15-A-7

D-Case 導入によるシミュレーションＳ／Ｗの期待結果明確化と合意形成 [1]

1. 概要

本編では、企業におけるシミュレーションソフトウェア開発において D-Case を適用した事例を紹介する。

D-Case とは、一般社団法人 ディペンダビリティ技術推進協会（DEOS 協会）で提唱されている、システムのディペンダビリティについて説明責任を果たし、合意形成するためのツールである。近年欧米で普及している安全性ケース（Safety Case）を拡張して開発された[1]。

一方、ソフトウェア開発において、事前に期待結果を明確化することが困難な場合、システムとしての妥当性確認が適切に実施されない、確認方法についてステークホルダ間で合意が得られない、というリスクがある。このリスク対策として、対象製品に求める特性を定義し、見える化することにより、妥当性確認項目を洗い出す、という活動が必要である。その考え方は、D-Case の目的である説明責任の遂行、合意形成と一致しており、ソフトウェア開発における妥当性確認に D-Case を導入することにより、期待結果の明確化が困難な対象でも妥当性確認が可能となり、ディペンダビリティを確保できるものと考える。

本事例で紹介する活動では、シミュレーションソフトウェア開発に D-Case を導入することにより、期待結果の明確化とステークホルダ間の合意に成功し、手戻りを防止することができた。その詳細について紹介する。

2. 取組みの目的

ソフトウェア開発において、ソフトウェアの要求事項の妥当性を確認する方法が曖昧で、かつステークホルダ間で合意されないまま開発が進み、開発後期の大きい手戻りや出荷後不具合が発生するケースがある。

本事例の対象製品のシミュレーションソフトウェアは、ユーザーが設定した複数のパラメータを用いてシミュレーションを行い、パラメータ間の順位付けをするという仕様である。ユーザーはシミュレーション結果を参考に、最良のパラメータを選択する。このソフトウェアのテストでは、順位付けが妥当であることを確認する必要があるが、そもそも既知でない情報を獲得するためのシミュレーションであり、事前に明確な期待結果を定義することは困

[1] 事例提供: 三菱電機株式会社 通信機製作所　森 素子 氏

難である。

2012 年度に対象製品の第 1 バージョンを開発した際、開発の後期まで誰も期待結果を決めることをせず、関数単体レベルの確認しか行わなかった。結果、ソフトウェア適格性テスト完了後に有識者が確認した際、経験則との不一致が発覚し、仕様を根本から見直す手戻りが発生した。完全な正解がわからないにしろ、何を拠り所として、期待結果をどうするか、という点においてステークホルダ間での合意が欠落していたことが原因である。このような合意の不足は、シミュレーションだけでなく、保守性、操作性、信頼性等の非機能要求のように、定量的に要求を決めにくい場合も発生しうる。

2013 年度、対象製品の第 2 バージョンを開発することになり、第 1 バージョンのような期待結果不明確による手戻りを防止する対策が必要であった。ちょうど同時期、ソフトウェア品質シンポジウム（SQiP シンポジウム）2013 において D-Case が紹介され、D-Case の目的である説明責任の遂行、合意形成が本事例の課題とも一致すると考えた。そこで、シミュレーションの期待結果を明確にし、ステークホルダ間で合意を形成することにより、開発の手戻りコストを削減することを目的に、D-Case 導入に取り組むこととなった。

3. 取組みの対象、適用技術・手法、評価・計測

3.1. 対象製品

本事例の取組み対象とする製品は、ある特殊なドメインに関するシミュレーションを行うソフトウェアである。ユーザーが設定した複数のパラメータ群に対し、シミュレーション計算によりそれぞれの評価値を算出し、評価値の順位を表示する。ユーザーは表示された順位を参考に、実環境で使用するパラメータを決定する。

対象製品の概要を図 15-A-7-1 に示す。ソフトウェアの種類はエンタープライズ系、開発言語は C#（画面）・MATLAB（シミュレーション計算）、規模は約 50KL、開発期間は約 1 年（2012 年度～2013 年度）であった。

