社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

シーケンス図が持つメッセージ順序の曖昧性除去手法の提案

楠　　野明† 　岡野　浩三† 　楠本　真二†

† 大阪大学大学院情報科学研究科，吹田市
E-mail: †{k-noa,okano,kusumoto}@ist.osaka-u.ac.jp

あらまし　本稿は，仕様設計の段階で使用されるシーケンス図においてメッセージ順に関する欠陥を検出，および修正する手法を提案する．提案手法はシーケンス図に含まれる全てのメッセージに対して，メッセージ送受信の順序が逆転してしまう可能性 (メッセージ順序の曖昧性) が存在するかをモデル検査の手法に基づき検査し，検出した場合はその可能性を除去する修正候補を開発者に提案する．評価の結果，妥当な時間で使用者にメッセージ順序の曖昧性を持つ箇所を，40%近い割合で示すことがわかった．
キーワード　シーケンス図，モデル検査，LTL 式，XML

Removing Possibility of Ambiguous Message Ordering in Sequence Diagram

Noa KUSUNOKI†, Kozo OKANO†, and Shinji KUSUMOTO†

† Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871, Japan
E-mail: †{k-noa,okano,kusumoto}@ist.osaka-u.ac.jp

Abstract　This report proposes a method to detect and repair software faults in sequence diagrams which are used for software design. Our proposed method checks possibility of ambiguous message ordering in sequence diagram using a model checker. If it finds the possibility, a developer is proposed modification which remove this possibility by the tool.Experimental evaluation finds that the tool can point out a user where to modify in reasonable time.
Key words　Sequence Diagram, Model Checking, Linear Temporal Logic, XML

1. ま え が き

ソフトウェアの信頼性は重要であり，ソフトウェアの品質を保証するための手法として欠陥の検出，および修正に関する手法が盛んに研究されている．例えば，プログラムのソースコードとテストケースを入力として，欠陥箇所を検出し，修正案を出力する手法 [1] や，ソースコードを既存のコードを用いて修正する手法 [2]〜[4] などソースコードを入力とした様々な手法が提案されている．しかし，ソフトウェアが欠陥を生じるのは実装の段階だけとは限らない [5]．例えば，仕様設計の段階で欠陥を生じてしまう場合もある．そのような欠陥が実装を行う段階で初めて発覚した際は，開発者は仕様設計の段階まで遡って修正し，再び実装しなければならない場合もある．この場合，欠陥の修正にかかるコストは大きくなってしまう．

本研究は，仕様設計の段階で使用されるシーケンス図においてメッセージ順に関する欠陥を検出，および修正する手法を提案する．対象とする欠陥は非同期モデルを表すシーケンス図において行われるメッセージ送受信の順序が状況により逆転してしまうことによる欠陥とした．この欠陥を本研究では “メッセージ順序の曖昧性” と定義して以降の説明を行う．提案手法が検出するメッセージ順序の曖昧性は必ずしも修正すべきものとは限らないため，開発者に修正すべきかを確認しながら修正する半自動的な手法としている．

手法は，まずシーケンス図からモデル検査に用いるモデルを生成する．次にシーケンス図に含まれるオブジェクト同士が行うメッセージの送受信から，モデル検査のもうひとつの入力である検査式を生成する．ここで生成される検査式はオブジェクトごとに，そのオブジェクトに関連するメッセージにメッセージ順序の曖昧性が存在するかをチェックするものである．そのためシーケンス図が含むオブジェクトとメッセージの数に応じて，検査式は複数得られる．次に得られたモデルと検査式を用いてモデル検査を行う．モデルが検査式に表された検査したい性質を満たさない場合，つまりモデルがメッセージ順序の曖昧性を持つ場合，提案手法は開発者にそのメッセージ順序の曖昧性を除去するためにシーケンス図の修正候補を提案し，その修正を適用するかどうかを選んでもらう．すべての検査式に対してモデル検査を行い，修正候補が提案されなくなれば，開発者によって適用すると選ばれた修正候補を用いてシーケンス図を

