2007 JSME Conference on Robotics and Mechatronics

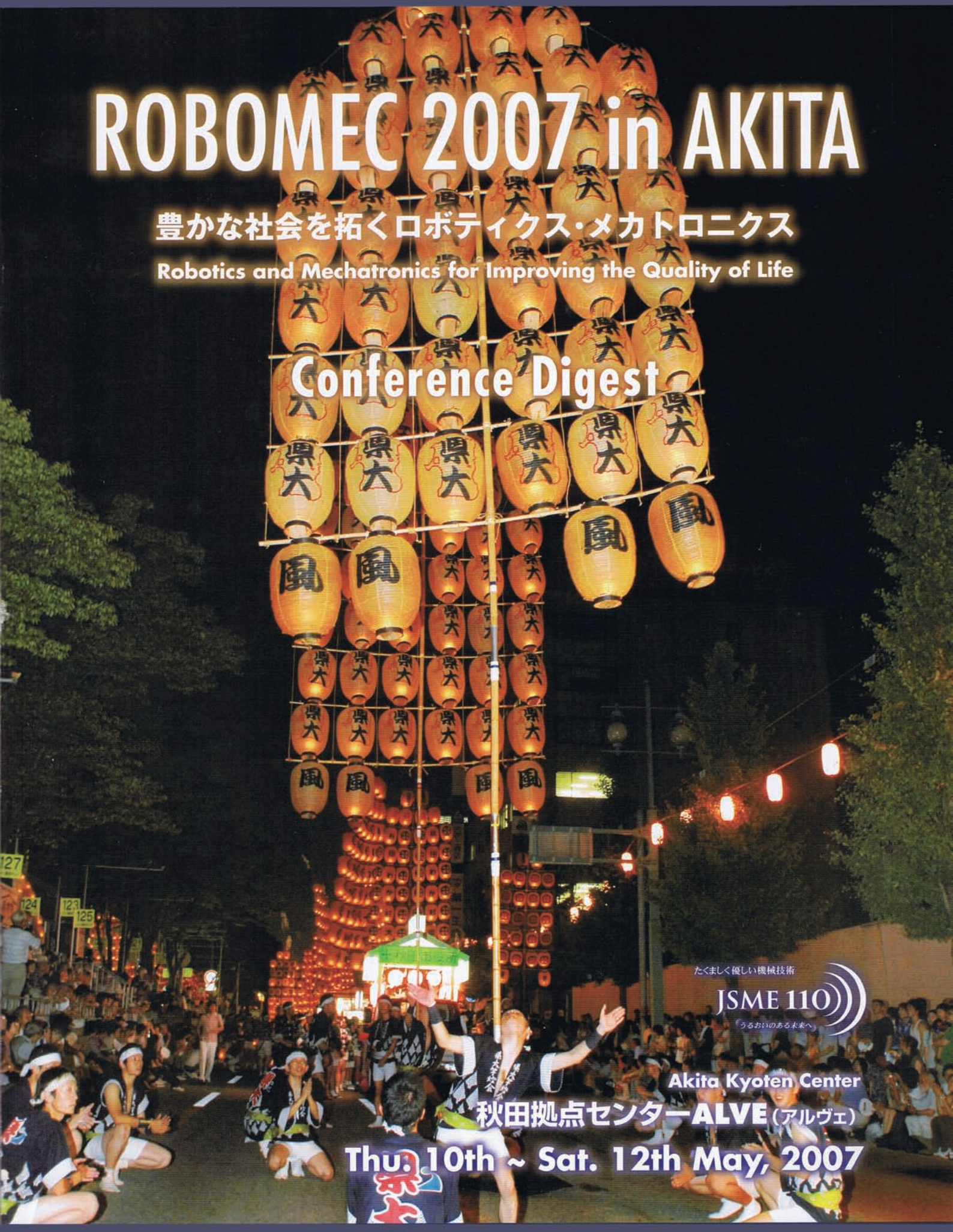
主催　社団法人 日本機械学会 ロボティクス・メカトロニクス部門

ROBOMEC 2007 in AKITA

豊かな社会を拓くロボティクス・メカトロニクス

Robotics and Mechatronics for Improving the Quality of Life

Conference Digest



たくましく優しい機械技術

JSME 110

「うるおいのある未来へ」

Akita Kyoten Center

秋田拠点センターALVE（アルヴェ）

Thu. 10th ~ Sat. 12th May, 2007

社団法人 日本機械学会

〒160-0016 東京都新宿区信濃町35番地 信濃町煉瓦館5階 TEL. (03) 5360-3500　FAX (03) 5360-3508

RT ミドルウエアを基盤としたロボット統合開発環境

RT-Middleware based Integrated Development Environment for Robots

○正 安藤慶昭 (産総研) 正 神徳徹雄 (産総研)
正 末廣尚士 (産総研) 正 北垣高成 (産総研)

Noriaki ANDO, Tetsuo KOTOKU, Takashi SUEHIRO, Kosei KITAGAKI
National Institute of Advanced Industrial Science and Technology
{n-ando, t.kotoku, t.suehiro, k.kitagaki}@aist.go.jp

The RT-Middleware is a software platform for RT-System development. In RT-System development based on the RT-Middleware, the system consists of RT-Components that are modularized RT functional elements. The system development life-cycle often can be iterative process across the whole development phase. In such development process, integrated development environment (IDE) can be an effective tool for system developers. In this paper, we propose the basic RT-Systems development life-cycle and the integrated development environment that consists of a lot of helper tools to realize concurrent development process for RT-systems.

