コンポーネント指向 RT システム開発工程における有用なツール群に関する調査

○高橋 三郎（産総研），花井 亮（産総研），BIGGS Geoffrey（産総研），
原 功（産総研），安藤 慶昭（産総研）

Investigation of Software Development Tools for Component-based RT System

○ Saburo TAKAHASHI (AIST), Ryo HANAI (AIST), Geoffrey BIGGS (AIST),
Isao HARA (AIST), and Noriaki ANDO (AIST)

Abstract： In this paper, we showed the survey results about the tools and the methods to develop the RT system. Especially we focused in RT middleware with the standard software development process. At first, we explained about ESPR v2.0 as the standard software development process. Then we showed the investigating results for each development process. Finally, we proposed new tools and components that are required in the future development for using RT middleware.

1. 緒言

本稿では，ロボットシステム構築を効率化する RT ミドルウェアおよび RT ミドルウェアを用いた開発と親和性の高いコンポーネント指向設計に着目し，ロボットシステム製品開発を行う際に，各開発工程でどのような手法およびツールが利用が可能であるかについての調査を行った．

コンポーネント指向設計 (CBSE：Component-based software engineering) [1] とは，疎結合なインタフェースを有する再利用可能なコンポーネント群で構成する設計技法である．ロボットシステムにおける上記コンポーネントは RTC（Robotic Technology Component） [2] と呼ばれ，ソフトウェア標準化団体 OMG にて規格化されている．

本稿で調査対象とする開発プロセスモデルは，組み込みソフトウェア開発プロセスモデルのデファクトスタンダードとなっている ESPR (Embedded System development Process Reference) v2.0 [3] とした．ESPR v2.0 で定義されるプロセスモデルの概略を図 1 に示す．ソフトウェア開発プロセスモデルの代表的な規格としては，ISO/IEC12207 (Systems and software engineering-Software life cycle processes)，ISO/IEC15288 (Systems and software engineering-System life cycle processes) などが存在するが，上記の代表的な規格が体系的に整理され，多くの製品開発現場で用いられているといった観点から，本稿では ESPR v 2.0 を調査対象プロセスモデルとして選択した．

過去の研究では，ロボットシステム開発においてシステムモデリング記述言語 SysML [4] を利用することで，システムエンジニアリングプロセスおよびセーフティエンジニアリングプロセスにおけるトレーサビリティの確保，要求分析，設計効率化に対しての有効性が示されている [5]．一方，ソフトウェアエンジニアリングプロセスにおいては，利用が可能なツールが世の中に数多く存在しているが，利用できる状況，条件については体系的に整理されていない．また，サポートプロセスについても，数多くのプロジェクト管理手法，ツールが提唱されているが，その選択は開発体制や対象となるソフトウェアの種別に大きく左右されるため，今後の検討課題と考え，本稿では言及していない．

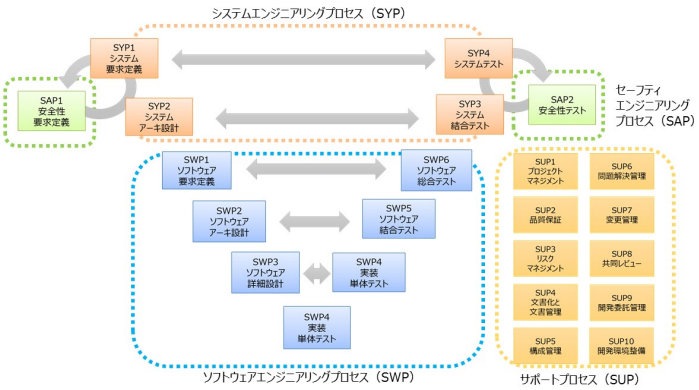

Fig. 1: Structre of ESPR v2.0

2. 開発プロセスのマッピング

コンポーネント指向設計は，コンポーネントの再利用性を重視した技法であり，コンポーネント内部の設計，実装

および単体テストについては言及されていない．そこで，本稿では ESPR v 2.0 記載の「機能ユニット」を「コンポーネント」相当と仮定し，ソフトウェア・アーキテクチャ設計ではコンポーネントの外部仕様の設計，ソフトウェア詳細設計ではコンポーネントの内部仕様の設計が行われると想定した．また，上記に伴い，コンポーネント内部は複数のプログラムユニット（関数やメソッドなど）で構成されるものとした．上記のプロセスモデル，工程のマッピング概念を図 2 に示す．

