CONFERENCE DIGEST

ロボティクス・メカトロニクス講演会2012
2012 JSME Conference on Robotics and Mechatronics

ROBOMEC2012 IN HAMAMATSU

グリーン&ライフイノベーションで未来を拓くロボティクス・メカトロニクス

Robotics and Mechatronics
- Pioneer the Future with Green & Life Innovations -


浜名湖（弁天島）


浜松城


家康公像

May 27 Sun. - 29 Tue., 2012 ACT CITY HAMAMATSU

主催　一般社団法人 日本機械学会 ロボティクス・メカトロニクス部門
The Japan Society of Mechanical Engineers, Robotics and Mechatronics Division

高信頼 RT ミドルウエアの開発

産総研 ○安藤慶昭, ビグズ・ジェフ, 中坊嘉宏, 水口大知, 藤原清司, 原功, 神徳徹雄
株式会社セック 近藤理良, 豊田光弘, 池添明宏, 中本啓之, 草間康利, 長瀬雅之
ゼネラルロボティックス株式会社 齋藤元, 株式会社グローバルアシスト 坂本武志

Development of Dependable RT-Middleware

Noriaki Ando, Geoffrey Biggs, Yoshihiro Nakabo, Daichi Mizuguchi, Kiyoshi Fujiwara, Isao Hara, Tetsuo Kotoku, AIST,
Masayoshi Kondo, Mitsuhiro Toyoda, Akihiro Ikezoe, Hiroyki Nakamoto,
Yasutoshi Kusama, Masayuki Nagase, SEC Co.,Ltd.,
Hajime Saito, General Robotix, Inc., Takeshi Sakamoto, Global Assist Co., Ltd.

Abstract—
Dependable RT-Middleware (d-RTM) is implemented to realize component based safety RT-system development in this paper. RT-system which can be harmful to human beings should be dependable and be guaranteed its safety. The d-RTM, which provides RT-Component framework with safety functionalities, is developed according to the IEC 61508 standard for functional safety. Its safety concepts, safety requirement specifications are shown with examples of actual coding with d-RTM.

Key Words: Functional safety, dependable systems, RT-Middleware, RT-Component

1. はじめに

人と接触し得る環境で利用される RT システムは、誤作動により人間に危害を加える危険性がある。こうしたシステムでは、ハードウエアおよびソフトウエアに対して設計時から安全に対して多大な注意を払い開発を進めなければならない。特に、複雑な制御ソフトウエアを安全に実装することは難しく、高いコストが掛かるため、RT システムの開発・市場への投入にとって大きな障壁となっている。

機能安全 (Functional Safety) とはシステムの安全を確保する機能 (安全関連系: Safety Related System, SRS) をリスクと許容目標から構成する考え方で、電気・電子・プログラマブル電子 (E/E/PES: Electric/Electronic/Programable Electronic) の機能安全に関する国際規格 IEC 61508[1] が、RT システムを含む様々な分野の安全規格の基礎となっている。

IEC 61508 では、安全度水準 (SIL: Safety Integrity Level) に応じて、ソフトウエア開発時のプロセスおよび手法などについて多くの守るべき基準が定められている。複雑化したソフトウエアにおいて、こうした多くの基準を満たしつつ開発を進めることは多大な労力を要するため、著者らは [2] において、コンポーネント指向開発による、ディペンダブルなシステムの開発手法について 3 種類のアーキテクチャを提案した。

本稿では、機能安全規格を満たすソフトウエアをコンポーネント指向で開発するためのミドルウエアとして高信頼 RT ミドルウエア (Dependable RT-Middleware: d-RTM) を実装する。d-RTM 自体も IEC61508 に定められた全安全ライフサイクルを満たすプロセスに基づいて開発する。プロセス内で定められる d-RTM の基本概念を定義する安全コンセプト、安全コンセプトに基づき導出される安全要求仕様、および d-RTM の利用例について述べる。

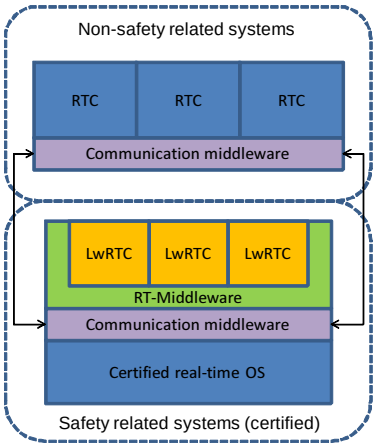

Fig.1 RTC based non-SRS and LwRTC based SRS architecture.

