido no ***Temporizador de PN*** Receber do AEst da estação de origem a confirmação de que o sinal foi aberto Receber informação do AEst desta mesma estação de que o pedal de saída foi actuado Receber do AEst da estação de origem confirmação do fecho do sinal (AEst fecha sinal com actuação do pedal de saída) Iniciar ***Temporizador de Percurso*** para contar o tempo de percurso do comboio até à estação de destino de forma a validar janela de tempo de actuações do pedal de estacionamento da estação de destino
Pós-condições	Itinerário formado Sentido e trânsito do bloco encravados Temporizador de percurso a decorrer

DESTRUIR ITINERÁRIO

Identificação	
Objectivo	Operação do Subsistema de Encravamento que consiste em destruir um itinerário quando um comboio chega à estação de destino, após confirmação telefónica entre o Chefe de Linha e o Chefe de Comboio
Actores	PCC AEst
Sistema	AEnc
Descrição de funcionamento	
Pré-condições	Comunicação com PCC operacional Comunicação entre AEnc e AEst da estação de destino operacional Itinerário formado Temporizador de percurso a decorrer
Sequência	Pedal de estacionamento da estação de destino actuado Receber informação do AEst da estação de destino de que o comboio pisou o pedal de estacionamento Verificar que o tempo obtido no temporizador se encontra no intervalo aceitável para realização do percurso AEnc envia 1ª mensagem ao PCC (Terminal RESI) para comando de confirmação de destruição de itinerário (1ª mensagem: “Confirma comboio XXX na estação YYY ?”) AEnc recebe 1ª mensagem do PCC (Terminal RESI) de confirmação de destruição de itinerário AEnc envia 2ª mensagem ao PCC (Terminal RESI) para comando de confirmação de destruição de itinerário (2ª mensagem: "Confirma destruição de itinerário ZZZ ?") AEnc recebe 2ª mensagem do PCC (Terminal RESI) de confirmação de destruição de itinerário Destruir itinerário
Pós-condições	Itinerário destruído Desencravamento de sentido e trânsito do bloco

ANULAR ITINERÁRIO

Identificação	
Objectivo	Operação do Subsistema de Encravamento que consiste em anular um itinerário
Actores	PCC AEst
Sistema	AEnc
Descrição de funcionamento	
Pré-condições	Comunicação com PCC operacional Comunicação entre AEnc e AEst da estação de origem operacional Itinerário formado
Sequência	Operações no PCC: PCC pede comando de anulação PCC pede autenticação do Chefe de Linha (inserir palavra-chave) PCC envia comando de anulação de itinerário ao AEnc Operações no Autómato: AEnc recebe comando de anulação por parte do PCC AEnc envia pedido de confirmação ao PCC (mensagem) PCC confirma anulação de itinerário actuando Chave de Segurança correspondente ao itinerário (só serão aceites anulações do itinerário seleccionado anteriormente para anulação) Anular itinerário: • AEnc envia comando de fecho do sinal ao AEst da estação de origem • Recebe do AEst da estação de origem a confirmação do fecho do sinal • Pára ***Temporizador de Percurso*** • Desencrava trânsito e sentido do bloco
Pós-condições	Desencravamento de sentido e trânsito do bloco ***Temporizador de PN*** anulado ***Temporizador de Percurso*** anulado

2.2.4 Arquitectura do Sistema

A arquitectura do Sistema de Encravamento já foi sumariamente evocada nos pontos anteriores. O sistema é composto pelo Posto de Comando Centralizado onde se encontra o MSCP, um Autómato de Encravamento onde é executado o MAEnc e vários Autómatos de Estação, um para cada estação, no qual está o MAEst. O PCC serve de interface do AEnc ao utilizador, neste caso o operador de linha, que lhe permite enviar comandos de formação, anulação e destruição de itinerários para o Autómato de Encravamento. O MSCP apresenta um sinóptico com os vários itinerários em fase de realização, i.e. a serem percorridos, e informações sobre alarmes, avisos ou simples informações. Todas estas informações são interpretadas e actualizados a partir de dados enviados pelo Autómato de Encravamento. O MAEst controla todos os equipamentos da estação onde está inserida bem como as Passagens de Níveis (PNs) adjacentes a estação isto é aquelas cujo sinal de PN (SPN) coincide com o sinal de saída de estação. Está encarregado de executar comandos de abertura e fecho de sinal enviados pelo Autómato de Encravamento e enviar informação sobre o estado dos aparelhos ao Autómato Central. O MAEnc gera toda a Lógica de Encravamento a saber a formação, anulação e destruição de itinerários, a verificação de condições de segurança. Para além desta gestão ele deve monitorizar as comunicações tanto com o PCC como com os vários Autómatos de Estação.

Capítulo 3

Técnicas de Modelação no Domínio Ferroviário

Neste capítulo apresenta-se algumas técnicas de modelação utilizadas no domínio ferroviário assim como algumas ferramentas empregadas no âmbito da modelação de sistemas A primeira parte descreve em que contexto entra a modelação no processo de desenvolvimento de sistemas para gestão de linha ferroviárias. Na segunda parte descrevem-se algumas técnicas assim como algumas ferramentas mais utilizadas para esse fim. A última parte descreve um pouco a técnica bem como a ferramenta escolhidas para modelar o Sistema de Encravamento proposto para o projecto.

3.1 Normas

No domínio ferroviário da União Europeia a concepção e desenvolvimento de sistemas considerados críticos e vitais, como os de sinalização, tem de responder a certas normas de modo a garantir a integridade de segurança dos sistemas.

Foram desenvolvidas as normas:

- EN 50126 é relativa ao sistema no seu conjunto
- EN 50129 é relativa à segurança funcional dos sistemas
- EN 50128 é relativa à concepção e desenvolvimento de software integrado nos sistemas

Estas normas são chamadas normas filhas da norma IEC61508 mais geral, equivalentes às que se aplicam a outras áreas específicas tais como por exemplo a EN61511 para os processos industriais, a EN62061 para as máquinas, EN61513 para o nuclear.

A IEC 61508 é a norma que estabelece os requisitos para o desenvolvimento e implementação de sistemas eléctricos e electrónicos de segurança crítica para sistemas industriais em geral. Esta norma está dividida em sete partes que permitem cobrir os vários aspectos dos sistemas E/E/EP (*Electrical / Electronic / Programmable Electronic Safety-related systems*). Entre elas estão princípios gerais, princípios relacionados com software, e exemplos de métodos para determinar o nível de integridade e de segurança, os chamados SIL (*Safety Integrity Level*) definidos por esta norma.

Na IEC61508 existem 5 níveis de integridade de segurança que vão de zero até quatro sendo que quanto maior é o nível menor é o risco de acontecimento de uma falha perigosa. A norma IEC aplica-se tanto a sistemas de segurança que funcionam por solicitação como a sistemas que trabalham em permanência num estado considerado não perigoso. Para definir estes dois tipos de sistemas, o SIL é especificado de duas maneiras: PFD (Probability *of Failure on Demand*) e PFH (*Probability of Failure per Hour*). Para o cálculo tanto do PFD

como do PFH deve-se ter em conta todos os problemas em todos os elementos que compõe o sistema, sendo que um dos factores importantes para este cálculo é a taxa de falhas críticas.

A partir dos resultados obtidos consegue-se determinar o SIL.

Para cada elemento de um sistema é determinado um PFH (ou PFD) e o SIL correspondente mas não significa que se possa garantir para o sistema no global um SIL igual. Por exemplo um sistema composto por dois elementos com SIL2 não é garantindo que o SIL do sistema seja o mesmo, isto é, se os elementos forem colocados em série poderá ser garantido um SIL inferior aos SIL dos elementos enquanto que se forem colocados em redundância, desde que bem realizada, podem levar o sistema a obter SIL3. O cálculo do SIL de um sistema é efectuado somando todos os PFD (ou PFH) e verificada a correspondência, consoante a tabela que se encontra abaixo. Salienta-se que o elemento com menor SIL é o que maior influência terá sobre o SIL final do sistema.

No que diz respeito ao domínio ferroviário os cálculos do SIL são efectuados tendo em conta os PFH.

Escala dos Níveis do SIL

SIL*	Solicitações do Sistema: raras PFD**	Solicitações do Sistema: frequentes PFH***	Factor de redução do risco
4	$\geq 10^{-5}$ até $< 10^{-4}$	$\geq 10^{-9}$ até $< 10^{-8}$	de 10000 até 100000
3	$\geq 10^{-4}$ até $< 10^{-3}$	$\geq 10^{-8}$ até $< 10^{-7}$	de 1000 até 10000
2	$\geq 10^{-2}$ até $< 10^{-3}$	$\geq 10^{-7}$ até $< 10^{-6}$	de 100 até 1000
1	$\geq 10^{-1}$ até $< 10^{-2}$	$\geq 10^{-6}$ até $< 10^{-5}$	de 10 até 100

* Safety Integrity Level, Nível de Integridade e Segurança

** Probability of Failure on Demand, Probabilidade de haver falha (realizanda a função prevista) na altura da solicitação

***Probability of a dangerous Failure per Hour ou Probability of Failure on high demand, probabilidade de uma falha perigosa por hora

Figura 1 – Escala dos níveis do SIL – ref: DOC_IEC_61508

Na EN50126 estabelece-se o ciclo de vida de um sistema desde a fase de conceito até a fase de desmantelamento, são definidas as várias etapas de cada uma das 14 fases do ciclo nomeadamente:

- Conceito: define-se o âmbito e o propósito do projecto.
- Definição do sistema e condições de aplicação: entre outras tarefas realiza-se nesta fase uma análise das Situações Potencialmente Perigosas (SPP) que podem afectar a disponibilidade e segurança do sistema. Esta análise pode ser do tipo HAZOP. Para além desta análise deve-se definir o Plano de Segurança.
- Análise de Risco: determinação dos riscos associados às SPP. A partir desta análise pode-se determinar qual o SIL necessário para obter um sistema seguro.
- Requisitos do Sistema: Definir requisitos do sistema, nomeadamente aqueles relativos ao RAMS (*Reliability, Availability, Maintainability and Safety*), conforme o Plano de Segurança. Definir o plano de aceitação.
- Alocação dos requisitos do sistema: alocação dos requisitos do sistema aos componentes e elementos externos bem como a definição dos sub-sistemas.

- Projecto e realização: registo de todas as tarefas de validação do RAMS desta fase, especificação dos planos para futuras tarefas do ciclo de vida, redacção do Plano de Segurança Geral e Manual de Segurança.
- Fabrico: efectuar implementação de processos de fabrico dos subsistemas e componentes validados pelo RAMS e planificar processo de instalação.
- Instalação: montagem e instalação dos subsistemas e componentes, documentação do processo de instalação, definição dos procedimentos de apoios logísticos do sistema.
- Validação do Sistema: Verificação e Validação (V&V) da conformidade entre os requisitos, o projecto e realização, o fabrico e a redução de risco.
- Aceitação do sistema: avaliação das tarefas de V&V do sistema, do V&V do RAM, Manual de Segurança. Aceitação do sistema.
- Exploração e Manutenção.
- Monitorização do desempenho
- Modificação e Reengenharia: a modificação deve ser efectuado tendo em conta os requisitos do RAMS.
- Abate, Desactivação e Desmantelamento: deve ser mantida uma análise relativa ao ciclo de vida do RAMS de forma a servir para futuros sistemas.

Tal como mencionado anteriormente a norma EN 50128 refere-se ao software que será integrado no sistema. Entre outras indicações a norma refere um conjunto de considerações a ter relativamente a escolha de compiladores, linguagens de programação, e algumas regras de implementação. Define igualmente o ciclo de vida que deverá ser devidamente escolhido segundo alguns critérios e documentado. Deverá ser definido um Plano de Garantia de Qualidade do Software e documentada a sua fase de integração com o hardware. As fases do Ciclo de Vida do Software são:

- Especificação dos Requisitos do Software: além desta especificação há também a Especificação dos testes dos Requisitos do Software e o Relatório de Verificação dos Requisitos
- Fase de Arquitectura e Concepção do Software: Especificação da Arquitectura do Software e Especificação da Concepção do Software. Após verificação obtém-se o Relatório de Verificação da Arquitectura e Concepção do Software
- Fase de Concepção dos Módulos de Software: Especificação da Concepção dos Módulos de Software e Especificação dos Testes dos Módulos de Software. Após verificação obtém-se o Relatório de Verificação dos Módulos de Software.
- Fase de Codificação: Código Fonte e Documentação de Suporte. Após verificação obtém-se o Relatório de Verificação do Código Fonte do Software.
- Fase de Testes dos módulos de Software: Relatório dos Testes dos Módulos de Software
- Fase de Integração do Software: Relatório dos Testes de Integração do Software
- Fase de Integração Software / Hardware: Relatório dos Testes de Integração Software / Hardware
- Fase de Validação do Software: Relatório de Validação do Software
- Fase de Avaliação do Software: Relatório de Avaliação do Software

- Fase de Manutenção do Sotware: Registos de Manutenção do Software e Registos de Alteração do Software

Salienta-se que os requisitos do software, principalmente aqueles que estão relacionados com a segurança do sistema, são especificados, após identificar todas as funções de segurança atribuídas ao software, consoante os Requisitos do Sistema, os Requisitos de Segurança do Sistema e o Plano de Segurança do Sistema.

3.2 Ferramentas de especificação e gestão de requisitos

Colocando esta norma de um ponto de vista mais prático, a primeira fase seria a definição e especificação textual dos requisitos, especificação dos casos que devem ser testados com todos os testes necessários e um documento relatando todas as Situações Potencialmente Perigosas. Realizados os documentos procede-se ao controlo da consistência dos requisitos que serão validados por um especialista na área de sistemas críticos e do domínio ferroviário. O passo seguinte consiste na modelação dos requisitos funcionais, procedendo, após o controlo da consistência dos modelos obtidos, à verificação dos requisitos do software a partir dos modelos. Verificados os requisitos prossegue-se com a simulação efectuada a partir dos modelos obtidos. Esta simulação é posteriormente validade pelo especialista. Logo que os requisitos e os modelos sejam validados pelo especialista pode-se avançar para a implementação a partir da modelação de requisitos realizada na segunda fase. Esta descrição apresenta o processo de desenvolvimento dos requisitos para sistemas europeus de encravamento ferroviário conforme as normas europeias e as recomendações da UCI (União Internacional dos Caminhos de Ferro), conforme ilustra a figura abaixo.

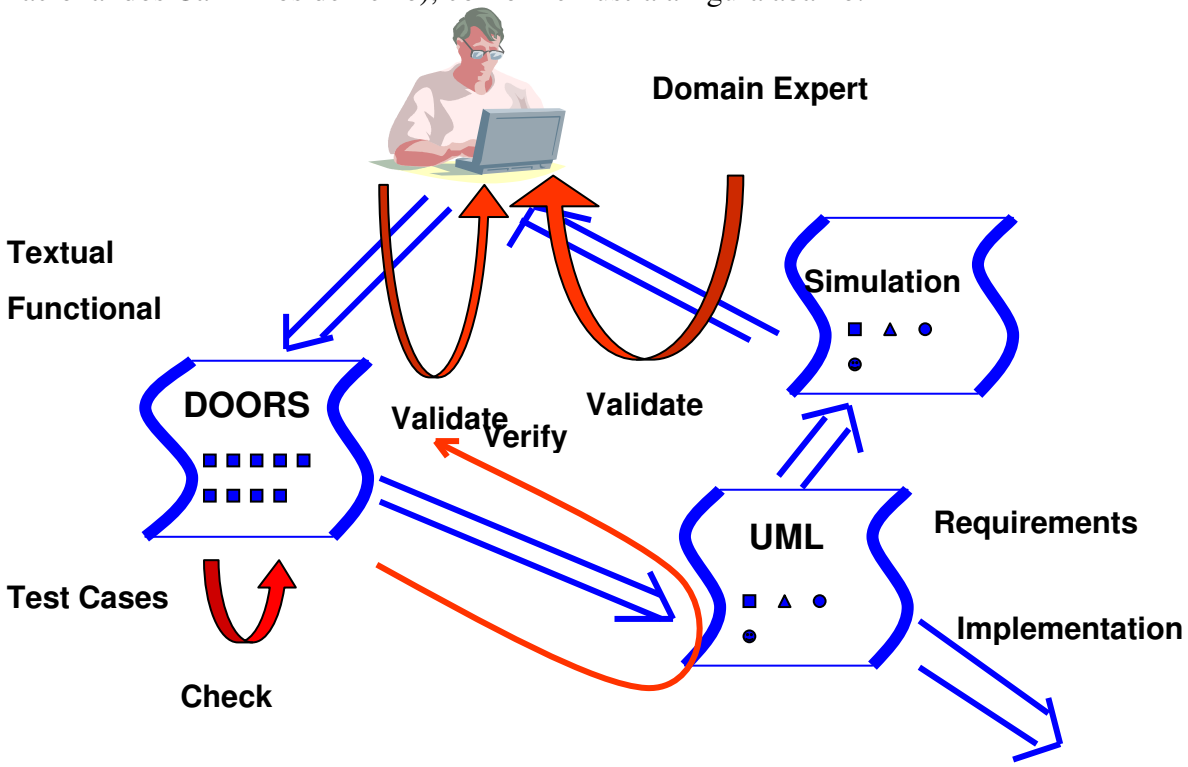


Figura 2 – Requirements Development Process – ref: UIC

Existem algumas ferramentas para a definição e gestão de requisitos. Aquela que é mais utilizada nas áreas de sistemas críticos, como são os casos da ferrovia e da aeronáutica, é o DOORS da Telelogic. Empresas como a Airbus, Boeing e Thales, entre outras, utilizam esta ferramenta para a definição dos seus requisitos. O DOORS é uma ferramenta multi-

plataforma que permite capturar, ligar, traçar, analisar e gerir a evolução das informações de forma a garantir a conformidade do projecto com os requisitos e as normas. É também um gestor de base de dados proprietário. A gestão dos requisitos consiste portanto em trazer tudo (modelo UML, Modelo B, especificação textual, ...) nesta base de dados e colocar ligações entre os vários elementos. O DOORS fornece meios de rastreamento de forma a seguir estas ligações. Ele permite igualmente a interoperabilidade com ferramentas de tratamento de documentos tais como as ferramentas da Microsoft (Word, Excel, Project, ...) entre outros.

