監視制御システム向けアプリケーションフレームワーク

非会員 小 島 泰 三 (三菱電機(株)産業システム研究所)
正 員 中 田 秀 男 (三菱電機(株)産業システム研究所)
非会員 丹 下 正 純 (三菱電機(株)系統変電・交通システム事業所)
正 員 西 田 正 吾 (大阪大学基礎工学部)

GRASS: An Object-Oriented Framework for SCADA Systems

Taizo Kojima, Non-member, Hideo Nakata, Member, Masazumi Tange, Non-member (Mitsubishi Electric Corp.), Shogo Nishida, Member (Osaka University)

This paper describes GRASS, an object-oriented framework for SCADA(Supervisory Control And Data Acquisition) systems. The proposed framework consists of class libraries and tools, that support development of SCADA system software. The basic functional requirements of SCADA systems are similar, regardless of the particular application field, so substantial software sharing is possible. Nevertheless, productivity of SCADA software is impeded by dependence on details of the target system. The task of configuring the SCADA software for each target system is complex, typically involving manual input of a lot of system description data. GRASS simplifies this task by automatically generating system description data and providing editing tools for subsequent customization.

キーワード：監視制御システム，電力系統，自動生成，フレームワーク，カスタマイズ

1. はじめに

工業プラントや電力系統等の大規模システム向けの監視制御システムの構築では，システム機能の高度化や大規模化に伴い，そのソフトウェア開発の生産性低下が問題となっている。これまで，監視制御システムのソフトウェア開発の生産性向上を目的とし，共通パッケージの利用等，様々なソフトウェア共用化の試みがなされている。監視制御システムの基本機能は，監視制御対象の所属する分野に大きく依存している訳ではない。対象設備の現在状態を CRT 画面上に表示する監視機能，アナログ値の設定，バルブや開閉器等の動作を制御する操作機能，時系列データのトレンド表示や帳票表示等，基本的な機能は各分野において共通している。まして同一分野における個々のシステムでは，機能の多少はあっても，個々の機能の内容自体は変わらない。このため，各機能の実現においては，ルーチンの共用化が可能である。残念ながら，これら従来の取り組みだけでは，監視制御システムのソフトウェア生産性向上において，十分な効果を達成できていない。

監視制御システムソフトウェアの生産性向上を阻む大きな要因は，機能の違いではなく，その対象依存性にある。監視制御対象であるプラントや系統，センサやアクチュエータ

等の監視機器，監視機器から監視制御システムの間のデータ伝送系等が対象システムにより異なり，システムをそれらに合わせる必要がある。各機能を司るソフトウェアの対象依存部分が，システム定義データの形でソフトウェア本体と分離されていたとしても，現状ではシステム定義データの入力作業量が大きい。また，実際にはシステム定義データの変更だけで対処できる場合が少なく，対象に合わせたソフトウェア本体の改変が必要となる。さらに，同一機能であっても，ソフトウェア開発の担当者が対象システム毎に異なる場合がある。その際，改変のために既存のソフトウェアを理解することを放棄し，新規開発に傾きやすい。結果として，機能が変わらないのに，次々と新しいソフトウェアが再開発されていくことになる。従って，監視制御システムのソフトウェア生産性を向上させるには，ソフトウェアの共用化だけでなく，対象依存部分の扱いについても十分に考慮する必要がある。

従来から，様々な監視制御システムソフトウェアが開発されている。例えば，FIX DMACS⁽²⁾や看太郎⁽¹⁶⁾は，小規模な監視制御向け流通ソフトウェアである。これらソフトウェアは，監視制御機能やデータ作成ツール等，システム構築に必要な一連の機能を提供している。また，S. P. I. D. E. R.⁽¹⁾と呼ぶ，電力系統監視制御システムのためのソフトウェア

アーキテクチャも開発されている。

残念ながら、従来のシステムは、監視制御システムにおけるシステム依存の定義データの作成の効率化について十分な考慮をしていない。確かに流通ソフトウェアを用いることにより、監視制御システムのプログラム部分の作成は不要となる。流通ソフトウェアでは汎用性を高くするために、アナログデータやデジタルデータをタグと呼ぶ表現で統一し、画面へのデータ割り付けや制御演算等の全ての処理をタグを用いて記述する。タグはプリミティブレベルの表現であり、何万点もの入出力信号を扱う大規模システムでは、このタグの管理に多くの手間を要す。また、特定分野向けのシステムでは、流通ソフトウェアと比較し、分野に応じた入力支援を導入することができるが、伝送系の違いにより、顧客毎に特殊化しなければ、十分な入力支援を行えない場合が多い。このため、従来の監視制御システムでは、特定分野向きといえども、完全なカスタムソフトウェアとして開発する場合以外は、定義データの作成支援については十分な配慮をしていなかった。

