MDAと連携した組込みソフトウェア開発における仕様要件分析法

小澤 陽平[†] 鶴見 知生[††]
岡本 鉄兵[†††] 小泉 寿男[†]

近年，組込みソフトウェアでもMDA(Model Driven Architecture)の活用のために研究や試使用が行われている．MDAには，モデル記述にUMLを用いたxUMLが存在する．しかし，xUMLはそれだけでは動くモデルの記述が難しい．そのため，モデルを動かすことによる早期開発を目指すxUML本来の利点を発揮できていない．本稿では，xUMLにおける組み込みソフトウェア向けの要件分析法を規定し，分析方法を明確にすることにより，分析結果を定量化する．これにより，xUMLの特徴を活かしたMDAと連携した組込みソフトウェア開発における仕様要件分析法を目指す．

A Proposal of specification design method in development in Embedded software connecting with MDA

Yohei Ozawa[†] Tomoo Tsurumi[††]
Teppei Okamoto[†††] Hisao Koizumi[†]

Recently, the use of MDA is studied on the embedded software development and trial case are reported. In MDA, xUML uses UML for the model description. However, xUML is difficult to describe the executable model. Therefore, the advantage of xUML to develop the softwares in the early period of time by moving the model cannot be showed. This paper defines an analysis method of specification requirement of embedded software, and quantize the result of the requirement analysis by making clear the analysis method. As a result, this paper aims to build a proposal of specification design method in development in embedded software connecting with MDA by using the advantage of xUML.

1. はじめに

組込み機器とは，PC, WS以外にマイクロプロセッサが介在した機器であり，電気ポット，洗濯機等の家電製品やエレベータ等が相当する．組込みソフトウェアは，上記のような機器の制御のために内蔵されたソフトウェアであり，近年は応用分野の多様化より大規模化・複雑化する傾向がある．その一方で製品の開発サイクルの短期化も進んでいる．このことから，現在の組込みソフトウェアでは，構造化手法をはじめとする従来の開発手法では品質と納期の維持が難しく，開発手法の改善が必要となっている．上記のような状況に対応するために，ソフトウェアの設計図で分析・設計・実装の一連の開発を行うMDA(Model Driven Architecture)が主としてビジネスアプリケーション開発の分野で注目されており，様々な研究[1][2]が行われている．また，MDAのモデル記述には，オブジェクト指向開発において標準とされているUMLが用いられるものがあり，このようなMDA向けのUML(Unifiled Modeling Language)[3]として xUML(ExecutableUML)[4][5]が挙げられる．しかしながら，これはUMLを元にしたMDA開発プロセスであるため，必然的にUMLを用いた分析方法の問題点も引き継いでいる．特に，システムの要件を記述するためのユースケース図では，詳細な要件抽出のガイドラインが規定されていないため，分析者によって抽出されるユースケースの粒度にばらつきが生じることがある．また，ユースケース自体の曖昧さ故に，ユースケース図がUMLの他の図のために活用できないという問題も挙げられる．以上から，UMLでは分析・設計工程間にギャップが生じ，以後の設計工程で用いる他の図にシステムの機能要件が正確に反映されないという問題が発生する．xUMLでは，この問題により動くモデルの記述が難しく，結果として早期開発の妨げとなっている．

本稿では，このような問題を解決のためにxUMLのユースケース図(UMLのユースケース図と同様)をいくつかの方法で改良し，組込みソフトウェア向けの要件分析法を規定する．この要件分析法では，xUMLを用いた組込みソフトウェア分析を容易にすると共に，一定の規定した変換規則に従うことでxUMLにおけるPIMに変換することを可能としている．分析方法の改善により，MDA(xUML)プロセスにおける組込みソフトウェア開発において，その本来の特徴である早期開発を十分に発揮できるようにすることを目指す．

†東京電機大学大学院理工学研究科
Graduate School of Science and Engeneering,
Tokyo Denki University
††サイクス(株)
Cyx.Inc
†††ペンタックス(株)
PENTAX Corporation

