要求工学プロセスの知識に関する研究

－発見的知識の体系とそれを表現するためのフレームワーク－

妻　木　俊　彦

目次

図一覧

表一覧

はじめに

本論文は，要求工学プロセスにおける偶発的な問題を解決するために必要な知識の体系化と表現法の試みについて述べたものである．

はじめに，こうした研究を行うに至った筆者の個人的な経緯について簡単に述べる．筆者は，旧日本ユニバックに入社以来，様々なソフトウェアの開発に立ち会い，要求の多様さとそれを定義することの難しさを目の当たりにしてきた．オンライン・トランザクションプロセッサの開発では，要求は開発者自身がソフトウェアの製品仕様に基づいて独自に定義しなければならなかったし，アプリケーションパッケージの開発では，要求をクライアント独自の要求と該当業務に共通した要求に分離することの難しさを知らされた．また，エキスパートシステムの開発に携わったときには，いわゆる領域の専門家といわれる一筋縄ではいかない人たちを相手に，専門的な知識を聞き出し，システムの目標と機能を定義するには様々な技法が必要なことを痛感した．オブジェクト指向の分析設計手法の研究を行いながら，構造化手法からオブジェクト指向への社内のパラダイムシフトを図ったこともあったが，静的モデルのような空間認識は，多くの技術者にとって習得の難しい技術であるのに対し，動的モデルのように対象の変化を認知することは容易な作業であることを知らされた．

こうした経験を通して学んだことは，単一の手法によってカバーできる問題の範囲は極めて狭いこと，また，人の認知特性の違いによって使用できる技法に適性があるということであった．一方，OMGのプロセス工学WGでのソフトウェア開発プロセスのメタモデル化の作業を通して，個々の手法は，独自に組み合わせて使用することが可能であることを学んだ．さらに，個人的な経験から，多くの技法は，その本質的な性質を理解さえしていれば，応用範囲が格段に広がるということも分かった．

しかし，現実のソフトウェア開発の現場では，要求の重要性が認識されながら，プロジェクト失敗の原因の多くが，相も変わらず，要求工学プロセスにあるという状

態から脱皮できないでいる．要求工学プロセスへの投資コストの少なさを問題とする意見もあったが，筆者は，要求工学技術自体が抱える2つの問題に注目した．

その1つは，要求工学技術の多様性の問題である．要求工学はソフトウェアそのものの意味を扱う唯一の技術であり，ソフトウェアの意味を定義するために，要求の発生源である複雑で曖昧な現実世界と，要求された機能を実現する厳密で正確なソフトウェアという2つの異なった世界を扱わなければならない．ステークホルダとの共同作業を通して，ソフトウェアが備えるべき機能や品質，性能などを明らかにし，それを開発するのに必要かつ十分な情報を収集するのが要求分析者の仕事である．しかし，決められた期間内に，問題領域の知識やソフトウェアへの要求を正確に理解するだけでなく，ステークホルダが欲している要求の本質を見抜き，あいまいな知識を明確にし，矛盾した要求を調整することも要求分析者の仕事の一部である．要求工学が，論理学やコンピュータサイエンスといった工学的な技術から，認識論や人類学，言語学といった社会科学的な知識をも含む学際的な技術であると言われる所以である[105]．ソフトウェア開発の現場で日々忙しく働いている多くの要求技術者が，このように多様で膨大な要求工学技術を習得し，利用できるようにするにはどうすればよいかということが，筆者にとっての1つ目の課題であった．

もう1つの問題は，熟練技術者が持つ経験的な知識や技術の取り扱いである．実際のプロジェクトでは，要求技術者が抱えている多くの問題を要求工学技法だけで解決することは難しく，熟練技術者がもっている知識やノウハウといった発見的知識の助けを借りなければならないことが多い．しかし，明文化されていない，その属人的な知識の実体を把握することは極めて難しい．著名な要求技術者らがどのように問題を解決しているかという Hikey らの調査によっても，豊富な経験とスキルにもとづいた熟練技術者の振る舞いの本質が明らかになったわけではなく，幾つかの個別のノウハウが垣間見えただけである[62]．実際，熟練者たちは独自のノウハウによって，プロジェクトの状況を判断し，問題解決に有効と思われる手法を適切に選択し，それらを上手に組み合わせて問題を解決している．もとより，熟練技術者のノウハウを一般の技術者がそのまま受け入れることは容易なことではない．筆者にとっての要求工学のもう1つの課題は，要求工学に関する様々な発見的知識を，個人的なノウハウから技術へと昇華することであった．

この2つの課題を解決したいというのが，本研究を行うことになった直接の動機

である．以下に，本論文の研究目標について，簡単に述べる．

研究目標

本研究の目標は，多様な要求工学知識を容易に利用できるようにすることであり，実際のプロジェクトで必要とされる要求工学知識のフレームワークを提案することである．高度情報化時代の到来とともに，ソフトウェアに対する要求は進化と多様化の波に晒され，それを定義するための作業はますます困難になりつつある．複雑な問題を解決するために多くの技術が開発され，提案されてきたが，技術の多様化は，一方で，過酷な現場で日々の作業に追われている要求技術者たちにとって，技術習得の障害となっている．本研究で提案するフレームワークが，要求工学に関する多様な知識に再利用の道を開き，現場の技術者たちを支援することを期待している．

本研究で提案するフレームワークは，工学的知識と発見的知識を含む知識体系とその表現法を含んでいる．体系化とは，複雑で理解が困難な問題を整理し，理解を容易にするための技術で，分割や抽象化，構造化といった方法がある[18]．本論文では，熟練技術者が個人的に持っているが明文化されていない経験則を発見的知識と呼び，未だ公開されていない個人的な技法や，工学的な知識の効果的な使用法（第3章），工学の対象外の心理学的あるいは社会学的な知見をも含んでいる．一方，確立された要求工学技法によって規定され，明文化され，公開された手順や制約に関する知識（第4章）のことを工学的知識と呼んでいる．これを技術とは言わずに知識と呼んでいるのは，規則そのものではなく，それを習得した人間の知識体系に焦点を当てようとしているからである．これらの知識の何れが欠如しても，問題を解決することは難しい．学而不思則罔、思而不学則殆．

知識の体系化

本論文では，要求工学技法の選択における発見的知識と，モデルの変更における工学的知識の体系化について提案する．要求技術者たちの個人的な経験に根ざした個別の発見的知識を再利用できるようにするには，その中から普遍的な知識を選別し，明文化する必要がある．しかし，選別されただけの知識は，たとえ普遍性を持っていたとしても，依然として個別のままであることに変わりはない．

関連した知識を集約し，知識体系を整備することが，収集された個別の知識を，より多くの技術者たちが有効に活用できるようにするための道である．発見的知識の体系化でよく使用される方法は抽象化である．もともと断片的な知識である発見的知識を体系化するには，知識が捨象されるリスクを回避するよりも，より本質的な知識を発見することの方が重要である．一方，要求工学技法の中で明示的に定義された工学的知識である制約条件を厳密に定式化するのに有効な方法の1つは，規則化である．技法を構成する規則は，それがどんな些細なものであっても捨象することは許されない．それゆえ，工学的知識を体系化するための方法として，規則の構造化がよく用いられる．規則の構造を定義するためにメタ知識が用いられることもある．しかし，メタ知識は規則全体の形式的構造を統語的に表現するだけで，その意味まで表現することはできない．

知識の表現法

体系化された知識を再利用できるようにするには，それらを何らかの形式で表現し，明示化することが必要となる．さらに，知識は，その性質によって表現の形式が異なっている．多くの表現法の中でもっとも自然な方法は，自然言語による記述である．自然言語による記述には，冗長性やあいまい性が伴う反面，文脈や構文といった付随的な機構による豊富な意味世界の表象が可能であり，属人的な発見的知識を表現するのに適している．第2章では，パターンの記述に自然言語を使用している．

一方，絵や図式といった図形を使った表現は理解の容易性に優れているが，表現できる情報量には限界があり，補完的な手段が必要となる場合が多い．第2章の状況遷移図や第4章のモデル連携依存関係は図式表現を使用することによって知識の全体像を直感的に表現しているが，それを理解するためには自然言語による詳細な説明が必要である．

対象同士の相対的な関係を視覚的に表現するために，座標空間を使用することがある．座標空間上の位置関係から，座標軸以外の新たな軸を発見できることもある．第3章の要求工学技術マップは，座標空間を使って要求工学技術の特徴を視覚的に表現している．

知識を規則の形で表現する方法は，情報同士の論理的関係を表現するのに適しており，機械化も可能である．第4章では，規則表現の一種であるプロダクシ

ョン・ルールを採用することによって，モデル連携規則の厳密な記述を目指している．

論文の構成

本論文では，第2章から第3章に亘って次の3種類の異なった知識の体系化と表現法について議論する．

- プロジェクトにおける個別の問題解決のための知識（第2章）
- プロジェクトに適した要求獲得技法の選択に関する知識（第3章）
- 要求変更におけるモデルの一貫性保証に関する知識（第4章）

これらの3種類の知識は，一見，無関係そうに見えるかもしれない．しかし，本論文のなかでは，発見的知識と工学的知識，個別の知識と体系化された知識という議論によって，要求工学に関する知識の全体像を構成していることが示される．以下に，各章の概要を示す．

第2章では，プロジェクトにおける偶発的な問題を解決するための発見的知識について議論する．異なった知識と経験を持った人たちが共同で作業を行う要求工学プロセスでは，しばしば予期せぬ問題が発生する．提案されている多くの技法や手法は，予想可能な問題に対する解決法を提示してはくれるが，偶発的な問題に対しては多くの場合無力である．手法による解が期待できない問題に，熟練技術者の知識が有効な場合がある．本章では，ソフトウェア開発における個別の知識を記述するための方法として注目を集めているパターン[53,54]を使って，発見的な知識を再利用する方法について提案する．しかし，これまでに提案されているパターンの多くはソフトウェアの設計に関する知識を記述したものであり[29]，プロセス上の問題を解決するための知識を記述したパターンの報告は少ない[59,142]．本論文では，ソフトウェア開発プロセスにおける個別の知識を記述したパターンをプロセスパターンと呼び，要求工学におけるプロセスパターン，すなわち，要求工学プロセスパターンの形式について議論するとともに，パターンをプロセスの進行に沿って体系的に表現するための状況遷移図を提案する．

第3章では，プロジェクトの特性をもとに，適切な要求工学技法を選択するための基盤となる知識の体系と，それを表現するための方法を議論する[143]．プロジ

ェクトを成功させる重要な要因の1つが，技法の選択にあることはよく知られている[62]．しかし，一般の要求技術者にとって，多様な要求工学技法の中から自身のプロジェクトにとって適切な技法を選択することは容易なことではない．本章では，代表的な要求工学技法を，その性質によって特徴付け，新たなプロジェクトが発足したり，プロジェクトの性質が変化した時，あるいは要求獲得作業中に障害が発生した時などに実施しなければならない適切な技法選択の基盤となるフレームワークを提案する．ここで扱う知識は，第2章で議論した要求工学プロセスパターンの中で個別に記述されていた手法選択のための発見的知識を集約し，体系化したものである．

第4章では，複数の異なった要求モデルを使用しているプロジェクトで，要求変更が発生した際に，要求モデルの一貫性を保証するのに必要な知識を取り上げ，その記述方式と体系化について提案する．状況の変化に応じて異なった手法を使用し，異なった要求モデルを作成したプロジェクトは，モデルの一貫性，すなわち，要求モデル同士の間に矛盾が無いということを保証する必要がある．新たなモデルを作成する際のモデル同士の一貫性の保障に関する研究はあるが[49]，要求モデルの変更に関する問題については，要求追跡の研究[141]以外，これまであまり議論されてはこなかった．ここで扱う知識は，第3章で扱ったような意思決定のための知識ではなく，具体的な作業を支援するための知識である．

これらの議論に先立ち，第1章では，要求工学プロセスにおける課題と，それを解決するための要求工学技術の全貌を概括し，本研究との関係について議論する．また，第5章では，全体のまとめを行い，本論文の成果について議論し，さらに，今後の課題として，多様な形式で記述された要求工学知識を，実際のプロジェクトに供するための支援システムについて述べる．

用語

要求工学は，ソフトウェア工学の他の領域に較べ，その成立が遅れたため，用語の慣用的な使用が長く続き，その定義が必ずしも統一されているとは言い難い状況にある．ここで，本論文で使用する若干の用語について言及しておくことにする．

「要求」という用語は，英語の“requirements”の日本語訳である．国内ではしばしば，要求の代わりに要件という用語が使われることがある．要求は顧客が望んでいること，要件は開発者と顧客が合意したことなどという説明がなされることもあるが，要件という言葉には，要求をソフトウェア側から捉えたという意味合いが強く現れており，それゆえ，システム仕様を含んで用いられることが多い．したがって，本論文では，requirements に対する用語として，「要求」という用語を統一的に用いる．ソフトウェア工学で使用する用語の定義をしている IEEE-610 における「要求」の定義は次のとおりである[68]．

“(1) a condition or capability needed by a user to solve a problem or achieve an objective; (2) a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents; (3) a documented representation of a condition or capability as in (1) or (2).”

ここでは，要求とは，問題を解決し，目標を達成するための機能や条件のことであると定義されている． Southwell は要求を，その内容によって機能要求と非機能要求に分け，さらに非機能要求を，効率，信頼性，インタフェース，設計制約に分類した[130]．IEEE においても，機能要求に加え，性能要求，インタフェース要求，設計制約，実装要求，物理的要求といった非機能要求が定義されている[68]．また，Robertson らは，要求をさらに詳細に検討することによって，機能要求，感覚的要求，使用性要求，性能要求，操作性要求，保守性と移植性要求，安全性要求，文化的政治的要求，法的要求といった9種類の要求に分類している[116]．本論文の第3章では，非機能要求のための技法の選択についても議論する．

必要かつ十分な要求を正しく定義し，それらを次工程の設計者たちに正確に伝達するには，それなりの技術やプロセスが必要となる．要求工学は，要求に関

わる技術を研究する領域である． Zave による要求工学の定義は，次のようなものである[152].

Requirements engineering is the branch of software engineering concerned with the real-world goal for, functions of, and constraints on software system. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families.

この定義によれば，要求工学は単に要求仕様書を作成するだけではなく，要求の進化や再利用にも関係している．Loucopoulos らも，要求工学を，問題を分析し，観察の結果を文書化し，理解の正しさをチェックするためのシステム的なプロセスであると定義している[89].

The term requirements engineering (RE) has been defined as the systematic process of developing requirements through an iterative co-operative process of analysing the problem, documenting the resulting observations in a variety of representation formats and checking the accuracy of the understanding gained. This definition is based on the premise that the RE process involves aspects of concern along three dimensions: the representation, cognitive and social dimensions.

要求工学の対象であるプロセスを，どのように定義するかというプロセスモデルの研究も盛んである．当初は，ウォーターフォール型[38]やスパイラル型[12]といったソフトウェア開発プロセス全体の中で捉えられることが多かったが，研究の進化に伴い，要求工学プロセスそのものの実行方式が提案されるようになってきた．Zeroualらは，要求獲得(eliciting)と要求表現(representing)という2つのステップを定義し[153]，Kramer らは，要求獲得(elicitation)，仕様化(specification)，検証(validation)という3つの繰り返し操作から構成されているとしている[85]．また，Rzepka は，要求獲得を，更に，要求の所有者の識別，要望書の収集，要望書の文書化と詳細化，要望書の統合，非機能要求の定義という5つの活動に分割している[121]．情報処理学会要求工学ワーキンググループでは，要求獲得，記述，検証，管理という4つの大きなステップを，スパイラルに実行する方式を提案している[107]．また，繰り返し型のアプローチとしては，事実

の発見，要求収集と分類，評価と論理化，優先順位付け，統合と確認という作業を必要に応じて自由に実行する方式[25]や，ソフトウェア開発プロセス全体の中で要求工学プロセスを必要に応じて繰り返し実行することのできる方式[86]なども提案されている．スパイラル型や繰り返し型は，要求変更への対応を容易にするために考えられたプロセスである．

日本では，要求工学プロセスの作業は，要求分析と要求定義という2つの部分に分けて考えられてきた．要求分析は，ステークホルダがもっている漠とした要望から，その意図を明らかにし，要求として明確にしてゆくための作業であり，最近は，要求獲得（Requirements elicitation）と呼ばれることが多い．また，分析の結果をモデル記述言語や要求記述言語によって仕様として記述する作業は，要求定義あるいは要求記述と呼ばれてきた．他方，システム化の目的やコストやスケジュールといったクライアントとの契約に関わる問題は，システム化計画という，要求分析とは異なったプロセスで行われるものと考えられてきた[138]．本論文では，要求獲得と要求定義という用語を使用する．また，プロジェクトの管理に関する問題についても議論の対象としている．

要求分析者とは，ステークホルダから要求を聞きだして，それを設計者につなぐ作業を担当する技術者のことである[25]．システムの利用が限られた人たちに限定されていた時代は，要求はユーザから獲得されるものと考えられていたが，情報化時代の到来とともに，その影響が多岐にわたるようになり，要求の提示者もステークホルダと呼ばれるようになってきた．すべてのステークホルダが，プロジェクトのメンバーとなるわけではないが，真の要求を知っているステークホルダがプロジェクトに参加しているかどうかは，要求の品質を大きく左右する．Christel は，購入者（customers / sponsors），ユーザ（users），開発者（developers），品質査定チーム（quality assurance team），要求分析者（requirements analysts）がステークホルダに含まれるとし[25]， Robertson らは，そのシステムの依頼者（client），ユーザ（user），購入者（customer），システムの操作員（operator），管理者（administrator），プロジェクトの責任者（owner），システムの開発者（engineer）を挙げている[116]が，本質的な違いは無い．むしろ，同じステークホルダでも，その背景や立場によって異なった要求を持っていたり，熟練した専門家，未熟な担当者，非常勤の作業者といったようなレベルの違いによって持っている知識や要

求が異なったりする[127]. 本論文では, 要求を提示する役割を担っている者のことをステークホルダと呼ぶことにしている.

最後に, 要求とは直接の関係は無いが, 技術に関係する用語について述べておく. 本論文では, techniquesに技法, methodsに手法, methodologiesに方法論という用語を割り当てている. 技法と手法の間に厳密な区別は無いが, 技法は, ドメイン分割やゴール指向分解[87]といった, 原始的な機能を提供するための方法を指し, 一方, 手法は, ユースケース法[74]やi*法[149]のような, 特定の目的を実現するための方法を指すといった使い分けをしている. しかし, どちらにも属する方法がある. 両者を指すときは, より原初的な技法という用語を使用している. それに対し, 方法論は, Objectory[74]やUnified method[86]のように, その実行方法まで規定した技法や手法の集合を指している[25].

謝辞

本論文をまとめるにあたり，終始ご指導とご教授を賜りました，東京大学大学院総合文化研究科，玉井哲雄教授には心より感謝申し上げます．玉井教授には，先生主催のゼミへの参加をお誘いいただき，ソフトウェア工学への扉を開いていただくとともに，論文の執筆を通した技術的な指導のみならず，その研究者としての姿勢をとおして筆舌に尽くしがたい薫陶をいただき，今日ここまで来ることができました．

本論文のテーマは，日本ユニシス株式会社での経験や情報処理学会ソフトウェア工学研究会要求工学ワーキンググループでの研究活動を通して培ってきたものです．日本ユニシス株式会社の上司や同僚には，多くの情報や示唆をいただき感謝いたします．なかでも，初期の段階で論文の執筆という世界へ誘っていただいた森澤好臣氏（現，北海道情報大学教授），オブジェト指向分析設計の共同執筆者として叱咤激励くださった故岩田裕道氏，ソフトウェア工学の何たるかを示してくださった山崎利治氏には大変お世話になりました．また，要求工学ワーキンググループの活動を通して，大西淳立命館大学教授，佐伯元司東京工業大学教授，筑波大学の中谷多哉子助教授には，多くの技術的な指導をいただきました．さらに，要求工学企業人グループ“mahoroba”のメンバー諸氏には，実務に関する議論を通して多くの示唆をいただきました．ここに，謹んで感謝の辞を呈します．

第 1 章

要求工学の現状と課題に関する調査

要求工学は，複雑なソフトウェアを開発するための中核技術であり[20]，そのゴールは優れた要求仕様書を作ることにある[25]．要求獲得，定義，検証といった活動を支援するために，様々な手法やプロセスモデルが研究され，提案されてきた．しかし，それにも関わらず，ソフトウェアの開発現場では，プロジェクトを破綻に導くような重大に問題に絶えず直面している[133]．本論文の目標は，現場で起きている重大な問題に適切に対処できるように，要求工学に関する工学的知識と発見的知識を整理することである．第2章以降の，要求工学知識の記述と体系化の議論に先立ち，本章では，研究の背景となる主要な技術を概括し，本論文の研究目標との関係について述べる．

1.1. 要求工学の課題

はじめに，プロジェクトが遭遇し，場合によっては破綻を引き起こすかもしれない要求工学上の問題を整理し，その原因について考える．

政策的問題

ソフトウェア開発プロジェクトにおける最初の，かつ，最も重要な仕事は，作成するシステムがどのように使用されるのかを知ることである．ステークホルダの関心は，コンピュータにあるのではなく，コンピュータが何をしてくれるかにある[41]．それにもかかわらず，システム開発者の視点は，コンピュータそ

のものに拘束されてしまって，システム化の目的やシステム境界といった本来の目標が明確にされないまま要求獲得作業が始まってしまう場合がある．そのようなプロジェクトで定義された要求の中には，実現の妥当性のない要求や，不要な設計情報などが紛れ込んでいたりする[25]．

はじめに，目標を確認しておくことが重要である．しかし，上位管理者の短絡的な思いや，現場からの風評だけに動かされたプロジェクトでは，クライアントの責任者に尋ねてみても，システム化の目的を明確にすることが困難な場合がある．このように，問題そのものが不明確な状況でも，ソフトな手法[24]を使って課題を明確にすることもできる．しかし，その地位にしがみつこうとする責任者の下で，目標を見失った多くのプロジェクトは，波間に揺れる木の葉のように，時間の暗闇を漂っているだけである．要求問題に入る前に，明確な方針を策定することが急務である．

経済的問題

ステークホルダの要求を実現するには，それに見合ったソフトウェアを作成するための時間とコストが必要となる．しかし，現実のプロジェクトが使用できる時間とコストには制約がある．それにも関わらず，分析者は，しばしば，制約を無視した要求に遭遇することがある．無謀な要求は，計画された期間や予算内での開発が難しいと判断されると，実装段階で無視されることになる．これはプロジェクトにおける経済的な問題である．プロジェクトの経済に影響を与えるもう1つの問題は，要求の変更である．要求変更が発生すると，開発や保守のためのコストが増加し，場合によっては代替案を採用する必要が出てくる．それゆえ，要求技術者には，代替案の選択とともに，ソフトウェア開発に関する時間とコストを見積もるための技術も必要とされる．こうした経済的な問題から，要求プロセスでの手抜きが強要される場合がある．しかし，手抜きによって生じる誤解や見落としといったエラーは，開発プロセスの後半に，より大きなしわ寄せとなって返ってくることに注意すべきである[11]．

これまで，経済的な問題の原因の多くはシステム化計画段階での見積もりのミスである場合が多かったが，今後は，要求の進化と多様化がその主要な原因となってくると予想される．経済的な問題は，技術的問題よりプロジェクト管理的な問題に発展することがある．第2章では，プロジェクトの環境

に強く影響を受ける偶発的問題を解決するための発見的知識について議論する.

組織的問題

プロジェクトが失敗する主要な原因の1つに, ステークホルダのプロジェクトへの不参加がある[133]. 実際, 要求技術者の多くがステークホルダの非協力的な態度に困惑している. ステークホルダ不在の問題は, インタビューやレクチャによる情報収集のときだけでなく, ゴール指向分析やブレーンストーミングといった, より本質的な要求を発見するためのグループワークの際などに, より大きな障害となる. 真の要求を知っているステークホルダを発見することは難しく, そのうえ, 多忙な彼らの協力を得ることは, さらに困難である. しかし, 問題領域に無知なステークホルダを参加させることは, ステークホルダの不在以上の混乱をプロジェクトに引き起こすことになりかねない[25].

その原因がステークホルダ側の問題であるにもかかわらず, クライアントとの力関係から, 問題の解決が要求分析者の側に委ねられる場合が多い. 要求技術者の経験と能力が, もっとも必要とされる場面である. 第3章では, プロジェクトの意思決定の構造を解析し, それを支援する方法について議論する.

文化的問題

プロジェクトの参加者たちの間に, 予期せぬ誤解が生じることがある. 誤解の主たる原因は, プロジェクト内における意思疎通の欠如である場合が多く, その結果,「要求された通りのソフトウェアを作っても, 顧客が喜ぶとは限らない」[119]という反省だけが残ることになる. 意思疎通を阻害する要因の1つは, メンバー同士のバックグラウンドの違いである. 知識や嗜好や立場の違いが, 意見の衝突を生む[25]. ステークホルダ同士の視点の違いによって生じる衝突を解消するには, 必要以上に多くの労力を要する場合が多い[14]. ステークホルダと要求分析者とでは, 問題領域に対する知識のレベルが異なるので[25], 両者の間で使用する知識や言葉の意味が異なるという場合もある[127]. 問題領域内だけで使用される独自の専門用語は, それを知らない分析者にさまざまな誤解を生じさせる. また, 会話の中で使用される曖昧な日常語も誤解を生む原因の1つとなる[25]. 俗語の使用も注意を要する. ステークホルダの発言が, 公式のものか非公式のものかを判別す

ることも重要である[41]. 一般常識と問題領域内での常識が異なっているために, ステークホルダの省略した常識に分析者が気付かないこともある[25]. また, 開発者同士の非公式の議論は, 分析者の目標に対する理解を変えてしまうことがあるので注意を要する[67]. だからといって, 分析者がステークホルダに用語の意味をその都度確認しているようでは, 作業が不効率になるだけでなく, ステークホルダの不信を招くことにもなる.

こうした問題を避けるために, 用語集の作成や議事録の確認などといった様々な方法が提案されている. しかし, 現象は多様であり, 問題が起きていることに気が付かない場合も多い. 問題の発見の仕方から解決法までを包含した, 広範な発見的知識が必要となるが, 集約化することの難しい知識の扱いについては, 第2章の個別知識の体系化で議論する

技術的問題

対象となる問題領域に関する知識や, 要求工学に関する技術の欠如が, プロジェクト崩壊の直接的な原因となる場合がある. 問題領域の特性やニーズをよく理解していないステークホルダの存在は, 不適切で曖昧な要求の原因となり, その結果繰り返される要求の訂正により, 何時まで経っても要求が確定されないという問題が発生する. また, 情報技術に対するステークホルダの理解不足が, 過剰な要求の原因となることもある. 問題領域に対するステークホルダの知識の不足を補うための分析者の知識や, 隠れた要求を引き出すための技術が期待される. 反対に, 情報量の多さが, 問題の焦点を曖昧にし, 誤解を生じることもある[85].

プロジェクトを破綻から救い出すには, それぞれのプロジェクトの状況を的確に判断し, それに適した手法を選択し[50], 問題を解決するための高度な技術と能力が要求技術者に要求される. しかし, 高度な技術を持った要求技術者を確保することは容易ではなく, 一方, 要求はますます高度化してゆく. 要求技術者の知識と技術力の向上は, 偏に, 技術者の精進にかかっているといってよい.

教育的問題

時代の進展とともに, コンピュータの利用環境はますます複雑化し, 情報システムが与える影響は, 単にシステムユーザの域に止まらず, 社会全体に広がってゆきつつある. 要求者が, ユーザからステークホルダに代わった所

以である．彼らは多様な視点と興味を持ち[41]，それに伴い，ソフトウェアに対するニーズも拡大と変化を続け[16]，それが要求の進化と多様性の引き金となっている[25]．一方，MDSOC[140] やオントロジー，エージェントといった，進化と多様性の問題に対応するための新たなソフトウェア技術の研究も進んでおり，利用者と開発者をつなぐ要求工学の役割がますます重大になっている．

進化と多様化に対応するには，問題の特性に合わせた柔軟な手法の選択と，それに伴うモデルの変更という問題を解決しなければならない．しかし，複雑化する問題を解決するには，いきおい技術そのものも進化せざるを得ず，現場の技術者に対する教育の重要性が増している．本論文の，要求工学知識に関するフレームワークの提案は，要求の進化と多様性という新たな問題の解決に如何に貢献できるかを具体的に問うものである．

1.2. 要求工学技術の概要

要求工学プロセスの中でプロジェクトが遭遇する問題を解決するために，様々な技術が開発され，要求工学として集大成されてきた．ここでは，要求工学技術を，その性質によって要求獲得技法と要求定義技法に分けて整理し，最後に，要求の多様性と進化という新たな問題に対応するための技術をまとめて議論する．

1.2.1. 要求獲得技法

要求獲得(requirements elicitation)は，要求の所有者であるステークホルダを通して，これから作成しようとしているソフトウェアに対する具体的な要求や，問題領域に関する知識や情報を収集するための作業であり，情報収集(information gathering)などとも呼ばれる．

B.Nuseibeh らは，要求獲得技法を，次のような6つのタイプに分類している[105].すなわち，インタビューに代表される伝統的技法，複数のステークホルダによるグループ技法，仮製品を評価するプロトタイピング，モデルを使って情報を分

析するモデル駆動型技法，特定の個人の振る舞いに注目する認知技法，観察された現象の普遍的なモデルを作ろうとする文脈的技法である．これは，要求獲得作業で使用する方法や道具に注目した分類で，技術の全体像を捉えるのには便利な分類法であるが体系的ではない．この分類法を参考にして，要求獲得のための代表的な技法を，モデル駆動型技法と非モデル駆動型技法，および関連技法とに分けて議論する．

複雑で困難な要求獲得作業を支援するために様々な技法（Techniques）や手法（Methods）が提案されてきた．現場の要求分析者にとっての課題は，そうした多様な技術の選択と適用である．分析者の視点による要求獲得技法の分類は，第3章で取り扱う．ここで取り上げるドメインモデル，ゴール指向モデル，ユースケースモデルはそれぞれ，第3章で定義する領域分割型，ゴール指向型，シナリオベース型の技法を代表するモデルであり，非モデル駆動型技法の多くはブレーンストーミング型技法に属している．

(a) モデル駆動型要求獲得技法

モデルは問題領域を抽象化したもので，モデルを使うことによって，要求に対するプロジェクトメンバー相互の理解を深めることができる．モデルとして抽象化された問題領域は，システム化の目標の妥当性や，個々の要求の完全性を検証することを容易にし，ステークホルダと要求分析者のコミュニケーションの基盤を提供する．

モデルという言葉は，模型や模範，あるいは理論に対する実現というように，様々な意味で使われるが，ソフトウェアの世界では，構造を持った対象の抽象的な記述をモデルと呼んできた[138]．モデル駆動型技法は，獲得された問題領域の知識や要求を，モデルによって表現し，分析するタイプの技法の総称である．現在使用されている主要な要求獲得技法の多くは，問題解決のためのアプローチが異なっていても，このカテゴリーに分類される．しかし，直感的理解には優れていても，図式には十分な説明能力があるとはいえず，図式によるモデルに論理的な厳密性を与えるために形式化が用いられることがある．形式仕様がモデルに分類される理由である．

ドメインモデル

ドメインモデルは，問題領域を構成する実体と，それらの関係を抽象化したものである．ドメインモデルは具体的で明快なモデルであり，問題の本質を明らかにし，問題領域に関する知識の曖昧さや矛盾を解消したり，異なった考え方を持ったメンバー間の理解と意見の統一を図ったりするのに有効である[83]．

実装言語の進化とともに，ドメインモデルの構成要素も変化してきた．問題領域をモデル化することの重要性をいち早く唱えたのは，M.Jacksonである．Jackson は，当時の中心的なプログラミング言語である手続き型言語を前提に，実体とその行動ならなる実体構造図を提案した[70]．しかし，その両者を同時にモデル化することは容易ではなく，その後登場したオブジェクト指向言語を前提とした多くの手法では，実体の静的側面と動的側面を別々にモデル化しようとしている．そこでは，実体の概念はクラス図によって，その振る舞いはシーケンス図などの補助的図式を使って表現される[119]．問題領域の複雑化と，ソフトウェアの実装技術の進化に伴い，よりフレキシブルなモデル化が求められるようになり，実体より，より本質的な構成要素である役割（role）をベースとしたモデル化が注目されるようになった．役割によるモデルは，エージェント指向設計[147]や，動的なオブジェクト指向設計[139]などを通して，多様な実装環境への柔軟な適用が可能である．

A.Davis は，問題領域をモデル化することのメリットとして，ステークホルダ間のコミュニケーション，システム境界の設定，問題領域の分割と抽象化，問題レベルでの分析，情報の衝突箇所の特定，分析のための知識構造の容易な変更を挙げている[35]．しかし，問題領域のモデル化には，対象領域を抽象化する能力が求められるので，困難な作業であることが多い．

ゴール指向モデル

ステークホルダの意図を基に，要求を達成すべきゴールとして構造化したのが，ゴール指向モデルである．ゴール指向モデルでは，組織や個人の目標を表現することによって，より本質的な要求を明らかにすることができる．ゴール指向モデルの一種であるエンタープライズモデルでは，企業内の組織や担当者は，与えられたゴール（目標）を実現するために行動するという前

提に立っている[90]. そういう意味で, ゴール指向モデルは, 組織の意図を反映したモデルであり, 実装環境に影響されないという特徴がある. ドメインモデルが機械論的世界観(the mechanistic view of the world)にもとづいたモデルであるのに対し, ゴール指向モデルは目的論的世界観(the teleological view of the world)にもとづいたモデルであるといえる.

