ROOM法とエージェントネットによるSOAシミュレータの提案

倉畑 宏行[†] 藤井 拓[‡] 宮本 俊幸[†] 熊谷 貞俊[†]

†大阪大学 大学院工学研究科 電気電子情報工学専攻
‡（株）オージス総研 技術部ソフトウェア工学センター

要旨

近年，企業のシステムのアーキテクチャとして SOA が注目されている．SOA ではサービス間の相互作用の複雑さから，設計や検証が困難である．我々は SOA に基づくシステムの設計とその動作検証用シミュレータについて研究している．本論文では SOA の設計手法として提案している拡張 ROOM 法について述べる．この方法はサービス間の相互作用からサービスの動作モデルの生成により SOA の設計の困難さを解決する．

1. 緒論

近年，情報通信技術の発展により企業間の競争が激化している．各企業は競争力を高めるために業務を市場の要求に迅速に対応させることが必要となってきている．このためには，短期間で業務のための情報システムを構築することが必要である．このような背景から，短期間で情報システムを構築可能なアーキテクチャとして SOA(Service Oriented Architecture)[1]が注目されている．

SOA とは大規模なシステムをサービスの集合によって構成するシステムのアーキテクチャであり，サービスとは業務の機能を単位とするソフトウェアまたはシステムである．システムを開発する際，必要なサービスが総て揃っているという理想的な状況では，システム開発者はサービスの組み合わせのみでシステムを設計できる．よって，このような状況では業務のためのシステムを短期間で構築可能である．しかし，現状では企業内のシステムのサービス化が進んでおらず，開発の際は新規サービスの開発や企業内の既存システムのサービス化を伴うことが多い．

SOA に基づきシステムを開発する際はサービスの相互作用の複雑さが問題となる．１つのサービスは非常に多くの動作のパターンを持っており，複数のサービスの連携によるシステムの動作パターンは膨大となる．この複雑さがサービスの動作の設計を困難にし, 加えて設計に対する網羅的な検証を困難にしている．サービスを開発する場合は他のサービスとの連携を考慮して設計する必要がある．また，設計に対する検証に抜けがあると必要な動作の不足や論理的なエラーが実装に残る．実装段階でこのような不備が発覚すると修正のための開発コストの増加や開発期間の長期化が起こる．

我々はこのような設計と検証の困難さを解決するためのシミュレータ[2]を提案している．このシミュレータでは UML によりシステムを設計しこれを実行可能モデルに変換して検証を行う．我々はシミュレーションに用いる設計方法として拡張 ROOM 法を提案している. 拡張 ROOM 法は既存の設計手法である ROOM 法[3][4]にサービス間の相互作用のモデルから各サービスの動作モデルのスケルトンを自動生成する手順を加えた手法である. 動作モデルの生成により前述のサービスの動作設計の困難さを克服している．また，UML モデルを実行可能モデルであるエージェントネットモデル[5]へ変換しモデルの実行を可能とすることにより検証の困難さを解決することが可能である．本論文では拡張 ROOM 法の概要と UML のコミュニケーション図で表現されたサービス間の相互作用のインスタンス（シナリオ）からサービスの動作の生成手法を示す. 詳細な拡張ROOM 法の作成手順ならびにエージェントネットへの変換方法は文献[2]を参照されたい．

以下，**2.**では拡張 ROOM 法の概要を説明する．また，**3.**では生成の手順と例題を説明する．**4.**ではこの手法に対する評価と考察を述べる．最後に **5.**では結論を述べる．

2. 拡張 ROOM 法 概要

ROOM 法は UML2.0 で表現する場合，コンポジット構造図によりオブジェクトとオブジェクト間の接続関係，シーケンス図により予測されるオブジェクト間のメッセージ授受の経路，ステートマシーンモデルにより構成要素の動作を記述する記法である．ただし，3 つのダイアグラムの順

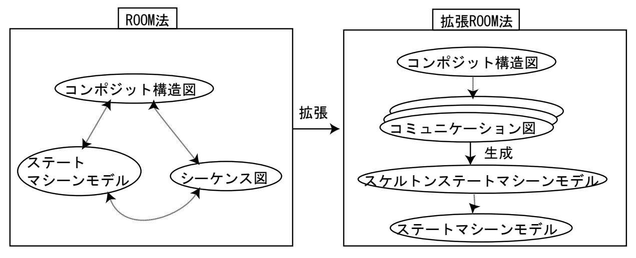

図 1　拡張 ROOM 法概要

序関係について定義していない．

ROOM 法を SOA に適用するには, 3 つのモデルのうちステートマシーンモデルを要求から直接設計することが困難であるという問題点がある．ROOM 法は，ステートマシンを単位にして、複数のイベントによる相互作用によりシステム全体の振る舞いをモデル化する手法である．要求で求められている 1 つのシステムの動作（以降シナリオと呼ぶ）は，複数のステートマシンがイベントを介して相互作用することによって実現する．ROOM 法のモデルが要求を満足するためには, 要求で求められる全てのシナリオを満足するための複数のステートマシンを考える必要がある．しかし，このようなステートマシーンモデルを要求から直ちに設計することは容易ではない．