オブジェクト指向監視制御フレームワーク GRASS (Generation based Reconfigurable Architecture for SCADA Systems) は、監視制御システム開発における対象依存性の問題を軽減し、生産性向上を達成すべく研究開発しているソフトウェアシステムである。筆者ら自身、既に監視制御システム向けデータベースや監視画面定義の作成に関する研究開発を行っている⁽¹⁰⁾⁽¹³⁾。これら研究は、既存の監視制御システムのデータ作成を支援するものである。これに対し、GRASSの研究では、監視制御システム全体について、その構築方法について見直しを行った。

本研究の最大の主眼点は、設備定義を元とした諸定義データの自動生成とカスタマイズにある。回路図やシステム構成図等の設備図面を用いて、設備定義を入力する。この得られた設備定義から、まず、デフォルトの監視画面定義や入出力データ項目定義を自動生成する。次に、生成された監視画面定義や入出力項目定義を、編集ツールを用いて最終的な定義に編集する。そして、得られた入出力項目定義を元に、入出力信号の割り付け表や、集計定義、さらには入力信号の変化に対してアラームを発する等の監視処理定義を自動生成する。監視制御システムを対象システム毎に適合させるための作業量を最小化するとともに、自動生成により誤りの混入をも最小化する。このためのソフトウェアアーキテクチャが GRASS である。

以下では、監視制御システムのソフトウェア構築のためのフレームワークについて論じる。まず、2 において、本フレームワークの基本設計について検討する。次に 3 において、開発した監視制御向けフレームワーク GRASS について、その構成及び概要を説明する。また、4 では、適用及び評価について述べる。

2. 設 計

応用プログラム生成系 (Application Generator) は、対象とする応用問題に関する仕様から、応用プログラムの全部あるいは一部を自動生成するシステムである^{(3)(4)(6)(7)(10)~(14)}。例えば、Hyproof⁽¹⁴⁾ では、オブジェクト指向方式とソフトウェア自動生成を組み合わせることにより、監視制御向けマンマシンシステムの構築を支援する。また筆者ら自身も、監視制御システム向けデータベース管理システムの生成⁽¹⁰⁾や監視画面定義の生成⁽¹³⁾において、その有効性を確認している。GRASS の設計では、この応用プログラム生成系の技術に基づき、監視制御システムの対象依存のプログラムや定義データを効果的に作成する手法について検討した。以下では、GRASS 設計上の要点について述べる。

〈2・1〉 オブジェクト指向方式の応用プログラム生成系 GRASS では、オブジェクト指向方式に基づき、各種応用プログラム生成系がアプリケーションフレームワークの中に体系化される。それら生成系は、記述言語方式と視覚的な方式の2種類の入力から、スクリプト言語による記述やデータ定義ファイル等を生成する。全ての応用プログラム生成系において、その扱う対象をオブジェクトとして表現する。オブジェクトとして扱うことで、情報隠蔽が促進されることを期待するだけでなく、オブジェクトとして、概念上、統一することにより、生成系を含む、複数のプログラム間において、共通のインタフェースを確保しやすくなると考えるからである。

オブジェクト指向方式の導入は、以下のように行う。まず、監視制御対象の設備をクラスによりモデル化する。ここでは、筆者らによる従来の研究⁽¹⁰⁾と同様、監視制御対象の設備や制御用機器のモデル化において、オブジェクト指向分析設計手法⁽¹⁵⁾を用いる。本フレームワークでは、このモデル化によって得られるクラスを、オブジェクト指向プログラミング言語で提供されるクラスと区別し、論理クラスと呼ぶ。一般にデータ設定には、同様のパタンが繰り返し現れる。この共通パタンを論理クラスに対して集中して定義する。

このモデル化において、個々の設備を設備オブジェクトとして表現、対応する各クラスの属性として、設備に付随するセンサー等の入出力接点情報を定義する。また、クラスで属性間の関係も記述できるようにする。通常、一つの設備には、相互に関係付けられた様々な入出力接点が付随する。例えば、遮断器の開閉や遠方制御権の有無を示す入力信号、制御操作を行う出力信号等がある。そして、これらの制御関係を定義する。設備を表すオブジェクトが与えられれば、クラスで定義された情報を用い、必要な入出力項目定義や設備を個別に制御するための命令シーケンスを自動生成することができる。