Key Words: RT, Middleware, System Integration, Integrated Development Environment

1. はじめに

著者らは, 体系的なシステムインテグレーションを支援する実装プラットフォームとして, RT ミドルウエアを提案してきた [1, 2]. これまで人に属していたシステムインテグレーションの知識を, 特定のコンポーネントモデルに基づく実装プラットフォーム上に再構成することで, 既存のシステムの構造が明確となり, システムの設計や分析に関する構造化されたパターンの抽出が可能になるものと期待できる. RT ミドルウエアを基盤とした RT システムにおいては, RT コンポーネントと呼ばれる RT 機能要素をモジュール化したソフトウエアコンポーネントの組み合わせによりシステムが構築される.

RT ミドルウエアには様々な利点と同時に, フレームワークを利用した開発特有の問題点も存在する. その問題点を克服し, RT ミドルウエアによるシステム開発の効率化を図るために, 開発プロセスを一貫して支援する統合開発環境が必要であると考えた.

本稿では, RT ミドルウエアに基づき RT システムを開発する際の開発プロセスについて考察する. また, 開発プロセスの各フェーズ毎の作業に着目することで, 統合開発環境が備えるべき機能を整理し, 開発を支援するツールを提案する. 最後に, 統合開発環境に搭載する機能の例として, コンポーネント設計を行う rtc-template および, システム設計を行う RtcLink を Eclipse 上に実装しこれを示す.

2. RT ミドルウエア

RT ミドルウエアに基づいたロボットシステム開発では, RT システムは RT コンポーネントの集合体として構築される. RT システムでは, 多種多様なデバイスが用いられるのみならず, 多様な OS, 多様なプログラミング言語が用いられる. RT コンポーネントは分散オブジェクト CORBA を利用したフレームワークを持つため,

- ネットワーク透過
- OS 非依存
- プログラミング言語非依存

といった特徴がある. すなわち, 多種多様なプログラミング言語で実装された, 様々な OS 上で動作するモジュールであっても, またそれらが物理的に離れていても, RT コンポーネントとして実装されていれば, 相互に接続しシステムに組み込むことが可能である.

2·1 コンポーネント化による利点

RT ミドルウエアを基盤とした RT システムにおいては, システム内の機能要素がコンポーネント化される事により, 以下のような利点が得られる.

