工學碩士 學位論文

GML과 SVG에 기반한 전자해도 시스템의 설계 및 구현

Design and Implementation of Electronic Navigational Chart System Based on GML and SVG

指導敎授 金 載 熏

2006年 2月

韓國海洋大學校 大學院

컴 퓨 터 工 學 科

甘 勝 哲

本 論文을 甘勝哲의 工學碩士 學位論文으로 認准함

委員長　　工學博士　　辛　沃　根　　印

委　員　　工學博士　　柳　吉　洙　　印

委　員　　工學博士　　金　載　熏　　印

2005年　12月

韓國海洋大學校　大學院

컴퓨터工學科　甘　勝　哲

목 차

Design and Implementation of Electronic Navigational Chart System Based on GML and SVG

Seung-Chul Kam

Department of Computer Engineering, Graduate School
Korea Maritime University, Busan, Korea

Advised by Jae-Hoon Kim

Abstract

With increasing Internet capability and evolving network services, geographical information is widely used in various systems, including marine and coastal geographical information systems. Although marine geographical information such as Electronic Navigational Chart has been applied to the ship navigations successfully, there may be some drawbacks such that it can be used only through some specialized systems. Furthermore, it is difficult to keep compatibility to interchange geographical data among systems because specific data formats are usually used.

To overcome these limitations, OGC published GML standard based on XML to represent geographical information efficiently, and W3C proposed vector graphic standard called SVG. In this thesis, we developed an Electronic Navigational Chart management system based on GML and SVG. It can provide services of Electronic Navigational Chart through web browsers like Internet Explorer.

The proposed system consists of four steps. In the first step, Electronic Navigational Charts are represented in the form of GML and then stored in a database. Secondly, GML document is retrieved from the database according to user query on the web. In the third step, the retrieved GML document is translated into SVG document using XSLT, and

finally, the SVG document is browsed on the web browser.

The proposed system was developed on the quite general standards so that it can be used for general purposes, while the existing systems are mainly used for specific purpose like navigation of ships. Also, it can efficiently manage and retrieve Electronic Navigational Charts by virtue of database, and further can overcome drawback of raster graphics by using vector graphics of SVG.

제 1 장 서 론

전자해도(ENC: Electronic Navigational Chart)란 기존의 종이해도에 표현된 모든 정보들을 특정한 표준형식에 따라 표현된 디지털 해도를 말한다. 전자해도에 표현된 해안선, 등심선, 수심, 항로표지, 위험물, 항로 등 선박의 항해와 관련된 모든 정보는 국제수로기구(IHO: International Hydrographic Organization)의 표준 규격인 S-57에 따라 제작되고 있다. 이러한 전자해도는 국제 공인 전자해도 표시시스템(ECDIS: Electronic Chart Display Information System)[1]을 사용하여 종이해도와 동일하게 선박항해 용도로 활용되고 있다.

하지만 이러한 형태의 전자해도 데이터는 전자해도 표시시스템이나 항해용 전자참고도(ERCS: Electronic Reference Chart System)와 같은 특수한 표시 시스템에 의존하고 있다. 즉, 전자해도 데이터는 특정 시스템에서만 해석이 가능한 형태로서 일반적인 시스템에서는 해석이 불가능하며, 다양한 시스템 간의 데이터 교환이 어렵고, 전용 브라우저가 필요하다는 단점을 가진다. 그러나 인터넷의 발달은 장소에 구애 받지 않고 전용 장비나 전용 브라우저 없이도 전자해도 데이터를 활용할 수 있는 시스템을 요구하고 있다.

한편, 현재 다양한 종류의 지리 정보 시스템이 인터넷을 통해 서비스되고 있지만, 육상 지도에만 국한되어 있어 전자해도 정보의 제공이 절실한 상황이다. 또한, 이러한 지리 정보 시스템의 가장 일반적인 형태는 독자적인 데이터 형식으로 지리 데이터를 구축하고, 검색된 결과를 이미지화하여 사용자에게 제공하는 것으로, 이런 형태의 서비스는 지리 데이터의 공유가 어렵고, 이미지를 통한 지도 출력은 해상도에 따른 정보의 손실이 발생한다.

이러한 문제점을 해결하기 위해 OGC(OpenGIS Consortium)는 범용의 시스템에서 사용될 수 있는 표준화된 지리 정보 표현 양식인 GML(Geography

Markup Language)을 제안하였고, W3C(World Wide Web Consortium)는 그래픽에 대한 논리적인 구조를 기술하고 비트맵의 단점을 극복할 대안으로 벡터 기반의 그래픽 표준인 SVG(Scalable Vector Graphics)를 제안하였다.

본 논문에서는 지리 정보 교환 표준인 GML을 이용해 전자해도 데이터를 표현하고, 표현된 전자해도를 인터넷을 통해 서비스할 수 있는 시스템을 제안한다. GML로 표현된 전자해도를 데이터베이스에 저장하여 전자해도의 관리와 사용자의 검색을 용이하게 하였고, 사용자의 검색조건에 따라 검색된 GML 전자해도를 SVG로 변환하여 웹 브라우저에 출력한다. 이를 위해 GML 문서를 SVG로 변환하기 위한 규칙을 정의하고 XSLT(eXtensible Stylesheet Language Transformations)를 이용하여 변환한다.

본 논문의 2장에서는 S-57 전자해도 표준, GML 지리 정보 표준, SVG 벡터 그래픽 표준, 그리고 XSLT 문서 변환에 대하여 살펴본다. 3장에서는 제안한 시스템을 데이터베이스 저장 모듈, 사용자 질의 처리 모듈, GML 문서 생성 모듈, SVG 문서 생성 모듈, SVG 문서 표현 모듈로 구분하여 설명한다. 4장에서는 제안한 시스템의 구현에 대하여 설명하고, GML 전자해도가 SVG로 변환되어 웹 브라우저로 출력되는 과정에 관하여 설명한다. 5장에서는 결론과 향후 연구 과제에 대해서 논의한다.

제 2 장 관련 연구

본 장에서는 S-57 전자해도 표준 포맷의 구조에 대해서 설명하고, 지리 정보를 표현하기 위한 표준 포맷인 OGC의 GML에 대해서 살펴본다. 그리고 2차원 그래픽을 표현하기 위한 W3C의 XML 그래픽 표준인 SVG에 대해서 살펴보고, XML 문서를 다른 형식의 문서로 변환하기 위한 W3C의 XSLT에 대하여 알아본다.

2.1 S-57 전자해도 표준

S-57은 전자해도 등 해양 데이터의 표현 및 교환을 위한 표준으로 국제수로기구가 2000년 11월에 버전 3.1을 공표하였다[2]. 그림 2.1은 S-57 객체 클래스의 구조를 UML(Unified Modeling Language)로 표현한 것으로, S-57은 지도상의 등대, 항구 등과 같은 피쳐(feature) 객체 및 위치 표현을 위한 공간(spatial) 객체로 구성되며, 각 객체는 객체식별자(identifier)와 속성(attribute)으로 구성된다는 것을 보여준다[2, 8].

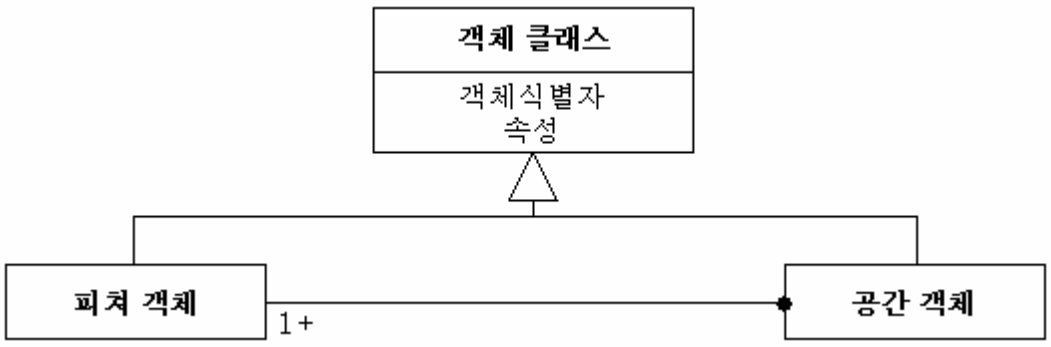

그림 2.1 S-57 객체 클래스 다이어그램

Fig. 2.1 Diagram of S-57 Object Classes

(1) S-57 데이터 모델

현실세계의 실체, 즉 객체는 피쳐 객체(feature object)와 공간 객체(spatial object)로 나누어진다. S-57에서는 이러한 실세계의 객체를 피쳐 레코드와 공간 레코드로 표현한다. 피쳐 레코드는 해당 객체를 식별하기 위한 식별자와 점, 선, 면의 지형속성을 갖는다. 또한 피쳐 레코드는 객체 간의 주종 관계(master-slave), 참조하는 공간 레코드에 대한 포인터, 다국적 언어를 위한 속성값을 포함한다. 공간 레코드는 피쳐 레코드가 참조할 수 있는 식별자와 공간 정보의 표현에 필요한 속성값을 갖는다. 피쳐 레코드는 공간 레코드의 유무에 관계없이 존재할 수 있으나 공간 레코드는 반드시 피쳐 레코드와 관련을 가져야 한다.

피쳐 객체는 다음의 4가지로 분류할 수 있다.

- Meta Object: 다른 객체들에 대한 공통 정보를 표현하는 객체
- Cartographic Object: 지도 제작과 관련된 정보를 표현하는 객체
- Geo Object: 실세계의 각종 개체(entity)를 묘사하는 객체
- Collection Object: 다른 객체들 간의 관계를 규정하는 객체

또한 공간 객체는 벡터 모델(vector model), 래스터 모델(raster model), 행렬 모델(matrix model)로 분류되지만 현재는 벡터 모델 만이 규정되어 있으며, 래스터 모델과 행렬 모델은 차후에 제공될 계획이다. 벡터 모델에서는 실세계의 지형지물을 점, 선, 면을 이용하여 객체를 표현한다.

(2) S-57의 데이터 구조

현실세계를 묘사하기 위한 개념적 데이터 모델은 데이터 구조 형태로 표현되고 시스템 상호간에 교환될 수 있다. 전자해도에 대한 정보를 표현하고 교환하

기 위한 데이터 구조는 다음과 같이 요약할 수 있다.

- 하나의 교환 세트(exchange set)는 한 개 이상의 파일로 구성된다.
- 하나의 파일(file)은 한 개 이상의 레코드로 구성된다.
- 하나의 레코드(record)는 한 개 이상의 필드로 구성된다.
- 하나의 필드(field)는 한 개 이상의 부 필드(sub-field)로 구성된다.

(3) S-57의 레코드

S-57의 레코드는 파일을 구성하는 기본 단위이며 다음의 다섯 가지 종류로 분류된다.

- Data Set Descriptive(meta): 데이터의 원천자료, 데이터가 제작된 환경, 사용된 좌표계와 투영법 등과 같은 데이터의 특성에 관한 기술과 데이터의 정확도에 관한 정보를 담고 있다.
- Catalogue: 전체 교환 파일을 참조할 때 목차와 같은 역할을 한다.
- Data Dictionary: 사용된 객체와 속성에 관한 설명을 담고 있다.
- Feature: 실세계 객체에 관한 정보를 표현한다.
- Spatial: 공간 정보를 표현한다.