この問題点を解決するため拡張 ROOM 法（図 1）を提案している．拡張 ROOM 法は各モデルに図 1 に示した設計順序を与えた. コンポジット構造図ではサービスとそれらの接続関係を要求から抽出し定義し，次に要求からシナリオを抽出しサービス間の連携モデルとして定義する．ここで，要求はシステムに必要な機能または動作や満たすべき条件の記述であり, シナリオは要求を満たす動作のインスタンスである．拡張 ROOM 法ではあるシナリオをコミュニケーション図により表現する．つまり，複数のコミュニケーション図でシステムの動作パターンを表現する．コミュニケーション図はシーケンス図に比べサービス間の接続関係の明示による理解の容易さと定義の容易さが利点である．次に，サービスの動作を定義するが，前述した問題点を解決するため複数のコミュニケーション図から抽出可能なステートマシーンモデルのスケルトンを生成する. スケルトンの生成によりサービスの動作のパターンを考慮しながら分岐を設計する必要や状態を発見するという困難さを克服できる. 最後により詳細なステートマシーンモデルを生成したスケルトンにデータに関する演算や状態の階層化を加えることにより定義する. このステートマシーンモデルはエージェントネットに変換することで更に検証を加える．

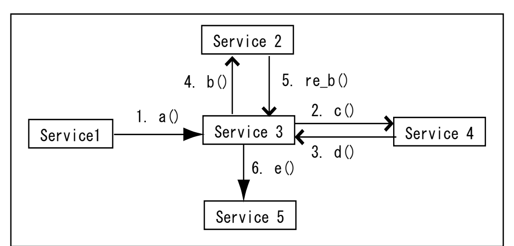

図 2　CD1

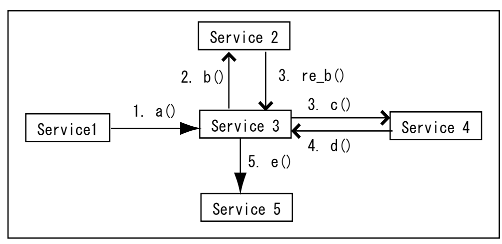

図 3　CD2

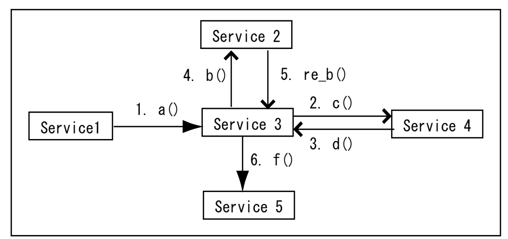

図 4　CD3

3. ステートマシーンモデルの生成

過去の研究[2]で同期メッセージで構成されたコミュニケーション図からステートマシーンモデルの生成方法を提案した．以下に概要を列挙する．

- コミュニケーション図ごとに，あるサービスに対して接続するメッセージのシーケンス番号による全順序集合の作成
- メッセージグラフの作成（サービスごと，同一メッセージは同一ノードにマージ)
- 変換規則に従いメッセージグラフをステートマシーンモデルへ変換

本論文ではこの方法を拡張し，非同期メッセージに対応させた生成方法を提案する．非同期メッセージを含んだステートマシーンモデルを生成する際，問題となる点はコミュニケーション図がメッセージの順序の制約を記述できない点である．つまり，メッセージの順序が異なり，同じシステムの挙動を表現した図が複数存在する可能性がある．この例を図 2〜図 4 に示す．図 2 では a(), e()は同期メッセージであり，b(), c(), d()は非同期メッセージである．また非同期メッセージの送信受信の関係は re_b()

1. 各コミュニケーション図iの各サービスjに対するメッセージ集合$CD_{(i,j)}$ とする
2. 各コミュニケーション図iの各サービスjについて同期メッセージに関する全順序$SM_{(i,j)}$を作成し $CD_{(i,j)}$に加える
3. 同様に、非同期メッセージの集合$aSM_{(i,j)}$を作成し$CD_{(i,j)}$に加える
4. メッセージグラフの作成（∀j について）