ユースケースモデル

多くのモデルは問題領域そのものを抽象化しようとしているが, システムに対する要求を, より直接的に表現しようとしたのがユースケースモデルである[74]. I.Jacobson の方法論 Objectry の中で使用されていたモデルが, UML(Unified Modeling Language)の中に取り込まれ, 広く使用されるようになった.

ユースケースは, ステークホルダの視点から捉えたシステムへの機能要求であり, その詳細はシナリオによって定義される. ユースケースモデルが表現できるのは機能モデルだけであるという批判に対し, Jacobson は, 非機能要求は機能要求の属性であり, 機能要求が定義できれば, FURPSモデル[57]を使って非機能要求の導出は可能であるとしている[77]. また, ユースケースシナリオとともに, 事前条件や事後条件などを明確にすることによって, ユースケースモデルを詳細化し, 厳密性を高めるためのユースケース記述が提案されている[28].

問題モデル

問題領域を別の時空間に写像することによって, 取り扱うべき問題を明らかにすることができる. M.Jackson は, 実世界を問題フレーム(Problem frames)空間に写像することによって, 未知の問題を既知の問題に収束させ, 解決法を提示しようとしている[73]. 問題フレーム空間は, 問題を抱えている実世界と, その問題を解決するための情報機械としてモデル化され, 問題はさらに小さな基本問題に分割されている. Jackson は, 問題フレームの記述法とともに, いくつかの基本問題に対する解も提示している. 基本問題に対する解の集合が, 元の問題の解となる. 問題モデルもドメイン分割手法に属するモデルということができる.

(b) 非モデル駆動型要求獲得技法

非モデル駆動型の要求獲得法は，具体的な要求を個別に収集する方法で，特定の対象に焦点を当てることによって，対象を様々な角度からより深く検討することができる．しかし，無作為に収集された要求は構造化されていないので，その全体像を把握することが困難である．収集が終わった後で，全体を整理しなおす必要がある．

調査技法

構造化されたインタビューは，情報収集のための最も一般的で強力な手法である[155]．一見簡単そうな技法であるが，正しい要求を効果的に収集するための適当な手順やツールをもたないため[153]，必要な情報が収集できなかったり，不必要な情報を大量に収集してしまったりすることがある．要求を聞き出すために，ステークホルダが行っているタスクに関する情報を収集する．しかし，タスクには，現在行われているタスクと将来の望ましいタスクがあり[79]，どちらのタスクを要求のベースとするかは，システム化の目標に依存する．要求技術者には，インタビュー技術とともに，収集した情報を整理したり，情報の真偽を判定するためのスキルが必要とされる．もちろん，インタビューを受ける側にもそれなりのスキルが要求される[10]．P.Ferdinandiは，Webアプリケーションのための要求獲得作業で使用する，インタビューのための具体的な質問を，目標，使用形態，獲得質問という形で列挙している[47]．

調査によって問題領域の知識や，個別の要求を収集する手段としては，インタビューの他にも，アンケートによる調査，ドキュメントの調査，作業現場の観察，分析者が実際に作業体験をする徒弟制などといった方法もある．何れの方法も，収集された情報の整理が課題として残る．

グループ技法

グループ技法は，チーム･アプローチなどとも呼ばれ，複数のステークホルダや要求技術者が一堂に会して，情報や意見を交換し，知識の共有化を図るための技法である．ステークホルダ自身が実際のグループ作業に参加

することによって，知識や要求の違いを，ステークホルダ同士の会話によって解消する．グループ作業の効率や，統合化された情報の品質などは，会議をリードするファシリテータの能力に依存するところが大である．また，唯でさえ忙しいステークホルダを，場合によっては，長期間拘束せざるをえないという問題が残る．

グループ技法は，新しいアイデアを創出するのに有効である．ブレーンストーミングは，アイデア発想法の1つで，グループメンバーの発想の違いから，さらに多くのアイデアを生み出そうという手法である．KJ法は，人間の直観を用いて問題の意味や構造を整理し，新たな解や発想を導出する[84]．

フォーカスグループは，マーケティングでよく使われる技法で，特定のテーマに関し，無作為に選んだメンバーによる話し合いによって，様々な意見を収集する[134]．特定の個人を対象にしたインタビューに比べ，多様な意見の収集が可能となる．

ステークホルダの声を直接反映し，コンセンサスを得るためにもグループ技法が用いられることがある．Joint Application Development（JAD）は，1970年代にIBMによって提唱された技法[151]で，ステークホルダ自身が問題を定義したり，解を発見するのを分析者が支援する．JADチームは，ステークホルダと熟練したファシリテータによって構成され，JADセッションを通して，要求を共有する．User Centered Design [101]や Participantory Design [58]も，JADに似た手法である．

シナリオ

シナリオは，現実世界の物語である[22]．CREWS(Cooperative Requirements Engineering With Scenarios)プロジェクトでは，現状の問題についてのシナリオを通して，ゴールを検証したり，新たなゴールが発見できることを明らかにした[94]．シナリオは，主に，自然言語によるテキストを使って記述されるが，簡単なマルチメディアを使う場合もある．自然言語は，人間の思考や感情を表現するために生まれた言語であり，曖昧で冗長である反面，それだけ多くの情報を含み，シナリオ分析などを通して，様々な発想を誘発することができる．シナリオから自然言語の持つ冗長性を排除するためのモデル化手法も試みられている．大西は，格文法[48]を使ったシナリオの解析法を提案している[106]．

プロトタイピング

プロトタイピングは，その品質を定量化できない要求や，ステークホルダの意見や反応を必要とする要求を確定するために用いられる手法であり[36]，感覚的な判断が必要とされるユーザインタフェースへの要求を確定するときなどに良く用いられている．プロトタイピングによって得られた情報は，グループ技法の議論のための情報として使われたり，アンケートやプロトコル分析のための基礎データとしても利用される．

プロトタイピングは，曖昧な要求を確定するためだけではなく，複雑なアルゴリズムの検証や，開発の可否を判定するためのデモンストレーションとしても用いられることが多い．このような作業を通して得られた情報も，要求の一種である．

認知技法

認知技法は，知識システム構築のために，専門家の知識を獲得するための知識獲得法の流れを引いている[129]．特定の個人の行動を解析し，その背景にある認知プロセスを通して，その知識構造を解明しようという技法である．

プロトコル分析では，その認知プロセスを観察者に伝えるために，指名されたステークホルダは自らの行動を声に出しながら作業する．ラダーリングは，事前に用意された調査用の定型的な質問を順次行い，その答えから知識の構造を解明してゆく．カードソーティングでは，ステークホルダに，キーワードの書かれたカードを，問題領域別に分類してもらう．レパートリ・グリッドは，実体ごとに，その属性と値をステークホルダに訊ねることによって，属性マトリックスを作成する．いずれも，特定のステークホルダの知識構造を調べるための手法である．

文脈的技法

文脈的技法は，伝統的な技法や認知技法の代替として1990年代に登場した[56]．他の技法が対象世界を抽象化しようとするのに対し，文脈的技法では組織的，社会的な理解が重要であるという考えのもとに，観察された現象の普遍的モデルを作ろうとする[113]．ここでは，対象の観察といった民俗学的な手法が使われ，会話分析では，会話のパターンを見つけるため

に詳細な分析が行われる[145].

（c）関連技法

獲得されたばかりの要求には，様々な矛盾や曖昧な要素が含まれている．工学的な技法だけでそれらの問題を解決することは困難であり，そうした場合には，発見的な知識の活用が有効なこともある．要求の品質を高めるために，いくつかの関連技術が提案されている．ここでは，技法として確立している，合意形成と優先順位付けのための技術を取り上げる．問題解決に有効な発見的知識の記述と体系化については第2章で扱う．また，優先順位付けのための技法は，第3章で議論する．

合意形成

利害関係の異なったステークホルダ同士が，異なった要求を持つことは避けられないことであるし，問題領域に対する共通の理解だけで，それが解消できるとも限らない．異なった意見を持ったステークホルダ同士の合意を形成するために，個々のステークホルダの貢献度をモデル化し[42]，妥協点を発見する必要がある[117]．これをネゴシエーションと呼ぶ．

WinWin アプローチは，ステークホルダ間での要求の違いを解消するのではなく，互いの妥協点を求めるための手法として提案された[14]．意見の相違は，condition，issue，agreement，option という4つの基本スキーマと，それらの関係によって定義され，これをもとに，ステークホルダ同士のネゴシエーションを行い，合意を形成しようとするものである[15]．このアプローチは，中東戦争をモデルにしたネゴシエーションのための1つの方法であるが，国際紛争の調停作業が多様であるように，すべての異なった要求がこのアプローチによって解決するわけではない．

優先順位付け

期間とコストという制約下にあるプロジェクトでは，ソフトウェア開発における優先度を決めるために，個々の要求に対し，優先順位づけを行う必要がある[69]．優先順位の決定は，解決すべき問題の重要度や緊急性，ステークホルダの好みなどを考慮して決定されるため，相反する複数の意見に対す

る意思決定機構が必要となる．優先順位付けのための代表的な技法に，品質機能展開，ソフトシステム方法論，階層化意思決定法などがある．

品質機能展開（QFD：Quality Function Deployment）は，顧客の声を製品開発の各ステージで適切な技術要求に変換する手法であり，関係マトリックスを用いて顧客の要求と製品の機能を関連付けている[1, 6]．この構造を要求獲得に応用し，関係マトリックスによって顧客要求の重要性を，製品機能の順位付けに反映することが可能である[25]．

ソフトシステム方法論（SSM: Soft Systems Methodology）は，複数のメンバー間の異なった考えを，問題状況に即して評価・決定する方法である．はじめに，問題状況の本質的な目標を認識する．次に，それを実現するための可能な問題解決案をメンバーから提示してもらい，モデル化する．モデル化に当たっては，顧客や変換プロセスといった6種類（CATWOE）の要素を使ってその解決案を定義し，それぞれの案を，効果や効率といった視点から評価する[24]．

階層化意思決定法（AHP: Analytic Hierarchy Process）は，複数の代替案に対する意志決定を行うための技法である．問題を階層構造あるいはネットワーク構造を使って表現し，個々の案を一対比較によって評価し，その構造内での相対的な関係をつくり上げる[123]．AHPを使うことによって，物理的な事象だけでなく，心理的な事象を評価することも可能である．

1.2.2. 要求定義技法

要求獲得技法を使って収集された要求や問題領域に関する知識は，その内容の厳密性を高めるためにモデルや仕様として定義され，要求仕様書に記述される．この作業を要求定義と呼ぶ．

要求定義では，問題領域に関する知識や要求をモデルとして表現する場合と，要求そのものを仕様として記述する場合があるが，要求を記述するために使用する言語は，要求獲得フェーズから引き続いて使用されることが多い．要求仕様書は，特別な教育を受けていない設計者にも伝わるように，自然言語を使ってより分かりやすく記述されることが多いが，リアルタイムシステムのように厳密性を要するシステムの場合には，状態遷移図や形式仕様記述言語がよく用いられる．

IEEE-830は，要求仕様書に記述されるべき項目についての提案を行っている[69].

ここで取り上げるのは，要求モデルや要求仕様を作成したり，保守したりするためのモデル記述言語や要求仕様記述言語である．工学的知識に属する知識の表現と体系化については第4章で議論する．

（ａ）モデル記述技法

モデル記述言語は問題領域を抽象化し，それを，図式などを使って記述するための言語であり，ビジュアル言語などとも呼ばれている．モデル化手法では，要求は，モデルの中から浮かび上がってくると考えられる．言い換えれば，モデルは要求のもう1つの表現である[138]．図式によって表現されたモデルは，テキストを使った表現に比べ，直感的な理解に優れている反面，論理的な説明能力には乏しいという側面を持っている．

モデルの記述能力は，モデル記述言語の記述能力に依存する．それぞれのモデル化手法は，独自のモデル記述言語をもっており，IDEF[52]や UMLのように，記法の標準化が進んでいるものもある．UMLは，様々なオブジェクト指向方法論のモデルの記法を統一するために標準化団体のOMGによって制定されたもので，複数の図式から構成されている．

要求をモデル化する最初の試みは，ステークホルダとシステムの動的な変化や機能的な振る舞いをモデル化することであった．構造化手法では，実世界（問題領域）の業務から情報（データ）の流れを抽出し，データを処理するためのシステムの機能を，データフロー図を使って表現した[38]．一方，オブジェクト指向における実体の振る舞いは，シーケンス図[119]や状態遷移図[74]などを使ってモデル化が図られている．初期のモデル化手法の多くは，要求工学技術マップ（第３章）では，シナリオ型の手法に属していることが分かる．形式的手法は，対象の状態変化を数学的に記述するもので，習得が困難であるが，事前条件から事後条件への変化を計算できるため，分析の自動化が期待できる[122]．

多くのドメインモデル記述言語は，プロジェクトメンバー間で共通の認識を得やすい実世界の実体に焦点を当て，その性質と振る舞いを記述しようとしている．JSD法の実体構造図が実体とその行動を1つの図式で表現する[70]のに対し，

現在のオブジェクト指向手法では，静的モデルと動的モデルのための記述言語は異なっている[119]．ドメインモデルは，システムへの要求を直接的に表現するというより，問題領域の構造を正しく理解し，実世界とソフトウェアの構造的な同一性を実現するために用いられることが多いので[96]，複数のモデルを使用した場合[103]の中心的なモデルとなる場合が多い．

KAOS[87]や i* [149]といったゴール指向型のモデル化手法では，組織の構造や目標，構成メンバーの役割や責務，タスクなどに焦点を当て，組織行動の基盤となっている目標という抽象的な概念を記述しようとしている．ゴール指向モデル記述言語では，特定の目標をゴール分解することによって，階層化されたサブゴール木が展開され，要求はゴールの操作化[87]や Means-end分解[148, 26]などを通して導出される．また，複数の上位ゴールに対する寄与度を計算することによって，採用すべきサブゴールを選択するような方式も提案されている[82]．

ドメインモデルやエンタープライズモデルのように問題領域を抽象化したモデルでは，要求はそれらのモデルの中に暗黙的に含まれているという立場をとっているが，より直接的に要求を表現したいという要求に対応したのがユースケース図である[74]．ユースケース記述言語が描くのは，アクターとそれに関連した機能要求としてのユースケースだけである．ユースケース図の厳密性を高めるために，シナリオを柱とした，自然言語によるユースケース記述も使用されている．ユースケースモデルは，単に要求を記述するだけでなく，ソフトウェア開発工数の見積もり[97]や，プログラムのテスト[91]などにも使用されている．

（ｂ）仕様記述技法

モデル記述言語が要求や問題領域を簡潔に表現することを目的としているのに対し，仕様記述言語はステークホルダの要求を厳密に記述することを目指している．機能要求や性能要求，品質要求といった様々な要求を表現するためには，機能，性能，品質，インタフェースなどを表現できる仕様記述言語が必要となる[69]．仕様を記述するために，論理表現[8]から自然言語[5]に至るまで，様々な形式的，準形式的，非形式的な言語が提案されているが，どの言語を使用するかは，仕様に何を記述するかによって異なってくる．そのことは，とりもなおさず，仕

様とは何かという問題に帰着する．仕様に関する代表的な考え方には，次のようなものがある．

M.Jackson は，環境と機械（システム）によって仕様を説明している[71, 72]．彼によれば，要求はシステムの外にある環境について述べたものであり，システムは要求を実現する手段である．一方，仕様とは，環境とシステムの境界上で，システムの振る舞いを，現象として記述したものである．このとき，環境特性（E）と仕様（S）が満たされることによって要求（R）が満足され（S, E ⊢ R），プログラム（P）とそれが動く機械（C）の特性によって仕様（S）が満足される（C, P ⊢ S）という命題が証明されなければならない．一方，D.Parnas は，利用環境とソフトウェアの間の4変数モデルによって要求と仕様との関係を説明している[110]．環境とソフトウェアの間には，センサーなどの入力デバイスと，アクチュエータなどの出力デバイスがあると考える．環境は，入力デバイスへの観測変数によって認識され，出力デバイスからの制御変数によって操作される．観測変数と制御変数の関係が要求であり，入力データと出力データの関係が仕様であるとしている．

要求獲得や分析の段階では図式表現であるモデルが用いられることが多いが，仕様の記述では自然言語や形式仕様記述言語のようなテキスト表現がよく用いられる[138]．厳密なモデルを描ためには，VDM [81]や Z [132]のような形式仕様記述言語が用いられる．形式仕様記述言語は，数学や論理学を使って，問題領域の実体や事象の生起を，正確かつ厳密に記述することができ，モデルの検証も可能であるが[27]，要求技術者と設計技術者の双方に，形式仕様に対する知識が必要となる．

要求変更時には，モデルの場合と同様，仕様の一貫性を保証することも重要な作業である．要求仕様記述言語を使用した場合は，もともと構造的な記述となっているが，第4章で議論するように，テキスト形式で記述された仕様も，構造化することによって，図式へのマッピングが可能となる．

（ｃ）仕様検証技法

検証は，英語の Verification＆Validation の訳である．両者の違いを，Boehm は，正しく製品を作っているか（verification）と，正しい製品を作っているか（validation）の違いであるとしている[12]．言い換えれば，Verificationはソフト

ウェアの開発工程の品質を検査することであり，Validation は製品そのものの品質を検査することであるといえる．しかし，日本語では何れの場合も定訳がない[138]．本論文では併せて検証と呼ぶことにする．

要求仕様検証の目的の1つは，記述された仕様が正しいかどうかを判定することである．しかし，仕様の正しさと一言で言っても，様々な正しさがある．検証の目的にあわせて，種々の検証技法の中から適切な技法を選択する必要がある．

仕様が形式的言語で記述されている場合は，自動推論[92]やモデル検査などによって，その一貫性と完全性を自動的にチェックすることが可能である[61]．SCRツールは，形式的に記述された仕様の形式論的な一貫性と完全性を自動的に検査する[61]．モデル検査は，モデルがシステムに要求される性質を満たすかどうかを自動的に検査する手段で，どのような経路を通ってもデッドロックのような望ましくない事象が発生しないという安全性（safety）と，望ましいことはいつか必ず起きるという活性（liveness）を検査する．ラベル付きの遷移システムとして記述したモデルの性質を，時相論理によるモデル検査手法でチェックする手法が多く提案されており[27]，SPIN のようなモデル検査系も提供されている[64]．

手動による検証を支援するシステムも提案されている．手動で検証を行う場合は，インスペクション[46]やウォークスルーなどの手法がよく用いられる．アニメーションは，システムの振る舞いを視覚的に表現することが可能であるため，対象の動的な変化を時間の経過にしたがって確認することができるが，正確な順序制御や矛盾のチェックを行うには向いていない[23]．また，プロトタイプは，本来，数量化できない品質要求を，擬似実装コードによって確定するための方法であり，デモンストレーションによって，多くのステークホルダの検証を受けたり，実行環境の中に組み込むことによって性能上の検証を行うこともできる．また，手動による検証には，要求を実現する上でのリスクの評価も含まれる．リスクには，人的リスク，コスト的リスク，技術的リスクなどがあり，いずれも，要求を実現する上で欠かせない条件である．環境の変化によってリスクも変化する［104］．

検証法の選択について，本論文では明示的には議論していない．検証作業は，要求工学プロセスの中で繰り返し行われるので，時宜に応じた検証法選択のための知識が必要となる．

1.2.3. 要求の進化と多様化のための技法

インターネットをはじめとする技術革新によって，ソフトウェアを取り巻く世界は，変化と多様性の度合いを急速に増加しつつあり，それは，ソフトウェアに対する要求の不安定性となって現われている[25]．要求が進化する原因は，ニーズの絶え間ない変化であり[124]，多様化の原因は，異なったステークホルダの異なった視点と興味である[41]．ここでは，要求の進化と多様化という問題に対応するための新たな技術をまとめて概括する．

(a) 初期要求

これまでの要求工学プロセスに先立って，システムが組織的要求に合致しているか，システムは何故必要かなどという基本的問題を定義したものを初期要求と呼ぶ．

ビジネスの将来像をモデル化することによって，要求獲得作業の前にビジネスの変化を予測しようとしたのが，ビジネスモデリングである．I.Jacobson らは，ユースケースモデルを使ってビジネスモデルを作成するための手法[75]と，ビジネスモデルからシステムモデルへの変換方法[76]について提案している．H.Eriksson らは，プロセスよりも詳細なビジネス活動に注目し[44]，C.Marshall らは，ビジネス活動を構成している組織や実体に注目したモデリング法を提案している[95]．これらドメインモデル系のビジネスモデルでは，将来モデルを描くために，ビジネス変革の方向性を定義しなければならないが，ビジネス戦略の立案には，企業経営やマーケティングといったソフトウェア工学とは異なった分野の知識を必要とするため，要求技術者には扱いにくい手法であった．要求獲得の前に本質的問題を明確にしようというビジネスモデリングは，初期フェーズ要求工学という考え方の先駆をなしている．

初期フェーズ要求工学という用語を確立したのは，i*法である[149]．そこでは，ビジネス上の意図をゴールとして，モデルに明示的に組み込み，アクターの活動の本質的な問題に焦点を当てようとしている．i*法が扱うビジネス上のゴールは，企業経営レベルのゴールではなく，アクター同士の依存関係という作業レベルのゴールであり，ソフトウェア技術者にも扱い易いものとなっている．i*法では，通常のハードゴールとともに，その実現の評価が難しいソフトゴールも明示的にモデル

化され，ゴール分解によって必要なタスクが導出されるようになっている．しかし，ソフトゴールのような感覚的なゴールには，個人の主観的な判断が色濃く反映される．異なった価値観を取り扱うための手法としてソフトシステム方法論[24]などが有効である．

(b)　多視点モデル

世界の複雑化に伴い，様々な立場のステークホルダが現れ，システムに対する視点も多様化してきた．多様な視点を敢えて統一することなく，それぞれの観点(perspective)からモデル化したものを多視点モデルと呼ぶことにする．ステークホルダの視点の違いは，所属する組織，組織内での責務，個人的な世界観や知識などの違いによって生じ，要求の矛盾や曖昧性を生み出す原因となる．しかし，一方で，異なった視点によって作成されたモデルを比較し，統合することによって，要求の正当性を確保することが可能となる．視点の違いに着目したモデル化や要求獲得法が提案されている．これらの手法は，対象範囲や分類基準は違うものの，本論文の議論と目指す方向は同じである．

Zachman らは，同一の製品であっても，計画者や設計者といったシステム開発における観点の違いによって，異なった目的や手法による記述が必要であることを指摘し，Information System Architecture として体系化した[150, 131]．

A.Finkelstein らは，システム開発チームのメンバーのスキルや知識や経験などのバックグラウンドと，開発チームの中で果たすべき役割を観点として捉え，異なったスタイルでそれを表現するための Viewpoints というフレームワークを提案している[49]．

CORE(COntrolled Requirements Expression and analyst)は，要求工学の基本的な枠組みの中に多視点を取り扱うための手法を組み入れている．はじめに，分析されるべき問題が定義され，次に，その問題に関連する視点が明らかにされ，それぞれの視点で，情報の収集が行われる．収集された情報は，それぞれの視点で図式表現され，制約分析を通して非機能要求が識別され，文書化される[136]．

M.Deutsch は，リアルタイムシステムを作成するための手法のなかで，顧客の視点による要求モデル，使用者の視点による操作モデル，実装者の視点による

実装モデルといった3種類のモデル提案している[39]. 多くの多視点モデルの中でも, 彼のアプローチは洗練されており, メタモデルを介したモデルの自動変換という新たな研究課題が視界に入ってくる[140].

(c) メソッド工学

進化し多様化する要求に対応するためのもう1つの技術は, 既成の手法や方法論をそのまま適用するのではなく, それぞれのプロジェクトの状況に合わせて個別の手法を選択的に適用しようというものである. こうした技術は, 方法論の開発, 評価, 利用などの技術を工学的に研究するメソッド工学として研究されている. 手法の選択と連携は, ここでの大きな課題である. 偶発的な問題への対応という考え方は, 本論文のプロセスパターンによる発見的知識の記述に通じるものがある.

個々の要求工学技術が, どのような問題に適しているかを知ることによって, プロジェクトの特徴や, 抱えている問題に合わせて, 適切な手法を選択することが可能となる. システムやプロジェクトの特徴から要求プロセスの不確定性を判定し, それをもとに要求の確実性を高めるための戦略を設定したり, 手法を選択するための方法が提案されている[99, 34].

MEMAモデルでは, オブジェクト(組織の視点), 情報(コンポーネントの視点), データ(処理の視点), 実装(運用と保守の視点)といった4つのモデルと, ゴールや機能やプロセスなどからなる8個の問題領域特性によって問題領域の特徴を決定し, それらをもとに, 用意されている手法断片との適合性の評価を行っている[114].

G.Vlasblom らは, 情報システムや問題領域の型によるプロジェクトの特徴と, ビジネス環境や既存の情報システムなどの環境要素という2つの側面からなるプロジェクト状況プロファイルをもとに, 最適のモデル状況プロファイルを探し, 開発モデルをプロジェクトのシナリオに合わせて変形するというアプローチを提案している[146].

メソッド工学のもう1つの課題は手法の連携である. 異なった手法, あるいは手法の断片を連携させる場合, これらの手法の間で意味上の一貫性を確保する必要があり[60, 18], 基本的には, それぞれの手法で作成されたソフトウェア成果物,

すなわちモデル同士の間での意味論的整合を保つという形で行われることになる．多くのメソッド工学手法では，今のところ，形式的に表現できる範囲内での意味論を扱うに留まっている．これらの研究は，本論文の要求モデル変更時の一貫性の保証と表裏の関係にある．

多視点でも取り上げた Viewpoint アプローチは，ステークホルダの観点ごとに異なった表現形式で記述された部分要求を，視点統合規則によって連携させ，表現形式の意味論的一貫性を保証するためのフレームワークを提案している[102]．視点統合規則は，それぞれの視点ごとに準備された Viewpointテンプレートの中に，実行されるべきアクションとして記述されている．

Viewpoint アプローチが異なったモデル同士の変換を規則によって統合しようとしているのに対し，H.Delugach は，異なった図式で記述されたモデルを，汎用的な中間モデルに変換し，同一形式の概念図の上で各モデルの一貫性をチェックする手法を提案している[37]．彼の研究では，ERD，DFD，STD などの図式で記述されたそれぞれのモデルを，中間モデルである概念図に変換し，モデル同士の一貫性をチェックしている．

一方，Situational Methods Engineering では，既存の手法から断片を切り出し，それらを組み立てて，プロジェクト固有の手法を構築しようとする．S.Brinkkemper らは，ソフトウェア成果物である製品断片とともに，その操作も工程断片として分割し，それぞれの断片を組み合わせて新たな手法を構築するために，メタモデルのレベルでの統合ガイドラインを示している[19]．

(d) 要求変更

要求変更の発生には，要求獲得作業の失敗と，外部環境の変化という2つの原因がある．前者は修正作業に無駄な追加コストがかかるので好ましくない変更，後者は製品の使用期間の長期化や販売数量の増加につながる好ましい変更であると言われている[116]．いずれにしろ，要求の変更は避けられない現象である．

ステークホルダの多様な視点や，プロジェクトの特性に合わせて選択された複数の異なる手法を使って作成されたモデルや仕様は，最初から意味論と形式論の一貫性を考慮して作成された手法や方法論に比べ，要求変更に対する脆弱性

を抱えている.

要求変更が発生したときには, その変更に如何に容易に対応できるかということが問題となる. ソフトウェア変更管理技術[55]は, 実際の変更箇所を特定するための技術であり[17], 構成管理[45]や要求追跡[78]などの技術と密接な関係を持っている.

要求仕様書への主たる変更作業は, 要求の追加, 削除, 修正である. 追加は, ステークホルダによる要求の変更や, 分析ミスなどによって生じる. 削除は, 開発コストや開発期間の超過によって発生し, 実装段階に入ってから, 定義済みの要求が取り消されるようなこともある[13]. さらに, 要求変更は, 要求モデル同士の矛盾や, 要求仕様の非一貫性という問題を引き起こすこともある. モデル間の矛盾が発見されたら[66], 必要な修正を施さなければならない. モデルの変更は, さらに, 同じモデルの他の部分に, 論理的な矛盾や衝突を引き起こすことがある. 変更による影響を分析し, 必要なら, 修正を実施する. 追跡リンクだけでは影響は十分に分析できないが, 異なった複数の視点を用いることによって影響の拡大を自動的に検出できる場合もある[43].

