UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Manutenibilidade da Semântica de Modelos de Dados de Produtos Compartilhados em Rede Interoperável

Vinícius Medina Kern

Tese de doutorado submetida ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Doutor em Engenharia de Produção.

Florianópolis, Dezembro de 1997.

Manutenibilidade da Semântica de Modelos de Dados de Produtos Compartilhados em Rede Interoperável

Vinícius Medina Kern

Esta tese foi julgada adequada para obtenção do título de "Doutor em Engenharia" e aprovada em sua forma final pelo Programa de Pós-graduação em Engenharia de Produção.

Banca Examinadora:

_____

Ricardo Miranda Barcia, Ph.D, orientador
Coordenador do Programa de Pós-Graduação em Engenharia de Produção - UFSC

_____

Jan Helge Bøhn, Ph.D., examinador externo

_____

Carlos Frederico Bremer, Dr., examinador externo

_____

Roberto Carlos dos Santos Pacheco, Dr. Eng.

_____

Alejandro Martins Rodriguez, Dr. Eng.

AGRADECIMENTOS

Sou grato aos contribuintes brasileiros, que pagaram por grande parte de minha formação. O amor de minha família tem sido vital. Obrigado pai, mãe, Dani e Celsinho. E Luciana, cujo carinho e apoio me ajudaram a superar dificuldades e frustrações inerentes a este trabalho.

O suporte financeiro foi essencial e veio de diferentes fontes, em épocas diferentes. Durante a maior parte do doutorado, incluindo parte do período no exterior, recebi apoio do CNPq, número de processo 200951/94-7. Tive apoio, também, da CAPES, do National Institute of Standards and Technology (NIST), e da UNIVALI, onde leciono para Ciência da Computação.

Sou grato pelos excelentes recursos colocados à minha disposição na Virginia Polytechnic Institute and State University (Virginia Tech) em Blacksburg, VA, EUA. Em especial, não teria sido possível escrever esta tese sem os recursos do Writing Center, da Newman Library, e do CADLAB. O mesmo é verdadeiro para a biblioteca, escritório, software e instalações providos pelo NIST em Gaithersburg, MD, EUA. O Centro Tecnológico para Informática (CTI) em Campinas, SP, ofereceu-me acesso a computadores e a oportunidade de debater com participantes do projeto B-STEP. Agradeço ao Ministério das Relações Exteriores (MRE) o pagamento das passagens para as visitas realizadas a instituições brasileiras de ensino e pesquisa, como parte do Programa de Estímulo ao Retorno de Talentos (PERT'95) do CNPq.

Agradeço ao prof. Ricardo Barcia, meu orientador, pela oportunidade. O prof. Jan Helge Bøhn (Virginia Tech, Department of Mechanical Engineering) aceitou ser meu orientador no exterior, o que muito me honra. De um modo amigável, entusiástico, objetivo e brilhante, Dr. Bøhn ofereceu o exemplo e me auxiliou no desenvolvimento de minhas habilidades como pesquisador, escritor, orador e orientador. A tese foi melhorada dramaticamente em relação à primeira versão graças, em grande parte, à sua revisão. Sou grato, também, ao prof. Osama K. Eyada, que me recebeu originalmente como orientando na Virginia Tech, Department of Industrial and Systems Engineering (ISE), e acolheu minha decisão de seguir a orientação de Dr. Bøhn, no Department of Mechanical Engineering, no melhor interesse de minha pesquisa. Reconheço a excelente oportunidade que me foi oferecida por Mary Mitchell, K.C. Morris e Peter Denno no NIST, onde me receberam como pesquisador convidado e supervisionaram meu trabalho. Agradeço aos amigos Drs. Roberto Pacheco e Alejandro Martins (PPGEP/UFSC), que dedicaram longas horas a discussões sobre a tese. André Luiz Tietböhl Ramos, Ph.D., "O Mighty Guruji", aconselhou sabiamente e ajudou de inúmeras maneiras durante minha estada em Blacksburg, na Virginia Tech. Ladislau Conceição (CTI) ofereceu ajuda e comentários críticos para o sucesso da tese, especialmente sobre a tradução EXPRESS-IDL.

Muitas outras pessoas foram importantes para o sucesso da pesquisa doutoral, incluindo: Dr. Carlos F. Bremer (USP e Univ. Aachen); Carlos Pittaluga Niederauer e Rejane Oliveira (CNPq); Fernando Montenegro e Dr. Oscar Lopez (UFSC); Edward Barkmeyer e Neil Christopher (NIST); Eliane Campregher (UNIVALI); Dr. Krishna K. Krishnan, Dr. Mauro J. Atalla e Nei Mueller (Virginia Tech); Manuel Montenegro (MRE); Marilena Deschamps; e Dr. Rogério Barra (CTI e PDES, Inc.). Ainda, muitas outras pessoas me apoiaram em assuntos não diretamente relacionados com a tese. Não posso mencionar todos (é todo um novo volume), mas por favor aceitem minha profunda gratidão. Finalmente, agradeço aos meus alunos, que me mantêm motivado para ser o melhor professor e orientador que eu puder ser.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Os dados manipulados por aplicações de engenharia têm sido tratados usando sistemas de gerência de banco de dados ou mecanismos dedicados embutidos em sistemas CAx. As tendências atuais de competitividade industrial apontam para a necessidade de integrar as aplicações de engenharia. Duas demandas principais se manifestam: o uso de um mecanismo para o acesso interoperável em rede aos dados, e a necessidade de manipular modelos de dados baseados em diferentes paradigmas. Esta tese de doutorado apresenta uma revisão sobre tecnologia de banco de dados com ênfase em aplicações de engenharia, introduz os problemas de intercâmbio de dados de produtos e interoperabilidade de aplicações, e discute o problema de perda semântica na tradução de modelos de dados de produtos compartilhados em rede e baseados em formatos padrão. Uma análise dos problemas que emergem nesta tradução, com o objetivo de avaliar a manutenibilidade da semântica dos dados ao longo de uma rede interoperável, é executada e comentada.

PALAVRAS-CHAVES

Intercâmbio de dados de produtos; Compartilhamento de dados; STEP (STandard for the Exchange of Product model data); PDES (Product Data Exchange using STEP); EXPRESS; SDAI (Standard Data Access Interface); CORBA (Common Object Request Broker Architecture); IDL (Interface Definition Language); Interoperabilidade; Bancos de dados para engenharia; Empresas virtuais; Empresas virtuais industriais

PREFÁCIO

A pesquisa de doutorado relatada nesta tese foi parte do programa de doutorado "sandwich" do CNPq. Os créditos foram cumpridos na Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia de Produção (PPGEP/UFSC, em Florianópolis, SC). De agosto de 1994 a novembro de 1995 a pesquisa foi conduzida na Virginia Polytechnic Institute and State University (Virginia Tech, em Blacksburg, VA, EUA). Entre novembro de 1995 e outubro de 1996, a pesquisa foi conduzida no National Institute of Standards and Technology (NIST, em Gaithersburg, MD, EUA). O projeto de tese foi submetido a um exame de qualificação no PPGEP/UFSC em junho de 1996, e a defesa da tese ocorreu em dezembro de 1997, também no PPGEP/UFSC.

Na fase de revisão da literatura, durante o período no exterior, foram publicados um artigo de disciplina (KERN 1994) e três artigos em congressos (KERN; BØHN 1995) (KERN; BØHN; BARCIA 1996) (KERN; BARRA; BARCIA 1996). Tanto os artigos quanto esta tese foram escritos em inglês. Esta versão em português foi gerada para atender à recente determinação da UFSC, que estabelece que as teses devem ser publicadas em nossa língua pátria.

A versão original em inglês foi incluída no apêndice. O autor recomenda fortemente a leitura do texto original. Há muitos termos técnicos cujo significado não deixa margem a dúvidas quando em inglês e no contexto adequado, enquanto geram longos e maçantes parágrafos explicativos quando se tenta vertê-los para o português (curiosamente, o tema desta tese é a perda semântica...). O mesmo ocorre em outras línguas -- os autores franceses escrevem em inglês, e a única língua na qual se encontra uma quantidade razoável de artigos significativos na área é o alemão, embora os falantes desta língua também publiquem predominantemente em inglês.

A versão em inglês é completa, com anexos e numeração própria de páginas, figuras e tabelas. Uma versão digital desta tese está prevista para ser disponibilizada na página do autor no PPGEP/UFSC, em http://www.eps.ufsc.br/~kern.

CAPÍTULO 1

INTRODUÇÃO

> "Uma mensagem para os fazedores de mapas: as estradas não são pintadas de vermelho, os rios não têm linhas de limite territorial correndo pelo meio, e não se podem ver curvas de nível em uma montanha".
> (KENT 1978)

1.1 Motivação

A tecnologia de automação industrial progrediu dramaticamente durante as últimas décadas. Os sistemas CAx (aplicações de engenharia, como: CAD, CAM, CIM, etc.) oferecem soluções de alto desempenho. Desde controle numérico (NC, CNC, DNC) até sistemas flexíveis de manufatura e estações de trabalho para *solid model-based design*, a tecnologia de automação avançou e tornou-se mais sofisticada para satisfazer as necessidades específicas da indústria (YANG 1993). A integração destas tecnologias é fator crítico para a competitividade industrial. A natureza complexa dos dados de engenharia pode dificultar a integração das aplicações. WILSON (1987) aponta "duas pedras no caminho" que têm impedido a integração efetiva de sistemas CAx:

1. Os sistemas CAx atuais foram projetados para entrada e saída de *dados*, em vez de *informação*; e
2. As ferramentas CAx atuais operam em níveis de abstração diferentes de um produto.

Portanto, a qualidade da modelagem da *informação* (dados com significado ou semântica--restrições, regras, procedimentos) sobre produtos é um dos fatores importantes para a integração de sistemas CAx. Além disso, os dados precisam ser trocados entre aplicações que estão baseadas em modelos de dados distintos, o que cria a necessidade de tradução dos dados. Para atacar este problema, a Organização Internacional de Padronização (ISO) lançou o Padrão para o Intercâmbio de Modelos de Dados de Produtos (ISO 10303-1 1992), conhecido informalmente como STEP.

STEP visa a representar todas as informações sobre um produto em todo o seu ciclo de vida. Diferentes aplicativos podem trocar informações através de um formato padrão definido por STEP. Os modelos de dados STEP são normalizados (isto é, em conformidade com as *formas normais* – vide 2.1) e escritos em EXPRESS (ISO 10303-11 1994), uma "linguagem objeto-assemelhada para a especificação de modelos de informação" (SCHENCK; WILSON 1994) que permite a especificação de modelos de dados complexos, com herança múltipla.

A figura 1 ilustra o problema do intercâmbio de dados de produtos com o exemplo do Programa EMB 145 da Embraer, um projeto para construir um avião contando com a parceria de diversos fornecedores, com o primeiro vôo previsto para 1995 (CECCHINI 1996). As caixas mostram nomes de fornecedores e seus respectivos sistemas CAD. Projetos necessitavam ser transferidos entre a Embraer e cada um dos parceiros, mas o sistema CAD usado pela Embraer (Intergraph) era diferente dos sistemas CAD usados pelos diversos fornecedores. Nesta situação, cada projeto transferido de um parceiro para a Embraer deveria ser traduzido para o modelo de dados do sistema Intergraph, enquanto um modelo de dados transferido a partir da Embraer teria que ser traduzido para o modelo de dados do sistema CAD do parceiro.

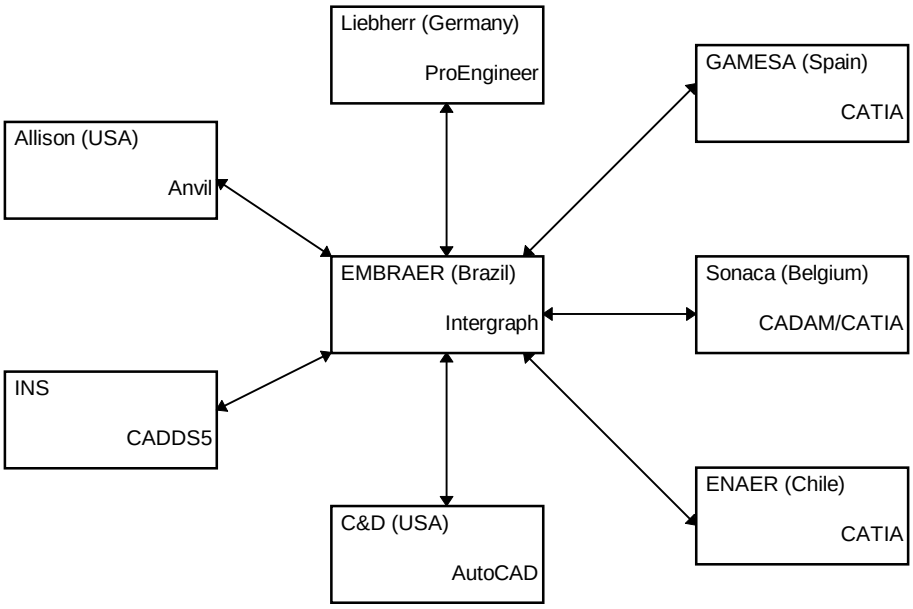


Figura 1 - Ilustração de intercâmbio de dados CAD (CECCHINI 1996)

Havia algumas soluções, naquela época, para o intercâmbio de dados entre dois modelos de dados específicos (na forma de um filtro implementado em um sistema CAD que traduzia dados para o modelo de dados de outro sistema CAD) e

entre um formato proprietário e um formato neutro (na forma de um pós- e um pré-processador em um sistema CAD que traduzia dados de e para um formato neutro, como IGES e SET). Entretanto, nenhuma destas soluções era considerada estável, eficiente, econômica, ou de outra forma justificável para adoção no âmbito do intercâmbio de dados no Programa EMB 145. A solução escolhida para o problema foi forçar os fornecedores a entregarem seus projetos de acordo com o modelo de dados do sistema CAD usado pela Embraer, implicando a necessidade de cada fornecedor comprar, treinar usuários, e usar uma versão do mesmo sistema CAD.

A situação recém-descrita era (e ainda é) muito comum na indústria. Naquelas circunstâncias, não era possível que cada parceiro usasse sua melhor experiência, seu sistema CAx específico, e tomasse parte no projeto EMB 145 sem que fosse necessário reentrar todos os dados recebidos no formato do sistema Intergraph no seu sistema específico, ou reentrar no Intergraph todos os dados produzidos no seu sistema específico, antes de enviar para transferência para a Embraer. Este trabalho redundante não seria necessário se houvesse um sistema que permitisse a conversão de dados entre vários sistemas CAx e o acesso aos dados em um nível de granularidade fino, isto é, em pequenas porções de informação, acessadas conforme a necessidade.

Além do problema do intercâmbio de dados de produtos, a necessidade de acesso aos dados entre as diversas aplicações de engenharia dá origem ao problema de *interoperabilidade* entre aplicações. As aplicações são desenvolvidas em linguagens de programação diferentes, para funcionar em sistemas operacionais diferentes, em localizações diferentes, e suportando paradigmas de banco de dados diferentes. Interoperabilidade é definida como "a interconexão efetiva de dois ou mais sistemas de computador, bancos de dados, ou redes diferentes, com o propósito de suportar computação distribuída e/ou intercâmbio de dados" (OFFICE OF SCIENCE 1994).

Para prover o acesso interoperável entre diferentes aplicações, o Grupo de Gerência de Objetos (OMG) lançou a Arquitetura de Corretor Comum para Chamadas a Objetos (CORBA), cujo cerne é o Corretor para Chamadas a Objetos (ORB). O ORB provê “os mecanismos básicos para fazer chamadas de modo transparente e receber respostas de objetos localizados local ou remotamente, sem que o cliente necessite ter conhecimento sobre os mecanismos usados para representar, comunicar-se com, ativar, ou armazenar os objetos (OMG 1993).

Em um ambiente de interoperabilidade em rede, como o descrito, os dados sobre produtos são representados em EXPRESS, e podem ser acessados através de chamadas escritas na Linguagem de Definição de Interface (IDL), uma linguagem

declarativa cuja sintaxe lembra a de C++, na qual interfaces de objetos são publicadas de acordo com a arquitetura CORBA. A ISO publicou um esboço do padrão de mapeamento da Interface Padrão de Acesso a Dados (SDAI), o documento STEP dedicado a implementações de compartilhamento (ISO 10303-22 1995), para IDL (ISO 10303-26 1997).

A necessidade de um mapeamento EXPRESS-IDL e mapeamentos posteriores para linguagens de implementação (uma vez que EXPRESS e IDL são linguagens declarativas) gera preocupações quanto à manutenibilidade da semântica dos modelos de dados de produtos compartilhados em rede heterogênea. De acordo com HARDWICK; LOFFREDO (1995), as seguintes características de EXPRESS podem requerer codificações ou outras manipulações para preservar a informação original de um modelo de dados nativo: entidades, herança, tipos primitivos, enumerações, seleções, e tipos agregados. KIEKENBECK et al. (1995) corroboram esta idéia afirmando que o mapeamento de EXPRESS para outra linguagem "requer um mapeamento abrangente do sistema de tipos EXPRESS para a linguagem de destino. A perda de partes do sistema de tipos EXPRESS através do mapeamento limitaria a utilidade do código gerado".

1.2 Definição do Problema

Esta tese aborda o seguinte problema de pesquisa:

> Quais são as perdas semânticas devidas especificamente ao mapeamento EXPRESS-IDL no compartilhamento baseado em padrões e interoperável em rede de modelos de dados de produtos; e como é possível aliviar estas perdas?

1.3 Relevância

Estima-se que 70% de todos os projetos industriais sejam re-projetos (KIGGANS 1996). Diferentes aplicações de engenharia têm de lidar com o mesmo projeto em evolução, em diferentes aspectos. Uma vez que as aplicações usam modelos de dados diferentes e podem funcionar em ambientes diferentes, há uma lacuna que impede a comunicação de informações sobre projetos.

A integração de aplicações de engenharia é fator crítico para a integração de empresas industriais. Conforme apontam Rando e Paoloni,

> "As indústrias de manufatura obtiveram êxito implementando soluções pontuais muito sofisticadas, porém, elas estiveram impossibilitadas de integrar estas soluções com sistemas empresariais genéricos." (RANDO; PAOLONI 1994)

No aspecto vertical de integração de empresas, *engenharia concorrente*, diferentes grupos de engenheiros trabalham colaborativamente e simultaneamente em diferentes tarefas do projeto e manufatura de um produto (KERN; BØHN 1995). No aspecto horizontal, *empresas virtuais*, várias corporações combinam suas especialidades para criar um produto. Uma empresa virtual deve poder formar-se rapidamente para investir em oportunidades, e dissolver-se também rapidamente quando a oportunidade cessa (HARDWICK et al. 1996). Esta tese aborda a manutenibilidade da semântica no compartilhamento de modelos de dados de produtos em rede heterogênea, em favor da integração das aplicações de engenharia, que é necessária para a realização da engenharia concorrente e das empresas virtuais.

1.4 Objetivos

O objetivo geral desta tese é desenvolver conhecimento e técnicas para a realização das empresas virtuais industriais.

Especificamente, esta tese tem os seguintes objetivos:

- Apresentar uma visão geral sobre gerência de bancos de dados com ênfase na adequabilidade e aplicabilidade de técnicas e ferramentas de bancos de dados para aplicações de engenharia;
- Descrever o problema de intercâmbio de dados de produtos e uma solução baseada em padrões para este problema;
- Apresentar o conceito de interoperabilidade de aplicações em rede heterogênea e uma solução baseada em padrões para a integração de modelos de dados de produtos para acesso interoperável em rede;
- Categorizar as oportunidades para perda semântica no compartilhamento em rede heterogênea de modelos de dados de produtos; e
- Apresentar uma avaliação das perdas semânticas no mapeamento de modelos de dados de produtos baseados em padrões e destinados ao acesso interoperável em rede heterogênea: descrever a natureza da perda, ações possíveis para aliviar a perda, e identificar as perdas que são intrínsecas à escolha das linguagens que representam os modelos de dados de produtos.

1.5 Escopo

Esta seção descreve o escopo da tese, que apresenta o acoplamento de STEP e CORBA como arcabouço para a integração baseada em padrões de modelos de dados de produtos compartilhados em rede heterogênea, introduz questões tecnológicas relativas a este acoplamento, e delineia o tópico específico de pesquisa.

A adoção de um padrão para o intercâmbio de modelos de dados de produtos (ISO 10303-1 1992), integrado ao uso de um padrão para a interoperabilidade em rede heterogênea (OMG 1995), permite viabilizar o acesso em rede a modelos de dados de produtos. STEP oferece o formato para a especificação dos modelos de dados de produtos e uma interface padrão, independente de linguagem, para o acesso aos dados (ISO 10303-22 1995). CORBA oferece "um conjunto de interfaces objeto-orientadas que suportam a construção e integração de componentes de software objeto-orientados em ambientes distribuídos heterogêneos" (BRANDO 1996).

O acoplamento STEP-CORBA define como representar e acessar dados de produtos. A seleção de técnicas específicas de gerência de banco de dados, como: controle de transações, controle de concorrência, e gerência de versões, está fora do escopo da integração STEP-CORBA, e fora do escopo desta tese. Considera-se que cada aplicação de engenharia tem suas próprias técnicas de gerência de banco de dados, seja usando um sistema de gerência de banco de dados (DBMS), ou um mecanismo embutido em um sistema CAx.

A realização da integração de modelos de dados de produtos depende da existência de Protocolos de Aplicação (APs), que são partes do padrão STEP. Um AP é um esquema conceitual escrito em EXPRESS para um certo domínio de aplicação. Os APs são os modelos conceituais a serem implementados em conjunto com um dos métodos de implementação STEP (veja a seção 2.2).

Nesta tese, o método de implementação STEP usado é SDAI, a interface padrão de acesso aos dados. SDAI é uma Interface de Programação de Aplicação (API) independente de linguagem, com um mapeamento para CORBA IDL já definido. Objetos STEP, representados em EXPRESS, são mapeados para IDL para a construção das interfaces a serem publicadas em ORBs para o acesso em rede. Neste mapeamento pode haver perda de informação. A manutenibilidade da semântica na tradução EXPRESS-IDL é o tópico específico abordado por esta tese.

A metodologia para a análise da perda semântica na tradução de modelos de dados de produtos baseados em padrões e interoperáveis em rede heterogênea foi ilustrada através da construção de uma série ou suíte de casos, cada um enfocando um aspecto dos tipos de dados EXPRESS. Os modelos de dados de produtos nesta

suíte foram traduzidos para IDL, sujeitos à especificação da linguagem EXPRESS (ISO 10303-11 1994) e ao mapeamento da SDAI para IDL (ISO 10303-26 1997). Esta tradução foi realizada com o uso de tradutores parciais de EXPRESS para IDL, alguma verificação e correções manuais, e código escrito à mão quando não havia tradutor disponível. O código IDL produzido foi comparado com os modelos de dados de produtos originais, e uma análise da perda semântica foi feita. Ações para aliviar o efeito destas perdas foram sugeridas, e direções foram recomendadas para trabalho futuro quanto à manutenibilidade da semântica de dados de produtos.

A pesquisa apresentada nesta tese foi baseada nas seguintes versões de padrões: EXPRESS - padrão internacional (ISO 10303-11 1994); SDAI - esboço do comitê normalizador (ISO 10303-22 1995); Mapeamento SDAI-IDL - esboço do comitê normalizador (ISO 10303-26 1997); CORBA - versão 2.0 (OMG 1995).

1.6 Organização do Documento

Esta tese é composta de cinco capítulos: (1) introdução, (2) revisão da literatura, (3) uma apresentação de uma abordagem baseada em padrões para a integração em rede heterogênea de modelos de dados de produtos, (4) uma análise da perda semântica na tradução de modelos de dados de produtos para acesso interoperável em rede heterogênea, e (5) conclusão. O documento é complementado por referências bibliográficas e apêndice.

O capítulo 2 apresenta uma revisão da literatura em três partes: primeiro, apresenta-se uma visão geral sobre tecnologia de gerência de banco de dados com ênfase na adequabilidade e aplicabilidade para aplicações de engenharia. A seguir, o problema de intercâmbio de dados de produtos (PDE) é introduzido, seguido por uma descrição de STEP, o padrão ISO 10303. Finalmente, o problema de interoperabilidade de aplicações em rede heterogênea é introduzido, juntamente com uma descrição da especificação CORBA.

O capítulo 3 apresenta uma abordagem baseada em padrões para o compartilhamento em rede heterogênea de modelos de dados de produtos. STEP é adotado como o padrão para o intercâmbio de dados de produtos, enquanto CORBA oferece a especificação padrão para a interoperabilidade de aplicações. No núcleo desta integração está a tradução de modelos de dados de produtos escritos em EXPRESS, a linguagem de modelagem de dados de STEP, para IDL, a linguagem na qual objetos publicam suas interfaces em ORBs, de acordo com a arquitetura CORBA.

O capítulo 4 apresenta uma análise da perda semântica na tradução de modelos de dados de produtos para acesso em rede heterogênea. Uma suíte de

modelos de dados é produzida, onde cada modelo enfoca um aspecto dos tipos de dados da linguagem EXPRESS. Os modelos são, então, traduzidos para IDL de acordo com o mapeamento padrão SDAI (ISO 10303-22 1995) para IDL (ISO 10303-26 1997). A perda semântica observada nesta tradução, definida como a disparidade entre os tipos de dados das duas linguagens, é analisada.

Finalmente, o capítulo 5 apresenta um resumo e as contribuições da tese, juntamente com recomendações para futuros trabalhos. Uma seção de referências bibliográficas complementa o texto, seguida de um apêndice que inclui a versão original da tese em inglês, com anexos e numeração própria de páginas, figuras e tabelas.

CAPÍTULO 2

REVISÃO DA LITERATURA

Neste capítulo, uma revisão da literatura em três áreas fundamentais é apresentada: gerência de banco de dados, intercâmbio de dados de produtos, e interoperabilidade de aplicações.

2.1 Gerência de Banco de Dados

Esta seção discute os aspectos mais importantes da gerência de banco de dados para a implementação de bancos de dados de produtos interoperáveis em rede heterogênea.

Modelagem de dados

Modelos de dados são usados para estabelecer o fundamento arquitetônico de bancos de dados. Cada modelo de dados objetiva "modelar o mundo" tão acuradamente quanto possível. Os modelos de dados podem ser classificados em:

- Hierárquicos: neste modelo, o banco de dados tem uma estrutura de árvore na qual cada registro tem apenas um ascendente, com a exceção do registro-raiz, que não tem ascendente. O modelo hierárquico deu a fundamentação para os primeiros sistemas de gerência de banco de dados (DBMS).
- Rede: é uma generalização do modelo hierárquico. O modelo de rede permite que cada registro tenha vários ascendentes e descendentes, como em um grafo.
- Relacional: suas fundações foram estabelecidas por CODD (1970). O modelo relacional suporta a abstração de sistemas de banco de dados que podem ser visualizados como coleções de tabelas e relacionamentos entre tabelas.
- Objeto-orientado: este paradigma tem os *tipos abstratos de dados*, a *herança*, e a *identidade* como seus aspectos mais fundamentais (KHOSHAFIAN 1993). Suporta o rico modelo de dados disponibilizado inicialmente através de linguagens de programação objeto-orientadas.

Há também os modelos de dados *semânticos*, que são conceitualizações em um nível de abstração mais alto. Sua aplicação não está restrita a um dos quatro

modelos acima. O modelo semântico mais conhecido é o modelo Entidade-Relacionamento (ER), de CHEN (1976).

As *formas normais* são um conjunto de critérios que visam a dar o nível de granularidade mais específico possível a tabelas. Embora criadas no escopo do modelo relacional, as formas normais vêm sendo usadas também em conjunção com o modelo objeto-orientado, objetivando a criação de modelos de dados *normalizados*. Dentre as várias regras normalização, a Forma Normal de Boyce-Codd (FNBC) é usualmente aceita como um bom critério de projeto. Segundo a FNBC, uma relação (tabela) está normalizada quando todo atributo ou conjunto de atributos cujo conteúdo determina o conteúdo de outro(s) atributo(s) deve necessariamente ser uma *chave* (ou seja, deve determinar o conteúdo de todos os atributos) desta relação. A figura 2 ilustra o redesenho da relação `Departamento`, que não atende ao critério da FNBC -- o código do funcionário chefe do departamento (`Chefe.#Func`) determina o nome do chefe (`Chefe.NomeFunc`), mas não é uma chave da relação. Para atender ao critério da FNBC, o atributo `Chefe.NomeFunc` foi projetado para fora da relação, gerando as relações `Departamento` e `Funcionário`, onde as chaves estão em negrito.

```
Relação não-normalizada:
Departamento (#Dep, NomeDep, Chefe.#Func, Chefe.NomeFunc )

Relações normalizadas segundo a FNBC:
Departamento ( #Dep, NomeDep, Chefe.#Func )
Funcionário  ( #Func, NomeFunc )
```

Figura 2 - Exemplo de uma relação e seu redesenho de acordo com a FNBC

Natureza dos dados

Os dados de um ambiente de negócios enquadram-se bem em tabelas ou arquivos contínuos, e são especialmente adequados para a gerência por sistemas relacionais (KERN 1994). Estes dados são cadeias de caracteres e números organizados em registros de tamanhos fixos, com apenas uma versão (a atual) dos dados em um certo momento.

Dados em um ambiente de engenharia, como aqueles manipulados por sistemas CAx, dificilmente enquadram-se em tabelas ou arquivos contínuos. Dados de aplicações de engenharia são, além de números e cadeias de caracteres, também gráficos bitmap, vetores e matrizes, etc. Objetos de engenharia variam em tipo e tamanho e relacionam-se conforme malhas complexas. Como exemplo de tamanho

dos dados em uma aplicação de engenharia, HARDWICK et al. (1996) registram a criação de um banco de dados restrito aos dados sobre a montagem mecânica para um eixo do Humvee, um veículo para qualquer tipo de terreno. O modelo de informação contém dois megabytes de dados no formato STEP, armazenados como 80.000 ocorrências. Se estes dados são expandidos para um banco de dados completo para o motor do veículo, os autores postulam que o banco de dados será 1.000 vezes maior, considerando que tratam-se apenas de dados sobre a montagem mecânica.

DBMSs

Sistemas de gerência de banco de dados, ou DBMSs, gerenciam dados independentemente de qualquer aplicação. Eles oferecem ferramentas para executar tarefas de gerência de dados, como: controle de acesso concorrente, evolução do esquema de banco de dados, e controle da unicidade da informação.

Os DBMSs que suportam o modelo relacional têm sido muito bem sucedidos no gerenciamento de dados de negócios. No entanto, eles apresentam uma série de insuficiências quando usados para gerenciar aplicações complexas, como sistemas CAx, conforme descrevem JOSEPH et al. (1991).

Os Bancos de Dados Objeto-Orientados (OODBs) emergiram como extensão do poderoso modelo de dados oferecido pelas linguagens de programação objeto-orientadas. HEILER et al. (1987) assinalam que o processo de definir uma versão inicial de um projeto de engenharia e alterar este projeto é muito semelhante ao processo de definir objetos e refiná-los sucessivamente, especificando restrições e construindo hierarquias de objetos. Neste sentido, a orientação a objetos é um modelo genérico que facilita o mapeamento do modelo mental de objetos do projetista para o sistema de ferramentas de projeto de engenharia, e da interface de usuário para as ferramentas do sistema subjacente (KERN 1994).

Enquanto os bancos de dados relacionais representam uma tecnologia sólida e têm uma linguagem de consulta padrão, o SQL, há muito ainda a evoluir no que tange a OODBs. Dois *manifestos* descrevem visões algo antagônicas sobre o que é ou deveria ser um OODB (ATKINSON et al. 1989, THE COMMITTEE 1990).

2.2 Intercâmbio de Dados de Produtos

A demanda pelo intercâmbio de dados de produtos surge quando dados de produtos necessitam ser transferidos entre diferentes aplicações. Isto pode ser causado por

necessidades de comunicação entre diferentes equipes de engenharia, departamentos, ou empresas (KERN; BØHN 1995).

Abordagens ao Intercâmbio de Dados de Produtos

Há soluções informais triviais para o problema de intercâmbio de dados de produtos, incluindo: enviar desenhos por fax, reentrar dados em um formato específico, ou eliminar o problema através da determinação de que todas as partes envolvidas devem usar um mesmo sistema CAx. Na maior parte dos casos, entretanto, estas soluções não são aceitáveis, sendo necessário traduzir os dados de um sistema para outro . Há duas abordagens possíveis para a tradução de dados, como mostra a figura 3: tradução direta e tradução com formato neutro.

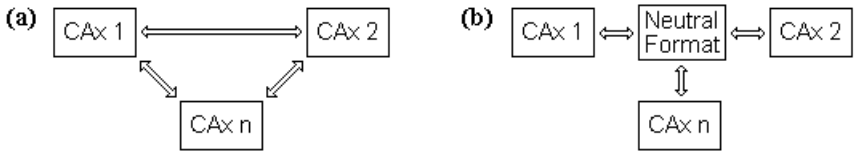


Figura 3 - Tradução direta (a) e com formato neutro (b)

Enquanto os tradutores diretos oferecem melhores oportunidades para otimizações e para a captura de idiossincrasias dos sistemas CAx envolvidos, cada nova versão de um sistema CAx adquirida pela empresa usuária demanda a existência de tradutores *de* e *para* todos os outros sistemas CAx. O uso de tradutor com formato neutro supõe a existência de uma especificação de formato de dados que seja neutra, de domínio público, e sobre a qual haja concordância generalizada quanto ao uso. Cada novo sistema CAx adquirido demanda a existência de um pré- e um pós-processador para executar traduções *de* e *para* o formato neutro. Assim, sendo *n* o número de sistemas CAx em uso, necessitam-se *n(n-1)* tradutores na tradução direta, e *2n* tradutores na tradução com formato neutro intermediário.

Evolução dos Padrões para o Intercâmbio de Dados de Produtos

As primeiras especificações para intercâmbio de dados tratavam, genericamente, de dados geométricos. Entre estas, a mais conhecida é o DXF, formato proprietário da Autodesk, e os padrões IGES (Estados Unidos), SET (França), e VDA/FS (Alemanha). PDES, um sucessor precoce do padrão IGES, foi realinhado em suporte do padrão internacional STEP (KERN; BØHN 1995), ou ISO 10303.

A tabela 1 apresenta uma comparação descritiva de vários padrões para o intercâmbio de dados de produtos, onde ACIS é o núcleo do modelo geométrico adotado por vários sistemas CAD comerciais, entre eles: AutoCAD, CADKEY, MicroStation, PE/Solid Designer, and SolidEdge (REDONDO 1996).

Tabela 1 - Comparação de padrões para intercâmbio de dados de produtos (REDONDO 1996)

	IGES	SET	VDA/FS	STEP	ACIS
Escopo	• Modelos estruturais • Modelos de superfície • Modelos sólidos • Modelos para análise de elementos finitos • Desenhos técnicos	• Modelos estruturais • Modelos de superfície • Modelos sólidos • Desenhos técnicos	• Modelos de superfície	• Modelos de produtos para todo o ciclo de vida do produto	• Modelos estruturais • Modelos de superfície • Modelos sólidos
Características da especificação	• Coleção de entidades • Formato de arquivo	• Coleção de entidades • Formato de arquivo	• Coleção de entidades • Formato de arquivo	• Especificação formal de um modelo de produto • Definição formal de sintaxe de arquivo	• Núcleo de um modelador geométrico

Arquitetura e Desenvolvimento de STEP

STEP foi concebido para manipular informações sobre um produto em todo o seu ciclo de vida, desde o início do projeto até sua retirada de circulação). O padrão STEP é, na verdade, uma família de padrões organizados em uma estrutura similar à de um sistema de banco de dados, inspirada na arquitetura ANSI/SPARC para sistemas de bancos de dados (TSICHRITZIS; KLUG 1978):

- Camada de aplicação: Modelos de informação desenvolvidos por peritos e específicos de uma área de aplicação. São os Protocolos de Aplicação (APs), padrões da série 200, e as Construções de Aplicação Interpretadas (AICs), série 500, introduzida recentemente.
- Camada lógica: Biblioteca de modelos de informações de produtos chamada Recursos Integrados (IRs), séries 40 e 100, dedicada a descrever todos os domínios de interesse de modo único e não-ambíguo.
- Camada física: Métodos de implementação, série 20, que trata do mapeamento de esquemas de aplicação para alguma tecnologia de computação específica.

A tabela 2 apresenta as séries de documentos STEP e correspondentes números dos documentos ou Partes.

Tabela 2 - Séries de documentos STEP e camadas da arquitetura em três níveis

Série de documentos STEP	**Número dos documentos**	**Camada da arquitetura**
Introdutórios	Zero a nove	
Métodos descritivos	Série 10	
Métodos de implementação	Série **20**	**Camada física**
Metodologia de teste de conformidade	Série 30	
Recursos integrados (IRs)	Série **40 e 100**	**Camada lógica**
Protocolos de aplicação(APs) **Construções de aplicação interpretadas**	Série **200** Série **500**	**Camada de aplicação**
Suítes abstratas para testes	Série 300	

Toda especificação em STEP é escrita em EXPRESS (ISO 110303-11 1994), a linguagem de descrição de dados de STEP, ou em inglês puro e simples. EXPRESS foi concebida para a utilização na modelagem da informação, permitindo a representação não apenas de dados, mas restrições, regras, funções, e procedimentos. A tabela 3 resume as características da linguagem EXPRESS.

EXPRESS tem um sistema de tipos de dados muito rico, como ilustra a figura 4. O conceito de entidade é similar ao conceito de *entidade* no modelo ER e ao conceito de *classe* no modelo objeto-orientado. O mecanismo de herança tem similaridades com a programação orientada a objetos, sendo que a herança múltipla é permitida.

Tabela 3 - Características da linguagem EXPRESS (KERN; BØHN; BARCIA 1996)

EXPRESS é:	EXPRESS não é:
uma Linguagem de Descrição de Dados (DDL)	uma Linguagem de Manipulação de Dados (DML)
independente de tecnologia	uma metodologia
objeto-assemelhada (mas não restrita a sistemas objeto-orientados)	uma linguagem de programação
legível por seres humanos, processável por computador (mas não-executável)	

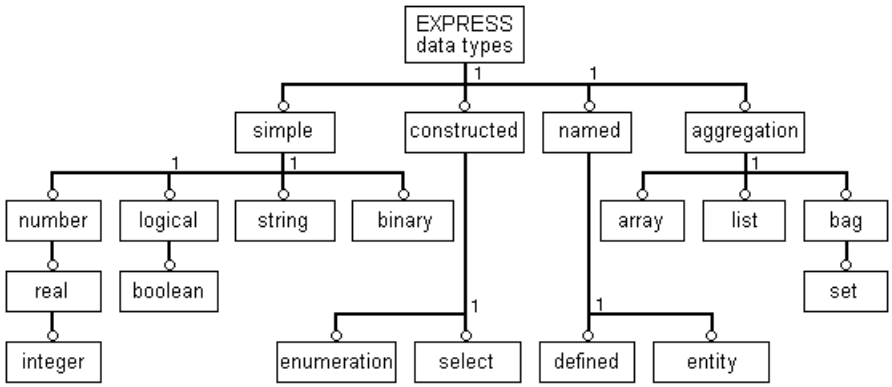


Figura 4 - Classificação de tipos de dados em EXPRESS, em EXPRESS-G (KIEKENBECK et al. 1995)

Modelagem da Informação em STEP

Os modelos de dados STEP são construídos utilizando uma abordagem *bottom-up* (de baixo para cima). Os IRs são uma coleção de esquemas normalizados, reutilizáveis, interconectados e não-ambíguos, divididos em *genéricos* e *de aplicação*. Os IRs genéricos prestam-se a modelagem de dados de propósitos gerais, enquanto os IRs de aplicação dedicam-se a modelar dados relativos a uma gama de aplicações. A figura 5 mostra um exemplo de entidade geométrica de propósito geral presente no IR *Representação Geométrica e Topológica* (ISO 10303-42 1992), com uma representação gráfica na figura 6, em EXPRESS-G. Os IRs servem como base para a construção dos APs, a segunda classe de modelos de informação STEP (KERN; BØHN; BARCIA 1996).

```
ENTITY path
  SUPERTYPE OF (ONEOF(open_path, edge_loop, oriented_path))
  SUBTYPE OF (topological_representation_item);
  edge_list  : LIST [1:?] OF UNIQUE oriented_edge;
WHERE
  WR1: path_head_to_tail(SELF);
END_ENTITY;
```

Figura 5 - Definição de uma entidade no documento STEP Parte 42

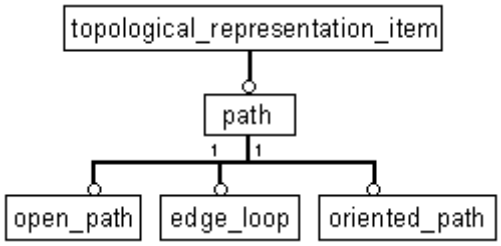


Figura 6 - Diagrama EXPRESS-G para as entidades presentes na figura 5

Os APs são os mais importantes e, com larga margem, os mais extensos dos padrões STEP (FOWLER 1995). O conceito de protocolo de aplicação foi desenvolvido para evitar implementações arbitrárias de apenas partes do padrão, como havia sido anteriormente o caso do padrão IGES. Para construir um AP, as estruturas nos IRs são reutilizadas e adaptadas para atender às necessidades de um domínio de aplicação específico. Os APs incluem uma seção sobre testes de conformidade que visa a verificar a conformidade com respeito ao padrão de uma implementação comercial de um AP.

Há, atualmente, vários APs em desenvolvimento pela ISO. O maior documento será, provavelmente, o AP 214 (ISO 10303-214 1995), que é uma especificação bastante abrangente voltada para aplicações da indústria automobilística.

Os modelos STEP normalizados e independentes de tecnologia são especificações conceituais que podem ser usadas por implementadores para construir aplicações que compartilham dados, sem importar qual a abordagem de banco de dados, rede, ou sistema operacional adotada em cada aplicação (KERN; BØHN; BARCIA 1996).

Implementação de STEP

A separação dos aspectos de implementação da modelagem da informação é uma das características-chaves em STEP. Para produzir uma implementação, um dos métodos de implementação STEP é combinado com um AP. Quatro níveis progressivos de implementação foram previstos:

- Nível 1: Intercâmbio de arquivos -- transferência passiva de arquivos-texto
- Nível 2: Intercâmbio em forma de trabalho -- transferência ativa assistida por software
- Nível 3: Intercâmbio de bancos de dados -- acesso compartilhado a bancos de dados

- Nível 4: Intercâmbio de bases de conhecimento -- integração de sistemas baseados em conhecimento

Os níveis 1 e 2 encontram-se bem desenvolvidos, e o nível 3 tem implementações comerciais e de pesquisa. Implementações no nível 4 encontram-se no estágio de pesquisa básica.

O documento STEP Parte 21 define uma maneira de codificar dados de produtos para o intercâmbio de arquivos, independentemente de aplicação. Arquivos "parte 21" permitem o intercâmbio apenas de esquemas STEP completos, sendo portanto adequados para implementações dos níveis 1 e 2.

A Interface Padrão de Acesso aos Dados (SDAI) é uma coleção de padrões STEP que suportam o compartilhamento de dados em nível de granularidade muito mais fino do que esquemas inteiros, aproveitando a característica de normalização dos modelos de dados STEP. SDAI especifica operadores de baixo nível e independentes de linguagem para criar, recuperar e excluir objetos, e para inserir, modificar e excluir atributos de objetos (URBAN et al. 1994). A figura 7 apresenta uma visão esquemática do compartilhamento de um banco de dados por várias aplicações, que acessam o banco de dados através da interface SDAI. Cada aplicação deve possuir um pré- e um pós-processador para executar a tradução de dados de produtos entre o seu modelo interno e o modelo de um AP.

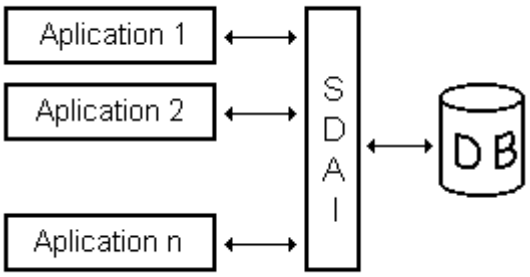


Figura 7 - Compartilhamento de um banco de dados STEP por várias aplicações

O documento SDAI principal é a Parte 22 (ISO 10303-22 1995). Há, ainda, mapeamentos de SDAI definidos para as linguagens: C++ (ISO 10303-23 1995), C (ISO 10303-24 1995), e CORBA IDL (ISO 10303-26 1997), com a possibilidade de surgirem novos mapeamentos. A Parte 26 não é propriamente um mapeamento para uma linguagem de programação, mas antes para a Linguagem de Definição de Interface (IDL), permitindo a integração de STEP com CORBA, visando à realização

dos bancos de dados de produtos interoperáveis em rede heterogênea. Existe uma quantidade de ferramentas públicas, disponíveis sem custo, que pretendem servir de suporte ao desenvolvimento de software para STEP (CLARK 1990, MORRIS 1990, LIBES 1993).

2.3 Interoperabilidade

A interoperabilidade é definida como “a interconexão efetiva de dois ou mais sistemas de computador, bancos de dados, ou redes diferentes, com o propósito de suportar computação distribuída e/ou intercâmbio de dados" (OFFICE OF SCIENCE 1994). Em um sistema de computação distribuído, as aplicações interoperam entre si, mas são executadas em diferentes locais, possivelmente sob diferentes plataformas de hardware e software.

Especificações para Interoperabilidade

Várias iniciativas de pesquisa, implementação e padronização têm abordado o problema da interoperabilidade. Três especificações (dois padrões e uma especificação proprietária) dedicadas a suportar a interoperabilidade entre aplicações foram lançadas recentemente: o Ambiente de Computação Distribuído (DCE), Embutimento e Ligação de Objetos/Modelo de Objeto Componente (OLE2/COM), e a Arquitetura de Corretor Comum para Chamadas a Objetos (CORBA).

DCE é um conjunto integrado de serviços projetado para suportar aplicações distribuídas. Foi desenvolvido pela Fundação para o Software Aberto (OSF). Chamadas de Procedimento Remotas (RPCs) acionam aplicações e sistemas operacionais para realizarem tarefas específicas, uma tarefa de cada vez.

OLE é uma tecnologia de objetos desenvolvida pela Microsoft, dedicada originalmente a suportar documentos compostos. COM foi desenvolvido em conjunto pela Microsoft e Digital Equipment Corporation para prover suporte a objetos distribuídos.

O Grupo de Gerência de Objetos (OMG), um consórcio de aproximadamente 600 empresas, desenvolveu CORBA como um padrão para interoperabilidade.

CORBA

A chave para o entendimento da estrutura da arquitetura CORBA reside no Modelo de Referência, ilustrado na figura 8 e definido na Arquitetura de Gerência de Objetos (OMA) da OMG (SOLEY; STONE 1995), que consiste de:

- Corretor para Chamadas a Objetos (ORB): Habilita os objetos a fazerem chamadas e receberem respostas de modo transparente em um ambiente distribuído.
- Serviços de Objeto: É uma coleção de serviços (interfaces e objetos) que suportam funções básicas para o uso e implementação de objetos. Os serviços são necessários para se construir qualquer aplicação distribuída e sempre são independentes de domínios de aplicações.
- Facilidades Comuns: É uma coleção de serviços que oferecem capacidades de propósito geral que são úteis para muitas aplicações (por exemplo: e-mail, impressão).
- Objetos de Aplicação: São objetos específicos de aplicações particulares de usuários. Não são padronizados pela OMG.

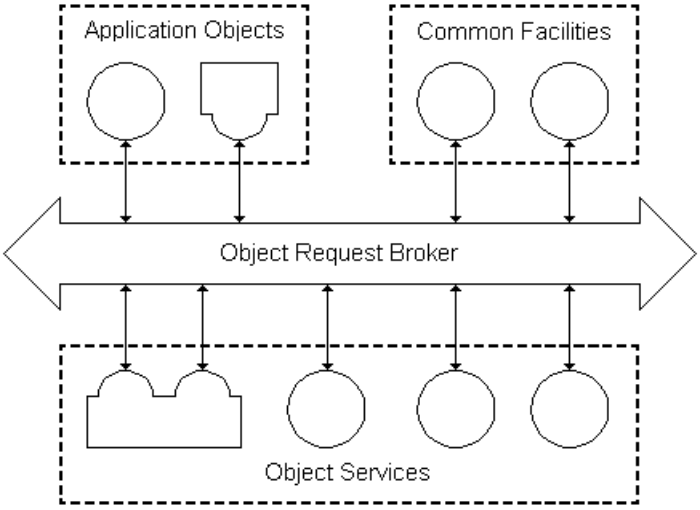


Figura 8 - Modelo de Referência OMA (SOLEY; STONE 1995)

Um ORB é como um comutador telefônico, oferecendo o mecanismo básico para fazer e receber chamadas (OMG 1995). Sua funcionalidade é explicada:

> "Usando um ORB, um cliente pode chamar de modo transparente e iniciar uma ação em um servidor de objetos, que pode estar na mesma máquina ou em outro ponto de uma rede. O ORB intercepta a chamada e localiza um objeto capaz de implementar a chamada. A seguir, passa os parâmetros ao objeto, invoca seu método, e retorna os resultados. O cliente não precisa estar ciente sobre onde o objeto está localizado, sua linguagem de programação, sistema operacional, ou qualquer outro aspecto do sistema que não é parte da interface do objeto. Desta forma, o ORB implementa a interoperabilidade entre aplicações e interconecta vários sistemas de objetos de modo imperceptível."
> (OMG 1993)

Os objetos disponibilizados através de um ORB publicam suas interfaces usando a linguagem IDL de CORBA. Em IDL, os atributos e operações de um objeto são especificados de modo independente de linguagem de programação. A gramática de IDL é um subconjunto do padrão proposto ANSI C++, com construções adicionais para suportar conceitos de objetos distribuídos (OMG 1995). A herança múltipla é permitida.

Uma vez que IDL é uma linguagem descritiva, as implementações são construídas usando mapeamentos de IDL para linguagens de programação. VINOSKI (1993) assinala que o fato de que IDL é declarativa intensifica a separação entre interface e implementação que é enfatizada no desenvolvimento de sistemas objeto-orientados.

Em comparação com DCE, CORBA representa uma abordagem objeto-orientada, enquanto DCE não (BRANDO 1996). Isto não implica em que DCE não pode ser usado com programação orientada a objetos, mas apenas que DCE “deixa de oferecer suporte explícito para o paradigma da orientação a objetos” (BRANDO 1996). VINOSKI (1993) dá um exemplo das desvantagens de tentar acoplar DCE e programação orientada a objetos:

> "Como despachar um objeto C++ através de uma rede de um processo a outro? ... O fato de que objetos C++ podem conter ponteiros ocultos para tabelas de funções virtuais acaba por criar uma pedra no caminho, mesmo para o programador mais determinado, especialmente quando redes de computadores heterogêneos estão envolvidas."

Como a maior parte dos ambientes objeto-orientados, CORBA suporta (BRANDO 1996):

- Encapsulamento
- Abstração
- Herança
- Polimorfismo
- Ligação tardia de acionamentos de operações para chamadas de funções
- Reutilização de classes e objetos

A criação de novas classes e objetos em tempo de execução, outra característica de sistemas objeto-orientados, não é suportada diretamente, mas CORBA inclui mecanismos suficientes para um implementador de ORB oferecer esta capacidade (BRANDO 1996).

De acordo com TIBBITS (1995), CORBA compartilha com OLE2/COM a utilização de princípios da programação orientada a objetos, incluindo herança, encapsulamento, e extensibilidade. Ambos contam com um ORB para fazer chamadas a objetos distribuídos. OLE 2.0, no entanto, não aborda o problema de integrar aplicações em plataformas não-PC. A especificação COM estipula como a funcionalidade OLE pode ser implementada em plataformas não-PC onde CORBA está sendo usada. Além disso, a OMG emitiu uma chamada de propostas (RFP) para estabelecer a interoperabilidade COM-CORBA.

MAYBEE et al. (1995) avaliam o estágio atual de OLE2/COM:

> "Até que COM ofereça suporte completo para objetos distribuídos, é prematuro fazer comparações diretas. Recentemente, a Microsoft começou a atacar estas questões usando DCE para oferecer o suporte subjacente para nomeação e ordenamento (*marshaling*). É adequado dizer que isto significa que COM terá os mesmos méritos e deméritos de DCE."

CAPÍTULO 3

MODELOS DE DADOS DE PRODUTOS COMPARTILHADOS EM REDE HETEROGÊNEA

Este capítulo introduz a integração dos padrões STEP e CORBA como tecnologia para habilitar o compartilhamento de modelos de dados de produtos interoperáveis em rede heterogênea. São apresentadas implicações desta escolha de padrões e os desafios à factibilidade da tradução de modelos de dados STEP. Uma análise da perda semântica nesta tradução é proposta.

