Vol.2015-SLDM-170 No.53
Vol.2015-EMB-36 No.53
2015/3/7

プロジェクト開発の不確定要素が及ぼす影響分析

山﨑 友貴[1,a)]　久住 憲嗣[2,b)]　福田 晃[3,c)]

概要：組込みシステムは各種デジタル家電や自動車，人工衛星などの製品に用いられており，現在，様々な分野において広く普及している．しかし，組込みシステム産業の進展とともに，新しい課題も生まれており，その一つに仕様変更による手戻りの防止がある．組込みソフトウェアの提供する機能の複雑化により，要求定義の時点で不確定な要素が存在する場合や，急速なビジネス変化によるソフトウェア発注者側の仕様変更が行われた場合，設計したモデルが変更に耐えられず，ソフトウェアの開発において手戻りが発生する．手戻りが起こった場合，要求仕様段階で作成したドキュメントの見直しを行う必要があり，設計段階で定義したモデルは修正を余儀なくされる．その手戻りの影響を軽減させるために，設計段階で作成される UML モデルから不確定要素による変更の影響度を示すことができれば，変更容易性の高いモデルかどうかを判定することができると考えられる．従って本研究では，不確定要素がプロジェクトに与える影響を設計段階で定量的に分析する尺度を提案し，検証実験によってその有用性を評価した．結果として，人間の UML 変更容易性と影響度には強い相関が見られることが分かった．

1. はじめに

組込みシステムは各種デジタル家電や自動車，人工衛星などの製品に用いられており，現在，様々な分野において広く普及している．しかし，組込みシステム産業の進展とともに，新しい課題も生まれており，その一つに仕様変更による手戻りの防止がある．組込みソフトウェアの提供する機能の複雑化により，要求定義の時点で不確定な要素が存在する場合や，急速なビジネス変化によるソフトウェア発注者側の仕様変更が行われた場合，設計したモデルが変更に耐えられず，ソフトウェアの開発において手戻りが発生する．不確定な要素とはつまり不確定な要求のことであり，本研究では，その要求を不確定要素と定義している．

手戻りが起こった場合，要求仕様段階で作成したドキュメントの見直しを行う必要があり，設計段階で定義したモデルは修正を余儀なくされる．従って，製造段階に影響が出ることは明確である．その手戻りの影響を軽減させるために，設計段階で作成される UML(Unified Modeling Language) モデルから不確定要素による変更の影響度を示すことができれば，変更容易性の高いモデルかどうかを判定することができると考えられる．よって本研究では，不確定要素がプロジェクトに与える影響を設計段階で定量的に分析する尺度を提案し，検証実験によってその有用性を評価することを目的とする．

[1] 九州大学大学院システム情報科学府
[2] 九州大学システム LSI 研究センター
[3] 九州大学大学院システム情報科学研究院
[a)] yamasaki@f.ait.kyushu-u.ac.jp
[b)] nel@slrc.kyushu-u.ac.jp
[c)] fukuda@f.ait.kyushu-u.ac.jp

2. 関連研究

2.1 不確定要素

システム開発において，必要な情報のうち，開発者によって認知され，定義され，命名され，かつ未解決となっているものを不確実性，または不確定要素と呼ぶ [1]．一般的にはシステム開発の初期段階において，開発者と顧客の間に一定のコミュニケーションの障壁があるため，曖昧な要求や矛盾点が存在する要求が要求仕様書に含まれる可能性がある [2]. 不確定要素とはつまり，不確定な要求のことであり，その要素が含まれたまま設計開発段階に移行するとプロジェクトの手戻りが発生する可能性が高まる．手戻りが発生した場合，プロジェクトは要求仕様の段階から見直さなければならないため，時間のコストがかかる．更に，プロジェクト開発には納期があるため，その期日までにシステムを完成させなければならず，人員増強によるコストも発生する．従って，手戻りは未然に防ぐ必要がある．