2. 要件分析法の基本概念

組込みソフトウェアの分析方法の改善を図るために，以下の点に着目し MDA(xUML)におけるユースケース図を改良した要件分析法を規定する．

(1) MDA(xUML)のユースケース分析における組込みソフトウェア向けのガイドラインの明確化

従来のユースケース分析では，分析の方法が明確でなく，そのため完成した図の妥当性に根拠が無かった．また，分析者によってユースケース自体にバラつきも見られた．要件分析法では，記述ガイドラインを規定することでユースケースのバラつきを抑えると共に，分析の目的を明確にしている．また，構成要素の対応関係を明確にしているので，組込みソフトウェアの全体像の把握を容易にしている．

(2) 新要素を導入することによる「要求」と「機能」の関連を明確化

ユースケースのバラつきを抑えるために，要件分析法では最小のユースケースを「1対の入力・出力イベントが対応付けられるもの」と定義する．また，新たにソフトウェアドメインやハードウェアエレメントと言った要素を追加することで，ユースケースがどのような機能(ソフトウェア，ハードウェアの両方)によって実現されているを明確にしている．これにより，従来のユースケース図が他の図に活用できないという問題を改善している．

3. 要件分析法の実現方法

3.1 要件分析法の構成要素

要件分析法では，ソフトウェアの構造の記述とは他に，UML のコラボレーション図とステートチャート図を用いてシステムの振る舞いを記述する．以下に要件分析法で使用される各構成要素である主アクター，副アクター，ユースケース，ソフトウェアドメイン，ハードウェアエレメントについて定義する．

主アクター：システムと相互作用するオブジェクトであり，主としてシステムの利用者であるユーザがこれに相当する．主アクターは，後に述べる副アクターを介してソフトウェアドメインまたはハードウェアエレメントを動作させるイベントを発行する．

副アクター：主アクターからの動作や命令によりイベントを発生させるデバイスである。ボタン、つまみ等がこれに相当する．ただし，タイマー等主アクターを介さずに自動的にイベントを発生させる副アクターもある．

主アクターと副アクターの例を図1に示す．

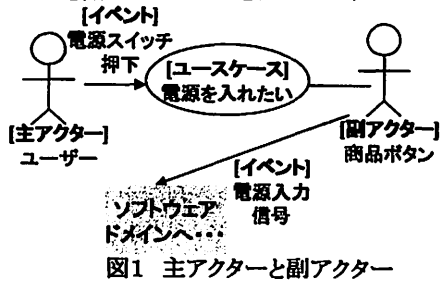

図1 主アクターと副アクター

ハードウェアエレメント：ハードウェアエレメントは，分析対象の組み込み機器を構成するハードウェアであり，ソフトウェアクラスによって制御される．ただし，複数のハードウェアエレメントが1つのソフトウェアクラスによって制御されることもある．ハードウェアエレメントの記述例を図2に示す．ハードウェアエレメントの基本的な記述方法は UML のクラスと同様であり，操作は概略程度で記述する．

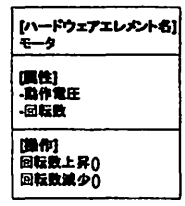

図2 ハードウェアエレメント

ソフトウェアドメイン：ハードウェアエレメントを制御するためのソフトウェア部品である．副アクターから発行されたイベントにより動作し，複数のハードウェアを制御することもある．ソフトウェアドメインの内部を図3に示す．また，本要素はAPI(またはデバイスドライバ)より上位のロジック的な制御(アプリケーションを含む)を対象とする．

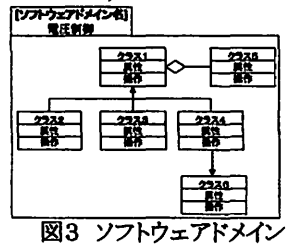

図3 ソフトウェアドメイン

イベント：ソフトウェアドメインを動作させるために与える情報である．イベントには，主アクターから発行されるものと副アクターから発行されるものがあり，前者(基本的にアナログ値)は，副アクターを介して後者(デジタル値)に変換される．また，複数のイベントがあって初めてユースケースが動くということは無い．

3.2 要件分析法による分析フロー

①主アクターの割り当て：主アクターは基本的に UML のユースケース図のアクターと同様であり，開発対象の組込み機器を利用する人間または関連する既存のシステム等が対象となる．