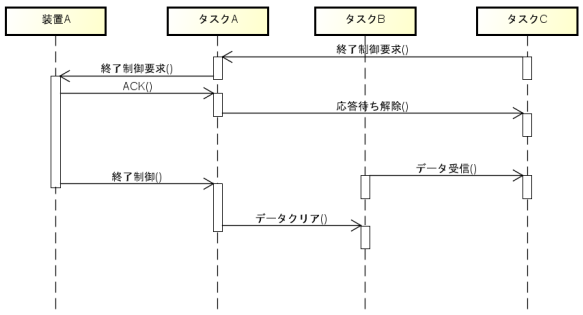

図 1　開発現場で使われるシーケンス図の例

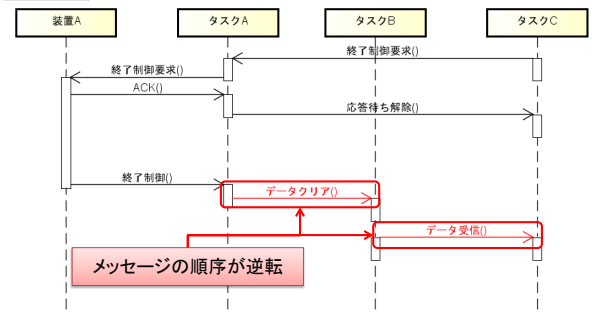

図 2　想定されるエラーの例

修正する．これにより，シーケンス図に含まれるメッセージ順序の曖昧性を除去することが出来る．また，手法によって生成されたシーケンス図を表すモデルは別途検査式を与えることで，異なる性質に対して検査を行うことも可能である．

提案手法をツールとして実装し，2 つの評価を行った．1 つ目の評価では収集，および生成した 11 個のシーケンス図を用いてツールが生成する修正候補の数や実行時間を評価した．2 つ目の評価では実際の開発現場の経験を元にして得られたメッセージ順序の曖昧性を持つシーケンス図にツールを適用した．その結果，得られた 10 個の修正候補について開発者に確認したところ，この内 4 個の修正候補に対して必要な修正であると判断された．

2. 研 究 動 機

図 1 はソフトウェア開発において使われているシーケンス図である．この図は要求定義や仕様設計の段階で用いられるものであり，開発者はすでに欠陥は取り除いたと考えている．しかし，開発の途中で図 2 のような欠陥が発見されることがある．図 2 は開発経験をもとに想定される欠陥の例であり，これはタスク B からタスク C へ “データ受信” というメッセージが送られる処理と，タスク A からタスク B へ “データクリア” というメッセージが送られる処理の順序が逆転してしまっている．この逆転によりデータ受信が完了する前にタスク B が保持していたデータが消えてしまい，データ受信の処理が失敗するという欠陥が起こっている．

シーケンス図では，各オブジェクトが処理を行う際に各オブジェクトのライフラインが実行状態となり，実行仕様が四角で表示される．例えば図 1 の装置 A においては，タスク A から “終了制御要求” のメッセージを受信する処理，タスク A に “ACK” のメッセージを送信する処理，タスク A に “終了制御” のメッセージを送信する処理の 3 つの処理が 1 つの実行仕様で行われている．また，タスク B においてはタスク C に “データ受信” のメッセージを送信する処理が 1 つの実行仕様で，タスク A から “データクリア” のメッセージを受信する処理が 1 つの実行仕様で行われており，2 つの処理が異なる実行仕様で行われている．このような異なる実行状態で行われる処理の間に依存関係がない場合は並列に実行されるように実装されることがある．そのように実装された場合，非同期モデルのシーケンス図では各処理の依存関係が存在しないために処理が行われる順序の逆転が発生してしまうのである．

本研究では，図 1 のようなある程度設計が進んだシーケンス図からメッセージ順序の曖昧性を除去する手法を提案する．

3. 提 案 手 法

3.1 概　　要

提案手法は半自動的にシーケンス図の修正を行い，シーケンス図が持つメッセージ順序の曖昧性を除去する．手法の入力はシーケンス図を表す xml ファイルで，出力は入力として与えられたシーケンス図からメッセージ順序の曖昧性を除去したシーケンス図を表す xml ファイルである．また，メッセージ順序の曖昧性はシステムの挙動に悪影響を与えない場合も存在する．これは入力として与えられる情報だけでは判断できないため，開発者に検出したメッセージ順序の曖昧性を除去するか判断してもらうようにした．そのため，提案手法は半自動的な手法となっている．手法の対象は非同期モデルを表し，メッセージ交換が行われているシーケンス図である．また，手法で用いるシーケンス図のバージョンは UML1.4 [6] とした．提案手法は図 3 のような流れで，以下の 4 つの STEP で行われる．