上記は、同一インスタンスに閉じる関係の集約定義といえる。一方、複数のオブジェクト間の関係においても、類

図 1 複合設備パターン

Fig. 1. Template of composite equipment

似た関係が存在する。例えば、電力系統の設備の場合、図 1 のように、用途に応じて、典型的な電力回路の接続形態というものが存在し、これらをパターン化することができる。また、図中、図形要素として現れていないが、リレー装置やセンサー等の監視用機器等も電力回路に付随しており、これらの設定もボタン化できる。このようなボタン化できる設備の集合は、個別に定義するより、ボタンを部品化し、この部品を用いて電力回路を定義するほうがデータの入力効率は向上する。

〈2・2〉属性への修飾情報の付加 様々な種類のプログラムを組み合わせで統合化システムを構築するためには、その基本部を構成、そして制御するための枠組が必要である。本フレームワークでは、対象を論理クラスとしてモデル化する際に、一つの論理クラス定義の中に、複数の応用ソフトウェアに関する情報を集約的に定義する。この集約化された情報に基づき、フレームワーク全体を統一的に制御する。

例えば図 2 のような監視画面の表示を考える。画面上には、設備の情報として、計測値やシミュレーションにより求まる値、名称等が混在して表示される。また、計測値とその前日の実績値が組み合わせて表示される。従って、画面表示を行う場合には、その内容に応じて、計測値を保持するサーバやシミュレータ等に問い合わせる必要がある。そこで、論理クラスの属性定義において、関係する計測値や名称を記述し、各属性に対し、その属性値の取得に係わるサブシステムを指定する。また、画面定義の各図記号に対し、対応する設備と、その設備のどの属性であるかを示す識別子を与える。画面表示時にはデータ問い合わせ要求を属性に付加されたサブシステムに振り分けるようにすれば、上記処理に対して一般化した解法を提供できる。

なお、属性定義における修飾情報は、前述の属性間の関係記述の手段として利用できる。例えば、計測値と上下限値の双方を属性として定義し、それらの間に計測値とその閾値であるという関係を設定する。そして計測値を表す属性に対し、入力接点の有無や工学値変換のための計算式を定義する。この修飾情報を用いれば、監視処理定義の生成や入出力データ項目の抽出処理を実現できる。

〈2・3〉生成・カスタマイズ 一般にソフトウェアの完全な自動生成は困難である。従来のソフトウェア自動生成に関する研究は、完全なソフトウェアの生成を目指すものであった。この場合、ソフトウェアの完全な仕様書が必要

図 2 属性修飾を用いた問い合わせ要求の処理

Fig. 2. Dispatch with attribute qualifiers

である。ところが、計算機で処理可能な完全な仕様書を用意することは、プログラミング言語を用いて必要なソフトウェアを実装するよりも、通常、困難である。このため、ソフトウェア自動生成は、部分的には実用化されていても、大規模ソフトウェアの作成では、実用化が困難であった。

グラフィカルユーザインタフェース (GUI) 分野において、ソフトウェア自動生成を補う方法として、北村らにより生成カスタマイズ手法が提案されている⁽⁸⁾⁽⁹⁾。北村らにより提案された手法では、まず、予め表示部品と、それと連係するデータ処理部品を用意する。次に、応用プログラム中に現れるデータや関数の定義から、上記部品を用いてデフォルトの GUI を自動生成する。そして、応用プログラム実行時に、動作確認を行ないながら、画面上における対話的な修正を通じて、生成された GUI を最終的な GUI に変形していく。

生成カスタマイズ手法は、ソフトウェア自動生成を補う手法である。本研究では、監視制御システムのソフトウェア開発における各種データ定義の作成において、生成・カスタマイズを導入する。生成カスタマイズ手法を各種応用プログラム生成系と組み合わせ、フレームワークとしてまとめ上げる。例えば、伝送系定義の自動生成と生成後の編集、さらにその結果を段階的に詳細化していくという一連の定義データ作成手順を、ツール等の支援を用いて体系化する。そして、これにより、ソフトウェア自動生成を大規模システムの開発の中で有効に作用できるようにする。

3. GRASS フレームワーク

以下では、GRASS フレームワークの実現について述べる。GRASS は、オンライン実行系である「エグゼキューティブ」とオフライン生成系である「コンフィグレータ」から構成される。以下では、まず、オンライン系のソフトウェア構成について説明する。次に、これらオンライン系をサポートするオフライン系のツール群の役割について説明する。その後、フレームワークの特長である論理クラスと生

