통신망 표현을 위한 확장용이 GUI의 객체 지향적 설계 및 구현 (Object-Oriented Design and Implementation of Extensible GUI for Presentation of Communication Network)

요약

본 논문에서는 객체 지향 기법을 사용하여 확장이 용이한 통신망 표현 GUI의 설계 및 구현 방법을 제시한다. 이를 위해 통신망을 구성하는 여러 요소들 및 통신망 관련 정보를 사용자에게 보여주는 여러 GUI 요소들을 객체 지향 기법을 사용하여 모델링 하였다. 그리고, 다른 시스템으로의 확장 가능성을 예를 통하여 보였다. 이 GUI는 확장이 용이하고 유지 관리가 편리하므로, 향후 제작될 통신망 표현 GUI들을 위한 참조 모델로서 제시하고자 한다. [1]

Abstract

This paper proposes a design and implementation of extensible GUI which represents communication network using object-oriented technique. We have modeled major elements of communication network and GUI using object-oriented technique, and demonstrated the extensibility into other system by showing examples. The proposed framework provides extensibility and maintenability, so this system can be a reference model for future GUIs of communication network.

[1] 본 논문은 한국 통신 연구 개발단의 학술 용역 연구 과제의 지원을 받아 작성된 것입니다.

1 서론

현재 우리 나라에는 전국적으로 약 800여개의 교환기가 설치되어 있고, 이 교환기들 사이에는 많은 통신 선로가 연결되어 전체적으로 통신망을 형성하고 있다. 전화 및 컴퓨터 통신의 일반화로 인해 이 통신망들 사이에는 항상 많은 양의 통신 부하가 걸리고 있고, 이렇게 통신 부하량이 증가함에 따라 통신망의 현재의 상태를 알아내어 통신망을 적절히 관리하는 것이 주요한 업무가 되었다. 이와 같이 기존의 통신망의 운용 효율을 극대화하고 통신망의 신뢰도를 향상시키며 다양한 통신망의 설계, 구축 단계로부터 망운용, 유지/보수 단계에까지의 종합적이고 체계적인 통신망의 관리를 위해서는 통신망의 구조 및 기능을 표준화된 모델에 따라 표현할 수 있어야 한다.[10,13,19]

선진국에서는 이러한 통신망의 계획, 설계 및 관리에 필요한 도구 프로그램을 그래픽 사용자 인터페이스(Graphical User Interface: 이하 GUI)를 통하여 개발, 활용하고 있다. 하지만, 국내에서는 각 단계별 특정 분야에 관한 도구 프로그램은 부분적으로 개발, 활용하고 있으나 전체 단계에 걸쳐 일관적이고 체계적으로 사용될 수 있도록 통신망의 구조를 모델링하여 주는 기법이 없는 실정이었다.

본 연구에서는 이러한 필요에 맞추어 전체 통신망의 상태를 일관적이고 체계적인 방법으로 사용자에게 모델링하여 줄 수 있으면서 또한 새로운 기능 요구를 쉽게 추가할 수 있도록 확장성을 갖춘 GUI의 작성을 위해 필요한 기반 기술을 연구하였고, 이 연구를 토대로 통신망의 상태를 모델링하여 주는 GUI 프로그램을 설계, 개발하였다.

먼저, 통신망의 표현 구조를 모델링하기 위하여 현재 각광받고 있는 객체 지향 기법을 사용하였다. 객체 지향 기법은 시스템을 구성하는 요소들을 클래스(class) 개념을 이용하여 모델링하고 이들간의 상호 작용을 통해 시스템을 구성할 수 있도록 해 주는 기법이다. 이 객체 지향 기법을 사용하여 통신망을 구성하는 여러 요소들 - 전화국, 중계소, 통신 케이블 등- 을 노드와 링크 클래스로서 모델링하였다. 그리고 , 노드와 링크를 화면에 표시할 때 관련 위치를 알기 쉽도록 하기 위해 지도를 함께 보여주어야 하는데, 이를 위해 지도 관련 기능을 수행하는 지도 클래스를 모델링하였다. 그리고, 이러한 정보들을 종합적으로 화면에 출력하여 주고, 사용자로부터의 명령을 입력받아 전체 통신망을 알기 쉽게 표현해 주고, 관련 정보들을 사용자에게 보여 주는 기능을 수행하는 GUI 윈도우 클래스를 모델링하였다. 또, 사용자에게 노드 및 링크 관련 상세 정보를 보

여 주기 위해 필요되어지는 GUI 요소들을 관련 클래스로서 모델링하였다.

본 시스템은 C++ 및 X & Motif 환경을 기반으로 작성되었다. 현재 객체 지향 기법 및 GUI의 확산 추세로 미루어 볼 때, C++ 언어 및 X & Motif 환경을 기반으로 한 시스템이 많이 작성될 것으로 예상된다. 본 시스템은 앞으로 작성될 통신망 관련 GUI의 설계와 구현에 대한 하나의 참조 모델로서 사용되어질 수 있을 것이다.

본 논문은 2장에서 객체 지향 및 GUI의 관련 기술에 대해서 서술하고, 3장에서는 통신망 표현 GUI를 작성하는데 있어서의 고려점들을 살펴 본다. 4장에서는 본 논문에서 제안한 통신망 표현 GUI의 객체 지향적 구성 방법과 이의 장점에 대해서 살펴 보고, 5장에서는 이러한 구조로서 만들어진 시스템의 기능 및 확장성에 대해서 살펴 본다. 6장에서는 시스템의 현재 상태 및 향후 연구 계획을 살펴 보고, 7장에서 결론을 맺는다.

2 관련 기술 및 연구

2.1 관련 기술

2.1.1 객체 지향 기술

객체 지향 기술은 시스템을 구성하는 요소들을 독립적인 객체로서 파악하고, 이들 객체들 사이의 상호 관계를 기술함으로써 시스템을 구성하는 기법이다. 이를 위해 클래스(class), 상속성(inheritance), 동적 바인딩(dynamic binding) 등의 개념을 제공한다.[2,3,8]

클래스는 같은 속성과 같은 행동 방식을 가지는 객체들을 종합적으로 표현하는 개념이다. 시스템 모델링시에 시스템을 구성하는 요소들은 클래스로서 모델링된다. 이 클래스 안에는 자료와 함수가 있어서 자료는 객체의 수행 상태를, 함수는 객체의 행위 내용을 기술하게 되는데, 이를 데이타 멤버(data member) 및 멤버 함수(member function)라고 부른다.

