ソフトウェア工学 83－5
（1992. 2. 6）

オブジェクト指向によるFA用外部装置の仮想化手法

碇崎 賢一[+] 大畑 浩司[++] 松本 俊哉[+] 打浪 清一[+] 植田 敬子[++]

[+]九州工業大学情報工学部

[++]安川情報システム（株）制御システム事業部

企業の基幹部門を制御するFAシステムは，適用範囲が拡大するにつれて大規模化かつ複雑化しており，その効率的な開発手法と，安定的かつ継続的に運用するための保守業務の効率化が強く望まれている．本報告は，FAソフトウェア開発の困難さの主要因である多様な外部装置に着目し，これに関連したソフトウェア開発工程の効率化をはかることによって，FAソフトウェアの生産性と品質を向上させようというものである．このような目標を実現するために，外部装置をオブジェクト指向の概念により仮想化するとともに，大規模なFAシステムの開発に適した自律化や組織化の機能を導入し，これに基づいてFAソフトウェアを開発する方式を提案する．

An Object-Oriented Abstraction Method for Factory Automation Devices

Ken'ichi KAKIZAKI[+], Kouji OHATA[++], Toshiya MATSUMOTO[+],
Seiichi UCHINAMI[+], Keiko UEDA[++]

[+]Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology
680-4, Kawazu, Iizuka, Fukuoka 820, Japan

[++]Yasukawa Joho System K.K.
5-5, Higashiouji, Yahatanishi, Kitakyushu 806, Japan

In recent years, Factory Automation Systems are widely used for the nucleus part of corporation activities, and the system scale and complexity are being increased. It is required an efficient development and maintenance methodology for the large and high quality systems.

This paper introduces our approach for the Factory Automation System Development. On the system development, the difficulties are caused by the diversity of the devices. In our approach, the devices for the Factory Automation Systems are abstracted based on Object-Oriented concept. The abstracted devices are incorporated some functions such as autonomy and systematization suited for efficient development for large systems.

1．はじめに

FA システムは企業の基幹部門に広く適用されるにつれて大規模化かつ複雑化しており，その効率的な開発手法と，安定的かつ継続的に運用するために必要な保守業務の効率化が強く望まれている．しかしながら，FA ソフトウェアの重要な位置を占める外部装置のプログラミングは，その対象が多種多様にわたり機能や操作法が統一されていないことと，外部の事象に同期をとらなければならないなどの時間にきびしいという問題があるために，非常に困難なものとなっている．このため，外部装置などのハードウェアにも詳しい熟練したソフトウェア技術者が必要である一方で，そのような技術者を配置しても生産性が向上しないという矛盾が生じている．

本研究は，FA ソフトウェアの開発，保守を効率的に行うための枠組を確立することを目標としている．このために，オブジェクト指向に基づくソフトウェアの部品化と標準化により，再利用性と変更の容易性を高め，開発および保守作業を効率化するとともに，ハードウェアに依存する部分と依存しない部分に分離し，技術者がその技能分野に適した作業に専念できるようにする．また，この様な目的にあった開発支援システムを構築することにより，設計，プログラミング，テスト，文書化などを効率的に行えるシステムの構築を目指している．

本報告では，FA システム開発をオブジェクト指向で統一的に行うためと，FA ソフトウェア開発の困難さの主要因である外部装置に関する開発効率を向上させるために，オブジェクト指向による外部装置の仮想化手法[1]に関して示す．また，仮想化された外部装置に，大規模な FA システムの開発に適した自律化と組織化の機能を導入し，これに基づいて FA ソフトウェアを開発する方式を提案する．

2．オブジェクト指向による仮想化

2．1 外部装置の仮想化

OA システムの分野では，端末，プリンタ，ディスクなどの主要な周辺装置がオペレーティングシステムによって仮想化され，そのインターフェースが標準化されているために，ソフトウェア技術者が周辺装置の多様性に影響されずにシステムを記述できるようになっている．このため，ハードウェアに詳しくなければならない技術者は限定されており，ハードウェアの詳細を知らない一般的な技能レベルの技術者が，ハードウェアの機能を効果的に利用したソフトウェアを一定レベルの品質を維持しながら効率的に開発できる環境が実現している．