図3 基本プロセス構成

Fig. 3. An example of the GRASS online system

成・カスタマイズについて述べる。

〈3・1〉 エグゼキューティブ エグゼキューティブは、スクリプト言語や定義データにより動作をカスタマイズできるプロセス群から構成されるオンライン系部分のソフトウェアである。実行可能なデータ構造やスクリプト言語を外から与えることで動的な振る舞いをカスタマイズできるソフトウェアを一般的にエンジンと呼ぶ。GRASSでは、カスタマイズ性を向上させるため、様々なソフトウェアをエンジン化している。図3にエグゼキューティブのプロセス構成例を示す。

図において、監視制御対象の設備の情報は、I/Oサーバ経由で、主計算機上の監視制御エンジンに入力される。監視制御エンジンでは、値の範囲チェックや事故判定等の監視処理が実行される。そして、その結果は共有値サーバ経由でGUIプロセスや他の応用プログラムに渡される。また監視制御エンジンでは、状態変化をトリガとして制御スクリプトが実行される。GRASSでは、監視制御処理を定義データにより柔軟にカスタマイズするため、Javaスクリプトと類似した制御構文を持つHGSと呼ぶスクリプト言語を開発した。そして、後述の論理クラス定義における処理の記述、図的編集系を用いて入力する制御シーケンスや分野特化の特殊な事故判定処理を、このHGSスクリプトのプログラムに変換し実行するようにした。

本フレームワークの開発では、プロセス構成の柔軟性を確保するため、まず、GNP (Grass Network Protocol)

と呼ぶプロトコルを開発し、全てのプロセス間の通信をGNPに統一した。GNPは、監視制御データの通信において、オブジェクトとその属性をキーとして、時刻付きの値の交換を効率良く実現できることを目標に、購読処理パターン(Publisher-Subscriber)⁽⁵⁾を基本として、GRASSフレームワークのために設計した専用プロトコルである。そして、リクエストブローカと呼ぶプロセスを配置して、各応用プログラムからのデータリクエストを、その内容に応じてディスパッチできるようにした。例えば、電力系統監視制御の場合は、潮流計算を用いた監視値の評価や制御操作に先立つ充停電区間の検査処理等が必要となる。この場合、まず、対象設備の電力回路のモデルを作成する。この時、各モデルとそのモデルの持つ状態値には、対応する設備のオブジェクトと属性を割り付ける。次に、このモデルにシステムの現在状態を反映させ、計算処理を行うサーバプロセスを作成する。そして、このサーバをリクエストブローカに登録することにより、各種プログラムがアクセスできるようにする。なお、本フレームワークのオンライン系においては、上記プロセス構成が必須ではない。

〈3・2〉 コンフィグレータ 図4にコンフィグレータのソフトウェア構成を示す。GRASSにおいて、コンフィグレータの基本的な役割は自動生成である。しかし、自動生成だけに頼ると柔軟性が低くなる。そこで各段階のデータやスクリプトの編集系を用意し、編集系によるカスタマイズを可能とすることで自動生成を補う。

対象分野が同一であるシステム毎の開発では、コンフィグレータによる生成カスタマイズ作業の繰り返しとなる。まず、分野向けに用意された設備定義編集系(回路図等の設備図面利用)を用いて設備定義を入力する。次に、設備定義から監視画面定義、入出力項目定義を生成し、それらを編集する。その後、入出力項目定義から、I/O割付け定義、集計定義、監視処理定義等を生成し、また、生成された定義を編集する。

カスタマイズが必要であったとしても、自動生成は生産性向上に大きな威力を発揮する。例えば、監視画面を例とする。設備図面から監視画面定義を自動生成した場合、生成された画面定義はデフォルト画面としての扱いになる。自動生成後、画面編集ツールで見かけ(レイアウト)を変更する必要がある。しかし、最初からの編集と比較して、編集作業の手間が少なくなる上、自動生成時に監視画面上の図記号と設備との結合が自動的になされているので、編集時にその設定を行なう必要がない。これにより作業時間が短縮される他、設定誤りも減少する。

前述のように、GRASS設計のキーポイントの一つは拡張性である。GRASSの開発対象とする監視制御システムには、あらかじめ準備された仕様に基づくパッケージソフトウェアだけでなく、客先の好みへの対応が必要なカスタムソフトウェアも含まれる。また、パッケージソフトウェアであっても差別化のため、機能を拡張していくことも想定される。この場合の最終的な拡張手段はプログラミングで

