分散制御指向 FA 開発アーキテクチャ

日本工営パワー・システムズ株式会社
事業開発部 ZIPC・DVC 開発セクション
石田　哲史

1．市場の概況

近年の FA 機器市場では、不況とデフレの煽りを受けて各企業とも業績の下方修正を強いられ、設備投資の大幅な削減がなされているが、その一方で設備の老朽化やリストラによる要員削減を補填するための設備自動化、段取り替え作業の簡略化を図る新しい製造ラインの構築、高機能ロボットの導入などは暫時進められている。

一方、エンドユーザからは費用対効果を最大限にするため、高機能化・短納期・高信頼（メンテンスフリー）という要求仕様を突きつけられ、要求に対する十分な性能を確保できないなどといった課題を数多く抱えている。

情報機器はムーアの法則に従って常に進化をし続けており、新技術出現のたびにそれまで障壁となっていた数々の技術的課題を容易に解決できるようになってきている。現在、FA 機器の世界においても製品競争力を確保するため、積極的に新技術を採用せざるを得ない状況にあるにもかかわらず、機器メーカ間の開発の足並みがまったく揃わず、機器毎に特化したプログラム記述が多くなっている。

2．FA 機器開発環境の現状

(1) 開発環境の課題

FA 機器開発に関して最も標準的な構成部品は PLC（シーケンサ）になる。

PLC はハードウエアの持つ信頼性、リレー回路設計をベースとした開発手法（ラダープログラミング）による容易な制御回路開発が特徴の機器として定着しており、最近では高性能な通信能力や豊富なツール群を持つ PC（パソコン）と組み合わせた FA 機器も市場をにぎわせるようになってきている。

技術の高度化に伴い、PLC の得意とする DI/O・AD・DA を中心とした入出力制御機能だけでは顧客満足を満たせず、HMI（ヒューマンインターフェイス）を中心とした監視・操作性の向上、複数の FA 機器をネットワークで結んだ高度な連係制御機能、パソコンを中心とした情報処理機器との連係などが要求されるようになっている。

これら新技術はリレー制御を元にした PLC の基本コンセプトとはまったく異なるため、汎用 PLC にこれらの機能を搭載することは極めて難しく、各メーカともハードウエアに性能

改良を加えながら、独自に新技術の対応を行っている。

その結果、PLCの標準性が奪われ、新技術（特殊ユニット・拡張基板）を利用するための方策として、PLCメーカが各社固有のラダープログラミングを採用するようになったため、開発者はメーカ固有のプログラミング手法を習得したり、開発環境の再整備を強いられるようになっていった（**図1**）。

また、ラダープログラムの標準性が奪われることにより、過去の知的資産を流用し難い状況も同時に生み出している。

こうした変遷の中で開発者は次第にPLCをPCの拡張基板の一部として捉えるようになり、情報処理の中心をPCに移行し始めている。これにより開発作業はラダープログラム以外のプログラミング技法（VC・VB・JAVA）やWindows開発技術、データベース開発技術を習得することが必要になった。

図1　ユーザーと技術者の悩み

(2) 機能増大における生産性と試験量の増大

FA 機器の高機能化に伴い、構成する PLC 数や IO 点数の増大、パソコン計装など監視制御機能の拡充に比例して開発コード（ステップ数）が増大している。

さらに、開発コードはラダー言語・SFC 言語・VC・VB・JAVA など多様な言語からシステムが構成されるため、機能増加によるステップ数の増大や、各プログラミング言語間のインターフェイスをとるためのコードなど主機能には関係のない部分の開発も無視できない作業量となってきている。

一方、試験フェーズにおいては、増大するステップ数によって膨大な確認項目数となっている。試験においては多くの装置と連係するケースが多いことから、試験条件において入力タイミングを厳密に管理するなどの試験手順や試験環境構築が複雑となってきており、現在の開発環境下における生産は限界に近づいてきていると言える。

表―１は電力監視・制御装置における試験時の課題をユーザ（電力会社）と装置開発メーカの双方にてアンケートを採った結果である。この結果からも生産環境における限界が読みとれる。