Outra ferramenta utilizada é a Reqtify da GEENSYS (*Global Embebbed Electronics and Networked System Solutions*) que permite igualmente a rastreabilidade e análise através do ciclo de vida dos projectos. Foi desenvolvido no âmbito do programa A380. O Reqtify permite a automatização da rastreabilidade de forma a seguir a implementação dos requisitos e assegurar-se de que eles estejam bem definidos no conjunto dos produtos. A esta especificidade acrescenta-se a análise do impacto e a geração de documentos automatizada.

As técnicas e ferramentas de modelação e simulação são abordadas na próxima secção.

3.3 Técnicas e ferramentas de modelação

A fase seguinte à definição e especificação dos requisitos consiste na concepção do software e a sua especificação. Para tal é recomendado a utilização da modelação. Ora, como está descrito abaixo, existem várias técnicas que podem ser utilizadas, sendo umas utilizadas mais frequentemente do que outras. Existe uma agência francesa de certificação ferroviária fundada em 1997, cujos membros representam todas as partes relativas ao mundo dos transportes ferroviários chamada CERTIFER. É um organismo de avaliação da conformidade representativa das competências em transportes ferroviários existentes a nível nacional e internacional. Segundo a CERTIFER, é primordial a utilização de métodos formais, dando o exemplo do método B (e variante B0), do SCADE, etc., bem como de métodos estruturais tais como SDL. Com a aparição do UML, a sua utilização na especificação e concepção dos softwares passou a ser recomendada.

Para alguns destes métodos de modelação existem várias ferramentas que permitem desenvolver os modelos, enquanto que para outros métodos está disponível no mercado apenas uma ferramenta, devido ao facto de que as empresas que as disponibilizam são proprietárias do método, isto é, ela própria definiu-o, como o SCADE da Esterel.

3.3.1 Eclipse

O Eclipse é um IDE (*Integrated Development Environment*) freeware que permite fornecer uma plataforma modular no âmbito da realização de desenvolvimentos informáticos. A I.B.M. está na origem do desenvolvimento do Eclipse mas esta disponibilizou à comunidade todo o código afim de continuarem o seu desenvolvimento. Esta ferramenta utiliza bastante o conceito de "plug-ins" na sua arquitectura. Tirando a plataforma do "Runtime", todas as outras estão desenvolvidas dessa forma. Este conceito permite fornecer um mecanismo para extensão da plataforma e oferecer a possibilidade a terceiros desenvolver funcionalidades não fornecidas na versão de base.

Os principais módulos fornecidos de base como o Eclipse são relacionados com Java mas ultimamente foram desenvolvidos mais módulos para outras linguagens como C / C++, e outros aspectos do desenvolvimento como base de dados e concepção utilizando UML. Todos

são desenvolvidos em Java ou pelo projecto ou por terceiros de forma comercial ou *open source*.

Os módulos agem sobre ficheiros incluídos no *Workspace* onde se encontram vários projectos em arborescência.

O Eclipse possui vários pontos fortes que estão na origem do seu enorme sucesso sendo os principais:

- Uma plataforma aberta para o desenvolvimento de aplicações e extensível devido ao seu mecanismo de plug-ins
- Vários versões de um mesmo plug-in podem co-habitar numa mesma plataforma
- Um suporte multi-linguagens graças a plug-ins dedicados: C, PHP, C#,...
- Um suporte de várias plataformas de execução: Windows, Linux, Mac OS X, ...
- Embora esteja desenvolvido em Java, Eclipse consegue ser rápido a executar devido à utilização da biblioteca SWT (Standard Widget Toolkit)
- Várias funcionalidades de desenvolvimento propostas pelo JDT
- Uma ergonomia configurável que propõe diferentes perspectivas consoante as actividades a realizar
- Um histórico local das últimas modificações
- A construção incremental dos projectos Java graças ao seu compilador que permite, além de compilar o código mesmo contendo erros, gerar mensagens de erro personalizadas, seleccionar o alvo e efectuar o *scrapbook* (permite testes de código)
- Propõe o necessário para o desenvolvimento de novos plug-ins.
- Possibilita a utilização de tecnologias open-source: CVS, Ant, Junit
- A plataforma é inteiramente internacionalizada numa dezena de línguas na forma de plug-ins

A sociedade Omondo propõe EclipseUML, um plug-in permitindo utilizar UML com Eclipse. EclipseUML disponibiliza uma versão grátis suportando onze diagramas do UML 2.0.

Para funcionar correctamente com Eclipse 3.0, o EclipseUML necessite de alguns plug-ins:

- GEF (Graphical Editor Framework) 3.0.1
- EMF (Eclipse Modeling Framework) 2.0.1
- UML2 1.0.1

A ferramenta RSA (*Rational Software Architect*) da IBM oferece igualmente um ambiente de modelação UML para Eclipse.

O Eclipse não é só utilizado para a modelação, mas também para todo o processo de desenvolvimento. No sector ferroviário foi por exemplo utilizado para a reestruturação da exploração das linhas ferroviárias holandesas.

3.3.2 Scilab / Scicos

Esta é uma ferramenta de software utilizada para modelação e simulação de sistemas dinâmicos híbridos editado pela INRIA (*Institut National de Recherche en Informatique et en Automatique*) e ENPC (*Ecole National des Ponts et Chaussées*).

O Scilab é um software numérico científico que fornece um poderoso ambiente de desenvolvimento para aplicações científicas e de engenharia. É distribuído de forma livre sendo actualmente utilizado no mundo inteiro em empresas, investigação e ensino. Scilab é hoje gerido pelo Consortium Scilab composto por vários membros sendo um deles a Thales.

O Scilab disponibiliza centenas de funções matemáticas com possibilidade de acrescentar interactivamente programas escritos em diversas linguagens (FORTRAN,C,C++,Java). Possui estruturas de dados sofisticadas (incluindo listas, polinómios, fracções racionais, sistemas lineares,...), bem como um interpretador e uma linguagem de programação de alto nível. Scilab foi concebido de forma a ser um sistema aberto no qual o utilizador pode definir novos tipos de dados e operações sobres estes tipos de dados.

Existem várias ferramentas: gráficos 2D / 3D e animação, álgebra linear, polinómios e fracções racionais, simulação (ferramentas que resolvem sistemas de equações diferenciais explícitas e implícitas), optimização LMI, optimização diferencial e não diferencial, tratamento do sinal, grafos e redes, estatísticas, interfaces com cálculos formais (Maple, MuPAD), interface TCL / TK.

Scilab funciona em Windows 9X / 2000 / XP, GNU / Linux e na maior parte dos ambientes UNIX.

Scicos é umas das ferramentas do Scilab que permite efectuar modelações e simulações de sistemas dinâmico híbridos. Fornece a maior parte das funcionalidades do software Simulink e SystemBuild: um editor gráfico de diagramas de blocos hierárquicos, paletes de blocos predefinidos, um compilador que determina o ordenamento necessário à simulação ou execução do código C gerado, um simulador híbrido de tempo discreto, continuo ou por eventos baseado no ordenamento calculado pelo compilador, ferramentas de resolução numérica de equações diferenciais (LSODAR), um gerador de código C para os subsistemas discretos do modelo.

Sendo uma ferramenta do Scilab, o Scicos dispõe de todas as funcionalidades do Scilab, tais com as funções de cálculos de filtros ou de reguladores para construção dos modelos, a linguagem de programação para conduzir as simulações em "*batch*", as ferramentas gráficas para o tratamento das simulações.

O compilador e o simulador do Scicos são baseados num formalismo preciso que pode ser considerado como uma extensão das linguagens síncronas. Dá-lhe uma semântica rigorosa relativamente a aspectos temporais. Desta forma um esquema de blocos Scicos representa um modelo onde cada bloco é activado explicitamente por um sinal de activação. Um mecanismo de herança de activação permite simplificar a implementação dos esquemas de blocos e de se aproximar dos esquemas mais tradicionais de "fluxo de dados" . Permite gerar de forma coerente dois tipos de tempos, contínuo e discreto, associados a dois tipos de funções contínuas ou discretas.

Finalmente, o Scicos pode gerar código C para uma execução mono-processador ou código SynDEx (ver secção sobre SynDEx a seguir) para efectuar uma execução com restrições de tempo real numa arquitectura multi-componente formada por processadores e

circuitos integrados específicos (ASIC, FPG) comunicando todos por ligações ponto-a-ponto ou multi-ponto (bus).

3.3.3 SynDEx

SynDEx é um software de modelação de funcionalidades, arquitecturas distribuídas embebidas e de implantações distribuídas com restrições de tempo baseado na metodologia AAA (Adequação-Algoritmo-Arquitectura). Possibilita a especificação de algoritmos aplicativos e de arquitecturas distribuídas bem como efectuar optimizações explorando manualmente ou automaticamente as diferentes implantações possíveis, respeitando sempre as restrições de tempo real. Gera código de forma automática para a implantação escolhida. Adaptado à prototipagem rápida e à realização de aplicações tempo real embebidas, possibilita o desenvolvimento conjunto hardware / software.

SynDEx é multi-plataforma (Unix / Linux, Windows, MacOs), tem como alvo as aplicações críticas de tempo reais no domínio dos transportes (aviação, automóvel, ferroviário, etc.), das telecomunicações, da robótica móvel, controlo industrial, etc.

Possui as seguintes funcionalidades acessíveis através de uma interface homem-máquina gráfica interactiva:

- especificação de funções aplicativas com a ajuda de gráficos e algoritmos (esquemas de blocos), ou de uma interface com linguagens síncronas (Scade / Esterel, SyncCharts, Signal), Scilab / Scicos, etc.;
- especificação de arquitecturas hardware de sistemas distribuídos embebidos recorrendo a gráficos “multi-componentes”, composto por componentes de software e /ou hardware, ligados entre si por meios de comunicação (ponto-a-ponto ou multi-pontos, memória partilhada ou canais de mensagens);
- especificação das associações entre funções e componentes da arquitectura quantificando ou redirigindo as distribuições possíveis;
- especificação das restrições de tempo real: períodos e latências;
- exploração das implantações realizadas manualmente e / ou automaticamente recorrendo a heurísticas de optimização fundamentadas em análises de distribuição e de ordenamento tendo em conta as quantificações e restrições de tempo real;
- visualização de um diagrama temporal simulando o comportamento tempo real das funções aplicativas no hardware;
- geração de executáveis distribuídos tempo real construídos à medida, a partir de um nó executável, específico a cada processador;
- geração de executáveis distribuídos em tempo real, tolerante a falhas, recorrendo a redundância de software efectuada automaticamente.

Syndex melhora a segurança de concepção por uma lado certificando-se da coerência entre a fase de especificação / concepção e a fase de implementação conduzindo ao código executável, e por outro lado garantindo que as propriedades lógicas sejam verificadas. Isto minimiza os testes em tempo real e, associado à geração automática do código, reduz de forma significativa o tempo de ciclo de desenvolvimento das aplicações distribuídas de tempo real, embebidas.

3.3.4 Matlab / Simulink

O Matlab consiste em primeiro lugar numa ferramenta de cálculo com base em cálculo matricial para a resolução de equações físicas, que integra bibliotecas de cálculo e de análise, uma linguagem e uma representação gráfica dos dados tratados. É frequentemente associado ao Simulink que consiste numa ferramenta de modelação, simulação e análise da dinâmica dos sistemas apoiados num conceito de biblioteca aberta integrando uma linguagem gráfica de representação.

Matlab-Simulink foi adaptado para a concepção de sistemas regidos por leis de controlo para diversas áreas. Com a sua boblioteca *StateFlow*, ele integra uma modelação de diagramas de estados que diferem dos diagramas utilizados em UML ao nível do formalismo e da execução.

Matlab / Simulink permite uma abordagem funcional da modelação de um sistema linear ou não, dos seus dados funcionais através de uma decomposição hierarquizada. A modelação decompõe da forma seguinte:

- Estrutural: blocos hierarquizados
- Comportamental: Modelação dos fluxos de dados e do tratamento de dados por blocos, vindos de bibliotecas (cálculo, diagrama de estados, código C, Fortran, Java...)

Simulink permite a modelação pela estimação de modelos a partir de dados experimentais. Possui uma linguagem própria e executável semanticamente similar à linguagem Pascal. A simulação é realizada ou em modo interpretado ou em modo compilado, e permite a manipulação gráfica dos dados e dos resultados da simulação dos modelos. O "*debug*" associado ao simulador permite a execução passo a passo das funções, a análise dos dados intermédios, e gestão de "*break points*", ou seja funcionalidades avançadas de depuração dos modelos.

Além da simulação, o Simulink integra (desde a versão R14) um modelo de verificação e de teste que permite:

- Dar a cobertura dos testes sobre o modelo
- Introduzir asserções para ponto de teste aquando da simulação
- Verificar regras de conformidade dos modelos

De referir que o Simulink integra desde a versão R14, interface com DOORS de forma unidireccional, de forma a ligar um bloco Simulink a um requisito Doors.

3.3.5 Redes de Petri

Uma rede de Petri é uma das várias representações matemáticas para sistemas distribuídos discretos. Tal como as linguagens de modelação, ela define graficamente a estrutura de um sistema distribuído como um grafo dirigido anotado. Possui nós de posição, nós de transição, e arcos dirigidos conectando posições com transições.

A qualquer momento durante a execução de uma rede de Petri, cada posição pode armazenar um ou mais *tokens*. Diferentemente de sistemas mais tradicionais de processamento de dados, que podem processar somente um único fluxo de *tokens* reentrantes, as transições de redes de Petri podem consumir e mostrar *tokens* de múltiplos lugares. Uma

transição só pode agir nos *tokens* se o número requisitado de *tokens* aparecer em cada posição de entrada.

Transições agem em *tokens* de entrada por um processo denominado **disparo**. Quando uma transição é disparada, ela consome os *tokens* de suas posições de entrada, realiza alguma tarefa de processamento, e realoca um número específico de *tokens* nas suas posições de saída. Isso é feito atomicamente. Como os disparos são não determinísticos, as redes de Petri são muito utilizadas para modelar comportamento concorrente em sistemas distribuídos.

As redes de Petri foram utilizadas por exemplo no projecto de Encravamento PIPC pela Alcatel Transport France e no METEOR a partir das quais foram extraídas máquinas abstractas B.

3.3.6 SDL

SDL (*Specification and Description Language*) é uma linguagem de descrição formal baseada nos autómatos de estados finitos comunicantes EFSM (*Extended Finite State Machine*), bastante utilizada no domínio das telecomunicações. No entanto hoje em dia a sua utilização não se limita a esta área sendo empregada com relativo sucesso nas áreas dos sistemas embebidos ou multimédia. SDL é uma linguagem normalizada e utilizada no mundo industrial apoiada por um ambiente de desenvolvimento integrado. Através de SDL, o universo divide-se em duas partes: o sistema e o seu ambiente. As especificações descritas em SDL são limitadas ao sistema.

Este método serviu por exemplo para modelar o sistema de contador de eixos utilizado num sistema de sinalização ferroviário italiano, utilizando para tal o software Cinderella SDL.

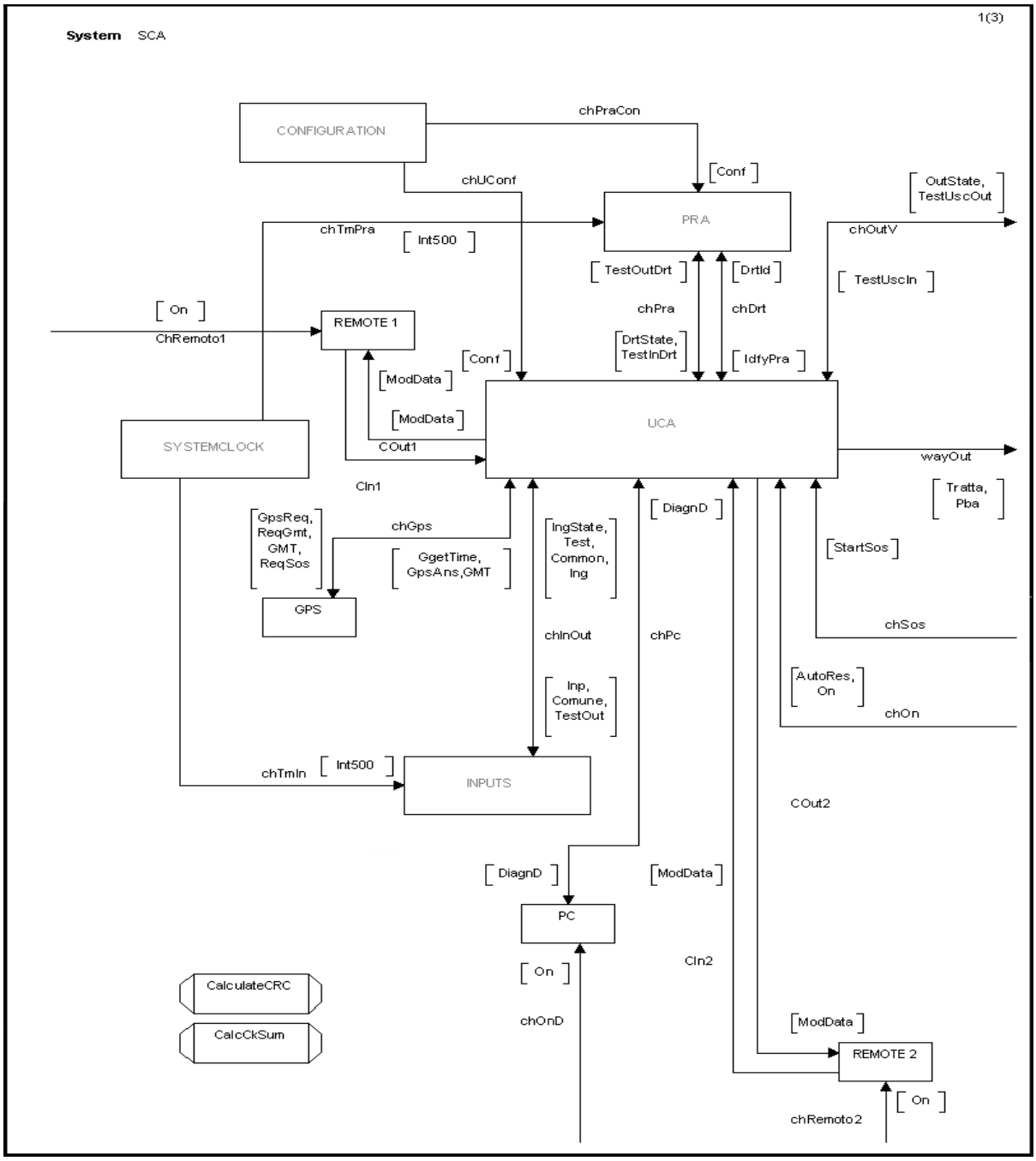


Figura 3 – especificação formal em SDL do SCA (Sistema Conta Assi,- sistema contador de eixos) – ref: SCA_SDL

Cinderella SDL

Cinderella SDL é uma ferramenta de modelação visual para desenvolvimento de sistemas de software embebidos, serviços de comunicação, protocolos, ou qualquer tipo de sistema baseado em messagens / sinais.

