R-4

初学者特有の問題点に着目した アクティビティ図のためのセルフレビュー支援手法の研究

橋浦研究室　　1145402　日原和志

1．はじめに

ソフトウェア設計では UML が標準として用いられている．要求分析の段階では UML の 1 つである．アクティビティ図が用いられているが，学習する際問題が多く発生する．

2．研究目的

前途で述べた問題点の発生はアクティビティ図が曖昧さを含む要求を表現する[1]必要があることが要因の 1 として挙げられる．初学者はこのような問題に対し，形式の確認や妥当性の検証ができず，このモデルは正しいかという疑問を持つことが多い[2]．本研究では，このような問題に対して問題点の修正方法などのレビューのクライテリアを提案する．このクライテリアを用いることで設計の問題点を初学者に認識させることが目的である．

表 1.問題点一覧

番号	問題点
1	開始ノードがない
2	終了ノードがない
3	パーティションがない
4	ガードなし
5	ガードが片側にしかない
6	次のノードが前のノードより上にある フローが他のフローと重なっている
7	分岐にデシジョンノードが使用されていない
8	分岐・合流を1つのデシジョンノード・マージノードで定義している
9	並行処理をデシジョンノード・マージノードで定義している
10	フォークノードしか使われていない
11	分岐以外でガードが記述されている
12	分岐にフォークノードを使用している
13	合流にジョインノードを使用している
14	アクションノードに実行が複数書き込まれている
15	フローが途切れている
16	分岐条件をノートに書いている
17	アクション代わりにノートを使って記述している
18	パーティションをノートで記述している
19	分岐条件がノートでアクションノードにつながれている
20	アクションノードに分岐条件が書かれている

3．提案手法

本研究で定義したアクティビティ図の問題点は表 1 に示す．これら 1 つ 1 つの問題点に対して，以下の項目からなるセルフレビューのためのクライテリアを定義した．

1)　修正前の図
2)　見つけ方
3)　理由
4)　修正手順
5)　修正後の確認基準
6)　修正後の例

4．実験

本研究で定義したルールの有用性を図るために以下の対象者と手順で実験を行った．

- 対象者
  アクティビティ図を学習したことがない日本工業大学情報工学科の学生
- 実験に使用したツール
  astah* community
- 比較するために使用したツール
  モデカツ![3]
- 実験手順
  それぞれ表 2 と表 3 に示す．

表 2.提案手法の場合

	実験手順
1	アクティビティ図の説明を行う
2	問題 1 を提案手法を使わずに解いてもらう(20分)
3	問題 1 を提案手法から修正・確認を行ってもらう(20分)
4	問題2を解いてもらう(30分)
5	アンケート

表 3.モデカツ！の場合

	実験手順
1	アクティビティ図の説明を行う
2	問題 1 を修正基準を使わずに解いてもらう(20分)
3	問題 1 を修正基準から修正・確認を行ってもらう(20分)
4	問題2を解いてもらう(30分)
5	アンケート

なお，問題 1 は実際に初学者に問題文からアクティビティ図を作成し，自らの手で修正する問題とであり，問題 2 は予め作成された問題点を含むアクティビティ図を修正する問題とする．

5. 評価手法

実験を行うにあたって評価する項目は，以下の 3 点である．

1. 修正・完成にかかった時間
2. 修正しきれなかった問題点の総数
3. 過程で使用したルール

2 の修正しきれなかった問題点の総数は，今回の提案手法で定義した問題の総数を評価する．

6. 実験結果と考察

表 4 から，提案手法とモデカツ！では問題を解く平均時間に差異は見られなかった．平均時間に差異が生じなかった原因としては，提案手法は確認基準が 20 項目であるのに対して，モデカツ！は 11 項目であり，アクションの修正基準のみを表示するため確認・修正作業が早く終わり，結果として差異が生じなかったと考えられる．

表 4. 問題回答の平均時間

	提案手法	モデカツ！
問題 1 修正前	0:15:54	0:19:08
問題 1 修正後	0:11:11	0:10:59
問題2	0:24:37	0:25:28

修正しきれなかった問題の総数は提案手法では 5 個，モデカツ！では 13 個となり，実際に今回定義したルールの問題点に対してはモデカツ！よりも改善が見られたと考えられる．

