知 能 と 複 雑 系　126－8
（２００１．１０．１２）

異分野エキスパート集団によるインタラクティブロボットの共同開発のための基本ソフトウェア

松坂 要佐　　小林 哲則

早稲田大学 理工学部 電気電子情報工学科
〒 169-8555　東京都新宿区大久保 3-4-1

03-5286-3379
{yosuke,koba}@tk.elec.waseda.ac.jp

インタラクティブロボットの共同開発を可能とする開発環境を設計・実装した。汎用ロボットなどの大規模統合システムの開発においては、複数の分野にまたがる専門知識が必要とされるため、単一の開発者が全体システムを見通すことは困難となる。そのため、異なる専門を持った複数の開発者たちの共同作業が不可欠となる。本稿においては、インタラクティブロボットの共同開発形態に適したシステムアーキテクチャについて検討し、そのアーキテクチャを実現する基礎となる基本ソフトウェア、開発者の活動を支援するツールを準備した。音声・音響・画像情報を統合的に利用した対話ロボットの実装を通して、高密度に統合されたシステムが短期間で開発できることを確認した。

キーワード: 統合システム開発, コンピュータヒューマンインタラクション, 黒板モデル, publish/subscribe モデル

System Software for the Cooperative Development of Interactive Robots by Experts with Their Own Specialties

Yosuke Matsusaka　　*Tetsunori Kobayashi*

Department of Electrical, Electronics & Computer Engineering
Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan

+81-3-5286-3379
{yosuke,koba}@tk.elec.waseda.ac.jp

We designed an environment that enables the collaborative development of interactive robot. In order to realize the large scale integrated systems, collaborations of many engineers with different specialties are unavoidable. Since the detailed knowledge from many fields is required to make them, it is difficult for a single developer to maintain the whole system. In this paper, we investigate the system architecture which suits for the collaborative development model, design the system software which gives the base of the architecture, and prepare the tools which support the proper activities of developers. We also evaluate the efficiency of the proposed environment through the experience in implementing the multimodal conversation robot.

Keywords: integrated system development, computer human interaction, blackboard model, publish/subscribe model

1　はじめに

近年、人間の生活空間で活動するロボット (コミュニケーション・介護・補助ロボットなど) に対するニーズが高まっている。人間の生活空間で生起する様々な事象に対処しながら活動するためにはさまざまな機能を持つ必要があり、かつその機能間の強い結合が必要となる。これらの要求にこたえるロボットは、膨大な機能を持った統合システムとなる。

多機能を持つシステムを構築するためには、さまざまな分野の知識が必要とされ、到底一人の力では構築しきることができない。システムの開発は、それぞれのモジュールを担当する複数の開発者同士の共同作業となり、共同作業を円滑化する枠組みが必要となる。

これまでロボットに代表されるインタラクティブシステムの開発においては、開発がトップダウン的に行われることが主流であった。すなわち、システムデザイナによる、動作環境・タスクの分析に基づいたシステム設計プロセスの完了の後に必要な機能の実装が行われていた。しかしながら、日常の雑多な作業をこなすロボットのシステム開発形態としては、タスク・機能を各々のモジュール開発者が各々の問題解決手段に基づいて分析・実装し、それら実装された機能を必要に応じて追加していけることが望ましい。これらの開発形態の差異をオープンソースモデルにおける議論 [2] に習って前者を「伽藍方式」、後者を「バザール方式」の開発モデルと呼ぶことにする。

バザール方式のシステム開発を考えた場合、各々の開発者によって開発された機能が全体の中で統合されるためには、開発者が全体のシステムの中から自らが利用可能な情報を選び出し、選び出した情報に自由にアクセスできる枠組みが必要である。また、機能の付加を考えた場合、すでに動作しているシステムに機能を追加していくわけであるから、付加した機能が、既存の機能の動作を妨げずに、かつ既存のシステムの中で密に統合されることを可能にするシステムアーキテクチャを検討する必要がある。

本稿では、ロボットを対象とした大規模統合システムの開発にあたって、バザール的なシステム開発を可能とするシステムアーキテクチャと基本ソフトウェア、開発者同士のインテグレーション活動を支援する開発環境について検討を行ったのち、多人数の開発者によるマルチモーダル対話ロボットの実装を通して検討したシステム開発環境の有効性を確認する。

