アーキテクチャ記述の振舞い検証支援ツールに関する研究

2007MI181 小栗 達也　　2007MI264 山内 宏也
指導教員　野呂 昌満　蜂巣 吉成

1　はじめに

本研究室では，組込みシステムの開発を目的としたアスペクト指向ソフトウェアアーキテクチャスタイル E-AoSAS++[4] を提案している．昨年度の研究 [5] では，アーキテクチャ記述をプロセス代数の CSP[1] に変換して，CSP のモデル検査ツールである FDR[2] を用いて検証する方法を提案した．提案された検証方法には，アーキテクチャ記述から CSP 記述に変換することと，検査項目を記述することを手作業で行なうことによる間違いが混入する問題がある．

本研究の目的は，計算機による E-AoSAS++ における実行前検査の作業の効率化を図ることである．検証に必要な情報の図式表現を考える．図式表現から CSP への自動変換を確立する．検査項目を作成するときの煩雑な作業を整理し，図式表現できない情報を生成する．研究の方針として，仕様の図式表現を提案する．仕様の図式表現を用いて，直接 CSP で書く作業を，GUI を用いて検査項目を作成することで，煩雑な作業を明確にする．

成果として，アーキテクチャ記述からライブラリを用いて CSP 記述を自動生成できた．仕様の UML 記述を提案し，ライブラリを用いて CSP 記述を自動生成できた．GUI を用いて検査項目を作成することで，CSP の知識がなくても検証ができるようになった．また，CSP を自動生成することで，手作業による間違いの混入を減らすことができた．検証支援ツールの評価として，実際にアーキテクチャ記述と仕様から CSP 記述の自動生成と，検査項目の作成を試みた．

2　基本的なアイデア

本研究では，E-AoSAS++ に基づくアーキテクチャ記述に対するモデル検査を用いた検証の支援をする．検証作業のプロセスを図 1 に示した．検証作業のプロセスと，本研究のアイデアを説明する．

モデル検査

モデル検査とは，システムの振舞いを網羅的に走査し，システムが満たすべき性質を自動的に検査する手法である．FDR を用いた検証には，アーキテクチャ記述と環境と仕様の CSP 記述が必要である．CSP 記述は CSP のライブラリに適合した形を使用する．

検証の手順は，アーキテクチャ記述の UML 記述から CSP 記述に変換する．環境と仕様を CSP で記述する．アーキテクチャ記述と環境と仕様の CSP 記述を合成し，検証の情報を追加して検査項目を作成する．作成した検査項目を FDR に入力して検証を行なう．

本研究では，アーキテクチャ記述と仕様の図式表現から CSP 記述を自動生成し，GUI を用いて検査項目の作成を支援することで，手作業による間違いの混入を減らし，CSP の知識なしで検証することができる．

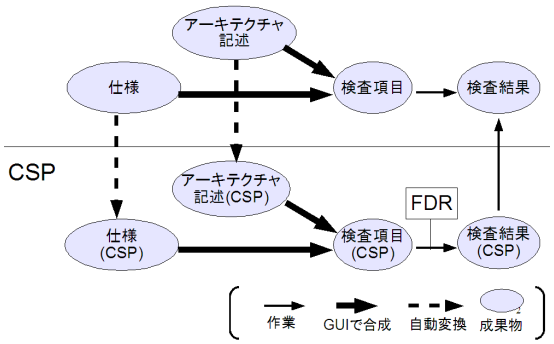

図 1　検証作業のプロセス

2.1　環境と仕様の UML 記述を CSP 記述に変換

検証をするには環境と仕様を直接 CSP で記述するので，CSP と FDR の知識が必要だった．また，仕様の書き方もきまっていなかった．今まで行なってきた検証の経験から仕様の記述に十分な要素を考え，仕様のモデルとする．この仕様のモデルを利用して仕様の図式表現を提案することで，CSP の知識なしで検証できると考えた．アーキテクチャ記述が UML で表記されていることから，仕様を UML を用いて図式表現をする．仕様の UML 記述から仕様のモデルに変換し，ライブラリを適用して CSP 記述に変換する．

