

2x2requirement チャート適用による 開発並びに運用コスト低減の評価

嶋津 恵子（慶應義塾大学） 古川 康一（嘉悦大学）

概要 我々は、IEEE が提供する標準の ConOps(CONcept of OperationS)の項目を効率的に特定するためのフレームワーク・ツールとして、2x2requirement チャートを考案した。その最大の狙いは、システム・エンジニアリングにおける要求開発作業を、初期の作業コストを増加させることなく充実させることであり、その結果、開発での予想外の作業発生を抑えることと、運用コストを低減することに貢献できると考えられる。我々は、すでに同チャートの運用コスト低減に対する貢献について報告した。一方、システムの運用コストは、開発されたシステムの完成度に依存することが多い。そこで、運用コストを計測した実験において、比較した2つのシステムの開発作業を再検討し、開発作業の影響を排除した上で、再度運用コストを精査した。この結果、2x2requirement チャートを利用すると、開発作業の品質に依らず運用コストの低下が実現できたことが判明した。さらに、今回の検証では、開発コストそのものの低減にも貢献していたことを確認した。

1. はじめに

1995年に発表された米国スタンディッシュ・グループが行った調査以降、プロジェクトの評価はQCD (Quality, Cost, Deliver : 品質, コスト, 納期) で論じられている。特に、“コスト”の超過は、その大きさによってはプロジェクトの中断を判断する最大の要素になっている。この原因の多くは予定外の欠陥修復作業であり、さらにその半数以上が初期段階の要求特定作業に起因している[1]。不十分な要求特定が開発段階で手戻り作業を発生させ、プロジェクト期間の延長になり納期遅れを生む。そして延長期間分のリソース投入が富み、コスト超過に繋がる(2.3節)。また、手戻り作業を通してアドホック的に付け加えられた機能は完成度が低くなり、要求満足で計測する“品質”を低下させる原因になる。さらにこれらは、設計そのものが不十分になることも多く、運用段階でも想定外の障害を発生させ得る。このように不十分な要求仕様は、システム・ライフサイクルに渡ってコスト超過を引き起こす特徴を持っている。これを防ぐには、初期段階に十分なリソースを充当し、要求の完成度を上げることが望まれている(2.4節)。

一方、現在の情報システムを取り巻く環境は、日常的に変化しているため、重要な問題の焦点が変化したり、想定していなかったことが新たな要求として浮上する。たとえ当初議論されなかったとしても、開発の作業中に新たに発生した要求にも対応が迫られることが多くなった。この状況は、所謂上流作業が無駄になるという懸念を生み、多くの情報システム構築の現場で質の高い要求

を作成しないまま、実装作業に移行することが多い。こういった背景を受け、我々は効率的に本質的な要求を発見する作業の助けになる2x2 requirement チャートを開発した。このチャートの狙いは、質の高い要求開発を限られたリソースで実現し、システムの品質を向上させ、開発と運用のコストを低減させることである。特に情報システム構築の現場では、運用コストの問題が認識されながらも、開発の品質が運用コストに影響するため、正確な検証は容易ではない。そこで、未だに開発コストを元に、実現手法選択が決定されることが多い[3]。

今回我々は、嶋津ら[2]が比較した2つのシステムの開発段階の作業を再検討し、この影響を排除した上での運用コストの比較を行った。さらに、チャートの利用による開発段階のコストへの影響の確認も行った。

本論文は次の構成を取る。2章に先行研究と関連研究を述べ、3章に2x2 requirement チャートの設計構想と、利用方法を説明する。4章には、同チャートの検証用に採用したエンタプライズ検索システムの概要と検証実験、およびその結果を示す。5章に考察を述べ、6章にまとめを記す。本論文の用語は、システム・エンジニアリングの標準のISO/IEC15288の対訳であるJIS X0170を採用し、対訳の無いものはそのまま英語表記を用いた。例えば要求として記述した文章は、statement とする。さらに、Initial Requirement, Wish Requirement, ユーザー要求などと称される最も初期の段階、もしくはプロジェクト開始前に作成される書類を初期要求と記す。

2. 先行研究・関連研究

2.1 上流作業の充実度に関する調査

Gruhl は、最近の 26 の宇宙事業プロジェクトを対象に、開発段階で発生した対予算超過と概念段階で費やしたコストの関係を調査した[4]。これによると概念段階に費やしたコストは、開発段階予算の 6%程度であることが多く、この場合すべてのプロジェクトで開発段階コストは予算を大幅超過していた。超過が 170%に上る（開発総コストが対予算 270%）ものも存在した。一方、開発予算の 10%を概念段階に費やすと対予算超過は平均 20%に収まり、20%を充当すると予算超過はほとんど発生していない。

2.2 開発コスト超過の原因に関する研究

プロトタイピングは、アジャイル開発方法の代表であり、ソフトウェア業界では広く一般化している。一方 Yap は、この開発方法の採用が多くのプロジェクトで、コスト超過を引き起こしていると報告した[5]。特に、ISO/IEC15288 が示したシステム・ライフサイクル段階決定関門(2.3 節)に基づく V モデル型開発を採用せずに、プロトタイピングを繰り返す方法を採用した場合この傾向が強く現れる。また Yap は、要求特定前に実装作業を開始し、漸増的に要求を特定していくとどうという問題に陥りやすいか、事例を収集し教訓として整理した。さらに Talby は、開発段階で発見された問題の原因が発生した段階と、それらの問題を修正する作業コストの関係を調査し、概念段階が原因である場合では、修復コストが肥大化すると報告した[6]。