상속성은 객체 지향 개념이 재사용성(reusability) 및 확장성(extensibility)을 가지도록 해 주는 중요한 개념이다. 시스템 모델링시에 클래스들을 보다 일반적인 개념을 가지는 클래스와 보다 자세한 개념을 가지는 클래스로 구성해서 이들 사이에 상속성 관계를 명시할 수 있는데, 이때, 일반적이면서 자신의 내용을 상속해 주는 클래스를 수퍼 클래스(super-class) 또는 부모 클래스(parent class), 보다 자세하면서 일반적인 내용을 상속받는 클래스를 서브 클래스(sub-class) 혹

은 자식 클래스(child class)라고 한다. 자식 클래스는 부모 클래스에 속하는 내용은 중복 정의할 필요없이 자신만의 데이타 멤버와 멤버 함수만을 정의하면 된다. 이러한 상속성의 개념을 통해 클래스의 재사용과 확장성이 가능해진다.

동적 바인딩은 클래스 상속 계층 구조 상에 존재하는 클래스들이 같은 이름의 멤버 함수를 가져서, 수행시(run-time)에 실제로 사용되는 함수를 지정하여 사용할 수 있도록 하여 시스템의 유연성을 높여주는 개념이다.

2.1.2 GUI

GUI는 시스템이 메뉴 및 아이콘 등의 그래픽 요소들을 갖추어서, 사용자가 이들 요소들을 통해 시스템을 보다 편리하게 사용할 수 있도록 해 주는 것을 목표로 한다. 기존의 명령어 입력 방식에 비해 명령어를 일일이 외울 필요가 없으며, 타이핑 노력을 줄여 주며, 메뉴 또는 아이콘을 선택만 하면 되므로 쉽게 시스템을 사용할 수 있는 장점이 있다.[1,5,6,14]

GUI는 이렇듯 쉽게 시스템을 사용할 수 있도록 해 주는 장점 때문에 많은 시스템이 프로그램과 사용자 사이의 중간의 인터페이스로서 사용하고 있으며, 최근에는 시스템 구성에 있어 거의 필수적인 요소로서 인식되고 있다.

2.2 관련 연구

현재 외국에서 개발되어 사용중인 통신망 표현 GUI들은 GUI 시스템 자체로서만 존재하는 것이 아니라 통신망을 관리하기 위한 전체적인 플랫폼 안에 존재하는 하나의 구성 요소로서 통신망 관리 시스템과 함께 존재한다. 이러한 관련 시스템들을 살펴 보면 다음과 같다.

Sun의 NetManager는 Manager-Agent 모델에 기반한 망 관리 시스템으로서 SNMP(Simple Network Management Protocol)에 기반한 통신망 관리 기능을 수행하고 있다. NetManager에서 제공되는 GUI는 OpenLook에 기반하고 있으며, Discover, Results Browser, Results Grapher 등의 응용 프로그램들로 구성되어 있는데, 이를 통해 통신망 상의 노드들의 작동 상황을 알려주는 정보들을 수집하고, 이들의 관리 상황을 그래픽 환경으로 보여 주어 통신망을 콘솔에서 관리자가 총괄적으로 관리할 수 있도록 해 준다.

HP의 OpenView는 SNMP에 기반한 망 관리를 위해 사용자 인터페이스, 망관리 응용 프로그램들, 그리고, 여러 라이브러리 함수들로 구성되어 있다. OpenView의 사용자 GUI인 NodeManager

는 Motif에 기반하고 있으며 SNMP Event Monitor, MIB Loader/Browser, Data Presentation Tool 등의 다양한 응용 프로그램들로 구성되어 있다. 이 GUI는 지도의 축소 및 확대 기능, 심볼을 이용한 통신망 표현, 다양한 레벨의 통신망 표현 등의 기능을 가지고 있다.

IBM의 NetView/6000은 SNMP에 기반한 통신망 정보 수집 및 관리 기능을 수행하는 시스템으로, IBM의 SNA 구조 안에서 통신망 관리 부분을 담당한다. 이 시스템은 Motif에 기반하고 있는 GUI를 제공하는데, 이 GUI는 통신망을 멀티 레벨로 표현하여 망의 계층 구조를 살펴볼 수 있도록 해 주며, 정보 수집 및 표현 기능, 경보 기능 등을 가진다.

Dec의 PolyCenter는 EMA(Enterprise Management Architecture)에서 통신망 관리를 담당하는 시스템으로 CMIP(Common Management Information Protocol)과 SNMP를 지원한다. 이 시스템 역시 Motif에 기반한 GUI를 제공하는데, 이 GUI는 아이콘으로 구성된 지도 형태의 인터페이스와 폼과 명령 입력 방식의 인터페이스 두 가지 형태의 통신망 표현 방식을 제공한다. 통신망을 계층적으로 살펴볼 수 있으며 통신망 관련 정보들을 수집 및 출력, 통신망 구조의 자동 인식, 경고 기능 등을 제공하여 준다.

위에서 살펴본 바와 같이 기존의 통신망 표현 GUI들은 별도의 시스템으로 존재하는 것이 아니라 하나의 망 관리 시스템 안에 종속되어 존재한다. 기존의 시스템들에 비해 본 시스템이 가지는 특징은 크게 다음과 같이 정리할 수 있다 :

- 특정 시스템에 종속적이지 않다.
- 망 관리를 위한 다양한 프로토콜 및 모듈과 쉽게 연동될 수 있는 인터페이스 구조를 갖추고 있다.
- 다양한 형태의 응용 프로그램에 대해 확장될 수 있는 구조를 갖추고 있다.

3 통신망 표현 GUI의 구성을 위한 고려 사항

이 장에서는 통신망을 표현하여 주는 GUI를 구성할 때 고려하여야 할 사항들에 대해 살펴 보기로 한다. 이러한 고려 사항들이 어떻게 본 시스템에 반영되었는가에 대해서는 4장에 설명되어 있다.

3.1 구조의 간단성

통신망을 표현하여 주는 GUI는 시스템의 기본 구조가 명확하고 간단하여야 한다. 통신망의 구성 내용을 모델링하여 주는 각각의 모듈들이 독립적으로 구성되어 있고 이들 사이의 상호 연결 관계가 명확하여야만 한다. 그래야만 시스템의 구현과 유지 보수가 쉽고, 다른 관련 응용으로의 확장시에도 쉽게 시스템을 이해할 수 있게 된다.