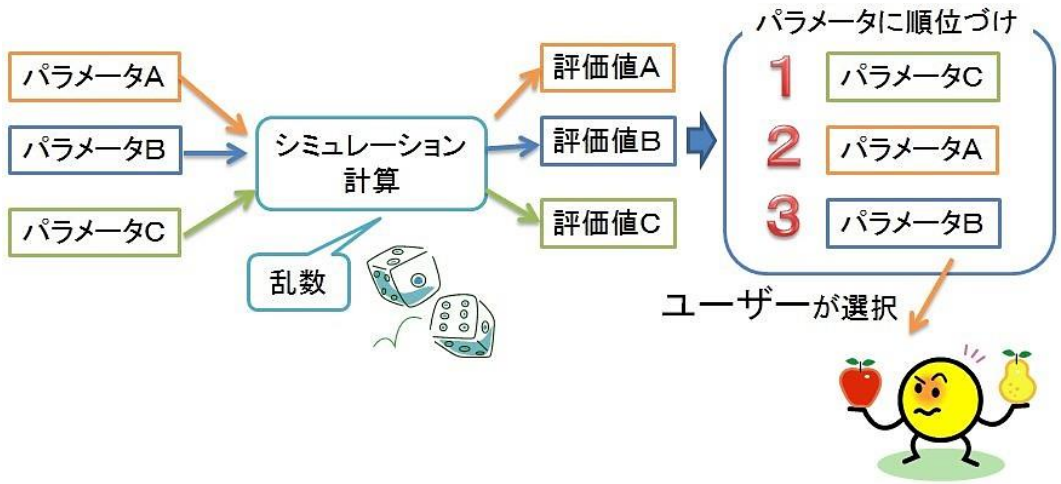

図 15-A-7-1　対象製品の概要

3.2. 開発体制

開発体制は、上流設計者 1 名、開発チーム 4 名の計 5 名であった。開発チームの内訳は、リーダー層 1 名、入社 5 年目 1 名、入社 2 年目 1 名、新人 1 名であり、若手中心のチームであった。

3.3. 開発における課題

3.3.1. 対象製品の特長

対象製品であるシミュレーションソフトウェアは、シミュレーション計算に乱数を使用しており、事前に期待結果を作成することが困難であった。また、特殊なドメインに関する計算であり、社内の一部の有識者以外、計算結果良否を見て直感的に判断することも難しい状態であった。

3.3.2. 第 1 バージョンにおける手戻り発生

対象製品は、2012 年度から 2013 年度にかけて開発され、2012 年度に第 1 バージョン、2013 年度に第 2 バージョンをリリースした。

前述の通り、シミュレーションの期待結果を事前に決めるのが難しい、という問題があったため、第 1 バージョンの開発時には、シミュレーション期待結果を明確に決め妥当性を確認する、という手順が抜けてしまった。

ソフトウェア適格性テストが完了し、シミュレーション結果を社内の有識者がチェックした際、「実環境における振る舞いとあまりにもかけ離れている」という指摘が挙がった。そのため、ソフトウェア要求分析フェーズにまでさかのぼり再検討、再設計を実施し、妥当なシミュレーション結果を得ることのできる方式に変更した。指摘された不具合が発生した原因は、対象をモデル化する際に簡易化しすぎた、というものである。また、この再検討、再設計による手戻り工数は、約 1 人月であった。

3.3.3. 第 1 バージョンにおける開発プロセスの問題

開発プロセスのどのフェーズにおいても、モデルの不適切な簡易化という不具合を発見できなかった原因について分析する。

ソフトウェア開発の各フェーズにおいて実施した設計と検証内容を表 15-A-7-1 に示す。

表 15-A-7-1　第 1 バージョンでフェーズごとに実施した設計・検証内容

No	フェーズ名	設計・検証内容
1	ソフトウェア要求分析	シミュレーション計算式の導出（不具合混入） 設計レビュー
2	ソフトウェア方式設計	画面、プロセス構成等の設計。設計レビュー
3	ソフトウェア詳細設計	計算関数の入出力詳細。設計レビュー
4	コーディング、単体テスト	計算関数単位の計算の妥当性確認
5	ソフトウェア結合テスト	パラメータ入力、エラーチェックの確認 計算制御（開始、完了、進捗・結果表示）の確認
6	ソフトウェア適格性確認テスト	ユーザーシナリオに沿い、パラメータ入力～計算結果表示までが実行できることの確認

当該不具合の混入フェーズはソフトウェア要求分析であり、不具合原因であるモデル化の計算式はソフトウェア要求仕様書に明記されていた。従って、検出すべきフェーズは、ソフトウェア要求分析のレビュー、またはソフトウェア適格性確認テストということになる。

ソフトウェア要求仕様書のレビューについては、社内の有識者を含めて実施していた。しかし、レビューでは不具合が指摘されず、製造工程へ流出してしまった。レビューで指摘されなかった原因は次のように推測される。

(1)　計算式がドキュメントの各章に分散して記述され、システム全体として計算が実行されたときにどのような結果になるか推測しにくい。

(2)　書かれた計算式が学術的に妥当か？という観点でチェックされており、モデル化の意図まではチェックできていない。

ソフトウェア適格性確認テストにおいては、「ユーザーが計算を実行できる」という動作の確認を実施しているものの、計算結果である順位付けが妥当かどうかに確認がなされていない。

これらの反省として、システム全体として動いたときの期待結果を決め、それに合致するかをレビューやテストによって検証する必要がある、ということになる。

しかし、前述の通り、このシミュレーションには「事前に期待結果を作成することが困難」という特徴があり、「決めたいけど決められない」というジレンマに陥ることになる。しかし、ソフトウェアの妥当性を確認するには、何か決めなければならない。

3.3.4. 第 2 バージョン開発における課題

2013 年度に対象製品の第 2 バージョンが開発されることになった。第 2 バージョンの開発内容は、基本構成を変えずシミュレーションの計算パターンを追加する、というものである。第 2 バージョンの開発において、第 1 バージョンのような手戻りを防止するには、「シミュレーションの期待結果を決め、検証する」ということを「何らか」の方法で実現する必要がある。