(4) S-57의 필드

S-57의 필드는 레코드를 구성하는 단위로서 객체에 대한 실제 정보를 담고 있다. 하나의 레코드는 여러 개의 필드로 구성되며 하나의 필드는 여러 개의 부 필드로 구성된다. 필드와 부 필드의 계층적 구조는 S-57 Ed. 3.0에 나와 있다[2].

2.2 GML 지리 정보 표준

GML은 웹 환경에서 지리공간 정보의 저장 및 전송을 위해 데이터를 구조화된 문서인 XML로 표현한다. OGC에서는 2000년 5월에 버전 1.0이 발표된 이후로, 2001년에 XML 스키마(schema)에 기반을 둔 GML 2.0을 제시하였으며, 2003년에 26개의 스키마를 가진 GML 3.0을 채택하였다[3].

그림 2.2는 GML 3.0에서 제공하는 스키마들의 관계로서, 스키마 간의 참조 관계를 확인할 수 있다. GML 3.0은 피쳐 스키마와 지오메트리(geometry) 스키마를 기본 스키마로 하고, 이 외의 부가적인 기능을 하는 스키마로 구성된다. 이러한 26개의 코어(core) 스키마는 지리 정보를 표현하기 위한 핵심적인 스키마이다. 코어 스키마를 어플리케이션에 적합하도록 확장한 것이 응용(application) 스키마이다.

(1) 피쳐 스키마

피쳐는 도로, 강, 사람, 운송 수단, 행정 경계와 같은 의미를 가지는 실세계의 객체이다. GML의 피쳐 스키마는 실세계의 피쳐나 피쳐 콜렉션(feature collection)을 생성하기 위한 틀을 제공하는 스키마이다.

GML에서 피쳐는 XML 요소(element)로 표현되며, 피쳐 특징을 표현하는 프로퍼티(property)들과 에트리뷰트(attribute)의 집합으로 기술된다. 피쳐 스키마는 지오메트리 스키마를 포함함으로써 피쳐의 위치 정보를 표현한다.

모든 GML 피쳐 스키마는 gml:location과 gml:boundedBy 프로퍼티를 선택적으로 가진다. gml:location 프로퍼티는 피쳐의 위치를 정의하고, gml:boundedBy 프로퍼티는 피쳐 인스턴스(instance)를 둘러싸는 경계를 정의한다. 그리고, 모든 피쳐는 고유한 식별자인 gml:id 에트리뷰트를 가진다.

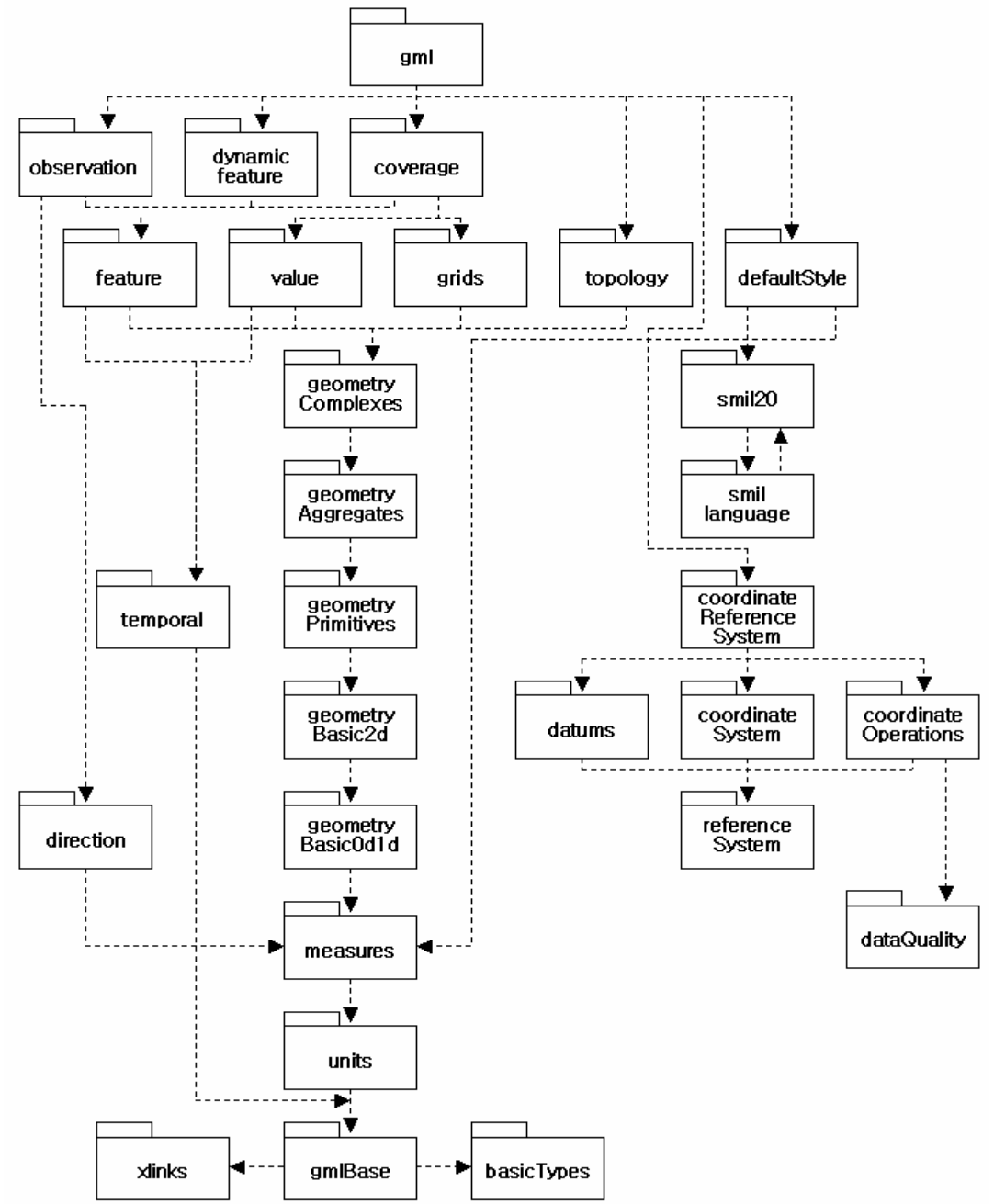

그림 2.2 GML 3.0 스키마들의 관계

Fig. 2.2 Relationship of GML 3.0 Schema

(2) 지오메트리 스키마

지오메트리 스키마는 공간 데이터를 표현하기 위한 스키마로서 지오메트리 타입 요소 및 에트리뷰트를 정의하고 있으며, 모든 지오메트리 요소는 (x, y)로 표현되는 좌표 참조 기법에 기반하여 피쳐의 위치 정보를 표현한다.

모든 지오메트리 요소들은 추상적인 수퍼타입(supertype)으로부터 직접적 또는 간접적으로 파생된다. 따라서, 지오메트리 요소는 식별자(gml:id), 이름(name), 설명(description)을 가진다.

GML에서는 지오메트리 정보를 표현하기 위해 Point, LineString, Box, LinearRing, Polygon과 여기서 확장된 MultiPoint, MultiLineString, MultiPolygon 등의 타입을 제공한다

(3) GML 응용 스키마

GML 응용 스키마는 특정 영역이나 분야에 맞게 실제 GML 문서에서 참조할 XML 스키마이다. GML에서 제공하는 기본 스키마들을 토대로 GML 문서의 구조를 정의함으로써 XML 문서를 GML로서 해석할 수 있는 기반을 제공한다. GML 응용 스키마는 GML 응용 스키마 생성 규칙을 따르는 W3C의 XML 스키마로 표현된다.

2.3 SVG 벡터 그래픽 표준

SVG는 XML을 기반으로 한 그래픽 표준 언어로서 2차원 그래픽을 표현하기 위해 W3C에 의해 2003년 1월에 권고안이 발표되었다[4]. SVG는 웹 관련 기술로 해상도에 관련 없이 변형 없는 확대와 축소가 가능하고 XML의 문법으로 기술되기 때문에 무한한 확장성을 가지고 있다. 또한 SVG는 다른 XML 문서에 의해 참조될 수 있으며 다른 SVG 그래픽 내부에도 포함될 수 있다. SVG는

벡터 그래픽 형태, 이미지, 텍스트의 3가지 그래픽 객체 타입을 가지며, 각 그래픽 객체들은 이미 표현된 객체를 그룹화하고, 스타일을 적용할 수 있으며, 변형하거나 합성할 수 있다. 또한 XML의 장점을 모두 수용하며, SMIL(Synchronized Multimedia Integration Language), GML 등 다른 XML 기반 언어들과 결합시켜 다양한 웹 어플리케이션으로 응용할 수 있다.

(1) 벡터 그래픽

SVG 형식은 벡터 기반의 웹 그래픽의 새로운 표준으로 떠오르고 있다. SVG는 벡터 기반의 문서나 이미지를 웹 브라우저에서도 그대로 읽을 수 있으며, 일반 벡터 그래픽의 장점을 그대로 살려 해상도에 따른 픽셀 변화에도 전혀 영향을 받지 않기 때문에 방대한 양의 이미지를 표현하기에 적합하다. 또한 표 2.1과 같이 다른 그래픽 표준에 비해 많은 유연성을 가지고 있다.

표 2.1 SVG와 다른 그래픽 표준의 비교

Table 2.1 Comparison of SVG and Other Graphic Standards

	SVG	Macromedia Flash	JPEG	GIF
이미지 형식	Vector	Vector	Raster	Raster
해상도 조절 가능	○	○		
이벤트 스크립트 지원	○	○		
HTML 표준	○	○	○	○
DB 연동 지원	○			
XML 기반	○			
이미지 내 텍스트 검색	○			
애니메이션 지원	○	○		○

(2) SVG 구성 요소

SVG는 XML 기반의 그래픽 표준으로 모든 그래픽 형식을 텍스트화하여 구성하고 있다. 이는 구조적인 그래픽 정의가 가능하고, 데이터베이스와 연동하여 동적인 그래픽 생성이 가능하다. 또한 각 요소는 좌표 시스템에 기반하기 때문에 지리 정보 시스템과의 결합이 용이하다. 표 2.2는 SVG 문서를 구성하는 주요 구성 요소를 보여준다.

표 2.2 SVG의 구성 요소

Table 2.2 Composition Elements of SVG

요소 구분	요소 명	요소 기능
루트 요소	svg	SVG 문서의 부분 정의, 중첩 가능
설명 요소	title	SVG 문서의 제목 설명
	desc	SVG 문서의 내용 설명
그룹핑 요소	g	그래픽 요소의 그룹핑 정의
기본 도형 요소	line	두 개의 좌표로 이루어진 선 그리기
	polyline(path)	좌표의 나열로 다중선 그리기
	circle	한 개의 좌표와 반지름 크기로 이루어진 원 그리기
	rectangle	두 개의 좌표로 이루어진 사각형 그리기
	polygon	시작점과 끝점이 같은 폐쇄형 선 그리기
심볼 요소	symbol	그래픽 요소 심볼화
	def	참조될 요소 정의
	use	참조 가능한 요소 사용
텍스트 요소	text	텍스트 속성 지정
링크 요소	a	하이퍼링크 제공
스크립트 요소	script	이벤트 스크립트 지정

2.4 XSLT 문서 변환 표준

웹 브라우저들은 HTML(Hypertext Markup Language)과 같이 이미 정의된 태그를 해석하여 출력한다. 그러나 HTML과 달리, GML은 미리 정의된 태그(predefined tag)를 사용하지 않고, 사용자에 의해 만들어진 태그를 사용한다. 이러한 GML의 특성으로 인해 브라우저는 GML 문서를 어떻게 출력해야 하는지 알 수가 없다.

