ネットワーク管理システムへのオブジェクト指向の適用*

川崎製鉄技報
26 (1994) 3, 119-123

Application of Object Oriented Methodology to Network Management System

永山 尚博
Naohiro Nagayama
川鉄情報システム(株)
基盤システム事業部
ネットワークシステム部
主任部員(主席課長)

下地 健一
Kenichi Shimoji
川鉄情報システム(株)
基盤システム事業部
ネットワークシステム部

要旨

LANから広域網までのネットワークを統合的に管理するネットワーク管理システムを開発した。ネットワーク構成の変更や成長，ユーザニーズに柔軟に応えられるシステムを実現するため，設計から実装まで開発の全工程においてオブジェクト指向の開発手法を適用した。本システムは，モジュール型でUNIX上で稼働するプラットフォームと構成，障害，性能，および機密管理機能を持つ標準アプリケーションより構成されている。本システムではネットワーク管理と親和性の高いオブジェクト指向を適用することにより，ネットワーク管理システムのモデル化を実現し，システム拡張時の生産性向上も実現できた。

Synopsis:

The authors have developed an integrated network management system, which can manage networks including LAN and WAN. In its development process from designing to implementation, an object oriented methodology has been applied, in order to realize a flexible system, which can cope with configuration changes and growth of network and to keep up with users' needs. The system is a modular type system composed of a platform working on UNIX and standard applications which contain management functions of configuration, faults, performance, and secrecy. As a result of appling the object oriented methodology, which is familiar with netwok management, modeling of the network management system has been realized, and productivity improvement during system expansion achieved.

1 はじめに

情報ネットワークを取り巻く環境の変化はダウンサイジングの浸透に伴ってさらに拍車がかかり，ネットワークの大規模化と複雑化，構成機器の多種多様化，すなわちマルチメディア，マルチベンダ，マルチネットワーク化が急激に進んできた。このため，ネットワークを管理する上で以下の問題点が出てきた[1]。

(1) 障害区分／解析の困難性
(2) 保守運用コストの増大
(3) 資産管理，計画管理の困難性
(4) ネットワーク運用管理者の不足

そこで，これらの問題点を解決できる経済的で信頼性，柔軟性，操作性，保守性，拡張性に優れたネットワーク管理システムの必要性が叫ばれてきた。

こうした状況下から，ネットワーク・インテグレーション事業の核となる商品としてネットワーク管理システムの開発に取り組むことにし，経済的で信頼性，柔軟性，操作性，保守性，拡張性に優れたシステムを実現するため，開発手法としてオブジェクト指向を適用した。

本報では，ネットワーク管理システム開発を通して実施したオブジェクト指向の適用について報告する。

2 ネットワーク管理システムの要件

市販製品に対し十分な競合力を持ち，差別化を可能とするため，開発するネットワーク管理システムに対する目標を以下のように定めた。

(1) 市販の管理アプリケーションに相当する機能が容易に追加可能であること。
(2) 新規の管理アプリケーションが容易に追加可能であること。
(3) ユーザのネットワーク構成の変化に柔軟に対応できること。

これらの要求を満たすためのシステムの条件を以下に示す。

(1) システムが部品化（モジュール化）され，しかも一つ一つの部品の独立性が保たれ，新規追加に対し他の部分への影響が最小限で済むような構成となっていること。
(2) 新規追加の際に，全く新規にすべてを作成するのではなく，既存の部品を有効に再利用できること。
(3) 新規追加部分についても統一的な操作が行えること。

これらの条件を実現するための開発手法として，オブジェクト指向が有効であるので，ネットワーク管理システム開発に適用することとした。

* 平成6年5月11日原稿受付

3 ネットワーク管理システムの種類

ネットワーク管理システムとしてすでに存在しているSNMP (simple network management protocol) マネージャ製品は，その実装形態から4種類のタイプに分かれる[2]。

(1) プラットフォーム型
(2) パッケージ型
(3) モジュール型
(4) アプリケーション

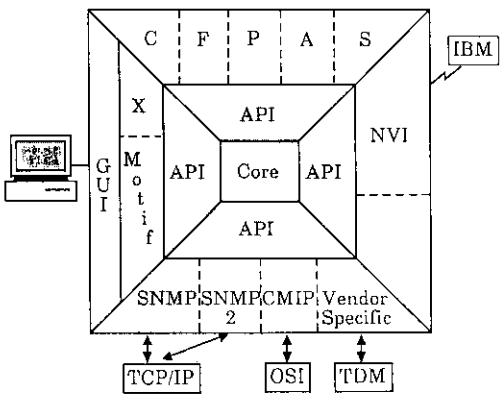
第1番目のプラットフォーム型とは，管理のための基本部分だけでできており，アプリケーションは別途購入または開発が必要である。