第 2 章

要求工学プロセス知識の記述

実世界とソフトウェアシステムの狭間にあって，曖昧な要望と厳密な仕様の双方を，その処理対象としている要求工学プロセスでは，プロジェクトの発足時には想定得なかった障害，いわゆる偶発的な問題がしばしば発生している．要求工学プロセス，なかでも，要求獲得の作業中に偶発的な問題が多発する理由は，プロジェクトと，そこで扱う問題の特殊性にあると考えられる[25]．

要求獲得は，システムへの要求提示者であるステークホルダと，システムの開発者である要求分析者の共同作業で成り立っている．しかし，プロジェクトの構成要員でもある両者の間には，持っている専門知識の違いをはじめとして，異なった価値観や相対立する利害関係など，互いの理解や意思の疎通を妨げる様々な障壁が横たわっている．そのうえ，ソフトウェアの意味を決定するはずの要求は，曖昧で，変化しやすく，時には矛盾さえ含んでいる．

要求獲得のための工学的な技法や手法では解決が難しい偶発的な問題に対する解決策の1つが，熟練技術者の経験則やノウハウ，すなわち発見的知識である．たしかに，発見的知識だけでこれらの問題がすべて解決するわけではない．しかし，いくつかの研究はプロジェクトで発生する偶発的問題解決への発見的知識の有効性について言及している[114,146]．

多くの発見的知識は，特定の技術者に固有の知識で，一般に公開されることもなければ，文書として明文化されてもいない．本章では，多くの要求技術者たちにとって，困難な問題を解決するための有効な方法である発見的知識を記述するための知識フレームワークについて議論する．

2.1. 発見的知識の記述方法

発見的知識を記述し，利用するための枠組みとして，知識工学が注目を浴びた時代があったが，問題解決のための知識がシステム内に隠蔽されてしまっていたり，個別の知識や一般常識の取り扱いなどに技術的な困難さがあったため，特別な場合を除いて用いられることが少なくなった．最近では，知識工学に代って，パターンという技術が用いられることが多い[53, 54]．パターンに記述された知識は利用者に公開されるので[29]，個々の技術者が持っている能力や技術力とともに，困難な問題の解決に寄与することが可能となる．

以下に，要求工学プロセスにおける発見的知識を記述するためのプロセスパターンの基本的な概念について述べる．

プロセスパターン

パターンは，ソフトウェア開発に関わる様々な発見的知識を記録し再利用するための有効な方式の1つであり，数多くのパターンが作成されてきた[30]．しかし，これまで提案されてきたパターンの多くは，ソフトウェア設計のための知識を記述するためのものであり，その対象はモデルであり，プログラムであった．設計知識を記述するためのパターン形式は，基本的には，問題と解の組み合わせから構成されている[3,29]．それに対し，プロセス知識を記述するためのパターンが対象にするのは，時間と共に変化する活動であり，体制であり，工程である．それゆえ，プロセスパターンには，問題と解だけでなく，解を特定するために必要なプロジェクトの状況に関する情報も必要となる．

課題

課題とは，解決すべき問題である．要求工学プロセスパターンでは，課題は，プロジェクトが遭遇する偶発的な問題と，それを解決するための発見的知識である解から構成される．様々な原因が複雑に絡み合い変化するプロセスでは，同じ問題に対し，何時も同じ解が適用できるとは限らないし，1つのプロジェクトで成功した解が他のプロジェクトでも成功するとは限らない．

状況

状況とは，個々のプロジェクトが置かれた環境であり，そのプロジェクトを特徴付ける性質である．要求工学プロセスパターンでは，これをプロセスに関

する外部状況と，プロジェクトに固有の内部状況として記述する．外部状況は所与の状況であり，内部状況は自ら変更可能な状況である．同じ問題であっても，プロジェクトの状況が異なれば，適用すべき解も異なってくる．

状況遷移

新たな問題が発生したり，その問題が解決するとプロジェクトの状況が変化する．プロジェクトに起きる連続した状況の変化を状況遷移と呼ぶ．状況遷移には，望ましい遷移と望ましくない遷移がある．望ましくない状況遷移とは，状況が悪化してゆく遷移である．状況遷移は，要求工学プロセスパターンの連鎖として表現できる．これを図式化したものが状況遷移図である．

本章では，要求工学プロセスにおける発見的知識を記述するためのプロセスパターンの形式と，その体系的表現である状況遷移図を提案する．

2.2. 要求獲得プロセスの構造

はじめに，実際の要求獲得事例の作業構造を解析し，それをもとに，プロセスパターンの形式を定義する．ここで，構造解析の対象とする事例は，化学薬品メーカーの営業支援システム開発における要求獲得作業である(図-2.1)．この事例をもとに，多くのプロジェクトに共通する要求獲得作業の構造を抽出する．解析にあたって，本アプリケーションに固有の要件は極力排除してある．また，技術的な知識に焦点を当てており，予算や期間といったプロジェクト管理に関する知識については触れていない．図-2.2に作業シナリオを示す．このシナリオは，6つ基本ステップから構成されている．

構造解析によって，このシナリオから明らかになった構造は次の2つである．

ステージ構造

1つ目の解析は，共通目標による作業ステップの統合化である．同じ目標を共有するステップ同士によるグループをステージと呼ぶことにする．プロジェクトには，正常なプロセスのためのステージと，問題解決のためのステージがある．この事例は，次の4つのステージで構成されている．

ステージ1　　要求獲得作業－講義

ステップ1:　講義形式による，領域専門家からの情報収集作業

ステップ2:　欠如した利用者ニーズの発見と代替案の提示

ステージ2　　要求獲得作業－インタビュー

ステップ3:　インタビュー形式による，利用者への情報収集作業

ステップ4:　多すぎる情報の整理

ステージ3　　要求分析作業－矛盾

ステップ5:　抽出された要求の矛盾の発見と解消

ステージ4　　要求分析作業－確認

ステップ6:　ステークホルダとの最終的な要求の確認と承認

活動サイクル構造

2つ目の解析は，ステップの分解による構造化である(図-2.3)．それぞれのステップ内の活動の共通する構造を活動サイクル構造と呼ぶことにする．本事例で検出された活動サイクルは，以下の通りである．

状況判断：　プロジェクトは，達成すべき目標，あるいは，解決すべき問題を持っている．これをプロジェクトの課題とする．課題を克服するために，プロジェクトは，自身の能力や制約条件を把握する．

目標設定：　プロジェクトは，課題や問題を解決するために，具体的な目標を設定し，その目標を実現するための戦略を立案する．

手法選択：　次に，プロジェクトは，戦略に沿って，自身の状況に適した手法を選択する．選択された手法は，問題に対する解である．

作業実行：　プロジェクトは，選択された手法を使って課題克服のための作業を実行する．しかし，その作業が必ずしも成功裏に完結するとは限らない．作業実行中に，新たな問題が発生したときには，そちらの問題の解決を先行させなければならないことがある．

結果評価：　作業が終了したら，目標が達成されたかどうかを判断し，達成していれば作業が完了したものとして，次のステージに進み，達成されていないときには，何らかの問題が発生していると考えられる．

某化学薬品メーカーでは，新製品の開発が盛んで，商品の品揃えが急ピッチで増加している．例えば，接着剤ひとつをとっても，接着する素材の種類や，接着面の状態，接着時の気温や湿度などによって数十種類の商品が用意されている．そのため，類似した商品同士の使い分けには専門的な知識が必要で，日々増加する商品に営業が対応できないという状況が発生していた．該社の情報システム部門は，営業部門に十分な情報が提供されていないことが問題の原因であると考え，詳細な商品知識を個々の営業に提供するためのシステムの開発プロジェクトを立ち上げた．
情報システム部門の担当者がプロジェクトのリーダーとなり，詳細な商品知識を持っている研究部門の研究者が1名，プロジェクトのメンバーとして参加することになった．また，技術的指導を依頼されたプロバイダーの要求技術者が分析者として参加した．必要な情報は研究部門のメンバーから入手し，それをもとに要求を定義し，最後に，営業部門との確認作業を行うというプロジェクトの作業計画が設定された．

図－2.1　プロジェクトの概要

ステップ1　講義形式による問題領域知識の獲得
（1.1）研究部門のメンバーを含むプロジェクトが開始された．
（1.2）プロジェクトメンバーは，営業支援情報を収集するために，
（1.3）研究部門のメンバーから，講義形式で情報を入手することにし，
（1.4）情報の収集作業を開始した．
ステップ2　利用者ニーズ欠如の発見と代替案の提示
（2.1）要求分析者は，研究部門からのメンバーが，営業部門のニーズを十分に理解していないことに気が付いた．
（2.2）しかし，直ちに作業方針を変更することはせずに，要求獲得作業を継続するため，
（2.3）営業部門のメンバーを，プロジェクトに追加するように要請した．
（2.4）しかし，この提案は営業部門から拒絶された．
ステップ3　インタビューによる要求の再調査と情報収集
（3.1）プロジェクト内での情報収集が不可能となったので，
（3.2）プロジェクトの外で情報を収集するため，
（3.3）営業担当者へのインタビューを提案し，許可されたので，
（3.4）プロジェクトメンバーは，インタビューを実施した．
（3.5）その結果，大量の情報を収集することができた．
ステップ4　入手された大量の情報の整理と洗練
（4.1）膨大な未整理情報から，要求を抽出することは困難と判断し，
（4.2）情報の内容を理解するために，
（4.3）営業活動の進展に合わせて，情報を分類することにし，
（4.4）情報の整理を行った．
（4.5）その結果，情報を整理することができた．
ステップ5　要求の矛盾の発見と解決
（5.1）収集した情報が整理されたので，
（5.2）要求を定義するために，
（5.3）収集された情報の中から要求を抽出し，
（5.4）それらの要求を評価した．
（5.5）その結果，営業部門内の役割によって要求に矛盾のあることが判明した．
ステップ6　要求仕様の確認と承認
（6.1）矛盾した要求が発見されたので，
（6.2）両者の合意点を見つけるために，
（6.3）複数の情報検索法という妥協案を作成し，
（6.4）両者に提示した．
（6.5）その結果，双方の合意を得ることができた．

図－2.2　構造化シナリオ

(1.1) 状況判断: プロジェクトの状況判断(要求獲得作業開始)
(1.2) 目標設定: 課題実現の目標設定(領域専門家からの情報入手)
(1.3) 手法選択: 課題実現のための手法選択(講義)
(1.4) 作業実行: 手法適用
(2.1) 状況判断: 異常検知と問題判定(利用者ニーズの欠如)
(2.2) 目標設定: 問題解決の戦略立案(作業形態継続)
(2.3) 手法選択: 問題解決のための手法選択(要員追加)
(2.5) 結果評価: 問題が解決したかどうかの検証(手法選択失敗)
(3.1) 状況判断: 異常検知と問題判定(要員追加不能)
(3.2) 目標設定: 問題解決の戦略立案(外部資源の利用)
(3.3) 手法選択: 問題解決のための手法選択(インタビュー)
(3.4) 作業実行: 手法適用
(3.5) 結果評価: 問題が解決したかどうかの検証(手法実行)
(4.1) 状況判断: 異常検知と問題判定(膨大な情報)
(4.2) 目標設定: 問題解決の戦略立案(情報整理)
(4.3) 手法選択: 問題解決のための手法選択(分類)
(4.4) 作業実行: 手法適用
(4.5) 作業結果: 問題が解決したかどうかの検証(分類情報)
(5.1) 状況判断: プロジェクトの状況判断(情報収集の完了)
(5.2) 目標設定: 課題実現の目標設定(要求定義)
(5.3) 手法選択: 課題実現のための手法選択(要求抽出)
(5.4) 作業実行: 方策実施
(5.5) 結果評価: 問題が解決したかどうかの検証(矛盾発見)
(6.1) 状況判断: 異常検知と問題判定(矛盾した要求)
(6.2) 目標設定: 問題解決の戦略立案(妥協点発見)
(6.3) 手法選択: 問題解決のための方策選択(妥協案)
(6.4) 作業実行: 方策実施
(6.5) 結果評価: 問題が解決したかどうかの検証(解消)

図-2.3 活動サイクルの構造

2.3. プロセスパターンの形式

プロジェクトに問題が発生すると，プロジェクトの状況が変化する．これは，問題を抱えた状況である．プロジェクトは，この状況から抜け出すために，さまざまなプロセス知識を使って問題を解決するための活動を行う．この問題解決活動の構造をもとに，要求工学プロセスパターンの記述形式を定義する．

要求工学プロセスでは，たとえ同じ問題であっても，プロジェクトの状況によって，その解決方法が異なる場合がある．すなわち，状況と課題は独立した事象として扱われなければならない．例を示す．ここで，:の前が状況，後ろが課題を

示している.

(イ) 要求を決定のできないワークショップ:領域専門家の不在

(ロ) 要求を決定のできないワークショップ:ステークホルダの不在

(ハ) 情報が収集できないワークショップ:ニーズを知らない要求者

(ニ) 訂正が頻発するワークショップ:ニーズを知らない要求者

(イ)と(ロ)は, 同一状況における異なった問題であり, (ハ)と(ニ)は, 異なった状況における同一問題の例である.

この例で分かるように, 同一の状況, 同一の問題であっても, 課題が異なれば, 異なった解が必要となり, 同じ課題であっても, 状況が異なれば, 異なった解が必要となる. このように, 問題解決のためのプロセス知識には, プロジェクトのおかれた状況を認識するための知識と, プロジェクト内部における問題を解決するための知識がある. この2種類の知識を記述するために, プロセスパターンを, 2つのセクションに分ける.

状況セクション: プロジェクトの状況を記述するためのセクション

課題セクション: 課題を克服するための知識を記述するためのセクション

状況セクションには要求獲得プロセスにおけるプロジェクトの位置やプロジェクト自身の特徴を記述し, 課題セクションにはプロジェクトが抱える問題とその問題を解決するための方策を記述する.

2.3.1. 状況セクションの形式

プロジェクトの状況を記述するための状況セクションの形式を定義する. 以下は, 状況セクションに記述されるべきプロジェクトの特徴である.

- 設定された課題を克服してゆくために, プロジェクトが遂行する作業の単位をステージとする. 各ステージには, 固有のステージ名と達成すべき目標が設定される. 目標が達成されると, プロジェクトは次のステージに移行する.
- ステージの中で問題が発生すると, プロジェクトの状況が変化する. 1つの問題が解決されても, 新たな問題が発生したり, 問題解決活動中に新たな問題が発生することによって, 更なる状況変化が発生する. 連続する状況の

変化を譲許遷移と呼ぶ．状況遷移のなかで，それぞれの状況は固有の名前を持つ．

- プロジェクトが置かれている環境下での制約や，プロジェクトそのものが持っている特性によって，プロジェクトの状況は異なる．前者を外部状況，後者を内部状況とする．
- 問題を抱えたプロジェクトは，問題解決のための活動を行う．問題が解決された状況を目標状況とする．
- プロジェクトが，ある状況に変化したと判断するための条件を事前条件，その状況から抜け出したと判断するための条件を事後条件とする．
- 状況変化には問題が解決に向かう良い変化と，更に悪化する悪い変化がある．悪い変化が続くと，プロジェクトは破綻するかもしれない．そこで，プロジェクトは，問題解決のために，新たな目標を設定し，新たなステージに移行する．問題が解決すると，もとのステージに戻る．

状況に関するこれらの特徴をもとに，要求獲得プロセスを記述するための状況セクションの構造を定義する(図-2.4)．状況セクションは，状況および状況タイプと条件タイプから構成される．状況タイプは，プロジェクトの状況の説明であり，外部状況と内部状況および目標状況がある．条件タイプは，その状況が成立するための条件で，事前条件と事後条件を含む．図-2.5に状況セクションの具体的な記述形式を示す．

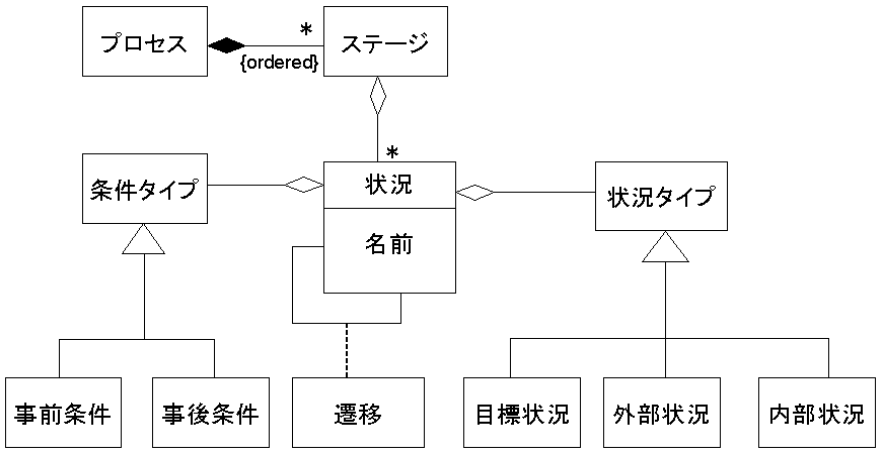

図-2.4　状況セクションの概念構造

状況名: 状況を特定するための名前
外部状況: プロジェクトが置かれているステージとステージ内での位置
内部状況: プロジェクト自身の人的, 技術的な特性
目標状況: 問題が解決した時の状態
事前条件: プロジェクトがこの状況に陥ったと判断するための条件
事後条件: プロジェクトがこの状況から抜け出したとする判断するための条件

図-2.5　状況セクションの記述形式

2.3.2. 課題セクションの形式

プロジェクトが抱えた問題とその解を記述するための課題セクションの形式を定義する. 課題セクションに記述されるべきプロジェクトの特徴を以下に示す.

要求工学プロセスにおけるプロジェクトの問題解決活動は, 状況判断－目標設定－手法選択－作業実行－結果評価という5つの活動から構成されているものとする. この構造を, 課題セクションの基本構造とする.

プロジェクトの活動は, 達成すべき目標や解決すべき問題で始まる. それを課題名とする.

目標や問題には, それが発生する, 何らかの理由が存在する. この理由が, 課題の背景である. 背景は通常のパターン形式[29]における“文脈”に相当する. しかし, 同じ背景から生じた現象が, どのプロジェクトにとっても同じ課題となるわけではない. それぞれのプロジェクトには, 現象から特定の課題が生じるための独自の原因がある.

プロジェクトは, その特性や内外で発生する事象の変化から, プロジェクトが抱える問題を特定し, それを解決するための目標を設定しなければならない.

しかし，問題を解決するための目標は，1つとは限らない．1つの問題に対する複数の目標同士は，代替関係にある．

プロジェクトは，目標を達成するために，戦略を策定し，その戦略を実行するための方法を選択する．選択された方法は，問題に対する解である．

1つの目標に複数の解が存在することもあれば，1つの解が複数の異なった目標を達成することもある．あるいは，1つの目標を達成するための解のセットが存在するかもしれない．目標達成のために選択された解には，選択されるがゆえの理由が存在する．解の選択幅を制約する目標は，通常のパターン形式における“フォース”に相当する．

解には，工学的手法によって実現される技術的な解と，プロジェクト管理手法によって実現される管理的な解がある．

事前に設定した目標と解の実行結果とを比較し，問題が解決されたかどうか，あるいは状況が良くなったかどうかを判断し，その結果を結果評価とする．

選択された解を実行したからといって，何時も期待する結果が得られるとは限らない．プロジェクトの他の要因が，活動の結果に予期せぬ影響を与える場合がある．目標は期待される状態であるが，結果は目標に対して取り得る可能なすべての状態の中の1つである．

目標は，問題が解決した状態であり，解は目標を達成するための手法である．この階層構造は，問題から目標，目標から解，解から結果というゴール分解の木構造を形成している．問題が定義できたら，木構造の下位の項目を順に選択してゆくことによって，問題解決の方法と，期待される結果を定義できる．

問題解決におけるこの特徴をもとに，要求工学のプロセスを記述するための課題セクションの構造を定義する．図-2.6に課題セクションの概念構造を，図-2.7に課題セクションの記述形式を示す．

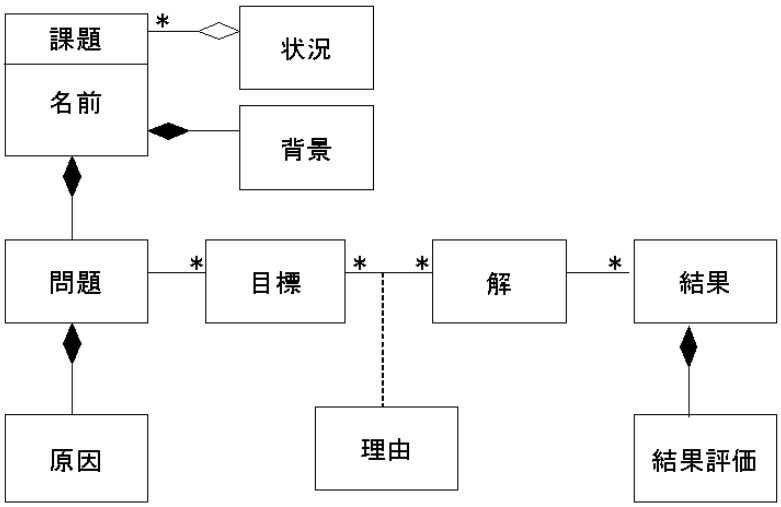

図-2.6　課題セクションの概念構造

課題名: 課題を特定するための名前
背景: プロジェクトに問題が発生した背景としての文脈
問題: プロジェクトが解決しなければならない問題
原因: 現象が問題となる理由や原因, フォース
目標: 問題解決のためのゴール
解: 問題を解決し, 目標を達成するための手段, 手法
理由: 問題と解の間の論理的な関係及び解の制約
結果: 解の適用によって新たに生じた状況
結果評価: 結果に対する相対的価値判断

図-2.7　課題セクションの記述形式

2.4. 状況遷移図の形式

プロジェクトは問題の発生と解決を繰り返しながら推移してゆく. これを状況遷移と呼ぶ. 要求技術者の役割は, 遭遇した問題を解決するために, 自身のプロジェクトが, 遷移する状況の中のどの位置にいるのかを判断し, プロセスパターンを使って適切な解決策を選択し, 実行することである. しかし, 大量のプロセスパターンの中から, 適切なパターンを選び出すことは困難である. この作業を支援するのが状況遷移図である.

状況遷移図は, プロジェクトの状況遷移を, 有効グラフによって表現したものである. 状況遷移図には, 要求工学プロセスにおける作業ステージと, それぞれのステージのなかで発生する可能性のあるすべての状況変化が描かれている. 要求技術者は, 状況遷移図を使うことによって, 自身のプロジェクトの置かれた状況をプロセスの流れのなかで捉えることができ, より適切なプロセスパターンの選択が可能になる.

プロセスパターンを適用すると, プロジェクトの状況はさらに変化してゆく. この変化には, 状況の好転や悪化という相対的な価値判断が伴う. 状況が好転し, 問題が解決すれば, プロジェクトはもとのステージに戻ることができるが, 問題が解決しなかったり, 新たな問題が発生したりして状況が悪化すると, そのステージの中でプロジェクトそのものが破綻終了してしまうかもしれない. 複合的な問題の場合は, 1つの問題が解決すると, それまで隠れていた別の問題が表面化する場合もある.

状況遷移図の構成要素と, その表記法は以下の通りである.

ステージ: ステージは状況遷移図の中の大きな四角形で表し, ステージ間の移動は矢印付き線分で表す. ステージはステージ名で識別される.

状況: 状況はステージの中の四角形で表す. また, 状況の変化は, 2つの状況の間の矢印付き線分で表す. 状況は, 次の3つの領域から構成される.

- **名前**: 状況の名前. すなわち, 状況セクションの状況名.
- **状況**: プロジェクトの状況. すなわち, 状況セクションの内容.
- **課題**: 解決すべき課題とその解. すなわち, 課題セクションの内容. このセ

クションで，目標と解は，その組み合わせを表すために，状況の中の小さな四角形で表す．1つの目標に複数の解が存在する場合，OR関係は並列構造で，AND関係は直列構造で解を示す．

結果評価：結果評価は，状況変化に対する，要求技術者による価値判断である．状況変化の価値は，状況同士の相対価値によって定義され，良い変化は線上の(＋)，悪い変化は(－)記号で表現する．

開始／終了：ステージの移行や状況遷移の開始と終了を表す．

図-2.8に状況遷移図の概念構造をクラス図で示す．

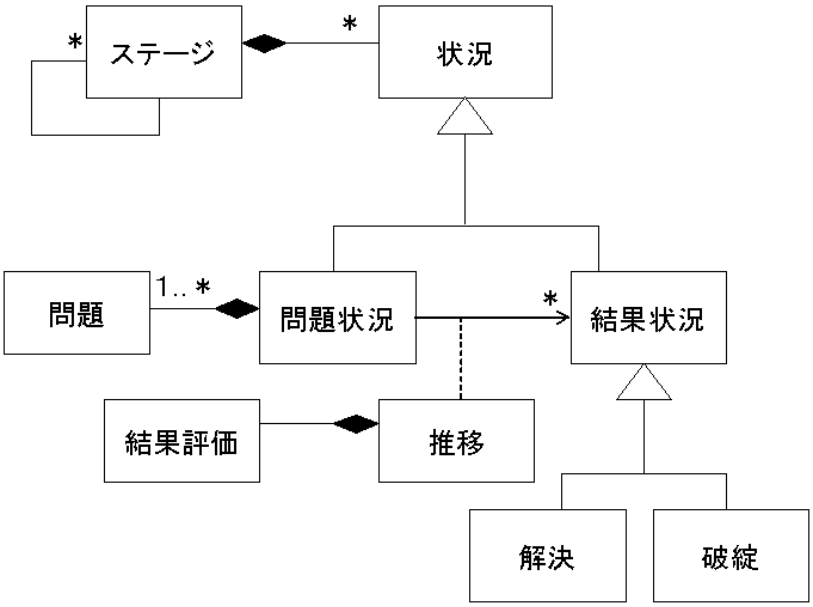

図-2.8　状況遷移図の概念構造

問題の種類によって，必要とされる情報が異なる場合がある．状況遷移図には，表示する情報の違いによって，次の3種類の表現形式を用意する．

完全型：プロセスパターンの状況セクションと課題セクションの内容を，すべて記述する．

部分型：状況名と課題名だけからなる中間的な状況遷移図である．

簡易型：状況名だけからなる状況遷移図であり，状況遷移の全体像を描くのに使用される．

状況遷移図のそれぞれの表現形式の違いを，図-2.9に示す．

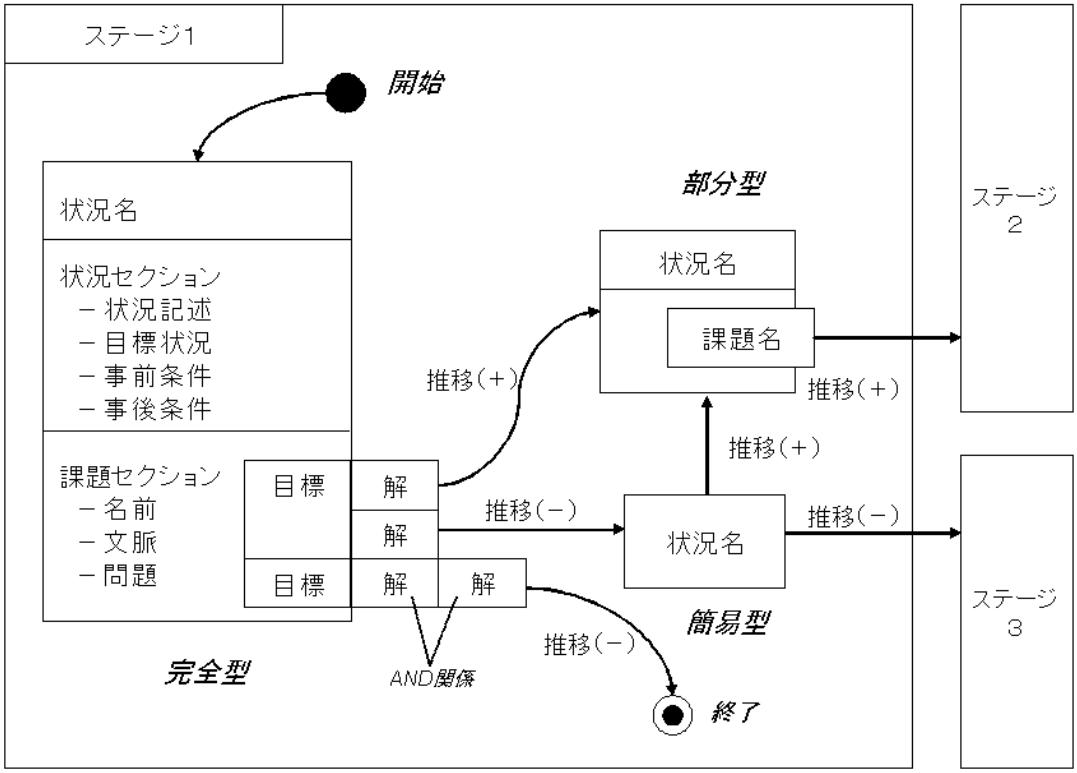

図-2.9 状況遷移図

2.5. 要求工学プロセスパターンの記述例

本節では，上記構造解析で使用した営業支援システムの事例を使って，具体的な要求工学プロセスパターンと状況遷移図の記述例を示す．

2.5.1. 状況セクションと簡易型状況遷移図

各状況セクションの各項目を簡単に記述した例を，図-2.10に示す．これは，「情報が収集できないワークショップ」という状況から始まる一連の状況変化の例である．ここでは，状況セクションは4つのステージから構成されている．ステージの1から3では状況に変化が起きているが，ステージ4では状況変化は発生していな

い.

また, これらの状況セクションを, 簡易型の状況遷移図によって表したものを図-2.11に示す. なお, この状況遷移図の中で, 破線で描かれた状況と状況遷移は, 本構造解析のシナリオにはなかったパスである. 本来は実線で表されるべきものであるが, ここでは, 便宜的に破線で表してある.

実際のプロジェクトの状況は, いくつかの異なった状況が組み合わされた複合型の状況であることが多い. したがって, 状況遷移も, 複数の状況遷移の複合型となる. こうした複合型問題への対応の仕方には, 2つのアプローチがある. 1つは, 複合状況を分解し, 単独状況の集合として扱う方法である. プロジェクトは, それぞれの単独状況の遷移に順番に対応してゆけばよい. しかし, 複合型の状況が相乗効果を発揮し, 新たな問題を生じる場合がある. 複合型状況は, それ自身を1つの状況とみなして対応することが必要である. これは, 人間の病気と治療の関係に似ている. 西洋医学では, 病気の原因を個別に治療してゆくが, 東洋医学では, 複合症状そのものに対応しようとする. 問題の特性に合わせて対応法を変えてゆくというアプローチは, ソフトシステム・アプローチ[24]とも呼ばれ, プロジェクトの問題解決にも役立つであろう. どちらの方法を採用するかに関する知識もまた発見的知識である.

ステージ1：専門化からの情報収集

状況セクション	
状況名	情報が収集できないワークショップ
外部状況	専門家から講義形式で情報収集をしている.
内部状況	必要な情報の収集が進まない.
目標状況	ユーザニーズが入手されている.
事前条件	ユーザニーズの情報入手先が無い.
事後条件	ユーザニーズの情報入手先がある.

ステージ2：インタビューによる情報収集

状況セクション	
状況名	情報の整理ができない分析者
外部状況	利用者にインタビューを行い多量の情報が収集された.
内部状況	多量の未整理情報がある.
目標状況	収集情報が整理されている.
事前条件	多量の未整理情報がある.
事後条件	未整理の情報が無い.

ステージ3：要求の評価

状況セクション	
状況名	矛盾した要求を抱えた分析者
外部状況	収集された状況から要求を抽出している.
内部状況	矛盾した要求が含まれている.
目標状況	矛盾が解消されている.
事前条件	要求同士の間に矛盾がある.
事後条件	要求同士の間に矛盾が無い.

