情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-SLDM-144 No.34
Vol.2010-EMB-16 No.34
Vol.2010-MBL-53 No.34
Vol.2010-UBI-25 No.34
2010/3/27

モデルベース開発における異種モデル間連携基盤の開発

竹辺靖昭† 鈴木康文† 川上真澄†

組込みソフトウェアのモデルベース開発においては，システム仕様記述や検証などの目的に応じて様々な種類のモデリング手法が用いられる．本研究では，複数のモデリング手法を用いるモデルベース開発におけるモデルからモデルへの変換技術の有効性を評価するため，UML モデルから UPPAAL モデルへの変換をおこない両者を連携した開発を支援するツールを QVT の処理系を用いて開発し，開発工数を計測した．また，開発した支援ツールを用いてケーススタディをおこない，UML モデルから UPPAAL モデルへの変換を確認した．

その結果，UML モデルから UPPAAL モデルのようにセマンティックギャップがあるモデル間の変換であっても，変換ツールの開発は 6.2 人月でおこなうことができ，開発工数全体が 31 人月程度のプロジェクトであれば，手動で検証のためのモデルを記述するよりも仕様モデルから検証モデルへの変換ツールを開発した方が開発コストが低減できることが試算できた．

Development of Multiple Modeling Platform for Model Based Development

Yasuaki Takebe†, Yasufumi Suzuki†
and Masumi Kawakami†

In model based development of embedded system software, we often use multiple modeling methods for system specification, verification, etc. In this research, we developed a model translator from UML models to UPPAAL models using a QVT processor and measured the development cost of the translator to evaluate how effective the model-to-model transformation technologies is in model based development using multiple modeling methods. We also did a case study using the translator we developed to test UML model to UPPAAL model translation.

As a result, even for UML models to UPPAAL models translation which has severe semantic gap, we can develop a translator with 6.2 person month work. Using this result, we estimate that if the total development cost of a system is more than 31 person month, we can reduce the development cost by developing a translator from specification model to verification model than writing verification model by hand.

1. はじめに

近年，組込みソフトウェア開発の大規模化，複雑化が進む一方，その開発にかける期間は短縮される傾向にある．そのような背景の中，市場に出まわった製品が，ソフトウェアに起因する不具合によって回収される事例が増加しており，ソフトウェアを含めた製品システム全体の品質低下が大きな問題となっている．出荷後の製品回収は多大なコストが発生し，さらに，企業の信用低下やブランドイメージの悪化を引き起こすため，組込みソフトウェア開発におけるシステムの品質確保が今まで以上に強く求められている．

品質を確保しつつ, かつ, 低コスト, 短期間でソフトウェア開発を実現するために，ソフトウェア工学の見地から様々な手法が提案，研究されてきた．近年では，そうした開発手法の一つとして，ソフトウェアの構造や振舞いを抽象化した表現であるモデルを用いたモデルベース開発(MBD)技術の浸透が進んでいる．

モデルベース開発技術導入の利点の一つとして，製品開発の初期段階であるシステム仕様設計の段階からシステム仕様をモデル化し，そのモデルに基づいたシミュレーション，テスト，形式検証等により仕様の正しさを確認することによって，開発の後段階における手戻りを削減し，効率良く品質を確保することが可能となることが挙げられる．

このような開発工程全体にわたるモデルベース開発の利点を享受するために，開発の前工程で使われるシステム仕様設計のモデルと後工程で使われる検証用のモデルとの整合性をとるためのモデルからモデルへの変換技術が注目されている．OMG は QVT(Query/View/Transformation)[1]というモデルからモデルへの変換のための言語標準を策定しており，標準に準拠した QVT の処理系がツールベンダーやオープンソースにより実装されている．こうした変換技術を使うことにより，システム仕様設計モデルから検証モデルへの変換を行い，両者を用いた開発を支援するツールの作成が容易になることが期待できる．

しかし，現実の開発に用いられるモデルでは，前工程で用いられるモデルと後工程で用いられるモデルでセマンティックギャップがあることが多く，両者の間の相互変換を行うのは容易ではないと考えられる．QVT の処理系などのモデルからモデルへの変換技術によりこうした異種のモデル間の変換器が開発できるかどうかは評価の余地がある．

我々は，モデルからモデルへの変換技術を用いたモデルベース開発技術の有効性を検証するため，QVT の処理系を用いて UML モデルから UPPAAL モデルへの変換を行い，両者を連携した開発を支援するツールを作成した．本論文ではこの変換ツールを

† (株)日立製作所
Hitachi Ltd.