②ユースケース候補の抽出：この工程で抽出されるユースケース候補は，組込み機器で行いたいこと，できることを粒度等を考慮せずに抽出する．このような観点でユースケース候補を抽出し，それぞれを「～したい」というユーザ側の視点で記述する．③以降の手順を経ることにより，他の要素が割り当てられなかったユースケース候補が除外されることで，ユースケース粒度のバラつきを抑えている．

③副アクターの割り当て：副アクターは主アクターを介して組込み機器をどうさせるイベントを発行させるもの(スイッチ，ツマミ等)を対象とし，抽出する．

④イベントの設定：ユースケース候補(後に述べる内部ロジックを示すソフトウェアドメインも含む)を駆動する入力となるイベントを抽出する．抽出時には，主アクターは分析対

象の組込み機器の外部での入力動作，副アクターは内部での情報(または信号)入力とし，ユースケース候補と副アクターを介して対応したものとする．

また，ユースケース候補が主アクターとの間に何も関係を持っていない場合は，そのユースケース候補は副アクターのみで駆動されるものとする．

⑤ハードウェアエレメントの割り当て：ユースケースを実現するために必要となるハードウェアを設計・配置する．これは，後の工程であるソフトウェアドメインの配置を考慮して「ユースケースを実現するために必要で，かつソフトウェアにより制御されるもの」を対象とする．具体的には，モータ，タイマー等が相当する．

ハードウェアエレメントは，基本的に UML のクラス図のクラスと同様な記述方法を採るが，使用するハードウェアが確定していない場合，記述は全体的に概略程度とし，その属性や操作は回転数・モータ逆回転といった抽象的なものとする．

⑥ソフトウェアドメインの設計：前述したハードウェアエレメントを制御するソフトウェア部分の構成を記述する．ソフトウェアドメインの内部は，UML のクラスと同様である．また，ソフトウェアドメイン内のクラスは，あくまでソフトウェアのみを記述ため，ハードウェアがソフトウェアドメイン内クラスとして記述されることはない．

3.3 要件分析法による分析フロー（振る舞いの記述）

ソフトウェアドメインの振る舞いを記述する．後に MDA(xUML)ツールを使用することを考慮して，MDA(xUML)の動的分析と同じ方法である UML のコラボレーション図とステートチャート図を使用する．

上記①～⑥の過程を経て要件分析法による分析図が完成となる．分析の結果，完成する図を図4に示す．

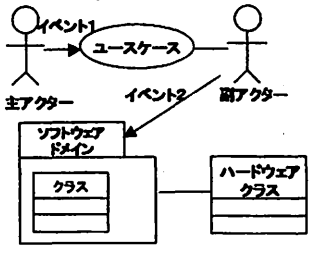

図4 分析図の記述

これによりハードウェアエレメントやソフトウェアドメインが全く割り当てら得なかったユースケース候補は取り除く．この後，図4に対する振る舞いの記述を行い，実装には MDA(xUML)ツールを利用する．

3.4 分析図から xUML(PIM)への変換

前節の過程でとりあえずは分析図が作成されるが，実際には必要に応じて何度かこの工程を繰り返すことになる．その繰り返しを経て完全な分析図が完成する．その後，完成した分析法図から必要な情報を抽出・再構成し xUML(PIM)へ変換する(図5参照)．

・ソフトウェアドメインを元にドメインチャートへ
開発対象を最上位ドメインとし，その下に各ソフトウェアドメインを配置
・ソフトウェアドメイン内のクラスを xUML のクラス図へ
・イベントをユースケースのパラメータへ
ユースケースの description(詳細)として記述する．
・ユースケース図の変更は原則的に変更は行わない
・動的分析のコラボレーション図とステートチャート図はそのまま利用する
・ハードウェアエレメントは，分析図で関連がある(繋がっている)ソフトウェアドメイン内のクラスと関連付けるが変換後のクラス図はソフトウェアドメイン内のクラスとハードウェアエレメントを明確に区別できるよう何かしらの「目印」が必要であると考えている．