3. 調査手法

本稿の調査では，開発におけるソフトウェアエンジニアリングプロセスに着目し，模擬的なロボットシステムの開発を通じて各工程のアウトプットを作成するために利用が可能なツール，手法についての調査および考察を行った．開発対象システムの諸元を表 1 に示す．開発対象としたシステムは，RT ミドルウェアとして OpenRTM-aist [6] を利用したロボットアームシステムであるが，本稿の内容としては対象となるロボットシステムに強く依存した内容となっていないため，対象システムについての詳細な解説は割愛する．

Table. 1: Target System

対象システム	多関節ロボットアームの制御
RT ミドルウェア	OpenRTM-aist
ロボット制御 PC	Windows，Linux
開発ホスト PC	Windows，Linux

4. 調査結果

本章では，ESPR v 2.0 記載の工程ごとに設計，テストに必要な要素を抽出し，各要素に対して利用が可能な手法およびツールについての考察を述べる．

4.1 ソフトウェア要求定義

ソフトウェア要求定義工程で明確化すべき要素は

・ ソフトウェアの機能要求
・ ソフトウェアの非機能要求
・ 開発上の各制約を加味した要求間の優先順位

である．

調査結果を表 2 に示す．機能要求および非機能要求に関しては，SysML のユースケース図，要求図，同様に UML [7] のユースケース図，ユースケース記述を利用することで記述が可能である．要件の優先順位については，一般的に開発ターゲットの特性を鑑みてビジネス上のステークホルダとの議論により決定されるものであり，ここでは不問とした．

Table. 2: Methods for S/W Requirement Definition

要素	利用可能な手法
機能要求の明確化	SysML（ユースケース図，要求図）
	UML（ユースケース図，ユースケース記述）
非機能要求の明確化	SysML（要求図）
要件間の優先順位決定	-

4.2 ソフトウェア・アーキテクチャ設計

ソフトウェア・アーキテクチャ設計工程で明確化すべき要素は

・ コンポーネント構成の検討
・ コンポーネントの振る舞い，責務分担の決定
・ コンポーネント間のインタフェース設計
・ コンポーネントの再利用設計
・ システムリソース成立性の検討（性能，メモリ量の見積もりなど）

である．

調査結果を表 3 に示す．コンポーネントの構成および振る舞いに関しては，SysML，UML を利用することで記述が可能である．また，複合コンポーネント [8] を設計する場合は複合コンポーネント作成支援ツール [9] を利用することで，GUI を用いて直感的に理解しやすい形で設計できる．インタフェース設計においては， NEDO 次世代ロボット知能化技術開発プロジェクトで策定が進められた共通インタフェース仕様書 [10] の利用が可能である．該当する機能を有するコンポーネントを開発する際に上記インタフェース仕様は大いに参照できると考えられる．また，文書の自動生成という観点では，doxygen [11] ，wasanbon [12] などのツールも有効である．コンポーネント再利用に関しては，現状，複数のコンポーネント管理サイトが存在し，独自の方式でコンポーネントを管理しているため，横断的な管理手法が存在せず，管理されている情報にばらつきがあるという課題が存在する．また，リソース成立性の設計に関しては，現状有効な手法・ツールは存在しなかった．

4.3 ソフトウェア詳細設計

ソフトウェア詳細設計で明確化すべき要素は

・ コンポーネント内のプログラムユニット構成の検討
・ プログラムユニットの振る舞い，責務分担の決定
・ プログラムユニット間のインタフェース設計
・ プログラムユニット内部のロジック設計
・ コンポーネント内リソース成立性の検討（メモリ量の見積もりなど）