2.2 既存の影響分析手法

本節では，プロジェクトの手戻りの影響を分析し，軽減させる影響分析の既存手法を 3 種類紹介する．

不確定要素を含む要求・運用・設計モデリング手法
モデル構築手法は，開発初期段階で仕様が確定できず開発後期や開発進捗に沿って仕様が決定するシステムを対象としている [2]．不確定要素が要求仕様に含まれている場合のシステム開発をターゲットとしており，不確定要素一覧シート，ケース一覧シート，フィーチャモデル，部分モデルを作成し，仕様変更が起きた際，手戻りを発生させることなく開発が進められるような手法を提案している．この手法の特徴として，不確定要素一覧シートやケース一覧シートなどで影響分析を行い，起こりうる全てのパターンを網羅した部分モデルを作成することで手戻りによる影響を極限まで抑えることができる，ということが挙げられる．しかし，この手法を導入するにあたり，必要となる知識が多く存在するため，導入コストが大きいことが欠点である．更に，全てのパターンを網羅する部分モデルを作成するコストも大きい．

Automated impact analysis of UML models
大規模プロジェクトを分析，設計する場合，多数の相互依存の UML モデルを用いる [3]．自動化ツールを用いたモデルベースの影響分析では，自動化ツール iACMTool を用いて，要求仕様から設計したクラス図，シーケンス図を基に変更前，変更後の差分を測定し，その影響度として影響分析尺度 (distance) を算出していた．この手法はコードベースではなく，モデルベースの影響分析であるため，モデルの変更段階でその後の影響度を確認できることが特徴として挙げられる．しかし，一般的に組込みシステム開発ではシーケンス図ではなく，ステートマシン図が多用されるため，この影響分析手法を適用することができない．

Software Change Impacts-an Evolving Perspective
ソフトウェアライフサイクル・オブジェクト (SLO) ベースの影響分析では，変更要求のセットで提案されている各ソフトウェアで変更される可能性が高いソフトウェアライフサイクル・オブジェクト (SLO) を識別することによって，リリース計画をサポートするものである [4]．この影響分析は，各ソフトウェアで変更される可能性が高い SLO の関連度を用いて分析を行っている．影響の種類には直接的な影響と間接的な影響があり，直接的な影響のある SLO は第 1 レベルの影響度，間接的な影響のある SLO は第 N レベルの影響度として定義され，それらを上に示すような影響分析尺度として影響分析を行っている．

2.3 メトリクス解析

2.2 の既存手法で用いられている影響分析尺度は非常に有用であると考えられる．なぜなら，影響分析を行うにあたり，モデルの変更による影響を定量的に数値で示すことが重要であるからである．その影響分析尺度に類似したものとして，メトリクスという概念が一般的に知られている．以降，提案手法で用いたモデルベースのメトリクスを紹介する．

2.3.1 クラス図のためのメトリクス：CK メトリクス

CK メトリクスとは，Chidamber と Kemerer によって提案されたメトリクスであり，オブジェクト指向ソフトウェアの複雑度を測定するものである [5]．オブジェクト指向ソフトウェアの性質を測定するメトリクスとして広く用いられている [6]．各メトリクスについて，以下で述べる．

WMC(Weighted Methods per Class)
そのクラスに定義されているメソッドに重み付けをして足し合わせた値である．重み付けの方法としてサイクロマチック数 [7] などが用いられる．

DIT(Depth of Inheritance Tree)
クラス階層における継承の深さを表す．

NOC(Number Of Children)
サブクラスの数を表す．

CBO(Coupling Between Objects)
関係しているクラスの数を表す．

RFC(Response For a Class)
クラスのレスポンスとは，そのクラスのオブジェクトにメッセージが送られた結果，実行される全てのメソッドの集合である．

LCOM(Lack of Cohesion Of Methods)
クラスの凝集とは，クラスのメソッドがお互いに関連している程度を表す．

2.3.2 ステートマシン図のためのメトリクス

ステートマシン図のメトリクスは，その図のサイズと構造複雑度を判断するために用いられる [8]．各メトリクスについて，以下で述べる．

- サイズメトリクス

  NEntryA(Number of Entry Actions)
  entry アクションの総数．すなわち，状態に入る際に実行されるアクション数．

  NExitA(Number of Exit Actions)
  exit アクションの総数．すなわち，状態から離れる際に実行されるアクション数．

  NA(Number of Activities)
  do アクティビティの総数．

  NSS(Number of Simple States)
  状態の総数 (合成状態内のシンプル状態の数も含む)．

  NCS(Number of Composite States)
  合成状態の総数．

  NE(Number of Events)
  イベント (トリガー) の総数．

NG(Number of Guards)
ガード条件の総数.