以上の過程を図示したものを図5に示す．

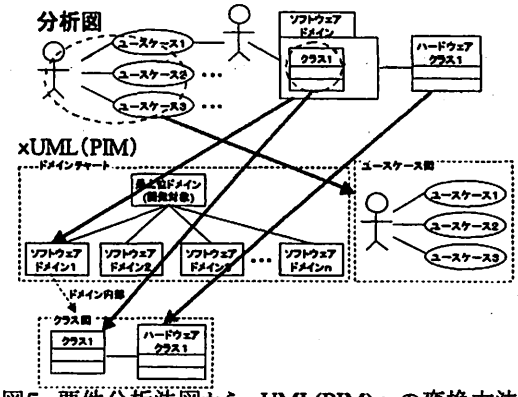

図5 要件分析法図から xUML(PIM)への変換方法

4 要件分析法の適用

要件分析法の評価を行うにあたり，今回はラジコンカーについて要件分析法を用いた開発プロセスを行った．ラジコンカーの要求仕様を表1の通りに想定する．

表1 ラジコンカーの要求仕様

・ユーザの持つ送信機から信号を受信機で受け取り、SW 制御によってステアリング情報とアクセル情報が各サーボに通知され方向転換、速度調整を行う。
・ユーザから自動走行の信号を受信すると、内蔵されているプログラムを読むことで数種類のデモ走行を行う機能がシステムには備わっている。
・デモ走行はユーザから停止信号を受信することで停止する。
・ユーザからの停止信号を受信しなくても、開始してから5分後には自動的に停止する。
・ラジコン本体のバッテリー残量を本体上のランプの点灯によりユーザに通知する。

ラジコンカーを要件分析法で分析する際の手順は，前節の①～⑥の過程を経る．以下，要件分析法を特定の機能要件のみに適用し，その分析過程を示す．

4.1 要件分析法のラジコンカーへの適用

①主アクターの割り当て

今回のラジコンカーでは，これを利用する既存の外部システムの類は存在しない．従って，アクターとして該当するものは利用者(ユーザ)のみとなる．以上から，ラジコンカー

のアクターに「ユーザ」を抽出した.

②ユースケース候補の抽出

表1の要求仕様から粒度等に関係なくアクターである「ユーザが行いたいこと, またはできること」という観点から抽出した. 抽出したユースケース候補は以下の通りである.

- ・曲がりたい(方向転換したい)
- ・走りたい(速度調整したい)
- ・デモ走行を行いたい
- ・デモ走行を停止したい
- ・自動停止できる
- ・バッテリ残量を知ることができる

③副アクターの割り当て

主アクターの動作を組込み機器が認識できる命令(信号など)に変換する入力デバイスを考える. 表1の要求仕様から, 送信機(すなわちリモコン)が副アクターとして挙げられるが, これ以上詳細な仕様がない. そのため, 今回はリモコンの入力デバイスである「ステアリングハンドル, スピードトリガー, デモ走行開始・停止スイッチ」を想定し, 副アクターとした.

④イベントの設定

要件分析法におけるイベントは, 主アクターから発行されるアナログ的な入力量としてのイベントと, 副アクターにより主アクター側のイベントをデジタル値に変換されたものとして発行されるものが存在する. ユースケース候補に対し, これらのイベントが両方存在するかを考慮して設定する. この時, ユースケース候補「曲がりたい」について考えると, その主アクター側イベントは「ハンドル操作量」であり, 副アクターはステアリングハンドルであることから「ステアリング角度」とした.

＊以後, ユースケース候補「曲がりたい(方向転換したい)」における要件分析法の分析過程を示す.

⑤ハードウェアエレメントの割り当て

ユースケース候補を実現するために必要となるハードウェアを割り当て, これを UML のクラスの形態で記述する. ユースケース候補「曲がりたい(方向転換したい)」について考えると, これを実現するために必要なハードウェアは方向転換用のモータ(ステアリング用モータ)であると考えることができる. また, この段階ではステアリング用モータの仕様が未確定であるため, 属性, 操作については概略を記述した.

⑥ソフトウェアドメインの設計

前工程で割り当てたハードウェアエレメントを制御するために必要となるソフトウェアを UML のクラス図の形態で設計する. この時, 制御対象となるハードウェアをクラスとして入れないことに注意する. 表1の要求仕様から「ステアリング情報とアクセル情報が各サーボに通知され・・・」とあるが, サーボ系をドライバまたは API であると見なし, その上位ロジックとして「ステアリング制御」をソフトウェアドメインとした.

