UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

A INTEROPERABILIDADE DE INFORMAÇÕES GEOGRÁFICAS EM AMBIENTE MARÍTIMO.

por

ROGÉRIO PERROTI BARBOSA

Junho/2005

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

.

A INTEROPERABILIDADE DE INFORMAÇÕES GEOGRÁFICAS EM AMBIENTE MARÍTIMO.

por

ROGÉRIO PERROTI BARBOSA

Dissertação submetida à avaliação, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Ubiratan Porto dos Santos
Orientador

Junho/2005

FOLHA DE JULGAMENTO

A Interoperabilidade de Informações Geográficas em Ambiente Marítimo.

Rogério Perroti Barbosa

Dissertação de Mestrado submetida ao corpo docente do Instituto de Matemática e Núcleo de Computação Eletrônica (IM/NCE) – Universidade Federal do Rio de Janeiro – UFRJ, como parte dos requisitos necessários à obtenção do grau de Mestre.

Aprovada por:

_____

Prof. Ubiratan Porto dos Santos, Ph.D.(Orientador)

_____

Prof. Jose Carlos Sícoli Seoani, D.Sc. (Co-orientador)

_____

Profª. Margareth Simões Penello Meirelles, D.Sc.

_____

Prof. Carlo Emmanoel Tola de Oliveira, Ph.D.

Agradecimentos

Em primeiro lugar agradeço a DEUS que me deu a oportunidade de nascer perfeito e capaz de realizar um trabalho que jamais esquecerei, pois, mais do que os conhecimentos adquiridos, a elaboração desta dissertação mostrou-me a necessidade de organização e planejamento em todas as tarefas difíceis da vida. Agradeço, também, ao meu chefe imediato João Alberto Viana Tavares, que sempre procurou disponibilizar o tempo necessário para que eu pudesse elaborar a grande pesquisa que se fez necessária; à minha esposa que soube compreender o meu nervosismo, quando muitas vezes me sentia perdido e sem encontrar todas as informações necessárias, passando um grande tempo dedicado à busca de elementos que pudessem contribuir, de forma efetiva, para o produto final; ao professor Cainho que despertou em mim o entusiasmo pelo trabalho, com seu exemplo de dedicação e capacidade; a comandante Tereza que com sua enorme boa vontade e paciência efetuou uma revisão de formato em todo o trabalho; e, finalmente, ao meu orientador que tornou o produto desta pesquisa um trabalho de final de curso para a obtenção do grau de Mestre em Ciência da Computação.

RESUMO

Cada vez mais a necessidade de uma padronização torna-se evidente em todos os ramos da navegação. O grande número de sistemas que tem surgido tenta oferecer recursos importantes que deverão, no futuro, substituir os meios tradicionais de condução de uma embarcação baseados em cartas de papel.

A Organização Hidrográfica Internacional elaborou uma forma de garantir a troca de informações entre os sistemas de navegação existentes, que visa manter não só os padrões pré-estabelecidos, bem como utilizar a metodologia de desenvolvimento baseada em especificações bem definidas, elaboradas por órgãos responsáveis pela navegação marítima ao redor do mundo.

Este trabalho descreve os padrões existentes e as tecnologias desenvolvidas para proporcionar meios para a confecção de Sistemas de Informação e Visualização de Cartas Eletrônicas, conhecidos como ECDIS, que devem respeitar as características propostas e seguir padrões de apresentação. Os padrões pesquisados são: S-57, S-52 e 8211. Apresenta, também, um panorama dos atuais sistemas que já se baseiam na aplicação direta destas tecnologias. Sistemas tais como: ORCA Navigator, Navi Sailor 3000, Tokimec EC-7500, Dkart Navigator e Navmaster Professional, são analisados apresentando suas próprias estruturas de armazenamento de dados chamadas de Sistema de Cartas Náuticas Eletrônicas, ou simplesmente SENC, cuja implementação tornou-se um segredo que proporciona o diferencial de eficiência na recuperação das informações armazenadas necessárias à navegação segura.

A apresentação de ferramentas de código aberto tais como OpenEV com a biblioteca Lib-S52, e a biblioteca de leitura do formato 8211, propicia, ainda, uma possibilidade de desenvolvimento de um Sistema de Informação e Visualização de Cartas Eletrônicas livre de associações a tecnologias proprietárias, possuidoras de patentes exclusivas de seus criadores.

ABSTRACT

Electronic navigation standards are an area of major concern, for safety and security at sea. A large number of systems have been developed to fulfill this need and, in the future, they should replace traditional nautical charts in paper.

The International Maritime Organization – IMO – has been the major player on this standardization process in order to assure interoperability between the electronic charts and navigation systems deployed by hydrographic bureaus from various countries.

The present work describes the standards and technologies available to provide means for production, storage and presentation of electronic charts. The standards are: S-57, S-52 e 8211. It also gives an overview of systems which currently utilize these technologies. Such systems are called Eletronic Chart Display and Information System. The ORCA Navigator, Navi Sailor 3000, Tokimec EC-7500, Dkart Navigator and Navmaster Professional, are analised. They adopt their own proprietary data structures for an efficient data manipulation and visualization which are crucial for navigation in real-time.

Finally, this work adopts open software and standards to circumvent the intellectual property restrictions, imposed by patent rights, for the development of an Open Electronic Chart Display and Information System – ECDIS.

SUMÁRIO

FIGURAS

TABELAS

ACRÔNIMOS E GLOSSÁRIO DE TERMOS UTILIZADOS

ActiveX

Conjunto de tecnologias que permite aos componentes do “software” interagir com qualquer outro “software”, em um ambiente de rede, independente da linguagem em que ele foi criado. ActiveX foi construído a partir de tecnologia COM e, por ser uma plataforma aberta, habilita a elaboração de aplicações a partir de padrões. Suporta uma enorme quantidade de ferramentas, incluindo Microsoft Visual Basic e Java, e permite sua integração com padrões de mercado, incluindo HTML, Java, TCP/IP e COM.

AIS “Automatic Identification System”

Sistema de comunicação e identificação automático para garantir uma navegação segura, propiciando uma operação eficiente no controle da embarcação.

Área

Primitiva geométrica de duas dimensões de um objeto que especifica sua localização.

ARCS

Padrão de cartas “raster”, com cobertura mundial, possuindo mais de trezentas e cinqüenta cartas que são atualizadas semanalmente.

ARPA

Sistema atualmente incorporado aos radares que permite o monitoramento automático de alvos.

Atributo

A característica de um objeto, podendo ser quantitativo ou qualitativo.

Borda

Um objeto espacial com uma dimensão, localizado por um ou mais pares de coordenadas.

BSB

Padrão de cartas “raster”, geradas a partir de cartas náuticas, que utiliza alta resolução e grande quantidade de cores na sua elaboração.

Célula

A unidade básica para a distribuição de uma ENC, que cobre uma área localizada entre dois meridianos e dois paralelos, cujo conteúdo não pode exceder cinco Mbytes de informações, sendo a mesma destinada a uma proposição específica de navegação.

Cadeia de Nós

Uma estrutura de dados cuja geometria é descrita em termos de bordas, nós isolados e nós conectados.

Carta

Uma carta é projetada para conter os requisitos marítimos de navegação, contendo, por exemplo: perigos à navegação, águas profundas, características de costa, etc.

Carta Eletrônica

Termo abrangente que descreve dados, “software” e o sistema eletrônico que tem a capacidade de mostrar informações das cartas.

Cartas C-Map CM93/3

Cartas vetoriais produzidas pela Morintech Navigation, cobrindo o globo em 9 zonas. Vem se tornando um padrão para vários tipos de aplicações comerciais.

Carta Eletrônica de Navegação (ENC)

A base de dados padronizada no seu conteúdo e formato para uso em ECDIS. É produzida por empresas hidrográficas autorizadas pelo governo.

Catálogo de Objetos

Provém de uma descrição das entidades do mundo real, contendo uma lista de objetos descritivos, espaciais e seus atributos.

CIE

Método para representação de cores, proposto pela Comission Internationale de l'Éclairge (CIE). Elaborado para suprir algumas deficiências do padrão RGB.

Classe de Objetos

Uma descrição genérica de objetos que possuem as mesmas características.

COM (Component Object Model)

Trata-se de uma plataforma aberta, que combina tecnologias Web e tradicional (desktop). Isto permite que os profissionais desenvolvedores e produtores Web criem aplicações únicas, integrando aplicação e conteúdo Web.

ECDIS (Eletronic Chart Display and Information System)

Um sistema de informações para a navegação, com características que permitem atualização das cartas, segundo a regulamentação V/20, da convenção de SOLAS de 1974, apresentando informações selecionadas de uma carta eletrônica de sistema de navegação (SENC).

Face

Objeto espacial de duas dimensões. A face é uma área contínua definida por duas ou mais bordas que a limitam.

Geoprocessamento

O termo denota a disciplina do conhecimento que utiliza técnicas matemáticas e computacionais para o tratamento de informação geográfica.

GNU

Licença que permite compartilhar e alterar um “software” que, uma vez modificado ou distribuído, deve seguir algumas restrições que visam garantir suas características de “software” livre.

GPS (Global Positioning System)

Sistema de posicionamento global operado pelo Governo dos Estados Unidos. Quando usado com correções diferenciais é conhecido como GPS diferencial (DGPS).

Gráfico Planar

Uma estrutura de dados com duas dimensões cuja geometria é descrita em termos de nós e bordas, ligados pela topologia.

IEC (International Electrotechnical Commission)

É a federação internacional dos organismos nacionais de normalização para a área elétrica. É uma organização não- governamental internacional, criada em 1906. No Brasil é representada pela ABNT - Associação Brasileira de Normas Técnicas.

IHO (International Hydrographic Organization)

Coordena as atividades das empresas hidrográficas, promovendo a elaboração de padrões para a produção de cartas náuticas e publicações.

IMO (International Maritime Organization)

A agência responsável por promover uma navegação internacional segura e controlar a poluição provocada por embarcações.

Meta Objeto

Um objeto descritivo que contém informações sobre outros objetos.

Modelo de Dados

Uma especificação de um conjunto de componentes e relacionamentos entre os elementos pertinentes a um determinado fenômeno, definido por um modelo da realidade.

Nó

Um objeto espacial de dimensão zero, localizado por um par de coordenadas.

NIST

National Institute of Standards and Technology, órgão americano responsável pela aprovação de padrões e tecnologias.

NMEA

Padrão de comunicações para vários dispositivos da indústria eletrônica marítima, tendo como exemplo o GPS (define requerimento de sinais elétricos, protocolos de transmissão de dados e tempos).

Objeto

Um conjunto de informações identificável, que pode conter atributos e estar relacionado a outros objetos, podendo ser espacial ou descritivo.

Objeto Cartográfico

Objeto descritivo que contem informações sobre a representação cartográfica (incluindo texto das entidades do mundo real).

Objeto Feição

Um objeto que não contém informações de localização das entidades do mundo real, contendo informações características pertinentes ao mesmo.

Pan

Abreviação de panorâmica.

Panorâmica

Movimento da câmera (da esquerda para a direita ou vice-versa) que mostra uma tomada geral de um objeto, cena ou pessoa que está sendo filmada, gravada ou exibida.

Relacionamento

Uma ligação lógica entre dois elementos do modelo de dados.

RGB

Método para representação perceptual de cores, utilizando as cores básicas vermelho (“Red”), verde (“Green”) e azul (“Blue”), proposto pela Comission Internationale de l'Éclairge (CIE).

S-52

Uma publicação da IHO destinada a especificar o conteúdo das cartas e aspectos de apresentação em ECDIS.

S-57

Uma publicação da IHO destinada a descrever um padrão de transferência para dados hidrográficosdigitais.

S-57 ENC

Cartas Náuticas Eletrônicas no formato S-57.

SENC

Carta eletrônica de sistema de navegação.

“software” livre

“software” de utilização livre. Todos podem contribuir com ele, seja no seu desenvolvimento, seja na correção de erros, seja na documentação, desde que a condição de liberdade seja mantida.(http://www.dextra.com.br/opensource/os-introducao.htm)

SOLAS

Convenção internacional para a vida segura no mar, desenvolvida pela IMO.

Topologia

Um conjunto de propriedades das formas geométricas definidas no modelo de dados.

UML

É a sigla de Unified Modeling Language – linguagem unificada de modelagem. Foi criada, basicamente, para que analistas e programadores falassem a mesma língua. Mas também para se criar um padrão de desenvolvimento de softwares.

Vetor

Conexão direta entre dois pontos, referenciado por dois conjuntos de coordenadas ou pela direção e distância entre estes conjuntos.

Waypoints

Pontos definidos para a utilização em confecção de rotas ou marcação de posições específicas nas cartas.

Zoom

Movimento de aproximação (zoom in) ou afastamento (zoom out) da câmera de um objeto, pessoa ou cena que está sendo filmada, gravada ou exibida.

CAPÍTULO 1

1 Introdução

Segundo CAMARA G [1], as ferramentas computacionais para Geoprocessamento, chamadas de Sistemas de Informação Geográfica (SIG), permitem realizar análises complexas, ao integrar dados de diversas fontes e ao criar bancos de dados georreferenciados. Sua aplicação vem se diversificando cada vez mais com o passar dos anos, sendo utilizados em várias áreas do conhecimento humano, tais como: cartografia, administração de recursos naturais, controle de epidemias, planejamento urbano, monitoramento costeiro, etc.

A terra possui a maioria de sua superfície composta por mares e oceanos, de forma que o domínio do conhecimento geográfico marítimo torna-se fundamental para os interesses de uma nação, principalmente para países como o Brasil, que possem grande extensão costeira. Neste cenário os SIGs marítimos desempenham um papel de extrema importância, contribuindo não só para a navegação comercial, como também para a navegação de patrulhamento, garantindo assim a soberania do país.

No futuro os SIG deverão compartilhar informações, atualizando os dados de forma padronizada, permitindo a manutenção da coerência do cenário representado.

Neste panorama se insere o presente trabalho, que explora a utilização de padrões de transporte de informações para que, no futuro, os Sistemas de Informações Navais possam interagir de forma atualizada e confiável, levando em conta a utilização de um modelo de objetos definidos no padrão de dados hidrográficos S-57. Esta padronização servirá como base para implementação de SIGs utilizados pela Marinha do Brasil.

1.1 Motivação

Atualmente a navegação exige bem mais do que apenas a consulta a mapas feitos em papel, os quais têm orientado os navegadores por muito tempo. A complexidade das embarcações e o grande fluxo existente, principalmente em áreas de ancoradouro, vêm exigindo uma solução que apresente maior velocidade de visualização das informações

pertinentes ao meio naval. Os sistemas existentes que utilizam, na sua maioria, cartas rasterizadas não contêm flexibilidade suficiente para prover uma navegação extremamente segura, não permitindo, assim, a exibição de um volume de informações necessárias à condução das embarcações modernas. Com este intuito surgiram os Sistemas de Informações Geográficos Navais, ou ECDIS (da sigla na língua inglesa para Sistemas de Informação e Visualização de Cartas Eletrônicas). Na realidade, estes sistemas desenvolvem um papel bem maior do que apenas meros visualisadores de cartas, sendo sistemas que interagem com sensores e operadores, os quais são capazes de obter rapidamente informações do quadro existente e tomarem as decisões corretas no tocante à condução da embarcação.

Pensando nisso, a Organização Hidrográfica Internacional (IHO), em conjunto com outros órgãos, introduziu um novo padrão para troca de informações destinado a sistemas de informações em ambientes navais. Este novo padrão está conduzindo o mercado de produção de ECDIS para um novo estágio de evolução, tornando-se aceito mundialmente e, conseqüentemente, exigindo um melhor estudo e aprimoramento das técnicas de implementação, bem como sua aplicabilidade. Estes Sistemas exigem uma forma de armazenamento de informações que estão diretamente ligadas às necessidades da aplicação. A preocupação em utilizar estas informações em ECDIS ou somente em sistemas ECS (Sistemas de Visualização de Cartas Eletrônicas) define o nível de armazenamento que será exigido para implementar as funções pertinentes a esses sistemas.

1.2 Objetivos

Os principais objetivos deste trabalho de pesquisa são:

- Analisar as necessidades de utilização de um padrão de interoperabilidade entre sistemas de informação navais;
- Efetuar um levantamento dos principais padrões de troca de dados espaciais existentes;
- Mostrar a possibilidade de utilização do modelo de orientação a objeto sobre dados geográficos, para utilização em sistemas de informação navais;
- Efetuar um levantamento dos principais padrões de apresentação de dados espaciais existentes; e

- Apresentar as tecnologias existentes que implementam soluções para desenvolvimento de aplicações e utilização dos padrões existentes em ECDIS.

1.3 Organização do Trabalho

A dissertação está distribuída nos seguintes capítulos:

O capítulo 2 descreve alguns conceitos básicos necessários para um melhor entendimento do trabalho, tais como:

- UML (Unified Modeling Language) e a Orientação a Objetos; e
- Conceitos sobre Sistemas de Banco de Dados Geográficos.

O capítulo 3 trata do estudo do padrão S-57, da IHO.

O capítulo 4 analisa o formato de troca da NIST (National Institute of Standards and Technology) - padrão 8211.

O capítulo 5 apresenta uma implementação e utilização de uma biblioteca de leitura para o padrão de troca, elaborado por Frank Warmerdam (http://home.gdal.org/~warmerda/) .

O capitulo 6 trata do estudo do padrão de apresentação S-52, originado pela IHO.

O capítulo 7 apresenta exemplos de utilização dos padrões expostos.

O capítulo 8 apresenta a conclusão.

CAPÍTULO 2

2 Conceitos Envolvidos

Faz-se necessária uma abordagem de determinados conceitos e definições para sedimentar o tema deste trabalho. Este capítulo descreve os pontos importantes relativos à atual tecnologia de Sistemas de Informações Geográficas e Banco de Dados Geográficos. Temas como o paradigma da Orientação a Objetos (OO), Sistemas de Informações Geográficas e formatos padrão de armazenamento de dados geográficos serão abordados também.

Segundo MITROVIC [2], o paradigma OO se adapta melhor a aplicações não tradicionais como SIG, CAD e CASE. A capacidade de representar o componente funcional (comportamento) das entidades (objetos), e não somente a sua estrutura, é a grande diferença entre o paradigma da Orientação a Objetos e os modelos de dados anteriores a este, tais como os modelos relacionais.

2.1 UML e Orientação a Objetos

Tendo em vista o interesse por novas abordagens de desenvolvimento de “software” que permitissem uma melhor compreensão do código gerado e uma possibilidade mais simples de expansão e alteração do sistema, nasce neste meio a tecnologia de Orientação a Objetos, criada por Grady Booch, James Rumbaugh e Ivar Jacobson, na Rational Software Corporation, como um dos paradigmas mais proeminentes na solução destes crescentes problemas.

A Orientação a Objetos permite que se represente, naturalmente, os objetos do mundo real, incluindo seus atributos, relacionamentos e comportamentos, tornando-os mais próximos da maneira de pensar do usuário final. Pode-se considerar uma aplicação Orientada a Objetos como sendo um conjunto de objetos com seus estados pertinentes e que interagem entre si, sendo que, devido à sua característica modular, tornam-se mais fáceis de manter. Uma alteração realizada no código de um objeto não altera os outros objetos do sistema. No paradigma da Orientação a Objetos o acoplamento do sistema é reduzido devido à eliminação

da necessidade de compartilhamento de áreas de dados, estimulando a reusabilidade, onde os objetos são autocontidos e podem ser usados em outras aplicações suficientemente similares.

No final dos anos 80 surge a UML [3], a sucessora dos métodos de análise e projeto Orientado a Objetos, unificando os três métodos mais populares da época: RUMBAUGH [4], JACOBSON [5] e BOOCH [6]. Esta proposta baseia-se em um conjunto de diagramas e uma metodologia unificada que visam facilitar o processo de desenvolvimento de aplicações Orientadas a Objetos por intermédio de uma notação gráfica.

Os conceitos básicos de UML (encapsulamento, abstração, associação e agregação) e de OO (classes, objetos, herança, polimorfismo, atributos e métodos) serão apresentados a seguir.

2.1.1 Encapsulamento

Segundo BOOCH, "e*ncapsulamento é o processo de divisão dos elementos de uma abstração que constituem sua estrutura e comportamento; encapsulamento serve para separar a interface contratual de uma abstração de sua implementação*" [6], podendo-se, portanto, vê-lo como a maneira pela qual a metodologia isola os elementos pertinentes da implementação, os quais só podem ser acessados via uma interface bem definida. RUMBAUGH, seguindo uma linha semelhante, diz que: "*o encapsulamento consiste na divisão dos aspectos externos de um objeto, os quais podem ser acessados por outros objetos, e dos detalhes internos pertinentes à sua implementação, que ficam ocultos aos demais objetos*" [4].

Com o encapsulamento, detalhes internos dos objetos podem ser alterados sem que as aplicações que os utilizam sejam afetadas, promovendo, assim, uma maior independência entre os mesmos. Qualquer alteração no código para eliminar um erro ou aumentar o desempenho fica restrita ao objeto que possui o código, facilitando sua evolução e manutenção de forma segura.

2.1.2 Abstração

RUMBAUGH define abstração como sendo uma análise seletiva de certos aspectos de um problema, isolando apenas os aspectos importantes [4]. Já BOOCH define abstração como sendo a principal característica de um objeto que o diferencia dos demais [6]. Resumindo-se, o objetivo principal da abstração é a identificação do comportamento de objetos para posterior implementação, podendo ser definida tanto como um processo como uma identidade. Como processo, representa a extração dos detalhes essenciais de um item ou grupo, desprezando os detalhes irrelevantes. Como identidade, denota uma representação resumida de uma entidade do mundo real.

A engenharia de "software" identifica dois tipos de abstração: de dados e de função. Na abstração de dados, tantos os dados como a implementação não são de interesse do usuário. Ela estende o conceito de abstração de função, que concentra sua atenção na interface da função, ignorando a sua implementação. A implementação de uma lista de dados pode ser feita através das funções que inserem e retiram de uma lista ("insert" e "delete"), escondendo a implementação das funções e a estrutura de dados. Deste modo, qualquer alteração na implementação das operações ou na estrutura de dados da lista não afeta os usuários.

2.1.3 Objetos e Classes

As entidades físicas ou conceituais que encontramos à nossa volta podem ter seus aspectos relevantes modelados por uma abstração de "software" conhecida como objeto. Podemos ter, dependendo do ponto de vista, diferentes abstrações. Para o engenheiro automotivo, os pneus, as portas, o motor, a velocidade máxima, o consumo de combustível, podem ser considerados objetos. Em um prédio, as portas, as janelas, os móveis, podem ser considerados objetos; o próprio prédio pode ser considerado objeto, se olharmos por um nível de abstração mais alto.

Na geomática, a identificação de objetos abre grandes possibilidades, pois, dependendo do objeto em questão, o especialista poderá utilizar uma maior quantidade de objetos para modelar o fenômeno de interesse. A composição da superfície da terra, do

subsolo, da atmosfera, das florestas e dos mares, pode levar a variações de representação, possibilitando uma rica utilização de objetos para sua modelagem.

Outro aspecto relevante na caracterização dos objetos é o que usualmente chama-se de estado. O estado pode ser descrito como uma condição momentânea ou um conjunto de circunstâncias que descreve o objeto. O estado de um motor poderia incluir o número de rotações por minuto, o estado de um rio poderia incluir a sua vazão, o estado de um liquidificador poderia ser “ligado” ou “desligado” e o estado de um líquido poderia incluir a sua temperatura.

Para entidades complexas tais como elementos da natureza, costuma-se utilizar os possíveis estados que são relevantes ao nosso modelo (abstração).

Deve-se levar em conta que para a comunicação entre objetos faz-se necessária a utilização de métodos e serviços, sendo que é por intermédio dos métodos que a definição do comportamento é efetuada. O objeto possui, usualmente, dois métodos reservados: o método “construtor” e o método “destrutor”, chamados sempre que o objeto é criado e destruído, respectivamente.

Os atributos de um objeto em geral não são acessados diretamente, o usuário deverá utilizar os métodos fornecidos para poder manipulá-los. Ver Figura 1.

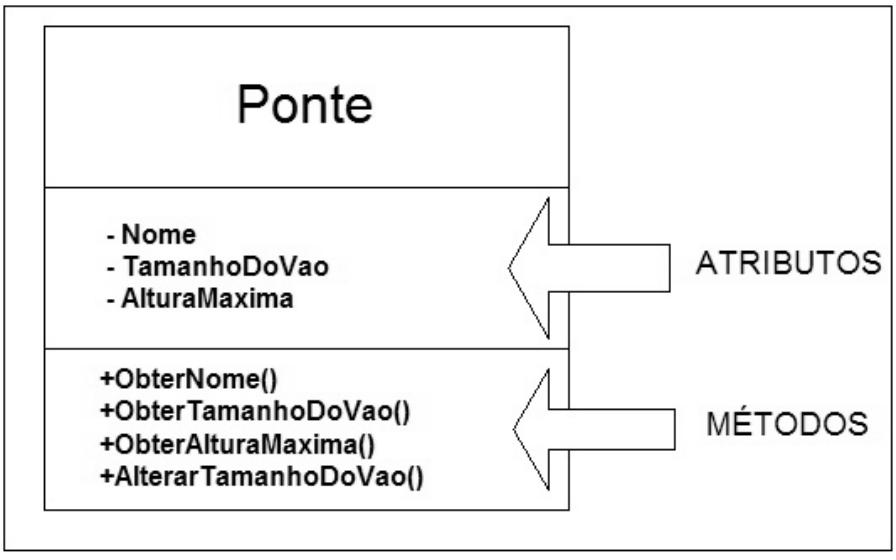


Figura 1 – Classe (Atributos e Métodos)

As classes possuem um comportamento bem definido, com relacionamentos bem semelhantes. Pode-se dizer que classe é um padrão que propicia uma descrição dos atributos e métodos comuns aos objetos de mesmo tipo. A modelagem de um porto, por exemplo, poderia possuir classes do tipo bóia, do tipo área de ancoradouro, do tipo canal, etc. Na Baía de Guanabara poderíamos ter a Ponte Rio-Niterói como um objeto da classe “Ponte”, onde este objeto possuiria atributos do tipo: “Tamanho do Vão”, “Altura máxima”, e assumiria métodos, tais como: “Obter Altura Máxima”, o qual retornaria, uma vez invocado, a altura máxima permitida para o cruzamento da ponte.

2.1.4 Herança

Herança pode ser definida como a propriedade que permite a um objeto adquirir características de um outro. Esta propriedade pode ser considerada um relacionamento entre os objetos, que permite definir uma nova classe de objetos partindo-se de outra já existente, onde a classe que cede as características é chamada de superclasse ou classe base. A nova classe herda as características da superclasse ou classe base (valores, referências, comportamentos), podendo substituir ou redefinir essas características. A classe “Bóia de Perigo Isolado” poderia herdar as características da classe “Bóia”, onde esta é uma generalização da primeira. Da mesma forma, “Bóia de Perigo Isolado” é uma especialização da classe “Bóia”.

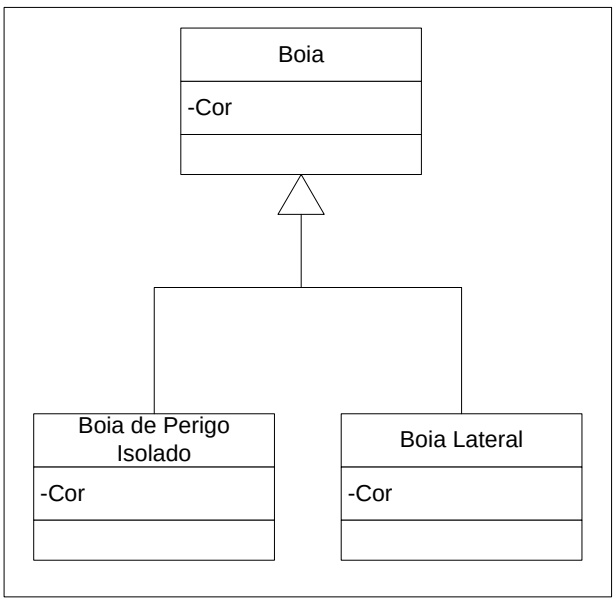


Figura 2 - Herança

2.1.5 Métodos e Comportamentos

Os métodos são o centro da tecnologia OO. Cada classe de objetos terá os métodos básicos herdados da classe base, podendo possuir outros comportamentos definidos internamente que são seus métodos intrínsecos. Os métodos podem ser do tipo:

- Métodos de valor (também conhecidos como atributos derivados), retornam uma resposta para a mensagem. Os resultados aparecem como atributos numa chamada. Ex: área, comprimento;
- Métodos reflexos, ocorrem automaticamente durante o ciclo de vida de um objeto. Ex: criação, modificação e exclusão;
- Métodos reflexos de validação, permitem a colocação de métodos de validação em cada classe de objetos. A mudança para um objeto referenciado é um método reflexo que permite a propagação de efeitos de um objeto para outro; e
- Métodos processuais, ocorrem por uma requisição do operador. Usados para checagem de dados, formação de polígonos e generalizações.

2.1.6 Métodos de Validação de Integridade de Dados

A integridade dos dados é, sem dúvida, a maior preocupação na manutenção de uma grande base de dados geoespacial . Para um ECDIS esta integridade torna-se fundamental para que se possa ter um controle seguro da navegação. O modelo de objetos de dados permite a definição da lógica, as regras e os métodos reflexos desses dados geoespaciais no esquema do banco de dados. Como estas regras são objetos do modelo, fica obrigatório seu uso na entrada de objetos e na alteração dos mesmos no banco. Quando um objeto está sendo modificado, mensagens são enviadas para o objeto em vários estágios:

- Uma mensagem antes que a modificação comece, para checar uma permissão; e
- Uma mensagem depois que a modificação terminou, para checar se a mesma foi válida.

Se algum método de validação reflexo retorna negativo, a transação inteira é desfeita e o estado do objeto retorna ao início. Este tipo de validação não serve para edições interativas.

2.1.7 Polimorfismo

É definido em Orientação a Objetos como sendo um código que possui “vários comportamentos” ou que produz “vários comportamentos”. Mais especificamente, um código que pode ser aplicado a várias classes de objetos, mantendo seu comportamento transparente para quaisquer tipos de argumentos; isto é, a mesma mensagem enviada a objetos distintos poderá provocar reações diferentes por parte destes objetos. Um método polimórfico é aquele que pode ser aplicado a várias classes de objetos sem que exista qualquer inconveniente. Segundo FERREIRA & JARABECK, um exemplo simples de polimorfismo é dado por um moedor de carne, que tem a função de moer carne, produzindo carne moída para fazer bolinhos. Desse modo, não importa o tipo de carne alimentada (classe), o resultado será sempre carne moída, não importa se de boi, de frango ou de qualquer outro tipo. As restrições impostas estão no próprio objeto e são definidas pelo fabricante, e não pelo usuário do produto [7].

2.1.8 O Modelo de Dados Orientado a Objetos

Em um banco de dados OO, as entidades do mundo real são representadas como objetos, sendo que todos pertencem a uma determinada classe de objetos. Uma classe pode ter vários objetos, porém um objeto só pode pertencer a uma classe que define quais valores ele pode assumir. Valores podem ser tipos de dados simples (inteiros, ”strings”, datas, etc.) ou complexos (geométricos, localização, tabelas “raster”). Objetos podem armazenar informações estruturais ou referências entre outros objetos.

Na Orientação a Objetos os métodos são definidos nos próprios objetos, sendo esses métodos responsáveis pelo comportamento dos mesmos. Deste modo, um método é chamado enviando-se uma mensagem, e este responde executando o comportamento pertinente e utilizando, possivelmente, valores e referências também armazenadas pelo objeto. A habilidade de definir comportamentos como parte do esquema do banco de dados, em vez de parte da aplicação, está contida no paradigma da Orientação a Objetos.

2.2 Representação Ativa de Objetos

Verifica-se o uso muito comum da representação ativa OO nos casos de apresentação de objetos, sendo que os mesmos podem desenhar-se a si mesmos, diferentemente a cada momento, dependendo da situação, ou seja, dinamicamente. Já na representação por aspecto, temos uma representação estática definida pela classe de aspectos. Pode-se verificar as diferenças na Tabela 1.

Representação Ativa OO	Representação por Aspectos
Dinâmica - os objetos podem se desenhar diferentemente a cada vez.	Estática – definida pela classe de aspectos.
Definida no banco de dados.	Definida no programa da aplicação.
Pode ser definida e alterada pelo cliente.	Só pode ser alterada por quem fornece o "software".
Pode ser influenciada por combinação de atributos.	Indexada por um atributo de código de simples aspecto.
Pode ser influenciada por atributos de outros objetos referenciados.	Cada aspecto é representado isoladamente.
Pode se adaptar a influências externas (ex: escala do mapa requerida).	Não é adaptativa.

Tabela 1- Comparação entre Representação Ativa e por Aspectos

A representação ativa é usada extensivamente na produção de Cartas Náuticas Eletrônicas para prover retorno ao operador da exatidão dos aspectos da codificação, atributos e relacionamentos entre objetos.

2.3 Processamento de Imagens Digitais

A idéia de utilizar a manipulação de imagens digitais surgiu com a intenção de agilizar as tarefas de análise das informações contidas, até então, em mapas cujo manuseio e atualização muitas vezes tornavam-se problemáticos. Surgem, assim, os Sistemas de Informação Geográfica que, segundo CÂMARA: "S*ão sistemas de informações construídos especialmente para armazenar, analisar e manipular dados geográficos, ou seja, dados que representam objetos e fenômenos, em que a localização geográfica é uma característica inerente e indispensável para tratá-los. Dados geográficos são coletados a partir de diversas fontes e armazenados, via de regra, nos chamados banco de dados geográficos*" [8].

2.4 O Banco de Dados de Objetos

Uma carta é um modelo de parte da superfície da terra, apresentada como uma ilustração gráfica. A produção de Cartas e ECDIS, para aplicações específicas, levam ao detalhamento de certos aspectos e a omissão de outros. No passado, a cartografia hidrográfica era feita capturando-se e compilando-se os dados necessários para produzir uma carta em particular. A ineficiência dos métodos tradicionais levou à utilização de meios eletrônicos para armazenar o modelo geoespacial do mundo, que pode ser atualizado. Partindo-se deste banco de dados, é possível produzir-se uma gama de produtos em diferentes escalas e diferentes especificações WOODSFORD [9]. Bancos de dados relacionais tradicionais não foram projetados para armazenar modelos de dados complexos e grandes volumes de dados envolvendo a construção e a integridade do mapeamento da geografia do mundo real. Nem, tão pouco, seria fácil produzir uma grande quantidade de produtos cartográficos com as usuais facilidades de representação estáticas dos tradicionais GIS e programas de mapeamento. Assim, os bancos de dados geoespaciais OO e suas representações associadas promovem um novo mundo, onde os objetos possuem comportamento próprio e podem ser armazenados de forma independente .

2.5 Banco de Dados com Características Espaciais

Segundo ORACLEESPACIAL [10], um SGBD (Sistema de Gerenciamento de Banco de Dados) com características espaciais possui um conjunto de funções e procedimentos que permitem que dados espaciais sejam armazenados, acessados e analisados eficientemente. Estes dados representam em sua essência características de localização de objetos reais ou conceituais, objetos estes que estão correlacionados com o espaço no qual eles existem.

Em geral pode-se ter uma SGBD relacional com características espaciais, como é o caso do ORACLE-ESPACIAL. Este SGBD provê um esquema SQL e funções que facilitam o armazenamento, a recuperação, a atualização e a consulta de coleções de feições espaciais no banco de dados. É constituído dos seguintes componentes:

- Um esquema (MDSYS) responsável pelo armazenamento, sintaxe e semântica dos tipos de dados geométricos suportados;
- Um mecanismo de indexação espacial;
- Um conjunto de operadores e funções que promovem consultas na área de interesse, bem como outras operações de análise espacial; e
- Utilitários administrativos.

Um componente espacial de uma feição espacial é a representação geométrica de sua forma em alguma coordenada espacial, sendo referenciada como sendo sua geometria.

2.5.1 O Modelo Objeto Relacional (ORACLE-ESPACIAL)

Um SGBD com suporte a objetos espaciais deve possuir características que permitam a representação das geometrias desses objetos. Se o mesmo é constituído de forma relacional, como é o caso do ORACLE, ele deve possuir um modelo que evidencie e especifique as feições geoespaciais. Surge, então, a definição do Modelo Objeto Relacional, cujos benefícios incluem:

- Suporte a um grande número de tipos de geometrias, incluindo áreas, círculos, polígonos compostos, linhas, etc;
- Fácil criação e manutenção de índices e execução de consultas espaciais;
- Manutenção dos índices criados por um servidor do banco de dados;
- Geometrias modeladas em uma única linha e coluna; e
- Performance otimizada.