2.3 運用コストに関する研究

嶋津らは、2x2requirement チャートの活用により、初期要求を読み下せた結果、発生頻度の低い作業に対しては高い運用段階のコストが発生するが、日常的に利用される機能のコストを低く抑える情報システムの開発に成功した[2]。一方、実施した 2 つの異なるケースによる比較検証では、システム開発作業の質の差の考慮はされていない。具体的には、運用コストが肥大化した事例の開発段階に、より多くのリソースが投入されていれば運用コストが下がった可能性がある。汎用性のある運用コスト低減方法を提案するには、開発作業の品質を十分に考慮する必要がある。

2.4 質の高い要求仕様構築のための標準

ISO/IEC15288 では、システム・ライフサイクルを 6 つの段階(stage)、つまり概念(concept)、開発(development)、製造(production)、利用(utilization)、支援(support)、廃棄(retirement)で管理すると効果的に高品質なシステムを構築できるとし、モデルとして提供している(図 1 上図)。さらに INOCSE(The International Council on Systems Engineering)は、この標準の利用によるベストプラクティスを収集し、実利用性を高めるためにモデルの詳細化と、俯瞰化を行なった(図 1 中および下図)。詳細化は各段階を複数のフェーズで区切り、俯瞰化は、概念段階を concept、開発と製造の段階を implementation、そして利用と支援と廃棄の段階を operation の 3 つの period に集約した。さらに ISO/IEC15288 では、フェーズや段階の重要な移行箇所に決定関門(decision gate)を置く。ここで、関係するステークホルダ全員で移行の可否を決定する。

さらに INCOSE では、質の高い要求の開発と、効率的なシステム構築を目指し、V モデルを薦めている。これは、反復型モデルを包含した滝型モデルに基づく作業方法モデルであり[7]、V モデル左部は対象となるシステムを部品とインタフェースに分解するプロセス(Decomposition Analysis Resolution process: DAR process)として、V モデル右部はそれらを統合するプロセス(Verification Analysis and Resolution process: VAR process)として利用する。V モデル左部で最下層部品(LCI: Lowest Configuration Items)まで到達すると、所謂“仕様”の完成になる。また LCI には、COTS (Commercial-off-the-shelf: 民製品)や、NDI (Nondevelopment Item: できあい品)を採用することを前提としている[8]。V モデル右部は、LCI の収集、それらの統合によるサブシステム化(Subsystem Integration)、さらにそれらの統合によるシステム化(System Integration)に進む。前述した決定関門が設置される場所は、V モデルに照らすと左部上の、いくつかの機能部品とそれらを繋ぐいくつかのインタフェースから構成されるかを定義する system architecture 作成時や、さらにそれぞれの部品内部をより詳細な部品とインタフェースによる構成で定義する system design 作成時である。要求開発の視点に立つと決定関門は、要求の基準線を徐々に清廉化していくための重要な役割を担う。LCI の特定に至る仕様の詳細化作業は、決定関門を通過するごとに基準線が精密になり、以降の作業はこの範囲内で議論と検討が実施される。

従って、要求の開発は、最初の基準線の設け方が重大な意味を持つ。この時に情報を構造的に整理するための

雛形が ConOps(Concept Operations)である[9, 10]. ConOps は、要求特定期間(図1の前半の2つの phase)の作業結果であり、初期要求を元にステークホルダ間の議論を通して合意される. ConOps の主たる狙いは、システムの導入と利用によって実現する将来の状態(to be)を、現状のそれ(as is)との差で明確にすることである. ISO/IEC15288 では、ConOps を元にシステム要求を特定する手順を採用している. ConOps の内容が十分に検討されたかどうかは、システムに対する要求の本質を的確にとらえているかどうかに関わり、不具合の発生やそれらの修復にかかるコストの大きさを決定づけると考えられる. このため、宇宙・航空産業等の大規模複雑かつ安心・安全を求められるシステム開発の現場では、その重要性が認識されている[9].

そして、この記載内容をより正確に作成する方法として、ロジャーは requirement statement の推敲(elaboration)の重要性を示し[11], また玉井は要求分析の型を分類し、目的指向型アプローチの重要性を指摘している[12].

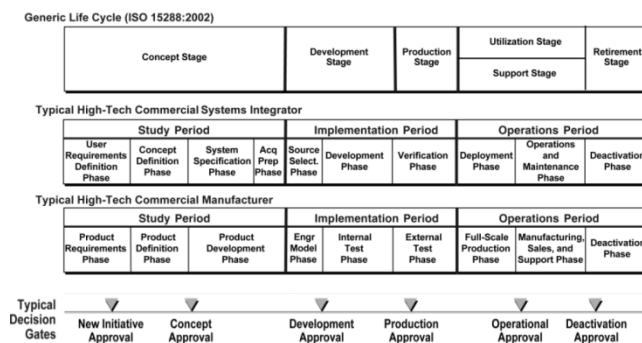


図1 ISO/IEC15288 によるシステム・ライフサイクル・モデルと典型的な決定関門 [13]

2.5 本書の報告の焦点

本章に述べた先行研究から、概念段階の正確性と十分性が開発段階での予想外のコスト発生を防ぐことができ(2.2 節), 方法論としては ConOps を作成しこれを反映した機能群からなるシステム要求の作成に移行するのが最適だと考えられる(2.4 節). 一方、情報システム構築の現場では、できるだけ概念段階を短縮したい要望がある(1 章および 2.1 節). これを受け嶋津らは、2x2Requirement チャートを開発し、これを利用すると開発段階コストの 6% 相当しか概念段階に充当しなくても効率的に ConOps を特定できることを示した[2].