그래서, GML 문서를 브라우저 상에 표현하기 위해 GML 문서가 어떤 방식으로 출력되어야 하는지를 결정하기 위한 메커니즘이 필요하다. 이러한 메커니즘의 하나로 CSS(Cascading Style Sheet)와 XSL(eXtensible Stylesheet Language)이 있다. 이 중 XSL은 HTML과 함께 사용되는 CSS보다 훨씬 복잡하고 다양한 기능을 지원한다.

XSLT는 XML 문서의 변환(transformation) 기능을 수행하는 언어로 입력 문서에 스타일시트 문서의 기술된 명령(instruction)을 적용함으로써 다른 문서 형식으로 변환하거나 재구성할 수 있다[5, 7].

(1) XSLT의 처리 과정

XSLT가 주어진 문서를 필요한 출력 형태로 변환하는 동안 이루어지는 처리 과정은 크게 두 가지로 분리할 수 있다.

- 구조적 변환 단계 : 입력된 XML 문서의 구조는 요구되는 출력 형태를 반영하는 구조로 데이터의 변환이 이루어진다.
- XSL 포맷팅(formatting) : 입력된 XML 문서의 구조는 HTML이나 PDF 등의 형식을 가지는 출력 형태를 갖는다.

스타일시트 문서의 코드는 선언적이다. 이미 정의된 명령들을 적절히 사용함으로써 의도하는 결과 문서를 만들 수 있다. 즉, 선언 규칙(writing rule)에 의해서 변환하고자 하는 의도를 선언할 수 있다. 이러한 선언 규칙이 입력 문서에서 출력할 내용을 선택하고, 스타일링(styling)을 결정하고, 입력 문서에 포함되지 않은 부분을 추가하여 출력한다.

XSLT는 프로세싱 하는 동안, 소스 문서가 XSL 스타일시트에서 이미 정의된 서식(template)에 일치하는 부분을 만나면 그 일치되는 부분에 대한 명령을 수행하여 결과 문서를 생성한다.

(2) XSLT의 서식 요소

XSLT는 원본 문서의 구조를 결과 트리(tree)로 변환하는 패턴(pattern)과 서식이 결합된 규칙으로 설명할 수 있다. XSLT는 서식과 함께 구조화하며, 서식은 문자열의 조합으로 된 형식으로 문서의 서식 요소를 추가한다. 또한 문서의 구조 요소를 새로이 생성하여 지정할 수도 있다. 이러한 서식은 각각 패턴으로 탐색된 XML 구조 요소를 선택하고 대응 요소를 서식 규칙으로 처리하여 그 결과를 문서로 생성한다. 결국 패턴 생성 결과로 XML 구조의 요소를 선택하고, 서식 내의 포맷 처리 언어로 선택된 구조 요소를 XSLT 처리기로 보내어 문서를 정의된 형식으로 변환한다. 적용하려는 서식을 검색하는 방법에서 주어진 XML 구조 요소에 대응하는 패턴을 갖는 서식은 다양하게 존재할 수 있지만 XSLT 처리기에서 처리되는 서식은 한 서식만 적용된다. 그리고 서식은 XML 이름 공간(XML Namespace)에 지정된 영역 집합을 가지는데, 접두어(prefix) ‘xsl’을 사용하여 XSLT 서식 요소를 사용할 수 있으며 XSLT 서식 요소는 크게 다음의 세 가지로 구분할 수 있다.

- 기본 제어 요소 : XML 문서의 내용을 처리하기 위한 XSLT 서식 요소
- 수정 및 처리 요소 : XML 문서의 구조 요소를 수정하고 처리하는 기능을 제공하는 요소
- 의사 결정 요소 : XSLT 문서에서 의사 결정을 위한 요소

각 요소에 속하는 XSLT의 서식 요소는 표 2.3과 같다.

표 2.3 XSLT의 서식 요소

Table 2.3 Template Elements of XSLT

요소 구분	요소 명	요소 기능
기본 제어 요소	xsl:stylesheet	서식 집합 정의
	xsl:template	선택된 구조 요소를 패턴으로 정의
	xsl:apply-templates	관련 서식을 탐색하여 처리기에 전달
	xsl:copy	패턴의 처리 결과를 다른 노드에 복사
	xsl:value-of	패턴의 처리 결과 값을 표현
수정 및 처리 요소	xsl:element	처리 결과에 요소 추가
	xsl:attribute	처리 결과에 속성 추가
	xsl:comment	처리 결과에 주석 추가
	xsl:processing-instruction	처리 결과로 처리 명령의 표현
의사 결정 요소	xsl:for-each	여러 구조 요소에 동일한 서식 적용
	xsl:if	서식의 조건 처리
	xsl:choose	여러 조건문을 실행하도록 구조화
	xsl:when	조건문에 따르는 서식 표현

(3) XPath

XPath는 XML 문서의 요소와 대응 관계를 표현하기 위한 부분으로 XML 문서 구조에 접근하기 위한 언어이다. XSLT를 개발하는 과정에서 문서의 일부분을 선택하는 XSLT의 표현 문법과 문서 간의 연결을 위해 개발된 XLink(XML Linking Language) 언어가 상당히 중첩되어 있다는 사실이 발견되어, 이러한 표현 언어의 중첩을 피하기 위해 새로운 단일 언어인 XPath 1.0을 정의하여 1999년 11월 16일에 W3C에서 권고하였다[6].

XPath는 XSLT의 하위 언어로서 그 역할을 수행하며, XSLT처럼 노드(node)로 이루어진 트리구조를 이용하여 XML 문서를 표현한다. 요소, 텍스트, 속성 등이 모두 XPath에서는 노드로 표현되며, XML의 계층 구조로 표현한 결과는 트리 형태와 유사하다. 그리고 XPath의 표현은 수치 계산이나 문자열 조작 또는 논리 연산자를 통한 조건문 판단을 위해 사용되지만, 가장 대표적인 용도는 입력 문서 내에서 처리되어야 할 부분을 추출하는 것이다[7].

제 3 장 전자해도 시스템 설계

본 장에서는 제안한 전자해도 시스템의 구조를 모듈별로 설명한다. S-57 전자해도를 GML로 변환하여 데이터베이스에 저장한다. 저장된 데이터베이스를 통해 질의처리가 이루어지며, 질의 결과는 다시 GML 문서로 생성된다. 생성된 GML 문서는 XSLT를 이용하여 SVG 문서로 변환된 후에 웹 브라우저를 통해 출력된다.

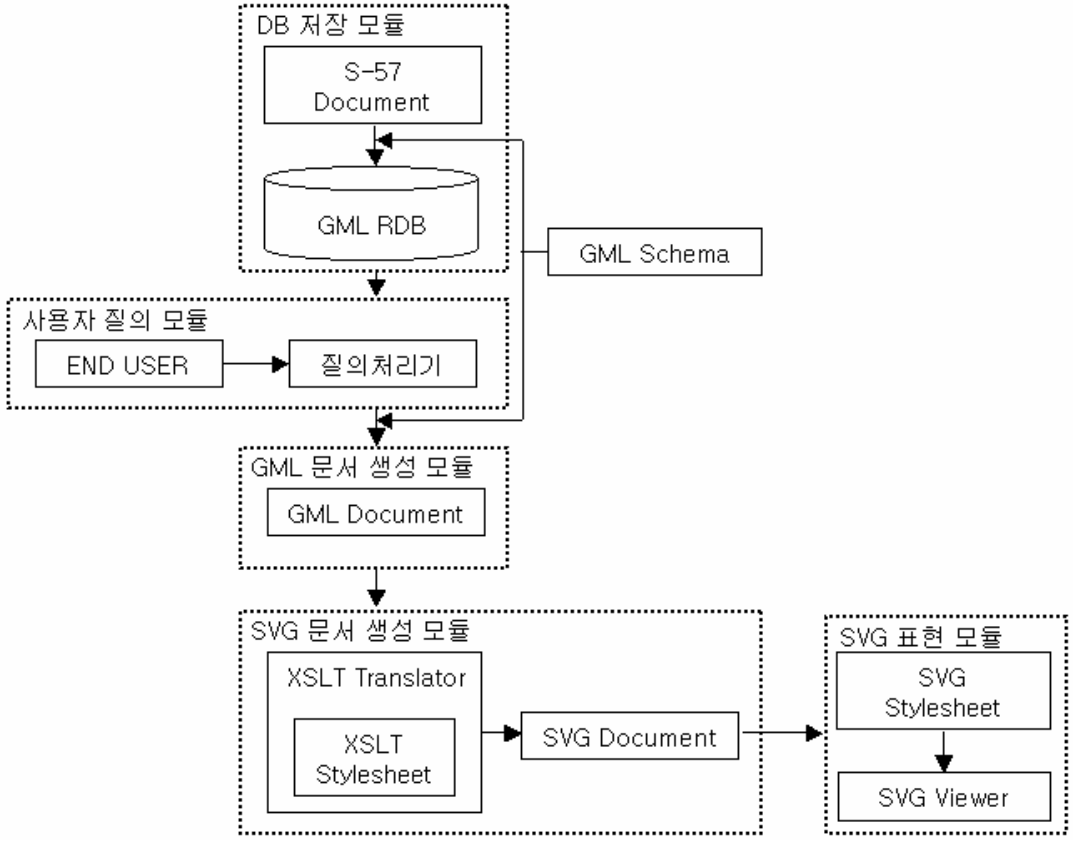

그림 3.1 시스템 구성도

Fig. 3.1 System Architecture

3.1 S-57 문서의 데이터베이스 저장 모듈

전자해도를 위한 데이터 표준인 S-57은 객체 클래스(object class)들의 집합으로 구성된다. 객체 클래스는 지도상의 등대, 항구 등과 같은 지리적 특징 객체와 위치 표현의 공간 객체, 객체 식별자와 에트리뷰트로 구성된다[8]. 2진 코드로 표현된 S-57 전자해도 데이터는 텍스트로 변환된 후 객체 클래스 단위로 분할한다. 생성된 텍스트를 GML 스키마 구조에 맞게 변환한 후 XML 파서(parser)를 통해 데이터베이스에 저장한다.

데이터베이스는 GML 문서의 저장과 검색의 효율성을 고려하여 관계형 및 객체지향형 데이터베이스의 장점을 이용할 수 있는 객체 관계형 데이터베이스를 사용하여 시스템을 구성하였다. 데이터베이스는 단일 테이블로 구성하였으며, 객체 클래스에서 지도 명, 객체 명, 객체 내용 별로 데이터를 추출하여 저장하였다.

3.2 사용자 질의 처리 모듈

사용자 질의는 웹 서버와 데이터베이스 서버, 클라이언트 간의 동작으로 이루어진다. GML 문서가 저장된 데이터베이스에서 정보를 검색하는 방법에는 크게 구조 기반 검색과 내용 기반 검색이 있다[9]. 본 논문에서는 검색 인터페이스를 통하여 지역과 객체 단위의 구조 기반 검색을 웹 서버에 지시한다. 웹 서버는 사용자 질의를 데이터베이스 서버에 SQL(Structured Query Language) 형태로 전달한 후 결과 집합을 돌려 받는다. SQL은 데이터베이스에서 데이터를 선택하고 추가하고 삭제하고 갱신하는 등의 역할을 하는 구조화된 질의 언어로서, 본 논문에서는 선택 질의 만을 사용한다. 선택 질의는 다음과 같은 형태를 가진다. SELECT 절은 검색할 필드를 지정하고, FROM 절은 검색할 필드를 포함하는 테이블을 지정하며, WHERE 절은 검색할 필드의 조건을 지정한다.