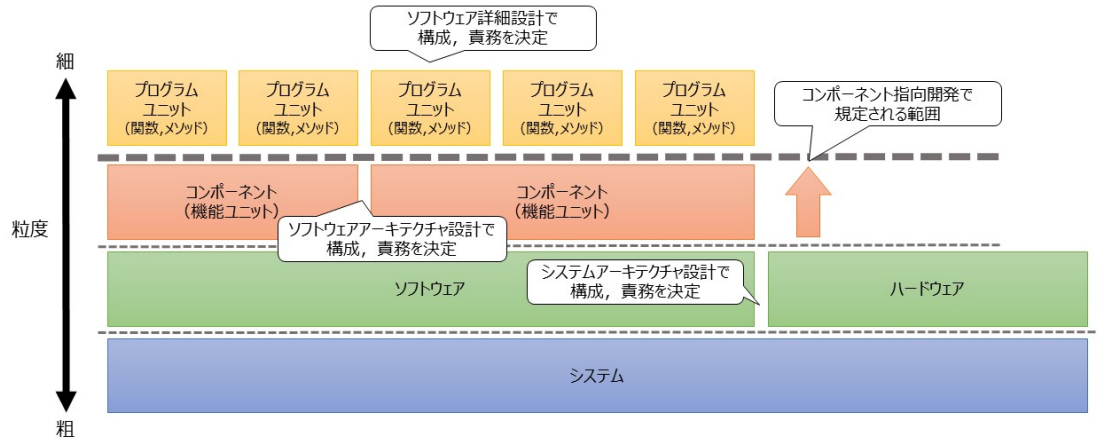

Fig. 2: Architecture of Engineering Process

Table. 3: Methods for S/W Architectural Design

要素	利用可能な手法
コンポーネント構成設計	SysML（コンポーネント図）
	UML（パッケージ図，コンポーネント図，クラス図，オブジェクト図）
	複合コンポーネント作成支援ツール
コンポーネント振る舞い設計	SysML（コンテキスト図，ブロック図）
	UML(シーケンス図，相互作用図，コミュニケーション図，タイミング図，ステートマシン図)
インタフェース設計	共通インタフェース仕様書
	wasanbon(仕様書の自動生成のみ)
	doxygen (仕様書の自動生成のみ)
コンポーネント再利用	OpenRTM-aist ホームページ
	OpenRTC-aist ホームページ
	wasanbon ホームページ
	各研究，教育機関ホームページなど
リソース成立性の検討	-

Table. 4: Methods for S/W Detiled Design

要素	利用可能な手法
プログラムユニット構成の検討	UML（パッケージ図，コンポーネント図，クラス図，オブジェクト図）
振る舞い，責務分担の決定	UML(シーケンス図，相互作用図，コミュニケーション図，タイミング図，ステートマシン図)
インタフェース設計	doxygen (仕様書の自動生成のみ)
ロジック設計	-
リソース成立性の検討	-

である．本節で示すプログラムユニットとは，機能をつかさどるコンポーネントを詳細化した実装における最小単位のことで，関数やメソッドなどを指している．

調査結果を表 4 に示す．基本的には通常のソフトウェア開発と同様に UML を利用した設計が有効である．しかし，ロジック設計に対して有用なツール・手法は無い．内部設計に関しては，ロボットシステムに限らない一般的なソフト開発と同等の課題であり，コンポーネントの内部設計，コアロジックに関しては取り扱わないという RT ミドルウェアのポリシー通りの結果とも言える．