本書は、これに対し開発コストの比較を行いさらに、2.3 節に示した課題、つまり開発作業の品質を十分に考慮し運用コストを正確に比較した結果を報告する.

3. 2x2 requirement チャート

システム構築に際し、前提となる情報システム開発方法によって、検証する手段の効果の大きさや影響の範囲が異なる. そこで、我々は、2x2 requirement チャートがより多くの実践の場で活用されるよう、国際標準のシステム・エンジニアリング手法 ISO/IEC15288 と、INCOSE の ISO/IEC15288 ベストプラクティス集を、情報システム開発の基本方式として利用する.

3.1 2x2 requirement チャート設計思想

システム・エンジニアリングにおける requirement は、視点を様々に変えて、多方面から議論されるが、生成される多くの statement は、ISO/IEC15288 が示すシステム・ライフサイクル・モデルに従うと図2の 2x2 requirement チャート上に整理できると考えた. 具体的には、生成される Statement は、時間的視点、つまり (X1)作成(implementation)時と(X2)運用(Operation)時、さらに目的と実現方法の視点、つまり (Y2)Objective と(Y1)Enabler のいずれかである. システム・ライフサイクル・モデルの期待する利用状態(i)と、その実現後に状態を維持継続させるために必要な支援(iv),そして、開発と製造の段階で、用意され得る人的・施設的・資金的・技術的な資源(iii)である. (iii)を投入することで、必要な機能や性能(ii)が作成される. ISO/IEC15288 が示す ConOps は、(i)と(iv)の情報からなり、システム要求は ConOps を実現するために、(iii)を活用し(ii)を明らかにしたものになる. このチャートをフレームワークとして利用し、作成された statement を評価(assessment)したとき、(i)と(iv)の情報で要求として完全性・無矛盾性を実現することが、効率的かつ最適な要求開発につながると考えた.

例えば、現在、情報システム構築に際し、多くの初期要求記載に目立つと言われる利用技術や実現方法[14]は、これらいずれの視点でも無い. (i)と(iv)の情報を揃えた時には、最適な方法では無いと判断されることも有り得る.

我々の提案は、採用技術優先で作成された statement があつた場合、その存在の適正具合に早期に気づかせるための道具としての意味を持っている.

次節に我々の提案する 2x2 requirement チャートの具体的な利用方法を示す.

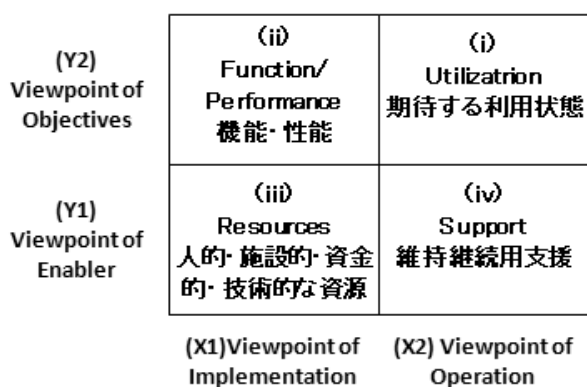


図 2 2x2 requirement チャート

3.2 2x2 requirement チャート利用手順

2x2 requirement チャートの基本的な利用方法は、図 3 のフローチャートと以下に示す手順の通りである。すべての作業は、システムエンジニアとステークホルダのグループワークによって実施する。使用例は、[2]を参照されたい。

- (1) 初期要求中の任意の statement を抽出し STM_01 とする。
- (2) STM_01 を定量表現に変換し、STM_01_en とする。
- (3) STM_01_en が、2x2 requirement チャートの(4つのうちの)どの視点で提示されたものか特定する。
- (4-1-1) STM_01_en が、2x2 requirement チャートの ii もしくは iii に相当する場合、STM_01_en が必要となる理由である statement を特定し、STM_01_en_01 とする。
- (4-1-2) STM_01_en_01 を対象に(2)の操作を行う。
- (4-2-1) STM_01_en が、2x2 requirement チャートの ii もしくは iii に相当しない場合(i もしくは iv に相当する場合)、ConOps 作成用情報として利用する。
- (4-2-2) 2x2 requirement チャート利用を終了し、ConOps 作成作業に移行する。

開発した 2x2 requirement チャートの最大の狙いは、実現方法や技術選択に偏りがちな requirement statements 生成作業を、利用者の利用形態や利用シナリオを想定した statements 生成に誘導することである。実際の要求開発作業では、必要に応じて systems requirement に展開する (ii) と(iii)の statement の検討や清廉化も同時に検討する。