表１：電力監視・制御装置の試験における課題

	課　　題
ユーザーの視点	・機能規模が大きく試験環境を構築することができない。
	・複雑な障害に対して再現することができない。
	・性能試験や限界試験などが充分に実施されない。
	・過去の不具合の事例などが教訓として活用できていない。
	・複数装置間の自動連係機能などについては現地調整となり試験ができていない。
	・ユーザーサイドの視点での試験が充分にされていない。
メーカーの視点	・試験進捗が要求仕様の信頼性に対してどの程度消化できているかが把握できない。
	・プログラミング規約など基本事項の実施確認が取れない。
	・試験シミュレータなどの開発が膨大。
	・品質指標（バグ発生率）の数値が正しく採取できない。
	・試験中におけるプログラムの構成管理が難しい。
	・試験項目が膨大で項目が洗い出しきれない。
	・試験工程が充分に確保できない。

(3) 信頼性の低下

PLC は FA 現場において、ミッションクリティカルな機能を司ることが多く、高信頼性が絶対条件となる。

FA 機器の情報処理の中心が PLC から PC に移行するにつれ、システム全体の信頼性が PC の影響を受けることが多くなってきた。PC は汎用機器であるため、ハードウエアプラットホーム・OS ともに幅広い機能において拡張性が確保されている。その反面、マルチベンダ環境においては信頼性を保証することが難しい状況にある。

また、PC は DISK や FAN などの回転部品が多く、信頼性確保のためにはクラスタ技術など高価で複雑な構成が必要となる。

一方、FA 現場は環境条件の厳しい大型機械加工場から空調の整った精密加工場まで利用される環境は幅広く、耐環境性に優れ、安定した動作が保証される PLC 技術は必要不可欠な状況にある。

(4) 要員配置と要員教育の課題

開発手法（言語や方法論）やハードウエアの技術進化により、多様な開発方法が選択できるようになっている。

これらの技術を幅広く習得し、活用するためには相応の知識と経験を必要とするため、実際の作業においては開発言語やプラットホーム別に担当を分けているケースが多いが、昨今では短納期・低コスト化・リストラなどにより、開発に必要な要員を確保することも難しくなってきている。

さらに、短納期や技術革新の早さも加わり、要員教育においても十分な研修期間や実地経験を積ませることが難しく、その結果、人的ミスによる不具合増加を助長している状況に陥っている。

(5) 知的資産活用とソリューション技術向上の課題

技術革新により生産性の改善は進んではいるものの、ソフトウエアやハードウエア技術の変遷が激しいため、過去に開発したプログラムなどを流用するためには最新環境に合わせた改良が必要となり、流用開発においては移植作業量が多くなっている。

また、上記(4)項による状況から、新しい開発手法を実践の場において習得することが多く、開発の道具であるはずの手法の習得に追われ、本来開発者が実施しなければならない監視のための知的センシング技術や高度な連係制御技術などの手法が進化していない。

(6) メンテナンス性の課題

ＰＬＣとＰＣが混在するシステム構成により、製品のライフサイクルが短いＰＣ部品の確保やＯＳも含めたソフトウエアの構成管理が複雑化している。

また、開発言語の多様化や開発ステップ数の増大に伴い、プログラムメンテンス時のトレーサビリティが低下している。

3．ＦＡ開発方法論の見直し

(1) 状態遷移表設計手法の適用

PLC を用いた FA 機器においては定周期にプログラムが繰り返し実行されるシーケンシャル制御を行っている。

PLC 用ラダープログラムはシーケンシャル制御を前提にしたプログラム手法で、はしご段に記載された処理を先頭からはしご段を降りるように順次実行し、処理が終了次第、再び先頭に戻る”繰り返し実行方式”となっている。

処理ステップ数が大きくなると各処理の起動時間や装置間での通信タイミングが変化し、設計や試験が難しいうえ、処理中に参照するメモリ状態が変化してしまい、メモリ参照タイミングによって実行結果が異なってしまうといった課題がある。

こうしたことから、設計時にタイミングが明確となる設計手法が望まれるようになってきている。タイミング設計においてはモデル記述型手法としてアローダイアグラムやシーケンス図・状態遷移図などの方法があるが、処理タイミングの抽出や処理の漏れ抜けを精査することが難しい。

また、条件記述型手法として、フローチャート・HCP チャートや状態遷移表設計手法などがあげられるが、最も処理タイミングが扱いやすい手法として状態遷移表を用いた設計が考えられる。