前述の通り、対象製品が扱うドメインについて、社内の有識者が少数ではあるが存在して

いる。その知見を使って期待結果を抽出することができないか？という案が浮上した。さらに、その内容について関係者間で合意する、ということが必要である、という考えにも至った。そのため第 2 バージョン開発に向けて、「期待結果を明確化し、関係者間で合意形成する」方法を確立することが課題となった。

3.4. 適用技術・手法

3.4.1. D-Case の導入

D-Case とは、「ステークホルダ間で要求の変化に対する合意議論を充分に行う事ができ、合意結果/結論に至った理由/議論の経緯を記録する」ための方法であり、合意記述の記法に基づいて 作成された合意文書もまた D-Case と呼ぶ。([2]より抜粋)

D-Case の基本構成を図 15-A-7-2 に示す。

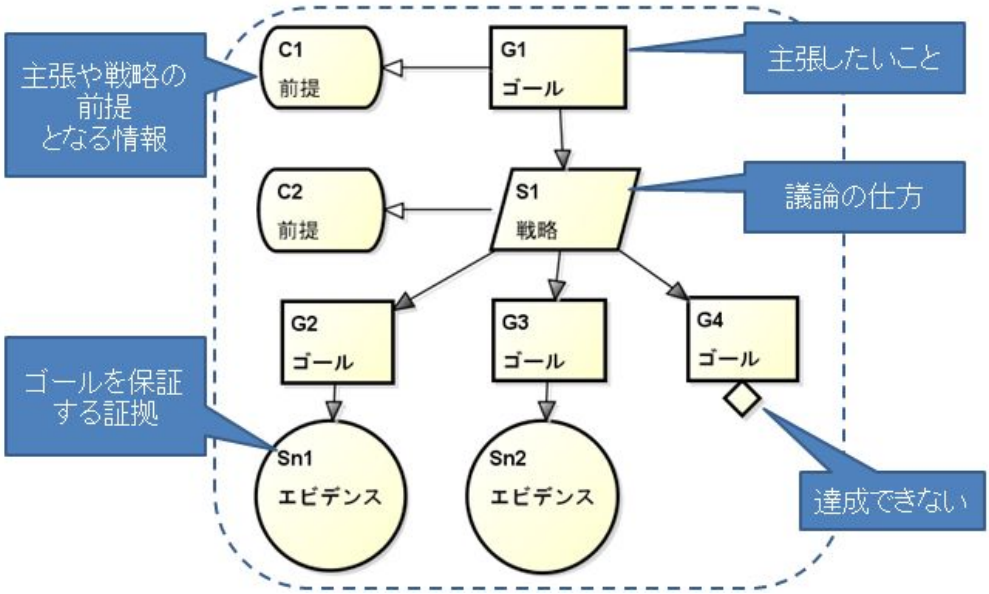

図 15-A-7-2　D-Case の基本構成

筆者は D-Case について、ソフトウェア品質シンポジウム（SQiP シンポジウム）2013 で知ることになる。SQiP シンポジウムの併設チュートリアルの 1 つとして、富士ゼロックス株式会社 秋山 浩一氏による「ＨＡＹＳＴ法によるテスト分析・テスト設計入門」が開講され、説明責任の全うや合意形成の手段として D-Case が紹介された。

D-Case を用いて、対象製品が何を満たせばディペンダブルであるかを示すことができれば、期待結果の明確化につながる。さらに早期に合意形成まで実現できれば、開発後期における手戻りも防止することができる。このような考えに基づき、対象製品の第 2 バージョン開発において、「期待結果を明確化し、関係者間で合意形成する」手段として D-Case を導入することにした。

3.4.2. D-Case 導入における課題

製品開発に D-Case を導入するにあたり、次の課題があった。

① D-Case 作成スキル獲得
メンバー全員が D-Case はもちろん議論の初心者である。また、社外にも実開発への D-Case 適用事例がほとんど存在しない。どのように、組織へ D-Case を導入し、作成スキルを獲得するか？

② 合意に導く議論構成
ステークホルダから客観的な合意を得るには、D-Case の作成においてどのような議論の構成が適切であるか？

③ 経験則の無い場合の議論
有識者の経験則を活かしたいが、経験則が無いものをどのように扱うか？

④ 合意記録の管理
合意の記録や証拠をどのように残すか？また、多忙なステークホルダに合意、承認をなるべく楽に実施してもらうにはどうすればよいか？

各課題に対する解決方法について、次章で説明する。

4. 取組みの実施、及び実施上の問題、対策・工夫

4.1. 組織への D-Case 導入

課題「①D-Case 作成スキル獲得」の達成のために、社外からの情報収集、社内勉強会を実施した。

4.1.1. 情報収集

2013 年当時、D-Case を企業の製品開発に適用した例はほとんどなく、インターネットの情報を参考にしたり、シンポジウムへ参加して講演を聴講するなど、手探りで導入を進めた。下記を主に参考にした。

（1） WEB サイト：dcase.jp（http://www.dcase.jp/introduction.html [2]）
（2） シンポジウム：ET2013 スペシャルセッション「オープンシステムディペンダビリティが世界を変える～DEOS（変化しつづけるシステムのためのディペンダビリティ向上技術）、いよいよ実用化へ！～」
（3） DEOS プロジェクト研究成果集：
（http://www.jst.go.jp/crest/crest-os/osddeos/data/DEOS-FY2013-SS-01J.pdf ）