図 4 GRASS 全体構成
Fig. 4. Organization of GRASS framework

あり、容易に拡張できるようにフレームワーク設計を行った。しかしながら、プログラミングにはオブジェクト指向言語を使える技量と、設計されたフレームワークに対する理解が必要となる。そこで、システムの拡張時におけるプログラミングの必要性を減少させるため、クラス定義編集系や制御ロジック編集系等のメタレベルのツールを用意した。

〈3・3〉論理クラス定義 GRASS フレームワークでは、論理クラス定義に従って、エグゼキューティブやコンフィグレータの各ソフトウェアが動作する。図 5 にクラス定義例を示す。図では、クラス `ConnectedEquipment` を親クラスとして持つ 2 つのクラス `LineSwitch` と `Transformer` を定義している。

クラスやその属性は、自動生成等の制御のための修飾情報を持つ。クラス定義において、クラス名や属性名の後の「:」で区切られた残りの部分が修飾情報を与える部分である。例えば、クラス `LineSwitch` には、6 つの属性が定義されている。このうち文字列型の属性 `type` には、`persistent` と指定されている。`persistent` は、属性の格納に関する修

飾子であり、これにより属性 `type` の値は、設備定義としてデータベースに保存することを意味する。そして、図 4 のプログラム生成系は、`persistent` と記された属性について、データベースのスキーマやアクセスライブラリの生成処理を行う。一方、属性 `openCloseState` には、`raw input` と `iospec` という修飾がなされている。これは、2 接点により処理される入力点を有する属性であることを意味する。そして、`sporadic` により、データ転送においては、変化時にのみ実施することが適切であることを指定する。エグゼキューティブでは、この情報を用いて、データ転送を定周期や状態時に行う等の処理を切替える。

論理クラス定義では、属性間の関係も修飾情報として記述する。例えば、クラス `Transformer` において、属性 `inputVoltageHighLimit` と属性 `inputVoltageLowLimit` は、それぞれ、属性 `inputVoltage` の上下限值である。また、同じくクラス `Transformer` において属性 `inputAvr` は、属性 `inputVoltage` を定周期で平均値を集計したものである。なお、属性修飾において、`shared` と記されているもの

```
class LineSwitch : inherit ConnectedEquipment {
String type: persistent;
boolean openCloseState:
    raw input, iospec doubleDI,
    shared,sporadic,
    action {
        if (openCloseState == 1)
            postAlarm();
    };
boolean openCloseCtrl: raw output, iospec
    control port of openCloseState;
boolean selectState:
    raw input, iospec singleDI,
    shared,sporadic;
boolean selectCtrl: cooked output,
    control port of selectState,
    select port of openClose;
boolean disableUpdate: shared,
    update inhibit of openCloseState,
    alarm inhibit of openCloseState;
};

class Transformer : inherit ConnectedEquipment {
float voltageRank: persistent;
float ratedVoltage: persistent;
float inputVoltage: cooked input,shared,periodic;
float inputVoltageHighLimit: shared,
    high limit of inputVoltage,init(11000);
float inputVoltageLowLimit: shared,
    low limit of inputVoltage,init(9000);
float inputAvr: compute, avr of inputVoltage;
float inputMax: compute, max of inputVoltage;
float outputVoltage:
    cooked input,shared,periodic;
};
```

図 5 論理クラス定義例

Fig. 5. An example of logical class description

は共有値サーバに、compute と記されているものは集計エンジンに格納される。

〈3・4〉生成処理 論理クラス定義から設備定義編集系を作成する部分については、筆者らによる従来の実装⁽¹⁰⁾を改良したものとなっている。以下では、新たに導入した生成機能について述べる。

GRASS では、設備定義を元に入出力項目定義、I/O 割り付け定義、集計定義等の様々なシステム依存のデータを自動生成する。GRASS において、設備定義の入力に先立ち、論理クラス定義から、データベーススキーマ等のプログラム生成を行い、設備定義編集系を作成する。そして、この編集系を用いて設備定義を入力する。設備定義の入力は、論理クラスのオブジェクトのインスタンス生成に対応する。論理クラスの各属性には、図 5 の様に、そのデータ型や入出力接点に関する情報が付加されている。従って、インスタンスを数え上げれば、必要なデータ領域の計算や入力接点の割り付け、また集計定義の作成を機械的に実行できる。図 6 に生成処理を示す。