Cinderella SDL tem por base a linguagem de especificação SDL que é uma poderosa notação gráfica orientada a objectos que suporta análises, design, implementação e testes.

- Análise e simulação dos modelos SDL
- Notações normalizadas como SDL, MSC e ASNI

- Importação / Exportação de / para Telelogic Tau e ObjectGeode

O Editor Gráfico SDL Integrado, analisador, e simulador, suportam toda a linguagem de especificação SDL.

O Analisador Incremental SDL Cinderella executa em background enquanto se desenvolve a especificação. Ele fornece igualmente um corrector automático para alguns tipos de erros.

O simulador "built-in" simula a especificação de forma imediata, i.e. não é necessário compilar para obter um ficheiro executável para simular a especificação. Pode-se até simular e actualizar especificações parciais enquanto o software está a simular.

Object Geode

É uma ferramenta que permite especificar, verificar e gerar o código de sistemas comunicantes descritos pela linguagem SDL.

Dispõe de um módulo de verificação baseado num simulador interactivo e de um simulador exaustivo (*ModelChecking*). O módulo de verificação permite a produção de cenários de teste no formato MSC. O módulo de verificação permite:

- A simulação passo a passo ou aleatória
- A simulação exaustiva (break point sobre condições, prova sobre estados,...)
- Execução de casos de testes (ficheiros MSC – *Message Sequence Chart*)

ObjectGeode permite a geração de código executável (C, C++) a partir de uma especificação SDL.

A sociedade Telelogic substituiu as ferramentas ObjectGeode e Tau 1 pelo ambiente TAU G2.

RTDS G3

É um ambiente de desenvolvimento de software que suporta a norma definida pelo UIT, a linguagem SDL Z.100. As suas principais características são:

- Importação / Exportação de ficheiros SDL 88, 92 e 96, SDLK-PR Z100,
- Depuração gráfica do código SDL: visualização do sistema de informação SDL, das variáveis, dos break points no código SDL,...
- Suporte das linguagens C, C++, SDL, SDL-RT e UML num só ambiente,
- Integração ao Debug Multi 4.0 de Greenhills software, para funções de depuração de baixo nível

Real Time Developer Studio G3 é a primeira ferramenta do mercado que suporta ao mesmo tempo UML, SDL, e SDL-RT no mesmo ambiente. SDL-RT é uma extensão tempo real à linguagem SDL. Introduz conceitos tais como os semáforos e a integração da linguagem C para servir àqueles que desenvolvem sistemas tempo real.

Suporte UML de Real Time Developer Studio G3: Diagramas de Classe, Diagramas de Casos de Utilização, Diagramas de Implantação e Diagramas de Sequência.

De referir que os diagramas de sequência UML são equivalentes ao SDL-RT MSC (*Message Sequence Chart*).

3.3.7 UML

UML é uma linguagem gráfica de modelação orientada por objectos não proprietária muito utilizada em engenharia de software.

UML é composto por 13 tipos de diagramas. Decompõe-se em vários subconjuntos:

- As vistas: São o que se pode ver do sistema. Descrevem o sistema de certo ponto de vista, que pode ser organizacional, dinâmico, temporal, arquitectural, geográfico, lógico, etc. Combinando todas as vistas é possível definir o sistema completo.
- Os diagramas: São elementos gráficos que descrevem o conteúdo das vistas, sendo que estas são noções abstractas. Os diagramas podem ser parte integrante das vistas.
- Os elementos de modelação: São os elementos dos diagramas UML. Estes modelos são utilizados em diversos tipos de diagramas. Exemplo de elemento: caso de utilização, classe, associação, etc.

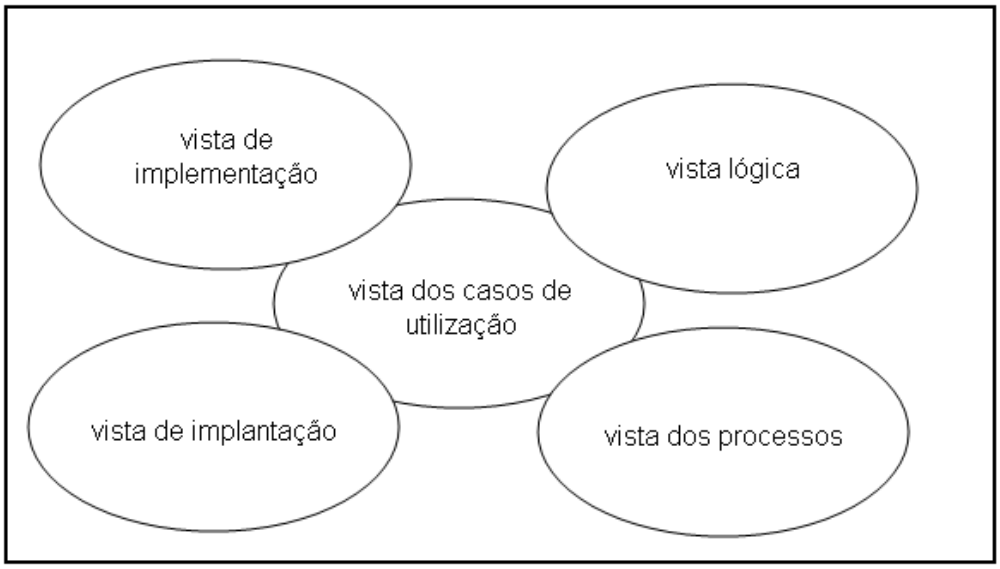


Figura 4 – Vistas possíveis do sistema utilizando de UML

- Vista dos casos de utilização: é a descrição do sistema visto pelos actores do sistema. Corresponde as necessidades esperadas por cada actor.
- Vista lógica: É a definição do sistema visto por dentro. Explica como podem ser satisfeitas as necessidades dos actores
- Vista de implementação: define as dependências entre os módulos
- Vista dos processos: é a vista temporal e técnica, que leva a noções de tarefas concorrentes, controlo, sincronização, etc.
- Vista implantação : vista que descreve a posição geográfica e arquitectura física de cada elemento do sistema.

ARTISAN Studio

Baseado em UML para tempo real, o ARTiSAN Studio permite modelar e documentar todas as especificações dos sistemas e softwares dos projectos.

Foi a primeira ferramenta UML a integrar o suporte da engenharia dos sistemas, por extensões desde 1997. Na altura chamava-se Real-Time Studio.

Três diagramas proprietários eram propostos:

- O diagrama de restrições, para definir o conjunto de especificações funcionais assim como as restrições gerais (este diagrama foi posteriormente substituído pelo diagrama de requisitos ou paramétrico do SysML)
- O diagrama de arquitectura do sistema, para definir o contexto e ambiente do sistema. Delimita a fronteira do sistema e destaca os operadores humanos e sistemas periféricos externos (substituído pelo diagrama de blocos SysML ou de estrutura composta do UML2)
- O diagrama de arquitectura de software, para mostrar a estrutura e sincronização específicas aos tratamentos de tempo real e / ou multi-tarefas, bem como a transmissão e exclusão mútua dos recursos. (este ponto de vista ainda subsiste)

Suporta a concepção de software de forma mais convencional, com geração de código completamente parametrizável e engenharia reversa.

Pela integração da ferramenta de engenharia dos requisitos Reqtify, o ARTISAN Studio oferece a ligação com os requisitos em texto provenientes de DOORS, MS WORD, RequisitePro, mas também TestDirector, Cantata++, Matlab / Simulink através do diagrama de requisitos do SysML. Suporta as arquitecturas : DoDAF, MoDAF, TOGAF, NATO, CAN, ADO, Agate. Propõe a geração e engenharia reversa para as linguagens C, C++, Java, Ada, SGL e IDL Corba.

Permite animar e simular o comportamento dinâmico do sistema. Quem desenvolve pode assim controlar a execução dos modelos (eventos, parâmetros ou temporização), sendo que o código fonte C ou C++ é gerado automaticamente. Suporta a geração do código a partir dos modelos UML do ARTISAN Studio e a sincronização automática das modificações trazidas ao nível do código até ao modelo de origem. A geração de código pode ser personalizada.

Todos os modelos criados são guardados num referencial centralizado numa base de dados, multi-utilizador, totalmente aberto via OLE Automation. O ARTISAN Studio permite, através da ferramenta de gestão da configuração, de atribuir direitos de acesso no próprio projecto.

TAU G2

A primeira versão de Telelogic TAU fazia a ligação entre a notação UML e a linguagem SDL. Esta ligação permitia a geração de modelos SDL a partir do diagrama de classes e dos diagramas de estados. O modelo SDL podia portanto ser verificado através de uma ferramenta de simulação (simulação interactiva e *model-checking*). Esta fase de verificação possibilitava a produção de casos de teste. Acessoriamente, era possível gerar código. A passagem do TAU G1 ao TAU G2 visou colocar à disposição uma ligação entre o UML e o tempo real. A versão

G2 perdeu a ligação com o SDL e portanto perdeu também a ligação com as ferramentas de verificação.

Telelogic TAU G2 é um ambiente de desenvolvimento por modelação (MDD – *Model Driven Development*) dotado de funcionalidades avançadas para o ciclo de desenvolvimento aplicativo, desde a engenharia do sistema e o desenvolvimento de software até à geração de testes. Graças ao seu suporte de modelação UML 2.1, ao MDA, ao SysML, à simulação dinâmica dos modelos e à geração de código, o TAU G2 permite automatizar as melhores práticas em matéria de concepção, desenvolvimento e teste de sistemas embebidos, de sistemas estratégicos e outros sistemas avançados.

Associados ao suporte de arquitecturas de hardware conhecidas, de sistemas operativos de tempo real e de ambientes de desenvolvimento integrados, as funções de modelação do TAU G2 permitem àqueles que desenvolvem especificar todos os aspectos da concepção de um sistema, de simular e verificar o comportamento e de assegurar-se de que o projecto está bem encaminhado ao longo do ciclo de desenvolvimento.

Disponível com TAU G2 (http://www.telelogic.com):

- Visualização da concepção do sistema
- Verificação da concepção do sistema
- Assegurar a coordenação entre análise de requisitos, engenharia de sistema e desenvolvimento de software
- Desenvolvimento em paralelo
- Geração de documentos automática
- Concepção, desenvolvimento e entrega de software de alta qualidade
- Geração, execução e gestão automática dos cenários de testes

O Telelogic TAU G2 propõe ligações com outras ferramentas, nomeadamente com ferramentas da Telelogic como o SYNERGY (gestão de versão, gestão de tarefas, ...) e DOORS (gestão de requisitos).

3.3.8 Statemate

Statemate MAGNUM permite a validação de especificações e a prototipagem rápida pela modelação comportamental e funcional executável. Statemate é a ferramenta suportando na origem dos *statecharts* inventados por David HAREL. Estes *statecharts* diferem pelo detalhe tanto ao nível do formalismo como da execução em relação ao modelo dos diagramas de estados utilizados em UML.

Statemate associa um método inerente à ferramenta segundo o qual a descrição de um sistema pode ser feito segundo três pontos de vista:

- Estrutural (Como): módulo Charts (caído em desuso)
- Funcional (Quê): activity charts, em que as funções são hierarquizadas
- Comportamental (Quando): *Statecharts* reforçados por outros meios de descrição: Tabelas de Verdade, blocos contínuos, sequencias (organigramas chamados *flowcharts*), código C ou ADA

A linguagem de acção dos "statecharts" é proprietária, de tipo pseudo-Pascal, semi-formal e semi-síncrona e é executável:

- Verificação por simulador de tipo interpretado
- Geração de código C, Ada. Para o C, um gerador de código optimizado está igualmente proposto.

Statemate suporta uma aproximação semi-objecto com instanciação com parametrização formal – real)

Vários módulos de verificação e de teste estão igualmente à disposição:

- Model Checker (Validação)
- Certifier (prova)
- ATG (Automatic Test Generation)

As noções de base para o tempo real como os estados, transições, tempo, concorrência, etc. são suportadas.

Um quarto ponto de vista está fortemente integrado: os painéis gráficos. Esta vista é utilizada durante a simulação para facilitar o desenvolvimento do modelo e pode estar ligado ao código C gerado a partir do modelo comportamental de forma a criar um executável autónomo servindo de demonstrador (protótipo virtual). Ferramentas mais elaboradas (Altia design & VAPS) podem igualmente serem utilizadas.

Em meados dos anos 90, as 2 ferramentas existentes, Express VHDL / Verilog para a concepção de hardware de ASIC ou FPGA de um lado, e Statemate C / Ada do outro, unificaram-se sob a designação de "Statemate Magnum" com uma tentativa de constituir uma ferramenta de co-design HW/SW que não vingou, levando a I-Logix a concentrar-se nos aspectos dos softwares de modelação. Na origem, Statemate integrava uma ferramenta de gestão e de rastreabilidade dos requisitos integrados, que foi abandonada em 1997, em detrimento de uma interface com DOORS e TRM, as duas ferramentas emergentes naquela época. O interface com DOORS era unidireccional, a geração de documentos é realizada com MSWord.

O Statemate integra blocos de base de gestão da configuração das versões dos seus modelos, possuindo uma interface com ferramentas clássicas: PVCS, ClearCase, Synergy, etc.

No ano 2000, houve uma tentativa de actualização suportando diagramas de sequência (*Live Sequence Charts*) e casos de utilização UML. Todavia, este suporte UML era apenas parcial já que a noção de objecto era incompleta, a semântica dos statecharts divergia bastante da que era utilizada em UML, incluindo daquela utilizada na outra ferramenta I-Logix Rhapsody, que é baseada em UML.

As ferramentas dividiam-se no mercado desta forma:

- Statemate para a especificação de sistemas de tempo real e embebidos,
- Rhapsody para a codificação visual por UML, de softwares de tempo real e embebidos.

Esta ferramenta ainda é mantida em certos mercados que ainda não adoptaram UML. Com o aparecimento do SysML no Rhapsody, este parece ter entrado em concorrência com Statemate.

3.3.9 Método B

O método B é um método formal de desenvolvimento de software que permite modelar de forma abstracta na linguagem B o comportamento de um programa, e por sucessivas refinamentos, chegar a um modelo concreto, subconjunto codificável em linguagem Ada ou C.

Uma actividade de provas formais permite verificar a coerência do modelo abstracto e da conformidade de cada refinamento com o modelo superior (provando assim a conformidade do conjunto das implementações concretas com o modelo abstracto).

Distinguem-se 3 versões:

- B clássico, tal como está definido no B Book escrito em 1996. O software de suporte é o AtelierB, ou o B-Toolkit.
- B por eventos, que é uma evolução utilizando unicamente a noção de eventos para descrever acções e não as operações (mais próximas de rotinas informáticas). Desta forma o método pode ser aplicada em variados sistemas como a electrónica e não só em programas. Realiza-se assim desenvolvimentos incrementais de sistemas provados. Para tal continua-se a utilizar AtelierB mas em conjunto com o Click'n'Prove.
- B# é uma reformulação do B por eventos com elementos da notação Z. A ferramenta mudifica-se e passa a chamar-se Rodin.

Os objectivos de B passam por formalizar a especificação, explicitar a concepção e simplificar a programação. Este método cobre:

- A especificação,
- A concepção por refinamento sucessivos ,
- Arquitectura em camadas,
- Geração de código executável.

Este método foi por exemplo utilizado para validação do sistema implementado no Metro de Paris.

AtelierB

Esta ferramenta é talvez a mais utilizada no domínio ferroviário no que diz respeito à utilização do método formal B. A verificação formal é efectuada pela implementação de uma fase de prova de lemas matemáticos. Estes lemas matemáticos são conhecidos como "obrigação de prova" (OP). O processo de geração das OP está integrado no método. A partir do modelo B, é possível gerar código (C, C++, ADA) que poderá ser associado a código escrito manualmente. Foi utilizado para o desenvolvimento dos automatismos de segurança do metro automático SAET-METEOR realizado pela Siemens.

O Atelier B é uma ferramenta comercial que permite verificar a sintaxe, a semântica e a correcção de um modelo B. A correcção de um modelo é realizado através de uma etapa de prova formal. Esta prova formal tem como propósito evidenciar a correcção do modelo inicial e a correcção do(s) refinamento(s). A cada etapa do desenvolvimento B, as obrigações de prova são geradas afim de garantir a validade do refinamento e a consistência da máquina

abstracta (formalismo associado ao método que permite o desenvolvimento incremental da especificação até ao código através da noção de refinamento).