情報収集によって習得した知識は、主に下記の通りである。

（1） ディペンダビリティ、DEOS プロセスの考え方

[2] 2015 年現在は、http://www.dcase.jp/p/jintro1.html　に変更

（2） D-Case の基本的な記述方法

（3） D-Case の適用事例

このうち、対象製品に D-Case を適用するために必要な、（2）D-Case の基本的な記述方法について社内勉強会を実施し、メンバーのスキルアップを図った。

4.1.2. 社内勉強会の実施

製品に適用する前に D-Case に慣れるという目的で、約 1 か月間チーム内の D-Case 勉強会を開催した。勉強会では簡易ツールを題材にした D-Case 作成演習を実施した。

簡易ツールを題材にした D-Case 作成演習

Redmine[3]に作業時間を自動登録する、という簡易ツールを題材に、D-Case 作成演習を行った。このツールは REST[4]を使用したクライアントツールであり、作業開始時にチケット[5]を選択してスタートボタンを押下すると時間計測を開始し、ストップボタン押下時にチケットに自動的に作業時間を計上する、という仕様である。

まず、簡易ツールに対するトップゴール決めを行った。ここではトップゴールを抽出するために、マインドマップ図で簡易ツールに要求することを列挙した。図 15-A-7-3 に作成したマインドマップ図を示す。列挙した内容を分類すると、「楽」「正確」というキーワードが見えてきた。そこで、「ツールは簡単に正確な時間を計上できる」をトップゴールとした。後にわかったことであるが、本来の手順としては、「楽」「正確」というキーワードは、トップゴールに対する「前提」として抽出されるべきである。本演習時はそこまで理解が及んでいなかった。

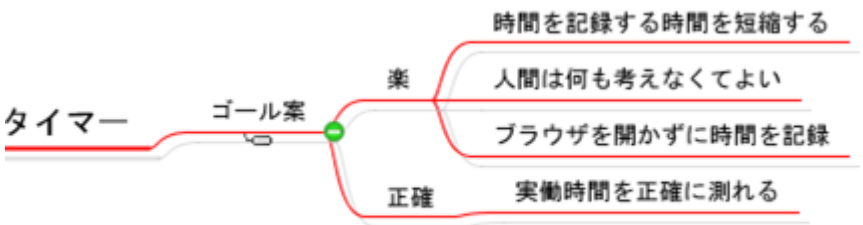

図 15-A-7-3 トップゴール決めのマインドマップ図

次に、サブゴールへの分解を行った。初めはツールの操作手順で分解し、それぞれの手順

---

[3] Redmine：オープンソースのプロジェクト管理ソフトウェア

[4] REpresentational State Transfer ：分散システムにおいて複数のソフトウェアを連携させるのに適した設計原則の集合。出典 http://e-words.jp/w/REST.html

[5] チケット：Redmine 上でタスクを管理する作業指示書に対応するもの
http://redmine.jp/glossary/i/issue/

が「楽」「正確」を満たすことを論証しようとしたが、手順がドキュメントで細かく決められていたため、その手順をなぞるだけの分解になってしまい（図 15-A-7-4）、本質的な特性について論証できなかった。