ステージ4：要求の確認

図-2.10　状況セクションの例

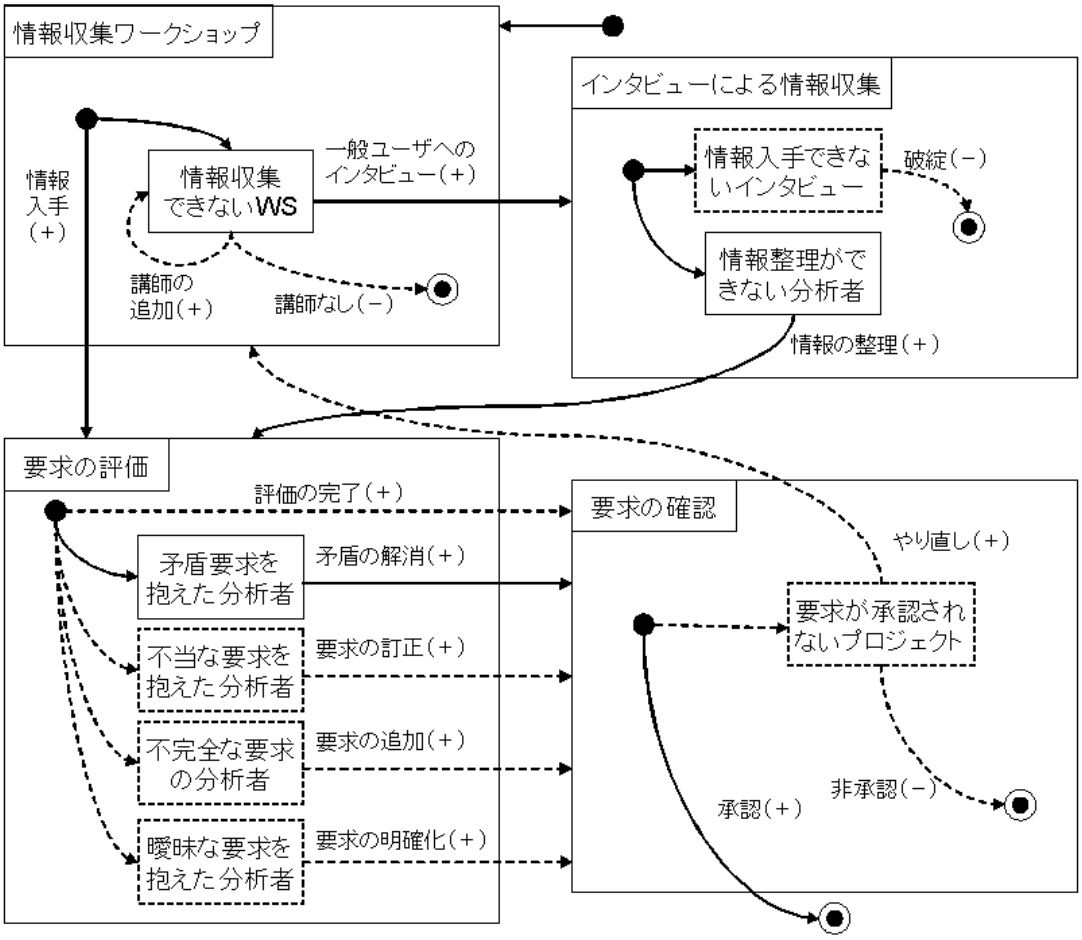

図-2.11　簡易型状況遷移図の例

2.5.2. 課題セクションと完全型状況遷移図

図-2.12に示した例は，課題セクション「情報が収集できないワークショップ」という状況における「ユーザニーズを知らない要求者」という課題である．ここでは，「ユーザニーズを知らない要求者」という課題に対し，「新たな情報源を求める」と「既存の情報を利用する」という2つの目標と，それらの目標に対する3つの解のみを示しているが，当然，それ以外の目標も存在するし，それぞれの目標に対しても，ここに挙げた以外の解が存在する．

状況：情報が収集できないワークショップ

課題セクション			
課題名			ユーザニーズを知らない要求者
背景			ワークショップで情報収集しているが，作業が計画通り進まない．
問題			要求者がユーザ要求を知らず，質問に答えられない．
原因			要求者は研究所からのメンバーで，営業活動を知らない．
A	目標		新たな情報資源を求める．
	1	解	営業部門からのメンバーを追加する．
		理由	現在の作業形態を継続する．
		結果a	（＋） 営業要員がメンバーに参加して，必要な情報が得られる．
		結果b	（－） 営業要員が参加するが，必要な情報が得られない．
		結果c	（－） 営業部門から断られる．
	2	解	ユーザにインタビューする
		理由	プロジェクト内に情報資源が無い．
		結果a	（＋） インタビューを行い，必要な情報が得られる．
		結果b	（－） 未整理の膨大な情報が得られる．
		結果c	（－） インタビューが断られる．
B	目標		既存の情報を利用する．
	1	解	獲得済みの情報から要求を類推する．
		理由	他に情報源が無い．
		結果a	（＋） 要求が抽出された．
		結果b	（－） 要求を抽出するだけの情報が無い．

図－2.12　課題セクションの例

図-2.13には，その前後の状況を含めた完全型の状況遷移図を示す．

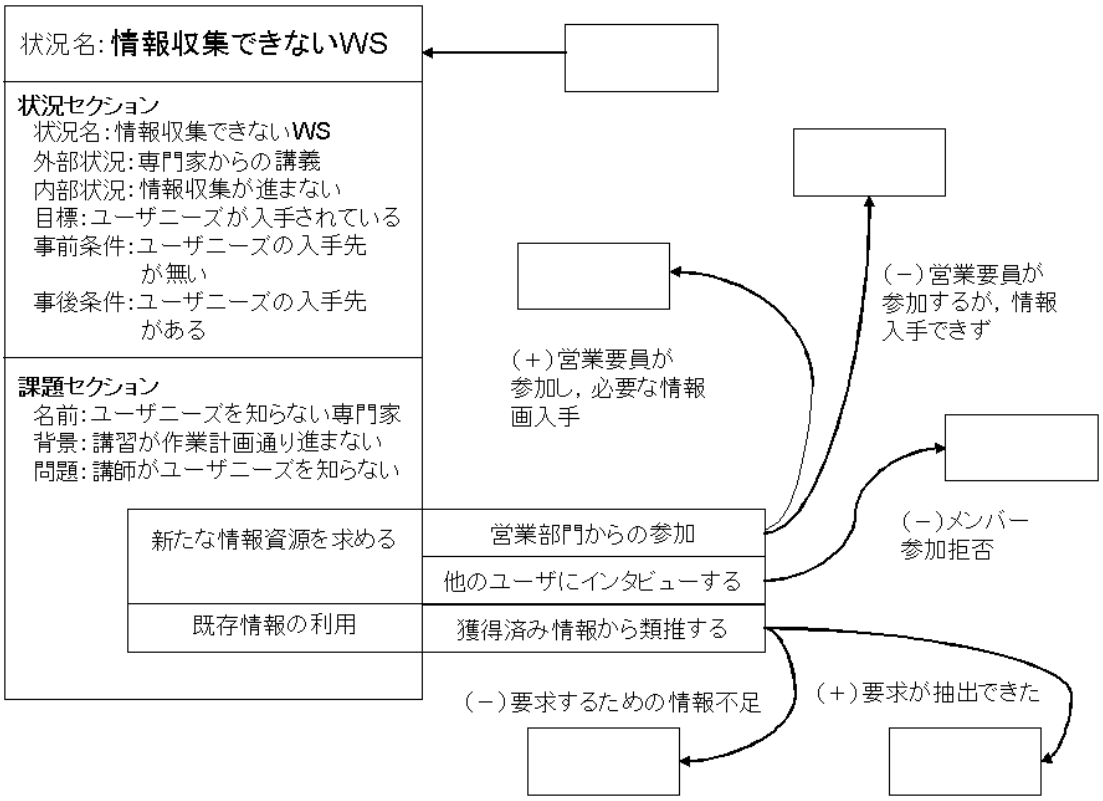

図-2.13　完全型状況遷移図の例

2.6. 事例研究：航空運輸システム

要求工学プロセスパターンを適用したプロジェクトの事例について述べる．ただし，当該プロジェクトの活動は極めて多岐に渡っており，ここで述べる事例は，その中の要求獲得作業における情報の整理という問題だけに焦点を当てている．

（a）プロジェクトの概要

航空運輸業界は，スカイフリーに代表される規制緩和，中国を中心とした物流の変化，各国通関業務の電子化，アライアンスを締結したグループ企業による顧

客の囲い込み，動的な価格設定による利益の管理，顧客への新規サービスの提供など，大きな変化が始まりつつある．こうした時代の潮流に乗り遅れないためには，航空運輸会社は基幹システムを改造する必要に迫られており，数年計画でシステムの大規模改修を実施することになった．

システム改修プロジェクトの構成と役割は次のように決められた．

社内の承認オペレーション：　航空会社の情報企画部門のメンバー　3名

システム改修項目の提案：　コンピュータメーカーの航空運輸担当SE　3名

（b）要求工学プロセス

ステップ1：　プロジェクトの開始

プロジェクトの開始に当たって，情報企画部門からコンピュータメーカーに，次の2つの条件が提示された．

改修項目に抜けが無いように，業務全般にわたって調査すること

社内の承認を得るために，新たな魅力的機能を提案すること

ステップ2：　情報収集と整理

コンピュータメーカーの航空運輸担当SEらは，情報の見落としを防ぐために，できるだけ多くの情報源を探し，以下のような情報の収集に当たった．

現行システムの機能の調査

過去3年に渡る業界紙による業界動向の調査

運輸コンサルタントを使ったロジスティックス・サービスの調査

空港現場での，作業担当者へのヒアリング調査

こうした調査の結果，約2000件に上る情報や要求が収集された．この情報をもとに，コンピュータメーカーから航空会社の情報企画部に，改修項目の提示と新たな機能の提案を行なうことになった．メーカーの担当SEたちは，収集された膨大な情報の整理を始めたが，その量のあまりの多さに，整理作業が進まず，時間だけが徒に過ぎていった．メンバーたちは，その原因を，絶えず増え続ける情報と，クライアントの矛盾した意見のせいにした．

ステップ３：　プロセスパターンの適用：情報の整理

プロジェクトは，大量情報の整理という問題を解決するために要求工学プロセスパターンの適用を図ることにした．

パターンの選択にあたって，はじめに，情報整理作業を行っているメーカーの担当SE3名による状況の判断を行った．状況遷移図を使って，自分たちの置かれている状況を選択することにし，「情報を整理できない分析者」という状況が同定された．次に，その状況の中から「大量情報の整理」というパターンが選択された．「増加し続ける情報」や「矛盾した情報」というパターンも選択されたが，情報量との比較によってそれらの重要性は低いとみなされた．

選択されたパターンには，解として次の4つの分類法が提示されていた．

業務の種類の違いによって情報を分類する
関係者のタイプの違いによって情報を分類する
目的の違いによって情報を分類する
時間の経過にしたがって情報を分類する

これを参考に，現行の組織体制をもとにした業務と，個々の要求の上位目的という2つの軸を使って情報を分類することにし，それぞれ同じグループ内で，類似した情報を統合した．次に，選択された2つの軸によって分類した項目をさらに統合し，最終的に，30個の要求項目に整理することができた．その殆どは，航空運輸制度の変更に対する対応策であった．この分類統合作業は1人のSEによって行われ，作業に要した期間は2週間であった．

ステップ４：　新機能の創出

航空会社からのもう1つの要請である新機能の創出については，発想支援法を用いていくつかのアイデア創出を行った．しかし，それは要求プロセスパターンとは，異なった作業である．

（ｃ）考察

「大量情報の整理」という問題は，要求工学プロセスにおいてしばしば遭遇する典型的な問題の1つである．この問題を解決するための基本的な戦略は，情報分類のために可能な分類軸を設定し，それを順次試して行くという方法である．しかし，その戦略を実行するには膨大な時間が必要となる．問題は，情報の分類統合のための手法と，有効な分類軸を如何に早く発見できるかということにある．

本事例では，プロセスパターンを使って情報の分類軸の候補を提供することによって，3人のSEが1ヶ月かかっても整理できなかった情報の整理を，1人のSEが2週間でできるという結果を得ることができた．もちろん，問題領域についてのある程度の知識をもった担当SEにとって，提示された分類軸の中から，有効な軸を発見することは比較的容易であり，一旦，分類軸が決まれば，それに沿って情報を整理することは，さして難しい作業ではなかった．この事例から分かったことは，抽象化作業に慣れていない一般のSEにとって難しいのは，情報の理解ではなく，それを整理するための分類軸の発見であるということであった．もしプロジェクトの状況を無視して，問題だけを特定しようとしていたら，「増加し続ける情報」や「矛盾した情報」という異なった問題が選択され，分類軸の提示という結論には至らなかったかもれない．状況セクションを独立させ，状況認識を優先させたプロセスパターンの有効性が示された例である．しかし，状況の特定は困難な作業である．プロジェクトメンバーによってしばしばその認識が異なる場合がある．特に，地位の高い技術者は，自分にとって都合の悪い状況は認めたがらない傾向がある．客観的な状況判断にもとづいたメンバー同士の合意の形成と問題の事前検討が問題解決の重要な鍵となる．この例では，状況遷移図の使用によって，「情報の分類が出来ない分析者」という状況を特定することができ，情報の分類に有効な分類軸を提示が可能となった．

（ｄ）その他の知見

本事例で学んだことの1つは，ちょっとしたアイデアが問題解決に大いに有効な場合があるということである．発見的知識とは，そういうものであるかもしれない．もちろん，そうしたアイデアを受け入れるためには，受け入れる側に，思考の柔軟性が必要であることは言うまでも無い．

ステップ4における問題は，いわゆる営業時のオーバーコミットから生まれたもので，本来の要求工学の対象範囲外である．こうした人間の創造性に関わる問題に対しては，要求工学技術以外の技術が必要となり，要求工学プロセスパターンとは異なった発想支援パターンのようなものが必要となるだろう．

2.7. 関連研究

ソフトウェア開発における個別の問題を解決するための発見的知識をカタログ化するための有効な記述方法であるパターンが提案されて以来，ソフトウェア工学の様々な局面で，種々のパターンが提案されてきた．方法論がシステム開発プロジェクトの成功シナリオを提供するのに対し，経験に基づいた知識を記述したパターンは，方法論では提供されない，偶発的な問題に対する解を提供することを目的としている．しかし，SCRUMのように，方法論そのものをパターンで記述している例もある[126]．

パターンの基本的なアイデアは，建築家であるC.Alexander によって提案されたものである[2]．Alexander は，問題と文脈と解というパターンの主要な構成要素を提示することによって，その後のパターン発展の礎を築いた[3]． Alexander は建築家だったので，彼が提示したパターンも建築設計に関するものである．その後のパターン研究の主流が設計型パターンであった遠因が，ここにあるのかも知れない．

ソフトウェアパターンにおける最初の大きな仕事は，GoF によるデザインパターンである[54]．このデザインパターンは，著者らの発見的知識が集大成されたもので，オブジェクト指向プログラム設計上の有益な23個のパターンから構成されている．その基本的なアイデアは，継承よりも合成や委譲を優先させて再利用可能なプログラム構造を作ろうとするもので，そのアイデアを，14種類の構成要素によって記述している．GoF 本の成功に刺激され，その後，アナリシスパターン[53]やアーキテクチャパターン[21]など，様々なソフトウェア設計に関するパターン集が提案された．それらの記述項目には，いくつかの共通要素が見られる．なかでも，特に，問題と文脈と解は，対象問題に拘わらない本質的な要素である． Coplien は，これらの形式を発展させ，ソフトウェアパターンを記述するための一般的なパターン形式を提案した[29]．この Coplien 形式と呼ばれる形式は，パターン名，問

題，文脈，フォース，解，根拠，結果という7つのパターンセクションから構成されており，本研究でも，この形式を念頭においている．Fowler のアナリシスパターンは，異なった問題領域に共通する実体構造を，抽象度の高い概念モデルとして表現したものである．問題領域の基本構造である概念モデルをパターン化し，そこから共通の要求を導出しようというものであり，その意味で，対象領域を理解するための発見的知識の1種と位置づけることができる．Jackson の問題フレーム[73]も，対象を解決すべき問題と捉えるという違いを除けば，同様の文脈で捉えることができる．パターンが建築業界よりも情報システム業界で受け入れられたのは，建築パターンが意匠設計への適用だったのに対し，情報システムパターンがソフトウェアの構造設計への適用であったからかもしれない．感性が大きな比重を占める意匠設計では，他人のパターンを受け入れることは難しい．

パターンは，その後，個別の問題解決のための知識を記述するための代表的な方式として定着し，優れたソフトウェアを作るためのソフトウェアパターンとともに，開発プロセスを支援するためのプロセスパターンも提案されるようになった．要求工学プロセスのためのパターン形式に関しては，Hagge らが，目標，文脈，問題，解，アプリケーション領域，フォース，構造，結果，事例と経験という9つの要素を提案しており[59]，本研究との類似性が高い．いわゆる設計パターンなどにはない目標や結果という概念をプロセスパターンに導入しているところに彼らの独自性があり，本論文でもその重要性が再認識されている．しかし，パターン化の対象が小規模プロジェクトに限定されているため，状況変化という概念が欠如している．例えば，“Provide Specification Guideline by Reverse Envisioning” というパターンでは，プロジェクトの状況を特定するための文脈の記述は，「分散プロジェクトが発足した後の要求獲得作業」というように，手順を示しているだけで，プロジェクトの状況の説明にはなっていない．そのため，「異なった視点を統合しなければならない」という問題が，プロジェクトのどのような問題に寄与するのかの具体的なイメージがつかめない．また，内部状況についての記述もないので，「ガイドラインの作成」という解が，どの程度のメンバー構成のプロジェクトなら有効であるかなどを判断することが困難である．プロジェクトが大きくなればなるほど，そのプロジェクトが遭遇する状況は複雑化してゆく．そのため，問題とは独立した状況の定義をしなければ，解の選択は困難となる．しかし，彼らの目的は，プロセスパターンの形式を研究することではなく，要求パターンを収集することにあり，本研究とは目的

が異なっている.

パターンの有効性が示されて以来, PLoP[30]をはじめとする多くのパターン集が出版された. しかし, 大量のパターンの中から目的とするパターンを選択することは至難の業である. その上, それぞれのパターンは他のパターンと密接に結びつくことによって, パターンの構造的な複雑さが増加する. 多くのパターンが提案されたにもかかわらず, 参照されるパターンがGoFの設計パターンに集中していることの理由がそこにある. Alexander は, 複数のパターンをまとめて記述するパターンランゲージ[2]を提案しているが, パターンランゲージは, 1つの建築対象に適用すべき連続したパターンの集合であり, そこから特定のパターンを発見することを目的とはしていない. 一方, R.Johnson らは, 複雑なパターン構造を記述するための方法として有効グラフの使用を提言している[80]. 彼らは描画エディタの使用法を10ステップのフレームワークとし記述し, それらの関係を有効グラフによって示しており, それは本論文の状況遷移図と同じアプローチである. ただし, 描画エディタの使用法は, 彼らも言っているように完結したパターンランゲージの一種であり, 偶発的問題を対象としている本研究とはそこが異なっている.

2.8. 考察

本章では, 要求工学プロセスにおける問題解決のための発見的知識の記述方式について議論し, 状況と問題の分離がプロセス知識の記述にとって有効であることを具体的な事例を通して示した.

熟練技術者が持つ個別の発見的知識は, それを普遍化することによって再利用が可能となる. パターンは, そのような普遍的な知識を記述するのに向いている. 本論文で提案したプロセスパターンにおける状況セクションと課題セクションの分離は, たとえ同一の問題であっても, 状況が異なれば, 異なった解が必要となるような工学プロセス特有の問題構造, すなわち, 状況と課題の非同期構造を素直に表現するのに適している. ステージ, 状況, 問題という階層構造は, 問題の段階的な絞り込みを可能にすることによって, 選択の容易性や記述の明快さという問題に寄与している. さらに, 結果の提示は, 適切と思われる手法を選んだからといって, 必ずしも良い結果が得られるとは限らない工学プロセスでの, 解の選択に

対するリスクを軽減するのにも役立つ.

しかし, 普遍化されただけで未整理のままの知識は, 抽象化された知識に比べ, その量が膨大となることが避けられない. 大量で複雑な情報を整理する方法として, 分割, 抽象化, 構造化などがある[13]. 本論文で提案している状況遷移図は, 構造化による発見的知識の整理であり, 状況の変化に焦点を当てることによって大量のパターンを体系化するための図式である. 状況遷移図を使って, 状況別, 問題別にパターンを整理し, 解析することによって大量のパターンの中から適切なパターンを探すことが容易になる. さらに, 有向グラフが示す状況変化の前後関係は, 問題の予測, 回避の事前処置, コストの見積もりなどに関する基礎情報を提供してくれ, 予期せぬ問題を検討することにも役に立つ. また, 図式は, 異なった立場の人間同士の合意形成に役立つので[138], 多くの経験者との合意によって発見的知識の有効性を判断するのにも効果がある.

ソフトウェア工学の様々な分野で, 大量のパターンが提案されてきたが[30], 一般によく使用されているのは, そのうちの僅かなものに過ぎない. 利用者にとって価値のあるパターンとは, 同様の問題がよく発生し, その解が普遍的な価値を持つものである. デザインパターンやアナリシスパターンが多くの技術者に受け入れられた理由の1つは, パターンの普遍化が図られており, 少ない知識で高い再利用の可能性を秘めているからである. 選択の容易性, 記述の明快さ, 解の普遍性などが再利用を促進する要因であり, パターンの記述にはある程度の熟練性を要することになる.

状況遷移図で見たように, 構造化は多様で曖昧な問題領域を理解する有効な方法の1つである[13]. そうした視点から, 構造主義[88]などとの関連について検討を行うことは今後の課題である.

第 3 章

要求工学技法の選択

要求工学プロセスの中で，要求技術者はいくつかの意思決定を行わなければならない．その中でも，最も困難な作業の1つは，プロジェクトに適した要求工学技法を選択することである．プロジェクトが遭遇する問題の多様性に合わせて様々な技法が提唱されてきたが，これは，プロジェクトに適した要求工学技法を選択するにはどうすれば良いかという新たな問題を提起した．

プロジェクトは，対象問題領域や，システムのタイプ，大きさ，組織の文化といった様々な要因によって特徴付けられる．プロジェクトの特性に合わない要求工学技法が選択されると，そのプロジェクトは障害に陥ってしまうかもしれない[144]．確かに，要求選択の問題に関して，いくつかの研究がある[93, 62]．しかし，論文や教科書，チュートリアル，ツールのマニュアルといった要求工学関係の文献の多くは，特定の技法の優位性について強調はするが，その適用領域や限界について十分に言及することがない．プロジェクトに適した手法を選択する上で重要なのは，要求手法とプロジェクトの特徴を関係付ける汎用的なフレームワークである．

要求工学技法の適用上でのもう一つの課題は，プロジェクトの状況変化にどのように寄与できるかということである．要求工学作業中であろうと，要求仕様が完成した後であろうと，ビジネス環境やステークホルダあるいは適用技術などの変化によって，プロジェクトは重要かつ実質的な変化を経験することになる．プロジェクトは変化していないのに，プロジェクトの性質に対する要求技術者の理解が変化することもある．要求工学プロセスが障害に遭遇してしまってから，プロジェクトの特性に対する最初の判断が間違っていたということに気付くこともある．このような

状況に陥ったプロジェクトは，直面している障害を回避するために，新たな要求工学技法を選択し，適用しなければならない．

優れた技術者は，良い選択をし，選ばれた技法を上手に適用する能力を持っている．しかし，限られた教育や乏しい経験しか持たない未熟な技術者は，手法選択の幅が限定されている．一度，プロジェクトに適さない手法が選択されると，そのプロジェクトは破綻に帰することになってしまうかもしれない．本章では，代表的な要求工学技法の特徴を整理し，プロジェクトの開始時や，プロジェクトの特性が変化したり，障害に遭遇したときなどに，適切な技法を選択できるようにするためのフレームワークを提案する．

3.1. 要求獲得技法の分類

初期要求工学プロセスである要求獲得技法に焦点を当てる．

要求工学技法の特徴を検討するために，ドメイン分割，ゴール指向，シナリオベース，ブレーンストーミングという，広く使われている4つの代表的なアプローチを取り上げる．オブジェクト指向モデリングや状態モデリングといったモデリング技術とその記法は重要であり，いくつかの要求分析技法はそのようなモデリング技術に強く結びついているが，ここでは，純粋な要求獲得技法にのみ注目することにする．この4つの技法は，単に重要というだけではなく，互いに明確に異なった特性を有している．以下に，4つのアプローチを概観する．

ドメイン分割

ドメイン分割によって対象ドメインは，順次，サブドメインに分割されてゆく．扱うサブドメインが，十分に小さいく管理が可能なときは，要求を漏れなくリストアップすることが容易である．この技法を使って，小さなサブドメインや要求を，より大きなものにまとめ上げてゆくというボトムアップ式の処理も可能である．このようなドメイン分析手法の研究は，長い歴史を持っている．代表的な例は，R.Rieto-Diaz による仕事で[115]，J.Neighbor によって Draco プロジェクトに引き継がれた[100]．このアプローチは，分類手法として一般化することがきる．ビジネスプロセスやステークホルダのグループ化など，ドメイ

ン以外の要求に関わる様々な実体を分類するためのアプローチも，これに含まれる．

ゴール指向

多くのゴール指向アプローチが提案されてきた[7, 33, 87, 98]．そうした研究の中で，KAOSが形式的なアプローチを採用しているのに対し，GBRAMは非形式的なアプローチをとっている．

シナリオベース

シナリオベースのアプローチも一般的である[63, 112]．ユースケースは，シナリオの統合セットと考えられる．それゆえ，ユースケースアプローチ[32, 74]は，基本的に，このカテゴリーに属するが，より後期の要求工学フェーズで使用するのに向いている．

ブレーンストーミング

ブレーンストーミングは，システム開発経験が乏しかったり，新たなソフトウェアの開発が期待されているような領域での要求獲得に，特に効果がある．KJ法は，40年以上前に作られた手法であるが，洗練されたブレーンストーミング法の1つで，実業界では広く使用されている[84]．KJ法のソフトウェア要求工学への適用は，N.Takeda らによって報告されている[137]．

要求工学技法の違いを，その適用という観点から特徴付けるために，2つの次元について考える．1つは，獲得操作のタイプに関するもので，もう1つは，対象物のタイプに関するものである．それぞれの次元を，さらに，分類する．すなわち，操作タイプを，静的(static)と動的(dynamic)という2つのカテゴリーに分け，対象タイプを閉鎖的(closed)と開放的(open)という2つのタイプに分ける．表-3.1に，2つの次元からなる空間上に配置された4つの技法を示す．

要求工学技法を分類するに当たって，この2つの次元を使用する．

最初の次元は，要求獲得プロセスのなかで，静的(static)な構造と動的(dynamic)な振る舞いの，いずれに焦点を当てて活動しているかということである．

表-3.1　技法と次元空間

操作 / 対象	静的(Static)	動的(Dynamic)
閉鎖的(Closed)	領域分割	シナリオベース
開放的(Open)	ゴール指向	ブレーンストーミング

静的

要求は，静的な構造として捉えた問題領域から，構造的あるいは体系的な方法で収集され，整理される[18]．一般に，このタイプの技法は，静的な構造を持った問題領域を扱うのに適している．(サブ)ドメインやゴールあるいは要求それ自身といった，対象に関する要求を分解／合成，あるいは，分類／集約するための規則やガイドラインがある．

動的

要求は，動的な文脈に焦点を当てた問題領域から，創造的かつ非手法的な方法で収集される．このタイプの技法は，問題領域を動的な文脈で扱うのに適している[22]．多くのステークホルダにとって，手続き的な振る舞いや時間的な状況変化について考えることは，比較的容易な場合が多い．要求は，ステークホルダや分析者の想像力によって列挙されたり生成されたりする．しかし，このような手法の使用は，動的なドメインに限定されるわけではない．実際，このタイプの代表的な手法であるブレーンストーミングは，静的な性質を持った要求を獲得するのにも使われている．

第2の次元は，分析されるべき対象空間が閉鎖的(closed)か，開放的(open)か，という特性に関係している．

閉鎖的

対象空間は，比較的安定しており，良く知られていて，閉じている．そうした空間は，基本的に構造化が可能であり，形式論的に多様な型に整理することができる．それゆえ，主として，対象の形式や構文に注目することによって理解することが可能である．

開放的

対象空間は，安定しておらず，あまり知られていず，開いている．そうした空間を探索するには，対象の意味を詳細に検討しなければならない．

他の様々な要求工学技法も，この2つの次元によって展開される平面上に写像することが可能である．この平面を要求工学技術マップと呼ぶ．図-3.1は，代表的な技法を配置した要求工学技術マップである．技法の配置は基本的に，筆者の経験をもとに行われ，筆者もその一員として活動している実務家たちによる要求工学研究グループ Mahoroba のメンバーによって確認された．

このマップ上に，いくつかの注目に値する類型が見られる．

1．右半分の上部から下部にかけて，MSC，ユースケース，シナリオ，物語法，ロールプレイといった一連のシナリオベースの技法が見られる．これらの技法は，動的なイベントの進行におけるユーザや専門家の経験と直感をベースにしているという，要求獲得上の共通した性質をもっているが，決して硬直的ではない．上方部の技法群は，形式あるいは構文を意識しながら，閉鎖的な空間を調べるのに比較的適している．それに対し，下方部の技法群は，開放的な空間を調査するのに適しており，より人間性を指向する傾向を持っている．プロトタイピングアプローチも，同様の性質を持っており，その中心的課題は，動的な会話をシミュレートすることである．
2．左上から右下にかけての対角線に沿って，ドメイン分割，構造化インタビュー，カードソーティング，抽象化，KJ 法，ブレーンストーミング，民俗学的技法といった，一連の知識獲得分類型のアプローチが見られる．この対角線は，左上に向かっては収束的，右下に向かっては発散的という方向性を持った軸とみなすことができる．
3．ゴール指向アプローチの中で，形式的な KAOS は上方に位置づけられ，非形式的な GBRAM は下方に位置づけられる．i*フレームワーク[149]は，ゴール指向アプローチにオーバーラップしているが，右方向に拡張している．ゴール指向アプローチとシナリオベースアプローチを連結させた多くの提案があり[118]，それらは，要求工学技術マップ上には示されていないが，左下と右上の象限の架け橋である．

4．　前に述べたように，実体を分割するトップダウンと，合成するボトムアップの両方のアプローチが左上象限に含まれている．KJ 法は，ブレーンストーミングによって作られたカードを分類するプロセスを持っており，より閉鎖的な左上方へ位置づけられる．Problem frames アプローチ[73]は，トップダウン階層型技法によっても並列並行型技法によっても構造化されない分割であるが，他のドメイン分割と関連付けることができる．

5．様々なアイデア生成タイプの技法が右下象限に含まれている．例えば，アブダクションベースのアプローチ[120]もここに配置される．

いくつかの手法は，異なった象限の中に分類された技法を合成している．例えば，静的－閉鎖的象限に位置づけることのできる Multi Viewpoints アプローチは，複数の視点の分類をもとに要求獲得を行うが，多くの視点からの要求を処理するためにシナリオやゴールやその他の技法を合体させる．