O último refinamento é chamado implantação é muito próximo das linguagens aplicativas como C, C++, ADA, PASCAL, etc. O Atelier B dispõe de um gerador de código que permite, a partir de um modelo B, gerar o código final da aplicação.

O software é realizado na forma de um modelo em camadas que é composto de uma modelação geral formal e de uma modelação detalhada formal sendo ambas compostas por máquinas abstractas de diferentes níveis (máquina, refinamento e implementação). A fase de validação consiste em gerar as obrigações de prova de coerência e de refinamento e a tentar a prova como ilustrado na figura abaixo:

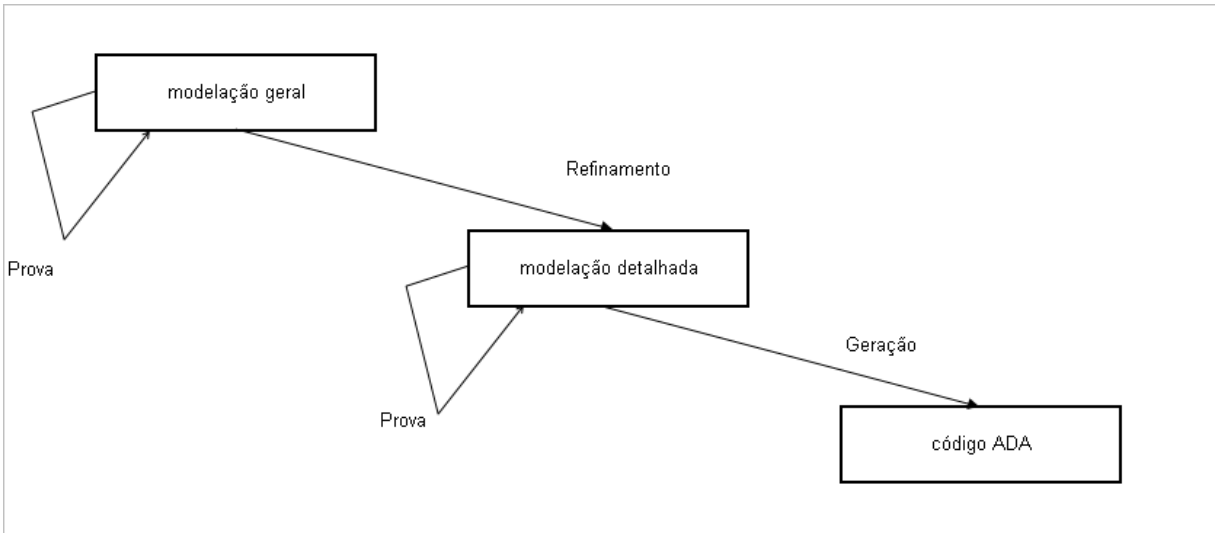


Figura 5 – Ciclo de desenvolvimento B – MeMVaTEx

O paradigma da modelação em B é baseado na introdução de propriedades que o modelo deve verificar e que serão posteriormente refinadas. Por exemplo a seguinte propriedade pode ser simplesmente exprimida em B:

- P0: não pode haver risco de colisão (os domínios de marcha dos comboios não estão em intersecção).

A pilotagem automática do SAET-METEOR é constituída por aproximadamente 115000 linhas de código B e de 115000 linhas de código ADA (dados incluídos) repartidas por 3 equipamentos. No âmbito do SAET-METEOR, houve trabalhos de validação da base de regras do provador e um trabalho de qualificação do gerador de código.

```
MACHINE
    swap
VARIABLES
    xx, yy
INVARIANT
    xx : NAT & yy : NAT
INITIALISATION
    xx :: NAT ||
    yy :: NAT
OPERATIONS
    echange =
    BEGIN
        xx := yy
    ||  yy := xx
    END
END

REFINEMENT
    swapR
REFINES
    swap
VARIABLES
    xr, yr, zr
INVARIANT
    xr= xx & yr = yy & zr : NAT
INITIALISATION
    xr, yr, zr:= 0, 0, 0
OPERATIONS
    echange =
    BEGIN
        zr := xr
    ;   xr := yr
    ;   yr := zr
    END
END
```

Figura 6 – exemplo de máquina abstracta com o seu refinamento – Wikipedia

3.3.10 SCADE

O Scade é um ambiente para o desenvolvimento de aplicações embebidas críticas como por exemplo as aplicações aviónicas que são sujeitas a fortes restrições de certificação (DO-178B Level A ou IEC 61508 SIL3 / SIL4). O Scade é baseado em modelos síncronos escritos em linguagens especializadas concebidas para uma programação segura de sistemas reactivos e de tempo real, sendo que um sistema reactivo é um sistema mantém uma interacção permanente ou frequente com o seu ambiente e que responde continuamente, em função do seu estado interno, aos estímulos externos a que está sujeito. Agora os modelos são utilizados noutras áreas como a concepção de alto nível de circuitos complexos e programação de sistemas embebidos. As linguagens síncronas são dotadas de uma semântica matemática precisa e têm um comportamento deterministico.

Scade reúne um conjunto de diagramas de fluxo baseados na linguagem síncrona LUSTRE, o modelo gráfico de estados é o das SSM (*Safe State Machine* baseado nos SyncCharts da Esterel). Estes diversos formalismos possuem uma semântica comum, e de facto, uma especificação SCADE pode combiná-los, simulá-los e desta forma validar o modelo produzido.

Este suporte semântico permite obter uma concepção correcta por construção. A cada etapa de uma concepção SCADE as actividades de verificação e de validação podem ser executadas. Estas actividades recorrem a métodos formais e à ferramenta Design Verifier integrada no ambiente e que apresenta a vantagem de ser exaustiva em relação às técnicas de teste. De facto, para testar que um dada propriedade é respeitada pelo sistema é necessário escrever os casos de teste para todas as combinações possíveis das entradas, o que se torna impossível devido à explosão combinatória destas entradas. Deste modo são efectuadas escolhas de forma a assegurar uma cobertura importante mas incompleta do teste. A utilização da técnica da prova formal possibilita uma verificação exaustiva em todo o espaço de estados do sistema, para assegurar a verificação da propriedade quaisquer que sejam as combinações de entradas.

A geração de código em Scade é possível para diferentes suportes de execução de tempo real como OSEK (utilizado na área automóvel) e MicroC/OS-II (utilizado na aviónica).

Scade propõe diferentes *gateways* para ferramentas complementares como Simulink de forma a permitir transformar uma modelação Simulink em SCADE ou integrar um modelo Simulink num nó Scade e fazer desta forma “co-simulação”. Existe também uma para Rhapsody, software de modelação UML para sistemas embebidos, e uma ligação à ferramenta DOORS para a gestão da rastreabilidade dos requisitos desde a definição dos requisitos até a geração de código.

No domínio ferroviário, o Scade foi por exemplo utilizado pela Eurostar (linha Londres – Paris / Bruxelas), nomeadamente na concepção e implementação do seu sistema de sinalização ferroviária.

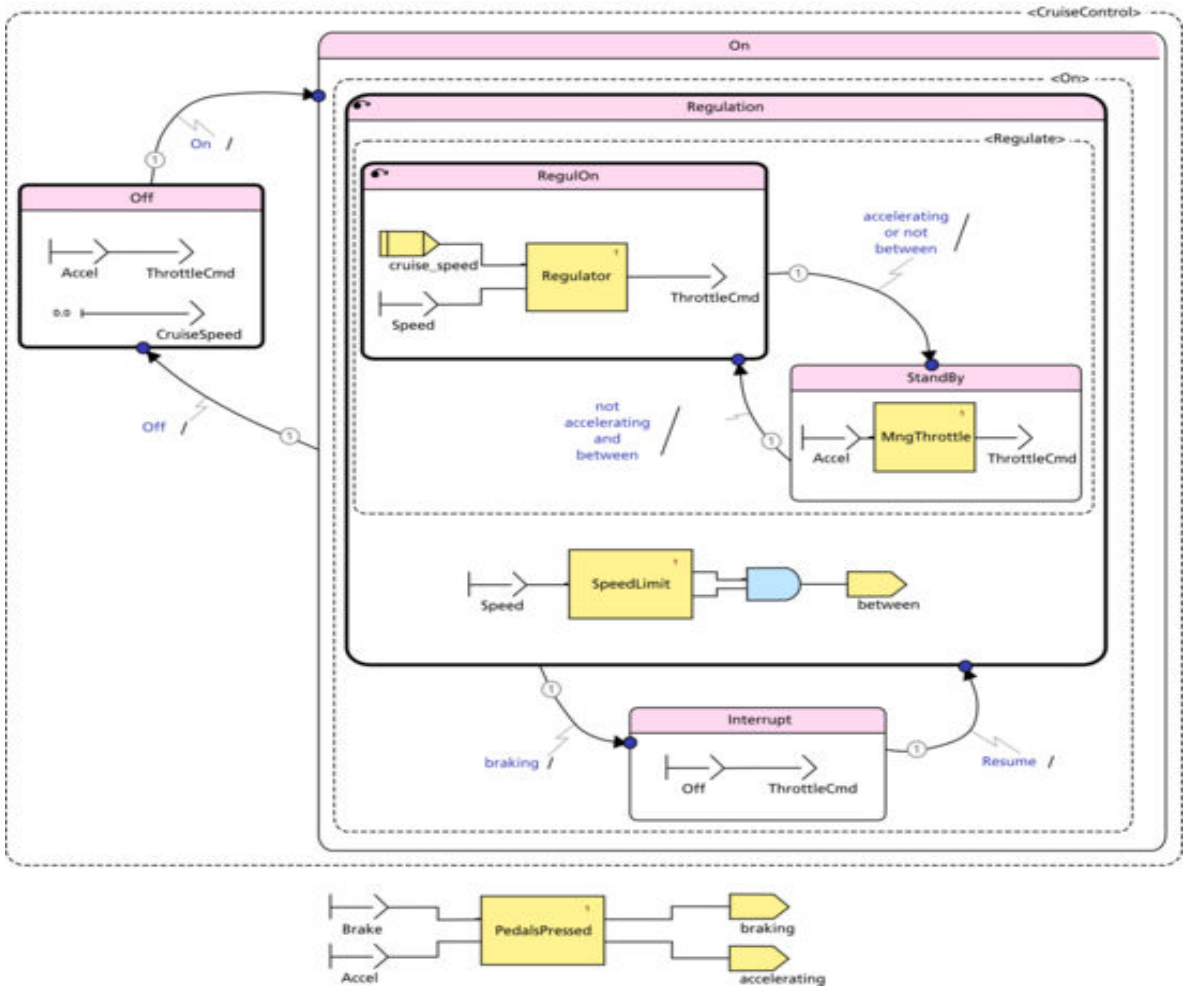


Figura 7 – Sample SCADE Suite 6.0 diagram used for Cruise-control – Wikipedia

3.3.11 LARIS / EURIS

EURIS (*EUropean Railway Interlocking Specification*) e LARIS (*LAnguage for Railway Interlocking Specification*) são métodos / linguagens de especificação para sistemas de controlo de encravamentos desenvolvidas e utilizadas pela NS Railinfrabeheer, companhia exploradora das linhas ferroviárias holandesas.

EURIS é uma linguagem de especificação graficamente orientada, paralela, conduzida por eventos, imperativa e fracamente tipificada. A especificação EURIS consiste numa descrição gráfica dos componentes na forma de LSCs (*Logic and Sequence Charts*), os quais estão conectados por portos. Cada LSC consiste num procedimento de um carácter imperativo, chamado *flow*. Os componentes podem enviar estruturas de dados chamados *telegrams* aos outros através dos portos. A recepção de um *telegram* por parte de um componente implica a execução de um *flow* no LSC pertencente a este componente. Este *flow* é determinado pelo *telegram* e pelo canal através do qual o componente recebeu o *telegram*. A alteração do valor de uma variável pode igualmente desencadear a execução de um *flow*. EURIS assume dois tipo de dados standard, booleanos e inteiros. Nos booleanos o 1 representa *true* e o 0 *false*. Existem três funções standard: adição, subtracção e multiplicação.

Flow: procedimento construído com base em:

- Estado de execução que indica em que circunstância é executado o *flow*. Existem duas possibilidade: recepção de um *telegram* ou alteração da variável associada ao estado de execução.
- Caso avalia o valor de uma variável: o valor de retorno influencia a execução seguinte do *flow*
- Uma tarefa interpreta o valor de uma variável
- Um símbolo de terminação indica o fim da execução de um *flow*
- Um *send action* p→T indica que um *telegram* T é enviado através do porto p do componente. O *send action* é sempre seguido pela terminação do *flow*.

LSCs: consiste num certo número de representações gráficas de *flows*. O LSC tem por base uma lista de variáveis internas, pela qual as variáveis carregam os seus símbolos de versão.

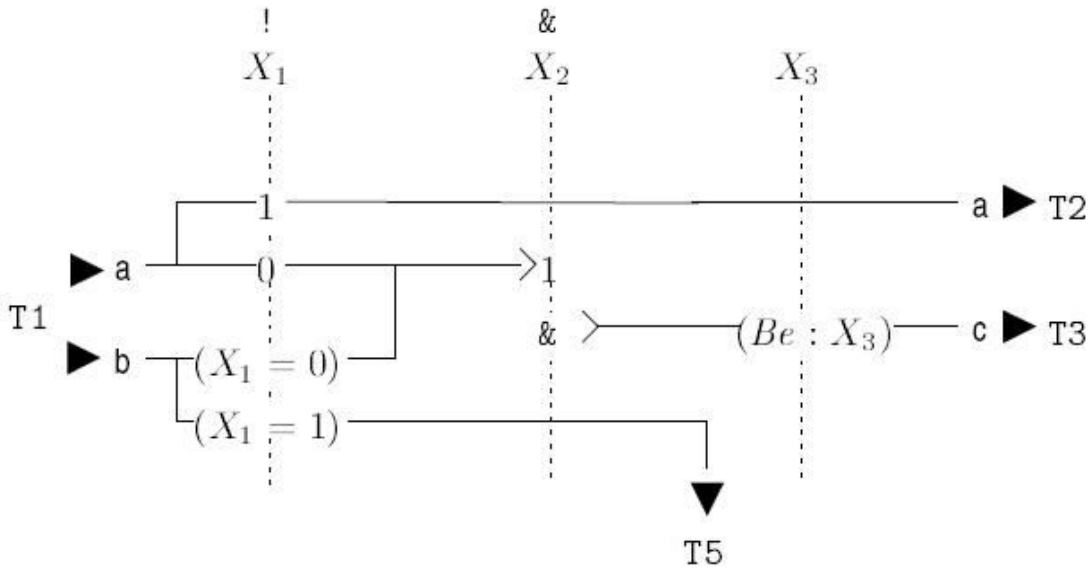


Figura 8 – Exemplo de LSC – ref: EURO_LARIS

LARIS é uma linguagem textual para especificar os sistemas distribuídos de comunicação que agem como intermediário entre um nível logístico e uma infra-estrutura. LARIS é como uma variante textual de EURIS.

3.4 Rhapsody

No presente projecto pretende-se modelar um sistema de sinalização ferroviário de uma linha de baixo tráfego utilizando o UML. Para tal foi necessário escolher uma ferramenta. A partir das descrições de várias ferramentas sobre as suas capacidade e bem como das suas possibilidades de interface, o Rhapsody acabou por ser a ferramenta escolhida para o projecto. Um dos pontos de decisão importantes em relação a outras, foi a existência de uma *gateway*

para SCADE sendo que esta ferramenta possibilita a geração de código C segundo as normas IEC 61508 e EN 50128.

3.4.1 Descrição geral

O Rhapsody foi a segunda ferramenta desenvolvida pela I-Logix. Foi criada logo que o OMG (Object Management Group) estabeleceu o UML. David HAREL, que inventou os *statecharts* utilizados na primeira ferramenta I-Logix Statemate e co-fundador da I-Logix, legitimou esta escolha de desenvolvimento de uma ferramenta UML orientada a tempo real.

Apresenta uma concepção para a testabilidade que inclui:

- Simulação dos modelos
- Testes baseados nas especificações
- Geração de testes automáticos
- Depuração ao nível dos modelos

Visualização de código e afinação ao nível do código (erros de sintaxe detectados aquando da compilação), break points, controlo da execução (pelo IDE ou directamente) etc..

Possibilidade de personalizar qualquer que seja o RTOS (Real Time Operating System)

A linguagem de acção dos *statecharts* é a linguagem utilizada para a geração de código que é desta forma executável pela geração de código C, C++, Java e Ada. Suporta o engenharia reversa.

Não é possível misturar num mesmo modelo várias linguagens, por exemplo uma combinação Java / C, o que torna mais difícil a rastreabilidade dos requisitos no caso de uma aplicação que utiliza várias linguagens (Java para a IHM e C para o código de baixo nível por exemplo), portanto neste caso seriam necessários vários modelos.

Vários módulos de verificação e de teste estão disponíveis

- Model Checker (validação)
- Certifier (prova)
- ATG (*Automatic Test Generation*)

A interface com os requisitos é assegurada pelo módulo *Gateway*, versão OEM limitada de Reqtify de TNI, assegurando a ligação com a ferramenta de gestão dos requisitos como DOORS, Requisite Pro ou então com os documentos criados em MS Word, Excel, ... em interoperabilidade com outras ferramentas de modelação (Simulink).

A geração de documentos em MS Word está disponível. Rhapsody integra os blocos de base da gestão de configuração das versões dos seus modelos, tendo uma interface com ferramentas clássicas: PVCS, ClearCase, Synergy, etc..

O Rhapsody funciona em ambiente Windows.

3.4.1.1 Aprendizagem e utilização

O software Rhapsody disponibiliza uma série de manuais de utilização bastante completos e de tutoriais sobre a sua utilização nas diferentes linguagens disponíveis para geração de código. Entre os manuais de utilização existe um manual geral, outro de iniciação,

um relativa a todo o tipo de propriedades, outros sobre a modelação de comunicação do tipo COM, disponível unicamente se utilizar como linguagem C++. Graças a toda esta documentação é possível conseguir uma boa utilização desta ferramenta. No entanto, requer bastante tempo de aprendizagem, sendo que, por exemplo, o manual de utilização contém 1400 páginas e o manual de propriedades 2200.