Vol.2010-SLDM-144 No.34
Vol.2010-EMB-16 No.34
Vol.2010-MBL-53 No.34
Vol.2010-UBI-25 No.34
2010/3/27

モデル連携基盤と呼ぶ．UML モデルはシステム仕様設計モデルの，UPPAAL モデルは検証モデル典型的な例である．また，両者の間にはセマンティックギャップがありモデル間の対応は自明ではない．実際にそのツールを用いてケーススタディを行うことによりモデル変換の正しさを確認するとともに，ツールの開発工数を計測し，多くの組込みソフト開発組織にとって現実的な工数でモデルからモデルへの変換器が開発可能であることを確認した．

本論文の構成は以下である．第 2 節では関連研究について述べる．第 3 節では実際のソフトウェア開発でモデル変換が利用されるケースを考察するとともに本研究で開発を行ったモデル連携基盤について述べる．第 4 節では開発したモデル連携基盤を用いたケーススタディについて述べる．第 5 節ではモデル連携基盤の開発工数および QVT 処理系の有効性の考察をおこなう．最後にまとめと今後の課題を述べる．

2. 関連研究

2.1 システム仕様設計モデルと検証モデル

モデルベース開発においてはシステム仕様設計や検証のために様々なモデリング手法が使われている．

システム仕様設計のためのモデリング言語の例としては UML[2]が挙げられる．UML は幅広く使われているモデリング言語であり，様々な図を組み合わせることによりシステムの様々な側面を記述することができる．UML に対応したモデリングツールとしては Eclipse 等のオープンソースのツールや Enterprise Architect[3]等の市販のツールが普及している．

UML は主にシステムが機能として満たすべき機能要求をモデル化する言語であるが，組込みシステムの仕様のモデル化に重要な非機能要求に関しても拡張が行われている．例えば，組込みシステム向けの拡張プロファイル UML profile for MARTE[4][5]では，時間制約を始めとする非機能要件のモデリングが可能になっている．

実際の組込みソフト開発においては，UML のような標準的なモデリング手法だけでなく，製品分野に特化した手法を用いる場合がある．我々は，デジタル家電製品等の仕様記述に適した状態遷移モデルの記述をサポートするコンカレント型開発プロセス支援ツールを開発し，評価を実施した[6].

検証のためのモデリング言語の例としては，モデル検査ツールの SPIN で用いられる Promela の他に，NuSMV，Alloy[7]，UPPAAL[8]等のモデル検査ツールで用いられる言語や VDM++等の形式仕様記述言語がある．

このうち UPPAAL はスウェーデンの Uppsala 大学とデンマークの Aalborg 大学で開発されたモデル検査ツールであり，時間オートマトンを用いて検証対象のモデルを記述することで時間制約を含むモデル検査が可能となっている．組込みソフトウェア開発への適用を考慮した際の UPPAAL の利点としては，モデル検査ツールとして良く用いられる SPIN と比較して，時間制約を記述可能なため，時間制約を含むような検証が可能であることや，GUI を用いたツールであるためモデルや検証結果の可視化が容易であることが挙げられる．

このような形式検証手法を実際のソフトウェア開発に適用するにあたっての問題点として，形式検証技術自体の難解性や，従来のソフトウェア開発に加え，形式検証のために形式検証用言語を用いた検証用の仕様記述が追加で必要となることによる開発コストの増加が指摘されている．

そのため，特別な言語を用いて検証用仕様記述を作成するのではなく，本来作成する必要のあるシステム仕様設計モデルを何らかの手法により変換し検証対象とすることで，形式検証固有のスキルとコストの必要性を低減することを目指した研究もおこなわれている．

Anastasakis らは，UML によるモデル記述を Alloy の記述に変換する UML2Alloy を開発している[9]．しかし，UML2Alloy ではデータ構造を UML のクラス図に属性として定義し，各データが満たすべき制約を UML のクラスに対する制約式(Constraints)として式で与える必要がある．このことは，UML2Alloy で Alloy モデルを生成するためには，Alloy モデル生成に特化した UML モデルの記述が必要であることを意味している．また，制約式を直接記述する必要があるなど，モデル検査ツールに習熟していない開発者が記述するのは困難である．形式上はモデル変換を実現しているものの，実質的にはそのモデル形式を UML の記法に対応しているに過ぎない．

UML から検証用のモデルへの変換としては，この他にも Lassen らが UML を VDM++へ変換する研究をおこなっている[10]．この研究では，UML のシーケンス図を元にモデル変換することでモデル検査を実現しようと試みているが，モデルの一部分しかモデル変換に対応しておらず，自動生成されたモデルを手動で修正しないとモデル検査が実行できないという問題を抱えている．

このように，システム仕様設計モデルから検証モデルに変換するにあたっては，両者のモデルの表現力の違いにより，片方のモデルで表現可能な情報がもう片方のモデルでは表現できない，もしくは表現困難であるということがしばしば発生する．さらに，両者のモデルにおいて表現可能であるような情報に対しても，両者のモデルにおけるモデルが持つセマンティクスの違いにより，異なる振舞いを意味していることがあり，それによりモデル間の齟齬が発生することがある[11].