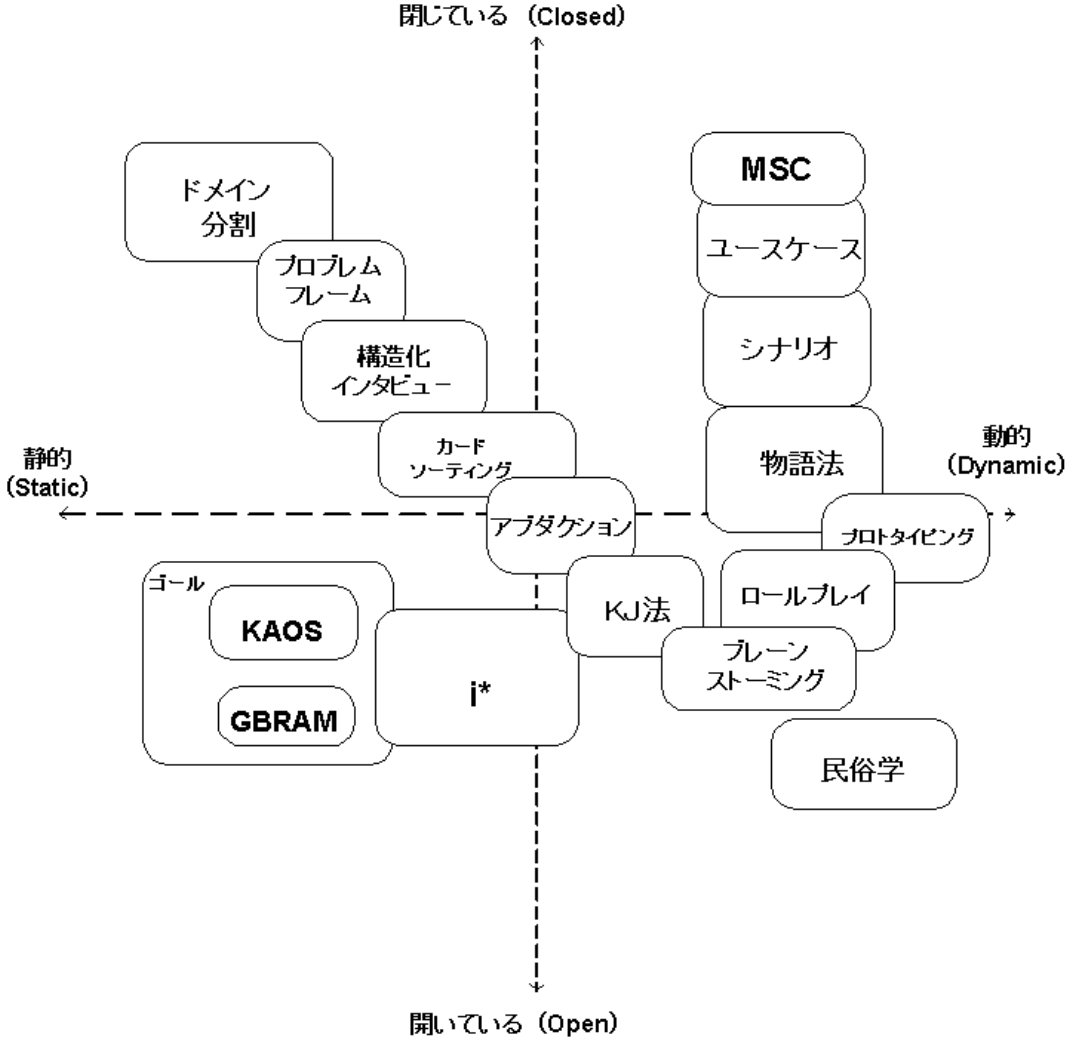

図-3.1　要求工学技術マップ

3.2. プロジェクト特性への要求工学技法の適合性

要求工学技法は，要求技術者の直感や経験をもとに選択され，それぞれのプロジェクトに適用される[4, 62]．優れた技術者は優れた選択をし，選ばれた技法を適切に適用する能力を持っている．しかし，訓練と経験に乏しい未熟な技術者は，選択肢の幅が限定されている．ひとたび，プロジェクトに適さない技法が選択されると，プロジェクトは奈落の底に落ちてゆくことになるだろう．

前節で述べた要求工学技法の特性は，プロジェクトの特性との適合性が判定され，手法選択のプロセスをガイドしてゆくために使われるべきである．

本節では，アプリケーションドメイン，要求技術者のタイプ，情報資源，ユーザの参画，要求特性という5つの要因によってプロジェクトを特徴付ける．この5つの特性は，筆者の経験をもとに，要求工学技術マップへの適応性を考慮して選んだものである．J.Naumann らは，プロジェクトの偶発性の要因を，プロジェクトのサイズ，構造化の程度，ユーザタスクの理解度，開発者タスクの理解度という4つのカテゴリーに分類しており[99]，筆者の選択が妥当であることが分かる．

ここで取り上げる要求工学技法は，どのような大きさのプロジェクトにも適用可能であると考えられる．「構造化の程度」は本研究におけるアプリケーションドメインの安定性に対応し，「ユーザタスクの理解度」は情報資源とユーザの参画に対応し，「開発者タスクの理解度」は情報資源と技術者のタイプに対応している．さらに，要求品質と非機能要求という2つの要因を取り入れることにする．これによって，選択のための意思決定要因はさらに豊かになるだろう．

手法の適合方向を示すために，プロジェクト要因を要求工学技術マップ上に写像することにする．図-3.2に示したような，四角形の中の双頭の矢印によって，それぞれの要因を象徴的に示す．四角形の位相は図-3.1の要求工学技術マップ平面に対応している．

3.2.1. アプリケーションドメイン

図-3.2で示した方向は，アプリケーション領域の安定性に対応している．上方向は，比較的安定的なアプリケーションのタイプである．対象ドメインに類似したシステムの開発経験の知識がよく蓄積されており，問題領域は閉じた空間として捉

えられ，機械的，形式論的な方法で調査できる．

下方向は，比較的新しく，あまり知られていない非安定的なアプリケーションである．このようなアプリケーションの要求を探すためには，深層レベルでの意味解析が必要となる．問題領域は開いた空間と見做される．

両ケースとも，静的－動的に関する次元とは比較的無関係である．例えば，あまり知られていないアプリケーション領域の要求獲得は，ゴール指向タイプのアプローチでもブレーンストーミングタイプのアプローチでもよく，選択は他の要因を考慮して行うべきである．

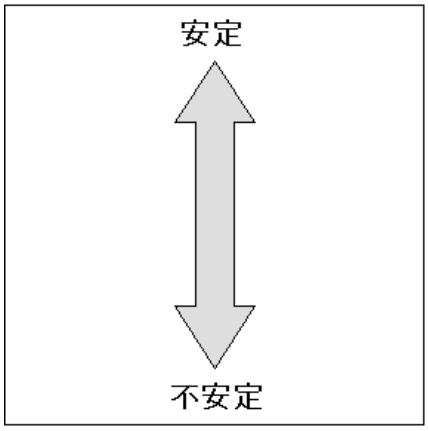

図-3.2 ドメインの安定性

3.2.2. 要求技術者タイプ

図-3.3の方向は，要求技術者の性質に対応している．左方向の技術者は，論理的かつシステマティックな思考法を好む傾向にある．右方向の技術者は，むしろ想像性や直感的思考法を好む傾向にある．別の見方をすれば，左側の人たちは比較的理論志向で，右側の人は経験志向であるともいえる．要求工学技術マップ上で左右に分類された技法群と，このグループの分類は対応している．

要求分析チームは異なった背景を持つ多くの人たちによって構成されているが，多くの場合，要求工学プロセス全体をコントロールしているリーダーがいる．例えば，J.Carroll と P.Swatman は次のように述べている[22]．「･･･ひとたび，チームがフィールドに入り，情報収集を始めたら，チーム活動の多くは，A1の仕事を

支援するために組織化される. A1はプロジェクトの要求技術者で, 問題領域と, その空間内の主要実体やシステムや情報要求を分析する.」ここで, A1は, 技術者, 社会学者, 電子コマースの専門コンサルタント, 研究者という7人のメンバーで構成される要求工学チームの2人のリーダーのうちの一人である.

単一の優位なリーダーを発見するのが困難な場合もある. マップはそのような場合でも有効である. すべての代表的技術者のタイプをマッピングすることによって, 全体としてチームの性質を示すことができる.

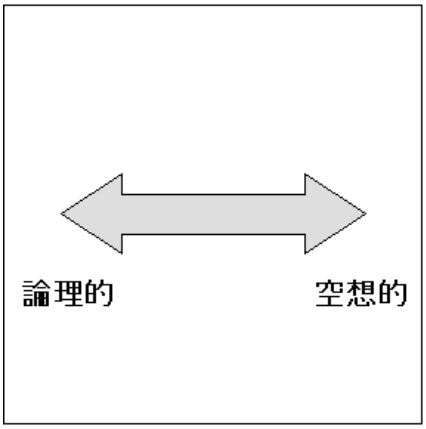

図-3.3 技術者の気質

3.2.3. 情報資源

文書や知識, あるいは既存のシステムの調査を通して入手できる利用可能な情報が大量にある場合は, 探索すべき空間は限定され, 閉鎖側の技法が適する. 情報は整理され, 主として, 構文ベースの方法によって整頓されるだろう. 一方, 使用可能な情報が少ないとか隠されているといった場合には, 要求は, 開放側の技法を使って, 開いた空間の中を探される.

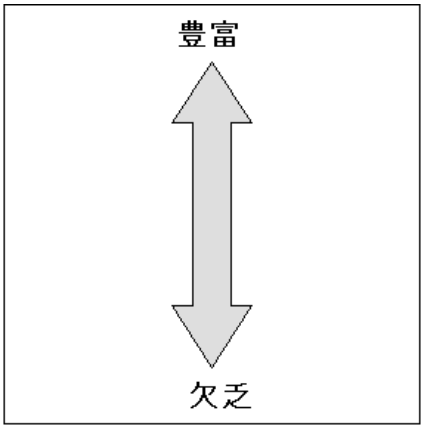

図-3.4　情報資源

3.2.4. ユーザの参加

ユーザの参加は，情報資源の要因と関係がある．もし，情報が形式的な知識として利用可能なら，要求収集のためのユーザの参加は不要である．しかし，ゴール探索やブレーンストーミングのような手法を実行するには，ユーザの参加は不可欠である．

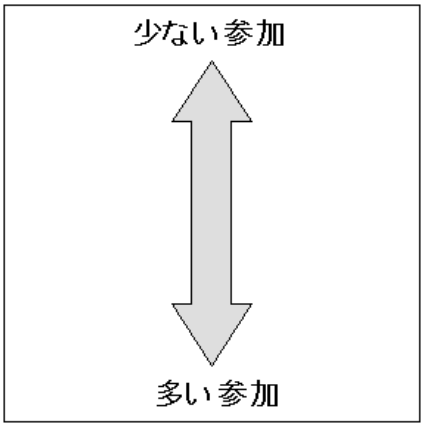

図-3.5　ユーザの参加

3.2.5. 要求品質

要求の品質は，多くの尺度によって評価される．IEEE標準830-1998 には，正当性，非あいまい性，完全性，一貫性，ランク付け，検証性，変更性，追跡性があげられている[69]．これらの特性はすべて重要だが，それらの間の優先順位はプロジェクトの性質によって異なってくるだろう．

静的－閉鎖的象限の技法は，問題領域を組織的にカバーするのに適している．それゆえ，対象領域を完全にカバーするという意味での完全性が達成できる．動的－閉鎖的象限の技法，特にシナリオベースの技法は，振る舞いの一貫性を保証するのに適している．特に，ワークフローやデータフローの一貫性は比較的容易に検出することができる．同様に，要求を動的な文脈で入手し，実装システムの振る舞いに一致させることができるので，追跡性についても実現することは容易である．静的－開放的象限の技法，特にゴール指向アプローチは，ゴール間の矛盾を発見するのに向いている．いくつかの矛盾解消手法とともに，このタイプの技法を使って要求の論理的一貫性を保証することが期待される．動的－開放的象限の技法は，期待される将来の要求も含め，異なった観点から多様な要求を作るのに向いているので，それらは要求への適合性を予測するのに役立つ．

図-3.6　要求品質

3.2.6. 非機能要求

非機能要求のいくつかは，特定の技法によって容易に獲得することが可能である．ゴール指向アプローチの流れは，ソフトゴールの概念を用いた非機能要求をカバーすることでよく知られている[98]．

セキュリティ関係は，シナリオベースのアプローチによって効果的に扱うことができる．何故なら，攻撃者がシステムに侵略するかもしれないすべての可能なシナリオを発見することが，システム攻撃を防ぐための本質であるからである．可能性のあるシナリオを挙げることは，発生するかも知れないその他のリスクを防ぐ効果もある．

使い易さは，会話とインタフェースに関する感覚的経験の機会をユーザに提供するプロトタイプ手法によって，特によくカバーされる．

3.3. 状況変化への貢献

プロジェクトは，前節の分析によって選択された適切な要求工学技法によってスタートする．しかしながら，新たな課題が発生し，それが解決されると，第2章で述べたように，プロジェクトの状況は変化する．要求分析フェーズ中にプロジェクトが障害に遭遇するかもしれないし，後期フェーズで要求仕様の不備が発見されるかも知れない．我々の経験と M.Christel らによるコンサルティング文献[25]をもとに，要求獲得活動で発生する問題を表-3.2に示した．

これらは，3.2.5節で扱った要求品質のネゴシエーションに関わる問題でもあるが，ここではそのような問題が検知されるイベントについて考える．そうしたイベントが検出されたときは様々な判断がなされる．使用している要求工学技法を変更することは，考え得る最も効果的な戦略の1つである．要求の品質と獲得技法群の関連については既に述べたが，表-3.2に挙げたような要求問題が顕わになったときにとるべき戦略は，それぞれの要求品質に対応している獲得技法グループ内の技法を単純に適用することではない．注意すべき事柄を，以下に示す．

表-3.2　要求獲得の問題

不完全な要求	
	ニーズの不完全な理解
	不完全な領域知識
	ユーザの協調性不足
	暗黙的仮説の見落とし
間違った要求	
	間違ったシステム境界
	システムの目的の間違った理解
曖昧な要求	
	類義語や同義語
	異なったユーザの観点の違い
確定しない要求	
	変動する要求
	際限のない追加要求の受け入れ
多すぎる要求	
	組織化されていない嵩張った情報源
	多すぎる要求
	営業によるオーバーコミット
	不要な設計思考

完全性

左上象限の技法は，不完全性を発見するのに効果があるが，不十分な要求を満たすのには必ずしも適しているわけではない．ゴール指向やブレーンストーミングのような，要求工学技術マップの下半分の技法は，見失った要求を獲得するために使うのがよい．そのような手法を通して新たな要求を収集した後，完全性をチェックするために，分類スキーマのもとでそれらをソートし，配置するために，左上象限に戻ることが好ましい．

矛盾

要求同士の間に矛盾が発見されたら，それが振る舞い上の矛盾の場合はシナリオベースのアプローチを，論理的な矛盾の場合はゴール指向アプローチを適用することが効果的である．

膨大な要求

左上象限のいくつかの技法はトップダウン分割のために使われ，いくつかはボトムアップ集約のために使われる．問題が要求量の多さにある場合は，集約型技法が役に立つ．問題は，それらの間にどのような優先度をつけるかである．AHPのような評価機構を組み合わせたゴール指向アプローチは効果がある．

あいまい性

要求表現のあいまい性は，基本的には，要求獲得技法とは直交した問題であり，モデリングや表記用言語に依存している．しかしながら，もし，問題が具体性の欠如なら，要求工学技術マップの上方向へ移動することによって，操作ゴールを入手するためのゴール指向アプローチを実行したり，シナリオをメッセージシーケンスチャートとして形式化することによって具体化を支援することができる．

3.4. 事例研究

ケーススタディとして，2つのプロジェクトを取り上げる．1つは研究発表会支援システムで，もう1つは核燃料棒管理システムである．

3.4.1. 研究発表会支援システム

(a) プロジェクトの概要

このシステムは，企業内のソフトウェア工学シンポジウムのためのものである．プロジェクトの立ち上げ時には，本研究で報告した要求工学技法の分類と選択のためのガイドラインは完成していなかった．プロジェクトは，基本的に要求分析者の

経験とスキルをもとに進行したが，プロジェクトの観察を通して本研究のアイデアを洗練させることができた．

このシンポジウムは毎年行われており，スタッフメンバーから社員への論文投稿が呼びかけられている．シンポジウムは事務局によって組織化されるが，実際には，過去10年間は事務局のただ一人の事務局員によってシンポジウムが運営されてきた．この事務局員の定年が近づいたので，彼が持っている知識を残す必要が生じた．システム開発のもう1つの目的は，シンポジウムの管理作業をより効率的に行うことである．

プロジェクトを始めることが決定されたが，当該事務局員は忙しすぎてプロジェクトに参加できないことが判明した．代わりに，過去10年間の未整理の文書が要求分析者に渡された．そこには，会議議事録やレターや事務局員の個人的なメモなどが乱雑に含まれていた．

(b) プロジェクトの特徴

プロジェクトの基本的な特徴は，以下の通りであった．

- 対象領域は比較的限定されていて，小さい．
- 他方，開発組織は似たようなシステムの開発経験が無く，問題領域は彼らにとって新しい．
- ステークホルダの参加は望めない．
- 使用可能な情報ソースは大きいが，知識は組織化されていない．

(c) 要求工学プロセス

実際の要求分析プロセスは，次の4つのステップで行われた．

ステップ1：　ドメイン知識を入手するための情報の整理

最初のステップは，シンポジウムを組織化するプロセスと，事務局員の行うべきタスクを理解することだった．事務局員から情報を入手することができないので，情報資源は組織化されていない資料だけであった．この状況に，

静的－閉鎖的象限の分類技法がよく適していた．問題領域が比較的小さく閉鎖的であったという事実は，その適合性を高めた．シンポジウムを開催するための仕事は，計画立案，論文募集，論文選択，シンポジウム開催，論文誌の発刊という流れに沿って進むので，情報を整理するために，活動プロセスによる分割が選択された．この判断は，資料を調べた分析者によって発見され，論文を投稿した一人の著者が，一般的な従業員から論文投稿者，入選者，発表者へと変化するという事実によって確証された．

中核プロセスは，さらにサブプロセスに分割された．例えば，計画立案プロセスは，組織コミッティの立ち上げ，プログラムコミッティの立ち上げと査読者の選定，組織とプログラムのコミッティ会議の開催，スケジュールの決定という具合に分割された．

すべての文書は，このプロセスの分割構造をもとに整理され，冗長で重要性の無い情報は破棄された．

ステップ２：　シンポジウムを組織化するタスクの理解

文書は作業の進行順序に沿って並べられ，プロセスの構造を理解する助けとなったが，それぞれの具体的なタスクは未だ良く理解されていなかった．こうしたタスクを明らかにするために，動的－閉鎖的象限のシナリオベースの手法がよく適合した．過去のシンポジウムにコミッティメンバーとして参加した人たちへのインタビューによって得られた情報と，並び替えられた文書をもとに，分析者は，自分自身でシナリオを記述しなければならなかった．シナリオの例は，次のようなものである．

論文募集

1．特別セッションのテーマの設定
2．論文募集要項の作成
3．PCメンバーへの論文募集要項の送付と必要な修正
4．組織コミッティへの論文募集要項の送付と承認
5．論文募集要項の印刷依頼
6．全従業員への論文募集要項の配布

出来上がったシナリオは正当性をチェックするために事務局員に送られ，若干のミスが指摘された．

ステップ３：　要求の識別

前のステップは，分析者が対象問題領域とプロセスを理解するためのものであり，構築するシステムのための要求とは，明確に識別されなければならなかった．要求を識別するために，静的―閉鎖的象限の分類技法が再び使われ，要求空間を分割するために，2つの基準が用いられた．1つは，機能，性能，品質という要求タイプであり，他方はビジネス，操作員，システムという要求元である．このステージでは，事務局員はプロジェクトのために多少の時間を取れるようになっていたので，この作業を進めるために，分析者は事務局員との週1回の会議を持つことになった．表-3.3に，結果を単純化して示す．

表-3.3　要求マトリックスによる要求の整理

	機能要求	性能要求	品質要求
ビジネス	計画立案， 論文募集， 査読者の選択， 論文審査， 論文執筆者への告知	データ量： 300論文	執筆者情報の保護， 査読者情報の保護， 不正アクセスの防止
操作者	イントラネットによる処理， テンプレートの使用	操作応答： 30秒	容易な操作性， ガイダンスとヘルプ
システム	イントラネットの代替パス	MTBF：1日	論文ファイル互換性

ステップ４：　要求の洗練

要求は獲得できたが，十分に具体的なレベルにはないということが明らかになった．要求を洗練するために，ゴール分割法が使われ，ゴールが操作レベルに分解された．その作業を通して，いくつかの新たな要求が発見され，表-3.3の要求マトリックスに追加された．同様に，ゴールは「Why？」という疑

問詞によって上位方向に拡張され，その結果は，プロジェクトの承認を得るために管理者に伝えられた．

(d) 考察

要求獲得作業への要求者の非協力的な姿勢は，要求技術者を悩ましている最も大きな問題の1つである[133]．本事例では，要求工学技術マップと状況変化への技法選択戦略をもとに，4つの異なった手法を順次適用することによって，危機的な状況を回避することができた．技法選択の指針が無ければ，プロジェクトは破綻に陥るか，不完全な要求仕様に基づいて作成されたプログラムに対する係争問題に発展していたかもしれない．

システム化対象領域は限定的で，サイズも小さいので，この事例プロジェクトで使われた要求獲得技法の殆どは，要求工学技術マップの上半分に属するものであった．この事実は，3.2.1節の所見に一致している．問題領域は小さく，比較的閉鎖的であったけれど，アプリケーションのタイプは，ソフトウェア技術者にとってはむしろ新しかったので，彼らには問題領域は開いていると見做された．それが，最終ステップで，ゴール指向アプローチが使われた理由の1つである．しかしながら，ゴール分割を使った主な理由は，上で述べたように，要求を操作レベルに詳細化するためだったので，実際に適用されたゴール指向技術の性質は，形式的あるいは制約的な方向を指している．この手法を使ったもう1つの目的は，管理者にプロジェクトの正しさを示すことであった．

プロジェクトの大きさに較べ，この要求分析プロセスでは，4ステップという入念な方法が採用された．この4ステップというプロセスは，プロジェクトが始まる前には計画されておらず，プロセス実行中の意思決定の結果であった．それは，3.3節で議論した状況変化による，実際の要求獲得プロセス遷移の1つの例である．

3.4.2. 原子燃料棒管理システム

(a) プロジェクトの概要

システムのゴールは，電力会社における核燃料集合体とその取り扱いを管理することである．全体プロセスは，燃料倉庫への新たな核燃料集合体の搬入，検査とプールへの保管，炉心への移動，使用済み燃料の破棄，再処理のための搬出から構成されている．システムの重要な機能は，核燃料集合体を処理するための一連の作業指示書を生成することである．システムが関心を持っている主な作業は，製造工場から新たな倉庫へ，倉庫からプールへ，プールから炉心へ，炉心からプールへ，プールからプールへ，プールから再処理施設への燃料集合体の移動である．システムは，また，個々の燃料集合体の状態と位置に関する情報を保持し，それらの履歴を記録する．

作業計画立案者からの一連の作業指示を受け取り，指示書を出力しているCUIベースの古いシステムがあった．作業計画立案者は紙上のシミュレーションを通して計画を立て，結果をシステム端末に入力していた．この作業は長い時間を要するため，計画立案者にとって多大な作業負担となっていた．プロジェクトの目的は，燃料集合体の状態と位置をグラフィカルに示し，作業計画立案者であるユーザによって提供される作業指示に従って，その動きをシミュレートし，検証された一連の作業指示を印刷する機能を定義することであった．プロジェクトへの参加者は次の通りである．

クライアント:　電力会社本体の管理スタッフメンバーが2名

ソフトウェア技術者：　現行システムを開発したソフトハウスから5名

問題領域の法律や類似システムに関する知識を持ったコンサルタントが1名

要求技術者1名（筆者）

(b) プロジェクトの特徴

プロジェクトの特徴は以下の通りであった．

- 既に現行システムが稼動しているので，ワークフローは明確であり，アプリケーションドメインは比較的安定している．
- 要求技術者のタイプは，想像的というより論理的である．
- 情報資源は十分にある．現行システムとそれに関する文書は，良い情報資源であり，そこには全てのタスクの処理が文書化されていた．

・ ユーザは参加していたが，原子力設備が遠隔地にあったため，システムの最終利用者である作業計画立案者は参加できない．
・ 必要とされる要求品質の中で，完全性と構造的一貫性は極めて重要と考えられたが，操作の一貫性と変更可能性に関する優先度は比較的低かった．何故なら，システムの振る舞いには殆ど複雑さが無く，作業プロセスの変更は考えられていなかった．

(c) 要求工学プロセス

ステップ１：　プロジェクト特性による要求工学技法の最初の選択

上で示したすべてのプロジェクト特性は，最初の選択では，左上象限の要求工学技法が適していることを示唆しており，第2の選択肢として左下象限の技法があった．こうして，要求を収集し分類するためにドメイン分割が選ばれた．しかし，核燃料集合体の取り扱いは，動的な操作の特徴も持っているので，要求分析の後半に，右上象限の技法，特にユースケース法を適用するのが良いだろうということが予想された．

ステップ２：　新らたな要求

しかしながら，予期せぬ新らたな要求が現れた．システムの中心的な目標は核燃料集合体のデータを保持することであったが，政府が電力会社に核物質の在庫を報告させるための法律を作ろうとしていたため，核物質の在庫管理という新機能をシステムに加えるべきであるという決定がなされた．これはシステムに異なった性格を持ち込むことになった．その機能は，メンバーにとって，よく知られていない新たなものであり，法律に関する情報も不十分で不確定であった．この要求は，プロジェクトに状況変化をもたらした．

さらに，はじめに仮定していた要求を再検討した結果，GUIを使って作業手順のシミュレーションを行うという機能を新たに追加することになり，基本的に現行システムの踏襲を前提としていた当初の機能に比較して，異なった性格をもつことになった．その結果，次の3つの機能局面が顕になった．

燃料集合体操作局面

シミュレーション局面
核物質在庫管理局面

燃料集合体操作局面は，元々予想されたのと同じプロジェクトの性質を保持しているが，シミュレーション局面は，新たな機能として導入されたので，その位置を要求工学技術マップの下方向に移動すると考えられた．こうして，これらの局面を分析するために，最初にブレーンストーミング，次にプロトタイピングを追加することが決定された．核物質在庫管理は，導入される機能が全体的に新しいので下方向への移動が考えられたが，要求は論理的で静的であったため左舷域に止まったままであった．こうして，ブレーンストーミングを使ってこの局面の分析を始め，ドメイン分割と論理的分析を続けることが決定された．

ステップ３：　操作指示語の新たな課題

この時点で，燃料集合体の操作の中核局面に新たなアイデアが導入された．コンサルタントによって提案されたそのアイデアは，操作指示語セットの変更であった．古い指示語セットは，基本的な操作指示語から構成され，単一の燃料集合体に対する単一の操作指示であった．提案された新しい指示語セットは，複数の基本的指示語を組み合わせた複合指示語から構成され，操作計画立案タスクの効率を高めることが期待された．同様の指示語を使用している他の電力会社の類似システムでの成功例も説明された．

しかし，この提案はクライアントに受け入られなかった．ユーザは不慣れな指示語を使うことを嫌うかもしれないという懸念が生じ，さらに，基本指示語は，設計された操作指示列の微調整に必須であるということが判明した．また，クライアントは同業他社の仕様を受け入れることを嫌い，独自の指示語セットを欲した．

これは，一種のステークホルダ間の要求の矛盾という状況である．矛盾を解消するために，3.2.5節のガイドラインに沿ってゴール指向アプローチが使われた．その結果，提案された複合指示語セットが採用されたが，クライアントの懸念を和らげるために古い基本指示語も残すことにした．この時点で要

求工学プロセスにおけるユーザの不在が重大な問題として顕在化した．ゴール指向分析を実施している間，要求チームは遠隔地の核設備にいる最終利用者の意見を聞かなければならず，その回答は通常一週間遅れで報告された．あらゆる要求工学プロセスが遅れ出した．

ステップ４：　ユースケース分析

ステップ3の結論が最終ユーザによって承認された後，燃料集合体の操作局面に関する要求工学プロセスには，ステップ1で予想されたように，ユースケースアプローチが実施された．チームの内の3人のメンバーがユースケースを記述したが，200個ほどの大量のユースケースが作られ，作業はスケジュールされた期間内には終わらないと思われた．さらに，記述されたユースケースのスタイルと粒度が，記述者によって相当に異なっていた．この問題を解決するために，いくつかのユースケースに共通のパターンを抽出し，具体的なユースケースを記述するときのテンプレートとしてそれらを利用することにした．[移動－検査－移動]といった指示語の典型的な組み合わせがパターンとして抽出され，位置と場所といった指示の対象をパラメータ化した．この解決法は，大量の情報を，ドメイン分割によって分類するという規則の適用と考えることもできる．

(d) 考察

本事例の特徴はプロジェクトの状況が時々刻々変化していることで，プロジェクトが使用する技法や戦略は，時宜に応じて変化させなければならない．それぞれの状況とそこで発生した問題ごとに，要求工学技法マップと技法選択戦略知識が使用されることになる．プロジェクトの発足時に，適用可能な技法の候補を選択することができたので，その後の具体的な要求分析戦略を立案することが可能であった．また，クライアント間の要求の矛盾やスケジュールの遅延といった状況の変化が認識されたときには，マップとそれに関連付けられたガイドラインを使用することは特に役に立った．さらに，要求獲得作業だけでなく，複合局面の出現時に局面を分析するための手法を選択するのにも有効であった．

通常，1つの手法には複数の機能が含まれており，個々の機能に着目すれば，

手法の適用範囲は格段に拡大することになる．しかし，それぞれの手法には，それぞれ独自の目標があり，適切な条件下での使用によって最も有効な効果を発揮する．手法の単純な機能比較[93]からは，適切な適用条件を想定することは難しい．プロジェクトの戦略に手法を対応付けようとする方法もあるが[99]，手法の全体像を見渡すことができず，また，集約された戦略によって手法選択の幅が限られてしまう．要求工学技術マップは，適用対象の特徴と処理の特徴という2つの視点から手法の全体像を示すとともに，手法同士の相対的関係を視覚的に示している．このプロジェクトの要求分析プロセスを通して，要求工学技術マップが適切な要求工学技法の選択をガイドするのに効果のあることが検証された．

（ｄ）その他の知見

この事例研究で学んだ重要なことの1つは，システムは複数の局面を持っていて，システムを局面によって特徴付けるか，個々の局面に対する特別の性質を考えることが役に立つということである．また，技法の選択は，機械的にはゆかない．システムの特徴とその環境は一意ではないし，プロジェクトの特徴から要求工学技術への対応付けは，代替選択のための余地を残しておかなければならない．まとめれば，フレームワークはこのプロジェクトの要求技術者に，チームをガイドし，他の参加者を納得させる上での多くの自信を与えることができたといえる．

3.5. 関連研究

この研究と似た目標を持った研究はそう多くはないが，その中で，A.Hickey & A.Davis と N.Maiden & G.Rugg の研究は，関連が強い．Hickey & Davis は，9人の高名な専門家へのインタビューを行い，彼らがどのように要求獲得を行ったかを訊ねている[62]．インタビューのスタイルは制約が無く，インタビューごとに大きく異なっている．彼らが問題にどのように取り組んでいるかということに対する回答を提供するために，短文で記述された比較的曖昧で，意図的な4つの状況が，インタビューアーに提示された．結果は，1)技法を使う時期，2)4つの状況への一般的な回答，3)その他の情報，としてまとめられた．調査は興味深く，役に立つ情報を提供しているが，要求技術とプロジェクト状況の関係は明らかにされず，バ

ラバラに議論されている. Maiden & Rugg の研究は, より体系的である[93]. 12種類の要求獲得技法が選ばれ, 要求の目的, 知識のタイプ, 知識の内部フィルタリング, 観察可能な現象, 獲得文脈, 手法の相互依存性という6つの側面から比較されている. 結果は, 相互依存性を除く5つの側面に関する5つの表によって表わされ, 表のそれぞれのエントリーはチェックまたは表象的重みを表している. 提案されたフレームワークは, 要求獲得技法を選択するときに使われるこれらの表を含んでいる. これらは, 選択時のチェックリストとして役に立つが, 事例の報告が無い.