3.2 사용의 편리성

통신망을 표현하여 주는 GUI는 사용자가 사용하기에 편리한 환경을 갖추고 있어야 한다. 기존의 메뉴를 통한 명령의 처리는 물론이고, 중요한 기능들은 쉽게 사용될 수 있도록 아이콘을 통해 제공되어야만 한다. 또한, 통신망 구성 요소의 표현이 명확하여 사용자가 통신망의 구성 상태를 한 눈에 쉽게 이해할 수 있어야 한다.

3.3 확장 용이성

통신망 표현 GUI가 관련 응용에 널리 사용되기 위해서는 시스템은 쉽게 확장될 수 있는 구조를 갖추어야만 한다. 새로이 필요되어지는 기능이 쉽게 추가될 수 있어야만 다른 형태의 관련 응용에 쉽게 적용될 수 있을 것이다.

4 통신망 표현 GUI의 객체 지향적 구성

4.1 시스템의 구조

4.1.1 시스템 구성 요소

본 시스템은 그림 1에 나타난 대로 프레젠테이션 관리자(presentation manager), 지도 관리자(map manager), 노드 관리자(node manager), 링크 관리자(link manager), 노드 및 링크 객체 등으로 구성된다.

프레젠테이션 관리자는 사용자와 직접 대화하는 윈도우 객체로서, 메뉴 및 아이콘, 작업 영역 등을 갖추고 있으며, 사용자로부터의 명령에 따라 사용자가 원하는 작업 - 통신망 표현, 특정 지역의 선택 및 확대, 노드 및 링크 정보의 검색, 노드 및 링크의 에디팅 등 - 을 수행한다. 이 프레

젠테이션 관리자는 지도 관리자, 노드 관리자, 링크 관리자 등과 상호 연결되어 있는데, 지도 관리자를 통해 화면에 전국의 지도 및 사용자에 의해 선택된 지역의 지도를 사용자에게 보여 주며, 또한, 노드 관리자 및 링크 관리자를 통해 해당 지역에 속한 노드 및 링크를 화면에 출력하여 준다. 또한, 통신망 정보를 사용자에게 보여 주는 다이어로그 객체들을 생성하여 주기도 한다.

지도 관리자는 지도 데이타 화일을 관리하며, 지도 정보를 읽어 들여 좌표 변환 기능을 수행하고, 프레젠테이션 윈도우 상에 전국 지도 및 사용자가 선택한 특정 지역의 확대된 지도를 그려주는 기능을 수행한다.

노드 관리자는 노드 데이타 화일을 관리하며, 노드 정보를 읽어들여 내부적으로 노드 객체의 연결 리스트(linked list)를 유지한다. 또, 사용자에 의해 선택되어진 영역 안에 속하는 전화국 및 중계소 등의 노드들을 프레젠테이션 윈도우 상에 출력하여 주고 해당 노드의 정보를 검색하여 사용자에게 보여 주는 기능을 수행한다.

링크 관리자는 링크 데이타 화일을 관리하며, 노드와 마찬가지로 링크 정보를 읽어 들여 링크 객체의 연결 리스트를 유지한다. 또, 지정된 지도 영역 안에 속하는 광 케이블, 동축 케이블 및 마이크로 웨이브 등의 링크들을 프레젠테이션 윈도우 상에 출력하여 주고, 해당 링크의 정보를 검색하여 사용자에게 보여 주는 기능을 수행한다.

노드 및 링크 객체는 통신망 상에 존재하는 여러 종류의 노드 및 링크들을 모델링하여 주며, 이들이 가지는 각종 정보와 이들 정보를 관리하기 위한 멤버 함수들을 포함한다.

4.1.2 시스템 구성 요소들간의 연결 관계

각 관리자들 사이의 연결은 메시지를 통하여 이루어진다. 프레젠테이션 객체는 사용자로부터의 명령을 해석하여 지도 및 노드, 링크 관리자에게 관련 기능을 수행할 것을 요청하는 메시지를 보낸다. 그러면, 각 관리자들은 요청받은 기능을 수행하고 수행 완료를 알리는 메시지를 프레젠테이션 객체에 보냄으로써 하나의 기능을 수행하게 된다.

프레젠테이션 관리자와 지도 및 노드, 링크 관리자 사이의 메시지의 전달 관계는 그림 2와 같다.

그림 1: 시스템의 구성

그림 2: 관리자들 사이의 메시지 전달 관계

4.1.3 본 시스템 구조의 장점

위와 같이 시스템을 구성함으로써 본 통신망 표현 GUI는 앞서 살펴 본 고려 조건을 다음과 같이 만족하고 있다.

구조의 간단성 본 시스템은 통신망을 표현하기 위해 꼭 필요한 요소들만을 독자적인 자료 구조와 함수를 갖춘 독립적인 객체로서 모델링하였으며, 또, 이들 객체들간의 상호 작용을 메시지 개념을 이용하여 객체 지향적으로 명시함으로써 시스템 구성 및 작동 구조가 간단, 명확하다. 또, 여러 함수들이 기능적으로 집중되어 광역 자료들을 사용하는 형태의 시스템 구성에 비해 본 시스템은 구조가 기능별로 모듈화되어 있어 시스템을 이해하기가 쉽다.

사용의 편리성 프레젠테이션 객체는 윈도우로서 구성되어 있고 메뉴 뿐만 아니라 아이콘을 제공하고 있으며, 사용자가 명령을 선택할 때마다 마우스 사용 관련 정보를 상태 메시지를 통해 알려 주므로 시스템의 사용이 편리하다.

확장 용이성 시스템 모듈이 각각 독립적인 객체들로 구성되어 있으므로 기능의 구분이 명확하고, 새로운 기능의 추가는 해당 기능과 이에 필요한 자료 구조를 포함하는 독립적인 새로운 모듈만을 추가하고 기존 모듈들과의 메시지 전달 관계만을 명시함으로써 쉽게 구현될 수 있다. 이것의 구체적인 예는 5.2절에 예시하였다.

4.2 시스템을 구성하는 클래스들

이 절에서는 본 시스템을 구성하고 있는 클래스들에 대해 살펴 보기로 한다. 본 시스템을 구성하는 클래스들의 계층 구조는 그림 3에 나와 있다.

4.2.1 PresentationWindow 클래스

PresentationWindow 클래스는 프레젠테이션 관리자 기능을 수행하는 클래스이다. 이 클래스는 사용자 명령에 따라 다른 관리자들의 처리 함수를 호출하여 주는 함수들, 그리고, 통신망 표현을 위해 필요되어지는 화면(작업 영역)의 구성을 위한 함수들을 정의하고 있다. 이 클래스의 객체들은 메뉴바, 아이콘 리스트, 정보 메시지 패널, 안내 메시지 패널, 지도와 노드 및 링크의 출력을 위한 작업 영역들을 갖춘 윈도우 구조를 갖는다.