SELECT [field1, field2,...]

FROM [table1, table2,...]

WHERE [Condition1, Condition2,...]

검색된 결과 집합은 GML 응용 스키마 구조를 수용하는 GML 문서로 재구성된다. 그림 3.2는 사용자 질의 처리 과정을 보여준다.

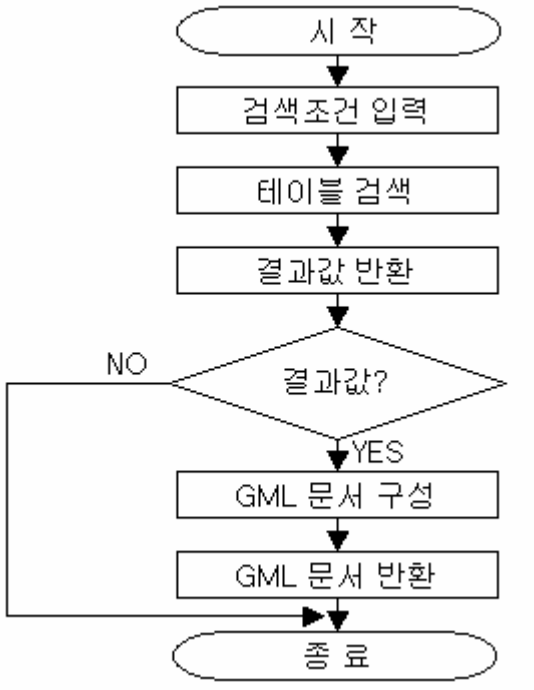

그림 3.2 사용자 질의 처리

Fig. 3.2 Processing User Query

3.3 GML 문서 생성 모듈

반환된 결과는 GML의 피쳐 스키마와 지오메트리 스키마를 기반으로 한 응용 스키마를 만족하는 GML 문서로 생성된다. 그림 3.3은 GML 응용 스키마의

구조를 보여준다. GML 응용 스키마는 피쳐 객체의 집합으로 구성되며, 각 피쳐는 요소 선언부와 타입 선언부로 구분된다. 요소 선언부에서는 피쳐의 타입을 정의하고, 타입 선언부에서는 피쳐를 구성하는 에트리뷰트, 속성 요소, 공간 객체 요소를 정의한다.

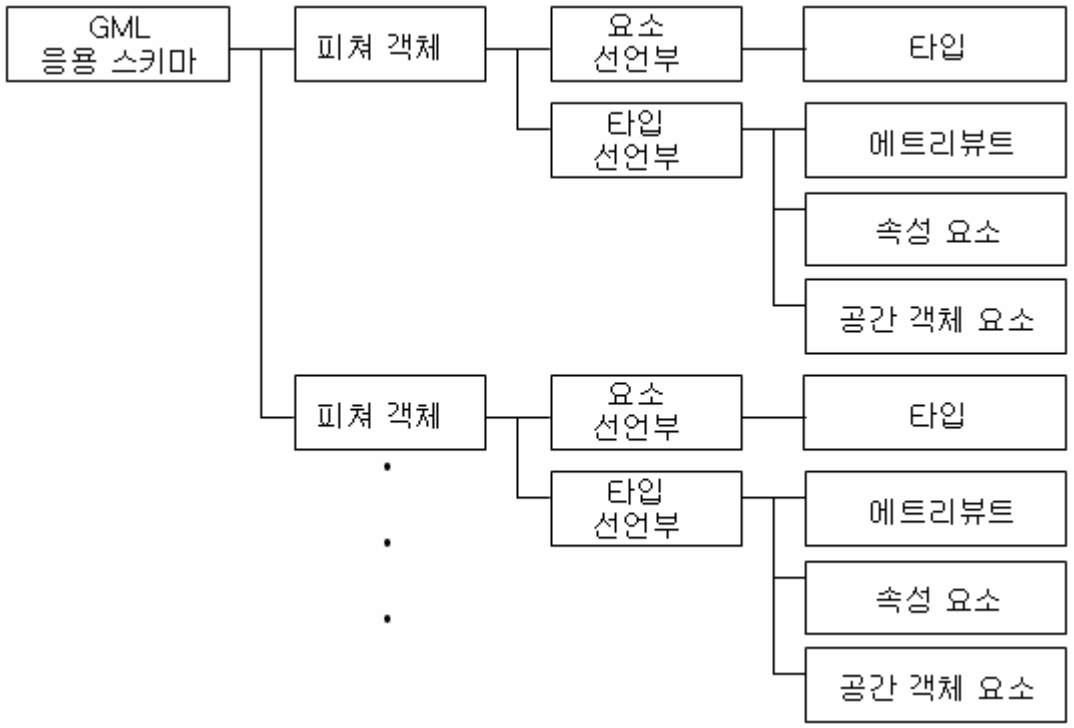

그림 3.3 GML 응용 스키마의 구조

Fig. 3.3 Structure of GML Application Schema

생성된 GML 문서에서 점(point)은 위도와 경도 좌표를 가지는 위치 정보이며, 선(line)은 점과 점의 연결, 면(area)은 연속된 점들의 연결로 구성된 다각형(polygon)으로 표현된다. 그림 3.4는 갈도스사(Galdos Inc.)에서 제공하는 S-57 데이터의 GML 응용 스키마의 일부로서 등대 객체의 타입과 구성 요소를 보여준다[10].

```
<!--======================== element ========================-->
<element name="Light" type="s57:LightType" substitutionGroup="gml:_Feature">
    <annotation>
        <documentation>The "Light" feature corresponds to the s57 object "LIGHTS". A luminous or lighted aid to navigation.
        </documentation>
    </annotation>
</element>
<!--====================== complexType ======================-->
<complexType name="LightType">
    <complexContent>
        <extension base="s57:AbstractFeatureType">
            <sequence>
                <element name="acronym" type="string" fixed="LIGHTS"/>
                <element ref="s57:catlit" minOccurs="0"/>
                <element ref="s57:colour" minOccurs="0"/>
                <element ref="s57:datsta" minOccurs="0"/>
                                    .
                                    .
                                    .
                <element ref="s57:sordat" minOccurs="0"/>
                <element ref="s57:sorind" minOccurs="0"/>
                <element name="position" type="s57:NodePropertyType" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

그림 3.4 Galdos사의 응용 스키마

Fig. 3.4 Application Schema of Galdos Inc.

그림 3.5는 갈도스 응용 스키마에 의해 생성된 GML 문서의 예를 보여준다. 응용 스키마에 따라 등대에 대한 속성 정보가 순서대로 작성되었으며, 요소의 발생 횟수를 지정하는 minOccurs가 “0”이기 때문에 데이터가 없는 일부 요소는 표시되지 않았다. 등대의 위치 정보는 노드 타입으로 선언되었으므로, (x, y)의 좌표 정보로 표현되었다.