2.5.2 Introdução aos Dados Espaciais (ORACLE-ESPACIAL)

O ORACLE–ESPACIAL foi projetado de forma a prover uma utilização voltada para informações espaciais a serem utilizadas em aplicações em GIS. Uma vez armazenadas, estas informações devem ser facilmente recuperadas e manipuladas, tais como quaisquer outras informações armazenadas no banco.

Um típico exemplo de dado espacial pode ser visto em uma carta náutica. Esta possui objetos bi-dimensionais que contêm pontos, linhas e polígonos que podem representar pontes, linhas de costa e inúmeros outros elementos terrestres e marítimos. A carta náutica é uma visualização da informação geográfica onde a localização dos elementos que existem na superfície da terra e que compõem a carta são projetados sobre a mesma, preservando suas posições relativas e distâncias.

Os dados que indicam as localizações destes objetos (latitude e longitude, ou altura e profundidade) são considerados dados espaciais. Estes dados são utilizados para projetar a localização dos objetos em um plano bi-dimensional e, por intermédio de um GIS, pode-se armazenar e recuperar esses dados, provendo um meio eficaz de se manter as proporções e posicionamentos relativos.

2.5.3 Tipos Geométricos (ORACLE-ESPACIAL)

No modo vetorial de representação da realidade, a geometria de um objeto é uma seqüência ordenada de pontos ou vértices que são conectados por segmentos de retas ou arcos circulares. A semântica da geometria é determinada pelo seu tipo. No caso do ORACLE–ESPACIAL, pode-se ter vários tipos, chamados primitivos, e compostos geométricos, construídos a partir dos próprios tipos primitivos, permitindo a composição de uma coleção. Como exemplo:

- Pontos e conjunto de pontos;
- Linhas (cadeia de linhas);
- Polígonos de n-pontos;
- Linhas compostas por arcos;
- Polígonos compostos por arcos;
- Polígonos compostos;
- Linhas compostas;
- Círculos; e
- Retângulos otimizados.

Observa-se estes elementos na Figura 3.

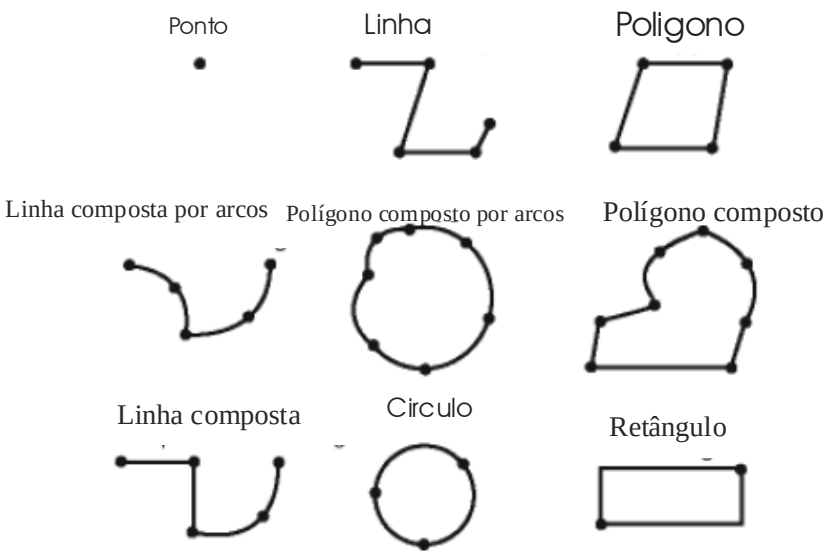


Figura 3 - Tipos Geométricos
Extraída de [10] ORACLEESPACIAL

2.5.4 Modelo de Dados

O modelo de dados espaciais em ORACLEESPACIAL [10] é composto por uma estrutura hierárquica de elementos, de geometrias e camadas, correspondendo à representação do dado espacial. Camadas são compostas por geometrias que são constituídas por elementos. Um ponto pode representar um local construído, uma cadeia de linhas pode representar uma rodovia, um polígono por sua vez pode representar um quarteirão. Desta forma, torna-se necessária uma melhor descrição para as entidades que compõem o modelo de dados.

2.5.4.1 Elemento

Elemento é a estrutura básica de composição de uma geometria, sendo que pontos, cadeias de linhas e polígonos são os tipos de elementos espaciais suportados. Pode-se ter, por exemplo, limites de países representados por polígonos, sendo cada coordenada de um elemento armazenada em um par X,Y. O anel exterior e o anel interior de um polígono com buracos são considerados como dois elementos distintos que unidos, formam um polígono complexo.

Dados referentes a pontos consistem em uma coordenada. Dados referentes a linhas consistem de duas coordenadas que podem representar um segmento de linha de um elemento. Dados de polígonos consistem de pares de valores de coordenadas, sendo uma par para cada vértice do polígono, que devem ser definidos em ordem ao redor do mesmo (orientados no sentido contrário dos ponteiros do relógio para o anel exterior de polígonos complexos, e sentido dos ponteiros do relógio para o anel interior).

2.5.4.2 Geometria

A geometria é a representação da característica espacial, modelada como um conjunto ordenado de elementos primitivos. Pode ser composta por um elemento simples, sendo uma instância de um tipo primitivo suportado, ou uma coleção homogênea ou heterogênea de elementos. A utilização de polígonos múltiplos pode ser percebida na representação de um conjunto de ilhas apresentadas como coleção homogênea. Em uma coleção heterogênea existem tipos diferentes tais como, ponto e polígono.

Pode-se ainda exemplificar como geometria, uma área construída em uma cidade, onde sua representação utilizaria polígonos, com buracos que corresponderiam a áreas não edificadas.

2.5.4.3 Camadas

Uma camada é na realidade uma coleção de geometrias que possuem o mesmo conjunto de atributos, assim, em um GIS pode-se ter uma camada que inclua as características topográficas, e em outra as características populacionais. Em GIS navais, pode-se ter uma camada que possua os elementos terrestres, uma segunda camada com elementos marinhos, uma terceira englobando as linhas de profundidade, e finalmente, outra incluindo os perigos a navegação, tendo-se assim, quantas camadas forem necessárias.

2.5.4.4 Sistemas de Coordenadas

Um sistema de coordenadas é definido em ORACLEESPACIAL [10], como sendo a atribuição de coordenadas a uma localização, onde estas coordenadas possuem entre si

relacionamentos, permitindo a interpretação deste conjunto como sendo a representação da posição no espaço do mundo real.

Qualquer dado espacial possui sempre um sistema de coordenadas a ele associado, que deve ser georreferenciado[1] ou cartesiano[2]. O dado georreferenciado possui uma unidade de medida padrão associada, no entanto pode assumir uma unidade diferente caso a situação assim o exija, por exemplo: sendo metros a unidade padrão, pode-se ter como resultado de retorno de uma função um valor em milhas.

Dados espaciais podem ser associados a sistemas de coordenadas cartesianas, geodésicas (geográficas), projetadas ou locais.

Desta forma, pode-se ter:

- Coordenadas Cartesianas, são coordenadas que medem a posição de um ponto a uma origem definida sobre eixos que são perpendiculares entre si e que representam um espaço bi-dimensional ou tri-dimensional;
- Coordenadas Geodésicas, também chamadas de coordenadas geográficas, são coordenadas angulares (longitude e latitude) intimamente relacionadas a coordenadas esféricas polares, sendo relativas a um datum[3] geodésico terrestre ;
- Coordenadas Projetadas, são coordenadas cartesianas planares, que resultam da aplicação de um mapeamento matemático de um ponto da superfície terrestre em um plano. Existem, para tanto, diversos tipos de mapeamentos matemáticos, sendo cada um utilizado segundo uma necessidade específica; e
- Coordenadas Locais, são coordenadas cartesianas que não são georreferenciadas, ou seja, não estão correlacionadas a pontos da superfície da Terra.

2.5.4.5 Tolerância

[1] Relacionado a uma representação específica.da terra.
[2] Não relacionado a uma representação específica da terra.
[3] Datum, segundo INPE [17], é um modelo matemático que substitui a Terra real nas aplicações cartográficas, utilizando uma superfície de referência posicionada em determinada localização.

A tolerância é utilizada para associar o nível de precisão aos dados espaciais. Representa o quanto pode-se afastar dois pontos, de modo que a distância entre eles continue sendo considerada a mesma. O valor da tolerância deve ser sempre positivo e maior do que zero. Seu significado depende se os dados espaciais estão ou não associados a um sistema de coordenadas geodésico.

- Para dados geodésicos, identificados por coordenadas de longitude e latitude, o valor de tolerância é dado em metros, por exemplo: um valor de tolerância de cem indica uma tolerância de cem metros; e
- Para dados não geodésicos, o valor da tolerância é um número de unidades que estão associadas ao sistema de coordenadas relacionado ao dado, por exemplo: se a unidade de medida for milha, uma tolerância de 0.005 (1/200 de milha) significa aproximadamente cento e cinco pés.

2.5.5 O Modelo de Consultas

Segundo ORACLEESPACIAL [10], tem-se dois passos para a realização de consultas espaciais, sendo que o conjunto resposta para a consulta é obtido após a execução de ambos os passos.

Estes passos são chamados de operações de filtragem primária e secundária, possuindo características particulares.

- A filtragem primária permite uma rápida seleção de registros candidatos, que são passados para a filtragem secundária, efetuando comparações de aproximações geométricas e, em conseqüência, reduzindo a complexidade computacional. É considerado um filtro de baixo custo, retornando um conjunto de elementos bem maior do que o da solução da consulta; e
- A filtragem secundária aplica às geometrias selecionadas pelo filtro primário os procedimentos computacionais necessários à realização da consulta. Sua utilização é computacionalmente dispendiosa, porém, é aplicada apenas aos registros selecionados na filtragem inicial, e não a todo o conjunto de dados, reduzindo consideravelmente o custo computacional total.

Pode-se observar o relacionamento entre os filtros primário e secundário na Figura 4.

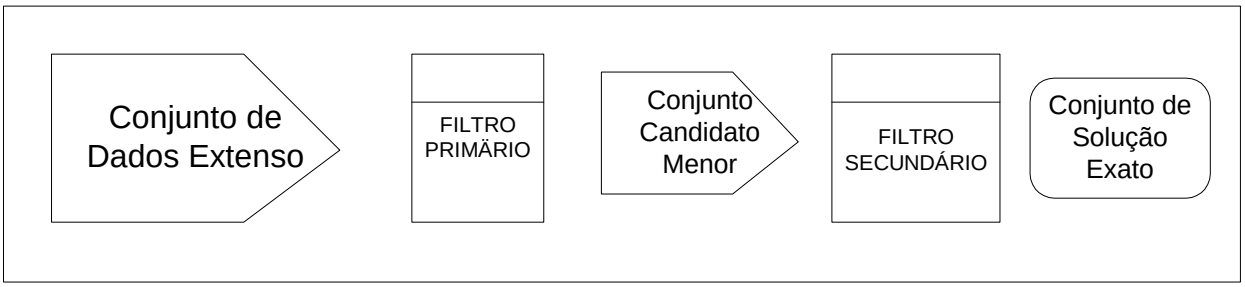


Figura 4 – Modelo de Consulta (ORACLE – ESPACIAL)
Extraída de [10] ORACLEESPACIAL

Nem sempre a utilização dos dois filtros se faz necessária. Dependendo do caso, apenas o filtro primário é suficiente, por exemplo: numa consulta para apresentar os elementos de um mapa, que possuam interação com um retângulo que representa limites visíveis, um filtro primário poderia retornar, rapidamente, um conjunto de elementos superior aos elementos intrínsecos à área em questão. Aplicando-se rotinas de "clipping"[4] sobre a área de interesse, pode-se obter a visualização desejada, sem a necessidade de aplicação do filtro secundário.

2.5.6 Indexação de Dados Espaciais

A indexação espacial provê um mecanismo de limitação de buscas como toda indexação, porém, neste caso, tem-se critérios baseados em interseção e relações de conter e estar contido. Deste modo, um índice espacial é necessário quando:

- Deseja-se achar objetos que interagem com um dado ponto ou uma área de interesse (busca de janela); e
- Deseja-se achar pares de objetos contidos em duas áreas que interagem espacialmente entre si (junção espacial).

[4] Clipping é o efeito pelo qual há uma porção visível de um objeto em uma janela de exibição ou de interesse, e porções invisíveis deste objeto fora da janela.

No caso do ORACLE-ESPACIAL [10], tem-se a utilização de indexação via “R-tree” (por padrão), “Quadtree” ou ambas. Cada tipo de indexação torna-se mais apropriada, dependendo da situação .

Utilização de indexação com “R-tree”:

- As aproximações das geometrias não podem ser reduzidas, pois, existe a utilização de retângulos mínimos de adjacências;
- A criação de índices é facilitada;
- Se a aplicação utilizar consultas com vizinhanças próximas, elas tornam-se mais rápidas;
- Se existirem atividades intensas de atualização espacial, a indexação por “R-tree” pode não ser uma boa escolha; e
- Podem ser indexadas até quatro dimensões.

Utilização de indexação com “Quadtree”:

- As aproximações das geometrias podem ser reduzidas, ou seja, podem ter um ajuste mais preciso, alterando-se o nível de fragmentação, bem como o número de fragmentos;
- O ajuste se torna mais complexo e a escolha de seus parâmetros pode afetar sua performance de modo significativo;
- Se a aplicação utilizar consultas com vizinhanças próximas, elas tornam-se mais lentas; e
- Atividades intensas de atualização espacial não afetam o desempenho da indexação por “Quadtree”.

2.5.6.1 Indexação por “R-tree”

Pode-se ter até quatro dimensões indexadas por uma “R-tree” de índices espaciais, sendo que cada geometria é aproximada para um retângulo e o mesmo deve envolver de forma mínima toda a geometria em questão. Este retângulo recebe o nome de Retângulo de Limite Mínimo (MBR), ver Figura 5

Figura 5 – MRB Envolvendo uma Geometria
Extraída de [10] ORACLEESPACIAL

Para uma camada de geometrias uma “R-tree” de índices é constituída de uma seqüência hierárquica dos índices dos MBRs das geometrias contidas nesta camada.

Observando-se a Figura 6 deve-se perceber que:

- *1* a *9* são geometrias em uma camada;
- *a*, *b*, *c* e *d* são os nós folha de índices da “R-tree”. Eles contêm os retângulos mínimos que envolvem os retângulos de geometrias;
- *A* contém o MBR de *a* e *b*, e *B* contém o MBR de *c* e *d*; e
- O nó raiz contém o MBR de *A* e o MBR de *B*.

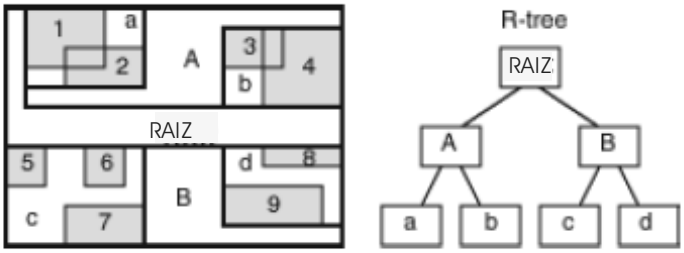


Figura 6 - “R-tree” Hierárquica de Índices sobre MBRs
Extraída de [10] ORACLEESPACIAL

2.5.6.1.1 Qualidade de uma “R-tree”

A utilização de muitas operações de inserção e exclusão de índices em uma “R-tree” pode afetar de maneira significativa o desempenho das consultas. Pode-se ter, então, uma degradação da qualidade da estrutura na qual ela se baseia.

De forma geral, o desempenho para consultas em uma “R-tree” de índices é intrinsecamente proporcional às áreas e perímetros cobertos pelas geometrias cujos índices pertencem à árvore. Em geral, a área representada pelo nível mais alto da árvore usualmente sofre alterações que implicam em degradação da eficiência das consultas. Para evitar este tipo de efeito, deve-se refazer os índices da árvore, procurando sempre manter a integridade da mesma.

2.5.6.2 Indexação por “Quadtree”

Em um esquema de indexação linear por “Quadtree”, o espaço de uma camada no qual todos os objetos geométricos estão contidos é submetido a um processo chamado “mosaico”, que define, de forma exclusiva e exaustiva, uma cobertura composta por fragmentos (retângulos) para toda a geometria existente. Este processo é efetuado visualizando-se o espaço como um retângulo, que deve ser decomposto em retângulos menores. Esta divisão é feita tomando-se a metade das dimensões das coordenadas, formando quatro retângulos que, unidos, formam o retângulo de nível superior. Este processo deve se repetir até que algum critério de parada seja atingido, por exemplo: o tamanho do retângulo de decomposição ou o numero máximo de retângulos responsáveis por cobrir uma geometria é atingido.

Pode-se, ainda, definir um tamanho fixo ou variável para os fragmentos que cobrem uma geometria. Retângulos de tamanho fixo são controlados pela resolução. O processo termina quando o espaço em questão é decomposto, em um número específico de vezes, em fragmentos que possuem tamanho e formato fixos. No caso de tamanho variável de fragmentos, a divisão é controlada pelo número de fragmentos, ou seja, o processo termina quando este número é atingido para a cobertura da geometria dada.

Os retângulos de um determinado nível da árvore podem ser ordenados de forma a compor uma curva de preenchimento, como visto na Figura 7

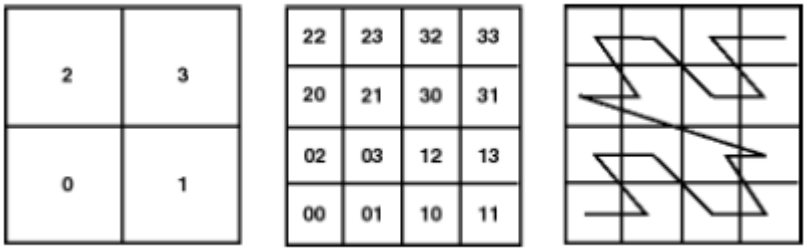


Figura 7 – Decomposição por “Quadtree”
Extraída de [10] ORACLEESPACIAL

2.5.6.2.1 Indexação Fixa

A indexação espacial fixa utiliza fragmentos de igual tamanho para cobrir a geometria de interesse. Como conseqüência, todos os índices possuem o mesmo tamanho, provendo um excelente rendimento em operações que envolvam comparações. Este rendimento é afetado diretamente pela variação do tamanho das geometrias existentes em uma camada. Se a escolha do tamanho dos fragmentos for apropriada às geometrias de pequeno tamanho, para as de tamanho maior o número de fragmentos ficará excessivamente elevado. Por outro lado, ao escolher-se fragmentos de tamanho apropriado para geometrias de áreas maiores, esses mesmos fragmentos não se aplicarão corretamente às áreas menores.

Na Figura 8 pode-se observar que os fragmentos de tamanho reduzido são aplicados, corretamente, às pequenas geometrias. Porém, um número excessivo de fragmentos é exigido para cobrir as grandes geometrias.

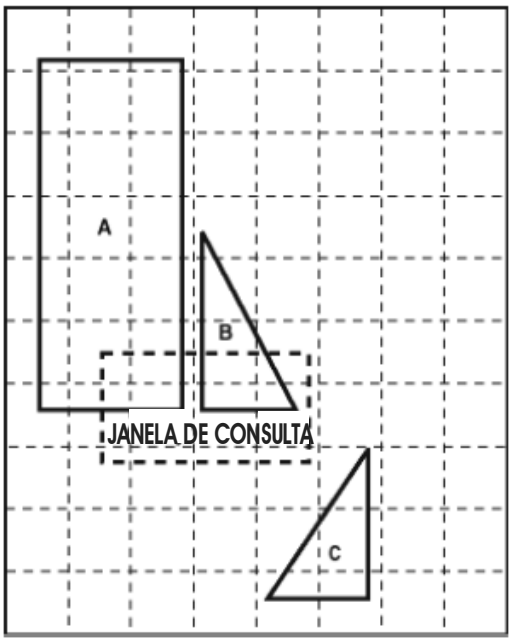


Figura 8 – “Mosaico” com Tamanho Fixo e Fragmentos Pequenos
Extraída de [10] ORACLEESPACIAL

Na Figura 9 pode-se observar que os fragmentos de tamanho maior se aplicam bem às grandes geometrias, exigindo um número pequeno de retângulos para envolvê-las. Entretanto, a seletividade se torna bem reduzida.

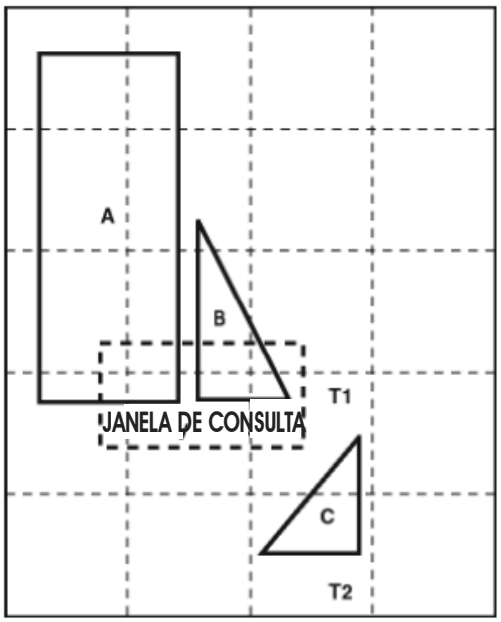


Figura 9 – “Mosaico” com Tamanho Fixo e Fragmentos Grandes
Extraída de [10] ORACLEESPACIAL

2.5.7 Relações Espaciais

As relações espaciais são formadas sobre localizações geométricas, sendo baseadas em topologias e distâncias. Como exemplo, pode-se ter uma fronteira de uma área, constituída de um conjunto de curvas que separam a área do resto do espaço. O interior de uma área é constituído por todos os pontos pertencentes a esta área que não se localizam sobre suas fronteiras. Tomando-se este preceito, duas áreas são ditas adjacentes quando elas compartilham parte de suas fronteiras, mas não compartilham nenhum ponto sobre seu interior.

Define-se distância entre dois objetos espaciais como sendo a menor distância entre quaisquer dois pontos pertencentes a estes objetos. Desta forma, a distância entre eles estará dentro de uma distância fornecida, se esta for maior do que a distância entre os objetos em questão.

Segundo ORACLEESPACIAL [10], pode-se ter operadores que determinam relações espaciais, implementando modelos de interseção entre pontos, linhas e polígonos. Cada objeto espacial possuirá:

- Uma fronteira, constituída de pontos ou linhas que separam seu interior de seu exterior, por exemplo: a fronteira de uma linha são seus pontos terminais. As fronteiras de um polígono são compostas por linhas que descrevem seu perímetro;
- Um interior, constituído de pontos que pertencem ao objeto, mas não à sua fronteira; e
- Um exterior, constituído de pontos que não estão no objeto.

Segundo ORACLEESPACIAL [10], pode-se identificar relacionamentos topológicos, tais como:

- Disjunção – os limites e os interiores não se interceptam;
- Toque - os limites se interceptam, mas não os interiores;
- Sobreposição por Disjunção – o interior de um dos objetos intercepta os limites e o interior de outro, mas não os limites dos dois. Este relacionamento pode ocorrer

em situações tais como: uma linha se origina fora de um polígono e termina dentro dele;

- Sobreposição por Interseção – os limites e os interiores dos dois objetos se interceptam;
- Igualdade – os dois objetos possuem os mesmos limites e interiores;
- Contém – o interior e os limites de um objeto estão completamente contidos no interior de outro objeto;
- Cobre – o interior de um objeto está completamente contido no interior de outro objeto e seus limites se interceptam;
- Dentro – é o oposto de Contém;
- Coberto por – é o oposto de Cobre;
- Sobre – o interior e os limites de um objeto estão sobre os limites de outro objeto e o segundo objeto cobre o primeiro. Este relacionamento pode ocorrer, por exemplo: no caso de uma linha estar situada nos limites do polígono; e
- Qualquer Interação – os objetos não são Disjuntos.

Os relacionamentos topológicos são apresentados na Figura 10

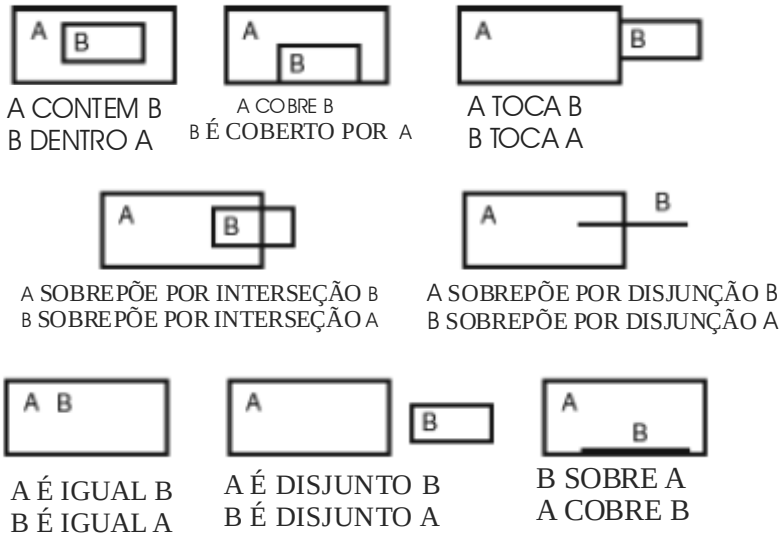


Figura 10 – Relacionamentos Topológicos.
Extraída de [10] ORACLEESPACIAL

2.6 Características dos Sistemas de Informações Geográficas

2.6.1 Definição

Segundo MOREIRA [11], qualquer SIG apresenta duas características principais:

- Permite inserir e integrar, numa única base de dados (banco de dados), informações espaciais provenientes de diversas fontes, tais como: cartográficas, imagens de satélites, dados censitários, dados de cadastro rural e urbano, dados de rede e de MNT (Modelo Numérico de Terreno). CAMARA [8] considera que o MNT representa uma grandeza que varia continuamente no espaço, usualmente associado à altimetria, podendo ser também utilizado para modelar unidades geológicas, como: teor de minerais, propriedades do solo ou subsolo, exemplo: aeromagnetismo. Conforme BURROUGH [12], podemos citar como Modelos Numéricos de terreno:
  - Armazenamento de dados de altimetria para gerar mapas topográficos;
  - Análise de corte-aterro para projeto de estradas e barragens;
  - Cômputo de mapas de declividade e exposição para apoio à análise de geomorfologia e erodibilidade;
  - Análise de variáveis geofísicas e geoquímicas; e
  - Apresentação tridimensional (em combinação com outras variáveis).

- Oferece mecanismos para combinar várias informações, através de algoritmos de manipulação e análise, bem como de consulta, recuperação, visualização e plotagem do conteúdo daquela base de dados georreferenciados.

De modo geral, conforme INPE [13], qualquer SIG é capaz de:

- Representar graficamente informações de natureza espacial, associando a estes gráficos informações alfanuméricas tradicionais;
- Representar informações gráficas sob a forma de vetores (pontos, linhas e polígonos) e/ou imagens digitais [5]"raster";

[5] Matrizes de pixels

- Realizar operações de aritmética de polígonos, tais como: união, interseção e diferença. Gerar polígonos paralelos ao redor de elementos ponto, linha e polígono;
- Limitar o acesso e controlar a entrada de dados através de um modelo de dados previamente construído;
- Oferecer recursos para a visualização dos dados geográficos na tela do computador, utilizando, para tal, uma variedade de cores;
- Interagir com o usuário através de uma interface amigável, geralmente gráfica;
- Recuperar de forma ágil as informações geográficas, como o uso de algoritmos de indexação espacial;
- Possibilitar a importação e a exportação de dados de/para outros sistemas semelhantes ou para outros “softwares” gráficos;
- Oferecer recursos para a entrada e a manutenção de dados, utilizando equipamentos como “mouse”, mesa digitalizadora e “scanner”;
- Oferecer recursos para a composição de saídas e geração de resultados sob a forma de mapas, gráficos e tabelas, para uma variedade de dispositivos, como: impressoras e “plotters”; e
- Oferecer recursos para o desenvolvimento de aplicações específicas, de acordo com as necessidades do usuário.

Na Figura 11 observa-se a estrutura de um SIG.

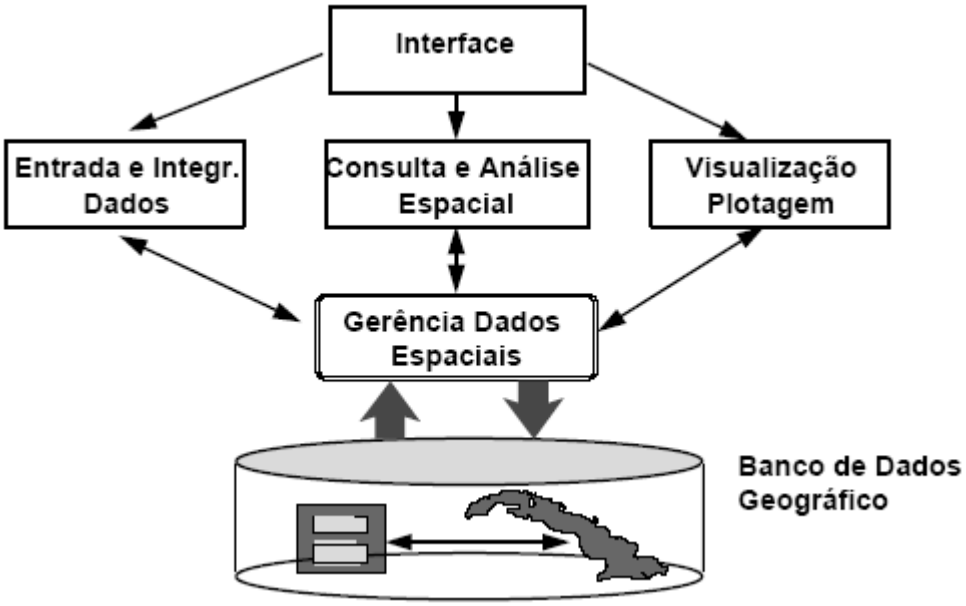


Figura 11 - Estrutura Geral de um Sistema de Informação Geográfica

Extraída do documento cap1-repres.pdf
www.dpi.inpe.br/gilberto/livro/bdados/cap1-repres.pdf
acessado em novembro de 2004

2.6.2 SIG Orientado a Objetos

Baseia-se no armazenamento de dados geográficos utilizando objetos. Esta função é realizada por um SGBD (Sistema de Gerenciamento de Banco de Dados) Orientado a Objetos, sendo toda a operação do SIG baseada em um modelo de dados Orientado a Objetos, incluindo características gráficas, alfanuméricas e de aspectos do comportamento do objeto. Na prática, a implementação de um SIG OO utiliza uma arquitetura cliente-servidor, onde o tráfego entre os núcleos do cliente e do servidor constitui-se de objetos. A comunicação entre o servidor e o banco de dados Orientado a Objetos pode ser implementada utilizando-se padrões como ODTP (“Object Data Transfer Protocol”) ou CORBA (“Common Object Request Broker Archiecture”), sendo que, usualmente, a escolha é por uma implementação proprietária, com o núcleo servidor fortemente integrado ao gerenciador Orientado a Objetos.

Uma maior semelhança com o formato existente em sistemas abertos é provida com a utilização da filosofia de Orientação a Objetos nesta arquitetura, pois, todos os esforços são destinados a estabelecer a interoperabilidade entre SIGs, baseados na padronização dos objetos.

Observa-se na Figura 12 o esquema de um SIG Orientado a Objetos.

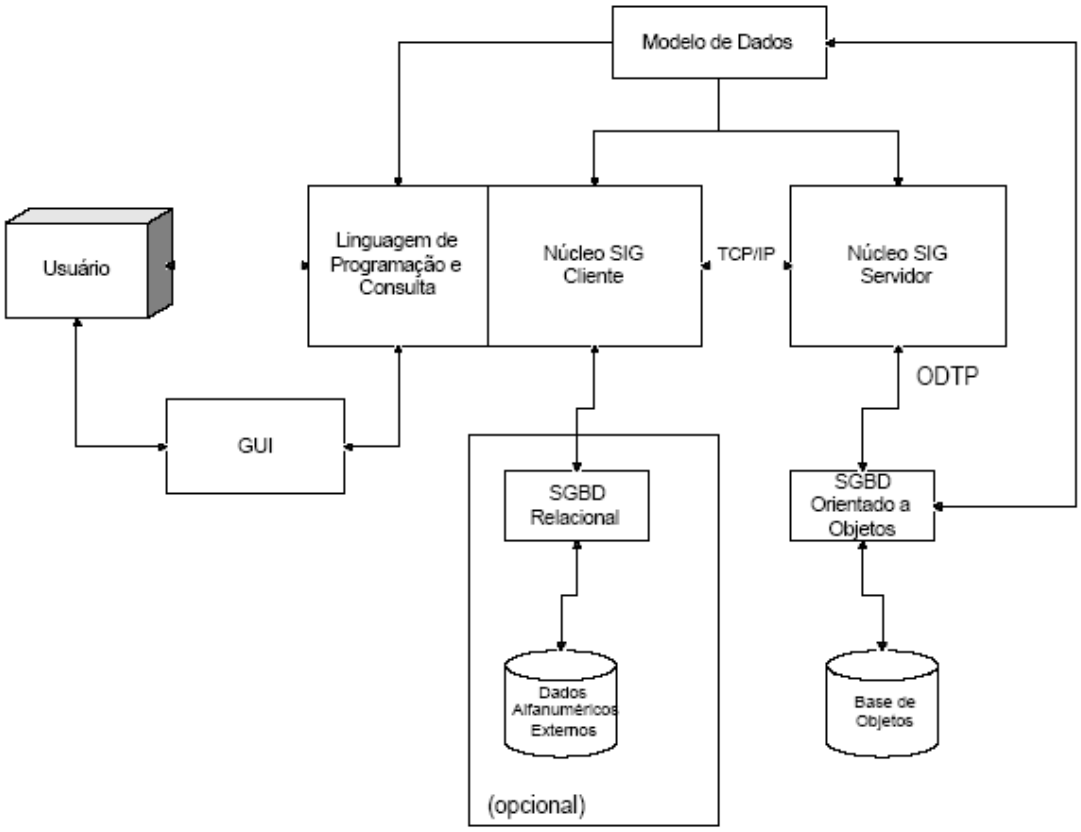


Figura 12 – SIG Orientado a Objetos

2.6.3 Características dos SIGs Marítimos (ECDIS)

Uma definição resumida para ECDIS, seria: "*o termo ECDIS refere-se a um sistema de informações com "backup" e atualizações de cartas segundo uma regulamentação V/20 (convenção SOLAS [14]), com funções de apresentação de informações selecionadas de uma carta de sistema eletrônico de navegação SENC, com posicionamento, recebendo informações de sensores de navegação para auxiliar no planejamento de rotas marítimas, monitoração e exibição de informações complementares caso sejam necessárias"*, IMO [15]. Torna-se necessária uma definição mais formal para SENC: "*a base de dados gerada pela transformação das cartas náuticas eletrônicas pelo ECDIS, para serem usadas pelo mesmo, visando à geração de apresentações e outras funções de navegação*", IMO [15]. Neste sistema, a apresentação está baseada nas SENCs e não nas cartas náuticas eletrônicas ENC.

Na condução de uma embarcação existem, geralmente, três preocupações principais:

- Navegação, os responsáveis pela navegação processam informações provenientes de diferentes fontes, sejam elas de satélites ou de outros sistemas: plotando as informações em uma carta de papel; determinando se a embarcação encontra-se em posição segura; projetando a posição à frente e contingências futuras. Este planejamento é a função mais importante da navegação, dependendo, diretamente, da qualidade de plotagem e da precisão dos sensores;
- Evitar a colisão, os responsáveis pela condução da embarcação estão sempre calculando o ponto de maior aproximação dos contatos existentes, procurando, desta maneira, calcular sua rota de modo a evitar uma colisão; e
- Gerenciamento da embarcação, uma ponte[6] com funções integradas permite respostas mais eficientes em situações de risco, propiciando um aumento de tempo livre para ser empregado no gerenciamento e acompanhamento de contatos na tarefa de evitar colisões.

A utilização de ECDIS baseados em PCs, segundo KAY [16], proporciona:

- Sua instalação sem interrupção da rotina da embarcação, permitindo um treinamento no próprio navio;
- As atualizações do sistema são facilmente aplicadas sem maiores problemas;
- Com a retirada de operação da embarcação, o sistema pode ser removido;
- Com a possibilidade de utilização de sistemas operacionais mais conhecidos, os tempos de aprendizado são reduzidos;
- Cartas adicionais podem ser enviadas por e-mail para a embarcação;
- A manutenção do “software” é mais rápida e sua atualização pode ser efetuada no próprio navio, sendo enviada por e-mail; e
- Monitores repetidores podem ser dispostos em locais estratégicos, segundo determinadas situações, por exemplo: em cada ponte em situações de manobra.