STEP-1: モデル記述生成
STEP-2: 検査式生成
STEP-3: 修正候補群導出
STEP-4: シーケンス図修正

以降では，行う処理を各 STEP ごとに説明する．

STEP-1: モデル記述生成

この STEP では，入力として与えられたシーケンス図を表す xml ファイルからモデル検査で用いるモデルを生成する．提案手法が用いる xml ファイルは astah* professional [7] を用いて取得する．また，モデルは SPIN モデルチェッカ [8] で用いる promela 言語で記述されたものを生成する．生成方法は Lima らの手法 [9] を参考にしており，シーケンス図に含まれるオブジェクトは実行仕様ごとにモデルにおけるプロセスに，メッセージはモデルにおけるチャネルと変数でそれぞれ表現する．

図 4 を例として説明する．図 4 の上部はシーケンス図，下部はシーケンス図から生成したモデルを略記したものである．シーケンス図に含まれるオブジェクト B には 3 つの実行仕様が存在するので，プロセス B_1，プロセス B_2，プロセス B_3

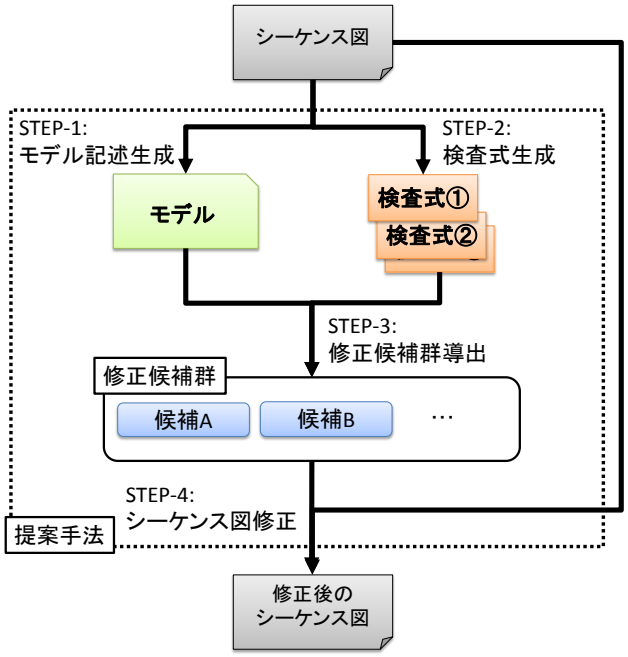

図 3　提案手法の流れ

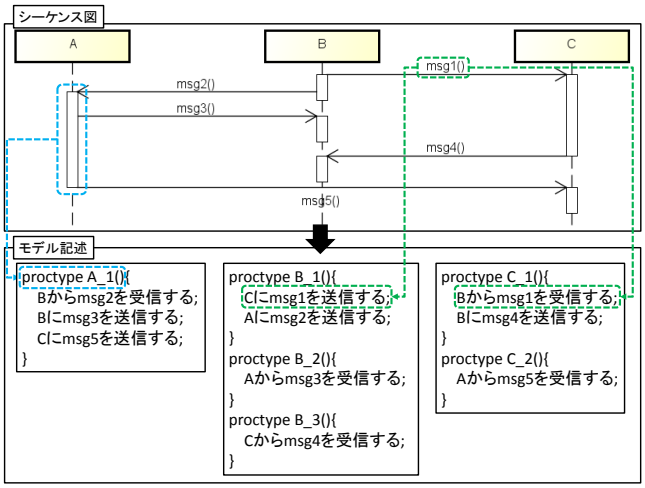

図 4　シーケンス図からモデルの生成例

としてそれぞれモデルが生成されている．このように実行仕様ごとにプロセスを生成することにより同一の実行仕様に含まれる処理の実行順を保証する．その一方で，異なる実行仕様に含まれる処理間でおこりえる非同期メッセージによるメッセージ順序の曖昧性をモデル化している．また，参考にした Lima らの手法では実行仕様ごとにプロセスを生成することは行われていない．また，オブジェクト B からオブジェクト C に対しての “msg1” の送信は，プロセス B_1 では 8 行目のオブジェクト C に対する “msg1” の送信に変換されている．また，プロセス C_1 においては 14 行目にオブジェクト B から “msg1” を受信する処理が記述されている．