規格による規定や取扱うツールの制限，モデル化の目的の違い等により，様々な異なるセマンティクスを持ったモデルが利用されている状況にあり，モデルの種類毎に持つセマンティクスの違いは無視できない問題である．

Vol.2010-SLDM-144 No.34
Vol.2010-EMB-16 No.34
Vol.2010-MBL-53 No.34
Vol.2010-UBI-25 No.34
2010/3/27

2.2 モデルからモデルへの変換技術

このように様々なモデル間の変換を記述し自動化するために，OMG は QVT というモデルからモデルへの変換のための言語標準を策定している．また，標準に準拠した QVT の処理系がツールベンダーやオープンソースにより実装されている．

例えば，ドイツ ikv++ techn ologies 社は medina[12]というツールで QVT 標準を実装している．medini はモデルの構成管理ツールとモデル間の変換器を統合したツールである．MOF(Meta-Object Facility)[13]に準拠したメタモデルを定義することにより，複数種類のモデル間の変換や構成管理をおこなうことができる．

モデルからモデルへの変換記述言語処理系としては, この他にも Eclipse 関連のプロジェクトである Model to Model T ransformation プロジェクトにおいて，QVT に準拠した処理系である QVTO や, INRIA が中心で開発した ATL(Atlas Transformation Language) などが実装されている．

こうした変換技術を使うことにより，モデルからモデルへ変換をおこなうツールの開発が容易になることが期待できる．しかし，著者らが知る限り，前述のセマンティックギャップを含むような現実的なモデル変換器を QVT 処理系で開発し，コスト低減効果を評価した報告はない．これらの技術がどこまでモデル変換ツールの開発を容易にするかということについては評価の余地がある．

3. モデルベース開発における異種モデル間連携

3.1 ソフトウェア開発でモデル変換が利用されるケースの考察

現実的な状況を想定してモデルからモデルへの変換技術の有効性の評価をおこなうため，実際のソフトウェア開発ではどのような形態でモデル変換が利用されるかを考察する．

本研究で想定するモデル変換の利用形態を図 1 に示す．

実際のソフトウェア開発においては，上流工程においてシステム仕様を設計し，それを段階的に詳細化し, 実装するというのが一般的である. システム仕様の段階では，全体的な方式検討がおこなわれ，システム全体の振舞いが高い抽象度で定義される．

次に，製品を構成する各部品や，その部品上で動作するソフトウェアに対して，それぞれ機能設計を実施する．この機能設計段階においては，各部品のハードウェアによる制約等がより具体化され，それに従ってソフトウェアが実装すべき機能も明確化される．

この仕様設計モデルから機能設計モデルへの詳細化は，一般に機能設計モデルの方が詳細であり，仕様設計モデルに含まれていない情報を含んでいるため，単純なモデル変換により自動化することは困難である．このため，設計者の人手によるモデル作成がおこなわれ，モデル間の不整合や不具合の原因となる．

そこで本研究では，次のようにモデル変換技術を用いることにより仕様設計モデルと機能設計モデルの整合性の検証がおこなわれると想定する．

まず，設計モデルから検証モデルへの変換器を作成し，設計モデルから検証モデルへの変換を自動化する．これにより設計モデルと検証モデルとの整合性を保つことができる．次に，仕様設計モデルと機能設計モデルをそれぞれ検証モデルへと変換し，モデル検査ツール等の検証器により，同一の検証項目に対して同じ結果を満たすことを確認する．このようにして仕様設計モデルと機能設計モデルの整合性を検証することができる．

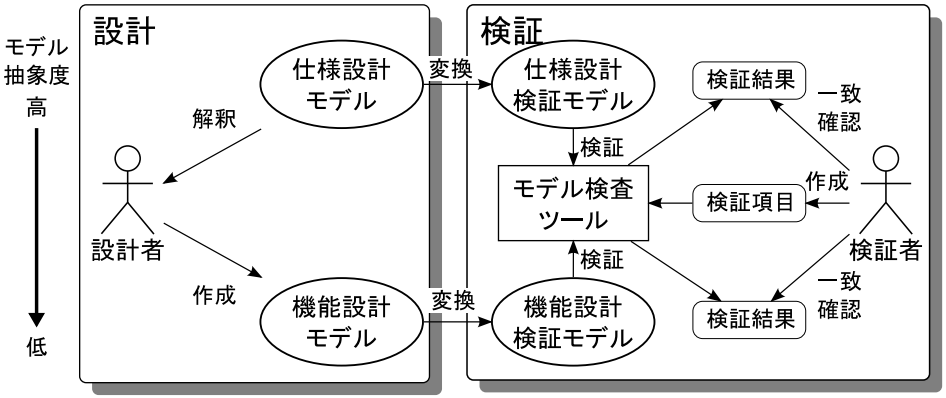


図 1　本研究で想定するモデル変換の利用形態

3.2 QVT 処理系による異種モデル間連携基盤の構築

本研究では, ドイツ ikv++ technologies 社の medini を使用し, UML モデルと UPPAAL モデルとの間の相互変換を実現するモデル間連携基盤を開発することにした．本研究において開発した異種モデル間連携基盤による各ツールの構成図を図 2 に示す.