FA システムではOA システムと異なり，応用分野ごとに使用される外部装置の多様性が極めて高いために，外部装置の仮想化が進められていなかった．このため，ほとんどの技術者に，ソフトウェア的な知識だけでなく，そのシステムで使用される外部装置のハードウェア的な知識が要求されている．また，同種の外部装置であっても操作概念や操作用の命令体系が異なっており，外部装置が変るごとに，外部装置に関係する処理をプログラム全体にわたって書き換えなければならないという問題もある．このような状態では，ソフトウェアを開発できるのは高度な技能を持つ技術者に限定されるだけでなく，そのような技術者であっても生産性が低くなると共に，ソフトウェアの信頼性，保守性，汎用性などが著しく損なわれることになる．

FA システムで使用する外部装置の特性は，いくつかの典型的なモデルに分類することが可能であると考えられるため，ディスク装置や端末装置などに用いられているのと同様な考え方を FA ソフトウェアの開発にも取り入れ，外部装置を仮想化することにより，その生産性，信頼性，保守性を大幅に向上させることができると考えられる．

2．2 オブジェクト指向

オブジェクト指向はソフトウェアをオブジェクトと呼ばれる構成要素に分割し，その機能と外部とのインターフェースを明確化することにより，プログラムの抽象度と独立性を向上させようとする概念である．オブジェクト指向は，現実に存在する＂物＂を抽象化することに特に適しており，外部装置を操作する FA ソフトウェアに非常に適した概念である．

オブジェクト指向では，処理対象を抽象化したオブジェクトの機能と外部インターフェースが明確になると共に，オブジェクト内部のデータの安全性を保証できるために，多人数によるプログラム開発を効率的かつ安全に行えるようになるという利点がある．オブジェクトの利用者は，そのオブジェクトの内部で行われている詳細な操作を知らなくてよく，どのような結果が得られるかさえ知っていればプロ

グラムを記述できるようになる．

オブジェクト指向の継承機能を利用することにより，すでに開発されているオブジェクトを基本として異なる部分の追加によってシステムを開発できるために，ソフトウェアを部品として利用し，効率よく開発を進められるという利点がある．また，継承機能を利用することによって，容易に系統的な標準化が行えるようになる．

オブジェクト指向では，オブジェクト間の手続き呼出はオブジェクト相互のメッセージ交換で行われ，各オブジェクトは他のオブジェクトと協調して処理を行う自律的な処理単位となっている．この様に，ソフトウェアの構成要素が自律的な処理機能を持つことは，大規模なシステムを構築する場合に強く望まれることである．

2．3　開発環境

従来は FA ソフトウェアに適したオブジェクト指向システムの開発環境がなかった．しかしながら最近では，C 言語にオブジェクト指向の概念を取り入れた C++[2] などの処理系が容易に利用できるようになってきており，今までに蓄積したソフトウェア資産をそのまま継承できると共に，ソフトウェア技術者が比較的容易に移行することができる．また，C++ では，抽象化機能や継承機能などのオブジェクト指向言語に特長的な高度な機能を利用しても，C 言語と同様に，処理速度が高くメモリ容量もそれほど増大しないという特長がある．

FA 用のコンピュータでは，プロセッサの性能やメモリ容量に制約が多いという問題があるが，最近では大容量のメモリと高速なプロセッサを低コストで導入できるようになり，オブジェクト指向でソフトウェアを実現するための十分なハードウェアが低コストで得られるようになってきている．一方，人件費は上昇の一途をたどっており，ソフトウェアの大規模化と複雑化にともなう人件費の増大による，システム開発コストの増大が大きな問題となっている．このため，オブジェクト指向の導入にともなってハードウェアのコストが多少上昇したとしても，ソフトウェアの開発，保守コストの低下額の方が大きくなり，システム開発の全体的なコストを抑えることができると考えられる．

3．仮想化外部装置への要求

FA システムで使用される外部装置の基本特性をオブジェクト指向の概念に基づきモデル化すると共に，大規模システムの構築に適した機能を付加することによって，以下に示す 5 項目の機能を実現する．