表 5，表 6 は適合率と再現率，F 値をまとめた表である．モデル要素に対して適合率 P と再現率 R を以下の式で求めた．

$$P = \frac{\text{対象者の要素の正答数}}{\text{対象者の全要素数}} \qquad R = \frac{\text{対象者の要素の正答数}}{\text{正答例の要素数}}$$

表 5. 提案手法の P と R，F 値

	問題1修正前			問題1修正後			問題2		
	P	R	F値	P	R	F値	P	R	F値
A	0.9375	1.0000	0.9677	0.9375	1.0000	0.9677	0.8400	1.0000	0.9130
B	1.0000	0.9375	0.9677	1.0000	0.9375	0.9677	0.8823	0.9375	0.9090
C	0.8750	0.8750	0.8750	0.8750	0.8750	0.8750	0.8235	0.8750	0.8485
D	0.5330	0.5310	0.5660	0.6000	0.7500	0.6666	0.8667	0.8125	0.8387

表 6. モデカツ！の P と R，F 値

	問題1修正前			問題1修正後			問題2		
	P	R	F値	P	R	F値	P	R	F値
E	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9231	0.7500	0.8276
F	0.3636	0.2500	0.2963	0.5000	0.3125	0.3846	0.7143	0.6250	0.6667
G	0.9412	1.0000	0.9697	0.9412	1.0000	0.9697	0.6667	0.7500	0.7059
H	0.7647	0.8125	0.7879	0.7647	0.8125	0.7879	0.6250	0.6250	0.6250

F 値を求めた結果から，問題 1 修正後では F 値の差異はあまり生じていないが，問題 2 では F 値の差異から提案手法を用いた方が良い結果となった．この結果となった原因は，並行処理の記述方法に対して，モデカツ！の対象者に多く間違えが見られたことが原因であると考えられる．

過程で使用されたルールの総数を表 7，表 8 に示す。これらを平均すると提案手法では問題 1 で 8.5 個，問題 2 で 4 個が平均として使用されている．これに対してモデカツ！では問題 1 で 7.75 個，問題 2 で 2.5 個が平均として使用されている．このような結果となった原因は，提案手法がモデカツ！よりも項目が多いことから多く使用すること，問題 2 では問題 1 使用したルールを覚えているため，問題 1 よりもルールを使用することが少なくなったと考えられる．

表 7. 提案手法使用総数

	問題 1 修正後	問題2
A	12	3
B	8	7
C	5	1
D	9	5
平均	8.5	4

表 8. モデカツ！使用総数

	問題 1 修正後	問題2
E	12	1
F	2	7
G	0	2
H	17	0
平均	7.75	2.5

7. まとめと今後の課題

本研究では，アクティビティ図の記述方法を中心とした提案手法を提案し，初学者に有効であるかどうかを比較対象と実験を行った．その結果，平均時間には差が生じないが，比較対象より定義した問題点が取り除かれることが結果として現れた．一方で，本実験から定義を行っていない問題点が発生することが確認された．具体例としては開始ノードがパーティションの枠外に配置されている，他のノードから開始ノードにフローが流れている等である．そのため，今後は今回発見された問題点に対して改善ルールを定義していく必要がある．

参考文献

[1] 小木曽 禎，遠山 紗矢香，湯浦 克彦，“学生向けモデリング演習支援システムの開発と評価，” 研究報告コンピュータと教育 (CE)，2011-CE-109 巻，5 号，pp. 1-9，2011-03-11.

[2] 赤山 聖子，久住 憲嗣，久保秋 真，部谷 修平，福田 晃，“効果的なオブジェクト指向モデリング教育のための実行可能モデリング言語の比較評価，” 情報教育シンポジウム 2013 論文集，2013 巻，2 号，pp. 125-132，2013-08-11.

[3] 斎藤 智之，平田 雅弥，“モデカツ！：アクティビティ図の記述支援機能を持つモデリングツールの研究”，平成２８年度日本工業大学情報工学科卒業論文抄録集，pp. 187-188，2017.