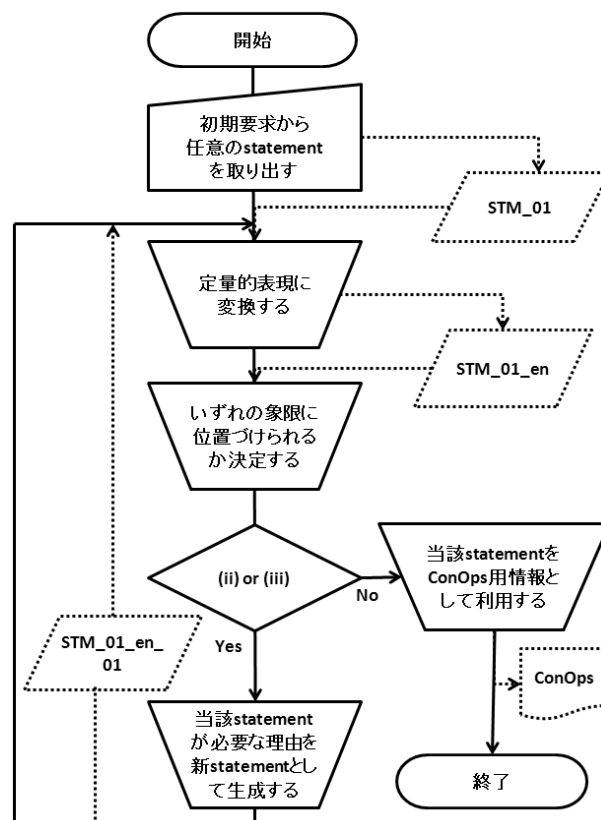


図 3 2x2 requirement チャートの利用手順

4. 有効性検証実験

嶋津らは、同一の初期要求を元に、2つのエンタプライズ検索システムを開発した[2]。これらは、Study Periodの作業方法のみが異なる。具体的には、2x2 requirement チャートを利用しないケース1と利用するケース2であるが、実施体制とともに、詳細は[2]を参照されたい。特に、Vモデル左部の決定関門の設置は、要求の品質を決定する基準線の清廉化であるので両ケースで厳格に等しく用いた。また、ハードウェア製品の場合、開発段階に続く製造段階のコストも重要であるが、エンタプライズ型情報システムの場合、製造物はサーバに設置され、利用されるので、製造段階を開発段階に吸収した。従って本章とそれ以降に述べる“開発”は、開発と製造の段階を指し、つまり implementation period と同一である。両システムとも開発後、実際に一般の利用者に6カ月間の operation period を提供した。この期間を“運用”とする。

4.1 実験結果：要求特定

ケース1、ケース2とも、入手した初期要求を元に、上流で systems requirement を整理した。4.1節と4.2節では、[2]の報告の要点を記す。初期要求の statements 中に、“System-Operators shall edit contents of an index

database.”(IR1)と, “End-users shall change the order of search result.” (IR2)が含まれていた. これらはいずれも 2x2 requirement チャートの(ii)に相当すると特定された. 次に, この statement が必要である理由を検討し, (i)に相当する statement を特定した. この結果, 「利用者は, 検索結果を, 特定の情報発信者, 特定の発信元キャンパス, または発信日で限定して入手できることとする」が, 求められている to-be 状態となった. 次に, この実現方法として IR1 と IR2 が最適かどうかを検討した. その結果新たな方法が提示された. 具体的には, 「検索対象情報に, 情報発信者と発信元キャンパスそして発信日をインデックスデータベースに付加できるインタフェースをシステムが備えることとする」と「システムは, 検索結果を, 特定の情報発信者もしくは特定発信元キャンパスそして特定の発信日に限定して表示する選択肢を提供することとする」である. そこで, これらを(ii)用の新たな statement として特定した. さらに, (iv)の statement として, 「各機能の設定更新作業は, 4 人時間以内で実施することとする」が生成された. 最終的に, これら3つの象限の statement を実現する方法として, (iii)に検索エンジン基盤ハードウェアソフトウェア一体型製品 (COTS_B)が選択された.

これに対しケース1は, IR1 と IR2 が, そのまま system requirement として採用された. そこで検索システムを構成する各部品を, エンドユーザと開発者によるプロトタイプ開発で行うことにした. これにより, すべてのシステム部品が詳細に渡り改変可能な基盤ソフトウェア製品 (COTS_A_s)と, これが実利用に耐えられる動作を保証する基盤ハードウェアサーバ製品(COTS_A_h)が選択された.

4.2 実験結果：システム・アーキテクチャ

ケース1と2で異なったシステム要求が生成されたことで, 次の作業であるシステム・アーキテクチャも異なる結果になった. 図4は, 標準の検索エンジン基盤ソフトウェアのアーキテクチャである.

