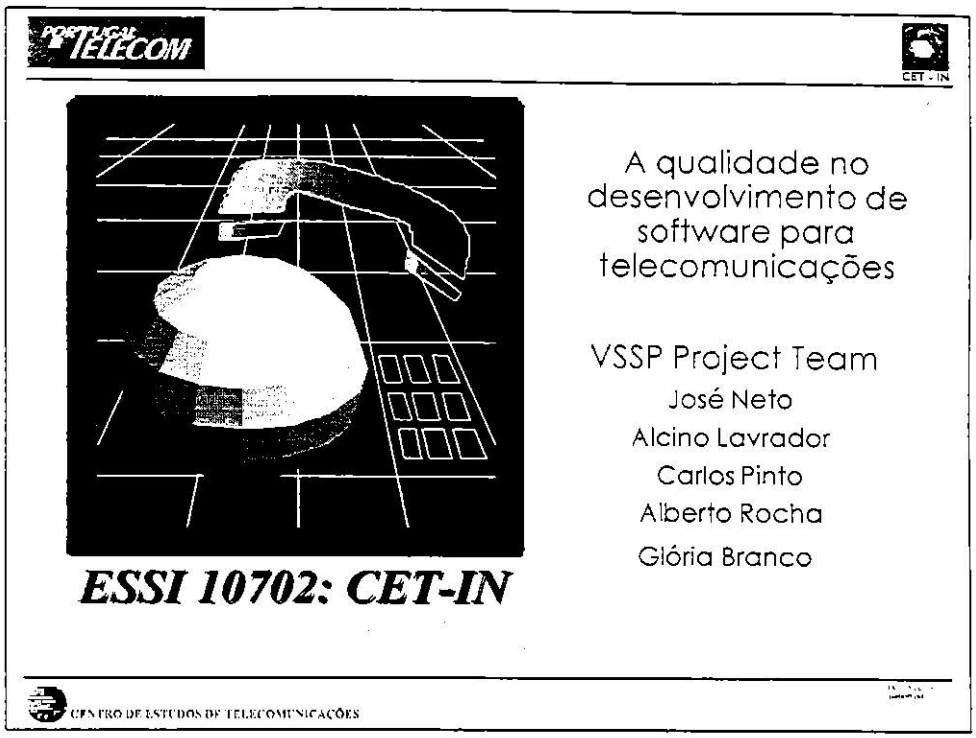


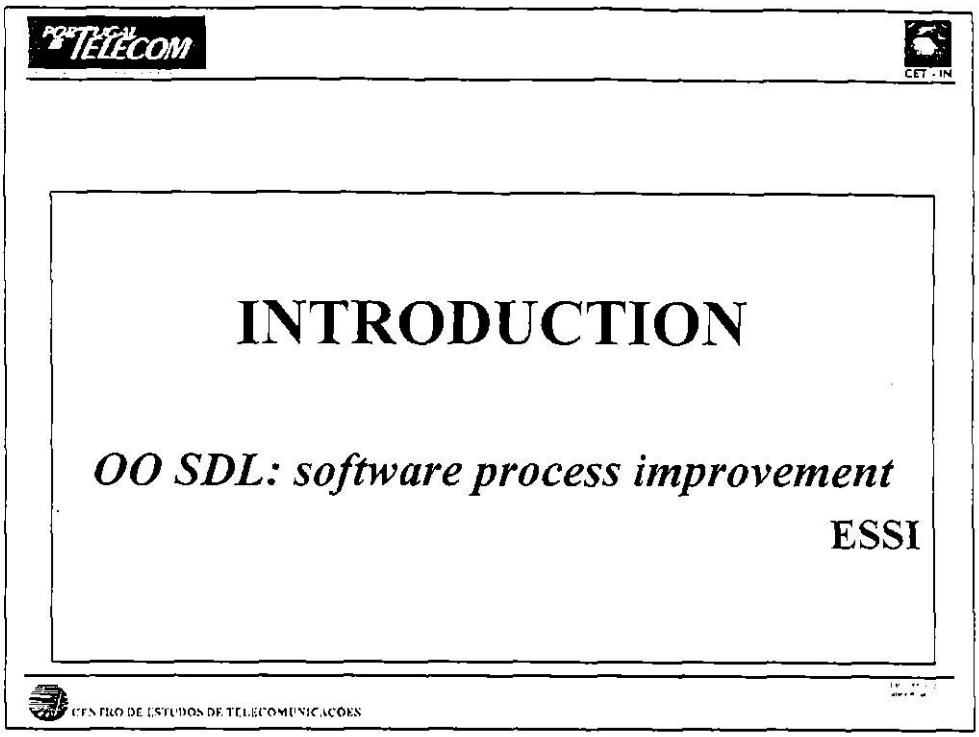
Esta apresentação é baseada no projecto ESSI 10702 (CET-IN). Aborda a temática da melhoria do processo no campo da engenharia de software de tempo real. O projecto baseia-se na aplicação da linguagem SDL-92 suportada pela ferramenta SDT para produzir um produto de telecomunicações para a rede pública. O trabalho realizado focou os aspectos de engenharia desde a especificação até à implementação.

As áreas de interesse potencial na experiência são: a aplicação do SDL-92 e a sua ferramenta de suporte; o uso de métodos para desenvolvimento com SDL-92. Embora a aplicação seja específica para a indústria de telecomunicações, as conclusões gerais sobre o uso do SDL-92 deverão ser aplicáveis a qualquer sistema que seja essencialmente de tempo real e de interfaces de mensagens discretas.

A experiência decorreu no CET (Centro de Estudos de Telecomunicações), o sector de investigação e desenvolvimento da Portugal Telecom em Aveiro que é o departamento responsável pela investigação e desenvolvimento de produtos e aplicações para a rede PT. O CET é composto por cerca de 200 engenheiros, dos quais 60 estão envolvidos em desenvolvimento de software. Aproximadamente 10 desses engenheiros têm estado envolvidos directa ou indirectamente na experiência.

O produto destina-se à aplicação na rede da PT e a primeira versão encontra-se já instalada em experiência piloto desde Julho/1995. A experiência decorre num ambiente de pressão real para cumprimento de prazos de entrega. Os resultados são portanto realistas e não idealistas. O trabalho que necessitava de ser realizado não abrangia a análise de requisitos ou uma quantidade apreciável de especificação de alto nível, pois o produto obedece a normas internacionais e a requisitos nacionais bem definidos.

A melhoria mais significativa é o grau de eficácia da técnica usada: desenho e implementação do sistema na linguagem SDL suportada pela ferramenta SDT (SDL Design Tool) da Telelogic, uma companhia sueca do grupo Saab Combitech. Embora o SDL tenha sido seleccionado neste caso devido à sua utilização em telecomunicações, é uma linguagem que pode ser aplicada a qualquer domínio de tempo real baseado em estímulo-resposta. O SDL tem sido utilizado numa ampla variedade de aplicações desde bancos a controle ambiental de gastos e aeronáutica.