4.4 ソフトウェア実装

ソフトウェア実装で実施すべき要素は

・詳細設計に従ったプログラムユニットの作成

のみである．

調査結果を表 5 に示す．OpenRTM-aist には標準でソースコードテンプレートとなるスケルトンコードを出力する機能は具備されている．コマンドラインから利用できる rtc-template や GUI フロントエンドを備える RTC ビルダが OpenRTM-aist と同梱で配布されている．また，Web ブラウザ上で RTC ビルダの機能を実装した RTC Builder on the Web [13] も存在する．上記に加え，設計情報からコードの自動生成が可能なツールについても Astah SysML-RTM 連携プラグイン [14]，PatternWeaver for RT-Middleware [15]，ZIPC-RT [16] などをベンダ各社がリリースしており，これらを用いることで設計を含めたトレーサビリティ確保が可能である．なお，これらのツールは設計ツールとしての側面も有しているが，いずれも UML，SysML の手法をベースとしているため，設計ツールとしての表記は割愛した．また，上記と目的は異なるツールとして，コアロジックとポートなどの RTC 依存のインタフェースを分離して実装可能な RTSeam [17] を利用することで，コード可読性および単体テスト効率の向上が可能である．

4.5 ソフトウェア単体テスト

ソフトウェア単体検証で実施すべきことは

・詳細設計に従った単体テスト仕様設計

Table. 5: Methods for S/W Implementation

要素	利用可能な手法
プログラムユニット作成	rtc-template （OpenRTM-aist）
	RTC Builder （OpenRTM-aist）
	RTC Builder on the Web
	Astah SysML-RTM 連携プラグイン
	PatternWeaver for RT-Middleware
	ZIPC-RT
	RTSeam

・ プログラムユニットのホワイトボックステスト実行
・ 検出された不具合の解析

である．

調査結果を表 6 に示す．本工程において，一般的なソフトウェア開発では CppUnit [18] に代表される xUnit 系のテストフレームワークが利用されるが，現状，RT ミドルウェアと連携するテストフレームワークは存在しない．テストの実行に関しては，RTSeam を利用することで，コアロジックを抽出した検証可能という観点から，コンポーネント内のホワイトボックステストの効率化が期待できる．

Table. 6: Methods for S/W Unit Testing

要素	利用可能な手法
単体テスト仕様設計	-
ホワイトボックステスト実行	RTSeam
不具合解析	-

4.6 ソフトウェア結合テスト

ソフトウェア結合テストで実施すべきことは
・ アーキテクチャ設計に従った結合テスト仕様設計
・ コンポーネントのブラックボックステスト実行
・ 検出された不具合の解析

である．

調査結果を表 7 に示す．OpenRTM-aist v 1.1.2 から同梱されている rtshell [19] をはじめ，コンポーネントの外部インタフェースに対してのデータ挿入など，本工程のテストを支援するツールは揃っている．また，オフィスソフトを操作するための RTC 群 [20]，RTC デバッガ [21]，RTStorage [22] については GUI フロントエンドも備えており，効率的なテスト・デバッグが期待できる．一方，検証仕様自体の作成や，非データポートのインタフェースであるサービスポート，コンフィグレーションを使用した検証は実施することができず，データ挿入で発生が困難な異常系（レイテンシ増大や例外発生など）の検証についても考慮されていない．

4.7 ソフトウェア総合テスト

ソフトウェア総合テストで実施すべきことは

Table. 7: Methods for S/W Integration Testing

要素	利用可能な手法
結合テスト仕様設計	-
ブラックボックス	rtshell（OpenRTM-aist）
テスト実行	RTSystemEditor（OpenRTM-aist）
	サンプルコンポーネント（OpenRTM-aist）
	オフィスソフトを操作するための RTC 群
	RTC デバッガ
	RTStorage
不具合解析	オフィスソフトを操作するための RTC 群
	RTC デバッガ
	RTStorage
	ロギング機能（OpenRTM-aist）

・ 要求仕様に従った総合テスト仕様設計
・ 機能要求仕様のテスト実行
・ 非機能要求仕様のテスト実行
・ 検出された不具合の解析

である．本工程は複数のコンポーネントを接続し，要求仕様に基づく機能が実現できているかを確認することが最大の目的である．

調査結果を表 8 に示す．手法・ツールとしては，ソフトウェア結合テストとほぼ同等の手法が利用可能であり，正常系の検証，コンポーネント間動作を確認できるツールは揃っている．また，本工程からハードウェアとの結合を意識した検証が必要となるため，RTC の組み込みが可能なシミュレータ choreonoid [23] を利用することで，後工程への流出する不具合を低減できると考えられる．しかし，ソフトウェア結合テストを同様に検証仕様の作成，異常系の検証については考慮されておらず，非機能要件（性能等）に関しても検証ができるツールは存在していない．