2　システム開発モデルの長所・短所

システム開発環境に関する議論をはじめる前に、各々の開発モデルの持つ特質について概観する。

a. 伽藍方式

明示的なシステムデザイナ、システムデザインプロセスを経た後、具体的な実装が行われる開発モデルである。

設計・開発の流れは以下のようにして進む。

1）システムデザイナによるタスクの仕様の検討
2）システム全体のデザイン
3）モジュールへの分割・インターフェースの定義
4）各モジュールの実装
5）接続試験
6）仕様の検証・モジュールの調整

長所:システム設計の段階からシステムの正しい動作を検証できる。各モジュールの実装段階においては、各々の開発者のするべき仕事、満たすべき仕様が明らかになっているため、開発が進めやすい。システムをタスクにあわせて最適化できるため無駄がない。

短所:システム全体に精通した明示的なシステムデザイナが必要。システム全体がシステムデザイナの定義した特定の問題解決手法の制約を受けて開発が進められるため、大規模・複雑なシステムを考えた場合、部分部分でその解決法が最適でない可能性がある。一度システム全体が完成したら、それ以上の機能の付加や仕様の変更に対応するのは難しい。

b. バザール方式

モジュール開発者が自らの担当するモジュールに関して自由な開発を行い、他のモジュール開発者との共同作業の中でシステム全体を作り上げていく開発モデルと定義する。

長所:明示的なシステムデザイナが不要。システムデザイナが把握不可能な多岐の分野にわたって高度な専門知識が必要な大規模システムの開発が可能となる。各分野の専門知識を持つ開発者がそれぞれ最適な問題解決手法によってモジュールを実装することができる。システム全体が完成した後も、継続的に機能の追加や改善を行っていくことができる。

短所:統制を取る枠組みがないと開発を進めることができない。一部開発者のルール違反によって全体のパフォーマンスが崩される危険性がある。システム全体の見通しを立ててから開発するわけではないため、系統だった動作確認ができない。各々が各々で開発を進めるため、開発者同士のコミュニケーションがうまくいかなかった場合、冗長な部分が多くなる可能性がある。

3　システム開発環境の要件

3.1　基本構成・基本的デザイニングプロセス

システムを構成する部分要素のことをモジュールと呼ぶ。モジュールを開発する主体のことをモジュール開発者と呼ぶ。

モジュール開発者が、その領域の情報を理解し、利用することができるだけの知識を持つ範囲のことを SoI (Scope of Interest) と呼ぶ。モジュール開発者が、その分野に対して深い知識を持ち、モジュー

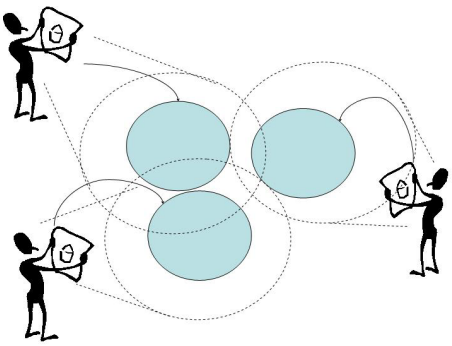

図 1: 基本的デザイニングプロセス

ルを開発することで新たな情報を提供できる範囲のことを AoP (Area of Profession) と呼ぶ。その領域を理解するだけの知識を持たず、直接的には利用することができない範囲のことを OoF (Outside one's Field) と呼ぶ。モジュール開発者が自分の AoP の範囲の情報を他のモジュール開発者に対して参照可能な形で公開することをコミットと呼ぶ。自らが AoP とする情報をその情報を SoI とする他のモジュール開発者からの要請によりコミットすることをコラボレーションと呼ぶ。