第2番目のパッケージ型はプラットフォーム型に簡単なアプリケーションを付加したものである。

第3番目のモジュール型は，パッケージ型に比べ標準機能としてのアプリケーションの他に種々のアプリケーションが用意されていて，これらをオプションとして組み合わせて管理システムを構築する。

第4番目のアプリケーションとは，プラットフォーム型製品上で動作するアプリケーション部分のみで構成される製品で，動作させるためには別途プラットフォーム製品の購入が必要である。

本システムは種類としてはモジュール型製品であるが，特長はプラットフォームとしてのアプリケーションからの可用性に優れているところにある。これはシステム開発の方針としてカスタマイズを強く意識し，大規模なLAN (local area network) やWAN (wide area network) のネットワーク管理への適用をターゲットとしているためである。このため，**Fig. 1** に示す4方向のインタフェースを有している。



C: Configuration　　S: Security
F: Fault　　X: X-Window
P: Performance　　NVI: NetView Interface
A: Account

Fig. 1 Openway NM Interfaces

4 システムの構成

本システムは前節で述べたように，モジュール型の実装形態となっており，**Fig. 2** に示すように以下の二つの部分から構成され，さらに拡張が可能になっている。

(1) プラットフォーム部分
ネットワーク管理のための基本機能部分
(2) アプリケーション部分
プラトフォーム上で動作する具体的な個別の管理機能部分

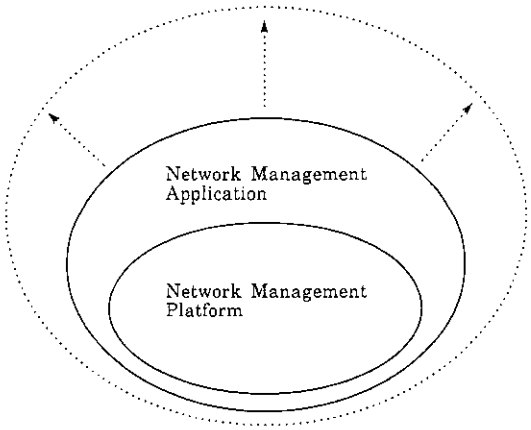


Fig. 2 Structure of system

4.1 プラットフォーム

システムを開発するにあたり，プラットフォームに要求される条件は，次の4点である。

(1) 全面的にオブジェクト指向で設計・実装されたシステムである。
(2) UNIXワークステーション上で動作する。
(3) ネットワーク管理システム構築用の基本機能を有している。
(4) システム拡張時に再利用できる部品（サブモジュール）を多く持っている。

本システムでは以下に示すように，これらの条件をすべて満足している。

(1) 全面的にオブジェクト指向を適用し，実装のプログラミング言語もオブジェクト指向言語であるC++を使用している。
(2) SUNワークステーション上で開発し動作する。
(3) 管理アプリケーションで使用する基本機能のインタフェースがライブラリ等で用意されている。
(4) 数百におよぶクラスが用意されており，サブクラスを定義することにより部品（サブモジュール）の再利用ができる。

4.2 アプリケーション

プラットフォーム上で管理機能を果たす標準的なアプリケーションとして次の四つの管理機能を有している。

(1) 構成管理機能
マップ表示，構成情報のデータベース化，ノードのグループ化
(2) 障害管理機能
アラーム設定／表示，トラブルチケット発行
(3) 性能管理機能
モニタ（リアルタイム情報），ログ（履歴情報）
(4) 機密管理機能
管理システムへのアクセス制御

5 ネットワーク管理とオブジェクト指向の親和性

今日のようにネットワーク構築が盛んに行われるようになると，

ユーザがネットワーク管理を考えざるを得ないのは当然であるが，ここでさらに問題となるのは構築したネットワークは今後も拡張・発展するということである。つまり，拡張・発展したネットワークを管理するため，管理システムも拡張・発展させなければならない。この時，管理対象は必ずしも従来から存在するものばかりとは限らず，全く新しいプロトコルや機器であることも多いと予想される。例えば，現在ではネットワーク管理を階層化して考える場合，まずLANとかWANとかいったドメインに分けているが，**Fig. 3** に示すように，仮にこれに新しいドメインが追加されたと仮定すると，従来の手続き型言語のシステムでは全体に及ぼす影響が大きい。システムが構造化されていたとしても，既存システムの修正やリコンパイルがあり，システム全体の内，数割は修正対象となる。これに対し，オブジェクト指向では，この追加されたドメインを一つのオブジェクトとして独立して取り扱うことができ，既存システムの影響は最小に抑えられ，生産性向上に貢献することができる。