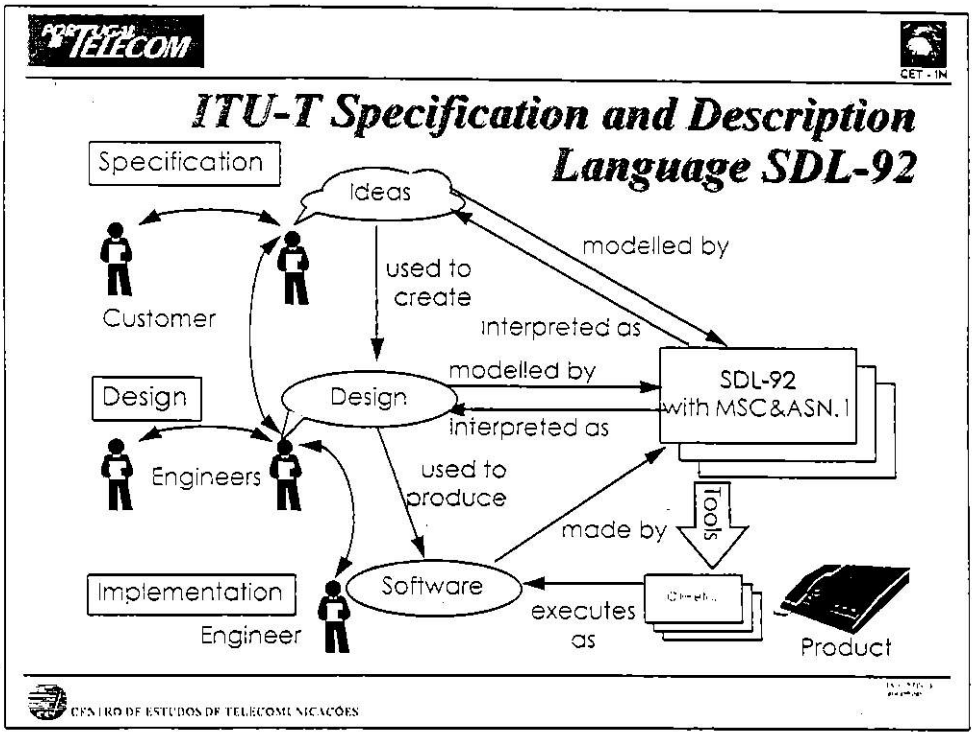
O desenvolvimento de software é uma vertente cada vez com maior relevo em diversas indústrias e serviços, e nomeadamente no CET. De facto, os desenvolvimentos de equipamentos, serviços e aplicações de gestão e operação são exemplos de actividades que o CET desenvolve para a Rede de Telecomunicações da Portugal Telecom (PT) que têm uma parcela de software muito grande e em expansão. Assim, a sistematização e o melhoramento dos processos de desenvolvimento de software no CET assume particular relevância. Neste contexto uma das iniciativas nesta área consistiu em apresentar uma candidatura ao programa comunitário ESSI (European Software System Initiative / Software Best Practice - European Commission DG III), a qual foi aprovada e deu início ao projecto ESSI 10702 (a decorrer desde 1/6/94 até 30/11/95). O projecto consiste na aplicação de uma metodologia suportada por uma ferramenta CASE (SDT), que permite a análise, documentação, especificação, geração automática de código, validação e teste. A ferramenta SDT3.0 implementa a linguagem gráfica formal SDL92, que é uma linguagem normalizada pelo ITU-T, para uso nas Telecomunicações, mas também com uso crescente noutras actividades, designadamente em qualquer aplicação com características de tempo real e de interface por mensagens discretas. O código gerado é "C", o que generaliza o uso em diversas plataformas. A aplicação prática seleccionada foi uma vertente do projecto CET-IN. O CET-IN é uma plataforma de serviços de telecomunicações, permitindo a criação e execução de serviços baseados no conceito das Redes Inteligentes, assumindo a normalização internacional relevante nesta área. Exemplos desses serviços, que em breve se encontrarão disponíveis na PT, são o Telecom Class Automático e o Número Pessoal. O CET-IN é um projecto de software com uma dimensão e complexidade consideráveis: os executáveis desenvolvidos para os diversos componentes do sistema têm cerca de 4 MBytes no seu conjunto. Várias técnicas são empregues usando tecnologias de ponta no desenvolvimento de software: Unix, metodologias de orientação a objectos -- ferramentas CASE: SDT, Objectory (análise/design) --, C, C++ Softbench, VisualBasic, base de dados relacional Oracle e ferramentas associadas, etc. O componente de software que serviu de base à experiência de aplicação é o VSSP (Virtual Service Switching Point), e o enquadramento encontra-se descrito na figura (no texto final da comunicação serão detalhados os aspectos relevantes). A sua primeira versão encontra-se já implementada com sucesso e aplicada na Rede PT desde Maio/95.

A versão Object-Oriented da ferramenta, disponível desde Março/95 é a base da versão seguinte, em preparação, prevista para Setembro/95. A comunicação apresenta resultados comparados de diversas práticas permitindo assim uma avaliação das melhorias alcançadas:

- Desenvolvimento sem recurso à ferramenta SDT
- Desenvolvimento com a ferramenta SDT2.3 (implementação actual)
- Desenvolvimento com a ferramenta SDT3 (versão Object-Oriented)

Os resultados são divulgados em diversos seminários e conferências permitindo assim a disseminação da experência, para eventual utilização por outros interessados.

Na comunicação será também focado um dos resultados mais relevantes do projecto: a descrição do processo de desenvolvimento de software desde os requisitos, análise, especificação, implementação e teste com base na ferramenta SDT.



Considerando na engenharia de sistemas três fase principais - especificação, desenho e implementação - o papel do engenheiro e da linguagem utilizada pelo engenheiro é diferente em cada fase. Durante a especificação, a preocupação primordial é a captura de ideias a partir dos requisitos e a sua modelização como uma especificação, para que o diálogo sobre as ideias possa ter lugar. Na fase de desenho, são utilizadas linguagens para a formalização, partição e estruturação dos conceitos, para que se possa dividir o problema em partes mais facilmente implementáveis. As linguagens formais são muitas vezes utilizadas para a comunicação entre engenheiros de desenho, e a linguagem utilizada para a especificação é um factor importante.

A chave para o sucesso de um sistema é a completa especificação e desenho. Isto exige uma liguagem de especificação adequada, obdecendo a:

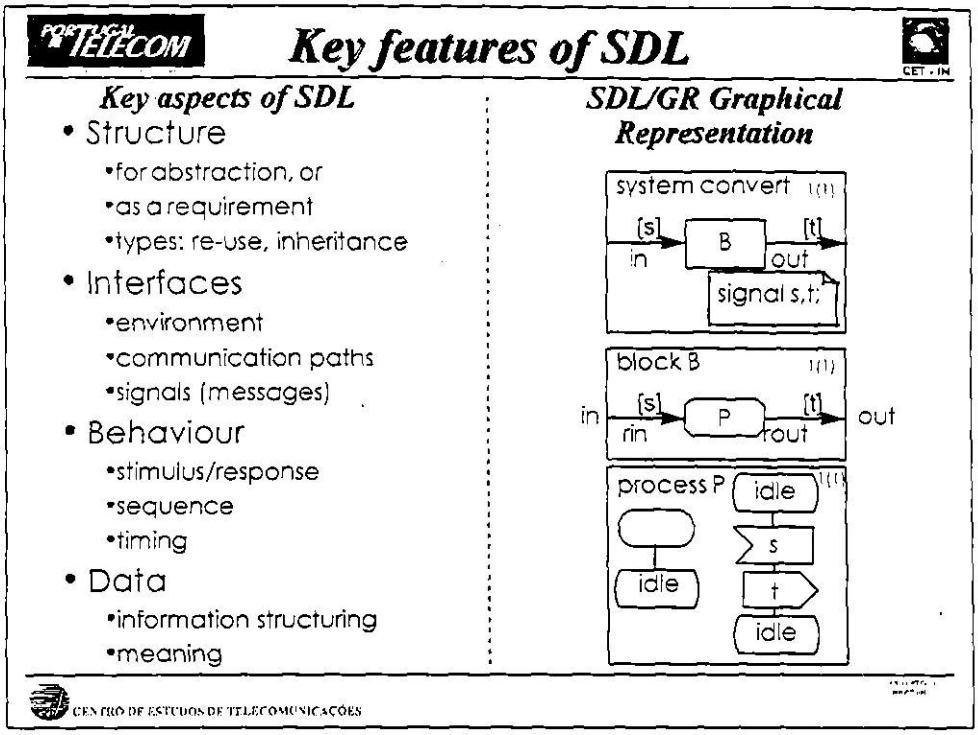
- *conjunto de conceitos* bem definidos;
- *especificações sem ambiguidade, claras, exactas e concisas;*
- uma base para *análise das especificações* no que diz respeito à determinação se estão *completas* e à sua *exactidão;*
- uma base para determinar a *conformidade* das implementações face às especificações;
- uma base para determinar a *consistência* entre si do conjunto das especificações;
- utilização de *ferramentas* baseadas em computador para criação, manutenção, análise e simulação das especificações.