4.2 ラジコンカーの振る舞いの記述

①～⑥までの工程では, 主として組込みソフトウェアの構造を記述したが, 本工程ではソフトウェアドメインの振る舞いについて記述する.

ここでは, ソフトウェアドメイン「ステアリングサーボ」の内部のクラス図(ステアリングサーボ内はクラスが一つしかないが)全体の振る舞いについて記述する. 手順としては, コラボレーション図を描いてから, これを元にステートチャート図を記述するため, まずコラボレーション図から考える.

コラボレーション図の記述に当たって必要なのは, 振る舞いをするために必要となる要素である. 今回の例であるステアリングサーボでは, 分析図の静的構造の要素に着目することで, ある程度抽出が可能である. また, ステアリングサーボの振る舞いの概略は, 「ハンドル操作があったらその操作量に応じて専用のモータを回転または逆回転させ, タイヤを右左折できるようにする」と考えることができる. 以上から, ソフトウェアドメイン「ステアリングサーボ」のコラボレーション図を記述した. 記述したコラボレーション図を図6に示す.

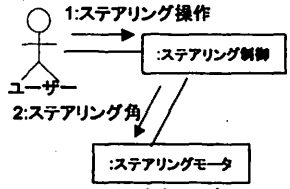

図6 ステアリングサーボのコラボレーション図

次に, ステートチャート図についてだが, これは基本的に図6のコラボレーション図を元に記述する. まず, 状態とイベントの抽出を行うが, 前者は「右折状態」と「左折状態」が必要であるのは明らかである. イベントについては, 抽出した状態から「ハンドル操作量(右または左)」が必要であることがわかる. 以上から, ソフトウェアドメイン「ステアリングサーボ」のステートチャート図を図7に示す.

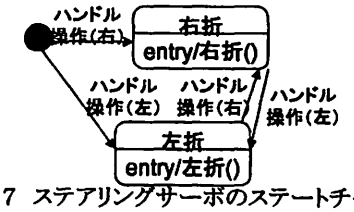

図7 ステアリングサーボのステートチャート図

＊以上の工程でユースケース「方向転換したい」の構成と振る舞いを記述した. 他のユースケースについても同様の工程を踏まえることでラジコンカーの分析図が完成する. 分析の結果完成した分析図(ユースケース「方向転換したい」のみ)を図8に示す.

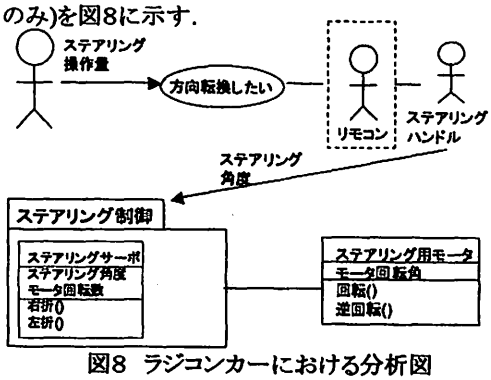

図8 ラジコンカーにおける分析図

(ユースケース「方向転換したい」限定)

4.3 分析図からxUML(PIM)への変換

「4.4」で述べた通りの変換法で，対象をユースケース「方向転換したい」のみに限定した図8のラジコンカーの分析図をxUML(PIM)に変換した(図9参照).

具体的には，以下の過程を経てxUML(PIM)におけるドメインチャート，ユースケース図，クラス図の3つに変換した.

- 分析対象の組込み機器(ラジコンカー)を頂点として，それにかにソフトウェアドメインを配置した
- 分析図の主アクターとユースケースのイベントを消去し，これをxUML(PIM)のユースケース図とした
- 分析図の各ソフトウェアドメイン内のクラスを抽出し，これを再構成しxUML(PIM)のクラス図とした.