さらに後述するクリーンルーム開発に欠かせない CASE ツールとしては「ZIPC」が存在する。

近年 IEC61131-3 で標準化された設計手法が FA 開発においては主流となっているが、FA のみを対象としているため、汎用ソフトウエアとの親和性が低く、オブジェクト指向設計などの生産性の高い設計手法への移行が難しい。

一方、状態遷移表は FA という機器に特化するのではなく、リアルタイムというプログラムの動作条件・機能目的に特化し、オブジェクト指向開発におけるビヘイビアモデルと同一の概念を持つなど次世代に適用しやすい手法と考え、状態遷移表設計手法を採用した。

(2) CASE ツールによるクリーンルーム開発

クリーンルーム開発はオブジェクト指向デザインパターン開発に代表される。特に UML を用いた CASE ツールはパターンジェネレータを具備することにより、モデル作成段階からモデルに適合しない設計を排除する方法を採っている。さらに、最近ではモデル検証を実施しながらのスパイラル開発によって短納期も可能としている。

一方、FA 機器開発においてはラダー用ツールを代表としてツール類がビジュアル化され、十分な操作性を備えているが、ラダー開発では設計の検証が試験実施後に判明するため、試験結果に課題があった際の設計への手戻りが大きい。そ

のため、設計ツールは設計検証も含めたツールが望ましく、プロトタイプ開発手法やスパイラル開発手法が容易な、プログラム完全自動生成型の CASE 環境が望ましい。

(3) ハードウエアオリエンテッドの設計からソフトウエアオリエンテッドの設計への変更

FA 機器開発は、プログラム開発というよりは電気設計に近い設計技術が必要であり、PLC についても同様の開発方法を踏襲し、ハードウエアユニット（DI/O、AD、DA）の処理を対象としていた。そのためハードウエアをどう操作し、どう利用できるかが重要となり、ラダープログラム等の PLC 用言語は言語機能拡張においてハード依存を強めることで記述の簡略化を追求した。

その結果、PLC プログラムは機種依存性が強くなり移植性が失われた。

また、プログラムの巨大化に伴い、ハードウエアを意識したプログラムではアセンブラと同様、生産性が期待できないなどの課題もある。

従来の PLC を用いた装置は、運用期間 5 年～10 年程度が一般的であったため、移植性を考慮することはそれほど重要ではなかった。

しかし最近ではコスト・信頼性・メンテンス性においてソフトウエアの資産化が重要な製品戦略となってきており、デザインパターン開発手法（ソフトウエアの部品化）も FA 機器開発で注目されつつあり、IEC61499-1 で FB を用いた定義化も進められている。

以上によりソフトウエアの独立性を高めインディペンデント・プラットホームな開発環境は、巨大化する FA プログラミングの世界でも必須と考える。

(4) 自律型ネットワークによる機能の能動構成

情報処理技術においては、CORBA・DCOM・JINI などに代表されるとおり、機能をネットワークで能動的に接続し、各機能がソフトウエアの要請に対してサービスを提供する分散コンピューティングが脚光を浴びている。

これは企業内基幹系システムにおいて、巨大サーバが処理（サービス）の全てを提供することの限界から、より身軽なシステムへの移行が始まり、シン・クライアントを含む 3 層構成によるクライアント／サーバ方式が主流となってきたことからも明らかである。オブジェクトベースの分散コンピューティングは機器の処理能力を有効に活用し、メンテンス性・信頼性・セキュリティ面で有効な手法であることが数々の導入事例で立証されてきている。

このことから、FA 機器においてもネットワークベースで機能を再構成できる具体的な手法が必要である。

(5) 資産活用を考慮した設計の階層化

ナレッジマネジメント（KM）は数年前より各企業で取り組む重要な企業戦略となっている。

ソフトウエアについても知的資産化を重要戦略としている企業が多いが、ソフトウエア内には知的資産部分（他の開発でも流用可能な部分）と機種や機器に依存した部分が共存することが多かったため、資産管理が行いにくかった。

しかしながら、デザインパターン開発においてはインディペンデント・プラットホームを指向しているために、デザインパターンは他への流用が可能な資産として期待が掛かっている。

また、デザインパターンはパターンジェネレータと併用することにより、デザインパターン設計とプログラムコードが完全一致するため、デザインパターンのみの管理でプログラム管理も可能となる。