STEP-2: 検査式生成

この STEP では，入力として与えられたシーケンス図を表す xml ファイルからモデル検査で用いる検査式を生成する．検査式は時相論理式 (LTL 式) を用いる．検査式はシーケンス図が

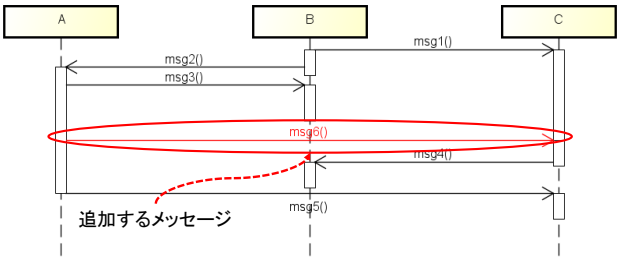

図 5　修正候補例

メッセージ順序の曖昧性を持つかどうかを検査する．隣接する全てのメッセージ組に対してメッセージ順序の曖昧性を検査するので，検査式はメッセージ，およびオブジェクトの数に応じて複数生成される．

例えば，図 4 のシーケンス図から検査式を生成する場合を考える．シーケンス図において，オブジェクト B は 4 つのメッセージ送受信を行っている．このオブジェクトに関して検査したい性質は以下のとおりである．

（1） “msg1” を送信する前に，“msg2” を送信するか
（2） “msg2” を送信する前に，“msg3” を受信するか
（3） “msg3” を受信する前に，“msg4” を受信するか

以上の 3 つの検査したい性質に対して，それぞれ以下に示す検査式が生成される．

（1） $\neg(msg2\text{ を送信する})U(msg1\text{ を送信する})$
（2） $\neg(msg3\text{ を受信する})U(msg2\text{ を送信する})$
（3） $\neg(msg4\text{ を受信する})U(msg3\text{ を受信する})$

STEP-3: 修正候補群導出

この STEP では，STEP-1 で取得したモデルと STEP-2 で取得した検査式を用いてモデル検査を行い，修正候補の導出を行う．修正候補とは，シーケンス図をどのように修正するかを示すものである．どのメッセージ送受信の組にメッセージ順序の曖昧性があるかの情報を保持しており，それらのメッセージ送受信の間にメッセージを追加するという修正を示す．メッセージの送信元は，メッセージを追加される組において先にメッセージ送受信を行うオブジェクトであり，メッセージの受信先は組において後にメッセージ送受信を行うオブジェクトとなる．

修正候補はモデルが，検査式が表す検査したい性質を満たさない場合に生成される．そのため，モデル検査の結果に応じて修正候補は複数生成される．修正候補は入力として使った検査式から導出される．検査式にはメッセージ順序の曖昧性が存在するかを判定したい対象として，2 つのメッセージ交換が書かれている．モデル検査の結果，モデルがメッセージ順序の曖昧性をもつと示されたとき，提案手法は検査式に書かれたメッセージ交換の間にメッセージの送受信を追加するという修正候補を生成する．例えば，図 5 では，モデルに対して，

“$msg3''$を受信する前に,“$msg4''$を受信するか

という性質を検査した結果，メッセージ順序の曖昧性が存在すると判定されている．そこでオブジェクト A には検査したい性質の前半に書かれている “msg3” を送信した後に，“msg6” を

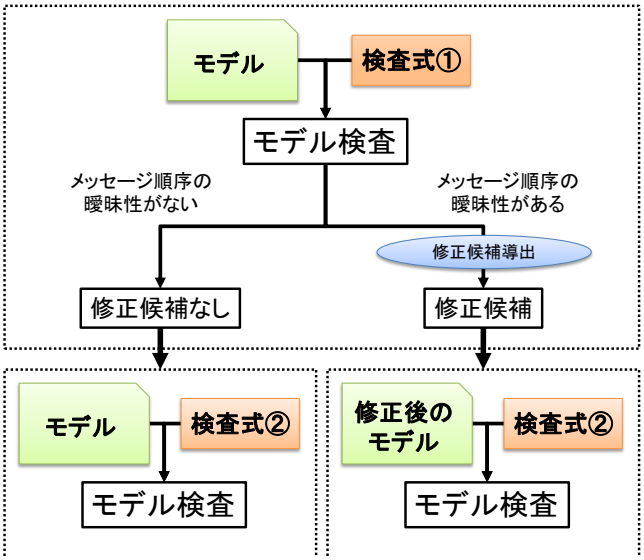

図 6　修正候補導出の流れ

送信する処理が追加されている．また，オブジェクト C には検査したい性質の後半に書かれている “msg4” を送信する前に “msg6” を受信する処理が追加されている．