また，ネットワーク管理の標準化においても，ネットワーク管理の持つ拡張性や階層性と言った性質を考慮し，オブジェクト指向が採用されている。したがって，今後ネットワーク管理システムの開発において，オブジェクト指向を適用することはごく自然であると言える。

さらに，将来的にはMIB (management information base) を実装するデータベースもOODB (object oriented data base) を使用すればシステムとの親和性も良く，今後拡張・発展するネットワーク管理に容易に対応できる。言い換えると，ネットワーク管理をオブジェクト指向で設計・実装することは，実世界のネットワークとそれに対する管理をシステム内に写像化する上で非常に適しているということである。

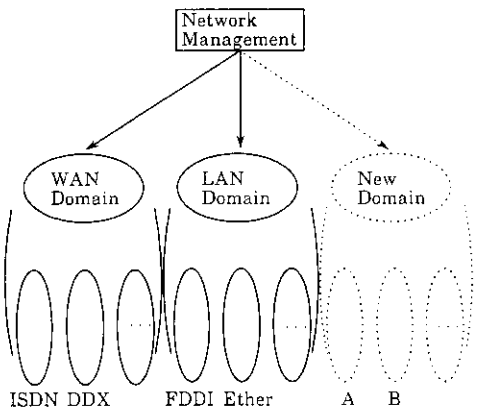


Fig. 3　Managed domain

6　ネットワーク管理におけるオブジェクト指向の適用

6.1　モデル化

オブジェクト指向設計で特に重要な作業がモデル化である。モデル化とは，実世界の現象を抽象化し，モデルとして表現することにより，システム内に写像化することである。したがって，このモデル化の作業はオブジェクト指向での開発プロセスの中で最も重要であると言っても過言ではない。ネットワーク管理システムに適用する場合も，もちろんモデル化は重要で，実現しようとする仕様をモデル化することに多くの時間を要する。

本システムでは，三つの異なる視点からシステムをモデル化した。

第1はオブジェクトモデルで，システムの静的，構造的，データ的な側面を表現しており，クラス階層とクラス定義を記述する。

第2は動的モデルで，システムの時間的，動作的，制御的な側面を表現しており，本開発ではナビゲーション・フローとシナリオで記述した。

第3は機能モデルで，システムの変換的，関数的，機能的な側面を表現しており，マクロ定義や関数定義にあたる[3]。

6.2　オブジェクト

ネットワーク管理システムにおいては，管理の対象となるものはすべてオブジェクトとして表現される。すなわち，ネットワークを構成するすべての機器（ノード），ノードの持つ管理情報としてのMIB，ノード間の接続を表すリンク，ノードやリンクの集合としてのクラスタといったネットワークの構成要素の他，障害管理におけるアラームやトラブルチケット，性能管理におけるポーリングやグラフもすべてオブジェクトとして表現される。それぞれのオブジェクトの定義はクラスと呼ばれ，システム内に存在するオブジェクトの実体はインスタンスとよばれる。

したがって，ネットワーク管理システムでオブジェクトとして扱われるものすべてについてクラス定義が必要であり，その定義ですでに用意されているクラスをどのように継承するかが設計上重要な点となる。

6.3　クラス定義

クラス定義では，必ず属性 (Attribute) と操作 (Method) が定義され，これによりシステムにおけるそのオブジェクトが意味を持ったものになる。ネットワーク管理システムにおけるクラス定義の例として，例えばMibNodeクラス定義は **Fig. 4** のようになる。

```
abstract class MibNode : PhysicalNode, Mib2NetElement,
                         InSmiPSPollNetElement {
    // Attributes
    rc  int                 macAddress;
    rw oidStringList        traps;
    //  C++ Actions
    //
    //  public:
    //      virtual uBoolean      Create(...);
    //      virtual uBoolean      Delete(...);
    //      virtual uBoolean      HandleSnmpTrap(...);
};
```