Para um sistema, podem haver especificações a diferentes níveis de abstracção. Uma especificação é a base para as implementações, mas deve abstrair-se numa primeira fase dos detalhes de implementação para

- dar uma panorâmica geral de um sistema complexo
- adiar decisões de implementação, e
- não excluir implementações válidas

Durante a implementação de software, os desenhos são considerados como um ponto de partida e o produto é descrito em termos de uma linguagem de programação. Felizmente, o SDL pode ser utilizado para todas as vertentes evitando alguns custos e fontes de erro quando se passa de linguagem para linguagem.

A abordagem de desenvolvimento baseada em SDL abrange a totalidade do ciclo de vida da engenharia de software. É apropriada para tempo real e aplicações distribuídas em que a preocupação primordial seja a divisão em unidades que comuniquem por passagem de mensagens. Não será tão apropriada, pelo menos directamente, a áreas em que predominem interfaces de utilizador para comunicação Homem-máquina bem como para o desenho de objectos em que haja uma grande envergadura de dados de informação (Bases de Dados). Os requisitos dos sistema de telecomunicações são particularmente exigentes pelo que se estes podem ser satisfeitos de um modo económico, então certamente que outros tipos de sistemas apresentarão poucos problemas à utilização de SDL.



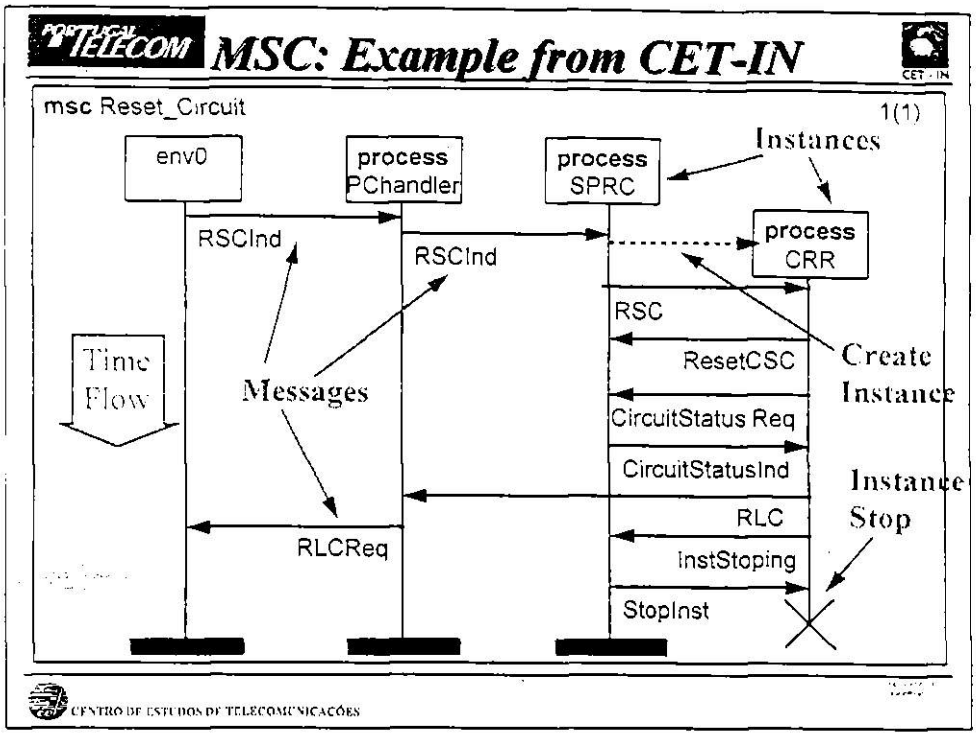
Estrutura - abarca a composição de blocos (**block**) e processos (**process**). O SDL é estruturado por forma a permitir a fácil compreensão de um sistema ou para reflectir a estrutura (pretendida ou realizada) de um sistema. Há uma relação forte entre estrutura e interfaces. Os tipos (**Type**) (p.e.. **blocos** e **processos**) podem ser utilizados para estruturar a descrição do **sistema** e para descrever as partes que podem ser re-utilizadas no sistema ou em outros sistemas. É possível a definição de **tipos** que "herdem" propriedades de outros **tipos**.

Interfaces - destina-se à descrição de sinais (**signal**) e vias de comunicação para sinais. A comunicação é assíncrona pelo que quando um sinal é enviado de um processo poderá haver um atraso antes de ele atingir o seu destino e o sinal pode ficar em fila de espera mesmo depois de chegar ao destino. As vias de sinais estão relacionadas com a estrutura do sistema. O comportamento do sistema é caracterisado pela comunicação apresentada nos interfaces exteriores. Os sinais podem conter dados e portanto a sua definição é dependente dos tipos dos dados.

Comportamento - tem a ver com o envio e recepção de sinais bem como com a interpretação das transições nos processos. A interpretação da comunicação de sinais entre processos é a base da semântica de uma definição SDL.. A intrepretação depende dos interfaces, da estrutura e particularmente dos dados.

Dados - são usados para armazenar informação. Os dados armazenados em sinais e processos são usados para decisões no interior dos processos. Excepto para os sistemas extremamente triviais a utilização de tipos de dados é fundamental para eliminar texto informal das definições SDL.

O SDL tem duas formas de representação: SDL/GR - Representação Gráfica e SDL/PR - Representação Textual. A maior parte dos utilizadores prefere a representação gráfica baseada em ferramentas, pois é mais fácil de ler e compreender. O CET usa SDL/GR. Nem tudo no SDL/GR é gráfico: alguns items, tal como nomes, não podem ser expressos graficamente pelo que essa parte do SDL/GR é textual. Esta parte comum textual cobre: definições de dados, expressões e atribuições. A linguagem na sua forma completa inclui de facto ambos SDL/PR e SDL/GR.



Os Mapas de Sequências de Mensagens - Message Sequence Charts (MSC) - são utilizados para especificar sequências de mensagens exemplificativas da utilização do sistema, para ocorrências normais ou excepcionais. Os MSCs são tão fáceis de compreender que podem ser considerados intuitivamente óbvios. Os MSCs podem ser utilizados como base para discussão com os utilizadores do sistema.

Frequentemente os MSCs são utilizados como esboços intermediários do comportamento do sistema bem como das suas partes internas. Isto ajuda o engenheiro a compreender o que o sistema deve de facto fazer.

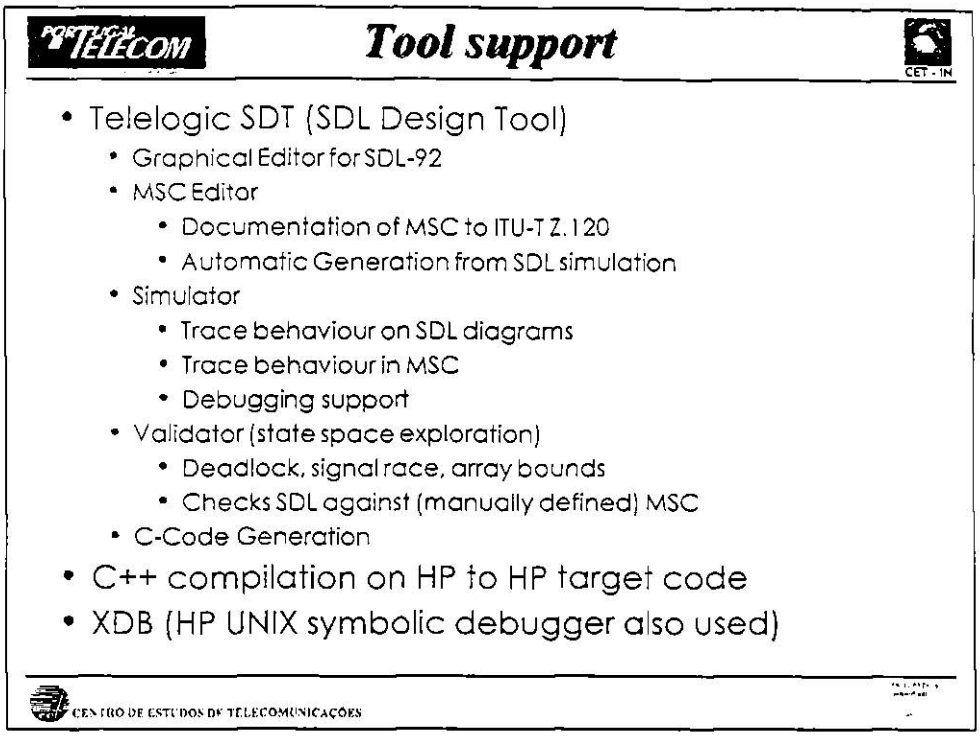
Os MSCs que tratam o sistema como uma instância (isto é como uma "caixa preta") podem ser parte dos requisitos de um sistema podendo até ser utilizados em documentos contratuais. Quando os MSCs são deste tipo, devem ser mantidos durante o ciclo de vida do projecto. Durante a elaboração de uma aplicação e da divisão do sistema em blocos e processos, os MSCs são utilizados para determinar a comunicação entre as diversas partes do sistema. O exemplo dado consiste na libertação de uma chamada telefónica. Estes diagramas fazem parte da descrição do sistema. Devem ser mantidos ao longo do ciclo de vida do projecto e aplicados na fase de teste da aplicação.