[6] Onde é efetuado o comando da embarcação.

2.6.4 O Modelo de Dados ENC

O termo ENC (Carta Náutica Eletrônica) surgiu com a necessidade de se possuir uma base de dados de informações que pudesse ser utilizada em SIGs navais (ECDIS). O padrão para ENC utilizado em ECDIS foi definido pela Organização Hidrográfica Internacional (IHO), publicação especial 57 (S57) IHB 1996 [17]. A edição 3.0 do padrão S57 foi publicada em setembro de 1996, constituída de um modelo de dados e um catálogo de objetos e atributos, com um formato de transferência. Pode-se resumir seu modelo como:

- Objetos espaciais, podem ser nós, bordas ou faces, e descrevem informações sobre posição geográfica. Eles devem possuir uma estrutura topológica de conectividade e adjacência;
- Objetos feição, descrevem as características das entidades do mundo real (uma linha de costa, uma área de ancoradouro). Eles se referem a objetos espaciais que os localizam. Mais de um objeto feição pode referenciar um nó ou uma borda, compartilhando um conjunto comum de coordenadas geométricas (X,Y). Um contorno de profundidade e duas áreas de profundidade podem compartilhar um conjunto de coordenadas, que formam uma borda comum, se ambas forem adjacentes;
- Hierarquia de coleções de objetos, apenas os objetos de baixo nível terão informações sobre a posição geográfica, por exemplo: um esquema complexo de tráfego pode ser constituído por uma coleção de esquemas de tráfego mais simples, cada um composto de linhas de tráfego; e
- Atributos e relacionamentos, complexos o suficiente para se construir e manter atualizados os objetos capturados.

As cartas náuticas eletrônicas (ENC) podem ser do tipo rasterizada ou vetorizada, assumindo vários formatos existentes (S-57, BSB-”raster”, ARCS, etc) e níveis diferentes de conteúdo e atribuição. Visando o esclarecimento destas questões, deve-se elucidar as diferenças existentes entre estas classificações e suas características.

Para se ter uma carta náutica eletrônica “oficial” é necessário que a mesma siga os padrões especificados pela IHO para um produto S-57 ENC, sendo gerada por empresas de hidrografia autorizadas pelo governo dos paises que assinam a convenção. Só assim ela

poderá ser utilizada por um ECDIS, que segue as normas de padronização IMO (Organização Marítima Internacional) para este tipo de sistema.

Uma definição resumida para ENC seria: "*uma base de dados padronizada no conteúdo, estrutura e formato, formalizada para o uso em ECDIS, gerada por empresas hidrográficas autorizadas pelo Governo, contendo todas as informações de cartas necessárias para uma navegação segura, e informações suplementares que possam vir a serem consideradas úteis*", IMO [15].

Tendo em vista as definições apresentadas, deve-se ter em mente que as diversas SENCs devem ser atualizadas, a "priori", em função da navegação. Ter-se uma ENC atualizada não é suficiente, pois, é a empresa hidrográfica que determina quais informações são oferecidas pelas cartas náuticas eletrônicas ENC e atualizações correlatas, enquanto são os navegadores que decidem quais informações oriundas das SENCs devem ser apresentadas.

2.6.5 Topologia e Estrutura ENC

Bancos de dados espaciais usam métodos e referências para implementar, internamente, suporte para as estruturas topológicas que formam os objetos que representam o mundo real. Regras são constituídas e o banco de dados pode aplicá-las para criar as ligações e nós necessários . A topologia de um objeto é mantida ao longo de sua edição, prevenindo, assim, o risco de sobreposição de contornos e outros problemas conseqüentes da alteração dos objetos envolvidos. Pode-se ter polígonos formados por composição de linhas ou diretamente formados por um conjunto de faces (entidade atômica de área). A Figura 13 apresenta a formação de polígonos.

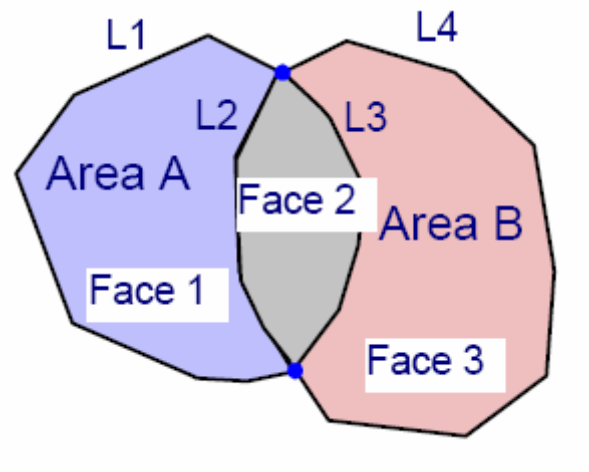


Figura 13 - Áreas Constituídas por Faces, com Ligações de Referência
Extraída de PGHARDY [18] .

O suporte a topologias é a característica mais importante da implementação do modelo de dados do padrão S-57. Com seus conceitos de objetos feição e objetos espaciais (nós, bordas e faces) o Banco de Dados Espacial fornece primitivas topológicas necessárias à manutenção automática das estruturas aplicadas aos dados armazenados nas projeções das cartas ou em Lat/Long no caso de estruturas em S-57.

2.6.6 Benefícios da Utilização de Orientação a Objetos na Produção de Dados em ECDIS

A descrição do paradigma OO, abordado nas seções anteriores, pode ser empregado na maioria dos dados de aplicações geoespaciais WOODSFORD [9]. Os itens apresentados a seguir demonstram as características que validam a força do emprego de banco de dados espaciais Orientados a Objetos e da utilização de mapeamento ativo na produção de Cartas Náuticas Eletrônicas:

- Modelagem de dados complexos, permitindo que o mundo real seja modelado em um banco de dados de objetos;
- Versatilidade de modelagem, utilizando-se um esquema de versões;
- Armazenamento e recuperação de dados, incluindo versões de objetos, propiciando uma eficiente atualização de dados;

- Captura, geração e manutenção eficientes de objetos das Cartas Náuticas Eletrônicas;
- Estruturas topológicas, incluindo criação automática e atualizações durante o uso, provendo estruturas de dados menos intrínsecas;
- Verificação de integridade de dados, propiciando a produção de dados de alta qualidade para serem utilizados em ECDIS;
- Métodos Orientados a Objetos generalizados simplificam a geração de produtos derivados, como exemplo: simplificações do uso da base de dados; e
- Representação Ativa apresenta os atributos dos objetos em formas destacadas nas cartas, permitindo ao operador verificar se a estrutura dos dados está de acordo com o padrão da Carta Náutica Eletrônica (ENC).

CAPÍTULO 3

3 O padrão S-57

3.1 O Modelo de Dados S-57

Modelo inicial – O padrão foi concebido com a intenção de permitir a transferência de dados que descrevem o mundo real. O mundo real é, na maioria das vezes, muito complexo. Para se ter uma representação prática, uma visão simplificada e altamente específica deve ser usada. Na realidade consegue-se esta representação modelando de forma reduzida a realidade.

As entidades que fazem parte do modelo devem ser aquelas que, no mundo real, são pertinentes à hidrografia. O modelo deve ser também geoespacial, implementando essas entidades como uma combinação de características chamadas feições e espaciais, definindo os objetos como: objetos do tipo feição e objetos do tipo espacial.

Um objeto é definido como um conjunto identificável de informações, contendo atributos e podendo se relacionar com outros objetos. Objetos do tipo feição contêm atributos que descrevem as características pertinentes ao seu modelo, sem conter nenhum atributo espacial[7]. Possuem, também, um relacionamento com um ou mais objetos espaciais, sendo que um objeto feição pode existir sem referenciar um objeto espacial. Porém, todo objeto espacial deve estar referenciado por um objeto feição.

Observa-se estas características na Figura 14.

[7] Informações sobre formato e posição da entidade no mundo real.

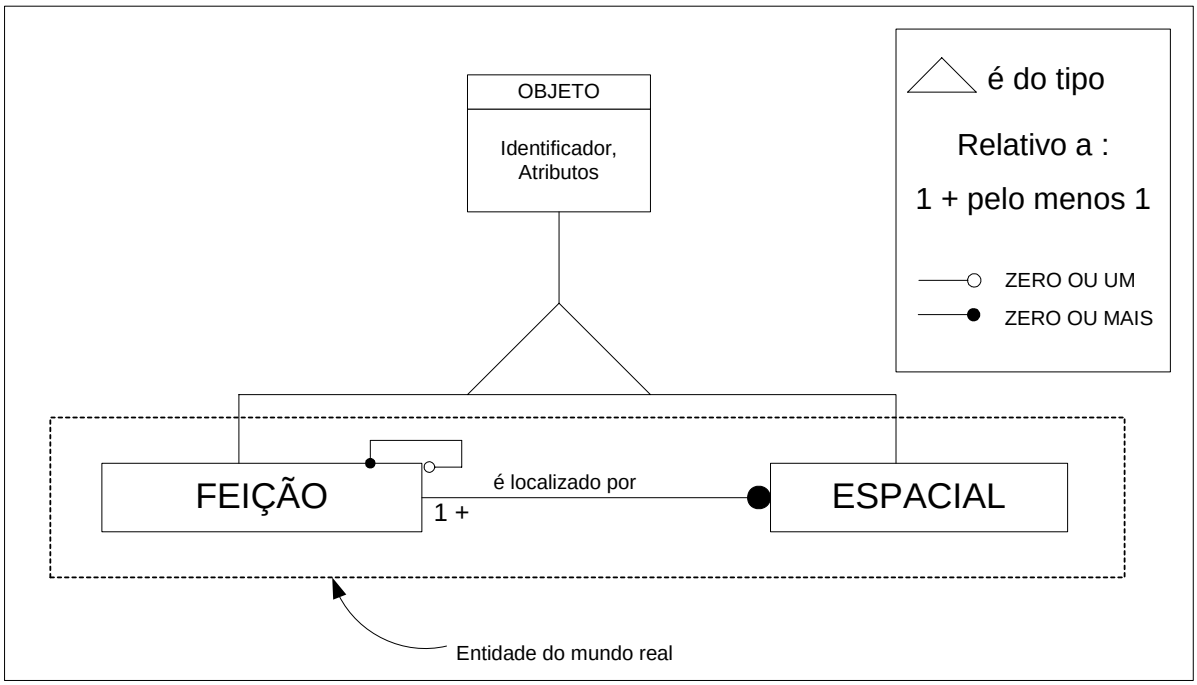


Figura 14 – Modelo S-57 (OBJETO)
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

3.2 Implementação do Modelo

3.2.1 Objetos Feição

- Meta-objeto - objeto feição que contém informações sobre outros objetos;
- Cartográfico - objeto feição que contém informações sobre a representação cartográfica das entidades do mundo real;
- Geo - objeto feição que contém características que descrevem os objetos do mundo real; e
- Coleção - objeto feição que descreve os relacionamentos entre outros objetos.

Os subtipos destes objetos feição estão definidos no catálogo de objetos da Organização Hidrográfica Internacional[8] .

[8] **http://www.iho.shom.fr/**, documento 31ApACH1.pdf.

3.2.2 Objetos Espaciais

As características espaciais das entidades do mundo real podem ser representadas de várias formas, como exemplo: vetorizada, rasterizada ou matricial. Assim sendo, estes objetos podem ser do tipo vetor, “raster” ou matriz. Segundo KIAN [19], a manipulação de informação espacial pode ser representada de maneira semelhante na forma de elementos rasterizados ou matriciais, ambos utilizando estruturas do tipo vetor, porém codificados de maneiras, até certo ponto, diferentes, dependendo das características dos dados envolvidos.

Pode-se observar na Figura 15 o modelo implementado para o tipo vetorial.

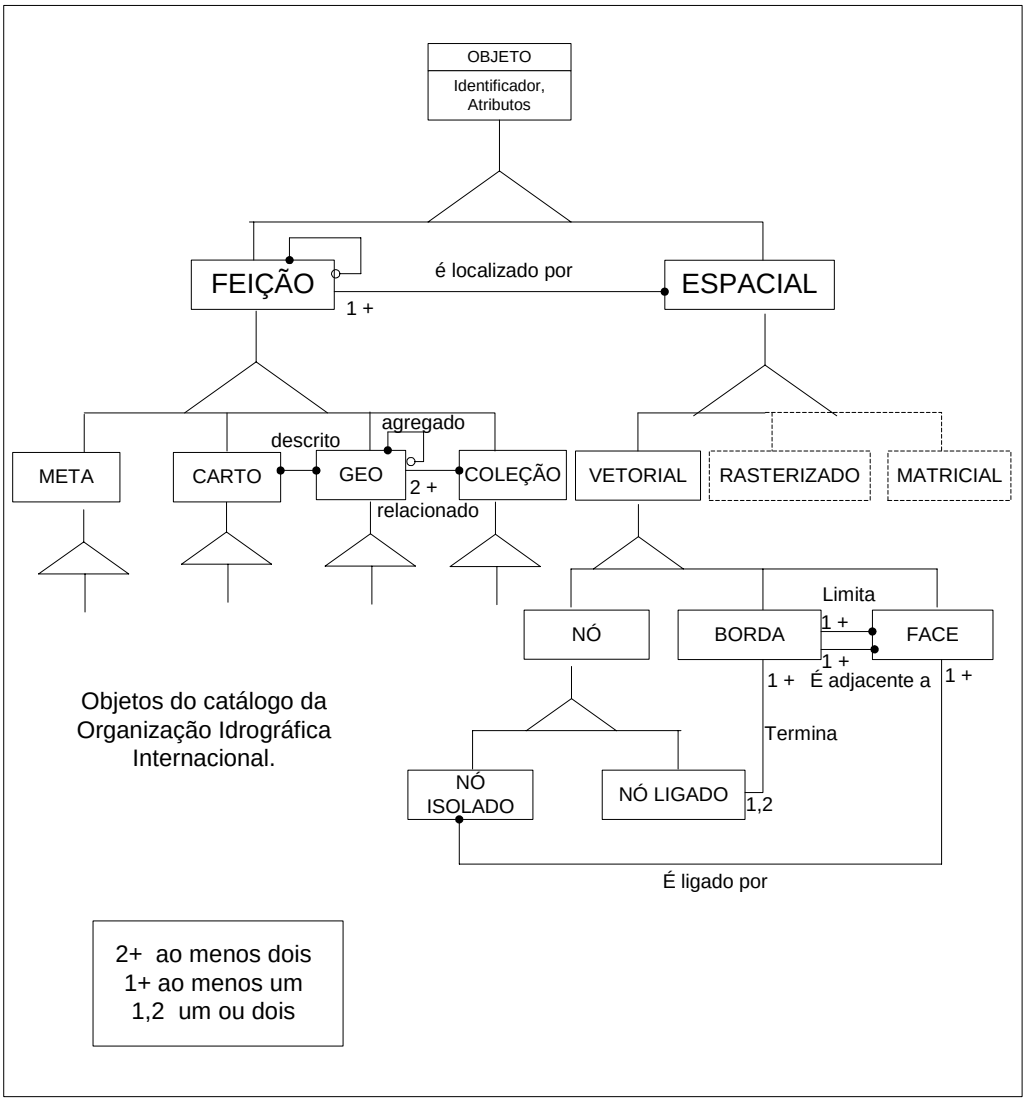


Figura 15 – Modelo Tipo Vetorial

Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

3.2.2.1 Modelo Vetorial

De forma simplificada, duas visões planas dimensionais da realidade são tomadas, objetos espaciais do tipo vetor podem ter zero, uma ou duas dimensões implementadas como nós, bordas e faces respectivamente. A terceira dimensão é expressa como um atributo do objeto. O modelo espacial é apresentado na Figura 16.

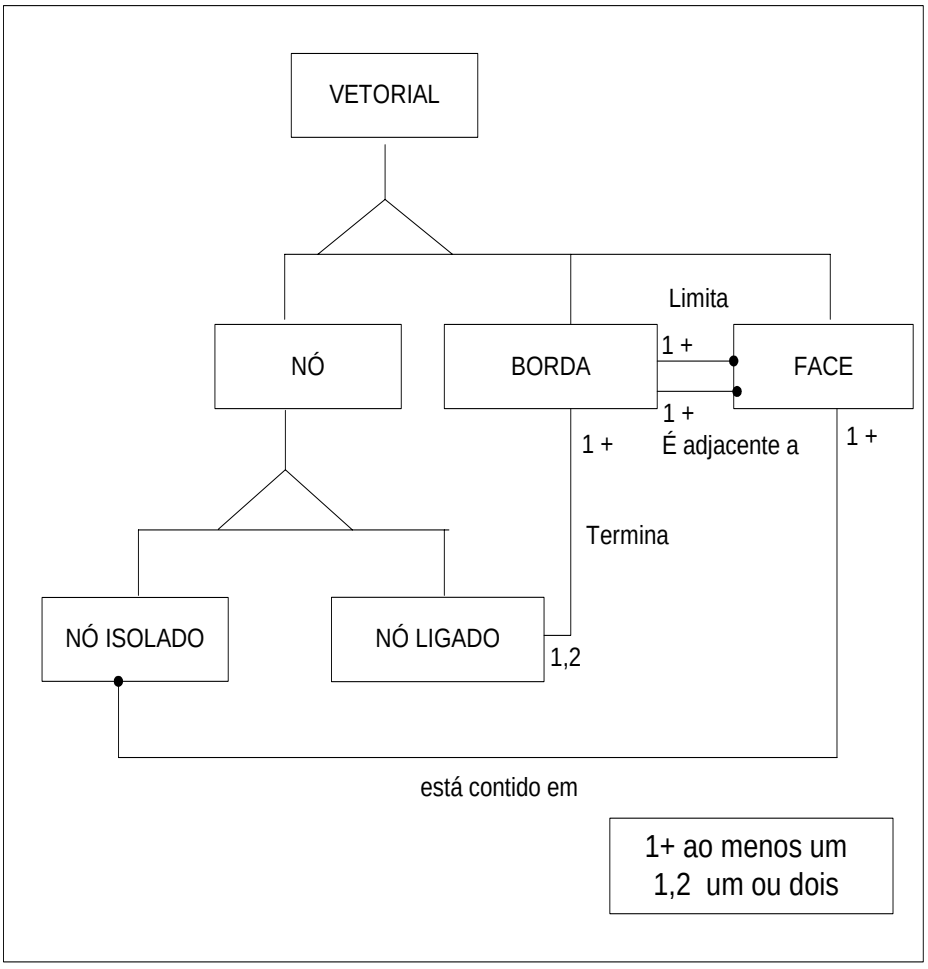


Figura 16 – Modelo Espacial

Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

Pode-se observar os seguintes relacionamentos:

nó isolado.....está contido..... face
face..... contém.....nó isolado
borda..... limita..... face
face..... é limitada por..... borda
nó ligado..... termina..... borda
borda..... é terminada por..... nó ligado

borda..... é adjacente a..... face

Estes relacionamentos podem ser usados para descrever quatro níveis de topologia:

- Espaguete cartográfico;
- Cadeia de nós;
- Gráfico planar; e
- Topologia completa (integral).

Nesses modelos as representações de ponto, linha e área são especializações do relacionamento “é localizado por”.

3.2.2.1.1 Espaguete Cartográfico

Definido como um conjunto de nós isolados e bordas. Objetos feição podem não compartilhar objetos espaciais. Pode-se ter pontos representados como nós isolados, linhas representadas como uma série de pontos interligados e áreas como cadeias fechadas de pontos interligados.

Para que exista uma lógica consistente, bordas coincidentes precisam ter geometria idêntica.

Observa-se na Figura 17 o modelo de espaguete cartográfico.

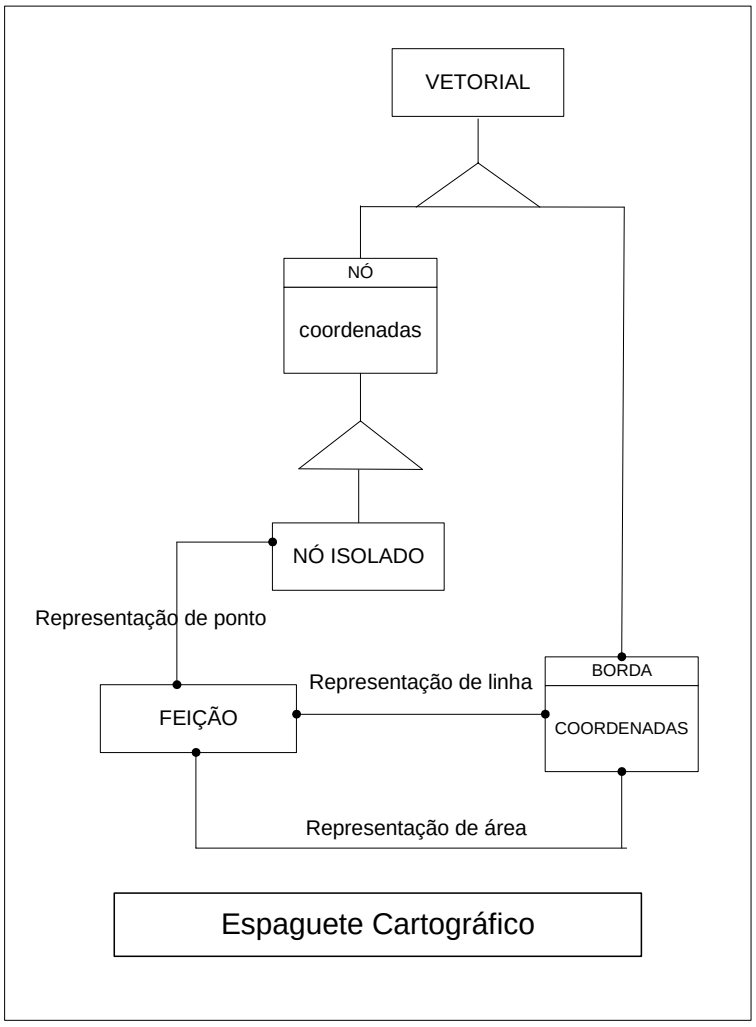


Figura 17 – Topologia Espaguete Cartográfico
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/..

3.2.2.1.2 Cadeia de Nós

Definida como um conjunto de nós e bordas. Cada borda precisa referenciar um nó conectado como sendo seu início e fim, podendo ser ou não o mesmo.

Pontos podem ser representados como nós (isolados ou conectados), linhas são representadas como uma série de bordas e nós conectados e áreas são representadas como “loops” fechados, começando e terminando em um nó comum. A Figura 18 apresenta o modelo Cadeia de Nós.

3.2.2.1.3 Gráfico Planar

Definido com um conjunto de nós e bordas, onde bordas não podem se cruzar e se tocam apenas em um nó conectado. Os objetos vetoriais podem ser compartilhados com a restrição de que bordas que se tocam sempre compartilham nós conectados e áreas adjacentes sempre compartilham as bordas que formam os limites comuns. A Figura 18 apresenta o modelo gráfico planar.

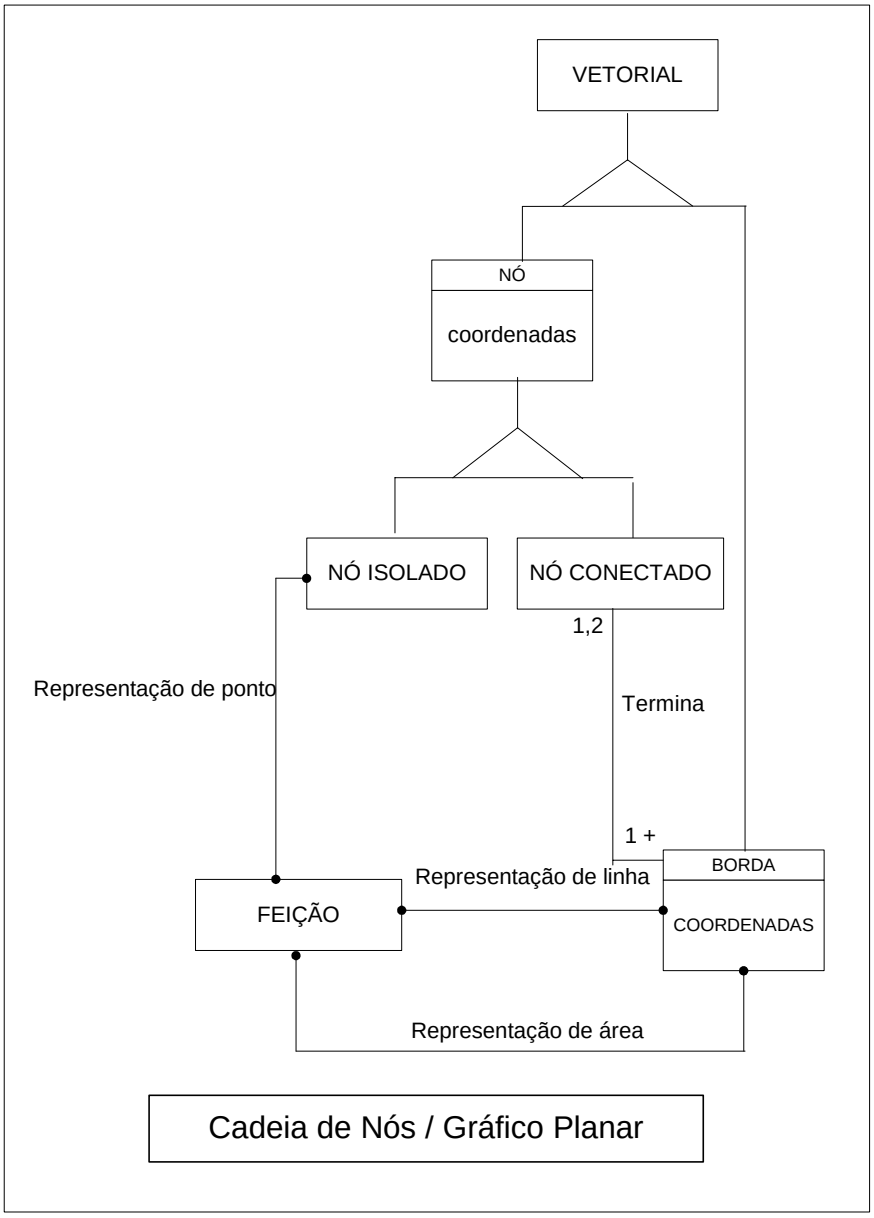


Figura 18 – Topologia Cadeia de Nós / Gráfico Planar
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

3.2.2.1.4 Topologia Completa

Definida como um conjunto de nós, bordas e faces. O universo é dividido em um conjunto de faces mutuamente exclusivas. Nós isolados devem referenciar as faces nas quais eles estão contidos e as bordas devem referenciar as faces esquerda e direita. Pontos são representados por nós (conectados ou isolados). Linhas são representadas por uma série de bordas e nós conectados. Áreas são representadas por faces. A Figura 20 apresenta o modelo de topologia completa.

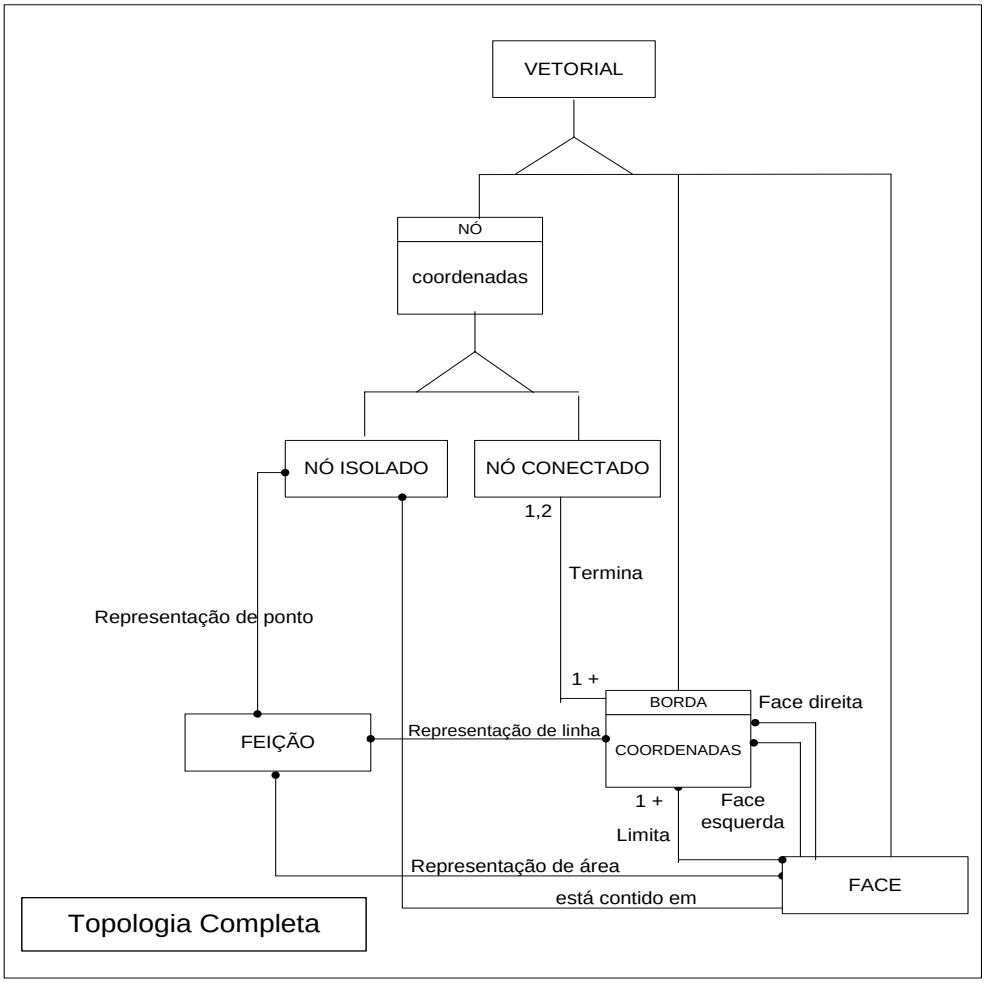


Figura 19 – Topologia Completa

Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

3.3 Vantagens da Utilização do Padrão S-57

A utilização do padrão S-57 provê algumas vantagens:

- É um padrão internacional não proprietário mantido pela Organização Hidrográfica Internacional [9];
- Usa o padrão de formato ISO/IEC 8211, com a intenção de encapsular os dados. Este formato provê um mecanismo básico para a transferência de dados de um sistema computacional para outro, independentemente da arquitetura. Permitindo não só a transferência dos dados, bem como a descrição da organização dos mesmos;
- Projetado como um formato cambiável, permite que o "software" cliente que vai ler os dados em S-57 seja diferente do "software" que gerou os mesmos dados, desobrigando, assim, a manutenção do mesmo pelo fabricante nas fases de produção e utilização;
- O dicionário de objetos e atributos é focado em dados hidrográficos; e
- Permite uma extensão do dicionário de dados pelo usuário, de modo que o mesmo possa criar seus próprios objetos e atributos.

3.4 O Modelo S-57 Utilizado pela ESRI[10] no ArcGis[11]

O modelo de dados ESRI, para o ArcGis, possui em seu conteúdo uma nova extensão responsável por implementar dados náuticos espaciais, propiciando, desta forma, um mecanismo de banco de dados náutico eficiente que pretende prover implementações e análises, conforme o padrão ENC S-57, criando relacionamentos entre entidades planares e os elementos feição utilizados pelo padrão.

O modelo possibilita uma completa solução de análise do comportamento dos dados contidos nas cartas e suas variações, tornando possível uma gerência efetiva sobre essas alterações. A manutenção e a edição dos dados tornar-se-á mais segura e confiável.

---

[9] http://www.iho.shom.fr/, arquivo 31Main.pdf.

[10] Empresa destinada a implementar soluções SIG e prestar serviços de Auditoria, Consultadoria, Projectos e Integração de Sistemas, Desenvolvimento de Aplicações, Suporte Técnico e Formação, além de serviços de constituição de bases de dados geográficas.

[11] ArcGIS é uma família de produtos de software que dão forma a um GIS completo construído segundo standards da indústria que fornecem capacidades excepcionais, contudo fácil de se usar.

A introdução de relacionamentos, que não existem no modelo S-57 puro, pode facilitar a modelagem de certos elementos que não seriam completamente representados, como exemplo: o relacionamento um para muitos entre feições primitivas do modelo. Na Figura 20 pode-se observar a modelagem em seis camadas temáticas.

- Camada para representar os elementos provenientes de batimetria;
- Camada para representar elementos de entrada de dados;
- Camada para representar a superfície da terra;
- Camada para representar perigos de navegação;
- Camada para representar instalações portuárias; e
- Camada que possui a representação vetorial dos elementos existentes na base de dados geo-espacial.

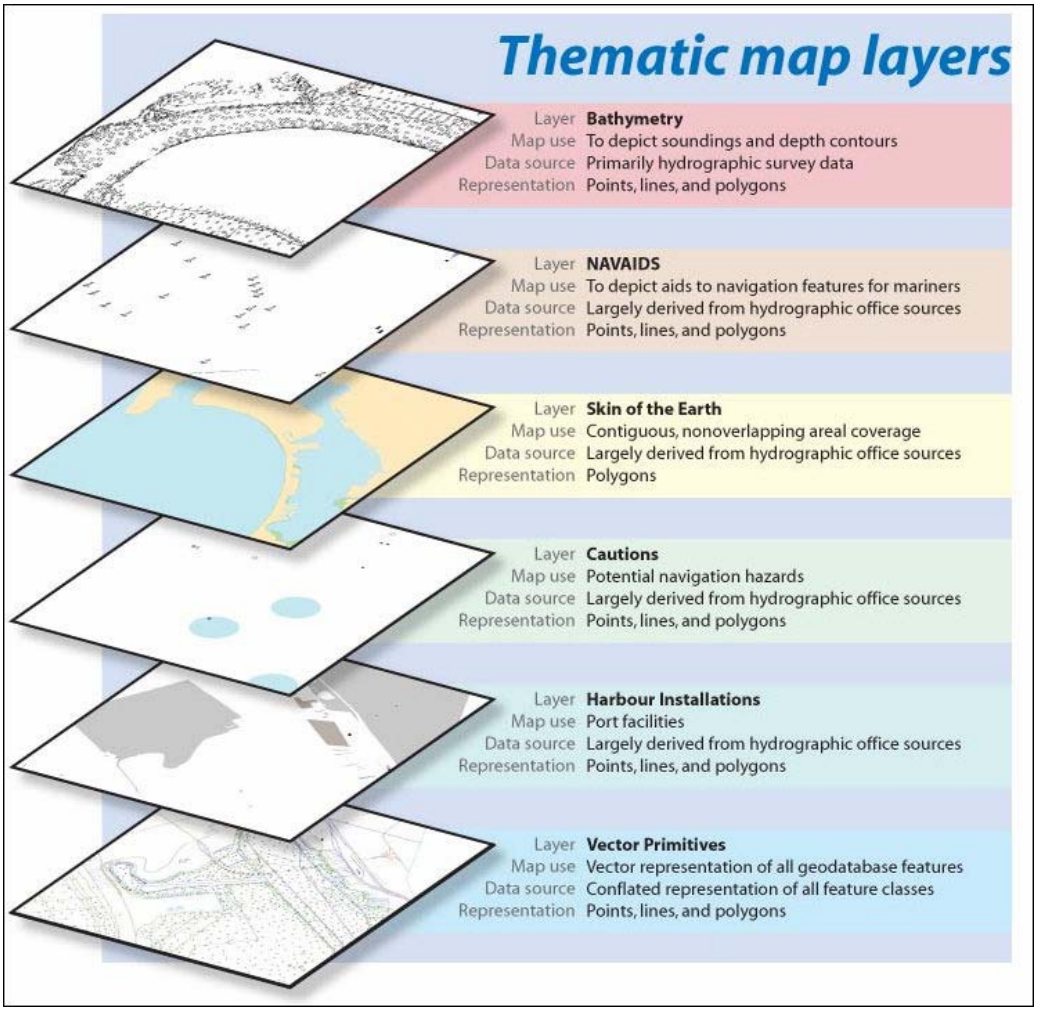


Figura 20 – Camadas de Representação do Modelo de Dados Náutico S-57 (http://www.esri.com/news/arcnews/summer04articles/nautical-data-model.html).

A conversão das cartas S-57 para o padrão ENC da ESRI permite uma importação dos objetos existentes no padrão da IHO, porém este modelo não deve ser utilizado para navegação. A ESRI apresentará, futuramente, o modelo para ser utilizado como base de dados em sistemas ECDIS, possibilitando, desta forma, manter os padrões estabelecidos para o desenvolvimento desses sistemas.

CAPÍTULO 4

4 O padrão 8211

4.1 Definição

Padrão destinado a encapsular dados, provendo um mecanismo básico para transferi-los de um sistema computacional para outro, independentemente de suas arquiteturas. Permitindo, também, a descrição da organização dos dados envolvidos.