그림 3: 클래스 계층 구조의 구성

PresentationWindow 클래스가 같는 주요 데이타 멤버들과 멤버 함수들의 구조는 다음과 같다 :

```
class PresentationWindow : public MainWindow {
    PresentationWindow   *prev_win;     // pointer to previous PresentationWindow object
    Map                  *display_map;  // pointer to associated Map object
    Widget menu, status_bar, message_bar, icon_list, work_area;
          //  declaration of Motif widgets needed for construction of PresentationWindow
    float scale,total_scale;            // window's scale factor
    short win_height, win_width;        // window's size
  protected:
    virtual Widget createWorkArea ( Widget ); // Creates the label
  public:
    PresentationWindow( char * );       // constructor
    ~PresentationWindow();              // destructor
    void    SetWindowSize(Position,Position);    // to size window
    void    InitMenuPanel();            // to initialize menu structure
    void    InitStatusPanel();          // to initialize status bar
    void    InitMessagePanel();         // to initialize message bar
    void    InitIconPanel();            // to initialize icon list
    void    InitWorkPanel();            // to initialize working area
    void    DrawPresentation();         // to draw presentation of network
    void    EditNetwork();              // to edit network
```

```
    void    ShowInformation();          // to show information of network
};
```

4.2.2 NodeManager 클래스

NodeManager 클래스는 노드 관리자 기능을 수행하는 클래스로서 노드(전화국,중계소) 객체를 처리하기 위해 필요한 자료 구조와 함수들을 포함한다.

NodeManager 클래스는 노드 데이타 화일을 관리하며, 노드 데이타 화일로부터 노드 정보들을 읽어들여 이를 노드 객체들에 설정한 뒤, 노드 객체들을 연결 리스트로서 구성하여 내부적으로 관리한다. PresentationWindow 클래스의 윈도우의 요청에 따라 이 윈도우에 나타난 지도 상의 영역에 포함되어지는 노드들을 선택하여 윈도우 화면에 출력하여 준다. 사용자가 작업중에 새로운 노드를 생성하면 생성된 노드를 연결 리스트에 삽입하여 주고, 기존의 노드를 삭제하면 선택된 노드를 연결 리스트로부터 삭제하여 준다. 그리고, 최종적으로 연결 리스트에 있는 노드 구조를 노드 데이타 화일에 저장하여 준다.

```
class NodeManager {
    Node        *nodes;                 // pointer to nodes read from node file
    Node        *last_node;             // pointer to last node
    unsigned    node_num;               // current node number in array
    int         nodes_modified;         // flag
    int         data_loaded;            // flag
  public:
    NodeManager ();                     // constructor
    virtual ~NodeManager ();            // virtual destructor
    ReadFile();                         // read node info.
    WriteFile();                        // write node info.
    void Draw();                        // draw nodes
    void DrawStation();                 // draw station
    void DrawRelay();                   // draw relay
    void ShowInformation();             // show node info.
    void CreateNode();                  // create node
    void ModifyNode();                  // modify node info.
    void DeleteNode();                  // delete node
};
```

4.2.3 LinkManager 클래스

LinkManager 클래스는 링크 관리자 기능을 수행하는 클래스로서 링크(광 케이블, 동축 케이블, 마이크로 웨이브) 객체를 처리하기 위해 필요한 자료 구조와 함수들을 포함한다.

LinkManager 클래스는 NodeManager 클래스와 마찬가지로 링크 데이타 화일을 관리하며, 링크 데이타 화일로부터 링크 정보들을 읽어 들여 이를 링크 객체들에 설정한 뒤, 링크 객체들을 내부적으로 연결 리스트로 구성하여 관리한다. PresentationWindow 클래스의 윈도우의 요청에 따라 이 윈도우에 나타난 지도 상의 영역에 포함되어지는 링크들을 선택하여 화면에 출력한다. 이때, 링크 양쪽끝에 연결된 노드 중에 하나라도 해당 영역안에 존재하면 표시하여 준다. 역시 마찬가지로 사용자가 새로운 링크를 생성하면 생성된 링크를 연결 리스트에 삽입하여 주고, 기존의 링크를 삭제하면 선택된 링크를 연결 리스트로부터 삭제하여 준다. 그리고, 최종적으로 연결 리스트에 있는 링크 구조를 링크 데이타 화일에 저장할 수 있도록 하여 준다.

```
class LinkManager {
    Link        *links;                 // array of links read from link file
    Link        *last_link;             // array of links read from link file
    unsigned    link_num;               // current link number in array
    int         links_modified;         // flag
    int         data_loaded;            // flag
  public :
    LinkManager ();                     // constructor
    virtual ~LinkManager ();            // virtual destructor
    Read_file();                        // read link info.
    Write_file();                       // write link info.
    void Draw();                        // draw links
    void DrawOptical();                 // draw optical link
    void DrawCoaxial();                 // draw coaxial link
    void DrawMwave();                   // draw microwave
    void ShowInformation();             // show link info.
    void CreateLink();                  // create link
    void ModifyLink();                  // modify link info.
    void DeleteLink();                  // delete link
};
```

4.2.4 MapManager 클래스

MapManager 클래스는 지도 관리자 기능을 수행하는 클래스로서 지도를 윈도우 상에 그려주기 위해 필요한 자료 구조와 멤버 함수들을 포함한다.

MapManager 클래스는 지도 데이타 화일들을 관리한다. 지도 데이타가 가지는 위도, 경도의 실제 좌표값을 윈도우 화면상의 x, y 좌표값으로 변환하는 기능을 수행하며, 사용자의 명령에 의해 특정 영역이 선택되고 확대 명령이 주어지면 선택된 지역을 확대하여 새로운 윈도우 상에 출력하여 준다.

```
class MapManager {
        int  xs, ys;                // original shift position on the original map
        int  xr, yr;                // relative position from the parent
        int  width, height;         // width and height of map
        float  scale_factor;          // scale factor
        float  rel_scale_factor;      // relative scale factor value
        Map* parent;                  // pointer to map in previous level
  public:
        MapManager();                 // constructor
        ~MapManager();                // destructor
        void CalcOriginalPosition();  // calculate original position
        void DrawMap();               // draw map
        void DrawCity();              // draw cities in map
        void DrawNet();               // draw grid line for map
        // other member functions to set and return map information
                    ...
};
```

4.2.5 Node 클래스

Node 클래스는 통신망 상에 존재하는 노드들을 모델링하기 위한 클래스로서 통신망 상의 노드들이 가지는 각종 정보와 이들 정보의 관리를 위한 멤버 함수를 포함한다. Node 클래스는 통신망 상의 노드를 보다 자세히 모델링 하기 위한 클래스인 StationNode 클래스 및 RelayNode 클래스의 부모 클래스가 된다.