1）抽象化
2）標準化
3）個別化
4）自律化
5）組織化

仮想化された外部装置は，図 1 に示すようにデバイスユニットと呼ばれ，ソフトウェア的に構成されるデバイスオブジェクトと，外部装置のハードウェアから構成される．デバイスオブジェクトの内部に定義されるデータや手続きは，外部装置の状態を保存したり，外部装置に指示を与える機能を持つ．なお，外部装置には，純粋な機械装置だけではなく，シーケンサーなどの制御装置が含まれる場合もある．

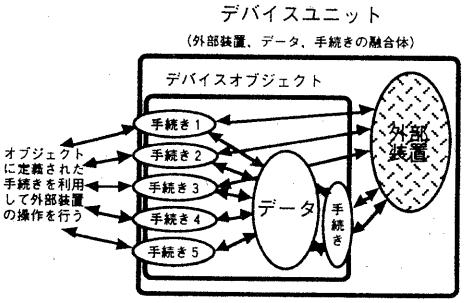

図 1　デバイスユニット

以下の節では，自動倉庫に使用される外部装置を例として，これらの機能を簡単に説明する．自動倉庫は，物品の在庫管理と入出庫の制御をコンピュータにより行うもので，代表的な外部装置として，クレーン，コンベア，カートラックなどがある．クレーンは，物品収納棚に対して，物品の格納と取り出しを行う．コンベアとカートラックは物品を運搬し，クレーンと外部との物品の入出庫を仲介する．

3．1　抽象化

一般的なソフトウェア技術者が，使用する外部装置の信号と時間的な要因も含めた複雑な制御シーケ

ンスなどの詳細を知らなくてもシステムを開発できるように，外部装置の機能と外部インターフェースをデバイスユニットという形式で抽象化する．このようにすることにより，外部装置の構成に依存する詳細な操作はデバイスオブジェクト内部に隠蔽されるために，システムのハードウェア依存性の局所化をはかることができ，外部装置に変更が必要となった場合に，その影響をデバイスオブジェクト内に限定することができると共に，外部装置の複雑な操作を把握し，その制御プログラムを記述することができる高い技能を持つ技術者に依存する部分を局所化することができる．

3．2　標準化

同種の外部装置を一貫した概念で操作できるように，基本的で共通した要素を取り出し，デバイスユニットの機能と外部インターフェースを標準化する．たとえば，使用するクレーンの種類が数種類あったとすると，それらの機能や操作法の詳細が異なっていても，基本的には同一の操作を行う外部機器である．したがって，同一の外部機器として操作できるように機能と外部インターフェースを標準化することにより，その利用者は共通の機能と操作法（外部インターフェース）を継続して使用できるため，それらを再修得する必要がなくなる．

標準化の考え方をさらに発展させて，同様な機能を持つ外部機器に限定せず，同じ概念に基づく機器の操作を，同一のインターフェースによって操作できるようにすることも望まれる．例えば，自動倉庫で使用される，クレーン，コンベア，カートラックなどの外部機器は，それぞれの種類の中で機能と外部インターフェースの標準化を行うのはもちろんであるが，それだけでなく，これらの外部装置の基本機能が荷物を運ぶことであり，荷物を取り出し，運搬し，格納するという共通の機能を持つことに着目して，物品の運搬装置としての標準的な機能を外部インターフェースを定める必要がある．

各デバイスユニットの機能とインターフェースを明確に定義し標準化することにより，仕様が異なる新たな外部機器を導入した場合でも，そのデバイスユニットを使用するシステムの設計，プログラミング共に技術の修得が容易になる．また，異なるソフトウェアであっても，同様なデバイスユニットを利用する用途に対しては，同一の機能とインターフェースを繰り返し使用して開発を行うことになるので，習熟による技術者の技能の向上により，生産性と品質の向上が期待できる．また，デバイスユニットは，その外部装置を利用するシステムでは繰り返し使用されるため，検査と改良が繰り返し行われることになり，品質の向上が見込まれる．