Fig. 4　MibNode class definition

6.4　性質継承とクラス階層

6.2節で述べたように新しいクラス定義を行う場合には，ほとんどの場合全く新しいクラスを定義するのではなく，既存のクラスを継承したサブクラスを定義することになる。したがって，新しいクラスがすでにシステム内に存在するクラスで構成されるクラス階層のどこに組み込まれるのかが重要である。ネットワーク管理システムにおけるクラス階層は特に重要で，クラス階層中に抽象化されたジェネリックなクラスが用意されていれば，新規のオブジェクト追加も比較的容易にできる。クラス階層の例としては，ネットワーク上のノードのクラス階層を **Fig. 5** に示す。

クラス階層による性質継承をFig. 5内のWorkstation Classの属性を例にして以下に示す。

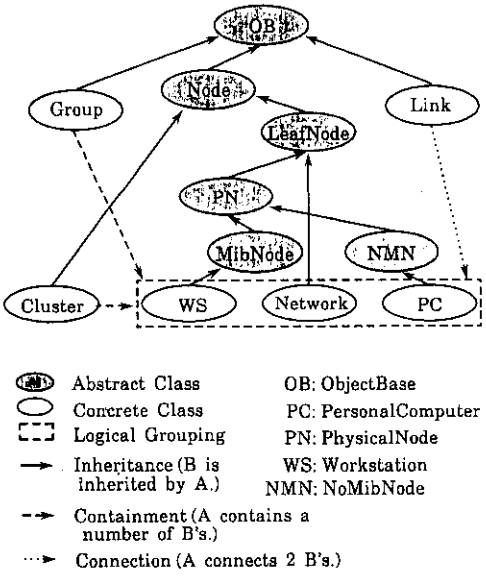


Fig. 5 Relationship among the objects

(1) 最上位に位置する ObjectBase Class は，すべてのオブジェクトのベースとなるクラスで，どのオブジェクトクラスもこのクラスを継承する。定義される属性は name のみである。

(2) Node Class で定義される属性は NodeType のみである。

(3) LeafNode Class で定義される属性は，
autoCreated
subType
groups
location
vendorName
versionNumber
である。

(4) PhysicalNode Class で定義される属性は ipAddress のみである。

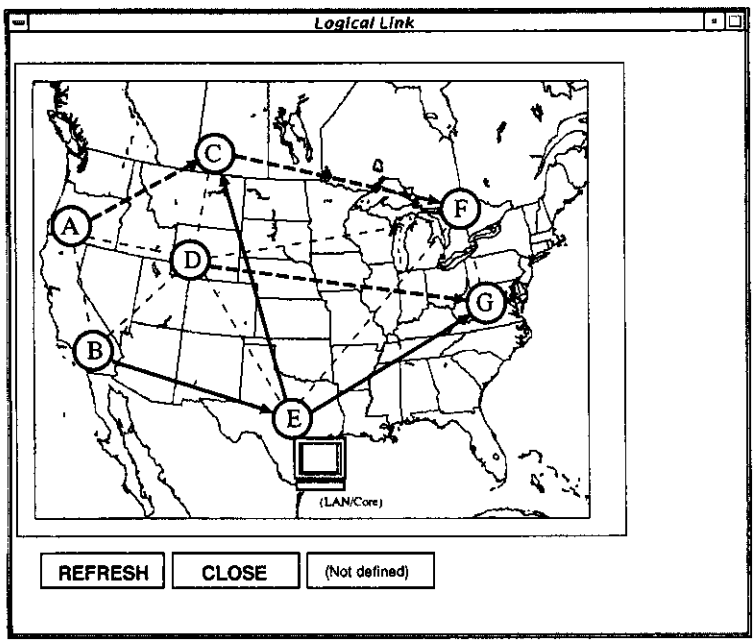
(5) MibNode Class で定義される属性は macAddress,traps である。

(6) Workstation Class は MibNode Class のサブクラスとして定義され，上位クラスの性質をすべて継承するので，特に定義をしなくても(1)～(5)で定義された属性をすべて持ったクラスとして定義される。

したがって，新規オブジェクトクラスの追加の際にも十分な分析を行い，既に存在するジェネリックなクラスのサブクラスとして定義するか，あるいは新しいジェネリックオブジェクトクラスを定義した上でサブクラスを定義するかして，上位クラスの継承を活用すべきである。

6.5 動的に変化する接続状態の GUI 化

ここで，システム拡張時でのオブジェクト指向の有効性を，本システム上で実施された拡張を例にして考察する。テーマは「動的に変化する接続状態の GUI (graphical user interface) 化」で，これは標準管理機能にある静的なノード間の接続情報とは異なり，動的に変化するノード間の接続状態を捉え，画面上にグラフィカルに表示することを課題としている。実際には **Fig. 6** に示すように，ネットワーク上に存在する特定のノードと，現在接続中の状態にあるのはどのノードなのかを収集した情報をもとにネットワーク管理マップ上に表示することである。