ケース1は, 標準のアーキテクチャを取る COTS の各部品をプロトタイプ開発の中で改変することにした. これに対しケース2は, 「システムは, 検索結果を, 特定の情報発信者もしくは特定発信元キャンパスそして特定の発信日に限定して表示する選択肢を提供することとする」が最終的に実現する姿だと特定された(4.1節参照). そこで, (ア)収集コンテンツのそれぞれに対し, 発信者と発信とキャンパスとさらに発信日を特定する装置 (LDB reader)と, (イ)それら専用のデータベース群と, (ウ)

それらに登録するインタフェースと, さらに(エ)検索結果を利用者の指定に応じて, それらを使って並び変えるルールベースを独自に作成することにした. 一方「各機能の設定更新作業は, 4 人時間以内で実施すること」も特定されていた. そこで, 想定される設定変更を整理し, これらのすべてが GUI 上のメニュー操作だけ実施できる基盤用ソフトウェア(COTS_B)を選択した. 図5は, 作成したシステム・アーキテクチャである.

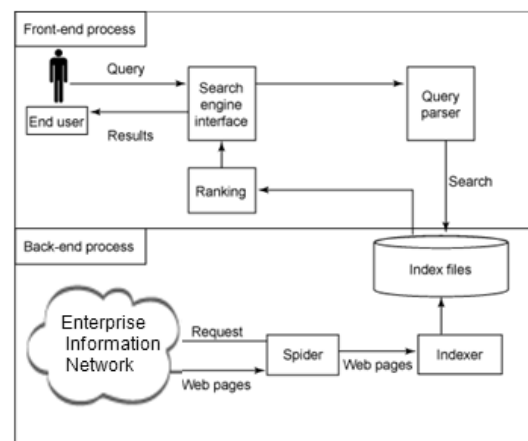


図4 検索エンジン標準アーキテクチャ

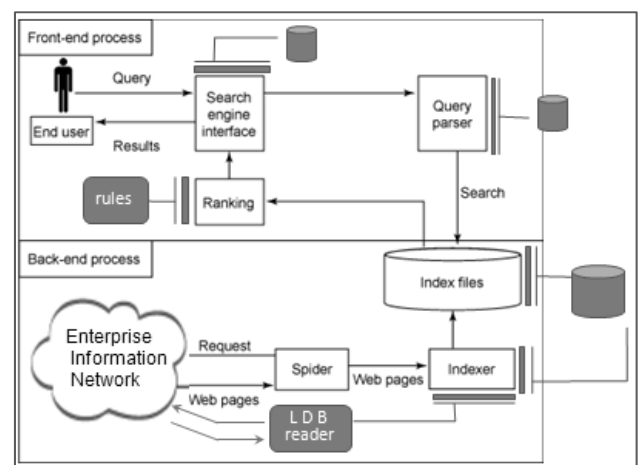


図5 ケース2で採用したシステム・アーキテクチャ

4.3 実験結果：開発期間

ケース1は, 予定通りの期間で Study Period から Implementation に移行したが, subsystem integration で予定外の障害が多発し, LCI の設定に戻り修復に当たった. ところが再度 system integration で障害が発生したため, COTS_A の提供会社から技術者の追加充当を行なった. システムの完成 (開発が終了し, 運用が開始) は, 4 カ月 10 日遅延した. ケース2は, 予定に対し2日早く Study Period が終了したが, 完成は予定より1週間遅延した. 両ケースとも, 一般利用開始後 URL をインターネットに公開し, 6 カ月一般利用者による運用期間を設けた. 図6

に、3つのperiodの当初予定期間と、それらに対する実績を示す。

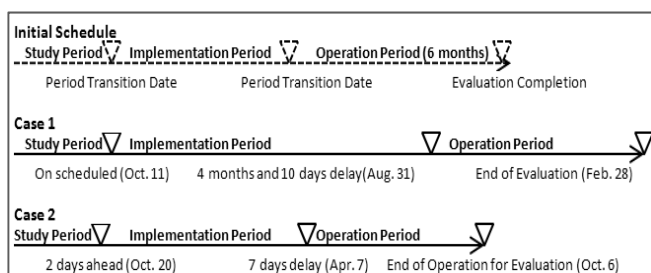


図 6 Period 移行の計画と実績

4.4 実験結果：開発コスト

ケース 1 は、IR1 と IR2 を満足させるために COTS_A_s と COTS_A_h が選定された。この価格は当初予算(2,000 万円)内で収まらず、開発作業直前に 1,200 万円増額し 3,200 万円に見積もりコストを変更した。ケース 2 で採用した COTS はハードウェアとソフトウェアの一体型であり、見込み予算から 680 万円減額した 1,320 万円が開発直前の見積もりコストになった。

また開発作業中の障害発生状況をみると、ケース 1 では、テストで 157 件の問題が発見された。うち 34 件が要求 statement の修正(矛盾解消と明確化)が必要なものであり、残り 123 件が正しく実装されていないものであった。これらの解決のために当初用意したメンバに、COTS 提供会社から新たに技術者を 4 名加えた。これによりさらに 1,000 万円の追加投入が行われた。つまり、開発直前の見積もりコストに対し(開発終了時点で)、45%超過した。

一方、ケース 2 では、開発中に発見された障害は 38 件であり、うち 1 件が要求 statement の修正が必要なものであり、残り 37 件が正しく実装されていないものであった。修復作業は、予定メンバがあたり予定就業時間内に終了した。つまり、開発直前の見積もりコスト通りに終了することができた。図 7 は、V モデル右部の 3 段階ごとのコスト累積額を示したものである。