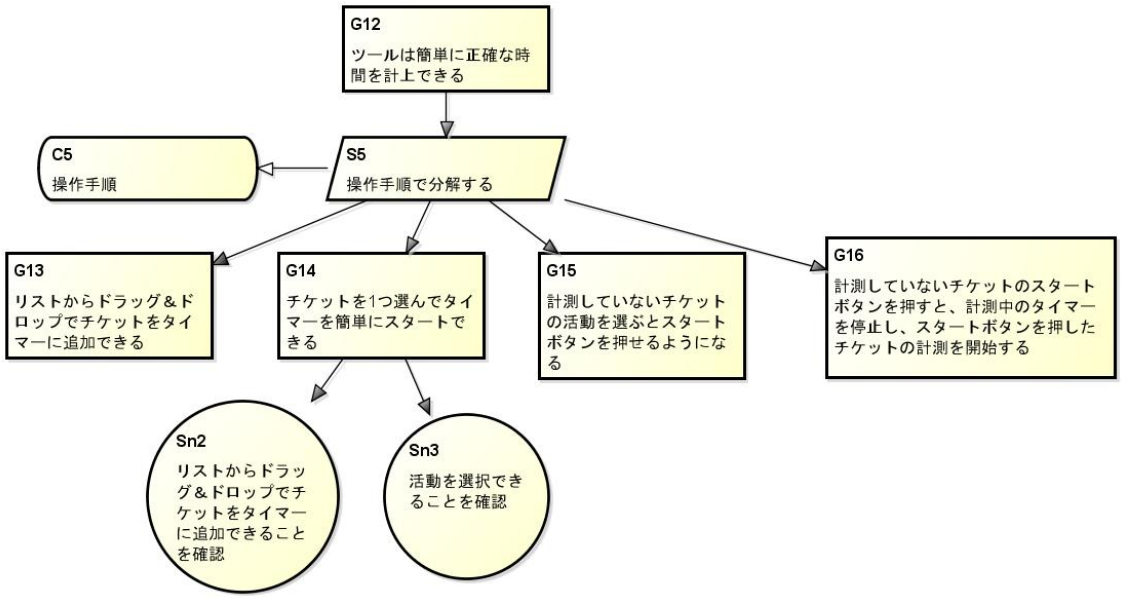

図 15-A-7-4　手順による分解

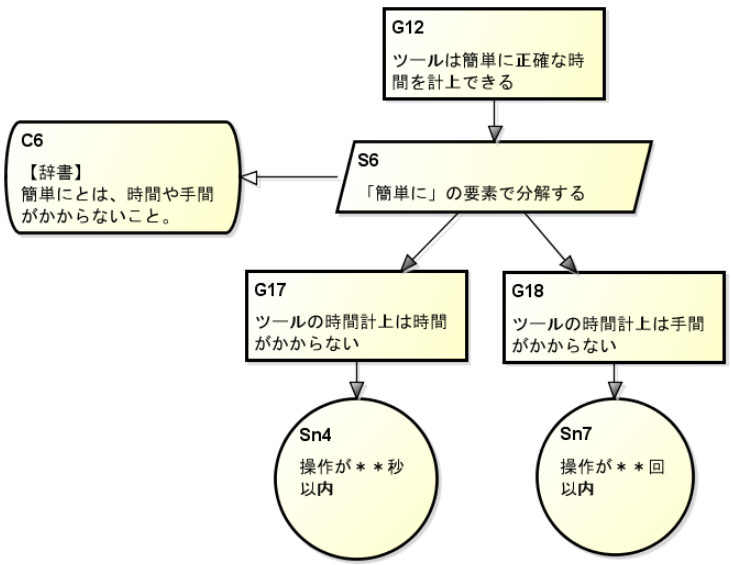

図 15-A-7-5　特性による分解

後日、社外の有識者からアドバイスをいただく機会があり、説明責任の遂行のためには、図 15-A-7-4 のような手順による分解も、図 15-A-7-5 のような特性による分解もどちらも必要であり、組み合わせて分解が網羅的になるようにするのがよい、というお話を伺うことができた。

本演習の成果として、チームメンバーは、D-Case の基本的な記述方法を習得した。また、小さいツールでもまじめに議論すると検討すべき事項が多数出てくる、という気づきもあった。

4.2. 対象製品への D-Case の適用

4.2.1. D-Case 適用理由

情報収集、勉強会を通じて、D-Case はシステムに要求される特性を見える化し、それを満たすことを証明できる強力なツールであることがわかった。このため、対象製品の開発に適用することを正式に決定した。

4.2.2. ステークホルダ

本事例におけるステークホルダと、それぞれの役割は次の通りであった。

（1） 有識者：経験則の提示、D-Case、エビデンスのレビュー

（2） 上流設計者：D-Case、エビデンスのレビュー

（3） 製造担当者（筆者含む）：D-Case の作成、エビデンス収集、D-Case の管理

ここで述べたステークホルダに対応する役割は、本事例固有のものであることに注意されたい。

4.2.3. 議論の方針

（1） トップゴールの「前提」

まず、トップゴールを「シミュレーション計算が妥当である」とした。課題「②合意に導く議論構成」に挙げたステークホルダから客観的な合意を得る議論を実現するために、対象製品に対する根幹的なシステム要求に立ち返って検討することとした。対象製品のシステム要求の内容は次の 2 点である。