システムの中心には、モジュール開発者の誰もが自由にアクセスできる共有エリアを設定する。これを黒板と呼ぶ。

基本的なデザイニングプロセスは以下のようになる。

1）黒板を見る
黒板を見に行くとシステム上で公開されているすべての情報が一覧できる。

2）必要な情報を選定する
黒板上の自分の SoI に該当する範囲の中からモジュールの構成に必要な情報を選定する。

3）情報を生成するモジュールを作成する
自分の AoP の範囲で、選定した情報や、新たに取り込んだセンサ入力から新たな情報を生成するモジュールを作成する。

4）モジュールを動作させる実行環境を用意する
モジュールを動作させるのに足りる実行環境 (プロセッサ) を用意する。

5）システムにプラグイン・情報を配信する
モジュールをプロセッサとともにシステムにプラグインする。モジュールで生成した新たな情報をコミットする。

6）全体の動作を見て調整する
モジュールのシステム全体の中での振る舞いを検証し、問題のある部分があった場合は修正する。

3.2 バザール的なシステム開発実現の要件

バザール的に統合システムを開発することを考えた場合、開発をスムーズに進めるためには、開発者を統制する枠組み (以下、開発の原則) が必要となる。幾つかの観点から、守るべき開発の原則を以下のように定めた。

第一に既存のシステムの動作を保証する観点から、

- 付加したモジュールは既存のモジュールに影響を与えてはならない。(不可侵の原則)

第二に後から付加されるモジュールへの配慮から、

- モジュールの生成する情報は、後から付加されるモジュールを含むどのモジュールに対しても公開されていなければならない。(公開の原則)

さらに、各々のモジュールの実装に依存するデータ構造のやり取りによるデータのブラックボックス化を防ぐ観点から、

- 公開される情報は、前提知識を必要とせずに解釈できるものでなければならない。(可解釈の原則)

その他に、ソフトウェアのソースコードのバージョン管理、並行開発などの従来からの開発論が必要とされる。

3.3 インタラクティブシステム実現の要件

インタラクティブシステム実現の要件として、自律性と応答性の 2 つが挙げられる。自律性は、システムの自発的な動作を表し、ユーザの指示を受けない状況下での活動や、ユーザに対する自発的な働きかけを実現するのに重要である。応答性は、システムを取り囲む状況の変化や、ユーザからの指示などに対して即時に応答を返せる能力のことを指す。インタラクティブシステムの開発環境について考えた場合、この自律性と応答性の 2 つの要素を容易に実現できる枠組みを提供する必要がある。

自律性を実装するためには、ステート遷移によりシステムの行動が制御される枠組み (以下、ステート駆動) が有効であることが知られている。応答性を実装するためには、イベント発生によりシステムの行動が制御される枠組み (以下、イベント駆動) が有効である。さらに、自律性・応答性両者に関わってくる要因として、レイテンシ、スループット、ジッタなどのシステムの処理時間に対する制約 (以下、リアルタイム性) を保証することが重要となる。

3.4 本研究における目標

上記した章の中で、バザール的なシステム開発環境の要件、インタラクティブシステム開発の要件について検討を行なったが、両者の要件を鑑みた上で、本研究において検討を行なう開発環境は、以下の条件を満たすことを目標とした。

明示的なシステムデザイナは不在とする。モジュールの開発者は、自分の製作するモジュールとその周辺のみの知識を持ち、システムが提供する環境の下、ほかのモジュール開発者とともにシステムを組み上げていく。

インタラクティブシステムの開発に適した枠組みを提供する。システムの自律性や応答性を容易に実現可能にする。作成するモジュールは処理の粒度において、ロボット制御から対話制御まで、幅広いスケーラビリティを持ち、それらがシームレスに統合される枠組みを持つ。

付加的開発が容易である。モジュールの変更や付加が、システム全体の性能に影響を及ぼさないようなアーキテクチャを定義し、そのアーキテクチャをできるだけ理想的に実現するシステムソフトウェアを提供する。

これらの条件を満たす枠組みとして、ステート駆動、イベント駆動の実装を容易にし、公開の原則を順守する情報公開モデル (4.1 章で詳述) と、システムのリアルタイム性の保証を容易にし、不可侵の原則を順守するアーキテクチャ (4.2 章で詳述) からなる開発環境を構築しユーザに提供した。