3.1 Uma Abordagem Baseada em Padrões para o Compartilhamento de Modelos de Dados de Produtos em Rede Heterogênea

Na revisão da literatura, no capítulo 2, discutiram-se vários padrões: SQL para manipulação de banco de dados; STEP, IGES, SET, e VDA-FS para o intercâmbio de dados de produtos, e CORBA para interoperabilidade de aplicações. Padrões representam uma alternativa ao uso de especificações proprietárias, que geralmente precedem a padronização. Há vantagens na utilização de padrões:

> "A diferença básica entre padrões e especificações proprietárias é: padrões para sistemas abertos são acordos de consenso entre múltiplos fornecedores comerciais. Tipicamente, especificações proprietárias contém documentação sobre uma única implementação, compilada após a existência desta implementação. Em contraste, um padrão é, geralmente, um pacto tecnológico prospectivo construído sobre um modelo de referência de longo alcance, tal como a Arquitetura de Gerência de Objetos da OMG. Em geral, os padrões oferecem uma base mais estável e com menos riscos para o desenvolvimento de sistemas."
> (MOWBRAY 1996)

Enquanto há vários padrões que são potencialmente relevantes para sistemas de informação, há organizações (por exemplo, ISO, IEEE, e NIST) que oferecem documentos de orientação para a escolha dos padrões a adotar, chamados *ambientes de sistemas abertos* (MOWBRAY 1996).

A abordagem adotada para a interoperabilidade em rede heterogênea de modelos STEP origina-se na arquitetura CORBA da OMG (OMG 1995). CORBA e STEP são, também, juntamente com gerência de conhecimento, fluxo de trabalho, e a Internet, tecnologias que estão no âmago do projeto "Protocolos de Infra-estrutura de Informação Industrial" (NIIIP) do governo norte-americano, que tem como um de seus objetivos estabelecer um protocolo de infra-estrutura de software aberto e baseado em padrões que integre processos, informações, e ambientes de computador heterogêneos e distribuídos em toda a base manufatureira norte-americana (NIIIP 1996).

As indústrias de manufatura têm implementado com sucesso soluções pontuais muito sofisticadas, mas não tem sido possível integrar estas soluções em sistemas industriais mais amplos. Há uma demanda por integração na empresa industrial:

> "O objetivo, nos dias atuais, é uma completa reengenharia dos processos de manufatura, incluindo a integração de uma variedade de processos que incluem: manufatura, projeto, análise, simulação, e práticas comerciais. A integração vertical destes processos dentro de uma única empresa recebe o nome de *engenharia concorrente colaborativa*, e ainda não foi plenamente realizada. Ainda assim, as exigências das demandas atuais têm forçado as empresas a buscar mais do que a integração vertical, estabelecendo o objetivo de integrar horizontalmente os processos de manufatura na chamada *empresa virtual* (VE). A VE é uma aliança temporária de organizações com o propósito de desempenhar uma tarefa que nenhuma das organizações poderia realizar isoladamente."
> (RANDO; PAOLONI 1994)

A realização da integração das empresas industriais depende da integração bem-sucedida de sistemas CAx. O acoplamento de STEP e CORBA é uma solução baseada em padrões para esta integração. No âmbito de STEP, este acoplamento é abordado através do mapeamento de SDAI para CORBA IDL (ISO 10303-26 1997). Esta também é a abordagem adotada no projeto NIIIP.

Papel da Interface Padrão de Acesso aos Dados (SDAI)

Em STEP, SDAI é o padrão dedicado à implementação de modelos de dados de produtos compartilháveis. Uma vez que SDAI é escrita em EXPRESS, uma linguagem não orientada para implementação, STEP define mapeamentos de SDAI para linguagens de programação, como C (ISO 10303-24 1995) e C++ (ISO 10303-23 1995). Também, um esboço do mapeamento de SDAI para CORBA IDL foi lançado (ISO 10303-26 1997), visando ao acesso interoperável em rede heterogênea a modelos de dados STEP.

A figura 9 ilustra o acesso interoperável em rede heterogênea a modelos de dados STEP: as aplicações de usuário e os bancos de dados STEP podem estar em lugares diferentes, redes diferentes, conectados a diferentes implementações de SDAI e a diferentes ORBs. Uma requisição de dados feita por uma aplicação é passada para um ORB; o ORB comunica-se com outros ORBs para localizar o banco de dados; uma aplicação de banco de dados recebe a requisição e, usando operadores de banco de dados traduzidos a partir dos operadores elementares SDAI, realiza o acesso e entrega os dados de volta à aplicação requisitante.

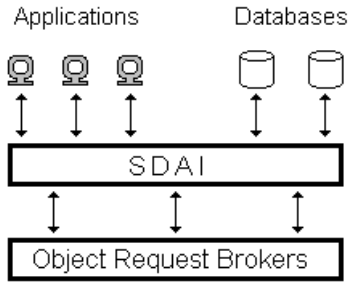


Figura 9 - Acesso interoperável em rede heterogênea a modelos de dados STEP

EXPRESS para IDL

Os modelos de dados STEP, escritos em EXPRESS, são objetos que podem ser compartilhados em uma rede heterogênea. Para publicar suas interfaces, os objetos STEP são traduzidos para IDL, conforme HARDWICK et al. (1996):

> "Para criar a infra-estrutura (para empresas de manufatura virtuais), a linguagem de definição de dados de CORBA, IDL, é combinada com a linguagem de definição de dados de STEP, EXPRESS. Estas duas linguagens têm propósitos distintos: IDL descreve interfaces para as aplicações, enquanto EXPRESS descreve modelos de dados normalizados. Ambas podem ser usadas para descrever objetos para aplicações de manufatura."

3.2 Implicações e Desafios Apresentados pela Integração Baseada em Padrões de Modelos de Dados de Produtos

A escolha de STEP e CORBA como padrões para a integração de modelos de dados de produtos interoperáveis em rede heterogênea implica em opções tecnológicas, e

apresenta desafios para a realização desta integração. HARDWICK (1996) identifica barreiras que impedem a comunicação de informação industrial em dentro de uma empresa virtual industrial:

- Controle de segurança insuficiente -- As empresas que colaboram em uma empresa virtual podem ser competidoras em outros projetos.
- Perda de controle sobre projetos -- As técnicas de controle diferem de empresa para empresa.
- Falta de interoperabilidade entre sistemas de aplicações -- Os dados produzidos por uma empresa não podem ser lidos pelo sistema de aplicações de outra empresa.
- Uso de tecnologias e sistemas de aplicações estranhos à empresa -- As tecnologias e sistemas de aplicações são, muitas vezes, complexas e requerem treinamento extensivo.

Os problemas de controle de segurança insuficiente e perda de controle sobre projetos estão fora do escopo do acoplamento STEP-CORBA; estes problemas estão sendo abordados com o desenvolvimento de protocolos de comunicação seguros (GANESAN; SANDHU 1994) e sistemas de controle de fluxo de trabalho (McCUSKER 1993). A integração de STEP e CORBA pode diminuir a barreira gerada por tecnologias e aplicações estranhas, uma vez que permite que cada empresa use os sistemas de sua escolha, compartilhando dados em formato padrão. De qualquer modo, a barreira mais importante que o acoplamento STEP-CORBA pode ajudar a vencer é a inabilidade de os sistemas de aplicações interoperarem. HARDWICK et al. (1996) acrescentam que “duas aplicações não conseguem fazer uso das funcionalidades uma da outra porque não podem invocar os recursos uma da outra”, referindo-se a esta problema como *interoperabilidade de código*.

As próximas seções relatam as implicações da escolha de STEP e CORBA e as dificuldades a serem superadas com respeito a três aspectos relacionados a interoperabilidade: gerência de banco de dados, interoperabilidade de aplicações, e representação de dados de produtos.

3.2.1 Tecnologia de Gerência de Banco de Dados

A adoção de STEP e CORBA não implica em qualquer restrição quanto à natureza da armazenagem. De acordo com McLAY; MORRIS (1990):

> "STEP proporciona a definição de informações sobre produtos, mas não define como os dados são armazenados. Algumas aplicações usarão arquivos de intercâmbio, enquanto outras podem demandar bancos de dados relacionais, ou ainda outras aplicações STEP podem usar bancos de dados objeto-orientados."

Questões relativas a desempenho ou controle de transações podem indicar direções quanto à opção por um sistema de banco de dados relacional, um banco de dados objeto-orientado, ou arquivos contínuos (*flat files*) (STI 1992), conforme descreve a tabela 4. Entretanto, mesmo que uma solução pontual ótima for encontrada, é improvável que seja igualmente adequada para toda a aplicação presente no ambiente distribuído de computação. A implementação de bancos de dados interoperáveis em rede heterogênea dependerá antes de uma combinação de sistemas e tecnologias diferentes, conforme anota KENT (1978):

Tabela 4 - Fatores que influenciam a escolha por uma tecnologia específica de banco de dados (resumido a partir de STI (1992))

Tipo de aplicação	• Aplicações de banco e companhias aéreas podem requerer características especiais de controle de concorrência, bloqueio, arquitetura cliente-servidor, etc., mas para a manipulação de geometria estas questões não são importantes. • Por exemplo, STEP Parte 41 [ISO 10303-41 1994] apresenta dados do tipo *lista de materiais*, que são apropriados para um banco de dados relacional.
Desempenho	• Um banco de dados relacional pode ser apropriado para poucas consultas. Para um número grande de consultas, e consultas sequenciais, bancos de dados relacionais são lentos, devido ao alto custo inicial das transações. • Arquivos STEP Parte 21 (arquivos-texto) representam uma escolha adequada para armazenagem se a aplicação processará todos, ou pelo menos a maior parte, dos dados em um conjunto. Todos os acessos subsequentes podem ser realizados na memória principal. • Bancos de dados objeto-orientados são a escolha mais promissora se a aplicação for acessar apenas alguns, ou um grande número de registros em um conjunto.
Controle de transações	• Bancos de dados relacionais são a melhor escolha para transações curtas. • Aplicações que processam apenas transações longas podem ser gerenciadas usando arquivos-texto. • Bancos de dados objeto-orientados são indicados para transações de duração mista. • Em muitas aplicações de engenharia, os usuários devem trabalhar em diferentes versões de um produto. Neste casos, o controle de versões é mais importante do que o controle de transações.
Versões	• Um bom sistema de versões deve facilitar a captura de mudanças de engenharia que, em um banco de dados, correspondem a muitas edições de muitos itens de banco de dados. • HARDWICK et al. (1995) relatam o uso de arquivos delta (texto) para comunicar mudanças de projeto.

> "Pois a verdade das coisas pode ser: coisas úteis acabam sendo feitas por ferramentas que são um amálgama de fragmentos de teorias."

Enquanto STEP é uma coleção de padrões para intercâmbio de dados de produtos prospectiva e independente de tecnologia, CORBA oferece a interoperabilidade das aplicações, que serve como um amálgama entre diferentes aplicações, redes, e bancos de dados baseados em paradigmas diferentes.
A adequação de cada tecnologia de banco de dados específica para aplicações complexas como sistemas CAx tem sido objeto de intenso debate. STONE; HENTCHEL (1990) observam, com respeito a esta 'guerra dos bancos de dados':

> "Como decidir qual tipo de banco de dados é o melhor quando mesmo os peritos não conseguem chegar a um acordo?"

A integração de STEP e CORBA permite a integração de diferentes abordagens de banco de dados, desde que as estruturas de dados possam ser mapeadas de acordo com os esquemas conceituais definidos nos APs.

HARDWICK; LOFFREDO (1995) descreveram mapeamentos entre EXPRESS e as linguagens de definição de dados (DDLs) de quatro DBMSs diferentes, e descobriram que algumas características do rico modelamento de dados em EXPRESS pode desafiar a capacidade das DDLs:

- Entidades;
- Herança;
- Tipos primitivos;
- Enumerações;
- Seleções; e
- Agregados.

O mapeamento de EXPRESS para SQL foi explorado por RAGHAVAN (1992), MORRIS (1990), MEAD; THOMAS (1989), e EGGERS (1988). SANDERSON; SPOONER (1993) descreveram mapeamentos de EXPRESS para os modelos de rede, hierárquico, e relacional, e mostraram que EXPRESS é semanticamente mais rico.

KERN; BØHN (1995) relatam a existência de implementações precoces de STEP usando bancos de dados objeto-orientados (OODBs). No entanto, a maior parte das implementações tratava o intercâmbio através de arquivos STEP Parte 21, usando

OODBs para armazenagem, mas não implementando o *compartilhamento* de modelos de dados de produtos.

3.2.2 Interoperabilidade de Aplicações

A escolha de CORBA como padrão para interoperablidade ainda é uma solução subtestada. Enquanto muitos autores elogiam a escolha, outros apontam os desafios a serem vencidos até a plena realização da interoperabilidade de aplicações com CORBA:

- Sistema de tipos fraco: CORBA não tem habilidade para organizar estruturas de dados com estruturas compartilhadas, tais como gráficos, que requerem cópias extras para mapear tipos de dados legados, e que códigos sejam passados por valor e não por referência (HEIMBIGNER 1995).
- Questões de tempo de execução: Seu suporte multilinguagem é limitado a C e C++ (HEIMBIGNER 1995).
- Segurança e autorização: Isto não é contemplado no escopo de CORBA.
- Persistência: Isto não é completamente contemplado no escopo de CORBA. De acordo com TIBBITS (1995):

> "CORBA oferece apenas um mecanismo que permite a uma aplicação converter uma referência a objeto para uma cadeia de caracteres e vice-versa. Esta cadeia de caracteres pode ser armazenada em disco e recuperada em algum momento no futuro, mas o processo não é exatamente um substituto para a armazenagem persistente. Basicamente, a persistência é deixada a cargo da implementação do ORB e do desenvolvedor da aplicação."

  Entretanto, há uma proposta para um Serviço de Objetos Persistentes (POS), que oferecerá uma interface genérica para persistência que permitirá que qualquer banco de dados seja acoplado como solução de armazenagem (*back end*) subjacente.

Apesar dos desafios apresentados, HEIMBIGNER (1995), um dos críticos, afirma que CORBA "é representativa dos problemas com outros padrões". Estes problemas não afetam a tendência de que CORBA ganhe espaço no mercado. Com efeito, WATSON (1996) se refere a um relatório recente sobre o mercado de objetos distribuídos produzido por Ovum Ltd., que estima que no fim da década de 1990 cerca de 9 milhões de computadores usarão CORBA, perfazendo cerca de 75% do mercado *middleware* de objetos distribuídos. VINOSKI (1993) afirma, em concordância:

> "A especificação CORBA está bem a caminho de se tornar a base-padrão para aplicações de objetos distribuídos. Seus componentes flexíveis permitem que sistemas legados trabalhem em conjunto com novas aplicações de modo imperceptível[1]."

3.2.3 Representação de Dados de Produtos

A escolha de STEP e CORBA para a realização da interoperabilidade em rede heterogênea de modelos de dados de produtos implica em que os dados de produtos serão representados em EXPRESS, de acordo com algum dos APs de STEP. Também implica em que o acesso interoperável em rede aos modelos de dados de produtos será efetivado através da publicação das interfaces dos objetos STEP para acesso pelos ORBs, de acordo com o mapeamento da SDAI para IDL. A efetividade desta representação dependerá de dois fatores: a qualidade dos modelos de dados STEP, e a qualidade da tradução dos modelos de dados escritos em EXPRESS para a publicação de suas interfaces em IDL para o acesso interoperável em rede.

Qualidade dos modelos STEP

STEP apresenta uma abordagem inovadora à modelagem de dados. O conceito de AP é um dos maiores avanços do padrão em relação a padrões e especificações anteriores. Ainda assim, persistem o debate sobre a conveniência de mudanças no processo de modelagem, bem como questões sobre a capacidade atual de STEP de preencher as necessidades da modelagem de dados de produtos.

Um dos requisitos que os modelos de dados STEP devem preencher é superar a diversidade na representação de dados gerada pelo uso de diferentes técnicas de projeto. De acordo com BARRA (1986), o único modo de possibilitar um intercâmbio de dados completamente isento de imperfeições é:

- Usar o mesmo sistema; e
- Usar as mesmas técnicas de modelagem.

Afora estas limitações, o melhor que se pode esperar é que os dados recebidos por um sistema CAx sejam *funcionalmente equivalentes* aos dados originados pelo sistema CAx transmissor. A satisfação desta condição está relacionada à noção de *completude*, um dos objetivos de STEP. À medida em que os dados sobre projeto de um produto são comunicados de uma aplicação a outra, os projetistas devem se certificar de que o sistema CAx receptor é capaz de interpretar corretamente os dados

[1] *Seamlessly*, no original.

recebidos. Além disso, dados sobre diferentes aplicações podem ser modelados por APs diferentes, criando a demanda por *interoperabilidade de APs*.

MEIS; OSTERMAYER (1996) propuseram mudanças ou melhorias ao processo de desenvolvimento STEP baseadas na abordagem da engenharia ontológica. METZGER (1996), um desenvolvedor de STEP, concluiu que "um modelo conceitual único para dados de produtos, descrito usando uma linguagem uniforme, não pode ser construído", e propôs "desistir da idéia de um modelo de dados unificado e completo", o que representaria uma mudança radical no processo de desenvolvimento STEP, que constrói os APs interpretando os IRs. Uma "delegação de tarefas de modelagem" é proposta:

> "A tarefa de modelagem deveria ser executada por grupos independentes e diferentes, não por um núcleo de generalistas 'integrando' o resultado. Cada grupo-tarefa deveria ser responsável pela qualidade de seu produto, a linguagem de domínio específico usada, a definição de termos adotada, o estilo de modelagem, etc." (METZGER 1996)

A proposta é apoiada com exemplos de como o conceito de uma *coisa* e *informação acerca de uma coisa* são confundidos em STEP, que tenta identificar conceitos diferentes usando uma linguagem uniforme (METZGER 1996), o que leva às seguintes inconsistências:

- No documento introdutório sobre STEP (ISO 10303-1 1992), *produto* é definido como "uma coisa ou substância produzida por um processo natural ou artificial".
- No documento STEP Parte 41 (ISO 10303-41 1994), um *produto* é "a identificação e descrição, no contexto de uma aplicação, de um objeto fisicamente realizável produzido por este processo".

Portanto, *produto* significa tanto a coisa quanto a informação sobre a coisa.

STEP é um esforço de padronização em evolução que já alterou alguns de seus métodos. As questões sobre a qualidade dos modelos STEP mencionadas acima indicam que a modelagem de dados STEP pode ainda sofrer mudanças, e mesmo mudanças importantes, que afetarão a qualidade dos modelos de dados de produtos.

Qualidade da tradução de modelos de dados EXPRESS para IDL

De acordo com SU et al. (1995), "EXPRESS tem uma semântica muito mais rica do que IDL, e tem um aspecto objeto-assemelhado". KIEKENBECK et al. (1995)

observam que o mapeamento de EXPRESS para outra linguagem demanda um mapeamento abrangente do sistema de tipos EXPRESS para a linguagem-destino, e a perda de partes do sistema de tipos EXPRESS através do mapeamento limitaria a utilidade do código gerado.

O sistema de tipos EXPRESS, ilustrado na figura 10, compreende os tipos: simples, agregados, nomeados, construídos, e generalizados, formando um gráfico de herança (KIEKENBECK et al. 1995). Os tipos simples, SELECT, agregados, e a herança e as entidades complexas têm peculiaridades desafiadoras para o mapeamento para outra linguagem, como é apresentado a seguir.

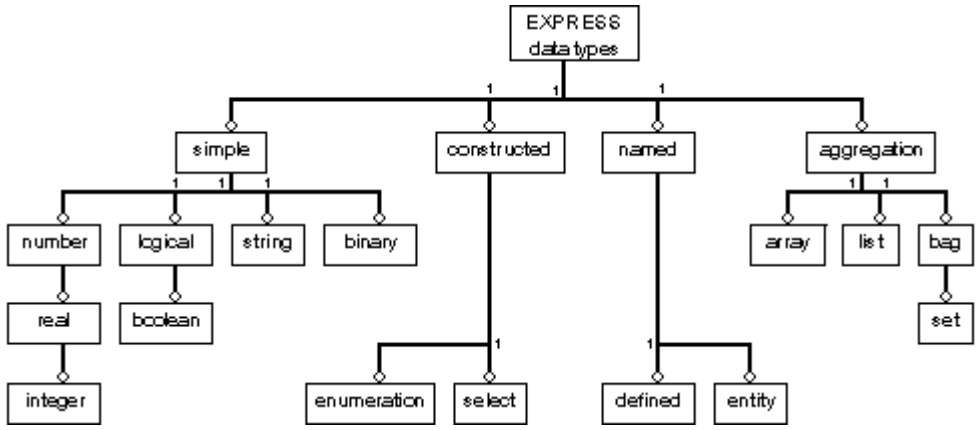


Figura 10 - Classificação dos tipos de dados EXPRESS, em EXPRESS-G (KIEKENBECK et al. 1995)

Tipos simples

Com respeito aos tipos simples na figura 10, algumas peculiaridades de EXPRESS são:

- O tipo NUMBER não é REAL nem INTEGER.
- O tipo de dados LOGICAL admite valores TRUE, FALSE, e UNKNOWN. O tipo BOOLEAN, entretanto, permite somente TRUE e FALSE.

SELECT

SELECT define uma coleção nomeada (o tipo-base) de tipos de dados (os tipos de dados subjacentes) (RANDO; McCABE 1996). A figura 11 ilustra o uso de SELECT, onde uma montagem de parede (`wall_mounting`) fixa um produto (`product`) a uma parede (`wall`) usando um método de fixação permanente (`glue` ou `weld`) ou temporário (`nail` ou `screw`).

```
TYPE attachment_method = SELECT(permanent_attachment,
                                temporary_attachment);
END_TYPE;
TYPE permanent_attachment = SELECT(glue, weld);
END_TYPE;
TYPE temporary_attachment = SELECT(nail, screw);
END_TYPE;
ENTITY nail;
  body_length : REAL;
  head_area   : REAL;
END_ENTITY;
ENTITY screw;
  body_length : REAL;
  pitch       : REAL;
END_ENTITY;
ENTITY glue;
  composition : material_composition;
  solvent     : material_composition;
END_ENTITY;
ENTITY weld;
  composition : material_composition;
END_ENTITY;
ENTITY wall_mounting;
  mounting : product;
  on       : wall;
  using    : attachment_method;
END_ENTITY;
```

Figura 11- Exemplo de uso do tipo SELECT (ISO 10303-11 1994)

Tipos agregados

Em STEP, *agregado* é o nome de coleções fortemente tipificadas. Geralmente, estas coleções apresentam o requisito de admitirem tamanhos variados. Por esta razão, não é prático implementar agregados fortemente tipificados em implementações de ligação tardia (RANDO; McCABE 1996), isto é, uma implementação que manipula qualquer modelo de dados EXPRESS, em contraste com uma implementação que faz a ligação cedo, ou seja, que é devotada a um modelo de dados EXPRESS específico.

Herança e tipo de dados entidade complexa

A herança é abordada em EXPRESS através do relacionamento entre entidades supertipo/subtipo. Algumas restrições de supertipo são usadas para restringir a combinação de tipos de entidades para compor uma instância válida de entidade: ONEOF (subtipos mutuamente exclusivos), AND (subtipos mutuamente inclusivos), e ANDOR (subtipos opcionalmente inclusivos). SCHENCK; WILSON (1994) comentam, sobre o rico modelo de herança em STEP:

> "EXPRESS permite criar relacionamentos entre supertipos e subtipos que não são possíveis com as linguagens objeto-orientadas corriqueiras, que tipicamente oferecem apenas hierarquias simples, com raiz única. EXPRESS permite raízes múltiplas (herança) e sobreposição dos nós na malha da hierarquia".

A figura 12 ilustra a herança múltipla em EXPRESS, com uma representação gráfica equivalente em EXPRESS-G, na figura 13.

```
ENTITY path
  SUPERTYPE OF (ONEOF(open_path, edge_loop, oriented_path))
  SUBTYPE OF (topological_representation_item);
  edge_list  : LIST [1:?] OF UNIQUE oriented_edge;
WHERE
  WR1: path_head_to_tail(SELF);
END_ENTITY;
ENTITY loop
  SUPERTYPE OF (ONEOF(vertex_loop, edge_loop, poly_loop))
  SUBTYPE OF (topological_representation_item);
END_ENTITY;
ENTITY edge_loop
  SUBTYPE OF (loop,path);
DERIVE
  ne : INTEGER := SIZEOF(SELF\path.edge_list);
WHERE
  WR1: (SELF\path.edge_list[1].edge_element.edge_start) :=:
       (SELF\path.edge_list[ne].edge_element.edge_end);
END_ENTITY;
```

Figura 12 - Herança múltipla em EXPRESS, em STEP Parte 42 (ISO 10303-42 1992)

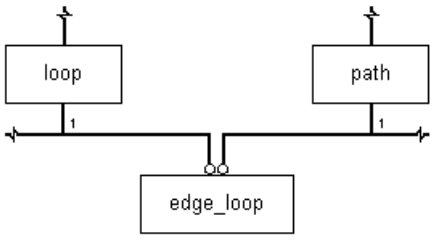


Figura 13 - Diagrama EXPRESS-G para as entidades na figura 12 (detalhe)

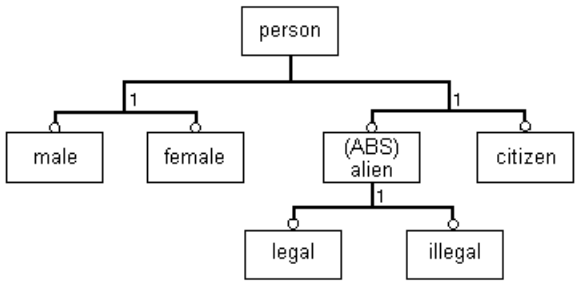
As entidades complexas são aquelas que não estão definidas explicitamente como entidades em um esquema EXPRESS, mas resultam da aplicação das restrições de supertipo AND e ANDOR. A figura 14 apresenta um esquema EXPRESS e o conjunto resultante de instâncias possíveis de entidades complexas.

Os supertipos abstratos são aqueles que não se pretendem instanciar diretamente, mas apenas em conjunção com pelo menos um de seus subtipos. Por exemplo, a entidade `alien` na figura 14, por causa da restrição ABSTRACT, pode ser instanciada apenas em conjunto com `legal` ou `illegal`.

(a)

```
SCHEMA example;
ENTITY person SUPERTYPE OF (ONEOF (male, female) AND
                            ONEOF(citizen, alien));
END_ENTITY;
ENTITY male SUBTYPE OF (person);
END_ENTITY;
ENTITY female SUBTYPE OF (person);
END_ENTITY;
ENTITY citizen SUBTYPE OF (person);
END_ENTITY;
ENTITY alien ABSTRACT SUPERTYPE OF (ONEOF(legal, illegal))
  SUBTYPE OF (person);
END_ENTITY;
ENTITY legal SUBTYPE OF (alien);
END_ENTITY;
ENTITY illegal SUBTYPE OF (alien);
END_ENTITY;
END_SCHEMA;
```

(b)



(c)

```
Allowable complex entity instantiations:
person&female&alien&illegal
person&female&alien&legal
person&female&citizen
person&male&alien&illegal
person&male&alien&legal
person&male&citizen
```

Figura 14 - (a) Esquema EXPRESS, (b) diagrama EXPRESS-G equivalente, e (c) entidades complexas possíveis. Adaptado de ISO 10303-11 (1994)

RANDO, McCABE (1996) assinalam que a estrutura supertipo/subtipo é um relacionamento de especialização, usado geralmente para expressar a abstração ISA

(“*is a*” ou “é um(a)”), mas pode também denotar o relacionamento PLAYS_ROLE_OF (“faz o papel de”). Por exemplo, observando a árvore de herança da figura 15 de baixo para cima, um estudante (`student`) é uma pessoa (`person`) e, na figura 16, um carro (`car`) é um veículo (`vehicle`). Porém, observando as árvores de cima para baixo, um veículo IS ALWAYS (sempre é) um carro, mas uma pessoa PLAYS_ROLE_OF (faz o papel de) estudante.

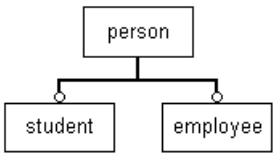


Figura 15 - Diagrama EXPRESS-G para a malha de entidades `person-student-employee`

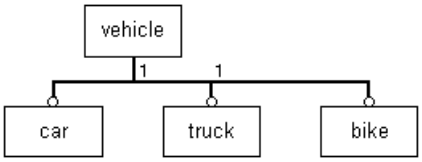


Figura 16 - Diagrama EXPRESS-G para a malha de entidades `vehicle-car-truck-bike`

Enquanto um carro nunca muda seu tipo após ser criado, instâncias de um relacionamento PLAYS_ROLE_OF, tais como `student` e `employee` na figura 15, podem mudar o tipo e assumir papéis múltiplos. A figura 17 ilustra a representação de conjuntos para a hierarquia de entidades da figura 15, com o surgimento do subtipo com papéis múltiplos `student-employee`, que não estava explícito na figura 15. A figura 18 apresenta uma representação de conjuntos para o relacionamento ISA da figura 16, com subtipos disjuntos e invariantes.

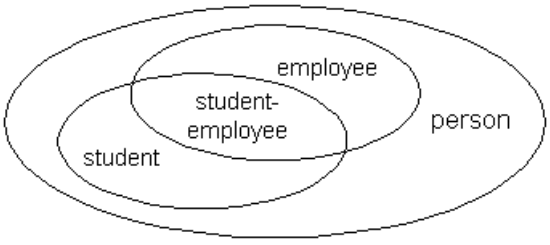


Figura 17 - Representação de conjuntos para as entidades na figura 15

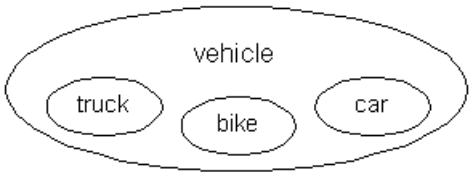


Figura 18 - Representação de conjuntos para as entidades na figura 16

Conforme RANDO; McCABE (1996), o relacionamento de especialização PLAYS_ROLE_OF é modelado em EXPRESS através do tipo de dados SELECT, que funciona como um “marcador de lugar” para outros tipos de dados. Uma instância do tipo SELECT pode mudar seu papel dinamicamente, mas apenas um papel por vez é suportado.

3.3 Considerações Finais sobre a Integração de Modelos de Dados de Produtos

Neste capítulo, o acoplamento de STEP e CORBA foi apresentado como base tecnológica para a integração de modelos de dados de produtos interoperáveis em rede heterogênea. Foram discutidas implicações deste acoplamento quanto a três áreas correspondentes às três subseções da revisão da literatura: tecnologia de gerência de banco de dados, interoperabilidade de aplicações, e representação de dados de produtos.

O acoplamento de STEP e CORBA não impõe qualquer escolha tecnológica quanto a gerência de banco de dados. Os bancos de dados STEP enfrentam os mesmos desafios e questões abertas que afligem qualquer banco de dados de engenharia. Os APs STEP oferecem os modelos conceituais para a representação e compartilhamento de dados entre sistemas de bancos de dados diversos, enquanto CORBA funciona como um amálgama que possibilita o acesso interoperável em rede a modelos de dados de produtos entre estes vários sistemas.

Quanto à interoperabilidade de aplicações, o uso de CORBA como tecnologia habilitadora é apoiado pela OMG e pelo padrão STEP através do mapeamento da SDAI para IDL. CORBA é adotada como solução para interoperabilidade de aplicações também no âmbito do projeto NIIIP. Os problemas e questões em aberto quanto à interoperabilidade de aplicações também não são particulares de CORBA.

A efetividade dos modelos de dados STEP para o acesso e compartilhamento em rede de modelos de dados de produtos depende da qualidade dos modelos de dados STEP (APs) e da qualidade da tradução dos modelos de dados de EXPRESS para IDL. Existe pesquisa em andamento para melhorar a qualidade dos modelos de dados STEP, especialmente no que diz respeito à interoperabilidade de APs, engenharia ontológica, melhoramentos na linguagem EXPRESS, e uma reconcepção do processo de construção de APs através da interpretação e reuso dos IRs.

Os modelos de dados escritos em EXPRESS são traduzidos para IDL com o propósito de construir as interfaces para o acesso interoperável em rede, de acordo com o mapeamento SDAI-IDL (ISO 10303-26 1997), que define como cada tipo de

dados EXPRESS é traduzido para IDL. Para que seja possível avaliar a traduzibilidade dos modelos de dados STEP para IDL, e a manutenibilidade da semântica dos modelos de dados através de uma rede distribuída, o próximo capítulo apresenta uma análise da perda semântica na tradução EXPRESS-IDL. Uma série de traduções de modelos de dados STEP é proposta, em que cada modelo enfoca um aspecto dos tipos de dados EXPRESS.

CAPÍTULO 4

ANÁLISE DA PERDA SEMÂNTICA NA TRADUÇÃO DE MODELOS DE DADOS DE PRODUTOS PARA ACESSO INTEROPERÁVEL EM REDE

Neste capítulo apresenta-se uma análise da perda semântica na tradução de modelos de dados de produtos STEP, escritos em EXPRESS, para a linguagem de definição de interface (IDL), da arquitetura CORBA, visando ao acesso interoperável em rede. As perdas são detectadas pela comparação das sintaxes das duas linguagens, e por uma avaliação *ad hoc* das possíveis perdas de significado (semântica) quando um objeto STEP é acessado através de sua interface IDL. Esta situação é típica do compartilhamento de dados de produtos baseado em padrões em empresas virtuais, onde todos os parceiros usam STEP para a representação de dados de produtos, e CORBA para o acesso interoperável em rede.

Os dados de produtos não são exatamente *traduzidos para* IDL, mas *acessados através de* interfaces IDL. A possibilidade de perda semântica é um problema diferente da perda de informação na tradução entre STEP de e para um modelo de dados específico de um sistema CAx. No último caso, os dados devem *migrar* para um modelo de dados *diferente*. No acesso em rede, a perda acontece apenas se a interface IDL não for capaz de transportar o objeto STEP acessado com todos os detalhes, ou se for permitido que dados espúrios sejam entregues. A análise apresentada neste capítulo é dirigida pela sintaxe: modelos de dados EXPRESS são comparados a suas interfaces IDL, e a perda é detectada nos casos em que há disparidade sintática entre as duas linguagens.

SANDERSON (1994) introduziu o modelo abstrato de dados (ADM), um metamodelo teórico de linguagens de modelagem de dados que apresenta as diferentes estruturas sintáticas que podem ocorrer em qualquer linguagem. A tabela 5 mostra as estruturas EXPRESS correspondentes às estruturas genéricas descritas pelo ADM, oferecendo uma visão abstrata ou genérica de EXPRESS. Estas estruturas foram analisadas quanto à perda semântica na tradução para IDL, visando à

construção de interfaces para o acesso interoperável. Para realizar a análise, uma *suite* ou série de traduções EXPRESS-IDL é apresentada na próxima seção.

Uma série de 17 esquemas (modelos de dados) STEP foi produzida, cada esquema enfocando um aspecto dos tipos de dados EXPRESS, como mostra a figura 19. Em cada caso analisado, uma "taxionomia da similaridade das estruturas de dados" é referida, conforme apresentaram BATINI *et al.* (1986) *apud* SANDERSON (1994):

Tabela 5 - Características de uma linguagem de modelagem genérica de acordo com o ADM, e as estruturas EXPRESS correspondentes

Estruturas ADM	Estruturas EXPRESS equivalentes	Observações
Construtoras	Entity, Select, Enumeration, Subtype-Supertype	Veja os casos t08, t09, t10, e t16.
De agrupamento	Array, Set, Bag, List	Veja os casos t05, t06, t07, t13, e t15.
De herança	Type-Subtype	Veja os casos t11, t12, e t17.
Primitivas	Binary, Integer, Logical, String	Veja os casos t01, t02, t03, t04, e t14.
Alfabeto	Conj.de caracteres EXPRESS	
Identificadores de objetos	Não especificados	

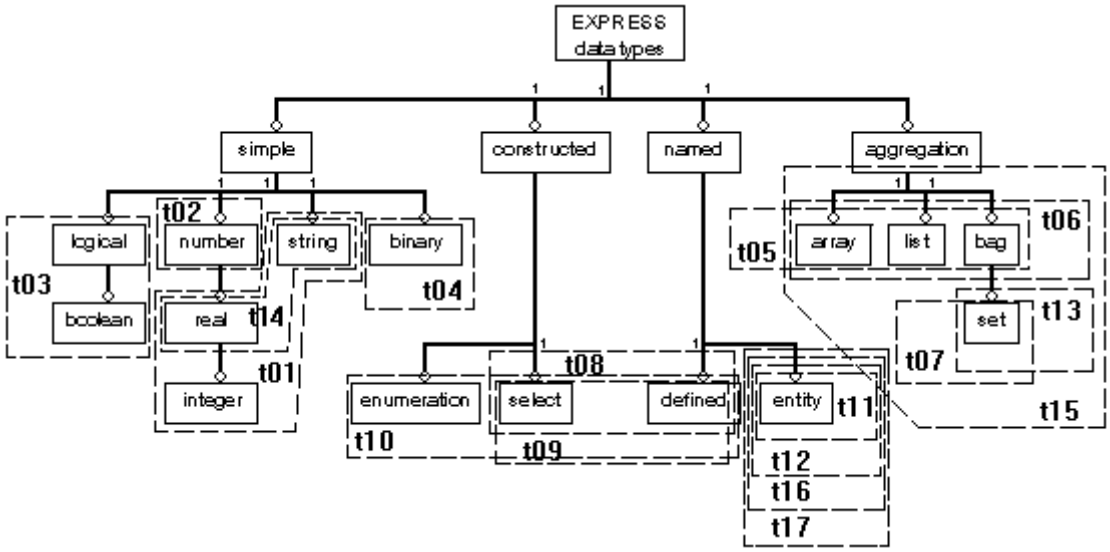


Figura 19 - Estruturas EXPRESS enfocadas nos 17 casos de tradução

- Idênticas – as estruturas são exatamente iguais, permitindo traduções sem perda;
- Equivalentes – as estruturas são baseadas em concepções equivalentes;
- Compatíveis – as concepções fundamentais dos dois modelos não estão em conflito, embora as estruturas não sejam equivalentes; e
- Incompatíveis – as concepções das duas linguagens estão em conflito, e tentativas de tradução levarão a perdas estruturais (sintáticas) e também, portanto, perdas semânticas.

Para que os 17 casos que compõem a *suite* de traduções fossem verossímeis, os esquemas STEP usados foram construídos a partir de partes de APs e IRs reais, obtidos do SC4 On Line Information Service at NIST (SOLIS 1996). Os APs e IRs foram esquadrinhados em busca de estruturas EXPRESS, de modo a representar na *suite* a riqueza de detalhes dos tipos de dados EXPRESS. A razão para este procedimento é o fato de que os APs são os esquemas a serem implementados em conjunto com SDAI ou outro método de implementação STEP, enquanto que os IRs são os modelos da informação genéricos que contém as estruturas básicas interpretadas nos (ou reusadas pelos) APs. Portanto, a escolha de um esquema EXPRESS adaptado de um AP leva a uma tradução de um esquema real, que corresponde a uma aplicação real. Alguns dos 17 casos da *suite* não foram extraídos de APs, mas de IRs, já que estes contém as estruturas que poderão ser usadas em futuros APs. O apêndice lista vários APs que estão em desenvolvimento.

Ao examinar os APs e IRs disponíveis, verificou-se que nem todos os tipos de dados EXPRESS estavam representados. Para suprir a falta, alguns exemplos contidos na especificação de SDAI (ISO 10303-22 1995) e EXPRESS, STEP Parte 11 (ISO 10303-11 1994), foram adaptados para construir os esquemas necessários. A tabela 6 lista, para cada um destes esquemas, o documento do padrão STEP a partir do qual foi adaptado e as entidades correspondentes, para que o leitor interessado possa reconstruir o esquema. Por exemplo, na tabela 6, a entidade `e0` do esquema t01 foi adaptada a partir de `ENTITY cartesian_transformation_operator`, que faz parte do AP 214.

Algumas estruturas EXPRESS tiveram mais do que um esquema construído para tradução, nos casos em que os documentos STEP mostram detalhes diferentes de uso da estrutura. Por exemplo, o caso t15 trata de tipos de dados agregados limitados `ARRAY`, `LIST`, e `BAG`, que também estão representados no caso t05. No entanto, o caso t15 descreve agregados com restrições (`OPTIONAL` e `UNIQUE`) que não estão presentes no caso t05.

Tabela 6 - Adaptações de documentos STEP para construir os casos de tradução (continua...)

Caso de Tradução	Adaptado a partir de	Estruturas presentes no esquema STEP original	Estruturas correspondentes no esquema gerado
t01 - REAL, INTEGER, e STRING	AP 214 (ISO 10303-214 1995)	cartesian_transformation_operator scale colour_rgb red curve_style name geometric_representation_context coordinate_space_dimension	e0 e0attr e1 e1attr e2 e2attr e3 e3attr
t02 - NUMBER	AP 214 (ISO 10303-214 1995)	sequential_method sequence_position	e0 e0attr
t03 - BOOLEAN e LOGIC	AP 201 (ISO 10303-201 1994)	b_spline_curve closed_curve camera_model_d2 view_window_clipping	e0 e0attr e1 e1attr
t04 - BINARY	Inventado – não há BINARY nos esquemas STEP.		
t05 - ARRAY, BAG, e LIST	AP 214 (ISO 10303-214 1995)	composite_curve segments composite_curve_segment using_curves curve_element_end_offset offset_vector rectangular_composite_surface segments surface_patch using_surfaces	e1 e1attr e2 e2attr e3 e3attr e6 e6attr e7 e7attr
t06 - ARRAY, BAG, e LIST sem limites	Inventado – não há agregados sem limites nos esquemas STEP.		
t07 - SET	AP 201 (ISO 10303-201 1994)	annotation_text composite_text collected_text trimmed_curve trim_1	e1 e3 e3attr e6 e6attr
t08 - SELECT	AP 201 (ISO 10303-201 1994)	geometric_representation_item curve point geometric_set offset_curve_2d geometric_set_select (TYPE)	ea0 eb0 eb1 eb2 ec0 sel

Tabela 6 (continuação)

t09 - SELECTs aninhados	Especificação EXPRESS, Exemplo 36 (ISO 10303-11 1994)	attachment_method (TYPE) permanent_attachment (TYPE) temporary_attachment (TYPE) nail screw glue weld wall_mounting	sel0 sel1 sel2 e0 e1 e2 e3 e4
t10 - ENUMERATION	AP 201 (ISO 10303-201 1994)	b_spline_curve b_spline_curve_form	e0 t_enum
t11 - tipo de dados entidade complexa	Especificação SDAI, Exemplo 8 (ISO 10303-22 1995)	a b c d e f g h	a0 b0 b1 c0 d0 d1 d2 d3
t12 - tipo de dados entidade complexa	AP 203 (ISO 10303-203 1994)	action_assignment change start_work	e0 e1 e2
t13 - SET sem limites	AP 204 (ISO 10303-204 1996)	representation representation_context	e0 e1
t14 - REAL e STRING	Inventado a partir de definições no doc. STEP Parte 11, 8.1.6 (ISO 10303-11 1994)		
t15 - ARRAY, LIST, e BAG	Inventado a partir de definições no doc. STEP Parte 11 (ISO 10303-11 1994)		
t16 - atributos de entidades	Inventado a partir de definições no doc. STEP Parte 11, 9.1 e 9.2 (ISO 10303-11 1994)		
t17 - herança múltipla	IR 42 (ISO 10303-42 1992)	path loop edge_loop	e0 e1 e2

A tradução de EXPRESS para IDL foi feita de acordo com o documento STEP Parte 26, mapeamento de IDL para SDAI (ISO 10303-26 1997). Neste mapeamento, apenas o aspecto estático dos modelos de dados STEP é considerado. O mapeamento de aspectos dinâmicos, tais como `PROCEDURE`s e `FUNCTION`s, não é parte de SDAI, da mesma forma que questões de gerência de banco de dados como transações, versões e controle de concorrência.

A tradução foi feita em parte manualmente e em parte usando o tradutor *exp2idl* do *toolkit* ST-Developer da STEP Tools. No momento da tradução, apenas parte dos tipos de dados EXPRESS é traduzível usando *exp2idl*. O código IDL produzido na tradução foi compilado usando o compilador IDL Orbix, da Iona, para verificação da sintaxe. Todas as traduções e compilações foram executadas em uma Sparcstation 10 da Sun, no Manufacturing Engineering Laboratory (MEL) do National Institute of Standards and Technology (NIST).

4.1 Uma *suite* de traduções EXPRESS-IDL

Esta seção apresenta os 17 casos da *suite* de traduções. Cada um consiste de um esquema EXPRESS e sua tradução para IDL, e uma análise da perda semântica ocorrida na tradução.

4.1.1 Tradução dos tipos de dados REAL, INTEGER, e STRING

O propósito deste caso de tradução é investigar a perda quanto aos tipos de dados REAL, INTEGER, e STRING. A figura 20 apresenta o esquema EXPRESS t01, seguido por sua tradução para IDL, na figura 21.

```
(* t01.exp

     Purpose of the schema: to represent simple data types
     REAL, INTEGER, and STRING.
*)

SCHEMA t01;

 ENTITY e0;
  e0attr : OPTIONAL REAL;
 END_ENTITY;

 ENTITY e1;
  e1attr : REAL;
 END_ENTITY;

 ENTITY e2;
  e2attr : STRING;
 END_ENTITY;

 ENTITY e3;
  e3attr : INTEGER;
 END_ENTITY;

END_SCHEMA;
```

Figura 20 – Esquema EXPRESS t01

```
/* t01.idl
*/


  interface  E0 {
     attribute double e0attr;
  };

  interface  E1 {
     attribute double e1attr;
  };

  interface  E2 {
     attribute string e2attr;
  };

  interface  E3 {
     attribute long e3attr;
  };
```

Figura 21 – Tradução do esquema t01 para IDL

Comentários relativos ao caso de tradução t01

A tradução para IDL foi produzida usando o tradutor *exp2idl*. Esta tradução automática gerou as interfaces representadas na figura 21. As interfaces foram geradas em arquivos diferentes referenciados (incluídos) em um arquivo chamado t01.idl. As interfaces foram reunidas manualmente na íntegra na figura 21 por simplicidade de apresentação. A sintaxe do código IDL resultante, compatível com o mapeamento definido no documento STEP Parte 26 (ISO 10303-26 1997), foi verificada com sucesso usando Orbix.

Perdas observadas no caso t01

As estruturas IDL da figura 21 são as interfaces a serem publicadas e disponibilizadas através de *object request brokers* para permitir o acesso interoperável em rede às estruturas EXPRESS correspondentes. Cada interface deve prover as estruturas apropriadas para representar todos os detalhes dos atributos das entidades.

No caso t01, as construções IDL `double`, `string`, e `long` permitem representar as estruturas EXPRESS correspondentes `REAL`, `STRING`, e `INTEGER`. Até este ponto, nada pode ser afirmado sobre quão adequadas são as estruturas IDL para a representação de suas estruturas correspondentes EXPRESS. Entretanto, pode-se notar que a sintaxe do código IDL se assemelha à das estruturas `double`, `string`, e `long` em C++. Estas estruturas são representações adequadas para

intervalo e tipo dos valores EXPRESS `REAL`, `STRING`, e `INTEGER`. Mas, IDL não é uma linguagem de implementação; pode ser implementada usando C, C++, ou outra linguagem de implementação para a qual um mapeamento de IDL estiver definido. A compatibilidade da tradução de `REAL`, `STRING`, e `INTEGER` deve ser examinada considerando a linguagem-cliente escolhida para implementação.

A perda semântica que *pode* ser observada é aquela intrínseca ao mapeamento EXPRESS-IDL, no caso t01, a falta de um equivalente em IDL para a diretiva `OPTIONAL` do atributo `e0attr` da entidade `e0`. `OPTIONAL` significa que uma instância do atributo em questão pode não ter valor. Esta característica é ignorada na tradução segundo o mapeamento IDL-EXPRESS (SDAI) (ISO 10303-26 1997). Portanto, um atributo `OPTIONAL REAL` que pertence a uma ocorrência de entidade EXPRESS disponibilizada para acesso interoperável em rede será acessado apenas como `REAL`. O raciocínio vale para atributos `REAL` ou `STRING`.

Quando uma ocorrência cujo valor não existe de um atributo `OPTIONAL REAL` é acessada como `REAL`, o resultado (o valor atribuído à ocorrência do atributo) é imprevisível, o que é um problema sério. Uma solução *ad hoc* possível para remediar esta perda é implementar objetos que transmitem um valor *flag*, isto é, um valor fora do intervalo previsto, que representa ou convenciona que esta ocorrência não existe. Esta solução pode funcionar se todos os clientes entendem esta convenção.

Em resumo, EXPRESS e IDL são *incompatíveis* quanto à característica `OPTIONAL REAL`. Outros tipos de dados com a diretiva `OPTIONAL` poderão gerar problemas similares se forem utilizados, no futuro, nos esquemas STEP. Com respeito às demais estruturas presentes no caso t01, EXPRESS e IDL são *equivalentes*.

4.1.2 Tradução do tipo de dados NUMBER

O propósito deste caso de tradução é investigar a perda semântica envolvendo o tipo simples de dados NUMBER. A figura 22 mostra o esquema EXPRESS t02, seguido por sua tradução para IDL, na figura 23.

Comentários relativos ao caso de tradução t02

Esta tradução para IDL foi produzida parcialmente através do tradutor *exp2idl*, que gerou a interface `E0`, sem os blocos `enum` e `union`, em um arquivo independente. A interface `E0` foi incluída manualmente no arquivo t02.idl apresentado na figura 23 para simplificar a apresentação. Os blocos `enum` e `union` também foram incluídos manualmente em t02.idl para que a tradução seja compatível com a especificação do

mapeamento IDL para SDAI (ISO 10303-26 1997). A sintaxe do código IDL produzido foi verificada com sucesso usando Orbix.

```
(* t02.exp

     Purpose of the schema: to represent simple data type
     NUMBER.

*)

SCHEMA t02;

 ENTITY e0;
  e0attr : NUMBER;
 END_ENTITY;

END_SCHEMA;
```

Figura 22 – Esquema EXPRESS t02

```
/* t02.idl
*/

  interface  E0 {
     enum Number_discriminant { Number_discriminant__long,
     Number_discriminant__double };

     union Number switch ( Number_discriminant ) {
     case Number_discriminant__long : long c1;
     case Number_discriminant__double : double c2;
     };

     attribute Number e0attr;
  };
```

Figura 23 – Tradução do esquema t02 para IDL

Perdas observadas no caso t02

O tipo de dados EXPRESS `NUMBER`, enfocado no caso t02, refere-se a atributos que podem ser `REAL` ou `INTEGER`. É usado quando “uma representação mais específica não é importante” (ISO 10303-11 1994). Qualquer *ocorrência* de um atributo `NUMBER`, entretanto, será implementada no *formato* `REAL` ou `INTEGER`. Neste sentido, o código IDL `Number` como `switch` (comutador) entre os tipos `double` e `long` é adequado

para representar a estrutura EXPRESS `NUMBER`, que pode ser `REAL` ou `INTEGER`. Portanto, não há perda semântica na tradução EXPRESS-IDL quanto ao tipo de dados `NUMBER`. Em resumo, EXPRESS e IDL são *compatíveis* quanto a `NUMBER`, isto é, suas estruturas não são equivalentes, mas também não estão em conflito.

4.1.3 Tradução dos tipos de dados BOOLEAN e LOGIC

O propósito deste caso de tradução é investigar a perda semântica relativa aos tipos de dados BOOLEAN e LOGIC. A figura 24 apresenta o esquema t03 seguido por sua tradução para IDL, na figura 25.

```
(* t03.exp
     Purpose of the schema: to represent simple data types
     BOOLEAN and LOGIC.
*)

SCHEMA t03;

  ENTITY e0;
      e0attr : LOGICAL;
  END_ENTITY;

  ENTITY e1;
      e1attr : BOOLEAN;
  END_ENTITY;

END_SCHEMA;
```

Figura 24 – Esquema EXPRESS t03

```
/* t03.idl
*/

  enum Logical { LFalse, LTrue, LUnset, LUnknown };

  enum Boolean { BFalse, BTrue, BUnset };

  interface  E0 {
     attribute Logical e0attr;
  };

  interface  E1 {
     attribute Boolean e1attr;
  };
```

Figura 25 – Tradução do esquema t03 para IDL

Comentários relativos ao caso de tradução t03

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 25 em arquivos diferentes, que foram incluídos manualmente no arquivo t03.idl para simplificar a apresentação. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

Perdas observadas no caso t03

O mapeamento IDL mostrado na figura 25 tem interfaces com atributos `Logical` e `Boolean`, similares aos atributos `LOGICAL` e `BOOLEAN` das entidades EXPRESS da figura 24. Os atributos IDL são construídos na forma de enumerações dos valores possíveis para os atributos das entidades, com a diferença que, em IDL, os atributos podem ser unset (`LUnset` ou `BUnset` -- valor não atribuído). Todavia, qualquer ocorrência de atributo acessada através de uma interface IDL deve possuir um valor válido, isto é, `LOGICAL` deve ser `TRUE`, `FALSE`, ou `UNKNOWN`; e `BOOLEAN` deve ser `TRUE` ou `FALSE`. Se o acesso não contempla um valor válido, o resultado é imprevisível e há perda semântica (se o dado recebido não é válido, como instanciar o atributo?).