いくつかの論文は, 似たようなトピックスを扱っているが, その目標あるいはアプローチは本研究のものとは相当に異なっている. S.Yadav らは, 要求分析技法のための比較フレームワークを提案しているが, 彼らが比較したのは, 要求分析の後半に使われると思われる DFD と IDEF0 であった[148].

似たような目標を持った, 偶発的アプローチを扱ったいくつかの研究もある. 例えば, J.Naumann らは, 偶発的アプローチに基づいてプロジェクト状況を要求確定戦略に結び付けている[99]. そこには我々の意図との類似性がある. しかし, 我々のアプローチが獲得フェーズに焦点を当てているのに対し, その対象プロセスは要求定義である. C.Sullivan も偶発的アプローチを取り上げており, 情報システム計画戦略全体を対象としている[135].

他の強力なアプローチは, メソッド工学をもとにしている. G.Vlasblom らは, プロジェクトの状況プロファイルによって開発手法を決定している[146]. T.Punter & K.Lemmen も, 手法を問題特性に一致させている[114]. V.Savolainen は, 組織的情報システムの開発のための参照フレームワークを提案している[125]. 我々のアプローチが要求フェーズに焦点を当てているのに対し, これらのアプローチに共通した目標は, 開発手法の選択である. 彼らは, 設計, 実装, 検証プロセスを含む開発プロセス全体をカバーしようとしている.

E.Hudlicka は, 知識工学と認知科学の視点から要求獲得技法を比較している[65]. そこでは, レパートリーグリッド分析, 多次元スケーリング, 階層化クラスタリングという3つの知識獲得技法が選ばれ比較されている. これらの技法は, 要求獲得に直接的な関係はないが, 情報の創作や分類技術を扱っているので, 要求工学技術マップの右下から左上への対角線上に配置することができる. これらの3つの技法は, 航空機の安全性検査という文脈の中での比較によって知見を

得ているが，それらがシステム要求を獲得するのにどのように使われたかということは明確に議論されていない．

J.Coughlan & R.Macredie は要求獲得プロセスのコミュニケーションを強調している[31]．その結果，彼らはMUST，Joint Application Design(JAD)，User-Led Requirements Construction(ULRC)，Soft Systems Methodology (SSM) といったユーザ参加型方法論を比較している．本研究とこれらのアプローチの違いは，手法タイプの選択だけでなく，彼らの焦点が技法ではなく方法論であるという点にもある．ブレーンストーミング，シナリオ，などのレベルの技法は，それぞれの方法論に採用されているが，共有されているものもあれば，共有されていないものもある．

M.Bergman & G.Mark は，高レベルで要求を扱っているために，プロジェクトが要求を決定するという通常の事例とは異なり，要求がプロジェクトを選定している[9]．NASAのプログラムが分析されている．そこでは，多くのプロジェクトが提案され，問題はNASAの高レベルの要求を満足する適切なプロジェクトのセットを発見することであった．確かに，そのような高レベルの要求も分析され決定されなければならず，それは論文の興味ある部分ではあるが，本論文の目標とは異なる．

A.Padula は， Hewlett-Packard での要求工学プロセスを選択する実例を報告している．獲得技術の視点ではなく，要求工学プロセスのために，2つのプロジェクトが比較されている[108]．

3.6. 考察

要求獲得技法の全体像を，2つの特徴軸による座標空間によって表現し，同一の位相空間上のプロジェクト特性を写像するという方法は，上に示した事例によって一定の効果があることが検証された．

しかし，我々の意図は，すべての要求技術者がマップ上のすべての手法を知るべきであるといっているわけではない．要求技術者が要求工学技術マップから得ることのできる最初の効果は，彼らが知っている手法の位置を知ることである．それぞれの技術者は，それぞれの象限内の少なくとも1つの手法については，よく知っていることが望まれる．たとえそうでは無い場合にも，知らない技法はチームの他

のメンバーによってカバーされるだろう．さらに，よく知らない技法の特徴も，既知の技法との相対的な位置関係から，簡単に捕まえることができ，学習を助けるだろう．

この研究の結果は，要求選択のフレームワークをシステマティックに使うためのチェックリストあるいは規則のセットとして使用されるのが望ましい．実際のプロジェクトにフレームワークを適用する中で，チェックリストを使って経験を積むということは良いことである．

第 4 章

要求モデルの連携

プロジェクトの状況の変化や[19], ステークホルダ間の観点(perspective)の相違[49]に対応するために, 要求分析者は, 異なった手法を選択し, 異なった要求モデルを作成することになる. しかし, 要求変更が発生すると, プロジェクトは, 作成済みの関連する要求モデル同士の間で矛盾がないように調整しなければならない. これをモデルの一貫性の保証と呼ぶ.

モデルの一貫性を保障するには, モデル同士の連携が必要になる. 複数のモデルの一貫性を保とうとする研究は, 主に, 要求追跡の領域で議論されてきた[78]. しかし, 要求追跡は, 要求モデルから設計モデルというように,形式論的な変換によって作成されたモデル同士を順次追跡してゆくもので, 同一対象に対する視点の違いを問題としている要求モデル同士の連携とは, その意味が異なる.

メソッド工学では, 新たなモデルを作成するときの要求モデル同士の一貫性の保証に焦点を当てており, 要求変更時のモデルの変更については, あまり議論されていない[19,102]. しかし, 既存モデルの変更では, 異なったモデル同士の一貫性の保障に加え, 変更による同一モデル内の影響箇所を探し出し, 修正しなければならず, 新たなモデルの作成よりも, 遥かに困難な作業であることが多い.

異なったモデル同士の一貫性の保証という作業を, ソフトウェア構成管理ツールなどを使って支援しようという試みも行われてきた[45]. 確かに, 多くの構成管理ツールは, ソフトウェアの版管理という問題では効果を挙げている[55]. しかし, モデル自身の変更の支援にまで踏み込んでいる例は少ない. 本論文では, 要求変更時に, 関連する要求モデル同士の間で変更箇所を特定し, 同一モデル内での2次的な影響の予測を支援するためのフレームワークを提案する.

4.1. 要求モデル連携の枠組み

メソッド工学は，異なったモデルの間の一貫性を保証するためのいくつかの機構を提案しているが[19, 51]，それらの多くは，モデル記述言語上の制約条件にもとづいた，モデル構成要素同士の存在制約に関する形式論的な一貫性を保証しようというものである．しかし，要求変更に伴って変更を受けた要求モデルでは，モデルの構造や属性が変更されるだけでなく，モデルが表現しようとしていた対象領域そのものの意味までが変更されてしまうことがあった．要求モデルの変更時に保証しなければならないのは，異なった要求モデル同士の形式論的一貫性と意味論的一貫性である．

形式論的一貫性

モデルとは，問題領域の抽象であり，多くのモデル化技法は抽象化された対象を表現するのに図式を使っている．図式で使用する図形の表記上の意味は，それぞれのモデル記述言語によって定義されている．したがって，特定のモデル記述言語の規約に則って記述されたモデルは，形式論的に唯一の解釈が可能である．一方，異なったモデル記述言語を使用して作成したモデル同士の間では，問題領域内の同一の対象が，異なった図式によって表現されることになる．しかし，同一問題領域内の同一対象は，表記法が異なっても，意味的に同一のものである．複数のモデル記述言語から構成されているUMLでは，モデルの表記法上の相違を，メタモデルによって形式論的に統一し，それらの一貫性を保証しようとしている．

意味論的一貫性

モデル記述言語は，モデルの記法上の制約を規定してはいるが，記述対象の意味までは規定していない．モデルによって表現された対象世界の意味，すなわち，モデルの意味論は，モデル作成者によって定義される．同一の問題領域に対する異なったモデル同士の間では，形式論的な一貫性とともに，意味論的な一貫性を保障することも重要である．

要求モデルを変更する場合に保証しなければならない一貫性には，モデル同士の一貫性と，モデル自身の一貫性がある．モデルの一貫性を保証するには，モデル同士の連携が必要となる．モデル連携は，外部連携と内部連携に分けられる．外部連携は，異なった要求モデル同士の連携であり，内部連携は，同一モデル内のモデル構成要素同士の連携である．

外部連携

モデルの外部連携には，2つのレベルの連携がある．1つはモデルの変更に際し，変更を同期させる必要のある他のモデルを特定するための連携で，もう1つは特定されたモデル同士の間で，変更対象となるモデル構成要素同士を特定するための連携である．作業の進捗によって複数の版が存在しているモデルもあれば，1つのモデルが複数に分割して描かれている場合もある．まず，変更が行われたモデルから，連携するモデルを探し，次に，変更対象となるモデル構成要素を特定する．外部連携には，モデル記述言語の規約に基づく形式論的な連携と，対象問題領域に依存した意味論的な連携がある．

内部連携

モデルは複数のモデル構成要素によって構成されており，1つのモデルの中で，モデル構成要素同士は，複雑に関連しあっている．あるモデル構成要素を変更すると，その影響が他のモデル構成要素に伝播し，モデル全体に広がってゆく．2次的な変更箇所を特定するために，同一モデル内での影響追跡を行う．影響追跡のための内部連携は，モデル構成要素同士の連携である．内部連携にも，形式論的な連携と意味論的な連携がある．

複数の異なった要求モデルを採用しているプロジェクトでの，モデル連携を使用した基本的な要求変更作業は，次の通りである．

1. 要求変更が発生したら，要求分析者は，要求モデルを使用して変更箇所を確認する．どの要求モデルを使用するかは，変更が手続き的なものに対するものか，構造的なものに対するものかによって判断し，変更箇所が確認できたら，該当の要求モデルに必要な変更を施す．

2. モデルの外部連携により，変更が必要な他の要求モデルの変更箇所を特定する．特定された要求モデルの変更方法を検討し，形式論的一貫性と意味論的一貫性を保証するために，必要な変更を施す．
3. モデルの内部連携により，変更が施されたそれぞれの要求モデル内で，2次的な影響を受ける可能性のある箇所を特定する．変更の必要性と変更方法を検討し，形式論的一貫性と意味論的一貫性を保証するために，必要な修正を施す．

4.2. モデル連携の構造

モデルの変更や修正には，モデル記述言語や問題領域に対する知識が必要とされるため，モデルの変更作業は，これまで，そのモデルを作成した技術者に任されることが多かった[133]．しかし，たとえ十分な知識をもっていたとしても，人手による作業にはミスがつきものであり，また，該当モデルの作成者がいつまでも保守作業に携わっているとは限らない．そこで求められるのは，モデルの変更に伴って発生するモデル間の矛盾箇所を指摘してくれる機能である．

本節では，具体的なモデルの変更実験の結果を通して，異なった要求モデルの変更に必要な知識とその構造を検討する．この実験に使用したアプリケーションは，3.4.1節の事例研究で用いた社内研究発表会支援システムである．企業内研究発表会の事務局を支援するためのシステムを，国際会議のプログラム委員長の業務を支援するシステムに変更し，要求モデルの変更箇所を洗い出し，それを解析した．主な要求変更は，以下の通りである．

- 社内組織上のステークホルダの削除
- 学会事務作業部門の追加
- 承認に関する社内手続きの削除
- 入選論文の審査方法の変更
- 論文審査に併設されているプログラムコンテストの削除
- 上記変更に伴う事務作業の追加・削除・変更

実験に使用したモデルは以下の5つのモデルである．ゴール指向モデル以外は，モデルを記述するためにUMLの図式を使用した．シナリオを除くモデルの実験結果を，図-4.1から図-4.4に示す．変更箇所を示すために，変更前と変更後のモデルをオーバーラップさせている．それぞれの図中で，×印は削除されたモデル構成要素，○印は変更されたモデル構成要素，△印は追加されたモデル構成要素を示す．なお，このモデルでは，モデル連携による変更箇所と，影響追跡による変更箇所を同時に示してある．

ゴール指向モデル　（図-4.1）

ゴール指向分解によってシステムに必要な機能を導出する．ここではシステム化の目的をゴールに設定している．図式は，KAOS[87]の記法に従っている．この技法は，要求工学技術マップの左下象限のゴール指向技法に属する．

オブジェクトモデル（図-4.2）

問題領域の静的な構造を表現するためのモデルで，UMLのクラス図によって記述する．この技法は，要求工学技術マップ（第3章）の左上象限のドメイン分割技法に属する．

ユースケースモデル（図-4.3）

機能要求を直接モデル化したモデルで，オブジェクトモデルとは独立に作成される．使用した図式はUMLのユースケース図である．ユースケース法は要求工学技術マップの右上象限のシナリオベース技法に属する．

シナリオ

ユースケースモデルのそれぞれのユースケースごとに，テキスト形式のシナリオが記述されている．シナリオはモデルではないが，本論文では，シナリオを構造化し，モデルと同様に扱えるようにしている．シナリオは，右上象限のシナリオベース技法に属する．

メッセージシーケンスモデル（図-4.4）

問題領域の動的な振る舞いを表現するためのモデルで，オブジェクトモデルとシナリオをもとに作成されている．図式は，UMLのシーケンス図を使って記述してある．この手法も，要求工学技術マップの右上象限のシナリオベース技法に属する．

本実験によって判明したことは次の通りである.

- 通常の要求追跡では, 要求仕様書から設計仕様書といった異なった文書間での追跡関係が必要となる. しかし, 要求モデル同士の連携では, 要求モデルは要求仕様書に記述されているので, 文書間の連携は必要ではない. ただし, 要求仕様書が複数に分冊されている場合は, 文書間連携も必要となる場合がある. しかし, 文書間連携については, 本論文では扱わない.
- 形式論的一貫性を保証するための外部連携は, UMLのように, 異なったモデル記述言語同士の間にメタモデルが定義されている場合は, 規則の形式で比較的容易に定義することができる. しかし, ゴール指向モデルとUMLのモデルのように, 独立したモデル記述言語同士の場合は, 連携が必要なモデル同士を特定し, 外部連携規則を作成するためのメタモデルないし変換ルールを定義する必要がある. しかし, いずれの場合も, 連携規則によって支援できるのは, 形式論的一貫性だけである.
- 意味論的一貫性を保証するための外部連携は, モデル作成者の意図を理解しなければならないため, モデル記述言語にもとづいた外部連携規則によって記述することは困難である. モデル作成者の意図を汲むのに有効なのは, 実際のモデル上で, 依存関係を使って連携を記述する方法である. この依存関係は, UMLで定義されているものであるが, UML以外にも適用可能である.
- 形式論的一貫性を保証するための内部連携は, モデル記述言語の規約を規則化することによって容易に記述できる. 影響分析はこの規則を使って, 修正箇所を予測する. しかし, 意味論的一貫性を保証するために, モデル構成要素ごとにあらゆる影響の可能性を洗い出すことは不可能に近い.
- テキスト形式であるシナリオの場合は, 1つのシナリオをモデルと考えて, その中の1つ1つの文をモデル構成要素とすることによって, モデルと同等に扱うことができる.

これらの知見をもとに, 一貫性保証のためのモデル連携の記述方式を提案するする.

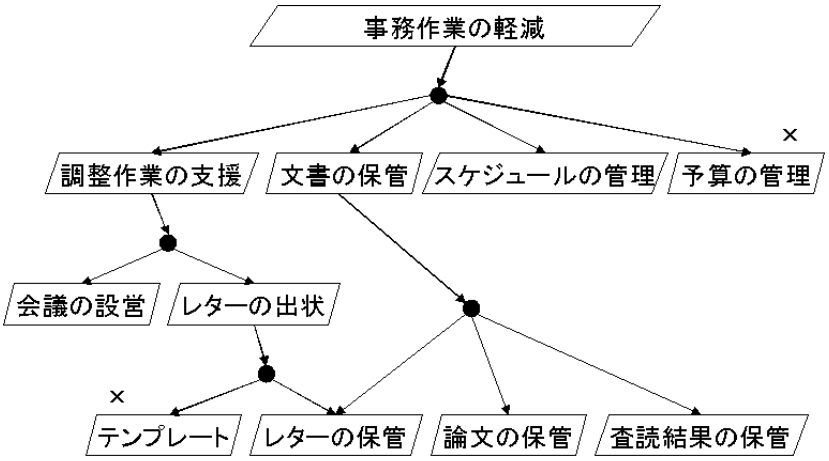

図-4.1　ゴール指向モデル

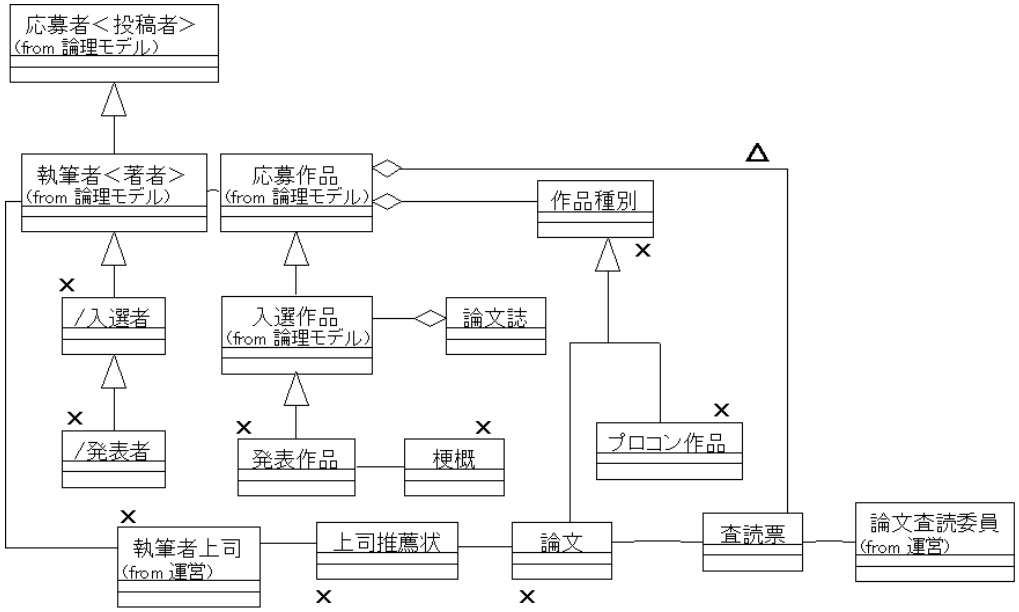

図-4.2　オブジェクトモデル

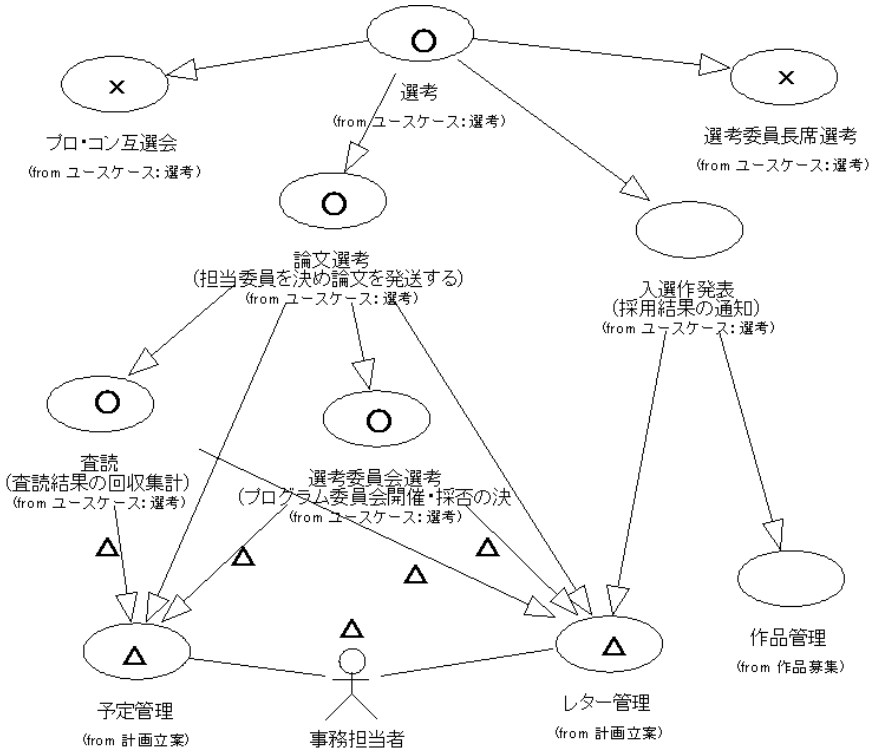

図-4.3　ユースケースモデル

図-4.4　メッセージシーケンスモデル

4.3. モデル連携の記述方式

モデル連携は，互いに一貫性を保証し合う必要のある要求モデル同士の関係で，外部連携と内部連携とがある．記法の制約条件にもとづく形式論的一貫性保証のための連携は，厳密な記述が可能な連携規則によって記述する．一方，作成者の意図を含む意味論的一貫性を保証するための外部連携の記述には，モデルそのものをベースとした連携依存関係を使用する．しかし，内部連携の記述は，作業負荷の割りに効果が少ないので実施しない．

4.3.1. 形式論的一貫性の保証

モデル連携規則では，変更の対象となるモデル構成要素とともに，それぞれのモデル記述言語上の制約条件をもとに，削除・修正・追加という変更操作のタイプと，必須・可能・不要という変更の必要性のレベルも表現することできる．

ここに示した規則は，個別のモデルのためのテンプレートである．末端クラス以外にも具象クラスの存在を認めたり，属性を公開して，他のクラスからの直接のアクセスを許可したり，具象ユースケースを共有ユースケースとして使用したり，あるいは境界オブジェクトや制御オブジェクトといったアプリケーション・アーキテクチャに依存したクラスを採用するなどといった，モデリング言語上では規定されていないモデル作成上の作法の違いによって，異なった規則が生成される場合があるが，ここでは考慮していない．そうしたモデル化作法の違いは，このテンプレートの規則の追加や変更によって対処することが可能である．

（ａ） 外部連携規則

実験で使用した5つのモデル間の連携関係の基本構造は，以下の通りである．

- オブジェクトモデルのステークホルダ・クラスと，それに該当するユースケースモデルのアクターの生滅は連動する．
- オブジェクトモデルのクラスと，それに該当するメッセージシーケンスモデルのオブジェクトの生滅は連動する．

- ユースケースモデルのアクターと，それに該当するメッセージシーケンスモデルのアクターの生滅は連動する．
- ユースケースの生滅にしたがって，それに対応するシナリオも生滅する．
- ユースケースモデルにおけるアクターの生滅は，シナリオに変更を与える．
- シナリオの変更と，メッセージシーメンスモデルの変更は連動する．
- ゴール指向モデルに対する形式論的一貫性を保証するための外部連携規則は存在しない．

オブジェクトモデルとユースケースモデルにおける外部連携規則の例を示す．

(Rule 1)　クラス図の具象クラスが追加・削除されると，ユースケース図の該当するアクターが追加・削除される．
(Rule 2)　クラス図の具象クラスが追加・削除されると，シーケンス図の該当するオブジェクトが追加・削除される．
(Rule 3)　クラス図の具象クラスが削除されると，シナリオの中の該当するオブジェクトが削除される．
(Rule 4)　クラス図に新たな具象クラスが追加されると，シナリオが変更される場合がある．
(Rule 5)　クラスの属性・操作が追加・削除・変更されると，シナリオが変更される場合がある．
(Rule 6)　クラスの属性・操作が追加・削除・変更されると，ユースケース図のアクターとユースケースの関係が変更される場合がある．

図-4.5　オブジェクトモデルの外部連携規則

(Rule 1)　ユースケース図にアクターが追加されると，クラス図に該当するアクターが追加される場合がある．
(Rule 2)　ユースケース図のアクターが削除されると，クラス図の該当するアクターは削除される．
(Rule 3)　ユースケース図にアクターが追加・削除されると，シナリオは変更される．
(Rule 4)　ユースケース図にアクターが追加・削除されると，シーケンス図は変更される．
(Rule 5)　ユースケース図にユースケースが追加・削除されると，クラス図のクラスは変更される．
(Rule 6)　ユースケース図にユースケースが追加・削除されると，新たなシナリオが追加・削除される．
(Rule 7)　ユースケース図にユースケースが追加・削除されると，新たなシーケンス図が追加・削除される．

図-4.6　ユースケースモデルの外部連携規則

（ｂ）内部連携規則

形式論的一貫性を保証するための内部連携規則は，それぞれのモデル記述言語の規約にもとづいている．以下に示すのは，オブジェクトモデルとユースケースモデルの内部連携規則であり，それぞれUMLの規約を使用している．

オブジェクトモデル内部連携規則（クラス図）

クラス図によって記述されるオブジェクトモデルの内部連携の基本構造は以下の通りである．

図-4.7は，基本構造の概念図である．四角形はクラスを示し，クラス間の関係はUMLの記法を用いて描いてある．影響の伝播は太い矢印で表してある．黒い矢印は影響が必ず伝播することを，白抜きの矢印は影響伝播の可能性があることを示している．

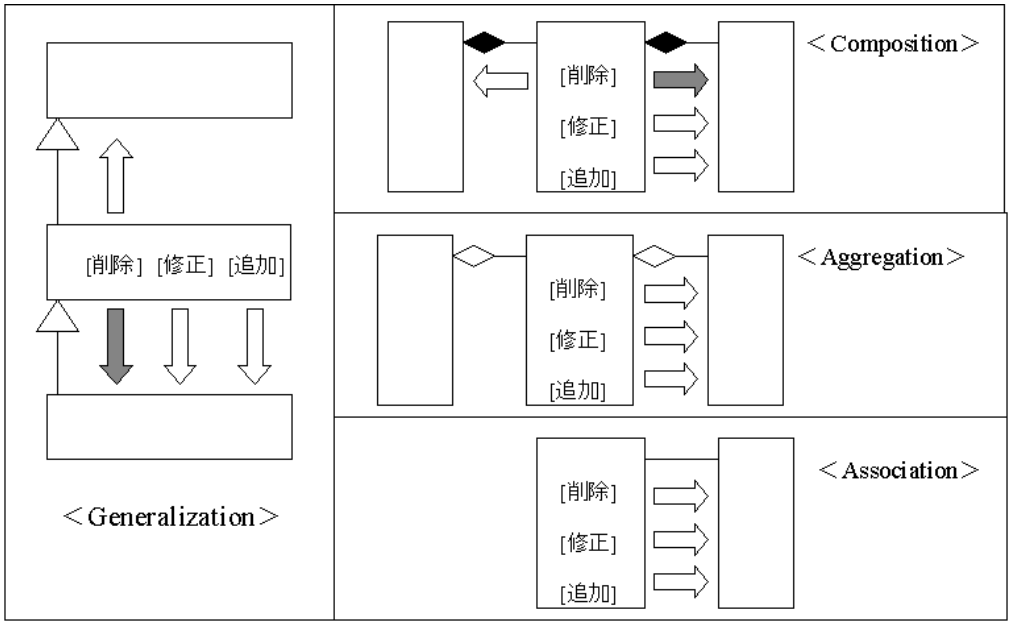

図-4.7　クラス図における内部依存関係

あるクラスの変更によって影響を受ける可能性のあるクラスは，そのクラスと汎化，関連，集約，合成関係で直接的に関連付けられているクラス，および属性や操作に対するアクセス関係を持っているクラスである．

あるクラスのすべての子クラスの削除は，それらの親クラスの削除を誘導する場合がある．

合成関係にあるクラスの親クラスの削除は，それに関連している全ての子クラスの削除を誘導する．

図-4.8に，オブジェクトモデルの内部連携規則の例を示す．

(Rule 1) 継承関係の中で，クラスが追加されると，そのクラスのスーパークラスとサブクラスが変更される場合がある．
(Rule 2) 継承関係の中で，あるクラスが削除されると，そのサブクラスは変更される．
(Rule 3) 継承関係の中で，全てのサブクラスが削除されると，それらの親クラスは変更される．
(Rule 4) 集約関係の中で，全体クラスが削除されると，部分クラスは変更される．
(Rule 5) 合成関係の中で，全体クラスが削除されると，部分クラスは削除される．
(Rule 6) 集約・合成関係の中で，すべての部分クラスが削除されると，全体クラスも削除される．
(Rule 7) 集約・合成関係の中で，全体クラスが変更されると，部分クラスも変更される．
(Rule 8) クラスの属性が追加されると，新たな操作が追加される場合がある．
(Rule 9) クラスの属性が削除されると，その属性にアクセスしている操作は変更される．
(Rule 10) クラスの属性が変更されると，その属性をアクセスしている操作は変更される場合がある．
(Rule 11) クラスの操作が追加されると，新たな属性が追加される場合がある．
(Rule 12) クラスの操作が削除されると，その操作を使用している他の操作は変更される．
(Rule 13) クラスの操作が削除されると，その操作によってアクセスされている属性が削除される場合がある．
(Rule 14) クラスの操作が変更されると，その操作を使用している他の操作が変更される場合がある．
(Rule 15) クラスの操作が変更されると，その操作によってアクセスされている属性が変更される場合がある．

図-4.8 オブジェクトモデルの内部連携規則

ユースケースモデル内部連携規則（ユースケース図）

ユースケース図によって記述されるユースケースモデルの内部連携の基本構造は以下の通りである.

図-4.9は, 基本構造の概念図である. ユースケース図における影響追跡の対象は, アクターとユースケースの関係および, include と extend の関係である. include 関係は複数のユースケースから共有される共有ユースケースの存在を示し, extend 関係は, 他のユースケースを拡張する拡張ユースケースの存在を示している.

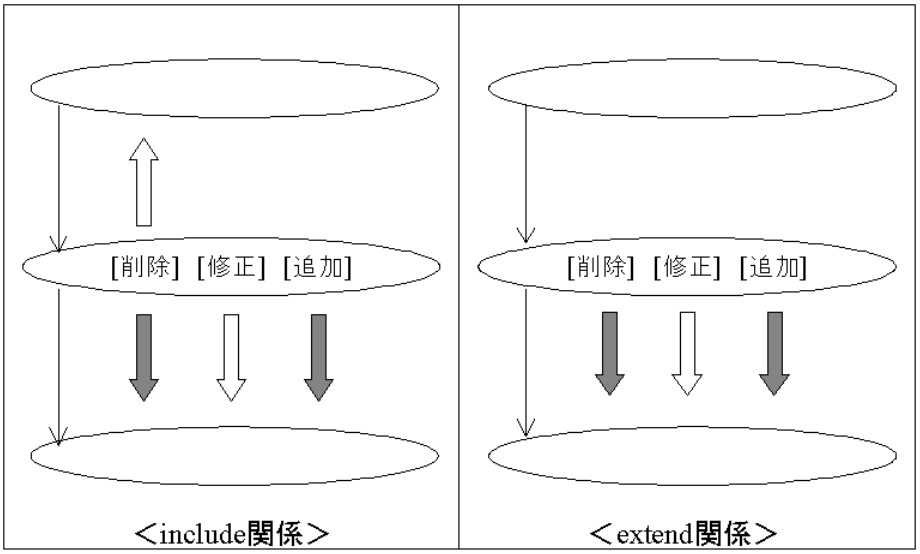

図-4.9 ユースケース図における内部依存関係

あるユースケースの変更によって影響を受ける可能性のあるユースケースは, そのユースケースと include関係, あるいはextend関係で直接的に関連付けられているユースケース, およびアクターである.

ユースケースは, 関連する全てのアクターが削除されると, ユースケース自体も削除される.

一部のアクターが削除されても, ユースケースそのものは削除されず, 対応するシナリオに変更が生ずる.