```
<Light gml:id="_540_2135148864_687">
  <acronym>LIGHTS</acronym>
  <catlit>
    <CategoryOfLight>
      <code>37</code>
     <value>lower</value>
    </CategoryOfLight>
  </catlit>
  <colour>
    <Colour>
      <code>75</code>
      <idList>1</idList>
    </Colour>
  </colour>
  <height>
    <Height>
      <code>95</code>
    </Height>
  </height>
  <position>
    <gml:Node gml:id="N120_668">
      <gml:pointProperty>
        <gml:Point gml:id="P120_668">
          <gml:pos>-32.498195 60.895926</gml:pos>
        </gml:Point>
      </gml:pointProperty>
    </gml:Node>
  </position>
</Light>
```

그림 3.5 GML 문서의 예

Fig. 3.5 Example of a GML Document

3.4 SVG 문서 생성 모듈

GML 문서를 SVG 문서로 변환하기 위해서는 응용 스키마를 분석하기 위한 스타일시트, GML 문서에서 피쳐를 추출하기 위한 스타일시트, 추출된 피쳐의 지오메트리 요소를 SVG의 그래픽 요소로 변환하기 위한 스타일시트로 구성된다. XSLT 변환기는 GML 문서의 요소와 스타일시트에 정의된 변환 규칙을 패턴 매칭(pattern matching)함으로써 SVG 문서를 생성한다. 그림 3.6은 SVG 문서 생성에 필요한 각 스타일시트 간의 관계를 보여준다. 응용 스키마를 분석하여 생성된 피쳐 스타일시트와 그래픽 스타일시트를 변환 모듈이 참조하여 GML 문서를 SVG 문서로 변환한다.

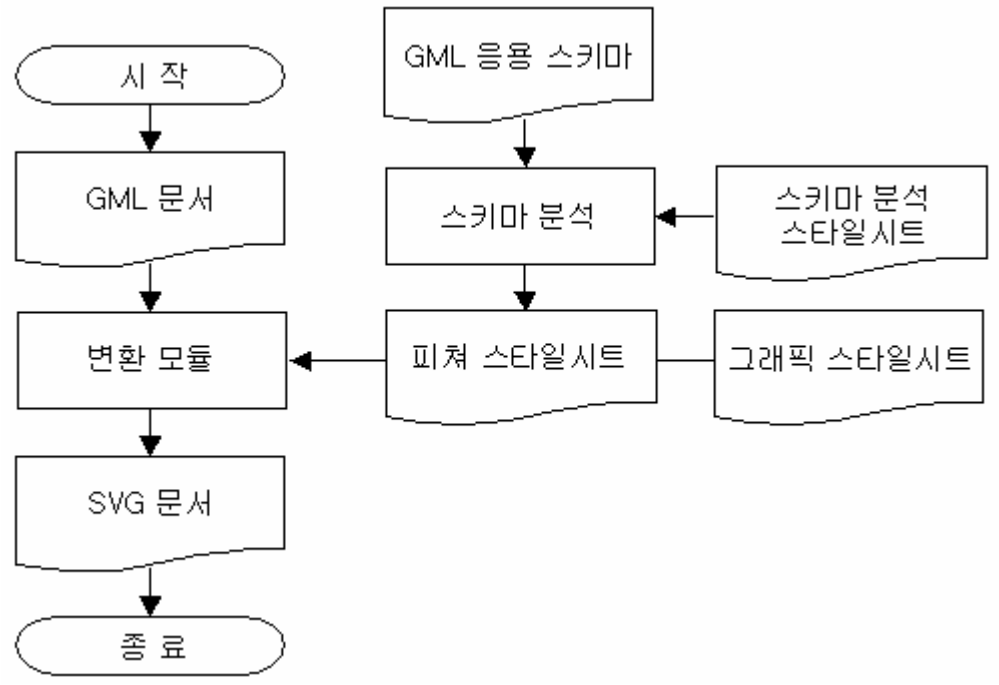

그림 3.6 SVG 문서 생성 모듈

Fig. 3.6 Generation Modules of SVG Documents

표 3.1은 GML 문서의 요소와 SVG 문서의 요소 간의 대응 관계를 보여준다. GML 문서와 SVG 문서 간의 대응은 1:1이 기본이지만, SVG가 더 많은 그래픽 요소를 제공하기 때문에 대응 관계는 1:M으로 구성할 수 있다. 다만, SVG는 점에 대한 그래픽 요소가 정의되어 있지 않으므로 사각형이나 원(circle) 요소를 이용하여 변환하였다. (x, y)로 표현되는 좌표 정보 외의 필요한 너비(w)와 높이(h)에 대한 정보는 해상도에 비례하여 정의하였다.

표 3.1 GML 문서의 요소와 SVG 문서의 요소
Table 3.1 GML Document Elements and SVG Document Elements

타입 구분	문서 구분	요소 형식
점	GML	<gml:Point > <gml:pos>x1,y1</gml:pos> </gml:Point>
	SVG	<rect x="x1" y="y1" width="w1" height="h1"/>
		<circle x="x1" y="y1" r="r1"/>
선	GML	<gml:LineString> <gml:pos>x1,y1 x2,y2 …</gml:pos> </gml:LineString>
	SVG	<line points="x1,y1 x2,y2 …" />
		<polyline points="x1,y1 x2,y2 …" />
		<path d="Mx1,y1 Lx2,y2 …Z" />
다각형	GML	<gml:Polygon> <gml:pos>x1,y1 x2,y2 x3,y3 … x1,y1</gml:pos> </gml:Polygon>
	SVG	<path d="Mx1,y1 Lx2,y2 Lx3,y3…Z" />
심볼	GML	<gml:Object1> <gml:Point ><gml:pos>x1,y1</gml:pos></gml:Point> </gml:Object1>
	SVG	<image x="x1" y="y1" " width="w1" height="h1" xlink:href="image path"/>
		<symbol id="symbol1" viewbox="x1 y1 x2 y2"/>
		<use x="x1" y="y1" width="w1" height="h1" xlink:href="symbol1">

(1) 스키마 분석 스타일시트

그림 3.7은 GML 응용 스키마를 분석하여 피쳐 스타일시트를 동적으로 생성하기 위한 스타일시트의 구조를 UML로 표현한 것이다. 응용 스키마에서 gml:_Feature에 대해 스타일시트를 반복적으로 적용하여 피쳐의 그룹, ID, 클래스, 지오메트리 타입을 추출한다. 지오메트리 타입은 Polygon, LineString, Point를 대상으로 한다.

이 스타일시트를 통해 응용 스키마가 수정되어도 피쳐 스타일시트를 동적으로 생성할 수 있다. 스키마 분석은 SVG 문서 정의 부분, 그래픽 스타일시트 정의 부분이 먼저 수행되고, 응용 스키마에 선언된 객체 단위로 반복해서 적용된다. 응용 스키마의 요소 선언부에서 ID와 객체 명을 추출하고, GML 문서의 gml:Point, gml:LineString, gml:Polygon 요소에 서식을 적용하도록 스타일시트가 구성되었다. 그림 3.8은 응용 스키마를 분석하기 위한 스타일시트의 일부분이다[11].

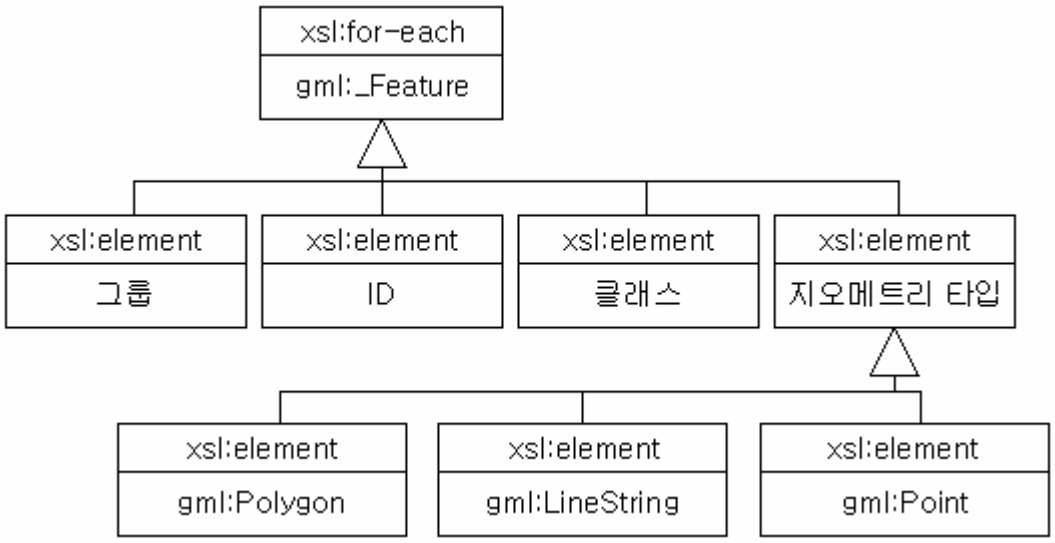

그림 3.7 응용 스키마 분석을 위한 스타일시트의 구조
Fig. 3.7 Structure of Stylesheet for Application Schema Analysis

```
<xsl:for-each select="//*[@substitutionGroup='gml:_Feature']">
  <xsl:variable name="tipo" select="concat(@name,'Type')"/>
  <xsl:variable name="nome" select="@name"/>
  <xsl:element name="xsl:element">
    <xsl:attribute name="name">g</xsl:attribute>
    <xsl:element name="xsl:attribute">
      <xsl:attribute name="name">id</xsl:attribute>
      <xsl:value-of select="@name"/>
    </xsl:element>
    <xsl:element name="xsl:attribute">
      <xsl:attribute name="name">class</xsl:attribute>
      <xsl:value-of select="translate(@name,'_','')"/>
    </xsl:element>
    <xsl:element name="xsl:for-each">
      <xsl:attribute name="select">
        <xsl:text>//*[name()='</xsl:text>
        <xsl:value-of select="@name"/>
        <xsl:text>']</xsl:text>
      </xsl:attribute>
      <xsl:element name="xsl:apply-templates">
        <xsl:attribute name="select">
          <xsl:text>.//gml:LineString</xsl:text>
        </xsl:attribute>
        <xsl:element name="xsl:with-param">
          <xsl:attribute name="name">ID</xsl:attribute>
          <xsl:element name="xsl:value-of">
            <xsl:variable name="resulta" select="//*[name()='complexType' and
@name=$tipo]//*[name()='element' and position()=1]/@name"/>
            <xsl:attribute name="select">
              <xsl:text>./*[name()='</xsl:text>
              <xsl:value-of select="$resulta"/>
              <xsl:text>']</xsl:text>
            </xsl:attribute>
```

```
            </xsl:element>
          </xsl:element>
      </xsl:element>
    </xsl:element>
</xsl:for-each>
```

그림 3.8 응용 스키마 분석을 위한 스타일시트

Fig. 3.8 Stylesheet for Application Schema Analysis

(2) 피쳐 스타일시트

그림 3.9는 응용 스키마 분석을 통해 생성된 피쳐 스타일시트의 구조를 UML 방식으로 보여준다. 피쳐 스타일시트는 응용 스키마에 선언된 모든 피쳐를 추출하여 그래픽 스타일시트를 반복적으로 적용함으로써, 그래픽 요소가 출력되는 순서를 지정하는 스타일시트이다. GML 피쳐의 모든 하위 요소에 대해서 출력되는 순서는 면, 선, 점 순으로 이루어지며, 모든 피쳐에 동일하게 적용된다. 그림 3.10은 등대 피쳐에 대해 출력 순서를 지정한 스타일시트이다.

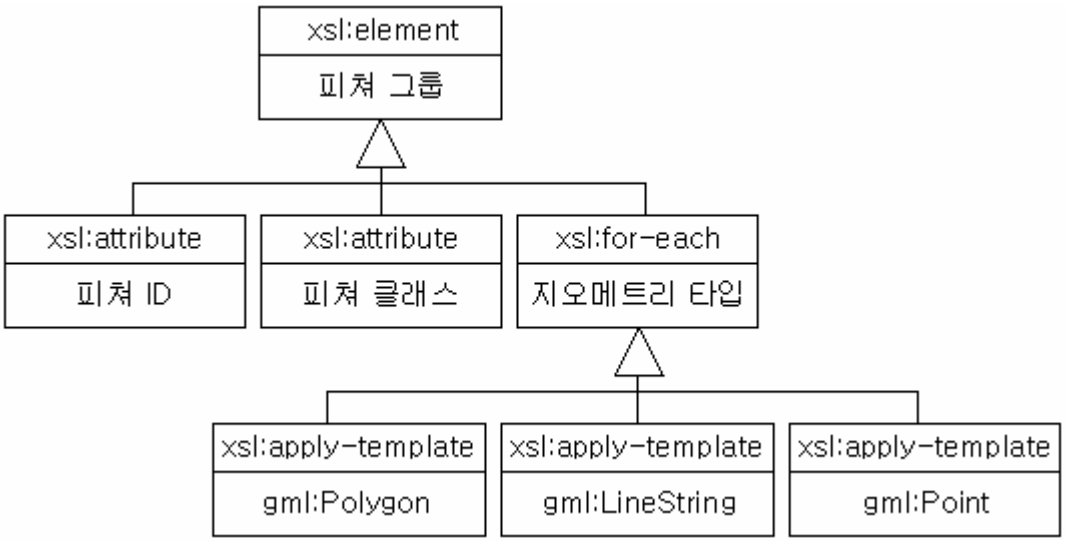

그림 3.9 피쳐 스타일시트의 구조

Fig. 3.9 Structure of Feature Stylesheet

```
<!--====================== Light element  ======================-->
<xsl:element name="g">
  <xsl:attribute name="id">Light</xsl:attribute>
  <xsl:attribute name="class">Light</xsl:attribute>
  <xsl:for-each select="//*[name()='Light']">
    <xsl:apply-templates select=".//gml:Polygon">
      <xsl:with-param name="ID">
        <xsl:value-of select="./*[name()='acronym']" />
      </xsl:with-param>
    </xsl:apply-templates>

    <xsl:apply-templates select=".//gml:LineString">
      <xsl:with-param name="ID">
        <xsl:value-of select="./*[name()='acronym']" />
      </xsl:with-param>
    </xsl:apply-templates>

