Dynamic Updating Controller自動合成のための環境モデル構築における関心事分離手法

山内 拓人[1] 鄭 顕志[1,2] 鷲崎 弘宜[1,2] 本位田 真一[1,2]

Separation of Concerns in Environment Modeling for Dynamic Updating Controller Synthesis

TAKUTO YAMAUCHI[1] KENJI TEI[1,2] HIRONORI WASHIZAKI[1,2] SHINICHI HONIDEN[1,2]

1. 研究背景

継続的な稼働が求められるシステムでは，システムの更新時においてもその実行を停止することなくシステムを更新することが求められる．このような実行時更新を行う場合，更新前後だけでなく更新途中も不具合が生じないよう注意深く更新手順を設計・検証する必要がある．しかし，この更新手順の設計は開発者にとって負担となっている．

イベント駆動制御の分野において，人手による更新手順設計の負担を軽減し誤り混入を防ぐために，更新手順の自動合成手法 [1] が提案されている．[1] では更新途中においても指定された安全性要求を満たしつつシステムの更新を行う Dynamic Updating Controller (DUC) を自動合成する手法を提案している．

[1] ではこの DUC の自動合成を離散制御器合成問題として定式化することで実現している．離散制御器合成では，制御器の制御対象となる環境を Labelled Transition System (LTS) としてモデル化した環境モデルと，満たすべき要求を線形時相論理式として記述した要求モデルを入力とし，2 プレイヤーゲーム理論に基づいて要求を満たすことが保証された制御器を自動合成する．DUC の自動合成において，DUC が制御対象としているシステムの更新自体を表す環境モデル E_{map} の入力が必要となる．開発者は以下の 3 つの関心事を 1 つの E_{map} としてモデル化する．

E_{map} を設計する上で意識される 3 つの関心事

- 更新前のシステム実行環境を表す環境モデル (E)
- 更新後のシステム実行環境を表す環境モデル (E')
- E と E' 間の状態の対応づけを表現する遷移 (T_{update})

[1] 早稲田大学
[2] 国立情報学研究所 (NII)

図 1 の左上部に E_{map} のコード例を示す．黒色は E，青色は E'，橙色は T_{update} を表している．[1] では単一の環境モデル内にこれら特性の異なる 3 つの関心事が入り混ざってモデル化されている．

これら 3 つの異なる関心事が LTS モデルの遷移として区別なくモデル化されている．このような関心事の混在はモデルの理解性を損なうだけでなく，モデルの保守・再利用時の障害となる．例えば，E' とは異なる新たな環境モデル E'' への更新を扱う新たな E_{map} をモデル化する場合，既存の E_{map} から再利用可能な部分を切り出すことが単純ではなく，開発者が LTS 上の個別の遷移に対して注意深く再利用の可否を判断する必要がある．そのため，不具合等が混入する要因となりうる．

2. 提案

そこで本論文では，DUC 自動合成のための環境モデルにおける関心事を分離するモデリング手法 (提案 1) と，分離したモデルから E_{map} を合成する自動合成手法 (提案 2) を提案する．図 1 に提案手法の全体像を示す．提案 1 では，E_{map} をモデル化する代わりに，3 つのモデル (E・E'・T_{update}) を構築する．E，E' は LTS で表現され，T_{update} は E の状態から E' の状態に対する更新時の遷移をモデル化したもので表現する．提案 2 では，E と E' に T_{update} の更新動作の遷移を追加することで E_{map} を合成する．この提案 2 の合成を支援する合成ツールを構築した．

提案 1 について，E_{map}，E，E' は LTS で表現される．E_{map} の要素で E と E' で表現できない更新用の遷移のみである．そこで，更新用の遷移として T_{update} を設計として新たに用意する．T_{update} は，E の状態，E' の状態，更新用の遷移で表される．これによって，提案手法による適用範囲の変化はなく関心事ごとの設計が可能になる．