2. 機能安全とコンポーネント指向開発

IEC61508 において、安全機能に関与するサブシステムを安全関連系、それ以外の部分を非安全関連系と呼び、安全関連系は IEC61508 が定める開発プロセスとベストプラクティスに従って開発する必要がある。

従来の単純なシステムでは安全関連系も単純なソフトウエアで実現できるため、規格に従い認証を取得することも比較的容易であった。一方、近年の複雑化したシステムにおいては、従来の手法のみで安全関連系のソフトウエアを構築することは、難しくなりつつある。RT システムにおいても安全関連系が、多数のセンサ情報の処理やアクチュエータによる複雑な機構の制御が含まれ複雑になる傾向があり、そのソフトウエアの安全性を証明し認証を取得することは容易ではない。

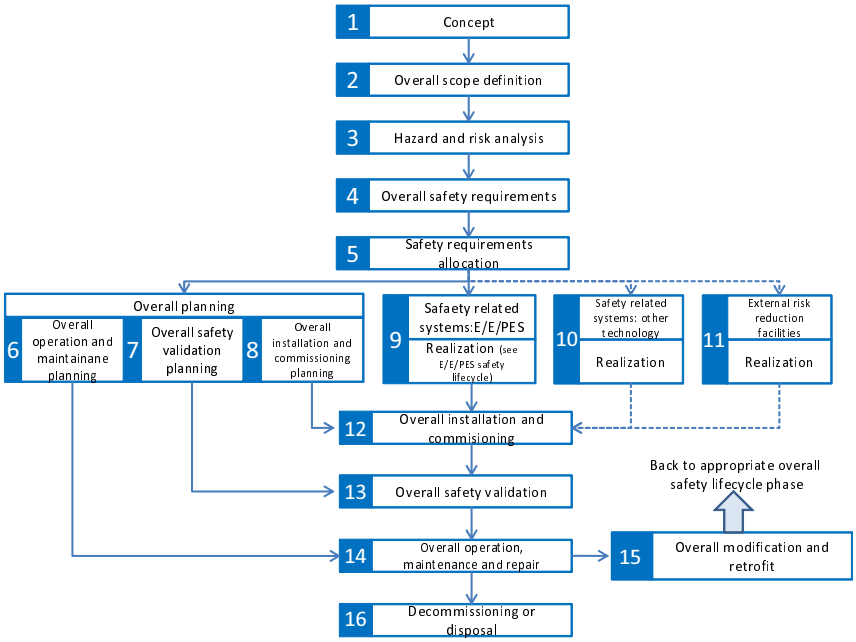

Fig.3 Safety Development Lifecycle. (IEC 1 646/98)

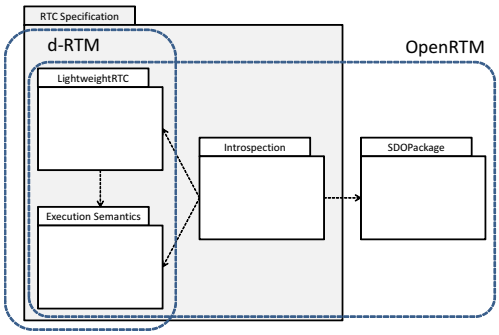

Fig.2 Supported packages of OMG RTC specification by OpenRTM-aist and RTMSafety.