3．3　個別化

外部装置の多様性と発展性に対処するために，標準化と対極的な要求であるが，外部装置の機能と外部インターフェースの個別化が必要となる．

FA　システムでは使用される外部装置が多種多様であるだけでなく，目的に応じて特殊な装置が特別に作成されることも頻繁にある．この様な装置を含めて，抽象化，標準化の効果を上げるためには，すでに抽象化，標準化されている外部装置を基にして，特別な外部装置にあわせたデバイスオブジェクトを容易に作成できるようにする必要がある．

3．4　自律化

システムが大規模になるにつれて，外部装置を集中的に管理することが困難になるために，各外部装置を自律的な単位として抽象化する必要がある．

外部装置を制御する方式としては，管理モジュールをシステムの核として用意し，すべての外部装置の状態を把握し，詳細な指示を与えさせる集中管理方式と，各外部装置に可能な限りの自律性を持たせ，管理モジュールは，必要最小限の管理しか行わない分散管理方式の２種類が考えられる．

たとえば，カートラックを指定位置に移動させる場合には，集中管理方式では図２に示すように，まず，管理モジュールがカートラックの駆動装置を起動して移動を開始させる．次に，管理モジュールはカートラックの位置を追跡し，所定の位置に達した時点で駆動装置を停止させる処理を行う．

管理モジュールに着目すると，１つ１つの外部装置を操作するために必要な手続きや情報は少なくても，外部装置が多くなってくると，それらを操作するための手続きや情報が多岐にわたる上に膨大になる．このために，システム全体を大局的に統括する管理モジュールに，外部装置に関する細かな操作手続きを多数組み込まなければならなくなるが，概念

レベルの大きく異なる処理が混在すると，処理の目的と内容が不明確になるために，多くのバグを生じる要因となる．また，プログラムを機能別に分割して効率よく開発することが困難になる．

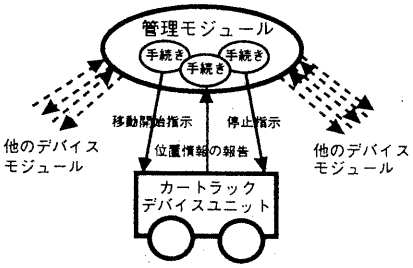

図2　集中管理方式

一方分散管理方式では，図3に示すように，管理モジュールがカートラックデバイスユニットに移動先を指示し，移動を開始させる．次に，カートラックデバイスオブジェクトはカートラックの位置を追跡し，カートラックが所定の位置に達した時点で駆動装置を停止させる処理を行う．この方式では，管理モジュールはカートラックに指示を出した後は，カートラックに関する処理を全く行う必要がない．

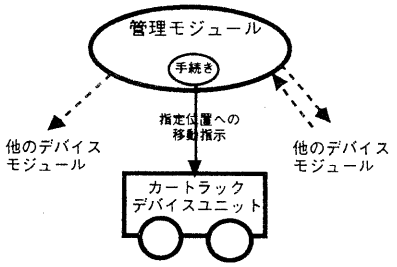

図3　分散管理方式

分散管理方式を利用した場合には，管理モジュールはシステム全体の大局的な管理とそれにともなう簡単な指示をデバイスユニットに出すだけでよいので，集中管理方式に発生するような複雑化の問題は生じない．また，性能上の問題などでマルチプロセッサ構成にする場合などに，システムの切り分けが容易に行えるという利点もある．

3．5　組織化

OA　システムでは，周辺装置間に物理的な接続関係がないのに対して，FA　システムでは，外部装置間が物理的に接続されており，物の移動などをともなうという点で大きな違いがある．FA　システムの外部装置が物理的に接続されるのと同様に，その制御を行うデバイスオブジェクトも，ソフトウェア的に接続し組織化する機能が必要であると考えられる．

FA　システムの複数の外部装置は，物品の受渡しなどをとおして関連付けられながら操作を進めて行く．このような処理では，複数の外部装置を関連付けるために，一般的に図4に示すように管理モジュールを介入させる方法が多用されている．

