情報処理学会論文誌 Vol. 52 No. 2 670–679 (Feb. 2011)

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2 × 2 requirementチャートの提案

嶋津恵子[†1] 古川康一[†2] 高野研一[†3]

我々は，IEEEが提供する標準のConOps（CONcept of OPerationS）の項目を適切に特定するための2×2 requirementチャートを考案し，その利用方法を開発した．システムの実現方法や採用技術に偏ることなくConOpsの構築にとって必要な情報をrequirementとして生成できることが，我々の提案の主眼である．この方法でConOpsを適切に構築することで，システムエンジニアリングにおけるStudy Periodを充実させ，結果的に運用（Operation Period）コストを低減させることが可能になる．今回我々は，実際にエンタプライズシステムを2つの方法で構築し，運用コストを比較した．これら2つは，顧客から提示された初期要求（initial requirement）をもとに，上流工程でConOpsを作成せず，システムアーキテクチャ作成とシステムデザインを経て開発工程（Implementation Period）に移行する方法（ケース1）と，同工程で我々が提案する2×2 requirementチャートを利用してConOpsを作成し，その後に引き続く作業を実施する方法（ケース2）である．この検証実験により，我々の開発したチャートの有用性を確認した．

Proposal of a 2 × 2 requirement Chart for Supporting ConOps Development to Reduce both Planning Time and Operation Cost

Keiko Shimazu,[†1] Koichi Furukawa[†2] and Kenichi Takano[†3]

We propose a 2×2 requirement chart enabling effective identification of items in ConOps, a standard provided by IEEE, and establish a methodology how to use it. The main focus of our proposal is to obtain requirements necessary for constructing CONOPS without any bias from realization methods and adopted techniques. By properly constructing ConOps through our chart, we are able to focus on the study period in systems engineering, and as a result reduce the costs in operation period. We developed two identical enterprise systems, each using a different method, and compared the cost in the operation period of each result. The first method, without creating ConOps in the study period, goes through systems architecture development and systems design based on initial requirements provided by the customers, the moves on to implementation period (case 1). The other method creates ConOps using our 2 × 2 requirement chart, then processes the rest in the same way as case 1 (case 2). Through this experiment, we verified the effectiveness of the proposed chart.

1. はじめに

1980年代後半以降，情報システムの導入から維持・管理等，システムを使い続けるために発生する費用の総額（TCO，Total Cost of Ownership）を問題にした議論が開始された．この時期は，企業を中心に情報システムの運用のアウトソーシング戦略が採用されるようになった頃でもある[1]．この戦略により，情報システムの運用コストが経営上目立つようになり，学界でも研究テーマとして取り上げられるようになったと考えられる．

一方，INCOSE（International Conference of Systems Engineering）は，システムのライフサイクルをシステム工学が考慮すべき全工程ととらえ，上流工程（Study Period）から始まり，開発工程（Implementation Period），運用工程（Operations Period）の順で移行すると定義した[2]．上流工程では，構築する情報システムに与えられた抽象的な目標が，具体的・工学的な仕様に変換される．一般に，情報システムは，(A) 工学的生産物を正確に製造し，(B) それを効率的・効果的に利活用し維持管理することで，目標どおりに機能する．この両者は互いに関係するので，仕様作成時には慎重な検討が必要とされる．特に後者の視点でシステムに求められる姿を具体化したものがConOps（CONcept of OPerationS）であり，IEEEがその標準構造を提案している[3]．換言すると，ConOpsは，(a) 対象領域の現在の状態（as is）と，課題が解決した際の同領域のあるべき状態（to be）を明示し，(b) 後者（to be）の実現のためには何を作る必要があるかを指示するものである．ConOpsが十分に検討されたかどうかはシステムの品質，特に運用の質を決定し，発生するコストに大きな影響を与えると考えられる．このため，宇宙・航空産業等の大規模複雑かつ安心・安全

†1 慶應義塾大学先導研究センター
Keio Advanced Research Center, Keio University
†2 嘉悦大学大学院ビジネス創造研究科
Graduate School of Business Innovation, Kaetsu University
†3 慶應義塾大学大学院システムデザイン・マネジメント研究科
Graduate School of System Design and Management, Keio University

を求められるシステム開発の現場では，重要視されている．

一方，情報システムを取り巻く環境は，日常的に変化している．このため，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したりする．これにより（たとえ上流工程で議論されなかったとしても），開発作業期間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てきた．この状況は，上流工程の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 ConOps を作成しない習慣が根付いている．特に，エンタプライズシステムは，実利用しながらサーバ上のプログラムを改良することで，漸進的に品質を上げることができる．これは，抽象性の高い要求仕様のまま，次工程に移行する大きな理由の 1 つとなっている．

しかしながら，我々は，エンタプライズ型情報システム構築プロジェクトにおいても，上流工程の充実は必要不可欠であると考える．ConOps を特定せずに情報システム開発を行うと，実際の利用開始後に使い勝手の悪さが発見されたり，重要だと思われていた機能の極端な利用度の低さが判明したりすることがある（2.2 節）．多くの機能を搭載したシステムは，監視する対象が増え，運用コストが増加する．それらが利用されない場合，システムの活用度にそぐわない高コスト発生もありうる．これらを根本的に解決するには，上流工程で ConOps を綿密に作成し，要求仕様の質を上げることが必須であるが，先に述べた背景から，上流工程の期間を短縮したいという要望に応えがちになる．そこで我々は，IEEE が提供する標準の ConOps を適切に特定するための，2 × 2 requirement チャートを開発した．これを利用すると，ConOps として特定する情報を想起しやすくなる．十分に質の高い ConOps を構築することで，上流工程を充実させ，結果的に運用工程で発生するコストの増加を防ぐことが可能になる．