Embora haja uma explicação sobre como modelar em UML, a utilização de Rhapsody requer um mínimo de conhecimento de modelação utilizando UML.

3.4.2 Observações

A versão utilizada é o Rhapsody 7.2 em versão Demo. Esta versão ainda é muito recente e foi lançada no mercado mais ou menos na altura da aquisição da Telelogic por parte da IBM. (completada no dia 3 de Abril de 2008). Esta aquisição levou a algumas alterações, o que complicou a requisição de um licença demo e acaba por ser uma das causas do atraso do projecto.

Capítulo 4

Desenho dos modelos

Neste capítulo apresenta-se a concepção e alguns aspectos de implementação dos modelos para o Sistema de Encravamento para a linha de baixo tráfego. O desenho foi realizado seguindo as linhas definidas na especificação de requisitos e a especificação funcional, que foi efectuada no âmbito do estudo HAZOP. Este estudo, embora não esteja finalizado, acabou por ter influência em algumas alterações realizadas na especificação funcional e portanto também serviu para a elaboração deste modelo.

O projecto visa apenas a parte da lógica de encravamento assim como o controlo dos equipamentos de via, suportados por autómatos, sendo que um contém o MAEnc (Módulo de Software do Autómato do Controlador Central de Encravamento) e os outros o MAEst (Módulo de Software do Autómato dos Controladores de Estação). Portanto o modelo diz respeito a estes dois módulos. Desta forma cada módulo é inserido num *package*, i.e. um *package* Autómato de Encravamento e outro Autómato de Estação. Dentro de cada um foram desenvolvidos 3 tipos de diagramas: um diagrama de casos de utilização, alguns diagramas de classes e diagramas de estados. Os diagramas de estados estão associados a algumas classes, e representam os vários estados em que uma instância da classe se pode encontrar durante o seu ciclo de vida.

Sendo o tempo algo escasso e certas análises do RAMS ainda não acabadas ou realizadas, o projecto não está modelado na sua totalidade, portanto decidiu-se implementar um dos casos de utilização: Formar Itinerário (ver Capítulo 2). Salienta-se que o projecto baseia-se num protótipo, no qual se considera uma linha com três estações como, representado no anexo A**.**

As três primeiras secções deste capítulo tratam respectivamente os três tipos de diagramas escolhidos, e a última secção aborda algumas opções tomadas.

De referir também que, no Rhapsody, pode aceder-se aos elementos de um diagrama através da parte gráfica, i.e., o desenho do diagrama, ou através de um *browser* em forma de uma árvore estruturada, organizada em directórios. É portanto possível trabalhar directamente sobre o *browser*, ou a partir dos desenhos dos diagramas. O *browser* pode ser organizado consoante a vontade do utilizador e das necessidades do projecto. Por exemplo, para este projecto, foram criadas dois novos *packages*, de maneira a dividir cada módulo de software, bem como os respectivos diagramas.

4.1 Diagrama de casos de utilização

A escolha pelo diagrama de casos de utilização deve-se a 2 factores ambos relacionados com a fase de análise:

- Permite ter uma vista daquilo que se pretende do sistema assim como a interacção com os vários elementos que estão ligados a ele.
- Permitirá realizar ligações ou indicar dependências entre casos de utilização e requisitos, de forma a ter uma vista gráfica das ligações.

Estes diagramas foram realizados com o pensamento na interface com o DOORS, ferramenta para gestão dos requisitos, de forma a criar as ligações entre os casos de utilização e os requisitos inseridos no DOORS.

Apresentam-se a seguir os diagramas de ambos os sistemas.

4.1.1 Autómato de Encravamento

Este diagrama encontra-se no *package* AutomatoEncravamentoPkg, e não difere muito do diagrama apresentado para o sistema AEnc, na especificação funcional do capítulo 2.

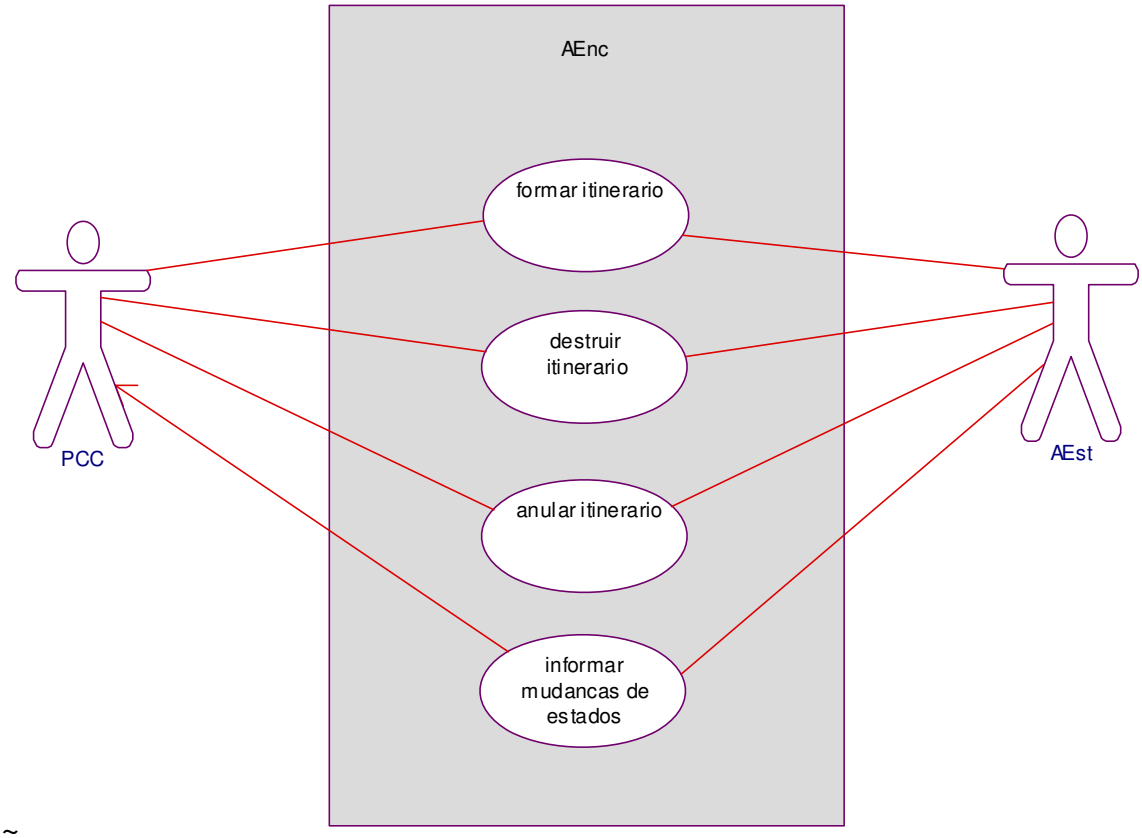


Figura 9 - Diagrama de casos de utilização do Autómato de Encravamento

4.1.2 Autómato de Estação

Tal como para o tipo de autómato anterior, o diagrama para este sistema não difere daquele especificado no capítulo 2.

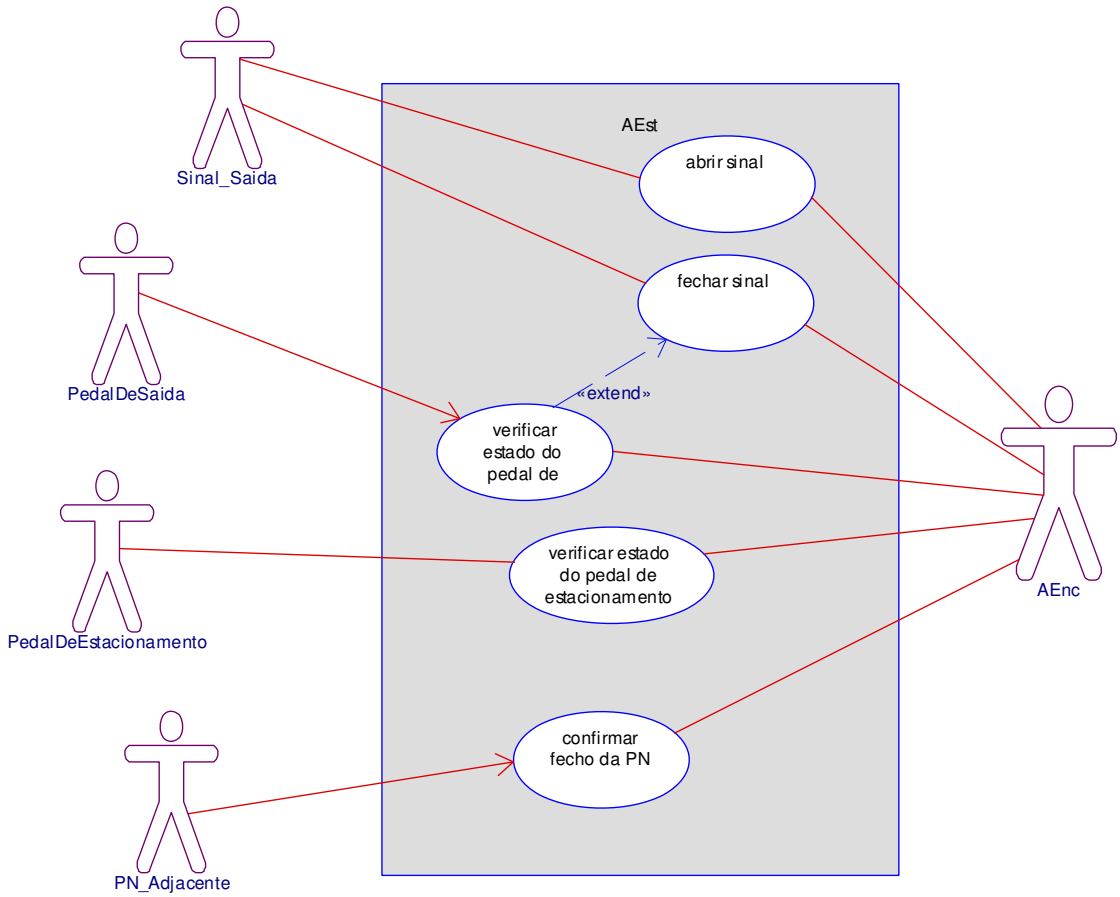


Figura 10 – Diagrama de casos de utilização do Autómato de Estação

4.2 Diagrama de classes

Este tipo de diagrama representa mais uma vista estrutural do software que se pretende desenvolver. Será a partir das classes definidas, que o Rhapsody irá gerar o código. Existem 3 tipos de classes: *Regular*, *Template*, ou *Instantiation*. Cada classe pode conter atributos, operações, associações e ligações (*links*) com outras classes, dependências e generalizações, *parts, ports*, e diagramas de estados (*statecharts*) e / ou de actividades. As classes podem ser normais ou então classes compostas (*composite classes*). Estas últimas chamam-se assim devido ao facto de poderem conter instâncias de outras classes, isto é por exemplo, uma classe camião pode ser composta por um objecto do tipo motor. Estes objectos são os chamados *parts*. Cada classe pode trabalhar no *thread* da classe principal, caso em que é denominada sequencial (*sequential*), ou então pode funcionar no seu próprio *thread*, isto é, em paralelo como o resto das classes, e neste caso a classe é activa (*active*).

Existem 5 tipos de operações no Rhapsody:

- Operações primitivas (*primitive operations*): são as operações simples, utilizadas para definir acções executadas pela classe ou pelas suas instâncias.
- *Triggered operation*: são operações que servem para desencadear transições de forma síncrona.
- Recepção (*reception*): são tipicamente eventos que, tal como as anteriores, desencadeiam transições nos diagramas de estados mas de forma assíncrona.

- Constructor: construtor da classe
- Destructor: operação que destrói uma instância da classe.

As ligações são instâncias de associações. De um modo geral as associações ligam as classes e os *links* ligam os objectos ou instâncias de classes. É possível definir as associações, ligações, generalizações e dependências entre classes pelo *browser*, mas através dos diagramas pode-se obter uma melhor visão destas.

O Rhapsody possibilita vários tipos próprios pré-definidos para os atributos e também outros derivados da linguagem escolhida para a geração de código. Esta ferramenta possibilita igualmente ao utilizador a oportunidade para definir tipos diferentes para alguns atributos. Existem cinco possibilidades para estes tipos:

- *Enumeration*: define-se um tipo de enumeração e os seus valores. Possibilidade de atribuir um inteiro para cada um dos valores.
- *Language*: Parecido com uma macro; define-se um tipo e o seu valor na linguagem escolhida para geração de código. Define-se por exemplo um tipo para o número de dígitos para o código de um alarme que se chamaria NUMERO_DE_DIGITOS:
  - const int %s = 4

  onde "%s" será substituído, aquando da geração de código, pelo nome do tipo, neste exemplo por NUMERO_DE_DIGITOS.
- *Structure*: é a definição de uma estrutura em C. Depois de lhe dar um nome define-se os tipos que se encontram neste tipo de estrutura. Podem ser de qualquer tipo pré-definido ou definido pelo utilizador, ou ainda uma classe.
- *Typedef*: permite redefinir um tipo de base num tipo que não existe na linguagem utilizada.
- *Union:* são tipicamente estruturas nas quais, em vez de haver uma zona de memória para cada atributo, existe uma zona de memória para a própria estrutura. Isto quer dizer que a variável só poderá ser de um dos tipos definidos na *union*. Por exemplo, dada uma *union* com um tipo inteiro e outro vector de caracteres, uma variável do tipo desta *union* poderá ser ou um inteiro ou um vector de caracteres.

4.2.1 Autómato de Encravamento

O Módulo de Software do Autómato do Controlador Central de Encravamento é aquele que gere todos os itinerários normais possíveis numa determinada linha de via única. Para tal necessita saber do estado em que os blocos (troços de via entre duas estações) se encontram, ter um retorno do estado dos equipamentos das diversas estações, e saber todas as condições relativas à formação, realização e destruição / anulação de cada itinerário. Tendo estes dados optou-se por definir quatro classes. São elas:

- Encravamento: classe principal
- Estação: classe que contém todos os equipamentos com os diversos estados
- Bloco: classe que gere os itinerários possíveis no bloco
- Itinerário: classe representante de um itinerário.

O passo a seguir foi o de definir quais eram as ligações entre estas várias classes. Era natural definir a classe Encravamento como uma classe composta, pois o encravamento é a gestão de uma linha de comboio, composta por estações e blocos. No caso do protótipo estudado, com três estações e portanto dois blocos, a classe Encravamento seria composta por 3 objectos do tipo Estacao e dois do tipo Bloco. Por sua vez um bloco contém dois itinerários, portanto a classe Bloco será igualmente uma classe composta por dois objectos do tipo Itinerário (representando os dois itinerários possíveis, em sentidos inversos). No entanto, sabe-se que os itinerários são mutuamente exclusivos, isto é, só um pode estar formado em cada bloco. Para tal define-se uma associação unidireccional da classe Bloco para a classe Itinerário. Esta classe representa o itinerário formado. Existe também uma relação entre os blocos e as estações, pois um bloco está ligado a duas estações e uma estação está ligada a um ou dois blocos, dependendo se esta estação se encontra no meio ou no extremo da linha. Portanto tendo definido estas ligações verificamos que seriam necessários no mínimo dois diagramas, mas optou-se por três, realizando desta forma um diagrama que retrata exclusivamente as associações entre Bloco e Itinerário. No final obtém-se os três diagramas das seguintes figuras:

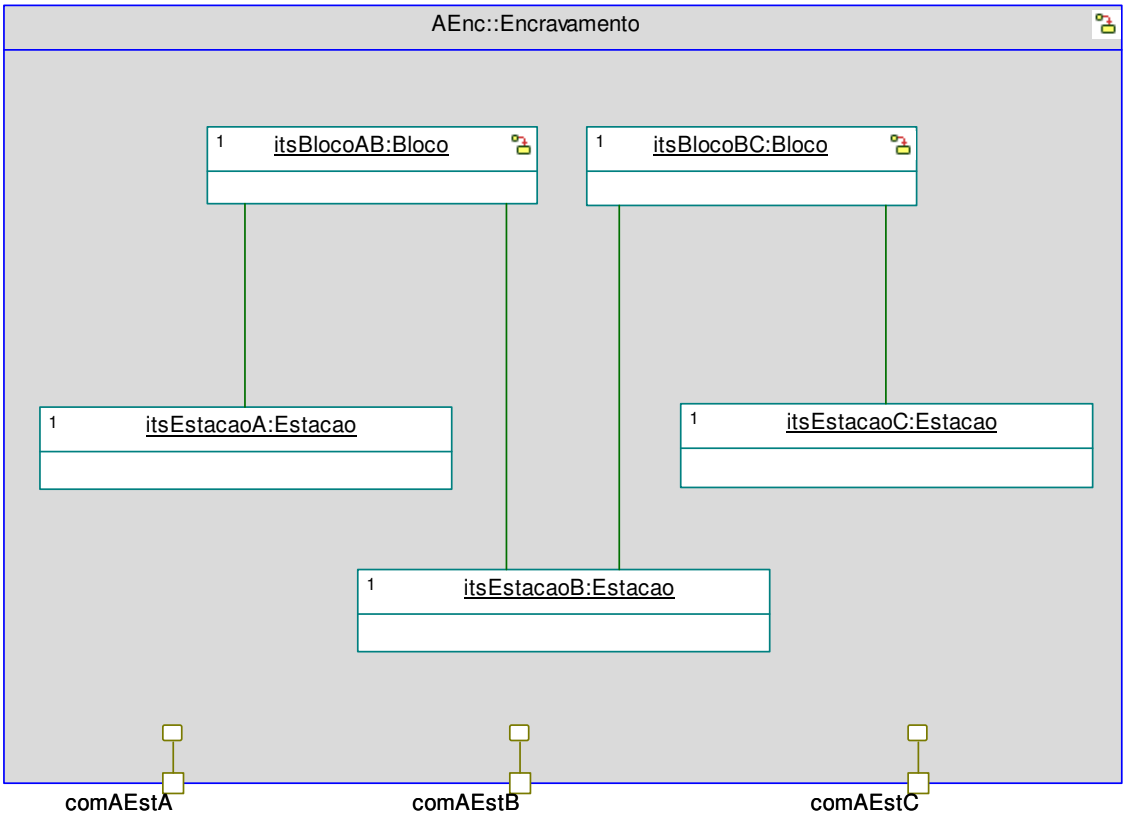


Figura 11 – Diagrama de classes definindo a *composite class* Encravamento

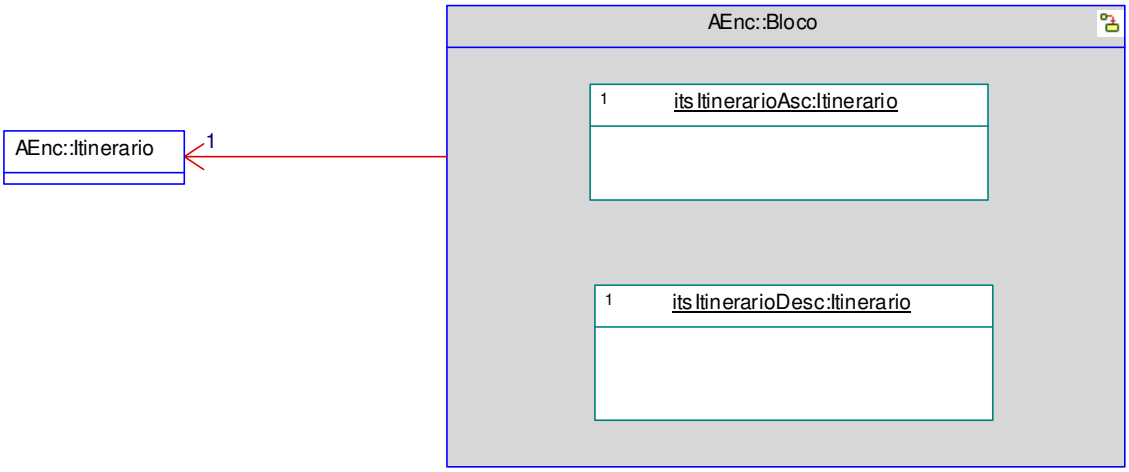


Figura 12 – Diagrama definindo a *composite class* Bloco associada à classe Itinerario

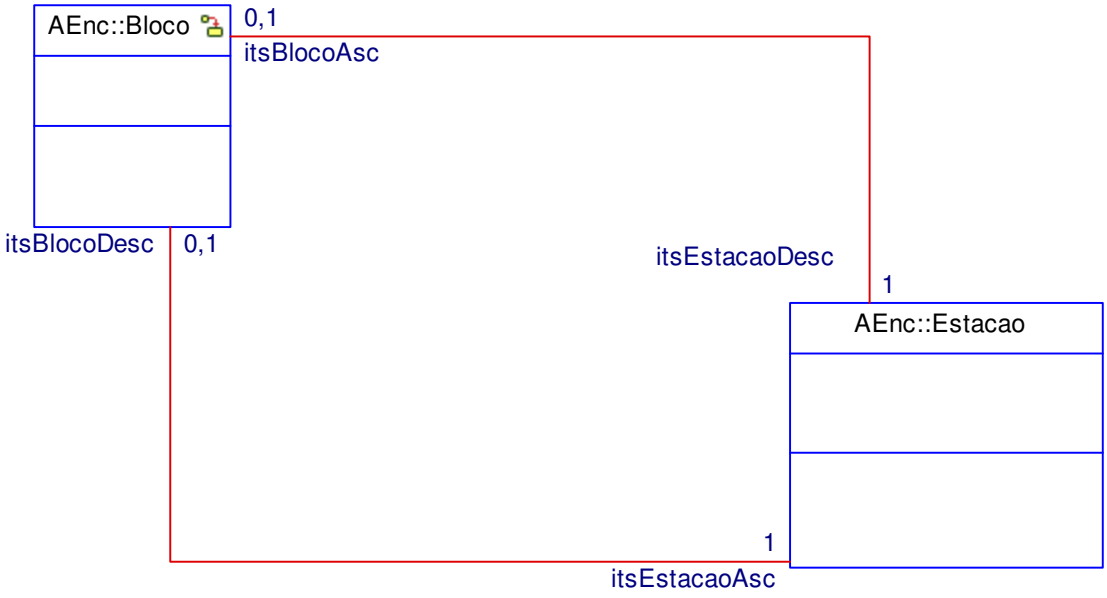


Figura 13 – Diagrama definindo as associações entre a classe Estacao e a classe Bloco

Na última figura nota-se que existem duas associações. Isto deve-se ao facto de se ter preferido definir dois tipos de associações, isto é, um bloco tem uma estação ascendente e uma estação descendente, e no caso de uma estação, esta pode ter um bloco ascendente e outro descendente. Salienta-se que as ligações no primeiro diagrama entre os *parts* de tipo Bloco e os *parts* de tipo Estacao são instâncias das associações entre a classe Bloco e a classe Estação.

Nota: uma linha é definida de uma estação inicial até uma estação final sendo que o ponto quilométrico (Pk) da primeira é 0km e o da segunda é igual ao comprimento da linha. O

sentido ascendente é o sentido do Pk menor até ao Pk maior e o sentido descendente é o sentido inverso. Para mais informações ver anexo A.

4.2.2 Autómato de Estação

O Módulo de Software do Autómato dos Controladores de Estação é aquele que deve controlar os equipamentos de estação. A tarefa deste módulo é o de verificar os estados dos diversos aparelhos, bem como executar comandos de abertura dos sinais luminosos de ferrovia, através da recepção de comandos vindos do autómato central de encravamento. Cada vez que acontece uma mudança de estado de um dos equipamentos ele deverá informar o autómato de encravamento. Outro aspecto importante, é o facto de o número de equipamentos variar consoante o tipo de estação controlada pelo autómato. Quer isto dizer que se o autómato controla uma estação que se encontra no meio da linha, este deverá controlar 2 sinais luminosos, 2 pedais de estacionamento, 2 pedais de saída e eventuais passagens de nível (PN). Se o autómato controlar uma estação que se encontra num dos extremos da linha, este deverá controlar apenas 1 sinal luminoso, uma eventual PN, um pedal de estacionamento e outro de saída. Salienta-se que o accionamento de um pedal de saída implica o fecho do sinal que se encontra no mesmo sentido que o pedal.

Tendo em conta todos estes aspectos, decidiu-se implementar uma classe de controlo e uma para cada tipo de equipamento. Mas devido ao facto de haver dois tipos de estação optou-se por representar as classes em três *packages* diferentes. Um no qual estão as classes respeitantes aos equipamentos, e nas outras, coloca-se uma classe de controlo de estação em cada uma sendo que uma das classes diz respeito ao controlo de uma estação intermédia (do meio da linha), e a outra, ao controlo de uma estação final (extremo da linha). Obtém-se assim:

- *Package* AestFinal:
  - ControloEstacaoF: classe principal de um autómato de estação que controla uma estação do extremo da linha.
- *Package* AestNormal:
  - ControloEstacao: classe principal de um autómato de estação que controla uma estação do meio da linha.
- *Package* Equipamentos:
  - Sinal: classe que controla um sinal;
  - PdS: classe refente a um pedal de saida;
  - PdE: classe referente a um pedal de estacionamento;
  - Pnadjacente: : classe referente a uma Passagem de nível adjacente.

Tendo definido as classes, o próximo passo consiste em definir quais são as relações existentes entre elas. As classes ControloEstacaoF e ControloEstacao são as classes principais e representam a estação. Ora se for uma estação final ela é composta por um sinal, um pedal de saída e outro de estacionamento. Pode eventualmente conter uma PN adjacente. Se for uma estação intermédia ela é composta por dois sinais, dois pedais de saida e dois de estacionamento. Pode eventualmente conter 0, 1 ou 2 PNs adjacente.

Portanto, tem-se

- ControloEstacaoF é uma classe composta por uma intância da classe Sinal, outra da classe PdS e outra da classe PdE e 0 ou 1 da classe Pnadjacente;
- ControloEstacao é uma classe composta por duas instâncias da classe Sinal, duas da classse PdS, duas da classe PdE e 0, 1, ou 2 da classe Pnadjacente.

Existe apenas mais uma relação entre classes e diz respeito às classes PdS e Sinal, pois o accionamento de um pedal de saída implica o fecho do sinal (se este se encontrar aberto). Neste sentido cria-se um associação unidireccional da classe PdS para a classe Sinal.

Obtêm-se desta forma os seguintes três diagramas:

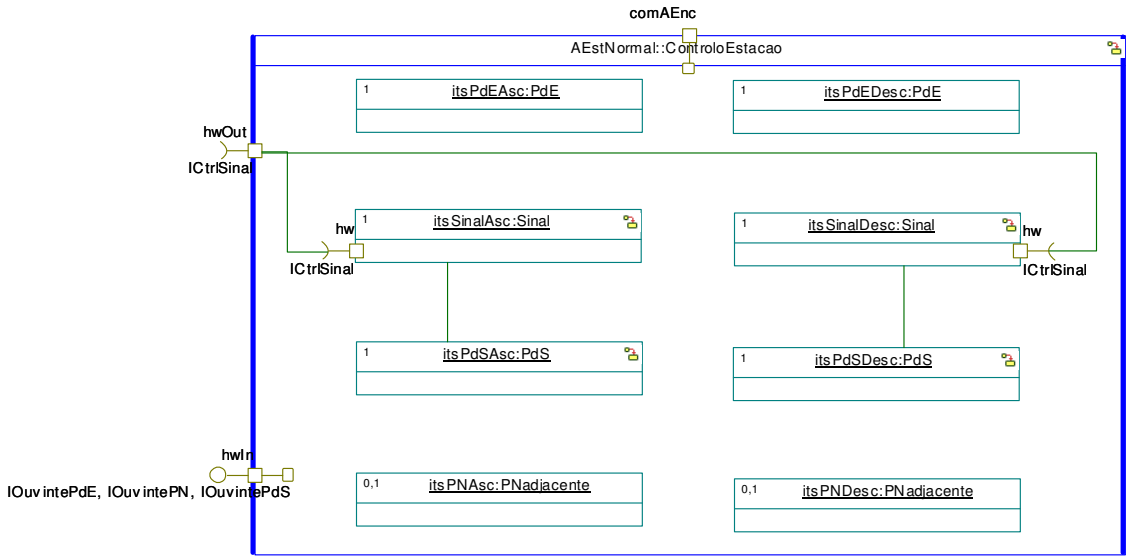


Figura 14 – Diagrama definindo a *composite class* ControloEstacao

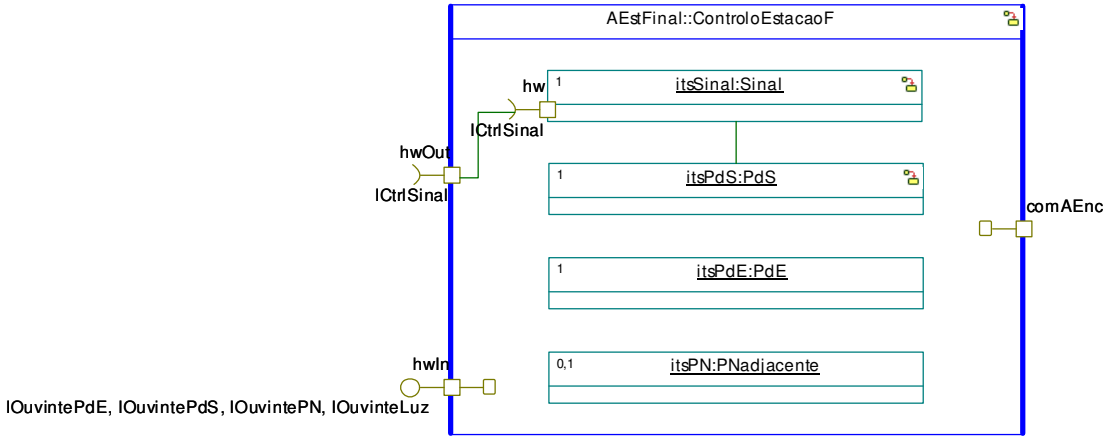


Figura 15 – Diagrama definindo a *composite class* ControloEstacaoF

Figura 16 – Diagrama definindo a associação entre a classe PdS e a classe Sinal

4.3 Diagrama de estados

Depois de definir as classes, é necessário definir os comportamentos ou os vários estados que as suas instâncias podem ter. Para tal associam-se diagramas de estados e / ou actividades as classes. Estes diagramas são responsáveis pela maior parte do código gerado. No caso do projecto de sinalização ferroviária, tratando-se de um sistema de tempo real que reage principalmente a eventos, optou-se por utilizar os diagramas de estados para definir o comportamento de algumas das classes definidas.

Um dos aspectos interessantes nos *statecharts* do Rhapsody é que há a possibilidade de colocar nas transições, condições de guarda requerendo que determinado objecto ou subestado (no caso de estados AND) se encontre num determinado estado graças à macro IS_IN(*state*). Refira-se que é necessário escrever algum código para as operações correspondentes a acções presentes nas transições ou nos estados.

4.3.1 Autómato de Encravamento

Neste *package*, as classes das quais importa modelar o comportamento são as classes Encravamento e Bloco. A primeira por ser a principal, sendo ela a receber o comando de pedido de formação e aquela que comunica com os autómatos de estação. A segunda porque é no bloco que se realiza o itinerário, e convém portanto definir qual o comportamento e quais são os eventos que desencadeiam uma transição de um estado para outro.

Apresentam-se nas figuras seguintes os diagramas de estados destas duas classes.