（∀i，∃j について） メッセージグラフ G_j を定義する

  0. $SM_{(i,j)} \in CD_{(i,j)}$ のメッセージ m_{i1} ......m_{ir} とする
  1. もし$m_{ir} = m_{i'r}$ なる メッセージを持つノードがG_jに存在すれば同一ノードとしてマージ
  2. そうでなければ 異なるノードとしてG_j にノードを定義

  3. $aSM_{(i,j)} \in CD_{(i,j)}$の各メッセージについて
  4. G_jに$aSM_{(i,j)} \subseteq aSM_{(i',j)}$ なる非同期メッセージの集合を持つノードがあれば，同一ノードとしてマージ
  5. そうでなければ，G_j に$aSM_{(i,j)}$を持つ非同期メッセージノードを定義

5. 総てのG_jについて同期メッセージ順序を逆転させ入出力関係を逆転させたノードの集合を接続
6. 全てのG_j について ステートマシーンモデルを作成
  1. 初期ノード，終了ノードを追加
  2. G_j が同期メッセージノードと非同期メッセージノードを持てば更にフォークノードとアークを追加する
  3. G_jを変換規則（資料1）にしたがって変換
  4. 出力状態が空である遷移の出力を終了状態に指定

図 5 変換アルゴリズム

のように明示している．ここで設計者の意図として b(), c()は同時に送信し，re_b(), d()の到着タイミングは不明であるとする．図 3 は異なるメッセージの順序で図 2 と同様の内容を表現している．図 3 では b(), c()の送信のタイミングが図 2 とは逆になっている．また，図 4 では e()ではなく f()という同期メッセージが発生する図となっている．

これらを同様の動作を判別し，ステートマシーンモデルを生成するアルゴリズムを図 5 に示す．このアルゴリズムでは, ある各コミュニケーション図上の各サービスの同期メッセージを全順序集合，非同期メッセージを半順序集合（ただし，順序関係は送受信のみに限定）として扱い，これらをメッセージグラフというメッセージの情報を保持した特殊なグラフへ変換している. メッセージグラフへの変換の際, 同期メッセージは同一メッセージである場合単一のノードへマージしている. ただし, 同期メッセージの返信タイミングはコミュニケーション図には明示されていない. よって, 同期メッセージの全順序を逆転させた返信メッセージを定義しメッセージグラフに追加している (STEP5)．また，非同期メッセージに関してはメッセージの集合が部分集合として含まれる場合（同一の場合も含む）に単一のノードとしてマージする．次に図 6 に示すメッセージグラフからステートマシーンモデルの変換規則によりメッセージグラフをステートマシーンモデルに変換する．ここで，は変換規則にガード条件の変換と繰り返しメッセージの変換規則も含まれていることに注意されたい．

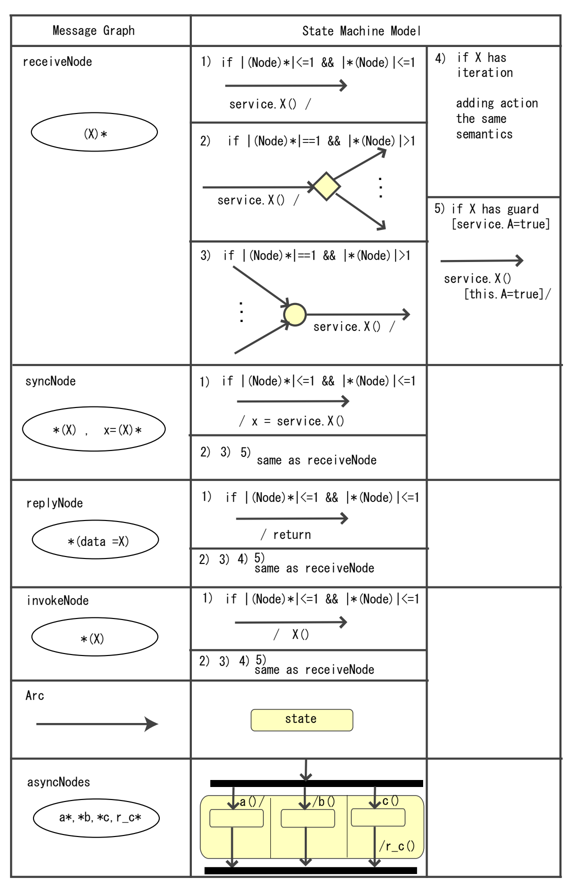

図 6 メッセージグラフからステートマシーンへの変換規則

図 2 から図 4 に対して図 5 のアルゴリズムを適用した結果を示す. ここでは特に service3 に限定する. まず, 各コミュニケーション図から得られるメッセージに関する同期, 非同期メッセージの集合は以下のようになる(STEP1-STEP3).

$$CD(1,3) = CD(2,3) = \{\{a \rightarrow e\},\{b \rightarrow re_b, c, d\}\}$$

$$CD(3,3) = \{\{a \rightarrow f\},\{ b \rightarrow re_b, c, d\}\}$$

ただし，CD(i,j)は CD(i)の Service(j)に接続するメッセージであり CD(i,j)={SM,aSM}である. このメッセージの集合から図 7 に示すメッセージグラフが作成される(STEP4)．最後にメッセージグラフがステートマシーンモデルへ変換すると図 8 を得る (STEP6).