さらに我々は，実際にエンタプライズシステムを 2 つの方法で構築し，運用コストを比較した．これらは，顧客から提示された initial requirement（初期要求）[⋆1] をもとに，上流工程で ConOps を作成せず，システムアーキテクチャ作成とシステムデザインを経て開発工程に移行する方法（ケース 1）と，同工程で我々が提案する 2 × 2 requirement チャートを利用して ConOps を作成し，続く作業を実施する方法（ケース 2）である．いずれのケースも開発工程後，運用工程に移行（つまりエンドユーザによる利用を開始）し，6 カ月間実稼働させ運用コストを記録した．

本論文は次の構成をとる．2 章に先行研究と関連研究を述べ，3 章に我々が提案する ConOps 作成用 2 × 2 requirement チャートの設計コンセプトと，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．また本論文では IEEE に準拠し，開発工程はシステムデザイン以降 Operation Period（運用工程）開始直前までを指す．さらに本論文では，システム工学上の要求として記述される文章を statement と表記する．

2. 先行研究・関連研究

2.1 運用コストに関する研究

IDC Japan が 2009 年 12 月 2 日に発表した情報システムの運用管理に関する調査によると，日本国内の約 500 社の回答者のうち 30%以上が “運用コストの削減” を，“情報漏洩対策” に続く重要課題として認識していた．

日経 BP 社が発行する経営や技術の情報誌の検索サイト[⋆2]で，“運用コスト” に関する記事を指定すると 1997 年以降，毎年数件から十数件ヒットする．そして Forsberg らは，多くの大規模複雑システムを対象に，各工程で発生するコストのライフサイクルコストに対する割合を調査した[4]．これによると，（結果的に発生した）総コストのうち，上流工程と開発工程で 40%消費され，残りの 60%は（ほぼすべて）運用工程で消費されている．さらに Swanson らは，運用工程で発生するコストは，(ア) 日常的な利活用に関するもの（操作：operation）と，(イ) トラブル防止ための監視および同発生時の対応に関するもの（保守：maintenance）に大別されるとし，保守コストがシステム開発費用に匹敵もしくは上回る場合もあると報告した[5]．しかも保守コストは情報システムのライフサイクルを通して，回避不能コストであると主張した．そして，学界からは，過去に発生したコストを計算式でモデル化し，将来に発生するコストを厳密に見積もる提案が出されている一方で，Glass は TCO を予測することは不可能だと示した[6]．社会的な環境変化を，高い精度で予測することができないことから，こういう試みは有効でないと説明している．

また，利用者の操作コストに注目した例としては，情報システムの業務コストに与える影響の調査報告がある[7),8)] が，いずれも運用コスト全体に与える影響はさほど大きくない．

2.2 上流工程と運用コストの関連に関する研究

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 IT プロジェクトの成否に関する調査によると，システムに実装された機能のうち 45%が「ほとんど利用しない/まっ

⋆1 業種，専門領域，国等によって，User Requirement，Wish Requirement 等とも呼ばれる．

⋆2 http://bizboard.nikkeibp.co.jp/

たく利用しない」，48%が「ときどき利用する」，そして「頻繁に利用する/つねに利用する」機能はわずか 7%であった．以後の同調査では，成功したプロジェクトの割合は 16%から 30%近くにまで伸びているものの，搭載機能の有効性に関しては大きな変化は見られない．多額の投資を行ったシステムが活用されていない様子が浮き彫りになっている．同様に，国内の情報システムの利用状況調査でも，この問題があげられた[9]．

最近になり，国内では情報システムの運用工程で最適なコストを実現するために，上流工程の成果物（設計結果等）との関連に注目した研究も行われている[10),11)]．これらはいずれも経験則に従って，上流工程の要求分析を充実させ，情報システムを成功裏に導入したものである．

2.3　同一システムを異なる方法で開発した比較実験

Anda らは，同一の要求仕様をもとに，複数の企業に情報システム開発を依頼し，それぞれで採用した手法や成果物に対する可変性（variability）と再現性/再利用性（reproducibility）に関し調査した[12)]．これによると，多くのシステムでは保守容易性（maintainability）に関し，運用工程における修復作業の高効率性と，高再利用性の成果をあげていた．この報告は，81 社に対し同一の情報システム開発の入札依頼を出し，返答のあった 35 社を対象にした大がかりな検証であり，実験結果の価値は高い．その一方で，完成度の高い（well-defined な statement からなる）同一の requirement を参加企業すべてに提示しており，質の違う requirement を対象に開発したシステムの運用の差は確認していない．

2.4　精度の高い要求仕様を構築する研究

上流工程の充実のために IEEE では標準書式である ConOps を整備している[3)]．これを精度高く特定しておくことで，どう作るかという視点で情報を整理する systems requirement[13)] も精度の高いものになる．反面，ConOps は（実現技術ではなく）システムやサービスが実現されたときに，対象となる業務の operation 面で，何をどう変化させたいかに注目して，情報を整理する枠組みである．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[14)]，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説明している[15)]．一方，現在，インタネットの普及にともない，先進的な情報技術を一般の利用者が日常的に使用することが多くなった．このため，情報システム構築に際し，先進的な技術を利用したいという潜在的な期待が大きくなり，技術先行で要求仕様に関する議論が展開されやすい．こうして作成された機能・性能・性質に関する statement に対し，何のためにそれらが必要かの確認作業が，後から発生するのは，前章に述べた上流工程に長時間をかけることにつながり，環境要件を満たしているとはいい難い．