Enterprise Architect で設計された UML モデルから UPPAAL で検証できるモデルに変換する機能を開発する．図 3 は，図 2 において medini 上に新規に開発した部分の詳細なソフトウェア構成図である．

新規開発部分は連携させるツール(Enterprise Architect および UPPAAL)の制御およびモデルの読出し，書出しを実現するツールコネクタ部分と，モデルをどのように変換するかを記述するモデル変換記述からなる．

ツールコネクタは, 各ツールが出力するモデルを medini が扱えるように各ツール用

Vol.2010-SLDM-144 No.34
Vol.2010-EMB-16 No.34
Vol.2010-MBL-53 No.34
Vol.2010-UBI-25 No.34
2010/3/27

のメタモデルに従ったモデルに変換する．メタモデルに従って変換することにより，medini が提供するバージョン管理等のモデル管理機能を利用することができる．

モデル変換記述に関しては，QVT に従って変換ルールを記述した．本研究では，UML 中のステートマシン図と UPPAAL モデルとの変換ルールを QVT で記述し，両者の変換を実現した．

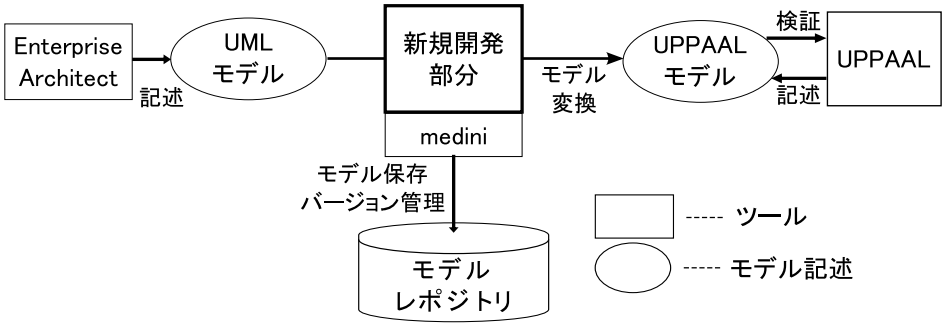


図 2　モデル間連携基盤における各ツールの構成図

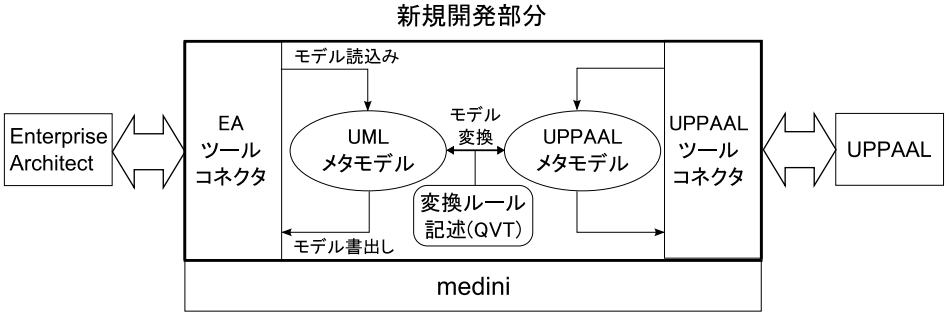


図 3　新規開発部分のソフトウェア構成図

3.3 モデル間の表現力およびセマンティクスの相違に起因する問題とその解決手法

本研究の対象である UML のステートマシン図と UPPAAL モデルはともに状態遷移図であるが，状態マシン間(UML においてはステートマシン間，UPPAAL においてはプロセス間を意味する)の相互作用であるシグナルに関して，その振舞いは大きく異なる．

具体的には，UML におけるシグナルは非同期シグナルであるのに対し，UPPAAL のシグナルは同期シグナルである．また，UML においてはシグナルの送信先を指定するのに対し，UPPAAL では同じチャンネルを共有する複数のプロセスがシグナルを受け取ることが可能である．

これらのセマンティクスの違いを吸収するために，本研究では，UPPAAL 上で非同期シグナルをエミュレートするシグナルバッファ機能を開発し，UML ステートマシン図から UPPAAL に変換する際に，シグナルバッファ機能を付加した上でモデルを出力することとした．

具体的には，シグナルを溜めておくことができるバッファのモデルを作成し，UPPAAL 上の各プロセスに対するシグナルは，すべてバッファを経由させてやり取りすることとした．

これにより，送信側はバッファに対してシグナルを送信することにより UML ステートマシン図のセマンティクスに合わせたモデル変換が可能となった．

3.4 モデル間連携基盤の開発コスト検討

本研究において提案する QVT 処理系上に構築するモデル連携基盤の開発にかかるコストは，大きく以下の 3 種類のコンポーネントの開発コストに分類することができる．

- 連携ツールとの間のデータのやり取りおよびツール動作の連携を実現するツールコネクタの開発
- 異種モデル間の変換規則の検討および設計
- 設計した異種モデル間変換規則の QVT による実装

ツールコネクタの開発にあたっては，既存のツールに対しては一度ツールコネクタを開発することによって，ソフトウェア資産を再利用することが可能である．また，新規開発ツールに対しても，XML 形式での入出力機構を持たせることにより，medini を用いた場合には XML 取り込み機能を用いることで比較的容易にツールコネクタの開発が可能となる．

異種モデル間の変換規則の検討および実装コストに関しては，前述のモデル間の表現力およびセマンティクスの相違の解決手法の開発にかかる工数が主なコストである．

4. モデル連携基盤を用いたケーススタディ

4.1 対象システムの仕様

本研究において開発したモデル間連携基盤を，図 4 に示すような単純化された搬送

Vol.2010-SLDM-144 No.34
Vol.2010-EMB-16 No.34
Vol.2010-MBL-53 No.34
Vol.2010-UBI-25 No.34
2010/3/27