このような設計レベルの階層化によって、管理すべき知識部分のみ管理していくことが今後の知識管理上重要である。

4. DVC fai の指向するフュージョン・アーキテクチャ

(1) クリーンルーム開発を可能とする、FA 開発用－CASE Tool “ZIPC for DVC”

DVC ではFA機器の将来を考慮のうえ、設計手法に状態遷移表を採用することで、設計時における状態遷移表記述、状態遷移表設計からのプログラムコードの自動生成、検証時における状態遷移制御処理確認といった機能について、組み込み開発用 CASE ツールである“ZIPC”をベースにCASE化を実現した。（注1）

さらに、状態遷移表に記述する処理(プログラム）に関しては機器（リレー）を制御する制御仕様の記述を可能とし、制御仕様記述言語（特許出願）をもとにプログラムコードの自動生成を実現した。

これにより全てのプログラムコードを仕様記述（モデル）から自動生成することでクリーンルーム開発を可能とした。図2にCASEツールでの作成例を示す。

また、制御仕様記述言語で記述した制御仕様は、C 言語ソースとしてコード生成されるため、DVC アーキテクチャ以外のシステム環境への移植も容易とする特徴を持つ。

(2) ソフトウエアの独立性を高めるネットワークベースのアーキテクチャ“DVC-Ware”

図3の通り DVC のアーキテクチャはネットワークをベースとして DVC-WareがOSやハードウエアの違いによる機種依存からソフトウエアの独立性を確保し、FL-net（OPCN-2）（注2）の持つネットワーク再構成機能により、システム動作中の再構成を可能とする。