2·1 ディペンダブル RTM (d-RTM)

著者らは、サブシステム間の相互作用を明確化かつ限定する、コンポーネント指向開発手法を安全関連系に適用する方法を提案している [2](図 1)。

著者らはこれまで、ロボットシステムのコンポーネント指向開発を実現するプラットフォームとして RT ミドルウエア: OpenRTM-aist[4], OpenRTM.NET[5] (以下両者をまとめて OpenRTM とする) の開発と、コンポーネントインターフェースの標準化 [3] を行なってきた。標準 [3] は従来の動的なコンポーネントのみならず、LightweigtRTC と呼ばれる静的な結合を想定したコンポーネントインターフェースを含む。これを高信頼コンポーネントのインターフェースとして利用することで、コンポーネント指向の利点を生かしつつ、安全関連系に適用可能なコンポーネントフレームワークが実現できると考えた。

2·2 OMG RTC 標準と d-RTM

安全関連系においては、ソフトウエアは決定論的振る舞いをすることが強く求められるため、一般に静的構造を持たなければならない。OMG RTC 仕様は、図 2 に示すように 3 つのパッケージ (LightweightRTC, Execution Semantics, Introspection) および 1 つの外部パッケージ (SDOPackage) から構成される。著者らの既存の実装である OpenRTM では、これらすべてのパッケージをサポートすることで、実行時の動的な振る舞い (ポートの接続や切断、RT コンポーネントと実行コンテキストのアタッチ・デタッチ等) を実現している。一方、d-RTM では、こうした動的機能は推奨されずかつ不要である。以上から、d-RTM を LightweightRTC および Execution Semantics パッケージにのみ準拠する形で実装することとした。

3. 開発プロセス

IEC 61508 は製品の企画から廃棄まで、図 3 に示すライフサイクルを規定し、それぞれのフェーズで何をどのように管理し、安全を担保する作業を行うべきかを定めている。

これらのフェーズでは、前提条件の定義、アセスメントや分析結果、文章変更履歴などの証拠 (以下 Evidence) の徹底した文書化が求められる。さらに、こうしたプロセスを管理する組織の体制やプロセスについても、厳密に定めることが求められる。

以下、これらのプロセスのうち、図 3 の 1, 4 および 5 を中心に説明する。

3·1 安全コンセプト

対象とするシステムが危険な状態に陥らないための対策や、対策が不能状態に陥らないための処置について、当該システムの設計に関する基本的な概念を定義したものを安全コンセプト (図 3 の 1 (Concept) に該当) と呼ぶ。

d-RTM のコンセプト自体は、従来の RTC が目的としている機能とほぼ同等であるが、これを実現する構成要素として、1) d-RTM Package, 2) 安全機能 Library Package, 3) Network (N/W) Protocol Library の 3 要素を規定する。従来の RT ミドルウエア：OpenRTM-aist と d-RTM との差異を図 4 に示す。

d-RTM Package は LightweightRTC ベースの RTC を駆動するための種々の機能を提供する部分であり、いわゆるミドルウエアに当たる。一方 安全機能 Library は d-RTM に特有な機能で、OS の安全機能の提供、生存監視機能・自己診断機能を提供することで、RTC およびシステム全体の安全性を向上するために利用される。N/W Protocol Library は、安全関連系の RTC 間の通信、および、非安全関連系の従来 RTC との通信を実現する部分であり、安全関連系内に実装するため非常に簡便なプロトコルで構成されている。CORBA と同様に CDR (Common Data Representation) によりマーシャリングが行われるが、RTC 間の固定長のデータ通信のみがサポートされる。

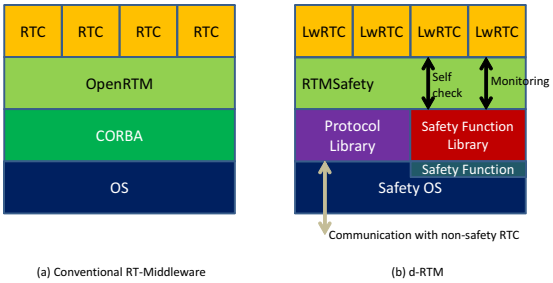

Fig.4 Structure comparison between OpenRTM-aist and d-RTM.

d-RTM の開発にあたっては、安全目標 (10 項目) を掲げ機能設計を行う。その一部を以下に示す。

- SIL3 の機能安全を有するロボットソフトウエアの実現
- コンポーネント間の相互作用を標準化されたインターフェースのみに限定することによる、サブシステム間の影響範囲の限定化
- 複数のタスク同士が衝突したりオーバーランしないよう、RTC フレームワークレベルでの実行タイミングの保証
- RTC の生存監視フレーワークの提供

3·2 安全要求事項

安全コンセプトおよび安全目標から、安全要求事項を定義する。全部で 45 項目の安全要求事項が定義された (図 3 の 4 (Overall safety requirements) に該当)。以下にその一例を示す。