再利用性の向上 様々な機能要素をコンポーネント化することで, ソフトウエアの再利用性を高めることができる.

柔軟性の向上 コンポーネントは, その振る舞いや入出力の仕様が明確に定義されているソフトウエアモジュールであり, 同一のインターフェース仕様を持つコンポーネントを容易に追加, 入替えができるため, 一般にシステムの柔軟性は向上する.

信頼性の向上 モジュール単位で開発することにより, テストが容易になりソフトウエアの信頼性の向上が期待できる.

体系的設計の実現, 属人性の排除 全てのモジュールが RT コンポーネントとして開発されることで, 均質な構成要素として扱うことができ, モジュール間の関係も明確になるため, システムの設計を体系的に行うことができる. 結果として, システム設計のノウハウの大部分をコンポーネント仕様として明確化できる.

システムの長寿命化 モジュール単位での交換が行えるため, 技術の進歩に適応することが容易であり, 結果としてシステムの長寿命化につながる.

2·2 コンポーネント化の問題点

一方で, RT コンポーネントは決められたフレームワークに則り開発する必要があるため実装上の制約が多い. 開発者にとっては, 作成するコンポーネント仕様を明確に定義し, 詳細なドキュメントも記述しなければならないといった開発以外での煩雑さを伴う. また, システム開発時には, コンポーネントの開発・デバッグやシステム全体の開発・デバッグ等, 幾つかの開発プロセス間を行き来し, これを繰り返し行う必要がある. RT コンポーネント化による利点を最大限生かすためには, これらの問題を解決する必要がある.

著者らが以前公開した RT ミドルウエア実装例 OpenRTM-aist-0.2.0[3] ではフレームワークに基づいた開発を容易にするために, 雛形コードを生成する rtc-template と呼ばれるツールを提供した. しかしながら, rtc-template はコンポーネン

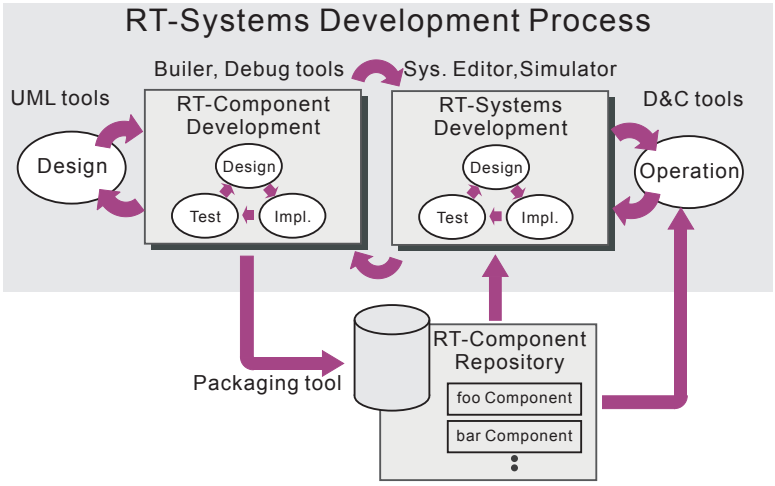


Fig.1 RT-Systems Development Flow, which consists of Component Development phase, System Development phase and System Operation phase.

ト作成の初期段階を補助するに過ぎず, これ以降のコンポーネント開発, システム開発および運用のプロセスにおいては, 開発者は上述の問題に直面することになる.

3. RT システム開発

本節では, 一連の開発プロセスにおいて, RT ミドルウエアが提供すべき機能について考察する. RT ミドルウエアを用いてシステムを開発する際には, 図 1 に示すように, システム設計, コンポーネント開発, システム開発, システム運用といったプロセスを経る.

そこで, 以下に述べるような開発プロセスを仮定する.