さらに、プラットホームに依存しない DVC アーキテクチャによって、設計したプログラムは設計仕様からプログラムコードを自動生成する際に、ターゲット

環境を指定するだけで Windows 上で動作するコードを生成し、設計仕様や生成されたコードに手を加えることなく動作を可能とする。

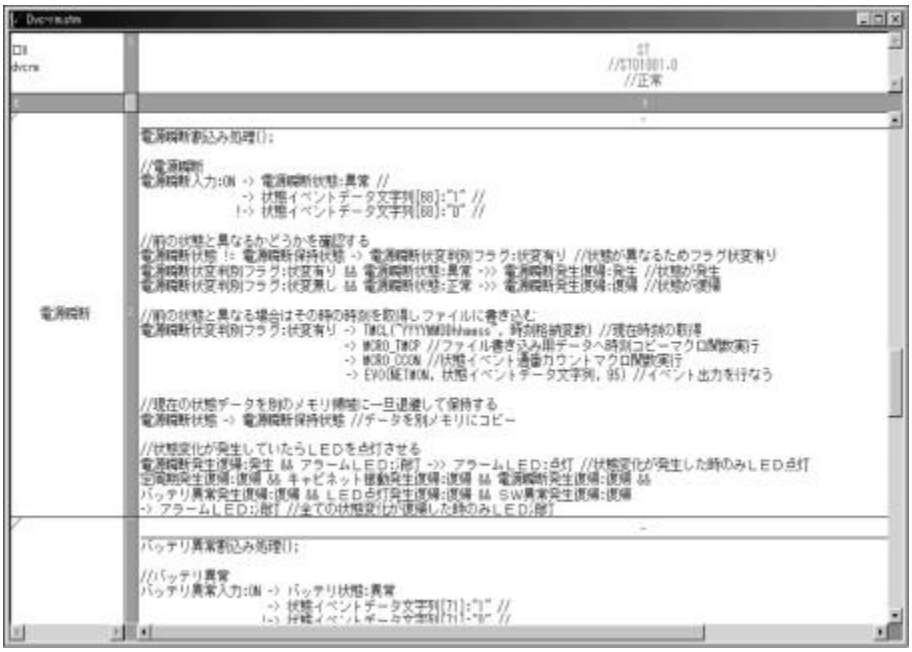

図２　制御仕様記述設計

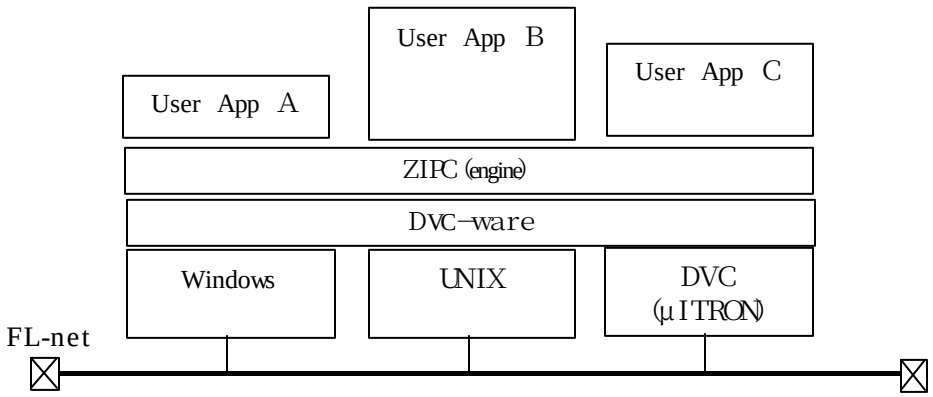

図３　システムアーキテクチャ

(3) ブロードバンド時代に対応した機能を装備

ブロードバンド時代の到来によって通信性能が飛躍的に向上してきている。DVC はブロードバンドを十分に生かす機能とハードユニットを装備し、付加価値の高いシステムの構築基盤を提供する。（図7）

以下にネットワークに対応した機能について述べる。

- DVC に DB クライアント機能を具備し、SQL（注3）言語を利用した制御機構の開発が可能。
- 広い通信帯域を利用し、画像ストリーミング機能や画像解析用データ生成機能を具備。
- ブラウザからの装置運転状態監視や、装置異常時における E メール通報機能など、モバイルやインターネットを活用した FA 装置の開発が可能。（注4）
- 複数のフィールドネットワークと連係することにより多彩なシステム構築を可能とするゲートウエイ機能を具備。（対象ネットワーク：Ethernet、FL-net、Device-net）

(4) CASE と設計データベースを連動した、資産活用機能を具備

DVC 開発は ZIPC を利用したビジュアルな開発環境を提供する。さらに ZIPC に登録した状態遷移表と制御仕様記述から生成したプログラムコードは、再利用とプロトタイプ開発を効果的に実現するため、設計データを全て DB に保管する機能を設けた。

DB に保管することで、過去の類似設計の参照や再利用部品の検索、メンテナンス対象部品の検索などの支援を可能とし、設計情報と知的資産管理の連係を可能とした。

(5) OS にμITRON を採用し、マイコン開発環境による拡張開発が可能

DVC-Ware を動作させるプラットホームには国産 OS のμITRON を採用し、国内の ITRON ベースプログラムの活用を可能とした。

また、DVC アーキテクチャによって設計され、自動生成されたソースコードはμITRON 上で動作可能なコードであることから、DVC 以外の ITRON ベースの機器への移植も容易とした。

DVC はμITRON のインターフェースを提供し（注5）、DVC アーキテクチャによる自動生成コード以外にユーザが作成した高度な機能を DVC に組み込むことを可能とする。ユーザの開発した高度な機能は、デバッガ（注6）を利用することで、マイコン基板同様の開発および試験を実施可能とした。

