3J-01

要求分析段階における モデル検査技術を用いた設計制約検証の自動化

池田　彩恵†　　松浦　佐江子‡

芝浦工業大学　システム理工学部　電子情報システム学科†‡

1. はじめに

複数機器間での通信によりサービスを提供する複合システムの開発では，通信で正常にデータをやり取りするために，通信発生のタイミングや通信時間，通信相手という設計制約を各機器が満たし同期をとる必要がある．設計，実装段階でこのような設計制約を満たさないことが判明すると作業の手戻りが生じるため，システム全体の流れを定義する要求分析段階で設計制約が満たされているか検証を行うことが重要である．システムの全体の流れは，UML モデルのワークフローで定義し，サブシステムの作業の流れと連携を確認することができる．しかし，サブシステムは並列動作するため，設計制約により各サブシステムが取り得る膨大な数の状態から機器が同期しない状態を人の目で確認することは困難である．そこで，UML モデルのフロー，シグナル送受信をモデル検査ツール UPPAAL モデルのプロセス，チャネルに対応付けて自動変換し，並列動作を検査して，設計制約を満たすことを検証する手法を提案する．本稿では，芝浦工業大学の PBL 課題である荷物自動搬送システムを事例として本手法の有効性を議論する．

2. 複合システムの要求分析

複合システムの要求分析では，システム全体の流れとサブシステム間の連携をワークフローで定義する．ワークフローは，システムに関わるサブシステムがシステムの要求を満たすように，システム全体のデータのやり取りや機器の振舞いをアクティビティ図で示した図である．アクティビティ図は業務や処理の手順を定義する UML モデルである．ワークフローは図 1 の左側のように，システム利用者やサブシステム毎にパーティションで区切り，その内部に各処理手順をフローで記述する．通信処理はシグナル送受信，待機時間等の時間条件は時間イベント受理アクションを用いることで，サブシステム間の連携を定義できる．ワークフローではサブシステム間のデータの授受を定義するため，サブシステム固有の振舞いはアクションノードで記述し，詳細はサブシステム毎にアクティビティ図のユースケース記述で表現する．

UML は，自然言語を用いてシステムの流れを自由に，静的に記述することができる．しかし複数機器間で通信を行うシステムを定義する場合，各機器が行う通信処理がどのタイミングで発生するか，どのデータを通信内容としてどの機器を相手に授受するかを記述する明確な要素が定められていないため，機器が同期を取るかの検証が困難である．並列動作する各機器が取り得る状態の数は膨大であり，静的に書かれたモデルを用いてそれらの状態を人の目で確認すると見落としが発生する可能性がある．またサブシステム固有の振舞いの中に通信処理が含まれる場合，固有の振舞いはアクションノードで記述されるため，通信処理の有無はユースケース記述を参照する必要がある．

3. 設計制約検証手法

3.1 モデル検査ツール UPPAAL

モデル検査技術は，システムの振舞いを有限状態遷移図に表し，検査したい性質を時相論理に基づく論理式（検査式）で記述してその性質を網羅的に検査する技術である．モデル検査ツールの UPPAAL [1]はシミュレータとベリファイアの機能を持ち，同一の挙動をするシステムを 1 つのプロセス，通信路をチャネルで定義して並列プロセスの同期を表現できる他，時間制約の表現と検査ができる．プロセスの挙動の定義をテンプレート，プロセスが取る 1 つ 1 つの状態をロケーション，状態間の遷移をエッジで定義する．モデルは GUI により編集し，XML ファイル形式で保存される．検査式 A[](not deadlock)によりデッドロックの検査を行うと，すべての実行パスでどれだけ時間が経過してもいずれのプロセスも遷移できない状態の有無を確認できる．システムを UPPAAL モデルで記述することで，並列処理の検証が可能となるが，UPPAAL はシステムの状態遷移を定義するため，処理手順を定義するワークフローを初めから UPPAAL で記述することは困難である．また，手動で UML モデルから UPPAAL モデルに変換する場合，図 1 のようにワークフロー内の各フローに対してテンプレートを書き出す必要があり，手間がかかる．

3.2 検証方法

UML モデルを UPPAAL モデルに対応付けて自動変換し，デッドロックの有無を検査して，設計制約を満たすか検証する手法を提案する．サブシステム間の連携を検証するため，シグナル送受信で記述したサブシステム間の通信処理に着目し，これを抽出して変換を行う．モデル要素の対応を表 1 に示す．パーティション内の開始ノードからフロー終了ノードまでを処理の一連の流れとみなし，この一連の流れ 1 つ 1 つをテンプレートに置き換える．通信処理は，シグナル送受信のステレオタイプに通信相手のパーティション名を記述し，アクション名に通信

Automatic Verification of Design Constraints on Requirements Analysis Model using Model Checking

†Sae Ikeda ‡Saeko Matsuura

†‡Department of Electronic Information System, Collage of System Engineering and Science, Shibaura Institute of Technology

内容を記述する．これらの記述が通信相手と通信内容の設計制約となる．ある 2 つのサブシステムの間で行われる，同じ内容の通信内容を扱う通信を，1 つの通信処理とする．UPPAAL モデルでは，送信または受信を行った状態をロケーションで表し，1 つの通信処理に対して 1 つのチャネルで通信路を表す．時間イベント受理アクションに記述した待機時間は，時間変数を用いて遷移の条件に置き換える．図 1 は，左のワークフローから，通信処理を含むフローの通信処理部分を抽出して，右のテンプレート群に変換している．検証の手順は以下の通りである．