2.2　アーキテクチャ記述の UML 記述を CSP 記述に変換

昨年度の研究で，E-AoSAS++ に基づくアーキテクチャ記述と CSP 記述の対応がとれている．CSP 記述への変換の手順は，UML 記述から E-AoSAS++ のアーキテクチャ記述モデルに変換し，ライブラリを適用して CSP 記述に変換する．E-AoSAS++ のアーキテクチャ記述モデルは，Java コード自動生成に関する研究で用いられているので，それを利用する．変換を 2 回行なうのは，アーキテクチャ記述モデルが既にあるのでそれを利用したかったからである．

2.3　アーキテクチャ記述と環境と仕様の CSP 記述を合成して検査項目を作成

検査項目の作成には，アーキテクチャ記述と環境と仕様以外の情報が必要である．今までは検査項目を一つ一つ手作業で書いていたので，記述し忘れや間違いが起こっていた．検査項目の作成作業を分析して明示する．検査項目に必要な情報を GUI で入力して，検査項目を自動的に作成する．

3 仕様の UML 記述の提案と CSP 記述の自動生成

これまで行なってきた検証の経験を基にした仕様のモデルから，仕様の UML 記述を提案する．提案した仕様の UML 記述から CSP 記述へ変換するさいの，仕様の UML 記述と仕様モデルの対応を示す．

3.1 仕様のモデル

CSP を用いて振舞い検証をする場合には，アーキテクチャと環境を組み合わせた振舞いを仕様として，イベントとイベント間の関係で表現する．これまで行ってきた検証の経験を基にした仕様のモデルを図 2 に示した．

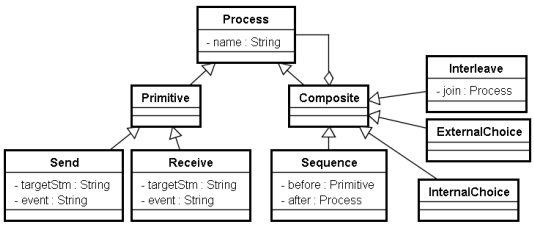

図 2　仕様のモデル

Process は Primitive または Composite からなる．Primitive はイベントを表わして，Composite はイベント間の関係を表わしている．イベントの種類として，イベントの送信とイベントの受信が存在する．また，イベント間の関係の種類として逐次，内部選択，外部選択，インターリーブの四つが存在する．

仕様のモデルをデザインパターン [3] の Composite パターンを用いて設計した理由として，Process は Composite であり，Composite は Process を持つという再帰構造になっていることが挙げられる．

3.2 仕様の UML 記述の提案

仕様を UML を用いて図式表現を行なう．仕様はイベントとイベント間の関係を制御フローで記述する．制御フローを表現できるアクティビティ図を用いることで仕様が直観的に理解しやすくなる．仕様の UML 記述にはアクティビティ図の記述方法を利用して，意味を新たに定義して拡張している． 図 3 にアクティビティ図で仕

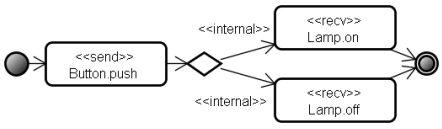

図 3　アクティビティ図を用いた仕様の例

様を記述した例を示した．UML のアクティビティ図の意味と大きく異なる点は，アクション名を”イベントの送信先 STM 名 ． イベント名”としていることである．また，アクションにステレオタイプをつけることで，イベントの送信と受信の意味を持たせている．アクション名にイベントの送受信の種類と STM 名とイベント名の情報を記述して，制御フローを用いてイベント間の関係を表現する．