システムに対して搬送時間の分析という課題を試行することにより評価した．

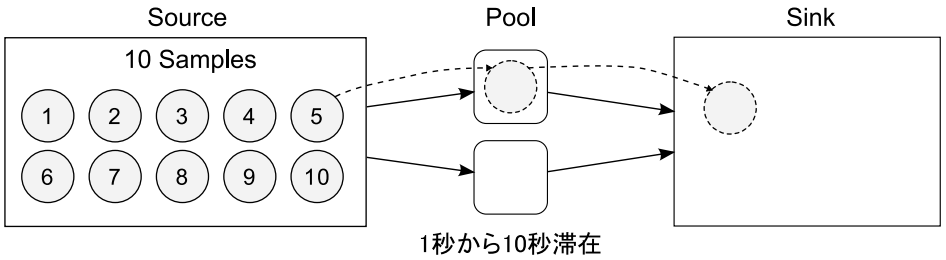


図 4　サンプル搬送課題の模式化

図 4 において，ソースに存在する複数のサンプルをプールを経由してシンクに搬送することを考える．各サンプルはプールにおいて，それぞれのサンプルに対して指定された時間滞在しなければならない．また，ひとつのプールには同時にひとつのサンプルしか滞在することができない．

以上のような制約条件において，様々なサンプルの入力条件に対して，すべてのサンプルがシンクに到達するのに必要な時間を分析することを課題とする．

本ケーススタディにおいては，10 個のサンプルがそれぞれ割当てられた番号と同じ秒数(1 秒から 10 秒)滞在するという条件であり，また，サンプルが搬送される順番は任意であるとする．

4.2　システム仕様設計検証モデルの記述

搬送機構の性能を分析するにあたり，まず，全体の仕様を検討する段階では，搬送されるサンプルの動きをモデル化することによる分析の方が容易であるため，UPPAAL でサンプルの動きをモデル化し搬送方式を検証した．なお本ケーススタディにおいては，システムの仕様が非常に単純であるため，図 1 に示す形態とは異なりシステム仕様設計検証モデルは手動で作成した．

仕様検討において用いられるサンプルの動きモデル(UPPAAL モデル)を図 5 に示す．図 5 では，サンプルに対して，そのサンプルが現時点でどの場所に存在しているかを状態として表わしている．初期状態にはソースに存在し，プールが空いていればプールに移動，プールではサンプルに指定された指定滞在時間(st)だけ滞在した後，シンクに移動するというモデルになっている．このようなモデルをクラス(UPPAAL ではテンプレートと呼ぶ)として，サンプルの数だけインスタンス(UPPAAL においてはプロセス)を生成させ，モデル検査をおこなう．

本ケーススタディにおいてはサンプル数は 10 を想定しているため，10 個のインスタンスを並列動作させることになる(図 6)．

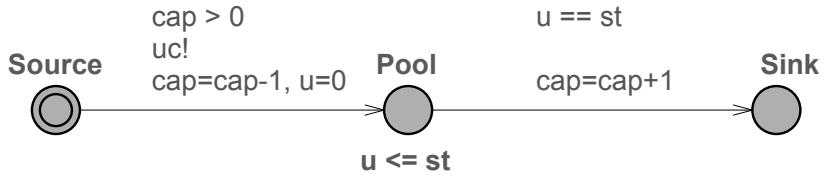


図 5　サンプルの動きの UPPAAL によるモデル化

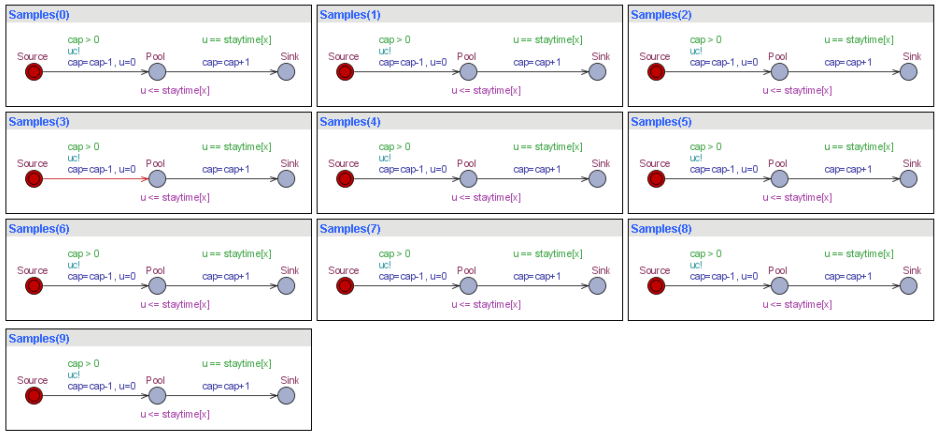


図 6　10 個のサンプルを並列動作させたシミュレーション

4.3　機能設計モデルの記述とモデル連携基盤による変換

次に，機能設計段階においては，装置を構成する各部品について，その部品の動作機構をモデル化することにした．仕様設計モデルにおいては搬送されるサンプルを対象に仕様を記述したが，機能設計段階においてはモデル化の対象は搬送システムを構成する個々の部品になる．これにより，個々の部品のハードウェア制約等を考慮し，搬送に必要な情報をどのように外界や他の部品から手に入れるかなどをモデル化する必要がある．