Node 클래스의 객체들은 NodeManager 객체가 노드 정보를 읽어들일 때 생성되어 관리되며, NodeManager 객체의 요청에 따라 노드 관련 정보를 설정하거나 리턴하여 준다.

```
class Node : public NetworkComponent {
```

```
        int     ID;             // node ID
        char    name[80];       // name of node
        float   latitude, longitude;   // real coordinate of node
        int     scr_x, scr_y;   // screen coordinate of node
        char    address[100];   // address of node
        char    phone[20];      // phone no.
    public :
        Node();                 // constructor
        ~Node();                // destructor

        // member functions to set and return node information
                    ...
};
```

4.2.6 Link 클래스

Link 클래스는 통신망 상에 존재하는 링크들을 모델링하기 위한 클래스로서 통신망 상의 링크들이 가지는 각종 정보와 이들 정보의 관리를 위한 멤버 함수를 포함한다. Link 클래스는 통신망 상의 링크를 보다 자세히 모델링 하기 위한 클래스인 CoaxialLink 클래스 및 OpticalLink 클래스, MicrowaveLink 클래스의 부모 클래스가 된다.

Link 클래스의 객체들은 LinkManager 객체가 링크 정보를 읽어들일 때 생성되어 관리되며, LinkManager 객체의 요청에 따라 링크 관련 정보를 설정하거나 리턴하여 준다.

```
class Link : public NetworkComponent {
        int     node1_ID;       // node ID of one end of link
        int     node2_ID;       // node ID of other end of link
        int     link_ID;        // link ID
        int     capacity;       // capacity of link
    public :
        Link();                 // constructor
        ~Link();                // destructor
        // member functions to set and return link information
                    ...
};
```

4.2.7 다이어로그 클래스들

다이어로그 클래스들은 통신망 관련 정보를 사용자에게 보여 주기 위해 사용되어진다.

Dialog **클래스** 이 클래스는 다이어로그 윈도우가 갖추어야 할 기본적인 자료 구조와 함수들을 정의하고 있다.

NodeInfoDialog **클래스** 이 클래스는 Dialog 클래스를 부모 클래스로 하며, 노드 관련 상세 정보를 사용자에게 보여 주기 위한 클래스이다.

LinkinfoDialog **클래스** 이 클래스 역시 Dialog 클래스를 부모 클래스로 하며, 링크 관련 상세 정보를 사용자에게 보여 주기 위한 클래스이다.

OptionDialog **클래스** 이 클래스는 시스템 사용과 관련된 여러 옵션들을 사용자에게 보여 주고, 사용자로부터의 옵션 선택을 받기 위한 다이어로그 클래스이다. 역시 Dialog 클래스를 부모 클래스로 하고 있다.

4.2.8 기타 클래스들

본 시스템은 시스템의 확장성을 위해 응용 프로그램에 있어 기본적으로 필요되어지는 기능과 자료 구조들을 하나의 클래스로 정의한 Application 클래스와, 각각의 윈도우들을 다루는데 있어 필요되어지는 기능들과 자료 구조들을 하나의 클래스로 정의한 MainWindow 클래스를 시스템의 구성 요소로서 사용하고 있다. MainWindow 클래스는 조금 더 일반적인 형태로 세분되어 BasicComponent 클래스와 UIComponent 클래스를 부모 클래스로 갖도록 구성되어 있다. 이 네 클래스는 [7]에서 제공하는 클래스 라이브러리에 포함되어 있는 것이다. 그 밖에 통신망 정의를 위한 기본 클래스로서 NetworkComponent 클래스가 있다.

BasicComponent **클래스** 이 클래스는 GUI를 구성하는 모든 요소들이 갖추어야 할 기본적인 자료 및 함수를 정의하고 있다. GUI 구성 요소들이 기본적으로 갖추어야 할 것들로는 다음과 같은 것들이 있다 :

- GUI를 구성하는 widget 들의 tree에서 루트가 되는 'base widget'
- GUI 요소들을 위한 'base widget'에의 접근 방법
- widget tree를 manage 및 unmanage 할 수 있도록 해 주는 함수

UIComponent **클래스** 이 클래스는 BasicComponent 클래스의 자식 클래스이며, widget 소멸의 처리를 지원하며 클래스를 초기화시켜 주기 위한 자원 관리자 함수를 포함한다.

Application **클래스** 이 클래스는 모든 Motif 혹은 Xt를 이용한 응용 프로그램에 반드시 필요되어지는 일반적인 초기화 루틴을 포함하고 있다. 이 클래스를 사용함으로써 프로그래머는 XtAppInitialize() 함수 및 XtMainLoop() 함수를 프로그램 중에서 호출할 필요 없이 단지 Application 클래스의 객체 하나만을 생성하여 주면 이 객체의 초기화 과정에서 이러한 함수들이 호출되어 자동적으로 Motif 프로그램의 기본 구조를 갖추게 된다. 이 클래스는 또한 생성되어지는 모든 MainWindow 클래스의 윈도우들을 등록시키고 관리하여 주는 기능도 수행한다.

MainWindow **클래스** 이 클래스는 모든 독립적인 상위 레벨의 윈도우에 공통적으로 필요되어지는 요소들을 포함하는데, 특히, Motif 응용을 위해 반드시 필요되어지는 shell widget과 XmMainWindow widget의 생성 및 초기화를 지원하여 준다. 프로그래머가 자신에 맞는 윈도우를 생성하기 위해서는 이 클래스를 부모 클래스로 하는 유도 클래스를 만들어서 작업 영역을 초기화시켜 주는 작업만을 수행하도록 하기만 하면 된다.

NetworkComponent **클래스** 이 클래스는 통신망의 구성 요소들이 갖추어야 하는 기본적인 자료 구조와 함수들을 정의하고 있으며, 모든 통신망 구성 요소 클래스의 수퍼클래스가 된다.