A verificação da validade do atributo `LOGICAL` ou `BOOLEAN` recebido, forçando o cancelamento do acesso no caso de valor inválido, deve prevenir o acontecimento desta perda semântica. Este procedimento poderia ser parte de um mecanismo de transação SDAI, o que não faz parte de STEP no estágio atual.

Em resumo, com respeito aos tipos de dados `LOGICAL` e `BOOLEAN`, EXPRESS e IDL são *incompatíveis* por causa dos valores `unset` em IDL. Isto, no entanto, pouco deve prejudicar o compartilhamento de dados de produtos. Não deve haver perda semântica, já que os acessos deverão resultar em valores válidos (compatíveis com o esquema STEP em foco) para os atributos `LOGICAL` ou `BOOLEAN`.

4.1.4 Tradução do tipo de dados BINARY

O propósito deste caso de tradução é investigar a perda semântica quanto ao tipo de dados BINARY. A figura 26 apresenta o esquema EXPRESS t04, seguido por sua tradução para IDL representada no arquivo t04.idl, ilustrado na figura 27.

Comentários relativos ao caso de tradução t04

A tradução para IDL produzida através do tradutor *exp2idl* gerou código que não está em perfeito acordo com o mapeamento de IDL para SDAI (ISO 10303-26 1997). A tradução automática foi incluída na figura 27 como comentário (limitado por “/*” e “*/”).

A tradução correta em IDL para `BINARY` é `Binary`, precedido por sua definição (`typedef`), já que não é tipo primitivo de IDL. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

```
(* t04.exp
     There is no SOLIS-available schema that includes a
     BINARY construct.

     Purpose of the schema: to represent simple data type
     BINARY.

     Produced according to Part 11, item 8.1.7 (Binary data
     type).
*)

SCHEMA t04;

 ENTITY e0;
  e0attr0 : BINARY;
  e0attr1 : BINARY (8);
  e0attr2 : BINARY (4) FIXED;
 END_ENTITY;

END_SCHEMA;
```

Figura 26 – Esquema EXPRESS t04

```
/* t04.idl
*/

typedef sequence<octet> Binary;

interface  E0 {

     attribute Binary e0attr0;
     attribute Binary e0attr1;
     attribute Binary e0attr2;

     /* Part 26 5.2.1.7, mapping of BINARY to IDL: "The name
     of the sequence shall be Binary."
     The following is exp2idl's mapping from EXPRESS. Binary
     size is declared directly with the attribute.

     attribute sequence<octet> e0attr0;
     attribute sequence<octet, 2> e0attr1;
     attribute sequence<octet, 1> e0attr2;
     */
};
```

Figura 27 – Tradução do esquema t04 para IDL

Perdas observadas no caso t04

As entidades EXPRESS que possuem atributos `BINARY`, no esquema da figura 26, foram traduzidas para IDL usando o tipo `Binary`. Entretanto, perde-se o *tamanho*, ou número máximo de bits (que pode ser especificado como um número entre parênteses, mas não é mandatório) de atributos `BINARY`, bem como a diretiva `FIXED`, que determina que o tamanho do atributo é fixo (limite mínimo e máximo).

O tipo de dados `BINARY` pode ser usado para especificar tamanho variável ilimitado (como o atributo `e0attr0` na figura 26), tamanho variável limitado (como o atributo `e0attr1` na figura 26 – limitado a 8 bits), ou tamanho fixo (como o atributo `e0attr2` na figura 26 – tamanho fixo de 8 bits). A interface IDL gerada a partir da tradução do esquema EXPRESS da figura 26 representa todos os `BINARY` da mesma forma, isto é, sem qualquer informação sobre tamanho e caráter fixo. Isto é adequado apenas para representar binários de tamanho variado.

A tradução automática, comentada na figura 27, é adequada para representar limites, mas também omite a diretiva `FIXED`. Ainda, o mapeamento padrão EXPRESS-IDL deixa claro que `Binary` deve ser uma predeclaração `typedef` (ISO 10303-26 1997), o que impossibilita a representação do tamanho. Portanto, há perda semântica na tradução de `BINARY` para IDL.

Em resumo, EXPRESS e IDL são *incompatíveis* quanto ao tipo de dados `BINARY`. Sempre que um `BINARY` de tamanho limitado for acessado através de uma interface IDL, será lido sem informação sobre tamanho. Uma forma possível de aliviar esta perda é incluir, na interface IDL, um campo inteiro informando o número de bits associado ao atributo `BINARY`. No compartilhamento baseado em padrões, entretanto, não se espera que esta perda semântica tenha efeito, já que tanto transmissor quanto receptor devem usar o mesmo esquema STEP, e a falta de informação sobre tamanho na interface IDL não impõe perda.

4.1.5 Tradução dos tipos de dados ARRAY, BAG, e LIST

O propósito deste caso de tradução é investigar a perda semântica quanto aos tipos de dados ARRAY, BAG, e LIST. A figura 28 mostra o esquema EXPRESS t05, seguido por sua tradução para IDL no arquivo t05.idl, ilustrado na figura 29.

Comentários relativos ao caso de tradução t05

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 29 em arquivos diferentes, que foram incluídos manualmente no arquivo t05.idl para simplificar a apresentação.

Devido à falta de conformidade da tradução automática com o padrão, a seguinte mudança foi introduzida manualmente: o tipo `double` foi substituído por `Double`, que é a tradução padronizada para `REAL`. A sintaxe do código IDL resultante foi verificada usando Orbix, apresentando erros: a tradução de atributo `INVERSE` deveria ser um atributo apenas para leitura (*read-only*) com o sufixo "`__list`", o que não é aceito por Orbix. A razão para este erro é desconhecida, aparentemente o compilador trata de uma versão incompleta de IDL.

```
(* t05.exp

     Purpose of the schema: to represent aggregates ARRAY,
     BAG, and LIST (all but SET).  Observation: BAG appears
     only in INVERSE attributes in the STEP schemata.

*)

SCHEMA t05;

 ENTITY e1;
  e1attr : LIST[1:?] OF e2;
 END_ENTITY;

 ENTITY e2;
  INVERSE
   e2attr : BAG[1:?] OF e1 FOR e1attr;
 END_ENTITY;

 ENTITY e3;
  e3attr : ARRAY[1:3] OF REAL;
 END_ENTITY;

 ENTITY e6;
  e6attr : LIST[1:?] OF LIST[1:?] OF e7;
 END_ENTITY;

 ENTITY e7;
  INVERSE
   e7attr : BAG[1:?] OF e6 FOR e6attr;
 END_ENTITY;

END_SCHEMA;
```

Figura 28 – Esquema EXPRESS t05

```
/* t05.idl
*/

interface  E2;
typedef sequence<E2> E2__bounded_list;

interface  E1;
typedef sequence<E1> E1__bounded_bag;

typedef sequence<Double> Double__bounded_array;

interface  E7;
typedef sequence<sequence<E7> > E7__bounded_list__bounded_list;

interface  E6;
typedef sequence<E6> E6__bounded_bag;

interface  E1  {
     attribute E2__bounded_list e1attr;
};

interface  E2  {
     readonly attribute E1__bounded_bag e2attr;
};

interface  E3  {
     attribute Double__bounded_array e3attr;
};

interface  E6  {
     attribute E7__bounded_list__bounded_list e6attr;
};

interface  E7  {
     readonly attribute E6__bounded_bag__list e7attr;

/*   The above translation, according to Part 26, 5.2.6,
     was not accepted by the Orbix idl compiler.
     Errors annotated:
     49:(semantic): Identifier `E6__bounded_bag__list' not found
     49:(semantic): Name does not denote a type
     Translation accepted by the compiler:

     readonly attribute E6__bounded_bag e7attr;
*/

};
```

Figura 29 – Tradução do esquema t05 para IDL

Perdas observadas no caso t05

As interfaces IDL resultantes da tradução dos tipos de dados agregados ARRAY, BAG, e LIST não contemplam todos os detalhes destas estruturas. Índices e limites de tamanho não são representados em IDL. Isto pode ser considerado um problema menor nos casos de BAG e LIST porque atributos IDL genéricos (isto é, sem índices ou limites) podem ser usados para interfacear a transmissão de agregados EXPRESS indexados e limitados (e, num intercâmbio baseado em padrões, os dois lados do processo conhecem o esquema EXPRESS que detalha estes índices e limites). Mas, no caso de ARRAY, não há atributo IDL genérico. Qualquer *array* é sempre indexado e limitado. Um cliente receptor de um *array* acessado através de uma interface IDL não recebe informação sobre qual deve ser seu índice-base (que, no exemplo da figura 28, é 1) ou tamanho.

Em resumo, EXPRESS e IDL são *incompatíveis* quanto aos agregados limitados ARRAY, BAG, e LIST. No caso de compartilhamento onde transmissor e receptor não usam o mesmo esquema STEP, isto acarreta perda semântica. Apenas um controle externo pode remediar a perda (por exemplo, a inclusão na interface de dois inteiros representando tamanho e índice-base, contanto que todos os clientes na rede compreendam esta convenção). No compartilhamento baseado em padrões, no entanto, isto não é problema, já que o esquema STEP em uso impõe os detalhes perdidos na tradução para IDL.

4.1.6 Tradução dos tipos de dados BAG e LIST não-limitados

O propósito deste caso de tradução é investigar a perda semântica quanto ao tipo de dados agregados BAG e LIST não-limitados. A figura 30 apresenta o esquema EXPRESS t06, seguido por sua tradução para IDL representada no arquivo t06.idl, ilustrado na figura 31.

Comentários relativos ao caso de tradução t06

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas no arquivo t06.idl em arquivos diferentes, que foram incluídos manualmente na figura 31 para simplificar a apresentação. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

O esquema EXPRESS t06.exp foi inventado (não foi extraído dos documentos STEP) para construir um caso de tradução contemplando agregados não-limitados, que não estão presentes nos esquemas STEP atuais, à exceção dos SETs não-limitados (vide caso t13). Um ARRAY não-limitado foi inadvertidamente incluído no

```
(* t06.exp

     Purpose of this schema: to represent unbounded
     agregates ARRAY, BAG and LIST.
*)

SCHEMA t06;

(* ENTITY e0;
     e0attr: ARRAY OF REAL;
 END_ENTITY;

  Unbounded ARRAYs do not belong in STEP syntax, therefore
  this entity was taken off the translation suite.
*)

 ENTITY e1;
  e1attr : BAG OF INTEGER;
 END_ENTITY;

 ENTITY e2;
  e2attr : LIST of INTEGER;
 END_ENTITY;

END_SCHEMA;
```

Figura 30 – Esquema EXPRESS t06

```
/* t06.idl
*/

typedef sequence< Long > Long__list;
typedef sequence< Long > Long__bag;

interface  e1 {
     attribute Long__list e1attr;
};

interface  e2 {
     attribute Long__bag e2attr;
};
```

Figura 31 – Tradução do esquema t06 para IDL

esquema, e *exp2idl* o traduziu. EXPRESS não contém ARRAYs não-limitados, portanto a entidade entity `e0` foi comentada (posta entre "`(*`" e "`*)`"), retirada do esquema. Também, *exp2idl* traduziu `INTEGER` para IDL como `long` (da mesma forma que `BAG`

of INTEGER e LIST of INTEGER como long__list e long__bag, respectivamente). Isto foi alterado manualmente para Long, Long__list, e Long__bag, para atender ao mapeamento padrão, que estipula que o tipo traduzido deve ter o primeiro caracter maiúsculo.

Perdas observadas no caso t06

Os agregados não-limitados BAG of INTEGER e LIST of INTEGER, mostrados na figura 30, são mapeados para IDL respectivamente como Long__list e Long__bag, que são definidos (typedef) na figura 31. Esta tradução representa todos os detalhes dos agregados não-limitados EXPRESS. Nenhuma perda está associada à tradução EXPRESS-IDL do caso t06. Em resumo, EXPRESS e IDL são *compatíveis* quanto aos agregados não-limitados BAG e LIST.

4.1.7 Tradução do tipo de dados SET

O propósito deste caso de tradução é investigar a perda semântica quanto ao tipo de dados SET. A figura 32 apresenta o esquema EXPRESS t07, seguido por sua tradução para IDL representada no arquivo t07.idl, ilustrado na figura 33.

Comentários relativos ao caso de tradução t07

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 33 em arquivos diferentes, que foram incluídos manualmente no arquivo t07.idl para simplificar a apresentação. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

O SET of REAL na entidade e6 da figura 32 foi traduzido por *exp2idl* como double__bounded_set. Para tornar a tradução compatível com o padrão, isto foi alterado para Double__bounded_set.

Perdas observadas no caso t07

O código IDL da figura 33 foi construído para representar SETs limitados. Porém, Double__bounded_set não permite a especificação de tamanho e índices. Isto representa um problema menor no compartilhamento baseado em padrões, já que os dois lados usam o mesmo esquema STEP e um Double__bounded_set (sem limites e índices) *pode* conduzir dados relativos a um SET limitado. Em resumo, EXPRESS e IDL são *incompatíveis* quanto a SETs limitados, mas esta disparidade não deve afetar a qualidade dos dados compartilhados usando STEP e CORBA, já que o esquema STEP em uso impõe os detalhes perdidos na tradução para IDL.

```
(* t07.exp

     Purpose of the schema: to represent (bound) aggregate SET.

*)

SCHEMA t07;

  ENTITY e1;
  END_ENTITY;

  ENTITY e3;
      e3attr : SET [2:?] OF e1;
  END_ENTITY;

  ENTITY e6;
      e6attr : SET [1:2] OF REAL;
  END_ENTITY;

END_SCHEMA;
```

Figura 32 – Esquema EXPRESS t07

```
/* t07.idl
*/

interface  E1;
typedef sequence<E1> E1__bounded_set;
typedef sequence< Double > Double__bounded_set;

interface  E1  {
};

interface  E3  {
     attribute E1__bounded_set e3attr;
};

interface  E6  {
     attribute Double__bounded_set e6attr;
};
```

Figura 33 – Tradução do esquema t07 para IDL

4.1.8 Tradução do tipo de dados SELECT

O propósito deste caso de tradução é investigar a perda semântica quanto ao tipo de dados SELECT, e também ilustra a tradução de tipos *definidos*, que fazem parte dos

blocos TYPE. A figura 34 apresenta o esquema EXPRESS t08, seguido por sua tradução para IDL representada no arquivo t08.idl, ilustrado na figura 35.

```
(* EXPRESS schema t08.exp

     Purpose of the schema: to represent SELECT.
     Observation: Named (TYPE) types, inheritance, and a SET
     of selects are also represented.

          ea0
          |1
     ------------
     |    |    |
     eb0  eb1  eb2   * eb2 is an entity whith an attribute
     |               * which is a select set (SET [1:?] of sel)
     |               * of eb0 and eb1.
     ec0             * (Adapted from AP 201)

*)

SCHEMA t08;

  TYPE sel = SELECT
    (eb0,
     eb1);
  END_TYPE;

  ENTITY ea0
    SUPERTYPE OF (ONEOF (eb0,eb1,eb2));
  END_ENTITY;

  ENTITY eb0
    SUPERTYPE OF (ec0)
    SUBTYPE OF (ea0);
  END_ENTITY;

  ENTITY eb1
    SUBTYPE OF (ea0);
  END_ENTITY;

  ENTITY eb2
    SUBTYPE OF (ea0);
      a0attr : SET [1:?] OF sel;
  END_ENTITY;

  ENTITY ec0
    SUBTYPE OF (eb0);
  END_ENTITY;

END_SCHEMA;
```

Figura 34 - Esquema EXPRESS t08

```
/* t08.idl
*/


interface  Eb0;
interface  Eb1;
enum Sel_select {
     Sel__Eb0,
     Sel__Eb1};
union Sel switch (Sel_select) {
case Sel__Eb0 : Eb0 c1;
case Sel__Eb1 : Eb1 c2;
};

interface  Ea0  {
};

interface  Eb0  : Ea0 {
};

interface  Eb1  : Ea0 {
};

typedef  sequence< Sel > Sel__bounded_set;
interface  Eb2  : Ea0 {
     attribute Sel__bounded_set a0attr;
};

interface  Ec0  : Eb0 {
};
```

Figura 35 - Tradução do esquema t08 para IDL

Comentários relativos ao caso de tradução t08

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 35 em arquivos diferentes, que foram incluídos manualmente no arquivo t08.idl para simplificar a apresentação. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

A tradução com *exp2idl* produziu nomes de variáveis incompatíveis com a especificação-padrão. O nome `select_sel` foi gerado em ordem errada, e foi alterado para `Sel_select`. Também, outros nomes de variáveis (`sel__eb0`, `sel__eb1`, `sel`, e `sel__bounded_set`) tiveram que ser alterados manualmente para que a capitalização fosse compatível com o mapeamento padrão EXPRESS-IDL (ISO 10303-26 1997).

Perdas observadas no caso t08

Os SELECTs EXPRESS são representados como uma escolha (switch) dentre uma série de valores enumerados (enum) de *tipos* diferentes, como mostram as figuras 48 e 49. Esta é uma representação apropriada, uma vez que é exatamente o que é um SELECT: um atributo que pode assumir um valor cujo domínio está numa lista de diferentes tipos. Ainda, esta representação IDL não pode ser confundida com qualquer outra tradução de uma estrutura EXPRESS. Portanto, não há perda ou ambigüidade associada à tradução de SELECT para IDL. Em resumo, IDL é *equivalente* a EXPRESS com respeito à estrutura SELECT.

4.1.9 Tradução de SELECTs aninhados

O propósito deste caso de tradução é investigar a perda semântica quanto a SELECTs aninhados. A figura 36 apresenta o esquema EXPRESS t09, seguido por sua tradução para IDL representada no arquivo t09.idl, ilustrado na figura 37.

Comentários relativos ao caso de tradução t09

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces, com pequenas alterações, mostradas na figura 37 em arquivos diferentes, que foram incluídos manualmente no arquivo t09.idl para simplificar a apresentação. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

Assim como em casos anteriores, algumas alterações de menor importância foram feitas no código IDL para fazê-lo compatível com a tradução padronizada: select_sel0, select_sel1, e select_sel2 foram substituídos por Sel0_select, Sel1_select, e Sel2_select, respectivamente. A primeira letra teve a capitalização corrigida, tornando-se maiúscula.

Perdas observadas no caso t09

Este caso difere de t08 por tratar de SELECTs *aninhados*, isto é, SELECTs de SELECTs. Como em t08, a tradução de SELECT é uma escolha (switch) dentre uma série de valores enumerados (enum) de tipos diferentes, conforme as figuras 50 e 51. Nos casos onde estes valores provém igualmente de SELECTs, a tradução padronizada é implementada como uma escolha (switch) dentre valores enumerados (enum) de diferentes escolhas (switch) dentre valores enumerados (enum) de tipos diferentes.

```
(* t09.exp

     Purpose of the schema: to represent nested SELECT.
     Adapted from Part 11, Example 36.

*)

SCHEMA t09;

  TYPE sel0 = SELECT
    (sel1,
     sel2);
  END_TYPE;

  TYPE sel1 = SELECT
    (e0,
     e1);
  END_TYPE;

  TYPE sel2 = SELECT
    (e2,
     e3);
  END_TYPE;

  ENTITY e0;
  END_ENTITY;

  ENTITY e1;
  END_ENTITY;

  ENTITY e2;
  END_ENTITY;

  ENTITY e3;
  END_ENTITY;

  ENTITY e4;
    e4attr : sel0;
  END_ENTITY;

END_SCHEMA;
```

Figura 36 - Esquema EXPRESS t09

Em outras palavras, é possível estender o princípio de escolha dentre valores enumerados, característico da tradução de SELECTs em IDL. O código IDL traduzido, além disso, não tem outro possível correspondente em EXPRESS. Portanto, uma entidade STEP contendo SELECTs pode ser acessada em uma rede heterogênea interoperável sem que haja ambigüidade ou perda. Em resumo, EXPRESS e IDL são *equivalentes* com respeito a SELECTs aninhados.

```
/*t09.idl
*/

interface  E0;
interface  E1;
enum Sel1_select {
     Sel1__E0,
     Sel1__E1};
union Sel1 switch (Sel1_select) {
case Sel1__E0 : E0 c1;
case Sel1__E1 : E1 c2;
};

interface  E2;
interface  E3;
enum Sel2_select {
     Sel2__E2,
     Sel2__E3};
union Sel2 switch(Sel2_select) {
case Sel2__E2 : E2 c1;
case Sel2__E3 : E3 c2;
};

enum Sel0_select {
     Sel0__Sel1,
     Sel0__Sel2};
union Sel0 switch (Sel0_select) {
case Sel0__Sel1 : Sel1 c1;
case Sel0__Sel2 : Sel2 c2;
};

interface  E0  {
};

interface  E1  {
};

interface  E2  {
};

interface  E3  {
};

interface  E4  {
     attribute Sel0 e4attr;
};
```

Figura 37 - Tradução do esquema t09 para IDL

4.1.10 Tradução do tipo de dados ENUMERATION

O propósito deste caso de tradução é investigar a perda semântica quanto ao tipo de dados ENUMERATION. A figura 38 apresenta o esquema EXPRESS t10, seguido por sua tradução para IDL representada no arquivo t10.idl, ilustrado na figura 39.

```
(* t10.exp

     Purpose of the schema: to represent ENUMERATION.

*)


SCHEMA t10;

  TYPE t_enum = ENUMERATION OF
     (te0,
     te1,
     te2);
  END_TYPE;

  ENTITY e0;
      e0attr    : t_enum;
  END_ENTITY;

END_SCHEMA;
```

Figura 38 - Esquema EXPRESS t10

```
/* t10.idl
*/


enum T_enum {
     T_enum__te0,
     T_enum__te1,
     T_enum__te2,
     T_enum__unset
};

interface  E0  {
     attribute T_enum e0attr;
};
```

Figura 39 - Tradução do esquema t10 para IDL

Comentários relativos ao caso de tradução t10

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática `produziu o arquivo t10.idl com a interface `E0` da figura 39 em um arquivo diferente, que foi incluído manualmente em t10.idl para simplificar a apresentação. O identificador de enumerador `_unset`, requerido pela tradução padronizada, não foi gerado por *exp2idl*, sendo incluído manualmente. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

Perdas observadas no caso t10

As estruturas EXPRESS `ENUMERATION` na figura 38 foram mapeadas em IDL `enum`, com sintaxe semelhante. Esta tradução representa bem a estrutura sintática do original EXPRESS. A única disparidade existente é a permissão de uma instanciação `unset` (sem valor assumido) em IDL. No compartilhamento baseado em padrões, no entanto, isto não representa problema, pois o mesmo esquema EXPRESS é usado dos dois lados, e os dados devem ser compatíveis com o esquema, que não permite `unset`.

Em resumo, EXPRESS e IDL são *incompatíveis* com respeito a `ENUMERATION` por causa da alternativa `unset` em IDL. Ainda assim, nenhuma perda semântica é esperada, já que qualquer acesso baseado em padrões deve conduzir um valor válido (jamais `unset`) para um atributo do tipo `ENUMERATION`.

4.1.11 Tradução de tipos de dados entidades complexas

O propósito deste caso de tradução é investigar a perda semântica quanto ao tipo de dados entidade complexa, no qual há herança envolvida. A figura 40 apresenta o esquema EXPRESS t11, seguido por sua tradução para IDL representada no arquivo t11.idl, ilustrado na figura 41.

Comentários relativos ao caso de tradução t11

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 41 em arquivos diferentes, que foram incluídos manualmente no arquivo t11.idl para simplificar a apresentação. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

Todas as interfaces que representam entidades complexas (aquelas após o comentário `/* complex entity data types */` na figura 41) foram incluídas manualmente em t11.idl de acordo com a especificação SDAI (ISO 10303-22 1995), pois sua geração não foi feita por *exp2idl*. No bloco COMPLEX da figura 40, todas as

instanciações possíveis de entidades complexas foram incluídas e então traduzidas de acordo com o mapeamento padrão EXPRESS-IDL (ISO 10303-26 1997).

```
(* t11.exp

     Purpose of the schema: to represent complex entity data
     types.

           (ABS) a0
            ANDOR
           /     \
          /       \
        b0         b1
         |
         |
        c0
        AND
   ONEOF   ANDOR
  /    |   |    \
 /     |   |     \
d0     d1  d2    d3

*)


SCHEMA t11;
 ENTITY a0 ABSTRACT SUPERTYPE OF ( b0 ANDOR b1 ); END_ENTITY;
 ENTITY b0 SUBTYPE OF (a0); END_ENTITY;
 ENTITY b1 SUBTYPE OF (a0); END_ENTITY;
 ENTITY c0 SUPERTYPE OF (ONEOF (d0, d1) AND (d2 ANDOR d3))
  SUBTYPE OF (b0);
 END_ENTITY;
 ENTITY d0 SUBTYPE OF (c0); END_ENTITY;
 ENTITY d1 SUBTYPE OF (c0); END_ENTITY;
 ENTITY d2 SUBTYPE OF (c0); END_ENTITY;
 ENTITY d3 SUBTYPE OF (c0); END_ENTITY;

 (* According to Part 26 (N396, 5.2.7, Jan 18,96), the
 "COMPLEX" block below should be added.  However, no
 other details are given, and no other reference is made
 to this block in any STEP document.  The COMPLEX block
 is only illustrated here (it is commented out of the
 schema).
```

Figura 40 - Esquema EXPRESS t11 (continua...)

```
 COMPLEX;
 b0+b1;
 b1+c0;
 d0+d2;
 d0+d3;
 d0+d2+d3;
 d1+d2;
 d1+d3;
 d1+d2+d3;
 b1+d0+d2;
 b1+d0+d3;
 b1+d0+d2+d3;
 b1+d1+d2;
 b1+d1+d3;
 b1+d1+d2+d3;
 END_COMPLEX;
 *)

END_SCHEMA;
```

Figura 40 (continuação)

```
/* t11.idl
*/

interface  A0  {
};

interface  B0  : A0 {
};

interface  B1  : A0 {
};

interface  C0  : B0 {
};

interface  D0  : C0 {
};

interface  D1  : C0 {
};

interface  D2  : C0 {
};

interface  D3  : C0 {
};
```

Figura 41 - Tradução do esquema t11 para IDL (continua...)

```
/* complex entity data types */

interface B0B1  : B0, B1 {
};

interface B1C0  : B1, C0 {
};

interface D0D2  : D0, D2 {
};

interface D0D3  : D0, D3 {
};

interface D0D2D3  : D0, D2, D3 {
};

interface D1D2  : D1, D2 {
};

interface D1D3  : D1, D3 {
};

interface D1D2D3  : D1, D2, D3 {
};

interface B1D0D2  : B1, D0, D2 {
};

interface B1D0D3  : B1, D0, D3 {
};

interface B1D0D2D3  : B1, D0, D2, D3 {
};

interface B1D1D2  : B1, D1, D2 {
};

interface B1D1D3  : B1, D1, D3 {
};

interface B1D1D2D3  : B1, D1, D2, D3 {
};
```

Figura 41 (continuação)

Em resumo, uma malha de hierarquia de entidades EXPRESS foi traduzida para interfaces IDL. Cada entidade foi mapeada como uma interface, bem como cada entidade complexa (combinação de duas ou mais entidades que são subtipos permitidos pelas restrições de hierarquia SUPER/SUBTYPE OF). Por exemplo, B0B1 é

uma interface que implementa a entidade complexa que pode ser descrita como “um A0 que é um B0 e um B1 ao mesmo tempo”.

Perdas observadas no caso t11

Todas as interfaces no esquema IDL da figura 41 correspondem às entidades no esquema EXPRESS da figura 40. No entanto, a tradução para IDL não contempla todos os detalhes do esquema EXPRESS.

O *status* ABSTRACT dos supertipos é ignorado. Uma vez que supertipos abstratos não devem ser instanciados, a tradução pode levar ao compartilhamento de objetos que só deveriam ser instanciados em conjunto com seus subtipos. Portanto, os esquemas EXPRESS e IDL são *incompatíveis* quanto a herança e entidades complexas.

Da mesma forma que em alguns casos anteriores, esta disparidade não deve afetar o compartilhamento baseado em padrões, quando os dois lados conhecem o esquema STEP que detalha a estrutura dos dados compartilhados, onde supertipos abstratos não são instanciados. O número de entidades complexas em IDL, derivadas da malha de entidades EXPRESS, representa uma dificuldade séria para o compartilhamento, pois incorre no problema adicional de explosão combinatória. Isto, no entanto, diz respeito ao desempenho do compartilhamento, não à perda semântica.

4.1.12 Outra tradução de entidades complexas

O propósito deste caso de tradução é investigar a perda semântica quanto ao compartilhamento de entidades complexas, usando um exemplo muito menor do que aquele do caso t11, com uma restrição ANDOR (e/ou) implícita. A figura 42 apresenta o esquema EXPRESS t12, seguido por sua tradução para IDL representada no arquivo t12.idl, ilustrado na figura 43.

```
(* t12.exp

     Purpose of the schema: to represent a complex entity data
     type.

          (ABS) e0
      (implicit ANDOR)
           /     \
          /       \
        e1         e2
*)
```

Figura 42 - Esquema EXPRESS t12 (continua...)

```
SCHEMA t12;

 ENTITY e0 ABSTRACT SUPERTYPE;
 END_ENTITY;

 ENTITY e1 SUBTYPE OF (e0);
 END_ENTITY;

 ENTITY e2 SUBTYPE OF (e0);
 END_ENTITY;

 (*
 COMPLEX;
 e1+e2;
 END_COMPLEX;
 *)

END_SCHEMA;
```

Figura 42 (continuação)

```
/* t12.idl
*/

interface  E0  {
};

interface  E1  : E0 {
};

interface  E2  : E0 {
};

interface  E1E2  : E1, E2 {
};
```

Figura 43 - Tradução do esquema t12 para IDL

Comentários relativos ao caso de tradução t12

A tradução para IDL foi produzida usando o tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 43 em arquivos diferentes, que foram incluídos manualmente no arquivo t12.idl para simplificar a apresentação.

A interface `E1E2` representa uma entidade complexa que foi incluída manualmente em t12.idl, de forma a respeitar a especificação SDAI (ISO 10303-22 1995), pois sua geração não foi realizada por *exp2idl*. O bloco COMPLEX comentado

na figura 42 contém a entidade complexa que foi traduzida para a interface E1E2. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

Perdas observadas no caso t12

O caráter não-implementável (ABSTRACT) perde-se na tradução para IDL. De forma semelhante ao caso t11, todas as interfaces correspondem a entidades EXPRESS na figura 42, mas a tradução não contempla todos os detalhes do esquema original.

A perda da diretiva ABSTRACT torna EXPRESS e IDL *incompatíveis* quanto a entidades complexas. Nenhuma perda semântica é esperada, entretanto, se o compartilhamento de dados de produtos for baseado em padrões, já que nenhum esquema EXPRESS que detalha a estrutura dos dados compartilhados permite a instanciação de supertipos abstratos.

4.1.13 Tradução de tipos de dados SET não-limitados

O propósito deste caso de tradução é investigar a perda semântica quanto ao tipo de dados SET não-limitado. A figura 44 apresenta o esquema EXPRESS t13, seguido por sua tradução para IDL representada no arquivo t13.idl, ilustrado na figura 45.

```
(* t13.exp

     Purpose of this schema: to represent the unbounded
     aggregate SET.

*)

SCHEMA t13;

 ENTITY e0;
     e0attr : e1;
 END_ENTITY;

 ENTITY e1;
 INVERSE
     e1attr : SET OF e0
          FOR e0attr;
 END_ENTITY;

END_SCHEMA;
```

Figura 44 - Esquema EXPRESS t13

```
/* t13.idl
*/

interface E0;
typedef sequence< E0 > E0__set;

interface  E1;
interface  E0  {
     attribute E1 e0attr;
};

interface  E1  {
     readonly attribute E0__set__list e1attr;

/*   The above translation, according to Part 26, 5.2.6 ENTITY
     data type, was not accepted by the Orbix idl compiler.
     Errors annotated:
     14:(semantic): Identifier `E0__set__list' not found
     14:(semantic): Name does not denote a type
     Translation accepted by the compiler:

     readonly attribute E0__set e1attr;
*/
};
```

Figura 45 - Tradução do esquema t13 para IDL

Comentários relativos ao caso de tradução t13

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 45 em arquivos diferentes, que foram incluídos manualmente no arquivo t13.idl para simplificar a apresentação. A sintaxe padrão não foi aceita pelo compilador Orbix. O código foi, então, adaptado conforme a figura 45, e a seguir verificado com sucesso usando Orbix.

Perdas observadas no caso t13

O código EXPRESS `SET of e0` da figura 44 foi mapeado para IDL como `E0__set__list`, conforme a figura 45. Esta tradução oferece uma correspondência um-para-um entre EXPRESS e IDL, isto é, a estrutura EXPRESS é traduzida com todos os seus detalhes, e a estrutura IDL gerada não tem outra interpretação possível.

EXPRESS e IDL são *equivalentes* com respeito a SETs não-limitados. Nenhuma perda ou ambigüidade está associada à tradução desta estrutura. Ainda assim, no experimento com o compilador IDL Orbix da IONA, a sintaxe construída de acordo com a Parte 26 (ISO 10303-26 1997) não foi aceita, como aponta a figura 45. Aparentemente, esta versão de Orbix usa outra sintaxe para IDL.

4.1.14 Outra tradução de REAL e STRING

O propósito deste caso de tradução é investigar a perda semântica quanto a REAL e STRING, considerando aspectos da sintaxe que não foram contemplados nos casos anteriores. A figura 46 apresenta o esquema EXPRESS t14, seguido por sua tradução para IDL representada no arquivo t14.idl, ilustrado na figura 47.

```
(* t14.exp

     Purpose of the schema: to represent aspects of simple
     data types (REAL and STRING) that were not depicted in
     translation cases t01 to t04.
*)

SCHEMA t14;

 ENTITY e0;
  e0attr : REAL(8);   -- 8 is the precision-specification
                      -- (significant digits)
 END_ENTITY;

 ENTITY e1;
  e1attr0 : STRING (10);
  e1attr1 : STRING (10) FIXED;
 END_ENTITY;

END_SCHEMA;
```

Figura 46 - Esquema EXPRESS t14

```
/* t14.idl

typedef string<10> string_10;

  interface  E0 {
     attribute double e0attr;
  };

  interface  E1 {
     attribute string<10> e1attr0;
     attribute string<10> e1attr1;

};
```

Figura 47 - Tradução do esquema t14 para IDL

Comentários relativos ao caso de tradução t14

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 47 em arquivos diferentes, que foram incluídos manualmente no arquivo t14.idl para simplificar a apresentação. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

Perdas observadas no caso t14

O tipo EXPRESS `REAL`, tal como no caso t01, é mapeado para IDL `double`, perdendo referência a sua precisão, como se pode observar na figura 47. Atributos `STRING` são traduzidos para IDL como `string`, mantendo a informação sobre tamanho limitado em 10, embora a diretiva `FIXED` do atributo `e1attr1` seja ignorada. EXPRESS e IDL são, neste caso, *incompatíveis*. Mesmo com esta perda semântica na tradução, o compartilhamento de dados de produtos baseado em padrões não deve sofrer prejuízo, já que as interfaces IDL permitem conduzir dados com os detalhes faltantes, e um mesmo esquema STEP deve ser conhecido na fonte e no destino da transmissão, impondo sua a estrutura sintática aos dados trafegados.

4.1.15 Outra tradução de tipos de dados agregados

O propósito deste caso de tradução é investigar a perda semântica quanto aos tipos de dados ARRAY, LIST, BAG (limitados), enfocando aspectos da sintaxe que não estão presentes nos esquemas STEP, embora pertençam à sintaxe de IDL. A figura 48 apresenta o esquema EXPRESS t15, seguido por sua tradução para IDL representada no arquivo t15.idl, ilustrado na figura 49.

Comentários relativos ao caso de tradução t15

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 49 em arquivos diferentes, que foram incluídos manualmente no arquivo t15.idl para simplificar a apresentação. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

Perdas observadas no caso t15

Os atributos EXPRESS com domínios `ARRAY`, `LIST`, e `BAG` foram traduzidos para IDL de forma semelhante à tradução do agregado `SET` nos casos t07 e t13. O mapeamento IDL para estas estruturas permite representá-las, porém o tamanho e os índices associados aos agregados são perdidos. Além disso, as diretivas `OPTIONAL` e `UNIQUE` são ignoradas. Isto faz com que os códigos EXPRESS e IDL sejam

```
(* t15.exp

     Purpose of the schema: to represent aspects of aggregates
     (ARRAY, LIST, BAG) that were not present in SOLIS, and
     therefore absent in previous test cases.  SET is already
     covered in suites t07 and t13.

*)

SCHEMA t15;

 ENTITY e1;
  e1attr : ARRAY[?:?] OF OPTIONAL UNIQUE SET [1:10] OF INTEGER;
 END_ENTITY;

 ENTITY e2;
   e2attr : LIST[0:?] OF UNIQUE e1;
 END_ENTITY;

 ENTITY e3;
  e3attr : BAG[1:?] OF INTEGER;
 END_ENTITY;

END_SCHEMA;
```

Figura 48 - Esquema EXPRESS t15

```
/* t15.idl
*/

typedef sequence< sequence < long > >
long__bounded_set__bounded_array;

interface E1;
typedef sequence< E1 > E1__bounded_list;
typedef sequence< long > long__bounded_bag;

interface  E1  {
     attribute long__bounded_set__bounded_array e1attr;
};

interface  E2  {
     attribute E1__bounded_list e2attr;
};

interface  E3  {
     attribute long__bounded_bag e3attr;
};
```

Figura 49 - Tradução do esquema t15 para IDL

incompatíveis, embora não haja impacto sobre a qualidade do compartilhamento baseado em padrões, já que agregados genéricos *podem* ser usados para conduzir dados de agregados limitados e indexados, e todas as partes envolvidas no compartilhamento devem usar o mesmo esquema STEP.

4.1.16 Tradução de atributos de entidades

O propósito deste caso de tradução é investigar a perda semântica quanto a aspectos de atributos de entidades, sendo que apenas INVERSE e OPTIONAL foram investigados nos casos anteriores. A figura 50 apresenta o esquema EXPRESS t16, seguido por sua tradução para IDL representada no arquivo t16.idl, ilustrado na figura 51.

```
(* t16.exp

     Purpose of the schema: to represent aspects of entity
     attributes (only INVERSE and OPTIONAL were depicted in
     other cases).

*)

SCHEMA t16;


  TYPE positive = INTEGER;
  WHERE
    notnegative : SELF > 0;
  END_TYPE;


  ENTITY e0;
    e0attr : OPTIONAL INTEGER;
  END_ENTITY;


  ENTITY e1;
    e1_height : REAL;
  DERIVE
    e1_ideal_weight : REAL := ( e1_height - 1.0 ) * 100.0;
  END_ENTITY;


  ENTITY e2door;
    e2_handle : e3knob;
  END_ENTITY;
```

Figura 50 - Esquema EXPRESS t16 (continua...)

```
(* The following declaration means that knob only exists
    if they are used in the role of handle in one
    instance of a door *)
ENTITY e3knob;
INVERSE
  e3_opens : e2door FOR e2_handle;
END_ENTITY;


ENTITY e4;
  e4code : INTEGER;
  e4name: STRING;
UNIQUE
  ur1: e4code, e4name;
END_ENTITY;


ENTITY e5;
  attr : REAL;
END_ENTITY;
ENTITY e6;
  attr : BINARY;
END_ENTITY;
ENTITY e7 SUBTYPE OF (e5,e6);
WHERE
  attr_pos : SELF\e5.attr > 0.0 ;
  (* attr as declared in e5, not e6 *)
END_ENTITY;


ENTITY e8;
  e8attr0 : NUMBER;
  e8attr1 : OPTIONAL REAL;
END_ENTITY;
ENTITY e9 SUBTYPE OF (e8);
  SELF\e8.e8attr0 : INTEGER;
END_ENTITY;
ENTITY e10 SUBTYPE OF (e8);
  SELF\e8.e8attr1 : REAL;
END_ENTITY;
ENTITY e11 SUBTYPE OF (e8);
  e11attr : NUMBER;
DERIVE
  SELF\e8.e8attr0 : REAL := 1 / e11attr;
END_ENTITY;


ENTITY e12;
  e12attr : INTEGER;
DERIVE
  e12plus : INTEGER := e12attr + 1;
END_ENTITY;
```

Figura 50 (continua...)

```
  ENTITY e13 SUBTYPE OF (e12);
  WHERE
    e13big : e12plus >= 18;
  END_ENTITY;


END_SCHEMA;
```

Figura 50 (continuação)

```
/* t16.idl
*/

typedef    long  positive;

interface  e0  {
     attribute long e0attr;
};

interface  e1  {
     attribute double e1_height;
};

interface  e3knob;
interface  e2door  {
     attribute e3knob e2_handle;
};

interface  e3knob  {
     readonly attribute e2door__list e3_opens;

};

interface  e4  {
     attribute long e4code;
     attribute string e4name;
};

interface  e5  {
     attribute double attr;
};

typedef sequence<octet> Binary;
interface  e6  {
     attribute Binary attr;
};

interface  e7  : e5,  e6 {
};
```

Figura 51 - Tradução do esquema t16 para IDL (continua...)

```
interface  e8  {
     enum Number_discriminant { Number_discriminant__long,
     Number_discriminant__double };

     union Number switch ( Number_discriminant ) {
     case Number_discriminant__long : long c1;
     case Number_discriminant__double : double c2;
     };
     attribute Number e8attr0;
     attribute double e8attr1;
};

interface  e9  : e8 {
};

interface  e10  : e8 {
};

interface  e11  : e8 {
     attribute Number e11attr;
};

interface  e12  {
     attribute long e12attr;
};

interface  e13  : e12 {
};
```

Figura 51 (continuação)

Comentários relativos ao caso de tradução t16

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 51 em arquivos diferentes, que foram incluídos manualmente no arquivo t16.idl para simplificar a apresentação. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

Perdas observadas no caso t16

As diretivas EXPRESS `SELF`, `OPTIONAL`, `WHERE`, `DERIVE`, e `UNIQUE` na figura 50 foram perdidas na tradução para IDL. Há ambigüidade não resolvida na herança do atributo `attr` de `e5` e `e6`, em `e7`. EXPRESS e IDL são severamente *incompatíveis* com respeito a estas características de atributos, que no momento não fazem parte de qualquer IR ou AP de STEP.

As diretivas `SELF`, `WHERE` e `DERIVE` referem-se a características não-armazenáveis dos dados STEP. Portanto, não implicam em perda no intercâmbio

interoperável baseado em padrões. Do mesmo modo que no caso t01, o acesso interoperável a uma ocorrência de atributo `OPTIONAL` tem resultado imprevisível, e a adoção de um valor *flag* que represente valor inexistente parece a solução mais promissora para aliviar esta perda. A restrição `UNIQUE` se aplica aos dados armazenados e, embora não seja imposta em IDL, faz parte do esquema STEP que governa a estrutura dos dados armazenados, enviados, e recebidos.

Em resumo, ainda que haja perda semântica na tradução de EXPRESS para IDL, e mesmo que as estruturas representadas neste casos t16 venham a fazer parte de algum esquema STEP, o intercâmbio baseado em padrões não deve sofrer perda, já que os participantes do intercâmbio devem usar o mesmo esquema STEP, e as interfaces IDL permitem transmitir os dados (à exceção de `OPTIONAL`, cuja solução foi apontada).

4.1.17 Tradução de entidades com herança múltipla

O propósito deste caso de tradução é investigar a perda semântica quanto a entidades associadas como supertipos e subtipos, formando herança múltipla. A figura 52 apresenta o esquema EXPRESS t17, seguido por sua tradução para IDL representada no arquivo t17.idl, ilustrado na figura 53.

```
(* t17.exp

     Purpose of the schema: to represent multiple inheritance.

*)

SCHEMA t17;

 ENTITY e0
   SUPERTYPE OF (e2);
 END_ENTITY;

 ENTITY e1
   SUPERTYPE OF (e2);
 END_ENTITY;

ENTITY e2
   SUBTYPE OF (e0,e1);
 END_ENTITY;

END_SCHEMA;
```

Figura 52 - Esquema EXPRESS t17

```
/* t17.idl
*/

interface  E0  {
};

interface  E1  {
};

interface  E2  : E0, E1 {
};
```

Figura 53 - Tradução do esquema t17 para IDL

Comentários relativos ao caso de tradução t17

A tradução para IDL foi produzida através do tradutor *exp2idl*. Esta tradução automática gerou as interfaces mostradas na figura 53 em arquivos diferentes, que foram incluídos manualmente no arquivo t17.idl para simplificar a apresentação. A sintaxe do código IDL resultante foi verificada com sucesso usando Orbix.

Perdas observadas no caso t17

A herança múltipla do subtipo `e2` em relação aos supertipos `e0` e `e1` na figura 52 é mapeada para código IDL *equivalente*. Em IDL, a interface `E2` herda de `E0` e `E1`, conforme ilustra a figura 53. Esta tradução permite um mapeamento um-para-um entre EXPRESS e IDL, sem disparidade sintática ou ambigüidade, portanto sem perda semântica.

4.2 Considerações finais sobre a análise da perda semântica

Pode-se extrair da lingüística uma analogia para ilustrar o problema da perda semântica na tradução: a hipótese de Sapir-Whorf propõe que a *estrutura* da língua nativa influencia fortemente a visão de mundo que uma pessoa adquire enquanto aprende a língua (KAY & KEMPTON 1994). Whorf descobriu que a língua indígena norte-americana Hopi “não contém palavras, formas gramaticais, construções ou expressões que se refiram diretamente ao que chamamos ‘tempo’, ou passado, presente ou futuro, ou persistente ou duradouro, ou movimento no sentido cinemático, preferencialmente a dinâmico” (WHORF 1956). Uma expressão em uma língua

européia contendo um coletivo de períodos de tempo (por exemplo, “dez dias”) não tem equivalente em Hopi.

Esta disparidade sintática não é resolvida por nenhuma construção com significado semelhante em Hopi. Portanto, há perda de significado (semântica). Embora as linguagens processáveis por computador sejam notavelmente menos complexas do que as linguagens naturais, com gramáticas pertencentes às duas classes mais simples (regular e livre-de-contexto) da classificação de Chomsky, conforme descrita por TREMBLAY & SORENSON (1985), este exemplo reflete a mesma situação que ocorre quando se tenta traduzir código escrito em uma linguagem de computador para outra.

Perdas no compartilhamento de dados de produtos em rede heterogênea em empresas virtuais

Há três formas de tradução envolvidas na integração de modelos de dados de produtos baseada em padrões visando ao compartilhamento em rede:

1. Tradução de um modelo de dados de um CAx específico para STEP, e vice-versa;
2. Tradução dos modelos de dados STEP, escritos em EXPRESS, para IDL, visando à construção de interfaces para o acesso em rede; e
3. Tradução de interfaces IDL para uma linguagem de implementação, uma vez que IDL não serve para a implementação direta.

Estas traduções são potenciais geradoras de perda de informação, conforme descreve a seção 3.2.3. As perdas relativas à tradução de STEP para modelos de dados específicos e vice-versa (item 1 acima) dependem da qualidade dos tradutores (pré- e pós-processadores) entre STEP e o modelo de dados do sistema CAx específico. A implementação das interfaces IDL em linguagem de implementação pode resultar em perda adicional, uma vez que as estruturas IDL têm que ser mapeadas para estruturas na linguagem de implementação (item 3). A análise da perda semântica apresentada nesta tese, no entanto, enfoca as perdas intrínsecas à tradução EXPRESS-IDL (item 2).

A seguir, apresenta-se um sumário das perdas detectadas nos 17 casos de análise da perda semântica na tradução de modelos de dados de produtos STEP para interfaces IDL.

Tabela 7 - Perda semântica na tradução de EXPRESS para IDL

Caso	Estruturas EXPRESS em foco	Perdas observadas na tradução, de acordo com o mapeamento SDAI-IDL SDAI (ISO 10303-26 1997)
t01	REAL, INTEGER, STRING	A diretiva `OPTIONAL` é ignorada no mapeamento IDL.
t02	NUMBER	Nenhuma.
t03	BOOLEAN, LOGIC	`Logical` e `Boolean` permitem `unset` (valor não atribuído) em IDL.
t04	BINARY	Tamanho de atributo `BINARY`, quando especificado, é perdido. A diretiva `FIXED` também é ignorada.
t05	ARRAY, LIST, BAG	Tamanho e índice de agregados são perdidos (não há como detalhar esta informação, de acordo com o documento STEP Part 26).
t06	ARRAY não-limitado, BAG, LIST	Nenhuma.
t07	SET	Tamanho e índice de agregados são perdidos.
t08	SELECT	Nenhuma.
t09	SELECT aninhado	Nenhuma.
t10	ENUMERATION	Em IDL, `unset` (valor não atribuído) é permitido.
t11	Entidade complexa	O caráter `ABSTRACT` de supertipos é perdido.
t12	Entidade complexa	O caráter `ABSTRACT` de supertipos é perdido.
t13	SET não-limitado	Nenhuma.
t14	REAL, STRING (detalhes ausentes no caso t01)	A precisão de `REAL` e o caráter `FIXED` de `STRING` são perdidos.
t15	Agregados (detalhes ausentes nos APs STEP)	O caráter `OPTIONAL`, a restrição `UNIQUE`, bem como tamanho e índice de agregados são perdidos.
t16	Atributos de entidades	Perda generalizada de restrições sobre atributos, sua redeclaração, e a restrição adicional sobre atributos herdados.
t17	Herança múltipla	Nenhuma.

Sumário da perda semântica detectada na tradução EXPRESS-IDL

A tabela 7 apresenta um resumo das perdas detectadas nos 17 casos de tradução de modelos de dados STEP para IDL. A disparidade identificada entre EXPRESS e IDL implica em esforço adicional para ajustar, para que possa ser usado pelo sistema CAx destinatário, um modelo de dados de produtos compartilhado em rede heterogênea em uma empresa virtual.

Os procedimentos propostos para recuperar ou aliviar a perda, em cada caso de tradução, levam em conta apenas a comparação entre o poder de modelagem de

EXPRESS e IDL. É possível que o conhecimento específico sobre a natureza das aplicações de engenharia possa ajudar a desenvolver outras estratégias para evitar a perda da informação. Por exemplo, suponha-se que uma aplicação específica manipule arranjos (*arrays*). Em IDL, os índices e tamanhos dos arranjos são perdidos, mas a aplicação pode apresentar algum padrão de uso de arranjos (por exemplo, usar sempre o mesmo tamanho e índice-base). Neste caso, uma solução específica para aliviar o problema, baseada no padrão de uso, pode ser desenvolvida.

CAPÍTULO 5

CONCLUSÃO

Este capítulo apresenta um sumário da tese, resultados e contribuições, e recomendações para trabalhos futuros de pesquisa.

5.1 Sumário da tese

Esta tese apresentou uma visão geral sobre a gerência da informação em aplicações de engenharia com foco em modelos de dados de produtos compartilhados em redes distribuídas. Foram discutidas técnicas de gerência de banco de dados para aplicações tradicionais de negócios e para aplicações de engenharia. O problema de intercâmbio de dados de produtos (PDE) foi introduzido, juntamente com uma discussão sobre a evolução das especificações e padrões para PDE. Foram explicados a arquitetura, a construção da modelagem da informação, e a implementação do padrão ISO 10303, STandard for the Exchange of Product model data (STEP), com considerações sobre o futuro do padrão. O problema da interoperabilidade de aplicações foi apresentado, incluindo a descrição de especificações propostas para a solução do problema, especialmente CORBA (Common Object Request Broker Architecture).

O objetivo principal da tese, o desenvolvimento de conhecimento para a realização das empresas industriais virtuais, foi abordado através do conceito de bancos de dados STEP integrados em rede. Os bancos de dados STEP são vitais para ambos aspectos da integração das empresas industriais: engenharia concorrente (aspecto vertical) e empresas virtuais (aspecto horizontal).

Para a realização das empresas industriais virtuais, foi introduzida uma abordagem baseada em padrões para a interoperabilidade em rede de modelos de dados de produtos. STEP foi adotado como o padrão para a representação, arquivamento, intercâmbio e compartilhamento de modelos de dados de produtos. CORBA foi a arquitetura-padrão escolhida para a realização do acesso interoperável em rede a modelos de dados de produtos. A escolha de STEP e CORBA é suportada pela ISO, que produziu STEP e apóia a adoção de CORBA para interoperabilidade de modelos STEP através do mapeamento IDL para SDAI. STEP e CORBA são as tecnologias escolhidas para interoperabilidade de modelos de dados de produtos

segundo o projeto NIIIP (National Industrial Information Infrastructure Protocols) do governo norte-americano.