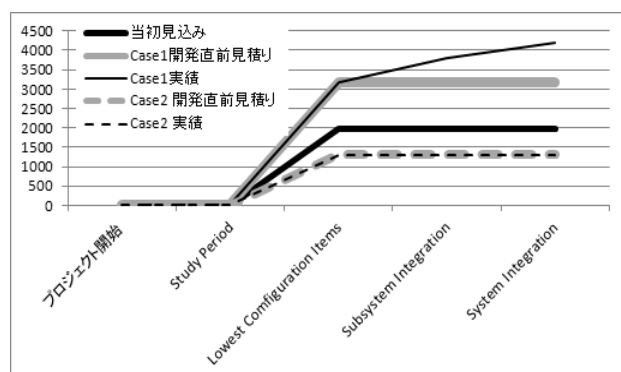


図 7 開発作業で発生したコスト比較

4.5 実験結果：運用コストと項目別分析

嶋津らは、2x2 requirement チャートを利用した時の運用コスト低減に対する効果を、ケース 1 とケース 2 の差を用いて示した[2]。今回、運用コストが大きくなったケース 1 について、開発リソースをさらに追加充当した場合、運用コストに変化が発生した可能性のある作業項目推定を試みた。具体的には、目標としていた“to be”状態を維持継続するための Operation Cost と、予想外の障害に対応するための Maintenance Cost の発生状況に注目した(表 1)。前者は、利用者の数や初心者が多さで変化するが、後者はシステムの完成度に依存することが多い。そこで、各運用作業項目の発生状況を時系列に整理し、発生頻度の変化を概観した(図 8)。

その結果、6つの Maintenance 作業のうち、クローラの再起動とインデックス DB の更新タイミング変更と、Web コンテンツ収集不可対応の 3 つの作業項目は、6 カ月の運用の間に、他の作業項目には見られないような大幅な発生回数の減少を示した。

表 1 発生した運用作業一覧 [2]

	作業内容	ケース1			ケース2		
		発生回数	作業時間小計	1回あたりの作業工数* (人時間)	発生回数	作業時間小計	1回あたりの作業工数* (人時間)
Operations Cost	クローラ設定内容の調整	2	10.0	5.0	6	9.0	1.5
	RDBMS内データの収集	2	8.0	4.0	1	27.0	27.0
	非httpサイト情報の収集	19	38.0	2.0	1	4.0	4.0
	ACL情報の収集	1	4.0	4.0	1	3.0	3.0
	検索語読み替え辞書登録	12	21.0	1.8	9	5.0	0.6
	小計(*は平均)	36	81.0	3.4	18	48.0	7.2
Maintenance Cost	システム全体の再起動	2	36.0	18.0	1	4.0	4.0
	クローラの再起動	102	50.0	0.5	1	0.2	0.2
	インデックスDBの更新タイミング変更	5	1.0	0.2	1	0.5	0.5
	システム起動不良対応	3	62.0	20.7	0	0.0	0.0
	クローラ起動不良対応	23	518.0	22.5	0	0.0	0.0
	Webコンテンツ収集不可対応	31	903.0	29.1	0	0.0	0.0
	小計(*は平均)	166	1570.0	15.2	3	4.7	0.8
合計(*は平均)		202	1651.0	9.3	21	52.7	4.0

ACL: Access Control List

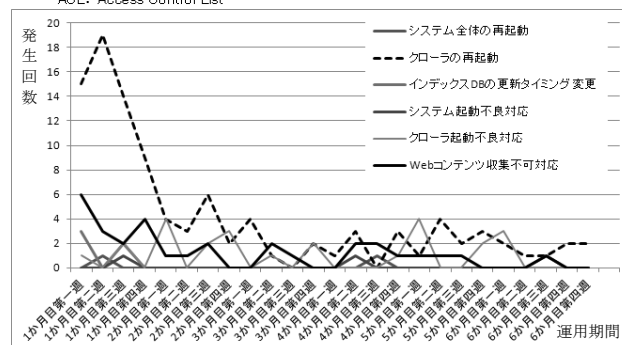


図 8 Maintenance 作業発生回数

5. 考察

今回我々は、2x2requirement チャートを利用した情報システム開発における開発と運用コストの低減を確認した。特に運用に関し、開発状況を考慮した詳細な確認を行った。

同一の初期要求を対象に情報システム構築を行い、開発コストに関し、同チャートを利用する時(ケース2)と利用しない時(ケース1)とを比較し、開発前の時点ですでに約60%削減できる見通しを得た。開発後にケース1が予定コスト内であるのに対し、ケース2は増額した修正見積りをさらに大幅に上回る追加コストを投入している。そして、この修正見積りの段階で、すでにケース1はケース2に対し、約2倍の値になっていた。また、開発期間を見ても、ケース2は7日の遅れで済んでいるのに対し、ケース1は予定していた開発期間の170%を要していた。これらの開発コスト差は、COTS 選択の差によると考えられる。ケース1は、搭載されている機能を人手で改編できる部品を、ケース2は、外部のサービスの出力結果を統合するインタフェースの充実したものを選択したと言える。ケース1で採用したCOTSは、突出した拡張性を持っていたが、扱う要員は製品に特化した高い専門知識が必要であった。この専門知識の特有性(汎用的技術の高さではなく製品特化性)を事前に把握することができず、予定外の費用の充実に繋がった。ケース2で採用したCOTSと付加機能部品の統合作業には、汎用的な技術力はケース1のそれと同様に求められたが、製品に特化した知識はほとんど必要なかった。プロジェクトを成功させるためには、概念段階に開発段階コストの20%の投入が望ましい(2.3節)。一方、我々はこれに対し現状と同様の6%で成功させることができた。これは我々が考案した2x2requirement チャートの導入により、効率的にConOpsを開発できたためだと考える。