5 통신망 표현 GUI의 기능 및 확장성

5.1 시스템의 기능

본 시스템은 전국에 존재하는 전화국 및 중계소로 구성된 노드들과 이 노드들 사이에 존재하는 광 케이블, 동축 케이블, 마이크로 웨이브 등으로 구성된 링크들의 상태를 지도와 함께 화면에 보여 주고, 해당 노드 및 링크에 대한 정보의 검색 및 새로운 노드와 링크의 생성, 정보 내용 수정, 삭제를 가능하도록 하여 준다. 또, 지도 안에서 임의의 지역을 선택한 뒤에 이 지역을 확대해서 보다 자세하게 노드와 링크의 연결 관계를 살펴 볼 수 있도록 해 준다. 여기서, 노드와 링크의 정보는 데이타 화일에 저장되며 노드와 링크 정보에 생긴 변화를 데이타 화일에 저장할 수 있다.

그림 4: 임의 지역의 선택 및 확대

본 시스템이 수행하는 기능들에 대해 살펴 보면 다음과 같다.

5.1.1 임의 지역의 선택 및 확대

시스템이 처음 구동되면 전국 지도를 보여주는 윈도우가 생성된다. 이 윈도우에서 'Select' 메뉴를 누른 후 임의의 지역을 마우스를 이용하여 선택한 뒤, 'Zoom' 메뉴를 눌러 확대 기능을 수행시키면 그림 4와 같이 선택한 영역이 확대되어 새로운 윈도우에 나타나게 된다. 이러한 작업은 사용자가 원하는 한 계속적으로 수행될 수 있다. 즉, 확대된 윈도우로부터 다시 특정 영역을 선택, 확대하는 작업을 계속 수행할 수 있다.

그림 5: 노드 및 링크의 화면 출력

5.1.2 노드 및 링크의 화면 출력

시스템 수행중에 'Load' 메뉴를 선택하여 노드 및 링크의 데이타 화일로부터 해당 데이타를 시스템으로 적재하면 해당 윈도우에 속하는 노드 및 링크의 위치와 연결 관계가 화면에 그림 5와 같이 나타나게 된다. 링크의 연결 관계를 쉽게 구별할 수 있도록 하기 위해 동축 케이블은 빨강색, 광 케이블은 노랑색, 마이크로 웨이브는 녹색 등과 같이 각 링크는 서로 다른 색으로 표시되어진다.

5.1.3 노드 및 링크 정보 검색

시스템 수행 중에 화면에 표시된 노드 및 링크에 대한 보다 자세한 정보를 알고 싶으면 'Show Node Info.' 혹은 'Show Link Info.' 메뉴를 선택한 뒤에 마우스를 이용하여 해당 노드 혹은 링크를 선택하면 그림 6과 같이 선택된 노드 혹은 링크의 자세한 정보를 보여 주는 다이어로그 윈도

그림 6: 노드 정보의 검색

우가 나타나게 된다.

5.1.4 그래픽 에디팅

시스템 수행 중에 새로운 노드 및 링크를 윈도우를 보면서 원하는 위치에 생성하거나, 기존의 노드 혹은 링크가 가지는 정보 내용을 변경하거나, 더 이상 존재하지 않는 노드 및 링크의 정보를 윈도우 상에서 삭제할 수 있다. 이렇게 변경된 내용은 노드 관리자 및 링크 관리자가 관리하는 연결 리스트에 반영되며, 화일 저장 기능을 통해 노드 및 링크 데이타 화일에 저장된다.

노드 및 링크의 생성 노드를 생성하기 위해서는 'Create Node'메뉴를 선택한 뒤 새로 생성될 노드의 화면 상의 위치를 마우스로 선택하면 된다. 그러면 새로이 생성될 노드가 가지게 될 정보를 입력받기 위한 다이어로그가 나타나며 여기에 관련 정보 - 노드 타입, 노드명, 노드 번호, 노드의 주소 및 전화번호 등 - 를 입력하면 하나의 노드가 사용자가 지정한 위치에 새로 생성되게 된다. 여기서, 노드의 위치 정보(위도, 경도값)는 생성된 노드의 화면 좌표값으로부터 변환되어 자동적으로 입력되어진다.

링크를 생성하기 위해서는 'Create Link' 메뉴를 선택한 뒤 화면 상에 존재하는 노드들 중에서 새로이 생성될 링크를 포함할 두 노드를 선택한다. 그러면 새로이 생성될 링크가 가지게 될 정보를 입력받기 위한 다이어로그가 나타나며 여기에 관련 정보 - 링크 타입, 링크의 용량 등 - 를 입력하면 하나의 링크가 새로 생성되게 된다. 여기서, 링크의 양끝에 위치하는 두 노드의 정보는 자동적으로 입력되어진다.

노드 및 링크 정보의 수정 이미 존재하는 노드의 관련 정보를 수정하기 위해서는 'Modify Node' 메뉴를 선택한 뒤 화면 상에서 수정할 노드를 선택한다. 그러면 변경될 노드 정보를 입력받기 위한 다이어로그가 나타나며 여기에 변경된 정보를 입력하면 기존의 정보가 바뀌게 된다. 여기서, 노드의 위치 정보가 변경되었으면 변경된 위치로 노드가 이동되어 출력된다.

이미 존재하는 링크의 관련 정보를 수정하기 위해서는 'Modify Link' 메뉴를 선택한 뒤 화면 상에서 수정할 링크를 선택한다. 그러면 변경될 링크 정보를 입력받기 위한 다이어로그가 나타나며 여기에 변경된 정보를 입력하면 기존의 정보가 바뀌게 된다.

노드 및 링크의 삭제 이미 존재하는 노드를 삭제하기 위해서는 'Delete Node' 메뉴를 선택한 뒤 화면 상에서 삭제할 노드를 선택한다. 그러면 노드 삭제 여부를 확인받기 위한다이어로그가 나타나며 여기에 삭제 여부를 확인하면 노드가 삭제되게 된다.

이미 존재하는 링크를 삭제하기 위해서는 'Delete Link' 메뉴를 선택한 뒤 화면 상에서 삭제할 링크를 선택한다. 그러면 링크 삭제 여부를 확인받기 위한 다이어로그가 나타나며 여기에 삭제 여부를 확인하면 링크가 삭제되게 된다.

5.1.5 기타 기능

이 밖에도 윈도우 상에 표시된 내용을 프린터로 직접 출력하여 주는 기능과, 관련 기능별 도움말 기능, 그리고, 사용자가 임의로 선택한 두 노드 사이에 존재하는 여러 경로들을 깜빡이게 하여 보여 주는 기능 등이 있다.