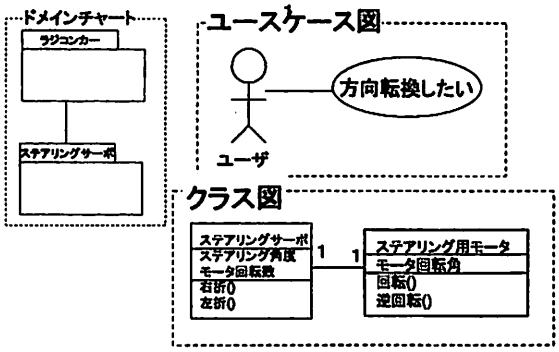

図9 ラジコンカーにおけるxUML(PIM)
(ユースケース候補「方向転換したい」限定)

5. 評価・検討

5.1 プロトタイプ構築とその評価方法

前節「4.1」と「4.2」では，ラジコンカーの要求仕様を想定し，規定した要件分析法を用いて分析を行い，その成果物である分析図からxUML(PI M)へ変換した.

今後，この成果物(分析図から変換したxUML(PIM))を元にMDAツールを用いて実装を行う(MDAツールにはiUMLを使用する予定である).これにより，まず「成果物が動いたか否か」について着目し，規定した要件分析法が利用できるものであるかを評価する.

5.2 要件分析法の評価

本稿では，MDAと連携した要求分析法を規定し，実際にこれを用いて分析を行った.そこで，これまでの分析における効果の，特に以下について検討する.

(1)分析図からxUML(PIM)に変換できたか

図8と図9から，分析図からxUML(PIM)への変換はできた.しかし，この状態では後に述べる一部情報の欠落があり，また，MDAツールでの動作確認も完全ではない.このため，今後は特にプロトタイプ構築による実装を通して現段階の要件分析法で動くものが作られるのかについて検討する.

(2)分析図からxUML(PIM)への変換で欠落した情報
(その理由・それがどのように問題になるか)

「5.3」で図8から図9のように分析図からxUML(PIM)へ変換したが，その際以下の情報が欠落した.

①主アクターが発行するイベント
②副アクター
③副アクターが発行するイベント

①，③は一見欠落してしまっているが，①はコラボレーション図のイベント(図6参照)，③は図8の分析図のものよりもさらに詳細な形用いられている(図7参照).このため，イベントに関しては完全には欠落はしておらず，むしろ詳細な形で利用されていると言える.次に，②については完全に他の図(変換後のxUML(PIM)や振る舞いの記述におけるコラボレーション図やステートチャート図)からは消えてしまっている.理由として，xUML(PIM，一般的なUMLも含む)におけるユースケース図と分析図とで副アクターの定義が異なることが挙げられるが，今後プロトタイプ構築による実装を通してこれが問題になるかについて検討していく.

まとめと今後の課題

MDA(xUML)を組込みソフトウェア向けに分析をいやすくするための要件分析法を規定した.また，ラジコンカーの要求仕様を想定し，規定した要件分析法の適用と評価を行った.

今後は，プロトタイプ構築を行い，これを通して要件分析法のさらなる改善(アクション言語との連携も含む)を行う.また，xUMLのみで分析した場合と要件分析法で分析した場合の組込みソフトウェアの開発プロセス全体の相違(利点，問題点等)について検討する.

参考文献

[1]ANNEKE KLEPPE・JOS WARMER・WIM BAST著，株式会社テクノロジックアート 訳，長瀬嘉秀 監修，日本コンピュウェア協力:「MDA モデル駆動型アーキテクチャ導入ガイド UML を基盤としたオブジェクト指向設計・開発手法」，インプレス，(2003)

[2]細川 卓誠・鶴見 知生・岡本 鉄兵・小泉 寿男:「組込みソフトウェア開発におけるユースケース分析・設計方式の提案」，情報処理学会第140回ソフトウェア工学研究会，140-2 pp9-14 (2003)

[3]オージス総研 著，千藤雅弘 監修:「かんたん UML オブジェクト思考モデリング言語がわかる本」，翔泳社，(1999)

[4]Atephen J.Mellor, Marc J.Balcer: "A Foundation for Model-Driven Architecture Executable UML" (2003)(スティーブ J.メラー・マーク J.バルサー著，二上貴夫・長瀬嘉秀 監訳，Executable UML 研究会 訳:「MDA モデル駆動型アーキテクチャの基礎 ExecutableUML」，翔泳社 (2003))

[5]Leon Starr 著，二上貴夫・長瀬嘉秀 監訳，Executable UML 研究会 訳:「クラス・モデルをいかに作成するか Executable UML 実践入門」，CQ 出版社，(2004)