また、GRASS では、設備図面からデフォルトの監視画面定義を自動生成することにより、監視画面定義の入力の手間を軽減する。例えば、電力系統設備の場合、設備図面は単線結線図を用いる。GRASS では、これを記述言語ベースと図的編集操作に基づくもの⁽¹³⁾の 2 種の変換処理を組み合わせることで、これらの設備図面から監視画面定義を生成する。一般に設備図面上の表現と監視画面上の設備の表現は、その監視目的に応じて異なる。例えば、監視画面定義

設備定義

生成

アナログ値監視定義

型	名称	種別	上限値	下限値
float	TX-A	inputVoltage	11000	9000
float	TX-B	inputVoltage	11000	9000

集計定義

型	名称	種別	演算	入力名称	入力種別
float	TX-A	inputAvr	avr	TX-A	inputVoltage
float	TX-A	inputMax	max	TX-A	inputVoltage
float	TX-B	inputAvr	avr	TX-B	inputVoltage
float	TX-B	inputMax	max	TX-B	inputVoltage

I/O 割付定義

型	名称	種別	接点形式	I/F 番号	ピン番号
bit	LS-A	openCloseState	double	0	0
bit	LS-B	openCloseState	double	0	1
bit	LS-C	openCloseState	double	0	2
bit	LS-A	selectState	single	1	0
bit	LS-B	selectState	single	1	1
bit	LS-C	selectState	single	1	2

図 6 定義データの生成

Fig. 6. Generation of description data

においては、監視の詳細度に応じて監視画面上には設備が省略表示される。あるいは、状態を示すための数値表示やグラフ表示が追加される場合がある。さらには、図記号の配置も設備図面と監視画面では異なる場合がある。設備図面から監視画面を自動生成するためには、これらの処理を扱えなければならない。

図 7 に記述言語ベースの変換ルール記述例を示す。図において、TransformerView は図記号のクラス名であり、設備図面においては、このクラスを用いて Transformer オブジェクトの入力を行う。図の変換ルールは、設備図面中の全ての TransformerView のインスタンスオブジェクトに対して適用され、各オブジェクトに対し、‘{ }’で囲まれたルール本体が実行される。ルール本体は、手続き型のスクリプト言語で記述する。図のルール適用により、対応する図記号の右上に、Transformer オブジェクトの inputValue について、その値が上限値を超えると赤色表示となる数値表示部品が生成される。

4. 適用及び評価

図 8、図 9 に電力系統監視制御システムを対象にしたコンフィグレータの画面例を示す。

```

rule TransformerView {
  if ($eval($$.voltageRank) > 500000) {
    $create NumberButton(
      x = $eval(x+40), y = $eval(y-50),
      width=40, height=16,
      value = $$inputValue,
      fillColor = $color("black"),
      foreground
        = ($$.inputValue > $$inputValueHighLimit
          ? $color("red") : $color("blue"));
    )
  }
}

```

図 7 記述式変換ルール例

Fig. 7. An example of transformation rule

図 8 において、画面左上は、コンフィグレータのメインメニューウィンドウ、画面右下は電力回路ボタン編集系のウィンドウである。メインメニューにおいて、ディスクを示す図記号は、各種定義データに対応する。この図記号をマウスでクリックすることにより、該当データの編集系を起動する。一方、太線矢印は生成処理を示しており、矢印をマウスでクリックすると、対応する生成処理を実行する。図中、電力回路ボタン編集系では、母線から送電線への引出し部分の回路について定義している。図において、8 個の主回路設備と、それら設備間の回路線が示されている。しかしながら、主回路設備以外にもリレーや制御用スイッチ等が回路図に付随してグループ化されており、定義全体では、約 100 個の入出力接点を持つ複合設備として定義されている。

図 9 に、電力回路の設備図面編集系と入出力項目定義編集系画面例を示す。設備図面編集系における設備の入力では、図 8 に示す電力回路ボタン編集系で作成した複合設備を組み合わせて全体回路を作成する。この時、回路図上には現れないセンサーや制御スイッチ等の設備は、図中右の階層的ツリーで設備構成を表示するウィンドウ経由で入力する。設備定義入力後、図 8 のメインメニューで生成処理を起動すると入出力項目定義が生成される。そして、入出力項目定義編集系等を用いて、カスタマイズ処理を行う。入出力接点情報の割付けにおいては、設備の物理的配置や利用する伝送装置を考慮する必要がある。電力系統監視制御分野においては、交流波形の位相比較等を必要とする制御が存在し、特定の入力接点を同一の I/F ボードで処理させる等の割付け制約が存在する。開発したシステムでは、フレームワークが提供する生成プログラム作成用ライブラリを用い、上記、制約を考慮した生成プログラムを実装、システムに組み込んだ。なお、メインメニューウィンドウ自身、コンフィグレータの監視画面編集系で作成したものである。監視画面ウィンドウとの違いは、そのウィンドウに付随するメニューボタン等だけであり、プログラムモジュールとしては、同一のものとなっている。これにより、監視処理を実行しながら画面編集をするオンライン画面編集のような、従来システムでは実現が困難であった機能を本フレームワークでは容易に実現できた。