修正候補の導出は図 6 のように全ての検査式に対して行われる．まず複数存在する検査式から 1 つを選び，その検査式とモデルを入力として SPIN モデルチェッカでモデル検査を実行する．検査の結果に応じて，以下のように異なる処理を行う．

メッセージ順序の曖昧性がない　同じモデルと別の検査式を用いてのモデル検査に移る

メッセージ順序の曖昧性がある　入力した検査式から修正候補の導出を行う．修正候補を適用するかを開発者に判断してもらい，適用すると判断された場合はこの修正候補を保持する．また，適用するかどうかにかかわらず，同じ反例がでないようにモデルを修正する．そして修正したモデルと，別の検査式を用いて再びモデル検査を行う．

これを繰り返し，全ての検査式に対するモデル検査が終了すればこの STEP を終了する．

STEP-4: シーケンス図修正

STEP-3 で開発者に適用すると選ばれた修正候補を用いて，入力として与えられた xml ファイルを修正する．修正候補は追加するメッセージの名前とメッセージを追加する箇所を示しているので，メッセージの定義やシーケンス図の各ライフラインに関するメッセージ情報をそれぞれ修正する．開発者に適用すると判断された修正候補全てを用いて，xml ファイルを修正すればこの STEP を終了する．

この修正を全て行った後，全ての検査式に対して違反がなければ，対象のシーケンス図がメッセージ順序の曖昧性を持たないことが保証される．

4. 評　　価

本研究では提案手法の性能を評価するために，Java 言語やシェルスクリプトを用いて手法をツールとして実装した．また，実装したツールを用いて以下に示す 2 種類の評価を行った．

- 評価 1: 生成したシーケンス図による，手法が生成する修正候補数，実行時間の評価
- 評価 2: 実際のシーケンス図による，手法が示す修正候補の評価

なお，評価に用いた環境は以下の通りである．

- OS: Windows 7 Professional
- CPU: Intel Xeon E5607 2.27GHz × 2
- Memory: 16.0GB
- SPIN: version 6.3.2

また，SPIN モデルチェッカの状態ベクトルのサイズはデフォルトの 1024bytes とした．

以降 4.1 章から 4.2 章では各評価の方法と結果について，詳細に説明する．

4.1 評 価 方 法

4.1.1 評　価　1

シーケンス図に関連する既存研究 [10]〜[14] やツール [15], [16] において，対象や例として示されているシーケンス図を収集した．また，収集したシーケンス図の内 1 つに対して，オブジェクト数，およびメッセージ数を増やす変更を加え，新たにシーケンス図を生成した．これらのシーケンス図に対してツールを適用し，シーケンス図に含まれるオブジェクト数やメッセージ数，およびツールが生成する修正候補の数と実行時間を計測した．

この評価を行う際は，開発者はツールが提案する修正候補を全て採用すると仮定して，実行時間を計測した．

4.1.2 評　価　2

実際の開発現場で使われているシーケンス図を表す xml ファイルを astah* professional [7] によって取得し，ツールを適用した．これによりツールが図 2 に示したような欠陥を検出できるかどうかを確かめた．また，ツールが生成した修正候補を開発者に確認してもらい，ツールが出力する修正候補が必要かどうかを確認した．

4.2 結　　果

4.2.1 評　価　1

11 個のシーケンス図にツールを適用した．評価の結果は表

表 1　評価 1 の結果

	obj	msg	修正候補数	実行時間 (秒)		
				STEP-1,2	STEP-3	STEP4
[10]	7	11	3	0.44	20.21	0.43
[11]	4	12	2	0.42	25.19	0.41
[12]	3	8	0	0.39	15.55	0.41
[13]	3	4	0	0.37	5.96	0.40
[15]	5	6	1	0.42	8.95	0.41
[16]	6	24	0	0.49	56.15	0.42
[14]	7	22	6	0.46	52.08	0.48
[14]-1	12	44	9	0.58	124.77	0.45
[14]-2	22	88	18	0.98	9136.59	0.52
[14]-3	32	132	24	0.97	14054.17	0.56
[14]-4	37	154	0	1.20	744.73	0.56