2.5　本提案の焦点

我々は，本章に述べた内容から，運用コストの最適化には上流工程の充実が不可欠であり，またこのコストのうち，保守コストに注目すべきであると考えた．そして，上流工程の短縮と運用コストの削減を両立させる requirement 開発ツールとして 2 × 2 requirement チャートを考案した．我々が提案する手法は，いわばロジャーが提唱する requirement statement 要求エンジニアリングの推敲（elaboration）を，目的を変えないまま，より効率的な実施を可能にすることを目指し，玉井が示した目的指向型戦略を支援するツールの提供が狙いである．また，質の異なる requirement をもとに同一のシステムを開発し，運用コストを比較するという前例の少ないアプローチをとっている．

3.　2 × 2 requirement チャート

3.1　設計コンセプト

システムエンジニアリングにおける requirement は，視点をさまざまに変えて，多方面から議論されるが，生成される requirement の多くは (X1) 作成，(X2) 運用さらに，(Y1) Product/Service，(Y2) User/Customer のいずれかの視点に立ったものとして整理することができる．これを 4 つの象限上に俯瞰的に表現したものが，我々が提案する 2 × 2 requirement チャートである（**図 1**）．多くの initial requirement は，利用技術や実現方法の記載が目立つ．2 × 2 requirement チャートの (X1) 作成の視点かつ (Y1) Product/Service の視点で requirement statements を充実させている状態といえる．こうして作成された要求仕様は，技術的優位性を主張したシステム設計が可能になるが，その一方で，なぜその技

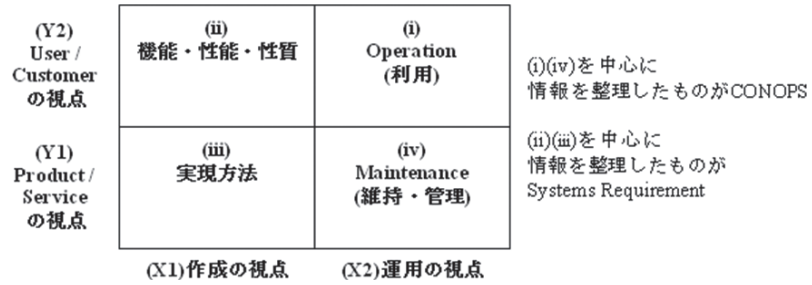

図 1　2 × 2 requirement チャート
Fig. 1　2 × 2 requirement chart.

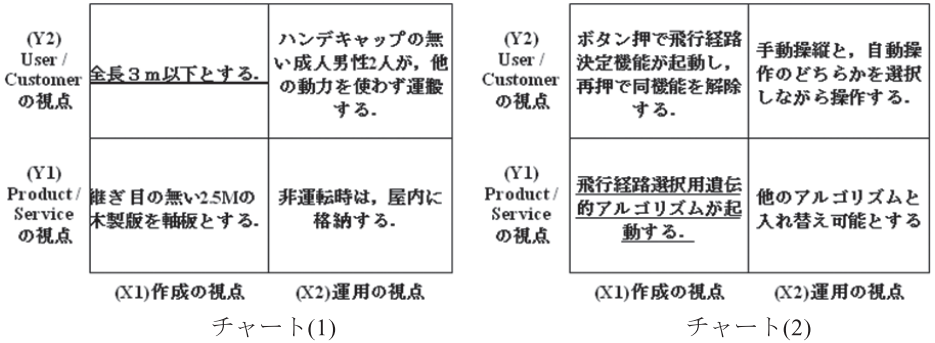

図 2　2 × 2 requirement チャートの利用例
Fig. 2　2 examples of applying the 2 × 2 requirement chart.

術や実現方法が必要かという確認が十分に行われない．その結果，利用者が（潜在的に）望む利用シナリオを実現することに直接つながらず，できあがったシステムが活用されないことが多い．一方，欧米では ConOps を中心に requirement を開発するのが主流になりつつある[16]．つまり，それは (Y2) User/Customer の視点かつ (X2) 運用の視点で statements が生成されることを意味する．実現した機能が十分活用される情報システムを構築するためには，ConOps に沿って整備された要求仕様書の構築が不可欠であると考えられる．つまり，図 1 の (i) と (iv) に相当する statements が requirement として十分に与えられ，それを直接実現するための機能・性能・性質と，さらにそれらを実現する具体的な手法や技術として，それぞれ (ii) と (iii) が検討されなければならない．

たとえば，1 人乗り飛行体を開発するプロジェクトの initial requirement 中の「簡単に取り扱えること」という記載を想定する．これを工学的表現に変換した結果，(a)「全長 3 m 以下とする」と (b)「飛行経路選択用遺伝的アルゴリズムが起動する」という 2 つの statements が生成されたとする．以下に，これら 2 つの statements から，2 × 2 requirement チャートを使って statements を増強する例を示す．

(a) と (b) は提案する 2 × 2 requirement チャートのそれぞれ (ii) と (iii) に相当する．それらを，**図 2** のチャート (1) とチャート (2) に示す．次に，変換された statement をもとに，他の 3 つの象限用の statements を生成する．図 2 の (1) に，新たに生成されたそれぞれの象限に対する statement「ハンデキャップのない成人男性 2 人が，他の動力を使わず運搬する」,「非運転時は，屋内に格納する」を示す．図 2 の (2) には，同様に作成された「手動操縦と，自動操作のどちらかを選択しながら操作する」,「他のアルゴリズムと入れ替え可能とする」を示す．これら 4 つの statements は，利用技術ではなく，どういう状態を実現したいかの特定である．したがって ConOps にふさわしい statements であるといえる．