システムを開発する場合, まず, システム設計に基づき必要なコンポーネントを揃える. 必要となるコンポーネントの大部分は, コンポーネントリポジトリと呼ばれる, 既存コンポーネントを蓄積するデータベースに存在し, これを利用する. リポジトリに存在しないコンポーネントは新たに作成する必要がある.

新たなコンポーネントを作成する際には, RT コンポーネントのフレームワークを利用するため, 大部分のコードを自動的に生成することが可能である. 開発者は独自のロジックをフレームワークに埋め込む形でコンポーネントを作成する. 作成したコンポーネントをシステムに組み込む際には, 事前に単体でテストを行い, 仕様通りの動作を行うかどうかテストを行う.

こうして必要なコンポーネントが揃った後, コンポーネントを組み合わせてシステムを構築する. コンポーネントの接続情報やパラメータ設定情報は XML 等の構造化されたフォーマットで記述され, 運用時にはこのファイルに基づきコンポーネントの起動, パラメータ設定, 接続を行い, システムが実際に動作する.

このような開発プロセスを支援する様々なツールが存在すれば, 開発者の作業量を軽減でき, システム構築の本質的な部分に集中することができると考えられる.

3·1 コンポーネント開発

コンポーネント開発時には, 図 2 に示すように, コンポーネントのコード作成, デバッグ, パッケージ化といった作業が想定され, 以下に示すような機能が必要と考えられる.

3·1.1 コンポーネントビルダ

ウィザード形式や, GUI などでコンポーネントの基本的仕様を入力し, コンポーネントの雛形コードを自動生成するツー

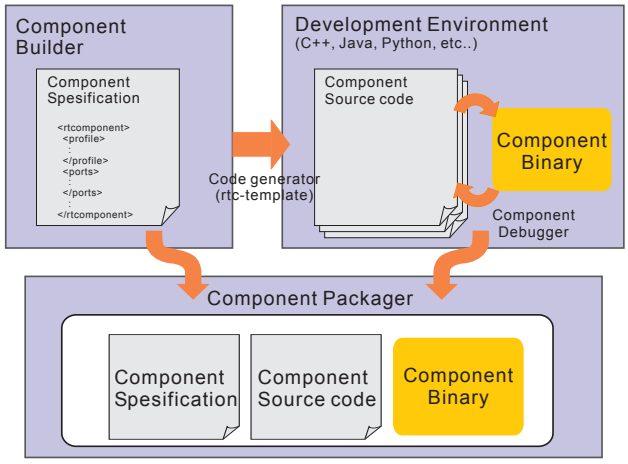


Fig.2 RT-Component development phase, which consists of Component Design phase, implementation phase and packaging phase.

ルが考えられる. 現状の rtc-template でも, 同等のことができるが, 生成後のソースコードの編集と, 仕様の変更などが同時並行的に行えるようなツールにより一層の効率化が望める.

3·1.2 コンポーネントデバッガ

システムにコンポーネントを組み込むには, 事前に十分なデバッグを行う必要がある. 従って, 作成したコンポーネントをシステムに組み込まずにテスト可能な, デバッガツールが考えられる. 通常のデバッガとともに, コンポーネントのデバッグに特化した機能が求められる.

例えば, 入力ポートに任意の入力を与えたり, 出力ポートからの出力をモニタするグラフ機能なども必要となる. また, 外部に対するインターフェースのオペレーションを手動でコールしたり, コンポーネント内のロジックをステップ実行しその挙動をモニタできる機能なども必要となる.