1) パラメータに相対的な順序付けができること

2) ユーザーがパラメータ選択の参考にできること

このシステム要求を「前提」とし、システム要求項目ごとにサブゴールに分解した。システム要求に基づいて分解した D-Case を図 15-A-7-6 に示す。

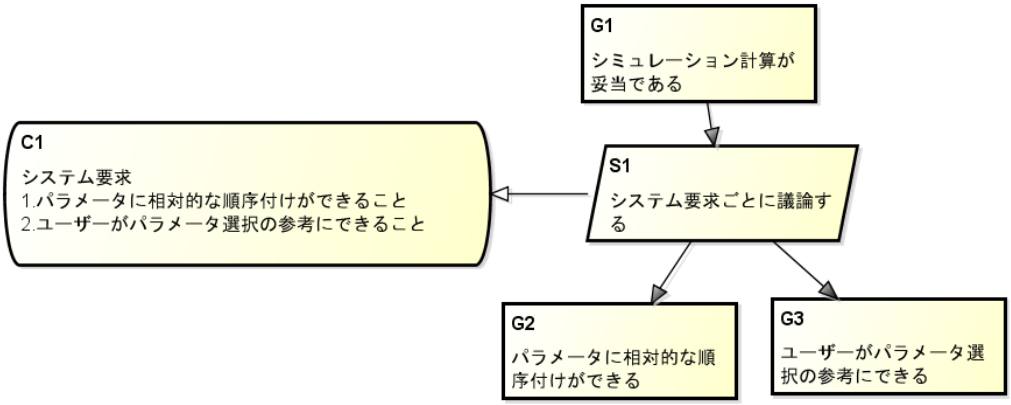

図 15-A-7-6　システム要求によるトップレベルの分解

本編では、システム要求のうち「パラメータに相対的な順序付けができること」についてどのように論証したかを紹介する。

(2)　パラメータに相対的な順序付けができることの議論

前述の通り、対象製品のシミュレーションについて、全ての条件に対する期待結果（順位）を事前に求めることはできないが、一方で有識者が知っているいくつかの経験則が存在することもわかっている。従って、パラメータに相対的な順序付けができることについての論証では、有識者の経験則を「前提」として活用することとした。有識者に経験則に基づくパラメータ条件と想定順位の組み合わせを列挙してもらい、その組み合わせごとに実際のシミュレーション結果と突き合わせた結果をエビデンスとする。有識者の経験則を「前提」として分解した D-Case を図 15-A-7-7 に示す。

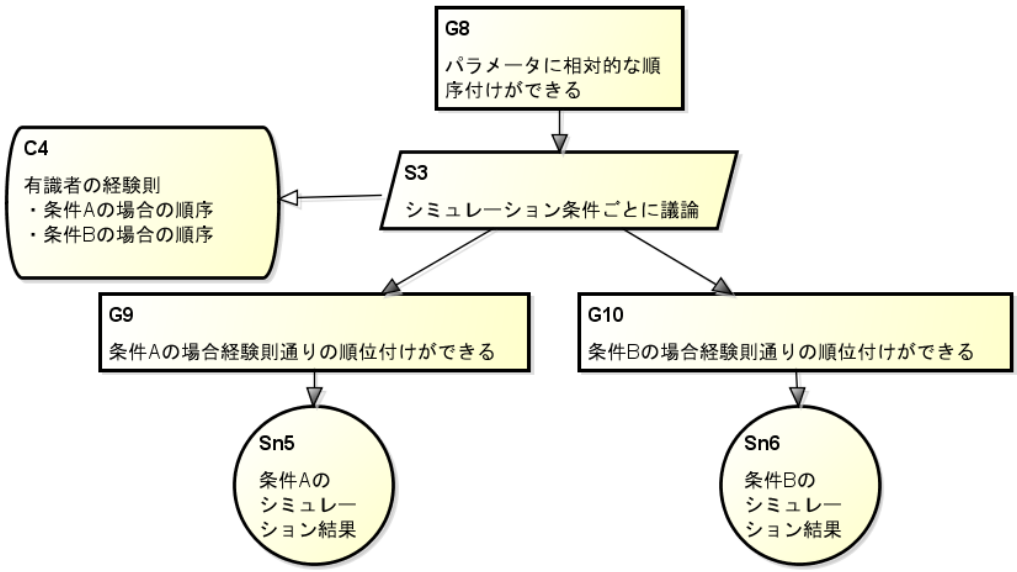

図 15-A-7-7　有識者の経験則による分解

さらに、課題「③経験則の無い場合の議論」として挙げた、有識者の経験則が存在しないシミュレーション条件の扱いについて検討し、シミュレーション条件を経験則のある場合と無い場合に分け、経験則のあるものは経験則を「前提」とし、経験則の無いものは事前に期待結果を導出する別の方法を検討し、見つからなければ「未達成」とした。ここでいう別の方法とは、物理法則や確率論に基づく Excel を使った計算などを指す。経験則の有無による分解を追加した D-Case を図 15-A-7-8 に示す。図 15-A-7-8 の右下のノード（G11）は有識者の経験則も、期待結果を導出する別手段も無いため、論証が未達成となっている。このように未達成のものについても明示的に示すことで、網羅的に議論がなされたこと記録とすることができる。

