Javaによる制御用ソフトウェアの開発

JDevice for Writing Control Software in Java

竹内　陽一郎　TAKEUCHI Yoichiro
上遠野　浩昭　KATONO Hiroaki

制御用アプリケーション ソフトウェアをJava(注1)言語で記述するのがデバイス制御用Java(JDevice)である。これにより情報系から現場系までのユニットのソフトウェアをJavaで構築することができ，システム構築を容易にし保守性の向上を実現できる。

LonWorks(注2)による分散制御ネットワークのモデルをオブジェクト指向言語であるJavaで実装することにより，制御対象をJavaBeansの形でコンポーネント化し，ビジュアル プログラミングを実現するなど，制御システムにおけるソフトウェア開発環境を改善できる。

JDevice is designed for device control to write control application software in Java. JDevice enables application software to be built in Java for both information systems and control systems, and provides easy configuration and high maintainability for these systems. By implementing the programming model of LonWorks network in Java, the object-oriented feature of Java language allows an abstraction of control objects in the form of JavaBeans and realizes visual programming. This improves the software development environment for the control systems.

1 まえがき

Javaは，実行環境に依存しないプログラム実行方式として急速に普及し，事実上の業界標準になりつつある。さらにJavaによる分散処理技術を発展させ，コンピュータ機器のインテリジェントな接続を可能にするソフトウェア技術としてJini(注3)が提案されている。SunMicrosystems社は，Javaの適用範囲を拡張すべく，Personal Java, Embedded Java, Java Cardなどの仕様を定義した。しかし，これらは制御システムにJiniのような分散コンピューティングの枠組みを提供するまでには至っていない。

当社とEchelon社は，Echelon社のLonWorks技術を核に，制御システムネットワーク上でのJava言語による分散処理フレームワークを実現するJDeviceの開発を進めてきた。LonWorksは，オブジェクト指向に基づく分散制御システムの枠組みをすでに実現している。しかし，LonWorksは，プログラミング言語としてEchelon社独自の仕様であるNeuron(注4) Cをベースとし，またハードウェア プラットフォームは，Echelon社のNeuron Chip(注5)だけを対象としている。JDeviceのねらいは，LonWorksの枠組みをJavaベースに拡張することで，インターネットにまでシームレスにつながる，分散制御システムの枠組みを実現することにある。

(注1)，(注3) Javaならびにその他のJavaを含む商標およびJiniは，米国SunMicrosystems社の商標。
(注2)，(注4)，(注6) LonWorks，Neuron，LonTalkは，Echelon社の商標。
(注5) Neuron Chipは，Echelon社がライセンスをもち，当社で製造しているLSI。

2 LonWorksのプログラミングモデル

LonWorksのプログラミングモデル概要を図1に示す。

LonWorksのシステムは，制御ネットワークと，その上に配置された分散制御ノード(ネットワークに接続する中継点)から構成される。各ノード内のプログラムは，タスクと呼ばれるプログラミングの最小単位から構成され，イベ

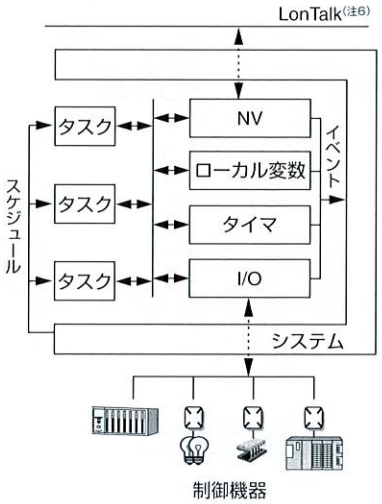

図1．LonWorksプログラミングモデル　各オブジェクトの状態変化に伴いイベントが発生し，タスクがスケジューリングされる。
LonWorks programming model

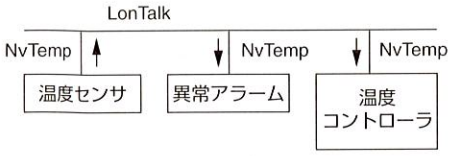

図2．NVの機能　温度センサノードの計測したデータがNV(この場合はNvTemp)により他のノードへ伝わる。
Effect of Network Variables

ントに結び付けられる。ローカル変数，ネットワーク変数(NV：Network Variable)などの状態変数は，制御に必要な種々の状態(例えば，温度や電圧など)を表現する。LonWorksの標準化団体であるLonMark(注7)では，これらの変数の標準的なデータ型(ビット表現とその意味付け)をSNVT(Standard NV Type)として定めている。