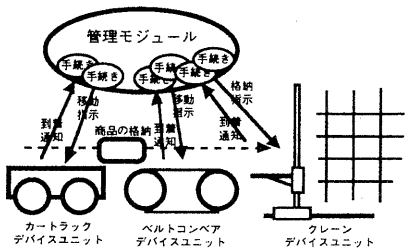

図4　デバイスユニットの個別管理

この場合も前節で述べたのと同様に，複数の外部装置が協調して行う処理が多くなればなるほど，それらの処理を行うための情報や手続きが管理モジュールに集中して複雑化し，結果としてソフトウェアの信頼性や生産性を低下させてしまうという問題を生じる．大局的な判断の必要がなく，該当する外部装置間のやり取りだけ行える処理は図5に示すように各外部装置間のやりとりに任せ，管理モジュールが関与しなくてよいようにする必要がある．

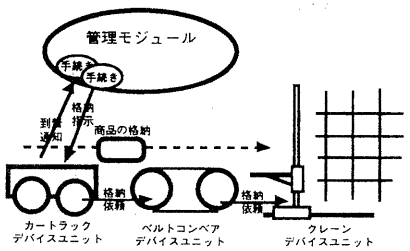

図5　組織化

複数の外部装置がどのように関連付けられるかということは，外部装置を利用する各応用プログラムごとに異なっており，デバイスユニットの利用者が行わなければならない．このためには，複数のデバイスユニットを自由に組織化することができる仕組みをデバイスユニットに組み込んでおく必要がある．

4．仮想化外部装置の構成

3章に示した要求を満たす仮想化外部装置の構成法を示す．

4．1　抽象化

自動倉庫に利用される外部装置は，物品を運ぶという共通の機能を持っており，その機能は，以下のように分類，定義される．

1）荷物の取得
2）指定位置への移動
3）荷物の格納

外部インターフェースとしては，以下の様なものが必要である．

1）動作の指示
2）状態の取得
3）制御情報の設定
4）割り込み

外部装置の仮想化には，C++ のクラス定義機能を用いる．各デバイスユニットに固有のデータは，データメンバを利用し，デバイスユニットの外部インターフェースは，メンバ関数を利用して定義する．

4．2　標準化

標準化は，C++ の多重継承を含むクラスの継承機能で実現することができる．継承機能を用いた標準化では，基本的に同様な機能をもつ外部装置の特性を基底クラスとしてまとめ，各デバイスユニットはこのクラスの機能を継承する派生クラスとして作成する．各派生クラスでは，そのクラス特有の機能に必要な処理を，基底クラスとの差分として追加，修正する．この様に継承機能を利用すると，共通する基底クラスを継承しているクラスのオブジェクトでは，機能や外部インターフェースが自然に標準化されると共に，新たなデバイスユニットの作成が容易に行えるという利点がある．

ここで，カートラック，コンベア，クレーンを仮想化したデバイスユニットをそれぞれ Cartrack, Convair, Crane とする．これらのデバイスユニットは，物品を運搬する機能を共通に持つので，まず，運搬機能を持つクラスをCarrier クラスとして定義し，Cartrack, Convair, Crane の各クラスは，Carrier の派生クラスとして定義する．さらに，物品を運搬するものに限定せず，外部装置に共通する基本的な機能だけを持つクラスを Device クラスとして定義すると，Carrier クラスは Device クラスの派生クラスとして定義することができる．この様子を図6に示す．

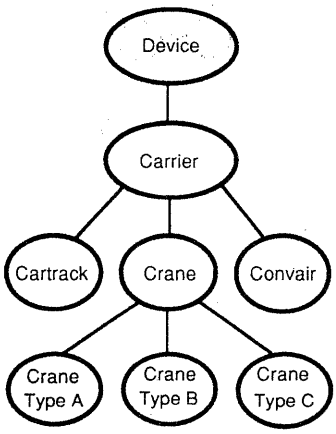

図6　継承による標準化

図6で Carrier クラスと Device クラスに定義されたメンバ関数は，Carrier の派生クラス Cartrack, Convair, Crane で自由に利用することができる．また，各クラスでそれぞれの装置の特性にあわせてメンバ関数を追加したり，修正し，基底クラスの定義を置き換え異なる処理を行うことになる．