- 構造複雑度メトリクス

NT(Number of Transitions)
遷移の総数 (ステートマシン図の最初と最後の遷移, 自己遷移, 内部遷移の数も含む).

McCabe(Cyclomatic Number of McCabe)
McCabe の循環複雑度 [7].
McCabe = |NT| − |NSS| + 2

上記したメトリクスと人間の理解容易性時間に相関があるかどうかをコルモゴロフ-スミルノフ検定を用いて調査した結果，9 つのメトリクスのうち，NA，NSS，NG，NT に有意な相関が見られた [8]．故に，本研究では，これら 4 つのメトリクスを影響分析に用いる．

3. 提案手法

3.1 概要

組込みシステムの開発段階において，プロジェクトの不確定要素部分が確定した場合，設計段階で設計したモデルはそのインパクトを受け，変更を余儀なくされる．その手戻りの影響度を定量的に分析するために，モデルベースの影響尺度を定義する．既存の影響分析手法の問題点は，組込みシステム開発で広く用いられているステートマシン図に適用できるものが存在しない，影響分析手法の導入コストが高い，であった．提案手法ではそれらの問題点を踏まえ，組込みシステム開発の設計段階の成果物であるクラス図，ステートマシン図に独自の複雑度算出式を適用し，手戻りの影響度を算出する方法を採用した．複雑度算出式は 2.3 で紹介したメトリクスを組み合わせて導出する．

3.2 クラス図の複雑度算出式

測定目的は，不確定要素による変更でクラス図に変更を加える際に，変更によってかかる時間コストがどの程度であるかを定量的に測定することである．一般的にクラス図の複雑度は CK メトリクスを測定することで得られるが，本研究は不確定要素による要求定義の変更がプロジェクトにどの程度の影響を及ぼすかについて考えているため，単なる CK メトリクスを測定しただけでは目的の結果が得られない．以上を踏まえて定義される測定課題としては，1 つのクラスが大きすぎる，クラス同士が結合しすぎている，の 2 つである．従って，CK メトリクスから選択されるメトリクスは，対象クラスの重み付きメソッド数を計測する WMC，対象クラスに結合しているクラス数を計測する CBO である．以下にクラス図の複雑度算出式を示す．

$$\text{class complexity} = \sum_{i=1}^{n} \left\{ \alpha(\text{WMC}_i) + \beta(\text{CBO}_i)) \right\} \quad (1)$$

(1) 式における i は測定対象クラス番号，n はクラス数である．また，第一項の重み付けを $\alpha = 0.5$，第二項の重み付けを $\beta = 0.5$ としている．

3.3 ステートマシン図の複雑度算出式

測定目的は，不確定要素による変更でステートマシン図に変更を加える際に，変更によってかかる時間コストがどの程度であるかを定量的に測定することである．ステートマシン図の複雑度においてもクラス図と同様，不確定要素を考慮して測定課題を定義する必要がある．課題としては，状態数の割に遷移が多い，システムに大きく影響を与えている要素が多い，の 2 つである．状態数の割に遷移が多いとは，状態のループが多いということである．ループが多用されていると，そのループの中心の状態が変更になった際に多くの遷移先の状態も見直さなければならず，変更コストが高くなってしまう．また，システムに大きく影響を与えている要素として，do アクティビティやガード条件などが挙げられる．do アクティビティはその状態の中で実行し続けるアクションである．更に，do アクティビティは遷移先の状態に遷移した瞬間実行し，その次の状態に遷移する瞬間実行を止めなければならないため，高度なリアルタイム処理が要求されている [8]．従って，ステートマシン図における do アクティビティの比重は，entry アクションや exit アクションに比べ大きく，その分影響度は大きいことが分かる．また，ガード条件の数が多いと，変更があった際にその遷移の端の状態の内容も見直す必要があるため，複雑な変更を要する．ステートマシン図を変更するためには，その図が人間にとって理解しやすいものではなければならない．つまり，人間の理解容易性と相関があるメトリクスが必要である．2.3.2 で紹介したメトリクスの中で，理解容易性と相関が見られ，かつ上記した課題を解決するメトリクスは，NSS，NA，NG，NT である．以下にステートマシン図の複雑度算出式を示す．

$$\text{stm complexity} = \sum_{i=1}^{m} \left\{ \alpha(|\text{NT}| - |\text{NSS}| + 2) + \beta\left(\frac{\text{NA}}{\text{NSS}}\right) + \gamma\left(\frac{\text{NG}}{\text{NT}}\right) \right\} / m \quad (2)$$