4　システムアーキテクチャの検討

4.1　情報公開・メッセージ通知モデル

各々のモジュールで生成された情報は、システム内のどのモジュールからも参照できなければならない。このような情報共有を実現する枠組みとして集中・開示型の情報公開モデルである黒板モデルを採用した。黒板モデルでは、各々のモジュールが共有の情報の開示場所に情報を「タグ」に書き込むことで、どのモジュールからもそれらの情報を参照することが可能となる。黒板モデルはデータ開示モデルとしては優秀なものであるが、データの変化を検知する用途には非効率である。そこで、メッセージ通知に適した枠組みである publish/subscribe モデルをあわせて採用した。publish/subscribe モデルでは、興味のあるデータを購読 (subscribe) リストに入れておけば、そのデータに対する変化の通知 (publish) を受けることができる。情報公開サーバは、黒板モデルに基づく情報開示サービスを提供するサーバと、publish/subscribe モデルに基づくメッセージ通知サービスを提供するサーバとの組合わせにより構成される (図 2)。

モジュール開発者は、情報開示サービスを利用して、自分にとって必要な情報を選定し、注目すべき情報はメッセージ通知サービスを利用して購読、新たに生成した情報を情報開示サービスを通してコミットする。同じデータに対するアクセス手段として、情報取得とメッセージ配信が等価的に選択できるため、データを状態としてアクセスできると同時にイベントとしても取得できる。この枠組みによって、

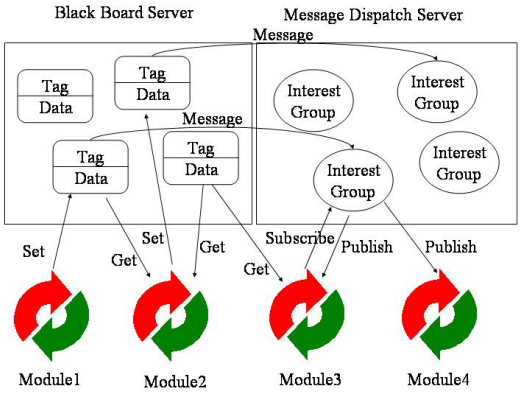

図 2: 情報公開・メッセージ通知モデル

インタラクティブシステム開発の要件となる自律性と応答性を等価的に扱える環境でシステムを開発することが可能となる。

4.2　リアルタイム性の確保

レスポンス性が高く、十分な転送容量を持つシステムバスと、マルチプロセッサの構成があれば、拡張性に長けるリアルタイムシステムが構成できる。システムに新たな機能を付加したいと思ったときには、機能とともに、その機能をリアルタイムに実行するに足りるプロセッシングパワーを付加すればよい。機能とプロセッシングパワーを組にして付加することで、機能の付加が既存のシステムのリアルタイム性能に与える影響は最小となる。

また、マルチプロセッサの構成は、大量の音声・画像情報を扱うマルチモーダルシステムとの親和性が高い。未処理のセンサデータは膨大なものであるが、センサから取り込んだデータを直接システムバスに流すのではなく、得られた膨大なデータから必要な情報を抽出するプロセッサをシステムバスに接続する前段階におくことで、システム全体のリアルタイム性の低下の原因となるバスへの負担 (バスの汚れ) を最小限に抑えることができる。

上記検討のもと、情報公開サーバと中心として複数のプロセッサをプラグインする形でシステムを組む構成を取った。

4.3　タグの命名規則・グループ購読

本システムにおいては、ソフトウェアのソースコードのバージョン管理、並行開発などの為に、UNIX の標準的なバージョン管理システムである CVS の機能を利用した。一般に CVS ツリーの構造はモジュール間の関連性の距離によって決定される。システムインテグレーションを考えると、モジュール間の関連性が高いほど頻繁にデータを交換する必要性がある。データの「タグ」の命名規則を CVS ツリーの構

図 3: システム全図

造に習ったものにした上で、正規表現によるタグのグループ指定での購読サービスを提供することで、ツリーのノード単位のメッセージ購読が可能とした。これら機能の提供により、メッセージ通知の頻度をモジュール間の距離に対応させることができる枠組みを提供した。

4.4 処理の粒度の異なるモジュールの統合

各々のモジュール間のメッセージ送受信は、非同期に行われる。情報開示サーバからのメッセージは、まずモジュール側に用意されたバッファへ格納される。メッセージの受信に自由度のあるバッファを採用することで、処理の粒度の異なるモジュール間の通信が実現できる。たとえば、上位の粒度のモジュールが下位の粒度のモジュールが出力する情報を参照する場合は、送られてくるメッセージを一時的にバッファに溜め込んでおいて自らの処理の周期にあわせてまとめて処理すればよい。また、下位の粒度のモジュールが上位の粒度のモジュールが出力する情報を参照する場合は、バッファをポーリングしてメッセージが入っていないときには、自らの処理を継続することができる。