Adicionalmente aos MSCs criados durante o desenho de uma aplicação, haverá necessidade da criação de MSCs complementares por foma a cobrir situações não usuais bem como sequências que nunca deveriam ocorrer. Estas últimas são bem importantes para testar a robustez do sistema.

Os MSCs podem expressar apenas alguns dos traços de comportamento do sistema. Não podem ser utilizados para especificar completamente o comportamento do sistema. Contudo, viabilizam uma visão alternativa útil e permitem comprovar o comportamento do sistema.

No projecto do VSSP, detalhado mais à frente , alguns problemas potenciais no desenho foram descobertos através da utilização sistemática de MSCs.

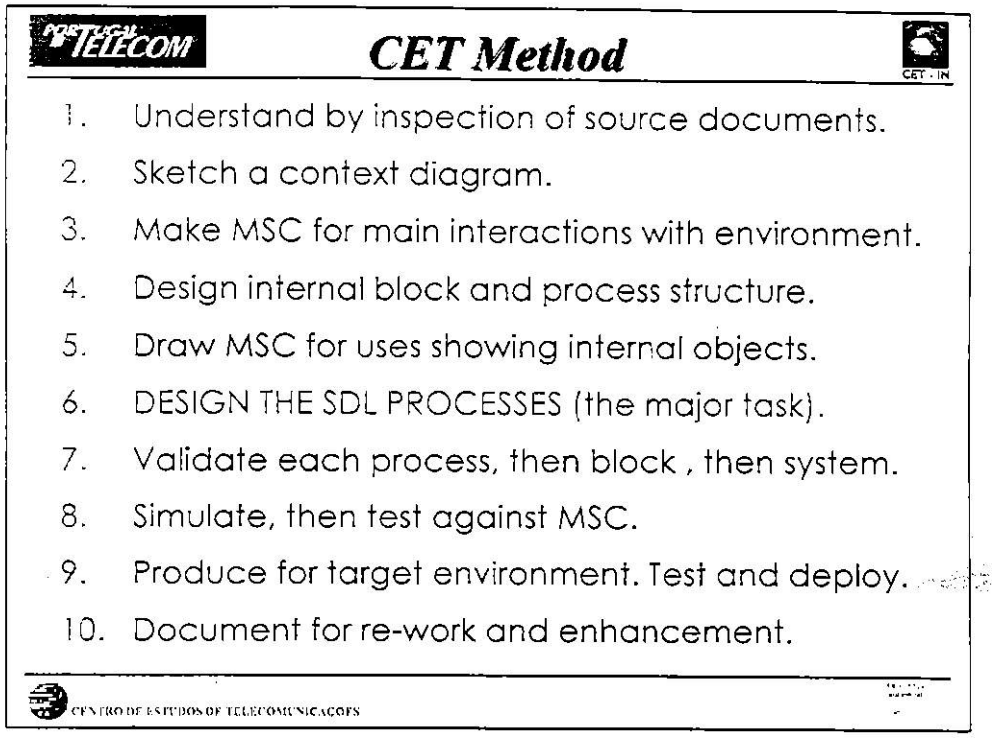
O MSC do exemplo omite algum detalhe nas mensagens. Muitas das mensagens têm parâmetros que são usados pelos processos, como por exemplo a identificação da origem e destino da chamada telefónica. Quando os processo são definidos o conteúdo das mensagens pode ser identificado e definido também. Para a realização de testes o conteúdo tem mesmo de ser definido efectivamente.



A utilização de uma linguagem como o SDL não pode ser separada da disponibilidade de suporte em ferramenta. A ferramenta de suporte SDT da Telelogic, permite a análise, especificação, documentação, geração automática de código e ainda o teste do software. A versão SDT3.0 suportando a versão SDL-92 da linguagem encontra-se disponível desde o primeiro trimestre de 1995. A anterior versão suportava o SDL-88. A versão actual contém:

- um editor gráfico para SDL-92 dotado de um analisador para a sintaxe e semântica. Mecanismos de importação e exportação de SDL textual (SDL/PR) para que possa haver comunicação com outras ferramentas que suportem SDL/PR.
- um editor gráfico de MSCs com a possibilidade adicional de geração automática de MSCs durante a simulação do SDL.
- Geração de código C com bibliotecas de simulação. Durante a simulação o comportamento do sistema pode ser rastreado directamente nos diagrams SDL e em MSCs gerados automaticamente durante a simulação.
- um Validador para detecção automática de falhas nas especificações SDL e validação automática de MSCs. O Validador verifica automaticamente o comportamento do sistema SDL no seu ambiente definido e detecta erros dinâmicos de tempo real tal como "dead-locks", "signal race", acesso deficiente a variáveis, etc. Para além disso, verifica código C que tenha sido incluído nas especificações SDL. Valida automaticamente que as operações do sistema (manualmente definidas através de MSCs) estão de facto implementadas no sistema em teste.
- Geração de código fonte C, para interfaces I/O, interfaces de hardware e sistema operativo, "kernel" final e o monitor de simulação interactiva, pelo que é assim possível gerar automaticamente aplicações completas e executáveis.

A versão anterior, SDT 2.3, apresentava um conjunto de facilidades excepto que não suportava SDL-92. Devido a este motivo a experiência foi repartida em 2 fases: utilização baseada em SDT 2.3, seguida de exploração das novas funcionalidades do SDL-92 e SDT 3.0.



Em muitos casos a quantidade de trabalho empregue na análise formal poderá ser trivial, pois existem no sector das telecomunicações normas bem documentadas, pelo que a maior parte do trabalho concentra-se na formalização. Os seguintes pressupostos são tidos em consideração na definição da metodologia:

1. A maior parte das aplicações do CET terão:
a) arquitectura bem definida, usualmente através de diagramas de blocos não SDL;
b) interfaces bem definidos, usualmente nas normas de telecomunicações, por vezes via linguagem formal ASN.1;
c) funcionalidades bem definidas (linguagem natural, nalguns casos complementada por pseudo SDL informal);

2. A análise da aplicação não é necessária, na maior parte dos casos.

A metodologia consiste portanto nas seguintes actividades:

1. Compreender a aplicação pela "inspecção" do material base (ver Olsen Chapter 6) .

2. Esboçar um diagrama de blocos (SDT, ou papel e lápis e qualquer ferramenta gráfica, ou através da utilização de um diagrama dos requisitos) descrevendo as entidades do sistema e ambiente envolvente da aplicação. Esta acção permite a identificação das fronteiras da aplicação bem como a fixação de nomes e entidades do sistema.

3. Fazer, ou identificar MSCs existentes para as principais interacções entre os objectos do meio envolvente e a aplicação. Identificar conjuntos de sequências e o texto correspondente.

4. Desenhar a estrutura de blocos dentro da aplicação usando a ferramenta SDT com canais e rotas de sinais.

5. Desenhar MSCs para as principais interacções entre as partes internas (blocos e processos) nos diagramas SDL. Juntar casos excepcionais e casos de erro em paralelo ao desenho dos processos.