(2) 式における i は測定対象のステートマシン図番号，m はステートマシン図の数である．また，第一項の重み付けを $\alpha = 2$，第二項の重み付けを $\beta = 10$，第三項の重み付けを $\gamma = 20$ としている．

3.4 プロジェクト全体の複雑度算出式

3.2，3.3 で示したクラス図，ステートマシン図それぞれの複雑度算出式を基に，プロジェクト全体の複雑度算出式を導出した．一般的に，プロジェクト 1 つに対して 1 つの

クラス図，1 クラスに対して内部の動きを表すステートマシン図が 1 つ設計される．そのため，クラス図の複雑度で算出した値と，各ステートマシン図の複雑度を合算した値の平均値で表すのが妥当であると思われる．以下に，その式を示す．

$$\begin{aligned}\text{project complexity} = &\text{class complexity} \\ &+ \text{stm complexity}\end{aligned} \tag{3}$$

式内の class complexity，stm complexity は，それぞれ (1)，(2) 式である．但し，(2) 式の重み付けの係数で使用した α，β，γ は γ，ζ，η に変更している．なお，各項の重み付けは，$\alpha = 0.5$，$\beta = 0.5$，$\gamma = 2$，$\zeta = 10$，$\eta = 20$ である．これら係数はクラス図，ステートマシン図のそれぞれのメトリクスの影響度が妥当なものとなるように設定している．例えば，NA/NSS や NG/NT は割合で表されるため，その他のメトリクスよりも複雑さに与える影響が小さくなる．従って，$\zeta = 10$，$\eta = 20$ とした．しかし，メトリクス値が妥当であるかどうかは適用するプロジェクトごとに異なるため，微調整はその都度必要であると思われる．

4. 検証実験

4.1 実験目的

この検証実験の目的は，人間のモデルの変更容易性と影響分析における影響度に相関があるかを調査するものである．モデルを変更する際に，変更者がそのモデルを変更しづらいものと考えるか，容易なものと考えるかはその人物の知識量，経験量に依存する．しかし，クラス図やステートマシン図をどのように設計すれば，設計者以外がそのモデルを理解しやすいかは，組込みシステム開発に携わった者であればある程度予測できると思われる．従って，本実験は組込みシステム開発の経験がある者を被験者として行う．

4.2 実験方法

本節は，検証実験の方法について述べる．検証実験は，架空の組込みシステム開発プロジェクトを用いて行う．架空プロジェクトは荷物の集荷，運搬システムの開発である．実験フェーズは，学習段階，要求仕様理解段階，UML 変更段階，評価段階の 4 段階に分かれているが，UML 変更段階については，複雑度の異なる 3 種類の UML モデル (要求仕様は同一) に対して行うため，その段階を 3 回繰り返すものとする．

4.3 実験で使用した UML モデル

本節では，実験で被験者に変更を依頼したクラス図，ステートマシン図について詳細を述べる．なお，図の掲載は実験 1 のみとする．実験 1，2，3 の UML モデルはそれぞれ同一の要求仕様に準するものであり，複雑度が異なるようクラスの分け方や状態の繋がりを変化させている．各実験の UML モデルの特徴としては，以下の通りである．

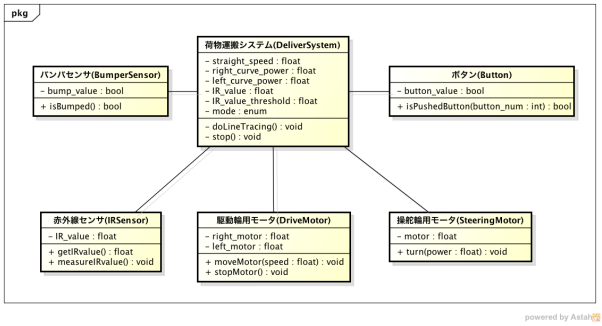

図 1 クラス図 (実験 1)