現在、GRASS フレームワークを用い、電力系統監視制

御システムを開発中である。実システム向けのデータを作成、これを用いて各種エンジンの性能を評価中である。本システムに開発においてプログラミング作業が必要となったのは、入力接点情報の自動生成機能と伝送装置固有のデータ設定機能と、そのオンライン系インタフェース部分だけである。一方、定義情報の入力については、設備及び伝送系定義については従来の 1 / 5 程度に、また、監視画面定義については、監視画面定義の自動生成と編集機能の向上により、従来の 1 / 3 程度にまで短縮できた。

近年、新設の変電所では、GIS (Gas Insulated System) を利用して電力回路を構成することが多い。この場合、予め GIS 本体を複合設備として定義しておき、複合設備を組み合わせることで全体電力回路を構成する。また、入力された設備定義から、入出力項目の抽出や I/O 割付け定義の生成を行うのであるが、新設の場合、生成される割付けを修正する必要性は少ない。このため、従来のシステム開発で数ヶ月要していた作業を数日で完了することが可能である。

GRASS フレームワークでは、オブジェクト指向方式をシステム基本部に導入し、システム全体を完全に新しい設計で実現した。監視制御システム、特に、オンライン系部分には、頑強性と安定性が強く求められる。このため GRASS フレームワークの実システムへの適用では、オフライン系ツールのみの導入と比較し、システム動作の検証に、より多くの手間を必要とする。実運用されているシステムに対して、GRASS で用いた技術の部分的な適用は、既に実施しているが、全面的な適用は今後の予定である。実適用を通して得られる情報、例えば、運用面から発生する定義データ変更要求に対するカスタマイズ性、また、オンライン処理部の拡張性等についての評価が、今後の課題として残されている。

5. ま と め

本論文では、監視制御システムの構築を題材に、生成カスタマイズ手法の枠組の中に、各種の応用プログラム生成系やメタツールを統合したアプリケーションフレームワークについて述べた。また、電力系統監視制御システムの開発に適用することにより、その有効性を示した。

本フレームワークでは、監視制御対象の設備について、オブジェクト指向方式のクラス概念を用いてモデル化する。次にこのモデルに従い、監視制御システムのプログラムやシステム依存の定義情報を自動生成する。そして、生成されたソフトウェアを、そのソフトウェア自体が持つ拡張性や編集ツール等を用いてカスタマイズし、対象とする最終的なソフトウェアを得る。これら一連の処理手順やツール群をフレームワークとしてまとめることにより、大規模監視制御システムのソフトウェア開発に対し、ソフトウェア自動生成技術を効果的に適用できることを示した。

生成とカスタマイズを組み合わせる場合、再生成の問題について検討する必要がある。これは、自動生成されたソ

図 8 画面例（トップメニュー）
Fig. 8. Screen example(top menu)

図 9 画面例（入出力項目表）
Fig. 9. Screen example(I/O description)

ソフトウェアを編集系等により修正した後、再度生成処理を実行する場合、カスタマイズ結果を再生成されたソフトウェアにどのように反映するかという問題である。一覧表のような形式で生成される単純なデータについては、カスタマイズ結果の反映処理を機械化することは比較的容易である。しかしながら、オブジェクト間の関係を変更するような場合は、その扱いは単純ではない。今後の課題として、この再生成の問題の一般化について検討してゆきたいと考えている。

(平成 11 年 1 月 11 日受付, 同 11 年 5 月 24 日再受付)