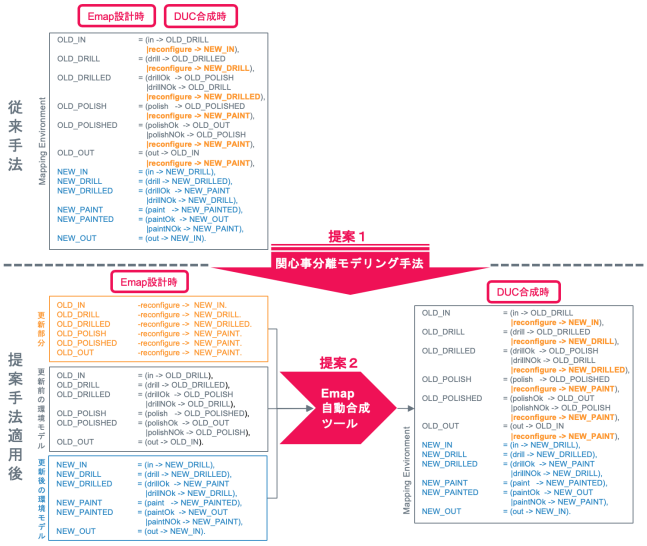

図 1 図左上 E_{map} の例とそれを分割設計する提案手法の概要

提案 2 について，T_{update} を用いて E と E' に更新用の遷移を追加し E_{map} を自動合成する．T_{update} で表された E の状態，E' の状態をそれぞれ E，E' から探索し，更新用遷移をその状態間に追加する．これによって E_{map} が合成できる．

3. 評価

本論文では，提案手法によって分離されたことによる影響を検証することによって，提案手法の有用性を評価する．提案手法による負の影響として，分離して設計を行った 3 つの関心事から E_{map} を自動合成する際にかかる合成時間がある．そこで [1] の論文中で用いられたモデルを用いて，E_{map} の自動合成にかかる時間を計測した．その結果が表 1 である．

表 1 E_{map} の合成時間と DUC 合成時間の比較

モデル名	E_{map} 合成時間 [ms]	DUC 合成時間 [ms]
choco	2.94	319
GSM	6.65	117
Industry	2.54	227
PowerPlant	2.87	294
ProductionCell	7.28	787

表より，E_{map} の自動合成にかかる時間は DUC の自動合成にかかる時間に比べて非常に小さいことがわかる．また，2 つの自動合成の使用上，自動合成にかかる時間が増加する要因は同じであるため，E_{map} の自動合成にかかる時間のコストが DUC の自動合成にかかる時間にとって表以上に大きくなることはないと言える．この結果から，提案手法によって生じる E_{map} 合成時間のオーバヘッドは無視できる程度とわかる．よって，提案手法は十分に有用であると言える．

4. 関連研究

DUC と関心事分離モデリング手法の効果について関連研究を用いて議論する．

実行時更新の関連研究として，自己適応システムにおける実行時更新を想定している [2] の論文がある．環境の変更に応じて，そのとき動作可能な制御器を判定し切り替える．この時，環境変更前で行われた動作全てを，モデルの初期状態から順に環境変化後の制御器で動作させることで，T_{update} を見つけることができる．これによって実行時更新を可能にしている．しかし，[2] では更新前の制御器で行えた動作は全て更新後の制御器で行える必要があるため，適用範囲は DUC に比べて小さい．

また，関心事分離モデリング手法の関連研究として，サービスロボットのアプリケーション設計で使われるドメイン固有言語 (DSL) での設計において，関心事分離モデリング手法を提案している [3] の論文がある．DSL をドメイン専門家，ロボティクス専門家，モデリング専門家の 3 つの関心事 (専門分野) によって設計を分けることでそれぞれの設計の検証・再利用が容易になると明示している．よって，本提案も [1] において同様の効果が見込まれる．

5. まとめ

本研究では 3 つの関心事ごとに個別に設計・保守可能にする，DUC 自動合成のための環境モデリング手法と，分離した 3 つのモデルから E_{map} を自動合成するツールを提案した．本提案によって，関心事の混在によって生じる E_{map} モデルの理解性損失が解消され，DUC 環境モデルの保守・再利用性が向上することが期待される．今後は，理解性・保守性・再利用性に関する評価の実施や，保守・再利用に関する支援手法の構築を行う予定である．

参考文献

[1] L. Nahabedian, V. Braberman, N. D'Ippolito, S. Honiden, J. Kramer, K. Tei, and S. Uchitel. Dynamic update of discrete event controllers. *IEEE Transactions on Software Engineering*, p. 21pages, 2018(Early Access).

[2] Nicolás D'Ippolito, Víctor A Braberman, Jeff Kramer, Jeff Magee, Daniel Sykes, and Sebastian Uchitel. Hope for the best, prepare for the worst: multi-tier control for adaptive systems. In *ICSE*, pp. 688–699, 2014.

[3] Kai Adam, Arvid Butting, Robert Heim, Oliver Kautz, Bernhard Rumpe, and Andreas Wortmann. Model-driven separation of concerns for service robotics. In *Proceedings of the International Workshop on Domain-Specific Modeling, DSM@SPLASH 2016, Amsterdam, Netherlands, October 30, 2016*, pp. 22–27, 2016.