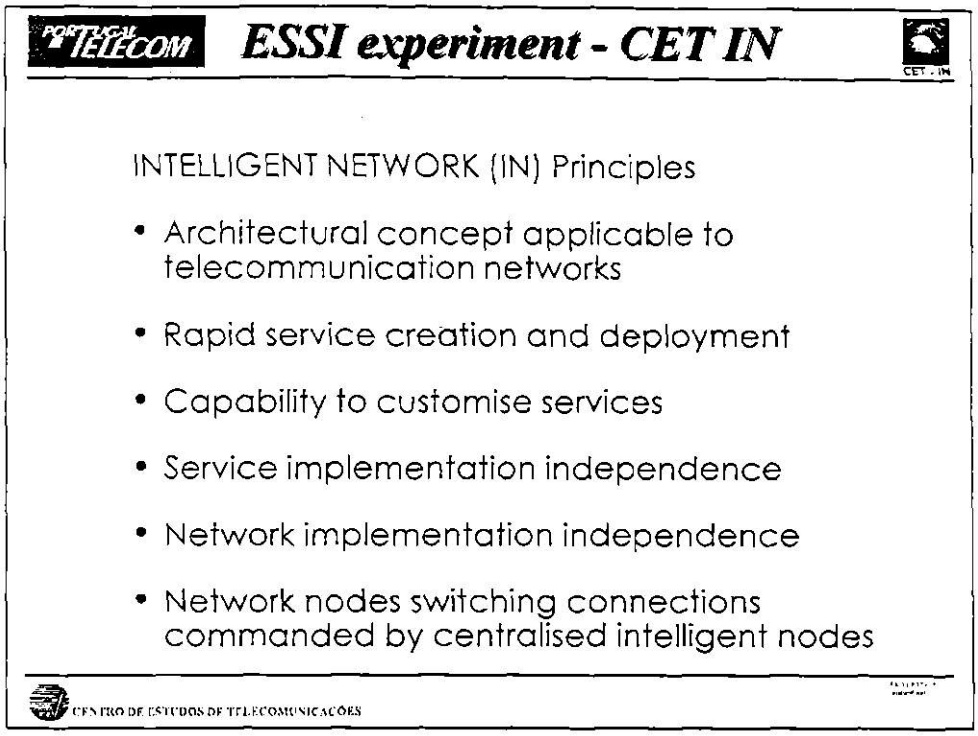
6. Desenhar os processos SDL. As etapas de formalização descritas em Olsen Capitulo 6 poderão ser utilizadas como guia. Na execução de um projecto é usual um processo ser atribuído para efeitos de implementação a uma pessoa.

7. Validar cada processo, depois os blocos e finalmente o sistema. Usar a técnica de "walkthroughs" e inspecção por outro elemento da equipa do projecto não envolvido na implementação de cada parte a inspeccionar. Cada processo completado é validado com base na utilização das facilidades oferecidas pela ferramenta.

8. O sistema é simulado e depois testado face aos MSCs.

9. O sistema é produzido para o ambiente alvo e testado, antes de entregue.

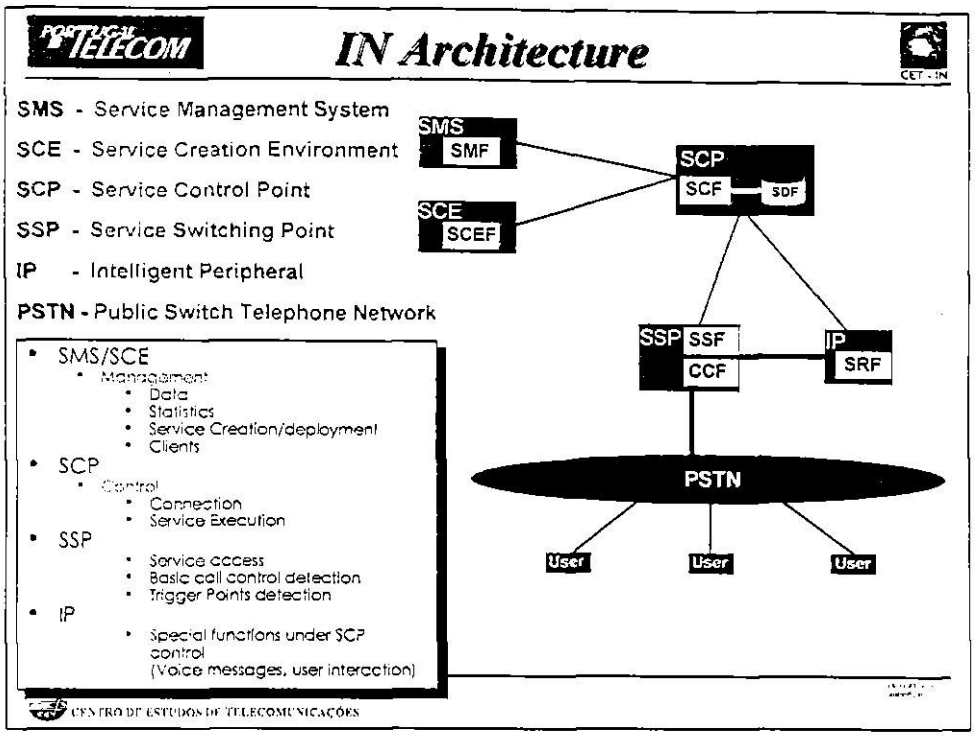
10. A documentação é finalizada para que outro engenheiro possa mais tarde re-abordar o sistema (ou corrigir um eventual erro de desenho) tendo conhecimento de como é que a aplicação funciona.



O termo Rede Inteligente - Intelligent Network (IN) - é utilizado para descrever um conceito arquitectural destinado a ser aplicável em todas as redes de telecomunicações. O objectivo é facilitar a introdução de novos serviços de telecomunicações de forma rápida e flexível, e poder modificar serviços já existentes dotando-os facilmente de novas funcionalidades.

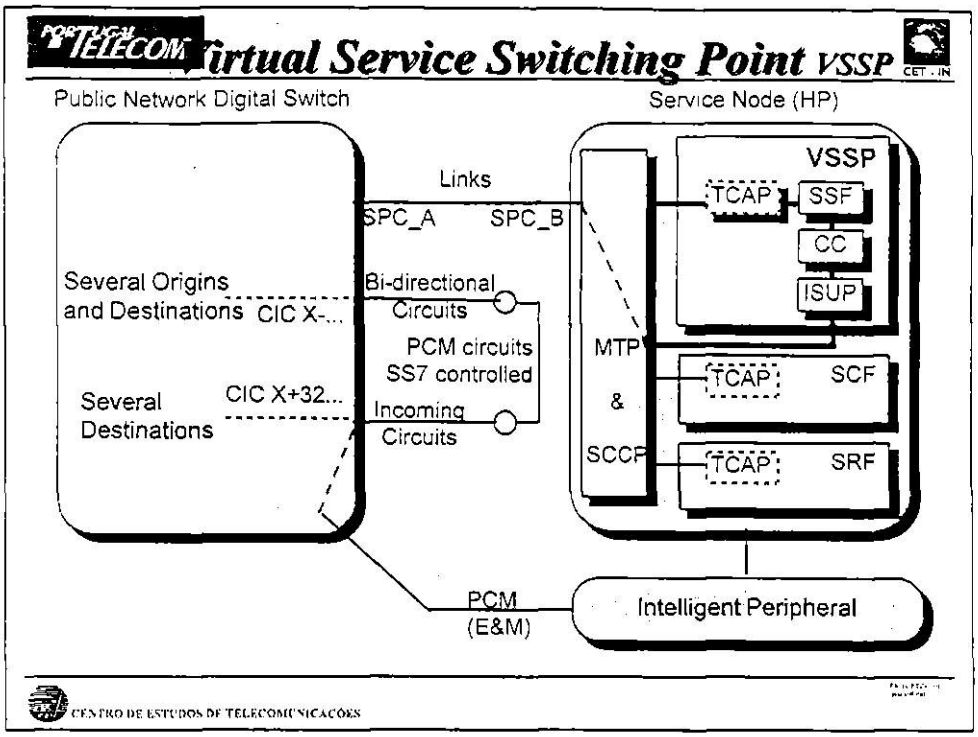
Os objectivos da IN são:

- permitir criação de serviços rápida e em moldes económicos
- independência da implementação dos serviços/rede num ambiente multivendedor. A independência da implementação garante ao fornecedor de serviço ser autónomo na disponibilização de novos serviços face aos vendedores tradicionais de equipamento de telecomunicações. Estes objectivos são conseguidos através da separação entre o controlo básico da chamada e do controle dos serviços. Os serviços são construídos com base em blocos básicos elementares que podem ser utilizados numa diversidade enorme de serviços. Tipicamente ambas as funções de controle de chamadas e controle de serviços são implementadas em entidades físicas diferentes, sendo o controle de serviços centralizado e o controle de chamadas distribuído pela rede.