3.2　2 × 2 requirement チャート利用手順

我々が提案する 2 × 2 requirement チャートは，実際の requirement 作業で，提示された statement がこのチャートの (ii) や (iii) に相当する場合，これらの精錬化作業に利用する．この作業は，提示された statement をもとに，(i) と (iv) の statement を Stakeholder 間で議論し，その結果をもとに (ii) の statement を（再）検討し，さらにこれら 3 つの statement を確実に実現する方法として (iii) を特定する．これにより，initial requirement をもとに，実現方法や利用技術に偏らない，利用者の意見を反映した十分な requirement が抽出され，その結果，ConOps に必要な情報が生成しやすくなると期待される．

具体的な 2 × 2 requirement チャートの使用方法は，以下に示す手順のとおりである．

① initial requirement 中の任意の statement を抽出し．それが工学的な表現でない場合，変換する．
② 上記 statement が，(i) から (iv) のどの象限に配置するのが最適かを特定する．
③ (ii) に特定された場合，その statement(FS_1) で示された機能・性能・性質が必要になる理由を Operation statement（OS）として Stakeholders 間で議論し[17]，(i) に配置する．
④ OS を実現するための（FS_1 以外の）別の新たな機能・性能・性質を検討し，1 つ以上特定する（FS_2 . . . FS_n）．
⑤ OS を実現する機能・性能・性質を，どのように維持管理すると，3 つの側面（Business と Budget とさらに Technical）上最適かを特定し[18]，(iv) の statement(MS) を設定する．
⑥ 第 2 象限用に生成された複数の statements(FS_1 . . . FS_n) の中で，MS を達成できるものを選択する（FS_MS）．
⑦ 選択した第 2 象限の機能・性能・性質である statement(FS_MS) を実現する方法（最適な COTS [⋆1]選択，実装/製造方法の決定）を，(iii) の statement として特定する．
⑧ 上記①〜⑦を繰り返す．

このように，我々が提案する 2 × 2 requirement チャートを用いることで，実現方法（技

⋆1 commercial off-the-shelf

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 生成に誘導しやすくなり，ConOps を効果的に開発することが可能になる．

4. 有効性検証実験

今回我々は，同一の initial requirement をもとに，2 つのエンタプライズ検索システムを開発した．これらは，上流工程の作業方法と成果物が異なる．具体的には顧客から提示された initial requirement をもとに，上流工程で ConOps を作成せずシステムアーキテクチャ作成とシステムデザインを経て，開発工程に移行する方法（ケース 1）と，同工程で我々が提案する 2 × 2 requirement チャートを利用し ConOps を作成したうえで，続く作業を実施する方法（ケース 2）である．両ケースとも，開発工程から運用工程への移行時に，開発したシステムをインターネット上に公開し，半年間一般の利用者に使用させた．そして，この半年間の運用工程で発生した作業コストを比較した．

4.1 開発したエンタプライズシステム概要

エンタプライズ検索システムは，（全情報を集中管理したときと同質な）異種情報の統合利用の実現を目指す[19]．つまり，このシステムの基本機能は，社内のネットワーク上に分散している全情報を対象とした検索であり，全社規模による情報活用や，KM（Knowledge Management）の基盤システムとして必要不可欠だとされている[20]．今回検証対象として，慶應義塾大学の学術コンテンツを対象にしたエンタプライズ検索システムの構築を採用した．

4.2 比較実験仕様概要

同じ目的を持つエンタプライズ検索システムを，2 つの異なるケースで開発し，実際に利用者に提供した．両者は，上流工程の作業方法とこの工程での成果物以外の差はほぼ存在しない．特に，上流工程から開発へのフェーズ移行判断基準（Decision Gate）が両ケースで異なると，正確な比較が困難になるため，INCOSE が提案する雛型[18] を両ケースとも用いた．すなわち，上流工程内の 4 工程（User Requirements，Concept Definition，System Specification，Acquisition Preparation）を順に実施し，開発工程に移行した．両ケースのプロジェクト実施体制は，次のとおりである．

【ケース 1】

検証期間

2005 年 10 月 3 日から 2006 年 4 月 2 日（6 カ月）

この期間に上流工程と開発工程を含む．2006 年 4 月 2 日に一般利用者による利用開始．

開発工程見込み予算

2,000 万円（上流工程で決定されるハードウェアとソフトウェアの購入費を含める．ただし人件費は含めない）

systems engineering メンバ

次のメンバで，上流工程と開発工程および運用工程を担当

システム設計担当（学部卒 IT 系勤務経験 5 年，1 件の情報システム開発プロジェクトを経験，実装経験言語 java，C，Ruby）

システム実装担当 a（修士 2 年生，実装経験言語 java，C）

システム実装担当 b（修士 2 年生，実装経験言語 java，C）

システム実装担当 c（修士 2 年生，実装経験言語 java，C）

上流工程の成果物

initial requirement をもとに，systems requirement，systems architecture そして system design を作成した．

上流工程予算

次のように開発工程見込み予算の約 6%を充当．

期間：2005 年 10 月 3 日から同 10 月 11 日（7 稼働日）

人員：全 systems engineering メンバ

【ケース 2】

検証期間

2007 年 10 月 1 日から 2008 年 3 月 31 日（6 カ月）

この期間に上流工程と開発工程を含む．2008 年 3 月 31 日に一般利用者による利用開始．

開発工程見込み予算

2,000 万円（上流工程で決定されるハードウェアとソフトウェアの購入費を含める．ただし人件費は含めない）

systems engineering メンバ

次のメンバで，上流工程と開発工程および運用工程を担当

システム設計担当（修士卒 IT 系勤務経験 3 年，2 件の情報システム開発プロジェクトを経験，実装経験言語 java，C，Ruby）

システム実装担当 d（修士卒 IT 系企業経験 3 年，情報システム開発プロジェクト経験なし．実装経験言語 java，C，Ruby）

システム実装担当 e（修士院 2 年生，実装経験言語 java，C）

表 1　上流工程での要求開発実施作業
Table 1　Requirement development jobs in our study period.