5.2 시스템의 확장 용이성

본 시스템은 그림 3과 같은 클래스 구조를 기반으로 해서 새로운 기능의 추가 요구에 대해 쉽게 확장될 수 있다. 이는 객체 지향 기법이 제공하는 상속성 및 복합화의 개념에 기반하고 있는데,

그림 7: 통신망 설계 시스템의 구성(그림 5의 클래스 계층 구조로부터의 확장)

통신망 설계 기능 및 SNMP(Simple Network Management Protocol)를 이용한 통신망 관리 기능으로의 확장 예를 통해 이를 살펴 보기로 한다.

5.2.1 통신망 설계 기능으로의 확장

통신망 설계 기능을 위해서는 다음과 같은 요소가 필요되어 진다 :

- 설계 작업을 수행할 수 있는 사용자 윈도우
- 설계 정보를 입력하고 검색할 수 있는 다이어로그 윈도우
- 설계 관련 기능을 수행하는 설계 관리자

이러한 요소들은 각각 PlanningWindow 클래스, PlanningInfoDialog 클래스, PlanningManager 클래스등의 여러 클래스들로서 모델링 될 수 있고, 기존의 시스템을 통해 그림 7과 같은 형태로 확장될 수 있다.

그림 7에서 PlanningWindow 클래스는 MainWindow 클래스의 자식 클래스로서 모델링함으로써 MainWindow 클래스가 가지는 기본적인 윈도우 구조를 상속받으면서 설계 작업을 위해 필요되어지는 기능만을 정의하면 된다. PlanningInfoDialog 클래스는 Dialog 클래스의 자식 클래스

로 모델링함으로써 다이어로그의 기본 기능을 상속받으면서 설계 정보의 입력 및 검색을 위한 구조만을 정의하면 된다. 설계 기능을 수행하는 PlanningManager 클래스는 새로운 클래스로서 정의한다. 여기서, PlanningManager 클래스는 통신망 설계 작업을 위해서는 통신망 구조를 여러 가지 형태로 변경해 보고 시뮬레이션하는 작업을 필요로 하므로, 노드 관리자 및 링크 관리자를 참조(reference) 할 필요가 있게 된다. 이것은 NetworkManager 클래스를 NodeManager 클래스와 LinkManager 클래스와 복합화(composition) 관계로서 연결함으로써 모델링할 수 있다. 이러한 방법을 통해 쉽게 설계 시스템으로 기능이 확장될 수 있다.

5.2.2 통신망 관리 기능으로의 확장

최근 일반적인 통신망 관리 시스템으로서 SNMP가 많이 사용되어지고 있다. 이 SNMP 시스템을 지원하는 통신망 관리 시스템으로의 기능 확장을 위해서는 다음과 같은 요소가 필요되어 진다 :

- 관리 작업을 수행할 수 있는 사용자 윈도우
- MIB(Management Information Base) 정보를 입력하고 검색할 수 있는 다이어로그 윈도우
- 각 장치들로부터 관리 정보를 수집하고 장치들에 관리 정보를 설정하여 주는 에이전트(Agent)
- 통신망 관리 기능을 통괄하는 통신망 관리자(Network Manager)

이러한 요소들은 각각 SNMPWindow 클래스, MIBDialog 클래스, Agent 클래스, NetworkManager 클래스 등의 여러 클래스로서 모델링될 수 있고, 기존의 시스템을 통해 그림 8과 같은 형태로 확장될 수 있다.

그림 8에서 SNMPWindow 클래스는 MainWindow 클래스의 자식 클래스로서 모델링함으로써 역시 MainWindow 클래스가 가지는 기본적인 윈도우 구조를 상속받으면서 관리 작업을 위해 필요되어지는 기능만을 정의하면 된다. MIBDialog 클래스는 Dialog 클래스의 자식 클래스로 모델링함으로써 다이어로그의 기본 기능을 상속받으면서 MIB 정보의 입력 및 검색을 위한 구조만을 정의하면 된다. 통신망 관리 기능을 수행하는 NetworkManager 클래스와 정보 수집 및 설정 기능을 수행하는 Agent 클래스는 새로운 클래스로서 정의하여 이것이 다른 클래스들과 상호 작용하도록 구성하면 된다. 특히, NetworkManager 클래스는 통신망 관리를 위해 통신망 구성 요소

그림 8: 통신망 관리 시스템의 구성(그림 5의 클래스 계층 구조로부터의 확장)

인 노드 및 링크들을 관리할 필요가 있으므로 기존의 NodeManager 클래스 및 LinkManager 클래스로부터 상속받도록 모델링한다. 이러한 형태로서 시스템을 모델링 함으로써 기존의 시스템을 쉽게 통신망 관리 시스템으로 확장할 수 있다.

5.3 시스템의 동시성 제어 기능

시스템의 동시성 제어 기능은 크게 다음과 같은 두 가지의 형태로 나누어 진다 :

- 프로그램 내에서의 동시성 제어 : 본 시스템이 수행될 때 생기는 다수의 윈도우들이 동시에 갱신 작업을 수행할 때의 동시성 제어
- 프로그램들 사이의 동시성 : 다수의 사용자가 본 시스템을 수행하여 갱신 작업을 수행할 때의 동시성 제어

5.3.1 프로그램 내에서의 동시성 제어

프로그램 내에서의 동시성 제어 기능은 다음과 같은 메카니즘을 통해 수행되어진다 :

- 모든 노드 및 링크 정보의 관리는 시스템 안에 유일하게 존재하는 노드 관리자 및 링크 관리자를 통해서만 수행되어진다.

- 시스템 윈도우들은 노드 및 링크 정보의 변경을 위해 노드 및 링크 관리자와 통신한다.
- 하나의 윈도우에서 노드 및 링크 정보에 변경이 생겼을 때 이는 노드 및 링크 관리자에 의해 다른 윈도우에 알려지고, 이에 따라 다른 윈도우들도 변경된 내용을 포함하도록 갱신되어진다.

5.3.2 프로그램들 사이에서의 동시성 제어

프로그램들 사이의 동시성 제어는 현재 시스템이 노드 및 링크 정보를 데이타 화일 형태로서 관리하고 있기 때문에 현재로서는 구현이 어렵다. 하지만, 이는 향후 연구 과제로서 삼고 있는 본 시스템과 데이타베이스와의 연결 작업을 통해 쉽게 구현될 수 있을 것으로 본다. 즉, 노드 및 링크 정보의 검색 및 변경을 데이타베이스를 통해 작업하면 데이타베이스가 제공하는 동시성 제어 기능을 이용할 수 있으므로 다중의 사용자가 동시에 본 프로그램을 이용하여 노드 및 링크 정보를 변경할 때 동시성 제어를 수행할 수 있게 된다.

