擬似マイクロカーネルに基づく成長型ロボットソフトウェアアーキテクチャ

Growing Robot Software Architecture based on Microkernel Emulation

○正 杉原知道（九大）

Tomomichi Sugihara, Kyushu University, zhidao@ieee.org

A highly extensible software architecture which emulates microkernel is proposed. The kernel function modules are implemented as shared objects to keep data visibility, while they behave as independent units to achieve encapsulation. It is valid in situations where the initial knowledge about operations and users is insufficient.

Key Words: Software architecture, Microkernel emulation, Growing system, Data visibility, Encapsulation

1. はじめに

人型ロボットやレスキューロボットなど，運用場面や環境，使用者に関する情報が設計時に十分得られないロボットの開発は，次世代ロボティクスにおけるチャレンジである．開発初期にシステムの全体像を固定せず，逐次的に機能を高めながら利用範囲を拡大していく手法を，金広ら [1] は発展的実現と呼んだ．これに最も重要なのは，ハードウェアの変更，開発者の変更，基盤ソフトウェアの変更などにも頑健に耐える拡張性である．再利用性，保守性の高さともほぼ同義である．このための基本技術が部品化，カプセル化であることは，従来より指摘されてきた [2][3][4].

一方，不確かさを多く含む実世界に開かれ，物理法則の支配を受けながら振る舞うロボットのシステムは，単純な階層構造や有限状態機械の形態を必ずしもとらない [5][6]．高次知能に相当するプログラムが，末端にあるセンサやモータの情報を直接参照するなどの必要が頻繁に生じ得る．このため重要なのはデータの可視性であり，カプセル化技術とは一般に相反する性質である．したがって，形態としてはカプセル化に則りつつも，全体としてモノリシックなシステムを形成し，システム内部でデータに自由にアクセスできる手段を残すことが望ましい．

本稿では，個々の機能ソフトウェアモジュール（デーモン）を共有オブジェクトとして実装するアーキテクチャを提案する．各デーモンはリクエスト・レスポンス用ポート，および初期化・終了処理部を独立に持ち，マイクロカーネルに似た高い拡張性を実現する．それらを単一のプロセスに結合しメモリ空間を一にすることで，デーモン間のデータ可視性を確保する．この意味で，提案するアーキテクチャを擬似マイクロカーネルと呼ぶ．

2. 共有オブジェクトによる擬似マイクロカーネル

2.1 システムの全体構成

提案するアーキテクチャの構成を図 1 に示す．システムは，ハブプロセスとデーモン群から成るカーネル，カーネル外部から通信するクライアント群，それら全てを対話的に起動・終了させるシェルから成る．このうち，シェルは標準的な OS で用意されているものを利用できる．以下，それぞれについて説明する．

カーネル　ロボット身体と密接に関連し，自律的な振る舞いを決定するソフトウェア．実時間処理が中核となる．

デーモン　カーネルの機能単位となるソフトウェアモジュール．共有オブジェクト（実行時リンクオブジェクトまたはプラグインオブジェクト）として実装される．各々が別個に初期化ルーチン，終了ルーチン，クライアントとの通信経路，必要に応じて実時間処理を行うスレッドを持つ．通信の実現形態は特に限定せず，名前付きパイプ，共有メモリ，ソケットなどから用途に合わせて選ぶ．実装上，全てのデーモンはフラットな関係にある．一方これらは，機能上は部分的階層関係をとり得る．その階層構造はコーディングの段階で決定する．これについては後述する．

ハブプロセス　デーモン群をメモリ空間で結びつけるソフトウェアモジュール．main 関数を持つ．起動直後に各デーモンを生成・結合した後，終了シグナルを待つ（実質的に何もしない）．

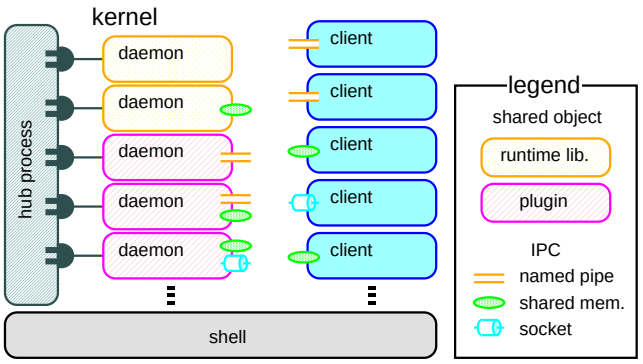

Fig.1 Microkernel emulation system consisting of shared objects

クライアント　特定のデーモンに対しリクエストを発行する，あるいはデーモンの持つ情報を参照するための外部プロセス．

2.2 共有オブジェクトによるカーネル実装

各デーモンは，他のデーモンにデータまたはメソッドを公開する．これらは，他のデーモン全てに共用されるデータまたはメソッドを公開するものと，特定のデーモンにメソッドのみを公開するものの二種類に分類される．前者は，実行時リンクオブジェクトとして実装される．また後者は，実行後にリンクするいわゆるプラグインオブジェクトとして実装され，次のような共通のプロトタイプを持つただ一つのメソッドを開示する．

```
void *exec(void *dat, void *util, void *ret);
```