共有関係にある子ユースケースの削除は，それらの親ユースケースに対し，何らかの変更を誘導する．

ユースケースの変更は，ユースケース・シナリオの変更を伴う．

図-4.10に，ユースケースモデルの内部連携規則の例を示す．

(Rule 1) アクターが追加されると，そのアクターに対するユースケースが追加される場合がある．
(Rule 2) あるユースケースに対するすべてのアクターが削除されると，そのユースケースは削除される．
(Rule 3) ユースケースが追加されると，そのユースケースに対するアクターが追加される場合がある．
(Rule 4) ユースケースが削除されると，そのユースケースだけを使用しているアクターは削除される．
(Rule 5) include 関係の中で，共有ユースケースが追加・削除・変更されると，その親ユースケースは変更される．
(Rule 6) include 関係の中で，共有ユースケースのすべての親ユースケースが削除されると，その共有ユースケースは削除される．
(Rule 7) extend 関係の中で，拡張ユースケースが削除・変更されると，拡張される側のユースケースは変更される．

図-4.10に，ユースケースモデルの内部連携規則

4.3.2. 意味論的一貫性の保証

異なった要求モデル間の意味論的一貫性を保証するのに，モデル連携依存関係を使用する．

モデル記述言語はモデル構成要素を形式的に規定してはいるが，それぞれの要素にどのような意味を与えるかはモデルの作成者に任されている．個々のモデル構成要素の意味論的な制約条件を連携規則で記述することは可能であるが，モデルの作成者が図形によって表現しようとした意図を，規則で表現することは極めて困難である．それがモデル同士の連携依存関係を使用する理由である．

モデルの外部連携における，モデルおよびモデル構成要素という2種類の連携依存関係の違いは，依存関係のステレオタイプによって区別する．モデル同士の連携依存関係にはステレオタイプ《model-trace》を，モデル構成要素同士の連携依存関係にはステレオタイプ《element-trace》を使用する(図-4.11)．

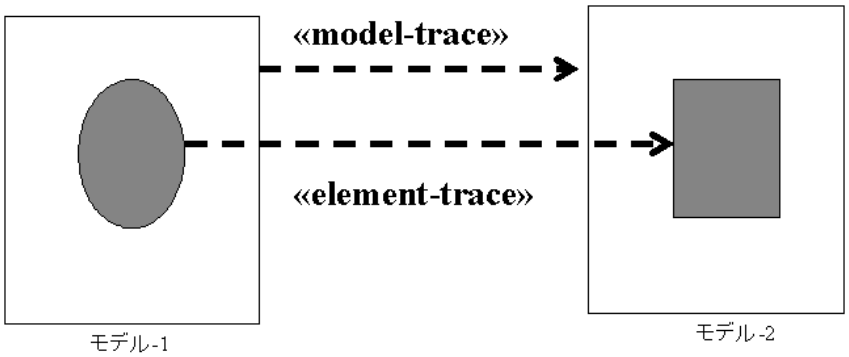

図-4.11　モデル連携を表す2つの依存関係

モデル同士の意味論的連携にも，制約条件がある．実験で使用した5つのモデル間の連携依存関係における意味論的な制約条件は，以下の通りである．

- ユースケースモデルのユースケースは，ゴール指向モデルのゴールを含んでいる．ユースケースに写像されていないゴールは，採用されなかったゴールである．
- メッセージシーケンスモデルのドメインオブジェクトは，オブジェクトモデルのクラスで定義されている．
- メッセージシーケンスモデルのアクターは，ユースケースモデルのアクターである．
- ユースケースモデルのアクターは，オブジェクトモデルのクラスで定義されている．しかし，ソフトウェアによって管理する必要のないアクターは，定義されていない．
- メッセージシーケンスモデルのメッセージパッシングは，シナリオに依存する．
- シナリオは，ユースケースモデルのユースケースに対応している．
- オブジェクトモデルのクラスに対応するオブジェクトは，シナリオの中で参照されている．

図-4.12に，異なったモデル同士の意味論的関係を示してある．それぞれのモデルの連携依存関係は，モデル設計者によって指定される．

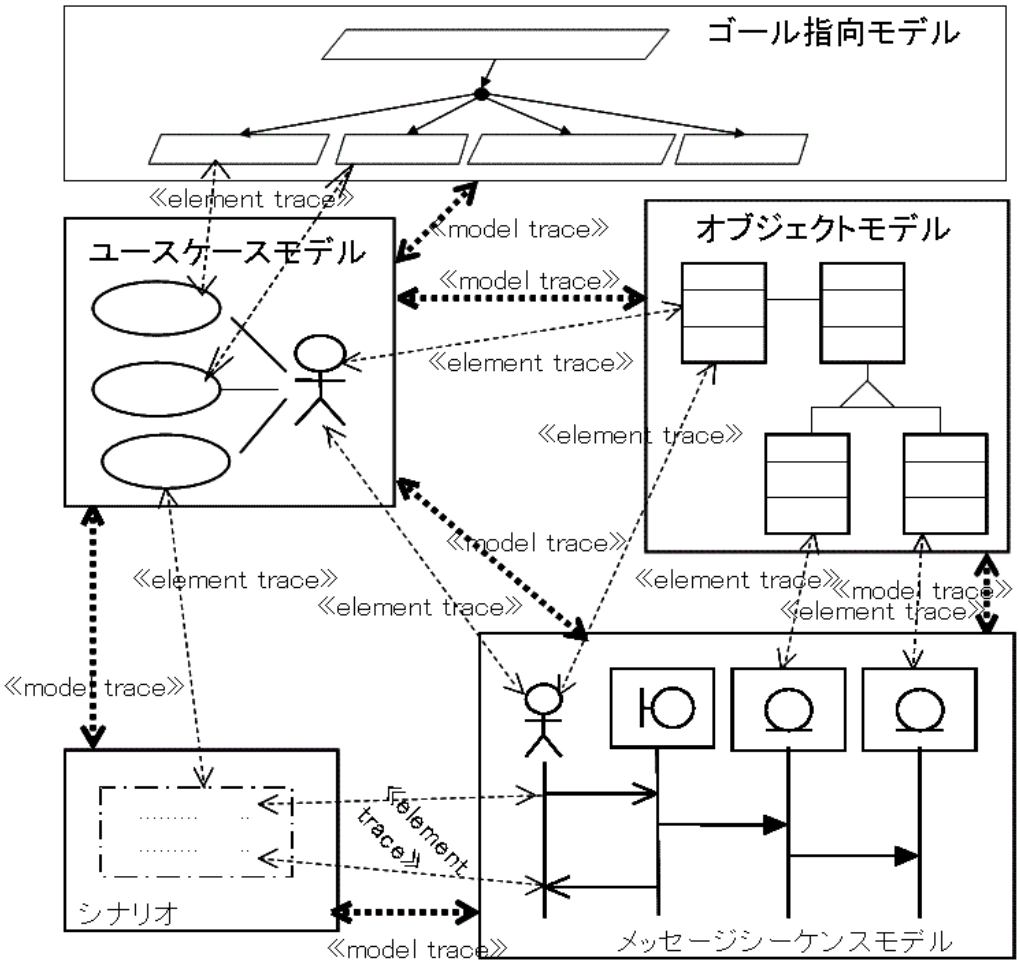

図-4.12 モデル連携依存関係

4.4. 事例研究： 対外系パッケージにおける要求変更管理

金融向けパッケージソフトウェアのモデル連携について述べる.

（a）プロジェクトの概要

本事例研究で取り上げる対外系ソフトウェアとは，バンキング・システムにおいて，SWIFTなどの公的システムや，大手顧客の固有のシステムなどとの接続を支援するためのソフトウェア・パッケージである．システムが提供する機能は，提供される金

融新商品や新たな通信機器の出現にあわせて，頻繁に，追加変更されるが，そのためのソフトウェア保守コストが大きな負担となっていた．それまでメインフレームで提供していた該パッケージソフトウェアを，サーバシステムへ移行するに当たって，機能の追加変更の容易性が大きなテーマとなり，ソフトウェアの開発は，保守の容易性という観点から，それまでのCOBOLベースからオブジェクト指向言語ベースに変更することになった．

ソフトウェアの開発プロセスは，要求モデリング，設計実装，保守の3つのフェーズに分けられ，作業期間はほぼ2ヶ月を要した．要求分析チームの構成は，仕様を決定するリーダーが1名，要求の提供者が3名，モデル作成者が3名およびアドバイザーが1名であった．

パッケージソフトウェアに対する要求は，クライアントから直接聞きだすことができないため，該当業務の専門化や客先担当のSEを要求者とし，プロジェクトのリーダーが仕様の最終決定に当たった．決定された要求仕様は，製品主管部のメンバーによってモデル化された．こうして定義された仕様にもとづくソフトウェアの設計と開発は，メインフレーム上のシステムを開発していたソフトハウスに外注されたが，開発後のソフトウェアに対する保守は，プロダクト主管部の要員3名が当たることになった．彼らの主たる任務は，変更要求への対応と要求モデルの更新であり，実際のプログラム変更作業は，担当ソフトハウスに外注された．

（ｂ）要求のモデル化プロセス

はじめに，要求モデルの作成とモデル連携の定義を行った．

ステップ１：要求モデルの作成

要求モデリングでは，次の4種類の要求モデルの作成を行うことにした．

ユースケースモデル：　クライアントに提供されるサービス機能の定義
オブジェクトモデル：　問題領域の構成要素とソフトウェア内部の情報の関係
シーケンスモデル：　それぞれのサービス機能ごとの情報交信手順
シナリオ：　詳細な機能と，関連する非機能要求の定義

要求獲得の結果，相手先起動集信やセンタ起動集信といったサービス機能を表す最上位のユースケースの総数は10個，また，加入者やサービス種別など，問題領域の構成要素や情報を表すクラスの総数は約70個であった．それぞれのサービス機能（ユースケース）ごとに，シーケンス図を描き，共通する情報交信の単位（サブシナリオ）を抽出し，それらを順次組み合わせることによって，5レベルの階層からなるサービス機能の構造が定義された．この階層構造は，共有サブユースケースの階層構造を表している．表-4.1に，相手先起動集信の一部を示す．

表-4.1　相手先起動集信のサブシナリオ階層構造

サービス機能 ユースケース	共有サブユースケース			
相手先起動集信	開局電文受付	電文入力	開局要求受付	
			電文検査	
			使用者通知	運用者通知
		相手先判定	加入者確認	確認コード検査
				パスワード検査
				サービス時間検査
				通話年月日記録
			通話記録	回線状況交信
		電文出力	開局回答創出	
			使用者通知	運用者通知
	ファイル転送開始	電文入力	開始要求受付	
			電文検査	
			使用者検査	運用者検査
		対象ファイル判定	通話記録確認	回線状況確認
			種別加入者確認	加入者取得
				レコードID検査 ：

最右列のサブユースケースは，クラスの間での情報交信パターンであり，それぞれシーケンスモデルによってその交信パターンが定義されている．それらを順次組み合わせてゆくことによって最上位のサービス機能が構成される．こうして，サービス機能を定義している10個のユースケースから，約80個の共有サブユースケースを抽出することができた．

ステップ２：モデル連携の定義

プロジェクトは，ユースケースモデル，オブジェクトモデル，シーケンスモデルという複数の要求モデルを使用したので，外部連携規則と内部連携規則およびモデル連携依存関係によってモデルの連携を定義することになった．

外部連携規則は，UMLの専門技術者が事前に作成したものを使用し，このモデル連携規則をもとに，モデル連携依存関係を設定することにした．本プロジェクトでは，連携関係にあるモデル構成要素同士の関係を，規則や依存関係ではなく，モデル連携対応表によって記述し，モデルの変更時に参照できるようにした．図-4.2に対応表の例を示す．1つのモデル構成要素の変更に対し，複数のモデルの複数のモデル構成要素が影響を受ける場合がある．

図-4.2　モデル連携対応表

連携元モデル		連携先モデル		関係属性	
ユースケースモデル		オブジェクトモデル			
モデル	モデル構成要素	モデル	モデル構成要素	可能性	関係タイプ
集信可否検査	集信状況記録	集信	集信状況	必須	形式
	通話記録	回線	回線状況	可能	意味
:	:	:	:	:	:

対応関係には，2つの属性を指定できるようにした．1つは修正の可能性に関するもので，もう1つは関係タイプに関するものである．関係タイプでは，形式論的関係か，モデリング作法上の関係か，意味論的関係かを区別できるようにした．要求変更が行われるたびに，この表は更新されることになる．

一方，影響分析のために，一般的な内部連携規則をチェックリストの形でまとめたものを作成したが，具体的なモデルをベースに影響関係を定義することは，要求モデルの構造そのものを定義することと変わらず，負荷が大き過ぎるため取りやめた．

（ｃ）要求の変更プロセス

ソフトウェアの開発後，製品主管部は，保守フェーズに入り，ソフトウェアのリリースと機能追加作業のための要員を残して，開発プロジェクトは解散した．

保守要員の役割は，顧客へのソフトウェアのリリース，要求モデルと設計仕様書の保守および，外注ソフトハウスへのプログラム修正の発注である．要求モデルの保守は，要求変更依頼が上部機関で承認されたのち，連携規則，連携依存表，影響追跡チェックリストを使って行われた．初年度5名いた保守要員は，次年度から3名となり，現在まで若手要員へのローテーションを継続しながら続いている．

（ｄ）考察

モデルの形式論的構造はメタモデルなどによって定義することは可能であるが，そのモデルに固有の意味論的構造を機械的に定義することは困難である[11]．しかし，モデルの作成者自身がモデルの意味論的一貫性やプロジェクト独自のモデリング作法を連携依存関係に反映させることにより，モデル変更作業における作業負荷の軽減と正確性の大幅な向上が可能となった．本事例では，チェックリストによる修正箇所の確認が可能となり，保守作業に開発要員が関与する必要が無くなった．コードを直接作成することのできない外注プログラムの保守では，その効果は一層顕著であった．また，手作業によるモデル変更作業では，人間のミスによる誤りが発生することがあるが，モデル連携規則はそうした誤りを発見するのにも役に立った．

モデル連携のもう1つの効果は，モデル連携規則の適用を通して，ソフトウェア保守チームのオブジェクト指向技術や該当モデルに対する理解が深まり，要員のローテーションがスムーズに行えるようになったことである．2年目以降の保守要員の数を減らすことができたのも，この教育効果の成果である．

影響追跡では，内部連携規則のチェックリストを用いたが，修正の対象を絞り込むことには十分役に立った．具体的なモデルの内部連携規則は極めて多様かつ複雑であり，しかも，要求変更発生時に実際に使われるのは，その極一部である．したがって，それらのすべてをモデル作成時に指定させることは現実的ではないと判断された．

（ｄ）その他の知見

本事例で学んだことは，モデル記述言語上の制約，モデリング作法上の制約，意味論的制約をそれぞれ異なったグループにまとめておくことによって，モデル変更の理由の違いごとに，該当するグループの規則だけを適用すればよく，作業の効率向上に役立ったということである．また，モデル連携依存関係は，モデルが大きくなるにつれて図式上の依存関係より表の方が使い易いことが分かった．しかし，モデル上に直接依存関係を指定できる支援ツールがあれば作業負荷は，さらに削減されるだろう．

本事例研究は，すでにメタモデル上での制約関係が定義されているUMLモデル同士の連携に関する作業であったため，連携規則を作成することは比較的容易であった．しかし，これが全く異なったモデル同士の連携であれば，規則を作ることの難しさが増加するが，それだけ，効果も大きいと考えられる．モデル連携を機械化する方法として，プログラムの構造解析と同様，入れ子構造の解析や共有情報を介したモデル構成要素間の関係を評価する方法などが考えられるが[154]，本研究では，その検証にまでは至っていない．

4.5. 関連研究

モデルの連携に関する知識は，メソッド工学の一貫性保証のための知識に似ている．メソッド工学には，複数の異なった手法によって作成されたモデル間の形式上の違いを，変換規則などを使って変換してゆく方式[102]と，個々の対象の形式に左右されないメタモデルを介して変換を図ろうとする方式[19]がある．しかし，メソッド工学は，ステークホルダの観点の違いやプロジェクトの状況の変化に応じて

適宜選択される手法により作成された異種モデル間の一貫性を保証するための技術であり，要求変更が発生した時の一貫性の保証についてまでは言及していない．既成のモデルを変更するときの一貫性の保証は，より困難な問題である．

Multi Viewpoint アプローチは，前者に属する代表的な手法である[51, 102, 103]．ソフトウェア開発に従事するステークホルダは異なった立場と観点を持っており，異なったモデルを作成する．それを Viewpoints と呼び，それぞれの Viewpoints を記述するためのフレームワークを提案している．個々のフレームワークは，そこで使用されるモデルの形式などを記述するためのスタイルスロットや，異なったモデル間での一貫性をチェックするための規則を記述するための作業計画スロット，開発中のシステムに関する Viewpoint の関心事領域を指定するためのドメインスロット，問題領域の記述である仕様スロット，開発状況の履歴を格納しておくための作業記録スロット，といった5つのスロットから構成されている．このフレームワークには，一般的な知識や規則を格納しておくためのテンプレートと，そのテンプレートから生成した個別のプロジェクトのためのインスタンスがある．彼らの考え方は，本研究との類似性が高く，特に，モデル間の連携に関する規則は，彼らの inter-ViewPoint check アクションに似ている．テンプレートとインスタンスという考え方も，本論文の研究と似ている．しかし，ViewPointは，モデル作成時の一貫性の保証に焦点を当てており，それゆえ，モデルの形式論的な問題しか対象としていない．一方，本研究では，作成されたモデルに対する要求変更時の一貫性の保証を対象としており，プロジェクトのモデリング作法やアプリケーションの意味論も扱っている．影響分析機能も，要求変更独自の問題であり，ViewPointの研究対象範囲外のテーマである．

Situational Methods Engineering [19]は，メタモデルを使って概念間の連携を図ろうとしている．すなわち，既存の手法から手法の断片を切り出し，それらの断片を組み立て直して，プロジェクトに固有の手法を構築する．個々の手法断片は，メタモデルによって形式論的な統合が図られる．この方法は，成果物である製品断片だけでなく，プロセスそのものである工程断片の連携も可能としている．モデル間の一貫性については，概念レベルでの一貫性保証規則が提案されている．この手法は，形式論的なモデル同士の一貫性を保証するための洗練された手法である．しかし，メタモデルを使った方式では，アプリケーションの意味論的な一貫性までを保証することは難しい．また，個々のモデルの言語仕様が定義されていること

が条件となり，不特定のモデルを扱うのには向いていない．

要求変更に伴うモデル間の変更の追跡は，むしろ，要求追跡の領域で研究されてきた[78]．要求追跡では，要求モデルの変更箇所を設計モデル，実装モデルというように，モデルの作成順序に従って追跡してゆくが，要求モデルという同一レベルのモデル同士では，基本的にモデル間の生成関係は存在しない．しかし，本論文における連携の基本的な機能については，要求追跡機能がベースとなっている[141]．要求追跡では，追跡作業を支援するいくつかのツールが提案されているが，その多くは，ハイパーカード機能などを使ってモデル連携作業を支援するものであり[40,111]，参照先のモデル構成要素まで特定することはできない．また，要求追跡に関する研究においても，影響追跡についての議論は少ない．

4.6. 考察

これまで，モデルの修正作業は，モデル作成者自身が行うことが多かった．しかし，モデル作成者が，いつまでも同一システムに携わっていられるとは限らない．担当者の変更と共に要求変更時の作業負荷が激増し，要求変更作業が不可能となるとともに，ソフトウェアそのものが陳腐化してゆくことになる．本研究では，外部連携依存関係や内部連携依存規則によって，モデル作成者がいなくても，異なったモデル同士の一貫性保証や同一モデル内の影響箇所の特定という作業の困難さを軽減できることを，具体的な事例を通して証明した．このことは，モデル連携依存関係さえ記述できていれば，状況に応じて異なったモデリング技法を選択できるという，多様性の克服という問題にも寄与できることを意味している．また，内部連携規則による影響追跡は，単一の要求モデルの場合にも必要な機能であり，外部連携が問題の多様さの解消に貢献するのに対し，モデル内部の矛盾の解消への寄与度が高い．オブジェクト指向技術の登場などによって，要求変更によるソフトウェア構造の劣化問題が一段落した現在，モデル変更に伴う2次的な影響箇所の追跡は，ソフトウェア進化のための鍵の1つであると考えられる．

要求モデル連携のための知識と，メソッド工学におけるモデルの一貫性保証のための知識は，意味的に同じものであるが，その使用目的の違いによって表現形式が異なっているだけである．同一の知識を別々に持つことは，冗長である．共通

する知識のフレームワークについては今後の検討課題である.

モデル同士の意味の連携に関しては, 今のところ, 分析者の努力に頼る以外に, 良い方法が見つかっていない.要求モデルそのものから, その意味を抽出するには, 要求モデルの設計時に, アナリスパターン[53] のような集約的な知識の適用を検討する必要があるが, これも今後の課題である.

連携規則を作成できるのが, 該当モデル記述言語に精通している専門技術者であることなどから, 要求モデルの連携は, プロセス知識の記述や要求工学手法の選択に較べて, そのコストは高価につく. しかし, 次の3つの観点から, その効果はコストに十分見合っていると考えられる. 第1に, テンプレートとなる連携規則は, 1度作ってしまえば, 新たな負担無しに, 再利用可能であり, 再利用の回数が増えれば増えるほど, 効果に対するコストの割合は低下してゆく. 第2に, モデルの一貫性の保証は, 異なったモデルを作成する場合には必須の作業であり, テンプレートにもとづいて, それを明示化するための作業は付加的なものである. これはメソッド工学にもいえることである. 第3に, ソフトウェアライフサイクルにおける保守コストの削減は, ソフトウェア工学の重要な課題である[19,74,119]. さらに, 実際のプロジェクトでは, 高度な技術を持ったモデル作成者を, いつまでも保守作業に携わらせておくことができないことを考えれば, 新たな保守担当者が仕様を理解する上でも, モデル依存関係はコストの削減に寄与するといえる.

第 5 章

まとめと関連する課題

最後に，本研究全体を総括し，関連する課題として，これまでに述べた知識を利用するための支援ツールの実現方法について言及する．

5.1. 研究のまとめ

要求工学プロセスは，ソフトウェアのライフサイクルを通して，ソフトウェアそのものの意味を定義するための唯一のプロセスである．このことは，要求工学が，現実世界とコンピュータという仮想空間をつなぐための学際的な技術であるということを意味している．実際のプロジェクトでは，要求分析者と呼ばれるソフトウェア技術者と，ステークホルダと呼ばれる非ソフトウェア技術者とによる，共同作業が行われており，そこでは，工学的な技術だけでは解決できない数多くの問題が発生している．熟練技術者の知識と技術が，それらの問題を解決するのに，おおいに役立っている．

本論文では，要求工学作業で直面する3つの典型的な問題を題材に，発見的知識と工学的知識の体系化と，それを表現するための方式について議論した．発見的知識には個別の普遍的知識とそれらを集約した抽象的知識があり，工学的知識にも個別の制約規則と，それらを定義するための構造的知識が必要となる．さらに，それぞれの知識には，それを適切に表現するための様々な表現法がある．それぞれの章では，問題ごとに，これらの知識の体系化と表現方式について議論した．

「はじめに」では，本研究の動機と目標について述べた．熟練技術者の経験にもとづく発見的知識は，定形的な形式を持たないため，それらを利用するには，それぞれの知識を体系化し，明示化するためのフレームワークが必要であるという問題を提起し，それを本研究の目標として提示した．

第1章では，要求工学の現状を技術的な側面から概括し，本研究のテーマである発見的知識の表現と体系化との関連について言及した．

第2章では，要求工学プロセスにおける偶発的な問題を解決するための発見的知識の記述方法について議論した．要求工学プロセス知識を記述するために，ソフトウェア設計用のパターンの形式を拡張し，状況セクションと課題セクションという2つの構造を提案した．さらに，プロジェクトの状態遷移の全体像を表示するための状態遷移図を提案した．

第3章では，プロジェクトにおける要求獲得問題を解決するのに必要とされる要求工学手法の選択という知識のフレームワークについて議論した．代表的な要求工学手法を，適用領域の特徴と処理操作の特性という2つの次元によって分類し，要求工学技術マップという座標空間上に配置した．また，プロジェクトの特徴を分類し，要求工学技術マップ上の具体的な手法を選択するための戦略的方法を提案した．

第4章では，要求変更の発生時に，モデルの一貫性を保証するための知識とその記述方法について議論した．要求変更の発生時に，複数の異なった要求モデル同士の一貫性を保つには，モデル同士の連携が必要であり，それを外部連携と内部連携という観点から整理し，それらを記述するための連携規則と連携依存関係を提案した．

3つの章の関係を図-5.1に示す．現実世界には，様々な知識が存在している．再利用可能な知識を収集するために，発見的知識には普遍化を，工学的知識には規則化を用いた．収集された雑多な知識を整理するための主要な方法として，分割と抽象化と組織化がある[18]．要求プロセスパターンのように個別に表現されている知識は，その中から同一の範疇に属する知識を選び出し，それらを組織化することによって，知識の全体構造を明らかにすることができる．一方，適切な技法の選択といった発見的知識の体系化には，抽象化を用いている．抽象化によって特殊な知識は捨象され，知識の全体構造が明らかになる．また，モデルを記述するための制約条件のような工学的知識は，知識全体を構造化するこ

とによって体系化を図ることができる．工学的知識が捨象されることは無い．

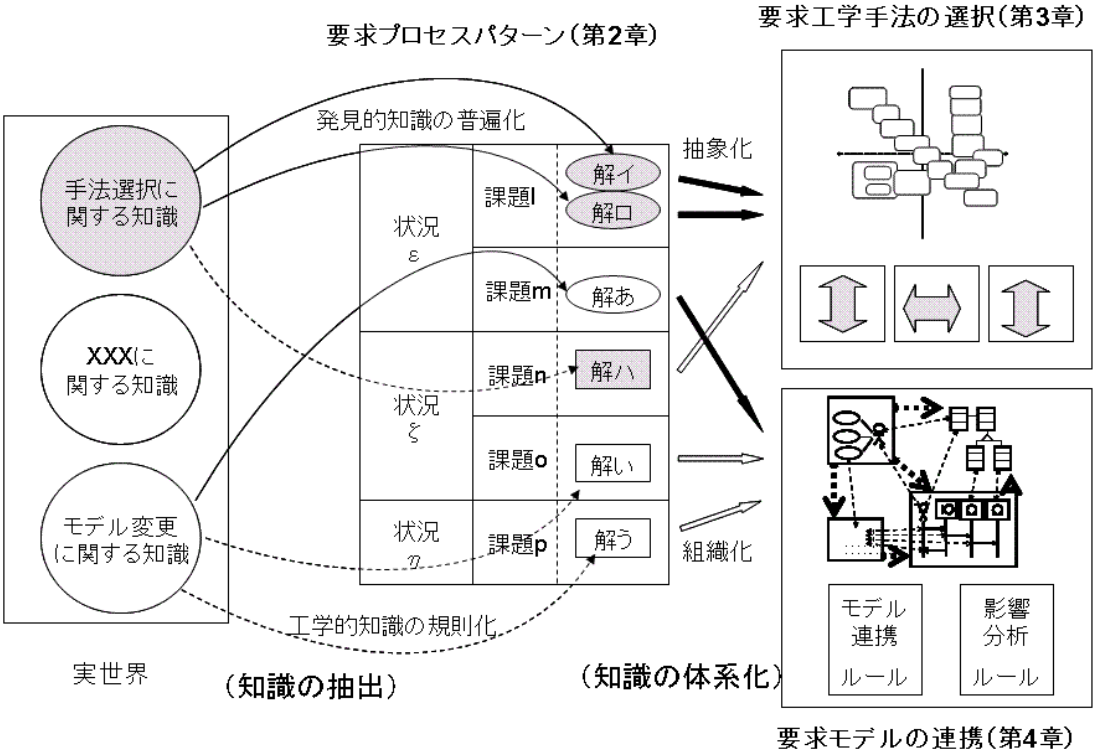

図-5.1　知識の抽出と体系化

知識は，それぞれ独自の表現形式を使って表現されることになる．本研究で使用した，それぞれの知識の表現形式をまとめると，以下のようになる．()内は，それぞれの知識表現が用いられた章である．ただし，知識とその記述法の関係は固定的なものではなく，他の方法を使用することも可能である．

テキスト　：プロセスパターン記述(2章)，手法適合原則(3章)
表　：プロジェクト問題(3章)
ルール　：モデル連携規則(4章)，影響追跡規則(4章)
座標空間　：要求工学技術マップ(3章)，プロジェクト特性(3章)
図式　：状況遷移図(2章)，モデル連携依存関係(4章)

図式や座標軸を使って表現される知識は，大局的な構造が重要な知識であり，利用者の感性を通して理解が可能な知識である．表やルールによって表現される知識は，個々の知識が明確に判別できるもので，その表現法は利用者に明快

な解を与えることができるが，説明能力に乏しい．自然言語による表現は，複合的な条件を持っている問題に有効であり，解の説明能力が高いが，冗長であるため，その記述はパターンのように抽象化する必要がある．

5.2. 研究の評価

本研究の，要求工学への効果について評価する．個別の研究評価は各章の考察で述べているので，本章では，共通の評価基準をもとに，定性的評価と定量的評価を行う．

（a）定性的評価

要求工学プロセスでは，クライアントとの機密保護契約や，プロジェクトごとに異なる課題や，再現できない作業環境などから，同一条件下での観測データを組織的に採取することが難しい．ここでは，本研究の成果を，筆者の主観を交えて，定性的に評価する．評価は，それぞれのプロジェクトでリーダー的役割を担っていた技術者から入手した感想を，筆者の経験もとに整理し直したもので，本研究で提案した方法が，それを使用しなかった場合に較べどれほどの効果があったかを，4段階で定性的に判定している．評価項目は以下の通りである．

- 有効性：　本来なら解決できなかった問題を，本研究の提案方法を利用することによって，どれだけ解決できたかを評価する．評価は，現場の技術者たちへの聞き取り調査に基づいている．
- 効率性：　単位作業に対する投資量で，時間的効率性やコスト的効率性があるが，ここでは人的コストに注目した効率性について評価している．評価は計画値と実績値の比較で行った．時間的効率性については，次の定量的評価の項で議論する．
- 完全性：　作業が正確かつ漏れなく行われたかどうかという作業品質を評価するもので，ここでは，要求仕様書の完成度によって評価することにした．要求仕様書の完成度は，IEEE830 の仕様

書評価項目の中から正当性，非あいまい性，完全性，一貫性を使って，全要求数に対する総違反数の比率を定性的に求めた．

- 独創性：　発見的知識でも対応できない予期せぬ問題が発生したときに，プロジェクトメンバーがその問題に対応できたかどうかを評価する．評価は，どれだけ適切な代替案を提示できるかという要求技術者のイマジネーション能力向上の評価でもある．評価は，筆者の独断的評価に顧客満足度を加味して行っている．
- 教育性：　本研究が提案する方法を使用することによって，要求技術者の技術力が向上したかどうかを判定する．担当技術者への聞き取り調査を通して知識量の向上を評価した．

評価結果は，以下の通りである．ここで，◎は非常に効果があった，○はそれなりに効果があった，□は変化が無かった，△は負の効果があったことを示している．

表-5.1　知識フレームワークの定性的評価

		有効性	効率性	完全性	創造性	教育性
2章	プロセスパターン	○	○	○	□	◎
	状況遷移図	◎	◎	□	○	□
3章	要求工学技術マップ	◎	◎	○	□	◎
	プロジェクト特性	○	○	□	○	□
4章	外部連携関連図	◎	◎	○	□	◎
	外部連携規則	○	△	◎	□	◎
	内部連携規則	○	△	◎	△	□