    <xsl:apply-templates select=".//gml:Point">
      <xsl:with-param name="ID">
        <xsl:value-of select="./*[name()='acronym']" />
      </xsl:with-param>
    </xsl:apply-templates>
  </xsl:for-each>
</xsl:element>
```

그림 3.10 피쳐 스타일시트의 예

Fig. 3.10 Example of Feature Stylesheet

(3) 그래픽 스타일시트

그림 3.11은 GML 문서에서 지오메트리 요소를 추출하여 SVG 문서의 그래픽 요소로 변환하기 위한 스타일시트의 구조를 UML 방식으로 보여준다. xsl:template는 추출하기 위한 지오메트리 요소를 지정하고, xsl:param은 지오

메트리 요소의 ID를 지정하고, xsl:element는 추출된 지오메트리 요소와 대응하는 SVG 그래픽 요소를 지정하며, xsl:attribute를 통해 지오메트리 요소의 좌표 정보를 추출한다.

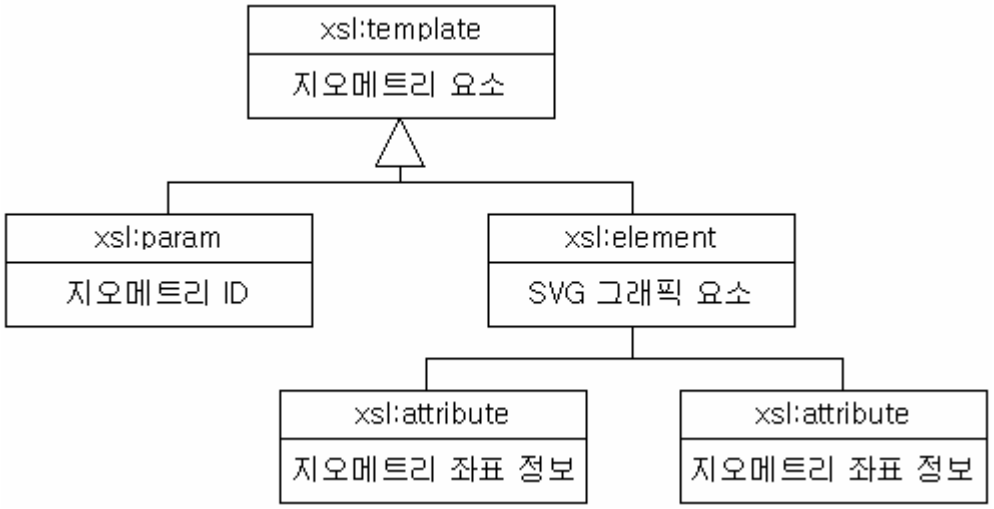

그림 3.11 그래픽 스타일시트의 구조
Fig. 3.11 Structure of Graphic Stylesheet

그림 3.12는 그래픽 스타일시트의 일부이다. GML 문서에서 점에 해당하는 gml:Point, 선에 해당하는 gml:LineString, 다각형에 해당하는 gml:Polygon 요소와 스타일시트의 서식이 매치되면 스타일시트에 선언된 명령을 수행하여 GML 문서의 지오메트리 요소를 추출한다. 추출된 지오메트리 요소는 SVG 그래픽 요소의 형식에 맞춰 변환된다.

```
<!--LineString-->
<xsl:template match="gml:LineString">
    <xsl:param name="ID"> </xsl:param>
    <xsl:element name="polyline">
        <xsl:attribute name="id">
            <xsl:value-of select="$ID"/>
        </xsl:attribute>
        <xsl:attribute name="points">
            <xsl:value-of select="normalize-space(gml:pos)"/>
        </xsl:attribute>
    </xsl:element>
</xsl:template>

<!--Polygon-->
<xsl:template match="gml:Polygon">
    <xsl:param name="ID"> </xsl:param>
    <xsl:element name="path">
        <xsl:attribute name="id">
            <xsl:value-of select="$ID"/>
        </xsl:attribute>
        <xsl:attribute name="d">
            <xsl:for-each select="*/gml:LineString">
                <xsl:text>M</xsl:text>
                <xsl:value-of select="translate(normalize-space(gml:pos),' ','L')"/>
                <xsl:text>z</xsl:text>
            </xsl:for-each>
        </xsl:attribute>
    </xsl:element>
</xsl:template>

<!--gml:Point 요소에 대한 서식이 지면 관계상 생략 되었다. -->
```

그림 3.12 그래픽 스타일시트의 예

Fig. 3.12 Example of Graphic Stylesheet

3.5 SVG 문서 표현 모듈

변환된 SVG 문서는 어도비사(Adobe Inc.)의 SVG 플러그인을 통해 사용자에게 보여진다. 그림 3.13은 SVG 표현 모듈로서 SVG 문서, 프리젠테이션 스타일시트, SVG 뷰어 간의 관계를 보여준다. 현재 버전 3.02인 SVG 뷰어(viewer)는 윈도우와 매킨토시 플렛폼에서 플러그인을 제공하며, 대부분의 웹 브라우저를 지원한다. 또한 플러그인은 출력된 SVG 문서에 대해 자체적으로 줌(zoom), 패닝(panning), 복사, 소스 보기 등의 기능을 제공한다.

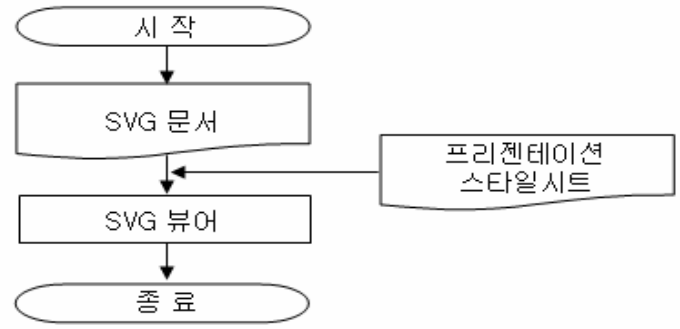

그림 3.13 SVG 문서 표현 모듈
Fig. 3.13 SVG Document Presentation Module

현재는 별도의 SVG 뷰어를 설치해야 하지만 XML 기술이 점점 보편화됨에 따라 모든 웹 브라우저에서 SVG 뷰어가 구현될 것으로 예상된다.

(1) 프리젠테이션 스타일시트

그림 3.14는 생성된 SVG 문서가 웹 브라우저에 표시될 때 적용될 프리젠테이션 스타일시트의 구조이다. 스타일시트에서 지정하는 요소로는 크게 채움(fill) 색깔, 외곽선(stroke) 색깔, 외곽선 두께를 지정할 수 있다. 그림 3.15는 프리젠테이션 스타일시트의 일부로서 SVG 문서의 요소에 정의된 클래스 명과 프리젠테이션 스타일시트의 이름이 일치되면 선언된 스타일 요소가 적용된다.

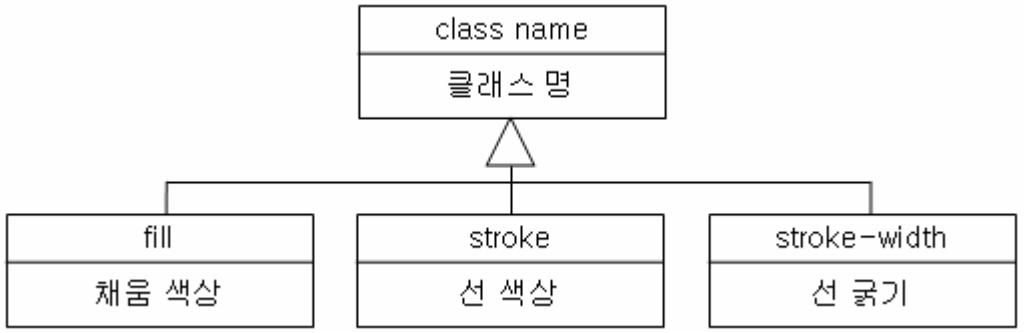

그림 3.14 SVG 문서의 프리젠테이션을 위한 스타일시트의 구조

Fig. 3.14 Structure of Stylesheet for SVG Document Presentation

```
.DepthArea
{
fill:blue;
stroke: blue;
stroke-width:0.2%;
}
.LandArea
{
fill:tan;
stroke: tan;
stroke-width:0.2%;
}
.Light
{
fill:yellow;
stroke: red;
stroke-width:0.2%;
}
```

그림 3.15 SVG 문서의 프리젠테이션 스타일시트 예

Fig. 3.15 Example of Stylesheet for SVG Document Presentation

제 4 장 전자해도 시스템의 구현 및 실행 예제

본 장에서는 제안한 전자해도 시스템을 실제로 구현하고, 구현 결과를 예제를 통해서 설명한다.

4.1 구현 환경

본 논문에서 제안하는 전자해도 시스템의 구현 환경은 표 4.1과 같다.

표 4.1 구현 환경
Table 4.1 Implementation Environment

구 분	구 성	
하드웨어	웹 서버	HP PLOIANT 5000
	DB 서버	HP PLOIANT ML370
	클라이언트	펜티움4 1.7GHz 데스크탑
운영체제	서버	Windows Server 2000
	클라이언트	Windows XP Professional
데이터베이스	SQL Server 2000	
웹 브라우저	Internet Explorer 6.0	
심볼 편집기	SVG Factory 1.0	
개발 언어(Tool)	Visual Studio C#.Net 2003	

4.2 구현된 시스템

본 절에서는 구현된 시스템의 전체 구성 및 각 모듈에 대한 기능을 설명한다.

(1) 시스템 구조도

구현된 시스템의 구성은 다음과 같다.

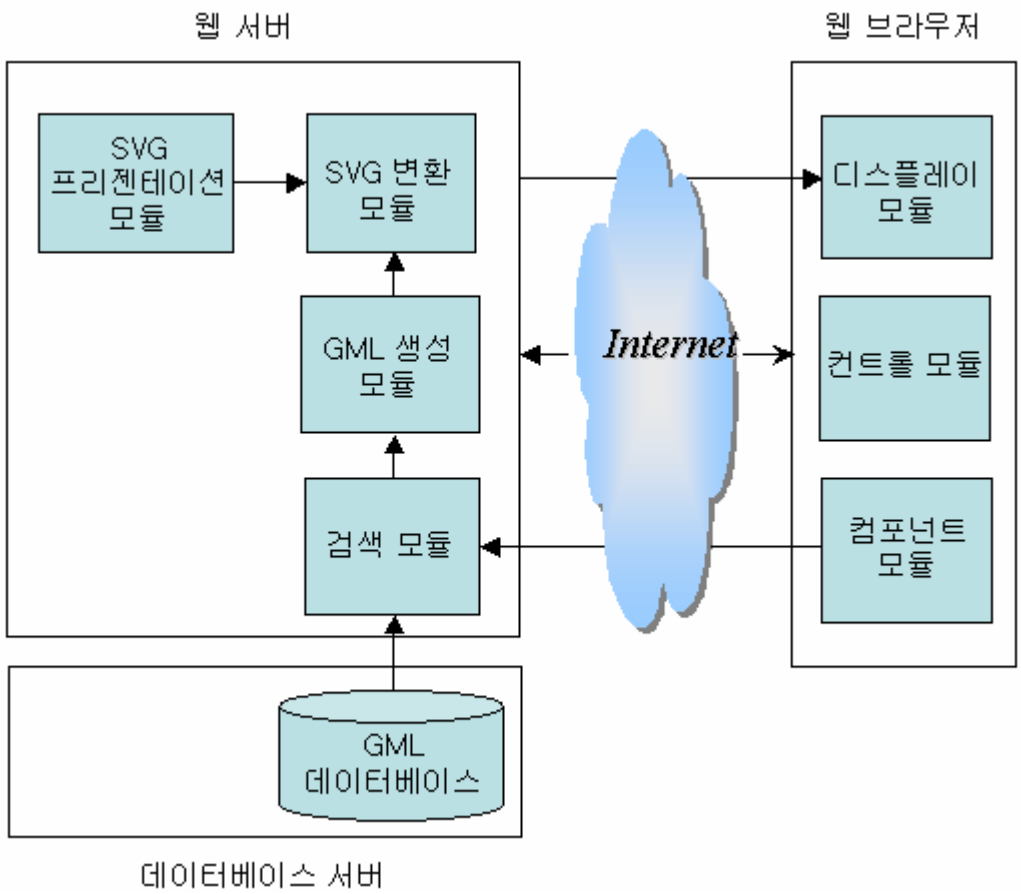

그림 4.1 구현된 시스템의 전체 구조

Fig. 4.1 Overall Structure of Implemented System

(2) 모듈 설명

본 논문에서 구현한 시스템은 크게 클라이언트, 웹 서버, 데이터베이스 서버로 구성되며, 웹 서버상에는 GML 문서를 SVG 문서로 변환하기 위한 미들웨어가 탑재되어 있다.

- **클라이언트**

클라이언트의 디스플레이 모듈은 SVG 문서를 화면에 출력하는 부분으로 인터넷 익스플로러와 같은 웹 브라우저가 사용된다. 컨트롤 모듈은 출력된 지리정보를 제어하기 위한 모듈로서 줌 기능, 패닝 기능 등을 제공한다. 컴포넌트 모듈은 클라이언트가 웹 서버에 검색을 지시하기 위한 모듈로서 전자해도 이름과 객체 종류를 선택할 수 있다.

- **서버**

서버는 웹 서버와 데이터베이스 서버로 구성된다. 웹 서버는 클라이언트와 데이터베이스 서버 간의 질의 검색을 처리하기 위한 웹 기반 사용자 인터페이스를 제공하며, 데이터베이스 서버는 GML 전자해도 정보를 관계형 데이터베이스로 저장한다.

- **미들웨어**

검색된 전자해도를 클라이언트에게 보여주기 위해 웹 서버상에는 미들웨어가 탑재되어 있다. 미들웨어는 크게 검색 모듈, GML 생성 모듈, SVG 변환 모듈, SVG 프리젠테이션 모듈로 구성된다.

검색 모듈은 전자해도 데이터베이스 서버로부터 사용자가 원하는 정보를 추출하기 위한 모듈로서 클라이언트 상의 컴포넌트 모듈을 통해 웹 서버에 지시

한다. 지시에 따라 검색된 결과는 GML 생성 모듈을 통해 새로운 GML 문서로 재구성된다. 생성된 GML 문서는 SVG 변환 모듈을 통해 스키마 분석 스타일시트, 피쳐 스타일시트, 그래픽 스타일시트를 적용 받아 SVG 문서를 생성한다. 마지막으로 SVG 프리젠테이션 모듈은 생성된 SVG 문서가 웹 브라우저에 출력될 때 사용자에게 보여질 프리젠테이션 스타일을 정의하는 모듈이다.

(3) 객체에 대한 심볼 정의

GML 문서를 SVG 문서로 변환할 때 GML 문서의 선과 면은 SVG로 즉각 변환이 가능하다. 반면에 등대, 부표 등과 같은 객체에 대한 그래픽 정보는 GML에서 별도로 지정하지 않을 뿐만 아니라, 표현하더라도 GML 문서와 SVG 문서의 용량이 대폭으로 증가한다. 이를 해결하기 위해 S-57에서 정의한 각각의 객체에 대한 그래픽을 미리 비트맵으로 작성한 후에 심볼 편집기를 통해 SVG로 변환하였다. 변환된 SVG 정보를 심볼화시켜 해당 객체의 좌표 위에 출력되도록 하였다. 그림 4.2는 등대를 표현한 비트맵을 SVG 문서로 변환한 문서를 보여준다.

하지만 본 기능은 현재 버전의 SVG 뷰어에서는 정상적으로 동작하지 않는다. SVG 문서 안에 다른 SVG 문서를 삽입하여 심볼화한 경우 심볼이 정상적으로 출력되지 않는 문제가 발생했다. 따라서 제안하는 심볼화 방법은 심볼 기능을 정상적으로 지원하게 되는 차기 버전에 도입하고자 한다. 대신 현재 버전에서는 심볼 대신 사각형 요소로서 심볼을 대체하였다.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
"http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303-
stylable.dtd">
```

```
<svg viewbox = "-32.500000 60.866667 0.04999999999716 0.10000000000142">
  <g id="Light">
    <image width="15.000000" height="33.000000" xlink:href="
iVBORw0KGgoAAAANSUhEUgAAAA8AAAAhCAIAAACJEmZbAAAAfUlEQVR42mP4Tw
fRCDdVBORw0KGdANSR7ziBAAA8OKDSDFJAhCAqeFDCJEEZbGFKfUqwEVR12FWZC
Y5qTA04VWMajy+ dYFVNbBokSjVdANSR7ziBsmtiyipaqIRooKzUZhNOIVw0AvR7ziBtoG
RCDdVBORw0KGdANSR7ziBAAA8OKDSDFJAhCAqeFDCJEEZbGFKfUqwEVR12FWZC
W4AAAAASUVORK5CYII=" />
  </g>
</svg>
```