両者は単にリンクのタイミングが異なるだけで，内部的には同じものであるが，利用方法が異なる．実行時リンクオブジェクトは，デーモンクラスに共通なエラー処理やシンボル解決関数の定義，ロボット状態の記憶などに適している．プラグインオブジェクトは，それが有するデータやメソッドのシンボルを隠蔽でき，複数のプラグインが同一のシンボルを持っていても衝突が起こらない．このため機能単位のカプセル化に適する．

標準的な共有オブジェクト開発ライブラリは，オブジェクトの生成・消滅時に自動実行される初期化・終了ルーチンの定義方法（たとえば GNU C では関数属性 `constructor`, `destructor`）を用意している．これらを用いれば，各オブジェクトを独立したモジュールとして扱い，システムの拡張性を高くすることが可能である．クライアントからはあたかも個々のデーモンが独立にサービスを提供しているように見える．

2.3 データフローの管理

複数の独立モジュールが同一のデータやハードウェアにアクセスしながら処理を行うシステムでは，モジュール間の処理内容に

矛盾が生じたり，データの処理順序に乱数的要因が生じたりする危険性を排除しなければならない．従来のアーキテクチャでは，データフロー管理部分を全てのモジュールの外部に置くスーパーバイザリシステムをとることが多かった [4][7]．この構成は，モジュール依存関係に関する情報の分散や，モジュールが処理できるデータ型の拘束などにより，拡張性の低下を招きがちである．

提案システムでは，デーモン群の依存関係を管理するスーパバイザを設けず，次のようなモジュール設計規約を定めることによってデータフローの矛盾を回避することとした．

1. ある（プラグイン型の）デーモンを下位機能として利用できるデーモンは，一つだけとする（フロー分岐の防止）．
2. デーモン間の上位・下位関係が逆転するフローは作らない（フロー循環の防止）．
3. 実時間スレッドの数は必要最低限とし，同期をとる必要があるものは初めから一つに統合する．（乱数的フローの防止）
4. 上記項目を満たす範囲で，一つ一つのデーモンはなるべく低機能に作る．（冗長フローの防止）

システム全体の整合性を，モジュールの処理内容に依らず保証する仕組みを作ることはできない．結局，プログラマが個々のモジュールから中身を吟味して開発していくことと，その際に上記のような規約を定めることが重要である．

3. 人型ロボット mighty における実装

提案アーキテクチャに基づいて開発した人型ロボット mighty[8] 制御システムの概要を図 2 に示す．これは Linux（カーネル 2.6）の上に作られており，シェルには標準的な tcsh を利用している．以下に，各デーモンとそれらが提供する機能を記す．角括弧内はそれぞれが依存するデーモンである．

mky40d　センサ・モータの入出力制御．[based]
motord　関節サーボのためのモータ制御．
sensed　センサ信号の処理と物理量への変換．
based　センソリモータ系をなす実時間スレッドの定義．[motord, sensed, ctrld]
ctrld　全身制御のための運動計算．[ikd, seqd]
ikd　逆運動学，分解重心速度制御．
seqd　運動軌道の記憶と管理．[navid]
navid　操縦システム [9] とのソケット通信．
agentd　ロボットの力学モデルによる状態の記憶と予測．
daemond　デーモンクラスで共通利用する関数群の定義．

小型な mighty システムでは，演算能力の高いプロセッサをロボットに搭載するのが難しいため，体外に置いたホスト PC とオンボディ PC とをピアツーピア接続し，イーサネットを構成している．ホスト PC 上では，メインシステムを稼動させている．オンボディ PC 上では，デーモンの一つである mky40d をリモート起動し，メインシステムのデーモン based とソケット通信させている．mky40d は，CUnet による体内ネットワーク上の分散プロセッサ群を 3[ms] 周期で制御する．based はスレッドを一つ持ち，mky40d に牽引される形でソフトリアルタイムに動作する．