Os desafios para a realização da interoperabilidade de modelos de dados de produtos usando STEP e CORBA foram discutidos, e uma análise da perda semântica na tradução de modelos de dados de produtos interoperáveis em rede foi conduzida. Para tanto, foi construída uma *suite* de modelos EXPRESS (a linguagem para representação de modelos STEP) e respectivas traduções para IDL. Cada um dos 17 modelos enfocou algum aspecto dos tipos de dados EXPRESS. As inconsistências entre as sintaxes de EXPRESS e IDL foram identificadas, e os possíveis impactos no compartilhamento de dados de produtos foram discutidos. Ações para aliviar a perda semântica resultante das inconsistências foram sugeridas.

5.2 Resultados e contribuições

Pode-se concluir, a partir da análise da perda semântica, que modelos de dados de produtos compartilhados em rede usando STEP e CORBA estão sujeitos a inconsistências entre os tipos de dados EXPRESS e IDL. Ações para aliviar estas perdas, com base na disparidade sintática entre EXPRESS e IDL, foram propostas no capítulo 4. Nas recomendações, na seção 5.3, outras estratégias para a recuperação de perda semântica são sugeridas, levando em conta não apenas a disparidade semântica, mas também o conhecimento sobre o uso dos dados em uma aplicação específica de engenharia.

A figura 54 ilustra o acesso interoperável em rede a dados de produtos usando STEP e CORBA, bem como as traduções necessárias. Primeiramente, os dados de algum banco de dados ou sistema CAx, armazenados segundo um modelo de dados específico, têm que ser ajustados (traduzidos) a um modelo de dados STEP -- um dos protocolos de aplicação (AP). Os dados representados por modelos STEP podem ser acessados através de operações da interface padrão para acesso aos dados (SDAI). Para o acesso a objetos STEP em nível de granularidade fina, é preciso que se publiquem interfaces IDL e sejam disponibilizadas através de ORBs. Isto requer a tradução da estrutura de entidades escrita em EXPRESS para IDL. Finalmente, para a implementação do acesso, as interfaces IDL devem ser mapeadas (traduzidas) para uma linguagem de implementação.

Qualquer destas três traduções pode impor perda semântica nos dados compartilhados. Esta tese tratou especificamente da perda semântica na tradução de EXPRESS para IDL, que é um subconjunto do problema geral de perda semântica em bancos de dados de produtos interoperáveis em rede.

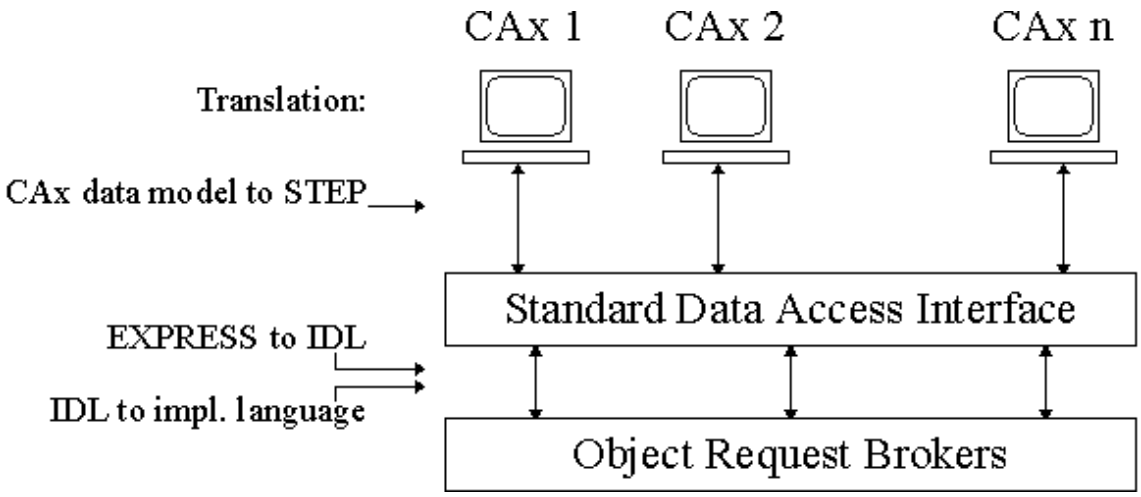


Figura 54 - Traduções no compartilhamento de modelos de dados de produtos em rede

EASTMAN (1994) considera que a necessidade de conversão de tipos de dados é, fatalmente, um impeditivo ao intercâmbio de arquivos bem-sucedido. SANDERSON (1994) sustenta que o problema de perda de informação na tradução de dados nunca será completamente resolvido, mas que é possível tratar de subclasses do problema genérico. De fato, com base nas experiências anteriores com intercâmbio de arquivos em formato neutro, esperava-se que o uso de STEP gerasse algum nível de perda semântica no intercâmbio de informações. Porém, o trabalho adicional para superar a disparidade imposta pelo uso de várias linguagens e modelos de dados deve ser comparado ao esforço atual para reconstruir, ou para traduzir diretamente conjuntos de dados de um sistema CAx para outro. Mesmo enfrentando perda semântica, a experiência da indústria sugere que a integração em rede de modelos de dados de produtos é uma alternativa mais viável e econômica.

A construção do padrão STEP é suportada por um grande número de organizações, tanto desenvolvedoras quanto usuárias de sistemas CAx. A razão para este apoio não é somente econômica, mas também estratégica. A condição de padrão internacional deve impor uma tendência à adoção de STEP sempre que o intercâmbio de dados de produtos for necessário, até o ponto de tornar-se uma exigência para a participação em negócios que envolvem o compartilhamento de dados de produtos.

Contribuições

Esta tese oferece as seguintes contribuições, consoantes com o objetivo de desenvolver conhecimento e tecnologia para a realização de bancos de dados STEP interoperáveis em rede:

- A construção de uma *suite* de esquemas EXPRESS adequados para a análise da perda semântica na tradução. São 17 esquemas que enfocam os diversos aspectos dos tipos de dados EXPRESS, obtidos a partir dos IRs, APs, e outros documentos STEP. Estes esquemas podem ser usados em casos de tradução e análise de disparidade sintática entre EXPRESS e outras linguagens;
- A identificação da disparidade semântica na tradução de modelos de dados de produtos, baseada na disparidade sintática existente entre EXPRESS e IDL; e
- A avaliação do impacto da disparidade semântica no compartilhamento baseado em padrões (isto é, STEP-CORBA) de modelos de dados de produtos compartilháveis em rede, e a proposição de procedimentos informais para minorar ou eliminar a ocorrência de perdas.

As contribuições descritas acima referem-se a esquemas STEP genéricos. A próxima seção apresenta recomendações para pesquisa futura sobre perda semântica no compartilhamento de dados conforme um AP específico, e outras questões de pesquisa em aberto.

5.3 Recomendações

A análise da perda semântica apresentada nesta tese revela quais dados representados segundo um esquema em EXPRESS poderão ser perdidos quando acessados através de uma interface IDL. O compartilhamento de dados de produtos baseado em padrões, usando STEP, sempre é realizado usando um AP específico. Até o presente, apenas os APs 201 (ISO 10303-201 1994) e 203 (ISO 10303-203 1994) foram publicados como padrões internacionais. A extensão do impacto das perdas identificadas nesta tese no compartilhamento de dados de produtos segundo um AP específico é assunto a ser investigado. Tal investigação demanda conhecimento perito sobre os dados de produtos modelados no AP específico e as aplicações de engenharia que podem usar os dados para compartilhamento. Uma pesquisa deste tipo deve responder as seguintes questões:

- Quais, dentre as estruturas EXPRESS suscetíveis à perda semântica, ocorrem no AP específico?;
- Quão freqüentemente estas estruturas estão presentes em conjuntos de dados típicos a serem compartilhados segundo o AP específico?;

- Qual é o impacto ou implicação de cada tipo de perda semântica para as aplicações que compartilham dados segundo o AP específico?; e
- Quais são, se existem, os procedimentos a serem adotados para corrigir a perda no compartilhamento segundo o AP específico? Há uma forma padronizada para a correção? É automatizável?.

Outras perdas, ainda, são possíveis, além daquelas citadas até aqui. Dependendo da linguagem escolhida para a implementação das interfaces IDL (por exemplo, C ou C++), mais perdas podem ser impostas aos dados STEP acessados. A identificação destas perdas e de soluções para sua correção são tema que merece ser abordado em pesquisa futura.

Fora do escopo da tecnologia de empresas virtuais e do compartilhamento interoperável em rede usando o mapeamento IDL para SDAI, a ISO publicou outros rascunhos de padrões (*draft standards*) que definem mapeamentos de SDAI usando linguagens de programação como C e C++, permitindo o acesso de múltiplas aplicações a um banco de dados STEP, conforme descreveu a seção 2.2. Há perda semântica inerente à escolha da linguagem de implementação, para a qual uma análise como a desenvolvida nesta tese é tema de pesquisa relevante. No início da pesquisa relatada nesta tese, propunha-se incluir também uma tradução EXPRESS-C++ em cada um dos 17 casos, para então identificar perdas e comparar com a tradução EXPRESS-IDL. No entanto, este intuito não pode ser realizado devido ao estágio muito inicial e incompleto do mapeamento C++ para SDAI, o que não permitiu construir as traduções. À medida que o processo de padronização avança e o mapeamento C++ para SDAI é completado, este estudo passa a ser viável. Adicionalmente, um mapeamento Java para SDAI deverá ser construído no futuro, oferecendo outra oportunidade de pesquisa para a exploração de perda semântica e sua correção ou atenuação.

Ao mesmo tempo em que os documentos STEP passam pelas etapas de padronização da ISO, um debate intenso sobre a qualidade do padrão acontece. Uma das áreas de debate diz respeito à construção dos APs, considerados o "coração" de STEP. Duas correntes alternativas propõem a construção de APs independentes e o uso de ontologias comuns para a construção dos IRs (para gerar APs de melhor qualidade).

METZGER (1996) propõe construir cada AP independentemente e "desistir da idéia de um modelo de dados completo e unificado". A realização desta proposta representaria uma mudança radical no processo de desenvolvimento de STEP, que constrói APs a partir da interpretação e reuso dos IRs.

O uso de ontologias comuns na construção de modelos da informação STEP vem sendo proposto para resolver o problema de redundância gerada pela abordagem *bottom-up* adotada, que impõe grande dificuldade à compatibilização de APs relativos a aplicações de engenharia que tem áreas de interesse comuns. Ontologias comuns foram usadas originalmente no compartilhamento de conhecimento. Há pesquisa atual investigando a aplicabilidade da engenharia ontológica a STEP:

> "As ontologias ... podem ser usadas para atingir a unificação da tradução de modelos EXPRESS para outras linguagens, ou a interoperabilidade de modelos EXPRESS com outros modelos representados em outras linguagens". (MEIS; OSTERMAYER 1996)

A unificação foi definida no âmbito de STEP como "o processo pelo qual dois enunciados em lógica são reconhecidos como equivalentes", conforme EASTMAN (1994). Mas, o desenvolvimento do SUMM (*Semantic Unification Meta-Model*), uma "abordagem matemática rigorosa para a unificação e integração de modelos, independentemente da linguagem na qual os modelos foram formulados" (EASTMAN 1994) foi descontinuado.

O uso de ontologias comuns pode trazer os benefícios da unificação de volta ao processo de desenvolvimento de STEP. A tabela 8 apresenta um paralelo entre o uso de STEP para compartilhamento de dados de produtos e o uso de ontologias comuns para o reuso e compartilhamento de conhecimento.

Tabela 8 - Paralelo entre o uso de STEP para compartilhamento de dados de produtos e ontologias comuns para compartilhamento de conhecimento

STEP para PDE e compartilhamento	Ontologias comuns para compartilhamento de conhecimento
EXPRESS é a linguagem de descrição de dados usada para construir todos os modelos da informação do padrão STEP.	KIF (Knowledge Interchange Format) é uma forma canônica, uma linguagem formal para o intercâmbio de conhecimento.
EXPRESS pode ser lida por pessoas, embora não seja este seu maior propósito.	KIF pode ser lida por pessoas, embora não seja este seu maior propósito.
Os IRs são modelos conceituais gerais para o reuso de modelos de dados de produtos.	Ontologias comuns são vocabulários de termos para representação e reuso de conhecimento.
Há perda semântica na tradução de modelos de dados STEP.	"Nem toda expressão em KIF pode ser traduzida para todas as linguagens-objetos" (MEIS; OSTERMAYER 1996)

GRUBER (1992) propõe a aplicação de restrições sobre KIF para a construção de ontologias portáveis. Um tópico de pesquisa promissor é a investigação de um conjunto de limitações a serem impostas sobre modelos de dados escritos em

EXPRESS, de forma a faze-los portáveis (sem perda semântica) no âmbito de um modelo representacional, que poderia ser, por exemplo: (1) o modelo de ACIS, de forma que todo sistema CAx que usa o *kernel* ACIS fosse capaz de interpretar corretamente o modelo de dados STEP; ou (2) os tipos de dados IDL, de forma que modelos de dados de produtos compartilhados em rede interoperável fossem isentos de disparidades sintáticas entre EXPRESS e IDL.

REFERÊNCIAS BIBLIOGRÁFICAS

ATKINSON, M.; DeWITT, D.; MAIER, D.; BANCILHON, F.; DITTRICH, K.; ZDONIK, S. "The object-oriented database system manifesto". In: *Deductive and object-oriented databases* (eds.: KIM, W.; NICOLAS, M.; NISHIO, S.). Elsevier, pp. 223-240, 1990.

BARRA, R. "Projeto B-STEP: Uma Iniciativa Brasileira em STEP". In: *II Seminário Internacional Aplicações de STEP para a Integração de Sistemas*. Instituto de Pesquisas Tecnológicas, São Paulo, 27 de Novembro de 1996.

BATINI, C.; LENZERINI, M.; NAVATHE, S. "A Comparative Analysis of Methodologies for Database Schema Migration". *ACM Computing Surveys* 18 (December 1986), 323-364, 1986.

BRANDO, T. "Comparing CORBA and DCE". *Object Magazine* 6 (1), pp. 52-57, March 1996.

CECCHINI, M. "Experiência da Embraer em Troca de Dados de Produtos". In: *II Seminário Internacional Aplicações de STEP para a Integração de Sistemas*. Instituto de Pesquisas Tecnológicas, São Paulo, 27 de Novembro de 1996.

CHEN, P. "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Transactions on Database Systems* 1 (1), pp. 9-36, 1976.

CLARK, S. *The NIST working form for STEP*. National Institute of Standards and Technology, Report NISTIR 4351, 1990.

CODD, E. "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM* 13 (6), pp. 377-387, 1970.

EASTMAN, C. "Out of STEP?" (Comment). *Computer Aided Design* 26 (5), pp. 338-340, 1994.

EGGERS, J. *Implementing EXPRESS in SQL*. Technical Report ISO TC184/SC4/WG1 Doc. 292, ISO, October 1988.

FOWLER, J. *STEP for Data Management, Exchange and Sharing*. Technology Appraisals, UK. 214 pp., 1995.

GRUBER, T. *Ontolingua: A Mechanism to Support Portable Ontologies*. Knowledge Systems Laboratory, Stanford University, 36 pp., 1992. Available at: http://www-ksl.stanford.edu/knowledge-sharing/papers/README.html.

HARDWICK, M. *Data Protocols for the Industrial Virtual Enterprise*. Source unknown (possibly EUG'96 - EXPRESS Users Group Conference), 1996.

HARDWICK, M.; SPOONER, D.; RANDO, T; MORRIS, K. "Sharing Manufacturing Information in Virtual Enterprises". *Communications of the ACM* 39 (2), pp. 46-54, 1996.

HARDWICK, M.; DOWNIE, B.; KUTCHER, M.; SPOONER, D. "Concurrent Engineering with Delta Files". *IEEE Computer Graphics &* Applications 15 (1), pp. 62-68, 1995.

HARDWICK, M.; LOFFREDO, D. "Using EXPRESS to Implement Concurrent Engineering Databases". In: *Proceedings of the Computers in Engineering Conference and the Engineering Database Symposium*. (eds: BUSNAINA, A.; RANGAN, R.). ASME, Boston, MA, pp. 1069-1083, September 17-20, 1995.

HEILER, S.; DAYAL, U.; ORENSTEIN, J.; RADKE-SPROULL, S. "An Object-Oriented Approach to Data Management: Why Design Databases Need It". *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pp. 335-340, 1987.

HEIMBIGNER, D. *Why CORBA Doesn't Cut It or Experiences with Distributed Objects*. SETT Presentation, University of Colorado, Boulder, 30 June 1995.

ISO 10303-1. *Product Data Representation and Exchange - Part 1: Overview and Fundamental Principles*. Committee Draft, September 15, 1992.

ISO 10303-11. *Product Data Representation and Exchange - Part 11: EXPRESS Language Reference Manual*. Document TC184/SC4/WG5 N65(P2). ISO International Standard, November 1, 1994.

ISO 10303-201. *Product Data Representation and Exchange - Part 201: Application Protocol: Explicit Draughting*. ISO International Standard, 1994.

ISO 10303-203. *Product Data Representation and Exchange - Part 203: Application Protocol: Configuration Controlled Design*. ISO International Standard, 1994.

ISO 10303-214. *Product Data Representation and Exchange - Part 214: Application Protocol: Core Data for Automotive Mechanical Design Process*. Document TC184/SC4/WG3 N436. Committee Draft, August 8, 1995.

ISO 10303-22. *Product Data Representation and Exchange - Part 22: Implementation Methods: Standard Data Access Interface*. Committee Draft, May 31, 1995.

ISO 10303-23. *Product Data Representation and Exchange - Part 23: Implementation Methods: ++ Programming Language Binding to the Standard Data Access Interface*. Committee Draft, December 25, 1995.

ISO 10303-24. *Product Data Representation and Exchange - Part 24: Implementation Methods: Standard Data Access Interface - C Language Late Binding*. Committee Draft, July 28, 1995.

ISO 10303-26. *Product Data Representation and Exchange - Part 26: Implementation Methods: Interface Definition Language Binding to the Standard Data Access Interface*. Document ISO TC184/SC4/WG11 N019. Committee Draft, 17 March 1997.

ISO 10303-41. *Product Data Representation and Exchange - Part 41: Integrated Resources: Fundamentals of Product Description and Support*. ISO Draft International Standard, 1994.

ISO 10303-42. *Product Data Representation and Exchange - Part 42: Integrated Resources: Geometric and Topological Representation*. Document TC184/SC4/WG3 N125a. ISO Committee Draft, July 31, 1992.

JOSEPH, J.; THATTE, S.; THOMPSON, C.; WELLS, D. “Object-Oriented Databases: Design and Implementation”. *Proceedings of the IEEE* 79 (1), pp. 42-64, 1991.

KAY, P.; KEMPTON, W. “What is the Sapir-Whorf Hypothesis?” *American Anthropologist* 86 (1), pp. 65-79, 1984.

KENT, W. *Data and Reality*. North-Holland, 211 pp., 1978.

KERN, V.; BARRA, R.; BARCIA, R. “Implementation of Standardized Shareable Product Databases” *Proceedings of the II International Congress of Industrial Engineering* (CD-ROM), Piracicaba SP, Brazil, October 7-10, 1996.

KERN, V.; BØHN, J.; BARCIA, R. “The Building of Information Models in STEP”. *Proceedings of the II International Congress of Industrial Engineering* (CD-ROM), Piracicaba SP, Brazil, October 7-10, 1996.

KERN, V.; BØHN, J. “STEP Databases for Product Data Exchange”. In: *Proceedings of I International Congress of Industrial Engineering*. Vol. III, pp. 1337-1341, São Carlos SP, Brazil, September 4-7, 1995.

KERN, V. “Database Systems for CAD”. In: *Computer-Aided Design I - Fall 1994 Term Papers*. Mechanical Engineering Department, Virginia Polytechnic Institute & State University, Blacksburg VA, pp. 67-74, 1994.

KHOSHAFIAN, S. *Object-Oriented Databases*. John Wiley & Sons, 362 pp., 1993.

KIEKENBECK, J.; SIEGENTHALER, A.; SCHLAGETER, G. “EXPRESS to C++: A Mapping of the Type-System”. *EXPRESS User's Group (EUG) Conference '95*, 1995.

KIGGANS, R. Development and Implementation of ISO 10303 (STEP). Speech at the *II Seminário Internacional Aplicações de STEP para a Integração de Sistemas*. Instituto de Pesquisas Tecnológicas, São Paulo SP, Brazil, November 27, 1996.

LIBES, D. *The NIST STEP Part 21 Exchange File Toolkit: An Update*. National Institute of Standards and Technology. Report NISTIR 5187, 1993.

MAYBEE, M.; HEIMBIGNER, D.; OSTERWEIL, L. *Multilanguage Interoperability in Distributed Systems: Experience Report*. University of Massachusetts Amherst, Report UM-CS-1995-075, August 1995.

McCUSKER, T. "Workflow Takes on the Enterprise". *Datamation* 39, December 1993.

McLAY, M.; MORRIS, K. *The NIST STEP Class Library (STEP into the Future)*. National Institute of Standards and Technology. Report NISTIR 4411, 1990.

MEAD, M.; THOMAS, D. *Proposed Mapping from EXPRESS to SQL*. Rutherford Appleton Laboratory, May 1989.

MEIS, E.; OSTERMAYER, R. *Recommendations to the STEP Committee*. Kactus Consortium, Document KACTUS-01-RPK-D007 v. 1.1, 24 pp., September 28, 1996. Available from http://swi.psy.uva.nl/projects/NewKACTUS/home.html.

METZGER, F. "The Challenge of Capturing the Semantics of STEP Data Models Precisely". In: *Proceedings of the first PAKM'96, Practical Aspects of Knowledge Management Conference*. Basel, Swiss, October 30-31, 1996.

MORRIS, K. *Translating Express to SQL: A User's Guide*. National Institute of Standards and Technology. Report NISTIR 4341, 1990.

MOWBRAY, T. "How to Apply Open Systems to OO Architectures". *Object Magazine* 6 (1), pp. 84-86, 1996.

NIIIP. *NATIONAL INDUSTRIAL INFORMATION INFRASTRUCTURE PROTOCOLS CONSORTIUM - The NIIIP Reference Architecture*. Vol. Report NTR96-01, Cycle 0, Revision 6, 652 pp., January 16, 1996.

OFFICE OF SCIENCE and Technology Policy. Federal Coordinating Council for Science, Engineering, and Technology. *High Performance Computing & Communications: Toward a National Information Infrastructure*. Report by the Committee on Physical, Mathematical, and Engineering Sciences, Washington D.C., 176 pp., 1994.

OMG. *OBJECT MANAGEMENT GROUP - The Common Object Request Broker: Architecture and Specification*. Revision 2.0, July 1995.

OMG. *OBJECT MANAGEMENT GROUP - Object Request Broker Architecture*. OMG TC Document 93.7.2, Framingham MA, 1993.

RAGHAVAN, V. *STEP Relational Interface*. Master's Thesis. Rensselaer Polytechnic Institute, Troy, New York, December 1992.

RANDO, T.; McCABE, L. *SDAI: An Object-Oriented Information Sharing Standard*. To be published, 1996.

RANDO, T.; PAOLONI, M. "Mapping EXPRESS/SDAI into the CORBA Standard". In: *4th EXPRESS User Group (EUG'94) Conference*. 1994.

REDONDO, A. "Padrões para Troca de Dados CAD: ACIS, IGES e STEP". In: *II Seminário Internacional Aplicações de STEP para a Integração de Sistemas*. Instituto de Pesquisas Tecnológicas, São Paulo SP, Brazil, November 27, 1996.

SANDERSON, D.; *Loss of Data Semantics in Syntax Directed Translation*. Ph.D. Thesis. Troy, NY: Rensselaer Polytechnic Institute, Computer Science, 1994.

SANDERSON, D.; SPOONER, D. "Mapping Between EXPRESS and Traditional DBMS Models". In: *Proceedings of the EXPRESS Users Group EUG'93*. Berlin-Germany, October 2-3, 1993.

SCHENCK, D.; WILSON, P. *Information Modeling: The EXPRESS Way*. Oxford University Press, New York, 388 pp., 1994.

SOLEY, R.; STONE, C. (eds.). *Object Management Architecture Guide*. Third edition. John Wiley & Sons, Framingham MA, 164 pp., 1995.

SOLIS. *SC4 On Line Information Service*. Repository of STEP documentation, including STEP IR and AP schemata, as of July 17th, 1996. Available at ftp://ftp.cme.nist.gov/pub/subject/sc4.

STI. STEP Tools, Inc. “What is Your Application?” *STEP Tools News* 1 (2), September, 1992.

STONE, C.; HENTCHEL, D. “Database Wars Revisited”. *Byte* October 1990, pp. 233-242, 1990.

SU, S.; LAM, H.; LEE, T.; ARROYO, J. “On Bridging and Extending OMG/IDL and STEP/EXPRESS for Achieving Information Sharing and System Interoperability”. *EXPRESS User's Group (EUG) Conference '95*, 1995.

THE COMMITTEE for Advanced DBMS Function. *Third Generation Data Base System Manifesto*. U.C. Berkeley Memorandum No. UCB/ERL M90/28, April 9, 1990.

TIBBITTS, F. “CORBA: A Common Touch for Distributed Applications”. *Data Communications* 24 (7), pp. 71-75, 1995.

TREMBLAY, J.; SORENSON, P. *The Theory and Practice of Compiler Writing*. McGraw-Hill, 1985.

TSICHRITZIS, D.; KLUG, A. (eds.) “The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems”. *Information Systems* 3, pp. 173-191, 1978.

URBAN, S.; SHAH, J.; ROGERS, M.; JEON, D.; RAVI, P.; BLIZNAKOV, P. “A Heterogeneous, Active Database Architecture for Engineering Data Management”. *International Journal of Computer Integrated Manufacturing* 7, pp. 276-293, 1994.

VINOSKI, S. “Distributed Object Computing with CORBA”. *C++ Report* July/August 1993.

WATSON, A. “The OMG After CORBA 2”. *Object Magazine* 6 (1), pp. 58-59, March 1996.

WHORF, B. *Language, Thought, and Reality*. MIT Press, 278 pp., 1956.

WILSON, P. “Information And/Or Data?” *IEEE Computer Graphics &* Applications 7, pp. 58-61, 1987.

YANG, Y. “The STEP Integration Information Architecture”. In: Law, K. (ed.): *Engineering Data Management: Key to Success in a Global Market*. Proceedings of the 1993 ASME International Computers in Engineering Conference and Exposition, San Diego-CA, pp. 39-47, 1993.

APÊNDICE

VERSÃO ORIGINAL EM INGLÊS

Conforme declara o prefácio, esta tese foi produzida originalmente em inglês. A seguir, apresenta-se a versão original. Por simplicidade de apresentação, a versão em inglês tem numeração própria de páginas, figuras e tabelas.

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

MAINTAINING SEMANTICS IN THE INTEGRATION OF NETWORK INTEROPERABLE PRODUCT DATA MODELS

Vinícius Medina Kern

Doctoral thesis submitted as a partial fulfillment of the requirements for a degree of 'Doutor em Engenharia de Produção' at the Universidade Federal de Santa Catarina, Brasil.

Florianópolis, December of 1997.

Maintaining Semantics in the Integration of Network Interoperable Product Data Models

by

Vinícius Medina Kern

Thesis submitted to the Graduate Program in Production Engineering of the Federal University of Santa Catarina (PPGEP/UFSC) in partial fulfillment of the requirements for the degree of DOCTOR IN ENGINEERING.

Approved:

_____

Ricardo Miranda Barcia, Ph.D, adviser
Coordinator of the Graduate Program in Production Engineering - UFSC

_____

Jan Helge Bøhn, Ph.D., external participant

_____

Carlos Frederico Bremer, Dr., external participant

_____

Roberto Carlos dos Santos Pacheco, Dr. Eng.

_____

Alejandro Martins Rodriguez, Dr. Eng.

ACKNOWLEDGMENTS

Brazilian taxpayers supported great part of my education. I am very thankful and obliged to the Brazilian people. The love of my family was vital. Thanks Dad, Mom, Dani and Celsinho. And Luciana, whose love and support helped me to get through the ups and downs of this long effort.

Funding was essential and came from different sources, at different times. During most of my doctorate, including part of the 2-year period abroad, the research was funded by the Brazilian National Research Council (CNPq), under process number 200951/94-7. I had funding also from CAPES, the National Institute of Standards and Technology (NIST), and UNIVALI, where I teach for Computer Science.

I am grateful for the excellent resources offered to me by the Virginia Polytechnic Institute and State University (Virginia Tech). Especially, I would not be able to write this thesis without the help and resources from the Writing Center, Newman Library, and the CADLAB at the Department of Mechanical Engineering. The same is true about the library, office, software, and installations provided by NIST at Gaithersburg, MD, USA. The Computer Technology Center (CTI) in Campinas, Brazil, provided me with a computer account and the opportunity to converse with the participants of Project B-STEP. I am also obliged to the Brazilian Ministry of Foreign Relations (MRE), which payed the tickets used in the visits to Brazilian education and research institutions, as part of the Talented Researchers Return Program (PERT'95) of CNPq.

I thank Dr. Ricardo Barcia, my adviser, for the opportunity. Dr. Jan Helge Bøhn (Virginia Tech, Department of Mechanical Engineering) agreed to be my adviser in the U.S., and I feel very lucky for this. In a friendly, enthusiastic, objective, and brilliant way, he set the example and helped me to develop my skills as a researcher, writer, speaker, and adviser. This thesis was dramatically improved in relation to the first draft, thanks mainly to his review. Dr. Osama K. Eyada first received me as an advisee at Virginia Tech, Department of Industrial and Systems Engineering (ISE), and supported me in the choice to move to Mechanical Engineering, under Dr. Bøhn's advising, in the best interest of my research. I am indebted to Mary Mitchell, K.C. Morris, and Peter Denno, who offered me an opportunity as a guest researcher at NIST and supervised my research. Many thanks also to Dr. Roberto Pacheco and Dr. Alejandro Martins at UFSC, who spent a long time discussing the thesis with me. My good friend Dr. André Luiz Tietböhl Ramos, "O Mighty Guruji", gave me wise advice about the thesis, and helped me in so many ways during my stay in Blacksburg, at Virginia Tech. Ladislau Conceição (CTI, now working for Microsoft) provided critical feedback and help, especially regarding EXPRESS-to-IDL translation.

Several other people helped me in critical things for the success of the doctoral research, including Dr. Carlos F. Bremer (USP and Univ. Aachen); Carlos Pittaluga Niederauer and Rejane Oliveira (CNPq); Fernando Montenegro and Dr. Oscar Lopez (UFSC); Edward Barkmeyer and Neil Christopher (NIST); Eliane Campregher (UNIVALI); Dr. Krishna K. Krishnan, Dr. Mauro J. Atalla, and Nei Mueller (Virginia Tech); Manuel Montenegro (MRE); Marilena Deschamps; and Dr. Rogério Barra (CTI and PDES, Inc.). Still, I am indebted to several other people who helped me in ways that are not directly related to the doctorate. I can't afford to mention everybody (it's a whole new volume), but please accept my deep gratitude. Finally, I thank my students, who are great and keep me motivated to be the best professor and adviser I can be.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

ABSTRACT

Data in engineering applications has been managed using database management systems or dedicated mechanisms embedded in CAx systems. Current industrial competitiveness trends point to the necessity of integrating engineering applications. Two major demands arise: the use of a mechanism to provide for network interoperable access to data, and the necessity of handling data models supported by different paradigms. This doctoral thesis introduces the problems of product data exchange and interoperability among applications using standard formats, and discusses the problem of semantic loss in the translation of product data models for network interoperable access. An analysis of the problems that emerge in this translation, with the objective of assessing the maintainability of data semantics across a distributed network, is performed.

KEYWORDS

Product data exchange; Data sharing; STEP (STandard for the Exchange of Product model data); PDES (Product Data Exchange using STEP); EXPRESS; SDAI (Standard Data Access Interface); CORBA (Common Object Request Broker Architecture); IDL (Interface Definition Language); Interoperability; Engineering Databases; Virtual Enterprises; Industrial Virtual Enterprises (IVE)

FOREWORD

The doctoral research reported in this thesis was part of the “sandwich” program from the Brazilian National Research Council (CNPq), with the credits taken at the Federal University of Santa Catarina, Graduate Program in Production Engineering (UFSC/PPGEP, Brazil). From August, 1994, to November, 1995, the research was conducted at the Virginia Polytechnic Institute and State University (Virginia Tech, USA). From November, 1995, to October, 1996, the research was conducted at the National Institute of Standards and Technology (NIST, USA). At UFSC/PPGEP, the thesis project was submitted and approved in a qualifying examination in June, 1996, and the thesis was defended in December, 1997.

During the literature review, one term paper (Kern 1994), and three congress papers (Kern & Bøhn 1995) (Kern, Bøhn, & Barcia 1996) (Kern, Barra, & Barcia 1996) were published. The language in which the papers and the thesis was written is English. In order to comply with UFSC policies, the thesis was published in Portuguese with the title “Manutenibilidade da Semântica de Modelos de Dados de Produtos Compartilhados em Rede Interoperável”, and this English version was included in the appendix.

CHAPTER 1

INTRODUCTION

> “A message to mapmakers: highways are not painted red, rivers don't have county lines running down the middle, and you can't see contour lines on a mountain.”
> (William Kent, *Data and Reality*, 1978)

1.1 Motivation

Industrial automation technology has improved dramatically over the past decades. CAx systems ("Computer-Aided anything", or: CAD, CAM, CAE, CIM, etc.) have provided engineering applications with high-performance solutions. Integration of these technologies is a major issue for industrial competitiveness. According to Yang:

> "Over the past 40 years, industrial automation has seen dramatic advances in terms of the capabilities and precision of the available technology. From numerical control (NC) in the fifties, through the first design graphics applications and computer controlled production operations in the sixties, Computer Numerical Control (CNC) and Distributed Numerical Control (DNC) in the seventies, and Flexible Manufacturing Systems (FMS) and solid model-based design workstations in the eighties, automation technology has continued to advance and become more sophisticated in order to meet the individual needs of industry. However, as industry moves into the nineties, a new industrial need is becoming the critical problem to solve: the integration of these diverse automation systems (e.g., CAD, CAM, CIM, CAE)." (Yang 1993)

The complex nature of engineering data may hinder the integration of engineering applications. As Wilson (1987) points out, two “stumbling blocks" that prevent the effective integration of CAx systems are:

1. Current CAx systems have been designed to input and output data rather than information; and
2. Current CAx tools operate on different levels of abstraction of the mechanical product.

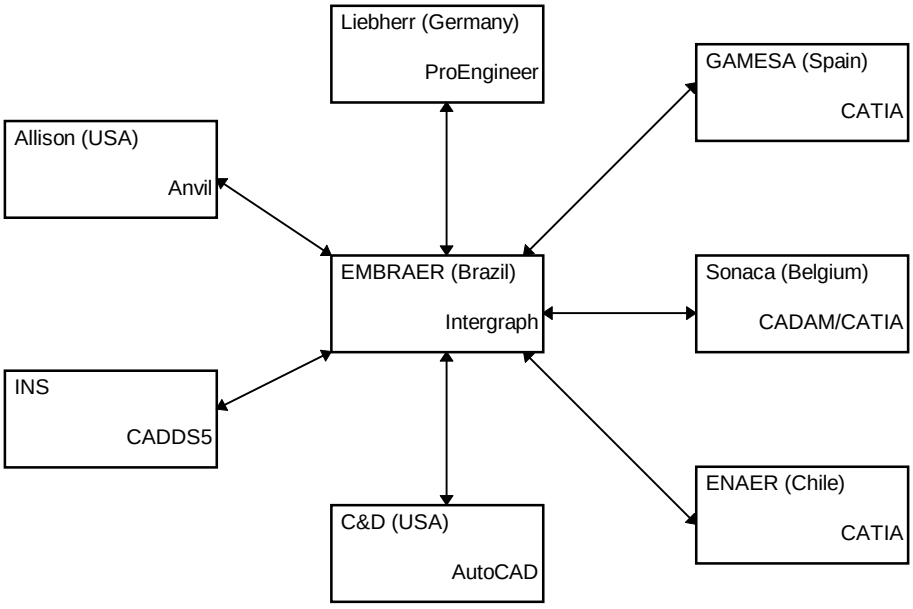


Figure 1 - Illustration of CAD data exchange – Program EMB 145 (Cecchini 1996)

Therefore, information (data with meaning) modeling is a major issue for CAx systems integration. Moreover, data has to be transferred between applications. This creates the need for data translation to fit the receiving system's data model. To deal with this problem, the International Standards Organization (ISO) launched the STandard for the Exchange of Product model data - STEP (ISO 10303-1 1994), aimed at the representation of all information about a product throughout its entire life cycle. STEP allows different applications to exchange information using a standard format. All data models in STEP are normalized (i.e., in conformity with the *normal forms*, described in section 2.1.2) and written in EXPRESS (ISO 10303-11 1994), an "object-flavored information model specification language" (Schenck & Wilson 1994) allowing for the specification of complex data models with multiple inheritance.

Figure 1 illustrates the problem of product data exchange, as it occurred in the following situation: Program EMB145 was a project to build a jet plane at Embraer in partnership with several suppliers, with its first flight in 1995 (Cecchini 1996). The boxes show companies' names and CAD systems. Designs had to be transferred between Embraer and each one of the suppliers, but Embraer used a CAD system (Intergraph) that was different from the several CAD systems used by its suppliers. In this situation, designs transferred from a supplier to Embraer would have to be translated into Intergraph's data model, while a design transferred from Embraer would have to be translated into the supplier's CAD system's data model.

At the time of the project there were some solutions developed for the exchange between two specific data models (in the format of a filter implemented in one CAD

system which translated its data into another CAD system's data model) and between a proprietary and a neutral format (in the format of pre- and postprocessors implemented in each CAD system which translated data to and from neutral formats such as IGES or SET). However, none of these solutions were considered stable, efficient, economic, or otherwise justifiable for adoption in EMB145 data interchange. The solution of choice was to force the suppliers to furnish designs using Embraer's CAD system data model. This implies that suppliers had to buy, train users, and use a version of the same CAD system used by Embraer if they wanted to take part in Program EMB145.

The situation just described was (and still is) very common in the industry. Under the circumstances, it was not possible for each partner to use its best expertise, with its specific CAx system, and still take part in Program EMB145 without re-entering all data into Intergraph before sending for transfer, or from Intergraph after receiving. This redundant work would not be necessary if there was a system that allowed for the conversion of data between the various CAx systems, and for the access to data at a fine level of granularity, i.e., small pieces of data accessed as needed.

The need for fine-grain data access gives rise to the problem of interoperability among applications. Interoperability is defined as "the effective interconnection of two or more different computer systems, databases, or networks in order to support distributed computing and/or data exchange" (Office of Science 1994). Engineering applications are developed under different programming languages, implemented on specific operating systems, in different locations, and support different database paradigms. In order to provide for network interoperable access to data, the Object Management Group (OMG) launched the Common Object Request Broker Architecture (CORBA) whose core is the Object Request Broker (ORB). The ORB "provides the basic mechanism for transparently making requests to - and receiving responses from - objects located locally or remotely without the client needing to be aware of the mechanisms used to represent, communicate with, activate or store the objects" (OMG 1993).

Product data in such a network interoperable environment is represented in EXPRESS, and might be accessed and delivered through calls written in the Interface Definition Language (IDL), a declarative language with a syntax resembling that of C++, in which object interfaces are published according to the CORBA architecture. ISO has published a draft of the IDL binding (ISO 10303-26 1997) to the Standard Data Access Interface--SDAI (ISO 10303-22 1995), the implementation aspect of STEP.

The need for an EXPRESS-IDL mapping, and further mappings to programming languages (since EXPRESS and IDL are not aimed at implementation) arouse

concerns about the maintainability of semantics of product data models being shared in a network. According to Hardwick and Loffredo (1995), the following features of EXPRESS may require encodings or other manipulations to preserve the original information within the native data model: entities, inheritance, primitive types, enumerations, selects, and aggregates. Identifying these constructs as the EXPRESS type system, Kiekenbeck, Siegenthaler, and Schlageter (1995) corroborate the idea stating that the mapping of EXPRESS to another language "requires a comprehensive mapping of the EXPRESS type system to the destination language. Loss of parts of the EXPRESS type system through the mapping would limit the usefulness of the generated code."

1.2 Problem Statement

This thesis addresses the following problem:

> What are the losses in the standards-based sharing of network interoperable product data models which are specific to the EXPRESS-IDL mapping, and how to alleviate these losses?

1.3 Relevance

An estimated 70% of all industrial designs are redesigns (Kiggans 1996). Different applications have to deal with the same evolving design, in different aspects. Since applications use different data models, and may run in different environments, there is a gap that hinders the communication of information about designs.

The integration of engineering applications is critical for the integration of industrial enterprises. As pointed out by Rando and Paoloni:

> "Manufacturers have been successful in implementing very sophisticated point solutions, however, they have been unable to integrate these solutions into enterprise wide systems."
> (Rando & Paoloni 1994)

Enterprise integration comprises a vertical and an horizontal aspect. In the vertical aspect of industrial integration, i.e., *concurrent engineering*, different groups of engineers work simultaneously and collaboratively on a product's different design and manufacturing tasks (Kern & Bøhn 1995). In the horizontal aspect, i.e., *virtual enterprises*, corporations combine their specialties to create a product. A virtual enterprise must be able to form quickly in response to new opportunities and dissolve just as quickly when the need ceases (Hardwick et al. 1996).

This thesis addresses the maintainability of semantics in the sharing of network interoperable product data models, in favor of the integration of engineering applications, which is necessary for both concurrent engineering and virtual enterprises.

1.4 Objectives

The general objective of this thesis is to develop knowledge and techniques for the realization of industrial virtual enterprises.

Specifically, this thesis has the following objectives:

- To present an overview of database management, with emphasis on the adequacy and applicability of database tools and techniques to engineering applications;
- To describe the problem of product data exchange, and a standards-based solution for the problem;
- To present the concept of applications' network interoperability, and a standards-based solution for the integration of product data models for network interoperable access;
- To categorize the opportunities for information loss in the sharing of network interoperable product data models; and
- To present an evaluation of the semantic loss in the translation of standards-based product data models aimed at the network interoperable access: to describe the nature of the loss, possible actions to alleviate it, and identify the losses which are intrinsic to the choice of languages that represent product data models.

1.5 Scope

This section describes the scope of the thesis. It presents the coupling of STEP and CORBA as the standards-based framework adopted for the integration of network interoperable product data models and introduces technological issues related to this coupling. An outline of the specific topic developed in this thesis is also presented.

The adoption of a standard for the exchange of product data models (ISO 10303-1 1992), integrated with the use of a standard for network interoperability (OMG 1995), allows for the network interoperable access to product data models. STEP provides the format for product data models and the language-independent Standard Data Access Interface (ISO 10303-22 1995). CORBA provides "a set of object-oriented interfaces that support the construction and integration of object-oriented software components in heterogeneous distributed environments" (Brando 1996).

The STEP-CORBA coupling defines how to represent and access data. The selection of specific data management techniques, such as transaction management, concurrency control, and version management, is beyond the scope of this integration, and beyond the scope of this thesis. Each application on the network has its own data management implementation, be it a database management system, or a mechanism embedded in a CAx system.

The realization of the integration of product data models depends upon the availability of Application Protocols (APs). An AP is a conceptual schema written in EXPRESS for a certain application domain. The APs are part of STEP. They are the conceptual models meant to be implemented in conjunction with one of STEP's implementation methods (see section 2.2.4).

In this thesis, the implementation method used is the SDAI, a language-independent application programming interface (API) which has a binding defined to CORBA's IDL. STEP objects, written in EXPRESS, are mapped into IDL for the building of interfaces to be published in ORBs to provide for network access. In this mapping, a loss of information may occur. The maintainability of semantics in the EXPRESS-IDL translation is this thesis' specific topic.

For the analysis of the semantic loss in the translation of standards-based, network interoperable product data models (described in chapter 4), the methodology was illustrated by the building of a suite or series of cases, each one focusing on one aspect of the EXPRESS data type. The product data models in this suite were then translated into IDL, subject to the specification of EXPRESS (ISO 10303-11 1994) and the IDL binding to the Standard Data Access Interface (ISO 10303-26 1997). This translation was achieved using partial translators from EXPRESS into IDL, some manual checking and corrections, and custom code where there was no translator available. The IDL translated code was then compared with the original product data models, and an analysis of the semantic loss was made. Actions to alleviate the losses are suggested, and directions for future work regarding the maintainability of product data semantics are recommended.

The research presented in this thesis is based on the following versions of standards: EXPRESS - International Standard, 1994 (ISO 10303-11 1994); SDAI - Committee Draft, 1995 (ISO 10303-22 1995); IDL binding to the SDAI - Committee Draft, 1997 (ISO 10303-26 1997); CORBA - version 2.0, 1995 (OMG 1995).

1.6 Text Organization

This thesis is comprised of five chapters: (1) the introduction, (2) a literature review, (3) a presentation of a standards-based approach to network sharable interoperable product data models, (4) an analysis of the semantic loss in the translation of product data models for network interoperable access, and (5) the conclusion. References and appendix complement the text.

Chapter 2 presents a threefold Literature Review: First, an overview of database management technology, with emphasis on the adequacy and applicability for engineering applications, is discussed. Then, the problem of product data exchange (PDE) is presented, followed by a description of STEP, ISO 10303. Also, an ontology engineering approach to STEP is commented. Finally, network interoperability is introduced, along with a description of CORBA.

Chapter 3 presents a standards-based approach to network sharable interoperable product data models. STEP is adopted as the standard for product data exchange, while CORBA provides the standard specification for application interoperability. The core of this integration is the translation of product data models written in EXPRESS, STEP's data modeling language, into IDL, the language in which objects publish their interface in CORBA.

Chapter 4 presents an analysis of the semantic loss in the translation of product data models for network interoperable access. A suite of data models is developed, each of which focuses on one aspect of the EXPRESS data type. The models are translated into IDL using the mapping specified in the standard binding from the SDAI (ISO 10303-22 1995) to IDL (ISO 10303-26 1997). The semantic loss observed in this translation, defined as the mismatch between the data types of two languages, is analyzed.

Finally, chapter 5 presents a summary and the contributions of this thesis, along with recommendations for future works. Annexes to the thesis include a list of acronyms used throughout the text, an overview of World Wide Web resources used, and a list of current STEP components.

CHAPTER 2

LITERATURE REVIEW

In this chapter, a review of the literature in three fundamental areas of this research is presented: database management, product data exchange, and application interoperability. Given the nature of the research, just a few books and articles have been published in a regular way. As a result, an estimated 50% of the main sources are in-progress standard documents, papers, and postings to technical e-mail lists downloaded from the Internet.

2.1 Database Management

This section discusses important aspects of database management for the realization of network interoperable product databases. Architecture and classification of databases and data models are presented. Database management techniques for business data and for engineering data are discussed and compared.

2.1.1 Database Systems Architecture

Date (1986) defines *database* as "a collection of stored operational data used by the application systems of some particular enterprise." However, databases involve complex concepts and systems. One can “view” a database in different ways. A more complex definition, a three-level architecture for database systems, was developed by the ANSI/X3/SPARC study group on database management systems (Tsichritzis & Klug 1978):

- Internal level -- it is the closest to the physical storage, concerned with the way in which data is actually stored. Examples of internal view of a database system are COBOL and PL/I user views of a record (Date 1986).
- External level -- it is the closest to the users, concerned with the way data is viewed by individual users. For instance, a data flow diagram of a specific application, referencing data deposits, is an external view of a database.
- Conceptual level -- it is a "simulation level" between the other two, a community user view. Enterprise-wide conceptual data models, such as Entity-Relationship diagrams (Chen 1976) and data structure diagrams, are good examples of conceptual-level rendering of a database.

This three-level architecture is illustrated in figure 2. There is exactly one internal view of the database, representing the database as it is physically stored, and several user views, each one representing some portion of the total database. The conceptual level is a view of the entire database, compatible at the same time with the internal storage structures and the several partial external (user) views.

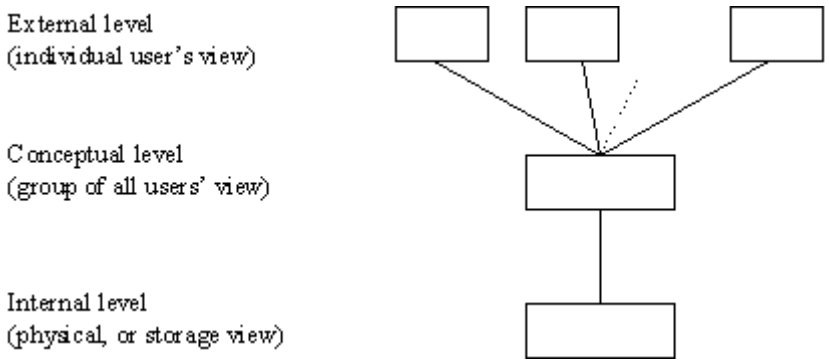


Figure 2 - Three-level architecture for database systems (Date 1986)

2.1.2 Data Models and Database Classification

Data models are used to establish architectural foundations for databases. Although they aim at "modeling the world", Kent (1978) maintains that data models are tools that do not contain in themselves the "true" structure of information. According to Eastman and Fereshetian:

> "The objective in all data and information modeling is to describe a universe of discourse (UoD). ... The task of information modeling is to provide a sound basis for mapping between the portion of the world of interest and a representation of it that can be used as a specification for defining a database and/or applications."
> (Eastman & Fereshetian 1994)

Data models can be generally classified as *semantic* or *representational*. Representational models define how data should be *represented*, and thus imply a choice for a specific technology. They can be classified as:

- Hierarchical: in this data model, the database has a tree structure where each record has only one ascendant or parent, with the exception of the root, which has no parent. Hierarchical data models supported early commercial database management systems (DBMSs).

- Network: this is a generalization of the hierarchical data model, in which the network model allows for records with many ascendants and descendants, as in a graph.
- Relational: with its foundations established by Codd (1970), the relational model supports a database abstraction that can be viewed as a collection of tables and relationships among tables.
- Object-oriented: this paradigm has *abstract data types*, *inheritance* and *aggregation relationships*, and *object identity* as its most fundamental aspects (Khoshafian 1993; Staub & Maier 1995). It supports the rich data model made available through the object-oriented programming languages (OOPL).

While the object-oriented model can be considered to be at the same level as hierarchical, network, and relational data models, it also has a partial correspondence with a higher class of data models called *semantic data models* (Joseph et al. 1991). Semantic data models are conceptualizations at a high level of abstraction. Their application is not constrained to one of the four previous data models. The most well-known semantic data modeling technique is the Entity-Relationship (ER) model (Chen 1976). Semantic data modeling leads to a database design that is less subject to specific data model limitations. However, once it is used, the semantic (also known as *conceptual*) model representation has to be translated into a representational model.

Early DBMSs supported hierarchical and network data models. Although implementations of hierarchical and network databases can still be found, they were replaced largely by relational databases. More recently, object-oriented databases begin to gain DBMS market share.

Relational model

A database is said to be *relational* if it can be viewed as a collection of *relations* (or *tables*, in a less mathematical terminology), and relationships between relations. Each relation is composed by tuples (or table *lines*), all containing the same attributes (or table *columns*).

A relations' design should follow the *normal forms*, which is a set of criteria designed to give tables in the relational model the correct level of granularity. The Boyce-Codd normal form (BCNF) is a normal form that is usually accepted as a good design criterion. It states that, in a relation, any attribute or minimum set of attributes which give access to the content of another attribute *must* be a key, or identifier of the tuple. Figure 3 illustrates the redesign of the relation `Department`, using BCNF, where each tuple of the relation Department have the attributes `Dep#` (department code), `DepName` (department name), `Empl#` (identification of the employee who is

department manager), EmplName (name of employee-manager), Phone (a collection of phone numbers), and the amount of its annual Budget. The BCNF-normalized relations have their keys represented in bold in figure 3. Other keys, known as *foreign keys*, shown in italic in figure 3, allow for the reconstruction of the information contained in the original relation. For instance: Empl#, in Department, holds the (foreign) key to recover the manager's name from Employee.

```
Non-normalized relation:
Department (Dep#, DepName, Empl#, EmplName, (Phone), Budget)

BCNF-normalized relations:
Department ( Dep#, DepName, Empl#, Budget )
Employee   ( Empl#, EmplName )
DeptPhone  ( Dep#, Phone )
```

Figure 3 - Example of a relation and its redesign according to the BCNF

Although normal forms were first used in the scope of relational models, they are also increasingly being used in object-oriented modeling to create *normalized* data models. This is the case of STEP data models, which is discussed in section 2.2.4.2.

Object-oriented model

Objects "cleanly separate external specification from internal implementation" (Blaha et al. 1988). Databases supported by the object-oriented data model store the objects' internal state (their private data), and applications combine the data with the objects' procedures (the external protocol) in response to other objects' calls.