Table. 8: Methods for Comprehensive S/W Testing

要素	利用可能な手法
総合テスト仕様設計	-
機能要求仕様	rtshell（OpenRTM-aist）
テスト実行	RTSystemEditor（OpenRTM-aist）
	choreonoid
	オフィスソフトを操作するための RTC 群
	RTC デバッガ
	RTStorage
非機能要求テスト実行	-
不具合解析	オフィスソフトを操作するための RTC 群
	RTC デバッガ
	RTStorage

5. 現状の課題

これまでの調査結果から，今後拡充が期待される機能について表 9 に整理した．

Table. 9: Methods & Tools to Be Implemented

工程	拡充が期待される代表的な機能
ソフトウェア要求定義	-
ソフトウェア	再利用可能な RTC レポジトリの整備
アーキテクチャ設計	リソース成立性設計支援
ソフトウェア詳細設計	-
ソフトウェア実装	-
ソフトウェア単体テスト	テスト仕様自動生成
	テスト自動実行環境
ソフトウェア結合テスト	テスト仕様自動生成
	異常系テスト実行支援
ソフトウェア総合テスト	テスト仕様自動生成
	非機能要求テスト実行支援

本調査の結果を踏まえ，今後拡充が望まれる代表的なツール，手法は下記の三点であると結論づけた．

再利用可能な RTC レポジトリの整備

コンポーネント指向設計のメリットを享受するためにも，横断的なコンポーネントレポジトリの作成が必要である．OSS（Open Source Software）として代表的なロボットソフトウェアプラットフォームである ROS (Robot Operating System) では，パッケージ単位で管理することで簡単に機能の取り込みができる仕組み [24] を有している．本機能実現のためには，コンポーネントを管理する上で必要な情報についての精査・定義，著作権の取り扱いの明文化およびコミュニティでの合意形成が必要となる．

テスト仕様および実行環境自動生成

RTC は規格化されたコンポーネント形式で設計，実装されているため，動作，インタフェース仕様が明確である．この制約は，テスト仕様およびテストコードの自動生成との親和性が高いと考えられる．RTC のインタフェースであるデータポート，サービスポート，コンフィグレーションを網羅的にテストできるようなテスト自動化の仕組みを導入することで，単体テスト，結合テスト，総合テスト工程の大きな省力化が期待できる．

異常系・非機能要求テスト実行支援

より品質の高いロボットシステムを構築するためには，ソフトウェアテストの段階で擬似的な故障や例外，遅延の増大をエミュレートできる仕組みや，その際の性能を計測できる仕組みが必要だと考えられる．故障挿入テスト（Fault Injection Test [25]）可能なモック，スタブコンポーネントおよび性能を計測するためのプロファイラ，トレーサの実現が期待される．

6. 結言

本稿では，標準的なソフトウェア開発プロセスモデル ESPR v2.0 に準拠した RT ミドルウェアを用いたシステム及びコンポーネント開発において，利用が可能なコンポーネント，ツール，手法についての調査結果を報告した．また，得られた知見に基づき,「再利用可能な RTC レポジトリの整備」,「テスト仕様および実行環境自動生成」,「異常系・非機能要求テスト実行支援」を将来的に拡充すべきとの提言を行った

今後は，本稿で得られた知見を元に下記についての調査を継続して行う予定である．

・ 実システム開発での手法，ツールの定量的効果検証
・ セーフティエンジニアリングプロセスに関する調査
・ サポートプロセスに関する調査

参考文献

[1] W. Kozaczynski, G. Booch, “Component-Based Software Engineering,” IEEE Software Volume: 155, Sept.-Oct. 1998, pp. 34–36.

[2] Object Management Group，“Robotic Technology Component (RTC),” http://www.omg.org/spec/RTC/.