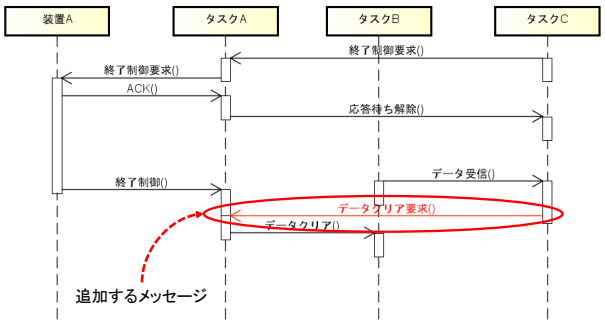

図 7　修 正 候 補

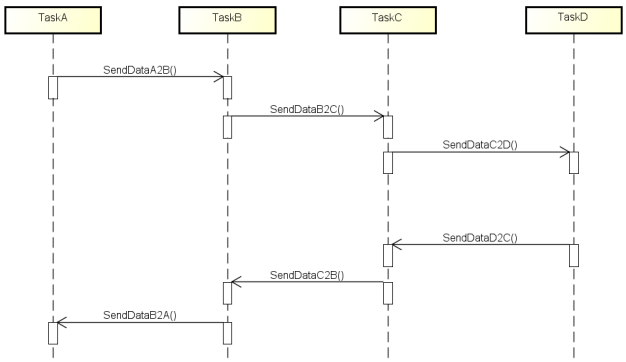

図 8　開発現場で使われるシーケンス図の例 2

1 のようになった．対象の内，7 個は収集したものを，4 個は文献 [14] のシーケンス図のインスタンス数を変更し，生成した (以降 [14]-1，2，3，4 と略記) ものである．表 1 の 1-3 列目は対象としたシーケンス図の情報を，4-7 列目までは対象にツールを適用した結果を示している．表の 1 列目はシーケンス図の取得元を，2 列目の “obj” はシーケンス図に含まれるオブジェクト数を，および 3 列目の “msg” はシーケンス図に含まれるメッセージ数をそれぞれ示している．また，4 列目の “修正候補数” はツールが生成した修正候補の数を，5-7 列目はツールの適用にかかった実行時間をそれぞれ示している．実行時間に関しては STEP ごとに計測した．ただし，5 列目の STEP-1，2 に関しては，STEP-1 の処理と STEP-2 の処理が同じシーケンス図を入力として，並行して処理が行われるため 2 つの STEP の実行時間を合わせて計測している．

STEP-1，2 の実行時間は対象による大きな差はないが, [14]-2，3，4 の値は比較的大きくなっている．これはシーケンス図に含まれるオブジェクトの数やメッセージの数が大きいためだと考えられる．また，STEP-3 の実行時間はシーケンス図に含まれるオブジェクト数やメッセージ数の値が大きくなるほど，永くなっていることがわかる．STEP-4 の実行時間についても大きな差はないが，生成した修正候補の数に応じて大きくなっていることが考えられる．

表 1 において [14]-4 の STEP-3 の実行時間はオブジェクト数やメッセージ数の値が非常に大きいにもかかわらず, [14]-3 の値に比べて非常に小さくなっている．これは [14]-4 のシーケンス図を表すモデルを用いてモデル検査を行った際にメモリ不足でモデル検査が行えなかったためである．そのため，修正候補数の値も 0 となっている．

4.2.2　評　価　2

ソフトウェアの開発現場で実際に使われるシーケンス図を 4 つ提供してもらい，その中でメッセージ順序の曖昧性に関するエラーシーケンスが想定される 2 つに対してツールを適用した．ツールを適用したシーケンス図を，図 1 と図 8 に示す．ツールを適用した結果は表 2 のようになった．表 2 が含む情報は表 1 と同様になっている．