文 献

- (1) "S.P.I.D.E.R. EMS/DMS/SCADA Open System Architecture", ABB Network Control & Protection.
- (2) "FIX DMACS システム開発", Intellution R, Inc. (1994).
- (3) Batory, D., Barnett, J.R., Garza, J.F., Smith, K.P., Tsukuda, K., Twichell, B.C., and Wise, T.E.: "GENESIS: An Extensible Database Management System", *IEEE Trans. Softw. Eng.*, Vol.14, No.11 (1988).
- (4) Cheatham, T. E.: "Program Refinement by Transformation", *Proc. of 5th Int. Conf. on Softw. Eng.*, pp. 430-437 (1985).
- (5) Coad, P., North, D., and Mayfield, M.: "Object Models: Strategies, Patterns, and Applications", Prentice Hall (1997).
- (6) "DEC VUIT User's Guide", Digital Equipment Corporation (1991).
- (7) Johnson, S.C.: "Yacc - Yet Another Compiler Compiler", *UNIX Programmer's Manual 2*, Section 19, AT&T (1979).
- (8) 北村操代, 杉本明: "GUI 生成・編集機能を持つクラスライブラリ GhostHouse", 情報処理学会 ソフトウェア工学研究会, Vol.92, No.59, pp. 27-34 (1992).
- (9) 北村操代, 杉本明: "生成・カスタマイズ手法によるソフトウェア開発方式", 情報処理学会 第 48 回全国大会 (5), pp. 165-166 (1994).
- (10) 小島泰三, 中田秀男, 川岸元彦, 上原拓: "監視制御システム向けデータベースの構築", 電気学会 論文誌 C, Vol.117-C, No.4, pp. 458-468 (1997).
- (11) Lesk, M.E.: "Lex - A Lexical Analyzer Generator", *UNIX Programmer's Manual 2*, Section 20, AT&T (1979).
- (12) Neighbors, J.M.: "DRACO: A Method for Engineering Reusable Software Systems", in *Software Reusability Vol.I Concepts and Models*, Biggerstaff, T.J., and Perlis, A.J., eds., ACM Press, pp. 295-319 (1989).
- (13) 中田秀男, 小島泰三, 杉本明: "設備を管理する図面から監視画面を自動生成するプラント監視システムの構築", 電気学会 論文誌 C, Vol.118-C, No.6, pp.856-864 (1998).
- (14) 大脇隆志, 廣田敦彦, 松田伸之, 阿部嘉治: "オブジェクト指向形マンマシンシステム構築支援システム Hyproof/MMS", 日立評論, Vol.73, No.8, pp. 73-78 (1991).
- (15) Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorenson, W.: "Object-Oriented Modeling and Design", Prentice-Hall (1991).
- (16) "つばさモニターメーカー看太郎 32 ユーザーズガイド", 椿本チェイン, (1996).

小島泰三 (非会員) 1960 年生。1982 年 3 月東京大学工学部船舶機械工学科卒業。同年 4 月三菱電機(株)入社。同社中央研究所にて精密機械の計測と制御, 設計支援システム, オブジェクト指向言語などの研究に従事。現在, 同社産業システム研究所にて, 監視制御システム, イントラネット産業応用システムの研究開発に従事。工学博士。情報処理学会, IEEE 各会員。

中田秀男 (正員) 1964 年生。1987 年 3 月京都大学工学部情報工学科卒業。1989 年 3 月同大学院工学研究科情報工学専攻修士過程修了。同年 4 月三菱電機(株)入社。産業システム研究所にて, オブジェクト指向システム, ユーザインタフェース, 視覚的プログラミングなどの研究に従事。情報処理学会会員。

丹下正純 (非会員) 1960 年生。1984 年 3 月愛媛大学大学院工学研究科電気工学修了。同年 4 月三菱電機(株)入社。系統変電・交通システム事業所にて電力系統監視制御システム, 変電所監視制御システムなどの設計に従事。

西田正吾 (正員) 昭和 27 年 1 月 5 日生。昭和 49 年 3 月東京大学工学部電子工学科卒業。昭和 51 年 3 月同大学院修士課程修了。同年 4 月三菱電機(株)入社。同社中央研究所システム基礎研究部研究員, グループマネージャーを経て, 平成 7 年 4 月, 大阪大学基礎工学部システム工学科教授。現在, 大阪大学大学院基礎工学研究科システム人間系専攻教授。システム技術, ヒューマンインタフェース

技術, メディア技術の研究に従事。工学博士。昭和 59-60 年 MIT メディアラボ客員研究員。61 年度, 平成 5 年度電気学会論文賞, 平成 4 年度電気学会著作賞, 平成 7 年度電気学会進歩賞受賞。電気学会, 情報処理学会, IEEE ほか会員。IEEE Fellow。著書は, 「ヒューマン・コンピュータ交流技術」(オーム社, 共著), 「メディア工学」(朝倉書店) など。