The object-oriented model is considered to be a better metaphor and a more natural way of modeling real-world objects, especially engineering objects (Hardwick & Spooner 1987; Joseph et al. 1991). However, there is much debate about what an object-oriented model for databases should be. Some authors advocate the extension of the relational technology with object characteristics (The committee 1990; Stonebraker 1991), while others hold that an object-oriented approach to databases should be built from scratch (Atkinson et al. 1990). Eastman & Fereshetian (1994) point out that, while hierarchical, network, and relational are general-purpose data models, object-oriented databases (OODBs) have their own special data models, and that no general-purpose data model for OODBs has yet emerged.

2.1.3 Databases for Business Applications

Business data fit in the relational model of flat files or tables (Kern 1994). These are strings and numbers organized in fixed-sized records, with only one version (the current) of data at a time. Concurrent access control can be managed using locking mechanisms: a small portion of the database is made inaccessible when the database is about to be updated, and is released immediately after the update is committed, thus causing a delay which is usually imperceptible by the users.

These *traditional* databases (currently relational, and formerly hierarchical and network) are usually implemented in well-established systems for business data management. Database management systems (DBMSs) provide for application-independent data management and offer several tools for data management tasks, such as concurrency control, database schema evolution, and relational integrity control (uniqueness of keys, validity of foreign keys). Typical users of traditional databases today include banks, which use the relational model to process large volumes of data.

2.1.4 Databases for Engineering Applications

DBMSs supporting the relational data model have been very successful in the management of business data. However, they have a series of shortcomings when used to manage complex engineering applications. Heiler et al. (1987) point out that the engineering design process of defining an initial version and then changing the design is very similar to defining objects and then successively refining them, specifying constraints and building hierarchies of objects. In that sense, the object-oriented model can easily map the designer's mental model of the design objects interfacing with the system design tools and the underlying system facilities (Kern 1994). Examples of tasks that involve manipulating large number of objects include manufacturing automation, multimedia applications, and very large scale integration (VLSI). These objects generally have widely different structures, exhibit complex behavior, and are interconnected by intricate networks. These data, which generally exceed a computer’s virtual memory, need to be shared among several users at different locations and preserved after the processes which generate them terminate. The following section will examine the nature of these data and the resulting shortcomings of conventional database systems in more detail.

2.1.4.1 Nature of Data

Data in engineering applications may include strings, numbers, arrays, bitmap graphics, and objects of different types and sizes, all with complex interrelationships.

This diversity of data formats which do not fit in tables is a challenge for traditional DBMSs. Also, the use of data is much more complex in engineering applications than in business applications. Urban et al. describe the engineering design environment, with an account of the data involved, and the tasks performed during product design:

> "In a large company engaged in the design and manufacture of industrial or consumer products, it is common to see several hundred people, in dozens of disciplines, engaged in development and production activities spanning several years. These decision-makers may be organized into many departments, at locations that may be physically distant. There may be many products, and versions of products, that the company markets. Each product typically consists of many individual parts, some of which may have been purchased in finished or semi-finished form from other companies (vendors). Typically, the development and production is serial, with each department waiting for information from the previous task, and passing it on to the next. There may be some concurrent activities, if two tasks are independent. With the current state of information technology, it is difficult to achieve concurrency between interdependent tasks. In the typical serial process for product engineering, the sequence of major activities may proceed as follows: (1) determination of product specifications; (2) conceptual design; (3) engineering design and analysis; (4) detailed design and blueprint review; (5) manufacturing process planning; (6) quality assurance planning; (7) material requirements planning, production planning and scheduling; (8) production/quality assurance; (9) assembling the products; and (10) packaging and shipping. Each of these activities is a complex aggregation of many activities, which use diverse sources of knowledge, information, and data. All of these activities need to be coordinated and must be performed under constraints set up by decisions made by many departments, each examining different aspects of the product."
> (Urban et al. 1994)

Table 1 presents two somewhat similar classifications of data related to industrial product design.

A characteristic from engineering data that challenges the capacity of traditional database management techniques is the size, and size variability of data. Hardwick et al. report on an experiment that illustrates the growth of engineering databases:

> "We created a STEP database for an axle of the Humvee all-terrain vehicle. The database was created by translating a Pro/Engineer CAD system model of the axle to STEP. The information model for the database contains two megabytes of data, stored as 80,000 instances. If this experiment is scaled up to create a complete database for a motor vehicle, we postulate that the database will be 1,000 times larger. Note that this database covers only the mechanical assembly data for a Humvee vehicle. When STEP expands to include other kinds of data, the number of definitions in the database and the number of data instances increase further."
> (Hardwick et al. 1996)

Table 1 - Classifications of design data

Product definition data	**Modeling data**
Reference: (Encarnação et al. 1986)	Reference: (Zeid 1991)
Geometrical data: Determine shape and dimensions of a product model.	Shape (CAD): Geometric/topologic information, part features.
Representational data: Indicate how an object is to be represented graphically, i.e. color, line thickness, line type, angle at which a model is viewed, etc.	Non-shape (CAD): Graphics data such as shaded images and model global data such as measuring units.
Organizational data: Identify an object or parts of an object during the production process and permits assignment of planning data to the object. These include part number, name, release status, etc.	Design (CAD/CAM) Information from geometric models for analysis purposes.
Technological data: Specify the object more accurately. This includes material data, production data, or calculation information.	Manufacturing (CAD/CAM): Tooling, NC tool parts, tolerancing, process planning, tool design, bill-of-materials.

In summary, the nature (format, size, and relationships) of engineering data is much more complex than the nature of business data. This represents a problem for traditional DBMSs, which are appropriate to manage data in the form of tables only. However, not only the nature of data hinders the use of traditional databases for engineering applications, but also other technological issues, as discussed next.

2.1.4.2 Shortcomings of Conventional Database Systems

Some of the limitations of relational technology for engineering database management are: poor performance, poor modeling power, impedance mismatch, lack of an appropriate transaction mechanism, and lack of support for versioning.

Poor performance

According to a study by Cheng and Hurson (1991), implementing a CAD application on a relational DBMS increased the time for data retrieval by a factor of five compared to a non-database (file-based) implementation. Indeed, Hurson et al. (1993) report that "many CAD tools were built on top of raw file systems to gain efficiency." Joseph et al. (1991) explain this poor performance by comparing a typical relational transaction to a CAD task: while a relational transaction consists of queries and updates in tuples, a CAD task begins by selecting data, then goes on with several operations of recovery and storage, all while navigating through a web of objects.

A CAD query would be too costly for a relational database. The pointers in the normalized objects in relational databases are many times arranged in the wrong

direction for navigation, imposing extra queries (e.g., *intersection curves* and *seam curves* are both specializations which point to *surface curves*, but a *surface curve* instance has no pointer to allow access to the specialized instance of either *intersection* or *seam curve*). Also, the checking of integrity rules performed during each relational update represent an excessive load for a CAD query.

Poor modeling power

Despite its semantic or conceptual character, the entity-relationship (ER) and other popular modeling techniques were conceived in the relational era, and have been used mostly to model databases implemented in relational DBMSs. However, when applied to engineering objects, ER modeling does not provide all the necessary constructs and concepts, and therefore stumbles on the complex relationships, functions, and procedures associated with the data. This complexity, combined with the varied data types used in engineering, suggest that engineering databases and applications would be better served by object-oriented techniques.

Impedance mismatch

There is a critical difference between data models and object manipulation paradigms for languages and conventional databases. Generally, a database's manipulation and data model cannot be mapped perfectly into an implementation language's manipulation and data model, and vice-versa. This difference is referred to as *impedance mismatch* (Joseph et al. 1991). Impedance mismatch makes it difficult to map language-supported models into database models. Even if some rich data modeling is used, much of the programming effort will be spent in translations in order to overcome the differences between language and database models.

Lack of an appropriate transaction mechanism

Transactions in relational databases follow the Update-Commit-Rollback mechanism (Elmasri & Navathe 1994) in which a transaction either succeeds (Commit) or fails (Rollback) completely. If a transaction fails, it has no effect on the database and must be submitted again at a later time. Relational database management assumes that transactions are short and lock small portions of data, therefore the cost of a rollback can be considered low. However, in engineering applications like CAD, a transaction may last longer than a computer session, and the information stored can be very large, making the conventional locking mechanism impractical. According to Kern (1994), a CAD transaction may last for a period longer than a computer session. In this case, a

crash in an uncommitted update would provoke rollback, wasting the expensive partial update.

Lack of support for versioning

While the process of making design changes is meaningful for engineering design, it has been neglected in the development of engineering database environment (Urban et al. 1994). Traditional databases do not support the evolution and management of different versions of data. Schema and data are considered to have only one version in a traditional database: the current version. However, CAD designers must try various versions of design objects until they are able to decide on which one is better. Also, in concurrent engineering, several CAD designers need to exchange incomplete designs, thus allowing for the coexistence of several design versions. Therefore, the relational strategy for versioning is inadequate.

Summary of shortcomings

The relational database structure of relations and tuples is efficient and simple to use for most applications known as "data processing." Nevertheless, they are not complete and efficient enough to represent and manage engineering data.

2.1.4.3 Database Requirements for Engineering Applications

In order to appropriately support engineering applications, database systems should have the following characteristics: a rich data modeling; environment support for version management; navigational and query access; seamlessness; an appropriate transaction mechanism; and the capability of object sharing among application systems.

Rich data modeling

Joseph et al. (1991) maintain that the modeling power of object-oriented programming languages, usually treated in the transient memory, should be extended to applications that need to deal with persistent data. Application data and relationships can be modeled in a natural manner using object-oriented concepts. The extension of this capability will benefit applications which need to deal with persistent data.

Data modeling characteristics that are needed in engineering data management include (Hardwick & Loffredo 1995; Catell 1991): the ability to manipulate unusually complex data models, procedure encapsulation, composed objects, multimedia

objects, and relationships of hierarchy. These characteristics are poorly treated, or not treated at all, by traditional database systems.

Support for versions

Engineering applications need to support different versions of their data. This is because design is a trial-and-error activity in which design objects evolve, assume different requirements, and are abandoned or resumed. Each stage of the design process may generate several alternative versions which can be evaluated, compared, and potentially chosen as appropriate design solutions. Therefore, it is important that the database can support the existence of multiple concurrent versions.

Navigational and query access to objects

Objects in large engineering applications are organized in complex object graphs. It should be possible to navigate those graphs after they are stored in a database, as well as recover data using queries.

Seamlessness

Seamlessness means the opposite of impedance mismatch. The integration of the database with the rest of the environment should not obstruct the rich data modeling discussed above.

Kaplan & Wileden (1996) assert that object-oriented database technology virtually eliminates impedance mismatch, resulting in a significant evolution of the underlying models and languages used in information systems applications. Wood (1992) reports on some OODBs in which the same language is used for data manipulation and application programming.

Appropriate transaction mechanism

Conventional concurrency control and transaction mechanism should be supported (Joseph et al. 1991). Long transactions need a different transaction mechanism, since locking an object during a long period is undesirable.

Some new techniques for transaction management are: *check-in/check-out*, which checks an object out from the global to a private workspace, updates it, and then check in again, enforcing concurrency control; and *hypothetical transactions* (Kim et al. 1990), a kind of what-if experiment, which always abort.

Object sharing among application systems

Application systems for engineering, like CAD/CAM systems, have been developed with their own built-in storage systems. Goh et al. argue that

> "... the end user is concerned with using the CAD/CAM tools for design, analysis and development and is usually not interested in aspects such as data structures, access methods, and so on. This has led to incompatible data representations and consequently, there arises a difficulty in sharing and exchanging data between applications." (Goh et al. 1994)

Engineering applications need to communicate and exchange data, regardless of their implementation languages, between (Encarnação et al. 1986):

- Different technical sectors of design (bodywork design, engine design),
- Design, production preparation, and production,
- Manufacturers and suppliers or branch factories,
- Time-sequential development of models,
- Different CAD/CAM systems, and
- Different versions of CAD/CAM system.

Conclusion

Database management for the so-called next-generation applications have been object of intense research. From the requirements above, this thesis focus on object sharing among application systems, which is introduced in the two remaining sections.

2.2 Product Data Exchange and Sharing

Product data exchange (PDE) refers to the task of expressing and transferring information about a given product in digital format. In this section, the reasons and alternatives to PDE are presented; the existent technology for PDE is summarized; and an international standard is presented, allowing for not only PDE, but also product data sharing, i.e. the simultaneous access to product data by several engineering applications.

2.2.1 The Quest for Product Data Exchange

The need for PDE arises when product data has to be transferred between different applications. This need may be caused by demands of communication between

different engineering teams, departments, or companies, for purposes of design, analysis, manufacturing, or product support.

Encarnação et al. refer to the need for product data exchange and sharing in CIM:

> "The term CIM (Computer-Integrated Manufacturing) has become a buzz-word in the late 1980s for the attempts to connect properly the various computer support systems used in these various areas into a well integrated system in which all information, once it is produced, becomes immediately available at all places where it is needed." (Encarnação et al. 1990)

Product data is usually generated and manipulated using a vendor system, and stored in a proprietary format, according to a specific data model. There is a big number of computer-aided systems (CAx) for engineering, and the interchange of information among them may become expensive and time consuming. Hardwick & Loffredo (1995) identify as successful data models those of CATIA (Dassault), CADAM (Lockheed) and Unigraphics (McDonnel Douglas), and observe that users may become locked into the modeler of a single vendor, which is undesirable.

The demand for cost reduction is another motivator for PDE. Costs associated with data reentry do not add any value to the product. Redondo (1996) presents examples that characterize the problem:

- Shell (the oil company) estimates that interfaces (between systems) are responsible for 25-70% of the system development costs.
- BMW (the auto company) costs associated with CAD data exchange with suppliers are about DM 10 million (approximately US$ 6.5 million, 1997), considering only data conversion direct costs.

Demand for the effective communication of product information is illustrated in figure 4, from Project EMB145 of the aeronautic company Embraer. EMB145 is a jet plane developed in partnership with several suppliers, with its first flight in 1985. A total of 12120 designs were produced. Embraer transferred about 1500 designs to its partners, and received 3000. The traditional solution was adopted for the exchange: suppliers were required to furnish data translated into Embraer's CAD system data model (Cecchini 1996).

A general classification of industry needs for PDE is presented in (Digital Equipment 1992), illustrated and exemplified in figure 5.

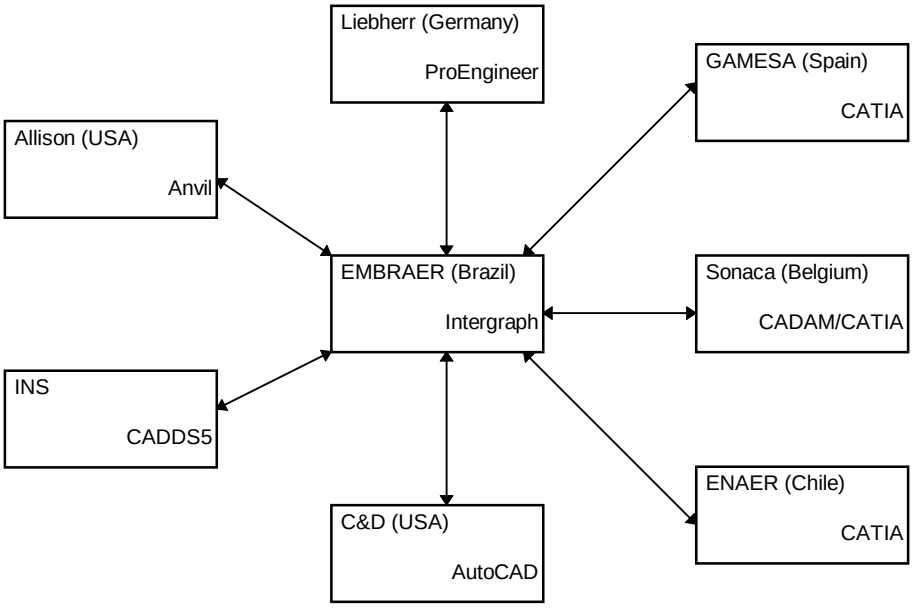


Figure 4 - CAD data exchange in the EMB145 airplane project (Cecchini 1996)

A Classification of Industry PDE Needs

- Product design needs
  (Shorter development cycles, improved quality, cost reduction, ...)
  - Needs of groups of design engineers
    (Data sharing, navigation at different granularity levels)
  - Needs across product life cycle
    (Management of work planning, configuration and versions, releases, ...)
  - Needs between enterprises
    (Shared, controlled, secure access to data)
- Software technology needs
  (Freedom to focus on each one's specialty)
  - Data storage technology requirements
    (Handling of heterogeneous, distributed environments)
  - Application development
    (Incorporation of new technology, isolation of applications and data storage)
  - System integration technology
    (Integration across hardware and software platforms, different conceptual models)

Figure 5 - Classification of industry PDE needs

According to Encarnação et al. (1986), the advantages expected from the exchange of product data can be summarized as follows:

- Reduction of throughput times;
- Minimization of errors;
- Clarity due to improved quality;

- Improved access to information;
- Reduction of repetitive work;
- Reduction of administrative costs; and
- Availability of standardized and purchased parts.

2.2.2 Approaches to Product Data Exchange

Manual re-entry of data, and standardization on a single system have been used as informal alternatives to PDE. This may be feasible in some situations, for instance: finite element analysis needs only 10-20% of detail of a full design (Digital Equipment 1992). However, most data transfers between engineering applications are feasible only through data translation.

There are two approaches to data translation, as illustrated in figure 6: *direct* translation, and *neutral format* translation.

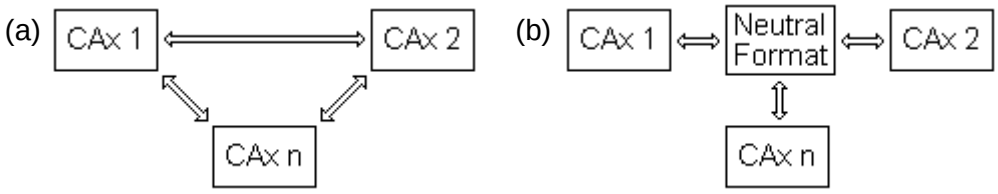


Figure 6 - Direct (a) and neutral format (b) data translation

While direct translators offer better opportunities for optimizations and for capturing idiosyncrasies of the CAx systems involved, every time a new CAx system version is acquired, translators to and from all other systems are required. The usage of neutral format translation presumes the existence of a public domain, agreed-upon neutral data format. Every new CAx system needs a pre- and a postprocessor to translate to and from the neutral format. Hence, being *n* the number of CAx systems in use, there is a need of *n(n-1)* translators for direct translation, and *2n* translators when using neutral intermediate translation.

One of the major issues in product data translation is that of completeness in data conversion. For instance, a circular arc may be represented as (Owen 1993):

- Center, radius, start angle, swept angle;
- Center, radius, start angle, finish angle;
- Center, radius, start point, end point;
- Center, radius, bulge factor;

- Ellipse (major axis = minor axis);
- General conic;
- Rational B-spline curve (with control points);
- Rational B-spline curve (flagged as circular); and
- Polyline.

Any alternative would be accepted for graphical representation, but some application could demand one of the alternatives, requiring a conversion from the original format to the target format. In order to allow for this equivalence, or completeness in the conversion, a neutral format data model has its size significantly augmented.

Table 2 accounts for the main advantages and disadvantages of direct or neutral format translation.

Table 2 - Comparison between direct and neutral format data translation

Direct translation	Neutral format translation
Advantages	
Better opportunity for accurate translations One (direct) translation	Only two translators for each CAx system Independence of supplier Possibility of use for archiving, protection against obsolescence
Disadvantages	
Explosion in the number of translators (n^2 - n) Expensive implementation and support Developer needs to keep experts in other systems	Long time to develop Prone to limitations on coverage Two translations, double opportunity for errors

2.2.3 Product Data Exchange Standards Evolution

Early data exchange specifications focused primarily on geometrical data. Among these were proprietary specifications like Autodesk's DXF, and national standards such as IGES (United States), SET (France), and VDA/FS (Germany). Encarnação et al. (1990) affirm that the “industrial application of computer-aided design originally concentrated mainly on drafting (two-dimensional representations) and on approximate representation of three-dimensional objects with wireframe models."

IGES (Initial Graphics Exchange Specification) was first released as an ANSI standard in 1981. Encarnação et al. (1986) observe that:

> “... flaws existed not only as deficiencies of the IGES translators, but also within the IGES standard itself. ... Problems with IGES relate to its large file size, file organization, and the entity set. Problems with IGES translators have been misinterpretations of the standard, programming errors, and implementations of different IGES entity subsets.”

The nonexistence of complex and realistic test cases was another deficiency. IGES reflected CAD systems of the 1970's. In 1983, IGES organization formed a committee to determine what could be done to meet the new needs of CAx applications. Laurance notes that:

> "... in many cases we have blamed IGES for the shortcomings that were inherent in the CAD system themselves - a case of blaming the messenger."
> (Laurance 1994)

SET (Standard d'Exchange et de Transfert) was based on IGES, but with a radically different file format, in which data can be shared between records, reducing file size significantly (Wilson 1993). The first major release of SET was in 1984.

The VDA/FS (Verband der Deutschen Automobilindustrie - Flachenschittstelle) standard format was designed to allow for the exchange of free form surface data between German automobile manufacturers, handling only a narrow section of the CAD spectrum but, within these confines, it is well applicable (Wilson 1993).

PDES, an early proposed successor to IGES, was later realigned in support of STEP, the international STandard for the Exchange of Product model data, ISO 10303 (ISO 10303-1 1992). Wilson reports that the committee formed to begin work on PDES recommended that:

> "... the PDES specification should be developed on three levels, roughly corresponding to the three level schemas used for databases. Each application area should define its own view of the information to be transferred, corresponding to a sub-schema definition in a database. ... The separation into these levels accomplishes several goals. It enables application experts to concentrate on the 'meaning' of their data without having to be concerned with how it is embedded in a physical medium. It focuses the overall data structure at one level which can be (conceptually) divorced from specific applications and file formats. A number of different formats can be specified to meet differing transfer requirements in a similar manner to the different language embeddings of the graphics standards like GKS or PHIGS."
> (Wilson 1993)

Table 3 presents a comparative description of several product data exchange standards, where ACIS is the industry-standard geometric modeling kernel of AutoCAD, CADKEY, MicroStation, PE/Solid Designer, and SolidEdge (Redondo 1996).

Table 3 - Comparison among product data exchange specifications (Redondo 1996)

	IGES	SET	VDA/FS	STEP	ACIS
Scope	• Wireframe models • Surface models • Solid models • FEM models • Technical drawings	• Wireframe models • Surface models • Solid models • Technical drawings	• Surface models	• Product models for the entire life cycle	• Wireframe models • Surface models • Solid models
Specification characteristics	• Collection of entities • File format	• Collection of entities • File format	• Collection of entities • File format	• Formal specification of a product model • Formal definition of file syntax	• Geometric modeler kernel

As an illustration of preliminary results, a survey on concurrent engineering based on standards, performed by the Packard Commission, and reported by Redondo (1996), yielded the following results:

- Reduced project changes up to 50%.
- Reduced product development leadtime up to 60%.
- Reduced re-work up to 75%.
- Reduced manufacturing costs up to 40%.

2.2.4 ISO STEP

STEP is the unofficial name of ISO 10303, the STandard for the Exchange of Product model data. It is the biggest standardization effort ever, being developed by Technical Committee 184 (TC 184, Industrial Automation Systems), Sub-Committee 4 (SC4, Manufacturing Languages and Data) of ISO.

ISO TC184/SC4 first met in 1984, in Washington, D.C., United States. At that time, many lessons had been learned from previous standards. According to Owen (1993), there was a consensus that none of the existing initiatives was acceptable to be used as an interim international solution. However, existing standards such as IGES, SET, and VDA/FS would be used in support of STEP development:

> "Technical work will be accomplished by existing and future national projects, organizations, and resources which will be coordinated and monitored by the SC4 committee. SC4 will set design objectives, establish priorities, arbitrate differences, and ensure that objectives are met and consistency is maintained."
> Resolution 1, July 1984, Washington (Owen 1993)

By 1996, more than US$ 500 million had been spent on STEP's development and transfer to industry (Redondo 1996).

STEP has, among its goals:

- To provide an international and multi-disciplinary consensus description of product data that supports life-cycle (concept to retirement, not just engineering to manufacturing) functions.
- To enable the capture of information comprising a computerized product model in a neutral form without loss of completeness and integrity.
- To enable product data exchange and sharing between: different software applications and platforms, different organizations involved in the product lifecycle, and physically dispersed sites.

The purpose of the standard is "to prescribe a neutral mechanism capable of completely representing product data throughout the life cycle of a product" (Edwards-Iwe 1993). STEP data models encompass virtually all aspects of design, analysis, manufacturing, documentation, and retirement of products. Trapp (1993) enumerates some key features that make STEP better than previous standards:

- It encompasses all product data;
- It is founded in information modeling languages;
- It separates the information model from the data instances;
- It will be implemented in accordance with application (specific) protocols; and
- It requires conformance testing of implementations.

The next subsections approach STEP's architecture and development; the building of normalized, object-oriented information models; STEP's implementation, which is separate from information modeling; the possible impact of ontology engineering in STEP development; and the future of the standard.

2.2.4.1 Architecture and Development

STEP is a complex standard with huge-sized documents, and was developed as if it was a database itself, adopting the ANSI/SPARC architecture for database systems (Tsichritzis & Klug 1978). Yang et al. state that:

> "The STEP architecture is a unique solution to a very complex problem: the meaningful communication of product information between unspecified industrial product automation systems. It is also an innovative solution with respect to the design and use of conceptual models".
> (Yang et al. 1993)

The development of a conceptual schema language and a framework for application protocols is regarded by Koch (1992) as "the most important achievement of STEP compared to earlier standards." The adoption of the architecture for STEP development reclines on a number of major design goals (Owen 1993):

- Completeness: STEP should provide for the complete representation of a product, for both exchange and archiving;
- Extensibility: STEP should provide a framework into which extensions of domain can be built;
- Testability of additions: Document releases should be subjected to peer review and, if possible, undergo further testing by being implemented;
- Efficiency: STEP should be efficient in terms of both file size and computer resources needed for processing;
- Compatibility with other standards: As far as possible, STEP should be compatible with other standards in order to ease migration from existing standards;
- Minimal redundancy: There should be only one way of representing a particular concept;
- Computing environment independence: No hardware/software dependence should exist in STEP;
- Logical classification of data elements: STEP should define (standard) subsets for implementations as it would clearly be a large standard; and
- Implementation validation: A framework for conformance testing should be part of the standard.

Owen addresses the issue of STEP subdivision:

> "It was recognized that the standard was going to be very large, and that implementations were needed of subsets. The concept of application protocols, already being developed in the IGES/PDES Organization, was brought forward to ensure that the earlier practice of vendors choosing, on an *ad hoc* basis, which constructs to implement would not occur for STEP. Coupled with the need for a more explicit framework for the product information models and the need for development of sections of STEP to progress at different rates, STEP was divided into a number of classes of parts, with well-defined relationships between them." (Owen 1993)

The ANSI/SPARC three-level architecture is paralleled by the STEP architecture of application, logical, and physical layers (Fowler 1995; Hardwick et al. 1995; Owen 1993):

- Application layer: External level, comprised of information models specific to an application area, developed by experts. These are the Application Protocols (APs), STEP Parts 201-1199.
- Logical layer: Conceptual level, a library of product information models called Integrated Resources (IRs), Parts 41-99 and 101-199, dedicated to describe all domains of interest in a unique, unambiguous way.
- Physical layer: Internal level, a series of Implementation Methods, Parts 21-29, dealing with the mapping of the application schemata onto a specific computer technology.

Table 4 presents the various STEP document series and the correspondent part numbers.

Table 4 - STEP documents series and architectural organization

STEP document series	**Document part numbers**	**Architectural layer**
Introductory	single digits	
Description Methods	10 series	
Implementation Methods	**20 series**	**Physical layer**
Conformance Testing Methodology and Framework	30 series	
Integrated Resources (IRs)	**40 and 100 series**	**Logical layer**
Application Protocols (APs) **Application Interpreted Constructs**	**200 series** **500 series**	**Application layer**
Abstract Test Suites	300 series	

The layered documents and the other infrastructure and testing series are summarized next:

Introductory documents series

Introductory documents describe the overall structure of the standard. Only Part 1, "Overview and Fundamental Principles" (ISO 10303-1 1992), have been developed so far.

Description methods series

This series comprise documents related to the languages and methods used to create standard representations of product data.

Part 11 (ISO 10303-11 1994) describes EXPRESS, a Pascal-like, declarative, object-flavored information modeling language. It is composed of constructs such as entities, types, rules and functions.

Description methods series includes also Part 12, EXPRESS-I, an instantiation language used to specify test data, and Part 13, "Architecture and methodology reference manual."

The Physical layer: Implementation Methods

The specification of Implementation Methods (how to exchange data) separated from the Information Models (how to represent data) allows for the development of new implementation mechanisms, as the computer technology advances. This is another outcome of the adoption of ANSI/SPARC DBMS framework for STEP development.

Four different levels of implementation were identified (Fowler 1995):

- Level 1: Passive file transfer;
- Level 2: Active file transfer;
- Level 3: Shared database access; and
- Level 4: Integrated knowledgebase.

In passive file transfer, a STEP pre-processor in the sending system converts the product data model to an ASCII physical file format. This file is transferred to a receiver system, which converts standard data into its internal representation using a post-processor.

Implementation level 2 is an extension to Level 1, where the physical file is converted to a "working form." According to Fowler (1995), since most file-based implementations use this approach, the distinction between levels 1 and 2 have disappeared.

Levels 3 and 4 represent a significantly different technical approach: they address not only static data *exchange*, but dynamic data and information *sharing*.

Part 22, the Standard Data Access Interface (SDAI) (ISO 10303-22 1995), is the core document for the enabling of shared database access. It describes an Application Programming Interface (API) for access and manipulation of STEP data. SDAI describes, in EXPRESS, a set of operations to store, manipulate, and share data. Since EXPRESS is not intended to be implemented, SDAI is broken up in several bindings to programming languages (C, C++, etc.) and the CORBA Interface Definition Language (IDL) (OMG 1995) for network interoperable access.

Knowledgebases are object of basic research and development (Higa et al. 1992; Meis & Ostermayer 1996), with wide areas of open research issues, not exclusively of STEP.

Conformance testing methodology and framework

Conformance testing is defined as "the testing of a candidate product for the existence of specific characteristics required by a standard in order to determine the extent to which that product is a conforming implementation" (Owen 1993).

Standard testing methodologies were specified for STEP because no two implementations interpret the standard in exactly the same manner. The goal of the STEP conformance testing is to ensure (Wilson 1993):

- Repeatability: Results are the same whenever they have been made;
- Comparability: Results are the same wherever they have been made; and
- Auditability: Tests procedures can be confirmed as having been correctly performed by a review of the test records.

Some other standards previous to STEP had conformance testing, but they were only available several years after the standard's publication. Therefore, if early independent testing was required for an implementation, the user had to undertake it himself (Owen 1993). In STEP, an implementation is tested for conformance to the standard using the conformance testing methodology and framework, and the abstract test suite associated with the application protocol.

Conformance testing in STEP includes "General concepts" (Part 31), "Requirements on testing laboratories and clients" (Part 32), "Abstract test methods for Part 21 implementations" (Part 33) and "Abstract test methods for Part 22 implementations" (Part 34).

The Logical layer: Integrated resources

Integrated Resources (IRs) are divided into *Generic* (40 series) and *Application* (100 series). This division reflects the fact that some resources are generic in nature, while others are suited for a range of applications. IRs are the first class of information models in STEP. They provide a conceptual model, written in EXPRESS, which is independent of any specific application or implementation.

Integrated Resources include “Fundamentals of product description and support” (Part 41), “Geometric and topological representation” (Part 42),

“Representation structures” (Part 43), “Product structure configuration” (Part 44), and “Visual presentation” (Part 46).

The collection of all IRs is a set of reusable, interconnected, unambiguous schemata that serve as the basis for the building of Application Protocols, the second class of information models in STEP.

The Application Level: Application Protocols

Application Protocols (APs) are "the bulk of the standard" (Laurance 1994). They define the context, scope and information requirements for a designated specific application context and specify elements of the IRs that are used to meet these requirements (Yang et al. 1993). APs are idealistically separate from each other (Wilson 1993).

STEP implementations are based on an Implementation Method and a conceptual schema in an AP.

Application Interpreted Constructs

AICs provide semantic integration between APs, when identical requirements are shared by two or more APs. Burkett (1993) states that an AIC is like a mini-AP: it specifies the structures and interpretations for a narrow and specific context.

Abstract Test Suites

Abstract Test Suites are designed to be developed for each AP in STEP. They provide the set of abstract test cases to be used during conformance testing of any implementation of the AP. The abstract test cases are derived from the conformance requirements section of the AP and the test purposes documented in the abstract test suite. Abstract Test Cases are both human-readable and computer processable. They are written in EXPRESS-I.

The development and balloting process

STEP documents are developed independently from one another, by distinct working groups. Each one undergoes a process of balloting for approval. It takes typically two iterations of the balloting process before a Part gets voted to Draft International Standard (DIS) status (Wilson 1993).

The process is administered by the National Institute for Standards and Technology (NIST). Development status is annotated as: working draft, project draft, released draft, technically complete, editorially complete, and ISO Committee Draft (CD) or DIS, before being voted as International Standard (IS). Figure 7 presents the

twelve documents present in the first released of STEP. For an extensive list of STEP standard documents, see the appendix.

Introductory	**Integrated resources**
Part 1 - Overview and fundamental principles	Part 41 - Fundamentals of product description and support
Description methods	Part 42 - Geometric and topological representation
Part 11 - The EXPRESS language reference manual	Part 43 - Representation structures
Implementation methods	Part 44 - Product structure configuration
Part 21 - Clear text encoding of the exchange structure	Part 46 - Visual presentation
Conformance testing methodology and framework	Part 101 - Draughting
Part 31 - General concepts	**Application Protocols**
	Part 201 - Explicit draughting
	Part 203 - Configuration controlled design

Figure 7 - The first release of STEP as an International Standard (in May, 1994)

The next subsections extend the presentation of the documents which belong to the three-level architecture: information models (logical and application layers) and implementation methods (physical layer).

2.2.4.2 Information Modeling in STEP

Regarding the advances of STEP in comparison to earlier standards, Laurance (1994) states that “a central tenant in the STEP credo is that IGES is deficient in that it does not contain the semantics of the information, only the data.” STEP focuses on the use of information in the product design process. It tries to capture the semantics by looking at the context in which product information is used.

***Information* or *Data* Modeling**

The terms *information modeling* and *data modeling* are used interchangeably at times, and the difference between them may be subtle. The former defines what is to be done, while the latter defines how it is to be done in a particular implementation environment (Wilson 1993). While relational data modeling, for instance, is bound to allow for the construction of schemata for relational databases, product information modeling represents product data regardless of the implementation technology.

An information model is defined by Wilson as:

> "... an implementation independent specification of the entities defining individual pieces of information necessary for some enterprise, the relationships among the entities, and the constraints on and between the entities and relationships."
> (Wilson 1993)

Or, more plainly, information modeling is defining all the meanings and then selecting a unique word for each meaning (Wilson 1993).

STEP product data models are normalized. Normalizing the data models in STEP is a necessary but expensive characteristic of the standard (Hardwick et al. 1996). It is necessary because manufacturing data needs to be shared by as many applications as possible. It is expensive because normalization often makes a model harder to process, since information about an object may be distributed in several tables.

Similarly as in a relational database, engineering and manufacturing applications need to access data at different levels of granularity. Normalized models allow for this selective access, however carrying some typical problems. Figure 8 presents the entity `product` from STEP. Hardwick et al. observe that

> "All of the entities that describe a product (including its geometry) may be found from this entity, but the navigation can be complex because the pointers in a STEP model describe constraints so they are sometimes arranged in the wrong direction for navigation."
> (Hardwick et al. 1996)

```
ENTITY product
     id   : identifier;
     name : label;
     description     : text;
     frame_of_reference    : SET [1:?] OF product_context;
END_ENTITY
```

Figure 8 - The STEP entity `product`

The EXPRESS language

Choosing STEP as the standard for product data exchange implies using EXPRESS as the data description language. EXPRESS allows for the building of conceptual schemata based on the Entity-Relationship model and constraint-specification constructs (Owen 1993).

Table 5 presents characteristics of the EXPRESS language.

EXPRESS was designed to be STEP's modeling language because no other language seemed to be appropriate to represent the richness of product data models. Other languages and methods had been tried before, such as NIAM and IDEF1X (FIPS 1993), but they demonstrated lack of semantic adequacy for design data (Eastman & Fereshetian 1994).

Table 5 - EXPRESS characteristics

EXPRESS is:	EXPRESS is not:
a Data Description Language	a Data Manipulation Language
technology independent	a methodology
“object-flavored” (but not restricted to object-oriented systems)	a programming language
entity-centered	
Pascal-like	
human-readable, computer-processable (but non-executable)	

The purpose of EXPRESS is to describe the characteristics of information that someday might exist in an information base (Schenk & Wilson 1994). An EXPRESS model can be mapped to multiple data processing technologies and can be used as the common basis for product data exchange between existing and future engineering systems (Hardwick & Loffredo 1995).

EXPRESS (ISO 10303-11 1994) includes a graphic version, EXPRESS-G. Figures 9 and 10 show representations in EXPRESS and EXPRESS-G. Also, there is an ever-growing number of EXPRESS extensions and variations, such as EXPRESS-I (an instantiation language), EXPRESS-C (a dynamic modeling language), EXPRESS-M (aimed at the interoperability of Application Protocols), and EXPRESS-X (allowing to define mappings between information models written in EXPRESS).

Classification of Information Models in STEP

Information models in STEP are built using a bottom-up approach. According to Yang (1993), STEP makes use of abstraction as the principle integration mechanism. This results in a hierarchical conceptual schema that permits data constructs to be “re-used” in different contexts. Each information model comprises one or several interconnected schemata.

STEP information models are:

- Integrated Resources (IRs): generic information models, not sufficient to support the requirements of specific applications. They provide the basic constructs for the representation of all product information within STEP;
- Application Protocols (APs): information models designed for specific applications, built using the basic constructs from the IRs; and
- Application Interpreted Constructs (AICs): groups of modeling constructs shared by different APs.

Integrated Resources

IRs, the first class of information models in STEP, are divided into *Generic* (40 part series, general purpose) and *Application* (100 part series, concerning a range of applications).

The collection of all IRs is a set of reusable, interconnected, unambiguous schemata. A single concept is represented only once within the IRs.

Figure 9 shows an example of a general-purpose geometric entity present in the IR *Geometric and Topological Representation*, Part 42 (ISO 10303-42 1992). Here, the entity `path` inherits properties from `topological_representation_item`, and it is a supertype for three other kinds of specialized `paths`. It has a list of edges as attribute, and the clause `WHERE` specifies a rule that makes the end vertex of each edge equal to the initial vertex of the next edge.

```
ENTITY path
   SUPERTYPE OF (ONEOF(open_path, edge_loop, oriented_path))
   SUBTYPE OF (topological_representation_item);
   edge_list  : LIST [1:?] OF UNIQUE oriented_edge;
 WHERE
   WR1: path_head_to_tail(SELF);
 END_ENTITY;
```

Figure 9 - Example of an entity in an IR

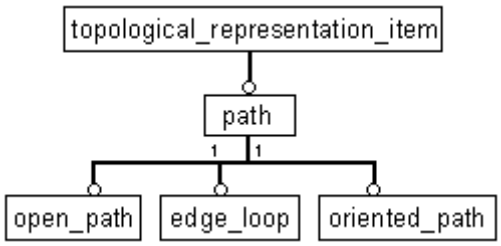


Figure 10 - EXPRESS-G diagram for the entities showing in figure 9

Part 42 is divided into three sections -- geometry, topology, and geometric shapes. Figure 11 presents a hierarchical view of a portion of the geometry section in Part 42: the entity `curve` and its specializations.

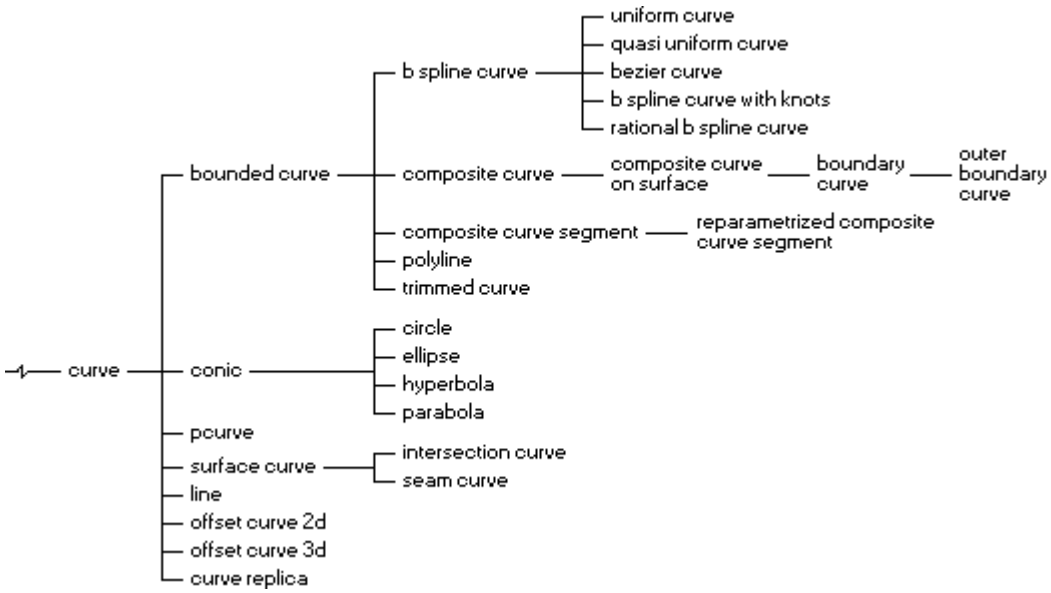


Figure 11 - Entity `curve`, and specialized entities in Part 42 (ISO 10303-42 1992)

IRs serve as building blocks for the Application Protocols (APs), discussed next.

Application Protocols

APs are the most important and, by far, the largest class of STEP Parts (Fowler 1995). They are the STEP information models that are intended to be implemented. APs define all the information required for a specific application domain, and they are ideally independent of one another (Wilson 1993).

The concept of APs was developed to avoid arbitrary implementations of only parts of the standard, as it is the case with IGES. Without APs, CAx vendors would be free to implement translators comprising non-standardized subsets of STEP. Sauder et al. comment on the building of APs:

> "Support for specific application areas is provided by application protocols (APs) which clearly and unambiguously describe all data needs for a particular industrial application. A consistent representation of common data needs between APs is maintained by reusing, as building blocks, a set of general data specifications called Integrated Resources (IRs). This approach is practical since there are overlapping data needs among both the business processes and industry disciplines (i.e., electrical, mechanical, civil engineering)."
> (Sauder et al. 1994)

The building of an AP is similar to that of a database design, comprising four stages:

1. Define scope (AAM--Application Activity Model);
2. Specify information requirements (ARM--Application Reference Model);
3. Generate the EXPRESS model (AIM--Application Interpreted Model); and
4. Establish conformance requirements.

Figure 12 illustrates the usage of several documents in the building of an AP. The four stages are described next.

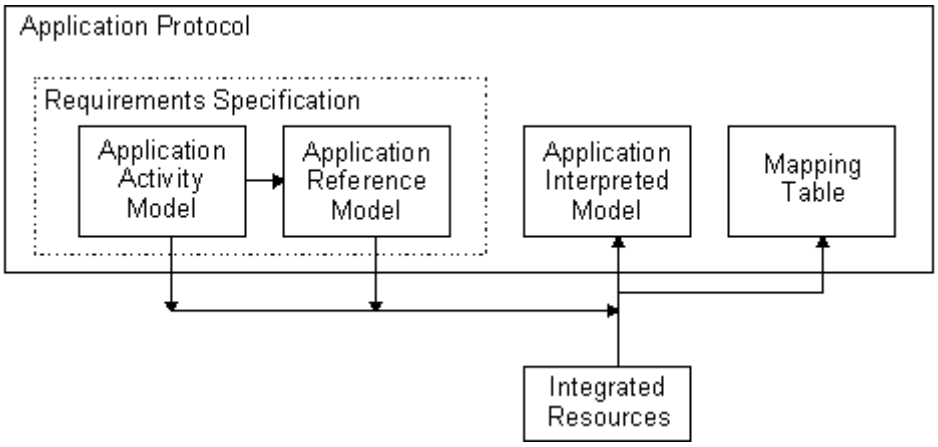


Figure 12 -Application interpretation, the process of building STEP information models

In the *first stage*, the scope of the model is defined in a document called the Application Activity Model (AAM). It provides an initial evaluation of the standard in terms of an application's processes and information flow.

In the *second stage*, the processes identified in the AAM have their information requirements defined in a document called the Application Reference Model (ARM). The ARM specifies the context-specific information to be communicated via the AP. ARMs are defined and documented using NIAM, IDEF1X, or EXPRESS-G. The IRs are not considered at this stage (Burkett 1993).

The *third stage* consists of mapping the IR and ARM constructs to produce the Application Interpreted Model (AIM). An AIM is an interpreted subset of the general-purpose IRs for use in the specific application context of the AP. The AIM includes a short form of an EXPRESS schema. It references the constructs of the IRs that are used, the schema in which each construct is defined, and additional constraints. Figure 13 presents an excerpt from the short listing in AP 202 (ISO 10303-202 1995). Figure 14 illustrates how the entity `date_and_time`, referenced from IR 41 in figure 13, is represented in the AIM extended schema. A mapping table shows how each

requirement in the ARM is satisfied by EXPRESS constructs in the AIM. Table 6 displays an example from the mapping table in AP 202: the *Application element* column shows application object names; *Source* is the number of the corresponding STEP part; *Rules* refers to numbered rules that apply to the current AIM element or reference path; and *Reference path* "documents the role of an AIM element relative to the AIM element in the row succeeding it" (ISO 10303-202 1995), allowing for the specification of a reference path through several related AIM elements. The connector '<=' in table 6 can be read as "is subtype of."

```
SCHEMA associative draughting;
USE FROM aic_advanced_brep;           -- ISO 10303-514
USE FROM aic_draughting_annotation;   -- ISO 10303-504
(...)
USE FROM date_time_schema             -- ISO 10303-41
  (date_and_time);
USE FROM geometric_model_schema       -- ISO 10303-42
  (edge_based_wireframe_model,
  geometric_curve_set,
  shell_based_wireframe_model);
(...)
ENTITY draughting_pre_defined_colour
  SUBTYPE OF (pre_defined_colour);
WHERE
  WR1: SELF.name IN ['black', 'red', 'green', 'blue',
                     'yellow', 'magenta', 'cian', 'white'];
END_ENTITY;
(...)
END_SCHEMA;
```

Figure 13 - AIM short listing in AP 202 (ISO 10303-202 1995)

```
ENTITY date_and_time;
  date_component : date;
  time_component : local_time;
END_ENTITY; -- date_and_time
```

Figure 14 - Entity `date_and_time` in the extended AIM schema in AP 202 (ISO 10303-202 1995)

In the *fourth stage*, tests based on the ARM and the long form of the AIM are developed in order to verify the conformance to constraints of the instances in the AIM.

The information model generated by the above procedure is a single schema, the normalized conceptual model in the AIM. It may therefore be tested by a conformance testing tool.

Table 6 - Mapping table for the AIM in AP 202 (detail) (ISO 10303-202 1995)

Application elements	AIM element	Source	Rules	Reference path
2D_CARTESIAN_ COORDINATE_SPACE	[geometric_representation_ context] [global_unit_assigned_ context]	42 41	6, 16	
2D_ELEMENTARY_ GEOMETRIC_CURVE_ SET	elementary_2d_geometric_ curve_set_shape_ representation	202	41	elementary_2d_geometric_curve_set_shape_representation <= shape_representation
2D_GEOMETRIC_ CURVE_SET	geometrically_bounded_2d_ wireframe representation	503	41	geometrically_bounded_2d_wireframe_representation <= shape_representation

AP development status

There are several APs currently under development by ISO (see the appendix for an extensive list). The largest document will probably be AP 214 (ISO 10303-214 1995), which is a very broad specification dedicated to automobile industry applications. Figure 15 shows the development status of APs as of January 1st, 1996: AAM (Application Activity Model), ARM (Application Requirements Model), AIM (Application Interpreted Model), CD (Committee Draft), DIS (Draft International Standard), or IS (International Standard).

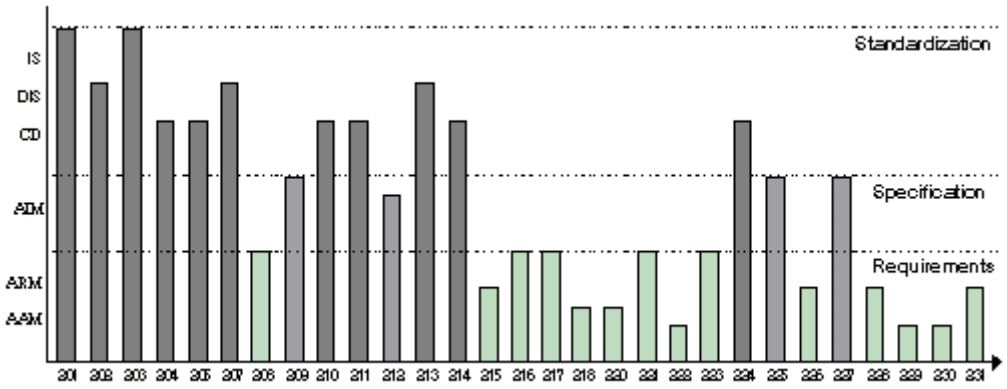


Figure 15 - STEP AP development status as of Jan. 1st, 1996 (P.D.I.T. 1996)

Table 7 presents some developments in the U.S. of application protocol usage prototypes, demos, and pilot developments, as of January, 1996.

Table 7 - STEP AP pioneer implementations (P.D.I.T. 1996)

Project	Application Protocol	Participants
AeroSTEP	AP203	Boeing, GE
AWS	AP203	IBM, Lockheed, NG
GM/EDS Prototype	AP207	GM/EDS
PAS-C	AP209	Lockheed, Vought, Boeing
PAP-E	AP211	Intermetrics, TRW, WH
PDXI	AP231	Chevron, DuPont, and others
PlantSTEP	AP227	The top 10 Plant Design Vendors
PREMP	AP210	Hughes, MMA, Rockwell, Boeing
RASP	AP210	Ford, UTC, Hughes, MMA, TI
PEP	AP201 and AP203	DoD Joint Center - FCIM/IPDE
RAMP	AP224	Grumman and U.S. Navy

AIMs tend to be very large documents and often overlap. This creates a need for AP integration, which will be discussed next.

Application Interpreted Constructs

AICs are interpretations of IRs which can be shared by various APs, aiming at the interoperability between APs. An AIC is a specification for a narrow domain area shared by two or more APs. It provides semantic integration for common functional requirements, for instance: geometric representation (B-rep, surface models, etc.) may be used in several different application domains such as automotive design, sheet metal tooling, and shipbuilding (Fowler 1995). An AIC can be developed and used to fulfill this functional requirement, as observed by Fowler:

> "The existence of an AIC identifies the *potential* for reuse of implementation code, and for the sharing of data between applications. This latter point is particularly important: since the data specification for surface models is common across the three applications identified above, data instances may be shared between them." (Fowler 1995)

STEP information models implementation

The APs are normalized database definitions that allow many different applications to share data. These databases can be implemented on file systems, relational databases, and object oriented databases. The next section discusses the implementation of STEP product data models.

2.2.4.3 STEP Implementation

The separation of STEP data models (*what* is to be exchanged) from implementation (*how* it is to be exchanged) is one of the standard's key features. To produce a STEP

implementation, an implementation method, from STEP's physical layer, is combined with an Application Protocol (AP), an application-specific information model specified in the application layer. Both information model and implementation method specifications are written in EXPRESS.

The implementation methods define the mapping of an application schema onto a specific computer technology. Four progressive implementation levels were devised:

- Level 1: File exchange -- passive text file transfer
- Level 2: Working form exchange -- software assisted, active file transfer
- Level 3: Database exchange -- shared database access
- Level 4: Knowledgebase exchange -- integrated knowledge-based systems

Levels 1 and 2 are well developed, and level 3 has research and commercial implementations. Level 4 implementations are subject of early research and development.

Physical file and working form exchange

STEP Part 21, "Clear Text Encoding of the Exchange Structure" (ISO 10303-21 1993), defines an application-independent way of encoding product data for file exchange. Part 21 files are suitable for data exchange of entire STEP schemata (implementation levels 1 and 2).