A figura mostra simultâneamente as entidades funcionais e as entidades físicas da arquitectura IN, a qual é caracterizada pelo seguinte:

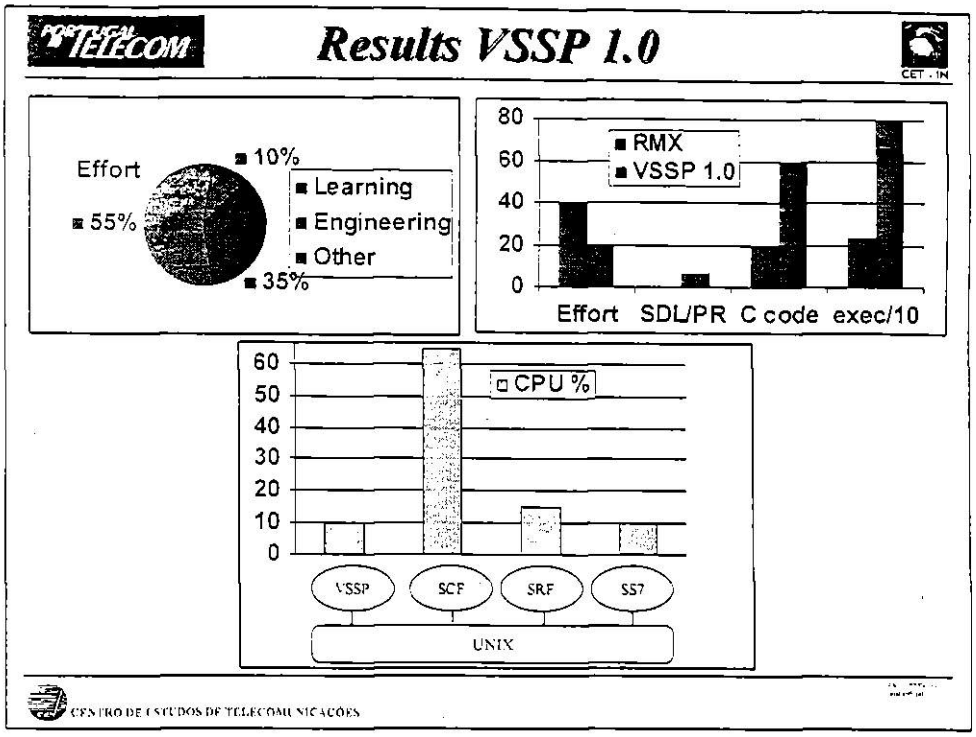
- Pontos de Controlo do Serviço (Service Control Point): nós centralizados da rede concentrando a inteligência. O SCP contem a função Service Control Function (SCF) com a lógica de serviços para tratamento das chamadas do tipo IN. Nalguns casos o SCP contem também a função de acesso a dados (Service Data Function) com uma base de dados integrada. O SCF possui interfaces e interactua com outras entidades do sistema IN: SSF, SDF and SRF. O SDF proporciona uma visão lógica para o SCF sobre os dados de serviço e de rede.
- Pontos de controle de comutação (Service Switching Points): nós da rede que são responsáveis pelo estabelecimento de ligações físicas através da rede de transporte por comando do SCP. Descrimina as diversas chamadas em curso reconhecendo as que deverão ter um tratamento IN, interactuando por um lado com o controlo básico de chamada (CCF) e com o SCF por outro lado.
- Periférico Inteligente (Intelligent Peripheral): entidade que implementa a função de recursos especializados - Specialised Resource Function (SRF) - proporcionando as interacções entre a rede e os utilizadores como, por exemplo, anúncios e recolha de informação do utilizador via marcação telefónica.
- Ambiente de Criação de Serviços (Service Creation Environment): o SCE suporta a criação de serviços. Disponibiliza os meios necessários para a definição de serviços, bem como para a sua verificação e teste. Gera informação com a descrição da lógica de serviços, que servirá para guiar o SCF na execução de cada serviço.
- Sistema de Gestão de Serviços (Service Management System): o SMS inclui a Função de Gestão de Serviços (SMF) e apresenta interfaces com todos os outros elementos da arquitectura IN. A função SMF é responsável por gerir o fornecimento de serviços aos clientes e utilizadores, instalação de novos serviços e controle geral dos componentes da rede. Possibilita o acesso a todas as entidades funcionais IN para a transferência de informação relacionada com a lógica de serviço e com os dados de serviço.
- Interfaces normalizados designadamente entre o SCP e o SSP para permitir a troca de informação entre a lógica de serviço e a rede de telecomunicações e ainda entre o SCP e o IP. Estes interfaces normalizados facilitarão a competição entre fornecedores de equipamento permitindo uma maior independência dos operadores de rede face a soluções proprietárias de alguns fornecedores.



Tradicionalmente, o SSP como entidade de comutação, inclui a função CCF pois esta função necessita de aceder aos recursos de um comutador para a sua missão de providenciar comutação entre circuitos de entrada e saída. No entanto, é possível pensar numa arquitectura derivada em que o CCF se localize numa entidade física diferente da que realmente executa a comutação de ligações. A única condição é que o CCF tenha conhecimento das ligações entre circuitos de entrada e saída, utilizando para isso o esquema base da figura.

A aplicação seleccionada para a experiência é o desenvolvimento de um interface de suporte a um pacote de serviços de telecomunicações (Cartão Virtual de Chamadas, Número Pessoal, Linha Pessoal). Estes serviços são do tipo dos definidos no conjunto de normas IN designado por ETSI CS-1 (Capability Set 1). Uma plataforma comercial de hardware/software (HP 9000/827) é utilizada como infraestrutura de suporte à implementação de um "Virtual Service Switching Point" (VSSP). O software na experiência implementa um interface à sinalização de rede entre comutadores (Message Transfer Part - MTP of ITU-T Signaling System No 7). Este interface complementado com o tratamento do controle de chamada e SSF, possui então um interface ao SCF e ao SRF controlando um Periférico Inteligente (IP) ligado ao comutador.

Na figura pode observar-se uma panorâmica da arquitectura do VSSP. O VSSP é uma das partes constituintes do nó de serviços implementado na plataforma HP. As chamadas de entrada provenientes da rede pública invocando serviços prestados pelo nó de serviços são encaminhadas via circuitos bidireccionais associados ao Ponto de Sinalização atribuído ao nó. A sinalização utilizada para controlar as chamadas é o ISUP do SS#7 sobre MTP. Os circuitos são ligados em "loop" ao comutador evitando-se a função de comutação no nó de serviços, que se serve das funcionalides inerentes ao comutador da rede pública. Cada circuito com CIC (Circuit Identification Code) número X é sempre ligado ao circuito X+32. Esta relação fixa entre os 2 circuitos é utilizada pelo ISUP (ISDN User Part, sinalização entre comutadores digitais) no VSSP para "comutar" a chamada. Quando uma chamada necessita de ser ligada ao destinatário ou ao IP, a ligação é controlada pelo VSSP através do envio de mensagens ISUP para a rede pública para o circuito X+32. O módulo VSSP do nó de serviços, comunica com o SCF e com o SRF, e este controla o IP que também está ligado à rede pública. O VSSP trata o reconhecimento de "Trigger Points" no processamento da sinalização (ver recomendações ITU-T da série Q.1200), comunica estes eventos ao SCF e aguarda instruções provenientes do SCF. A comunicação com o SCF baseia-se nas operações normalizadas INAP (Q.1218) sobre TCAP.