4．3　個別化

個別化の機能は，C++ の多重継承を含む継承機能，差分の追加と修正で実現することができる．多重継承を利用すると，継承による標準化だけではなく，複数のクラスを同時に継承することによる個別化をはかることも可能になる．継承機能を利用した個別化は，基本的には標準化で示した派生クラスの作成と同様で，必要となるほとんどの機能を持つ基底クラスを継承し，それだけでは不足する機能を派生クラスの定義として新たに付加することによって行う．

4．4　自律化

自律化を実現するために，まず，デバイスユニットに組み込む比較的低機能の関数を保護されたメンバ関数として定義し，公開されたメンバ関数は，これらの関数の組み合せで機能を実現するようにする．この様にすると，外部からは，自律的にふるまう高機能のインターフェースのみが見えるようになり，低機能の関数を隠蔽することができる．また，この様にして作成されたデバイスユニットを基底クラスとして継承するデバイスユニットでは，保護されたメンバ関数として用意されている低機能の処理を組み合わせて新たな機能を作成できるようになる．

4．5　組織化

デバイスユニットは，オブジェクト指向の概念に基づいて構成されているので，複数の外部装置が通信しあって処理を進めて行く方式は，デバイスユニット間のメッセージの交換で処理を進めるという形式で自然に導入することができる．

デバイスユニットの組織化を実現するための方式として，コールバックと呼ばれる機能を用いる．コールバック機能は状態，手続き，手続きに渡すデータの組み合せをオブジェクトに設定するもので，オブジェクトの内部状態が指定した状態になった場合に，指定した手続きが指定したデータを伴って呼び出される．この様にして呼び出される手続きはコールバックルーチンと呼ばれる．コールバックルーチンの起動条件としての状態は，抽象化の際に定めた状態をそのまま利用できる．

たとえば，システムの初期化ルーチンにおいて，カートラックとベルトコンベアそれぞれに，ベルトコンベアまで物品を運んだ場合にベルトコンベアを起動させるコールバックと，クレーンまで物品を運んだ場合にクレーンを起動させるコールバックを設定しておく．この様な設定の下で，カートラックに指示を出すと，物品をクレーンで物品棚に格納するまでの処理が，管理モジュールの指示を受けることなく自動的に行われる．

コールバック機能を利用する方式では，システムの初期化の多くの部分は，各デバイスオブジェクトへのコールバックルーチンの設定によって行われることになる．この方式は，デバイスオブジェクトの組織化を実現できるだけでなく，コールバックルーチンを設定する系列が，プログラムにおける外部装置間の関係を示すことになり，その系列を参照することによって，システムの全体的な構成を一目で把握することができるという利点も生じる．

コールバックルーチンの設定は，必要に応じて実行時に行うこともできる．また，コールバックルーチンは，基本的にはデバイスユニット間の組織化のために用いられるが，デバイスユニット間だけの関係では，大局的な判断をともなう処理を行いにくいという問題がある．しかしながら，コールバックルーチンの内容はどのような処理でも記述できるために，大局的な判断を行いながら処理を進めたい場合には，コールバックルーチンの内容として，スケジューラ等の管理機構に問い合わせを行いながら処理を遂行するような手続きを記述することで柔軟に対処することができる．

5．仮想化外部装置の応用と研究の方向

この章では，仮想化外部装置の応用と今後の研究の方向について簡単に示す．

5．1　シミュレータによるテストとデバッグ

デバイスユニットの機能とインターフェースが明確に定義されているために，そのシミュレータを作成することができる．システムのテスト，デバッグ時には，実際に使用するデバイスユニットとシミュレータを置き換えることによって，外部装置がない状態で応用プログラムの機能検査を行うことができる．このため，外部装置を開発現場に持ち込んだり，外部装置の設置場所に出向いてテストやデバッグを行う必要がなくなり，非効率的な作業を軽減できる．

FA　分野における外部装置は可動部分が多いため，それらの状態をグラフィックス表示できるようなシミュレータを作成すれば，技術者が視覚的な情報を入手して大局的な監視を行いながら，効率良くシステムのデバッグと調整を行うことができる．シミュレータは，応用プログラムから受けた指示を実行し，その結果を表示するだけでなく，その逆の機能も必要である．シミュレータに，システムの設置環境から外部装置に与えられる事象を発生させる機能を付加することにより，予測される様々な事象系列に対して，システムがどのように反応するかということをテストすることができる．さらに，シミュレータ