Vol.2010-SLDM-144 No.34
Vol.2010-EMB-16 No.34
Vol.2010-MBL-53 No.34
Vol.2010-UBI-25 No.34
2010/3/27

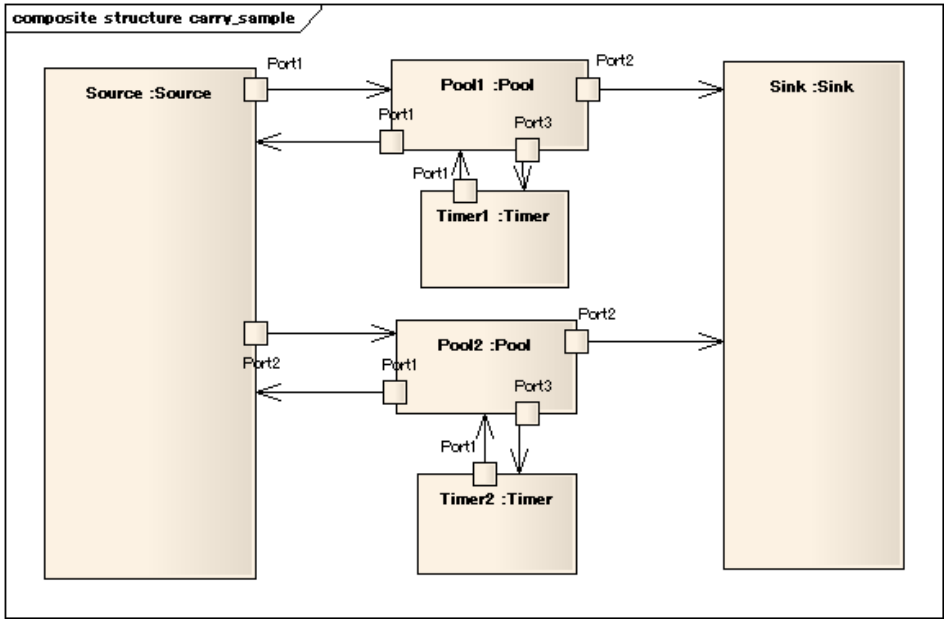
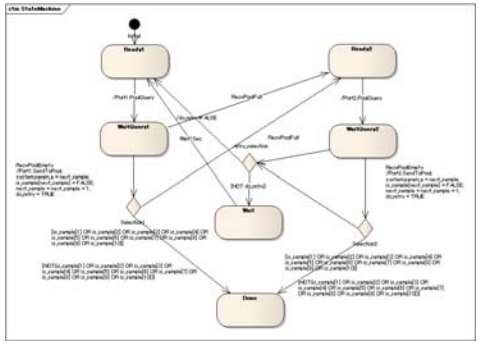


図 7　UML による機能設計モデル(コンポジット構造図)

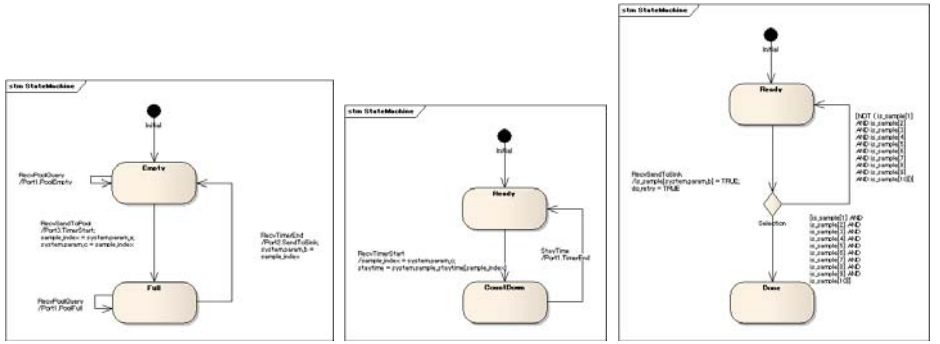
サンプル搬送課題に対して，ソース，プール，シンクの各部分を UML を用いて機能設計したモデルを図 7，図 8 に示す．本ケーススタディにおいては，プールに関しては Pool と Timer という 2 つの部品に分割して設計した．

図 7 の UML コンポジット構造図においては，各部品の構成および部品間のシグナルのやり取りを定義している．図 8 中の各部品に対して，それぞれその内部動作をステートマシン図で表わしたのが，図中の(a)，(b)，(c)，(d)である．

次に，図 7，図 8 に示した UML で記述された機能設計モデルを，開発した異種モデル間連携基盤を用いて UPPAAL モデルに変換した．得られた UPPAAL モデルの一部を図 9 に示す．図 9 のモデルは，異種モデル連携基盤によって自動的に処理されることを想定しているため，開発者が見るのに適した配置にはなっていない．しかし，一般に開発者が手動で記述するには複雑なモデルが自動的に生成されていることが分かる．



(a) Source



(b) Pool　(c) Timer　(d) Sink

図 8　UML による機能設計モデル(ステートマシン図)