5 提案アーキテクチャの実装

5.1 システム構成

上記検討に基づき、ロボットと情報処理 PC からなる実験システムを構築した (図 3)。

ロボットは、頭部に 2 台のカメラ・2 台のマイク・2 本のアームを持ち、移動台車による自立移動が可能である。このロボットに、CVS サーバと情報公開サーバを中心に 7 台のクライアント PC を接続した。モジュール作成者は、これら 7 台のクライアント PC 上におのおのの作成したモジュールを動作させる。

5.2 サーバプログラム

サーバプログラムは 2 つのプロセスからなる。ひとつは情報開示サービスを提供するものであり、ひとつはメッセージ配信サービスを提供するものである。

黒板サービスを提供するサーバは、クライアントからの黒板への書き込み・読み出しの要求に対して応答を返す。黒板上のどのデータに書き込む [読み出す] かは、ユーザ指定可能の文字列 (タグ) により区別される。各々の書き込み・読み出し要求には、書き込み先 [読み込み元] を指定するタグと、それに引き続いて書き込むべきデータが送信される。また、書き込み・読み出し要求以外に、現在黒板に書き込まれているタグの名前の種類を取得する (モニタする) ための API を用意した。モニタ要求が入力されると、指定された数字 ID から、その番号をインデックスとするタグ文字列を返す。

メッセージ配信サービスを提供するサーバは、クライアントからの購読要求・メッセージ配信要求を受け付け、各々のクライアントにメッセージを配信する。購読要求は正規表現による指定を使ったグループ単位での購読が可能である。メッセージの配信は、UDP を利用したパケットのブロードキャストが有効であるが、パケットの到着の確実性を保証できないことから、本システムでは TCP による配信を複数回行うという形で実装を行った。

5.3 モジュールの基本構成

各々のモジュールは、上記したサーバに接続されたクライアントであり、ソケット通信によってサーバにアクセスする。メッセージ配信サーバからのメッセージは、まずクライアント側に用意されたバッファへ格納される。クライアントプログラムは、メッセージがバッファに届いているかいないかをブロッキングなしで確認する (ポーリングする) か、メッセージが届くまで指定時間ブロッキングするかを引数として指定できる。また、UNIX のプロセス管理の枠組みを利用して、下位の作業を担当する子プロセスを生成した上で、親プロセスがプロッキング読み出しでメッセージ受信を検出し、子プロセスにシグナルを送ることで、受信割り込みを実現することができる。

高級開発言語とのインターフェースとして、C++、PERL、JAVA 用のクライアントライブラリを用意した。各々のライブラリはクラスライブラリとして定義され、言語による API の差はほとんどない。

5.4 開発ツール

開発者用のツールとして、コマンドライン動作するツールをいくつか提供した。黒板に書き込まれているタグとデータのペアをリアルタイムにモニタするツール、黒板の内容を標準出力にダンプ [アンダンプ] するコマンド、購読したメッセージへの通知を標準出力に出力するコマンドなどがあり、標準出力のパイプやリダイレクトなどを使って、UNIX の標準コマンドと組み合わせることで、動作のログをとって分析したり、状況の再現をしたりできる。

6　システムの稼動状況

これまでのシステムの稼動期間において、ロボットに、音声・音響・画像・言語処理を行いながら多人数の話者との会話に参加するシステム [9] や、状況に応じた表情生成を行いながらユーザとの対話を行うシステム [10]、ロボットアームを通した指示表現と音声による言語表現を用いてユーザと空間的情報を共有する対話システム [11]、インターネット経由でのロボットの遠隔操作システム [12] などの多数のシステム [13] を実装する機会を得た。

開発者となったのは、音声画像処理・パターン認識・人工知能・機械制御などを専門とする学生たちであり、どのシステムも平均して 8 人ほどの開発者がシステム開発・統合作業に関わり、現在 3 年を超えようとしている稼動期間を通して 15 人ほどの開発者がシステムにかかわってきた。