The EXPRESS schema which is mapped to the Part 21 implementation format is the AIM specified in an AP. It is an ASCII file, sequential, free-format (no column indentation), no CR character (one stream), with two sections: header and data, as illustrated in figure 16. The STEP file structure is based on an unambiguous context-free grammar, expressed in Wirth Syntax Notation -- WSN (Wirth 1977).

```
ISO-10303-21;
HEADER;
... (author, creation date, version of STEP, etc.)
ENDSEC;
DATA;
... (entity instances. Each entity has an id of the form #N,
     N being a unique integer, in any order. Entities may be
     referenced before they are defined)
ENDSEC;
END-ISO-10303-21;
```

Figure 16 - Structure of a STEP Part 21 exchange file

For a given CAx system to implement STEP Part 21 exchange, it is necessary to develop a front-end pre-processor and a back-end post-processor, which translate to and from Part 21 format into the internal representation format.

Figure 17 presents an example of Part 21 usage, with EXPRESS schema definitions and an exchange file with data structures compliant to the schemata.

(a)

```
SCHEMA example_geometry;  (* edited for brevity *)

                                    -- short name:
ENTITY cartesian_point              -- cpt
SUBTYPE OF (point);
  x_coordinate  : length_measure;
  y_coordinate  : length_measure;
  z_coordinate  : OPTIONAL length_measure;
END_ENTITY;

END_SCHEMA;

SCHEMA example_topology;

REFERENCE FROM example_geometry;

ENTITY vertex                       -- vx
SUBTYPE OF (topology);
  vertex_point : OPTIONAL point;
END_ENTITY;

ENTITY edge                         -- ed
SUBTYPE OF (topology);
  edge_start   : vertex;
  edge_end     : vertex;
END_ENTITY;

ENTITY edge_logical_structure    -- ed_strc
  edge_element : edge;
  flag         : BOOLEAN;
END_ENTITY;

ENTITY edge_loop                    -- ed_loop
SUBTYPE OF (loop);
  loop_edges : LIST [1:?] OF edge_or_logical;
END_ENTITY;

END_SCHEMA;
```

(b)

```
ISO-10303-21;

HEADER;
/*
   edited for brevity
*/
FILE_SCHEMA(('EXAMPLE_GEOMETRY',
'EXAMPLE_TOPOLOGY'));
ENDSEC;

DATA;
/*
THE FOLLOWING 13 ENTITIES REPRESENT A
TRIANGULAR EDGE LOOP
*/
#1=CPT(0.0,0.0,0.0);  /* CARTESIAN POINT ENTITY */
#2=CPT(0.0,1.0,0.0);
#3=CPT(1.0,0.0,0.0);
#11=VX(#1);          /* VERTEX ENTITY */
#12=VX(#2);
#13=VX(#3);
#16=ED(#11,12);      /* EDGE ENTITY */
#17=ED(#11,13);
#18=ED(#13,12);
#21=ED_STRC(#17,.F.); /* EDGE LOGICAL
                                   STRUCTURE ENTITY */
#22=ED_STRC(#18,.F.);
#23=ED_STRC(#16,.T.);
#24=ED_LOOP(#21,#22,#23));  /* EDGE LOOP ENT. */
/*
NOTE: OTHER SYNTACTIC REPRESENTATIONS FOR
A TRIANGULAR EDGE LOOP WERE POSSIBLE.
EACH REPRESENTATION MAY CARRY DIFFERENT
SEMANTICS, WHICH MAY REPRESENT
CHARACTERISTICS OF PARTICULAR APPLICATION
SYSTEMS.
*/
ENDSEC;

END-ISO-10303-21;
```

Figure 17 - EXPRESS schema definitions (a) and an exchange structure example (b)

Part 21 files allow for the exchange and archiving of data models. In exchange, the sender CAx systems generates an exchange file using its pre-processor, which is transferred to the receiver CAx system, which translates it into its internal representation format, using the post-processor. In archiving, the generated file is stored in a format that can be read by current and future applications. This is a strong reason for CAx systems to support STEP Part 21.

A number of public tools that support the building of applications using implementation levels 1 and 2 were made available (Libes 1993; Clark 1990). According to Fowler (1995), implementation level 2 is an extension to Level 1, where the physical file is converted to a "working form." Since most file-based implementations use this approach, the distinction between levels 1 and 2 have disappeared.

STEP implementation levels 3 and 4 implementations address a different set of problems: those associated with data and knowledge sharing, making use of the SDAI, STEP Part 22, discussed next.

The SDAI

Fowler approaches STEP level 3 implementations:

> "A level 3 implementation of STEP combines translation with data access. The translation element is equivalent to that for file exchange, i.e. converting from the internal format of a CAx system into the form prescribed by STEP. However, rather than storing the STEP data in a file for the purpose of exchange, an underlying database or repository is used to store the data. The data access element of a level 3 implementation is a standard interface to this underlying database, that allows applications to store, manipulate, and above all share the data in a standard manner."
> (Fowler 1995)

The Standard Data Access Interface is a collection of STEP standards supporting data sharing at a much finer level of granularity, taking advantage of the normalized schemata. It specifies language-independent low-level operators to create, delete, and retrieve objects, and to insert, modify, and delete object attributes.

The application programming interface (API) defined in the SDAI is based on a conceptual model. Different database technologies can be used for data storage without the need to alter the interfaces used between the applications and the database. With SDAI, database developers do not have to worry about how each application handles data, and application programmers can use SDAI functions without concern about the details of the underlying database technology.

Figure 18 presents a schematic view of STEP data sharing using a single database: applications access STEP data in a common database through SDAI. Data must comply with an information model defined in an application protocol (AP). Each application must provide a pre- and post-processor to translate between its own data format and the AP.

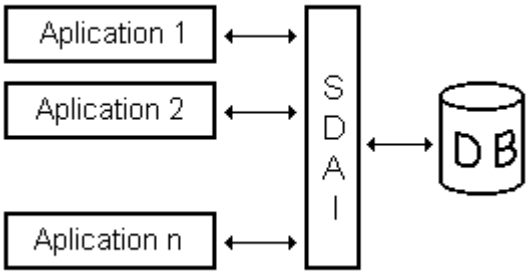


Figure 18 - Single-database STEP data sharing

Goh et al. (1994) comment on how SDAI can be used to share data and metadata between different applications and databases:

- Each database's application programming interface provides the means by which the data independence is perceived by the applications. However, these interfaces differ from database to database. Such differences impede data sharing and exchange.
- SDAI addresses this issue by means of a consistent set of mechanisms, regardless of the underlying data storage technology.
- Any application whose data is modeled in EXPRESS can utilize SDAI, thereby making the actual database transparent.

Figure 19 presents an example of SDAI functionality. The operation *End transaction access and commit* ends the sequence of operations started by *Start transaction*, and commits (makes persistent) all changes to the entity instances made since the most recent *Start transaction*.

The process of storing data from a CAx system in a database via the SDAI involves three distinct stages (Fowler 1995):

- Translation: The same as in file exchange. The result will be data stores in "in-memory" data structures within the interface software.
- Access: SDAI function calls are used to create and populate data structures within the underlying database.
- Storage: The SDAI implementation translates each "standard" function call into the operations necessary to create and populate data structures within the physical database, (e.g. tables, objects...)

Input

Transaction: **sdai_transaction;**
The transaction allowing access to entity instances which is to be terminated.

Possible error indicators

SS_NOPN	Session not open.
FN_NAVL	Function not available.
TR_NEXS	Transaction does not exist.
TR_EAB	Transaction ended abnormally.
TR_NAVL	Transaction not available.
SY_ERR	Underlying system error.

Effect on the SDAI environment

Transaction shall be deleted.

The **active_transaction** attribute of the currently open **sdai_session** shall be unset.

schema_instance.change_date for any modified or created **schema_instance** shall be set to the current date.

sdai_model.change_date for any modified or created **sdai_model** shall be set to the current date.

Figure 19 - Specification of the SDAI operation *End transaction access and commit* (ISO 10303-22 1995)

SDAI's main document is STEP Part 22 (ISO 10303-22 1995), which defines a language-independent access interface. Bindings are being defined to programming languages such as C++ (ISO 10303-23 1995), and C (ISO 10303-24 1995). Part 26 (ISO 10303-26 1997) is a SDAI language binding for the IDL (Interface Definition Language) of CORBA (Common Object Request Broker Architecture), allowing for the integration of STEP with CORBA, aiming at the realization of network shareable interoperable product databases, discussed in chapter 3.

SDAI implementations vary in binding style, i.e., the way in which the database software is structured (Hardwick & Loffredo 1995):

- Early binding, or code generation: The application provides access functions for a specific information model written in EXPRESS.
- Late binding, or data dictionary: The implementation is a general purpose software that is independent of information model, allowing access to data defined by any EXPRESS information model. It is more flexible, and offers lower performance than early binding implementations.

Besides the standard mappings to programming languages, there is work addressing the mapping of EXPRESS into the relational standard query language, SQL (Morris 1990). However, the relational model does not have the richness of EXPRESS.

It is expected that object-oriented languages and databases will yield better STEP implementations.

SDAI does not specify, at least in its current release, details about concurrency control, transaction management, and connection management. These out-of-scope issues are to be handled by each database management system. However, the SDAI provides some of the basic constructs for the building of a data sharing environment.

Knowledgebase implementations (STEP implementation level 4) are still object of early research and development (Higa et al. 1992). This domain has been approached by ontology engineering (Meis & Ostermayer 1996), discussed next.

2.2.4.4 Ontology Engineering and STEP

Ontology is a term borrowed from philosophy, meaning "a systematic account of existence" (Gruber 1993). The concept of ontology for Computer Science, according to Guarino (1996), is "a still debated issue", with the following being the most cited definition in the knowledge sharing community:

> "An ontology is an explicit specification of a conceptualization." (Gruber 1994)

A conceptualization is an abstract, simplified view of a domain which is object of knowledge representation. Therefore, an ontology is a well-defined terminology for the representation of concepts, objects, and relationships that hold in a domain.

Ontologies have been used as an enabling technology for the sharing and reuse of knowledgebases. Gruber (1992) enounces that knowledge sharing is more effective than building knowledge bases from scratch but, since knowledge bases have been built typically as *ad hoc* designs intended for narrow tasks, knowledge sharing is a difficult task.

In order to achieve sharable, reusable knowledge bases, there is need to (Gruber 1991):

- Specify a **canonical form** for declarative knowledge: One such canonical form is KIF (Knowledge Interchange Format), a formal language for the interchange of knowledge among disparate computer programs (Genesereth & Fikes 1992).
- Define **common ontologies**: Vocabularies of representational terms (classes, relations, functions, object constants) with agreed-upon definitions (restrictions, hierarchies, facts, and other axioms).

Ontologies play the role of coupling interfaces among shared knowledge bases, in a way similar to the formal argument list of a procedure in a conventional software library, or a conceptual schema in a database system (Gruber 1992). According to Gruber (1993), a conceptual schema provides a logical description of data, allowing for data sharing among application programs and databases without having to share data structures. While a conceptual schema provides definitions on *Data*, an ontology defines terms with which to represent *Knowledge*.

Connections between STEP and ontology engineering approaches

Gruber describes the importance of the use of ontologies for knowledgebases:

> "Ontologies capture reusable intellectual *content* of a representation effort--the choices about classes and relations that are potentially relevant for describing a domain or performing a task. Each ontology embodies a set of ontological commitments in a form that enables one to build knowledge bases and tools based on those same commitments. Given a common language and vocabulary, one can build knowledge bases that instantiate and specialize the shared classes and relations. The instantiations and specializations of the ontologies carry application-specific information. Thus, the role of ontologies is to specify a *modular coupling* among bodies of knowledge and the tools that operate on them, serving as knowledge-level protocols for input, output, and communication."
> (Gruber 1991)

This is similar to the use of STEP in product databases: to provide protocols for the representation and communication of product data.

Meis & Ostermayer (1996) suggest that an ontology engineering approach to STEP could improve the quality and efficiency of STEP's conceptual model development process. During conceptual design, formal modeling decisions are made about *what* is being implemented, but no decisions are made about *how* something is being implemented. The authors maintain that the same principle used to improve the quality and efficiency in the development of software by an early formal documentation is also applicable to the development of conceptual models. Conceptual models are also software products, and the early formal documentation can be performed with the aid of an approach to ontology engineering. The early design phase documented using an ontology engineering approach is called *conceptualisation* (Meis & Ostermayer 1996). During conceptualization, according to figure 20, the entities and relationships existing in the domain being modeled are named and described, and the explicit specification of the result is called an *ontology* (Meis & Ostermayer 1996).

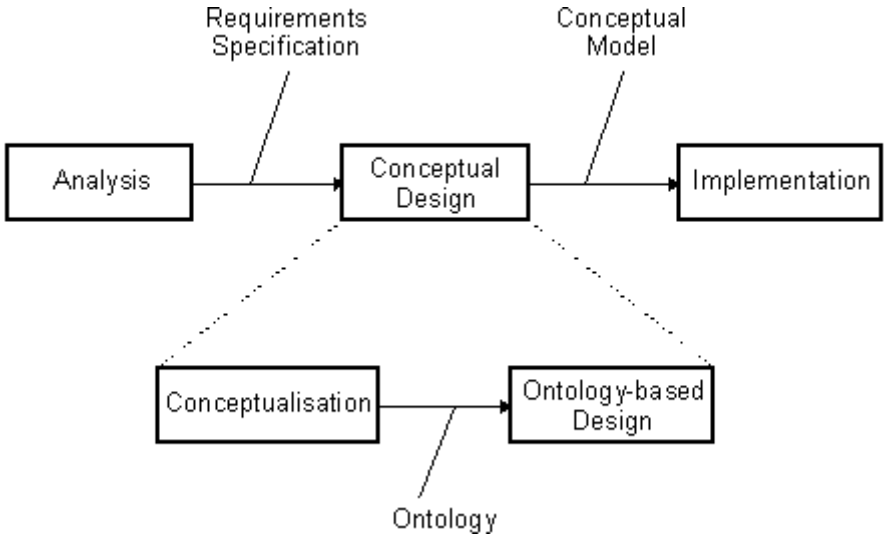


Figure 20 - Software design phases (Meis & Ostermayer 1996)

In STEP, the role of ontologies is performed by the IRs within the development of AIMs. Figure 21 shows a specification taken from STEP Part 41, *Fundamentals of product description and support* [ISO 10303-41]. The entity `action_relationship` is a relationship because it adheres to the *relationship template*, a design pattern defined in Part 41. Meis & Ostermayer (1996) argue, however, that the terms `entity` and `relationship` are not defined in STEP, potentially leading to inconsistent and non-uniform application of the constructs in other information models. An ontology approach to STEP would require a formal definition of `entity` and `relationship`, thereby improving quality and efficiency of the conceptual model development process.

```
ENTITY action_relationship;
    name            : label;
    description     : text;
    relating_action : action;
    related_action  : action;
END_ENTITY;
```

Figure 21 - Specification of a relationship in STEP Part 41

A number of “weak points” was identified in the STEP approach (Meis & Ostermayer 1996):

1. STEP information models are too complex and extensive for humans and machines to master dealing with them.
2. STEP information models are inconsistent and non-uniform.
3. The development and maintenance of STEP information models is inefficient (this is a consequence of points 1 and 2).
4. There is no sureness (i.e., no stability) in the development and standardization of STEP information models.
5. The development and use of STEP information models is not open with regard to other modeling approaches.

And some potential benefits of an ontology engineering approach to STEP are outlined, regarding the “weak points” above (Meis & Ostermayer 1996):

1. Reduction of the complexity and extent of STEP information models.
2. Increase of the quality of STEP information models.
3. Increase of the efficiency in the STEP development and maintenance (this is a partial consequence of 2).
4. Increase of the sureness in the development and standardization of STEP.
5. Increase of the openness of the STEP approach with respect to other modeling approaches.

2.2.4.5 STEP’s Future

As an international standard, it is plausible that the use of STEP data models be a requirement for the participation in international contracts in the near future. Also, the integration of industrial enterprises depends largely upon the adoption of standards.

Concurrent engineering, the vertical aspect of industrial enterprises integration, "reduces the time needed to get a product to market by allowing different groups of engineers to work in parallel on different design and manufacturing tasks" (Hardwick et al. 1995). STEP is a key enabling technology for the realization of concurrent engineering, allowing for the data exchange between the applications that take part in a product’s design.

Regarding the horizontal aspect of industrial enterprises integration, virtual enterprises, STEP has been adopted as the enabling technology for product data representation, exchange, and sharing for the National Information Infrastructure, according to the architecture defined by the National Industrial Information Infrastructure Protocols (NIIIP) Consortium (NIIIP 1996). The NIIIP Consortium is a

joint effort of United States' government and industrial enterprises for the deployment of the enabling technology for the realization of virtual enterprises.

STEP usage has economic advantages. It reduces dramatically the expenses associated with data translation. It permits that sophisticated CAx systems be used for very specialized tasks, while ordinary tasks are executed using general-purpose CAx systems. It allows data to be archived in a format that will be (presumably) continuously supported by CAx systems in the future.

The confidence in the standard's success, however, is far from absolute. The appearance of articles containing criticism regarding STEP's quality, and even with the expression "Out of STEP" (Eastman 1994; Evans 1994) in their titles, account for hindrances for the full realization of STEP's objectives. Laurance manifests the concern that STEP may fail because of its weight, and offers the question:

> "Will STEP work?"
> (Laurance 1994)

The author states that, as a replacement for IGES, the answer is *yes*. In comparison with previous PDE standards, STEP is the only one which is forward-looking, not retrospective (Owen 1993), making good use of the conceptual character of its specifications. Even STEP documents from the physical layer, like the SDAI, state their definitions in a technology-independent way, leaving implementation details to be established in language bindings, which are also standard documents. Also, the separation of implementation methods and information modeling, and the concept of application-specific protocols are great advances in relation to previous standards.

The success, however, is less clear for the broader picture. Problems reported by Laurance (1994) are:

- The bottom-up approach leads to lots of redundancy and difficulty to coupling multiple STEP application protocols.
- The first two standardized APs (AP 201 and 203) are more than 500 pages long, establishing the tendency that APs are going to be very large documents.

Burkett (1993) maintains that implementation considerations are not comprehensively or consistently accounted for in the development of STEP, and that STEP lacks an *implementation schema*, claiming that:

- Implementations based on the conceptual AIMs will be less than optimal; and
- Implementation Schemata are needed as part of the standard as a logical model which comprehensively and consistently accounts for implementation efficiency and optimization.

Wilson (1993) reports that STEP “has changed its scope, its methods of working, its technologies, and so on, since ... 1984." Another plausible change in the STEP development process would be the adoption of an ontology engineering approach, as described in the previous section. This has already been approached in STEP through the SUMM (Semantic Unification Meta Model), a “mathematically rigorous approach to the unification and integration of models independently of the languages in which those models were formulated” (Eastman 1994). According to SUMM, the unification of a set of models should be achieved by mapping the constructs of each model to a superset, a common abstract model (Meis & Ostermayer 1996). Nevertheless, SUMM’s development by the IGES/PDES Organization ended in 1992, and has not been applied to STEP so far.

2.3 Application Interoperability

Interoperability is defined as:

> "the effective interconnection of two or more different computer systems, databases, or networks in order to support distributed computing and/or data exchange."
> (Office of Science 1994)

Distributed computing, or the computing of *distributed systems*, is defined as the implementation of a collection of components or applications that interoperate with each other, but that execute in separate address spaces, and may execute on separate hardware or software platforms (Maybee et al. 1995).

2.3.1 Approaches to Interoperability

A number of research, implementation, and standardization initiatives have addressed the problem of interoperability. Four specifications (two open standards, one from a vendor consortium, and one proprietary) aimed at supporting application interoperability are fairly widespread: the Distributed Computing Environment (DCE), the proprietary Object Linking and Embedding / Component Object Model (OLE/COM), the DSOM(Distributed System Object Model)/OpenDoc vendor consortium specification, and the Common Object Request Broker Architecture (CORBA).

DCE was developed by the Open Software Foundation (OSF). It is an integrated set of services designed to support distributed applications. These services include:

- Remote Procedure Call (RPC), which calls applications and operating systems to perform specific tasks, one task at a time;
- Directory Service;
- Time Service;
- Security Service; and
- Threads Service.

OLE (Object Linking and Embedding) is an object technology from Microsoft, originally meant to support compound documents. The Component Object Model (COM), developed jointly by Microsoft and Digital Equipment Corporation, aims at providing support for distributed objects.

The Object Management Group, a consortium of currently 600 companies, developed CORBA as a standard for interoperability, discussed next.

2.3.2 OMG CORBA

The Object Management Group (OMG) was formed in 1989 with the purpose of creating standards allowing for the interoperability and portability of distributed object-oriented (OO) applications. A strength of this approach is that most of the major players in the commercial distributed OO computing arena are among the several hundred companies that belong to the OMG (Vinoski 1993).

The key to understanding the structure of the CORBA architecture is the Reference Model, defined in the OMG's Object Management Architecture (OMA) (OMG 1993), consisting of:

- Object Request Broker (ORB): Enables objects to transparently make and receive requests and responses in a distributed environment.
- Object Services: A collection of services (interfaces and objects) that support basic functions for using and implementing objects. Services are necessary to construct any distributed application and are always independent of application domains.
- Common Facilities: A collection of services that provide general purpose capabilities useful in many applications (e.g., e-mail, printing services).

- Application Objects: Objects specific to particular end-user applications. They are not standardized by OMG.

Figure 22 illustrates the Reference Model.

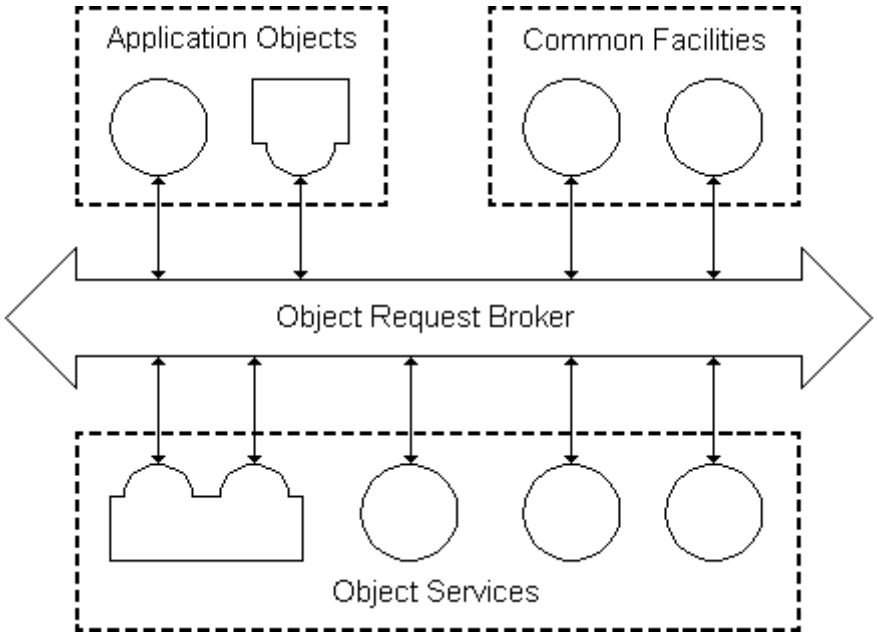


Figure 22 - OMA Reference Model (Soley & Stone 1995)

Primary goals are the reusability, portability and interoperability of object-based software components in distributed heterogeneous environments.

The ORB is like a telephone exchange, providing the basic mechanism for making and receiving calls (OMG 1995). It's functionality is explained:

> "Using an ORB, a client can transparently call on and initiate an action on a server object, which can be on the same machine or across a network. The ORB intercepts the call and locates an object that can implement the request, pass it the parameters, invoke its method, and return the results. The client does not have to be aware of where the object is located, its programming language, its operating system, or any other system aspects that are not part of an object's interface. In this way, the ORB provides interoperability between applications and seamlessly interconnects multiple object systems."
> (OMG 1993)

Watson (1996) observes that the OMG design philosophy is like that of a microkernel operating system, where CORBA is the kernel, providing communication

between the system components and little else. The facilities necessary in a distributed object environment are provided by a growing set of Object Services.

Objects made available through an ORB publish their interfaces using CORBA's Interface Definition Language (IDL). In IDL, object attributes and operations are specified in a programming language-independent way. The IDL grammar is a subset of the proposed ANSI C++ standard, with additional constructs to support distribution concepts (OMG 1995).

Vinoski discusses the need of a language-independent interface:

> "Unfortunately, shipping C++ objects across a network typically requires a violation of object encapsulation. (...) To avoid breaking encapsulation, distributed OO applications must deal only with object interfaces and should not care whether the object implementations are in the same process or on another machine halfway around the world. This ideal requires an object model that allows applications to transparently utilize both local and remote objects without sacrificing efficiency. Such an object model must address issues faced by all developers of distributed applications, providing a standard object programming interface that is not only system-independent but is also language-independent."
> (Vinoski 1993)

The author enumerates reasons why not adopting only a declarative subset of C++ as interface definition language (Vinoski 1993):

- The CORBA specification is intended to be language-independent; using a subset of C++ for IDL lessens the chances of CORBA being widely accepted in the distributed OO programming community.
- C++ is known to be difficult to parse; using a subset of C++ might ultimately limit the number of IDL compiler implementations available.
- Some features of C++, notably pointers, make marshaling of data difficult.
- Any declarative subset of C++ chosen would most likely be different enough from normal C++ so as to be confusing to experienced C++ users.

Since IDL is purely a descriptive language, clients are written in languages for which mappings from IDL concepts have been defined. The mapping of an OMG IDL concept to a client language construct will depend on the facilities available in the client language. The OMG IDL specification is expected to track relevant changes to C++ introduced by the ANSI standardization effort (OMG 1993). Also, IDL's status has been updated recently to ISO international standard 14750 (Siegel 1997).

CORBA 2.0, adopted in December of 1994, defines true interoperability by specifying how ORBs from different vendors can interoperate. Requests are the only way that ORBs interact. For request interoperability, the elements which need to be addressed include (OMG 1995):

1. Finding the appropriate ORB;
2. Communicating with that ORB;
3. Representing the target object reference;
4. Identifying the operation;
5. Translating the data parameters, exception, and context; and
6. Passing authentication and security information.

Particularly, general requirements for the ORB implementations are:

- Ability for two vendors' ORBs to interoperate without prior knowledge of each other's implementation.
- Support of all ORB functionality.
- Preservation of content and semantics of ORB-specific information across ORB boundaries (for example, security).

Comparing CORBA to DCE

CORBA resembles DCE in that both have an Interface Definition Language--CORBA's IDL is based on C++, while DCE's is based on C. Nevertheless, CORBA represents an object-oriented approach to system architecture, whereas DCE does not (Brando 1996). This doesn't imply that DCE excludes object-oriented programming; only that it "fails to provide explicit support for the object-oriented paradigm" (Brando 1996). DCE's Remote Procedure Calls are inherently procedural. CORBA, on the other hand, is inherently object-oriented: entire applications, as well as combinations of applications and data, can invoke and encapsulate one another (Tibbits 1995). Rather than writing a specific procedure call to link applications, the developer can actually encapsulate one application into another larger application.

Like most object-oriented environments, CORBA support (Brando 1996):

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

- Late binding of operation invocations to function calls
- Reuse of classes and objects

The creation of new classes and objects at runtime, another feature of object-oriented systems, is not supported directly, but CORBA includes sufficient mechanisms for an ORB vendor to provide that capability (Brando 1996).

Comparing CORBA to OLE/COM

Maybee et al. evaluate the current stage of OLE/COM:

> "Until COM provides complete support for distributed objects, it is premature to make direct comparisons. Recently, Microsoft has begun to address these issues by using DCE to provide the underlying naming and marshaling support. It is fair to say that this means that COM will have the same merits and demerits as DCE."
> (Maybee et al. 1995)

According to Tibbits (1995), CORBA shares with OLE/COM the characteristics of using object-oriented programming principles, including inheritance, encapsulation, and extensibility. Both rely on an ORB to make requests to distributed objects. OLE 2.0, however, does not address the problem of integrating applications on non-PC platforms. The Common Object Model (COM) specification stipulates how OLE functionality can be implemented on non-PC platforms where CORBA is being used. Also, the Object Management Group has issued a request for proposal for establishing COM and CORBA interoperability.

CHAPTER 3

NETWORK INTEROPERABLE PRODUCT DATA MODELS

This chapter introduces the integration of the standards STEP and CORBA as enabling technology for the sharing of network interoperable product data models. Implications of this standards-based choice are presented, challenges in the translation of EXPRESS product data models are outlined, and an analysis of the semantic loss in this translation is proposed.

3.1 A Standards-Based Approach to Network Interoperable Product Data Models

Chapter 2 presented a literature review in which several standards are accounted for: SQL for database manipulation, STEP, IGES, SET, and VDA-FS for product data exchange, and CORBA for application interoperability. Standards are alternatives to the use of proprietary specifications, which usually precede standardization. According to Mowbray:

> "The key difference between standards and proprietary specifications is: open systems standards are consensus agreements among multiple vendors. Typically, proprietary specifications are after-the-fact documentation of a single implementation. In contrast, a standard is usually a forward-looking technology agreement that builds on a long-range reference model, such as the OMG's Object Management Architecture. In general, standards provide a more stable, low-risk basis for system development."
> (Mowbray 1996)

While there are several standards that are potentially relevant for information systems, there are organizations (e.g., ISO, IEEE, and NIST) that provide guidance documents, called *open systems environments* (OSEs) (Mowbray 1996).

The approach adopted for STEP network interoperability comes from the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) (OMG 1995). CORBA and STEP are also, along with workflow and knowledge

management, and the Internet, core technologies of the NIIIP (National Industrial Information Infrastructure Protocols) project, which has as one of its objectives:

> "... to establish an open, standards-based software infrastructure protocol that integrates heterogeneous and distributed processes, data, and computing environments across the U.S. manufacturing base." (NIIIP 1996)

Rando & Paoloni (1994) point out that manufacturers have successfully implemented very sophisticated point solutions, but they have been unable to integrate these solutions into enterprise wide systems, The authors outline the need for industrial enterprise integration:

> "Today's goal is a complete re-engineering of manufacturing processes, including the integration of a variety of processes including manufacturing, design, analysis, simulation and business practices. The vertical integration of these processes within a single enterprise, *collaborative concurrent engineering*, is a goal that has not yet been fully realized. Nevertheless, the exigencies of today's requirements have forced manufacturers to look beyond this goal toward the horizontal integration of manufacturing processes in the *virtual enterprise* (VE). The VE is a temporary alliance of organizations for the purpose of performing a task which none of the organizations could accomplish on its own."
> (Rando & Paoloni 1994)

The realization of industrial enterprise integration depends on the successful integration of CAx systems. Coupling STEP and CORBA is a standards-based solution for this integration. It is approached in the scope of STEP through the CORBA's Interface Definition Language binding to the SDAI (ISO 10303-26 1997); it is also the approach adopted in the NIIIP project.

Role of the Standard Data Access Interface

In STEP, SDAI is the standard concerned with the implementation of sharable product data models. Since SDAI specification is written in EXPRESS, a language not aimed at implementation, STEP comprises bindings from programming languages such as C (ISO 10303-24 1995) and C++ (ISO 10303-23 1995) to the SDAI. Also, a draft standard of the CORBA IDL binding to the SDAI (ISO 10303-26 1997) has been released, allowing for the network interoperable access to STEP data models.

Figure 23 illustrates the network interoperable access to STEP data. End-user applications and STEP databases may be at different locations, in different networks, connected to different SDAI implementations and ORBs (Object Request Brokers). A data request from an application is passed on to an ORB. The ORB communicates

with other ORBs in order to locate the database. A database application receives the request and, using database operators translated from the SDAI operators, performs the access and delivers data back to the application.

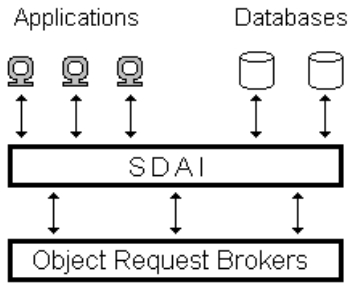


Figure 23 - Network interoperable access to STEP databases using CORBA

EXPRESS to IDL

STEP data models, written in EXPRESS, are objects which can be shared across a network. In order to publish their interfaces, STEP objects are translated into IDL, as Hardwick et al. discuss:

> "To create the infrastructure (for virtual manufacturing enterprises), the data definition language of CORBA, IDL, is combined with the data definition language of STEP, EXPRESS. These two languages have different purposes: IDL describes interfaces to applications; EXPRESS describes normalized data models. Both can be used to describe objects for manufacturing applications."
> (Hardwick et al. 1996)

3.2 Implications and Challenges Presented by the Standards-Based Integration of Product Data Models

The choice of STEP and CORBA as standards for the integration of network interoperable product data models imply technological choices, and poses challenges to the realization of this integration.

Hardwick (1996) identifies barriers hindering communications of industrial information within an industrial virtual enterprise:

- Insufficient security control -- Companies collaborating in a virtual enterprise may be competitors in other projects.
- Loss of control over projects -- Control techniques differ among the companies.

- Inability of application systems to interoperate -- Data produced by one company cannot be read by the application systems of another company.
- Unfamiliar technologies and application systems -- Technologies and application systems are often complex and require extensive training.

The problems of insufficient security control, and the loss of control over projects, are outside the scope of the STEP-CORBA coupling; they are being addressed with the development of secure communication protocols (Ganesan & Sandhu 1994) and workflow control systems (McCusker 1993). The integration of STEP and CORBA can lessen the barrier generated by unfamiliar technologies and applications, since it allows for each company to use the systems of their choice, and share data in standard format. Nonetheless, the most important barrier which the STEP-CORBA coupling can help overcome is the inability of application systems to interoperate. Hardwick et al. (1996) add that “two applications cannot make use of each other’s functionality because they cannot invoke one another’s resources,” naming this barrier *code interoperability*.

The next sections report on the implications of the choice of STEP and CORBA, and the difficulties to be overcome, regarding three aspects related to interoperability: database management, application interoperability, and product data representation.

3.2.1 Database Management Technology

The adoption of STEP and CORBA does not imply any restriction on the nature of the storage. According to McLay & Morris:

> “STEP provides the definition of product information, but it does not define how the data are stored. Some applications will rely on exchange files, other may require relational databases, and yet other STEP applications may use an objet-oriented database."
> (McLay & Morris 1990)

Questions related to performance or transaction control might indicate directions on whether to opt for a relational database system, an object-oriented database, or flat files (STI 1992), as described in table 8. However, even if an optimal point solution can be found, it is unlikely that it would be good enough for every application in a distributed computing environment. The implementation of network interoperable product databases will rather depend upon a combination of different systems and technologies. As Kent pointed out,

> "Thus the truth of things may be this: useful things get done by tools which are an amalgam of fragments of theories."
> (Kent 1978)

While STEP is a forward-looking, technology-independent collection of standards for product data exchange, CORBA provides for application interoperability, serving as an amalgam among different applications, networks, and databases based on different paradigms.

Table 8 - Factors influencing the choice for database technology (summarized from STI (1992))

Application class	• Applications for bank or airline companies may require special characteristics for transaction control, locking, client-server architecture, etc., but for geometry manipulation these issues are unimportant. • For example, STEP Part 41 [ISO 10303-41] features bill-of-materials style data suited to a relational database.
Performance	• A relational database may be adequate for few queries. For a high number of queries, and sequential queries, relational databases are slow due to the high initial costs of transactions. • STEP Part 21 files (text files) is an adequate storage choice if the application is going to process all or most data in a data set. All subsequent accesses can be performed at main memory. • Object-oriented databases are the most promising choice if the application does not know if it will access just a few, or very many records in a set.
Transaction Control	• Relational databases are the best choice for short transactions. • Applications which process only long transaction can be managed using text files. • Object-oriented databases are indicated for transactions of mixed duration. • In many engineering applications, users are required to work on different versions of a product. In this case, version control is more important than transaction control.
Versioning	• A good versioning system should make it easy the capture of engineering changes which, in a database, correspond to many edits to many database items. • Hardwick et al. (1995) report on the usage of delta (text) files to communicate design changes.

The adequacy of specific database technologies for complex applications like CAx systems has been the object of intense debate. Stone & Hentchel observed, regarding this ‘database war’:

> “How do you decide which type of database is best when even the experts can't agree?”
> (Stone & Hentchel 1990)

The integration of STEP and CORBA allows for different database approaches to be integrated, provided that they can map their data structures in accordance to the conceptual schemata defined in STEP APs.

Hardwick & Loffredo (1995) described mappings between EXPRESS and the data definition languages (DDLs) of four different DBMSs, and found that some rich modeling features of EXPRESS can challenge the capability of the DDLs:

- Entities;
- Inheritance;
- Primitive types;
- Enumerations;
- Selects; and
- Aggregates.

Mappings from EXPRESS to SQL have been explored by Raghavan (1992), Morris (1990), Mead & Thomas (1989), and Eggers (1988). Sanderson & Spooner (1993) describe mappings from EXPRESS to network, hierarchical, and relational models, and show that EXPRESS is semantically richer.

Kern & Bøhn (1995) report on early STEP implementations using object-oriented databases (OODBs). However, most implementations addressed data interchange using STEP Part 21, using OODBs for storage, failing to realize the *sharing* of product data models.

3.2.2 Application Interoperability

The choice of CORBA as the standard for application interoperability is still an undertested solution. While most authors praise this choice, others point out challenges to the realization of application interoperability using CORBA:

- Weak type system: CORBA does not have the ability to marshal data structures with shared substructure, such as graphs, which require extra copying to map legacy data types, and that typecodes be passed by value and not by reference (Heimbigner 1995).
- Run-time issues: Its multilanguage support is limited to C and C++ (Heimbigner 1995).
- Security and authorization: This is not addressed within CORBA.
- Persistence: This is not fully addressed. According to Tibbits,

> "CORBA only provides a mechanism that allows an application to convert an object reference to a string and vice versa. This string could be stored to a disk and recovered at some point in the future, but the process is not an exact substitute for persistent store. Basically, persistence is left to the ORB implementation and the application developer."
> (Tibbits 1995)

However, there is a proposal for a Persistent Object Service (POS), which will provide a generic persistence interface that will allow any database to be hooked in as a back end.

Despite the challenges presented, Heimbigner (1995), one of the critics, states that CORBA "is representative of problems with other standards." These problems don't affect CORBA's tendency to gain market share. In fact, Watson (1996) refers to a recent report on the distributed object marketplace by Ovum Ltd., estimating that by the end of this decade over 9 million computer systems will use CORBA, making up some 75% of the distributed object middleware market. Accordingly, Vinosky maintains that:

> "CORBA specification is well on its way to becoming the standard base for distributed OO applications. Together, its flexible components allow legacy systems to work seamlessly with new applications."
> (Vinoski 1993)

3.2.3 Product Data Representation

The choice of STEP and CORBA for the realization of network interoperable product data models implies that product data will be represented using EXPRESS, according to STEP's Application Protocols. It also implies that the network interoperable access to product data models will be provided by publishing the interfaces of STEP objects for access by the Object Request Brokers according to the IDL binding to the Standard Data Access Interface. The effectiveness of this product data representation is dependent upon two factors: quality of STEP models; and quality of the translation of EXPRESS data models into IDL for network interoperable access.

Quality of STEP models

STEP presents an innovative approach to data modeling. The concept of Application Protocols is one of the major advances of the standard. Nonetheless, debate over the appropriateness of changes in the modeling process remains, as well as questions about STEP's current capacity to fulfill product data modeling needs.

Overcoming the diversity in data representation generated by different design techniques is one of the requirements that STEP models are expected to fulfill. According to Barra (1996), the only way to make possible a completely seamless data interchange is:

- Using the same system; and
- Using the same modeling techniques.

Apart from these limitations, the better which can be expected is that data received by a CAx system be *functionally equivalent* to data originated by the sender CAx system. The fulfilling of this condition is related to the notion of completeness, one of STEP's objectives. As product design data is communicated from one application to another, the designers have to make sure that the receiving CAx system can interpret correctly the data received. Also, data for different applications may be modeled by different Application Protocols, creating a need for *AP interoperability*.

Changes, or improvements to the STEP development process have been proposed by Meis & Ostermayer (1996), based on the ontology engineering approach. Metzger (1996), a STEP developer, concludes that "a single conceptual model for product data, described using a uniform language, cannot be designed," and proposes to "give up the idea of a complete and unified data model," a radical change in relation to the current process of development, which builds Application Protocols interpreting the Integrated Resources. A "modeling task delegation" is proposed:

> "The task of modeling should be done by different and independent task groups, not by a core group of generalists 'integrating' the result. Each task group should be responsible for the quality of their product, the domain specific language used, the definition of the terms used, the modeling style, etc."
> (Metzger 1996)

Metzger supports his proposal by giving examples of how the concept of a *thing*, and *information about a thing*, are mixed in STEP. STEP attempts to identify different concepts using a uniform language (Metzger 1996), but this leads to the following mismatches:

- In the overview of STEP, *product* is defined as "a thing or substance produced by a natural or artificial process."
- In STEP Part 41, a *product* is "the identification and description, in an application context, of a physically realizable object that is produced by a process."

Hence, *product* means both a thing and the information about the thing.

STEP is an evolving standardization effort which has already changed some of its methods. The issues about the quality of STEP data models mentioned above indicate that STEP data modeling can still undergo changes, even major ones, which will impact the quality of product data models.

Quality of the translation of EXPRESS data models into IDL

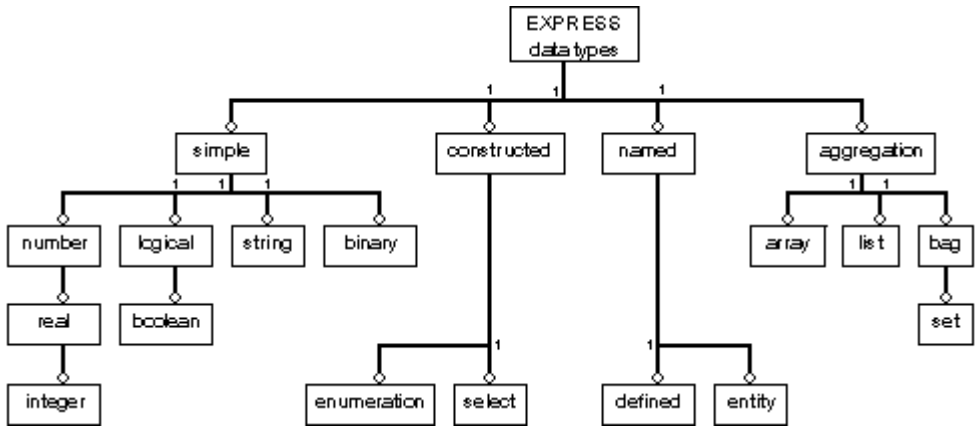


Figure 24 - Classification of EXPRESS data types, in EXPRESS-G (Kiekenbeck et al. 1995)

According to Su et al. (1995), "EXPRESS is semantically much richer than IDL and has an object-oriented flavor." Kiekenbeck et al. (1995) observe that the mapping of EXPRESS to another language requires a comprehensive mapping of the EXPRESS type system to the destination language, and loss of parts of the EXPRESS type system through the mapping would limit the usefulness of the generated code.

The EXPRESS type system, illustrated in figure 24, comprises simple, aggregation, named, constructed, and generalized data types, forming an inheritance graph (Kiekenbeck et al. 1995). Simple, SELECT, aggregates, and inheritance and complex entity data types have peculiarities that can challenge the mapping to another language, as presented next.

Simple types

Regarding the simple types in figure 24, some peculiarities of EXPRESS are:

- NUMBER data type is neither a REAL nor an INTEGER.
- LOGICAL data type allows the values TRUE, FALSE, and UNKNOWN. BOOLEAN data type, however, only allows TRUE and FALSE.

SELECT

SELECT defines a named collection (the base type) of data types (the underlying data types) (Rando & McCabe 1996). Figure 25 illustrates the use of SELECT, where a `wall_mounting` attaches a `product` onto a `wall` using either a permanent (`glue` or `weld`) or temporary (`nail` or `screw`) attachment method.

```
TYPE attachment_method = SELECT(permanent_attachment,
                                temporary_attachment);
END_TYPE;
TYPE permanent_attachment = SELECT(glue, weld);
END_TYPE;
TYPE temporary_attachment = SELECT(nail, screw);
END_TYPE;
ENTITY nail;
  body_length : REAL;
  head_area   : REAL;
END_ENTITY;
ENTITY screw;
  body_length : REAL;
  pitch       : REAL;
END_ENTITY;
ENTITY glue;
  composition : material_composition;
  solvent     : material_composition;
END_ENTITY;
ENTITY weld;
  composition : material_composition;
END_ENTITY;
ENTITY wall_mounting;
  mounting : product;
  on       : wall;
  using    : attachment_method;
END_ENTITY;
```

Figure 25 - Example of usage of the SELECT construct (ISO 10303-11 1994)

Aggregates

In STEP, aggregate is the name of strongly typed collections. Usually, they are required to be variably sized. For this reason, it is not practical to implement strongly typed aggregates in a late binding implementation (Rando & McCabe 1996), i.e., an implementation which can handle any EXPRESS data model, in contrast to an early binding implementation which is devoted to a specific EXPRESS data model.

Inheritance and complex entity data type

Inheritance is addressed in EXPRESS through the supertype/subtype relationship of EXPRESS entities. Supertype constraints are used to restrict the combination of entity types to compose a valid entity instance, i.e., ONEOF (mutually exclusive), AND

(mutually inclusive), and ANDOR (optionally inclusive subtypes). Schenck & Wilson comment on the rich EXPRESS inheritance model:

> "EXPRESS allows you to create relationships between supertypes and subtypes that are not possible with the run of the mill Object Oriented language which typically only provides for simple hierarchies with a single root. EXPRESS allows multiple roots (inheritance) and overlap of the nodes in the lattice."
> (Schenk & Wilson 1994)

Figure 26 illustrates multiple inheritance, with an EXPRESS-G representation in figure 27.

```
ENTITY path
  SUPERTYPE OF (ONEOF(open_path, edge_loop, oriented_path))
  SUBTYPE OF (topological_representation_item);
  edge_list  : LIST [1:?] OF UNIQUE oriented_edge;
WHERE
  WR1: path_head_to_tail(SELF);
END_ENTITY;
ENTITY loop
  SUPERTYPE OF (ONEOF(vertex_loop, edge_loop, poly_loop))
  SUBTYPE OF (topological_representation_item);
END_ENTITY;
ENTITY edge_loop
  SUBTYPE OF (loop,path);
DERIVE
  ne : INTEGER := SIZEOF(SELF\path.edge_list);
WHERE
  WR1: (SELF\path.edge_list[1].edge_element.edge_start) :=:
       (SELF\path.edge_list[ne].edge_element.edge_end);
END_ENTITY;
```

Figure 26 - Multiple inheritance in EXPRESS, in STEP Part 42 (ISO 10303-42 1992)

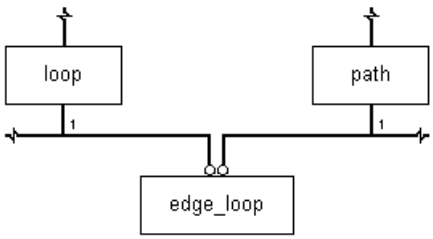


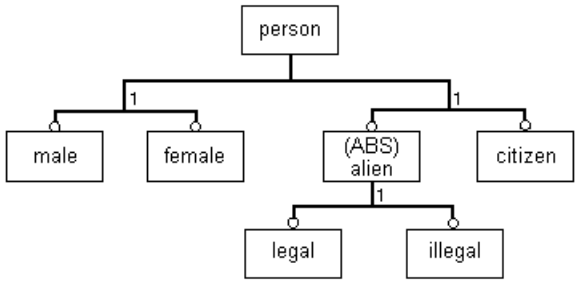
Figure 27 - EXPRESS-G diagram for the entities in figure 26 (detail)

Complex entities are those which are not explicitly defined as entities in an EXPRESS schema, but which result from the application of supertype constraints AND and ANDOR. Figure 28 presents an EXPRESS schema and the resulting allowable complex entity instantiations. Abstract supertypes are those that are not intended to be directly instantiated, but instead be indirectly instantiated in conjunction with at least one of its subtypes. For instance, the entity `alien`, because of the constraint ABSTRACT, may be instantiated only in conjunction with either `legal` or `illegal`.

(a)

```
SCHEMA example;
ENTITY person SUPERTYPE OF (ONEOF (male, female) AND
                            ONEOF(citizen, alien));
END_ENTITY;
ENTITY male SUBTYPE OF (person);
END_ENTITY;
ENTITY female SUBTYPE OF (person);
END_ENTITY;
ENTITY citizen SUBTYPE OF (person);
END_ENTITY;
ENTITY alien ABSTRACT SUPERTYPE OF (ONEOF(legal, illegal))
  SUBTYPE OF (person);
END_ENTITY;
ENTITY legal SUBTYPE OF (alien);
END_ENTITY;
ENTITY illegal SUBTYPE OF (alien);
END_ENTITY;
END_SCHEMA;
```

(b)



(c)

```
Allowable complex entity instantiations:
person&female&alien&illegal
person&female&alien&legal
person&female&citizen
person&male&alien&illegal
person&male&alien&legal
person&male&citizen
```

Figure 28 - (a) EXPRESS schema, (b) equivalent EXPRESS-G diagram, and (c) allowable complex entities, adapted from (ISO 10303-11 1994)

Rando & McCabe (1996) point out that supertype/subtype is a specialization relationship commonly used to express the ISA ("is a") abstraction, but that also can denote the PLAYS_ROLE_OF relationship. For instance, looking up the inheritance tree in figure 29, a student ISA person and, in figure 30, a car ISA vehicle. However, when looking down the tree, this vehicle IS ALWAYS a car, but this person PLAYS_ROLE_OF a student (Rando & McCabe 1996).

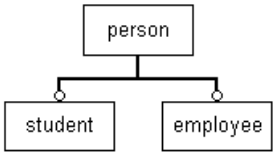


Figure 29 - EXPRESS-G diagram for the `person-student-employee` entity lattice

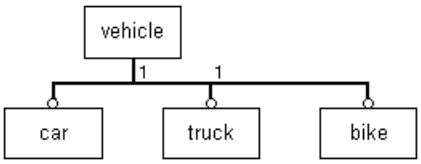


Figure 30 - EXPRESS-G diagram for the `vehicle-car-truck-bike` entity lattice

While a `car` doesn't change type after being created, the instances of the PLAYS_ROLE_OF relationship, such as `student` and `employee` in figure 29, may change types and assume multiple roles. Figure 32 presents a set representation for the ISA relationship in figure 30, with disjoint and invariant subtypes. Figure 31 illustrates the set representation for the entities in figure 29, with the new multiple-role subtype `student-employee`, which is not explicit in figure 29.

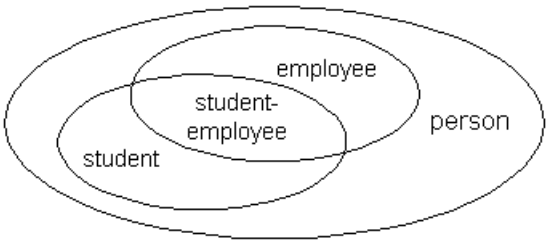


Figure 31 - Set representation for the entities in figure 29

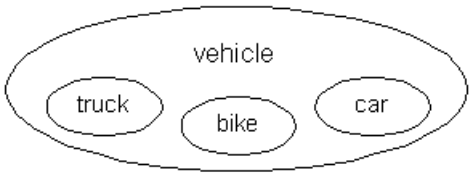


Figure 32 - Set representation for the entities in figure 30

The PLAYS_ROLE_OF specialization is modeled in EXPRESS with the SELECT data type, which works as a placeholder for other data types (Rando & McCabe 1996). An instance of a SELECT type can change its role dynamically. However, only a single role at a time is supported (Rando & McCabe 1996).

3.3 Concluding Remarks on the Integration of Product Data Models

In this chapter, the coupling of STEP and CORBA was presented as the standards-based technology for the integration of network interoperable product data models. Implications of this coupling were discussed relatively to three subject areas, corresponding to the three subsections of the literature review: database management technology, application interoperability, and product data representation.

The coupling of STEP and CORBA does not impose any technological choice on database management. STEP databases face the same challenges and open issues that any engineering database do, as discussed in section 2.1.4. STEP APs should provide the conceptual models for product data representation and sharing among disparate database systems, while CORBA works as an amalgam allowing for the network interoperable access to product data models among these different systems.

Regarding application interoperability, the use of CORBA as the enabling technology is supported by OMG, and by STEP through the IDL binding to the SDAI. CORBA is also used for application interoperability within the NIIIP project. Problems and open issues regarding application interoperability are not particular to CORBA.

The effectiveness of STEP data models for network interoperable access and sharing depends on the quality of the STEP data models (APs) and the quality of the translation of EXPRESS data models into IDL. Research is under way to provide for better quality of STEP data models, especially regarding AP interoperability, ontology engineering, improvements to the EXPRESS language, and a re-thinking of the building process using general-purpose IRs. STEP data models written in EXPRESS are translated into IDL for network interoperable access, according to the IDL binding to the SDAI (ISO 10303-26 1997), which defines specific mappings for each EXPRESS data type construct.