6 시스템 구현 상태 및 향후 연구 계획

6.1 시스템의 구현 상태

본 시스템을 구현하기 위해 사용한 구현 환경은 다음과 같다 :

- 기반 환경 : Unix & X 윈도우 시스템 [9,15,18]
- 그래픽 사용자 인터페이스 라이브러리 : Motif widget을 사용 [12,16,17]
- 사용 프로그래밍 언어 : GNU C++ [4,8,11]

시스템은 현재 약 15,000 라인 정도의 C++ 코드로 구성되어 있으며, 앞서 명시한 시스템의 기능들은 모두 아무 이상없이 수행되고 있다.

6.2 시스템의 구현을 통한 경험

본 시스템을 구현하면서 객체 지향 기법이 통신망의 구조 표현 및 GUI를 사용하는 시스템에 아주 잘 적용될 수 있음을 살펴 볼 수 있었다. 클래스 개념을 통한 통신망의 모델링은 그 구현이 간

단하고 쉬웠으며, 재사용성 및 확장성 등을 고려할 때 적절한 선택이었음을 알 수 있었다. 차후의 연구 과제로 삼고 있는 데이타 베이스와의 접목 및 질의 처리 역시 기존의 클래스에 데이타 베이스 처리를 위한 클래스를 정의하고 이들로부터 상속을 받는 구조로 전환하면 되므로 쉽게 이루어질 수 있으리라고 생각한다.

6.3 향후 연구 계획

향후 연구 계획으로는 다음과 같은 것들이 있다 :

- 노드 및 링크 관련 정보의 DB화
- 데이타 베이스와의 인터페이스 클래스 구현
  ⇒ 사용자로부터의 질의를 데이타 베이스로 전달하고, 해당 결과를 리턴하여 주는 인터페이스 클래스의 구현
- 사용자로부터의 질의 처리 및 질의 결과의 화면 출력 구현
  ⇒ 예를 들어, "수원에서 대전까지의 모든 광케이블의 표시"라고 하는 질의를 받아들이고, 해당 결과를 화면에 출력(깜빡여서 보여 줌)

7 결론

본 논문에서는 재사용성 및 확장 용이성, 유지/보수의 용이성 등의 특징을 가지는 객체 지향 기법을 이용하여 전국적인 통신망의 상태를 모델링하고 표현하여 주는 확장이 용이한 통신망 표현 GUI 시스템의 설계 및 구현 기법에 대해서 살펴 보았다.

이 시스템은 통신망을 구성하는 요소들인 노드 클래스와 링크 클래스, 지도 관련 기능을 수행하는 지도 클래스, 이러한 정보들을 종합적으로 화면에 출력하여 주고 사용자로부터의 명령을 입력받아 전체 통신망을 알기 쉽게 표현해 주는 GUI 윈도우 클래스, 그리고, 관련 정보들을 사용자에게 보여 주는 기능을 수행하는 GUI 요소들을 모델링하였으며, 이들 클래스들의 상호 연결 관계를 보였다. 또한, 기존의 시스템에 새로운 기능의 추가 요구시 시스템이 쉽게 확장될 수 있음을 통신망 설계 기능 및 통신망 관리 기능으로의 확장을 위한 모델링 예를 통해 보였다.

현재 객체 지향 기법 및 GUI를 사용한 시스템이 많은 사람들의 관심을 끌고 있고, 특히, C++ 언어 및 X & Motif 환경을 기반으로 한 시스템이 많이 작성될 것으로 예상된다. 본 시스템은 앞

으로 작성될 통신망 관련 GUI의 설계와 구현에 대한 하나의 참조 모델로서 사용되어질 수 있을 것이다.

감사의 글

본 연구를 진행하는데 많은 도움을 준 한국 통신의 조 성원 연구원님께 깊은 감사를 드린다. 그리고, 이 논문을 심사해 준 익명의 심사 위원님들께도 감사드린다.

참고 문헌

[1] Ben Shneiderman, "Designing the User Interface",2nd Ed., *Addison Wesley,* 1992

[2] Bertrand Meyer, "Object-Oriented Software Construction", *Prentice-Hall,* 1988

[3] Brad J. Cox, "Object-Oriented Programming", *Addison-Wesley,* 1991

[4] Bjarne Stroustrup, "The C++ Programming Language", 2nd Ed., *Addison Wesley,* 1992

[5] David R. Barstow, Howard E. Shrobe, Erik Sandewall, "Interactive Programming Environments", *McGraw-Hill,* 1984

[6] Deborah J. Mayhew, "Principles and Guidelines in Software User Interface Design", *Prentice Hall,* 1991

[7] Douglas A. Young, "Obejct-Oriented Programming with C++ and OSF/Motif", *Prentice Hall,* 1992

[8] James O, Coplien, "Advanced C++ Programming Styles and Idioms", *Addison-Wesley,* 1992

[9] Jonhson & Reichard, "Advanced X Window Application Programming", *MIS Press,* 1990

[10] Jun Matsuda, "Technical Trends in Network Engineering & Management", *NTT R&D,* Vol.42 No.2 1993

[11] Margaret A. Ellis, Bjarne Stroustrup, "The Annotated C++ Reference Manual", *Addison-Wesley,* 1990

[12] Marshall Brain, "Motif Programming", *Digital Press,* 1992

[13] Maureen Crozier,Steve Reld, "Flexible Netwok and Product Representations for Planning Tools", *GLOBECOM'87*

[14] Nan C. Shu, "Visual Programming", *Van Nostrand Reinhold Company,* 1988

[15] Oliver Jones, "Introduction to the X Window System", *Prentice-Hall,* 1989

[16] Open Software Foundation, "OSF/Motif Programmer's Guide", *Prentice Hall,* 1993

[17] Open Software Foundation, "OSF/Motif Programmer's Reference", *Prentice Hall,* 1993

[18] Paul J. Asente, Ralph R. Swick, "X Window System Toolkit", *Digital Press,* 1990

[19] Thomas W. Kennedy, "Network Operations & Management Tool Requirements for the 90's", *NOMS'92*

[20] NOMS'94 Tutorial 6: Network Management Platforms