に外部装置の定格を逸脱したり，定義されていない機能の実行を要求する指示が与えられた場合には，指示の内容とその指示が与えられた時点での外部装置の状態を報告すると共に記録する機能を付けると効果的であると考えられる．

5．2　プロトタイピング

シミュレータの利用は，開発されたプログラムのテストやデバッグなどのシステム開発の後半の工程だけではなく，プロトタイピングや外部装置間のスケジューリングなどのアルゴリズムの評価などのシステム開発の早い時期でも有効に利用することができる．これにより，技術的な実現可能性の評価だけではなく，プロトタイプを参考にして発注者との仕様の確認を早期に行えるという利点がある．

5．3　仮想化外部装置の使用者支援システム

オブジェクト指向によるシステム開発では，各クラスの機能やインターフェースを把握するために，クラスの階層関係や利用できるメンバ関数などを容易に参照できる必要がある．また，処理内容に関しては，メンバ関数の上書きや多態性により，処理の実際の定義の参照が困難であったり，メッセージの流れや，データメンバの定義と使用を把握しにくいといった問題が生じる．したがって，仮想化外部装置を用い，オブジェクト指向でシステムを設計，記述するためには，オブジェクト指向に特有なこれらの情報を効率良く参照するための支援システムが必須と考えられる．

ソフトウェアの生産性向上のためにCASEツールを用いた設計開発手法が注目されている．CASEツールでは，システムの分析や設計を行うために，データフロー図を記述して，データの流れと処理主体の関係を定義する．データフロー図で表されるデータの流れをメッセージ，処理主体をオブジェクトととらえると，オブジェクト指向の考え方を視覚的に表現したものだと考えることができる．したがって，CASEの手法とオブジェクト指向の概念を導入した仮想化外部装置とは，非常に親和性が高いといえる．

データフロー図に示される処理主体の間に，どのような関係があるかという情報は，どのデバイスユニットにどのデバイスユニットに対するコールバックルーチンを付加し，それの起動条件と使用されるデータは何かということと対応付けることができる．この様な対応付けを行うと，上流CASEで記述したデータフロー図の矢印からコールバックルーチンの定義系列を生成し，実行可能なコードを直接生成できると考えられる．

5．4　仮想化外部装置の開発者支援システム

外部装置の仮想化を進めるためには，仮想化外部装置の使用者の支援だけではなく，仮想化外部装置を開発するための支援システムも必要である．この様な支援システムでは，仮想化する外部装置の機能とインターフェースを分析するとともに，すでに定義されている近い機能を持つ仮想化外部装置との関係を調べ，標準化に適し，プログラムの新規の記述が少なくてすむようなクラス階層，機能，インターフェースを効率よく定義するための機能が望まれる．また，実際の仮想化外部装置の定義から，シミュレータの基本部分を自動的に生成する機能も必要であると考えられる．

6．まとめ

FAソフトウェアの生産性と品質を向上させる上でボトルネックとなっている外部装置関連のプログラミングに着目し，外部装置を仮想化することによって問題を解決する手法を提案した．また，仮想化外部装置に要求される機能を分析し，それらの要求をオブジェクト指向の概念に基づき実現する手法と，仮想化外部装置の概念を発展させ効果的に利用するための将来的な展望を示した．

現在，オブジェクト指向と仮想化外部装置による開発，保守などの業務の有効性や問題点を明らかにすると共に，技術者のオブジェクト指向技術の修得と移行の問題点を明らかにするために，本報告で提案した仮想化外部装置の概念を適用した自動倉庫のシミュレーションシステムを開発中である．その評価を基に，さらに研究を進めたいと考えている．

参考文献

[1] 砲崎賢一 他: FA ソフトウェアにおける外部装置の仮想化手法，精密工学会，FA ソフトウェア開発環境専門委員会，1991年度活動報告書

[2] B. Stroustrup: The C++ Programming Language, 2nd Ed., Addison-Wesley, (1991).