(1) ワークフローを astah[2]で作成する．
(2) (1)の astah ファイルをモデル自動変換ツールに入力する．変換が完了すると UPPAAL モデルの XML ファイルと，UML モデルのどの要素が UPPAAL モデルのどの要素に変換されたかを一覧にした対応表が出力される．
(3) 出力された XML ファイルを UPPAAL で開き，ベリファイアの検査式に A[](not deadlock) を入力し，検査を実行する．UPPAAL はすべての実行パスに対して自動で網羅的に検査を行う．
(4) UPPAAL に検査結果が表示される．赤字で「属性は満たされませんでした」とある場合，シミュレータを開く．シーケンス図と時間オートマトン図により，反例に到達した状態が表示される．
(5) (2)で出力された対応表を参考に，元の UML モデルと比較して修正を行う．

表 1. モデル要素の対応

ワークフロー	UPPAALモデル
パーティション、フロー	テンプレート
シグナル送受信 <<signal sending>> <<配達担当ロボット>> 荷物を送る <<signal receipt>> <<中継所>> 荷物を受け取る	チャネル、ロケーション
時間イベント受理アクション 20s	時間変数 clock t;

3.3 モデル自動変換ツール

astah ファイルを入力すると表 1 の対応に基づき自動変換を行い，UPPAAL モデルの XML ファイルと対応表を出力するツールを java 言語で作成した．サブシステム固有の振舞いについては，アクション名とユースケース名が一致する時，そのアクションの位置でユースケース記述内に定義された処理が行われるとみなし変換する．

4. 適用実験と考察

事例として芝浦工業大学の PBL 課題である荷物自動搬送システムを用いる．本システムは，2 台の自律走行車両型ロボットがそれぞれ収集と配達という役割を果たして，与えられた環境および条件のもとで自動的に荷物を宅配する．図 2 は荷物自動搬送システムの全体図である．車両と中継所は LEGO MINDSTORMS EV3，その他のサブシステムは 1 台の PC 上で実現する．荷物の情報はサブシステム間の通信でやり取りする．

図 1 は左が荷物自動搬送システムのワークフローであり，右が変換した UPPAAL モデルである．検査を行うと，デッドロックが発生した．UPPAAL のシミュレータから反例を確認すると，受取人宅のプロセスの「確認した結果を送る」というロケーションでデッドロックが発生したと分かる．ワークフローにおけるこのロケーションの元となる要素を対応表で確認すると，配達担当ロボットに対して，「確認した結果」の送信であると分かる．そこで，配達担当ロボットのフローを確認すると，「確認した結果」に対する受信が定義されておらず，代わりに「受取人が正しいこと」の受信が定義されていた．異なる表現で同じ通信処理について記述すると，読み手によって解釈が異なる可能性があると判断し，双方とも「受取人が正しいこと」と表記を統一させた．このように，反例が発生した箇所の記述を確認し，設計制約の記述を修正することができる．最終的にデッドロックが発生しなくなることで，定義したモデルにおける各通信処理について同期が取れていることが確認できた．

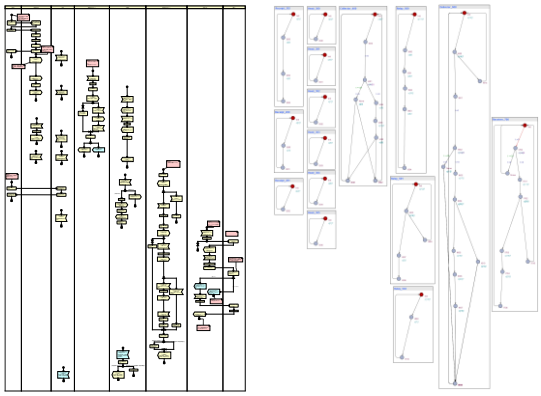

図 1. ワークフローと変換後の UPPAAL モデル

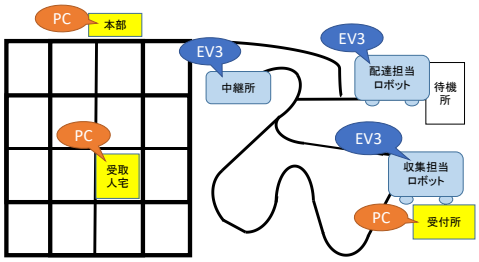

図 2. 荷物自動搬送システム

5. まとめ

開発の早期に作成したモデルの検証を行うことにより，サブシステムの同期が取れていることを保証したモデルで後の工程を行うことが可能となる．本研究の手法では，通信相手と通信内容の設計制約の記述を修正できる．しかしシステム全体の流れには，システムの各処理の事前条件，事後条件を記述し，フローの順序，繋がりを定義することが必要である．今後の課題は，UPPAAL モデルに変数を加え，事前条件，事後条件による各フローの繋がりを追えるようにすることである．

参考文献

[1] UPPAAL, http://www.uppaal.com/ (2017/6/20 参照)
[2] astah, http://astah.net/ (2017/6/20 参照)