NVは，LonWorksの分散制御の枠組みを実現するためのキーオブジェクトである。NVの状態値が更新されると，関連付けられた他のノードへ自動的に伝播(ぱ)され，イベントとして認識される。このメカニズムによって，各ノードは互いに情報およびタイミングを共有し連携して動作することができる。NVの関連付けは，システムインストール時に，プログラムとは独立した管理ツールによって行われる。したがって，プログラムを変更することなく，ノード単位のシステムの再構成を行うことができる。

NVの実例を図2に示す。図2の温度センサノードのプログラムは，温度を測定し，変化があればNV(この場合はNvTemp)上の温度値を更新する。NvTempが更新されると，これを参照するアラームノードおよび温度コントロールノード上ではイベントが発生し，関連するタスクが起動される。各ノードは，NvTempというNVによって，物理的に離れた場所にあっても連携して動作することができる。さらに，LonWorksにおける温度の表現方法は，SNVTの一つとして標準化されており，これに従えば，各ノードを独立に設計された別のノードに置き換えても，システムの動作は保証される。

LonWorksのプログラミングモデルは，抽象化されたオブジェクトの概念によって組み立てられている。しかし，実際にプログラムを記述するNeuron C言語はオブジェクト指向言語ではなく，LonWorksの概念とは大きな乖(かい)離がある。このギャップが大きな問題点である。

3 Javaへのマッピング

Javaはオブジェクト指向言語であるため，これをベースとすることでJDeviceは，より直接的にLonWorksの概念を表現することができる。

(注7) LonMarkは，Echelon社の商標。

まず，イベントおよびタスクは，JDeviceでは，Javaのオブジェクト(クラス)として表現される。イベントおよびタスクを表わす基本クラスに，アプリケーションプログラム(以下，アプリケーションと略記)ごとに必要なイベントの定義やタスクの処理をメソッドとして追加して，必要なクラスを定義していけばよい。イベントとタスクとの関連付け機能，およびスケジューリング機能は基本クラス内に隠蔽(ぺい)される。

I/O(入出力)やタイマなどシステム機能に対応したオブジェクトは，対応するJavaのクラスを定めこれに対応させる。これらのオブジェクトは，JDeviceのアプリケーションからみると，対応するシステム機能が隠蔽された実行時システムとのインタフェースとして動作する。

状態変数も，Javaのクラスとして，より本来の概念に近い形で表現できる。しかし，この場合，SNVTなど，既存のLonMark標準とのバイナリ互換性を維持する必要がある。また同じクラスを，NVや，コンフィギュレーションプロパティ(ネットワーク管理ツールによって管理されるノード上の初期化パラメータ)など，複数の異なる記憶クラスに適用できるようにすることも必要である。JDeviceでは，これを実現するための枠組みとしEnhancedVariableを導入した。

EnhancedVariableは図3に示すように，4層のクラス拡張と，システムポートと呼ばれるシステムとのインタフェースを担うオブジェクトから構成される。最下層では，バイト配列と，この内部を基本的なデータ型でアクセスするためのアクセスメソッドが用意されている。次の層では，実装するデータ構造に固有のアクセスメソッドが定義できる。例えば，LonMarkのSNVTなどのデータ型はここで定義される。第3層では，各アクセスメソッドに対して副作用としてのI/Oとのインタフェースを記述できる。最上層

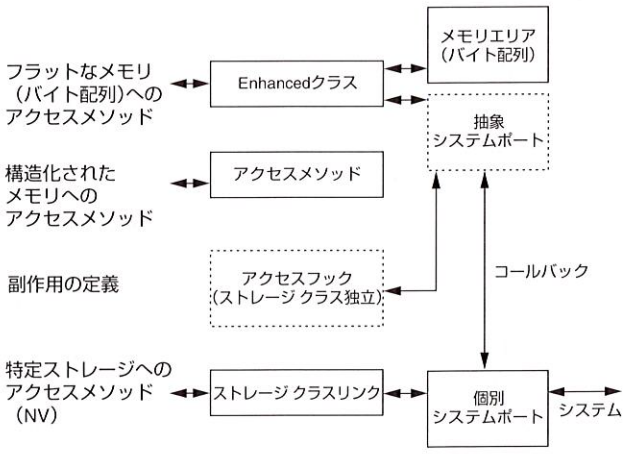

図3．EnhancedVariableの構造　4層のクラス拡張と，システムポートから構成される。
Scheme of EnhancedVariable

で，具体的な副作用の実体をシステムポートのインスタンスとして指定する。

4 JDeviceの実装

JDeviceを実行するJava Virtual Machine（Java VM：Java仮想マシン）の実装としては，以下の2種のケースを想定している。

(1) SunMicrosystems社標準のフルJava VM上に実装し，I/Oなどのライブラリはリアルタイム基本ソフトウェア（OS）上のネイティブドライバを利用する。