4.2 Introdução à Estrutura

4.2.1 Registros

O padrão 8211 surgiu com a necessidade de armazenamento de informações, sendo escolhido para compor a estrutura organizacional destinada ao transporte de dados no formato S-57. Ele é, na realidade, um encapsulamento de uma estrutura de arquivos, composta, basicamente, por registros classificados em cinco categorias :

- Conjunto de dados descritivos (“Data Set Descritive- meta”);
- Catálogo;
- Dicionário de Dados;
- Feições (“features”); e
- Espaciais.

O conjunto de registros referentes aos dados do tipo descritivo contém informações sobre as identificações de propósito geral que podem ser: sistemas de coordenadas envolvidos, projeções, “datuns” verticais e horizontais, escalas, unidades, origem do conjunto de dados e descrição da precisão dos elementos de localização nos registros espaciais.

O conjunto de registros referentes ao catálogo possui informações pertinentes à localização de arquivos, permitindo a execução das decodificações. Eles podem ser

considerados uma tabela de conteúdo, possuindo informações referentes a relacionamentos especiais existentes entre registros individuais do conjunto de troca.

Como o formato S-57 possui implementações binárias e em código ASCII, a referência ao tipo de implementação também é contida no Catálogo, sendo que este é sempre implementado em ASCII.

O conjunto de registros referentes ao Dicionário de Dados resume-se no conjunto das descrições dos objetos, atributos e seus valores utilizados. Têm-se três divisões:

- Informações que definem as classes de objetos e os atributos;
- Informações sobre o domínio dos valores dos atributos; e
- Informações que identificam quais atributos são válidos para uma classe de objetos (“Schema”).

O conjunto de registros do tipo feição são as referências não espaciais, ou seja, que não referenciam os objetos por seus atributos de localização, e sim por suas características do mundo real. Eles podem ser dos seguintes tipos:

- Meta;
- Cartográficos;
- Geo; e
- Coleção.

O conjunto de registros espaciais contém dados de localização. Eles podem ser do tipo vetor, “raster” ou matriz, possuindo as mesmas estruturas, variando apenas em detalhes de implementação. No caso especifico de S-57, o vetor é tomado como referência. Assim sendo, registros vetoriais possuem as coordenadas geométricas relacionadas com os registros do tipo feição, incluindo atributos espaciais e topológicos, podendo ser de tipo nó, borda ou face.

4.2.1.1 Meta dados

Os meta dados podem ser encontrados em três níveis, como especificado abaixo:

- Meta informação, que provê o valor padrão para todo o conjunto de dados nos quais estes registros estejam contidos;
- Informação definida por meta objetos, que são especificados no catálogo de objetos da Organização Hidrográfica Internacional; e
- Informação definida pelos atributos dos objetos individuais derivadas dos objetos descritivos.

4.2.2 Convenções Gerais de Codificação

4.2.2.1 Identificador de Registro

O identificador de registro é o identificador básico da estrutura de dados, sendo utilizado para manter o relacionamento topológico. Todo registro do conjunto de troca baseado em S-57 deve conter um identificador, sendo que o mesmo se divide em “Record Name” (RCNM) e “Record Identification Number” (RCID). São encontrados no primeiro campo de todos os registros.

A concatenação destes dois subcampos forma a identificação do registro conhecido como NAME. Como registros são organizados em arquivos, esta concatenação de RCNM e RCID deve ser única dentro do arquivo que contém este registro.

4.2.2.1.1 Tipos de Registros e Nomes (RCNM):

- Informações Gerais – DS;
- Referencia Geográfica – DP;
- Histórico - DH;
- Precisão - DA;
- “Catalogue Directory” – CD;
- Referencia Cruzada do Catálogo – CR;

- Definição no Dicionário de Dados – ID;
- Domínio no Dicionário de Dados – IO;
- Esquema no dicionário de Dados (“Schema”) – IS;
- Atributo feição (“Feature”); e
- Vetor :
  - Nó isolado - VI;
  - Nó Ligado - VC;
  - Borda – VE; e
  - Face – VF.

4.2.2.1.2 Número de Identificação do Registro (RCID)

Os valores destinados para a identificação do registro, por definição variam de 1 até 2^{32} - 2 (dois elevado a trinta e dois menos dois).

4.2.2.2 Convenção de Terminação para Campos e Subcampos

Todos os subcampos de tamanho variável devem ser terminados por um terminador (UT), sendo que todos os campos em S-57 devem ser terminados por um “terminador de campo” (FT). Os valores de UT e FT podem variar, dependendo do conjunto de caracteres escolhido para o campo, estabelecendo-se, assim, níveis léxicos.

Nível léxico	UT	FT
Nível 0	(1/15)	(1/14)
Nível 1	(1/15)	(1/14)
Nível 2	(0/0) (1/15)	(0/0)(1/14)

4.2.2.3 Valores de Ponto Flutuante

Como diferentes plataformas computacionais armazenam valores reais utilizando diferentes convenções, estes valores são armazenados como inteiros. Assim, a conversão de valores fica a cargo da utilização de um fator de multiplicação, sendo que para valores de coordenadas e profundidade estes valores são os mesmos para todo o padrão, já os outros

campos que armazenam valores reais devem armazenar o valor de multiplicação no próprio registro.

Valor inteiro = valor real * fator de multiplicação

4.2.3 Convenções de Codificação para Registros do Tipo Meta

Os dados do tipo Meta podem ser hierarquicamente codificados em três diferentes níveis. Formam um conjunto de dados descritivos, definidos como meta registros. São especificados como:

- Registro de informações gerais do conjunto de dados;
- Registro de referências geográficas do conjunto de dados;
- Registro de histórico do conjunto de dados;
- Registro de precisão do conjunto de dados; e
- Registro de referência ao Diretório do catálogo.

4.2.3.1 Topologia

No modelo de dados teórico, são definidos vários níveis de relacionamentos topológicos. Estes níveis são implementados em estruturas de dados do tipo vetor. Desta forma, o tipo de estrutura de dados utilizada precisa ser expressa, explicitamente, no subcampo DSTR do campo DSSI (Estrutura de Informação do Conjunto de Dados). Pode-se ter então:

- CS – Espaguete cartográfico;
- CN – Cadeia de nós;
- PG – Gráfico planar;
- FT – Topologia completa; e
- NO – Topologia não relevante.

Tendo-se definido a estrutura de dados no DSTR como ”espaguete cartográfico” (CS), apenas registros do tipo “nó isolado” e “borda” podem estar presentes no conjunto de dados.

Da mesma forma, o registro do tipo “nó conectado” só pode estar presente se a estrutura de dados for definida como “Cadeia de nós” , “gráfico planar” ou “topologia completa”. Assim, também o registro do tipo “face” só é permitido se a estrutura do tipo “topologia completa” for definida.

4.2.3.2 Sistemas de Coordenadas, Unidades e Projeções

Neste formato, como já foi visto, dados hidrográficos possuem componentes descritivos e espaciais. Para facilitar a transferência de componentes espaciais deve-se definir um sistema de coordenadas. O registro de referência geográfica é usado para a troca de detalhes dos sistemas de coordenadas envolvidos. Estes detalhes possuem dois componentes básicos:

- Unidades das coordenadas; e
- Projeção.

4.2.3.2.1 Unidades das Coordenadas

Coordenadas podem ser codificadas por três diferentes métodos. O tipo de unidade é codificado no campo “Coordinate Unit” (COUN), sendo este um subcampo do campo “Data Set Parameter” (DSPM). Pode-se ter três opções válidas para o campo COUN:

- LL – latitude e longitude (graus de arco) com suas subformas
  - Graus decimais; e
  - Graus, minutos e segundos.
- EM – “ Easting/Northing”(metros); e
- UC – Unidades na carta/mapa (milímetros).

4.2.3.2.2 Projeções

Nas transformações de unidades, convertendo-se coordenadas em latitude e longitude para posições geográficas, ou seja, referenciando a superfície da terra, torna-se necessária a utilização de alguns dados, a saber:

- A projeção empregada na carta ou mapa; e
- Um número suficiente de pontos registrados[12].

Até quatro parâmetros podem ser definidos no campo projeção. A Tabela 2 define estes parâmetros.

Nome	valor	parâmetro 1	parâmetro 2	parâmetro 3	parâmetro 4
Albert Equivalente	ALA	Meridiano Central	Parale. padr. perto equad.	Parale. padrão dist equador	paralelo de origem
Azimutal Equivalente	AZA	Longitude da tangente	latitude de tangência		
Azimutal Equi-dististante	AZD	Longitude da tengente	latitude da tangente		
Gnomonica	GNO	Longitude da tengente	latitude da tangente		
Obliq de Merc Hot	HOM	Longitude da projeção de origem	Lat da projeção de origem.	Azimute. Eixo x na projeção de origem.	Fator de esc na proj de Origem
Cônica de Lambert	LCC	Meridiano Central	Paral perto equador	Parale. Mais afast equador	parel origem
Lambert Equivalente	LEA	Meridiano Central			
Mercator	MER	Meridiano Central	Lat da escala verdadeira	Paralelo da Origem	
Obliqua de Mercator	OME	Longitude do ponto refer. gr círculo.	Latitude do ponto ref gr círculo.	Azimute.Do gra.Circ.pt de referência	
Ortográfica	ORT	Longitude da tangente	Latitude da tangente		
Polar Estereografica.	PST	Meridiano Central	Lat da escala verdadeira		
Policônica	POL	Meridiano Central			
Transversa de Mercator	TME	Meridiano Central	Fator da escala central	Paralelo da Origem	
Sterografica Obliqua	OST	Longitude da origem	Latitude da origem	Fat Escala na Origem	

Tabela 2 - Projeções
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em
http://www.iho.shom.fr/

[12] Pontos em que a unidade das coordenadas e a posição geográfica são conhecidas.

Fator de Multiplicação para Dados de Batimetria

Os valores de profundidade são convertidos para inteiro, uma vez que na sua codificação ficam armazenados desta forma. Como são valores reais em sua medição, devem possuir um fator de multiplicação que é denominado (SOMF).

4.2.3.2.3 Verificação de Integridade dos Dados

Algoritmos de validação de CRC (Verificação Redundante Cíclica[13]) devem ser utilizados para verificação da integridade dos dados. Estes algoritmos fazem a inspeção se nenhum dado foi corrompido no processo de transferência. Esses algoritmos podem variar e esta variação depende, diretamente, da aplicação, que deverá especificar qual o método de validação.

O apêndice A possui uma sugestão de algoritmo de validação de CRC.

4.2.4 Convenções de Codificação para Registros do Tipo Feição

4.2.4.1 Descrição Geral

A classe do tipo feição possui objetos que representam as características dos elementos do mundo real, sendo descritos no apêndice A do catálogo de objetos da Organização Hidrográfica Internacional (IHO). O catálogo define quatro categorias de classes de objetos:

- Meta;
- Cartográfico;
- Geo; e
- Coleção.

Cada categoria é implementada por uma estrutura de registros do tipo feição, codificados semelhantemente.

[13] Tipo de método de detecção e correção de erros aplicado em transmissão de dados.

Registros do tipo feição possuem os campos:

- Campo identificador do registro;
- Campo identificador do objeto;
- Campos de atributos;
- Campos de ponteiros de controle; e
- Campos de ponteiros.

Obs: Os campos ponteiros de controle são utilizados apenas para a atualização.

4.2.4.2 Campo Identificador de Registros do Tipo Feição

Ele consiste em grupos de subcampos como especificado a seguir:

- Identificador do registro (RCNM, RCID);
- Primitiva de geometria do objeto (PRIM);
- Grupo (GRUP);
- Rótulo (“label”) do objeto (OBJL);
- Versão do registro (RVER); e
- Instrução de atualização do registro (RUIN).

O sub-campo Primitiva de geometria do objeto (PRIM) é responsável por armazenar os elementos de geometria básica ou primitiva do objeto codificado. Este sub-campo pode assumir os valores:

- P – ponto;
- L – linha;
- A - área; e
- N - objeto que não é referenciado diretamente por nenhuma geometria (coleção de registros do tipo feição).

Nos casos de geometria de profundidade, as primitivas são definidas como:

- “P” (ponto);

- Subcampo Grupo (GRUP), utilizado para agrupar objetos do tipo feição; e
- Subcampo “Rótulo” (OBJL), referencia o objeto no catálogo da IHO.

4.2.4.3 Campo Identificador de Objeto Feição

É composto pelos seguintes subcampos:

- Agência produtora (AGEN);
- Numero de identificação do objeto (FIDN); e
- Subdivisão da identificação do objeto (FIDS).

Exemplo de Estrutura de dados de um objeto do tipo <42> (DEPARE):

001 : <16>
FRID [RCNM]<100>[RCID]<102>[PRIM]<3>[GRUP]<1>[OBJL] <42>...
FOID [AGEN] <3168>[FIDN]<971180368>[FIDS]<1>
ATTF...

4.2.4.4 Campo de Atributo do Registro do Tipo Feição

Os atributos são definidos no registro chamado “Registro de Atributos de Feição” (ATTF). Os atributos válidos são definidos no catálogo de objetos da Organização Hidrográfica Internacional.

4.2.4.5 Campo de Atributos Nacionais do Registro do Tipo Feição

Estes atributos são codificados no “Registro de Atributos do Tipo Feição” (NATF), sendo também definidos no catálogo de objetos da Organização Hidrográfica Internacional.

4.2.4.6 Campo Ponteiro de Registro do Tipo Feição para Objeto do Tipo Feição

É utilizado para estabelecer um relacionamento entre objetos do tipo feição. Identificado como (FFPT), possui um subcampo (LNAM) que é a chave estrangeira para o

objeto que está sendo referenciado. Possui, também, o subcampo “indicador de subcampo” (RIND), utilizado para qualificar o relacionamento, como por exemplo, mestre ou escravo.

4.2.4.7 Campo Ponteiro de Registro do Tipo Feição para Objeto do Tipo Espacial

O campo ponteiro de registro do tipo feição para objeto do tipo espacial (FSPT) é utilizado para ligar um objeto do tipo feição à sua geometria. Possui como subcampo principal o campo (NAME), onde se encontra armazenada a chave para o objeto espacial referenciado. Possui, ainda, o subcampo (ORNT) para definir a orientação,onde o sub-campo (USAG) indicador de uso e o subcampo (MASK) máscara de referência.

É a geometria de um objeto feição que determina o uso do campo ponteiro. Registros do tipo geo, cartográfico e meta, podem ser dos tipos primitivo, ponto, linha ou área. Registros do tipo feição, cuja geometria é representada por ponto, linha ou área, podem referenciar apenas objetos do tipo vetor.

4.2.4.7.1 Campo Ponteiro de Registro do Tipo Feição para Registro Espacial Usado por Feições do Tipo Ponto.

No caso de estruturas de dados do tipo: cadeia de nós, gráfico planar e topologia completa, o ponteiro feição pode referenciar nós isolados ou nós conectados. Já no caso do espaguete cartográfico, o ponteiro feição deve referenciar somente nós isolados.

Em geral, ponteiros de registros do tipo feição devem referenciar apenas um registro vetor. Assim, múltiplos ponteiros para o mesmo registro não devem ser permitidos, a não ser no caso particular de registros de feição de batimetria.

4.2.4.7.2 Campo Ponteiro de Registro do Tipo Feição para Registro Espacial Usado por Feições do Tipo Linha.

Para facilitar a decodificação de feições lineares que compreendem múltiplas bordas, os registros no vetor destas bordas devem estar seqüencialmente referenciados.

A direção em que uma borda é interpretada em uma feição linear em particular torna-se importante. Esta direção e codificada por intermédio de um valor armazenado no subcampo (ORNT) e pode assumir os valores abaixo:

- F (1) - Para frente;
- R (2) - Reverso; e
- N (255) - Direção não é relevante.

Pode-se observar na Figura 21 as diferentes configurações de seqüência de nós, conforme a topologia a ser utilizada.

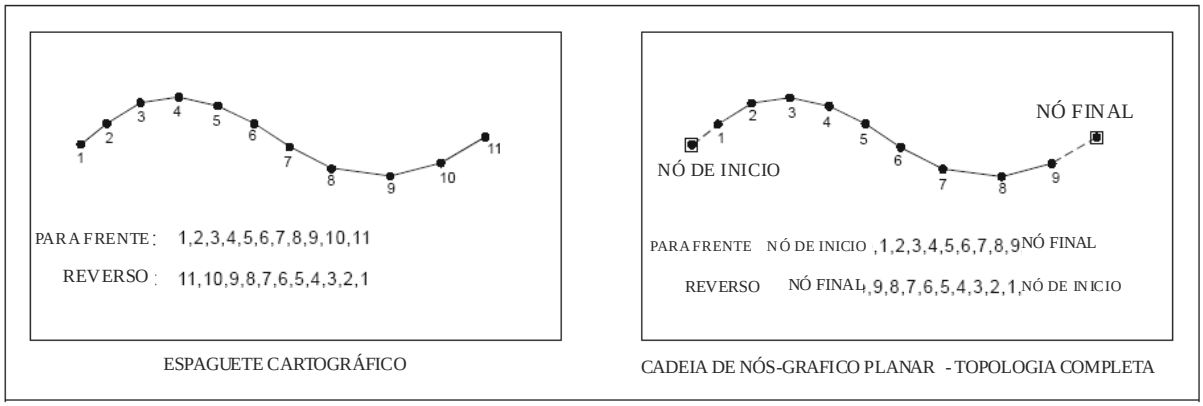


Figura 21 – Seqüência de Nós
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

Uma vez que o subcampo (USAG) tenha valor igual a “N” (255), o sub-campo MASK especifica onde a borda referenciada deve ser ou não mascarada , assumindo como valores permitidos:

- M (1) - Mascara;
- S (2) - Mostra; e
- N (255) - Critério de máscara não é relevante.

Observa-se na Figura 22 a aplicação da máscara, quando se faz necessária a sua ocultação da borda.

Figura 22 – Máscara de Borda
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

4.2.4.7.3 Campo Ponteiro de Registro do Tipo Feição para Registro Espacial Usado por Feições do Tipo Área.

Utilizando-se a modelagem de "Topologia Completa", as áreas são codificadas como uma ou mais faces. Os registros do tipo feição referenciam as faces e estas, por sua vez, seus limites de borda. Para todos os outros tipos de topologia as faces não são estruturas válidas.

No caso de estrutura do tipo "Espaguete Cartográfico", as áreas são codificadas como um conjunto de bordas próximas. Esta codificação se mantém para os casos de estruturas do tipo "Cadeia de Nós" e "Gráfico Planar", porém, nestes casos, em sua codificação utiliza-se também um conjunto de nós conectados.

O fechamento dos limites das áreas e faces precisa ser explícito, sendo que, no modelo "Espaguete Cartográfico", o primeiro ponto da primeira borda limite precisa ser o mesmo do último ponto da última borda limite.

No caso de estruturas do tipo “Cadeia de Nós” e “Gráfico Planar”, a primeira e a última bordas limite de uma área, precisam se encontrar em um nó conectado. Na “Topologia Completa”, a primeira e a última bordas limite de uma face precisam se encontrar em um nó conectado comum. Desta forma, para facilitar a decodificação de áreas, os limites devem ser referenciados seqüencialmente, codificando-se primeiro qualquer limite exterior completamente, para só então se codificar o limite interior.

Limites exteriores de áreas devem ser codificados no sentido dos ponteiros do relógio. Desta forma, limites interiores por sua vez devem ser codificados em sentido contrário. Assim, o codificador deve indicar qual direção (para frente ou reversa) as coordenadas devem ser usadas a fim de produzir a direção desejada. Esta orientação deve ser indicada no subcampo “Orientação” (ORTN), podendo assumir os valores:

- F (1) - Para frente; e
- R (2) - Reverso.

A Figura 23 apresenta os limites exteriores e interiores de áreas.

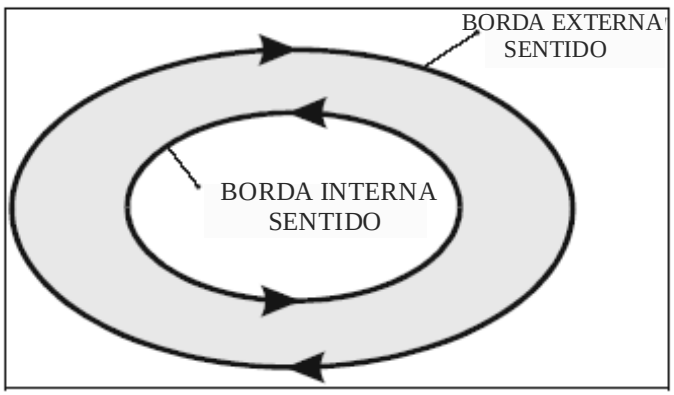


Figura 23 – Limites Exteriores e Interiores de Áreas
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

No caso de áreas que possuem uma borda externa e bordas internas que não se interceptam (áreas com buracos), um indicador (USAG) deve ser utilizado. Este subcampo

deve distinguir os limites interior e exterior. Este indicador é utilizado também para indicar que um limite exterior é truncado por um limite, podendo assumir os valores:

- E (1) - Limites exteriores;
- I (2) - Limites interiores; e
- C (3) - Limite exterior truncado por um limite.

A Figura 24 apresenta os limites de bordas que não se interceptam.

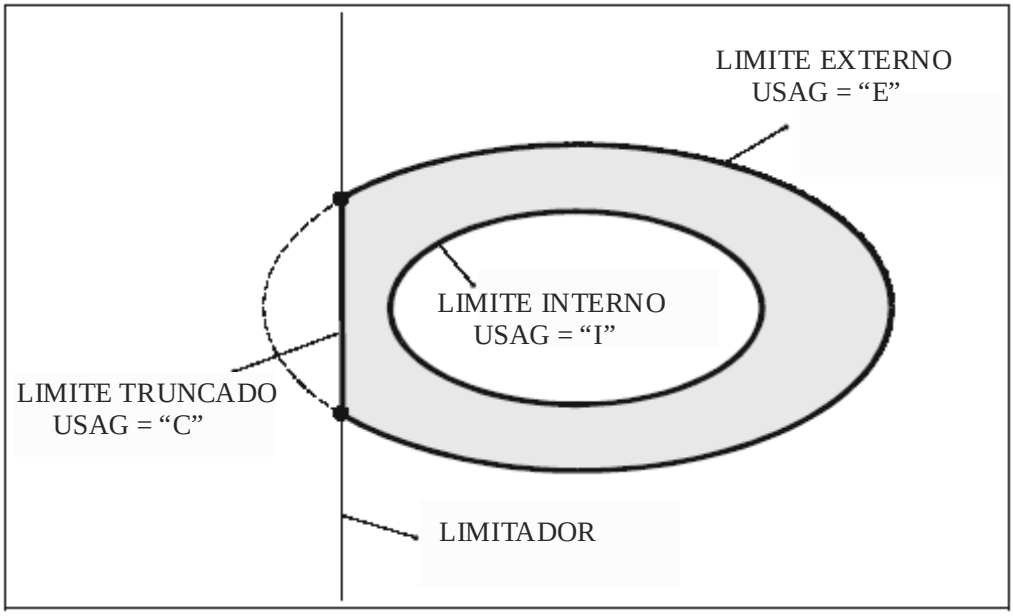


Figura 24 – Limites de Bordas que não se Interceptam.
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

Quando um limite interno é truncado por um limitador, ele se torna um limite externo, como se pode observar na Figura 25.

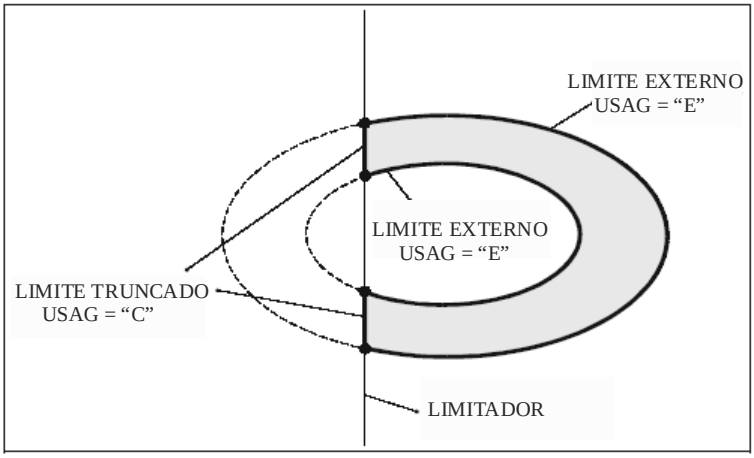


Figura 25 – Limites Interior e Exterior Truncados
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

Em certas circunstâncias, faz-se necessária a supressão da simbologia de uma ou mais bordas que definem o limite interno ou externo de uma área (Figura 22). Esta supressão é controlada por um “Indicador de máscara” (MASK), que pode assumir os valores:

- M (1) - Máscara;
- S (2) – Mostra; e
- N (255) - Máscara não é relevante.

4.2.5 Convenções de Codificação para Registros do Tipo Espacial

Este trabalho ocupa-se, exclusivamente com registros espaciais do tipo vetorial.

4.2.5.1 Registros Vetoriais

Como definido, um registro vetorial pode ser do tipo nó isolado, nó conectado, borda ou face. A geometria de elementos de batimetria é considerada um caso especial de um nó isolado.

Têm-se, assim, como campos de um registro vetorial:

- Campo identificador do registro;

- Campo atributo;
- Campo ponteiro de controle;
- Campo ponteiro;
- Campo controle de coordenada; e
- Campos de coordenadas (o padrão define vários tipos de campos de coordenadas, os quais são mutuamente exclusivos).

4.2.5.1.1 Campo Identificador do Registro

Este campo armazena o identificador (chave). Ë utilizado, também, para diferenciar os diversos tipos de registros. Neste caso, o subcampo (RCNM) é usado, podendo assumir os valores:

- VI (110) - Nó isolado;
- VC (120) - Nó conectado;
- VE (130) - Borda; e
- VF (140) - Face.

4.2.5.1.2 Campo Atributo

Em geral, atributos devem ser codificados no campo “Atributo de registro vetorial” (ATTV), que possui um subcampo (ATTL) onde é armazenado o atributo “Label/Code” dos objetos do catálogo de objetos da Organização Hidrográfica Internacional. Possui, também, o campo (ATVL) que deve ser uma “string” de caracteres.

4.2.5.1.3 Campo Ponteiro

O “Ponteiro para registro vetor” (VRPT) é utilizado para manter um correto relacionamento topológico entre os dados. Podendo ser utilizado nos casos de topologias do tipo “Cadeia de Nós”, “Gráfico Planar” e “Topologia Completa”, para registros do tipo borda. É também utilizado nos casos de Topologia Completa para registros do tipo nós isolados e

face, não sendo usado para registros do tipo nós conectados. Deve-se ressaltar que no caso de Espaguete Cartográfico este campo não se aplica.

4.2.5.1.3.1 Campo Ponteiro Usado por Nós Isolados

No caso de topologia completa, um nó isolado pode referenciar uma face na qual ele está contido. Neste caso utiliza-se o subcampo “Indicador de Topologia” (TOP), que contém o valor:

- F (5) - Face Contida.

4.2.5.1.3.2 Campo Ponteiro Usado por Bordas

No caso de Cadeia de Nós, Gráfico Planar e Topologia Completa, o começo e o fim de uma borda são explicitamente codificados como nós conectados. Os nós conectados são referenciados pela borda. Nos casos de bordas fechadas, elas devem referenciar o mesmo nó duas vezes. Deve-se observar que, ainda no caso de Topologia Completa, as bordas deverão referenciar as faces tanto pela direita, quanto pela esquerda. Todas as referências são obrigatórias, visto que a omissão de alguma acarretará em uma estrutura topológica corrompida.

Os valores pertinentes a este subcampo (TOP) são:

- B (1) - Nó inicial;
- E (2) - Nó final;
- S (3) - Face esquerda; e
- D (4) - Face direita.

Pode-se observar o exemplo de codificação de borda na Figura 26.

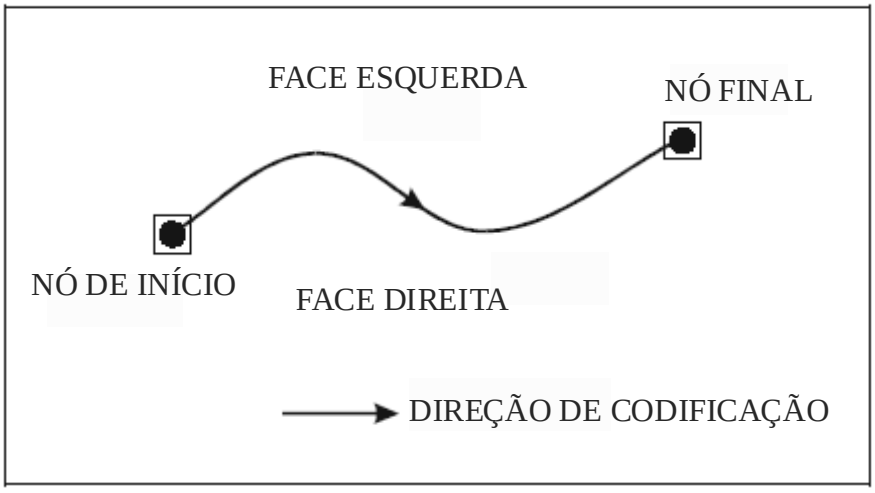


Figura 26 – Codificação de Borda
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

4.2.5.1.3.3 Campo Ponteiro Usado por Faces

Um registro do tipo face só pode referenciar bordas. A direção em que uma borda é codificada é armazenada no subcampo “Orientação” (ORTN). Utiliza-se o subcampo (USAG) para definir se esta borda faz parte do limite interior ou exterior de uma face, ou ainda, se coincide com o limitador. Neste caso, o subcampo “Indicador de máscara” (MASK) define se a borda referenciada deve ser mascarada ou não. Tem-se, então, uma combinação de valores dos três campos: ORTN, USAG e MASK, possuindo como valores válidos:

- ORTN :
  - F (1) - Para frente; e
  - R (2) - Reverso.
- USAG:
  - E (1) - Limite exterior;
  - I (2) - Limite interior; e
  - C (3) - Limite exterior truncado pelo delimitador.
- MASK:
  - M (1) - Mascarar;
  - S (2) - Mostrar; e

- N (255) - Máscara não é relevante.

4.2.5.1.4 Campos de Coordenadas

A geometria espacial dos objetos é codificada nos campos de coordenadas, tendo-se como tipos de campos:

- Campo de coordenada 2-D;
- Campo de coordenada 3-D; e
- Campo de Arco/Curva.

Os campos definidos acima são mutuamente exclusivos, sendo sua utilização vinculada ao tipo de registro no qual está sendo codificado.

Os campos de coordenadas se dividem em:

- Campos de coordenadas utilizados em elementos de batimetría. Para se promover uma maior eficiência, elementos provenientes de batimetria podem ser agrupados em um único registro. Sendo assim, uma estrutura especial deve ser utilizada . Esta estrutura é conhecida como coordenada 3-D ou campo vetor de batimetria, onde os elementos coordenada Y, coordenada X e profundidade se repetem;

- Campos de coordenadas utilizados por nós isolados. Codificados em “coordenadas 2-D”, campo (SG2D), constituído de um par de coordenadas;

- Campos de coordenadas utilizados por nós conectados. Codificados em “coordenadas 2-D”, campo (SG2D), constituído de um par de coordenadas.

- Campos de coordenadas utilizados por bordas. Devem ser codificadas utilizando-se campos de “Coordenadas 2-D” (SG2D) ou “definição de Arcos e Curvas” (ARCC). Na utilização de campos SG2D, opta-se por repetidos pares de coordenadas ligados implicitamente por linhas retas. Já na codificação por ARCC, utiliza-se derivadas matemáticas dos arcos e curvas. A utilização destas definições é mutuamente exclusiva. No caso de estruturas de dados do tipo “Cadeia de Nós”,

"Gráfico Planar" e "Topologia Completa", o início e o fim de uma borda precisam ser explicitamente codificados como nós conectados, sendo que a geometria destes nós não faz parte da borda. No caso de codificação via arcos e curvas, o tipo de interpolação deve ser armazenado no subcampo "Tipo de Arco / Curva" (ATYP).

Pode-se ter dois tipos para a representação de arcos:

- Arco de três pontos centrado, descrito por três pontos: o ponto de início (STPT), o ponto central (CTPT) e o ponto final (ENPT), formando assim um vetor, tendo como sentido de desenho o dos ponteiros do relógio. Este tipo é apresentado na Figura 27; e
-

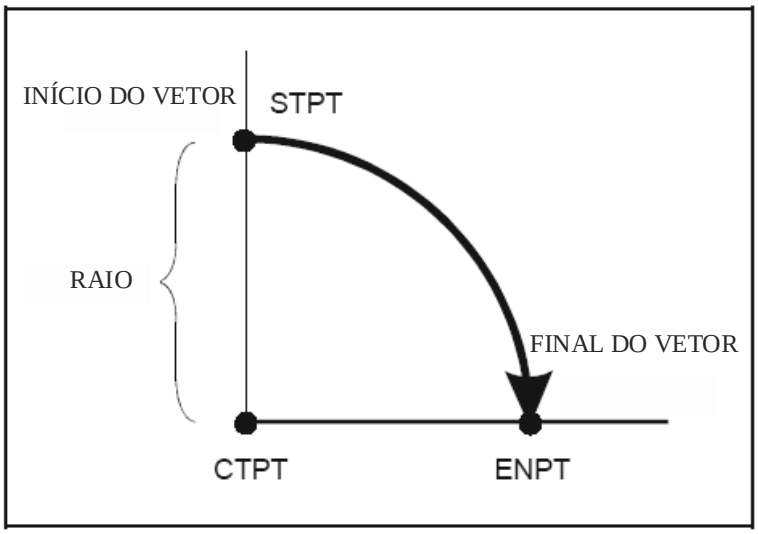


Figura 27 - Representação de Arcos (Três Pontos)
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

- Arco elíptico, descrito por cinco pontos: iniciando no ponto (STPT), onde se tem o início do vetor, finalizando em (ENPT), centrado em (CTPT), passando pelo ponto sobre o eixo maior (CDPM) e o ponto sobre o eixo menor (CDPR). Este tipo é apresentado na Figura 28.

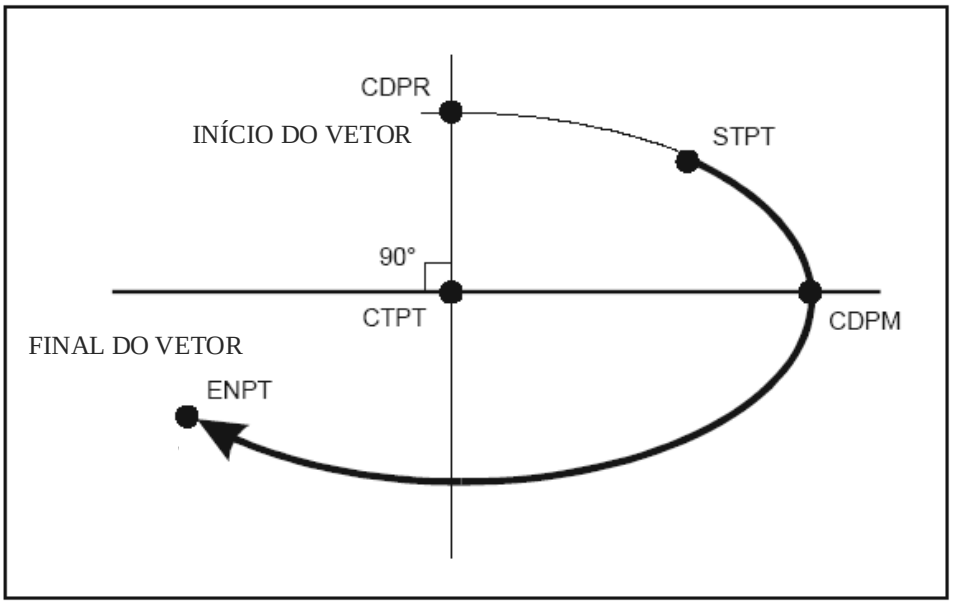


Figura 28 - Representação de Arcos (Elíptico)
Extraída da especificação S-57, versão 3.1, documento 31main.pdf em http://www.iho.shom.fr/.

Na representação por curvas, a codificação engloba uma junção complexa de segmentos de reta e pontos de controle suficientes para que a mesma seja apresentada como uma curva suave para o observador. Levando-se em consideração esta implementação, tem-se uma baixa complexidade da estrutura de armazenamento, porém verifica-se como desvantagem o grande volume de dados. Ainda na questão da apresentação, tudo dependerá da escala de visualização, pois, a mesma pode chegar a valores aos quais o número de pontos da representação não seja suficiente para tornar a aparência da curva suave.