3·1.3 コンポーネントパッケージャ

RT コンポーネントは, その入出力やインターフェースの仕様を明確に定義し, ドキュメント化されて初めて, 再利用可能な価値のあるコンポーネントとなる. また, 再利用のためには, コンポーネントのソースコードやバイナリ, 仕様定義ファイルやドキュメントなどを共通フォーマットでパッケージ化

する必要がある．これらの作業の定型的な部分は，ツールによる補助で効率化が見込める．

3·2　システム開発

RT コンポーネントを組み合わせてシステムの設計・開発を行う際には，静的または動的なシステム構成や，構成したシステムのシミュレーションといった作業が想定される (図 3)．静的システム構成とは，コンポーネント間の静的接続を構成する作業であり，動的システム構成とは，時間軸およびイベントによりシステム構成を動的に切り替え一連のシナリオを実行するシーケンスを作成する作業である．

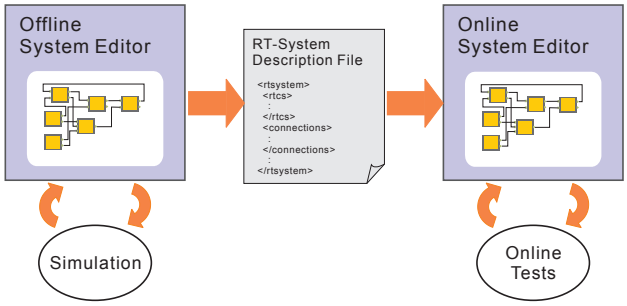


Fig.3 RT-Systems Development. Offline design phase includes simulation. Online design phase includes debugging in real systems.

3·2.1　コンポーネントリポジトリ

コンポーネントリポジトリとはパッケージ化されたコンポーネントを蓄積するデータベースである．開発者は，開発に必要なコンポーネントをリポジトリから参照し，システムに組み込む．

3·2.2　システムエディタ

静的システム構成には大きく分けて 2 種類の構成法が考えられる．一つは，動作中のコンポーネントを接続しシステムを構築するオンラインシステム構築．もう一つは，システム構築時に動作していないコンポーネント対して，コンポーネント定義のみを利用してシステムを構築するオフラインシステム構築である．

オンラインシステム構築，オフラインシステム構築ともに，構成情報はシステム接続情報を XML 等で記述したファイルに保存され，運用時にコンポーネントの起動やパラメータ設定，接続等に利用される．

3·2.3　シナリオエディタ

上述のシステムエディタは，静的なコンポーネントの接続関係を記述するのみであるが，実際に運用されるシステムでは，システム構成は時間軸やイベントなどにより動的に変化するのが一般的である．

シナリオエディタは，タイムライン上に沿って，あるいはイベントに対して動的に変化するシステム構成を記述するためのツールである．

3·2.4　シミュレータ

システム全体のテスト，デバッグを実機で行うことは安全面，コスト面などから困難な場合が多い．システム全体の検証のためにも，シミュレータは非常に重要なツールである．また，RT コンポーネントは実デバイスコンポーネントとシミュレーションコンポーネントを容易に入れ替えることが可能であるため，部分的にシミュレーションを行うことも可能である．

3·3　システム運用

構築された RT システムを運用する際には，図 4 のように，実行アプリケーションがシステム接続情報に基づき，コンポーネントをネットワーク上のコンピュータに配布，起動，接続およびパラメータの設定等を行いシステムが動作する．このような運用を想定する場合，基盤ツールとして以下のようなものが考えられる．

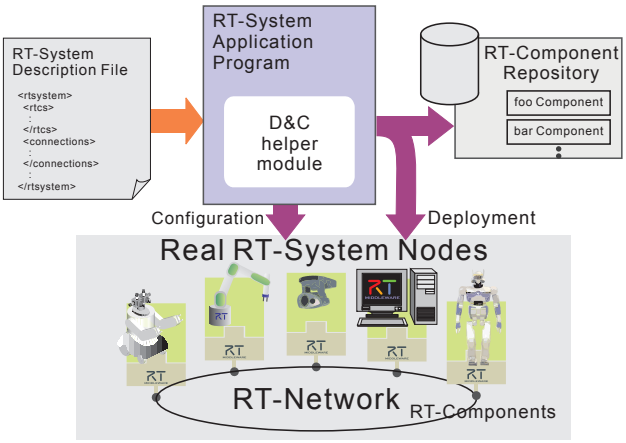


Fig.4 RT-Systems Operation. A RT-System application constructs and runs RT-System network through deployment and configuration helper interfaces.