(2) ネイティブコードにコンパイル(注8)　RTOS（Real Time Operating System）またはファームウェア上で直接実行する。Neuron Chipはこのケースである。

JDeviceは，ネイティブコードへのコンパイルを前提に，仕様自体が最適化を考慮して設計されている。例えば，EnhancedVariableは，つねにアクセスメソッドを介して値がアクセスされるが，コンパイル時にメソッド呼出しが容易にインライン化できるしくみになっている。また，Neuron Chip上では実現が難しいガーベッジコレクション(注9)がなくても実装できるように，実行時に生成されたオブジェクトの参照範囲はタスク内に限定している。さらに，8ビットのNeuron Chipへのコード生成では，不必要な32ビット演算を8/16ビットに換える最適化が行えるよう仕様上考慮されている。

5 ビジュアル プログラミング

JDeviceは，JavaBeansとこれを用いたビジュアル プログラミングを想定している。このようすを図4に示す。

プログラミングの最小単位は，センサ，電子的制御信号を物理的行動に変換する機構であるアクチュエータなど，制御対象を抽象化したオブジェクトを実装したJavaBeansとなる。これらのオブジェクトのインタフェースおよび行動は，LONMARKにおいて，各アプリケーション分野ごとのプロファイル（属性）として標準化されている。したがって，一度プログラミングすれば，属性値の変更だけで容易に再利用できる。

各ノードのプログラミングは，これらの個々のソフトウェア コンポーネントであるBeansのライブラリから，必要なBeansを選び出し，対象となるノード上に配置し，属性値の指定を行えば完了する。これらを，グラフィカルユーザーインタフェース（GUI）ツールで行うことによって，制御アプリケーションのビジュアル プログラミング化が可能になる。

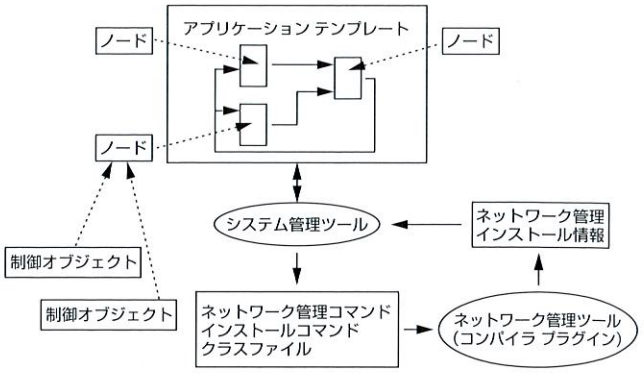

図4．ビジュアルツールによるシステム開発　GUIツールは，ネットワーク管理ツールと連携してプログラムのインストールおよび実行の監視を行う。
System development with visual tool

生成された各ノード用のプログラムはBeans形式で保存される。これによって，システム運用時の管理ツールも同様のしくみで実現できる。すなわち，特定のシステム全体にかかわる環境を指定するテンプレート上に，必要なBeansを選択し，初期化時のパラメータを属性として指定してこの上に配置すれば，システム構築が完了する。

6 あとがき

JDeviceのENC（Enhanced Network Computing）アーキテクチャ全体のなかでの役割は，分散制御ネットワークとインターネットとのシームレスな接続を実現するうえでの統一的なデータ表現およびアプリケーション動作の記述方法を提供することにある。これは，分散制御システムにおいて必要な機能を集約したLONWORKS抽象モデルとJavaでの実装により実現可能になった。このことは，アプリケーションのモジュール独立性と再利用性を飛躍的に高め，制御システムにおけるソフトウェア開発およびシステム運用コストを画期的に改善するものと期待される。JDeviceは，制御対象を抽象化し相互運用性を進めるLONMARK標準をそのまま組み込んでいる。当社とEchelon社は，JDeviceを新たな業界標準として推進する準備を進めている。

(注8) プログラミング言語で作成されたプログラムを，コンピュータが理解できる機械語に変換すること。
(注9) 使用されていないオブジェクトを終了させ，メモリ領域を整理して再度プログラムから利用できる状態にすること。

竹内　陽一郎　TAKEUCHI Yoichiro
東芝アメリカ社。
Java技術標準化活動に従事。
Toshiba America,Inc.

上遠野　浩昭　KATONO Hiroaki
情報・社会システム社 産業・電機・計装システム事業部 産業システム部主務。
LON応用制御システムの開発・設計に従事。
Industrial Systems Div.