3.3 UML 記述から CSP 記述への対応

仕様の UML 記述から仕様のモデルへの対応を図 4 に示す． 仕様のモデルとライブラリは 1 対 1 の関係と

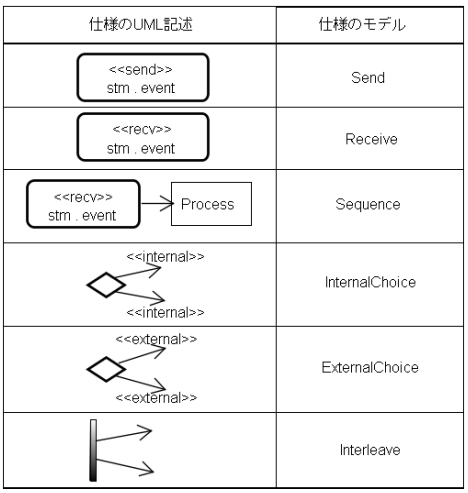

図 4　仕様の UML 記述とモデルの対応関係

なっているので，UML 記述から仕様のモデルへの対応をとることで CSP 記述を自動生成することができる．

4 アーキテクチャ記述から CSP 記述を自動生成

アーキテクチャ記述モデルから CSP のライブラリへの対応を示す．図 5 に E-AoSAS++ のアーキテクチャ記述モデルを示した．アーキテクチャ記述モデルとライブラリの対応をとることで CSP 記述へ自動生成ができる．

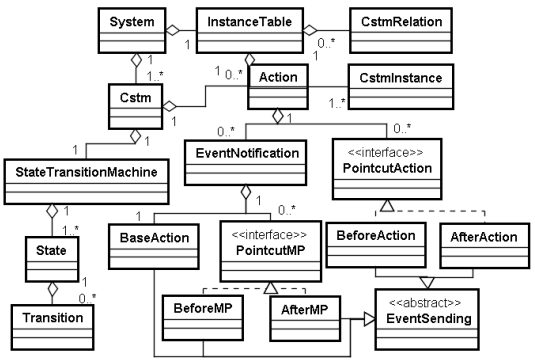

図 5　E-AoSAS++ のアーキテクチャ記述モデル

アーキテクチャ記述モデルからライブラリへの対応を表 1 に示す．ライブラリと CSP 記述は 1 対 1 の関係となっている．アーキテクチャ記述モデルからライブラリへの対応をとることで CSP 記述を自動生成できる．

表 1　アーキテクチャ記述モデルとライブラリの対応関係

アーキテクチャ記述モデル	ライブラリ
Cstm, StateTransition Machine	STM(stm,state)
State, Transition	StateTransition (stm, state, event)
EventSending	SEND(stm,event), RECV(stm,event)

5　GUI を用いた検査項目の作成

検査項目の作成に必要な情報を分析する．分析結果から GUI で必要な機能を分類する．

実際に行なっている検証作業の中から，検査項目の作成作業を分析した．分析した項目を自動的に作成するうえで，GUI に必要な機能を分類した．

- 入力を行なう
  - アーキテクチャ記述
  - 環境
  - 仕様
- 選択を行なう
  - CSP ライブラリの入力
  - 検証の種類
  - キューの長さ
- 一覧の表示と選択を行なう
  - 着目する状態遷移機械の一覧の表示と選択
  - 同期するイベントの一覧の表示と選択
  - 着目するイベントの一覧の表示と選択

これらの機能を GUI を用いて提供する．着目する状態遷移機械と同期するイベントと着目するイベントは，検証するアーキテクチャ記述の情報から取り出すことで表示を行なう．読み込んだ情報を GUI を使って一覧で表示することで，誤入力を減らすことができる．

6　検証支援ツールの設計

E-AoSAS++ のアーキテクチャ記述モデルと仕様のモデルの設計と，GUI の設計を示す．

アーキテクチャ記述から CSP 記述を自動生成

E-AoSAS++ のアーキテクチャ記述モデルから CSP 記述への変換は，モデルに対する処理が変更される可能性があることから，デザインパターンの Visitor パターンと Composite パターンと Interpreter パターンを用いた設計をした．設計したモデルを図 6 に示す．この図の AbstractStructureTree が E-AoSAS++ のアーキテクチャ記述モデルの構文木にあたる．