4.4 仕様設計モデルの機能設計モデルの整合性検証

まず，図 6 に示した仕様設計検証 UPPAAL モデルを用いてシミュレーションすることにより，搬送システムの最悪搬送時間を検証した．その結果，最悪の場合においても 32 秒以内に搬送が完了すること，および，最悪ケースにおける搬送順序がサンプル番号で 1，2，3，4，5，7，6，9，8，10 の順に搬送することなどが検証できた．

次に，異種モデル間連携基盤によって自動変換された機能設計検証 UPPAAL モデルを用いて同じ検査項目を検査した．その結果，同じ結果が出ていることが確認できた．

このようにして本ケーススタディにおける仕様設計モデルと機能設計モデルとの整合性が確認できた．

Vol.2010-SLDM-144 No.34
Vol.2010-EMB-16 No.34
Vol.2010-MBL-53 No.34
Vol.2010-UBI-25 No.34
2010/3/27

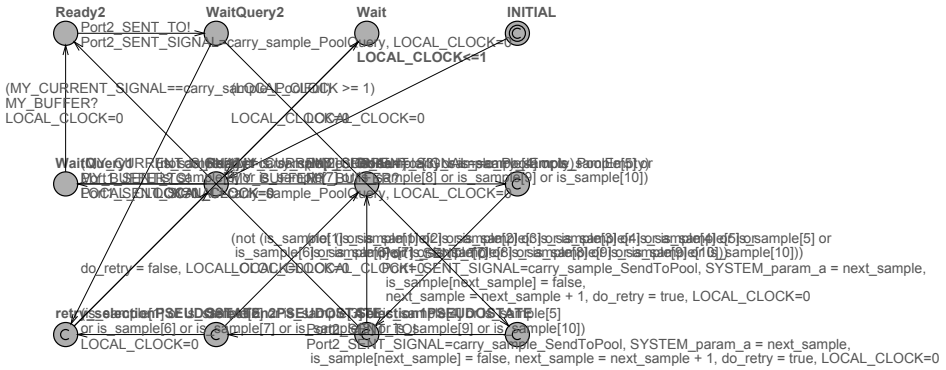


図 9　モデル間連携基盤によって自動生成された UPPAAL モデルの例

5. 評価

開発した異種モデル間連携基盤の開発工数およびその効果について評価する．開発に必要な工数としては，3.4 節に挙げたように，ツールコネクタの開発，モデル変換規則の検討，モデル変換規則の QVT による実装の 3 種類がある．

まず，ツールコネクタの開発に関しては，Enterprise Architect とのツールコネクタの開発に 0.5 人月，UPPAAL とのツールコネクタの開発に 0.7 人月を要した．これらの工数には，連携する各ツールの学習および仕様の調査を含んでおり，主な工数はそれらの調査であった．

実際には，ツールコネクタに関しては，連携するモデリングツールの実装に習熟していれば容易に開発が可能であると考えられる．我々が開発のコンカレント型開発プロセス支援ツール用のツールコネクタの試作を実施しところ，既存のライブラリを活用することにより 1 人日のみでの開発が可能であった．

次に，モデル変換規則の検討および設計は 4.0 人月，QVT での実装は 1.0 人月を要した．具体的には，両者を合わせて 2 名の開発者により 2.5 カ月を要した．したがって，異種モデル間連携基盤全体の開発工数は 6.2 人月であった．

モデル変換規則の検討および設計は変換対象のモデリング手法の理解に必要な期間を含んでいるため，実装言語にあまり依存せず，短縮できないことが予想される．設計の 4.0 人月と比較して実装は 1.0 人月と短期間であるのは，QVT の実装言語としての生産性の高さを示すものと考えることができる．

モデル変換ツールをこれだけのコストをかけて開発することの有効性は以下のように考えることができる．

まず，比較のため，今回のケーススタディにおいて作成した機能設計モデルに対する検証用モデルを手動で記述したところ，機能設計のモデルの記述と比較して 1.8 倍の工数を必要とした．また，組込みソフトウェア産業実体調査[14]によると，ソフトウェア詳細設計は組込みシステム開発の工数全体のうちの 11.1%を占めることがわかる．これより，詳細設計モデルに対して手動で検証用のモデルを記述するのに必要な工数は組込みシステム開発工数全体の 20.0%程度と推測できる．

以上の試算により，人員面でのコストのみを考慮した場合，システム開発工数の 20%が 6.2 人月以上，つまりシステム開発工数が 31 人月以上の規模のプロジェクトの場合には，本研究のようにモデルからモデルへの変換技術を用いた方が開発コストが削減できると推測することができる．

以上の試算は medini および UPPAAL の価格を考慮していないが，Eclipse プロジェクトによるモデルからモデルへの変換技術等のオープンソースの実装を活用することで試算に近いレベルまで費用の低減が可能であると考えられる．

6. おわりに

6.1 結論

本研究では，現実的なソフトウェア開発ではどのような形態でモデル変換が利用されるかを検討し，そのような状況下で QVT の処理系を用いてモデル連携基盤を開発し，その開発コストを計測することにより，モデルからモデルへの変換技術の有効性を評価した．著者らが知る限り，モデルからモデルへの変換技術によるモデル変換器の生産性向上によるコスト低減効果を定量的に評価した報告はこれまでなかった．