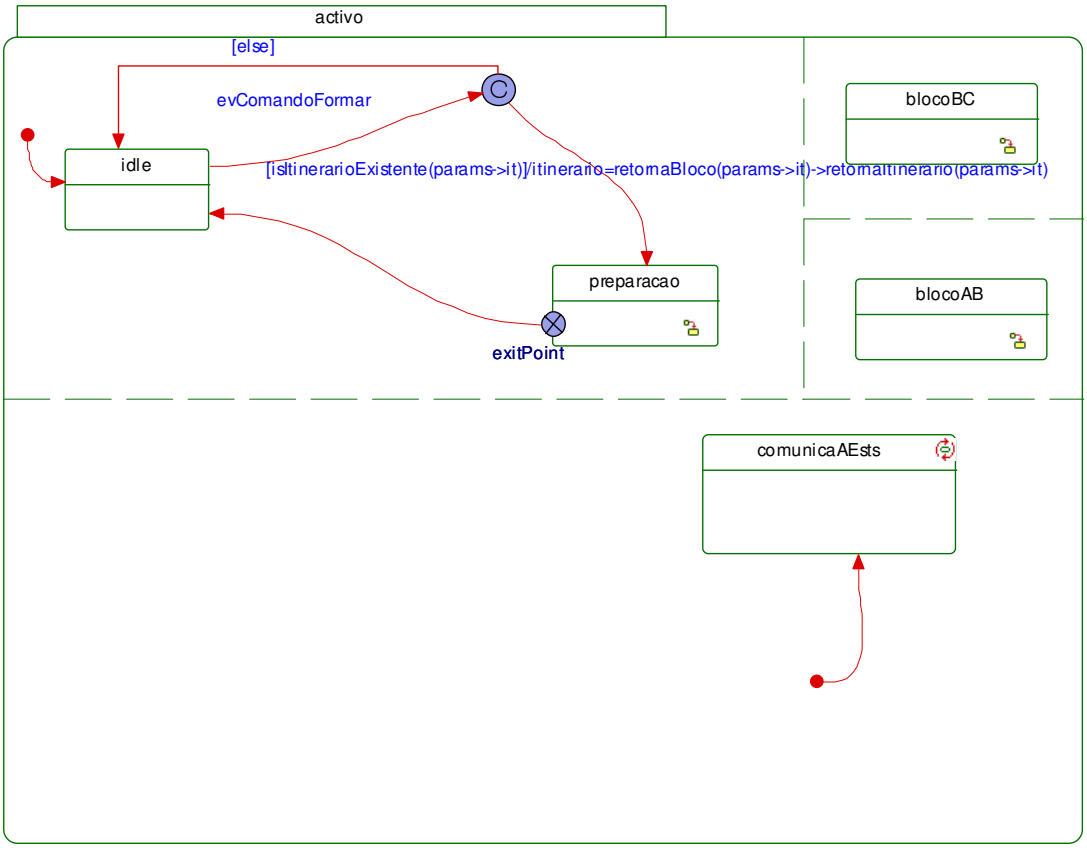


Figura 17 – *Statechart* da classe Encravamento

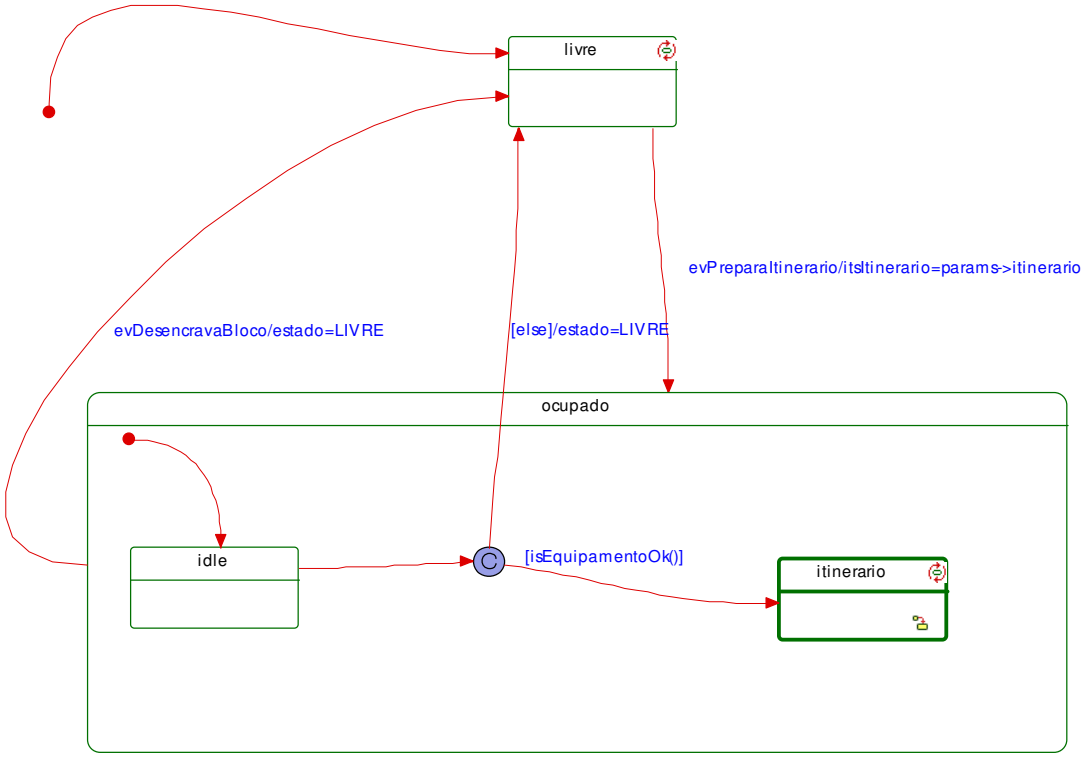


Figura 18 – *Statechart* da classe Bloco

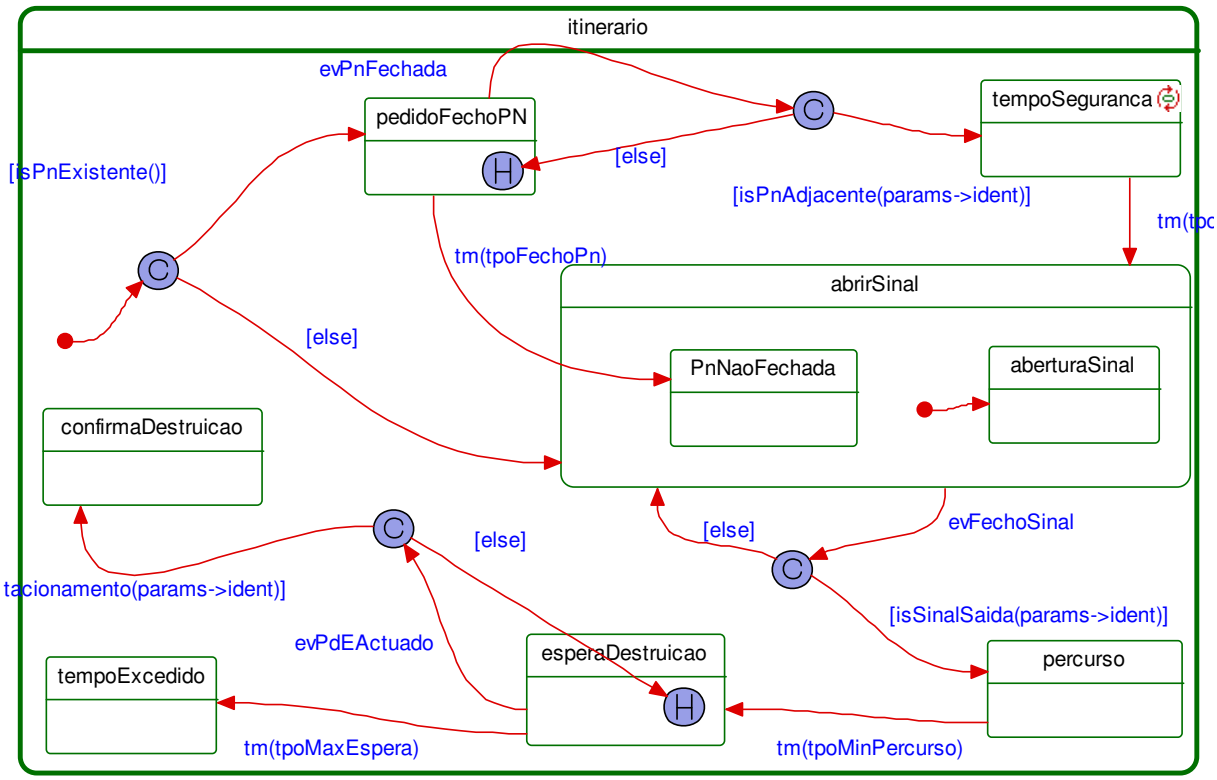


Figura 19 – *Statechart* do sub-estado itinerario do *statechart* da classe Bloco

Statechart da classe Encravamento

O estado `preparacao` encarrega-se de pedir os estados dos equipamentos às estações envolvidas num itinerário e de verificar se o bloco está disponível.

Os macro-estados `blocoAB` e `blocoBC`, do *statechart* da classe Encravamento, correspondem à realização do itinerário, i.e., o comboio encontra-se no bloco. Estão separados do resto para permitir que, logo que um itinerário tenha sido formado, ou seja o comboio autorizado a sair da estação, o Chefe de Linha possa pedir a formação de um outro itinerário. Desta forma, permite-se que possa haver um itinerário formado em cada bloco. Estes estados verificam em que estado se encontra o objecto do tipo Bloco correspondente para enviar comandos aos autómatos de estação implicados no itinerário.

O estado `comunicaAEsts` espera por eventos enviados pelas classe ControloEstacao e ControloEstacaoF, contendo como argumentos destas operações os estados de um ou de todos os equipamentos.

Statechart da classe Bloco

O diagrama do sub-estado `itinerario` mostra todas as fases possíveis para a realização de um itinerário, e quais os eventos que levam à mudança de estado.

4.3.2 Autómato de Estação

Neste *package* definiu-se um diagrama de estado apenas para as classes de controlo, a classe Sinal e a classe PdS. No caso das classes de controlo (ControloEstacao e ControloEstacaoF) os diagramas são idênticos. Dispõem apenas de um estado

(`comunicaAEnc`) que, tal como para a classe Encravamento, recebe eventos que desta vez correspondem a comandos de controlo do sinal e pedidos dos estados dos equipamentos. A classe Sinal requer a especificação do comportamento, porque a mudança de estado implica a execução de um comando. Como ilustrado na figura abaixo existem dois macro-estados AND, sendo que o de cima corresponde ao controlo do sinal e o segundo verifica a correspondência do estado do sinal. Caso haja alguma anormalidade ele requer o seu fecho imediato.

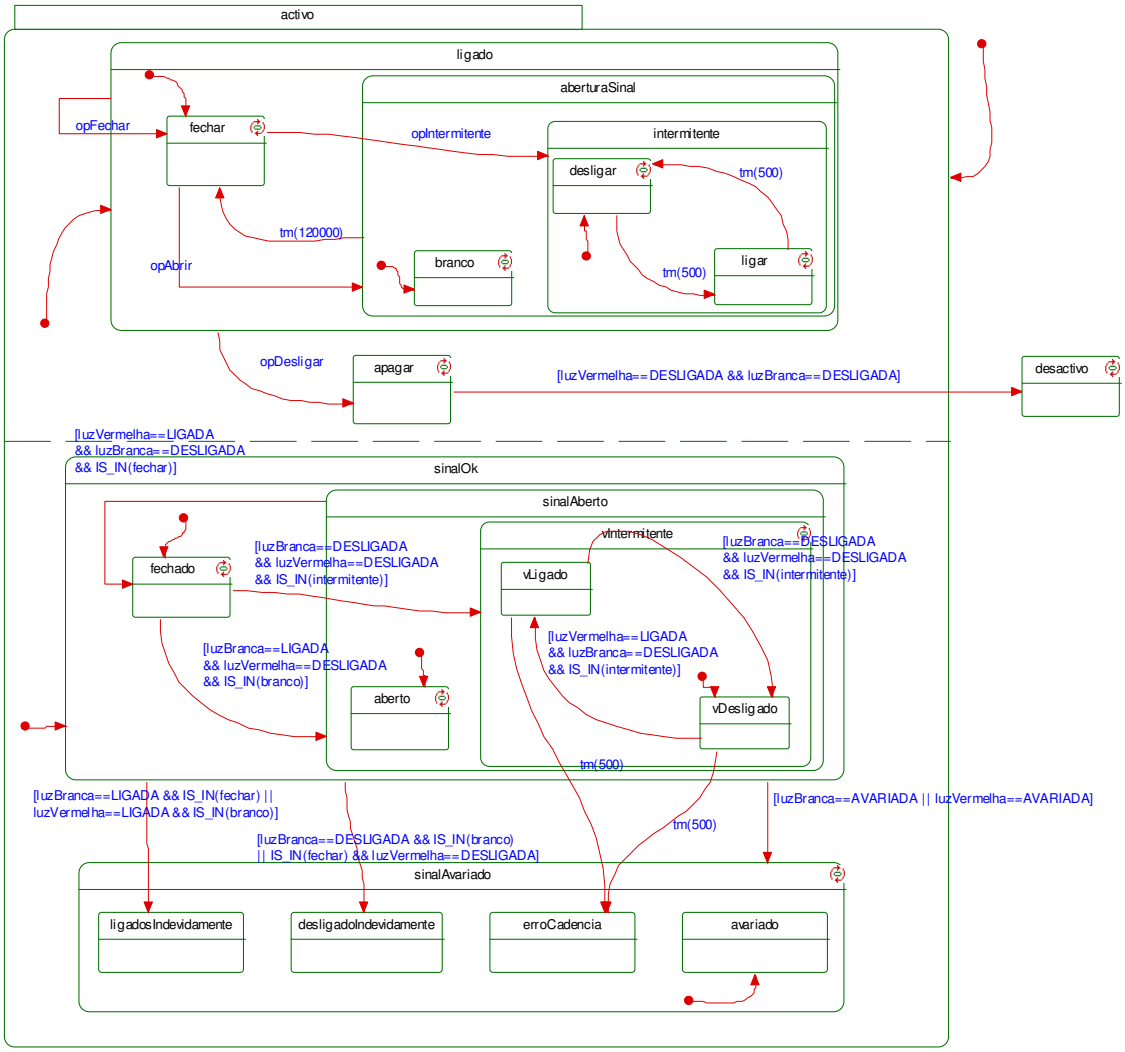


Figura 20 – *Statechart* da classe Sinal

A classe PdS, que representa um pedal de saída, requer igualmente a implementação de um *statechart* devido ao facto de que quando este se encontrar no estado accionado, ele gera um evento para o sinal de forma a provocar o seu fecho.

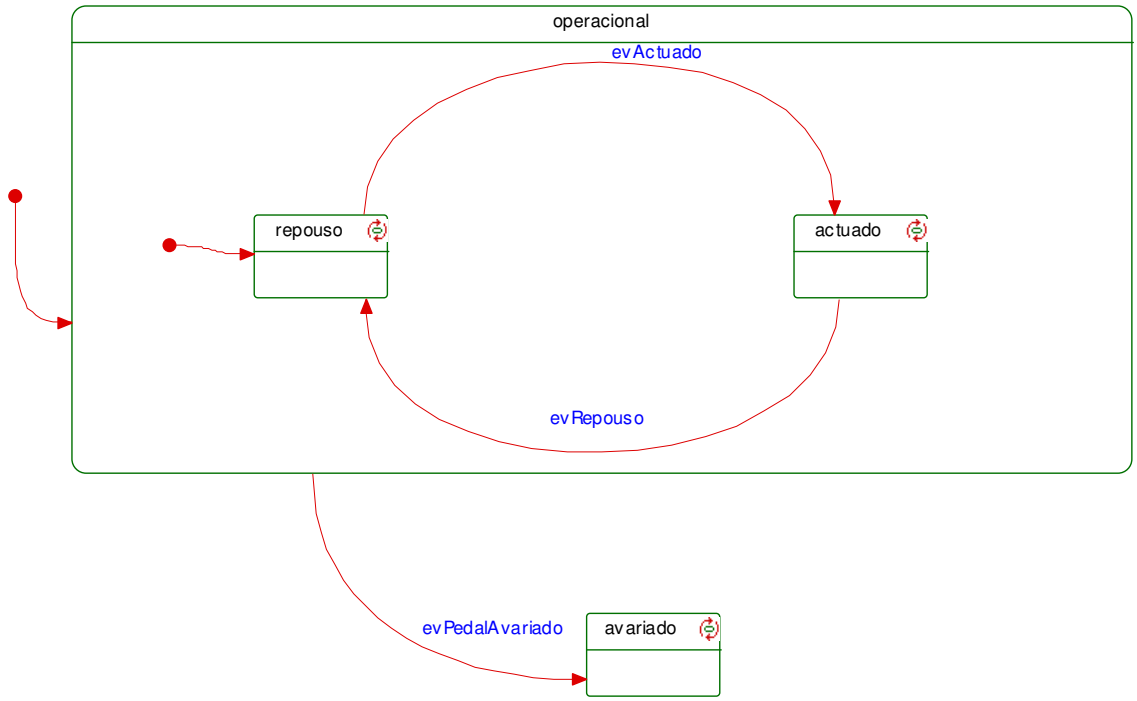


Figura 21 – *Statechart* da classe PdS

4.4 Opções tomadas

Nesta secção referem-se algumas opções tomadas. No que diz respeito a vários atributos relativos aos estados, foram definidos como tipos enumerados. Por exemplo o estado de um sinal pode ter os valores: APAGADO – FECHADO – ABERTO – INTERMITENTE. Como se trata de um sistema de segurança crítica, requer-se limitar as opções que uma variável pode tomar. E esta acaba por ser uma das melhores soluções.

A comunicação entre os autómatos não foi modelada. Sendo que a única possibilidade é a modelação de comunicação do tipo COM, e sabendo que os autómatos têm o seu próprio protocolo de comunicação, que seria mais do tipo TCP / IP, optou-se por não a modelar. Portanto limitou-se a incluir *behavioral ports* no qual se geram eventos para a outra classe.

Quanto à linguagem, sendo que a versão de demonstração do Rhapsody possibilita apenas C++ e Functional C, optou-se pela linguagem C++.

4.5 Simulações

Estas simulações foram observadas através da animação dos *statecharts*. Elas dizem respeito a três casos de utilização do autómato de Encravamento. São eles:

- Abrir Sinal
- Fechar Sinal
- Detectar Actuação do Pedal de Saída

Os testes foram nesta sequência no que diz respeito à classe Sinal e ao seu *statechart*:

1. Envio de um comando de abertura do sinal branco
2. Envio de um comando de fecho de sinal
3. Envio de um comando de abertura do sinal intermitente
4. Espera até fecho do sinal
5. Envio de um comando de abertura do sinal
6. Detecção da Actuação do pedal de saída e consequente fecho do sinal
7. Avaria do Sinal

Os testes para o pedal de saída foram nesta sequência:

1. Actuação do pedal
2. Pedal em repouso
3. Avaria do pedal

As figuras seguintes ilustram as duas destas fases de forma a dar um exemplo:

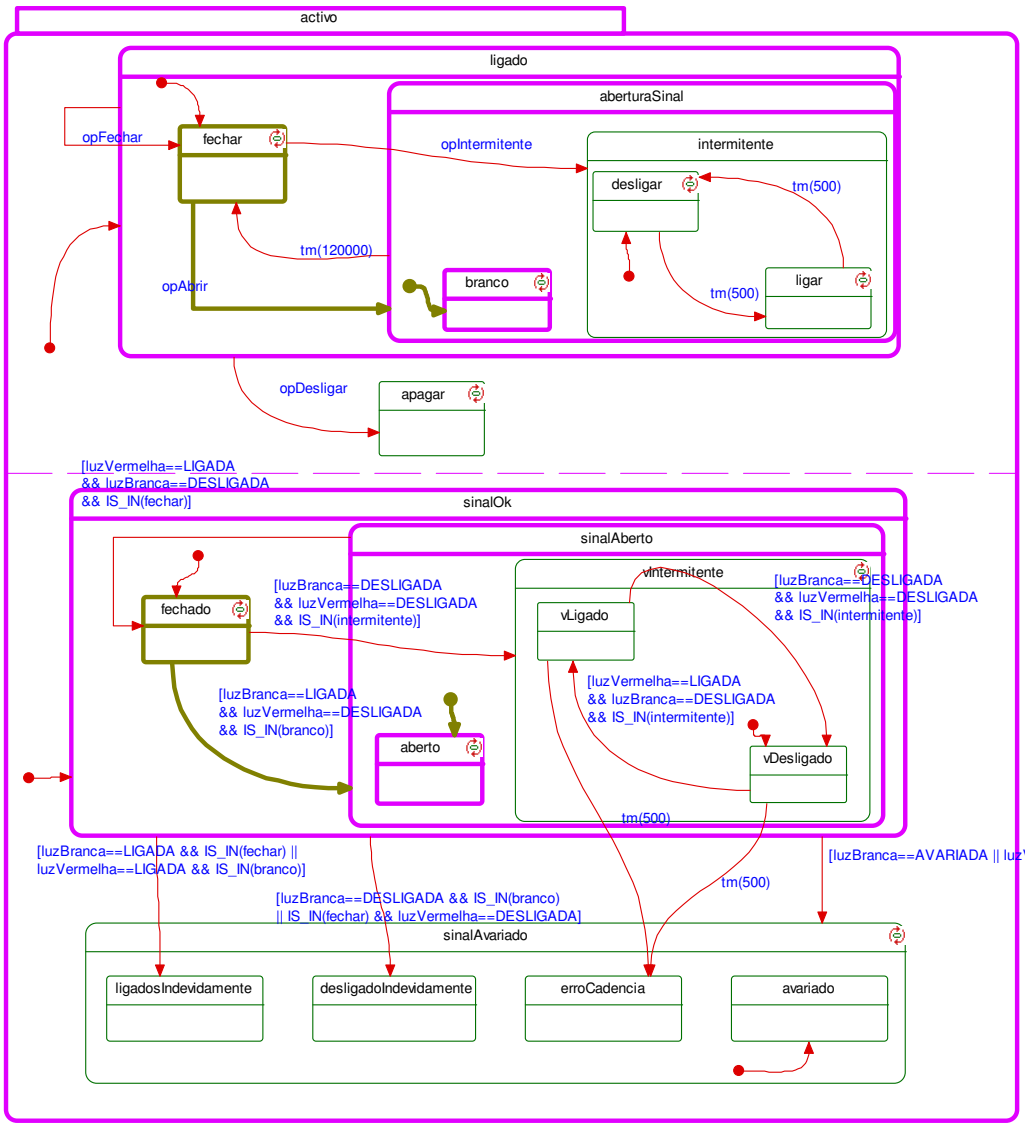


Figura 22 – *Animated statechart* correspondente às fases 1 e 5 do sinal

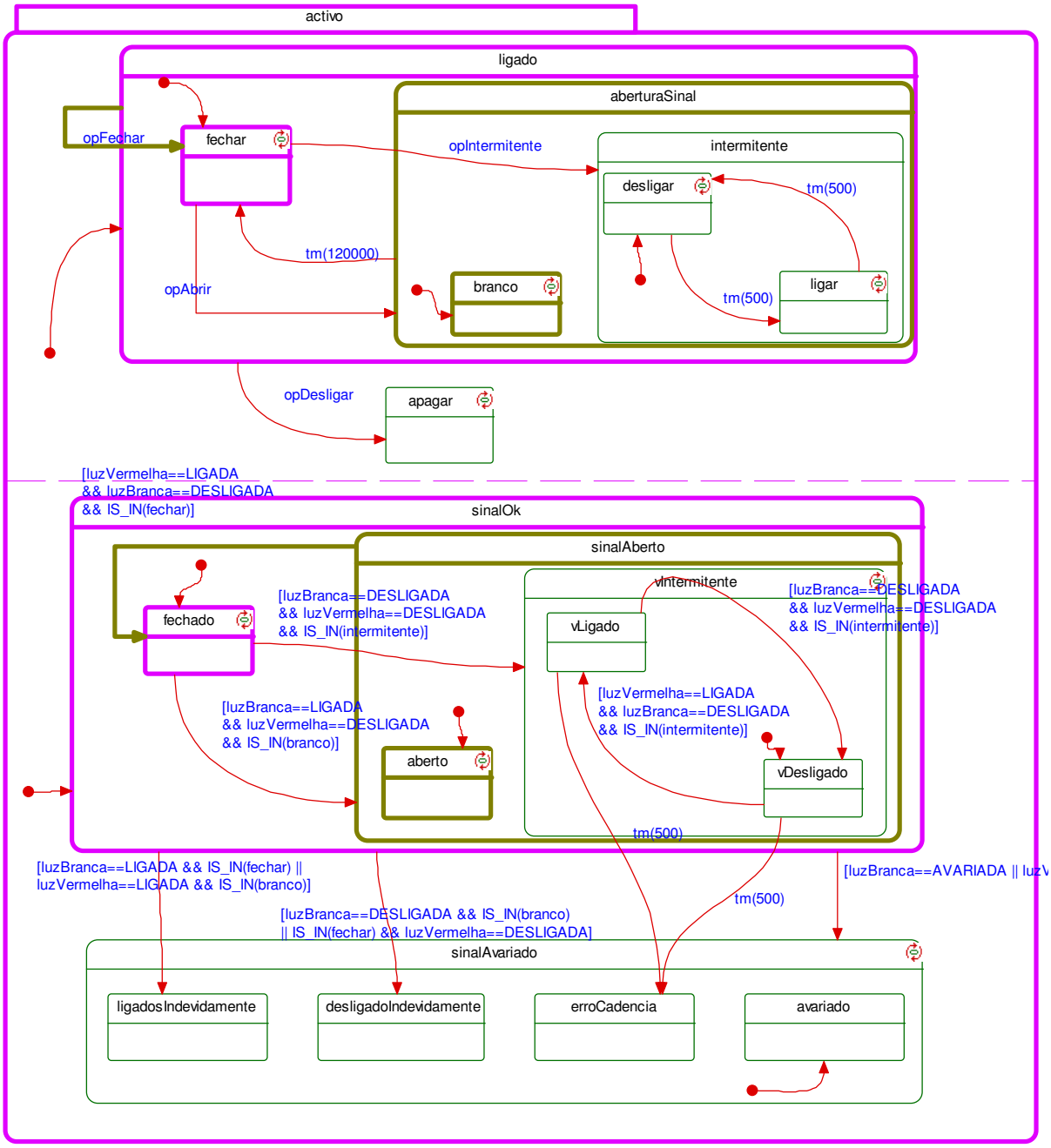


Figura 23 – *Animated statechart* relativo às fases 2 e 6 da sequência do sinal

No primeiro diagrama gerou-se um evento do tipo envio de comando abrir sinal. No segundo gerou-se um comando de fecho de sinal, ou por recepção de um comando, ou por actuação do pedal.

Capítulo 5

Conclusões e trabalho futuro

O projecto contemplou a utilização de UML (*Unified Modeling Language*) de duas formas: como técnica para especificação funcional, nomeadamente utilizando o diagrama de casos de utilização, e como técnica de modelação de um sistema, utilizando para desenho dos modelos uma ferramenta para UML tempo real. Este sistema (Sistema de Encravamento) visa apoiar a gestão de uma linha ferroviária de baixo tráfego, controlando os equipamentos de via, nomeadamente os sinais luminosos que se encontram nas estações.

A ferramenta para a modelação, escolhida entre várias como Tau G2 da Telelogic, foi o Rhapsody também da Telelogic (comprada há sensivelmente 3 meses pela IBM), utilizada em diversas áreas que envolvem desenvolvimentos de sistemas tempo real, como é o caso da ferrovia. É uma ferramenta que disponibiliza um vasto leque de opções relacionadas com UML, geração de código a partir dos modelos, animação dos modelos, teste dos modelos e também análises de sistemas.

5.1 Satisfação dos objectivos

Inicialmente os objectivos contemplavam a modelação do sistema com todas as suas funcionalidades e posteriormente a simulação dos modelos. Uma série de razões levaram a que os objectivos ficassem reduzidos à implementação de um dos casos de utilização, o Formar Itinerário. Este caso de utilização tem como fim receber um comando vindo do Posto Central de Comando (PCC) e verificar se há condições para autorizar o comboio a entrar no troço de via única entre duas estações, chamado bloco. Se as condições são verificadas, o sistema permite a formação do itinerário, i.e., abre o sinal de saída para o comboio entrar no bloco. Se alguma das condições não é verificada, como por exemplo a presença de um outro comboio no bloco, o sistema não forma o itinerário, isto é, não autoriza o comboio a entrar no troço de via.

As razões que levaram à redução dos objectivos iniciais são:

- Falta de experiência na aplicação de métodos a utilizar, nomeadamente a análise HAZOP, que originou o atraso na definição das protecções a implementar de forma a evitar situações perigosas.
- Leitura das normas relativas ao sector da ferrovia, em particular a EN50126 e a EN50128.
- Manuais de utilização da ferramenta completos mas extensos. Não houve tempo de aprender a utilizar as potencialidades da ferramenta, sendo que, segundo os próprios vendedores, é bastante difícil de explorar.

- Atraso no fornecimento da versão de demonstração devido à compra da Telelogic por parte da IBM e consequente mudança de política.

Com estas condicionantes e revistos os objectivos, estes acabaram por ser satisfeitos a nível da modelação, tendo incluído, gerado e compilado o código, e também tendo preparado o modelo para a implementação das outras funcionalidades. Faltou apresentar algum resultado a nível de simulação, isto é, um teste em que se tentava formar itinerário em diversas situações como:

- Formar itinerário verificando todas as condições
- Formar itinerário estando o bloco já ocupado
- Formar itinerário com equipamentos de estação avariados
- Formar itinerário com passagem de nível adjacente não fechada

Há no entanto alguns casos de utilização que foram simulado em parte, referente ao autómato de estação. São o Abrir Sinal, Fechar Sinal, Detectar Actuação do Pedal de Saída. Verifica-se através da animação dos *statecharts* das classes Sinal e PdS.

Este é um projecto bastante motivante, foi pena o tempo ser demasiado curto para um projecto desta complexidade.

5.2 Trabalho futuro

O trabalho futuro passa essencialmente por efectuar a simulação do primeiro caso de utilização do autómato de Encravamento (Formar Itinerário). Depois implementar-se-á o resto dos casos de utilização para simular todos os testes possíveis, que serão definidos posteriormente pelos especialistas de sinalização ferroviária, de forma a verificar que os modelos cumprem os requisitos de segurança. O objectivo inicial seria a de importar as classes para o SCADE, que a partir delas, geraria código C certificado segundo as normas IEC 61508 e EN50128.

Outra hipótese seria a de efectuar toda a modelação em SCADE, o que poderia ser mais viável de um ponto de vista económico.

Neste momento o protótipo é realizado através de *Sequencial Function Chart* para uns autómatos Hima, autómatos estes de nível de integridade de segurança 3 (SIL 3), que possui o seu próprio protocolo de comunicação (Safeethernet). A lógica está a ser implementada com os *Sequencial Function Chart* seguindo os modelos definidos na especificação. O objectivo final é que no caso de avançar para um projecto real, utilizar autómatos industriais com um código de alto nível, baseado na modelação realizada.

Referências

[EFACEC] Efacec Sistemas Electrónica, S.A. – Especificação de requisitos de SW do Sistema de Encravamento do Sistema de Sinalização para Linha Ferroviária de Baixo Tráfego – 18 Abril de 2008

[NP EN 50126] Aplicações Ferroviárias – Especificação e Demonstração de Fiabilidade, Disponibilidade, Manutibilidade e Segurança (RAMS) – 2006

[NP EN 50128] – Aplicações Ferroviárias – Sistemas de Sinalização, Telecomunicações e de Processamento de Dados – Software para Sistemas de Protecção e Comando Ferroviário – 2001

[NP EN 50129] – Aplicações Ferroviárias – Sistemas de Sinalização, Telecomunicações e de Processamento – Sistemas Electrónicos de Segurança para Sinalização – 2001

[MeMVaTEx] – Arnauld Albinet, Marie-Agnès Péraldi-Frati, Dumitru Potop-Butucaru, Yves Sorel, Olivier Casse, Jean-Louis Boulanger, Hubert Dubois, Stephane Badreau et Sylvain Begoc – Méthode de Modélisation pour la Validation et la Traçabilité des Exigences – Projects et Outils du domaine de MeMVaTEx – 1 de Março de 2007

[UIC] Union Internationale des Chemins de Fer – The Euro-Interlocking Consortium – Standards for Interlocking Systems in Europe

[DOC_IEC_61508] Dossier sécurité de process – Normes – L'IEC 61508 s'impose, sa famille aussi – Novembro de 2005

[Wikipedia] Méthode B – http://fr.wikipedia.org/wiki/Methode_B

Redes de Pétri – http://pt.wikipedia.org/wiki/Redes_de_petri

Signalisation Ferroviaires – http://fr.wikipedia.org/wiki/Signalisation_ferroviaire

Système Européen de contrôle des trains – http://fr.wikipedia.org/wiki/Système_européen_de_contrôle_des_trains

[EURIS] Fokko van Dijk, Wan fokkink, Gea Kolk, Paul Van de Ven, Bas Van Vlijmen – EURIS, a Specification Method for Distributed Interlockings

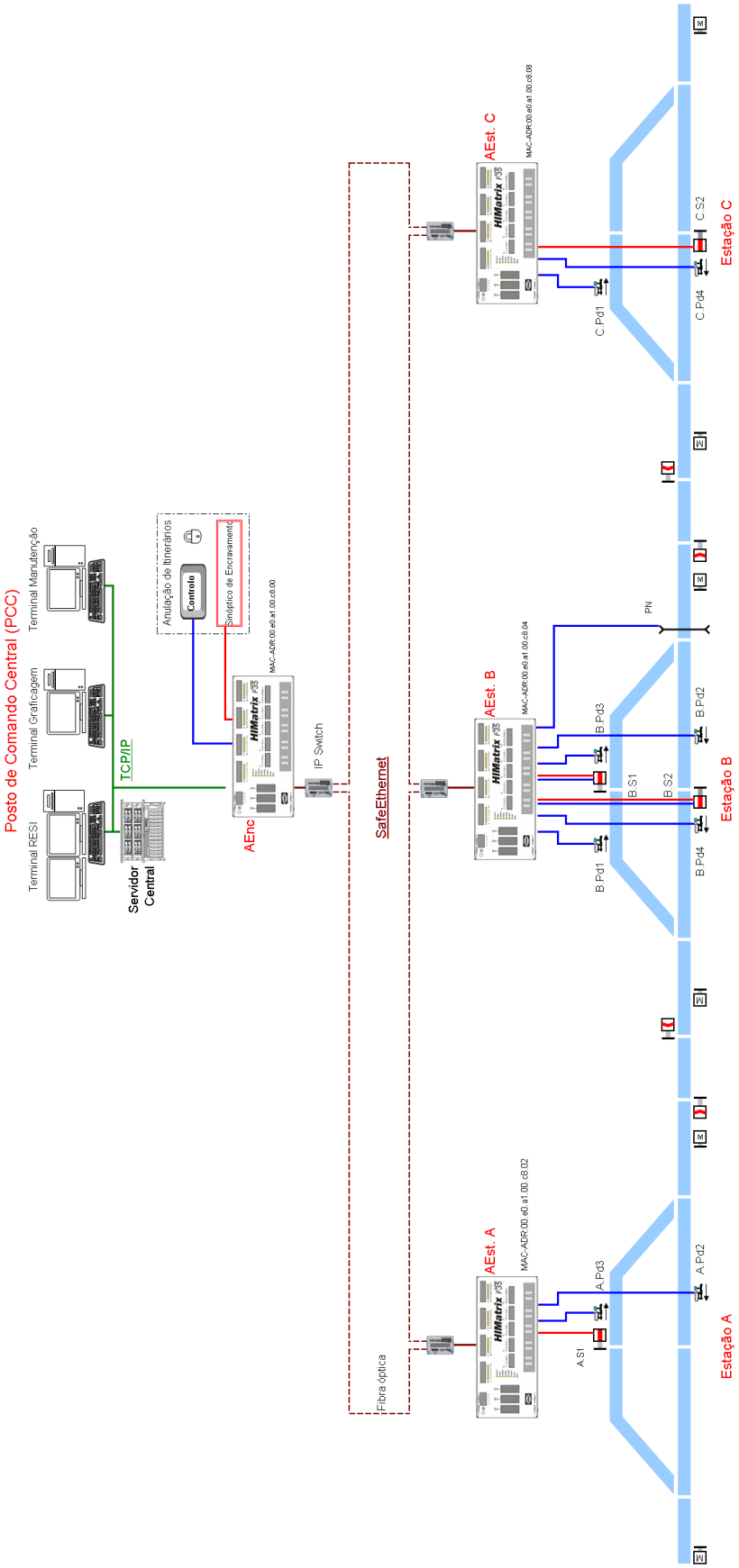
[EURIS_LARIS] Wan Fokkink, Jan Friso Groote, Marco Hollenberg and Bas van Vlijmen – LARIS 1.0, LAnguage for Railway Interlocking Specification

[CNRS / UTC] Jean-Louis Boulanger et Karim Berkani – Centre National de la Recherche Scientifique / Université de Technologie Compiègne – UML et la conception de système certifiable: problématique – Lille – 24 de Março de 2005
[CERTIFER] Jean-Louis Boulanger – CERTIFER – Certification d'Application Ferroviaire développée formellement ou non – 3 de Fevereiro de 2004 - http://www.certifer.asso.fr/

[SNCF] Service National des Chemins de fer Français – Implementation of a formal proof of interpretable specification for interlocking system – Braunschweig – 26, 27 de Janeiro de 2007

[SCA_SDL] Stefano Bacherini, Simone Bianchi, Leonardo Capecchi, Carlo Becheri, Alessandro Felleca, Alessandro Fantechi, Emilio Spinici – Modelling a Railway Signalling System Using SDL

ANEXO A: Arquitectura do Sistema



A.1 Breve descrição

A arquitectura está definida no capítulo 2 do relatório. Portanto neste anexo descreve-se os equipamentos de via e as suas respectivas colocações.

Convém informar primeiro que na ferrovia existem nomes específicos para descrever os sentidos. Estes estão relacionados com os pontos quilométricos (Pk), sendo que a estação mais a esquerda é aquela com o Pk menor e a estação mais a direita aquela com o Pk maior. Portanto o sentido da esquerda para a direita é dito ascendente e o sentido da direita para a esquerda é dito descendente. No caso do projecto em questão, quando o comboio realiza um itinerário ascendente, ao chegar a uma estação, ele colocar-se-á na linha de cima e, se realiza um itinerário descendente colocar-se-á na linha de baixo.

Reportando-se à figura, s1 e s2 representam os sinais luminosos de saída. Sendo que s1 é o sinal Ascendente da estação e s2 o sinal descendente. Pd3 e Pd4 representam os pedais de saída, indicando ao sistema, aquando da sua actuação, que o comboio está a sair da estação. O primeiro representa o pedal de saída ascendente e o segundo o pedal de saída descendente. Pd1 e Pd2 são os pedais de estacionamento, indicando ao sistema que o comboio chegou à estação. Pd1 é o pedal de estacionamento ascendente enquanto que Pd2 é o pedal de estacionamento descendente. Da mesma forma uma PN (Passagem de Nível) que se encontra a direita da estação é a PN Ascendente e um PN que se encontra a esquerda da estação é a PN descendente.

A.2 Autómatos

Os autómatos representados no esquema são autómatos de segurança, com SIL 3 (Nível de Integridade de Segurança). Estes autómatos comunicam entre si segundo um protocolo de comunicação próprio, designado como Safeethernet que, como o nome o indica, é um protocolo seguro.