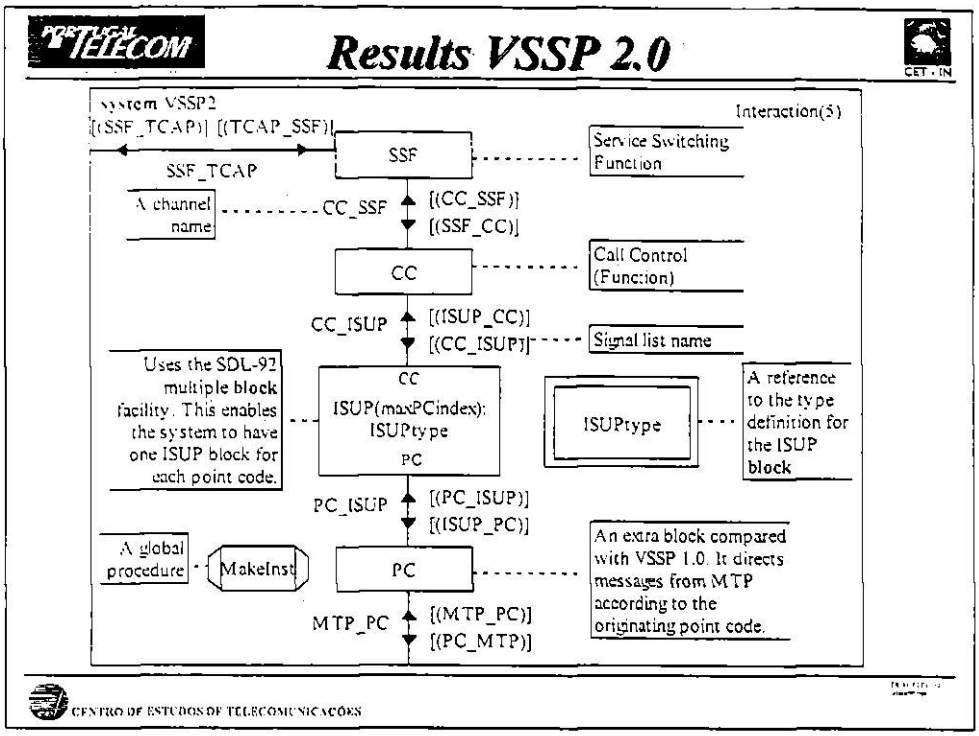
O esforço para a produção com base em SDT do VSSP 1.0 foi de cerca de 20 Homens-Mês. Esta versão é funcionalmente equivalente a uma versão produzida anteriormente (baseada em C e RMX) cujo esforço foi de ceca de 80 HM. Os dois números não podem ser comparados simplisticamente pois:

- os engenheiros envolvidos foram os mesmos e conheciam bem a aplicação anterior bem como toda a problemática envolvente;
- o sistema anterior foi concebido para outra plataforma (RMX);
- era a primeira vez que o SDL estava a ser utilizado;
- provavelmente 40 HM seriam suficientes para desenvolver de novo o software da primeira versão, utilizando o método anterior.

Embora utilizando o SDL/GR, utilizou-se como termo de comparação o número de linhas de SDL/PR textual gerados a partir do SDL/GR gráfico. O SDL/PR tem exactamente o mesmo significado do SDL/GR mas sendo o texto substituído por símbolos gráficos, viabilizando uma leitura mais facilitada aos utilizadores. O SDL/PR é gerado automaticamente como base para a análise e geração de código. Complementarmente ao código gerado pela ferramenta há mais cerca de 1K linhas de código C para o ambiente envolvente. A conclusão é que para a mesma funcionalidade o código objecto gerado é cerca de 3,4 vezes maior, mas foi escrito 2 vezes mais rapidamente. Os números acima indicados correspondem ao SDT 2.3 e ao VSSP 1.0.

Um erro de menor importância foi descoberto durante a operação, e o número de erros descobertos no processo de desenho propriamente dito foram significativamente em menor quantidade do que na experiência prévia. O erro descoberto na operação era um valor incorrecto na descodificação de um parâmetro de uma determinada mensagem. O parâmetro deveria ter sido ignorado, mas um valor incorrecto causava a geração de uma mensagem inesperada em determinada fase da chamada o que era de imediato tratado abortando a chamada em curso. Algumas chamdas falhavam então devido a isso. A ocorrência desse valor não tinha sido testada previamente.

Na abordagem prévia (C e RMX) o sistema tinha sofrido de uma qquantidade apreciável de erros de desenho associados ainda ao envio de mensagens para o destino incorrecto. O uso de SDL e SDT ajudaram a eliminar este tipo de erros, embora ainda houvesse alguns erros lógicos e de codificação no VSSP 1.0 antes dos testes e correcções, que foram rapidamente ultrapassados.



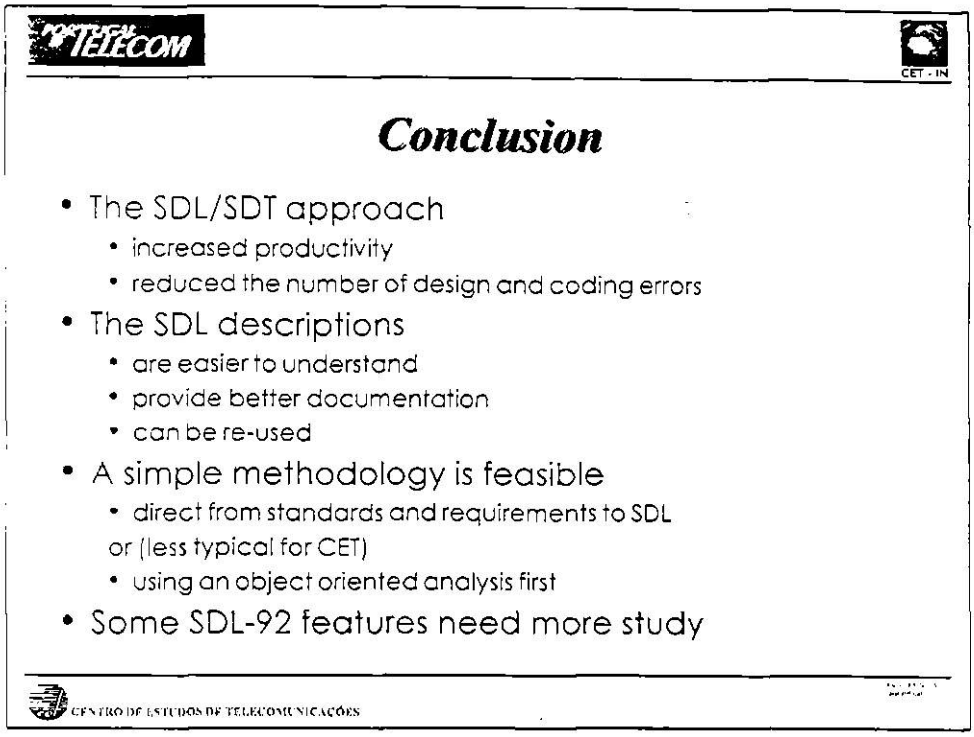
Como pode ser observado na figura as vias de comunicação para sinais são claramente indicadas em SDL. Os diagramas SDL têm a vantagem de apresentarem a estrutura e comunicação de uma forma natural para que à primeira vista pareçam desenhos informais frequentemente utilizados na descrição de sistemas. Na realidade expressam a definição e uso de nomes SDL e devem ser cuidadosamente definidos e ligados para que a definição SDL seja válida. Por exemplo os parêntesis rectos [...] situados perto de uma seta no diagrama definem os sinais transportados na direcção da seta na via correspondente, e um nome entre parêntesis curvos dentro dos [...] é a notação para uma lista de sinais definida algures nos diagramas. O SDT verifica se todos os sinais definidos na lista CPCIsignals são sinais provenientes de CPCI (processo em ISUPType) estão incluídos na lista ISUP_PC, e podem ser recebidos por um processo dentro do bloco PC. Estes casos exemplificam verificações (e as correspondentes melhorias de engenharia) que existem com o SDL-88 e com o SDT2.3.

O VSSP 2.0. cujo desenho é orientado a objectos, apresenta algumas vantagens face ao VSSP 1.0. O encaminhamento de sinais é simplificado, pois o conjunto de processos para um circuito é encapsulado dentro de um bloco, declarado como um tipo e instanciado. Para a comunicação intra-bloco os sinais não têm de ser enviados para um processo de entre vários: há uma instância de cada tipo por cada circuito. O SDL-88 não suportava a tipificação de blocos.