開発者たちは、明示的なシステム設計者が存在しない状況で、開発者同士の情報交換によりシステムを組み上げていった。各々のモジュールの統合作業の開始から、システムが満足のいく動作をするまでにかかる時間は、長い場合で 1ヶ月ほどであり、特に既存のシステムの拡張や手直しによって対応できる課題である場合は、1〜2 週間の短期間でシステムを構築することができた。

多人数の話者との会話に参加するシステムを例としてあげると、開発の結果、100 種を超えるタグが定義された。このタスクは、音声・音響・画像情報などの下層の処理を統合的に利用しながらも、ユーザとの会話のための言語処理などの高度な処理を同時に行なうという要件を持っている。最終的なモジュール間の参照関係から、ユーザの追跡のための画像処理とモータ制御の間の高速なフィードバックループが形成される一方で、音声・画像認識など中粒度の処理、処理結果を統合する非同期動作のモジュール、対話を担当する大粒度のモジュールなどが形成され、統合されたシステムを形作っていることが見て取れた。

7　まとめと今後の課題

開発者のシステムインテグレーション活動において、バザール的なシステム開発を可能とするシステムアーキテクチャと、その活動を支援する開発環境について検討を行った。多人数の開発者によるマルチモーダル対話ロボットの実装を通して検討したシステム開発環境が良好に動作することを確認した。

参考文献

[1] 橋本周司, 成田誠之助, 白井克彦, 小林哲則, 高西淳夫, 菅野重樹, 笠原博徳, “ヒューマノイド –人間型高度情報処理ロボット–,” 情報処理, Vol.38, No.11, pp.959-969, 1997.

[2] E.S.Raymond, *The Cathedral & the Bazaar,* O’Reilly & Associates Inc., 1999.

[3] L.D.Erman, F.Hayes-Roth, V.R.Lesser, D.R.Reddy, “The Hersey-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty,” ACM Computing Surveys 12(2), pp.213-253, 1980.

[4] D.Goddeau, E.Brill, J.Glass, C.Pao, M.Phillips, J.Polifroni, S.Seneff, V.Zue “GALAXY: A Human-Language Interface to On-Line Travel Information,” Proc. ICSLP’94, pp. 707-710, 1994.

[5] T.Matsui, H.Asoh, I.Hara, N.Otsu, “An event driven architecture for controlling behaviors of the office conversant mobile robot, Jijo-2,” Proc. IEEE-ICRA’97, pp.3367-3371, 1997.

[6] M.Inaba, “Remote-Brained Robotics: Interfacing AI with Real World Behaviors”, in T.Kanade, R.Paul (Ed.), *Robotics Research,* vol.6, pp.335-344, The International Foundation of Robotics Research, 1994.

[7] 矢向高弘, 菅原智義, 安西祐一郎, “μ-PULSER: パーソナルロボットを構築するためのオペレーティングシステム”, 信学論, vol.J77-D-I, no.2, pp.207-214, 1994.

[8] 肥田木康明, 益満健, 山岸則明, 中野裕一郎, 小林紀彦, 春山智, 小林哲則, 高西淳夫, “アイコンタクト機能を有する複数ユーザとの対話ロボット,” 情報処理学会 音声言語処理研究会, 97-SLP-17-1, pp.1-6, 1997.

[9] 松坂要佐, 東條剛史, 小林哲則, “グループ会話に参与する対話ロボットの構築”, 信学論 vol.J84-D-II, no.6, pp.898-908, 2001.

[10] T.Tojo, Y.Matsusaka, T.Ishii, T.Kobayashi, “A Conversational Robot Utilizing Facial and Body Expressions,” IEEE Proc. SMC2000 pp.858-863, 2000.

[11] 古川賢司, 東條剛史, 早田啓介, 藤江真也, 松坂要佐, 小林哲則, “空間情報共有のための身体表現機能を有する対話ロボット,” 第 17 回 日本ロボット学会学術講演会予稿集, pp.1181-1182, 1999.

[12] 中野鐵兵, 山田泰資, 松坂要佐, 東條剛史, 古川賢司, 久保田千太郎, 小林哲則, “自律ロボットの利用によるインターネットの実空間への拡張,” 情報処理学会 INTERACTION’99, IB-21, pp.167-168, 1999.

[13] ROBITA オンラインビデオライブラリ (http://www.tk.elec.waseda.ac.jp/robita/)