In order to help assess the translatability of STEP data models written in EXPRESS into IDL, and the maintainability of data model semantics across a distributed network, the next chapter presents an analysis of the semantic loss in the EXPRESS-IDL translation. A suite of STEP data model translations is proposed, where each model focuses on one aspect of the EXPRESS data type.

CHAPTER 4

ANALYSIS OF THE SEMANTIC LOSS IN THE TRANSLATION OF PRODUCT DATA MODELS FOR NETWORK INTEROPERABLE ACCESS

In this chapter, an analysis of the semantic loss in the translation of STEP product data models written in EXPRESS into IDL for network interoperable access is presented. Losses are assessed by comparing the syntax of the two languages, and by an ad hoc evaluation of the possible loss of meaning (semantics) when a STEP object is accessed through its IDL interface. This is the situation of the standards-based sharing of product data in virtual enterprises, where all partners use STEP for product data representation, and CORBA for network interoperable access.

Product data is not exactly *translated into* IDL, but rather *accessed through* IDL interfaces. The possibility of information loss is a different problem than the information loss in data translation between STEP to and from a specific CAx native data model. In the latter case, data has to *migrate* into a *different* data model. In the network access, loss happens only if the IDL interface is not capable of delivering the full detail of the STEP object accessed, or if spurious data is allowed to be delivered. For this reason, the analysis presented in this chapter is syntax-directed: EXPRESS data models are compared to their IDL interfaces, and loss is detected in cases where a syntactic mismatch between the two languages exist.

Sanderson (1994) introduces the abstract data model (ADM), a theoretical meta model of data modeling languages which presents the different features which can occur in any original modeling language. Table 9 shows the EXPRESS constructs corresponding to the generic features described by the ADM, allowing for an abstract or generic view of the language. These different types of EXPRESS constructs were analyzed regarding the semantic loss in the translation into IDL for the building of interfaces for network access. For the analysis of the semantic loss, an EXPRESS-IDL translation suite is presented in the next section. A series of 17 STEP schemata was produced, each one focusing on one aspect of the EXPRESS data type, as shown in figure 33. In each case analyzed, a “taxonomy of data structure similarity” is

referenced, as presented by Batini, Lenzerini and Navathe (1986) according to Sanderson (1994):

Table 9 - Features of a generic modeling language according to the ADM, and the corresponding EXPRESS constructs

ADM features	EXPRESS equivalent features	Observations
Constructors	Entity, Select, Enumeration, Subtype-Supertype	See translation cases t08, t09, t10, and t16.
Groupings	Array, Set, Bag, List	See translation cases t05, t06, t07, t13, and t15.
Inheritors	Type-Subtype	See translation cases t11, t12, and t17.
Primitives	Binary, Integer, Logical, String	See translation cases t01, t02, t03, t04, and t14.
Alphabet	EXPRESS character set	
Object identifiers	Not-specified	

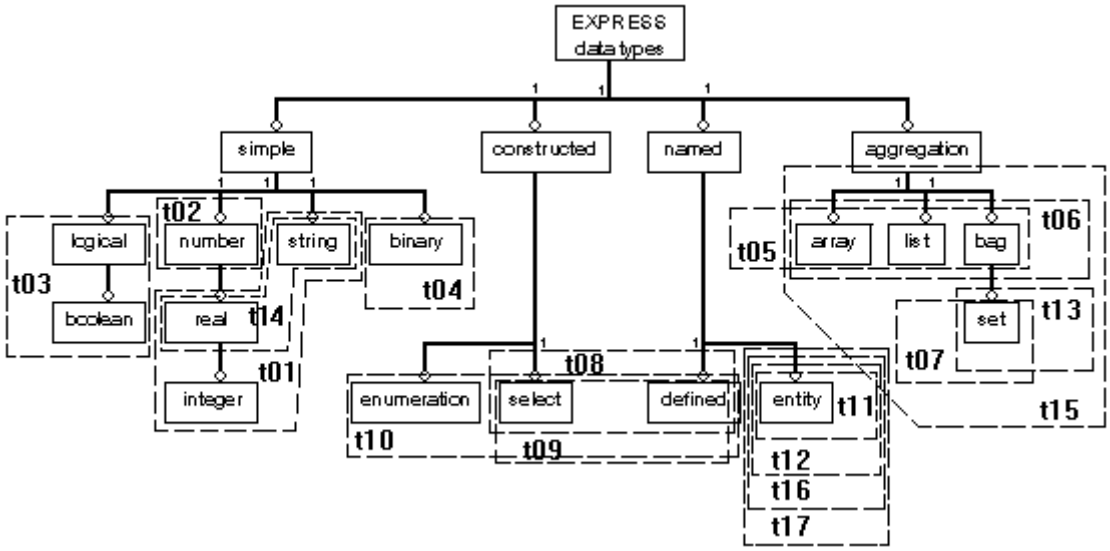


Figure 33 - Constructs from the EXPRESS data type tree focused in the 17 translation cases

- Identical -- the structures are exactly the same, leading to translations with no loss of information;
- Equivalent -- the structures are based on equivalent assumptions;
- Compatible -- the underlying assumptions of the two models are not in conflict, although the structures are not equivalent; and

- Incompatible -- there are basic assumptions in the two languages that conflict with each other, and attempts of translation will lead to structural (and therefore semantic) loss.

In order to produce realistic translation cases, the STEP schemata were built from parts of real STEP APs and IRs obtained from the SC4 On Line Information Service at NIST (SOLIS 1996). APs and IRs were searched for EXPRESS constructs which could represent the full detail of the EXPRESS data type. The reason for this choice is that APs are the STEP schemata to be implemented in conjunction with SDAI or other STEP implementation method, while IRs are the generic information models which provide the basic constructs interpreted (or *reused*) within the APs. Therefore, the choice for an EXPRESS schema adapted from the APs leads to a real-world application of the EXPRESS-IDL translation. In cases of data types which were not present in the APs, the adaptation was made from the IRs, since they provide the constructs which may be used in the future APs. The appendix lists several in-progress APs.

While examining the available AP and IR schemata, it was found that not all EXPRESS data types were present. To overcome this, necessary schemata were therefore adapted from examples in the SDAI specification (ISO 10303-22 1995) and EXPRESS specification, STEP Part 11 (ISO 10303-11 1994). Table 10 lists, for each of these schemata, the STEP document it was adapted from and the correspondent entities used, so the interested reader can reconstruct the schema. For instance, in table 10, case t01, the entity `e0` in the generated schema was adapted from `ENTITY cartesian_transformation_operator`, which was present in the AP 214. Some constructs of the EXPRESS data type had more than one schema built for translation, in which cases different details of the construct usage were depicted in the STEP documents. For instance, case t15 focuses on aggregated data types `ARRAY`, `LIST`, and `BAG`, which are also depicted in case t05. However, case t15 comprises aggregates in which restrictions (`OPTIONAL` and `UNIQUE`) that were not present in case t05 are depicted.

The translation from EXPRESS to IDL was performed according to the IDL binding to the SDAI, STEP Part 26 (ISO 10303-26 1997). In this binding, only the static aspect of STEP data models is accounted for. The mapping of dynamic aspects, such as `PROCEDURE`s, and `FUNCTION`s, is not part of the SDAI, nor does the SDAI currently address database management issues such as transactions, versioning, and concurrency control.

Table 10 - Adaptations from STEP documents to build the translation cases (continues...)

Translation case	Adapted from	Original structures present in STEP schemata	Corresponding structures in generated schemata
t01 - REAL, INTEGER, and STRING	AP 214 (ISO 10303-214 1995)	cartesian_transformation_operator scale colour_rgb red curve_style name geometric_representation_context coordinate_space_dimension	e0 e0attr e1 e1attr e2 e2attr e3 e3attr
t02 - NUMBER	AP 214 (ISO 10303-214 1995)	sequential_method sequence_position	e0 e0attr
t03 - BOOLEAN and LOGIC	AP 201 (ISO 10303-201 1994)	b_spline_curve closed_curve camera_model_d2 view_window_clipping	e0 e0attr e1 e1attr
t04 - BINARY	Made up -- no BINARY constructs were present in the STEP schemata.		
t05 - ARRAY, BAG, and LIST	AP 214 (ISO 10303-214 1995)	composite_curve segments composite_curve_segment using_curves curve_element_end_offset offset_vector rectangular_composite_surface segments surface_patch using_surfaces	e1 e1attr e2 e2attr e3 e3attr e6 e6attr e7 e7attr
t06 - unbounded ARRAY, BAG, and LIST	Made up -- no unbounded aggregates were present in the STEP schemata.		
t07 - SET	AP 201 (ISO 10303-201 1994)	annotation_text composite_text collected_text trimmed_curve trim_1	e1 e3 e3attr e6 e6attr
t08 - SELECT	AP 201 (ISO 10303-201 1994)	geometric_representation_item curve point geometric_set offset_curve_2d geometric_set_select (TYPE)	ea0 eb0 eb1 eb2 ec0 sel

Table 10 (continued)

t09 - nested SELECTs	EXPRESS specification, Example 36 (ISO 10303-11 1994)	attachment_method (TYPE) permanent_attachment (TYPE) temporary_attachment (TYPE) nail screw glue weld wall_mounting	sel0 sel1 sel2 e0 e1 e2 e3 e4
t10 - ENUMERATION	AP 201 (ISO 10303-201 1994)	b_spline_curve b_spline_curve_form	e0 t_enum
t11 - complex entity data types	SDAI specification, Example 8 (ISO 10303-22 1995)	a b c d e f g h	a0 b0 b1 c0 d0 d1 d2 d3
t12 - complex entity data types	AP 203 (ISO 10303-203 1994)	action_assignment change start_work	e0 e1 e2
t13 - unbounded SET	AP 204 (ISO 10303-204 1996)	representation representation_context	e0 e1
t14 - REAL and STRING	Made up from definitions in Part 11, 8.1.6 (ISO 10303-11 1994)		
t15 - ARRAY, LIST, and BAG	Made up from definitions in Part 11 (ISO 10303-11 1994)		
t16 - entity attributes	Made up from definitions in Part 11, 9.1 and 9.2 (ISO 10303-11 1994)		
t17 - multiple inheritance	IR 42 (ISO 10303-42 1992)	path loop edge_loop	e0 e1 e2

The translation was performed in part manually and in part using the translator *exp2idl* from the ST-Developer toolkit from STEP Tools. At the time of the translation, only part of the EXPRESS data type was translatable using *exp2idl*. The IDL sources produced were then compiled using IONA's Orbix IDL compiler for syntax checking. All translations and compilations ran on a Sun Sparcstation 10 at the Manufacturing Engineering Laboratory of the National Institute of Standards and Technology (NIST).

4.1 An EXPRESS-IDL translation suite

This section presents the 17 cases of the translation suite. Each consists of an EXPRESS schema and its translation into IDL, and an analysis of the potential semantic loss during translation.

4.1.1 Translation of REAL, INTEGER, and STRING data types

The purpose of this translation case is to investigate the loss regarding REAL, INTEGER, and STRING data types. Figure 34 presents the EXPRESS schema t01, followed by its translation into IDL, in figure 35.

Comments regarding translation case t01

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the interfaces in figure 35 separated in different files referenced (included) in a file called t01.idl. Then, the interfaces were manually made explicit within t01.idl for simplicity of presentation. The syntax of the resulting IDL source, compliant with the mapping defined in the IDL binding to the SDAI (ISO 10303-26 1997), was successfully verified using Orbix.

Losses observed in case t01

The IDL structures in figure 35 are the *interfaces* to be published and made available through object request brokers to allow for the network access to the correspondent EXPRESS entities. Each interface should provide the appropriate constructs to accommodate all of the entity's attributes in full detail.

In case t01, IDL constructs `double`, `string`, and `long` allow for the representation of the respective EXPRESS constructs `REAL`, `STRING`, and `INTEGER`. At this point, nothing can be said about how adequate the IDL constructs are for the representation of its EXPRESS equivalents, however it can be noted that IDL syntax resembles that of C++, and `double`, `string`, and `long` in C++ are adequate representations of the range and type of values of EXPRESS' `REAL`, `STRING`, and `INTEGER`. But, IDL is not an implementation language. It can be implemented in C, C++, or other implementation language whose binding to IDL is defined. The matching of `REAL`, `STRING`, and `INTEGER` should be examined with respect to the client implementation language of choice.

```
(* t01.exp

     Purpose of the schema: to represent simple data types
     REAL, INTEGER, and STRING.

*)

SCHEMA t01;

 ENTITY e0;
  e0attr : OPTIONAL REAL;
 END_ENTITY;

 ENTITY e1;
  e1attr : REAL;
 END_ENTITY;

 ENTITY e2;
  e2attr : STRING;
 END_ENTITY;

 ENTITY e3;
  e3attr : INTEGER;
 END_ENTITY;

END_SCHEMA;
```

Figure 34 - EXPRESS schema t01

```
/* t01.idl
*/

  interface  E0 {
     attribute double e0attr;
  };

  interface  E1 {
     attribute double e1attr;
  };

  interface  E2 {
     attribute string e2attr;
  };

  interface  E3 {
     attribute long e3attr;
  };
```

Figure 35 - Translation of schema t01 into IDL

The mismatch which *can* be observed, representing semantic loss in the translation case t01, is the lack of an IDL equivalent to the `OPTIONAL` directive of attribute `e0attr` of entity `e0`. `OPTIONAL` indicates that the attribute in question need not have a value. This is ignored in the mapping to IDL according to STEP Part 26 (ISO 10303-26 1997). Therefore, an `OPTIONAL REAL` attribute belonging to an EXPRESS entity instance made available across a network will be accessed simply as a `REAL` attribute. The same reasoning is valid for `REAL` or `STRING` attributes.

When an `OPTIONAL REAL` attribute whose instance does not exist is accessed as `REAL`, the result (the value assigned to the attribute) is unpredictable, and this is a serious problem. A possible ad hoc solution for this loss is to implement objects which deliver a flag value, i.e., a value out of the range, which represents that this instance does not exist. This ad hoc solution may work if all the clients understand this convention.

In summary, EXPRESS and IDL are *incompatible* with respect to the `OPTIONAL REAL` feature. Other simple data types with the `OPTIONAL` directive may generate similar problems if future STEP schemata use them. With respect to the other constructs present in case t01, EXPRESS and IDL are *equivalent*.

4.1.2 Translation of NUMBER data type

The purpose of this translation case is to investigate the semantic loss regarding the simple data type NUMBER. Figure 36 presents the EXPRESS schema t02, followed by the translation into IDL, in figure 37.

```
(* t02.exp

     Purpose of the schema: to represent simple data type
     NUMBER.
*)

SCHEMA t02;

 ENTITY e0;
  e0attr : NUMBER;
 END_ENTITY;

END_SCHEMA;
```

Figure 36 - EXPRESS schema t02

```
/* t02.idl
*/

  interface  E0 {
      enum Number_discriminant { Number_discriminant__long,
      Number_discriminant__double };

      union Number switch ( Number_discriminant ) {
      case Number_discriminant__long : long c1;
      case Number_discriminant__double : double c2;
      };

      attribute Number e0attr;
  };
```

Figure 37 - Translation of schema t02 into IDL

Comments regarding translation case t02

This IDL translation was produced in part using the *exp2idl* translator, which produced the interface E0 without the `enum` and `union` blocks in a different file referenced in t02.idl. The interface E0 was then included manually in t02.idl for simplicity of presentation, and the `enum` and `union` blocks were included also manually to make the IDL translation compatible with the specification of the IDL binding to the SDAI (ISO 10303-26 1997). The syntax of the resulting IDL source was successfully verified using Orbix.

Losses observed in case t02

The EXPRESS data type `NUMBER` depicted in case t02 refers to attributes that may be either `REAL` or `INTEGER`. It is used when “a more specific numeric representation is not important” (ISO 10303-11 1994). Any *instance* of a `NUMBER` attribute, however, will be implemented in the *format* of either `REAL` or `INTEGER`. In this sense, the IDL translation of `Number` as a `switch` between `double` and `long` types is adequate to represent the EXPRESS `NUMBER` which may be either a `REAL` or an `INTEGER`. Therefore, no semantic loss in the EXPRESS-IDL translation is observed regarding the `NUMBER` data type. In summary, EXPRESS and IDL are *compatible* with respect to the `NUMBER` data type, i.e., their structures are not equivalent, but they are not in conflict either.

4.1.3 Translation of BOOLEAN and LOGIC data types

The purpose of this translation case is to investigate the semantic loss regarding BOOLEAN and LOGIC data types. Figure 38 presents the EXPRESS schema t03, followed by the translation into IDL, in figure 39.

```
(* t03.exp

     Purpose of the schema: to represent simple data types
     BOOLEAN and LOGIC.
*)

SCHEMA t03;

  ENTITY e0;
      e0attr : LOGICAL;
  END_ENTITY;

  ENTITY e1;
      e1attr : BOOLEAN;
  END_ENTITY;

END_SCHEMA;
```

Figure 38 - EXPRESS schema t03

```
/* t03.idl
*/

  enum Logical { LFalse, LTrue, LUnset, LUnknown };

  enum Boolean { BFalse, BTrue, BUnset };

  interface  E0 {
     attribute Logical e0attr;
  };

  interface  E1 {
     attribute Boolean e1attr;
  };
```

Figure 39 - Translation of schema t03 into IDL

Comments regarding translation case t03

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the interfaces in figure 39 separated in different files which were included manually in t03.idl for simplicity of presentation. The syntax of the resulting IDL source was successfully verified using Orbix.

Losses observed in case t03

The IDL translation in figure 39 provides interfaces with `Logical` and `Boolean` attributes similar to the EXPRESS entities with `LOGICAL` and `BOOLEAN` attributes in figure 38. These `Logical` and `Boolean` attributes are built as an enumeration of the possible values to be assumed by the entity attributes, with the difference that, in IDL, the attributes are allowed to be unset (`LUnset` and `BUnset`). However, any STEP entity attribute accessed through an IDL interface should have a valid instance value (i.e., a `LOGICAL` attribute should be either `TRUE`, `FALSE`, or `UNKNOWN`; and a `BOOLEAN` should be either `TRUE` or `FALSE`). If the access happens to fail in the deliverance of a valid value, then the result is unpredictable and causes semantic loss (i.e., how would the attribute be instantiated if the data received is not valid?). Checking for the validity of a `LOGICAL` or `BOOLEAN` attribute received (and forcing the rollback of an access with invalid instances) should prevent this semantic loss from happening. This procedure could be part of a SDAI transaction mechanism, which is not currently addressed in STEP.

In summary, regarding the `LOGICAL` and `BOOLEAN` data types, EXPRESS and IDL are *incompatible* because of the unset instance values in IDL. This, however, should have a minor effect in the sharing of product data, with no semantic loss expected, since the accessed data should always deliver a valid instance (not unset) of a `LOGICAL` or `BOOLEAN` attribute.

4.1.4 Translation of BINARY data type

The purpose of this translation case is to investigate the semantic loss regarding the data type BINARY. Figure 40 presents the EXPRESS schema t04, followed by the translation into IDL, in figure 41.

Comments regarding translation case t04

The IDL translation produced using the *exp2idl* translator generated code which is not completely in accord to the IDL binding to the SDAI (ISO 10303-26 1997) specification. This automatic translation is included in figure 41 as a comment (limited by "`/*`" and

```
(* t04.exp

     There is no SOLIS-available schema that includes a
     BINARY construct.

     Purpose of the schema: to represent simple data type
     BINARY.

     Produced according to Part 11, item 8.1.7 (Binary data
     type).
*)

SCHEMA t04;

 ENTITY e0;
  e0attr0 : BINARY;
  e0attr1 : BINARY (8);
  e0attr2 : BINARY (4) FIXED;
 END_ENTITY;

END_SCHEMA;
```

Figure 40 - EXPRESS schema t04

```
/* t04.idl
*/

typedef sequence<octet> Binary;

interface  E0 {

     attribute Binary e0attr0;
     attribute Binary e0attr1;
     attribute Binary e0attr2;

     /* Part 26 5.2.1.7, mapping of BINARY to IDL: "The name
     of the sequence shall be Binary."
     The following is exp2idl's mapping from EXPRESS. Binary
     size is declared directly with the attribute.

     attribute sequence<octet> e0attr0;
     attribute sequence<octet, 2> e0attr1;
     attribute sequence<octet, 1> e0attr2;
     */
};
```

Figure 41 - Translation of schema t04 into IDL

"*/"). The correct translation for BINARY is the IDL Binary type preceded by its definition (typedef), since it is not a primitive data type in IDL. The syntax of the resulting IDL source was successfully verified using Orbix.

Losses observed in case t04

The EXPRESS entities which carry BINARY attributes in figure 40 are translated into IDL interfaces using the Binary type. The *size* or maximum number of bits (which *can* be specified as a number between parentheses, but it's not mandatory) of BINARY attributes, however, are lost in the IDL translation, as well as the FIXED character.

The BINARY data type can be used to specify either varying unlimited width (as in attribute e0attr0 in figure 40), varying limited width (as in attribute e0attr1 in figure 40--limited to 8 bits), or fixed width (as in attribute e0attr2 in figure 40--fixed size of 4 bits). The IDL interface translated from the EXPRESS entity in figure 40 represents all binaries equally, i.e., without any information about limit or FIXED character. This is adequate to represent varying width binaries, only. The automatic translation produced using *exp2idl*, shown as comment in figure 41, is adequate to represent limits, but still does not represent the FIXED character. Moreover, the standard EXPRESS-IDL mapping makes it clear that Binary should be a typedef predeclaration (ISO 10303-26 1997), which makes it impossible to specify the size. Therefore, these is semantic loss in the translation of the BINARY data type into IDL.

In summary, EXPRESS and IDL are *incompatible* regarding the BINARY data type. Every time a limited width BINARY is accessed through an IDL interface, information about the width is missing. A possible way to alleviate this loss would be to include, in the interface, an integer field with the number of bits associated to the BINARY attribute. In the standards-based sharing, however, no loss is expected since both ends of the transmission use the same STEP schema, and whatever is missing in the IDL interface is filled in when data is checked against the receiver STEP schema.

4.1.5 Translation of ARRAY, BAG, and LIST data types

The purpose of this translation case is to investigate the semantic loss regarding the aggregate data types ARRAY, BAG, and LIST. Figure 42 presents the EXPRESS schema t05, followed by the translation into IDL, in figure 43.

Comments regarding translation case t05

The IDL translation was initiated using the *exp2idl* translator. This automatic translation produced the file t05.idl with the interfaces in different files, referenced

```
(* t05.exp

     Purpose of the schema: to represent aggregates ARRAY,
     BAG, and LIST (all but SET).  Observation: BAG appears
     only in INVERSE attributes in the STEP schemata.
*)

SCHEMA t05;

 ENTITY e1;
  e1attr : LIST[1:?] OF e2;
 END_ENTITY;

 ENTITY e2;
  INVERSE
   e2attr : BAG[1:?] OF e1 FOR e1attr;
 END_ENTITY;

 ENTITY e3;
  e3attr : ARRAY[1:3] OF REAL;
 END_ENTITY;

 ENTITY e6;
  e6attr : LIST[1:?] OF LIST[1:?] OF e7;
 END_ENTITY;

 ENTITY e7;
  INVERSE
   e7attr : BAG[1:?] OF e6 FOR e6attr;
 END_ENTITY;

END_SCHEMA;
```

Figure 42 - EXPRESS schema t05

(`included`) in t05.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 43.

Because the automatic translation was not standard-compliant, the following change was performed manually: the construct `double` was substituted by `Double`, which is the standard translation of `REAL`. The syntax of the resulting IDL file was checked using Orbix, with errors detected: the standard mapping of an `INVERSE` attribute should be a read-only attribute with the suffix "`__list`", which is not accepted by Orbix. The reason for this is unknown, but suggests a possible incompleteness of the compiler.

```
/* t05.idl
*/

interface  E2;
typedef sequence<E2> E2__bounded_list;

interface  E1;
typedef sequence<E1> E1__bounded_bag;

typedef sequence<Double> Double__bounded_array;

interface  E7;
typedef sequence<sequence<E7> > E7__bounded_list__bounded_list;

interface  E6;
typedef sequence<E6> E6__bounded_bag;

interface  E1  {
     attribute E2__bounded_list e1attr;
};

interface  E2  {
     readonly attribute E1__bounded_bag e2attr;
};

interface  E3  {
     attribute Double__bounded_array e3attr;
};

interface  E6  {
     attribute E7__bounded_list__bounded_list e6attr;
};

interface  E7  {
     readonly attribute E6__bounded_bag__list e7attr;

/*   The above translation, according to Part 26, 5.2.6,
     was not accepted by the Orbix idl compiler.
     Errors annotated:
     49:(semantic): Identifier `E6__bounded_bag__list' not found
     49:(semantic): Name does not denote a type
     Translation accepted by the compiler:

     readonly attribute E6__bounded_bag e7attr;
*/

};
```

Figure 43 - Translation of schema t05 into IDL

Losses observed in case t05

The IDL translation of the aggregates in figure 42 offers specific constructs for the implementation of interface attributes for ARRAY, BAG, and LIST. These interfaces,

however, do not cover the full detail of EXPRESS aggregates because the indexes and size limits are not represented in IDL. This may be considered a minor problem in the cases of BAG and LIST because generic IDL attributes (i.e., without indexes or limits) can still be used to convey the correspondent indexed and limited EXPRESS aggregates. In the case of ARRAY, however, there is no "generic" alternative. Any array is always an indexed and limited structure. A receiving client of an array accessed through an IDL interface has no information on how to determine the base index (which in the example of figure 42 is 1), or the size of the array.

In summary, EXPRESS and IDL are *incompatible* with respect to bounded aggregates ARRAY, BAG, and LIST. This is a serious problem of semantic loss in the case of ARRAY. Only an external control (for instance, the inclusion in the interface of two integers for each array carrying its size and the base index, making sure all clients in the network understand this) can alleviate this loss. The same principle can be applied to BAGs and LISTs. In the standards-based sharing, however, no loss is expected since both ends of the transmission use the same STEP schema, and whatever is missing in the IDL interface is filled in when data is checked against the receiver STEP schema.

4.1.6 Translation of unbounded BAG and LIST data types

The purpose of this translation case is to investigate the semantic loss regarding unbounded aggregate data types BAG and LIST. Figure 44 presents the EXPRESS schema t06, followed by the translation into IDL, in figure 45.

Comments regarding translation case t06

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t06.idl with the interfaces in different files, referenced (included) in t06.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 45. The syntax of the resulting IDL source was successfully verified using Orbix.

The EXPRESS schema t06.exp was made up to allow for the building of a translation case involving unbounded aggregates, which are not present in the STEP schemata, except for unbounded SETs (see translation case t13). An unbounded ARRAY was inadvertently included in the schema, and *exp2idl* translated it. However, upon verification, since unbounded ARRAYs do not belong in EXPRESS, the entity e0 was commented out (put between "(*" and "*)") of the schema. Also, *exp2idl* was translated INTEGER into IDL's long (as well as BAG of INTEGER and LIST of

```
(* t06.exp

     Purpose of this schema: to represent unbounded
     aggregates ARRAY, BAG and LIST.
*)

SCHEMA t06;

(* ENTITY e0;
     e0attr: ARRAY OF REAL;
 END_ENTITY;

  Unbounded ARRAYs do not belong in STEP syntax, therefore
  this entity was taken off the translation suite.
*)

 ENTITY e1;
  e1attr : BAG OF INTEGER;
 END_ENTITY;

 ENTITY e2;
  e2attr : LIST of INTEGER;
 END_ENTITY;

END_SCHEMA;
```

Figure 44 - EXPRESS schema t06

```
/* t06.idl
*/

typedef sequence< Long > Long__list;
typedef sequence< Long > Long__bag;

interface  e1 {
     attribute Long__list e1attr;
};

interface  e2 {
     attribute Long__bag e2attr;
};
```

Figure 45 - Translation of schema t06 into IDL

INTEGER into long__list and long__bag, respectively). This was changed manually to Long, Long__list, and Long__bag, to make it compliant with the

standard mapping, which states that the translated type should have the first character represented as a capital letter.

Losses observed in case t06

The entities with unbounded aggregates `BAG of INTEGER` and `LIST of INTEGER` in figure 44 are mapped into the IDL constructs `Long__list` and `Long__bag`, respectively, which are defined (`typedef`) in figure 45. This translation allows for a one-to-one EXPRESS-IDL mapping, i.e., it covers the full detail of the EXPRESS unbounded aggregates, and an IDL interface which contains unbounded aggregates have no other possible interpretation. Therefore, no loss or ambiguity is associated to the EXPRESS-IDL translation in case t06. In summary, EXPRESS and IDL are *compatible* with respect to unbounded aggregates `BAG` and `LIST`.

4.1.7 Translation of SET data type

The purpose of this translation case is to investigate the semantic loss regarding the aggregate data type SET. Figure 46 presents the EXPRESS schema t07, followed by the translation into IDL, in figure 47.

Comments regarding translation case t07

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t07.idl with the interfaces in different files, referenced (`included`) in t07.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 47. The syntax of the resulting IDL source was successfully verified using Orbix.

The `SET of REAL` in entity `e6` in figure 46 was translated by *exp2idl* as `double__bounded_set`. In order to comply with the standard translation, this was manually changed to `Double__bounded_set`.

Losses observed in case t07

The IDL translation in figure 47 allows for the representation of EXPRESS bounded SETs in IDL. However, the translated `Double__bounded_set` does not permit the specification of the size and indexes. This may be considered a minor problem because a generic SET (i.e., without indexes or limits) can still be used to convey the correspondent indexed and limited SET. In summary, EXPRESS and IDL are *incompatible* with respect to bounded SETs. In the standards-based sharing, however, no loss is expected since both ends of the transmission use the same STEP schema; sizes and indexes are filled in when data is checked against the receiver schema.

```
(* t07.exp

     Purpose of the schema: to represent (bound) aggregate SET.
*)

SCHEMA t07;

  ENTITY e1;
  END_ENTITY;

  ENTITY e3;
      e3attr : SET [2:?] OF e1;
  END_ENTITY;

  ENTITY e6;
      e6attr : SET [1:2] OF REAL;
  END_ENTITY;

END_SCHEMA;
```

Figure 46 - EXPRESS schema t07

```
/* t07.idl
*/

interface  E1;
typedef sequence<E1> E1__bounded_set;
typedef sequence< Double > Double__bounded_set;

interface  E1  {
};

interface  E3  {
     attribute E1__bounded_set e3attr;
};

interface  E6  {
     attribute Double__bounded_set e6attr;
};
```

Figure 47 - Translation of schema t07 into IDL

4.1.8 Translation of SELECT data type

The purpose of this translation case is to investigate the semantic loss regarding the SELECT data type. It also illustrates the translation of EXPRESS defined data types,

i.e., those defined in TYPE blocks. Figure 48 presents the EXPRESS schema t08, followed by the translation into IDL, in figure 49.

```
(* EXPRESS schema t08.exp

     Purpose of the schema: to represent SELECT.
     Observation: Named (TYPE) types, inheritance, and a SET
     of selects are also represented.

          ea0
          |1
     ------------
     |    |    |
     eb0  eb1  eb2   * eb2 is an entity whith an attribute
     |               * which is a select set (SET [1:?] of sel)
     |               * of eb0 and eb1.
     ec0             * (Adapted from AP 201)
*)

SCHEMA t08;

  TYPE sel = SELECT
    (eb0,
     eb1);
  END_TYPE;

  ENTITY ea0
    SUPERTYPE OF (ONEOF (eb0,eb1,eb2));
  END_ENTITY;

  ENTITY eb0
    SUPERTYPE OF (ec0)
    SUBTYPE OF (ea0);
  END_ENTITY;

  ENTITY eb1
    SUBTYPE OF (ea0);
  END_ENTITY;

  ENTITY eb2
    SUBTYPE OF (ea0);
      a0attr : SET [1:?] OF sel;
  END_ENTITY;

  ENTITY ec0
    SUBTYPE OF (eb0);
  END_ENTITY;

END_SCHEMA;
```

Figure 48 - EXPRESS schema t08

```
/* t08.idl
*/

interface  Eb0;
interface  Eb1;
enum Sel_select {
     Sel__Eb0,
     Sel__Eb1};
union Sel switch (Sel_select) {
case Sel__Eb0 : Eb0 c1;
case Sel__Eb1 : Eb1 c2;
};

interface  Ea0  {
};

interface  Eb0  : Ea0 {
};

interface  Eb1  : Ea0 {
};

typedef  sequence< Sel > Sel__bounded_set;
interface  Eb2  : Ea0 {
     attribute Sel__bounded_set a0attr;
};

interface  Ec0  : Eb0 {
};
```

Figure 49 - Translation of schema t08 into IDL

Comments regarding translation case t08

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t08.idl with the interfaces in different files, referenced (`included`) in t08.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 49. The syntax of the resulting IDL source was successfully verified using Orbix.

The *exp2idl* translation produced variable names not complying with the standard mapping specification. The order in which the name `select_sel` was presented is wrong. This was changed manually to `Sel_select`. Also, other variable names (`sel__eb0`, `sel__eb1`, `sel`, and `sel__bounded_set`) had to be manually altered to make the letter capitalization comply with the definition in the standard mapping.

Losses observed in case t08

EXPRESS SELECTs are represented in IDL as a choice (switch) among an enumeration of values (enum) of different *types*, as shown in figures 48 and 49. This is an appropriate representation since that is exactly what a SELECT attribute is; an attribute which can assume a value whose domain is one in a list of different types. Also, this IDL representation cannot be confused with any other EXPRESS construct. Therefore, no loss or ambiguity is associated to the EXPRESS-IDL translation of SELECT. In summary, EXPRESS and IDL are *equivalent* with respect to SELECT.

4.1.9 Translation of nested SELECTs

The purpose of this translation case is to investigate the semantic loss regarding nested SELECT data types. Figure 50 presents the EXPRESS schema t09, followed by its translation into IDL, in figure 51.

Comments regarding translation case t09

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t09.idl with the interfaces in different files, referenced (included) in t09.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 51. The syntax of the resulting IDL source was successfully verified using Orbix.

Minor changes had to be made to the IDL source to make it compliant to the standard mapping, similarly as in previous cases: select_sel0, select_sel1, and select_sel2 were changed to Sel0_select, Sel1_select, and Sel2_select, respectively. Also, variable names capitalization was reviewed, letting the first letter be capital.

Losses observed in case t09

This case differs from case t08 in that it presents *nested* SELECTs, i.e., SELECTs of SELECTs. As in case t08, the IDL translation of EXPRESS SELECT is a choice (switch) among an enumeration of values (enum) of different *types*, as shown in figures 50 and 51. In the cases where these different types were also SELECTs, the standard translation produces a choice (switch) among an enumeration (enum) of different choices (switch) among enumerations (enum) of different types. In other words, it is possible to extend the same principle to represent nested SELECTs in IDL. The IDL translated code, also, has no other possible correspondent in EXPRESS. Therefore, a STEP entity containing SELECTs can be accessed across an

interoperable network without ambiguity or data loss. In summary, EXPRESS and IDL are *equivalent* with respect to nested SELECTs.

```
(* t09.exp

     Purpose of the schema: to represent nested SELECT.
     Adapted from Part 11, Example 36.
*)

SCHEMA t09;

  TYPE sel0 = SELECT
    (sel1,
     sel2);
  END_TYPE;

  TYPE sel1 = SELECT
    (e0,
     e1);
  END_TYPE;

  TYPE sel2 = SELECT
    (e2,
     e3);
  END_TYPE;

  ENTITY e0;
  END_ENTITY;

  ENTITY e1;
  END_ENTITY;

  ENTITY e2;
  END_ENTITY;

  ENTITY e3;
  END_ENTITY;

  ENTITY e4;
    e4attr : sel0;
  END_ENTITY;

END_SCHEMA;
```

Figure 50 - EXPRESS schema t09

```
/*t09.idl
*/

interface  E0;
interface  E1;
enum Sel1_select {
     Sel1__E0,
     Sel1__E1};
union Sel1 switch (Sel1_select) {
case Sel1__E0 : E0 c1;
case Sel1__E1 : E1 c2;
};

interface  E2;
interface  E3;
enum Sel2_select {
     Sel2__E2,
     Sel2__E3};
union Sel2 switch(Sel2_select) {
case Sel2__E2 : E2 c1;
case Sel2__E3 : E3 c2;
};

enum Sel0_select {
     Sel0__Sel1,
     Sel0__Sel2};
union Sel0 switch (Sel0_select) {
case Sel0__Sel1 : Sel1 c1;
case Sel0__Sel2 : Sel2 c2;
};

interface  E0  {
};

interface  E1  {
};

interface  E2  {
};

interface  E3  {
};

interface  E4  {
     attribute Sel0 e4attr;
};
```

Figure 51 - Translation of schema t09 into IDL

4.1.10 Translation of ENUMERATION data type

The purpose of this translation case is to investigate the semantic loss regarding ENUMERATION data type. Figure 52 presents the EXPRESS schema t10, followed by the translation into IDL, in figure 53.

```
(* t10.exp

     Purpose of the schema: to represent ENUMERATION.
*)

SCHEMA t10;

  TYPE t_enum = ENUMERATION OF
     (te0,
     te1,
     te2);
  END_TYPE;

  ENTITY e0;
      e0attr    : t_enum;
  END_ENTITY;

END_SCHEMA;
```

Figure 52 - EXPRESS schema t10

```
/* t10.idl
*/

enum T_enum {
     T_enum__te0,
     T_enum__te1,
     T_enum__te2,
     T_enum__unset
};

interface  E0  {
     attribute T_enum e0attr;
};
```

Figure 53 - Translation of schema t10 into IDL

Comments regarding translation case t10

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t10.idl with the interface `E0` in figure 53 in a different file, referenced (`included`) in t10.idl. The enumerator identifier "_unset", required by the standard translation specification, was not generated by *exp2idl*, and therefore was included manually. The interface was then manually made explicit within the main file for simplicity of presentation. The syntax of the resulting IDL source was successfully verified using Orbix.

Losses observed in case t10

EXPRESS `ENUMERATION`s in figure 52 are mapped into the IDL construct `enum`, using a syntax which is very close to EXPRESS', as shown in figure 53. This translation allows for a one-to-one EXPRESS-IDL mapping, i.e., it covers the full detail of the EXPRESS construct `ENUMERATION`, and an IDL interface which contains the construct `enum` have no other possible interpretation. The only mismatch is that IDL allows that an `unset` value be assumed. However, any EXPRESS `ENUMERATION` accessed through an IDL interface should have a valid instance value, which does not include `unset`.

In summary, EXPRESS and IDL are *incompatible* with respect to `ENUMERATION` because of the `unset` value in IDL. Even so, no semantic loss is expected since the accessed data should always deliver a valid instance (not `unset`) for an `ENUMERATION`.

4.1.11 Translation of complex entity data types

The purpose of this translation case is to investigate the semantic loss regarding complex entity data types. Figure 54 presents the EXPRESS schema t11, followed by the translation into IDL, in figure 55.

Comments regarding translation case t11

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t11.idl with the interfaces in different files, referenced (`included`) in t11.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 55. The syntax of the resulting IDL source was successfully verified using Orbix.

```
(* t11.exp

     Purpose of the schema: to represent complex entity data
     types.

           (ABS) a0
            ANDOR
           /     \
          /       \
        b0         b1
         |
         |
        c0
        AND
   ONEOF   ANDOR
  /    |   |    \
 /     |   |     \
d0     d1  d2    d3
*)

SCHEMA t11;
 ENTITY a0 ABSTRACT SUPERTYPE OF ( b0 ANDOR b1 ); END_ENTITY;
 ENTITY b0 SUBTYPE OF (a0); END_ENTITY;
 ENTITY b1 SUBTYPE OF (a0); END_ENTITY;
 ENTITY c0 SUPERTYPE OF (ONEOF (d0, d1) AND (d2 ANDOR d3))
  SUBTYPE OF (b0);
 END_ENTITY;
 ENTITY d0 SUBTYPE OF (c0); END_ENTITY;
 ENTITY d1 SUBTYPE OF (c0); END_ENTITY;
 ENTITY d2 SUBTYPE OF (c0); END_ENTITY;
 ENTITY d3 SUBTYPE OF (c0); END_ENTITY;

 (* According to Part 26 (N396, 5.2.7, Jan 18,96), the
 "COMPLEX" block below should be added.  However, no
 other details are given, and no other reference is made
 to this block in any STEP document.  The COMPLEX block
 is only illustrated here (it is commented out of the
 schema).

 COMPLEX;
 b0+b1;
 b1+c0;
 d0+d2;
 d0+d3;
 d0+d2+d3;
 d1+d2;
 d1+d3;
 d1+d2+d3;
 b1+d0+d2;
 b1+d0+d3;
 b1+d0+d2+d3;
 b1+d1+d2;
 b1+d1+d3;
```

Figure 54 - EXPRESS schema t11 (continues...)

```
 b1+d1+d2+d3;
 END_COMPLEX;
 *)

END_SCHEMA;
```

Figure 54 - EXPRESS schema t11 (continued)

```
/* t11.idl
*/

interface  A0  {
};

interface  B0  : A0 {
};

interface  B1  : A0 {
};

interface  C0  : B0 {
};

interface  D0  : C0 {
};

interface  D1  : C0 {
};

interface  D2  : C0 {
};

interface  D3  : C0 {
};

/* complex entity data types */

interface B0B1  : B0, B1 {
};

interface B1C0  : B1, C0 {
};

interface D0D2  : D0, D2 {
};

interface D0D3  : D0, D3 {
};
```

Figure 55 - Translation of schema t11 into IDL (continues...)

```
interface D0D2D3  : D0, D2, D3 {
};

interface D1D2  : D1, D2 {
};

interface D1D3  : D1, D3 {
};

interface D1D2D3  : D1, D2, D3 {
};

interface B1D0D2  : B1, D0, D2 {
};

interface B1D0D3  : B1, D0, D3 {
};

interface B1D0D2D3  : B1, D0, D2, D3 {
};

interface B1D1D2  : B1, D1, D2 {
};

interface B1D1D3  : B1, D1, D3 {
};

interface B1D1D2D3  : B1, D1, D2, D3 {
};
```

Figure 55 - Translation of schema t11 into IDL (continued)

All interfaces which represent complex entity data types (those after the comment `/* complex entity data types */` in figure 55) were included manually in t11.idl according to the SDAI specification (ISO 10303-22 1995) because their generation was not implemented in *exp2idl*. In the COMPLEX block commented in figure 54, all the possible instantiations of complex entity data types are included and then translated according to the standard mapping definition.

In summary, an entity lattice in EXPRESS was translated into IDL interfaces. Each entity was mapped to an IDL interface, as well as each complex entity, i.e., each combination of two or more subtypes allowed by the super/subtype constraints. For instance, `B0B1` is an interface which implements a complex entity data type which can be described as “an `A0` which is a `B0` and a `B1` at the same time.”

Losses observed in case t11

All constructs in the IDL schema in figure 55 correspond to the ones in the EXPRESS schema in figure 54. However, the IDL translation does not cover the full detail of the EXPRESS schema.

The ABSTRACT status of supertypes is lost. Since abstract supertypes are not intended to be directly instantiated, the mapping of abstract supertypes into IDL interfaces may lead to the implementation of objects that should be instantiated only in conjunction with its subtypes. Therefore, the EXPRESS and IDL schemata are *incompatible* concerning inheritance. The number of complex entity data types in IDL derived from the EXPRESS entity lattice present the additional problem of combinatorial explosion. This, however, concerns performance, not semantic loss.

4.1.12 Another translation of complex entity data types

The purpose of this translation case is also to investigate the semantic loss regarding complex entity data types, using a much smaller and less complex schema than t11, where there is an implicit ANDOR constraint in the root entity. Figure 56 presents the EXPRESS schema t12, followed by the translation into IDL, in figure 57.

Comments regarding translation case t12

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t12.idl with the interfaces in different files, referenced (`included`) in t12.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 57. The syntax of the resulting IDL source was successfully verified using Orbix.

The interface `E1E2` represents a complex entity data type. It was included manually in t12.idl according to the SDAI specification (ISO 10303-22 1995) because its generation was not implemented in *exp2idl*. The COMPLEX block commented in figure 56 contains the complex entity that was translated into interface `E1E2` according to the standard mapping definition.

The entity lattice in t12.exp was translated into IDL interfaces. Each entity was mapped to an IDL interface, as well as one complex entity.

Losses observed in case t12

The ABSTRACT character of a supertype entity is lost when translated into an IDL interface. Similarly as in case t11, all interfaces in the IDL schema in figure 57 correspond to entities in the EXPRESS schema in figure 56, but the IDL translation does not cover the full detail of the EXPRESS schema.

```
(* t12.exp

     Purpose of the schema: to represent a complex entity data
     type.

          (ABS) e0
      (implicit ANDOR)
           /     \
          /       \
        e1         e2
*)

SCHEMA t12;

 ENTITY e0 ABSTRACT SUPERTYPE;
 END_ENTITY;

 ENTITY e1 SUBTYPE OF (e0);
 END_ENTITY;

 ENTITY e2 SUBTYPE OF (e0);
 END_ENTITY;

 (*
 COMPLEX;
 e1+e2;
 END_COMPLEX;
 *)

END_SCHEMA;
```

Figure 56 - EXPRESS schema t12

```
/* t12.idl
*/

interface  E0  {
};

interface  E1  : E0 {
};

interface  E2  : E0 {
};

interface  E1E2  : E1, E2 {
};
```

Figure 57 - Translation of schema t12 into IDL

The ABSTRACT status of supertypes is lost. Therefore, the mapping of abstract supertypes into IDL interfaces may lead to the implementation of objects that should be instantiated only in conjunction with its subtypes. The EXPRESS and IDL schemata are *incompatible* concerning inheritance. In the standards-based sharing, however, no loss is expected since both ends of the transmission use the same STEP schema, and the instantiation of abstract supertypes are not allowed when data is checked against the sender or the receiver STEP schema.

4.1.13 Translation of unbounded SET data types

The purpose of this translation case is to investigate the semantic loss regarding unbounded SET data type. Figure 58 presents the EXPRESS schema t13, followed by the translation into IDL, in figure 59.

Comments regarding translation case t13

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t13.idl with the interfaces in different files, referenced (`included`) in t13.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 59. The (standard) syntax of the resulting IDL source was not accepted by Orbix. The code was adapted, as shown if figure 59, so the IDL code could be successfully verified.

Losses observed in case t13

The EXPRESS `SET of e0` in figure 58 is mapped into the IDL construct `E0__set__list`, as shown in figure 59. This translation allows for a one-to-one EXPRESS-IDL mapping, i.e., it covers the full detail of the EXPRESS construct (unbounded) SET, and an IDL interface which contains the construct `set__list` have no other possible interpretation, making EXPRESS and IDL *equivalent* with respect to this construct. No loss or ambiguity is associated to this EXPRESS-IDL translation. Nonetheless, in the experiment with a particular IDL compiler (IONA's Orbix), the syntax produced according to Part 26 was not accepted, as pointed out in figure 59. Apparently, this version of Orbix was compliant with another syntax of IDL.

4.1.14 Another translation of REAL and STRING data types

The purpose of this translation case is to investigate the semantic loss regarding aspects of simple data types (REAL and STRING) that were not depicted in suites t01 to t04. Figure 60 presents the EXPRESS schema t14, followed by the translation into IDL, in figure 61.

```
(* t13.exp

     Purpose of this schema: to represent the unbounded
     aggregate SET.
*)

SCHEMA t13;

 ENTITY e0;
     e0attr : e1;
 END_ENTITY;

 ENTITY e1;
 INVERSE
     e1attr : SET OF e0
          FOR e0attr;
 END_ENTITY;

END_SCHEMA;
```

Figure 58 - EXPRESS schema t13

```
/* t13.idl
*/

interface E0;
typedef sequence< E0 > E0__set;

interface  E1;
interface  E0  {
     attribute E1 e0attr;
};

interface  E1  {
     readonly attribute E0__set__list e1attr;

/*   The above translation, according to Part 26, 5.2.6 ENTITY
     data type, was not accepted by the Orbix idl compiler.
     Errors annotated:
     14:(semantic): Identifier `E0__set__list' not found
     14:(semantic): Name does not denote a type
     Translation accepted by the compiler:

     readonly attribute E0__set e1attr;
*/
};
```

Figure 59 - Translation of schema t13 into IDL

```
(* t14.exp

     Purpose of the schema: to represent aspects of simple
     data types (REAL and STRING) that were not depicted in
     translation cases t01 to t04.
*)

SCHEMA t14;

 ENTITY e0;
  e0attr : REAL(8);   -- 8 is the precision-specification
                      -- (significant digits)
 END_ENTITY;

 ENTITY e1;
  e1attr0 : STRING (10);
  e1attr1 : STRING (10) FIXED;
 END_ENTITY;

END_SCHEMA;
```

Figure 60 - EXPRESS schema t14

```
/* t14.idl

typedef string<10> string_10;

  interface  E0 {
     attribute double e0attr;
  };

  interface  E1 {
     attribute string<10> e1attr0;
     attribute string<10> e1attr1;

};
```

Figure 61 - Translation of schema t14 into IDL

Comments regarding translation case t14

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t14.idl with the interfaces in different files, referenced (`included`) in t14.idl. These interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 61. The syntax of the resulting IDL source was successfully verified using Orbix.

Losses observed in case t14

As in case t01, the EXPRESS `REAL` in figure 60 is mapped into the IDL construct `double`, loosing reference to its precision, as shown in figure 61. The `STRING` attributes are translated into `string`, keeping information about their limited size of 10. This translation allows for a translation of EXPRESS constructs `REAL` and limited `STRING`, however the size of the `REAL` attribute is lost in IDL, as well as the `FIXED` directive of attribute `e1attr1`. In this case, EXPRESS and IDL are *incompatible* with respect to the size and fixed character of `REAL` and `STRING`. In the standards-based sharing, however, no loss is expected since both ends of the transmission use the same STEP schema, and whatever is missing in the IDL interface is filled in when data is checked against the receiver STEP schema.

4.1.15 Another translation of aggregate data types

The purpose of this translation case is to investigate the semantic loss regarding aspects of aggregates (ARRAY, LIST, BAG) that were not present in STEP schemata. Figures 62 and 63 present the EXPRESS schema t15 and its IDL translation.

Comments regarding translation case t15

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t15.idl with the interfaces in different files, referenced (`included`) in t15.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 63. The syntax of the resulting IDL source was successfully verified using Orbix.

Losses observed in case t15

The EXPRESS constructs `ARRAY`, `LIST`, and `BAG` are mapped into IDL similarly as in previous translation cases t07 and t13. The IDL mapping allows for the representation of these constructs; however, the size and indexes of the aggregates are not represented, as shown in figure 63. Moreover, the `OPTIONAL` (an ad hoc solution for this problem is reported in case t01) and `UNIQUE` directives are ignored in the IDL mapping. This makes the EXPRESS and IDL codes *incompatible*, although the impact on semantic loss should be minimal, since generic (i.e., without indexes or limits) aggregates still can be used to convey the correspondent limited and indexed aggregates. Especially in the standards-based sharing, no loss is expected since both ends of the transmission use the same STEP schema, and whatever is missing in the IDL interface is filled in when data is checked against the receiver STEP schema.