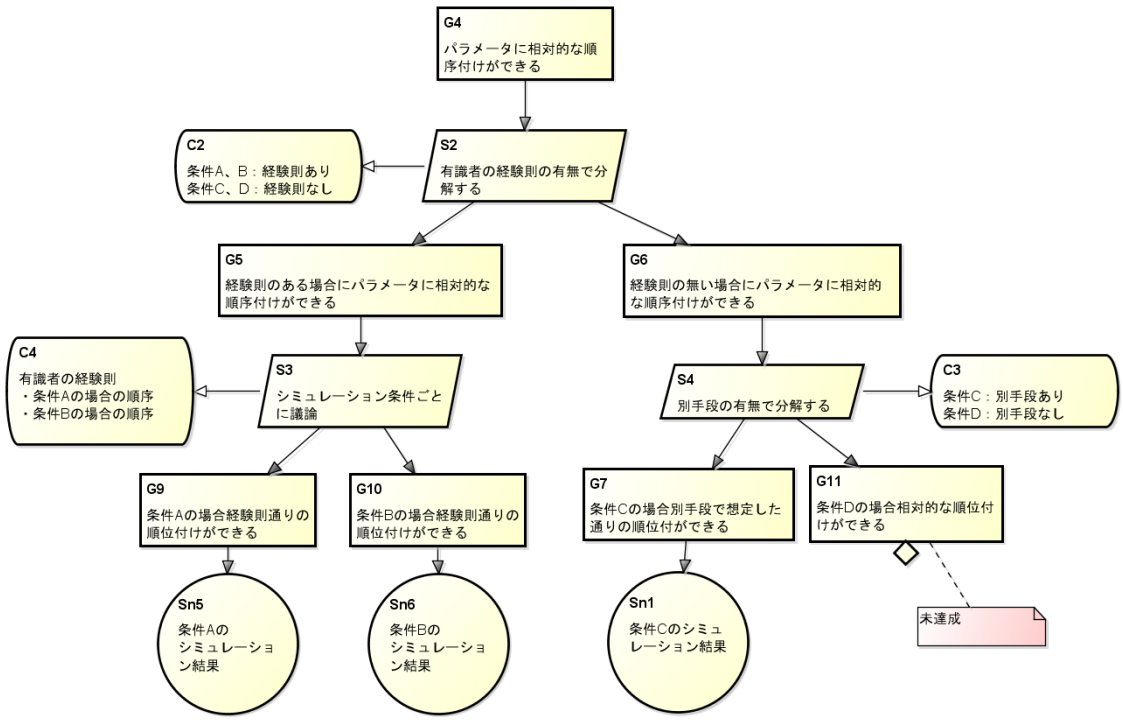

図 15-A-7-8　有識者の経験則有無による分類

4.2.4. D-Case を使用した合意形成

4.2.3 で紹介した D-Case について、開発の早い時期からステークホルダ間でレビューとフィードバックを繰り返した。また、実際にシミュレーション計算を行うモジュールを他のモジュールに先行して製造し、エビデンスを取集した。収集したエビデンスをステークホルダに確認してもらい、ゴールを論証できていることに合意を得た。

この活動により、ソフトウェア結合テストに入る前にシミュレーション機能に対する合意が達成され、第 1 バージョンのような手戻りを防止することができた。

合意形成のために D-Case について定期的なレビューを繰り返すことは、かなりの労力を必要としたが、幸い、本事例ではステークホルダが忙しい中時間を割いてくれたため、スムーズに進めることができた。D-Case による合意形成を成功するかどうかは、ステークホルダの理解と協力が得られるかにかかっている。

4.2.5. 合意形成データベースとしての Redmine 利用

課題「④合意記録の管理」合意への解決策として、Redmine を使用した。ゴール（サブゴール）、エビデンス一つ一つを Redmine のチケットとして登録した。合意を行う際には、Redmine にログインし、チケットに合意の入力を行った。Redmine のチケットには、更新履歴が残るため、いつ誰が合意したかの証拠になる。図 15-A-7-9 に Redmine 上に表示された合意の履歴例を示す。

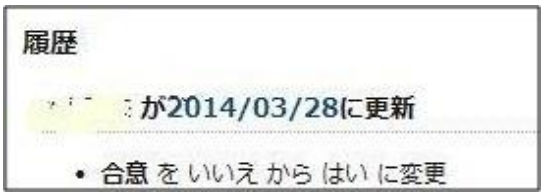

図 15-A-7-9　Redmine に表示された合意の履歴

本事例では D-Case のノード数が非常に多く、手作業による Redmine チケット化は困難だったため、Redmine の REST を使用して D-Case から Redmine チケットへ自動登録するツールを作成した。

このように合意形成のためのデータベースを構築し、ゴールやエビデンスの管理コストを削減することが、実開発への D-Case の適用には効果的である。

5. 達成度の評価、取組みの結果

5.1. 効果の定量的評価

コストに対する定量評価を図 15-A-7- 10 に示す。第 2 バージョンでは手戻りを防止することができたため、第 1 バージョン相当の手戻り（1 人月）を削減コストとみなすことができる。勉強会などの学習コスト、製品適用時の運用コストを差し引くと、約 30h の効果があった。本事例は D－Case を組織へはじめて導入し、何もわからない状態からの出発だったが、今後の適用においては学習コストが減り、効果が拡大することが期待される。その一方で、感覚的には運用コストが大きかったという印象が強く残った。

No	コスト種別	工数(h)	内訳分類	内容	工数(h)
1	導入コスト	123	学習コスト	D-Case勉強会	32
			運用コスト	D-Case作成、レビュー、エビデンス収集	91
2	削減コスト	150	手戻り削減	バージョン1での手戻り工数で換算	150
3	効果コスト (No2-No1)	27			

図 15-A-7- 10　コスト評価

5.2. 効果の定性的評価

対象製品開発終了後、ステークホルダに集まってもらい、KPT 法による振り返りを行った。KPT 法とは、アジャイルプロセスで一般的に用いられる PDCA の手法であり、チームメンバーが一堂に会し、Keep（良かった点）、Problem（問題だった点）、Try（挑戦したいこと）について順番に発言する。振り返りにおいて収集した意見を表 15-A-7-2 に示す。