(6) 信頼性高いハードウエア

1) PLC 同等の信頼性を具備

FAN レス・空調レス・DISK レスを実現し、電源規格に「電気協同研究 48-3、

電力規格B-402」に対応可能な電力制御用特殊電源を具備した。

2) ユニットのインテリジェント化による自律分散の実現

各ユニットはそれぞれがCPUを搭載することで分散処理を、また冗長構成機能を具備することで冗長に必要なユニットを増設するだけで冗長構成を可能とした（特許出願）。

冗長機能は 図４に示す通り、FL-netと構成制御信号によって、論理部のみならずIO入出力までの冗長を可能とし、２重化やN重化を自由に構成可能とした。

なお、冗長機能によってノンストップでプログラム変更・ハードウエア点検・装置交換を可能とした。

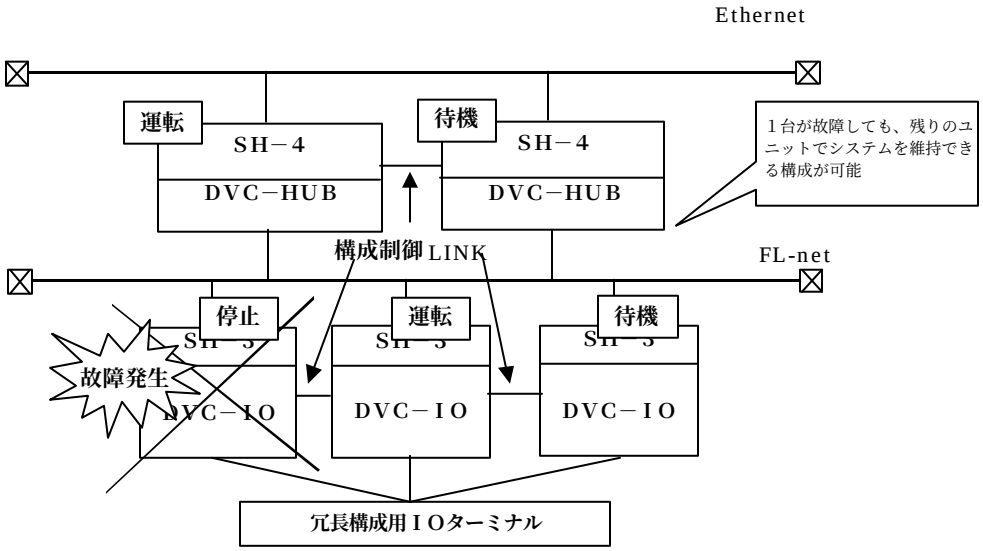

図４　冗長構成機構

(7) JAVA を用いたSCADA 開発ツールを具備

DVC は Ethernet 上にある Windows サーバとソケット通信を行う機能を持ち、Windows サーバ上にツールを組み込むことでリアルタイムにグラフィカル表示する SCADA 機能を具備する。さらに、Windows サーバに DB を構築することにより EJB に対応したソリューションの構築も可能とした。（図５）

リアルタイム監視は JAVA ソケット通信によってサーバ（サーバ側 JAVA）からクライアント（クライアントアプレット）へ情報更新トリガを与えることで、情報更新のリアルタイム化を実現した。

また、ブラウザに表示する画面は画面表示内容をデータ化することで、JAVA アプレットがデータを読み込んで画面制御を行う方式とし、画面データは CAD データ（DXF）を直接 JAVA アプレットが処理することで、CAD による画面作成を可能とした。（図６）

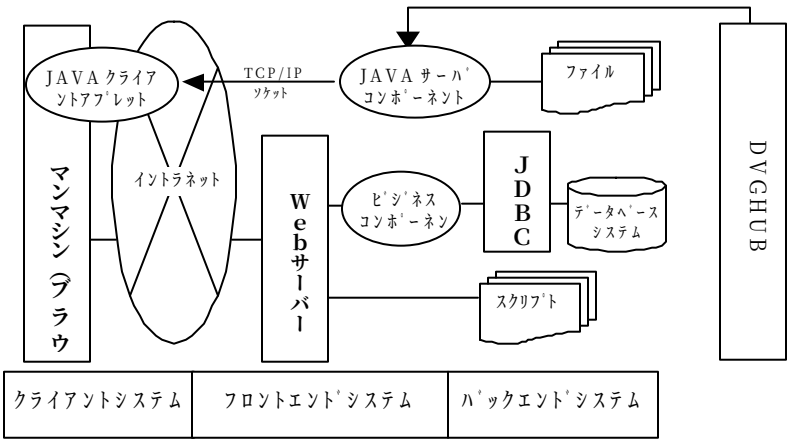

図５　３階層アーキテクチャ

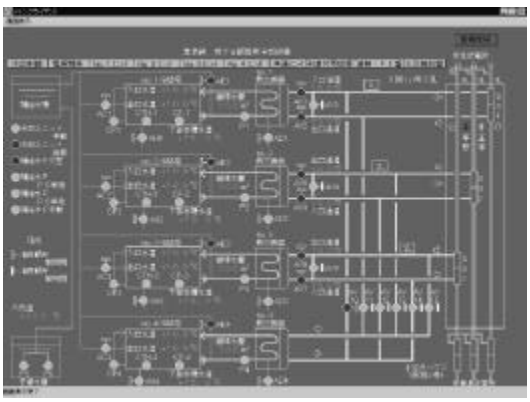

図６　ＪＡＶＡによる設備監視画面表示

5．効果

現在効果についてのデータを採取中であるが、現時点において以下の結果を得ている。

(1) DVC アーキテクチャの生産性

1) 制御仕様記述言語の生産性

DI/O を中心とする制御処理を C 言語で作成した場合、処理に必要なメモリを割り付け作業、接点入出力のためのドライバーIO 処理記述などハードウエアとソフトウエア構成を意識した開発を必要とする。