4.2.6 Codificação dos Relacionamentos

Relacionamentos entre registros podem se codificados de três maneiras:

- Utilizando um registro de “catálogo de referência cruzada”;
- Utilizando uma coleção de registros do tipo feição; e
- Definindo-se um registro feição “mestre”.

4.2.6.1 Registro de Catálogo de Referência Cruzada

Pode ser utilizado para ligar qualquer tipo de registro. Assim, em ambos os registros que se encontram ligados, o ponteiro se divide nos subcampos (NAM1) e (NAM2), registrando-se a natureza do relacionamento no subcampo “comentário” (COMT).

4.2.6.2 Registro de Coleção de Feições

Este registro é a implementação do objeto coleção, onde classes de objetos coleção são definidas no catálogo de objetos da Organização Hidrográfica Internacional. Sua referência diz respeito somente aos objetos feição, não possuindo nenhuma ligação com registros espaciais. A codificação utiliza um campo que armazena um ponteiro para um objeto feição. Desse modo, um registro de coleção de feições deve referenciar ao menos outros dois objetos feição, não podendo se auto-referenciar. Registros de coleção de feição podem referenciar outros registros do mesmo tipo.

O subcampo “Indicador de Relacionamento” (RIND) é utilizado para indicar a natureza do relacionamento, podendo assumir os valores:

- M (1) - Mestre;
- S (2) - Escravo; e
- P (3) - Par.

Deve-se levar em conta certas regras na composição do conjunto de relacionamentos de registros de coleção de feições:

- Só pode existir um relacionamento mestre por registro de coleção de feições, sendo que todos os outros relacionamentos deste registro devem ser escravos; e
- Se um relacionamento é definido como par (P), todos os outros relacionamentos referentes a esta coleção devem ser definidos como par.

4.2.6.3 Registro Feição Denominado Como “Mestre”

Um registro denominado como mestre deve conter um campo ponteiro registro feição para cada objeto escravo, por exemplo: um objeto bóia pode ter como escravos um sinal luminoso, um sinal de neblina, etc. Assim, deverá possuir um ponteiro para cada um destes objetos. Um registro mestre pode referenciar outros registros feição mestres, porém não pode referenciar-se.

4.2.7 Implementação Estrutural

Fisicamente os registros, campos e subcampos do padrão 8211 seguem uma estrutura bem definida, sendo que sua representação utiliza uma notação tabulada que visa representar a hierarquia na qual esses elementos se encontram.

4.2.8 Atualizações

As atualizações dos dados são baseadas em um modelo de troca, como apresentado na Figura 29. Pode-se observar que as atualizações nos dados base tem sua compilação fundamentada nas informações de mudança recebidas pelo produtor de dados. A informação de atualização resultante gera registros atualizados, que são enviados ao “aplicador”, e este efetua as alterações nos dados de destino. O processo de atualização divide-se em operações de atualização. Uma simples alteração de um registro é considerada uma operação deste processo, onde pode-se ter inúmeras operações de atualização.

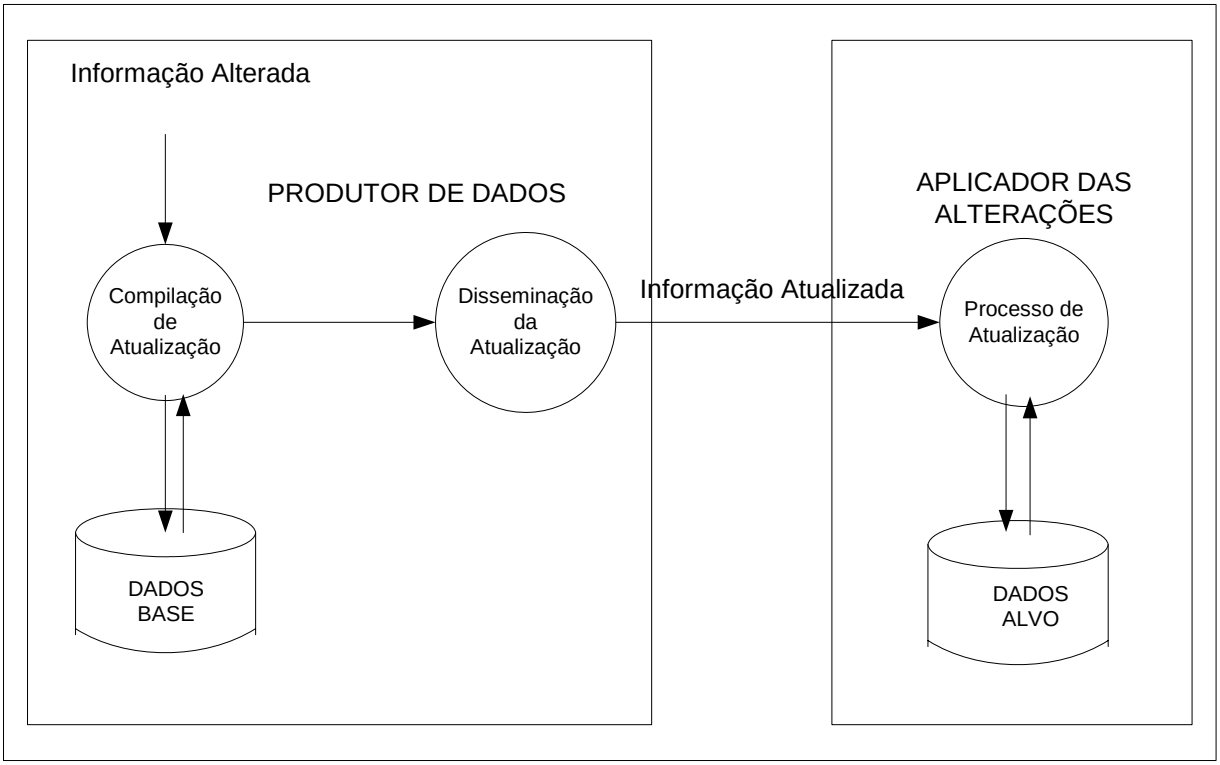


Figura 29 – Atualização dos Dados

Extraída da especificação do padrão 8211, em

http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=20305.

CAPÍTULO 5

5 A Biblioteca de Leitura

5.1 Introdução

Um arquivo no formato 8211 (DDFModule) possui uma estrutura bem definida composta por uma série de registros lógicos, sendo que o primeiro chamado de DDR (Registro de Descrição de Dados) é especial e contém descrições de todos os objetos de dados (campos ou objetos DDFFieldDefn) que possam ocorrer.

Os registros subseqüentes são conhecidos como “DRs” (Registros de Dados). Cada um contém um ou mais campos de dados (DDFField). Os campos que devem ou não aparecer são definidos pelo próprio padrão.

Essas ocorrências de campos possuem, cada uma, um nome e uma série de subcampos. Desta forma, cada tipo de campo possui sempre os mesmos subcampos, independentemente da ocorrência, sendo definidos no “Registro de Descrição de Dados” (DDFSubfieldDefn) e associados com suas definições (DDFFieldDefn).

As ocorrências de subcampos dentro dos campos possuem cada uma, também, um nome que as identifica. Pode-se observar a existência de vetores de subcampos dentro de campos, por exemplo: um campo de coordenadas pode possuir subcampos x e y, sendo que estes podem se repetir inúmeras vezes quando tem-se indicado uma série de pontos.

5.2 Descrição

- Classe DDFModule - É responsável por conter todas as informações do registros chamado “DDR”, Registro de Descrição de Dados. É formada por:

  - DDFModule - Construtor da classe;
  - DDFModule~ - Destrutor da classe;

- Open - Abre um arquivo 8211 para leitura. Caso a abertura do arquivo seja bem sucedida, o registro "DDR" é lido e as definições dos subcampos estarão disponíveis;
- Close - Responsável por fechar um arquivo do tipo 8211;
- ReadRecord - Responsável por ler o primeiro registro de dados do arquivo aberto;
- Rewind - Responsável por retornar para o primeiro registro do arquivo;
- FindFieldDefn - Responsável por procurar pelo campo indicado. Deve ser passado o nome do campo, sendo retornada sua definição;
- GetFieldCount - Responsável por procurar pelo índice de um determinado campo;
- GetField - Responsável por procurar pela definição de um campo passando-se um índice; e
- AddField - Responsável por inserir uma nova definição de campo.

- Classe DDFFieldDefn – Representa um container de informações sobre definições de campos. É formada por:

  - Dump - Responsável por prover as informações contidas dentro de um campo, sendo que todos os subcampos dentro deste campo serão também disponibilizados;
  - FindSubfieldDefn - Responsável por encontrar a definição de um subcampo, uma vez fornecido um ponteiro “TAG”[14];
  - GetDefaultValue - Retorna o valor padrão para uma determina instância do campo;
  - GetDescription - Retorna uma descrição para o campo;
  - GetFixedWidth - Retorna o tamanho do campo;
  - GetName - Procura um ponteiro para o nome do campo (“TAG”);
  - GetSubfield - Procura um subcampo pelo índice;
  - GetSubfieldCount - Retorna o número de subcampos;
  - IsRepeating - Procura por um “Flag” repetido, ou seja, retorna verdadeiro se o campo for repetido; e

[14] Este valor é determinado pela função GetName fornecendo-se o nome do campo

- SetRepeatingFlag - Marca um campo como repetido.

- Classe DDFRecord: Esta classe possui instâncias do registro de dados (DR). Os dados são armazenados sob a forma de uma lista de ocorrências de DDFField, repartindo os dados brutos em campos. É formada por:

  - AddField - Adiciona um novo campo o final do registro com tamanho zero;
  - Clone - Efetua uma cópia de um registro. Pode ser utilizado pela aplicação para ler registros, permitindo, inclusive, a leitura de registros novos;
  - CloneOne - Recria um registro referenciando um outro módulo. Funciona tal como o método Clone, porém pode-se ter um DDFModule diferente como referência;
  - CreateDefaultFieldInstance - Inicializa uma instância padrão. É utilizado, geralmente, pelo método AddField para iniciar uma nova instância de campo, com valores padrão para os subcampos;
  - DeleteField - Exclui um campo de um registro. O registro permanece com o mesmo tamanho, pois, a área de dados não é reaproveitada;
  - Dump - Permite que valores de campos e subcampos sejam escritos em um arquivo, possibilitando o monitoramento de valores;
  - FindField - Permite encontrar uma instância de um campo, passando-se o nome do mesmo;
  - GetData - Retorna um ponteiro para os dados do primeiro campo de um registro;
  - GetDataSize - Retorna o tamanho de um registro em bytes;
  - GetField - Retorna um ponteiro para o campo cujo índice é passado;
  - GetFieldCount - Retorna o número de campos (DDFFields) contidos no registro em questão;
  - GetFloatSubfield - Retorna o valor de um subcampo como um valor real;
  - GetIntSubfield - Retorna o valor de um subcampo como um valor inteiro;
  - GetModule – Retorna, tal qual DDFModule, o registro ao qual está associado;
  - GetStringSubfield - Retorna o valor de um subcampo como uma string dentro do registro em questão;

- ResizeField - Possibilita a mudança do tamanho dos dados de um campo, permitindo a alteração da quantidade de espaço reservada para alocar um dos campos existentes;
- SetFieldRaw - Permite a alteração dos dados de um campo, por exemplo: reposição dos valores do campo por valores de uma outra instância;
- SetFloatSubfield - Permite a alteração de um campo do tipo real;
- SetIntSubfield - Permite a alteração de um campo inteiro;
- SetStringSubfield - Permite a alteração de um campo string; e
- Write - Escreve um registro no final de um módulo. É utilizado, em geral, para escrever um registro no final de um arquivo.

- Classe DDFField - Representa as ocorrências de campos dentro de um registro do tipo DDFRecord, modelando uma instância dos campos de dados, diferentemente da classe DDFFieldDefn que possui a definição dos dados. É formada por:

  - Dump - Permite que valores de campos sejam escritos em um arquivo, para que exista a possibilidade de depuração;
  - GetData - Retorna um ponteiro para um registro. Uso interno;
  - GetDataSize - Retorna o número de bytes de um bloco de dados;
  - GetFieldDefn - Retorna os dados de definição de um campo;
  - GetInstanceData - Retorna os dados e o tamanho de uma instância de campo;
  - GetRepeatCount - Retorna o número de vezes que um subcampo ocorre em um campo; e
  - GetSubfieldData - Retorna um ponteiro para os dados armazenados em um subcampo do campo em questão.

- Classe DDFSubfieldDefn - Representa a definição dos dados contidos em um subcampo. É formada por:

  - Dump - Permite que valores de definição de subcampos sejam escritos em um arquivo, para que exista a possibilidade de depuração;
  - DumpData - Permite que valores de dados de subcampos sejam escritos em um arquivo, para que exista a possibilidade de depuração;

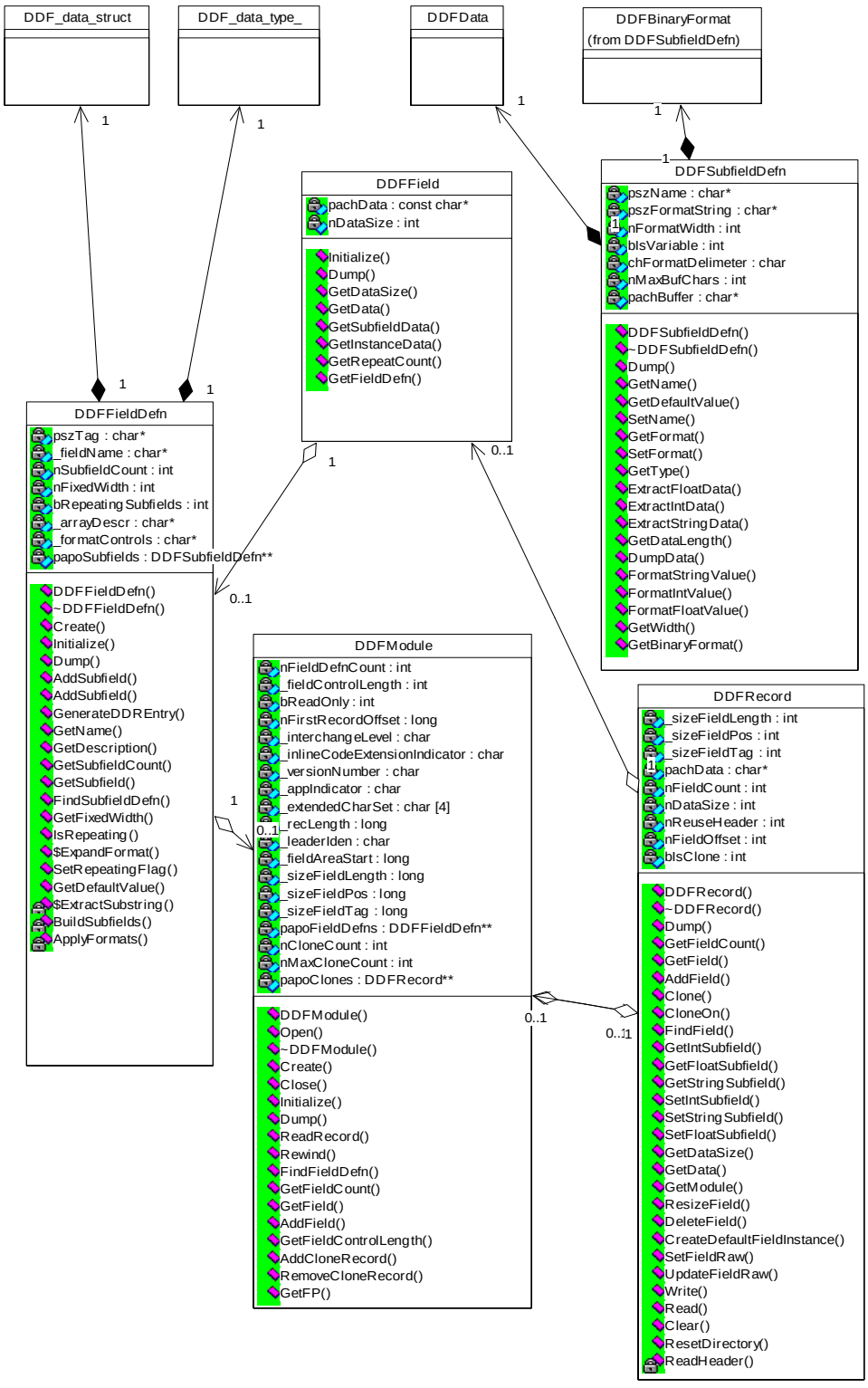
- ExtractFloatData - Permite a extração de valores de subcampos como reais;
- ExtractIntData - Permite a extração de valores de subcampos como inteiros;
- ExtractStringData - Permite a extração de valores de subcampos como strings;
- FormatFloatValue - Formata um subcampo de formato real;
- FormatIntValue - Formata um subcampo de formato inteiro;
- FormatStringValue - Formata um subcampo de formato string;
- GetDataLenght - Retorna o número de bytes dos dados de um subcampo;
- GetDefaultValue - Retorna o valor padrão de definição dos dados de um subcampo;
- GetFormat - Retorna um ponteiro para a string de formato do subcampo;
- GetName - Retorna um ponteiro para o nome do subcampo;
- GetType - Retorna o tipo de um subcampo; e
- GetWidth - Retorna o tamanho de um subcampo.

5.3 O Modelo

A Figura 30 apresenta um diagrama de classes da biblioteca de leitura de arquivos com objetos empacotados no padrão 8211.

5.4 Aplicabilidade

O apêndice A contém uma amostra da utilização da biblioteca de leitura, exibindo os valores de cada registro, campo e subcampo de uma carta no padrão S-57.



Classes da Biblioteca de Leitura do Arquivo no formato 8211

Figura 30 - Modelo da Biblioteca de Leitura Padrão 8211 – Cartas S-57

Obtida através da compilação das classes em

http://home.gdal.org/projects/iso8211/index.html.

CAPÍTULO 6

6 O Padrão S-52

6.1 Definição

Criado com o intuito de especificar o conteúdo das cartas e os aspectos de visualização em ECDIS, de forma a prover uma navegação eficiente e segura, satisfazendo, assim, as exigências de performance do padrão estabelecido pelas Organização Hidrográfica Internacional (IHO), Organização Marítima Internacional (IMO) e Comissão Eletrotécnica Internacional (IEC). Teve sua primeira publicação em novembro de 1995, que estabelecia detalhes técnicos de apresentação, atualização, cores e simbologia das cartas náuticas eletrônicas oficiais (ENC). Simultaneamente às suas elaboração e publicação, surge o padrão que define os requisitos necessários para a certificação de um ECDIS segundo a IMO. Este padrão especifica, em termos gerais, a apresentação das cartas vetorizadas autorizadas pelos órgãos governamentais, incluindo sua capacidade de atualização, planejamento e monitoração de rotas, posicionamento manual e apresentação contínua da posição do navio, proporcionando, assim, um panorama confiável, tal qual as cartas oficiais em papel.

O padrão especifica, ainda, funções adicionais para:

- Apresentação das informações do sistema em três níveis selecionáveis de detalhe;
- Garantia do correto carregamento das ENC e suas atualizações;
- Aplicação de atualizações automáticas ao sistema de apresentação;
- Proteção das cartas de qualquer alteração;
- Apresentação do conteúdo das atualizações;
- Armazenagem das atualizações separadamente e manutenção de um registro de aplicação das mesmas ao sistema;
- Apresentação de indicadores de sobre-escala ou sub-escala, caracterizando a imprópria utilização da escala na situação em questão;
- Sobreposição de imagens radar a informações ARPA no monitor;
- Apresentação de movimento real e relativo;
- Utilização das cores e símbolos definidos pela IHO;

- Utilização dos elementos e parâmetros de navegação especificados pela IEC;
- Utilização dos símbolos de tamanhos definidos, bem como, figuras e fontes apropriadas para as escalas especificas das cartas;
- Apresentação do navio como um símbolo ou em escala real;
- Apresentação do planejamento de rotas;
- Apresentação de uma visão clara das informações por mais de um usuário em condições diversas (diária ou noturna);
- Ajuste de “waypoints” e confecção de rotas em curva ou em linha reta;
- Apresentação de uma rota, mesmo não sendo a escolhida para a monitoração;
- Seleção de limite de navegação e a indicação de ultrapassagem dos limites de contorno seguro ou áreas proibidas;
- Apresentação de uma área distante da embarcação, simultaneamente à monitoração da rota escolhida;
- Escolha da prioridade de alarme, caso a embarcação ultrapasse uma área proibida ou de contorno seguro;
- Apresentação da posição da embarcação utilizando um sistema de posicionamento contínuo, com a precisão compatível com as necessidades de uma navegação segura;
- Identificação de discrepâncias entre os sistemas de posicionamento primário e secundário;
- Apresentação de alarme quando o sistema de posicionamento estiver fora de funcionamento;
- Apresentação de alarme quando o sistema de posicionamento e a carta em questão possuírem diferentes “datuns” geodésicos;
- Armazenamento e recuperação das últimas doze horas de navegação, permitindo a visualização das cartas utilizadas neste período;
- Necessidade de contínua ligação com sistemas que forneçam posição, rumo e velocidade;
- Permissão de não degradação das informações, devido à conexão com outros sensores;
- Condução de testes das funções prioritárias, com alarmes de mau funcionamento;
- Permissão de funcionamento com um sistema de força de emergência;

- Permissão de interrupção de fornecimento de energia até 45 segundos sem falhas ou necessidade de reinício; e
- Permissão de configuração de uma estação de “backup”, em caso de falha da estação principal.

6.2 Utilização

Com a intenção de facilitar a implementação do padrão, provendo um caminho seguro de utilização dos conceitos envolvidos em sua geração, o padrão S-52 propõe à elaboração de uma Biblioteca de Apresentação, “Presetation Library”, que deve ser seguida por todos que desejam implementar a abstrata descrição dos objetos no padrão S-57, em elementos visuais, em um ECDIS. Os conceitos e métodos presentes nesta biblioteca, que em seu âmago estão baseados nos conceitos do padrão S-57, afetam de maneira efetiva o projeto do ECDIS. Desta forma, os responsáveis por desenvolver estes sistemas devem se preocupar, logo no inicio de seu desenvolvimento, com o método de implementação da mesma.

6.3 Conteúdo

A Biblioteca de Apresentação implementa as especificações visuais, do padrão S-52, decodificando e simbolizando a SENC. Ela contém:

- Biblioteca de simbologia de um ECDIS, incluindo Símbolos de Navegação da IEC;
- Tabela de cores referentes aos modos de navegação (dia, noite e crepúsculo);
- Tabela de visualização, com as instruções simbólicas que ligam os objetos da SENC, com suas cores e símbolos apropriados, além de fornecer sua categoria IMO, prioridade de desenho, prioridade de apresentação sobre imagens radar, e sugestão de agrupamento de visualização para os objetos, formando grupos de visualização que reúnem objetos de características visuais semelhantes;
- Simbologias de procedimentos condicionais para:
  - Casos em que a simbologia do objeto dependa das circunstâncias, como exemplo: a escolha de um contorno seguro; e

  - Casos em que a simbologia é muito complexa para ser definida diretamente na Tabela de Visualização.
- Objetos de navegação marítimos, especificados no mesmo formato dos objetos das cartas, para fornecer um processamento conveniente no ECDIS (pontos de referência criados pelo operador); e
- Detalhes complementares, tais como:
  - "Software" de calibração de cor; e
  - Diagramas de testes de diferenciação de cores;

Os símbolos contidos na Biblioteca de Apresentação devem ser reproduzidos, fielmente, em forma e tamanho, usando-se qualquer formato conveniente. As tabelas de cores devem ser reproduzidas respeitando as tolerâncias dadas pelo próprio padrão. Os símbolos restantes devem seguir uma implementação conveniente de forma a prover uma semelhança de apresentação obtida com os objetos da biblioteca.

6.4 Estrutura do Modelo de Apresentação para um ECDIS

O modelo de apresentação para um ECDIS se divide em duas partes:

- Uma biblioteca de cores, estilo de linhas, estilos de preenchimento de áreas, símbolos pontuais e um conjunto de instruções simbológicas com as "look-up tables", que traduzem a descrição dos objetos em instruções simbológicas. Este conjunto é chamado de Biblioteca de Apresentação ("Presentation Library") do ECDIS, provendo, de forma eficiente, um formato que possibilita atualizações individuais.
- A descrição de uma estrutura programável, que serve de modelo funcional para a parte gráfica do sistema de um ECDIS, esclarecendo a utilização dos elementos da Biblioteca de Apresentação, e provendo um guia para a correta apresentação das estruturas de dados segundo o padrão S-57. Esta parte é chamada "Conceito Gerador de Apresentação".

6.5 Conceito Básico de um “Gerador de Apresentação” para um ECDIS

Baseado no fato de que os elementos da Biblioteca de Apresentação devem ser manipulados pelo Gerador de Apresentação, que possui seu projeto implementado, individualmente, por cada fabricante que se dispõe a produzir um ECDIS com as características dos objetos ditadas pelo padrão S-57, deve-se levar em conta os procedimentos propostos na Figura 31.

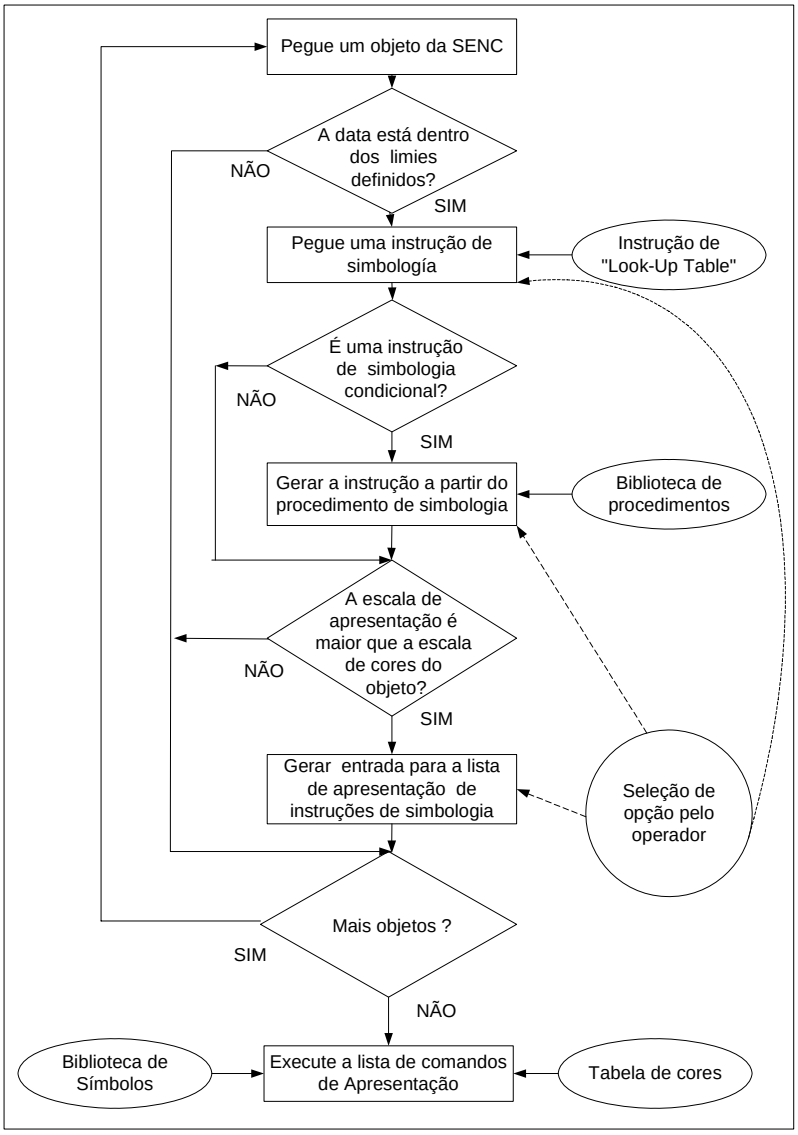


Figura 31 - Procedimentos Básicos para a Implementação de um Gerador de Apresentação
Extraída da especificação S-52 em http://www.iho.shom.fr/

Após todos os objetos terem sido analisados, a lista de exibição é preenchida com comandos, executados pelo módulo gráfico do ECDIS, que, por sua vez, carrega os símbolos da Biblioteca de Símbolos, adquirindo os valores corretos de cor, das tabelas de cores. Permitindo ao operador um controle efetivo do conteúdo e da aparência da apresentação, como por exemplo:

- Se o operador seleciona um outro contorno seguro “safety contour”, a lista de exibição é alterada para que as sombras das áreas de profundidade sejam modificadas. Isto é feito pela alteração das instruções simbológicas da classe DEPARE (área de profundidade); e
- Se os valores das cores para a apresentação durante o dia forem repostos para valores de cores de apresentação noturnos, altera-se apenas a tabela de cores.

6.6 Elementos utilizados pelo “Gerador de Apresentação”

O “Gerador de Apresentação” utiliza os elementos descritos a seguir:

- A biblioteca de símbolos, estilo de linhas e estilos de preenchimento;
- Um conjunto de diagramas que podem ser apresentados e que explicam a simbologia para o operador do ECDIS;
- Um esquema de codificação de cores, que inclui os padrões de apresentação IHO para dia e noite;
- Um conjunto de comandos de simbologia que, uma vez executados, resultam em instruções que apresentam a simbologia referente aos objetos do padrão S-57;
- Um conjunto de “look-up tables”, que relacionam a descrição do objeto no banco de dados da SENC com a instrução de simbologia apropriada, que pode variar:

  - Se a ligação relaciona diretamente a descrição do objeto e sua apresentação, tal como uma bóia ou uma área. Neste caso, a “look-up table” provê a instrução de simbologia apropriada para mostrar um símbolo ou o preenchimento e o estilo de linha necessários; e
  - Se a ligação for condicional, dependente das circunstâncias, como por exemplo: uma área de profundidade, cuja cor de preenchimento dependa da

escolha do **contorno seguro**. Neste caso, a "look-up table" transfere a decisão para um procedimento de simbologia condicional, que, então, seleciona as instruções de simbologia apropriadas.

- Um conjunto de procedimentos de simbologia condicionais, que decide que simbologia deve ser utilizada, no caso de uma seleção por parte do operador do ECDIS; e
- Um catálogo de classes de objetos de navegação, constituído de objetos que o operador pode adicionar a uma carta e que não estão definidos no padrão S-57.

6.6.1 O Esquema de Codificação de Cores

A Biblioteca de Apresentação utiliza um esquema de cores baseado em sua aplicação. Cada cor é representada por um símbolo composto por um código de cinco letras, que corresponde a uma definição de cor em um conjunto de tabelas de cores, para diferentes condições de brilho. Símbolos, estilos de preenchimento e estilos de linhas utilizam um esquema de codificação embutidos na própria definição do símbolo.

6.6.2 Símbolos, Estilos de Preenchimento e Estilos de Linha

A Biblioteca de Apresentação possui um conjunto de símbolos simples que são projetados para implementar os elementos pertinentes às cartas internacionais. Desta forma, é permitido ao ECDIS chavear a apresentação baseada em cartas de papel para uma apresentação que utiliza uma simbologia S-52 simplificada. Ele oferece, também, várias maneiras de preenchimento de áreas, baseadas em cores opacas ou com certo grau de transparência, possibilitando, ainda, o uso de padrões de preenchimento. Este procedimento é indicado como solução em certas situações especiais, quando utiliza-se cartas de papel tradicionais.

A simbologia escolhida para representar a direção do tráfego é feita por setas. A apresentação do símbolo pertinente a esta representação pode, às vezes, não aparecer devido ao tamanho e à posição da janela de visualização em uma carta do ECDIS, que não pode ser predeterminada. A visualização das setas utilizando-se o conceito de padrão de apresentação

tem um comportamento diferente, não sofrendo influência destes fatores. Pode-se ter, ainda, como estilos de linhas duas variações:

- Estilos de linhas simples, que podem ser sólidas, pontilhadas ou tracejadas, com variações de cor e espessura; e
- Estilos de linhas complexos, que são compostos por repetições de linhas padrões.

6.6.3 Instruções de Simbologia

A apresentação de um ECDIS é gerada por instruções de simbologia, que são traduzidas em palavras de comando designadas pela Biblioteca de Apresentação. Estes comandos são ordens que devem ser decodificadas para ações gráficas executadas pelo programa que implementa o ECDIS.

Existem cinco tipos de instruções de simbologia:

- Instruções para objetos do tipo linha;
- Instruções para objetos do tipo área;
- Instruções para objetos do tipo ponto;
- Instruções para objetos do tipo texto; e
- Instruções para chamada de procedimentos de simbologia condicionais.

6.6.4 Procedimentos de Simbologia Condicionais

A grande maioria dos objetos pode ser apresentada de maneira semelhante, utilizando-se instruções de simbologia para linhas, áreas ou símbolos. Já no caso de apresentações complexas, torna-se necessário o uso de simbologia condicional, que difere da simbologia tradicional, pois, decisões são tomadas durante a criação da apresentação do objeto, que podem afetar não só a simbologia do mesmo, bem como sua prioridade, grupo de visualização e categoria, dentre outros elementos.

6.6.5 As “Look-Up Tables”

As condições de visualização adversas, constituídas pela luz do sol ou pelo escuro da noite, em conjunto com o uso do monitor, levaram ao desenvolvimento de novos símbolos de representação, de forma a prover uma observação mais segura, proporcionando, assim, uma melhor resposta por parte dos operadores em situações de risco.

As “Look-Up Tables” contêm instruções que implementam a maneira pela qual uma instância de uma classe de objetos pode ser simbolizada. Elas são parte integrante da Biblioteca de Apresentação, que possui cinco tipos dessas tabelas:

- Tabela de Símbolos que representam pontos de cartas de papel;
- Tabela de Símbolos que representam pontos de apresentação simplificada (S-52);
- Tabela de Símbolos que representam linhas;
- Tabela de Símbolos que representam contornos plenos[15] de áreas ; e
- Tabela de Símbolos que representam contornos simbólicos[16] de áreas.

Cada linha da “Look-up Table”, chamada de entrada da tabela, contém o código da classe de objetos endereçada, sendo uma string constituída de uma combinação de valores de atributos, e instruções de simbologia ou uma chamada para um procedimento de simbologia condicional.

Para se encontrar a correta simbologia de uma instância de uma classe objeto, a “Look-up Table” recebe o código da classe de objetos em questão e valores de atributos relevantes na apresentação. As instruções de simbologia podem ser, assim, processadas pelo Gerador de Apresentação do sistema.

6.6.6 A Simbologia não Padronizada

Muitas vezes a navegação exige a utilização de símbolos que representam objetos, os quais não estão definidos no padrão S-57, por exemplo: uma linha que representa posições da

[15] Utilizado em escalas pequenas de apresentação.
[16] Utilizado em escalas grandes de apresentação.

embarcação ao longo do tempo ou mesmo pontos utilizados para representar posições passadas ou futuras. A Biblioteca de Apresentação vem com seu próprio catálogo de objetos de navegação, permitindo que, nestes casos, a representação seja feita por intermédio de símbolos, estilos de linha ou estilos de preenchimento.

6.7 O Sistema de Codificação de Cores

As cores utilizadas pela Biblioteca de Apresentação seguem uma codificação de cinco caracteres, que refletem sua utilização, como exemplo: pode-se ter CHMGD, que significa “chart magenta, dominant”. Estes nomes são usualmente chamados de símbolos de cores, que são usados por instruções de simbologia, símbolos, linhas e estilos de preenchimento de áreas, dando entrada na tabela de cores onde as cores são identificadas na forma de coordenadas CIE. A cor pode ser convertida para o sistema de coordenadas RGB, para finalmente ser apresentada em um monitor específico, levando-se sempre em conta as diferentes condições de iluminação na ponte da embarcação.

Os símbolos de cores são organizados em um esquema de cores que grupa os símbolos em seções de cores, as quais, por sua vez, contêm um conjunto de símbolos que são utilizados com um determinado propósito.

Mudanças no sistema de coordenadas de cores CIE são esperadas, com o acúmulo de experiência. Estas mudanças são relativamente simples de serem manipuladas. Já mudanças na organização do esquema de cores, embora venham a ser necessárias, devem ser evitadas, pois, seu custo é relativamente alto.

6.7.1 O Esquema de Cores

O apêndice B possui as tabelas com os códigos referentes aos esquemas de cores. Estes esquemas são separados em seções:

6.7.1.1 Seção de Uso Geral

As cores desta seção são utilizadas por todas as seções do esquema de cores, como exemplo tem-se:

- TRNSP - Significa uma cor cem por cento transparente. Um ponto com esta cor, quando se sobrepõe a outro com cor visível, não altera a aparência do segundo ponto, ou seja, comporta-se como invisível.
- NODTA – Significa sem dados. Este ponto não é coberto por informações provenientes de uma carta, nem por informações provenientes de outras fontes, como por exemplo: uma superposição de imagem radar no ECDIS. Pode ser chamada também de cor vazia de fundo.
- CURSR – Representa a cor do cursor. Na maioria dos sistemas o cursor pode ser tratado como um item autônomo, mantendo-se independente da área da carta a qual ele esteja se superpondo.