ツールによって得られた 10 個の修正候補の内 1 つについて説明する．図 7 が示す修正候補は，“タスク C” が “データ受信” のメッセージを受信する処理と，“タスク A” が “データクリア” のメッセージを送信する処理の間に，“タスク C” が “データクリア要求” のメッセージをを送信する処理と，“タスク A” がそのメッセージを受信する処理を追加する修正を示している．この修正候補は図 2 が示した “データ受信” のメッセージと “データクリア” のメッセージの順序が逆転してしまう欠陥を元に生成されている．よって，提案手法は図 2 のような欠陥を検出できるといえる．

この修正候補を含む 10 個の修正候補を開発者に確認してもらい，各修正候補が必要かどうかを判断してもらった．判断にあたって，開発者には各修正候補が示す修正が以下の 3 つのうちどれにあたるかを 1 つ選んでもらった．

（1）　この修正は行うべきである
（2）　この修正は行っても行わなくてもよい
（3）　この修正は行うべきではない

その結果，図 7 に示した修正候補を含む 4 個の修正候補については，(1) が選ばれた．また，5 個の修正候補については (2) と判断された．(3) と判断された修正候補は 1 個であった．

5.　妥当性の脅威

評価 1 では既存研究やツールからシーケンス図を収集し，ツールを適用することでツールが生成する修正候補の数や実行時間を評価した．今回，手法を適用したシーケンス図は 11 個であり，またシーケンス図に含まれるオブジェクト数やメッセージ数も限られた規模のものとなっている．そのため，規模の異なるシーケンス図を評価に用いた場合は異なる結果が得られる可能性がある．

次に，評価 2 では，実際に開発現場で発生した事態を元に想定されたエラーシーケンスを発生させる可能性を持ったシーケンス図を用いて，提案手法が欠陥を検出し，修正候補を提示できるかを確認した．今回の評価では対象が 2 つであるため，より一般的な結果を得るためにはより多くのシーケンス図を開発

表 2　評価 2 の結果

	obj	msg	修正候補数	実行時間 (秒)		
				STEP-1,2	STEP-3	STEP4
図 1	4	7	5	0.40	14.04	0.62
図 8	4	6	5	0.41	12.15	0.52

現場から取得し，評価を行う必要がある．

6. 関 連 研 究

Lima らは，シーケンス図から promela ファイルを生成し，モデル検査によって欠陥を早期に検出する手法を提案している [9]．この手法では UML2.0 の形式で書かれたほとんどのシーケンス図に対して，promela 言語における表現形式を提案している．また，この生成方法を eclipse プラグインとして実装し，適切な検査式を与えることで欠陥を検出できることを確認している．

宮本らは作成された状態遷移図と配置図で記述されたプログラムの仕様を promela 言語で記述された仕様に変換する手法を提案している [17]．この手法では，astah* professional が出力した状態遷移図と配置図を示す xml ファイルを入力として，配置図に含まれるインスタンスをプロセスに，状態遷移図に含まれる状態遷移を各プロセスが行う処理に変換することで promela ファイルを生成する．また，UML に含まれる仕様パターンの記述を LTL 式に変換することで，複雑な式を書くことなく SPIN でモデル検査を実行することを可能にしている．

長田らはシーケンス図などで記述された通信プロトコルの仕様から，通信プログラムのソースコードを生成する手法を提案している [18]．この手法では，シーケンス図やメッセージ形式定義言語で通信プロトコルを定義し，定義から通信プログラムを構築する．また構築の際，故障木図から例外処理を導出し，正常時処理に付加する．これにより，通信プログラムが例外発生時に行うべき例外処理の実装漏れを削減している．

Tiwari らはソフトウェアの仕様を記述したアクティビティ図を用いてテストケースを生成する手法を提案している [19]．この手法は，システムの処理の流れを表現するアクティビティ図からシステムが正常終了する条件を取得，その条件を反転してシステムの故障木図を取得する．これらを変換することでシステムの正常終了時のテストケース，および異常終了時のテストケースを生成する．

7. あ と が き

本研究では，非同期モデルでメッセージ交換が行われるシーケンス図を対象として，そのようなシーケンス図が持つメッセージ順序の曖昧性を除去する手法を提案した．また，2 種類の評価を行った．1 つ目の評価では提案手法が生成する修正候補の数や実行時間の評価を行った．2 つ目の評価では，開発現場で使われるシーケンス図に対して提案手法を適用し，修正候補を 10 個生成した．生成した修正候補を開発者に確認してもらったところ，実際のシーケンス図に含まれるメッセージ順序の曖昧性を妥当な時間で 40 %近い割合で検出し，修正候補を示せることが確認できた．今後の課題としては，より多くの実際に開発で使用されるシーケンス図に提案手法を適用することや，手法が示す修正候補のバリエーションを増やすことが考えられる．