表 15-A-7-2　振り返りの結果（KPT 法）

Keep （良かった点）	● ステークホルダ間で合意に至ったことは大きな成果
	● できないこと（未達成）についても合意を取ることができた
	● D-Case 作成段階での気づきが、設計に反映されることがあった
	● 体系的に検討書を残すことができた
	● 設計の進捗がわかりやすかった
Problem （問題だった点）	● 世の中に資料がなく、作成した図の議論が妥当か判断できない
	● 図が大きくなりすぎ、作成、合意やエビデンス整備に時間を要した
	● 誤った書き方をしていた（「前提」無しで分解）
	● 第三者が図を理解できるか不明
Try （挑戦したいこと）	● 今回は社内だったが、客先との調整に D-Case を使用してみたい
	● コスト、リスク、重要度から、優先順位を付けて作成したい
	● D-Case データの管理ツールを整備したい （ゴールとエビデンスのリンク、DB との連携など）

振り返りの結果から、メンバーは D-Case が効果的であることを十分実感したことが読み取れる。また、本事例の開発メンバーは若手が多く、若手の論理展開力が向上したという副次的な効果もあった。

同時に、D-Case を運用する上での課題（スキル、コスト）についても見えてきた。

6. 今後の取組みと考察

6.1. D-Case 適用における課題

本事例の取組みにより、D-Case は要求を明確化し、合意を形成するために有効なツールであることがわかった。しかし、適用にあたっては次のような問題がある。

（1） D-Case の作成、管理のコストがかかる。実開発ではサブゴール数が大きくなり、一つ一つについてエビデンスを収集し、ゴールと紐づける作業は非常に大変である。

（2） D-Case 適用方法が標準化されておらず、使いどころがわからない。また、どのような分解が適切かわからない。

これらの問題のため、適用範囲の拡大がスムーズに進まないのが現状である。対策として、「スコープの限定」を行うということが挙げられる。

6.2. スコープの限定

D-Case 適用の効果のある範囲を見極め、対象範囲を絞って適用する、という方法が有効である。逆にスコープ限定を行わず、システム全体に D-Case を作成するとコスト高になり、リスクの少ない箇所のトレーサビリティ確認に使用するのは、費用対効果が小さい。

効果のある対象範囲として、合意形成リスクのある部分、システムの重要な特性を選択するという方法が考えられる。

6.2.1. 合意形成リスクのある部分

要求仕様書の性能要求が曖昧に記述されている場合がある。例えば、「できるかぎり安全のこと」「いかなる時も速度性能を満たすこと」といった記述などには合意形成のリスクがある。このような曖昧な性能要求は、妥当性確認ができないためテストされず、後々ステークホルダ間でもめる原因や、運用時の不具合となることが多い。

また別なケースとして、ベンダーがツールなどを提案してくるような場合には、「このツールを使えば効率化ができる」「自動化できる」等のうたい文句を掲げることが多い。その具体的効果や責任範囲を明らかにしないまま導入してしまうと、期待した効果が出ずにベンダーとユーザーの間でトラブルになることもある。

一般的に、性能要求値、効果値、責任範囲は合意形成が不足するリスクが高い。従って、これらの具体値を「前提」として定義し、D-Case に分解してエビデンスを決め、ステークホルダ間で合意することで、運用段階で問題や手戻りが発生するリスクを軽減することができると考えられる。その他にも開発の過程で、合意に不安が発生した項目には、D-Case を書いてみることを勧める。

6.2.2. システムの重要な特性

何かシステムを開発する場合には達成しようとする課題があり、その課題がシステムに求められる重要な特性とみなすことができる。たとえば、プログラミング作業を効率化するた

めに導入するコード自動生成ツールであれば､｢プログラミング作業を効率化させることができる」が重要な課題である。反対に、外観が美しい、面白いといった魅力性は比較的重要ではない。従って､｢プログラミング作業を効率化させることができる」を重要な特性として選択し、具体的指標値（従来より○%以上削減など）に落とし込み、D-Case の「前提」とする。あとはプログラミング作業の手順にそって分解し、各手順が指標値以上効率化されていることを論証すればよい。ここで、外観の美しさや面白さまで論証し始めると、ノード数が増大し、メンテナンスのコストが高くなってしまう。

6.3. その他 D-Case を適用できるケース

D-Case による論証の対象は開発するシステムだけでなく、活動そのもののディペンダビリティも含まれる。

例えば､｢新製品のデモが効果的である」をトップゴールに D-Case を作成することもできる。この場合、効果的なデモとはどんなものかの基準を抽出し､｢前提」とする。｢前提」の例としては、費用対効果が明確である、技術の高さをアピールできている、デモの場にキーマンがいる、などが考えられる。さらにキーマンのリスト、アピールしたい技術ポイントなどを「前提」にサブゴールに分解し、でき上がった D-Case に基づいてリハーサルを実施する。

このように D-Case を用いて組織や自身の活動を評価したり、より良いものにすることができる。まずは身近な例から D-Case を作成してみることも、議論に慣れる 1 つの方法と考えられる。

参考文献

[1] dcase.jp：D-Case とは、http://www.dcase.jp/p/jintro1.html

[2] D-Case | DEOS の技術 | DEOS ：http://deos.or.jp/technology/dcase-j.html

独立行政法人情報処理推進機構 技術本部 ソフトウェア高信頼化センター（IPA/SEC）