しかしながら、制御仕様記述言語は、ラダー記述と同様の論理構成が日本語で記述でき、メモリ割付・開放・クリアなどのガベージコレクション機能を実装し、メモリの不正操作の防止や記述量の削減を可能とした。

また、メモリ操作が不要となったことにより、C 言語記述と比較してコーディング量が 30%～40%減少し、約 1.5 倍の生産性向上が得られた。制御仕様記述言語は設計仕様記述と同様に利用できるため、仕様作成・設計・製作作業を同時に進行することが可能となった。

2) CASE による設計・製作の合理化

状態遷移表設計により、処理タイミングを意識した動作設計が可能となった。

また、プログラム生成前に遷移表動作を ZIPC 上で確認できることから設計ミスを防止することができた。

製作においては、ZIPC が持つチェッカとジェネレータにてハード資源のアサイン違反や制御仕様記述規約違反などを自動検査し、動作可能なプログラムのみを生成するため、ヒューマンエラーによるケアレスミスを設計時点で排除できた。

さらにプログラムの動作試験中は、TAP 機能を利用することによって DVC 上の遷移状態を ZIPC ツール画面上に表示させることができ、タイミング試験による遷移表のカバレッジが把握可能となる。

また、LAN で接続した開発環境において、ブラウザを使用して FL ネットの状態・DI/O・AD・DA の状態をモニタしながら製作・試験を進めることができる。

(2) DVC アーキテクチャの信頼性

1) プログラム構造

メモリの設定・クリア、ドライバー連係処理など、システムに重大な影響を及ぼす処理を完全に隠蔽しているため、PLC と同様に信頼性の高いプログラムが可能となる。

また、ZIPC ジェネレータで設計コメント付きの C 言語ソースプログラムを生成するため、癖の少ないソースコードができ、保守時においてトレーサビリティの向上が図れる。

2) システム構成

図 7 のように必要な IO ユニットをネットワークに接続するだけで容易にシステムを構成できる。

また、ネットワークで容易にシステム

の再構成が可能となるため、電源を停止することなく点検の際に予備ユニットを接続し、運転を切り替えることでシステム停止を不要とすることができる。プログラム入れ替え時についても前記同様に予備ユニットを用いて停止レスでプログラム交換が可能であるため、メンテナンス性が向上した。