그림 4.2 등대 객체의 심볼 정의

Fig. 4.2 Symbol Definition of "Light" Object

4.3 실행 예제

본 논문에서 예제로 사용한 GML 전자해도는 갈도스사에서 제공한 GB5X01NW를 사용하였다. GB5X01NW 전자해도는 25,000:1의 해상도를 가지며, 레코드 구성은 표 4.2와 같고 S-57에 정의된 객체 중 69개가 사용되었다.

표 4.2 GB5X01NW 의 레코드 수

Table 4.2 Number of Records in GB5X01NW

레코드 구분	레코드 수
Meta Records	9
Geo Records	597
Collection Records	5
Isolated Node Records	103
Connected Node Records	763
Edge Records	1033

(1) 전자해도 및 객체 검색

객체 검색은 클라이언트의 컴포넌트 모듈을 통해 입력이 이루어지며, 웹 서버의 미들웨어가 데이터베이스에 접근해 검색 결과를 GML 문서로 생성한다. 그림 4.3는 클라이언트의 검색 화면, 변환된 검색 질의문, 질의에 대한 DB 검색 결과를 보여준다.

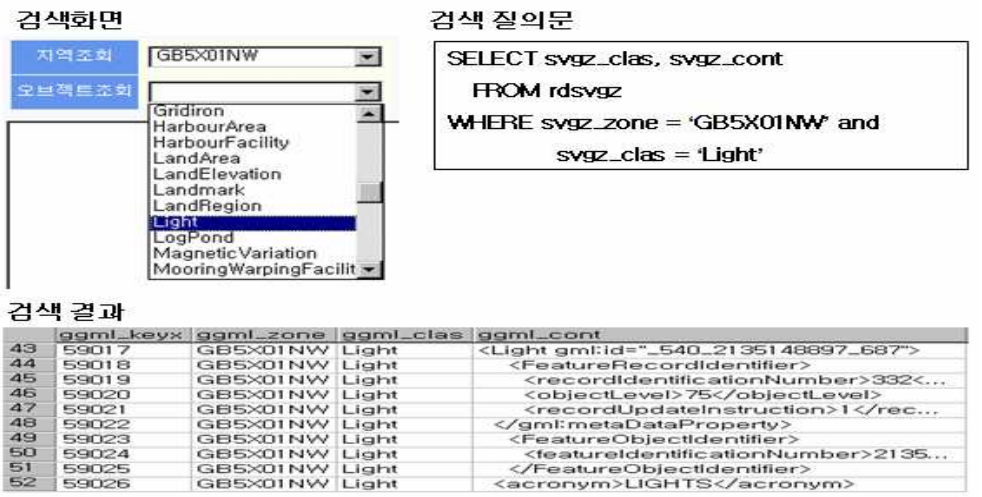

그림 4.3 객체 검색 및 결과
Fig. 4.3 Search Object and Result

(2) 생성된 GML 문서

그림 4.4는 검색 결과에 대해서 웹 서버의 미들웨어가 GML 문서를 생성한 결과를 보여준다. 생성된 GML 문서는 객체에 대한 메타 정보와 속성 정보를 포함한다. 메타 정보는 객체의 식별 번호 등에 대한 정보를 제공하며, 속성 정보는 객체에 대한 공간 속성과 비공간 속성 정보를 제공한다. 공간 속성은 객체의 위치 정보를 포함하며, 비공간 속성은 객체의 크기나 범위 등의 일반적인 정보를 포함한다.

```
<Light gml:id="_540_2135148864_687">
  <gml:metaDataProperty>
    <FeatureObjectIdentifier>
      <agency>540</agency>
      <featureIdentificationNumber>2135148864</featureIdentificationNumber>
    </FeatureObjectIdentifier>
  </gml:metaDataProperty>
  <acronym>LIGHTS</acronym>
  <catlit>
    <CategoryOfLight>
      <code>37</code>
      <value>air obstruction light</value>
    </CategoryOfLight>
  </catlit>
  <colour>
      <Colour>
        <code>75</code>
      </Colour>
  </colour>
  <position>
    <gml:Node gml:id="N120_668">
      <gml:pointProperty>
        <gml:Point gml:id="P120_668">
          <gml:pos>-32.498195 60.895926</gml:pos>
        </gml:Point>
      </gml:pointProperty>
    </gml:Node>
  </position>
</Light>
```