表 1 人間のモデル変更容易性

UML 名	平均	標準偏差
実験 1	3.80	1.245
実験 2	3.00	0.350
実験 3	3.30	1.161

実験 1 クラス図は単純だがステートマシン図が複雑である．

実験 2 クラス図とステートマシン図の複雑さが均等である．

実験 3 クラス図は複雑だがステートマシン図が単純である．

図 1，図 2 は実験 1 のクラス図，荷物運搬システムクラスのステートマシン図である．

4.4 実験結果

4.2 の実験方法に基づいて，10 人の被験者に実験を行った．被験者は全員が組込みシステム開発経験者であり，クラス図，ステートマシン図共に意味を理解している．実験結果は，人間が感じるモデルの変更容易性である．実験結果は，人間が感じるモデルの変更容易性を 5 段階で評価したものであり，1 が容易，5 が困難であることを表す．表 1 に結果を示す．

5. 複雑度算出式の妥当性検証

本章では，4 章で行った検証実験の実験値を基に，3 章で示した複雑度算出式の妥当性検証，評価を行う．

5.1 検証方法

複雑度算出式の検証方法を示す．まず，理論値を算出するために，4.3 で示した UML モデルを要求仕様の変更に沿って変更する．次に，変更前，変更後の 2 種類の UML モデルのそれぞれの複雑度を (3) 式 (プロジェクト全体の複雑度算出式) を用いて算出，その複雑度から以下の (4) 式を用いてプロジェクトの影響度を導出し，それを理論値と

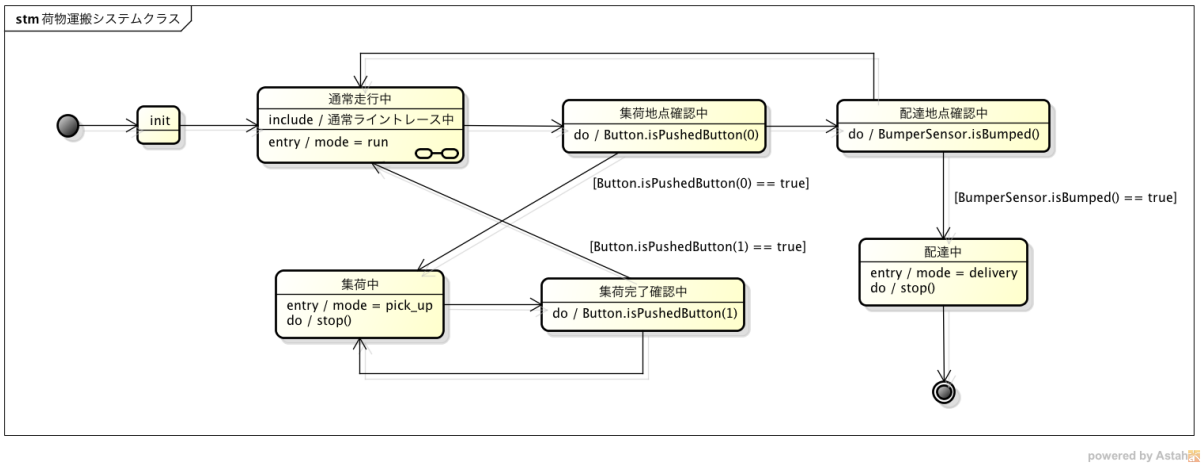

図 2 荷物運搬システムクラスのステートマシン図 (実験 1)

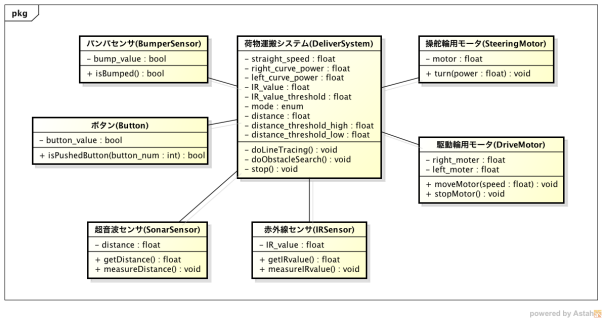

図 3 変更後のクラス図 (実験 1)

表 2 クラス図，ステートマシン図の複雑度

UML 名	測定対象	複雑度	
		変更前	変更後
実験 1	クラス図	9.50	11.50
	ステートマシン図	31.82	42.87
実験 2	クラス図	10.50	17.50
	ステートマシン図	22.55	24.91
実験 3	クラス図	12.00	18.00
	ステートマシン図	17.55	21.58