	フェーズ			
	User Requirements	Concept Definition	System Specification	Acquisition Preparation
Technical Aspectで実施すべきこと	ユーザからの初期要求の入手 CONOPSの作成 システム要求の選択	実現方法の候補群の作成	技術的実現方法の構築	実現担当の決定
ケース１での実施内容	1-1 Initial Requirementの入手 1-2 Initial RequrementのStatementを複数の工学的・定量的表現に変換 1-3 上記変換結果の中で最適なStatementsを選択し，Systems Requirementを作成	各システム要求を実現する方法を複数考案し，最適なものを選択した結果をSytems Archtectureとして作成	Component毎にどのCOTSで実現するか(もしくは製造するか), 統合作業はどのように実現と検証(Verify)するかを決定し, Systems Designを作成.	調達先(および製造担当)を特定
ケース２での実施内容 (提案する2x2Requirement Chartを利用)	2-1 Initial Requirementの入手 2-2 Initial RequrementのStatementを複数の工学的・定量的表現に変換 2-3 上記変換結果から想定できるOepration(操作)要求を生成し，COTSを作成 2-4 各操作要求を元にシステム要求Statementsを特定し, Systems Requirementを作成	各システム要求を実現する方法を複数考案し，最適なものを選択した結果をSytems Archtectureとして作成	Component毎にどのCOTSで実現するか(もしくは製造するか), 統合作業はどのように実現と検証(Verify)するかを決定し, Systems Designを作成.	調達先(および製造担当)を特定

システム実装担当 f（修士 1 年生，実装経験言語 java，C）

上流工程の成果物

initial requirement をもとに，ConOps，systems requirement，systems architecture そして system design を作成した．特に ConOps の作成に際しては，我々の提案する 2 × 2 requirement チャートを利用した．

上流工程予算

次のように開発工程見込み予算の約 6%を充当．

期間：2007 年 10 月 1 日から同 10 月 10 日（7 稼働日）

人員：全 systems Engineering メンバ．

4.3　実験結果：上流工程実施結果

ケース 1 は，開発工程に対する 6%分の日数を上流工程に充当した．事前に入手した initial requirement をもとに，上流工程で systems requirement を整理した．これを実現する systems architecture を作成し，component として最適なハードウェアと，ソフトウェア COTS を選定した結果，見込み予算内で収まらず，開発コストを 1,200 万円増額した．さらに，開発工程途中で障害が発生し，その解決のために別途 1,000 万円追加投入した．これにより，開発工程見込み予算に対し（開発終了時点で），3,200 万円超過した．ケース 2 も，開発工程に対する 6%分の日数を上流工程に充当した．同工程で作成した systems architecture を実現するために採用した COTS はハードウェアとソフトウェアの一体型であり，価格は見込み予算から 680 万円減額したものになった．**表 1** に，上流工程の各フェーズで実施した作業を整理した．同表に示すとおり，両ケースの作業は，user requirement フェーズで，3.2 節に示した手順に従って 2 × 2 requirement チャートを利用して ConOps を作成していること以外の差は発生していない．

表 2　2 × 2 requirement チャートを利用した requirement statement 生成例
Table 2　The actual examples of requirement statements using 2 × 2 requirement chart.

Initial Requirement	IR1: System-Operators shall edit contents of an index database.	IR2: End-users shall change the order of search result.
2x2 Requirementチャート利用手順②(3.2節参照)	第2象限(機能・性能・性質)に関するStatementとして特定	第2象限(機能・性能・性質)に関するStatementとして特定
2x2 Requirementチャート利用手順③(3.2節参照)	第1象限(操作)に関するStatementとして"System-Operators shall present additional information for each document on Search Result Page."を生成.	第1象限(操作)に関するStatementとして"End-users shall select documents published on particular Campus(es) ."を生成.
2x2 Requirementチャート利用手順④(3.2節参照)	上記statementを実現する，新たな第2象限(機能・性能・性質)用Statementとして"System-Operators shall input additional information in an index database, by identifying the specific document."を生成.	上記statementを実現する，新たな第2象限(機能・性能・性質)用Statementとして"End-user shall be able to select a particular campus for contents search."を生成.
2x2 Requirementチャート利用手順⑤(3.2節参照)	第1象限(操作)Statementの最適な維持・管理Statementとして第4象限に"System shall be maintained in 3 hours for each update operation."を生成.	
2x2 Requirementチャート利用手順⑥(3.2節参照)	上記維持・管理Statementを実現する最適な第2象限(機能・性能・性質)用Statementとして"System-Operators shall input additional information in an index database, by identifying the specific document."を選択.	上記維持・管理Statementを実現する最適な第2象限(機能・性能・性質)用Statementとして"End-user shall be able to select a particular campus for contents search."を選択.
2x2 Requirementチャート利用手順⑦(3.2節参照)	検索エンジンが参照するIndexDBに検索用キー情報を追加できる機能を持ったCOTSを選択：COTS_B	検索対情報をあらかじめ発信元URLで分類しておくことができる機能を持ったCOTSを選択：COTS_B