그림 4.4 생성된 GML 문서

Fig. 4.4 Generated GML Document

(3) 변환된 SVG 문서

그림 4.5는 생성된 GML 문서에 대해서 XSLT를 통해 SVG 문서로 변환된 결과를 보여준다. 변환된 SVG 문서에는 검색된 객체 외에 추가적으로 바다와 육지에 대한 정보를 포함한다. 이는 검색된 객체만을 화면에 출력할 경우 객체의 위치 등과 같은 정보를 한눈에 알기 힘들기 때문이다.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<?xml-stylesheet type="text/css" href="gml2svg_styles.css"?>
<svg width="100%" height="100%"
   viewBox="-32.500000 60.866667 0.049999999999716 0.100000000000142">
  <desc> Converted to SVG using GMLtoSVG.xsl</desc>
  <g id="DepthArea" class="DepthArea">
    <path id="DEPARE" d="M-32.493758 60.928285 -32.493763 60.932597 -
32.493758 60.933328 -32.493182 60.933529 -32.493308 60.933328z" />
    <path id="DEPARE" d="M-32.500000 60.884642 -32.500000 60.883333 -
32.499688 32.500000 60.884642 -32.499882 60.884628z" />
  </g>
  <g id="LandArea" class="LandArea">
    <path id="LNDARE" d="M-32.499722 60.898657 -32.498594 60.898996 -
32.498594 60.898657z" />
    <path id="LNDARE" d="M-32.497387 60.914768 -32.499326 60.916941 -
32.499326  60.914922 -32.497251 60.914922 -32.497387 60.914768z" />
  </g>
  <g id="Light" class="Light">
  <rect x="-32.498195" y="60.895926" height = "0.001" width = "0.001"/>
  <rect x="-32.495481" y="60.892104" height = "0.001" width = "0.001"/>
  <rect x="-32.491039" y="60.931863" height = "0.001" width = "0.001"/>
</svg>
```

그림 4.5 변환된 SVG 문서

Fig. 4.5 Transformed SVG Document

(4) SVG 문서의 브라우징

그림 4.6은 등대(Light) 객체에 대해서 변환된 SVG 문서가 웹 브라우저를 통해 사용자에게 출력된 결과를 보여준다. 그림 4.7은 길(Road) 객체가 웹 브라우저에 출력된 결과를 보여준다. 변환된 SVG 문서는 SVG 플러그인을 통해 사용자에게 보여지며, 검색된 GML 문서와 변환된 SVG 문서도 정보로서 제공한다.

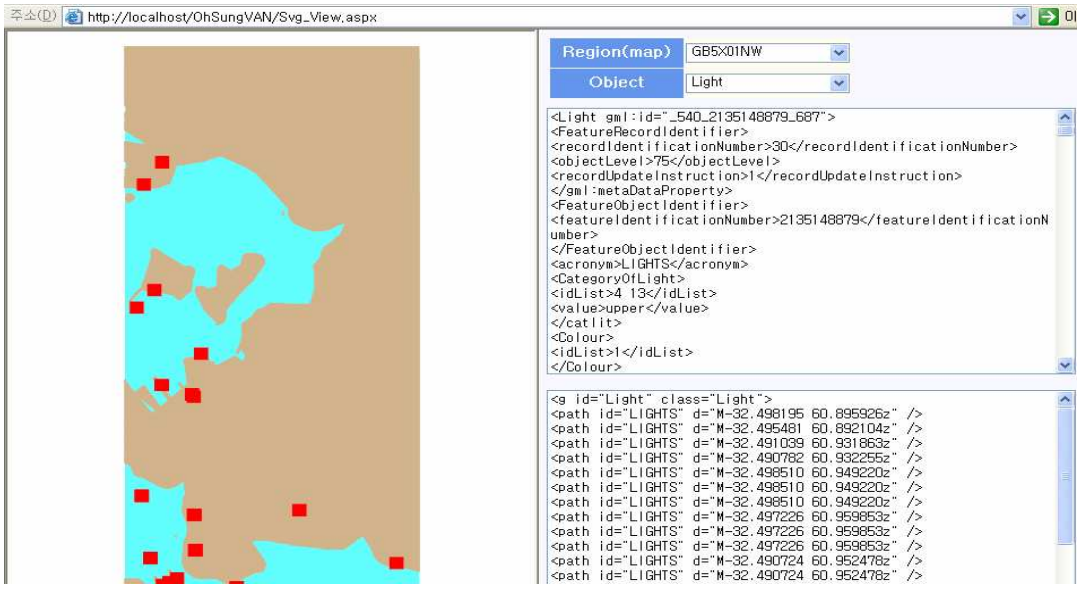

그림 4.6 SVG 문서의 브라우징 예 I

Fig. 4.6 Example I of SVG Document Browsing

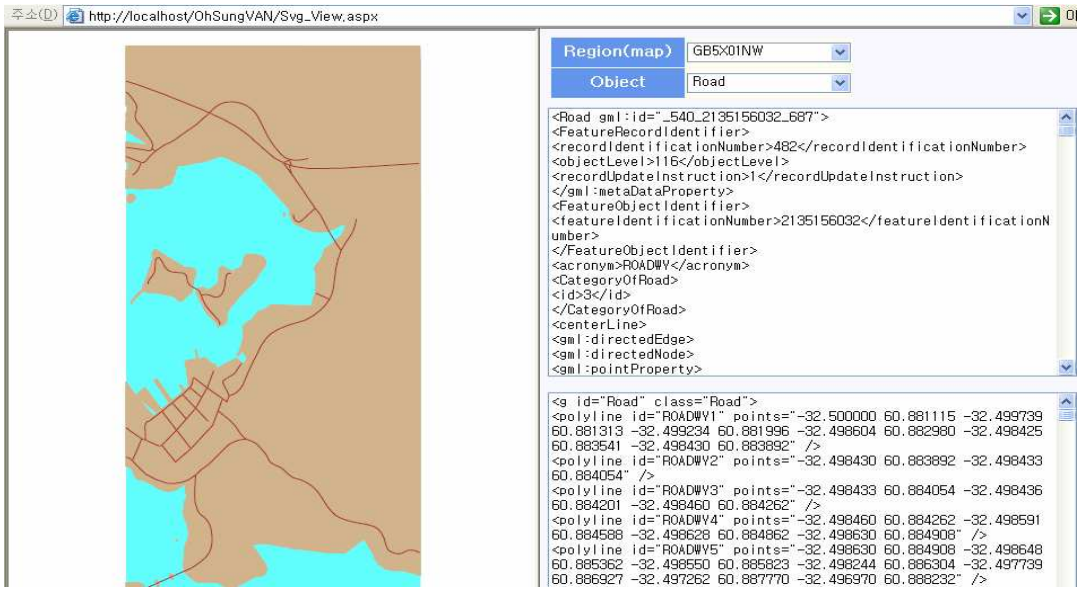

그림 4.7 SVG 문서의 브라우징 예 II

Fig. 4.7 Example II of SVG Document Browsing

제 5 장 결론 및 향후 연구 과제

기존의 전자해도는 특수한 데이터 포맷과 특수한 표시시스템을 사용하였기 때문에 선박의 운항에만 제한적으로 활용되어 왔다. 하지만, 전자해도에는 일반인들이 관심을 가질 수 있는 바다와 연안에 대한 수 많은 정보를 포함하고 있으며, 다양한 다른 시스템과의 정보 교환 등의 필요성이 부각되고 있다. 이러한 요구에 부응하기 위해서는 전자해도 데이터를 범용의 데이터 포맷으로 표현할 필요가 있고, 웹과 같은 접근성이 뛰어난 사용자 인터페이스를 제공할 필요가 있다.

본 논문에서는 S-57 전자해도 표준, GML 지리 정보 표준, SVG 벡터 그래픽 표준 등 세가지 표준을 따르는 전자해도 시스템을 설계하고 구현하였다. 본 논문에서 제안하는 방법을 통해 다음과 같은 장점을 얻을 수 있다.

① 인터넷 기반의 서비스로 범용의 전자해도 시스템 구축이 가능하다.
② 전자해도 데이터를 데이터베이스로 구축하여 전자해도의 효과적인 관리와 검색이 용이하다.
③ 데이터를 GML로 재구성함으로써 다른 지리 정보 시스템과의 정보 교환이 용이하다.
④ XSLT를 이용하여 벡터 기반의 그래픽 표준인 SVG로 변환하여 웹 브라우저를 통한 출력이 가능하다.
⑤ 벡터 그래픽을 통한 래스터 이미지의 한계를 극복할 수 있다.

향후에는 보다 빠른 검색과 개선된 사용자 인터페이스를 제공할 수 있도록 연구하고, 데이터의 보안성을 향상 시킬 수 있는 방안과 전자해도 데이터의 갱

신에 따른 데이터베이스의 수정과 다른 지리 정보 시스템과의 연계방안이 연구될 필요가 있다.

참고 문헌

[1] M. B. Brown, Developments in The NOAA Electronic Navigational Chart Program, NOAA, Office of Coast Survey, Marine Chart Division, U. S. Hydrographic Conference, 1999.

[2] International Hydrographic Bureau, IHO Transfer Standard for Digital Hydrographic Data, Edition 3.1, 2000.

[3] OpenGIS, Geography Markup Language (GML) Version 3.0, http://www.opengeospatial.org, 2003.

[4] W3C, Scalable Vector Graphics (SVG) Version 1.1, http://www.w3.org/TR/SVG11, 2003.

[5] W3C, XSL Transformations (XSLT) Version 2.0, http://www.w3.org/TR/xslt20, 2005.

[6] W3C, XML Path Language(Xpath) Version 1.0, http://www.w3.org/TR/xpath, 1999.

[7] 한국전자통신연구원, e-Logistics 문서 변환 및 출력 기술 개발에 관한 연구, 2004.

[8] 이성대, 강형석, 박휴찬, “전자해도용 XML 스키마의 정의 및 변환”, 한국해양정보통신학회논문지, 제8권, 제1호, pp. 200-212, 2004.

[9] 이성대, 곽용원, 박휴찬, “객체관계형 데이터베이스에 기반한 XML문서 저장 및 검색 시스템의 설계 및 구현”, 한국해양정보통신학회논문지, 제7권, 제2호, pp. 183-193, 2003.

[10] Galdos Inc, S-57 Schema and Related Tools Manual, S-57/GML Project, 2004.

[11] Vitor Rodrigues, GML to SVG using XSL, http://siglab.di.uminho.pt, 2004.

[12] W.T.M.S.B.Tennakoon, Visualization of GML data using XSLT, International Institute for Geoinformation Science and Earth Observation, 2004.

[13] 한국전산원, 지리공간 정보 엔코딩(Encoding) 표준 개발에 관한 연구, 2002.

[14] 감승철, 이성대, 곽용원, 박휴찬, “GML과 SVG에 기반한 전자해도 시스템의 설계 및 구현”, 한국정보과학회 2005 추계학술대회, pp. 127-129, 2005.

감사의 글

2년이라는 시간이 흘러 논문을 마무리하는 시점에 뒤돌아보니 감사를 드려야 할 분들이 참으로 많습니다. 먼저 본 논문이 완성되기까지 아낌없는 가르침으로 이끌어주신 박휴찬 교수님께 진심으로 감사를 드립니다. 교수님의 세심한 배려와 격려가 없었다면 본 논문의 완성은 어려웠을 것입니다. 또한 지도 교수님의 빈자리를 세심한 지도와 관심으로 돌봐주신 김재훈 교수님께 감사 드립니다. 그리고 바쁘신 중에도 본 논문이 작은 결실을 맺을 수 있도록 많은 가르침을 주신 류길수 교수님과 신옥근 교수님께도 감사를 드립니다. 학부 과정에서부터 많은 관심을 가져주신 손주영 교수님, 석사 과정에서 짧지만 많은 지식을 전달해주신 이장세 교수님께도 감사드립니다.

석사과정 동안 때로는 친구로 때로는 조언자로 본 논문이 완성되기까지 가장 큰 힘이 되어 주신 이성대 형님께 진심으로 감사의 마음을 전합니다. 또한 같은 파트타임으로서 지쳐있을 때 아낌없는 조언을 해주신 곽용원 형님, 커다란 학업의 성취 있으시길 바랍니다. 타지에서 학업에 열중하시는 이주현님, 함께 졸업하지 못해 아쉬움이 더 큰 유흥섭님과 박철현님께도 고마움을 전합니다. 앞으로 데이터베이스 연구실을 이끌어 갈 전성환님께도 감사와 건투를 보냅니다.

학부 때부터 지금까지 변함없이 많은 도움을 주신 강군호 조교님, 김경언 조교님께 고마움을 전합니다. 한마디 한마디가 모두 귀감이 되시는 음성처리 연구실의 하동경 형님과 박정임 형수님 감사 드립니다. 본 논문의 구현을 위해 아낌없이 조언해주신 유강주 형님과 남언규 형님께도 고마움을 전합니다. 인공지능 연구실의 듬직한 기둥이 되시는 박종일님과 원라경님, 김태진님께도 감사함을 전합니다. 나날이 발전하는 네트워크 연구실의 문성미님과

자연언어처리 연구실의 박은진님, 김강민님께도 감사 드립니다. 그 외 학부와 대학원 생활을 함께 한 모든 선배님, 후배님, 동기들에게 고마움을 전합니다. 특히나 지난 10년 동안 힘들 때마다 옆자리를 지켜준 친구이자 동기인 우진식님에게 고마움을 전합니다.

2년 동안 직장 생활과 학교 생활을 병행함에 있어 많은 배려를 해주신 윤영재 부장님, 김현찬 과장님께 감사 드립니다. 모든 일에 한발자국 먼저 전진하시는 직장 선배이면서 학교 선배인 김정우 대리님께도 고마움을 전합니다. 학교 생활하는 동안 빈자리를 채워준 최은옥씨에게도 이 자리를 빌어 고마움을 전합니다.

이제는 한 아이의 어머니가 된 첫째 여동생 현숙이와 나이 어린 형님을 불편하지 않게 대해준 매제, 막내답지 않은 막내 현정이의 사랑과 격려 덕분에 무사히 대학원 생활을 마칠 수 있었습니다. 또한 늘 사랑하는 마음으로 세상을 살아가게 해주는 그녀에게도 고마움과 사랑을 전합니다. 마지막으로 크신 사랑과 헌신으로 보살펴 주신 아버님과 어머님께 이 논문을 바칩니다. 감사합니다.