[3] 独立行政法人　情報処理推進機構　ソフトウェア・エンジニアリング・センター　編著，“ESPR Ver2.0 組込みソフトウェア向け開発プロセスガイド,” https://www.ipa.go.jp/files/000005126.pdf.

[4] Object Management Group，“OMG Systems Modeling Language,” http://www.omgsysml.org/.

[5] 中坊嘉宏，“SysML によるロボット介護機器のモデル化,” ロボティクス・メカトロニクス講演会 *2014(ROBOMECH2014)*，1P2-D01, 2014.

[6] Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T., Yoon, W.-K. “RT-Middleware:Distributed Component Middleware for RT (Robot Technology),” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pp. 3555-3560, 2005.

[7] 水川真, 坂本武志, 大原賢一, “UML と RT ミドルウェアによるモデルベースロボットシステム開発,” オーム社, 2009.

[8] Noriaki ANDO, et al., “Composite Component Framework for RT-Middleware (Robot Technology Middleware),” *2005 IEEE/ASME AIM2005*, pp.1330-1335, 2005.

[9] 宮本信彦，“複合コンポーネント作成支援ツールの開発,” 第 *16* 回システムインテグレーション部門講演会 *(SI2015)*，pp1043–1048, 2015.

[10] NEDO 次世代ロボット知能化技術開発プロジェクト，“共通インタフェース仕様書,” http://www.openrtm.org/openrtm/ja/project/Recommendation_CommonIF.

[11] Dimitri van Heesch., “Doxygen,” www.doxygen.org/.

[12] 菅佑樹, 尾形 哲也，“RT システムの再利用性を高めるためのオープンフレームワークの開発,” 第 *14* 回システムインテグレーション部門講演会 *(SI2013)*, 3G1-2, 2013.

[13] 原功, “RTC ビルダ on the Web,” 第 *13* 回システムインテグレーション部門講演会 *(SI2012)*，1G2-3, 2012.

[14] 株式会社チェンジビジョン, "SysML-RTM(OpenRTM-aist) 連携プラグイン," http://astah.change-vision.com/ja/feature/sysml-rtm-plugin.html.
[15] 株式会社テクノロジックアート, "PatternWeaver for RT-Middlewarer," http://pw.tech-arts.co.jp/pw/rt_middleware/.
[16] キャッツ株式会社, "ロボティクス開発環境「ZIPC-RT」,"http://www.zipc.com/special/zipc_rt/.
[17] 小田桐康暁，中本啓之，西之原寛，"RT コンポーネント開発へのテスト駆動開発手法の導入,"ロボティクス・メカトロニクス講演会 *2011(ROBOMECH2011)*, 1A1-H15, 2011.
[18] "CppUnit - C++ port of JUnit," https://sourceforge.net/projects/cppunit/.
[19] BIGGS Geoffrey, 松坂要佐, 安藤慶昭, 神徳徹雄, "rtshell 3.0: RT ミドルウェア用コマンドラインツール,"ロボティクス・メカトロニクス講演会 *(ROBOMEC2011)*, 2P1-L03, 2011.
[20] 宮本信彦，"オフィスソフトを操作するための RTC 群,"第 *15* 回システムインテグレーション部門講演会 *(SI2014)*，1A2-1, 2014.
[21] 株式会社セック, "RTC デバッガ,"http://www.sec.co.jp/robot/download_tool.html.
[22] "RT ミドルウェアのためのデータ記録・再生用ツール RtStorage," https://github.com/zoetrope/RtStorage.
[23] Shin'ichiro Nakaoka, "Choreonoid: Extensible Virtual Robot Environment Built on an Integrated GUI Framework," *2012 IEEE/SICE International Symposium on System Integration (SII2012)*, pp.79–85, December, 2012.
[24] Brian Gerkey and Ken Conley, "Robot Developer Kits," *Robotics & Automation Magazine, IEEE, September 2011*, pp.15, 2011.
[25] 黒田滋樹，柴山悦哉，"Software Fault Injection を用いた開発時テスト支援環境,"日本ソフトウェア科学会第 *22* 回大会（*2005* 年度）論文集, pp.82–86, 2005.