- ユーザに対してテンプレートを提供し、ユーザはコアロジックを埋め込むことで RTC を実装し、d-RTM は RTC を実行する環境を提供すること。
- 実行コンテキストは予測可能なタイミングでロジックを実行するリアルタイム性能を有すること。
- OS に対する拡張性を提供するため、OS 依存部・非依存部を明示的に分割すること。
- 問題・故障発生時の原因解析のためログ機能を有すること。
- ハードウエア・ソフトウエア故障を検出するウォッチドック機能によるリセット機能を有すること。
- IEC61508 において推奨されている手法・技法を使用すること。

3·3 安全要求仕様

開発する製品が、上記の安全コンセプト、および安全要求事項を満たすために、製品に実装されるべき機能に関する要求仕様を、安全要求仕様として規定する (図 3 の 5 (Safety requirements allocation) に該当)。すべての安全要求事項に対して、これを実現するための具体的な詳細仕様を定める。例えば、上述の安全要求事項の第 1 項目「ユーザに対してテンプレートを提供し ･･･」に対しては、9 項目の安全要求仕様が対応する。以下にその一例を示す。

- RTC コアロジックソースコードや RTC 設定情報ソースコードを作成するためのテンプレートを用意する。
- Action テンプレートは、Activity 制御機能が管理する RTC の状態遷移の各アクション実行時にコールバックされるインターフェースを規定する。
- RTObject テンプレートは RTObject 管理テーブルで構成する。
- Data Port テンプレートは Data Port 管理テーブルで構成する。
- Execution Context テンプレートは、タスクとしてメモリに割り当てられるタスクコードと Execution Context 管理テーブルで構成する。

このように、すべての安全要求事項は安全要求仕様によって、より詳細レベルでもれなく満たされ、これに基づいて実装を行わなければならない。

Table 1 d-RTM specification

対象 OS	μITRON, QNX Neutorino RTOS Safe Kernel
実装言語	C 言語サブセット
構成	高信頼 RT ミドルウエア+安全機能ライブラリ
RTC 数	1～ 16 個
データポート数	0～8 個/RTC, InPort/OutPort の総数
接続数	1 つの OutPort に対して 4 接続まで
実行周期	μITRON: 5ms～1s, QNX: 1ms～1s

4. ソフトウエアの実装

d-RTM は機能安全準拠 OS として、μITRON 系 OS および QNX Neutrino RTOS Safe Kernel 上で動作させることを前提として設計している。IEC61508 の SIL3 においては、C 言語のサブセット (コーディングルールの採用、動的メモリ確保の禁止等) を利用することが推奨されるため、これに従い実装言語として C 言語を採用することとした。d-RTM の諸元を表 1 に示す。

d-RTM を利用する開発者は、以下のようなプロセスに従って RT コンポーネントおよびシステム全体を開発する。

- RTC の設計に基づき、特定の ComponentAction コールバックを選択、実装する。
- RTC のデータポートを静的なデータポートテーブルに設定する。
- RTC を実行コンテキスト (EC) にアタッチする。
- 必要に応じてエラー処理を行う安全モニタコールバックを設定する。

図 5 に、Component Action コールバックの一つである on_execute 関数の実装例を示す。C 言語には名前空間がないため、各コールバック関数関数にはコンポーネント名 (ここでは MyRtc_) がプレフィックスとして付加されている。