謝辞 本研究は，日本学術振興会科学研究費補助金基盤研究 (S)(課題番号：25220003) の支援を受けて行われた．本研究の実施にあたり，アイデア，および実験データの提供を受けた三菱電機の原内聡様と村上享平様に感謝いたします．

文　　献

[1] S. Kaleeswaran, V. Tulsian, A. Kanade, and A. Orso, “MintHint: Automated Synthesis of Repair Hints,” In the Proceedings of the 36th International Conference on Software Engineering, pp.266–276, ACM, June 2014.
[2] W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest, “Automatically Finding Patches Using Genetic Programming,” In the Proceedings of the 31st International Conference on Software Engineering, pp.364–374, IEEE Computer Society, May 2009.
[3] H.D.T. Nguyen, D. Qi, A. Roychoudhury, and S. Chandra, “SemFix: Program Repair via Semantic Analysis,” In the Proceedings of the 35th International Conference on Software Engineering, pp.772–781, IEEE Press, May 2013.
[4] Y. Qi, X. Mao, Y. Lei, Z. Dai, and C. Wang, “The Strength of Random Search on Automated Program Repair,” In the Proceedings of the 36th International Conference on Software Engineering, pp.254–265, June 2014.
[5] R.H. Bourdeau and B.H. Cheng, “A formal semantics for object model diagrams,” IEEE Transactions on Software Engineering, vol.21, no.10, pp.799–821, 1995.
[6] “UML1.4”. http://www.omg.org/spec/UML/1.4/.
[7] “astah* professional”. http://astah.change-vision.com/ja/product/astah-professional.html.
[8] G.J. Holzmann, “The model checker SPIN,” IEEE Transactions on software engineering, vol.23, no.5, pp.279–295, 1997.
[9] V. Lima, C. Talhi, D. Mouheb, M. Debbabi, L. Wang, and M. Pourzandi, “Formal Verification and Validation of UML 2.0 Sequence Diagrams using Source and Destination of Messages,” Electronic Notes in Theoretical Computer Science, vol.254, pp.143–160, 2009.
[10] P. Baker, P. Bristow, C. Jervis, D. King, R. Thomson, B. Mitchell, and S. Burton, “Detecting and Resolving Semantic Pathologies in UML Sequence Diagrams,” In the Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering, pp.50–59, ACM, Sep. 2005.
[11] S. Bernardi, S. Donatelli, and J. Merseguer, “From UML Sequence Diagrams and Statecharts to Analysable Petri Net models,” In the Proceedings of the 3rd international workshop on Software and performance, pp.35–45, ACM, July 2002.
[12] D. Harel and S. Maoz, “Assert and Negate Revisited: Modal Semantics for UML Sequence Diagrams,” Software & Systems Modeling, vol.7, no.2, pp.237–252, 2008.
[13] H. Shen, R. Krishnan, R. Slavin, and J. Niu, “Sequence Diagram Aided Privacy Policy Specification,” IEEE Transactions on Dependable and Secure Computing, pp.1–1, 2014.
[14] B. Mitchell, “Characterizing Communication Channel Deadlocks in Sequence Diagrams,” IEEE Transactions on Software Engineering, vol.34, no.3, pp.305–320, 2008.
[15] “Lucidchart”. https://www.lucidchart.com/.
[16] “tracemodeler”. http://www.tracemodeler.com/.
[17] 宮本直樹，和崎克己，“UML 記述の仕様から SPIN モデル検査用 PROMELA モデルへの自動変換，” 情報科学技術フォーラム講演論文集，vol.9，no.1，pp.311–314，Aug. 2010.
[18] 長田知之，原内　聡，北村操代，山地　勉，上野泰秀，“故障木図からの例外処理の導出による通信プログラム構築手法，” 情報科学技術フォーラム講演論文集，vol.11，pp.45–48，Sep. 2012.
[19] S. Tiwari and A. Gupta, “An Approach to Generate Safety Validation Test Cases from UML Activity Diagram,” In the Proceedings of the 20th Asia-Pacific Software Engineering Conference, pp.189–198, IEEE, Dec. 2013.