一方，この差によって，両ケースで作成される systems requirement に記述される statements が大きく異なる結果になった．たとえば，initial requirement の statement に，“System-Operators shall edit contents of an index database.”（IR1）と，“End-users shall change the order of search result.”（IR2）があった．ケース 1 では，両 statements はこのまま systems requirement として採用され，systems architecture とその component となる COTS の選択に影響を与えた．具体的には，ケース 1 では，(ア) 検索用の index database に対し，その内容を自由に編集可能な機能を持ち，かつ (イ) 別途作成する component（—検索結果出力画面に対し，エンドユーザの指示によるオンデマンドな並べ変えを可能にする—）を付加できるインタフェースを持っている COTS が選択された．一方，ケース 2 では，2 × 2 requirement チャートを利用した．そして，**表 2** のとおり，3.2 節に示した手順で statement の生成と選択が行われ，IR1 と IR2 より，それぞれ清廉化された別の statement が適していることが判明した．これにより (ウ) 検索用の index database に対し，検索用キー情報を追加できる機能を持ち，かつ (エ) 検索対象情報をあらかじめ発信

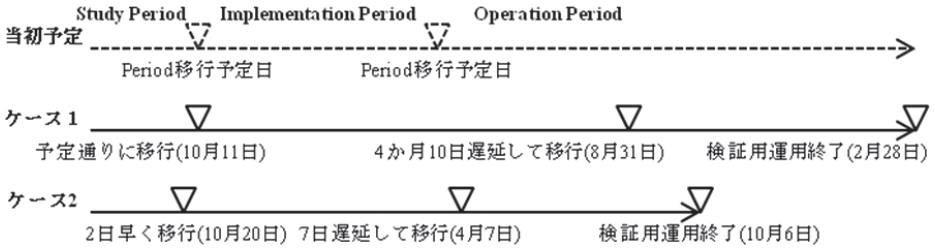

図 3　工程移行の計画と実績
Fig. 3　2 plans and actual performance of periods transfer.

元 URL で分類しておくことができる COTS が選択された.

4.4　実験結果：運用開始日

ケース 1 は，予定どおりの期間で上流工程が終了し開発に移行した．上流工程で作成した systems architecture を反映し，A 社製の検索エンジン（COTS_A）を導入した．subsystem integration 工程で予定外の障害が多発し，fabrication 作業に戻り修復にあたったが，再度 system integration 工程で重大な障害が発生した．A 社の技術者の追加充当を行ったが，システムの完成（開発工程が終了し，運用工程が開始）は，4 カ月 10 日遅延した．ケース 2 は，予定に対し 2 日早く上流工程が終了し，開発に移行した．上流工程で作成した systems architecture を反映し，ケース 1 とは異なる B 社製の検索エンジン（COTS_B）を導入した．システムの完成は，予定より 1 週間遅延した．両ケースとも，一般利用開始後，開発した検索システムの 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 3** に，上流工程と開発工程それに運用期間への移行予定と，それらに対する実施結果を示す.

4.5　実験結果：運用コスト

開発したシステムの運用コストは，2 種類 (a) 当初の目的どおりの利益を利用者に与えるための操作コスト（operation cost）と，(b) 正常な利用状態を保ち，またいずれかの stakeholders が正常な状態でなくなったと判断したとき，速やかに何らかの対応を行い正常な状態に戻す保守コスト（maintenance cost）に分類し，さらにそれぞれを作業別に記録した（**表 3**）．operation cost に関し，「ACL（Access Control List）情報の収集」作業は，両ケースとも運用期間 6 カ月の間に 1 回発生した．この作業に要した時間は，ケース 2 の方が 1 時間短い．その他の 5 つの作業すべてに関し，ケース 2 の方が，発生回数がない．また各作業の 1 回あたりの作業時間を見ると，「RDBMS 内データの収集」は 27 時間人であり，ケース 1 のそれの約 7 倍要している．operation cost のすべての作業を対象にした 1 回あたりの所要時間の平均は，ケース 2 が 7.2 時間人であり，ケース 1 のそれの約 2 倍である．ただし，「RDBMS 内データの収集」を除く 4 作業で見ると，1 回あたりの作業時間は 2.3 時間人でありケース 1 を下回る．maintenance cost に関し，「システム全体の再起動」は，ケース 1 では 2 回，ケース 2 では 1 回，それぞれ発生している．またこの作業の 1 回あたりの作業時間が，ケース 2 では 4 時間であるのに対し，ケース 1 のそれは 4 倍以上の 18 時間かかっている．また，「クローラの再起動」作業は，ケース 1 では（運用期間 6 カ月の間に）102 回発生し，1 回あたりの作業時間もケース 2 のそれの 2 倍以上要している．さらに，ケース 2 では発生せず，ケース 1 で発生した作業が 3 つある．具体的には，「システム起動不良対応」と「クローラ起動不良対応」，および「web コンテンツ収集不可対応」である．これらの 1 回あたりの作業時間は，いずれも 20 時間人を超えている.

表 3　運用工程で発生した作業一覧
Table 3　Whole jobs occurred in operation period.

		ケース 1			ケース 2		
	作業内容	発生回数	作業時間小計	1 回あたりの作業工数*	発生回数	作業時間小計	1 回あたりの作業工数*
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**
Maintenances 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

5. 考　　察

我々は，IEEE が提供する標準の ConOps の項目を効率的に特定するための，2 × 2 re-

quirement チャートとその利用方法を開発した．今回，これらを利用して情報システム開発を行った場合（ケース 2）と，そうでない場合（ケース 1）で，開発したシステムの運用コストにどのような影響が発生するかを検証した．今回確認した運用コストの差は，システムに必要な機能と性能を絞り込むこと（Boundary の特定[17]）に成功したかどうかの差であると考えられる．この絞り込みは，特に，実現したいシステムの利用シナリオと保守条件に焦点を当てたことが効果していると見られる．