する．但し，(4) 式は検証のために使用する影響度算出式である．今回は，不確定要素によるモデルの変更前後の複雑度の差分が，不確定要素によって生じた変更コストである．つまり，モデルに与えた影響を差分によって分析することができる．実際の影響分析における (4) 式の位置付けは 6 章で示す．最後に，理論値と検証実験の実験値に相関が見られるかどうかを判断する．

$$\text{influence rate} = abs((\text{project1 complexity}) - (\text{project2 complexity})) \quad (4)$$

5.2 理論値の算出

5.2.1 変更後の UML モデル

理論値算出の際に記述した，実験 1 の要求仕様による変更後の UML モデルを示す．

図 3 は，4.3 で示した変更前のクラス図の変更後のものである．実験 1 では荷物運搬システムクラスがロボットの動作制御の全てを行っていたので，障害物回避の動作もそのクラスで行うような変更方法を取った．図 1 と比較して，クラス構成はほぼ変化していないが，荷物運搬システムクラスの役割が非常に大きいことが分かる．

5.2.2 影響度の算出結果

4.3 の変更前 UML モデル，5.2.1 の変更後 UML モデルのそれぞれに 3.4 で示した (1) 式，(2) 式を適用し，クラス図，ステートマシン図の複雑度を算出した．算出結果を表 2 に示す．実験 1 の UML モデルはクラス図は単純だが，ステートマシン図が複雑であり，実験 3 はクラス図が複雑である代わりにステートマシン図が単純である．実験 2 に関しては，2 つの図の複雑さがバランスを保っている．表 2 に示す値を見ると，クラス図の複雑度が最も高いものは実験 3 であり，ステートマシン図の複雑度が最も高いものは実験 1 となっていることが分かる．よって，UML モデルの複雑さとそれをメトリクスにより定量的に算出した値との相関が見られることが確認された．

また，表 2 の複雑度から (4) 式を用いて変更による影響度を算出した．算出結果を表 3 に示す．各 UML モデルの影響度を見ると，ステートマシン図が複雑であるほどその度合いが高くなることが分かる．ステートマシン図はシステムの動的な流れを示したものであるため，変更するには現在の状態に追加する作業だけではなく，一から流れを組み直す必要性がある．そのため，ステートマシン図の複雑さがプロジェクトに大きく起因していると思われる．

5.3 検証結果

本節では，4.4 で示した人間のモデル変更容易性と，5.2.2 で示した複雑度算出式から算出した UML モデル変更による影響度に相関が見られるかを検証する．図 4 は理論値と実験値の比較結果である．左軸は実験値である人間の変更

表 3 プロジェクトの影響度

UML 名	複雑度		影響度
	変更前	変更後	
実験 1	41.32	54.37	13.06
実験 2	33.05	42.41	9.36
実験 3	29.55	39.58	10.03