4.1.16 Translation of entity attributes

The purpose of this translation case is to investigate the semantic loss regarding aspects of entity attributes (only INVERSE and OPTIONAL were depicted in previous cases). Figure 64 presents the EXPRESS schema t16, followed by the translation into IDL, in figure 65.

```
(* t15.exp
     Purpose of the schema: to represent aspects of aggregates
     (ARRAY, LIST, BAG) that were not present in SOLIS, and
     therefore absent in previous test suites.  SET is already
     covered in suites t07 and t13.  *)

SCHEMA t15;

 ENTITY e1;
  e1attr : ARRAY[?:?] OF OPTIONAL UNIQUE SET [1:10] OF INTEGER;
 END_ENTITY;

 ENTITY e2;
   e2attr : LIST[0:?] OF UNIQUE e1;
 END_ENTITY;

 ENTITY e3;
  e3attr : BAG[1:?] OF INTEGER;
 END_ENTITY;

END_SCHEMA;
```

Figure 62 - EXPRESS schema t15

```
/* t15.idl
*/

typedef sequence< sequence < long > >
     long__bounded_set__bounded_array;
interface E1;
typedef sequence< E1 > E1__bounded_list;
typedef sequence< long > long__bounded_bag;

interface  E1  {
     attribute long__bounded_set__bounded_array e1attr;
};

interface  E2  {
     attribute E1__bounded_list e2attr;
};

interface  E3  {
     attribute long__bounded_bag e3attr;
};
```

Figure 63 - Translation of schema t15 into IDL

```
(* t16.exp

     Purpose of the schema: to represent aspects of entity
     attributes (only INVERSE and OPTIONAL were depicted in
     other suites).
*)

SCHEMA t16;


  TYPE positive = INTEGER;
  WHERE
    notnegative : SELF > 0;
  END_TYPE;


  ENTITY e0;
    e0attr : OPTIONAL INTEGER;
  END_ENTITY;


  ENTITY e1;
    e1_height : REAL;
  DERIVE
    e1_ideal_weight : REAL := ( e1_height - 1.0 ) * 100.0;
  END_ENTITY;


  ENTITY e2door;
    e2_handle : e3knob;
  END_ENTITY;

  (* The following declaration means that knob only exists
     if they are used in the role of handle in one
     instance of a door *)
  ENTITY e3knob;
  INVERSE
    e3_opens : e2door FOR e2_handle;
  END_ENTITY;


  ENTITY e4;
    e4code : INTEGER;
    e4name: STRING;
  UNIQUE
    ur1: e4code, e4name;
  END_ENTITY;


  ENTITY e5;
    attr : REAL;
  END_ENTITY;
```

Figure 64 - EXPRESS schema t16 (continues...)

```
  ENTITY e6;
    attr : BINARY;
  END_ENTITY;
  ENTITY e7 SUBTYPE OF (e5,e6);
  WHERE
    attr_pos : SELF\e5.attr > 0.0 ;
    (* attr as declared in e5, not e6 *)
  END_ENTITY;


  ENTITY e8;
    e8attr0 : NUMBER;
    e8attr1 : OPTIONAL REAL;
  END_ENTITY;
  ENTITY e9 SUBTYPE OF (e8);
    SELF\e8.e8attr0 : INTEGER;
  END_ENTITY;
  ENTITY e10 SUBTYPE OF (e8);
    SELF\e8.e8attr1 : REAL;
  END_ENTITY;
  ENTITY e11 SUBTYPE OF (e8);
    e11attr : NUMBER;
  DERIVE
    SELF\e8.e8attr0 : REAL := 1 / e11attr;
  END_ENTITY;


  ENTITY e12;
    e12attr : INTEGER;
  DERIVE
    e12plus : INTEGER := e12attr + 1;
  END_ENTITY;
  ENTITY e13 SUBTYPE OF (e12);
  WHERE
    e13big : e12plus >= 18;
  END_ENTITY;


END_SCHEMA;
```

Figure 64 - EXPRESS schema t16 (continued)

Comments regarding translation case t16

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t16.idl with the interfaces in different files, referenced (`included`) in t16.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 65. The syntax of the resulting IDL source was successfully verified using Orbix.

```
/* t16.idl
*/

typedef    long  positive;

interface  e0  {
     attribute long e0attr;
};
interface  e1  {
     attribute double e1_height;
};

interface  e3knob;
interface  e2door  {
     attribute e3knob e2_handle;
};

interface  e3knob  {
     readonly attribute e2door__list e3_opens;

};

interface  e4  {
     attribute long e4code;
     attribute string e4name;
};

interface  e5  {
     attribute double attr;
};

typedef sequence<octet> Binary;
interface  e6  {
     attribute Binary attr;
};

interface  e7  : e5,  e6 {
};

interface  e8  {
     enum Number_discriminant { Number_discriminant__long,
     Number_discriminant__double };

     union Number switch ( Number_discriminant ) {
     case Number_discriminant__long : long c1;
     case Number_discriminant__double : double c2;
     };
     attribute Number e8attr0;
     attribute double e8attr1;
};
```

Figure 65 - Translation of schema t16 into IDL (continues...)

```
interface  e9  : e8 {
};

interface  e10  : e8 {
};

interface  e11  : e8 {
     attribute Number e11attr;
};

interface  e12  {
     attribute long e12attr;
};

interface  e13  : e12 {
};
```

Figure 65 - Translation of schema t16 into IDL (continued)

Losses observed in case t16

The EXPRESS directives `SELF`, `OPTIONAL`, `WHERE`, `DERIVE`, and `UNIQUE` in figure 64 are lost in the IDL translation. There is ambiguity in the inheritance of attribute `attr` from `e5` and `e6`, in `e7`, that is not resolved. EXPRESS and IDL are severely *incompatible* regarding these constructs. While the constructs in case t16 are not part of any STEP IR or AP, this mismatch is not a menace to the success of the STEP-CORBA coupling at the moment. The receiver STEP schema can fill in missing information about the data types, except in the case of `OPTIONAL`, but an ad hoc solution is possible, as reported in case t01. Should the constructs in t16 be included in STEP APs and accessed without checking for equal sender and receiver STEP schemata, then semantic loss becomes a serious problem.

4.1.17 Translation of entities with multiple inheritance

The purpose of this translation case is to investigate the semantic loss regarding multiple inheritance within an EXPRESS entity lattice. Figure 66 presents the EXPRESS schema t17, followed by the translation into IDL, in figure 67.

Comments regarding translation case t17

The IDL translation was produced using the *exp2idl* translator. This automatic translation produced the file t17.idl with the interfaces in different files, referenced (`included`) in t17.idl. The interfaces were then manually made explicit within the main file for simplicity of presentation, as shown in figure 67. The syntax of the resulting IDL source was successfully verified using Orbix.

```
(* t17.exp

     Purpose of the schema: to represent multiple inheritance.
*)

SCHEMA t17;

 ENTITY e0
   SUPERTYPE OF (e2);
 END_ENTITY;

 ENTITY e1
   SUPERTYPE OF (e2);
 END_ENTITY;

ENTITY e2
   SUBTYPE OF (e0,e1);
 END_ENTITY;

END_SCHEMA;
```

Figure 66 - EXPRESS schema t17

```
/* t17.idl
*/

interface  E0  {
};

interface  E1  {
};

interface  E2  : E0, E1 {
};
```

Figure 67 - Translation of schema t17 into IDL

Losses observed in case t17

The multiple inheritance of subtype `e2` and supertypes `e0` and `e1` in figure 66 is mapped into *equivalent* IDL code, where the interface `E2` inherits from `E0` and `E1`, in figure 67. This translation allows for a one-to-one EXPRESS-IDL mapping, i.e., it covers the full detail of multiple inheritance in EXPRESS.

4.2 Concluding remarks on the analysis of the semantic loss

An analogy can be drawn from linguistics to illustrate the problem of semantic loss in translation: the Sapir-Whorf hypothesis proposes that the *structure* of anyone's native language strongly influences or fully determines the world view he will acquire as he learns the language (Kay and Kempton 1994). Whorf found that native American language Hopi contains "no words, grammatical forms, constructions or expressions that refer directly to what we call 'time', or to past, present, or future, or to enduring or lasting, or to motion as kinematic rather than dynamic" (Whorf 1956). An expression in an European language containing a collective of time periods (e.g., "ten days") have no equivalent in Hopi. This structural or syntactic mismatch is not resolved by any other construct with a similar meaning in Hopi, therefore causing semantic loss. Although computer-processable languages are quite less complex than natural languages, with grammars belonging typically to the two simpler classes (*regular* and *context-free*) of Chomsky's classification as described by Tremblay and Sorenson (1985), this example reflects the same situation that occurs when one tries to translate code from one computer language into another.

Losses in the network sharing of product data in virtual enterprises

There are three types of data translation in the standards-based integration of product data models aimed at network sharing:

1. Translation from a specific CAx data model into STEP, and vice-versa;
2. Translation from the STEP models written in EXPRESS into IDL for the building of interfaces for network access; and
3. Translation of IDL interfaces into an implementation language, since IDL is not meant for direct implementation.

These translations are potential generators of information loss, as described in section 3.2.3. Losses related to the translation of STEP data to and from a specific CAx data model (number 1 above) depend on the quality of the translators (pre- and postprocessor) between an STEP AP and the CAx's specific data model. The implementation of the IDL interfaces in an implementation language may result in additional loss, since the IDL constructs have to be mapped into the implementation language constructs (number 3 above). The analysis of the semantic loss presented in

this thesis, however, focuses on the losses intrinsic to the EXPRESS-IDL translation (number 2 above).

Next, a summary of the 17 cases of analysis of the semantic loss in the translation of STEP product data models into CORBA's Interface Definition Language is presented.

Table 11 - Semantic loss in the translation from EXPRESS into IDL

Translation case	EXPRESS constructs depicted	Losses observed in the translation according to the IDL binding to the SDAI (ISO 10303-26)
t01	REAL, INTEGER, STRING	Attribute directive `OPTIONAL` is ignored in the IDL mapping.
t02	NUMBER	None.
t03	BOOLEAN, LOGIC	`Logical` and `Boolean` allow an `unset` value in IDL.
t04	BINARY	Information about size of `BINARY` attributes, when specified, is lost. The `FIXED` directive is ignored as well.
t05	ARRAY, LIST, BAG	Aggregate size and indexes are lost (Part 26 does not specify a way to carry this information).
t06	unbounded ARRAY, BAG, and LIST	None.
t07	SET	Aggregate size and indexes are lost.
t08	SELECT	None.
t09	nested SELECTs	None
t10	ENUMERATION	In IDL, an `unset` value is added to the range of values.
t11	complex entity data types	`ABSTRACT` status of supertype entities is lost.
t12	complex entity data types	`ABSTRACT` status of supertype entities is lost.
t13	unbounded SET	None.
t14	REAL, STRING (aspects not depicted in suite t01)	Precision of `REAL` and `FIXED` status of `STRING` are lost.
t15	aggregates (aspects not present in STEP APs).	`OPTIONAL` status and `UNIQUE` constraint are ignored in the translation. Size and indexes of aggregates are lost.
t16	entity attributes	Generalized loss of attribute constraints, attribute redeclaration, constraining of inherited attributes.
t17	multiple inheritance	None.

Summary of the semantic loss in the EXPRESS-IDL translation

Table 11 presents a summary of the losses detected in the 17 cases of translation of STEP data models from EXPRESS into IDL. The mismatch between EXPRESS and IDL identified here implies that additional work will have to be performed in order to adjust a product data model shared across a virtual enterprise's network to be used by a specific CAx system.

The procedures proposed in each translation case to recover from or alleviate the impact of losses take into account only a comparison between the modeling power of EXPRESS and IDL. It is possible that knowledge about the nature of engineering applications which share data in an interoperable network may help develop other strategies to recover from the semantic loss. For instance, suppose that a specific application, related to a specific AP, features arrays. In IDL, the indexes and size of arrays are lost, but the application may present a pattern for arrays (e.g., always the same size and base index), in which case a solution could be designed to reflect this pattern.

CHAPTER 5

CONCLUSION

This chapter presents a summary of the thesis, results and contributions, and recommendations for future work.

5.1 Thesis summary

This thesis has presented an overview of engineering information management with specific focus on product data models shared across distributed networks. Database management techniques for business and engineering objects were discussed. The problem of product data exchange (PDE) and sharing was introduced, along with a discussion on the evolution of PDE specifications and standards. The architecture, information modeling, and implementation of the STandard for the Exchange of Product model data (STEP), ISO 10303, were explained, together with comments on the possible impact of ontology engineering on STEP's information modeling development (as described in section 2.2.4.4), and considerations about the future of the standard. The problem of interoperability was presented, with an account of specifications for application interoperability, especially the Common Object Request Broker Architecture (CORBA).

The main objective of the thesis was to develop knowledge for the realization of network integrated STEP databases, which is vital for the integration of industrial enterprises in the vertical and horizontal aspects, namely concurrent engineering and virtual enterprises. For this purpose, a standards-based approach to the network interoperability of product data models was introduced. STEP was adopted as the standard for product data representation, archiving, exchange, and sharing. CORBA was chosen as the standard architecture for the realization of the network interoperable access to product data models. This choice of standards is supported by the ISO, which produced STEP, and supported the adoption of CORBA for STEP interoperability through the IDL binding to the SDAI. The coupling of STEP and CORBA is also the choice for product data model interoperability in the scope of the National Industrial Information Infrastructure Protocols (NIIIP) project.

Challenges to the realization of product data model interoperability using STEP and CORBA were discussed, and an analysis of the semantic loss in the translation of

network interoperable product data models was conducted. An EXPRESS-IDL translation suite was designed, comprised of 17 translation cases, each one focusing on one aspect of the EXPRESS data type. Losses due to the mismatch between EXPRESS and IDL data types were identified, the possible impact of these losses on the sharing of product data was discussed, and actions to alleviate the specific losses were suggested.

5.2 Results and contributions

From the analysis of the semantic loss, one can conclude that product data models retrieved from a shared network using STEP and CORBA are subject to deficiencies caused by the mismatch between EXPRESS and IDL data types. Actions to alleviate these losses, based on the syntactic mismatch between EXPRESS and IDL, were proposed in chapter 4. In the recommendations, in section 5.3, other strategies to recover from semantic loss are suggested, taking into account not only the syntactic mismatch, but also knowledge about the use of data in a specific engineering application.

Figure 68 illustrates the network interoperable access to product data using STEP and CORBA, and the data translations associated. First, data from a CAx system or a database, stored according to a particular data model, has to be translated into a STEP data model, which is one of the APs. These STEP data can be accessed through SDAI operations. To provide for the fine-grain network access to STEP objects, IDL interfaces have to be published and made available through ORBs. This requires a data structure translation from the entities written in EXPRESS into IDL. Finally, for the implementation, IDL interfaces have to be mapped into an implementation language.

Any of these three translations may impose semantic loss on the data being shared. This thesis addressed the semantic loss in the specific translation from EXPRESS into IDL, which is a subset of the general problem of semantic loss in network interoperable databases.

Eastman (1994) considers that the need for data type conversion is bound to prevent the successful neutral file exchange. Sanderson (1994) maintains that the problem of information loss in data translation will never be fully solved, but it is possible to address subclasses of the general problem. Indeed, from all the previous experiences with neutral format interchange, it was expected that STEP usage could yield data interchange with some level of semantic loss. The additional work imposed by the *semantic gap* between the languages and data models involved in the

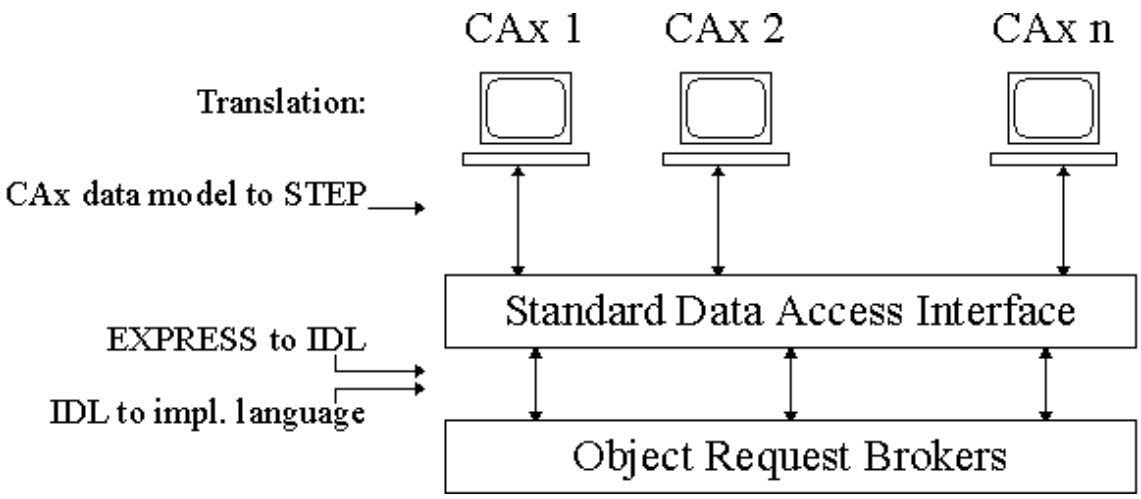


Figure 68 - Data translation in the network sharing of product model data

interchange should be compared to the current effort to rebuild, or directly translate a product data model from one CAx system to another. Even facing semantic loss, the network integration of product data models is a more feasible and economic alternative.

STEP is being supported by a great number of companies, both users and vendors of CAx systems, not only because of the economic aspect of data translation in relation to the traditional approaches of re-entering data or standardizing in a single CAx system, but also because its status of international standard is expected to set the tendency that adherence to STEP become an exigence to participate in business when product data interchange is involved.

Contributions

This thesis offers the following contributions, in accordance with the objective of developing knowledge and technology for the realization of network integrated STEP databases:

- The building of a realistic EXPRESS schema suite for analysis of semantic loss in translation, comprised of 17 schemata where each schema focuses on one aspect of the EXPRESS data type. The schemata are derived from the IRs, APs, and other STEP documents, and can be used to perform the analysis of data type mismatch between EXPRESS and other languages;
- The identification of semantic loss in the translation of product data models based on the data type mismatch between EXPRESS and IDL; and

- The proposal of ad hoc procedures to alleviate or eliminate the occurrence of each of the losses identified.

The contributions described above refer to generic STEP schemata. The next section gives recommendations for future work addressing the semantic loss in product data sharing according to a specific AP, and other open research issues.

5.3 Recommendations

The analysis of the semantic loss presented in this thesis reveals which data represented in EXPRESS may be lost when accessed through an IDL interface. Product data exchange and sharing using STEP, however, is always performed using a specific AP. To date, only APs 201 (ISO 10303-201 1994) and 203 (ISO 10303-203 1994) have been published as international standards. The extent in which the losses identified in this thesis affect the exchange and sharing according to one of these APs (or others, as they are being published as international standards) is a subject to be investigated. Such an investigation demand a thorough understanding about the product data modeled in the AP schema and the applications that may use it for data sharing, and should address the following questions:

- Which of the EXPRESS constructs susceptible to losses occur in the specific AP?;
- How frequently are these constructs present in typical data sets to be shared according to the specific AP?;
- What is the impact or implication of each kind of loss in the applications that share data through the specific AP?; and
- Are there, and what are the procedures to be taken to recover from each kind of loss in the sharing through the specific AP? Is there any pattern for this recovery? Is it automatable?.

Still, other losses are possible in addition to those identified in the EXPRESS-IDL translation. Depending on the language chosen for the implementation of IDL interfaces (e.g., C or C++), more losses may be imposed on the STEP data accessed. The identification and proposal of solutions to these losses is a subject which merit further investigation.

Out of the scope of virtual enterprise technology, apart from the network interoperable product data sharing using the IDL binding to the SDAI, ISO has published other draft standards defining bindings from programming languages such as

C and C++ to the SDAI, allowing the access from multiple applications to STEP data in a single database, as described in 2.2.4.3. There are semantic losses inherent to the language chosen for implementation, for which an investigation like the analysis in this thesis can also be conducted. In the early phase of this research, it was proposed that the translation suite included also an EXPRESS-C++ translation of each case, with the losses identified and compared to those in the EXPRESS-IDL translation. This, however, had to be dropped since the standard EXPRESS-C++ mapping was not complete enough to allow for the building of the translations. As the standardization process progresses and the C++ binding to the SDAI gets completed, this intent can be realized. In addition, a Java binding to the SDAI shoud be available in the future, thus offering another case for the exploration of semantic loss.

While the several STEP documents undergo the standardization process, there is intense debate regarding the quality of the standard. One area of the debate concerns the building of the APs, considered the core of STEP. Two recent propositions address the building of *independent* APs, and the use of common ontologies as a firmer ground upon which to build the IRs, leading to better APs.

Metzger (1996) proposes to “give up the idea of a complete and unified data model,” a radical change in relation to the current process of STEP data model development, which builds APs upon the interpretation of IRs.

The use of common ontologies in the building of STEP information models is being proposed to alleviate the problem of redundancy generated by the bottom-up approach, leading to difficulty in coupling multiple APs. Common ontologies were used originally for knowledge sharing, and there is research addressing the applicability of an ontology engineering approach to STEP:

> “Ontologies ... could be used to effect unification in the translation of EXPRESS models into other languages, or an interoperability of the EXPRESS models with models represented in other languages.” (Meis and Ostermayer 1996)

Unification was defined in the scope of STEP as the "process whereby two statements in logic are recognized to be equivalent," according to Eastman (1994). However, the development of the Semantic Unification Meta-Model (SUMM), an early “mathematically rigorous approach to the unification and integration of models independently of the languages in which those models were formulated” (Eastman 1994), was discontinued.

Table 12 - Parallel between the use of STEP for product data sharing, and common ontologies for knowledge sharing

STEP for PDE and sharing	Common ontologies for knowledge sharing
EXPRESS is STEP's data description language, used to build all of STEP's information models.	KIF (Knowledge Interchange Format) is a canonical form, a formal language for the interchange of knowledge.
EXPRESS is human-readable - even if it is not intended primarily for this purpose.	KIF is human-readable - even if it is not intended primarily for this purpose.
Integrated Resources are general conceptual models for the reuse of product data.	Common ontologies are vocabularies of representational terms for the reuse of knowledge.
There is semantic loss in the translation of STEP data models.	"Not all KIF forms can be translated further into all target languages" (Meis and Ostermayer 1996)

The use of common ontologies may bring the benefits of unification back into the STEP development process. Table 12 presents a parallel between the use of STEP for product data sharing, and the use of common ontologies for knowledge reuse and sharing.

Gruber (1992) proposes restrictions on KIF for portable ontologies. A promising research topic related to this proposition is the investigation of a possible set of limitations to be imposed on EXPRESS data models in order to make them portable (without semantic loss) across a determinate representation model, which could be, for instance: (1) the one of ACIS, so every CAx system which uses the ACIS kernel would be able to interpret correctly the STEP data model; or (2) the IDL data type, so product data models shared across a network would not suffer from the data type mismatch between EXPRESS and IDL.

REFERENCES

M. Atkinson, D. DeWitt, D. Maier, F. Bancilhon, K. Dittrich, and S. ZDONIK. "The object-oriented database system manifesto". In: *Deductive and object-oriented databases* (eds.: W. Kim, M. Nicolas, and S. NISHIO). Elsevier, pp. 223-240, 1990.

R. Barra. "Projeto B-STEP: Uma Iniciativa Brasileira em STEP". In: *II Seminário Internacional Aplicações de STEP para a Integração de Sistemas*. Instituto de Pesquisas Tecnológicas, São Paulo, 27 de Novembro de 1996.

C. Batini, M. Lenzerini, and S. Navathe. "A Comparative Analysis of Methodologies for Database Schema Migration". *ACM Computing Surveys* 18 (December 1986), pp. 323-364, 1986.

M. Blaha, W. Premerlani, and J. Rumbaugh. "Relational Database Design Using an Object-Oriented Methodology". *Communications of the ACM* 31(4), pp. 414-427, 1988.

T. Brando. "Comparing CORBA and DCE". *Object Magazine* 6 (1), pp. 52-57, March 1996.

W. Burkett. "The Implementation of STEP Schemas". In: K. Law (ed.): *Engineering Data Management: Key to Success in a Global Market*. Proceedings of the 1993 ASME International Computers in Engineering Conference and Exposition, San Diego-CA, pp. 25-38, 1993.

R. Catell (editor). "Introduction (What are Next-Generation Database Systems?)". *Communications of the ACM* 34 (10), pp. 31-33, 1991.

M. Cecchini. "Experiência da Embraer em Troca de Dados de Produtos". In: *II Seminário Internacional Aplicações de STEP para a Integração de Sistemas*. Instituto de Pesquisas Tecnológicas, São Paulo, 27 de Novembro de 1996.

P. Chen. "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Transactions on Database Systems* 1 (1), pp. 9-36, 1976.

J. Cheng and A. Hurson. "Effective Clustering of Complex Objects in Object-Oriented Databases". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, New York, pp. 22-31, 1991.

S. Clark. *The NIST working form for STEP*. National Institute of Standards and Technology, Report NISTIR 4351, 1990.

E. F. Codd. "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM* 13 (6), pp. 377-387, 1970.

C. Date. *An Introduction to Database Systems*. Fourth edition, Vol. I. Addison-Wesley, 639 pp., 1986.

Digital Equipment Corp. "Product Data Sharing using STEP Technologies". Version 0.2 (by R. Doty), part of: *Open Data: The Next Generation in Open Systems*, Digital STEP Technologies, 1992.

C. Eastman. "Out of STEP?" (Comment). *Computer Aided Design* 26 (5), pp. 338-340, 1994.

C. Eastman and N. Fereshetian. "Information Models for Use in Product Design: a Comparison". *Computer Aided Design* 26 (5), 1994.

C. Eastman. "Database Facilities for Engineering Databases. *Proceedings of the IEEE* 69 (10), pp. 1249-1263, 1981.

E. Edwards-Iwe. "A Client/Server Implementation of the Design Process using PDES/STEP 'Level 3' Data Sharing Architecture". In: K. Law (ed.): *Engineering Data Management: Key to Success in a Global Market*. Proceedings of the 1993 ASME International Computers in Engineering Conference and Exposition, San Diego-CA, pp. 15-23, 1993.

J. Eggers. *Implementing EXPRESS in SQL*. Technical Report ISO TC184/SC4/WG1 Doc. 292, ISO, October 1988.

R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. 2nd ed. Addison-Wesley, 873 pp., 1994.

J. Encarnação, R. Lindner, and E. Schlechtendahl. *Computer Aided Design*. Second edition. Springer-Verlag, 432 pp., 1990.

J. Encarnação and P. Lockemann (eds.). *Engineering Databases - Connecting Islands of Automation through Databases*. Springer-Verlag, 229 pp., 1990.

J. Encarnação, R. Schuster, and E. Voge (eds.). *Product Data Interfaces in CAD/CAM Applications - Design, Implementations and Experiences*. Springer-Verlag, 251 pp., 1986.

F. Evans. "Why Are We So Out of STEP?" *Computing & Control Engineering Journal* 5 (3), pp. 155-158, 1994.

FIPS Publication 184. *Federal Information Processing Standards. Integration Definition for Information Modeling (IDEF1X)*. National Institute of Standards and Technology (NIST), Computer Systems Laboratory, Gaithersburg, MD, USA. December 1993.

J. Fowler. *STEP for Data Management, Exchange and Sharing*. Technology Appraisals, UK. 214 pp., 1995.

R. Ganesan and R. Sandhu. "Securing Cyberspace". *Communications of the ACM* 37 (11), pp. 29-31, 1994.

M. Genesereth and R. Fikes. *Knowledge Representation Format, Version 3.0 Reference Manual*. Logic Group Report Logic-92-1. Computer Science Department, Stanford University, 1992.

A. Goh, S. Hui, B. Song, and F. Wang. "A Study of SDAI Implementation on Object-Oriented Databases". *Computer Standards & Interfaces* 16(1), pp. 33-43, 1994.

T. Gruber. "Towards Principles for the Design of Ontologies Used for Knowledge Sharing". *International Journal of Human and Computer Studies* 43 (5/6), pp. 907-928, 1994.

T. Gruber. *A Translation Approach to Portable Ontology Specifications*. Technical Report KSL 92-71. Knowledge Systems Laboratory, Stanford University, Revised April 1993.

T. Gruber. *Ontolingua: A Mechanism to Support Portable Ontologies*. Knowledge Systems Laboratory, Stanford University, 36 pp., 1992. Available at: http://www-ksl.stanford.edu/knowledge-sharing/papers/README.html.

T. Gruber. *The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases*. Knowledge Systems Laboratory, Stanford University, 1991.

N. Guarino. *Understanding, Building, and Using Ontologies - A Commentary to "Using Explicit Ontologies in KBS Development", by Heijst, Schreiber, and Wielinga*. LADSEB-CNR, National Research Council, Padova, Italy, 1996.

M. Hardwick. *Data Protocols for the Industrial Virtual Enterprise*. Source unknown (possibly EUG'96 - EXPRESS Users Group Conference), 1996.

M. Hardwick, D. Spooner, T. Rando, and K.C. Morris. "Sharing Manufacturing Information in Virtual Enterprises". *Communications of the ACM* 39 (2), pp. 46-54, 1996.

M. Hardwick, B. Downie, M. Kutcher, and D. Spooner. "Concurrent Engineering with Delta Files". *IEEE Computer Graphics &* Applications 15 (1), pp. 62-68, 1995.

M. Hardwick and D. Loffredo. "Using EXPRESS to Implement Concurrent Engineering Databases". In: *Proceedings of the Computers in Engineering Conference and the Engineering Database Symposium*. (eds.: A. Busnaina and R. Rangan). ASME, Boston, MA, pp. 1069-1083, September 17-20, 1995.

M. Hardwick and D. Spooner. "Comparison of Some Data Models for Engineering Objects". *IEEE Computer Graphics & Applications*, pp. 56-66, 1987.

S. Heiler, U. Dayal, J. Orenstein, and S. Radke-Sproull. "An Object-Oriented Approach to Data Management: Why Design Databases Need It". *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pp. 335-340, 1987.

D. Heimbigner. *Why CORBA Doesn't Cut It or Experiences with Distributed Objects*. SETT Presentation, University of Colorado, Boulder, 30 June 1995.

K. Higa, M. Morrison, J. Morrison, and O. Sheng. "Object-Oriented Methodology for Knowledge Base/Database Coupling. *Communications of the ACM* 35 (6), pp. 99-113, 1992.

A.R. Hurson, Simin H. Pakzad, and Jin-Bing Cheng. "Object-Oriented Database Management Systems: Evolution and Performance Issues". *IEEE Computer* February 1993, pp. 48-60.

ISO 10303-1. *Product Data Representation and Exchange - Part 1: Overview and Fundamental Principles*. Committee Draft, September 15, 1992.

ISO 10303-11. *Product Data Representation and Exchange - Part 11: EXPRESS Language Reference Manual*. Document TC184/SC4/WG5 N65(P2). ISO International Standard, November 1, 1994.

ISO 10303-201. *Product Data Representation and Exchange - Part 201: Application Protocol: Explicit Draughting*. ISO International Standard, 1994.

ISO 10303-202. *Product Data Representation and Exchange - Part 202: Application Protocol: Associative Draughting*. ISO Draft International Standard, August 8, 1995.

ISO 10303-203. *Product Data Representation and Exchange - Part 203: Application Protocol: Configuration Controlled Design*. ISO International Standard, 1994.

ISO 10303-204. *Product Data Representation and Exchange - Part 204: Application Protocol: Mechanical Design Using Boundary Representation*. ISO Draft International Standard, as accessed at SOLIS, ftp://ftp.cme.nist.gov/pub/subject/sc4, July 17th, 1996.

ISO 10303-21. *Product Data Representation and Exchange - Part 21: Implementation Methods: Clear Text Encoding of the Exchange Structure*. Draft International Standard, May 28, 1993.

ISO 10303-214. *Product Data Representation and Exchange - Part 214: Application Protocol: Core Data for Automotive Mechanical Design Process*. Document TC184/SC4/WG3 N436. Committee Draft, August 8, 1995.

ISO 10303-22. *Product Data Representation and Exchange - Part 22: Implementation Methods: Standard Data Access Interface*. Committee Draft, May 31, 1995.

ISO 10303-23. *Product Data Representation and Exchange - Part 23: Implementation Methods: C++ Programming Language Binding to the Standard Data Access Interface*. Committee Draft, December 25, 1995.

ISO 10303-24. *Product Data Representation and Exchange - Part 24: Implementation Methods: Standard Data Access Interface - C Language Late Binding*. Committee Draft, July 28, 1995.

ISO 10303-26. *Product Data Representation and Exchange - Part 26: Implementation Methods: Interface Definition Language Binding to the Standard Data Access Interface*. Document ISO TC184/SC4/WG11 N019. Committee Draft, 17 March 1997.

ISO 10303-41. *Product Data Representation and Exchange - Part 41: Integrated Resources: Fundamentals of Product Description and Support*. ISO International Standard, 1994.

ISO 10303-42. *Product Data Representation and Exchange - Part 42: Integrated Resources: Geometric and Topological Representation*. Document TC184/SC4/WG3 N125a. ISO Committee Draft, July 31, 1992.

J. Joseph, S. Thatte, C. Thompson, and D. Wells. "Object-Oriented Databases: Design and Implementation". *Proceedings of the IEEE* 79 (1), pp. 42-64, 1991.

A. Kaplan and J. Wileden. *PolySPIN: Support for Polylingual Persistence, Interoperability and Naming in Object-Oriented Databases*. University of Massachusetts Amherst, CMPSCI Technical Report 96-4, January 1996.

P. Kay and W. Kempton. "What is the Sapir-Whorf Hypothesis?" *American Anthropologist* 86 (1), pp. 65-79, 1984.

W. Kent. *Data and Reality*. North-Holland, 211 pp., 1978.

V. Kern, R. Barra, and R. Barcia. "Implementation of Standardized Shareable Product Databases" *Proceedings of the II International Congress of Industrial Engineering* (CD-ROM), Piracicaba SP, Brazil, October 7-10, 1996.

V. Kern, J.H. Bøhn, and R. Barcia. "The Building of Information Models in STEP". *Proceedings of the II International Congress of Industrial Engineering* (CD-ROM), Piracicaba SP, Brazil, October 7-10, 1996.

V. Kern and J.H. Bøhn. "STEP Databases for Product Data Exchange". In: *Proceedings of I International Congress of Industrial Engineering*. Vol. III, pp. 1337-1341, São Carlos SP, Brazil, September 4-7, 1995.

V. Kern. "Database Systems for CAD". In: *Computer-Aided Design I - Fall 1994 Term Papers*. Mechanical Engineering Department, Virginia Polytechnic Institute & State University, Blacksburg VA, pp. 67-74, 1994.

S. Khoshafian. *Object-Oriented Databases*. John Wiley & Sons, 362 pp., 1993.

J. Kiekenbeck, A. Siegenthaler, and G. Schlageter. "EXPRESS to C++: A Mapping of the Type-System". *EXPRESS User's Group (EUG) Conference '95*, 1995.

R. Kiggans. Development and Implementation of ISO 10303 (STEP). Speech at the *II Seminário Internacional Aplicações de STEP para a Integração de Sistemas*. Instituto de Pesquisas Tecnológicas, São Paulo SP, Brazil, November 27, 1996.

W. Kim, J. Banerjee, H. Chou, and J. Garza. "Object-Oriented Database Support for CAD". *Computer Aided Design* 22 (8), pp. 469-479, 1990.

T. Koch. "STEP-Based Modeling of Ship Product Definition Data". In: *Computer Applications in the Automation of Shipyard Operation and Ship Design, VII*. (eds.: VIEIRA, C et al.). Elsevier, pp. 365-376, 1992.

N. Laurance. "A High-Level View of STEP". *Manufacturing Review* 7, pp. 39-46, 1994.

D. Libes. *The NIST STEP Part 21 Exchange File Toolkit: An Update*. National Institute of Standards and Technology. Report NISTIR 5187, 1993.

M. Maybee, D. Heimbigner, and L. Osterweil. *Multilanguage Interoperability in Distributed Systems: Experience Report*. University of Massachusetts Amherst, Report UM-CS-1995-075, August 1995.

T. McCusker. "Workflow Takes on the Enterprise". *Datamation* 39, December 1993.

M. McLay and K.C. Morris. *The NIST STEP Class Library (STEP into the Future).* National Institute of Standards and Technology. Report NISTIR 4411, 1990.

M. Mead and D. Thomas. *Proposed Mapping from EXPRESS to SQL*. Rutherford Appleton Laboratory, May 1989.

E. Meis and R. Ostermayer. *Recommendations to the STEP Committee*. Kactus Consortium, Document KACTUS-01-RPK-D007 v. 1.1, 24 pp., September 28, 1996. Available from http://swi.psy.uva.nl/projects/NewKACTUS/home.html.

F. Metzger. "The Challenge of Capturing the Semantics of STEP Data Models Precisely". In: *Proceedings of the first PAKM'96, Practical Aspects of Knowledge Management Conference*. Basel, Swiss, October 30-31, 1996.

K.C. Morris. *Translating Express to SQL: A User's Guide*. National Institute of Standards and Technology. Report NISTIR 4341, 1990.

K.C. Morris, M. Mitchell, C. Dabrowski, and E. Fong. *Database Management Systems in Engineering*. National Institute of Standards and Technology. Report NISTIR 4987, 1992.

T. Mowbray. "How to Apply Open Systems to OO Architectures". *Object Magazine* 6 (1), pp. 84-86, 1996.

NIIIP. *National Industrial Information Infrastructure Protocols Consortium - The NIIIP Reference Architecture*. Vol. Report NTR96-01, Cycle 0, Revision 6, 652 pp., January 16, 1996.

Office of Science and Technology Policy. Federal Coordinating Council for Science, Engineering, and Technology. *High Performance Computing & Communications: Toward a National Information Infrastructure*. Report by the Committee on Physical, Mathematical, and Engineering Sciences, Washington D.C., 176 pp., 1994.

OMG. *Object Management Group - The Common Object Request Broker: Architecture and Specification*. Revision 2.0, July 1995.

OMG. *Object Management Group - Object Request Broker Architecture*. OMG TC Document 93.7.2, Framingham MA, 1993.

J. Owen. *STEP - An Introduction*. Information Geometers Ltd., Winchester, UK. 143 pp., 1993.

P.D.I.T. Product Data Integration Technologies, Inc. *STEP Management Overview*. Copies from the overhead presentation. Modules 1-7. Workshop on STEP sponsored by the National Institute of Standards and Technology (NIST), and developed by P.D.I.T. Gaithersburg MD, 1996.

V. Raghavan. *STEP Relational Interface*. Master's Thesis. Rensselaer Polytechnic Institute, Troy, New York, December 1992.

T. Rando and L. McCabe. *SDAI: An Object-Oriented Information Sharing Standard*. To be published, 1996.

T. Rando and L. McCabe. "Issues in Implementing the C Plus Plus Binding to SDAI". *Computer Standards & Interfaces* 16 (4), pp. 331-340, 1994.

T. Rando and M. Paoloni. "Mapping EXPRESS/SDAI into the CORBA Standard". In: *4th EXPRESS User Group (EUG'94) Conference*. 1994.

A. Redondo. "Padrões para Troca de Dados CAD: ACIS, IGES e STEP". In: *II Seminário Internacional Aplicações de STEP para a Integração de Sistemas*. Instituto de Pesquisas Tecnológicas, São Paulo SP, Brazil, November 27, 1996.

D. Sanderson. *Loss of Data Semantics in Syntax Directed Translation*. Ph.D. Thesis. Troy, NY: Rensselaer Polytechnic Institute, Computer Science, 1994.

D. Sanderson and D. Spooner. "Mapping Between EXPRESS and Traditional DBMS Models". In: *Proceedings of the EXPRESS Users Group EUG'93*. Berlin-Germany, October 2-3, 1993.

D. Sauder, M. Mitchell, and A. Feeney. *Challenges to the National Information Infrastructure: The Barriers to Product Data Sharing*. National Institute of Standards and Technology. Report NISTIR 5498, 1994.

D. Schenck and P. Wilson. *Information Modeling: The EXPRESS Way*. Oxford University Press, New York, 388 pp., 1994.

J. Siegel. *Re: What is the ISO reference of OMG-IDL?* E-mail message in-reply-to message <199709020335.XAA13658@emerald>. OMG expert's messages repository at http://www.omg.org/mhonarc/experts/, September 7, 1997.

R. Soley and C. Stone. (eds.). *Object Management Architecture Guide*. Third edition. John Wiley & Sons, Framingham MA, 164 pp., 1995.

SOLIS. *SC4 On Line Information Service*. Repository of STEP documentation, including STEP IR and AP schemata, as of July 17th, 1996. Available at ftp://ftp.cme.nist.gov/pub/subject/sc4.

G. Staub and M. Maier. "Object Modelling Technique (OMT) and EXPRESS - Comparison of 'Two Worlds'". *EXPRESS User's Group (EUG) Conference '95*, 1995.

STI. STEP Tools, Inc. "What is Your Application?" *STEP Tools News* 1 (2), September, 1992.

C. Stone and D. Hentchel. "Database Wars Revisited". *Byte* October 1990, pp. 233-242, 1990.

M. Stonebraker. "The Third-Generation Database System Manifesto: A Brief Retrospection". In: *Proceedings of the IFIP TC2/WG2.6 Working Conference on*

Object-Oriented Databases: Analysis, Design & Construction (eds.: R. Meersman, W. Kent, and S. Khosla). Elsevier, Windermere, UK, pp. 71-72, 1991.

S. Su, H. Lam, T. Lee, and J. Arroyo. "On Bridging and Extending OMG/IDL and STEP/EXPRESS for Achieving Information Sharing and System Interoperability". *EXPRESS User's Group (EUG) Conference '95*, 1995.

The Committee for Advanced DBMS Function. *Third Generation Data Base System Manifesto*. U.C. Berkeley Memorandum No. UCB/ERL M90/28, April 9, 1990.

F. Tibbits. "CORBA: A Common Touch for Distributed Applications". *Data Communications* 24 (7), pp. 71-75, 1995.

G. Trapp. "The Emerging STEP Standard for Product-Model Data Exchange". *IEEE Computer*, pp. 85-87, February 1993.

J. Tremblay and P. Sorenson. *The Theory and Practice of Compiler Writing*. McGraw-Hill, 1985.

D. Tsichritzis and A. Klug (eds.) "The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems". *Information Systems* 3, pp. 173-191, 1978.

S. Urban, J. Shah, M. Rogers, D. Jeon, P. Ravi, and P. Bliznakov. "A Heterogeneous, Active Database Architecture for Engineering Data Management". *International Journal of Computer Integrated Manufacturing* 7, pp. 276-293, 1994.

S. Vinoski. "Distributed Object Computing with CORBA". *C++ Report* July/August 1993.

A. Watson. "The OMG After CORBA 2". *Object Magazine* 6 (1), pp. 58-59, March 1996.

B. Whorf. *Language, Thought, and Reality*. MIT Press, 278 pp., 1956.

P. Wilson. "A View of STEP". In: WILSON, P.; WOZNY, M.; PRATT, M. (eds.) *Geometric Modeling for Product Realization*. Elsevier, pp. 267-296, 1993.

P. Wilson. “Information And/Or Data?” *IEEE Computer Graphics &* Applications 7, pp. 58-61, 1987.

N. Wirth. “What Can We Do About the Unnecessary Diversity of Notation for Syntactic Definitions?” *Communications of the ACM* 20 (11), pp. 822-823, 1977.

C. Wood. “Choosing an Engineering Object Data Management System”. In: CHASE, T. (ed.): *Engineering Data Management: Key to Integrated Product Development*. Proceedings of the 1992 ASME International Computers in Engineering Conference and Exposition, San Francisco-CA, pp. 1-14, 1992.

Y. Yang. “The STEP Integration Information Architecture”. In: Law, K. (ed.): *Engineering Data Management: Key to Success in a Global Market*. Proceedings of the 1993 ASME International Computers in Engineering Conference and Exposition, San Diego-CA, pp. 39-47, 1993.

X. Yang, J. Dong, and Z. He. “The Role and Application of STEP in CAD/CAPP/CAM Integration”. In: *IEEE TENCON'93*. Beijing, China, pp. 746-749, 1993.

I. Zeid. *CAD/CAM Theory and Practice*. McGraw-Hill, 1052 pp., 1991.

ANNEX A

ACRONYMS

AIC	Application Interpreted Construct
ANSI	American National Standards Institute
AP	Application Protocol
API	Application Programming Interface
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CAx	"Computer-Aided anything"; computer-aided engineering application systems such as CAD, or CAM
CIM	Computer Integrated Manufacturing
CNC	Computer Numeric Control
COM	Component Object Model
CORBA	Common Object Request Broker Architecture, from OMG
DBMS	Database Management System
DCE	Distributed Computing Environment
DDL	Data Description Language
DML	Data Manipulation Language
DML	Data Manipulation Language
DNC	Distributed Numerical Control
DXF	an Autodesk proprietary data format
ER	Entity-Relationship
EXPRESS	(not an acronym) Data description language used in STEP; ISO 10303-11
FEM	Finite Element Modeling
FMS	Flexible Manufacturing Systems
GKS	Graphical Kernel System, ISO standard
ICAM	Integrated Computer Aided Manufacturing Program, from the U.S. Air Force
IDEF1X	ICAM Definition, or: Integrated Definition Method for Information

	Modeling, extended.
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronic Engineers
IGES	Initial Graphics Exchange Specification
IR	Integrated Resource
ISO	International Standards Organization
KIF	Knowledge Interchange Format
NC	Numerical Control
NIAM	Nijssen's Information Analysis Method
NIIIP	National Industrial Information Infrastructure Protocols
NIST	National Institute of Standards and Technology
OLE	Object Linking and Embedding
OMA	Object Management Architecture
OMG	Object Management Group
OO	Object-oriented, or Object-orientation
OODB	Object-Oriented Database
OOP	Object-Oriented Programming
OOPL	Object-Oriented Programming Language
ORB	Object Request Broker
OSE	Open System Environments
OSF	Open Software Foundation
PDE	Product Data Exchange
PDES	Product Data Exchange Specification, or PDE using STEP
PHIGS	Programmer’s Hierarchical Interactive Graphic System, ISO/IEC 9592
RPC	Remote Procedure Calls
SC4	Sub-Committee 4 from ISO, responsible for STEP
SDAI	Standard Data Access Interface; ISO 10303-22
SET	Standard d'Exchange et de Transfert
SOLIS	SC4 On-Line Information Service
SPARC	Standards Planning and Requirements Committee
SQL	(pronounced 'Sequel') Structured Query Language
STEP	STandard for the Exchange of Product model data; ISO 10303
VDA/FS	Verband der Deutschen Automobilindustrie/Flächenschittstelle

ANNEX B

RESOURCES ON THE WORLD WIDE WEB

SOLIS (SC4 On-Line Information Service): the major source for online STEP documentation:
ftp://ftp.cme.nist.gov/pub/subject/sc4

ISO SC4 WWW site (at NIST):
http://www.nist.gov/sc4/

SC4 Mailing List Archives - e-mail exploder:
majordomo@cme.nist.gov ("subscribe sc4" in the message body)

MSID publication list (at NIST):
http://www.nist.gov/msidlibrary/pubs.htm

Project B-STEP (Centro Tecnológico para Informática, Brazil):
http://karran.ia.cti.br

ANNEX C

STEP STANDARD PARTS

The standards listed here are part of STEP. This list includes Parts that are either an International Standard (annotated "(IS)"), or a standard-to-be (committee draft, draft international standard, etc.). It should not be considered a comprehensive list, since standardization efforts may begin anytime and be discontinued without achieving IS status.

Overview and fundamental principles

ISO 10303-1: Overview and fundamental principles (IS)

Description methods

ISO 10303-11: The EXPRESS language reference manual (IS)

ISO 10303-12: The EXPRESS-I language reference manual

ISO 10303-13: Architecture and methodology reference manual

Implementation methods

ISO 10303-21: Clear text encoding of the exchange structure (IS)

ISO 10303-22: Standard Data Access Interface (SDAI)

ISO 10303-23: C++ language binding to SDAI

ISO 10303-24: C language binding to SDAI

ISO 10303-25: Fortran language binding to SDAI

ISO 10303-26: SDAI - IDL binding

Conformance testing methodology and framework

ISO 10303-31: General concepts (IS)

ISO 10303-32: Requirements on testing laboratories and clients

ISO 10303-33: Structure and use of abstract test suites

ISO 10303-34: Abstract test methods for Part 21 implementations

ISO 10303-35: Abstract test methods for Part 22 implementations

Integrated generic resources

ISO 10303-41: Fundamentals of product description and support (IS)

ISO 10303-42: Geometric and topological representation (IS)

ISO 10303-43: Representation structures (IS)

ISO 10303-44: Product structure configuration (IS)

ISO 10303-45: Materials

ISO 10303-46: Visual presentation (IS)

ISO 10303-47: Shape variation tolerances

ISO 10303-48: Form features

ISO 10303-49: Process structure and properties

Integrated application resources

ISO 10303-101: Draugthing (IS)

ISO 10303-103: Electrical and Electronic Connectivity

ISO 10303-104: Finite element analysis

ISO 10303-105: Kinematics

ISO 10303-106: Building construction core model

Application protocols

ISO 10303-201: Explicit draughting (IS)

ISO 10303-202: Associative draughting

ISO 10303-203: Configuration controlled design (IS)

ISO 10303-204: Mechanical design using boundary representation

ISO 10303-205: Mechanical design using surface representation

ISO 10303-207: Sheet metal die planning and design

ISO 10303-208: Life cycle product change process

ISO 10303-209: Composite and metallic structures analysis and related design

ISO 10303-210: Electronic printed circuit assembly, design and manufacture

ISO 10303-211: Electronic test diagnostics and remanufacture

ISO 10303-212: Electrotechnical plants

ISO 10303-213: Numerical control (NC) process plans for machined parts

ISO 10303-214: Core data for automotive mechanical design processes

ISO 10303-215: Ship arrangement

ISO 10303-216: Ship molded form

ISO 10303-217: Ship piping

ISO 10303-218: Ship structures

ISO 10303-220: Printed circuit assembly manufacturing planning

ISO 10303-221: Process plant functional data and its schematic representation
ISO 10303-222: Exchange of product definition data from design engineering to manufacturing engineering for composite structures
ISO 10303-223: Exchange of design and manufacturing product information for cast parts
ISO 10303-224: Mechanical products definition for process planning using form features
ISO 10303-225: Structural building element using explicit shape representation
ISO 10303-226: Ship mechanical systems
ISO 10303-227: Plant Spatial Configuration
ISO 10303-228: Building services: Heating, ventilation and air conditioning
ISO 10303-230: Building structural frame: Steel work

Abstract Test Suites

ISO 10303-301: Explicit draughting
ISO 10303-302: Associative draughting
ISO 10303-303: Configuration controlled design
ISO 10303-304: Mechanical design using boundary representation
ISO 10303-305: Mechanical design using surface representation
ISO 10303-307: Sheet metal die planning and design
ISO 10303-308: Life cycle product change process
ISO 10303-309: Design through analysis of composite and metallic structures
ISO 10303-310: Electronic printed circuit assembly, design and manufacture
ISO 10303-311: Electronic test diagnostics and remanufacture
ISO 10303-312: Electrotechnical plants
ISO 10303-313: Numerical control (NC) process plans for machined parts
ISO 10303-314: Core data for automotive mechanical design processes
ISO 10303-315: Ship arrangement
ISO 10303-316: Ship molded form
ISO 10303-317: Ship piping
ISO 10303-318: Ship structures
ISO 10303-320: Printed circuit assembly manufacturing planning
ISO 10303-321: Process plant functional data and its schematic representation
ISO 10303-322: Exchange of product definition data from design engineering to manufacturing engineering for composite structures
ISO 10303-323: Exchange of design and manufacturing product information for cast parts

ISO 10303-324: Mechanical products definition for process planning using form features

ISO 10303-325: Structural building element using explicit shape representation

ISO 10303-326: Ship mechanical systems

ISO 10303-327: Plant Spatial Configuration

ISO 10303-328: Building services: Heating, ventilation and air conditioning

ISO 10303-330: Building structural frame: Steel work

Application Interpreted Constructs

ISO 10303-501: Edge-based wireframe

ISO 10303-502: Shell-based wireframe

ISO 10303-503: Geometrically bounded 2D wireframe

ISO 10303-504: Draughting annotation

ISO 10303-505: Drawing structure and administration

ISO 10303-506: Draughting elements

ISO 10303-507: Geometrically bounded surface

ISO 10303-508: Non-manifold surface

ISO 10303-509: Manifold surface

ISO 10303-510: Geometrically bounded wireframe

ISO 10303-511: Topologically bounded surface

ISO 10303-512: Faceted boundary representation

ISO 10303-513: Elementary boundary representation

ISO 10303-514: Advanced boundary representation

ISO 10303-515: Constructive solid geometry

ISO 10303-516: Mechanical design context

ISO 10303-517: Mechanical design geometric presentation

ISO 10303-518: Mechanical design shaded presentation

VITA

I was born in Porto Alegre, Rio Grande do Sul state, Brazil, in 1964. My parents raised me and my two younger brothers in Três Coroas, a small town in the German area of Rio Grande. I learnt most of what I know from my family, and I think this is a good one: in 1977 I was in the seventh grade, and my two first monthly grades in Math were just terrible. My Dad visited the school and asked for the *preventive recuperation* (an early alternative to summer school), a right of any student (but I never heard of anybody who used that right). I spent two afternoons working with my Math teacher, and the result was an A in each one of the seven tests of the following month. I don't think you would read this if not because of that two afternoons. Later, I took civil engineering at the Federal University of Rio Grande do Sul (proud of it!), then moved to the neighbor state of Santa Catarina to get the master's degree. My first plan was to travel abroad for the doctorate, but I couldn't leave this paradisiac island. When the opportunity to enroll in the "sandwich" program (credits taken in Brazil, research abroad, defense back in Brazil) came up, I spent two years in the United States under the advising of Dr. Jan Helge Bøhn, at Virginia Tech. In my last year abroad, I worked as a guest researcher at the National Institute of Standards and Technology (NIST), occasionally traveling to Virginia to hear Dr. Bøhn's advice. These were great opportunities for my professional and personal growth. Moreover, NIST is the only place I know where people love football (the one played with the feet) so much as to sacrifice lunch time three times a week to play it. I love football and if it were not for my poor skills, I would have graduated from the same University where Falcão, Batista, Dunga, and Taffarel graduated -- Sport Club Internacional, beloved *colorado*. I also love teaching, which I consider more a privilege than work. A professor must be a free thinker, an independent, a role model for the students. This is the only way to justify the privilege of getting older among young, talented, potentially thinking people. For the future, I plan to keep teaching, advising, researching, and occasionally windsurfing around Santa Catarina Island, where I live. I´m married to Luciana; we have plans of raising a family in a house by the sea after she finishes *her* doctorate, but that´s another *vita*...