3·3.1　デプロイメント

デプロイメントとは，コンポーネントを適切なノードに配置し，ロードし，実行等のライフサイクルを管理する仕組みである．RT ミドルウエアは分散環境を前提としており，キーボードやモニタなどが無い多数の組込ノードの利用が想定されるロボットシステムにおいては，手動でこれらの作業を行うことは現実的でない．3·2.2 で述べたシステムエディタと連携し，システム構成情報から自動的にデプロイメントを行うツールを提供することで，単一ノード上の作業とほぼ同様の作業量で分散環境を構築できる．

3·3.2　コンフィギュレーション

システム構成時には，システムエディタから各コンポーネントの設定を行うことができるが，運用時にはシステムエディタは通常動作していない．従って，多数の RT コンポーネントから構成されるシステムにおいては，設定情報を集中的に管理し，各コンポーネントに対して設定できるスキームも必要となる．

4.　ロボットシステム統合開発環境

上述したシステム開発のプロセスは，実際には一方向的ではなく，横断的かつ繰り返し行われる．こうした一連の作業を，コンカレントに行うことができる統合開発環境を整備することにより，RT システム開発のより一層の効率化が図られるものと考えられる．上述の様に分類したが，実際には細かなツールの組み合わせで実現されるもので，これらのツールはフェーズ特有なものや，フェーズに関係なく利用可能なものが存在する．

また，上述した基本ツール以外に，ロボットの様々なアプリケーションに応じた特有のツールが必要な場合があり，それらのドメイン毎に用いられるツールは基本ツールとの連携も必要となる．

このように，RT ミドルウエアのための統合開発環境は，様々なツールを取捨選択して組み込むためのプラットフォームと

しての機能が必要である．このような目的に合致するプラットフォームとして，Eclipse が挙げられる．我々は，RT ミドルウエアのための統合開発環境プラットフォームとして，Eclipse を採用した．

5. Eclipse

Eclipse は Eclipse Foundation が開発するオープンソースの Java 言語およびその他言語のための統合開発環境であり，同時に開発環境のためのプラットフォームでもある [4]．Eclipse の基本部分は plug-in を実行するためのプラットフォームであり，様々な開発環境は多くの plug-in の集合体として構築される．デフォルトで付属する Java の開発環境自体も plug-in として実現されており，さらに plug-in を追加することで様々な開発環境に容易に拡張することが可能である．

Eclipse の特徴として，

- plug-in 機構により容易に拡張可能であり，plug-in 同士の連携も可能である．
- Rich Client Platform (RCP) と呼ばれる仕組みにより，plug-in をスタンドアロン化することが容易にできる．
- Java で実装されているため，多くのプラットフォームで動作する．

等が挙げられる．

著者らは，こうした Eclipse の特徴が，ロボット用統合開発環境を構築する上で有用であると考え，Eclipse 上に上述した様々なツールを構築することにした．本稿では，Eclipse 上に構築したコンポーネント開発ツールである rtc-template およびオンラインシステム開発ツールである RtcLink を例として示す．

5·1 rtc-template

OpenRTM-aist-0.2.0 に付属していたコードジェネレータ rtc-template を Java に移植し，Eclipse plug-in 化した．

コンポーネント開発者は Eclipse のエディタから，コンポーネントの仕様を入力し，RT コンポーネントの雛形コードを生成させることができる．さらに，Eclipse の代表的 C/C++ 開発ツールである，CDT(C/C++ Development Tools) と連携させ，コンポーネントの仕様作成からコードの実装までをシームレスに行うことができる．