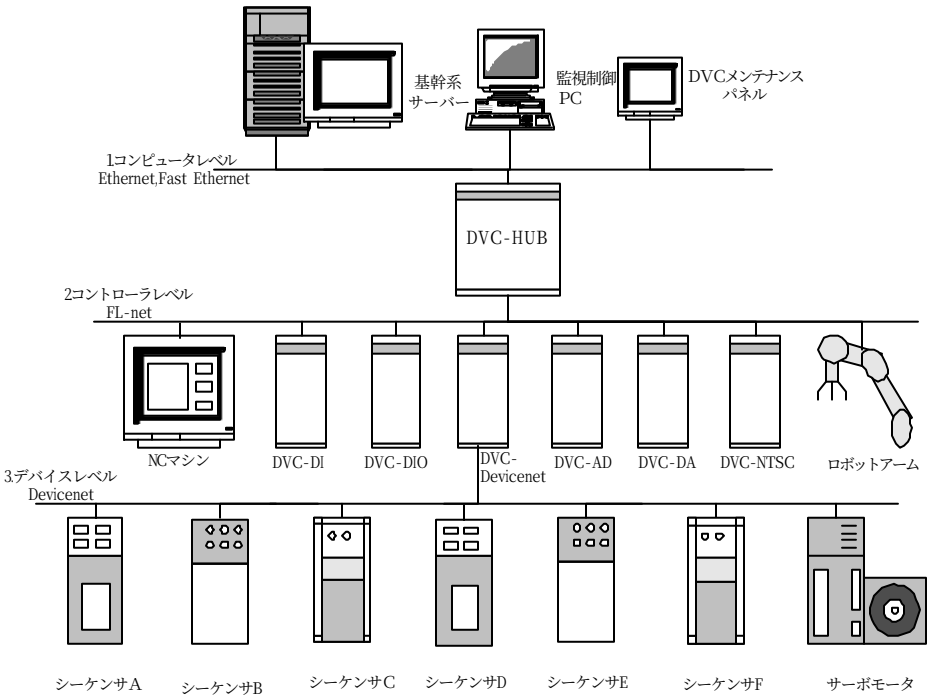

図７　製品ユニット群とシステム構成

6．課題

単純な DI/O ユニットもネットワーク機能をベースに動作するため、OS によるプログラム制御が必要となることから、OS レスまたは簡易 OS 程度でプログラムを動作させている従来型 PLC に比較して接点入出力が遅延する。そのため、各ユニット間で協調した分散処理が実施可能なマルチプロセッサプログラム動作方式を導入する必要があると考えられる。

プログラム面の課題としては、ラダー言語がモデル型言語であるのに対し、制御仕様記述型言語が記述型言語であるため、視覚的（直感的）な面でのプログラム動作把握が行いにくい面がある。

また、開発環境面としては、ZIPC から C ソースを生成し、オブジェクトに変換後ターゲットにダウンロードするという一連の作業時間が PLC に比較し複雑で時間を要している。

7．将来展望

近年、組み込みシステムの開発においてはオブジェクト指向設計がより進化したデザインパターン設計に移行しつつあることは既に述べた。ソフトウエアの部品化は生産性と信頼性の両面から求められている。特にパターンジェネレーターによるコード生成はかなり充実した設計支援環境となってきている。

デザインパターンはビジネス系（事務処理系）処理ではかなり一般化した設計手法となっているが、FA 系についてはパターンをどう作るかが現在研究段階にある。FA 設計においてはシーケンス図を多く用いるが、シーケンス図はオブジェクト指向設計でも取り入れられ、リアルタイム制御が必要な FA 系への適用も準備されている。

しかしながら、FA 系はソフトウエア実装がハードウエア毎に異なることから UML に対応したツールなどの適用が難しく遅れていた。DVC アーキテクチャはネットワークを利用したアーキテクチャであるため、実装部を切り離すことでオブジェクト指向へ対応できることから、オブジェクト指向のモデル化・抽象化が行いやすい UML 手法によるオブジェクト指向開発を導入していきたいと考える。

8．参考文献

1) 電気学会技術報告　2000 年 5 月　第 781 号　シーケンス制御技術の動向　　電気学会
2) 電気協同研究　1999 年 1 月　第 54 巻　第 4 号　集中制御所システム機能検証試験の効率化と品質管理　電気協同研究会
3) IEC61131-3　ハンドブック　PLCopen　JAPAN
4) 日本電機工業会ホームページ　http://www.jema-net.or.jp/Japanese/hyojun/hyojun.htm
5) 欠陥ゼロのソフトウエア開発　日経 BP
6) 計測技術　2000年6月　373号　特

集:多様なニーズに対応する PLC 最前線　日本工業出版
7) オートメーション　2001 年 8 月 46 号　特集:フィールドネットワーク技術の現状を探る　日刊工業出版プロダクション
8) オートメーション　2002 年 2 月 47 号　特集:生産・情報システム技術の新展開　日刊工業出版プロダクション
9) ソフトウエアの未来　FTP 編　翔泳社
10) Real-time UML:Developing Efficient Objects for Embedded Systems Bruce Powel Douglass
11) Executable UML: A Foundation for Model-Driven Architecture Stephen Mellor , Marc Balcer
12) 計測と制御　2002 年 2 月 41 号　特集:組み込みシステムの開発環境　計測自動制御学会
13) リアルタイム環境への CASE 導入　渡辺政彦・石田哲史・戴志堅著　電子開発学園出版局
14) 日本工営技術情報　NO21.2001　地中線設備情報監視装置の開発　石田哲史・荒川隆敦・田邊秀治・渡邊與志郎

注1) ZIPC はキャッツ(株)の組み込み開発支援ツールの名称。

注2) FL-net とは日本電機工業会が規格化した、FA 向けのネットワークプロトコルである。

注3) select 文においては、メモリ資源や通信性能の制約により記述に制限をかけ実装を可能とした。

注4) JAVA を利用した HMI 構築については、DVC 以外に PC サーバーを必要とする。

注5) μITRON の API は eSOL(株)の PrkernelV4 を提供。

注6) 高機能開発用デバッガーとして eSOL(株)の eBinder を装備。