仕様の UML 記述から CSP 記述を自動生成

仕様のモデルに対しても同様にデザインパターンを用いた設計を行なった．図 6 の AbstractStructureTree が

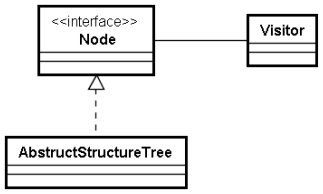

図 6　デザインパターンを適用した抽象構文木

仕様のモデルにあたる．デザインパターンを用いた理由は，処理が変更される可能性があるからである．

GUI を用いた検査項目の作成

5 章で行なった検査項目の分析を基に，設計した検査項目のデータ構造を図 7 に示す．

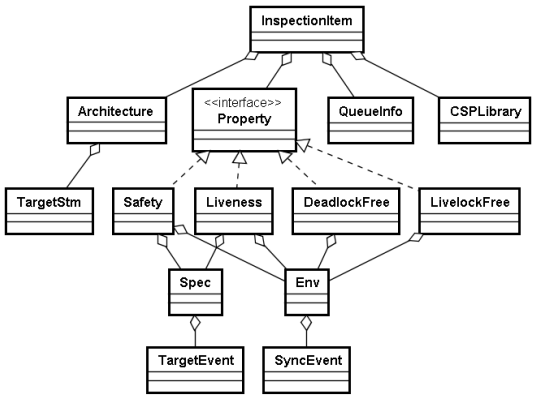

図 7　検査項目のデータ構造

5 章で行なった検査項目の分析を基に，設計した GUI を図 8 に示す．CSP ライブラリなどの選択をコンボ

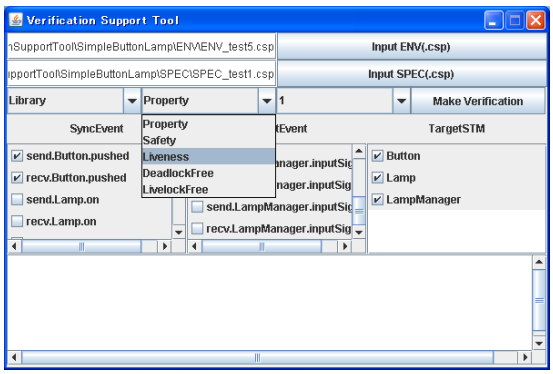

図 8　検査項目の作成画面のイメージ

ボックスで表示する．同期するイベントなどの一覧をリストで表示し，チェックボックスで複数選択することで情報の入力支援をする．

7 評価と考察

7.1 検証支援ツールの評価

ツールの評価として，アーキテクチャ記述と仕様から CSP 記述を正しく自動生成できるかと，検査項目を正しく作成できるかを確認する．

アーキテクチャ記述と仕様から CSP 記述を自動生成し，GUI を用いて検査項目を作成した．作成した検査項目を FDR に入力して検証を行なった．その結果，期待していた検査結果を得ることができたので，CSP 記述を正しく自動生成されているといえる．検証支援ツールによって，CSP 記述を正しく自動生成されていることが確認できた．

7.2 仕様モデルを基に自動生成した CSP 記述の妥当性の確認

仕様モデルを基に自動生成した CSP 記述の妥当性を，テストで確認した．テストは仕様の構文規則に当てはまっているかを確認した．

テストは仕様の構成要素ごとに用意した．Primitive は Send と Receive をそれぞれ 2 種類用意した．Sequence は Primitive と同じになるので 4 種類用意した．InternalChoice と ExternalChoice は選択するプロセスの数を最大で三つまでとして，それぞれ 2 種類用意した．Interleave は記述方法が 2 種類と，分岐するプロセスの数を最大で三つまでとして，4 種類のテストを用意した．Primitive だけではテストを行なうことはできないので，Composite の組み合わせを考え，深さの制限を 3 とした．全てのパターンを考えると膨大な数であるので，仕様の構成要素全てのテストが行なえるように組み合わせの中から 50 パターンを選んだ．テストの結果，テストした全てのパターンにおいて正しく生成できていたことから，CSP 記述の自動生成が正しく行なわれていることを確認した．以上のことから，仕様モデルを基に自動生成した CSP 記述は妥当であるといえる．