ケース 1 は，initial requirement で提示された機能に対し，どうしてそれらが必要かを検討しないまま，検索サービスとして想定されうる “高機能性” と，“拡張性” に重点を置いた COTS を選択し，systems architecture 中の component として採用した．ケース 2 では，initial requirement として提示された機能や性能が必要となる操作上の理由を議論し明確にした．その結果，高機能性と拡張性を必須とせず，外部のサービスの出力結果を統合するインタフェースが充実した COTS が選択された．そしてその COTS が装備していない（が，systems requirement 中に存在する）高機能（“RDBMS 内データの収集”）は，新たな別の component として開発した．ケース 1 で採用した COTS は，この機能を標準で装備しており，運用工程で利用したときの作業時間が，ケース 2 と比較し，1/7 になっている．その一方で，ケース 1 では，“システム起動不良” と “クローラ起動不良” そして “Web コンテンツ収集不可” の 3 つの問題が起き，maintenance 作業が発生した．これらは，いずれも検索サービスにとって（検索エンジンの標準アーキテクチャに含まれる）基本機能である[21]．つまり，ケース 1 は，半年の運用期間で 1 回だけ動作を求められる高機能を搭載した COTS を選択したが，基本機能の安定動作が得られなかったといえる．一方，ケース 2 では，基本機能と他のサブシステムを接続するためのインタフェースの充実度に注目して COTS を選択したことで，安定稼働が実現できた．

また，特に開発コストの予算内実施という点でプロジェクトを成功させるためには，上流コストに対し，開発コストの 20%の投入が望ましいとされていた（2.2 節）[11]．一方，我々はこれより少ない上流コストで成功させることができた．この成果も我々が考案した 2 × 2 requirement 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．

さらに，それぞれのケースで発生した運用コストを，（対応したエンジニアの工数と単価を使い）累積した結果が**図 4** のグラフである．両ケースとも，上流と開発の両工程までに要したコストを 1 として表している．これは両ケースで，開発までに要したコストが異なるので（4.4 節），運用コストの発生状況を直感的に理解しやすくするためである．2.2 節に示したように，システムのライフサイクルで見ると，多くの事例では，システムが運用に移行し

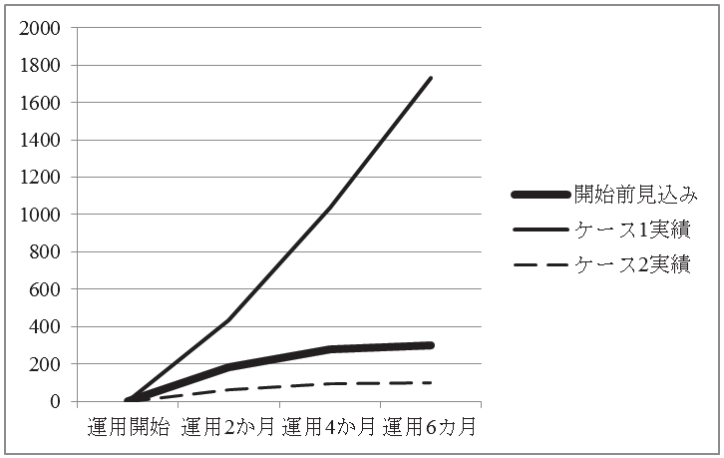

図 4　運用コスト（発生作業時間）
Fig. 4　Costs of operation period (working hours).

disposal されるまでに全プロジェクトコストの 60%が投入される[11]．ケース 1 に関し，今回の 6 カ月の運用期間終了時点で開発したシステムを廃棄する場合，この先行研究の報告どおりのコスト傾向となる．ただし，グラフに示すとおり，コスト発生の曲線は saturation 状態になっておらず，6 カ月を超えて運用を試みた場合，さらに指数関数的にコストが上昇した可能性がある．ケース 2 の場合，今回の運用期間と同じ上昇傾向をとったとしても運用期間が（6 カ月でなく）3 年であっても，全プロジェクトコストの 50%程度であると見積もることができる．一般に CPU の成長曲線や OS のバージョンアップ実績等を考慮すると，同一版のサービスが利用されるのは，最長で 3 年であろう．これらのことから，ケース 2 は，計画工程短縮と運用コスト低減の両面で，成功したといえる．一般に，実用システムを比較検討する場合，同様の環境で開発を行っても，正確性に疑問が生じる．具体的には，(A) 検証実施者が同じ場合，前ケースでの経験値が反映され，(B) 実施時期が異なる場合，新技術や製品の登場で，選択肢が著しく変化する可能性がある．今回，(A) に関し，両ケースでは同じ担当者は配置していない．また後者 (B) に関し，検証実験後の考察期間にケース 1 で採用した COTS の最新版を入手し，再度システムを組み上げ，同様の問題が発生することを確認した．このことから，両ケースの実施時期の差も，今回の比較検証の正確性を左右する理由にはならないと考える．

これまでの情報システム開発の研究報告では，同一のシステムを異なる方法で構築したものは（リソース面の問題が大きく），きわめて少ない．この点においても，本研究は新たな知見を提供したと考える．

6. ま と め

我々は，IEEE が提供する標準の ConOps の項目を効率的に特定するための，2 × 2 requirement チャートとその利用方法を開発した．我々の提案により，実現方法や採用技術に偏って抽出・生成されることなく，ConOps に展開できる requirements を容易に生成することが可能になる．我々は，提案する 2 × 2 requirement チャートを利用して上流工程を充実させた場合と，そうでない場合の 2 つの方法で，エンタプライズシステムを実際に構築し，運用コストに与える影響を検証した．2 × 2 requirement チャートを利用すると，上流工程で ConOps を効率的に構築し，運用工程を意識したシステムアーキテクチャの作成につなげることができた．具体的には，常時利用する機能の装備に限定した COTS をサブシステムとしての採用したシステム構成の考案であった．これにより，発生頻度の低い作業に対しては，高い運用コストが発生するが，日常的に利用される機能の運用コストを低く抑える情報システムの開発に成功した．