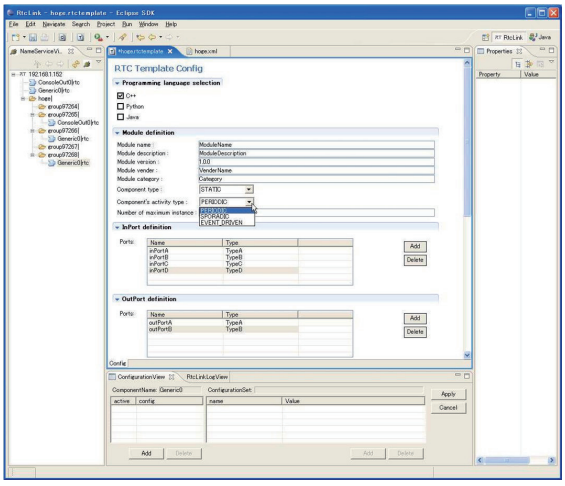


Fig.5 rtc-template on Eclipse. RT-Component design tool running on Eclipse IDE.

5·2 RtcLink

システム開発ツールとして OpenRTM-aist-0.2.0 に付属していた RtcLink を Eclipse に移植した (図 6)．RtcLink は Eclipse plug-in として実装されているため，新たに実装する Eclipse plug-in では RtcLink が持つ機能を容易に利用することができる．

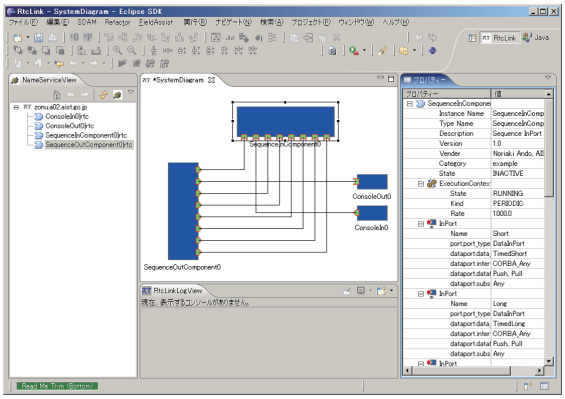


Fig.6 RtcLink on Eclipse. RT-System online design tool running on Eclipse IDE.

6. おわりに

本稿では，RT ミドルウエアに基づいたシステムを構築する際のプロセスを，設計，開発，運用とに分け，その際に必要となる作業ついて考察した．各開発フェーズにおける作業をサポートするツールを，機能毎に分類，整理し，必要なツールを提案した．

設計から運用に至るプロセスは一方向的ではなく，下流プロセスの結果を上流プロセスに容易にフィードバックするのが一般的である．これらの作業プロセスを透過的に行うために，各フェーズで必要となる作業を補助するツールを統合開発環境上に整備することを提案し，そのためのプラットフォームとして我々は Eclipse を選択した．

ツールの例として，コンポーネント設計のためのツール rtc-template，システム設計のためのツール RtcLink をそれぞれ Eclipse 上に構築した．

今後は，本稿で提案したツール群を順次作成し，RT ミドルウエアを用いたシステム構築のプロセスの効率化を目指す．

謝 辞

本研究の一部は，新エネルギー・産業技術総合開発機能 (NEDO) 次世代ロボット共通基盤開発プロジェクトの一環として実施されたことを記し，ここに感謝の意を表する．

参考文献

[1] 末廣 尚士，北垣 高成，神徳 徹雄，尹 祐根，安藤 慶昭，”RT コンポーネントの実装例.RT ミドルウエアの基本機能に関する研究開発 (その 1)”，第 21 回 日本ロボット学会学術講演会予稿集，p.1F27，2003.09

[2] Noriaki ANDO, Takashi SUEHIRO, Kosei KITAGAKI, Tetsuo KOTOKU, Woo-Keun YOON, “RT-Middleware: Distributed Component Middleware for RT (Robot Technology)”, 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005), pp.3555-3560, 2005

[3] OpenRTM-aist-0.2.0 Tutorial Web page, http://www.is.aist.go.jp/rt/OpenRTM-aist-Tutorial/

[4] Eclipse Fooundation, http://www.eclipse.org/