矢印で示したプロセス間通信とメソッド呼び出しによるデータフローに注目されたい．クライアントも含め，フローの出発点は多くあるが，それらは最終的に based，さらに最下流にある mky40d へと収斂する．同時に，図中に陽には描かれていないが，センサ情報や運動学，逆動力学によって獲得されるロボットの状態は agentd の機能によって全てのデーモンに共有されている．

4. おわりに

各々のモジュールが，共有オブジェクトとしてメモリ空間で繋がりながらも，独立したデーモンとして動く擬似マイクロカーネルアーキテクチャを提案した．これにより，高い拡張性とデータ可視性が両立される．実世界に開かれた成長型ロボットシステムのアーキテクチャとして有効であると考えている．

本研究は，次世代研究スーパースター養成プログラム（九州大学総長裁量経費）の支援を受けた．

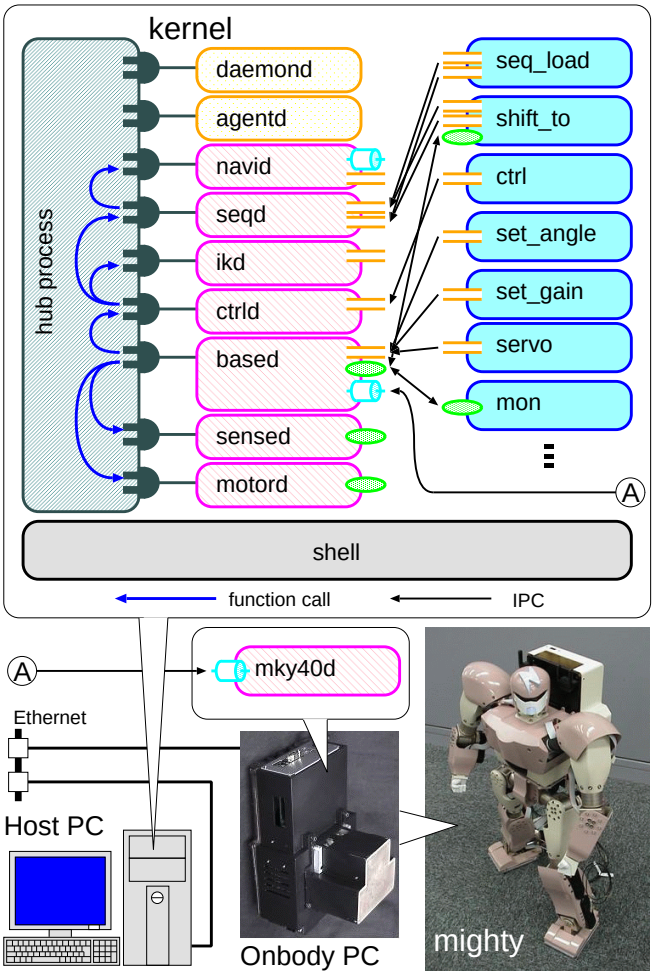

Fig.2　An implementation of the proposed system on the anthropomorphic robot mighty

文献

[1] 金広ら．人間型ロボットの発展的実現のためのソフトウェアシステム構成法．コンピュータソフトウェア, Vol. 17, No. 6, pp. 52–56, 2000.

[2] I. A. D. Nesnas et al. Toward Developing Reusable Software Components for Robotic Applications. In *Proc. of the 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2375–2383, 2001.

[3] A. C. Domínguez-Brito et al. Integrating Robotics Software. In *Proceedings of the 2004 IEEE Int. Conf. on Robotics & Automation*, pp. 3423–3428, 2004.

[4] 猪平ら．順次処理における動的再構成機能を備えた実時間制御ソフトウェア．日本ロボット学会誌, Vol. 22, No. 8, pp. 1021–1030, 2004.

[5] R. A. Brooks. Elephants Don't Play Chess. *Robotics and Autonomous Systems*, Vol. 6, pp. 3–15, 1990.

[6] 吉海．並列監視評価構造を備えたヒューマノイドの自律行動統合システムの構成法．博士論文, 東京大学大学院工学系研究科, 2004.

[7] 金広ら．HRP-2 の運動制御システム – プラグイン群とその実行制御 –．第 21 回日本ロボット学会学術講演会予稿集, 2003.

[8] T. Sugihara et al. Hardware design of high performance miniature anthropomorphic robots. *Robotics and Autonomous System*, Vol. 56, pp. 82–94, 2007.

[9] 小林ら．手動・自動操縦部位の非中断切替可能なヒューマノイドロボットの操縦システム．日本機械学会ロボティクス・メカトロニクス'08 講演会予稿集, 2008.