この例では、`InPort_read()` 関数によりインポートからデータを読み込み、

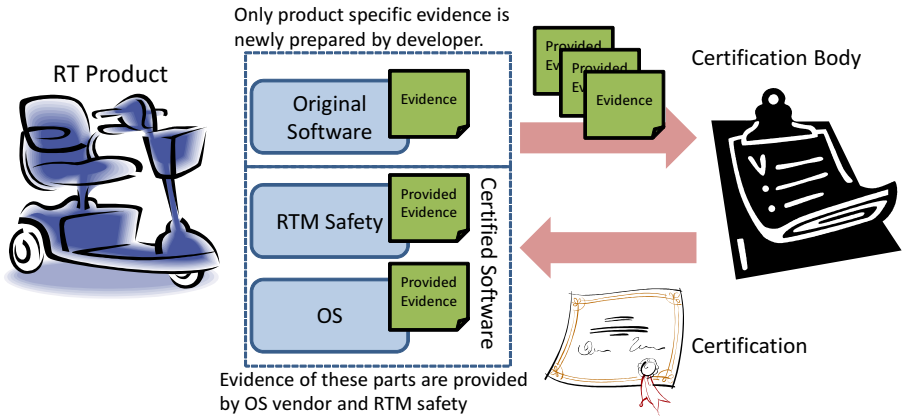

Fig.7 Certification process by using certified OS and d-RTM.

```
ReturnCode_t MyRtc_on_execute( void )
{
  : (中略)
  retVal = InPort_read(&gsDataPort_Input, temp,
                       sizeof(temp), &dataInfo);
  retVal = Marshalizer_demarshalUShort(temp,
                       &position,
                       dataInfo.byteOrder, &data);
  : (中略)
  return RTC_OK;
}
```

Fig.5 An example of on_execute function implementation.

```
ReturnCode_t
MyRtc_create(MyRtc_t* pself,
             const ObjectKey_t* psRtcId,
             const ObjectKey_t* psDataPortIds)
{
  /* RTC ID の設定 */
  pself->psRtcId = psRtcId;
  /* DataPort の設定 */
  pself->psDataPortIds = psDataPortIds;
  /* コールバック関数の設定 */
  pself->onInitialize  = InputRtc_on_initialize;
  pself->onFinalize    = InputRtc_on_finalize;
  pself->onStartup     = InputRtc_on_startup;
  : (略)
```

Fig.6 An example of RT-Component construction function.

`Marshalizer_demarshallUShort()` 関数により unsigend short 型のデータのアンマーシャリングを行い、データポート変数 data に格納している。API は異なるが、実装の仕方は OpenRTM と同様である。

d-RTM は C 言語による実装のため、C++実装である OpenRTM と異なり、言語によるクラスの仕組みが利用できない。したがって 図 6 の初期化関数に示すように、初期化時のコンポーネントのコールバック関数およびデータポートと RTC との関連付けを明示的に行う。

多くの実装はテンプレートとして実装されるため、OpenRTM と同様コンポーネント開発者はコアロジックに集中することができる。

4·1 d-RTM を用いた認証取得

安全認証を受ける上で重要なことは、図 3 の安全ライフサイクルを実施する上で作成される様々な文書 (Evidence) により各フェーズの作業が適切に行われ、フェーズ間に不整合がないことを証明することである。さらに、万一問題があった場合、あるいは改訂により製品を修正する場合のトレーサビリティを保証することにある。

認証済みのソフトウエアを利用し製品を開発する場合、通常ベンダはその Evidence を製品の一部として提供する。図 7 のように、認証済み OS および d-RTM を使用することで、大部分のソフトウエアについてはベンダが提供する Evidence を利用することができる。すなわち、開発者は新たに作成したソフトウエアについての Evidence を作成するにあたり、多くの部分を既存の Evidence を参照することで、労力を大幅に低減することできる。ただし、使用する OS やミドルウエアを含めた製品全体としてのリスク分析は依然として行う必要があることに注意しなければならない。

5. おわりに

本稿では、高信頼なシステムをコンポーネント指向で構築するためのミドルウエア d-RTM の実装について、安全コンセプト、安全要求仕様および実装の実際について述べた。d-RTM を機能安全規格 IEC61508 に定められた開発プロセスに従って Evidence となる文書の作成を行い、定められた仕様に従って実装を行った。

参考文献

[1] Functional safety of electrical / electronic / programmable electronic safety-related systems, IEC 61508, 2005
[2] 安藤 慶昭, 中坊 嘉宏, Geoffrey BIGGS, 大場 光太郎, "コンポーネント指向ディペンダブルシステム開発に向けて – 機能安全の観点からみた RT ミドルウエア –", 計測自動制御学会 システムインテグレーション部門 講演会 2010 (SI2010), pp.87-88, 2010.12
[3] OMG Specification, "Robotic Technology Component Specification", formal/08-04-04
[4] OpenRTM-aist, http://www.openrtm.org
[5] OpenRTM.NET, http://www.sec.co.jp/robot