A—G : Cluster including IWU

Active link from/to local IWU

Active link from/to neighbor IWU

Possible link

Fig. 6 Logical link view

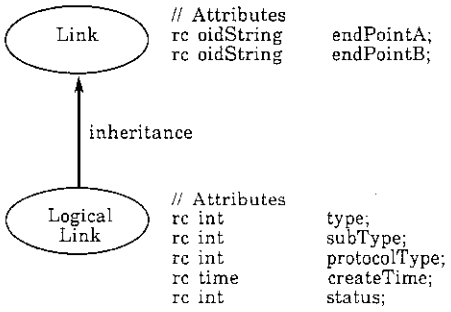


Fig. 7 Attributes of Logical Link Class

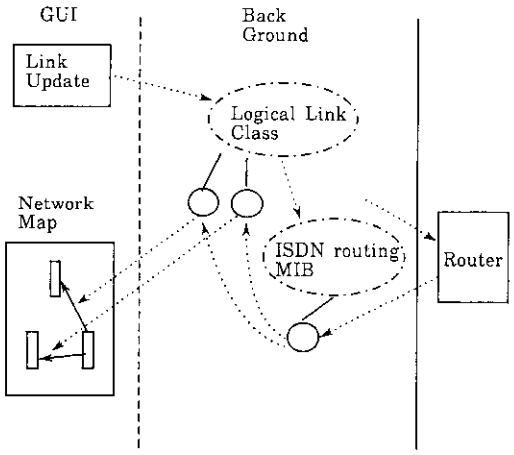


Fig. 8 Link update operation

まず，オブジェクトモデルを作成し，動的に変化する接続情報をLogical Link Classとして定義する。このオブジェクトは全くの新規オブジェクトではなく，既に存在するLink Classのサブクラスとして定義する。これにより**Fig. 7**に示すように接続オブジェクトとしての共通の性質であるリンクの両端（接続元と接続先）という属性が継承される。そのほかに，リンクの種別を表すtypeとsubtype，使用されるプロトコルを表すprotocolType，動的な変化を捉えるためのcreateTime，接続状態を示すstatusを属性として追加する。

また，操作としては生成，削除を定義する。特に生成時にGUI上にノード間をつなぐ線分として表示し，Statusによって線分の表示色が変化するよう，既存のメソッドを利用して定義する。これにより，定義した属性が入力されれば接続状態が表示可能となる。

次に，動的モデルとして，オブジェクトの生成から削除までの遷移を考える。オブジェクト生成の方法としては，定期的なポーリングにより情報を収集し自動的に生成する方法と，マウスでボタン等をクリックすることによりオブジェクトを生成する方法があるが，今回は，**Fig. 8**に示すようにLinkUpdateボタンのクリックにより，特定の管理ノードで収集できる接続情報（拡張MIB）を収集し，これをオブジェクトとして生成する方法を採用した。削除については，次回生成時自動的に行われる。

最後に，機能モデルについては，既存の関数を使用するため特に新しい関数は必要ではない。

このように，新しい機能の追加に関してもオブジェクト指向でのアプローチは有効である。特に，従来の開発手法に比べると実際に実現したい内容とシステム設計時に行うモデル化の内容が近く，従来のシステム開発では専門家を必要としたシステム化が容易である。また，オブジェクトクラスの継承により再利用性も高い。

7 結　言

ネットワークインテグレーション事業の強力なツールとなり，今後ますますその拡張性が要求されるであろうネットワーク管理システムを開発するために，開発手法としてオブジェクト指向を適用した結果，以下の成果が得られた。

（1） オブジェクト指向適用に不可欠なモデル化を，ネットワーク管理システムで実現した。

（2） オブジェクト指向の適用により，拡張時の効率化が実現できた。

今後，さらにアプリケーションを充実させ，顧客ニーズに柔軟に対応するために，オブジェクト指向適用の範囲を広げ，データベースにおいてもオブジェクト指向データベースを適用し，より高度なサービスを提供するネットワーク管理システムを開発することが課題である。

参考文献

1) 清水　豊：「ネットワーク管理の最新動向と技術展望」，(1990)，[(株)ソフト・リサーチ・センター]

2) 林　哲史：「SNMPマネージャ」，(1993)，[日経BP]

3) J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen: "Object-Oriented Modeling and Design", (1992), [Prentice Hall]