謝辞　本研究の一部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」によるものであることを記し，謝意を表す．

参 考 文 献

1) 溝口周二：情報システムのコスト・マネジメント，横浜国際社会科学研究，Vol.11, No.6, pp.593–609 (2007).
2) Forsberg, K., Mooz, H. and Cottenerman, H.: *Visualizing Project Management: Charts and Frameworks for Mastering Complex Systems*, *3rd edition*, Wiley, pp.84–89 (2005).
3) IEEE Std. Board: IEEE Guide for Information Technology, System Definition, Concept of Operations (ConOps) Document, IEEE Std., IEEE Computer Society, Reaffirmed 2007 (1998).
4) 榊原　康：会計システムに WTS を全面採用運用コストを年間 15 億円削減，日経コミュニケーション，2008.4.1, pp.102–105 (2008).
5) Swanson, E.B. and Beath, C.M.: *MAINTAINING INFORMATION SYSTEMS IN ORGANIZATIONS*, pp.4–9, John Wiley & Son, Chichester (1989).
6) Glass, R.L.: Predicting future maintenance cost and how we're doing it wrong, Software, Vol.19, Issue 6, pp.112–113, IEEE (2002).
7) 柴田博仁：大画面ディスプレイ・多画面ディスプレイの導入による業務効率化の測定，情報処理学会論文誌，Vol.50, No.3, pp.1204–1210 (2009).
8) Iwanaga, M.: System operation management solution of Hitachi, JP1, Version 5. Realization of TCO optimization of information system by effective operation management, *Business Communication*, Vol.36, No.10, pp.66–71 (1999).
9) 森山　徹：こんなシステムは作らない，日経システムインテグレーション，pp.22–38 (2005.3).
10) 桝田秀夫，小川剛史，町田貴史ほか：Diskless Linux を用いた情報教育システムの開発とその評価，情報処理学会論文誌，Vol.49, No.3, pp.1239–1248 (2008).
11) Forsberg, K., Mooz, H. and Cottenerman, H.: *Visualizing Project Management: Charts and Frameworks for Mastering Complex Systems*, *3rd edition*, pp.89–92, Wiley (2005).
12) Anda, B.C.D., Sjoberg, D.I.K. and Mockus, A.: Variability and Reproducibility in Software Engineering: A Study of Four Companies that Developed the Same System, *IEEE Trans. Software Engineering*, Vol.35, Issue 3, pp.407–429 (2009).
13) IEEE Std. 1233, 1998 Edition: IEEE Guide for Developing System Requirements Specifications, Digital Object Identifier: 10.1109/IEEESTD.1998.88826 (1998).
14) ロジャー S. プレスマン（著），西康　晴ほか（監訳），古沢聡子ほか（訳）：実践ソフトウエアエンジニアリング—ソフトウエアプロフェショナルのための基本知識，第 6 版，日科技連 (2009).
15) 玉井哲雄：ソフトウエア工学の基礎，pp.37–42, 岩波書店 (2004).
16) Halligan, R.: OCD&CONOPS in Capability Development, Materials of a Course Over Five Days, *Project Performance International*, pp.9–13, Adelaide, Australia (Aug. 2010).
17) Forsberg, K., Mooz, H. and Cottenerman, H.: *Visualizing Project Management: Charts and Frameworks for Mastering Complex Systems*, *3rd edition*, pp.8–18, Wiley (2005).
18) Forsberg, K., Mooz, H. and Cottenerman, H.: *Visualizing Project Management: Charts and Frameworks for Mastering Complex Systems*, *3rd edition*, pp.99–102, Wiley (2005).
19) Lewis, B.: Guest Editor's Introduction: A Glimpse at the Future of Enterprise Search, *IT Professional*, Vol.9, No.1, pp.12–13 (2007).
20) Huang, C.-C. and Kuo, C.-M.: The transformation and search of semi-structured knowledge in organizations, *Journal of Knowledge Management*, Vol.7, No.4, pp.106–123 (2003).
21) Hawking, D.: Web search Engine, *Computer*, June 2006, pp.86–88, ACM press (2006).

(平成 22 年 5 月 24 日受付)
(平成 22 年 11 月 5 日採録)

嶋津　恵子（正会員）

慶應義塾大学先導研究センター准教授．2002 年に慶應義塾大学より博士（政策・メディア）．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．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．情報システム学会，IEEE，INCOSE 各会員．

古川　康一（正会員）

1965 年東京大学工学部計数工学科卒業，1967 年同大学院工学系研究科修士課程修了．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．1982 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，1986 年同研究担当次長．1992 年慶應義塾大学環境情報学部教授に就任．1994 年同大学院政策・メディア研究科教授．2008 年 3 月定年退職．2010 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．

高野　研一

1955 年神奈川生まれ．1980 年名古屋大学大学院工学研究科博士課程前期修了．同年（財）電力中央研究所入所．1995 年名古屋大学より工学博士．1995 年から 1 年間英国マンチェスター大学 Visiting Research Fellow．2000 年人間工学会大島賞，2007 年慶應義塾大学大学院先導研究センターを経て，システムデザイン・マネジメント研究科教授．安全工学会編集委員，原子力技術協会，中央労働災害防止協会，NEDO，産業技術総合研究所等の専門委員会委員．技術システムのリスクマネジメントとヒューマンファクタおよび安全文化が専門分野．