次に、運用コストの項目別分析結果を考察した。4.5節で示した3つのMaintenance作業は、運用期間開始直後に作業が集中し、運用期間中に発生回数が激減している。前述の開発コストの状況と合わせて考察すると、ケース1の開発時に埋め込まれたと想定される不具合による運用コスト増を排除することで、日常的に利用される機能の運用コストをケース2とより正當に比較することができると考えた。つまり、開発期間中に目標となるシステム品質に到達できた場合を想定した両ケースの運用コストの比較である。具体的には、4.5節で特定した3つのMaintenance作業を除いた残りの作業を対象に考察した。図9の上側のグラフが取り除く前の運用コストで

あり、下側のグラフが取り除いた結果である。下側の図では、ケース1とケース2の運用コストの差が小さいことがわかる。効果が、下方修正されることになる。一方、それでも両ケースの運用コストを比較すると、6カ月の運用期間で約7倍の開きが認められる。また、これらの項目を除いたケース2の運用コストの累積結果は、除く前のその値と大きな変化が無いこともわかる。このことから、運用コストに影響するような開発作業品質の悪さは、ケース1では発生したものの、ケース2では発生しなかったと考えられる。

以上を総合的に判断すると、我々が開発した2x2requirement チャートの利用は、ISO/IEC15288 標準を前提とする限りにおいて、開発と運用のコスト低減に大きく貢献すると言える。

両ケースの差は、決定関門ごとの基準線の特定によるシステムの絞り込みの方向性の違いに拠るものだと考えられる。2.4節に示したとおりISO/IEC15288 標準では、重要な作業移行時に決定関門を設置し、真に必要な要求や仕様に的確に絞り込むよう明記されている。ところが、概念段階で要求を清廉化する方向が、利用者の使い勝手や利用目的に向かわず特定の機能や技術採用に向いていると、利用者の本来のニーズとは異なる方向に絞り込みが進むことになる。この方針でLCIのレベルまで決定し組み上げられたシステムは、完成度の低い先端技術の採用を促すことによる予想外の開発コストを引き起こしたり、運用開始後に利用者の使いがってにそぐわず支援のコストが増加すると考えられる。これに対し、2x2requirement チャートを利用すると、見失いがちな要求の特定を早期に確実に実施し、情報システムに求められる目的(どう使いたいのか、どうやって維持管理したいのか)を明確にすることができると考えられる。そして、システムの使用が利用者の目的に合致した方向で基準線特定の絞り込み作業を強力に支援することが可能になる。これが、今回の検証で、開発作業の品質を上げ、運用コスト削減を実現できた最大の理由であると言える。

現状では、開発直前に、運用コストをほとんど見通せず、開発コストの比較中心でシステムの実現方法が決定される[15]。このため、運用コスト削減に繋がる方法を採用したシステム開発が行われにくい。一方、我々は、2x2requirement チャートを使うことにより、開発コストだけでなく、運用コストが大幅に削減できることを示した。今回の開発例では、同チャートを導入することで、上流や開発のコストが、仮に1割から3割ぐらゐ上昇することになっても、最も費用の発生する運用の期間のコ