Os procedimentos remotos no SDL-92 proporcionam a realização de uma função remota de acesso seguro a dados, viabilizando uma especificação simplificada. Uma desvantagem é que a comunicação entre processos subjacente a essa funcionalidade não é por vezes óbvia pela simples leitura da estrutura dos diagramas.

O VSSP 2.0 também utiliza o conceito de procedimento global do SDL-92 para o caso do procedimento MakeInst. Este procedimento muito simples é o mesmo em diversos processos, mas em SDL-88 tinha de ser declarado em cada processo.

O uso de outras abordagens para os objectos SDL (processos e procedimentos) é a noção de armazém ("package") que está em estudo actualmente. O uso de tipos foi considerado, mas não foi considerado prático com SDT 3.0 devido ao facto de que a ferramenta não suporta parâmetros de contexto.

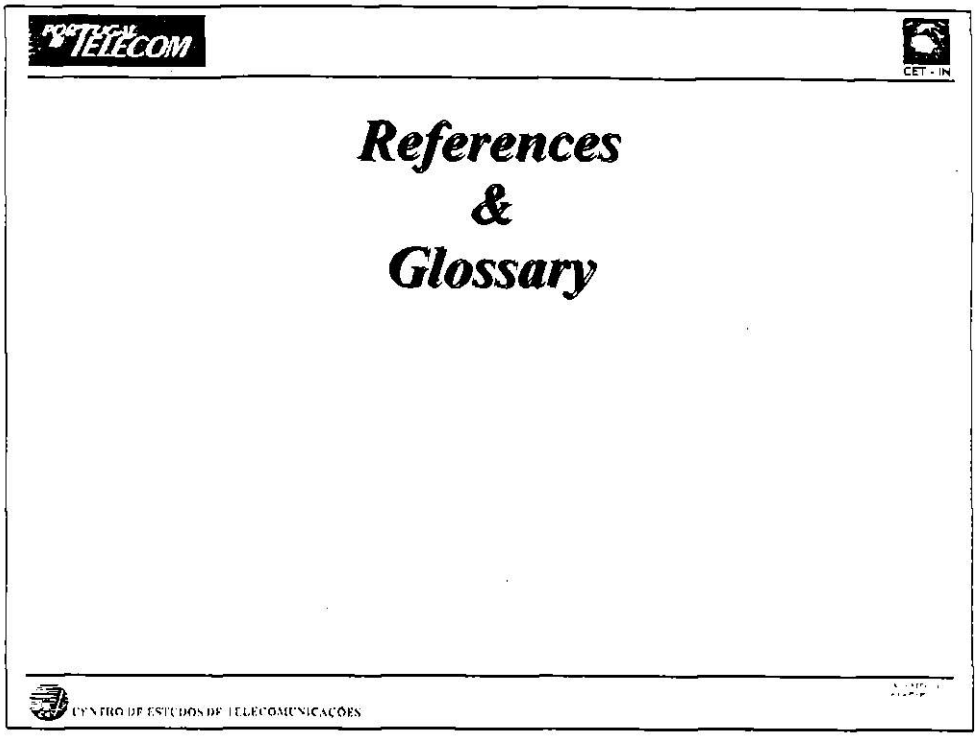


A experiência de facto teve como principal preocupação a re-engenharia do desenho e da geração de código, pelo que a maior parte da informação transmitida pela experiência é relevante para estas actividades. Os resultados mostraram claramente que esta abordagem proporciona uma melhoria de produtividade real e melhoria de qualidade. O rigor na determinação de objectivos em cada projecto é também uma outra vertente que certamente é beneficiada.

Algumas novidades (novos conceitos) do SDL-92 face ao SDL-88 foram já úteis e permitiram simplificações no VSSP 2.0. A ferramenta SDT3.0 apresentou uma migração suave da versão VSSP 1.0 realizada com base no SDT 2.3, acrescentando mais algumas novidades ao nível da ferramenta bem como obviamente ao nível da linguagem base.

Os melhoramentos conseguidos pela utilização do SDL-92, não foram tão significativos como os obtidos no salto inicial da implementação em C para a implementação em SDL-88. A razão para isso fica a dever-se a: 1- os desenhos do VSSP 1.0 foram a base de mais de 90% do VSSP 2.0 medida em que se realizou uma migração; 2- Um projecto de raiz poderia adequar-se mais à exploração das novas funcionalidades da linguagem; 3- Por outro lado, para fazer um uso completo da linguagem SDL-92 é necessário efectuar um re-arranjo substancial nos desenhos originais baseados na normalização existente nas telecomunicações. Estes são baseados em SDL-88 informal quase sempre. O benefício expectável seria a facilidade (e portanto a diminuição do custo) da manutenção, e re-utilização, mas este estudo está fora do âmbito deste trabalho.

O sucesso da experiência teve já como resultado que foi tomada uma decisão já em curso para a utilização do SDL/SDT para o re-desenho de um outro módulo do sistema CET-IN realizado previamente em C++: o SRF. Provavelmente, se também este caso for um sucesso, se poderá seguir outro módulo (SCF). Outros projectos no CET poderão também beneficiar da aplicação do SDT: o projecto LONGA, também com o sistema operativo HP-UX, e o projecto ELD, com o Sistema operativo RMX. Algumas experiências já foram realizadas neste último caso, com sucesso, para comprovar que era possível gerar código para um sistema operativo e Hardware diferentes do "host".



Glossário:

BOOST RACE project.
CASE Computer Aided Software Engineering.
CET Centro de Estudos de Telecomunicações, Portugal Telecom.
CET-IN Intelligent Network product of CET.
CS-1 IN Capability Set 1 (set of services)
ETSI European Telecommunication Standards Institute, Sophia Antipolis, France.
HP Hewlett Packard.
IN Intelligent Network (architecture for providing telecommunications services).
ITU International Telecommunication Union, Geneva, Switzerland.
MSC Message Sequence Chart.
MTP Message Transfer Part of ITU-T Signalling System No 7.
SCF Service Control Function.
SCP Service Control Point.
SDL Specification and Description Language. (OO SDL: Object-Oriented SDL)
SDT SDL tool produced by Telelogic.
SRF Specialised Resource Function.
SSP Service Switching Point.
VSSP Virtual Service Switching Point.

Referências:

Z.100 (03/93) "CCITT Specification and Description Language (SDL)" ITU-T, Geneva, 1994.
O. Færgemand and R. Reed, "SDL: the Standard for Engineering Telecommunication Software" pp 38-47 in "Software Standards Symposium", IEEE Computer Society Press, Los Alamitos, CA, 1993, ISBN 0-8186-4240-8.
Olsen et. al. "Systems Engineering Using SDL-92", North-Holland, 1994, ISBN 0-444-89872-7.
Z.120 (03/93) "Message Sequence Chart", ITU-T, Geneva, 1994.
Z.100 Appendices I and II (03/93) "SDL Methodology Guidelines, SDL Bibliography", ITU-T, Geneva, 1994.
Reed et. al. editors, "SPECS - Specification and Programming Environment for Communication Software", North-Holland, 1993, ISBN 0-444-89923-5.
R. Bræk and Ø. Haugen, "Engineering Real Time Systems", BCS Practitioner series, Prentice-Hall, Hemel Hempstead, UK, ISBN 0-13-034448-6.
MTS (94) 010 revision 3 "Methods for Specifications and Testing (MTS); Methodologies for Standards Engineering - Specification of Protocols and Services" - Draft for ETSI Work Item DE/MTS 00013, ETSI, Sophia Antipolis, France.
ETR 137, "IN Users Guide for CS-1", ETSI, Sophia Antipolis, France.
Recommendation Z.105 (10/94) "SDL Combined with ASN.1 (SDL/ASN.1)", ITU-T, Geneva, 1995.
Ivar Jacobsen, "Object Oriented Software Engineering", Prentice-Hall, 1992.
J. Rumbaugh et. Al., "Object-Oriented Modelling and design", Prentice-Hall, 1991.
X.680, "Abstract Syntax Notation 1(ASN.1)", ITU-T, Geneva, 1993.