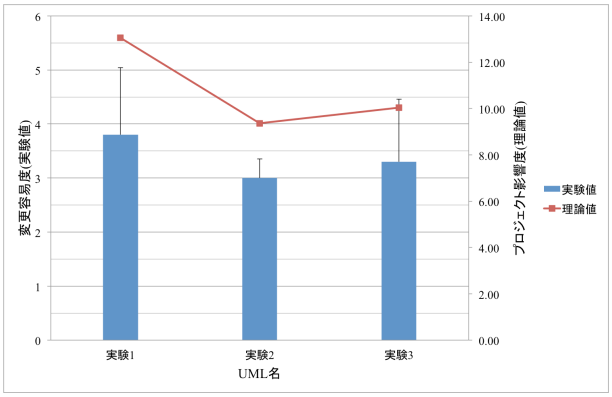

図 4 理論値と実験値の比較

容易性，右軸は理論値であるプロジェクト影響度を示す．下軸は複雑度の異なる UML 名を表している．また，棒グラフは実験値，折れ線グラフは理論値である．更に，実験値の標準偏差を棒グラフ上にプロットしている．前章でも述べたが，実験 1，2，3 のそれぞれの UML モデルの特徴は，クラス図は単純だがステートマシン図が複雑，クラス図とステートマシン図の複雑さが均等，クラス図は複雑だがステートマシン図が単純である，点である．比較結果を確認すると，実験 1，2，3 全てにおいて，理論値と実験値の特徴はほぼ一致していることが見てとれる．ピアソンの相関係数では 0.978 で正の強い相関が見られたが，有意確率は 0.1334 であった．有意確率が 0.05 以上となった原因として，人間のモデル変更容易性がその者の知識量，経験量に依存してしまう，ことが挙げられる．本研究では 3 種類の複雑度の異なる UML モデルを変更してもらったが，それを変更する被験者の経験が不足しており，3 種共に知識不足により変更が容易ではなかったと判断した可能性もあると思われる．今回使用している実験値は全て，組込みシステム開発経験のある被験者から取ったものであるため，全くの知識不足である者はいないと考えられる．しかし，その知識量にはばらつきが含まれているため，それが起因して有意水準が 95%に満たなかったものと推測する．以上を考慮してまとめると，やや有意性は劣るが，人間の UML 変更容易性と複雑度算出式から算出した影響度には強い相関が見られることが分かった．

6. おわりに

本研究では，不確定要素がプロジェクトに与える影響を設計段階で定量的に分析する尺度を提案し，検証実験によってその有用性を評価することを目的としていた．検証実験の被験者には，要求仕様理解段階を経た後，不確定要素による要求仕様の変更をシステムで実現できるよう UML モデルの変更を試みさせた．上に示した 3 種類の UML モデルを変更した後，評価段階において，変更段階で行った変更がどの程度難しかったかを定量的に示してもらうためのアンケートに答えてもらい，それを実験値とした．更に，事前に用意した変更前，変更後の UML モデルにプロジェクト全体の複雑度算出式を適用し，メトリクスによるプロジェクト全体の変更による影響度を算出し，理論値とした．そして，理論値と実験値に相関が見られるかどうかを判断し，複雑度算出式の妥当性検証，評価を行った．

結果として，ピアソンの相関係数では 0.978 で正の強い相関が見られたが，有意確率は 0.1334 であった．有意確率が 0.05 以上となった原因として，人間のモデル変更容易性がその者の知識量，経験量に依存してしまう，ことが挙げられる．今後の課題として，実際に影響分析を行う場合，設計したモデルに対して (3) 式を適用し，手戻りが起こりやすいプロジェクトであるかを確認する必要がある．その際，手戻りが起こる確率が高いモデルである基準値が必須であるが，その値は適用プロジェクトによって変動する可能性があるため，過去の類似プロジェクトに (3) 式，(4) 式を適用し，基準値を作らなければならない．その基準値を定めるプロセスの定義を行うとともに，より有意な相関が見られるプロジェクト全体の複雑度算出式を導出する予定である．

参考文献

[1] 中西 恒夫, 馬 立東, 久住 憲嗣, 福田 晃, “システム開発のための不確実性フレームワーク,” ETNET2014, pp.31–36, 2014 年 3 月.
[2] 陳 辰, 久住 憲嗣, 片平真史, 西原 雄次, 河合 東, 中西 恒夫, 福田 晃, “不確定要素を含む要求・運用・設計モデリング手法,” ESS2013, 2013 年 10 月.
[3] L.C. Briand, Y. Labiche, L. O ’Sullivan, and M.M. Sówka, “Automated impact analysis of UML models,” *The Journal of Systems and Software*, Vol.79, Issue 3, pp.339–352, Mar. 2005.
[4] Shawn A. Bohner, “Software Change Impacts-an Evolving Perspective,” *Proc. of the ICSM*, pp.263–272, 2002.
[5] S.R. Chidamber, C.F. Kemerer, “A metric suite for object-toriented design,” *IEEE Transactions on Software Engineering*, Vol.20, Issue 6, pp.476-493, Jun. 1994.
[6] 肥後 芳樹, 楠本 真二, 井上 克郎, “コードクローンを対象としたリファクタリングの有効性に関する調査,” 電子情報通信学会技術研究報告, SS2006-30, Vol.106, No.201, pp.37–42, 2006.
[7] T.J. McCabe, “A complexity measure,” *IEEE Transaction on Software Engineering*, Vol.SE-2, Issue 4, pp.308-320, Dec. 1976.
[8] C. Calero, M. Genero, and M. Piattini, *Metrics for Software Conceptual Models*, Imperial College Press, 2005.