4. 評価と考察

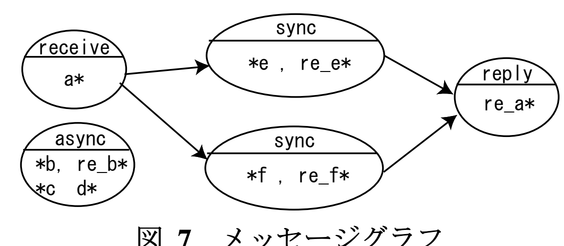

図 7　メッセージグラフ

図 8　スケルトンステートマシーン

提案した方法により作成したステートマシーンモデルは生成もとの全てのコミュニケーション図を実行可能である．しかし，生成したステートマシーンは定義したコミュニケーション図以外のシナリオ（並行シナリオ）も実行可能である．設計者の意図したモデルのセマンティクスによっては特定の並行シナリオが許されない場合もある．しかし，現状のアルゴリズムでは特定の並行シナリオの発生を許さないステートマシーンを生成することはできない．コミュニケーション図は作成が容易である反面，メッセージ間の複雑な制約が記述できない．このことが原因でメッセージの発生順序が不確定なステートマシーンモデルとなっている．

これを解決するための方法は非同期メッセージに対して前後に発生するメッセージに関する制約を付加する方法が考えられる．前後のメッセージを指定すれば厳密なメッセージ間の関係を持つステートマシーンが生成可能であると考える．同時に，メッセージの前後関係の指定によりループが発生する可能性があり制約の定義方法，制約の論理的にチェックする方法が必要である．また，制約の定義の労力が非常に大きくなる可能性もあり，``簡潔さ”というコミュニケーション図を用いる利点に逆行する可能性もある．これらを含めて解決する方法の提案が今後の課題である．

この分野の同様の研究として Jon Whittle[6]らはシーケンス図と OCL(Object Constraint Language)の事前，事後条件とマクロな変数の定義により記述されたシナリオからステートチャートの生成方法を提案している．OCL の解析により精密なステートチャートの生成が可能である．我々の方法は設計の困難さを克服するため簡素な方法で相互作用を定義している．

5. 結論

本論文では，SOA に基づくシステム検証用 UML シミュレータを構成する主要な技術の内，拡張 ROOM 法とステートマシーンモデルの生成方法を示した．作成されるステートマシーンモデルは定義したコミュニケーション図によるシナリオを実行可能なモデルであることが確認できた．

今後，コミュニケーション図の定義の際，メッセージの前後関係の制約の記述により更に厳密なステートマシーンモデルを生成する方法の提案する予定である．

謝辞　本研究の一部は文部科学省魅力ある大学院教育イニシアティブ「先端通信エキスパート養成プログラム」による助成を受けている．記して謝意を表す．

参考文献

[1] E. Thomas. Service-Oriented Architecture, PRENTICE HALL, (2004).
[2] 倉畑宏行，藤井拓，宮本俊幸，熊谷貞俊，``エージェントネットに基づく SOA 検証用 UML シミュレータの提案”，情報処理学会研究報告，2007-SE-157，pp103-110, (2007).
[3] B. Selic, G. Gullekson, P. T. Ward. REAL-TIME OBJECT-ORIENTED MODELING, WILEY PROFESSIONAL COMPUTING, (1997).
[4] B. Selic, J. Rumbaugh. ``Using UML for Modeling Complex Real-Time Systems’’, Rational Software Corporation , (1998). (http://www.ibm.com/developerworks/rational/library/content/03July/1000/1155/1155_umlmodeling.pdf)
[5] T. Miyamoto, S. Kumagai, ``A Multi Agent Net Modeling and the Realization of Software Environment’’, Proc. Of Workshop of Petri Nets to interligent system development in ICAPTN’99, pp.83-92, (1999).
[6] W. Jon, S. Johann. ``Generating Statechart Designs From Scenarios’’, 22nd International Conference on Software Engineering, pp314-323, (2000).