6.7.1.2 Conteúdo das Cartas

As cores desta seção se destinam à apresentação de cartas, mantendo sempre uma relação entre o espaço gasto com sua codificação (numero mínimo de bits) e flexibilidade suficiente para permitir futuras mudanças nas composições de cores. Algumas podem ser de uso geral, outras tem sua utilização restrita a propósitos específicos.

Como exemplo, pode-se ter:

- CHBLK, CHGRD, CHGRF, CHRED, CHGRN, CHYLW, CHMGD, CHMGF, CHBRN, CHWHT – utilizadas no projeto de símbolos, linhas e estilos de preenchimento de áreas, sendo que sua utilização não se aplica quando cores especiais são especificadas para a aplicação definida;
- OUTLW, OUTLL - utilizadas em simbologia de linhas externas, dependentes do fundo que elas delimitam (água/terra);
- LITRD, LITGN, LITYW – símbolos que representam luz. Possuem suas próprias cores, podendo ter suas características individuais alteradas conforme as circunstâncias;

- ISDGN – utilizada na representação de perigo isolado, que é configurado como um importante item dentro de um ECDIS, merecendo, assim, uma cor especifica;
- DNGHL – utilizada na representação de seleção efetuada pelo operador para elementos perigosos, durante planejamento de rotas;
- TRFCD, TRFCF – utilizadas na representação de rotas de tráfegos importantes, destacando-se das rotas de menor valor;
- LANDA - utilizada na representação de áreas de terra em geral;
- LANDF – utilizada na representação de atributos de áreas de terra;
- CSTLN – utilizada na representação de linhas de costa;
- RADHI, RADLO – utilizadas na representação de atributos de linhas de costa, para que se possa ter um controle total em quaisquer condições (dia/noite);
- SNDG1 – utilizada na representação de contatos sonar [17] que estejam mais fundos do que a profundidade segura;
- SNDG2 – utilizada na representação de contatos sonar que estejam dentro da profundidade segura;
- DEPSC – reservada para seleção de contorno seguro;
- DEPCN – utilizada na representação de todos os contornos de profundidade que não sejam contornos seguros;
- DEPDW, DEPMD, DEPMS, DEPVS, DEPIT – utilizadas para representar máscaras de profundidade, que podem ser:

    - Áreas mais profundas do que o contorno de profundidade selecionado pelo operador (DEPDW);
    - Áreas entre o contorno de profundidade e o contorno seguro selecionado pelo operador (DEPMD);
    - Áreas entre o contorno seguro e o contorno de águas rasas selecionado pelo operador (DEPMS);
    - Áreas entre o contorno de águas rasas e a linha de menor profundidade (DEPVS); e
    - Áreas entre o contorno de profundidade zero e a linha de costa (DEPIT).

[17] Alvos detectados pelo sonar, apresentados pelo sistema.

6.7.1.3 Sobreposição de Imagem Radar

6.7.1.3.1 Sobreposição Radar

Para uma imagem de sobreposição radar, pode-se ter uma cor para alta intensidade (RADHI) e uma cor para baixa intensidade (RADLO), podendo-se, ainda, utilizar uma interpolação entre estes dois valores, obtendo-se valores intermediários de intensidade.

Para alvos do tipo ARPA, utiliza-se uma simbologia de cor separada (ARPAT).

6.7.1.3.2 Imagem Radar Transparente

Opcionalmente, a imagem do radar pode passar de verde para transparente, sendo que esta transparência pode variar, alterando-se no pixel o seu percentual de transparência continua, que pode variar de 0% a 100%.

6.7.1.4 Informações de Navegação

Esta seção fornece cores para informações ao operador, utilizando SCLBR para representar a cor de geração da barra de escala, CHCOR para representar a cor de geração das correções manuais efetuadas nas cartas, e NINFO para representar a cor de geração das anotações de operador.

6.7.1.5 Futuras Aplicações

Esta seção fornece cores para requisitos futuros.

6.7.1.6 Rotas e Símbolo da Embarcação

Esta seção agrupa cores que se aplicam ao símbolo da embarcação e objetos associados a ela. Para vetores que representam curso e velocidade sobre a terra, utiliza-se a cor simbolizada por SHIPS. Os pontos históricos utilizam cores simbolizadas por PSTRK e SYTRK, sendo que as rotas planejadas são representadas por PLRTE e rotas secundárias por APLRT.

6.7.1.7 Interface do Usuário

Esta seção utiliza um grupo de dez símbolos de cores empregados na codificação da informação na interface com o usuário. As cores foram selecionadas de forma a garantir a visibilidade e a legibilidade da informação na área em questão, e, ao mesmo tempo, não desviar a atenção do operador das cartas. Certas cores, tais como UIBCK, mudam de branco, quando as cores de fundo são claras, para preto, quando as cores de fundo são escuras, provendo uma melhor adaptação quando as condições de luminosidade mudam da luz do sol para o escuro da noite.

6.8 A Linguagem Descritiva da Simbologia Vetorial

O formato utilizado pela Biblioteca de Apresentação para definir símbolos pontuais, estilos de linhas complexos e padrões de preenchimento de áreas, é descrito por uma linguagem que usa uma “pena” imaginária, que se movimenta por coordenadas cartesianas bi-dimensionais (x,y). Estas coordenadas podem assumir valores entre 0 e 32767 unidades, sendo que cada unidade representa 0.01mm. A origem das coordenadas (localização 0,0) encontra-se no extremo superior esquerdo do espaço.

O formato vetorial utiliza as seguintes instruções:

- ; - O ponto e virgula separa as instruções.Toda instrução deve terminar com um ponto e virgula;
- , - A virgula separa os parâmetros pertinentes a uma instrução;
- SP cor - Seleciona uma pena com a cor definida. O parâmetro cor é uma letra que identifica o símbolo de cor. A pena continua com a cor escolhida até que uma nova pena seja selecionada;
- ST transparência – Determina a transparência da cor que está selecionada no momento;
- SW espessura – Determina a dimensão da ponta da pena em unidades de 0.3 milímetros;
- PU coordenada-x,coordenada-y – Move a pena para a posição de coordenadas absolutas x,y;

- PD coordenada-x, coordenada-y – Abaixa a pena na posição atual e move-a para a posição de coordenadas absolutas x,y. Desta maneira, uma linha é desenhada usando a cor e a espessura correntes;
- CI raio - Desenha um círculo de raio especificado. A posição atual da pena é considerada o centro do circulo;
- AA coordenada-x, coordenada-y, ângulo – Desenha um arco baseado na posição atual da pena e o ponto central, com ângulo especificado. As coordenadas x e y especificam o centro do arco;
- PM n – Coloca o interpretador de comandos em modo de definição de polígono. Desta maneira, utilizando-se instruções PA,PD,CI e AA, que ficam armazenadas em um buffer do polígono, não são executadas até que o polígono esteja completamente definido;
- EP – Desenha o polígono que foi definido anteriormente pela instrução PM;
- FP – Preenche um polígono que tenha sido definido anteriormente pela instrução PM; e
- SC nome-simbólico, orientação – Chama uma definição de um símbolo, especificando se o mesmo vai ser desenhado em sua orientação original ou girado na direção da ultima instrução de movimentação.

6.8.1 Orientação e Tamanho de um Símbolo Vetorial

Para cada símbolo vetorial, a altura e a largura são dadas em unidades de 0.01mm, sendo que o tamanho do símbolo é proporcional à resolução de apresentação especificada no padrão S-52. Apenas o símbolo do próprio navio é apresentado em escala relativa à dimensão da embarcação.

Todo o símbolo possui seu ponto pivô definido como o ponto usado como referência para mudanças de escala e rotações. Quando o símbolo é apresentado, o ponto pivô é posicionado de forma que todos os elementos do símbolo fiquem dispostos geometricamente em relação a ele. A Figura 32 apresenta o ponto pivô.

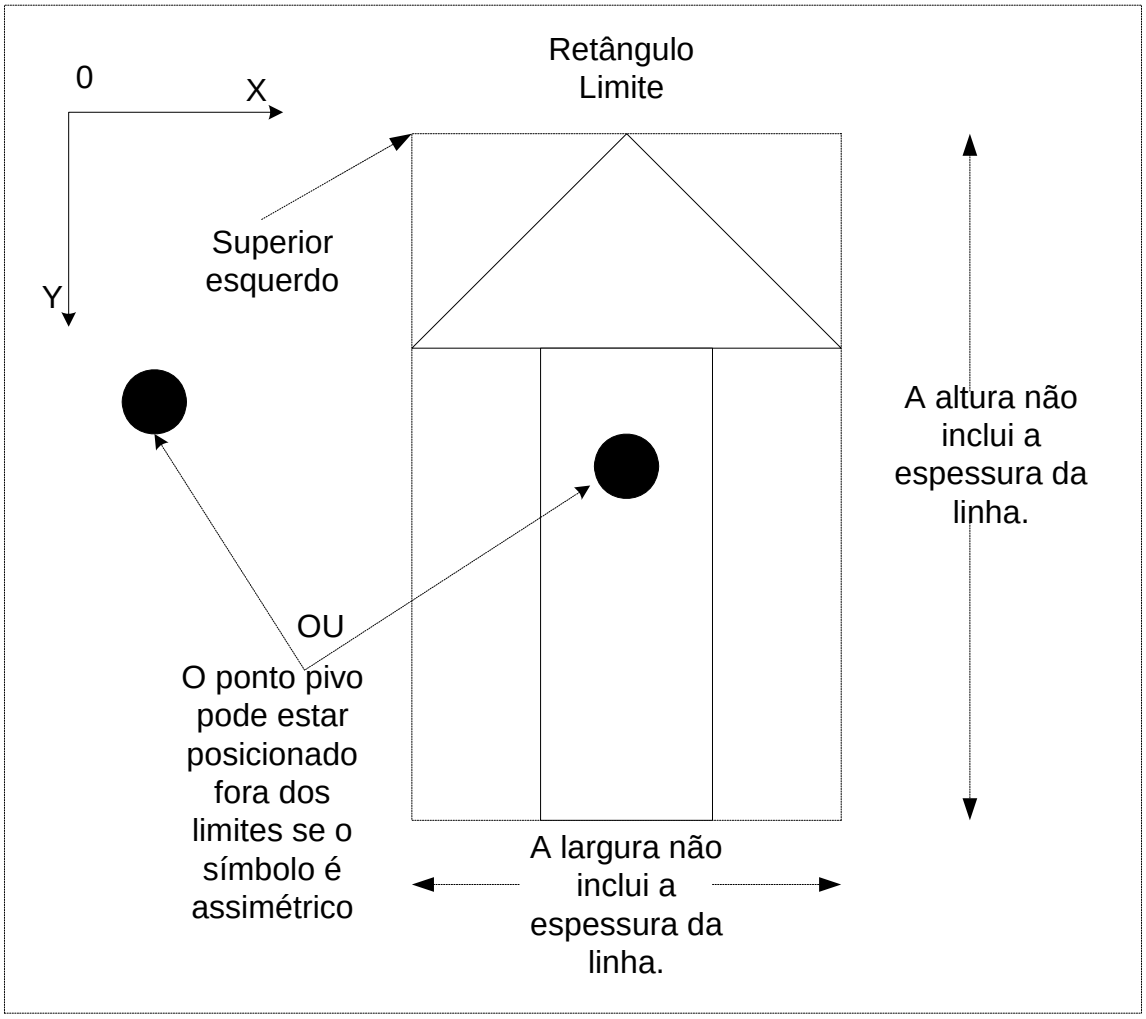


Figura 32 – Ponto Pivô

Extraída da especificação S-52 em http://www.iho.shom.fr./

6.8.2 Utilização de Estilo de Linha Complexo

O estilo de linha complexo é formado pela junção de símbolos. Ele possui um ponto pivô, que é utilizado como centro de rotações, sendo sua orientação dada pela direção entre dois vértices do objeto linha que será apresentado.

A Figura 33 apresenta o uso do estilo de linha complexo.

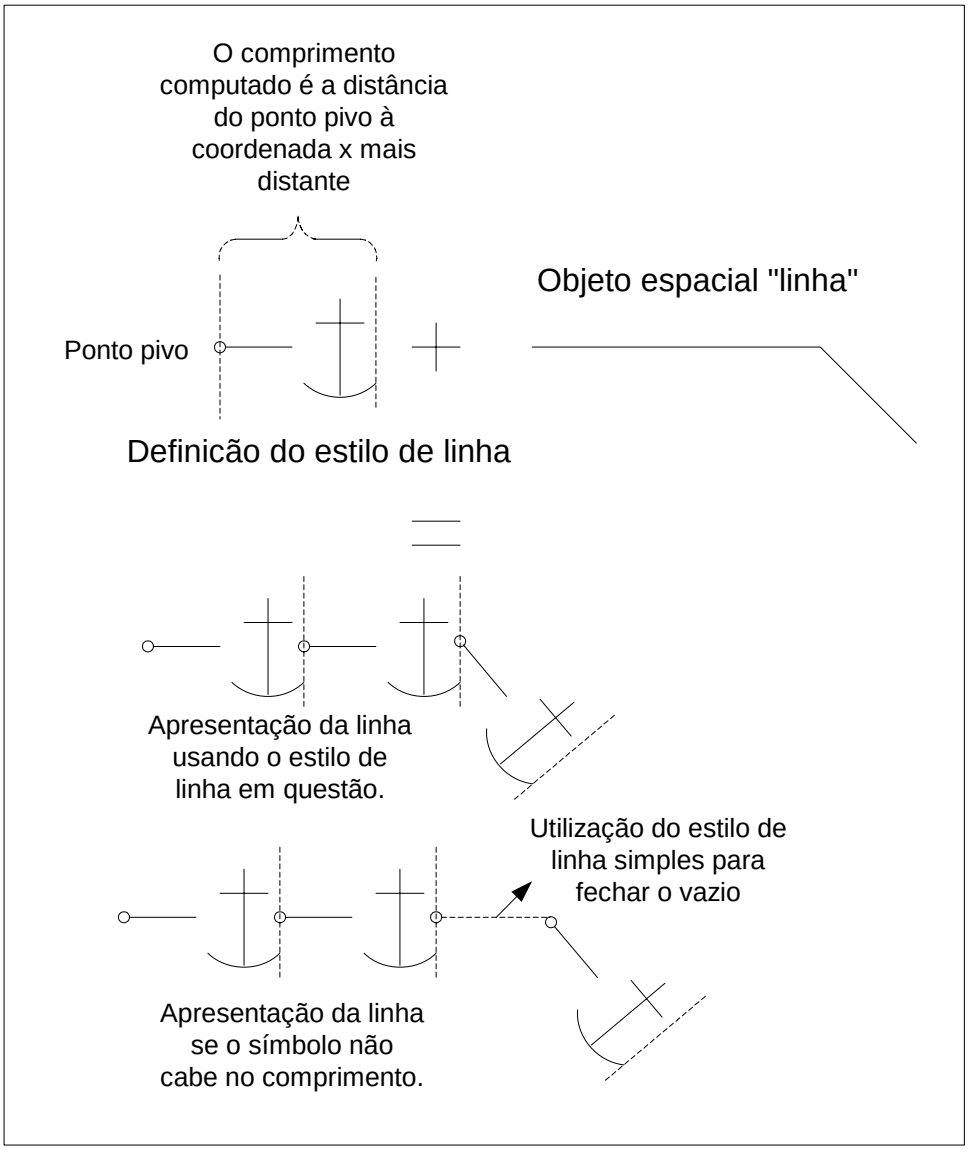


Figura 33 – Uso do Estilo de Linha Complexo
Extraída da especificação S-52 em http://www.iho.shom.fr./

6.8.3 Definições Simples no Formato Vetorial

Conforme visto, deve-se ter uma linguagem de descrição de simbologia vetorial que proverá, passo a passo, a elaboração da apresentação dos objetos pertencentes ao cenário que estará sendo exibido.

Observa-se os exemplos a seguir:

SPA;SW1;PU1000,1000;PD1000,2000;

Selecionar a pena “*A*”, com 1 x 0.3 mm, mover a pena para as coordenadas 1000,1000 sem desenhar a linha e, então, desenhar uma linha vertical deste ponto até a coordenada 1000,2000.

SPB;SW2;PU1000,1000;PD1000,2000,2000,2000,2000,1000,1000,1000;

Selecionar a pena “*B*”, com 0,6mm (2x0.3mm), mover a pena para as coordenadas 1000,1000 sem desenhar a linha e, então, desenhar um retângulo com canto superior esquerdo de coordenadas 1000,1000 e canto inferior direito de coordenadas 2000,2000.

SPB;ST2;PM0;PU1000,1000;PD1000,2000,2000,2000,2000,1000;PM2;FP;

Desenha o mesmo retângulo do exemplo anterior, porém agora definido como um polígono. Sua borda é automaticamente fechada pela instrução PM2. O polígono é preenchido com a cor da pena “*B*” e com transparência de 50%.

6.9 O Formato de Descrição de Símbolo no Padrão “RASTER”

Símbolos no padrão “raster” não são supridos pela Biblioteca de Apresentação. Desta forma, a implementação destes símbolos pode ser desenvolvida pelos fabricantes de ECDIS, respeitando sempre seus tamanho, cor e formato originais.

Como exemplo, pode-se ter um formato em que cada “pixel” é representado por uma letra, que no fundo especifica uma cor. Símbolos cujo código ASCII possuem valor superior a um determinado valor de código estabelecido, podem representar atributos tipo transparente (“pixel” invisível).

Na Figura 34 tem-se a representação de um símbolo âncora, onde a letra “*A*” representa a cor vermelha e a letra “*B*” representa a cor preta. O símbolo “@” foi utilizado para representar “pixels” invisíveis (transparência).

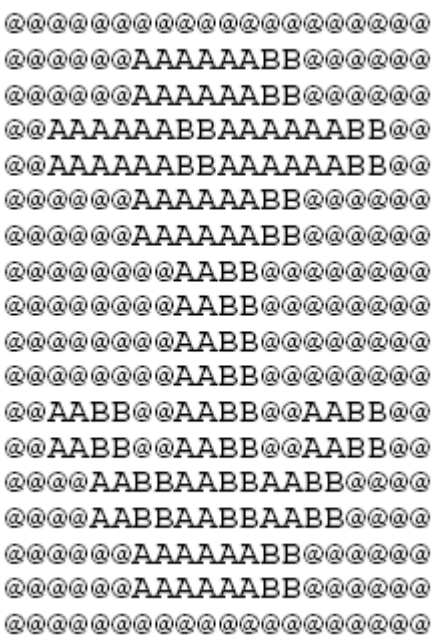


Figura 34 – Representação de Símbolos “Raster”
Extraída da especificação S-52 em http://www.iho.shom.fr./

O ponto pivô em símbolos “raster” é dado por um número de linha e por um número de coluna que servem apenas para posicionamento do símbolo, não sendo possível sua rotação.

6.10 Descrição das Instruções de Simbologia

Instruções de simbologia são utilizadas na “Look-up Table” para implementar a simbolização de objetos, resumindo-se em cinco instruções principais:

- SHOWTEXT – para apresentar os rótulos de texto;
- SHOWPOINT – para apresentar pontos e colocar símbolos dentro de áreas;
- SHOWLINE – para apresentar linhas e bordas de áreas;
- SHOWAREA – para apresentar áreas; e
- CALLSYMPROC – para chamar procedimentos de simbologia condicional.

As instruções de simbologia são formadas por comandos que, por sua vez, são constituídos de instruções que geram ações gráficas de baixo nível, tais como: “Encher uma área”, ”Desenhar uma linha”.

Os comandos possuem uma sintaxe definida pelo padrão apresentado na Figura 35.

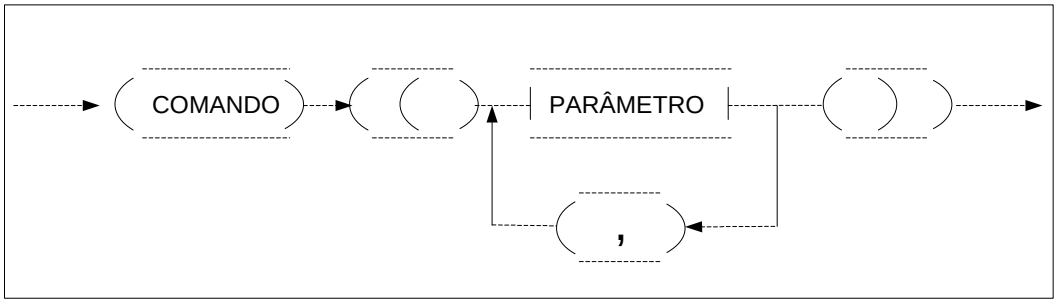


Figura 35 – Palavra de Comando de Simbologia
Extraída da especificação S-52 em http://www.iho.shom.fr./

CAPÍTULO 7

7 Exemplos de Utilização dos Padrões

A utilização dos padrões pesquisados neste trabalho vem, desde sua proposição, conduzindo a uma maior integração entre as diversas instituições responsáveis pelo desenvolvimento de sistemas de navegação em ambiente marítimo, proporcionando, assim, um caminho comum para a criação de sistemas confiáveis e padronizados, que podem ser avaliados e testados segundo normas bem definidas de avaliação, para que acidentes envolvendo embarcações, com prováveis prejuízos ambientais, sejam reduzidos, tornando possível uma navegação mais segura.

7.1 ECDIS e sua relação com os padrões

Os Sistemas de Navegação e Cartas Náuticas eletrônicas (ECDIS) passam a compor um cenário atuante de auxilio a navegação, uma vez que podem ser atualizados com freqüência. Esta atualização é feita através das Cartas Náuticas Eletrônicas, que possuem objetos relacionados ao mundo real, seguindo o padrão S-57 de modelagem, ficando seu empacotamento no formato de compatibilidade entre sistemas de plataformas diversas, o formato 8211,

Na utilização da base de dados comum, ou seja as cartas no formato S-57, esses sistemas podem contar com uma consistência na troca de informações, mantendo para tal uma estrutura interna em um formato favorável a recuperação da informação desejada.

No tocante a visualização os Sistemas de Navegação e Cartas Náuticas eletrônicas, devem manter uma semelhança que permita ao operador reconhecer com facilidade o ambiente tático, esta padronização refere-se ao padrão S-52, que alem de amarrar o formato e cor dos objetos apresentados, propõe uma biblioteca de implementação do cenário, proporcionando assim um caminho seguro de elaboração do elemento de visualização dentro do ECDIS, com um formato comum de apresentação que deve ser seguido independente da plataforma utilizada.

A figura 36 apresenta a forma com que os padrões apresentados pela IHO, interagem com a estrutura dos novos sistemas de navegação. A padronização conduzira a um nível de confiabilidade jamais atingido por Sistemas de Apoio a Navegação. Os sistemas antigos não possuíam meios eficientes de atualização das informações necessárias a navegação segura, nem mesmo a apresentação de forma dinâmica, provendo pouco auxilio em situações de risco real. Os novos sistemas baseados nos padrões propostos, conseguem apresentar uma grande flexibilidade e facilidade de operação pois sua interface é bem definida e sua estruturação é totalmente voltada para a realidade atual, onde as embarcações tornam-se cada vez maiores, e o fluxo de veículos torna-se mais complexo com o aumento das frota

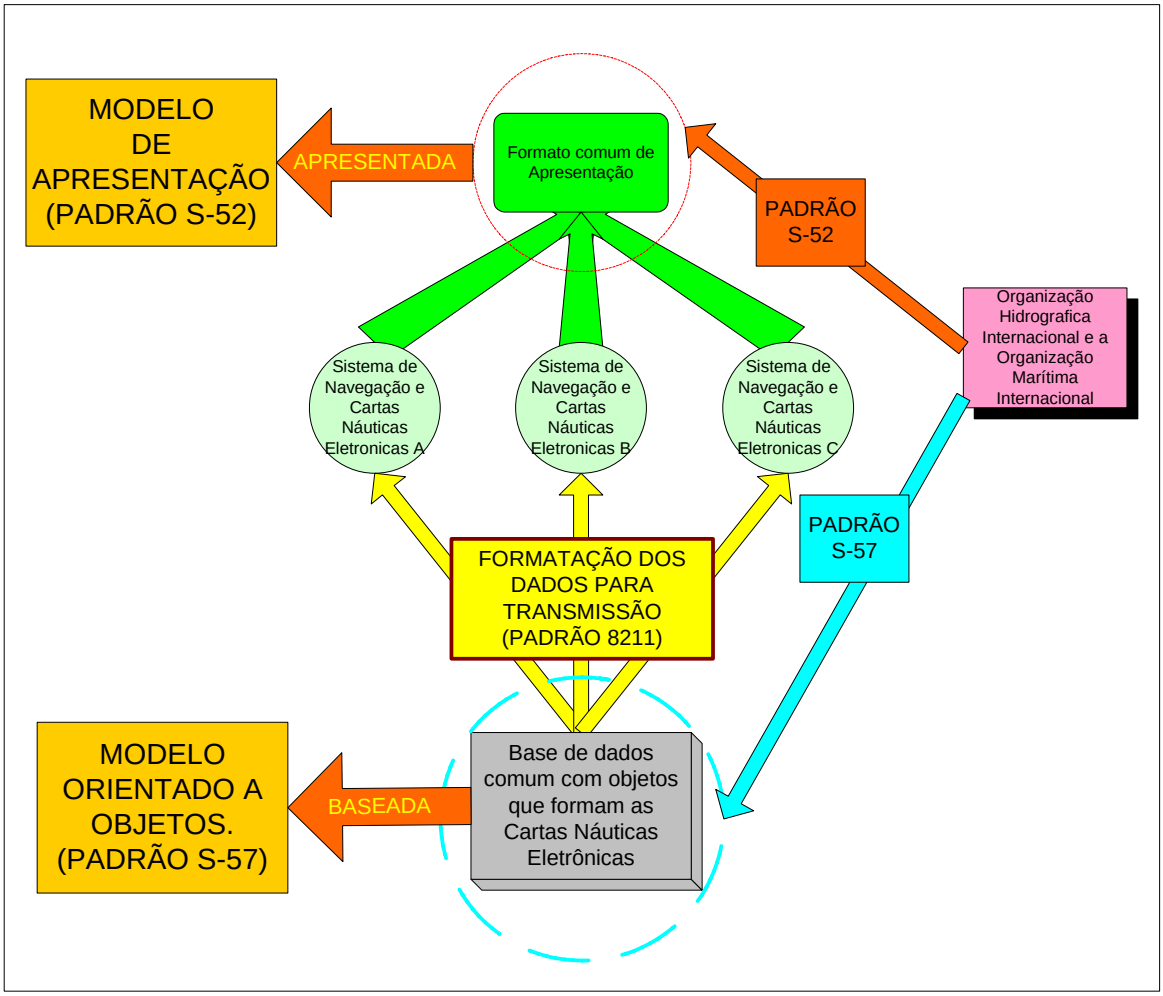


Figura 36 – Impacto dos Padrões Propostos na estrutura dos ECDIS

7.2 ECDIS, Algumas Implementações Existentes

Já existem no mercado ECDIS que foram desenvolvidos com a proposta de implementar e seguir os padrões de confiabilidade e interoperabilidade estabelecidos.

7.2.1 ECDIS ORCA Navigator

É um sistema de navegação, SEVENCS [21], que permite a utilização de cartas eletrônicas vetoriais diretamente obtidas da base de dados das companhias hidrográficas. É uma ferramenta de navegação eficiente devido à grande quantidade de informações atualizadas sobre obstáculos, informando ao operador, por voz, aproximações perigosas. Observa-se na Figura 37 a apresentação de uma carta eletrônica vetorial pelo sistema ORCA Navigator.

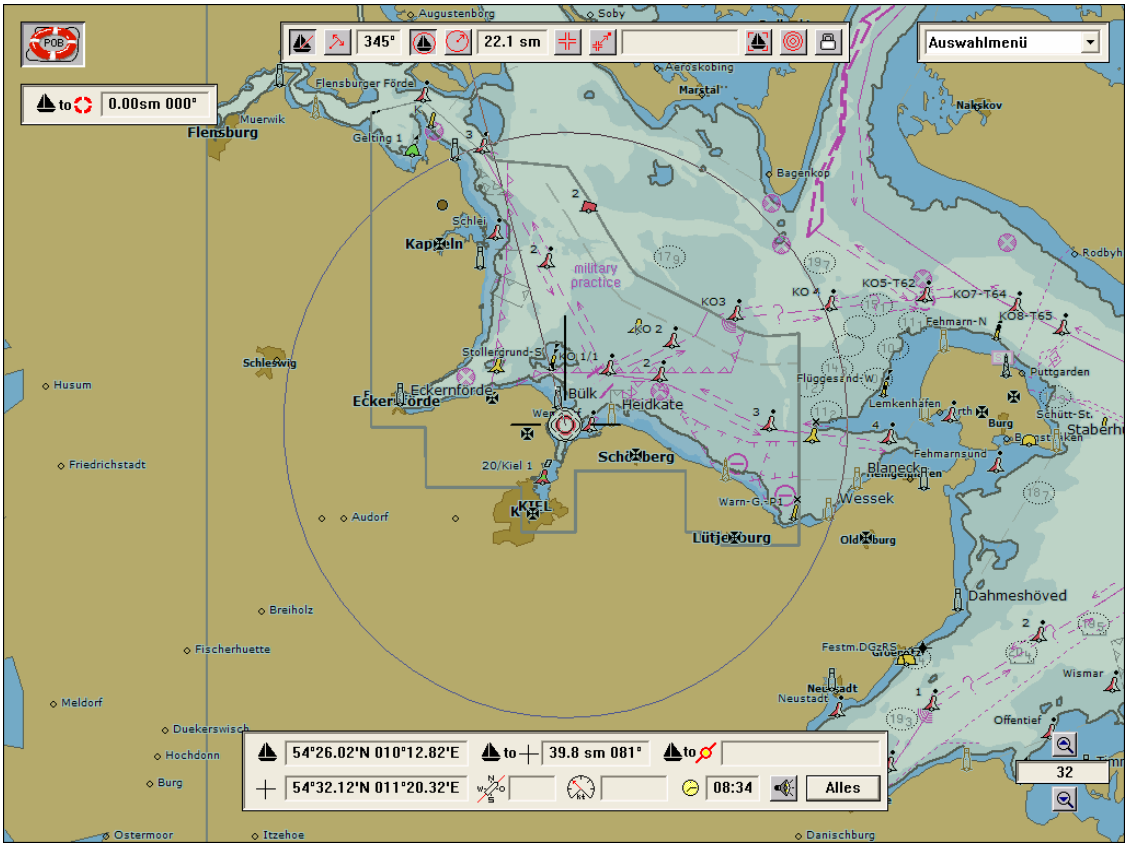


Figura 37 – Sistema ORCA Navigator
Extraída da imagem real da utilização da versão demo.

7.2.2 ECDIS Navi Sailor 3000

É um sistema de navegação baseado na cartografia eletrônica vetorial, TRANSAS [22]. Pode ser usado individualmente ou em conjunto com outras estações, aumentando, substancialmente, suas funcionalidades.

Características principais:

- Opera com a maioria dos formatos de cartas:
  - Cartas “raster”:
    - ✓ ARCS – Cartas britânicas;
    - ✓ NDI/BSB – Cartas americanas e canadenses; e
    - ✓ Seafarer – Cartas produzidas por companhias australianas.
  - Cartas Vetorias:
    - ✓ ENC – apresentação TX-97;
    - ✓ DNC – publicadas pela NIMA (USA); e
    - ✓ TX-97 – cartas vetoriais padrão da Transas, fabricante do Navi Sailor.
- Interface com grande variedade de sensores;
- Concebido segundo os padrões definidos para ECDIS;
- Fornece posicionamento em tempo real e monitoração de rotas;
- Fornece informações essenciais à navegação:
  - Posição do navio;
  - Informação de alvo ARPA;
  - Dados das interfaces dos sensores;
  - Fornece o contorno do navio na escala da carta em apresentação; e
  - Fornece informações sobre objetos perigosos nas cartas.

Pode-se observar na Figura 38 a versatilidade de conexões de um ECDIS Navi Sailor 3000.

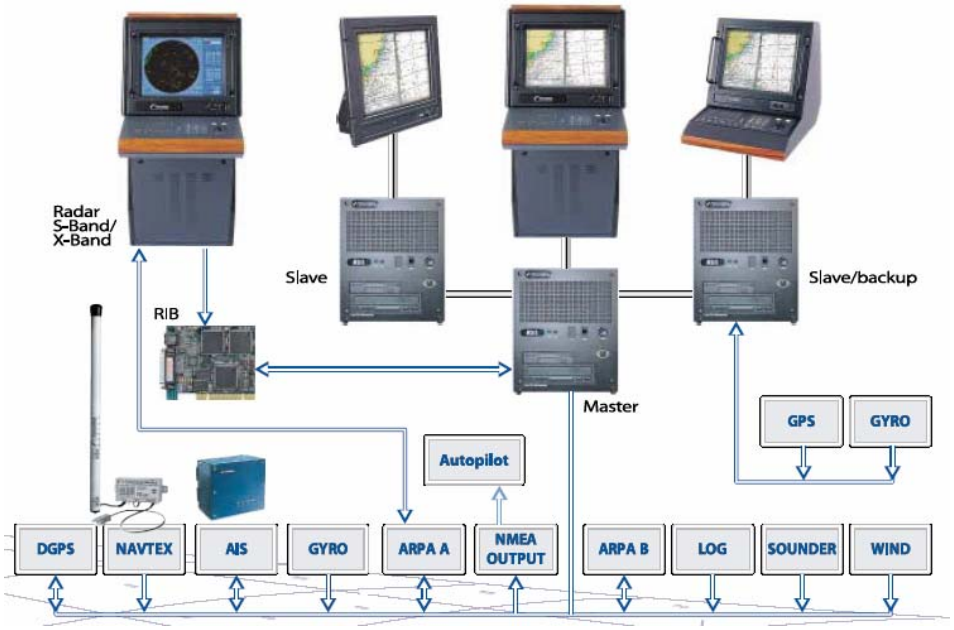


Figura 38 – Conexões de um ECDIS Navi Sailor 3000
Extraída do folheto de propaganda do equipamento.

7.2.3 ECDIS Tokimec EC-7500

É um sistema de navegação baseado na cartografia eletrônica vetorial, TOKIMEC [23], que possibilita a apresentação de cartas digitais, monitoramento de posição do navio, medição de posição de objetos, planejamento de rotas, monitoramento de navegação em rotas planejadas, sobreposição de imagem radar, dentre outras funções. A Figura 39 apresenta uma imagem deste modelo de ECDIS.

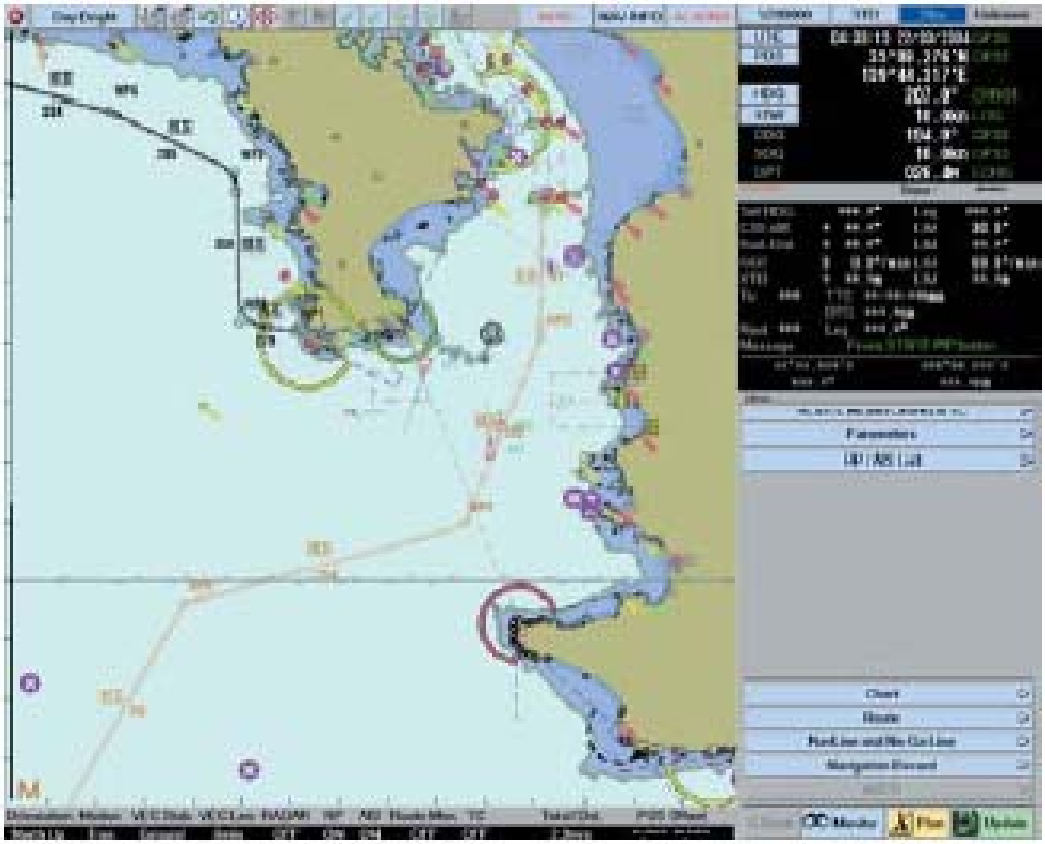


Figura 39 – ECDIS EC-7500 (Navegação)
Extraída da imagem real da utilização da versão demo.