変換対象のモデルとしては，システム仕様モデルとしては UML，検証モデルとしては UPPAAL を選択した．これらの間にはセマンティックギャップがあり，モデル間の対応は自明ではない．

また，実際にケーススタディにおいて UML モデルを変換し UPPAAL で検証することにより変換ツールの有効性を確認した．

開発工数の計測においては，モデル連携基盤全体のシステム構成を明確化し，QVT を用いた変換器を含む各構成要素の開発工数を計測している．これにより，実際にモデル変換による開発支援ツールを開発する場合に有効な開発コストが計測できていると考えられる．

開発コスト計測の結果，UML から UPPAAL へのモデル変換ツールの開発コストは 6.2 人月であり，システム開発工数が 31 人月以上の規模のプロジェクトの場合には，

Vol.2010-SLDM-144 No.34
Vol.2010-EMB-16 No.34
Vol.2010-MBL-53 No.34
Vol.2010-UBI-25 No.34
2010/3/27

本研究のようにモデルからモデルへの変換技術を用いた方が開発コストが削減できると推測することができた．

6.2 今後の課題

本研究では，現実的なソフトウェア開発を想定してモデルからモデルへの変換技術の有効性を評価しているが，モデル連携基盤の開発には以下のような追加のコストがかかることが想定でき，この評価が今後の課題である．

まず，本研究では UML と UPPAAL を対象として評価をおこなっているが，モデリング手法の記述力やモデル間のセマンティックギャップによりモデル変換器の開発コストは当然変わってくる．組込みシステムに用いられる様々なモデリング手法に対応するためには，様々な組合せに対して評価をおこなう必要がある．

次に，本研究でのモデル連携基盤は仕様設計モデルまたは機能設計モデルから検証モデルへと一方向の変換を考慮した．しかし，現実の開発においては，検証モデルにおいて問題を発見し，それを仕様設計モデルまたは機能設計モデルに反映したいという状況が頻繁に発生すると思われる．こうした場合に，現在のモデル連携基盤では検証モデルに修正を加えても元となる設計モデルに反映する仕組みがないため，開発者が手動で修正を行う必要がある．現実の開発に対応するためには，検証モデルから設計モデルへの逆変換機能も開発する必要があることが考えられる．このような機能を追加するとモデル変換規則の設計や実装の工数が増加する．

本研究で開発した UML モデルから UPPAAL モデルへの変換器は，モデル変換記述言語である QVT を用いても 5.0 人月という工数を要しており，複雑なソフトウェアであることは間違いない．一般にセマンティックギャップがあるモデル間の変換器はこのように複雑なものになると考えられる．一方で変換器にバグがあると検証結果に重大な影響を及ぼす．本研究においては，ケーススタディを実施することで変換器の正当性を確認したとしたが，現実のソフトウェア開発に適用するにはより厳しい検証が必要になる．こうした変換器の検証コストの評価，および検証コスト低減手法の検討も今後の課題である．

参考文献

1) Object Management Group: OMG’s Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT), http://www.omg.org/spec/QVT/
2) Object Management Group: UML the unified modeling language, http://www.uml.org/
3) Ent erprise Architect: http://www.sparxsystems.com.au/
4) S. Gerard, D. Petriu and J. Medina: MARTE: a new standard for modeling and analysis of real-time and embedded systems. *In Proc. of Euromicro Conf. on Real-Time Systems* (ECRTS 07), Pisa, Italy, Jul. 2007.
5) Object Management Group: UML profile for MARTE, http://www.omgmarte.org/
6) 川上真澄, 大脇隆志, 小泉忍: 組込みシステムの擦り合わせ型並行開発プロセスに対するモデルベース開発技法の適用, 組込みシステムシンポジウム 2009 (ESS2009).
7) D. Jackson: *Software Abstractions: Logic, Language, and Analysis*, MIT Press, 2006.
8) UPP AAL: http://www.uppaal.com/
9) K. Anastasakis, B. Bordbar, G. Georg, and I. Ray: UML2Alloy: a challenging model transformation, *In ACM/IEEE Intl. Conf. Model Driven Engineering Languages and Systems* (MoDELS 2007), Vol. 4735 of LNCS, pp. 436-450, 2007.
10) K. B. Lassen and S. Tjell: Model-based requirements analysis for reactive systems with UML sequence diagrams and coloured petri nets: *Innovations in Systems and Software Engineering*, Vol.4, No.3, pp. 233-240, August 2008.
11) P . Stevens: Bidirectional model transformations in QVT: semantic issues and open questions, *In ACM/IEEE Intl. Conf. Model Driven Engineering Languages and Systems* (MoDELS 2007), Vol. 4735 of LNCS, pp. 1-15, 2007.
12) medini: http://www.ikv.co.jp/
13) Object Management Group: OMG’s Meta Object Facility, http://www.omg.org/mof/
14) 組込みソフトウェア開発力強化推進委員会: 2008 年版 組込みソフトウェア産業実体, 調査報告書, 経済産業省, 2008.