ストが30%程度に収まることから、トータルコストの面で、同チャートの導入は価値が高いと考えられる。

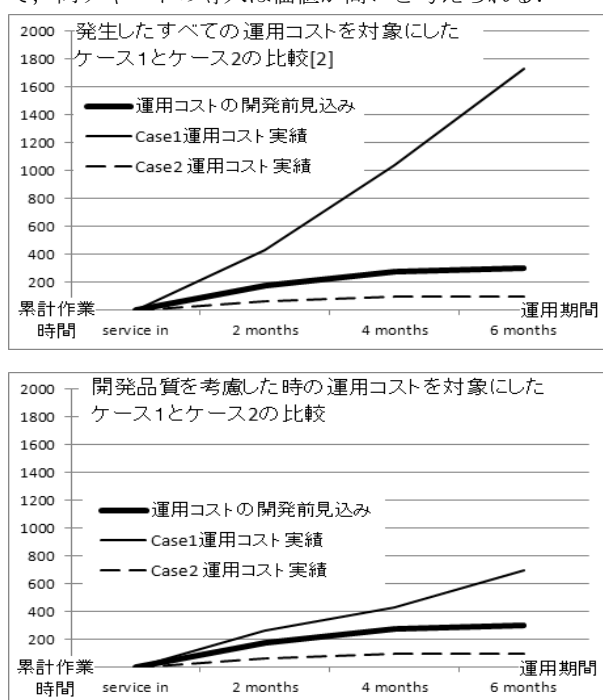


図 9 運用コスト比較

6. まとめ

我々は、IEEE が提供する標準の ConOps(CONcept of OperationS)の項目を効率的に特定するためのツールとして、2x2requirement チャートを考案した。その最大の狙いは、システム・エンジニアリングにおける要求開発作業を、初期段階の作業コストを増加させることなく充実させ、結果的に構築するシステムの完成度を向上させることにある。システムの完成度の低さが引き起こす影響は、運用作業でのコストという重大な問題となる。我々は、すでに同チャートの運用コスト低減に対する貢献について報告した。我々は、運用コストを計測した実験において、比較した2つのシステムの開発作業を再検討し、開発作業の影響を排除した上で、再度運用コストを精査した。この結果、2x2requirement チャートを利用すると、開発結果の完成度に依らず運用コストの低下が実現できることが判明した。さらに、今回の検証では、開発コストそのものの低減にも貢献していたことを確認した。

今後さらに、本書の検証で取り上げた両方のシステムで、実装された機能がどのくらい活用されたかを調査し、2x2requirement チャートの利用が最適な要求特定にどの程度貢献したのか、確認作業を行いたい。

謝辞 本研究は文部科学省グローバル COE プログラム「環境共生・安全システムデザインの先導拠点」に依る

ものであることを記し、謝意を表します。また、共同推敲者の小橋氏に感謝します。

参考文献

- 1) 日経コンピュータ, プロジェクトの実態, nikkei computer 2008.12.1, pp.44-49 (2008)
- 2) 嶋津恵子, 古川康一, 高野研一: 計画期間短縮と運用コスト低減を両立させる CONOPS 作成のための 2x2 requirement チャートの提案, 情報処理学会論文誌, Vol.52, No.2, pp.670-679 (2011)
- 3) 溝口周二: 情報システムのコスト・マネジメント, 横浜国立大学国際社会科学学会, 11(6), pp.1 - 17, 2007-02-20 (2007)
- 4) Forsberg, K., Mooz, H. and Cottnerman, H.: Visualizing Project Management: Charts and Frameworks for Mastering Complex systems, Wiley, 3rd edition, pp.84-89 (2005).
- 5) Yap, M.: Value based extreme programming, Agile Conference, p.8, Minneapolis, MN (2006).
- 6) Talby, D., Hazzan O., Dubinsky Y., Keren A.: Agile software testing in a large-scale project, IEEE software, pp. 30-37 (2006).
- 7) Haskins, C.: Systems Engineering Handbook, A guide for system life cycle processes and activities, INCOSE-TP-2003-002-3.2, January 2010, pp.24-28 (2010).
- 8) 4)に前掲, pp.162-163
- 9) Halligan, R.: OCD&CONOPS in Capability Development, Materials of a Course Over Five Days, Project Performance International, Adelaide, Australia, 9-13 August 2010 (2010).
- 10) IEEE Std. Board: IEEE Guide for Information Technology. System Definition. Concept of Operations (ConOps) Document, IEEE Std, IEEE Computer Society, Reaffirmed 2007 (1998)
- 11) ロジャーS. プレスマン: 実践ソフトウェアエンジニアリング ソフトウェアプロフェッショナルのための基本知識, 西康晴他監訳, 古沢聡子他翻訳, 日科技連, 第6版 (2009).
- 12) 玉井哲雄: ソフトウェア工学の基礎, 岩波書店, pp.37-42 (2004).
- 13) 7)に前掲, pp.26
- 14) Monson-Haefel, R.: 97 Things Every Software Architect Should Know, O'Reilly, (2009)
- 15) 野中誠: ソフトウェア開発コスト予測研究の動向と課題, ソフトウェアエンジニアリング最前線 2008, pp. 33-40, 近代科学社 (2008)

嶋津 恵子（正会員）

E-mail: shimazu@sdm.keio.ac.jp

慶應義塾大学 特任准教授。JAXA(独立行政法人 宇宙航空研究開発機構) 情報化事業外部委員, 2002 年に博士号取得。富士ゼロックス (株) 勤務の後, 慶應義塾大学デジタルメディア・コンテンツ研究機構 助教授を経て現職。システム・エンジニアリングの各種産業への応用, 特に情報システム工学, アプリケーション工学, 機械学習アルゴリズムのデータマイニングへの応用が専門。情報処理学会, 情報システム学会, IEEE, INCOSE 各会員

古川 康一（正会員）

E-mail: kfurukawa@kaetsu.ac.jp

昭和 40 年東京大学工学部計数工学科卒, 昭和 42 年同大学院工学系研究科修士修了。同年通産省工業技術院電気試験所 (現産業技術総合研究所) 入所。昭和 57 年 (財) 新世代コンピュータ技術開発機構 (ICOT) へ出向。第 2 研究室長, 第 1 研究室長を経て, 昭和 61 年同研究担当次長。平成 4 年慶應義塾大学環境情報学部教授に就任。平成 6 年同大学院政策・メディア研究科教授。平成 20 年 3 月定年退職。平成 22 年 4 月嘉悦大学大学院ビジネス創造研究科に就任, 現在に至る。東京大学工学博士 (情報工学)。慶應義塾大学名誉教授。

投稿受付 : 2011 年 8 月 5 日

採録決定 : 2012 年 1 月 18 日

編集担当 : 小橋喜嗣 (NTT アドバンステクノロジー)