7.2.4 ECDIS Dkart Navigator

É um sistema de navegação baseado na cartografia eletrônica vetorial, MADRES [24]. Foi certificado pela convenção de SOLAS, respeitando a resolução IMO A.694 para “software” de cartas marítimas. Possui algumas funções, tais como: piloto automático, visão tri-dimensional, sobreposição radar, monitoração de posição do navio, manipulação de cartas do tipo CM-93/3(C-Map), S-57, além de planejamento de rotas e outras funções de navegação, com todas as características que um ECDIS certificado deve possuir. A Figura 40 apresenta uma imagem deste modelo de ECDIS.

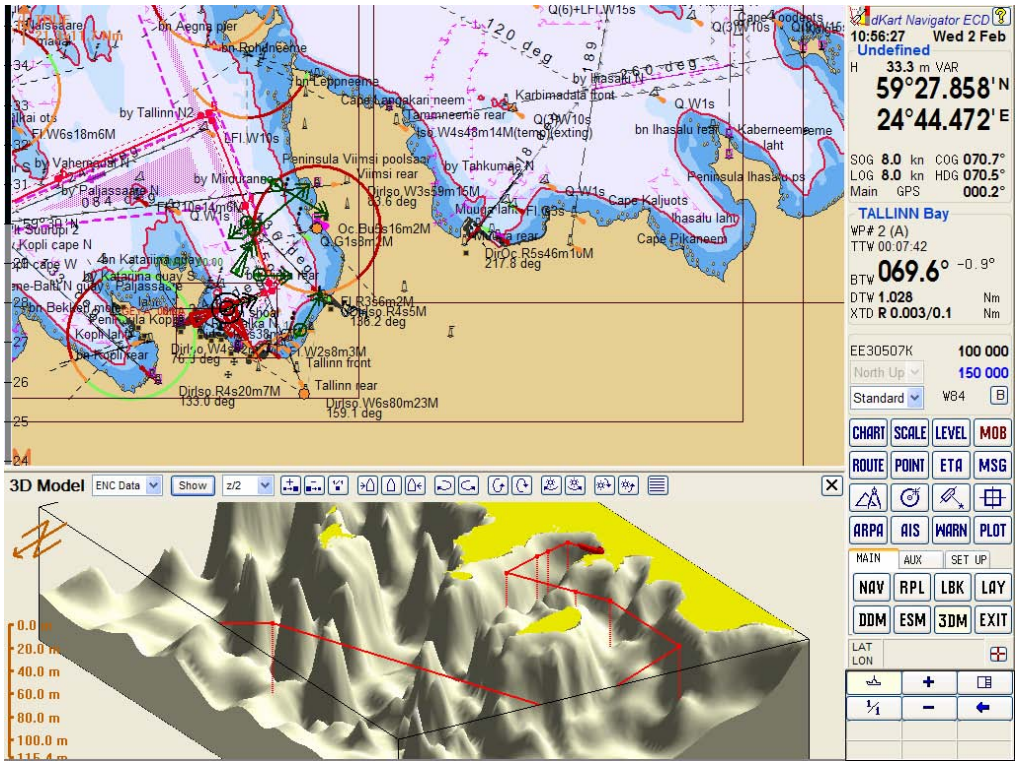


Figura 40 – ECDIS Dkart Navigator (Navegação)
Extraída da imagem real da utilização da versão demo.

7.2.5 ECDIS Navmaster Professional

É um sistema de navegação baseado na cartografia eletrônica vetorial, PCMARINE [25]. Foi desenvolvido segundo as especificações da IMO para ECDIS. Pode ser instalado em um computador PC padrão e conecta-se, via padrão NMEA, com instrumentos de bordo. Opera com ENCs oficiais, ARCS e cartas C-Map CM93/3. Sua imagem é apresentada na Figura 41.

Possibilita :

- Monitoramento da posição do navio e outras embarcações sobre as cartas eletrônicas;
- Planejamento e monitoração de rotas;
- Manutenção de cartas eletrônicas atualizadas;
- Geração de alarmes e advertências, segundo critérios do ECDIS;
- Apresentação de alvos ARPA; e

- Interface ajustável.

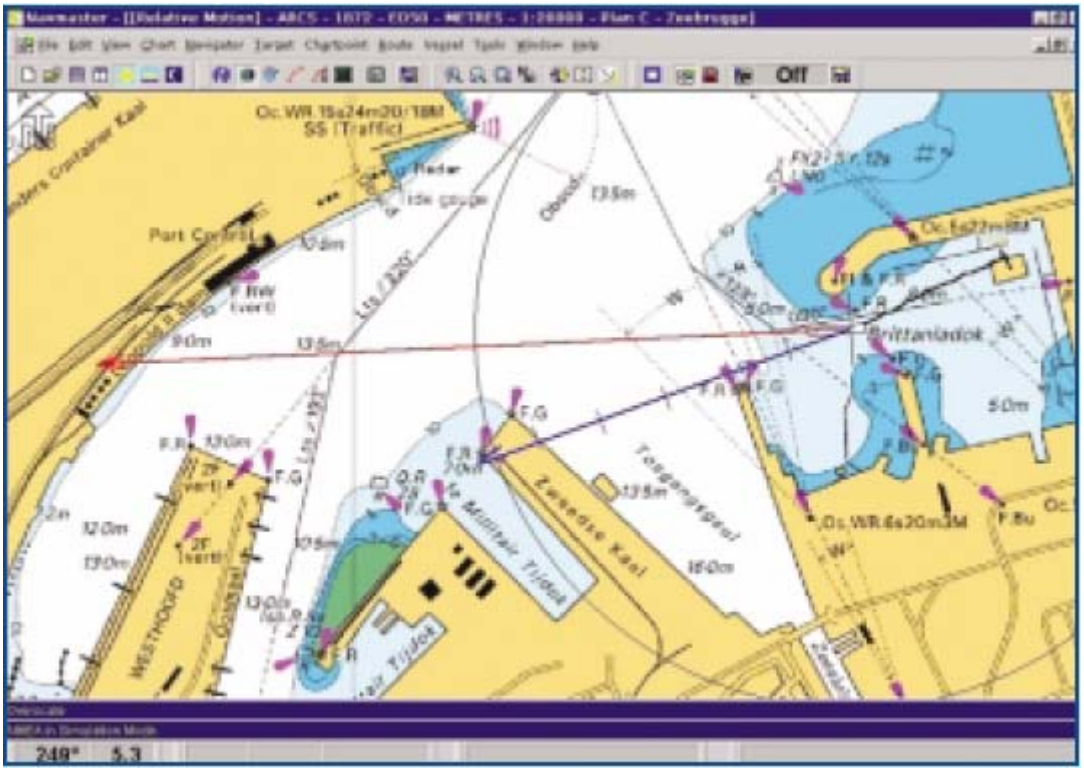


Figura 41 – ECDIS Navmaster Professional
Extraída da imagem real da utilização da versão demo.

7.3 Ferramentas de Desenvolvimento Disponíveis:

A utilização de sistemas de navegação marítimos, dentro dos padrões definidos para um ECDIS, conduzirá a uma revolução na navegação nas próximas décadas, permitindo assim, que as embarcações possam navegar com maior segurança. Com o intuito de agilizar a fabricação desses sistemas, alguns fabricantes estão disponibilizando ferramentas destinadas a ajudar, de forma mais rápida e eficiente, na produção e na manutenção do “software” necessário às suas implementações.

7.3.1 ECDIS Kernel

Desenvolvido pela empresa SevenCs, SEVENCS [21], o ECDIS Kernel EC2007 é um pacote de “software” para desenvolvimento, que permite que as companhias desenvolvam

seus próprios sistemas ECDIS. É composto por centenas de componentes de “software”, que se encontram de acordo com as normas de padrões da IMO para ECDIS (S-52/S-57). Possui as funções básicas e o suporte para leitura, apresentação e alteração de cartas digitais, fornecidas pelas Companhias Hidrográficas ao redor do mundo, permitindo cálculos de navegação, leitura de dados de sensores e fornecendo, também, uma interface com o usuário, diminuindo, assim, o tempo necessário para o desenvolvimento.

Características:

- Permite importar cartas de órgãos oficiais e privados;
- Apresenta cartas do tipo ARCS e “raster” BSB;
- Lida com vários tipos de projeção;
- Manipula cartas vetoriais do tipo S-57;
- Permite atualizações das cartas do tipo S-57;
- Permite escolha de área de operação;
- Permite impressão das cartas;
- Permite consultas em dados das cartas;
- Permite desenhar sobre as cartas os objetos do operador;
- Permite projetar rotas e verificá-las para averiguação de perigos;
- Permite monitorar e gravar o deslocamento da embarcação;
- Permite ler informações de sensores no formato NMEA;
- Apresenta alvos AIS e ARPA; e
- Permite monitorar as colisões.

7.3.2 ENC-X

Desenvolvido pela empresa CARIS, CARIS [26], permite a elaboração de sistemas com a capacidade de leitura de cartas no formato S-57, apresentando-as segundo o padrão S-52. Possibilita a apresentação de arquivos nos padrões Shapefiles, BSB, TIFF e imagens no formato GeoTIFF, propiciando, ainda, a adição de símbolos, linhas, polígonos e texto por sobreposição. A Figura 42 apresenta esta ferramenta.

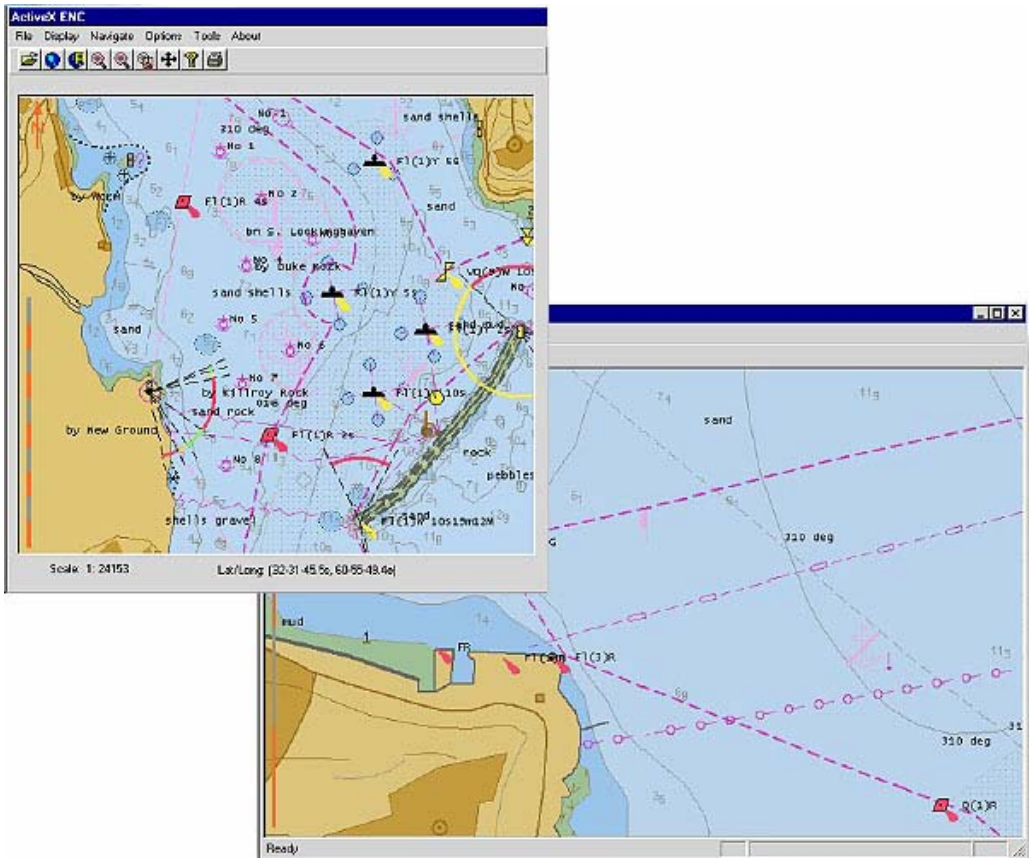


Figura 42 – CARIS ENC-X
Extraída do folder de propaganda do produto.

Proporciona uma facilidade de desenvolvimento de vistas, oferecendo uma extensa gama de funcionalidades:

- Pan;
- Zoom;
- Apresentação de diversas cartas na mesma janela; e
- Consultas com limites definidos pelo operador.

Confeccionada com uma arquitetura aberta, sendo uma ferramenta baseada em “ActiveX", sua utilização é compatível com ferramentas de desenvolvimento como: Visual Basic, Visual C++ e Delphi.

7.3.3 Openev – Biblioteca de Visualização de Dados Rasterizados e Vetoriais, sob os Termos de Licença GNU.

Ferramenta interativa, OPENEV [27], que possui características de:

- Rodar em plataformas populares;
- Manipular dados vetoriais 2D/3D;
- Manipular grande conjunto de dados rasterizados;
- Proporcionar múltiplas vistas do mesmo conjunto de dados, provendo diferentes interpretações;
- Possibilitar a apresentação de múltiplos conjuntos de dados em uma única visualização;
- Possibilitar interpretação de informação georreferenciada, propiciando mudanças de projeção, se necessárias;
- Possibilitar impressão;
- Possibilitar funções de manipulações de visualização (pan, zoom, rotação); e
- Utilizar a aceleração de "hardware", se presente.

7.3.3.1 Biblioteca libs52

A libs52 implementa a Biblioteca de Apresentação baseada no padrão S-52, revisão 3.2[18]. É parte integrante do projeto opENCview, destinado à elaboração de um visualisador de cartas náuticas eletrônicas (ENC), baseado na filosofia de "software" livre.

7.3.3.1.1 Especificação

Todas as especificações utilizadas na elaboração da biblioteca estão disponíveis na Internet para a aquisição gratuita, como apresentado na Tabela 3.

[18] (www.iho.shom.fr/general/ecdis/Cs_md3.pdf).

Organização	Especificação	Local / Local de obtenção
IHO	S-52	IHO S-52 USERS' MANUA , Ed. 3.2 www.iho.shom.fr/general/ecdis/pslb03_2.pdf
IHO	S-57	S-57 Appendix B.1 www.iho.shom.fr/general/ecdis/20SOC.pdf
OpenGIS	OpenGIS	OpenGIS especificação abstrata www.opengis.org/techno/specs.htm
SGI	OpenGL	Especificação OpenGL www.opengl.org/developers/documentation/index.html
X.Org	X Windows Sys	Sistema X Windows www.x.org

Tabela 3 – Especificações Utilizadas na Elaboração da Biblioteca libS52

7.3.3.1.2 Composição

A biblioteca é composta por:

- Módulo de manipulação dos dados S-57;
- Módulo para criação de apresentação no padrão S-52;
- Módulo de interface com a biblioteca gráfica de apresentação OpenGL; e
- Módulo de interface com a biblioteca Openev.

O arquivo S57data.c, apresentado no Anexo A, dá uma idéia de implementação da interface com os dados S-57.

CAPÍTULO 8

8 Conclusões

Muito se tem feito para a obtenção de uma padronização mundial dos sistemas de informações navais. Inúmeros esforços têm sido aplicados para tornar as fontes de dados, ou seja, as cartas eletrônicas, uma fonte confiável que deve sofrer constantes atualizações.

O mundo real possui uma grande variedade de elementos que devem ser modelados e representados nos SIGs. Suas geometrias e relacionamentos topológicos requerem conceitos complexos para sua representação.

Foi com o intuito de facilitar o desenvolvimento, que a Organização Hidrográfica Internacional, utilizando-se da filosofia OO, elaborou os modelos presentes no padrão S-57, padronizando, também, as formas de apresentação destes objetos.

Diversas companhias do mercado de navegação adiantaram-se e desenvolveram sistemas capazes de implementar os requisitos requeridos por estas proposições. O mundo foi dividido em regiões, ficando, assim, a cabo das nações responsáveis pelas áreas em questão o desenvolvimento das Cartas Náuticas Eletrônicas, as quais farão parte do banco de dados de navegação utilizado pelos sistemas que, no futuro, substituirão a navegação tradicional.

A divisão ficou a cargo do consórcio constituído dos países membros, sendo que no caso brasileiro, a Marinha do Brasil, especificamente a Diretoria de Hidrografia da Marinha, assumiu a tarefa de confecção das cartas.

O Brasil encontra-se em um estágio embrionário no estudo destes padrões e na confecção de sistemas que possam se comunicar com os ECDISs elaborados na padronização internacional. Este estudo pode funcionar como ponto de partida para o desenvolvimento de um ECDIS brasileiro independente de patentes estrangeiras, contando com a experiência oferecida pelas ferramentas de código aberto encontradas ao longo da pesquisa, que visou achar alternativas de implementação. Neste sentido foram criados e discutidos meios de implementação, que levam a confecção de um visualisador de Cartas S-57, com apresentação no padrão S-52.

Verificou-se que tais ferramentas, se forem utilizadas, necessitam ainda de melhorias e adaptações, para que se possa ter como produto um sistema confiável e compatível com os sistemas já desenvolvidos por grandes companhias internacionais.

8.1 Dificuldades Encontradas

O Brasil não possui trabalhos significativos na confecção de Sistemas de Informações Navais e pouco material de pesquisa existe disponível.

As grandes empresas internacionais não disponibilizam informações substanciais que possam tornar-se úteis na elaboração de sistemas que utilizem como base os padrões apresentados. A coleta de material, utilizado como fonte para a confecção deste trabalho, tornou-se um desafio ao não ser possível contar com o auxílio daqueles que detêm uma coletânea de elementos importantes para a implementação, ainda que embrionária, de um ECDIS genuinamente brasileiro.

Apesar das dificuldades encontradas, o trabalho de pesquisa exaustivo proporcionou a elaboração de uma proposta de implementação baseada em fontes de “software” aberto, que abrange a leitura de dados no padrão 8211, até a apresentação dos objetos do modelo S-57, no padrão S-52.

8.2 Contribuições

Este trabalho visou apresentar uma visão detalhada dos padrões propostos para a troca de informações entre sistemas de informação naval, abordando, também, uma forma padronizada de apresentação que já está sendo utilizada, em larga escala, pelas companhias de desenvolvimento de ECDIS ao redor do mundo, apresentando uma proposta de elaboração baseada em software aberto. Ressalta-se que este trabalho baseou-se em uma pesquisa detalhada em vários livros e artigos, principalmente levando-se em conta que:

- Não existe em âmbito nacional nenhum estudo detalhado sobre o assunto, este trabalho torna-se um documento único que poderá servir de base para estudos e

implementações necessárias dos modernos Sistemas de Informações e Apresentação de Cartas Náuticas Eletrônicas, pois como todos os países que precisam manter sua soberania no mar, o Brasil terá que se adaptar a esta nova realidade dentro do contexto mundial;

- Foram apresentados, inicialmente, conceitos importantes para o entendimento da arquitetura do modelo de dados S-57;
- Foi apresentado o modelo de dados S-57, com a intenção de prover informações suficientes para entender-se suas relações como os objetos do mundo real;
- Foi apresentado, com detalhes, o padrão de encapsulamento de dados para a transmissão, o padrão 8211, cujo entendimento se torna fundamental para a leitura das informações que os sistemas devem obter e converter para suas bases de dados proprietárias;
- Foi apresentada uma proposta de implementação de leitura para os dados no formato S-57, encapsulados no padrão 8211, sob a forma de uma biblioteca que pode servir de base para a implementação do módulo de leitura das cartas em um sistema de informações navais (ECDIS);
- Foi apresentado, com detalhes, o padrão de apresentação S-52, que deverá servir de base para dar entendimento à implementação da biblioteca de visualização; e
- Foi apresentada uma proposta de implementação de uma biblioteca de código aberto, para a visualização dos objetos modelados no padrão S-57 e apresentados segundo o padrão S-52. Esta biblioteca poderá servir de base para a elaboração de um visualizador de cartas neste padrão dentro do ECDIS, provendo, assim, um formato correto de visualização, que deve obedecer aos padrões internacionais estabelecidos.

8.3 Trabalhos Futuros

Este trabalho de pesquisa teve como finalidade um estudo das novas propostas de padronização de transmissão e apresentação de dados em ambiente marítimos, para que o desenvolvimento de sistemas capazes de manipular tais dados possam se tornar uma realidade dentro da Marinha do Brasil, provendo aos futuros SIGs navais capacidades tais como: controle de tráfego, navegação, avaliação de situações táticas, utilizando os dados geográficos fornecidos.

A Marinha do Brasil, deverá , então:

- Trabalhando em cima da proposição de implementação de uma biblioteca de leitura, desenvolver uma interface consistente de obtenção dos dados;
- Desenvolver uma estrutura de armazenamento para os dados, de forma a prover uma recuperação eficiente da informação armazenada, utilizando-se, ou não, um SGBD para tal;
- Desenvolver a implementação da biblioteca de apresentação baseada nas características comuns que os sistemas deverão possuir para se enquadrarem nos padrões propostos, podendo-se utilizar como base a OpenEV e a Libs52;
- Finalmente, com todos os itens satisfeitos, prover um ECDIS genuinamente nacional que, no futuro, poderá prover ainda, mediante novas pesquisas, desenvolvimento de uma interface para web.

8.4 A Tendência Futura dos Padrões

Segundo S57EDIC4 [20], os padrões utilizados até o momento devem evoluir para que se possa obter uma expansão da representação dos tipos de dados hidrográficos, não limitando-se, desta forma, aos tipos utilizados exclusivamente para a codificação de ENCs, necessárias aos ECDIS.

A próxima edição do padrão S-57 (Edição 4) deverá estar baseada nos padrões geoespaciais ISO, fornecendo uma reformulação e evitando as atuais limitações existentes na edição 3.1. Esta nova versão não será uma revisão incremental das edições anteriores, possuindo, além de um conteúdo estendido, um novo formato de troca de dados.

Observa-se algumas limitações na versão 3.1, tais como:

- Foi, inicialmente, desenvolvida com a proposição de satisfazer as necessidades dos ECDIS de padrão IMO;
- Possui um regime de manutenção que obriga o congelamento do padrão por um período de tempo;
- A estrutura atual não permitirá requerimentos futuros (ex: grade de batimetria);

- O encapsulamento do modelo de dados restringe a flexibilidade da utilização de uma variedade de mecanismos de transferência ; e
- Sua utilização foi baseada, exclusivamente, no propósito de produção e troca de dados das Cartas Náuticas Eletrônicas (ENC).

A próxima edição da S-57 deverá permitir o suporte de uma grande variedade de dados digitais hidrográficos, provenientes de várias fontes e produtores, incluindo dados no formato “raster” e matricial, alem de variações 3D e temporais (x, y, z e tempo), permitindo classificações, expandindo o horizonte dos ECDIS para a web e possibilitando a consulta, a análise e a transferência de dados.

Estas mudanças são apenas previsões, não existe nenhum informe oficial sobre alteração dos padrões atuais que, estarão ainda, sendo utilizados por um longo período de tempo. A próxima edição da S-57 só deverá estar pronto no final de 2006.

REFERÊNCIAS BIBLIOGRÁFICAS:

[1] CAMARA G, Bancos de Dados Geográficos, Representação Computacional de Dados Geográficos, Gilberto Câmara e Antônio Miguel Vieira Monteiro. http://www.dpi.inpe.br/gilberto/livro/bdados/index.html, Arquivo cap1-repres.pdf, acessado em janeiro de 2004.

[2] MITROVIC, A., 1996. OO paradigm meets GIS: a new era in spatial data management. Invited paper presented at YUGIS'96, pp 141-148, Belgrade, Yugoslavia, Mar.

[3] UML – The Current UML Specification Version 2.0, 2004, (http://www.uml.org/), acessada em maio de 2004.

[4] RUMBAUGH, J., 1991. Object-Oriented Modelling and Design. 1 ed. Addison-Wesley, Englewood Cliffs, N.J., USA.

[5] JACOBSON, I., MAGNUS, C., PATRIK, J., et al, 1992. Object-Oriented "software" *Engineering: A Use Case Driven Approach. 1 ed. Addison-Wesley, Reading, MA, USA.*

[6] BOOCH, G., 1994. Object-Oriented Analysis and Design with Applications. 2 ed. Benjamin/Cummings Publishing Company, Redwood City, CA, USA.

[7] FERREIRA, Marcelo; JARABECK, Flávio, 1991. Programação Orientada ao Objeto com Clipper, São Paulo, Makron Books.

[8] CÂMARA, G. CASANOVA, M.; HERMELY, A. S, 1996. Anatomia de Sistemas de Informação Geográfica. 1 ed. SBC, Escola de Computação, Campinas, SP, Brasil.

[9] WOODSFORD, Peter A., 1995. The Significance of Object-Orientation for GIS. IUSM Working Group on GIS/LIS, Hannover, September 25-28.

[10] ORACLEESPACIAL, 2002 Manual do Usuário e Referência do Oracle Spatial Versão 9.2, documento Oracle_Espatial.pdf,março de 2002, acessada em julho de 2004.

[11] MOREIRA, Maurício A., 2001. Fundamentos do Sensoriamento Remoto e Metodologias de Aplicação, 1 ed, São Jose dos Campos, SP.

[12] BURROUGH, P., 1986. Principles of Geographical Information Systems for Land Resources Assessment. Clarendon Press.

[13] INPE, 2004. Fundamentos de Geoprocessamento, Banco de Dados Geográficos, Divisão de Processamento de Imagens. (http://www.dpi.inpe.br/inpe/dpi/cursos/ fundamentos/geo2-bancodados_arquivos/frame.htm#slide0432.htm), acessada em julho de 2004.

[14] SOLAS., 1986. International Convention for the Safety of Life at Sea; Consolidated text of 1974 SOLAS Convention, the 1978 SOLAS Protocol, the 1981 and 1983 SOLAS Amendments. International Maritime Organization, London, 1 July 1986.

[15] IMO, 1995. Performance Standards for Electronic Chart Display and Information Systems (ECDIS), IMO Resolution A.817(19), International Maritime Organization, London, 23 November .

[16] KEY, 2001. Safety at Sea ECDIS Conference. by Kay Faulkner of PC Maritime. Rotterdam,September.(www.pcmaritime.co.uk/comm/library/ECSPortableApproach_0901.pd), acessada em Janeiro de 2004.

[17] IHB 1996, “IHO Transfer Standard for Digital Hydrographic Data, Edition 3.0”. Special Publication 57 (S57), International Hydrographic Bureau, Monaco

[18] PGHARDY. 1998, “S57 ECDIS DATA PRODUCTION AND UPDATE USING AN OBJECT – ORIENTED SPATIAL DATABASE “ , por P.G.Hardy em Ecdis98c.pdf acessada em novembro de 2003

[19] KIAN. Geographic information – Imagery and gridded data, by Kian Fadaie, arquivo 13007.pdf de: http://ess.nrcan.gc.ca/esic/ccrspub-cctpub/pdf/13007.pdf , acessada em março de 2005.

[20] S57EDIC4. S-57_Ed4_Information_Paper.pdf www.iho.shom.fr/ECDIS/S-57_Ed4_Information_Paper.pdf acessada em janeiro de 2005.

[21] SEVENCS – http://www.sevencs.com

[22] TRANSAS – http://www.transas.com

[23] TOKIMEC – http://www.tokimec.co.jp/marine

[24] MADRES – http://www.madres.com

[25] PCMARINE – http://www.pcmarine.co.uk

[26] CARIS – http://www.caris.com

[27] OPENEV - http://cvs.sourceforge.net/viewcvs.py/openev/#dirlist

ANEXOS

ANEXO A

Listagem da classe de visualização dos elementos que constituem o arquivo S-57 encapsulado no padrão 8211. A aplicação principal carrega uma carta e chama a biblioteca de leitura, que promove o desempacotamento e a apresentação dos dados. A carta utilizada foi a carta da Baía de Guanabara, confeccionada nos padrões internacionais S-57 pela Marinha do Brasil.

```
// Leitura8211.cpp : Define o ponto de entrada para o console da aplicação.
//
/*
******************************************************************************
 * $Id: 8211view.cpp,v 1.7 2003/08/21 21:22:46 warmerda Exp $
 *
 * Project:  SDTS Translator
 * Purpose: Programa exemplo de dumping de dados  em 8211 na saída padrão.
 * Author:   Frank Warmerdam, warmerda@home.com

******************************************************************************
 * Copyright (c) 1999, Frank Warmerdam
  * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this “software” and associated documentation files (the “software”),
 * to deal in the “software” without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the “software”, and to permit persons to whom the
 * “software” is furnished to do so, subject to the following conditions:
  * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the “software”.
  * THE “SOFTWARE” IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
```

```
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE “SOFTWARE” OR THE USE OR OTHER
 * DEALINGS IN THE “SOFTWARE”.
**************************************************************************
******
  * $Log: 8211view.cpp,v $
 * Revision 1.7  2003/08/21 21:22:46  warmerda
 * Special reporting facilities for the S-57 LNAM and NAME bitfields (provided
 * by Rodney Jensen).
 * Report DDFInt fields as unsigned if they come from an unsigned binary
 * field.
  * Revision 1.6  2001/07/18 04:51:57  warmerda
 * added CPL_CVSID
  * Revision 1.5  2000/06/16 18:02:08  warmerda
 * added SetRepeatingFlag hack support
  * Revision 1.4  1999/11/18 19:03:04  warmerda
 * expanded tabs
  * Revision 1.3  1999/05/06 14:25:15  warmerda
 * added support for binary strings
  * Revision 1.2  1999/04/28 05:16:47  warmerda
 * added usage
  * Revision 1.1  1999/04/27 22:09:30  warmerda
 * New
  */
#include <stdio.h>
#include "stdafx.h"
#include "iso8211.h"
#include "cpl_port.h"
```

```
CPL_CVSID("$Id: 8211view.cpp,v 1.7 2003/08/21 21:22:46 warmerda Exp $");
static void ViewRecordField( DDFField * poField );
static int ViewSubfield( DDFSubfieldDefn *poSFDefn,
                         const char * pachFieldData,
                         int nBytesRemaining );
/*
************************************************************************/
/*                              main()                                  */
/*
************************************************************************/
int main( int nArgc, char ** papszArgv )
{
    DDFModule   oModule;
    const char  *pszFilename = NULL;
    int         bFSPTHack = FALSE;
    for( int iArg = 1; iArg < nArgc; iArg++ )
    {
        if( EQUAL(papszArgv[iArg],"-fspt_repeating") )
            bFSPTHack = TRUE;
        else
            pszFilename = papszArgv[iArg];
    }
     pszFilename = "BR511512.000"; //Carregando carta S-57 da Baia de Gauanabara
    if( pszFilename == NULL )
    {
        printf( "Usage: 8211view filename\n" );
        exit( 1 );
    }

/* -------------------------------------------------------------------- */
/*     Abre o arquivo, os eeros são reportados em stderr */
/* -------------------------------------------------------------------- */
    if( !oModule.Open( pszFilename ) )
    {
```

```
        exit( 1 );
    }
    if( bFSPTHack )
    {
        DDFFieldDefn *poFSPT = oModule.FindFieldDefn( "FSPT" );
        if( poFSPT == NULL )
            fprintf( stderr,
                    "unable to find FSPT field to set repeating flag.\n" );
        else
            poFSPT->SetRepeatingFlag( TRUE );
    }
/* ---------------------------------------------------------------- */
/*     Repete lendo registros até não existirem mais                */
/* ---------------------------------------------------------------- */
    DDFRecord   *poRecord;
    int         iRecord = 0;
    while( (poRecord = oModule.ReadRecord()) != NULL )
    {
        printf( "Record %d (%d bytes)\n",
                ++iRecord, poRecord->GetDataSize() );
        /* ------------------------------------------------------- */
        /*      Percorre cada campo de um registro                 */
        /* ------------------------------------------------------- */
        for( int iField = 0; iField < poRecord->GetFieldCount(); iField++ )
        {
            DDFField    *poField = poRecord->GetField( iField );
                        for( long int tempo=0; tempo<100000000; tempo++){
                        };
            ViewRecordField( poField );
        }
    }
}
/*
**********************************************************************/
```

```
/*                      ViewRecordField()                                   */
/*                                                                          */
/*     Recupera o conteúdo de uma instância de um campo do registro */

/*
*****************************************************************************/
static void ViewRecordField( DDFField * poField )
{
   int        nBytesRemaining;
   const char  *pachFieldData;
   DDFFieldDefn *poFieldDefn = poField->GetFieldDefn();
   // Report general information about the field.
   printf( "    Field %s: %s\n",
         poFieldDefn->GetName(), poFieldDefn->GetDescription() );
   // Get pointer to this fields raw data.  We will move through
   // it consuming data as we report subfield values.
   pachFieldData = poField->GetData();
   nBytesRemaining = poField->GetDataSize();
   /* ------------------------------------------------------- */
   /*      Repete para os subcampos                           */
   /*      O repeat count  deve ser ao menos um       */
   /* ------------------------------------------------------- */
   int        iRepeat;
   for( iRepeat = 0; iRepeat < poField->GetRepeatCount(); iRepeat++ )
   {

      /* ------------------------------------------------------- */
      /*   Repete todos os subcampos deste campo , avançando   */
      /*   o ponteiro de dados.                  */
      /* ------------------------------------------------------- */
      int     iSF;
      for( iSF = 0; iSF < poFieldDefn->GetSubfieldCount(); iSF++ )
      {
         DDFSubfieldDefn *poSFDefn = poFieldDefn->GetSubfield( iSF );
```

```
        int        nBytesConsumed;
        nBytesConsumed = ViewSubfield( poSFDefn, pachFieldData,
                                       nBytesRemaining );
        nBytesRemaining -= nBytesConsumed;
        pachFieldData += nBytesConsumed;
      }
   }
}
/*
************************************************************************/
/*                        ViewSubfield()                        */
/*
************************************************************************/
static int ViewSubfield( DDFSubfieldDefn *poSFDefn,
                         const char * pachFieldData,
                         int nBytesRemaining )
{
   int        nBytesConsumed = 0;
   switch( poSFDefn->GetType() )
   {
    case DDFInt:
      if( poSFDefn->GetBinaryFormat() == DDFSubfieldDefn::UInt )
        printf( "       %s = %u\n",
                poSFDefn->GetName(),
                poSFDefn->ExtractIntData( pachFieldData, nBytesRemaining,
                                          &nBytesConsumed ) );
      else
        printf( "       %s = %d\n",
                poSFDefn->GetName(),
                poSFDefn->ExtractIntData( pachFieldData, nBytesRemaining,
                                          &nBytesConsumed ) );
      break;
    case DDFFloat:
      printf( "       %s = %f\n",
```

```
        poSFDefn->GetName(),
        poSFDefn->ExtractFloatData( pachFieldData, nBytesRemaining,
                                    &nBytesConsumed ) );
  break;
case DDFString:
  printf( "       %s = `%s'\n",
        poSFDefn->GetName(),
        poSFDefn->ExtractStringData( pachFieldData, nBytesRemaining,
                                     &nBytesConsumed ) );
  break;
case DDFBinaryString:
{
    int   i;
    //rjensen 19-Feb-2002 5 integer variables to decode NAME and LNAM
    int vrid_rcnm=0;
    int vrid_rcid=0;
    int foid_agen=0;
    int foid_find=0;
    int foid_fids=0;
    GByte *pabyBString = (GByte *)
       poSFDefn->ExtractStringData( pachFieldData, nBytesRemaining,
                                    &nBytesConsumed );
    printf( "       %s = 0x", poSFDefn->GetName() );
    for( i = 0; i < MIN(nBytesConsumed,24); i++ )
       printf( "%02X", pabyBString[i] );
    if( nBytesConsumed > 24 )
       printf( "%s", "..." );

    // rjensen 19-Feb-2002 S57 quick hack. decode NAME and LNAM bitfields
    if ( EQUAL(poSFDefn->GetName(),"NAME") )
    {
       vrid_rcnm=pabyBString[0];
       vrid_rcid=pabyBString[1] + (pabyBString[2]*256)+
          (pabyBString[3]*65536)+ (pabyBString[4]*16777216);
```

```
            printf("\tVRID RCNM = %d,RCID = %u",vrid_rcnm,vrid_rcid);
          }
          else if ( EQUAL(poSFDefn->GetName(),"LNAM") )
          {
            foid_agen=pabyBString[0] + (pabyBString[1]*256);
            foid_find=pabyBString[2] + (pabyBString[3]*256)+
              (pabyBString[4]*65536)+ (pabyBString[5]*16777216);
            foid_fids=pabyBString[6] + (pabyBString[7]*256);
            printf("\tFOID AGEN = %u,FIDN = %u,FIDS = %u",
                foid_agen,foid_find,foid_fids);
          }
          printf( "\n" );
        }
        break;
      }
      return nBytesConsumed;
  }
```

A verificação das funções da Biblioteca exigiu a confecção de um programa de teste, que teve como entrada uma carta da Baía de Guanabara, no Padrão S-57, gerada pela Diretoria de Hidrografia da Marinha do Brasil. Os registros, com seus respectivos campos e subcampos, são apresentados nas Figuras 43[19] e 44[20].