7.3 ほかの形式言語への対応

FDR 以外のモデル検査ツールを用いる場合，ほかの形式言語を用いなければならない．ほかの形式言語に対応させるときの労力について考察する．

現在，CSP のライブラリをアーキテクチャ記述と仕様にそれぞれ対応させて，CSP 記述を自動生成している．ほかの形式言語に対応させるとき，その形式言語のライブラリを整備して，アーキテクチャ記述と仕様に対応させる必要がある．本研究で定義した仕様のモデルは，CSP を用いて行なっていた検証を分析して作成した．ほかの形式言語を用いて検証する場合に，仕様のモデルとして情報が足りない可能性があるので，仕様のモデルを再検討する必要がある．また，検査項目に必要な情報も異なると思われるので，検査項目を分析する必要がある．以上のように，ほかの言語に対応させる場合にライブラリの整備と，仕様のモデルの再検討と，検査項目の分析の労力がかかると考えられる．

7.4 検証支援ツールの部品化

仕様の UML 記述から仕様のモデルを生成する処理を分析し，処理の整理を行なった．整理した処理を元に，処理の部品化ができるかを考察する．仕様のモデルが変更された場合には，仕様の UML 記述から仕様のモデルを生成する処理を変更する必要があり，変更には労力が必要になる．ツールの開発者は用意してある部品を利用することで処理を考える手間が減り，ツール開発の省力化ができる．

処理の部品化の例として，仕様のモデルの Send と Receive を生成する処理を分析する．Send と Receive を生成するさいには共通の処理が存在する．共通の処理は，アクティビティ図のアクションのステレオタイプを判別し，イベントをデリミタで区切って格納することである．異なる箇所は，ステレオタイプとイベントの二つのパラメータである．この二つのパラメータを引数としたメソッドを用意することで，ツールの開発者はメソッドを利用するだけで Primitive の作成が可能になるといえる．そのほかの例として，仕様のモデルの InternalChoice と ExternalChoice を生成する処理に対しても同様に，共通の処理とパラメータを設定してメソッドにできるといえる．以上のことから，共通の処理を異なるパラメータを引数としたメソッドを用意することでツールの開発の省力化になると考えられる．

8 おわりに

本研究では，アーキテクチャ記述から CSP 記述の自動生成と，仕様の UML 記述の提案と，仕様の UML 記述から CSP 記述の自動生成を行なった．検査項目を作成するときの煩雑な作業を整理し，GUI を用いた検査項目の作成を行なった．考察として，仕様モデルを基に自動生成した CSP 記述に対してテストで妥当性を確認した．今後の課題として，ほかの形式言語への対応が考えられる．

参考文献

[1] C.A.R. Hoare, *Communicating Sequential Process,* Prentice-Hall, 1985.

[2] Formal Systems(Europe), *"FDR2 User Manual,"* http://fsel.com/documentation/fdr2/html/fdr2manual.html, 2005.

[3] E.Gamma, R.Helm, R.Jhonson, and J.Vlissiders, *Design Patterns: Elements of Reusable Object Oriented Software*, Addison-Wealey, 1995.

[4] 加藤大地，蜂巣吉成，沢田篤史，野呂昌満，" アスペクト指向に基づくソフトウェアアーキテクチャの文書化方式，" 知能ソフトウェア工学研究会 (KBSE)，vol.108，no.449，pp.55-60，2009.

[5] 長谷川裕記，神谷浩翔, 岡嶋智史，" E-AoSAS++ における実行前検査に関する研究，" 南山大学数理情報学部情報通信学科 2009 年度卒業論文，2010.