[19] A execução do programa mostra o desencapsulamento dos dados armazenados no padrão 8211, apresentando os registros com seus respectivos campo, subcampos e atributos.

[20] A execução do programa mostra o desencapsulamento dos dados armazenados no padrão 8211, apresentando os registros com seus respectivos campo, subcampos e atributos.

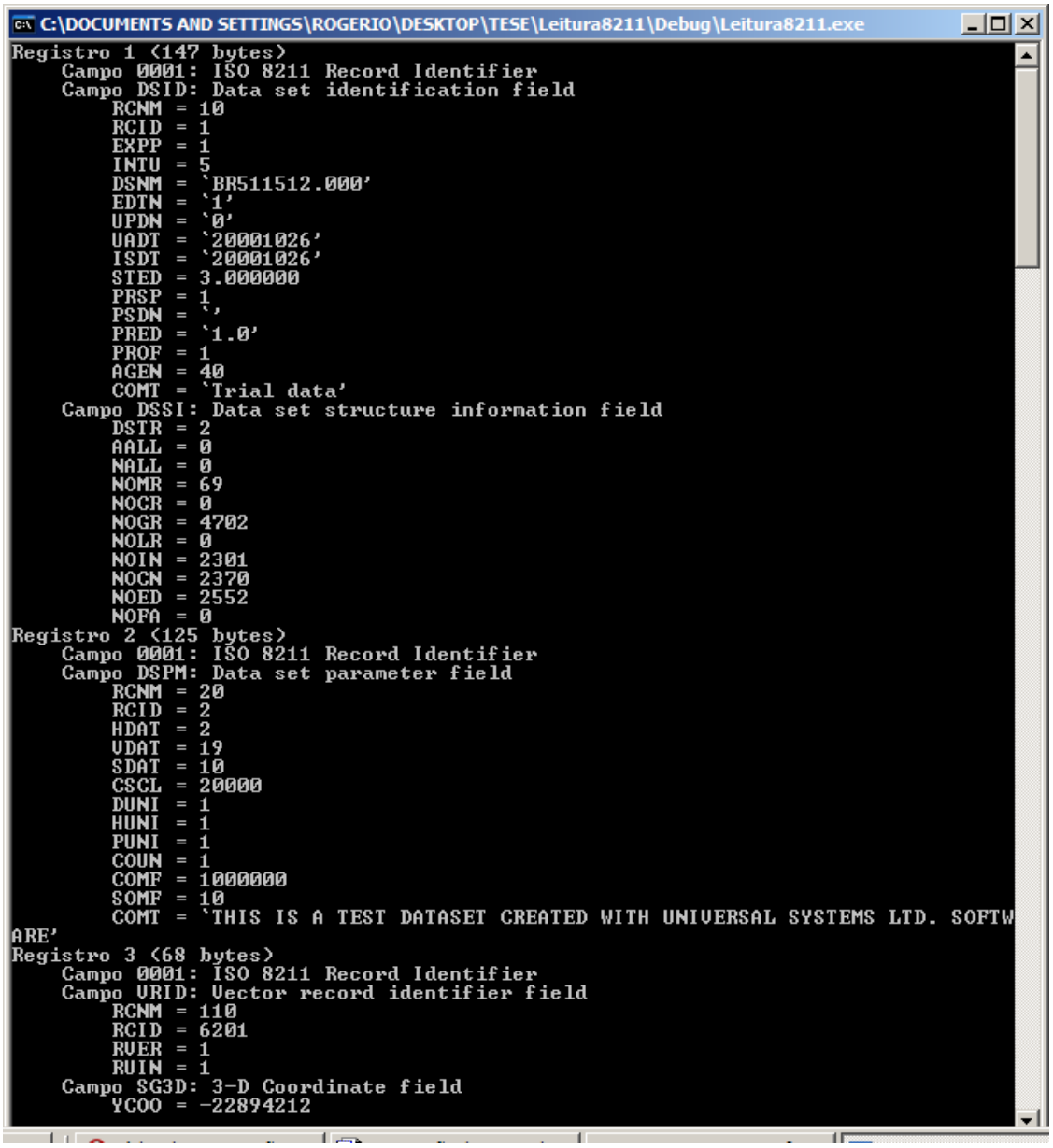


Figura 43 – Execução do Programa de Leitura da Carta no Padrão S-57

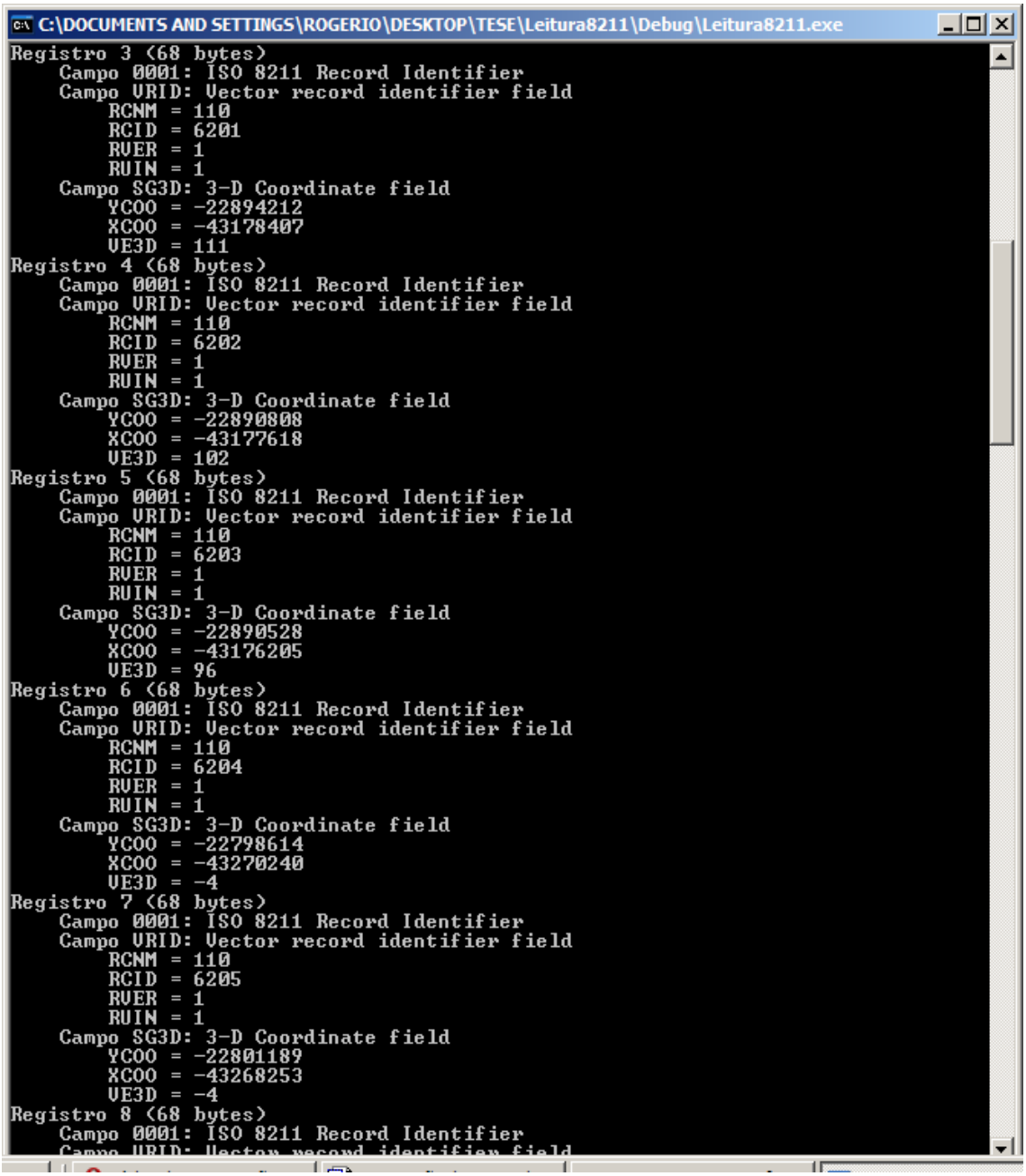


Figura 44– Execução do Programa de Leitura da Carta no Padrão S-57(continuação)

Algoritmo de validação de CRC, retirado da especificação do Padrão S-57, apêndice B.1, documento 10CRC.pdf.

O algorítmo não está otimizado para prover eficiência.

```
/* CRC_CHECK.C – Calcula o valor de CRC para um arquivo */
#include <stdio.h>
#include <math.h>
#include <string.h>
```

```
static int PolyCoefficients[] = {0,1,2,4,5,7,8,10,11,12,16,22,23,26,32};
unsigned long Polynomial = 0;
unsigned long crc_calc (unsigned short data, unsigned long accum)
{
int i=0;
for(i=8; i>0; i--)
{
if((data ^ accum) & 0x0001)
accum = (accum >> 1) ^ Polynomial;
else
accum >>=1;
data >>=1;
}
return(accum);
}
int crc_check(FILE *fileptr, unsigned long *crc_num)
{
int i = 0;
int ch = 0;
unsigned long Remainder = ~0;
Polynomial = 0;
for ( i=0; i < sizeof(PolyCoefficients)/sizeof(int); i++)
Polynomial |= 1L << (31 - PolyCoefficients [i]);
while((ch=fgetc(fileptr))!=EOF)
Remainder = crc_calc((short)ch, Remainder);
*crc_num = Remainder^~0;
return(0);
}
```

Biblioteca de visualização dos objetos S-57, utilizando o módulo de apresentação no padrão S-52.

```
// Project:  OpENCview/OpenEV
/*
   This file is part of the OpENCview project, a viewer of ENC
   Copyright (C) 2000-2004  Sylvain Duclos sduclos@users.sourceforgue.net
   This program is free “software”; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free “software” Foundation; either version 2 of the License, or
   (at your option) any later version.
   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free “software”
    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*/
#include "S57data.h"    // S57_geo
#include "S52utils.h"   // PRINTF()
#include <glib.h>       // GArray
#include <stdlib.h>     // exit()
#include <string.h>     // strlen()

// MAXINT-6 is how OGR tag an UNKNOWN value
// see gdal/ogr/ogrsf_frmts/s57/s57.h:126
// it is then turn into a string in gv_properties
#define EMPTY_NUMBER_MARKER "2147483641"  /* MAXINT-6 */

//typedef gvgeocoord geocoord;
static int id = 0;
typedef struct _pt3{ double x,y,z; } pt3;
typedef struct _rect {
    //double x;
    //double y;
    //double width;
    //double height;
    double x1;
    double y1;
    double x2;
    double y2;
} _rect;
// dta for glDrawArrays()

struct _prim {
    int mode;
    int first;
```

```
    int count;
} _prim;

typedef struct _S57_prim {
    //int     nbr;     // total number for primitive for this object
    GArray *list;     // list of _prim in 'vertex'
    GArray *vertex;
} _S57_prim;

// S57 object geo data
typedef struct _S57_geo {
    int          id;          // record id (debug)
    char         name[9];     // object name 6/8 + '\0'
    S52_Obj_t    obj_t;       // used in CS
    _rect        rect;        // lat/lon extend of object
    geocoord     pointxyz[3]; // point
    guint        linexyznbr;  // line
    geocoord    *linexyz;
    guint        ringnbr;     // area
    int         *ringxyznbr;
    geocoord   **ringxyz;
    //GArray      *XY;          // line/area list of projected geocoordinate
    S57_prim    *prim;
    GData       *attribs;
    S57_geo     *next;        // list of geo passed to CS
    void        *hGeom;       // opaque pointer to OGR geo
                              // used for CS
} _S57_geo;
////////////////////////////////////////////////////
// projection (PROJ4)
static projPJ pj    = NULL;
static projPJ pjsrc = NULL;          // projection source
static projPJ pjdst = NULL;          // projection destination
static int _setNewPROJ = TRUE;       // will set new projection
```

```
static void   _destroy_func(gpointer data)
{
   g_string_free((GString*)data, TRUE);
}
int       S57_doneObject(S57_geo *geoData)
{
   if (NULL == geoData){
      PRINTF("ERROR: deleting NULL geoData \n");
      return 0;
   }
   printf("Deleting geoData %d\n", geoData->id);
   //if (NULL != geoData->XY)
   //    g_array_free(geoData->XY, TRUE);
   if (NULL != geoData->linexyz)
      g_free((geocoord*)geoData->linexyz);
   if (NULL != geoData->ringxyz){
      int i;
      for(i = 0; i < geoData->ringnbr; ++i)
         g_free((geocoord*)geoData->ringxyz[i]);
      g_free((geocoord*)geoData->ringxyz);
   }

   if (NULL != geoData->ringxyznbr)
      g_free(geoData->ringxyznbr);
   if (NULL != geoData->attribs)
      g_datalist_clear(&geoData->attribs);
   g_free((gpointer*)geoData);
   return 1;
}
static int    _geo2prj3dv(guint npt, pt3 *data)
// convert to XY 'in-place'
{
   int   i = 0;
   pt3 *pt = data;
```

```
    int ret = 0;
    if (NULL==pjsrc || NULL==pjdst)
        return 1;
    // deg to rad --latlon
    for (i=0; i<npt; ++i, ++data)
        data->x *= DEG_TO_RAD,
        data->y *= DEG_TO_RAD;
    // rad to cartesian  --mercator
    ret = pj_transform(pjsrc, pjdst, npt, 3, &pt->x, &pt->y, &pt->z);
    if (0 != ret) {
        PRINTF("ERROR in transform (%i): %s\n", ret, pj_strerrno(pj_errno));
        exit(0);
    }
    return 1;
}
S57_geo   *S57_setPOINT(geocoord x, geocoord y, geocoord z)
{
    _S57_geo *geoData = g_new0(_S57_geo, 1);
    geoData->id          = id++;
    geoData->obj_t       = POINT_T;
    geoData->pointxyz[0] = x;
    geoData->pointxyz[1] = y;
    geoData->pointxyz[2] = z;
    //_geo2prj3dv(1,  geoData->pointxyz);
    return geoData;
}
S57_geo   *S57_setLINES(guint xyznbr, geocoord *xyz)
{
    _S57_geo *geoData = g_new0(_S57_geo, 1);
    geoData->id         = id++;
    geoData->obj_t      = LINES_T;
    geoData->linexyznbr = xyznbr;
    geoData->linexyz    = xyz;
    //_geo2prj3dv(geoData->linexyznbr,  geoData->linexyz);
```

```
    return geoData;
}
S57_geo   *S57_setAREAS(guint ringnbr, int *ringxyznbr, geocoord **ringxyz)
{
    _S57_geo *geoData = g_new0(_S57_geo, 1);
    geoData->id         = id++;
    geoData->obj_t      = AREAS_T;
    geoData->ringnbr    = ringnbr;
    geoData->ringxyznbr = ringxyznbr;
    geoData->ringxyz    = ringxyz;
    //int i = 0;
    //for (i=0; i<geoData->ringnbr; ++i)
    //    _geo2prj3dv(geoData->ringxyznbr[i],  geoData->ringxyz[i]);
    return geoData;
}
S57_geo   *S57_set_META()
{
    _S57_geo *geoData = g_new0(_S57_geo, 1);
    geoData->id       = id++;
    geoData->obj_t    = _META_T;
    return geoData;
}

int       S57_setName(S57_geo *geoData, const char *name)
{
    //strncat(geoData->name, name, 6);
    strcat(geoData->name, name);
    return 1;
}
char      *S57_getName(S57_geo *geoData)
{
    return geoData->name;
}
int       S57_setOGRGeo(S57_geo *geoData, void *hGeom)
```

```
{
   geoData->hGeom = hGeom;
   return 1;
}
void     *S57_getOGRGeo(S57_geo *geoData)
{
   return geoData->hGeom;
}
guint     S57_getRingNbr(S57_geo *geoData)
{
   if (AREAS_T != geoData->obj_t)
      return 1;
   else
      return geoData->ringnbr;
}
int       S57_getGeoData(S57_geo *geoData, int ringNo, guint *npt, double **ppt)
// helper providing uniform access to geoData
{
   if (NULL == geoData) {
      PRINTF("WARNING: geoData == NULL !?\n");
      *npt = 0;
      return 0;
   }
   if (ringNo<0 ||
      //(LINES_T==geoData->obj_t && 1 != ringNo)  ||
      (AREAS_T==geoData->obj_t && geoData->ringnbr<ringNo)) {
      PRINTF("ERROR: invalid ring number requested! \n");
      *npt = 0;
      exit(0);
      //return 0;
   }
   switch (geoData->obj_t) {
      case POINT_T:
         *npt = 1;
```

```
    *ppt = geoData->pointxyz;
    break;
  case LINES_T:
    *npt = geoData->linexyznbr;
    *ppt = geoData->linexyz;
    break;
  case AREAS_T:
    if (NULL != geoData->ringxyznbr) {
      *npt = geoData->ringxyznbr[ringNo];
      *ppt = geoData->ringxyz[ringNo];
    } else
      PRINTF("ERROR: atempt to access a tessed area .. \n"
          "(there is no geo anymore!!)\n");
    // WARNING: check if begin/end vertex are the same
    // (OpenGIS: closed and simple ring)
    // If so shorten it to help trace tesss (in combineCB)
    /*
    {
      int     n = 3 * (*npt-1);
      double *p = *ppt;
      if (p[0] == p[n+0] &&
        p[1] == p[n+1] &&
        p[2] == p[n+2]) {
        //--(*npt);
        // *npt -= 1;         // <<< shorten
      } else {
        PRINTF("not close/simple ring\n");
        //PRINTF("    START: x=%f y=%f z=%f\n", p[0],p[1],p[2]);
        //PRINTF("    END:   x=%f y=%f z=%f\n", p[n+0],p[n+1],p[n+2]);
        //exit(0);
      }
    }
    */
    //if (geodata->ringnbr > 1) {
```

```
            //      PRINTF("WARNING!!! AREA_T ringnbr:%i only exterior ring used\n",
geodata->ringnbr);
            //}
            break;
        default:
            PRINTF("ERROR object type invalid!\n");
            //return 1;
            exit(0);
    }
    return 1;
}
//GArray    *S57_getXY(S57_geo *geoData)
//{
//    // FIXME: reset array if projection change
//    if (NULL == geoData->XY)
//        geoData->XY = g_array_new(FALSE, FALSE, sizeof(double)*3);
//
//    return geoData->XY;
//}
S57_prim  *S57_initPrim(S57_prim *prim)
// set/reset primitive holder
{
    if (NULL == prim) {
        S57_prim *p = g_new0(S57_prim, 1);
        p->list   = g_array_new(FALSE, FALSE, sizeof(_prim));
        p->vertex = g_array_new(FALSE, FALSE, sizeof(double)*3);
        return p;
    } else {
        g_array_set_size(prim->list,   0);
        g_array_set_size(prim->vertex, 0);
        return prim;
    }
}
S57_prim  *S57_donePrim(S57_prim *prim)
```

```
{
   if (NULL != prim->list)   g_array_free(prim->list,   TRUE);
   if (NULL != prim->vertex) g_array_free(prim->vertex, TRUE);
   // failsafe
   prim->list   = NULL;
   prim->vertex = NULL;
   return NULL;
}
S57_prim  *S57_initPrimGeo(S57_geo *geoData)
{
   //if (NULL == geoData->XY)
   //    geoData->XY = g_array_new(FALSE, FALSE, sizeof(double)*3);
   geoData->prim = S57_initPrim(geoData->prim);
   //geoData->prim.list   = g_array_new(FALSE, FALSE, sizeof(_prim));
   //geoData->prim.vertex = g_array_new(FALSE, FALSE, sizeof(double)*3);
   return geoData->prim;
}
S57_geo   *S57_donePrimGeo(S57_geo *geoData)
{
   g_free(geoData->prim);
   geoData->prim = NULL;

   return NULL;
}
int       S57_begPrim(S57_prim *prim, int data)
{
   struct _prim p;
   p.mode  = data;
   p.first = prim->vertex->len;
   g_array_append_val(prim->list, p);
   return 1;
}
int       S57_addPrimVertex(S57_prim *prim, double *ptr)
{
```

```
    //g_array_append_val(prim->vertex, *ptr);
    g_array_append_vals(prim->vertex, ptr, 1);
    return 1;
}
int       S57_endPrim(S57_prim *prim)
{
    struct _prim *p = &g_array_index(prim->list, struct _prim, prim->list->len-1);
    p->count = prim->vertex->len - p->first;
    return 1;
}
S57_prim  *S57_getPrimGeo(S57_geo *geoData)
{
    return geoData->prim;
}
S57_prim  *S57_getPrimData(S57_prim *prim, int *primNbr, double **vertex)
{
    *primNbr =          prim->list->len;
    *vertex  = (double*)prim->vertex->data;
    return prim;
}
GArray    *S57_getPrimVertex(S57_prim *prim)
{
    return prim->vertex;
}
S57_prim  *S57_setPrimSize(S57_prim  *prim, int sz)
{
    g_array_set_size(prim->list,   sz);
    g_array_set_size(prim->vertex, sz);
    return prim;
}
int       S57_getPrimIdx(S57_prim *prim, int i, int *mode, int *first, int *count)
{
    struct _prim *p = &g_array_index(prim->list, struct _prim, i);
    if (NULL == p) {
```

```
        PRINTF("ERROR: no primitive at index: %i\n", i);
        return 0;
    }
    *mode  = p->mode;
    *first = p->first;
    *count = p->count;
    return 1;
}
int       S57_setExt(S57_geo *geoData, double x1, double y1, double x2, double y2)
{
    geoData->rect.x1 = x1;
    geoData->rect.y1 = y1;
    geoData->rect.x2 = x2;
    geoData->rect.y2 = y2;
    return 1;
}
int       S57_getExt(S57_geo *geoData, double *x1, double *y1, double *x2, double *y2)
{
    *x1 = geoData->rect.x1;
    *y1 = geoData->rect.y1;
    *x2 = geoData->rect.x2;
    *y2 = geoData->rect.y2;

    return 1;
}
S52_Obj_t  S57_getObjtype(S57_geo *geoData)
{
    if (NULL == geoData)
        return _META_T;

    return geoData->obj_t;
}
int       S57_samePtPos(S57_geo *geoA, S57_geo *geoB)
// return TRUE if 2 point are at the same position else FALSE
```

```
{
   // check if its the same object
   if (geoA == geoB)
      return FALSE;
   if (geoA->pointxyz[0] == geoB->pointxyz[0] &&
      geoA->pointxyz[1] == geoB->pointxyz[1] &&
      geoA->pointxyz[2] == geoB->pointxyz[2])
      return TRUE;
   else
      return FALSE;
}
// return the number of attributes.
static void   _countItems(GQuark key_id, gpointer data, gpointer user_data)
{
   char    *attName  = g_quark_to_string(key_id);
   if (6 == strlen(attName)){
      int *cnt = (int*)user_data;
      *cnt = *cnt + 1;
   }
}
int       S57_getNumAtt(S57_geo *geoData)
{
   int cnt = 0;
   g_datalist_foreach(&geoData->attribs, _countItems, &cnt);
   return cnt;
}
struct _qwerty {
   int currentIdx;
   char featureName[20];
   char **name;
   char **value;
};
static void getAttValues(GQuark key_id, gpointer data, gpointer user_data){
   struct _qwerty *attData = (struct _qwerty*)user_data;
```

```
    GString *attValue = (GString*) data;
    char    *attName  = g_quark_to_string(key_id);
    if (6 == strlen(attName)){
        strcpy(attData->value[attData->currentIdx], attValue->str);
        strcpy(attData->name [attData->currentIdx], attName );
        printf("inserting %s %s %d", attName, attValue->str, attData->currentIdx);
        attData->currentIdx += 1;
    } else {
        ;//      printf("sjov Att: %s  = %s \n",attName, attValue->str);
    }
}
// recommend you count number of attributes in advance, to allocate the
// propper amount of **. each char *name should be allocated 7 and the char
// *val 20 ????
int       S57_getAttributes(S57_geo *geoData, char **name, char **val){
  struct _qwerty tmp;
  tmp.currentIdx = 0;
  tmp.name       = name;
  tmp.value      = val;
  g_datalist_foreach(&geoData->attribs, getAttValues,  &tmp);
  //  strcpy(name[tmp.currentIdx], "x");
  //  strcpy(val[tmp.currentIdx], "y");
  return tmp.currentIdx;
}
GString   *S57_getAttVal(S57_geo *geoData, char *att_name)
// return attribute string value or NULL if:
//      1- attribute name abscent
//                  2-  its  a  mandatory  attribute  but  its  value  is  not  define
(EMPTY_NUMBER_MARKER)
{
    GString *att = (GString*) g_datalist_get_data(&geoData->attribs, att_name);
    // mandatory attribute with ommited value
    if (NULL!=att && (0==strncmp(att->str, EMPTY_NUMBER_MARKER, 10)))
        return NULL;
```

```
    // undefined value
    if (NULL!=att && 0==att->len)
        return NULL;
    return att;
}
GData     *S57_setAtt(S57_geo *geoData, const char *name, const char *val)
{
    GQuark   qname = g_quark_from_string(name);
    GString *value = g_string_new(val);
    if (NULL == geoData)
        return NULL;
    if (NULL == geoData->attribs)
        g_datalist_init(&geoData->attribs);
    g_datalist_id_set_data_full(&geoData->attribs, qname, value, _destroy_func);
    return NULL;
}
S57_geo   *S57_linkObj(S57_geo *geoData, S57_geo *geoNew)
// add geo to list of object, return geo
{
    // geo should not be linked to anyone
    g_assert(NULL == geoNew->next);

    // or linked to them self --degenerated case
    g_assert(geoData != geoNew);
    geoNew->next = geoData->next;
    geoData->next= geoNew;
    return geoData;
}
S57_geo   *S57_unlinkObj(S57_geo *geoData)
// remove the link between objects in list
{
    while (NULL != geoData) {
        S57_geo *tmp = geoData->next;
        geoData->next = NULL;
```

```
        geoData = tmp;
    }
    return geoData;  // NULL
}
static void   _printAtt(GQuark key_id, gpointer data, gpointer user_data)
{
    char    *attName  = g_quark_to_string(key_id);
    GString *attValue = (GString*) data;
    if (6 == strlen(attName))
        printf("\t%s : %s\n", attName, (char*)attValue->str);
}
S57_geo   *S57_nextObj(S57_geo *geoData)
// get next geo object in list --pop top node if TRUE
{
    if (NULL == geoData)
        return NULL;
    return geoData->next;
}
int       S57_dumpData(S57_geo *geoData)
// debug
{
    guint npt = 0;
    geocoord *ppt;
    switch (geoData->obj_t) {
        case POINT_T:
            npt = 1;
            ppt = geoData->pointxyz;
            //printf("POINT_T");
            break;
        case LINES_T:
            npt = geoData->linexyznbr;
            ppt = geoData->linexyz;
            //printf("LINES_T");
            break;
```

```
        case AREAS_T:
            if (NULL != geoData->ringxyznbr) {
                npt = geoData->ringxyznbr[0];
                ppt = geoData->ringxyz[0];
            } else
                PRINTF("ERROR: atempt to access a tessed area .. \n"
                    "(there is no geo anymore!!)\n");
            //if (geoData->ringnbr > 1) {
            //      PRINTF("WARNING!!! AREA_T ringnbr:%i only exterior ring drawn",
geoData->ringnbr);
            //}
            //printf("AREAS_T");
            break;
        default:
            //PRINTF("WARNING: invalid object type; %i\n", geoData->obj_t);
            ;
    }
    /*
    {
        int i   = 0;
        switch (geoData->obj_t) {
            case POINT_T:  printf("POINT_T"); break;
            case LINES_T:  printf("LINES_T"); break;
            case AREAS_T:  printf("AREAS_T"); break;
            default:
                PRINTF("WARNING: invalid object type; %i\n", geoData->obj_t);
        }
        printf("(%i): ", npt);

        for (i=0; i<npt; ++i) {
            printf("(%f,%f),", ppt[0], ppt[1]);
            ppt += 3;
        }
        printf("\n");
```

```
    }
    */
    printf("\tS57_ID : %i\n", geoData->id);
    g_datalist_foreach(&geoData->attribs, _printAtt, NULL);
    return 1;
}
int        S57_initPROJ()
{
    if (FALSE == _setNewPROJ)
        return 1;
    {   // setup source projection
        char *argssrc[] = { "proj=latlong", "ellps=WGS84"};
        if (!(pjsrc = pj_init(2, argssrc))){
            PRINTF("error init src PROJ4\n");
            exit(1);
        }
    }
    {   // setup destination projection
        char *argsdst[] = { "proj=merc", "ellps=WGS84","unit=meter" };
        if (!(pjdst = pj_init(3, argsdst))){
//      char *argsdst[] = { "proj=tmerc", "lon_0=11.8825", "lat_0=55.7290" , "a=6378137",
"b=6356752","k=0.99997" };
//      if (!(pjdst = pj_init(6, argsdst))){
            PRINTF("error init dst PROJ4\n");
            exit(1);
        }
    }
    {   // GL matrix setup
        char *args[] = { "proj=merc", "ellps=WGS84" };
        if (!(pj = pj_init(2, args))){
//        char *args[] = { "proj=tmerc", "lon_0=11.8825", "lat_0=55.7290" , "a=6378137",
"b=6356752","k=0.99997" };
//      if (!(pj = pj_init(6, args))){
            PRINTF("error init src PROJ4\n");
```

```
            exit(1);
        }
    }
    // FIXME: will need resetting for different projection
    _setNewPROJ = FALSE;
    return 1;
}
int       S57_donePROJ()
{
    if (NULL != pj)    pj_free(pj);
    if (NULL != pjsrc) pj_free(pjsrc);
    if (NULL != pjdst) pj_free(pjdst);
    pj    = NULL;
    pjsrc = NULL;
    pjdst = NULL;
    _setNewPROJ = TRUE;
    return 1;
}
projXY     S57_prj2geo(projUV pixel_xy)
// convert PROJ to geographic (LL)
{
    if (NULL != pj) {

        pixel_xy = pj_inv(pixel_xy, pj);
        if (0 != pj_errno) {
            PRINTF("x=%f y=%f %s\n", pixel_xy.u, pixel_xy.v, pj_strerrno(pj_errno));
        }
        pixel_xy.u /= DEG_TO_RAD;
        pixel_xy.v /= DEG_TO_RAD;
    }
    return pixel_xy;
}
projXY     S57_geo2prj(projUV pixel_xy)
// convert geographic (LL) to PROJ
```

```
{
   if (NULL != pj) {
      pixel_xy.u *= DEG_TO_RAD;
      pixel_xy.v *= DEG_TO_RAD;
      pixel_xy = pj_fwd(pixel_xy, pj);
      if (0 != pj_errno) {
         PRINTF("x=%f y=%f %s\n", pixel_xy.u, pixel_xy.v, pj_strerrno(pj_errno));
      }
   }
   return pixel_xy;
}
void       S57_getGeoWindowBoundary(double lat, double lng, double scale, int width, int height, double *latMin, double *latMax, double *lngMin, double *lngMax){
  S57_initPROJ();
  {
   projUV pc1, pc2;   // pixel center
   pc1.v = lat;
   pc1.u = lng;
   pc1 = S57_geo2prj(pc1); // mercator center in meters
   // lower right
   pc2.u = (width/2. +0.5)*scale + pc1.u;
   pc2.v = (height/2. +0.5)*scale + pc1.v;
   pc2 = S57_prj2geo(pc2);
   *lngMax = pc2.u;
   *latMax = pc2.v;
   // upper left
   pc2.u = -((width/2.)*scale +0.5) + pc1.u;
   pc2.v = -((height/2.)*scale +0.5) + pc1.v;
   pc2 = S57_prj2geo(pc2);
   *lngMin = pc2.u;
   *latMin = pc2.v;
  }
  printf("lat/lng: %lf/%lf scale: %lf, w/h: %d/%d lat: %lf/%lf lng: %lf/%lf\n", lat, lng, scale, width, height, *latMin, *latMax, *lngMin, *lngMax);
```

```
 //S57_donePROJ();
}
#if 0
int main(int argc, char** argv)
{
   return 1;
}
#endif
```

ANEXO B

Tabela de Cores

Uso Geral

Token		Colour	Usage
TRNSP	-	transparent	(invisible pixels)
NODTA	-	grey	(areas without chart data)
CURSR	-	orange	(cursor colour,VRM,EBL)

Seção de Cores I / Conteúdo das cartas

Token		Colour, day/night	Usage
CHBLK	-	black/grey	(general)
CHGRD	-	grey, dominant	(general)
CHGRF	-	grey, faint	(general)
CHRED	-	red	(general)
CHGRN	-	green	(general)
CHYLW	-	yellow	(general)
CHMGD	-	magenta, dominant	(general)
CHMGF	-	magenta, faint	(general)
CHBRN	-	brown	(general)
CHWHT	-	white	(general)
OUTLW	-	black	(symbol outline on sea area background)
OUTLL	-	pale/dark brown	(symbol outline on land area background)
LITRD	-	red	(red lights)
LITGN	-	green	(green lights)
LITYW	-	yellow	(white/yellow/orange/amber lights)
ISDNG	-	magenta	(isolated danger)
DNGHL	-	red	(danger highlight)
TRFCD	-	magenta, dominant	(traffic control features)
TRFCF	-	magenta, faint	(traffic control features)
LANDA	-	brown	(Land areas)
LANDF	-	brown	(Landforms, land features)
CSTLN	-	black/grey	(Coastline, shoreline constructions)
SNDG1	-	grey	(deep soundings > safety depth)
SNDG2	-	black/white	(shallow soundings <= safety depth)
DEPSC	-	grey	(safety contour)
DEPCN	-	grey	(depth contours)
DEPDW	-	white/black	(deeper than selected deep contour)
DEPMD	-	pale/dark blue	(safety contour to selected deep contour)
DEPMS	-	light/medium blue	(shallow contour to selected safety contour)
DEPVS	-	medium/light blue	(zero meter contour to shallow contour)
DEPIT	-	yellow-green	(high water line to zero meter contour)

Seção de Cores II / Imagem Radar

Token		Colour	Usage
RADHI	-	green	(high intensity echo or single int. echo)
RADLO	-	green	(low intensity echo & target trail)
ARPAT	-	green, dashed	(ARPA, target symbols & infos)

Seção de Cores III / Informações de Navegação

Token		Colour	Usage
SCLBR	-	orange	(scalebar)
CHCOR	-	orange	(chart corrections)
NINFO	-	orange	(Navigators Notes)
ADINF	-	yellow	(mariners' transparent area fill and manufacturers' points and lines)

Seção de Cores IV / Reservado para uso futuro

RESBL	-	blue	(reserved for line features)
RESGR	-	grey	(reserved for line features & screened areas)
RES04	-	not specified	(reserved for future use)
RES05	-	not specified	(reserved for future use)
RES06	-	not specified	(reserved for future use)
RES07	-	not specified	(reserved for future use)
RES08	-	not specified	(reserved for future use)

Seção de Cores V/ Símbolos do Navio e rotas

Token		Colour,day/night	Usage
SHIPS	-	black/white	(own ship, Co&SpMG vector)
PSTRK	-	black/white	(Past Track)
SYTRK	-	black/white	(Secondary Track)
PLRTE	-	red	(planned route & notations)
APLRT	-	orange	(alternate planned route)

Seção de Cores VI / Interface do Usuário

Token		Colour.day/night	Usage
UIBCK	-	white/black	(background user interface components)
UIBDR	-	grey, dominant	(user interface border components)
UIAFD	-	medium/light blue	(dominant fill colour)
UIAFF	-	brown	(faint fill colour)
UINFD	-	black/white	(dominant textual information)
UINFF	-	grey	(faint textual information)
UINFR	-	red	(textual information)
UINFG	-	green	(textual information)
UINFO	-	orange	(textual information)
UINFB	-	blue	(textual information)
UINFM	-	magenta	(textual information)