この定性的評価の結果から読み取れることは次の通りである．すなわち，全体

像を示す図表を伴った方法は有効性，効率性とも評価が高いが，作業の完全性にはあまり寄与していない．完全性は，全体像の把握より作業の綿密性により多く依存していると考えられる．事実，パターンや連携規則といった詳細な情報を提供する方法は，定常状態ではそれを利用するための付加作業が伴うために作業効率が低下する反面，作業の完全性は向上している．さらに，詳細な情報には，OJT的な使用を通しての教育的効果も認められた．連携規則の効率性に関する負の評価は，規則作成者の付加作業によるものであり，内部連携規則に対する担当技術者の創造性の負の評価は技術的に未熟な技術者が規則に依存してしまうためと考えられる．創造性については，予期せぬ問題への遭遇頻度が少なかったため，適切な評価ができていない．

（ｂ）定量的評価

本研究の定量的評価には，障害プロジェクトに対する有効性の評価を全障害数に対する回復障害数で測定するという方法や，成果物の品質向上性を製品の比較によって評価するといった方法もあるが，ここでは，筆者の経験をもとにした要求工学作業の時間的効率向上性について評価する．

作業の効率性は，定常状態での効率向上率と，障害状況での効率向上率という2つの視点でとらえることができるが，ここでは，具体的な事例をもとに，後者の評価を行う．ここで取り上げる3つの事例は，“多すぎる要求”という問題であり，安易なヒアリング調査を行ったプロジェクトがしばしば遭遇する問題である．それぞれの問題解決に当たっては，第2章の要求工学プロセスパターンを使用した問題解決戦略の設定か，第3章の要求工学技術マップを使った適切な手法の選択かの，いずれかの方法を採用している．以下に，定量的な観点からの問題解決プロセスについて簡単に記し，若干の考察を述べる．

事例１：航空運輸における要求項目の整理

ヒアリングや業界誌，管理者インタビューなどを通して，約2000件の要望や関連情報が収集され，その整理を3人で1か月ほど行ったが，整理できず，タイムオーバーランとなってしまった．適用した研究成果は，要求工学プロセスパターンである．業務をよく知っている担当者1名を指名し，領域分割法を適用

した結果，約半月で整理が行えた．(2.6(c)参照)

ここで収集された情報には，要求だけでなく業界や市場の動向も含まれており，要求プロセスでも最も初期段階での情報収集と位置づけられる．しかし，技術者は問題領域に関する十分な知識を持っており，それが領域分割法を選択した主要な理由である．彼らは，業界や市場動向の中から，暗黙の要求を抽出し，情報を洗練することができた．

事例２：介護支援における要求項目の整理

現行の介護支援管理システムの改築のために，現場担当者へのヒアリング調査の結果，約400件の要求が収集され，その整理を5人で10日ほど行ったが，検討メンバーは業務に疎く，要望の解釈がメンバーによって異なり，タイムオーバーランとなってしまった．適用した研究成果は，手法の選択である．業務に精通したメンバーがいなかったため，ゴール分割法を適用し，メンバー1名を指名し，クライアントの支援を受けながら，10日ほどで整理ができた．

収集された情報はクライアントからの具体的な要求であり，洗練された情報と位置づけることができる．しかし，技術者たちは，この新たなビジネスに対する知識は持っておらず，それがゴール分解を選択した理由である．

事例３：電力自由化対応における要求の整理

電力自由化対応のための機能は，電力会社も未経験のため，SIベンダーと共同で要求の取りまとめをすることになった．提案された項目は，現行システムの改良も含めて，150件ほどになった．これらを整理し，アプリケーションの構造を設計することになった．3名の担当者が1か月ほどかけて整理を行ったが，要求内容とソフトウェア構造の対応が確定せず，コストオーバーランとなってしまった．適用した研究成果は，手法の選択である．電力自由化の実体が不透明であったため，業務担当者やソフトウェア設計者など異なった観点からなるメンバー5人を指名し，KJ法を使って要求のグループ化を行った．整理は2日ほどで終わった．

ここでは，クライアントとの間で実現すべき要求の合意ができており，それらの要求をもとに，クライアントにアプリケーションの構造を提示するための作業であり，要求プロセスの最終段階である．単なる要求の分類ではなく，最終的なソフトウェアの構造や実現の可能性といった複数の分類基準を考慮しなければならず，それがKJ法を選択した理由である．

この3つの事例から作業量を定量的に定義すると，次のようになる．

- プロジェクトは，作業が3人月を超えるとコストオーバーランかタイムオーバーランとなる．
- プロセス知識パターンと手法の選択のいずれを採用しても，０．５～１．０人月で問題が解決している．

大量の情報の整理という問題に関して言えば，プロジェクトが障害に陥った時，なんらの手も打たなければ作業量は3人月を超え，そのプロジェクトは，遂には，タイムオーバーランかコストオーバーランに陥ることになる．しかし，ここで取り上げた事例によれば，作業量が2人月を超えない状態で，異常な状況を検知し，発見的知識の適用か適切な手法の選択を行えば，障害は回避できることになる．もっと多くの事例を収集すれば，より精度の高い有効基準値が得られるだろう．そうした基準値は，“多すぎる要求”以外の問題にも存在していると考えられる．これらの研究成果は，他の問題に対する障害克服のための作業効率の向上という形で寄与するであろう．

しかし，手法が適切に選択できればそれで問題が解決するわけではない．手法の適用の仕方を間違えば満足できる結果が得られない場合もあるし，手法に対する熟練度によって作業速度が異なることも事実である．しかし，個別の手法の使用法に関する発見的知識は，本研究の対象外である．この定量的評価で用いた事例では，それぞれの手法を適用した技術者の技量は，筆者が想定している平均的技術者のレベルである．

5.3. 関連する今後の課題

本研究では，知識の表記法と体系化について論じた．個々の課題は，それぞれの章で述べたので，ここでは，収集された知識の利用を支援するための機械化について述べる．本研究の成果である体系化された知識を使って要求工学作業を支援するための環境を提供することは，プロジェクトの破綻の最も多くの原因と

なっている要求工学上の問題[133]の解決に大きく寄与するだろう.

はじめに,それぞれの知識がどのように使われるかを検討する.知識の利用形態を図-5.2に示す.

知識の論理的な構造を条件部と結論部に分ける.条件部は入力情報から問題を分析し,結論部はそれに適した解を選択していると想定することができる.ここで,状況遷移図は,プロジェクトの問題とその背景情報をもとに,該当プロジェクトの状況遷移図上での位置を特定し,プロセスパターンは,状況遷移図上の位置情報をもとに実際のプロセスパターンを選択し,問題解決のための手法を提示することになる.

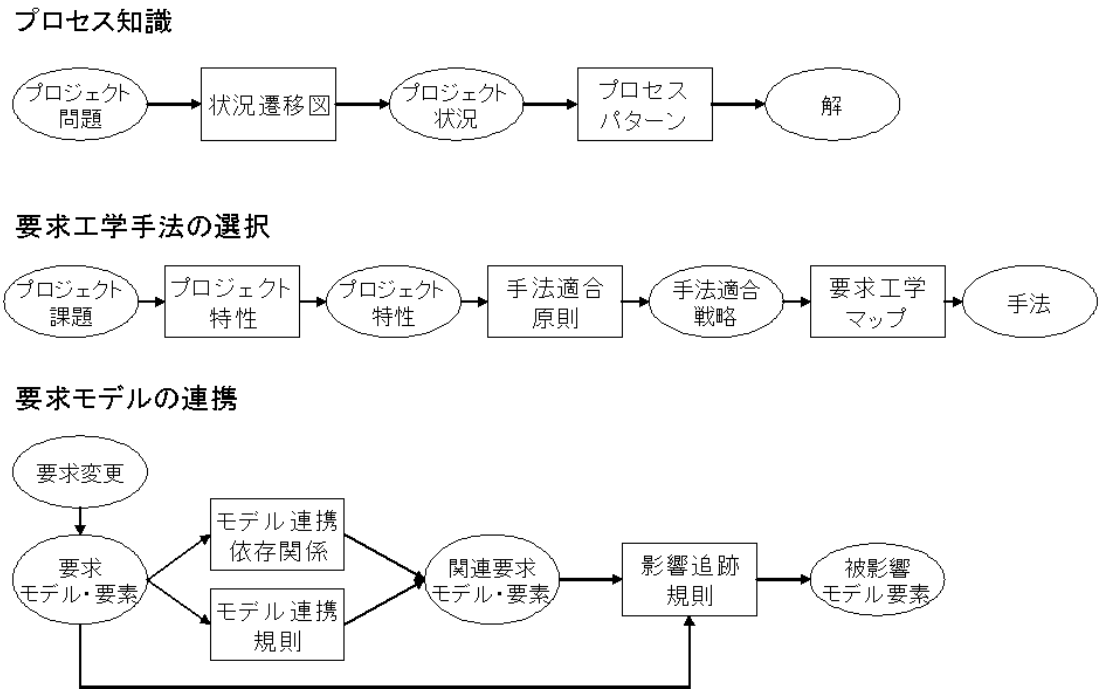

図-5.2　知識の利用形態

上記検討結果をもとに,図-5.3に要求工学支援ツールの論理構造を示す.要求技術者は,図-5.2に示した知識の利用手順に従って,問題を特定し,必要な解を得るものとする.支援ツールは,「プロセス知識」,「要求工学手法の選択」,「要求モデルの連携」などの知識ベースから成り,利用者の問題を特定し,適切な解を導き出す.それぞれの知識に固有の図や座標,あるいはテキストを使った

表現は，知識表現ベースとして参照可能となっている．

制御部は，ユーザインタフェース部を介してツールの利用者と会話し，利用者の抱えている問題を解決するのに必要な知識ベースを選択する．選択された知識ベースの条件部は，利用者から必要な情報を入手し，その利用者独自のプロジェクトモデルを作成する．結論部は，このモデルをもとに問題の解を求め，利用者にそれを提示する．

マルティメディア型の知識とルールベースの知識の連動や，新たな知識の追加機構など，解決すべき多くの課題が残っているが，教育的効果も視野に入れた支援ツールの実現については，今後の研究課題の1つである．

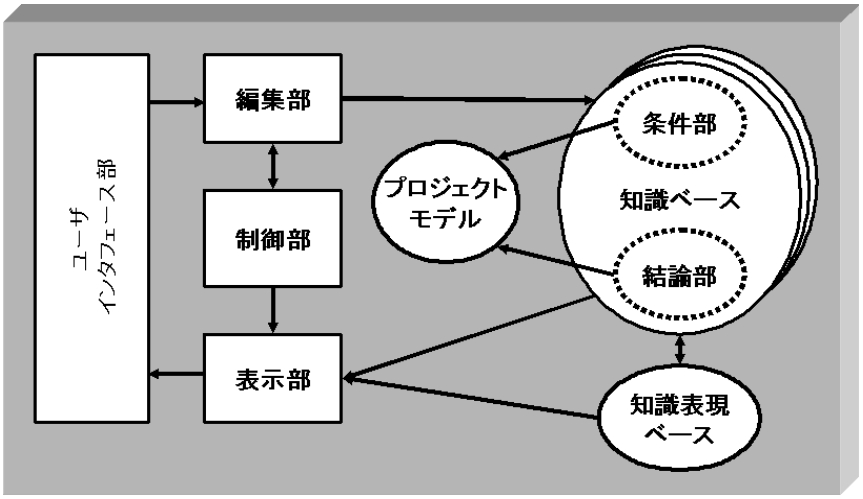

図-5.3 要求工学支援ツールの機能構造

5.4. 結び

本論文では，要求工学における，問題解決，手法選択，モデル連携という代表的な3つの問題における発見的知識と工学的知識を利用するためのフレームワークについて提案した．議論の対象となったこれらの知識は，要求工学プロセスの設計，実行，保守における代表的な知識である．こうした知識の活用については，本文中にも述べたように，偶発的アプローチとして多くの研究報告があるが，知識の体系化にまで言及した研究は少ない．発見的知識は，工学的な知識と

合体させることによって，要求技術者の抱える問題に適切に応えることが可能となる．本論文における知識フレームワークも，そうした総合的な知識の体系化を目指したものであり，知識の表現法にまで言及することによって，支援システムの可能性についても述べることができた．

要求工学は，その対象や作業形態から，学際的な色彩を持たざるを得ない領域であり，熟練した要求技術者の経験的知識が大きな比重を占め続けるだろう．発見的知識と工学的知識の融合と体系化およびその表現に関する本研究の意義がそこにあると考える．本研究で取り扱うことのできなかったその他の領域における知識の体系化は，支援システムの実現と共に，今後の研究課題とする．

参考文献

[1] Akao, Y. *Quality Function Deployment: Integrating Customer Requirements into Product Design*. Productivity Press, 1990.

[2] Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I. and Angel, S. *A Pattern Language*. Oxford University Press, 1977.

[3] Alexander, C. *The Timeless Way of Building*. Oxford University Press, 1979.

[4] Alexander, I, Robertson, S. and Maiden, N. What influences the requirements process in industry? *Proc. of 13th International Requirements Engineering Conference*, 2005, pp. 411-415.

[5] Ambriola, V. and Gervasi, V. Processing natural language requirements. *Proc. of 12th International Conference on Automated Software Engineering*, 1997, pp. 36-45.

[6] American Supplier Institute. *Quality Function Deployment: A Collection of Presentations and QFD Case Studies*. American Supplier Institute, 1986.

[7] Anton A. I. Goal-based requirements analysis. *Proc. of 2nd International Conference on Requirements Engineering*, 1996, pp. 136-144.

[8] Antoniou, G. The role of nonmonotonic representations in requirements engineering. *International Journal of Software Engineering and Knowledge Engineering* 8(3): 385-399, 1998.

[9] Bergman, M. and Mark, G. In situ requirements analysis: A deeper examination of the relationship between requirements determination and project selection. *Proc. of 11th International Requirements Engineering Conference*, 2003, pp. 11-22.

[10] Berlin, L.M. User-centered application definition: A methodology and case study. *Hewlett-Packard Journal* 40(5): 90-97, 1989.

[11] Boehm, B. *Software Engineering Economics*. Prentice Hall, 1981.

[12] Boehm, B. A spiral model of software development and enhancement. *IEEE Computer* 21(5): 61-72, 1988.

[13] Boehm, B. Software risk management: Principles and practices. *IEEE Software* 8(1): 32-41, 1991.

[14] Boehm, B., Bose, P., Horowitz, E. and Lee, M. Software requirements as negotiated win conditions. *Proc. of 1st International Conference on Requirements Engineering*, 1994.

[15] Boehm, B. and In, H. Identifying quality-requirement conflicts. *IEEE Software* 13(2): 25-35, 1996.

[16] Boehm, B. Requirements that handle IKWISI, COTS and rapid changer. *IEEE Computer* 33(7): 99-102, 2000.

[17] Bohner, S. and Arnold, R. *Software Change Impact Analysis*. IEEE Computer Society Press, 1996.

[18] Booch, G. *Object-Oriented Analysis and Design with Application*, Addison Wesley, 1994.

[19] Brinkkemper, S. and Joosten, S. Editorial: Method engineering and meta-modeling. *Information and Software Technology* 38(4): 259, 1996.

[20] Brooks, F.P. No silver bullet: Essence and accidents of software engineering. *IEEE Computer*, 20: 10-19, 1987.

[21] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M. *Pattern-Oriented Software Architecture - A System of Patterns*. John Wiley & Sons, 1996.

[22] Carroll, J.M. and Swatman, P.A. Managing the RE process: Lessons from commercial practice. *Proc. of 5th International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'99*, 1999, pp. 3-17.

[23] Cameron, J.R. Prototyping core functionality using JSD. *IEE Colloquium on*

Requirements Capture and Specification for Critical Systems (138): 2/1-2. 1989.

[24] Checkland, P. and Scholes, J. *Soft Systems Methodology in Action*. John Wiley & Sons, 1990.

[25] Christel, M.G. and Kang, K.C. Issues in requirements elicitation. *Technical Report CMU/SEI-92-TR-12*, 1992.

[26] Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[27] Clarke, J., Cross, O. and Peled, A. *Model Checking*. MIT press, 1999.

[28] Cockburn, A. *Writing Effective Use Case*. Addison Wesley, 2001.

[29] Coplien, J.O. *Software Patterns*. SIGS Books & Multimedia, 1996.

[30] Coplien, J.O. and Schmidt, D.C. *Pattern Languages of Program Design*. Addison Wesley, 1998.

[31] Coughlan, J. and Macredie, R. D. Effective communication in requirements elicitation: A comparison of methodologies. *Requirements Engineering Journal* 7(2): 47-60, 2002.

[32] Dano, B., Briand, H. and Barbier, F. An approach based on the concept of use case to produce dynamic object-oriented specifications. *Proc. of 3rd International Symposium on Requirements Engineering*, 1997, pp. 54-64.

[33] Dardenne, A., Lamsweerde, A.V. and Fickas, S. Goal directed requirements acquisition. *Science of Computer Programming* 20 (1-2): 3-50, 1993.

[34] Davis, G.B. Strategies for information requirements determination. *IBM Systems Journal* 21(1): 4-30, 1982.

[35] Davis, A.M. *Software Requirements: Analysis and Specification*. Prentice Hall, 1990.

[36] Davis, A.M. Operational prototyping: A new development approach. *IEEE Software* 9(5): 70-78, 1992.

[37] Delugach, H.S. *A multiple viewed approach to software requirements*. Ph.D. thesis, University of Virginia, 1991.

[38] DeMarco, T. *Structured Analysis and System Specification*. Prentice Hall, 1979.

[39] Deutsch, M.S. Focusing real-time systems analysis on user operations. *IEEE Software* 5(5):39-50, 1988.

[40] Domges, R. and Pohhl, K. Adapting traceability environments to project - Specific needs. *Communications of ACM* 41(12): 54-62, 1998.

[41] Dubois, E., Hagelstein, J. and Rifaut, A. Formal requirements engineering with ERAE. *Philips Journal of Research* 43(3/4): 393-414, 1988.

[42] Easterbrook, S. Resolving conflicts between domain descriptions with computer-supported negotiation. *Knowledge Acquisition: An International Journal* 3: 255-289, 1991.

[43] Easterbrook, S. and Nuseibeh, B. Managing inconsistencies in an evolving specification. *Proc. of 2nd International Symposium on Requirements Engineering*, 1995, pp. 48-55.

[44] Eriksson, H. and Penker, M. *Business Modeling with UML*. John Wiley & Sons, 2000.

[45] Estublier, J. Software configuration management: A roadmap. *Proc. of 22nd International Conference on Software Engineering - Future of Software Engineering*, 2000, pp. 279-289.

[46] Fagan, E. Design and code inspection to reduce errors in program development. *IBM systems journal* 15(3):182-211, 1976.

[47] Ferdinandi, P.L. *Requirements Pattern*. Addison Wesley, 2002.

[48] Fillmore, C.J. *The Case for Case*. in Bach and Harms (Eds.). Universals in Linguistic Theory, 1968.

[49] Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L. and Goedicke, M.

Viewpoints: A framework for integrating multiple perspectives in system development. *International Journal of Software Engineering and Knowledge Engineering* 2(1):31-58, 1992.

[50] Finkelstein, A. Requirements engineering: An overview. *Proc. of 2nd Asia-Pacific Software Engineering Conference*, 1993, pp. 152-164.

[51] Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J. and Nuseibeh, B. Inconsistency handling in multi-perspective specifications. *IEEE Transactions on Software Engineering*, 20: 569-578, 1994.

[52] FIPS183, *Federal Information Processing Standards Publication 183: Integration definition for Function Modeling (IDEF0)*. 1993.

[53] Fowler, M. *Analysis Patterns: Reusable Object Models*. Addison Wesley, 1997.

[54] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. *Design Patterns: Elements of Reusable Object- Oriented Software*. Addison Wesley, 1995.

[55] Ghezzi, C. and Nuseibeh, B. Guest Editorial - Managing Inconsistency in Software Development. *IEEE Transactions on Software Engineering* 24(11): 906-907, 1998.

[56] Goguen, J. and Linde, C. Techniques for requirements elicitation. *Proc. of 1st International Symposium on Requirements Engineering*, 1993, pp. 152-164.

[57] Grady, R. *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall, 1992.

[58] Grudin, J. and Pruitt, J. Personas, participatory design and product development: An infrastructure engagement. *Proc. of Participatory Design Conference*, 2002, pp. 144-161.

[59] Hagge, L. and Lappe, K. Pattern for RE process. *Proc. of 12th Requirements Engineering Conference*, 2004, pp. 90-99.

[60] Harmsen, F., Brinkkemper, S. and Oei, H. Situational method engineering for information system project approaches. *In Methods and Associated Tools*

for the Information Systems Life Cycle, pp. 169 - 194. Elsevier Science B.V., 1994.

[61] Heitmeyer, C.L., Jeffords, R.D. and Labaw, B.G. Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology* 5(3): 231-261, 1996.

[62] Hickey, A.M. and Davis, A.M. Elicitation technique selection: How do experts do it? *Proc. of 11th International Requirements Engineering Conference*, 2003, pp. 169-178.

[63] Holbrook, C.H. A scenario-based methodology for conducting requirements elicitation. *ACM SIGSOFT Software Engineering Notes* 15 (1): 95-104, 1990.

[64] Holzmann, G. The model checker Spin. *IEEE Transactions on Software Engineering* 23(5): 279-295, 1997.

[65] Hudlicka, E. Requirements elicitation with indirect knowledge elicitation technique: Comparison of three methods. *Proc. of 2nd International Conference on Requirements Engineering*, 1996, pp. 4-11.

[66] Hunter, A. and Nuseibeh, B. Managing inconsistent specifications: Reasoning, analysis and action. *ACM Transactions on Software Engineering and Methodology* 7(4): 335-367, 1998.

[67] Hutinsky, Z., Vrcek, N. and Bubas, G. *Communication Problems in Complex Information System Development Project*. University of Zagreb, 2001.

[68] IEEE, *Standard Glossary of Software Engineering Terminology*. Std 610.12-1990 (revision and redesignation of IEEE Std. 729-1983), 1983.

[69] IEEE, *Recommended Practice for Software Requirements Specifications*. Std 830-1998, 1998.

[70] Jackson, M. *System Development*. Prentice Hall, 1983.

[71] Jackson, M. *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*. Addison Wesley, 1995.

[72] Jackson, M. and Zave, P. Deriving specifications from requirements: An

example. *Proc of 17th International Conference on Software Engineering, 1996*, pp. 15-24.

[73] Jackson, M. *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison Wesley, 2001.

[74] Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G. *Object-Oriented Software Engineering*. Addison Wesley, 1992.

[75] Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G. *The Object Advantage*. Addison Wesley, 1995.

[76] Jacobson, I., Griss, M. and Jonsson, P. *Software Reuse*. Addison Wesley, 1997.

[77] Jacobson, I. Usecase and aspect - Working seamlessly together. *Journal of object technology* 2(4):7-28, 2003.

[78] Jarke, M. Requirements tracing. *Communications of ACM* 40(12): 32-36, 1998.

[79] Johnson, P. *Human-Computer Interaction: Psychology, Task Analysis and Software Engineering*. McGraw Hill, 1992.

[80] Johnson, R.E. Documenting frameworks using patterns. *Proc. of OOPSLA'92*, 1992, pp. 63-76.

[81] Jones, C.B. *Systematic Software Development using VDM, 2nd Edition*. Prentice Hall, 1990.

[82] Kaiya, H., Horai, H. and Saeki, M. AGORA: Attributed goal-oriented requirements analysis method. *Proc. of 10tn Requirements Engineering Conference*, 2002, pp. 13-22.

[83] Kang, C., Cohen, G., Hess, A., Novak, E. and Peterson, A. Feature-oriented domain analysis (FODA) feasibility study. *Technical Report CMU/SEI-90-TR-21*, ADA235785, Software Engineering Institute, 1990.

[84] Kawakita, J. *The Original KJ Method*. Kawakita Research Institute, 1991.

[85] Kramer, J., Ng, K., Potts, C. and Whitehead, K. Tool support for requirements analysis. *Software Engineering Journal* 3(3):86-96, 1988.

[86] Kruchten, P. *The Rational Unified Process*. Addison Wesley, 1998.

[87] Lamsweerde, A.V. Goal-oriented requirements engineering: A guided tour. *Proc. of 9th International Symposium on Requirements Engineering*, 2001, pp. 249-263.

[88] Lévi-Strauss, C. *Tristes Tropiques*. Plon, 1955.

[89] Loucopoulos, P. and Champion, M. Knowledge-based support for requirements engineering. *Information and Software Technology* 31(3):123-135, 1989.

[90] Loucopoulos, P. and Kavakli, E. Enterprise modelling and the teleological approach to requirements engineering. *International Journal of Intelligent and Cooperative Information Systems* 4(1): 45-79, 1995.

[91] Mader, M. *Designing Tool Support for Use-Case-Driven Test Case Generation*. Di-plomarbeit, FHTW Berlin, 2004.

[92] Maiden, N. and Sutcliffe, A. Exploiting reusable specifications through analogy. *Communications of the ACM* 34(5): 55-64, 1992.

[93] Maiden, N. and Rugg, G. ACRE: Selecting methods for requirements acquisition. *Software Engineering Journal* 11(3): 183-192, 1996.

[94] Maiden, N. CREWS-SAVRE: Scenarios for acquiring and validating requirements. *Automated Software Engineering* 5(4): 419-446, 1998.

[95] Marshall, C. *Enterprise Modeling with UML*. Addison Wesley, 2000.

[96] Meyer, B. *Object-Oriented Software Construction*. Prentice Hall, 1988.

[97] Mohagheghi, P., Anda, B. and Conradi, R. Effort estimation of use cases for incremental large-scale software development. *Proc. of 27th International Conference on Software Engineering*, 2005, pp. 303-311.

[98] Mylopoulos, J., Chung, L. and Nixon, B. Representing and using

nonfunctional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering* 18 (6): 483-497, 1992.

[99] Naumann, J., Davice, G. and Mckeen, J. Determining information requirements: A contingency method for selection of a requirements assurance strategy. *The Journal of Systems and Software* 1: 273-281, 1980.

[100] Neighbors, J. The Draco approach to constructing software from reusable components. *IEEE Transactions on Software Engineering* 10(5): 564-573, 1984.

[101] Norman, D. A. and Draper, S. W. *User-Centered System Design: New Perspectives on Human-Computer Interaction*. Lawrence Earlbaum Associates, 1986.

[102] Nuseibeh, B., Kramer, J. and Finkelstein, A. A framework for expressing the relationships between multiple views in requirements specifications. *IEEE Transactions on Software Engineering* 20(10): 760-773, 1994.

[103] Nuseibeh, B., Finkelstein, A. and Kramer, J. Method engineering for multi-perspective software development. *Information and Software Technology Journal* 38(4): 267-274, 1996.

[104] Nuseibeh, B. Ariane 5: Who Dunnit? *IEEE Software* 14(3): 15-16, 1997.

[105] Nuseibeh, B. and Easterbrook, S. Requirements engineering: A roadmap. *Proc. of 22nd International Conference on Software Engineering - Future of Software Engineering*, 2000, pp. 35-46.

[106] Ohnishi, A. Software requirements specification database based on requirements frame model. *Proc. of 2nd International Conference on Requirements Engineering*, 1996, pp. 221-228.

[107] 大西淳　要求工学ワーキンググループ活動報告. *情報処理学会研究報告, ソフトウェア工学研究会*-No.130: 127-134, 2001.

[108] Padula, A. Requirements engineering process selection at Hewlett - Packard. *Proc. of 12th International Requirements Engineering Conference*, 2004, pp. 296-300.

[109] Parnas, D.L. A technique for software module specification with examples. *Communications of the ACM* 15(5):330-336, 1972.

[110] Parnas, D.L. and Madey, J. Functional documentation for computer systems. *Science of Computer Programming* 25(1): 41-61, 1995.

[111] Pinherio, F. and Goguen, J. An object oriented tool for tracing requirements. *IEEE Software* 13(2): 52-4, 1996.

[112] Potts, C., Takahashi, K. and Anton, A. Inquiry-based requirements analysis. *IEEE Software* 11(2): 21-32, 1994.

[113] Potts, C. Requirements models in context. *Proc. of 3rd International Symposium on Requirements Engineering*, 1997, pp. 102-104.

[114] Punter, T. and Lemmen, K. The MEMA-model: towards a new approach for method engineering. *Information and Software Technology* 38: 295-305, 1996.

[115] Rieto-Di´az, R. Domain analysis: An introduction. *ACM SIGSOFT Software Engineering Notes* 15 (2): 47-54, 1990.

[116] Robertson, S. and Robertson, J. *Mastering the Requirements Process*. Addison Wesley, 1999.

[117] Robinson, W.N. and Volkov, S. Supporting the negotiation life-cycle. *Communications of the ACM* 41(5): 95-102, 1998.

[118] Rolland, C., Souveyet, C. and Achour, C. B. Guiding goal modeling using scenarios. *IEEE Transaction on Software Engineering* 24 (12): 1055-1071, 1998.

[119] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[120] Russo, A., Miller, R., Nuseibeh, B. and Kramer, J. An abductive approach for analysing event-based requirements specifications. *Proc. of 18th International Conference on Logic Programming*, 2002, pp. 22-33.

[121] Rzepka, W.E. A Requirements engineering testbed: Concept, status, and first

results. In Bruce D. Shriver (editor), *Proc. of 22nd Annual Hawaii International Conference on System Sciences*, 1989, pp.339-347.

[122] Saaltink, M. The Z/EVES System. *Proc. of 19th International Conference on the Z Formal Method*, 1997, pp. 72-88.

[123] Saaty, T.L. *The Analytic Hierarchy Process*. McGraw Hill, 1980.

[124] Sage, A.P. and Palmer, J.D. *Software Systems Engineering*. John Wiley & Sons, 1990.

[125] Savolainen, V. A dynamic framework of reference of information systems development. Dynamic modelling of information systems II (eds. Sol H G, Grosslin R L). *Elsevier Science Publishers*: 285-307, 1992.

[126] Schwaber, K. and Beedle, M. *Agile Software Development with SCRUM*. Prentice Hall, 2001.

[127] Sharp, H., Finkelstein, A. and Galal, G. Stakeholder identification in the requirements engineering process. *Proc. of Workshop on Requirements Engineering Processes - DEXA*, 1999, pp. 387-391.

[128] Shaw, M. Prospects for an engineering discipline of software. *IEEE Software* 7(6): 15-24, 1990.

[129] Shaw, M. and Gaines, B. Requirements acquisition. *Software Engineering Journal* 11(3): 149-165, 1996.

[130] Southwell, K., James, K., Clarke, B., Andrews, B., Ashworth, C., Norris, M. and Patel, V. *Requirements Definition and Design. The STARTS Guide, 2nd Edition*. National Computing Centre, 1987.

[131] Sowa, J.F. and Zachman, J.A. Extending and formalizing the framework for information systems architecture. *IBM System-J* 31(3): 590- 616, 1992.

[132] Spivey, J.M. *The Z Notation--A Reference Manual 2nd Edition*. Prentice Hall, 1992.

[133] Standish Group *Chaos*. Standish Group Inc, 1994.

[134] Stewart, D.W. and Shamdasani, P.N. *Focus Groups: Theory and Practice*. Sage, 1990.

[135] Sullivan, C.H. Systems planning in the information age. *Sloan Business Review* 26(2): 3-11, 1985.

[136] Systems Designers Scientific. *CORE: The Method*. Systems Designers Scientific, 1985.

[137] Takeda, N., Shiom, A., Kawai, K. and Ohiwa, H. Requirements analysis by KJ editor. *Proc. of 1st International Symposium on Requirements Engineering*, 1993, pp. 98–101.

[138] 玉井哲雄 *ソフトウェア工学の基礎*, 岩波書店, 2004.

[139] Tamai, T., Ubayashi, N. and Ichiyama, R. An adaptive object model with dynamic role binding. *Proc. of 27th International Conference on Software Engineering*, 2005, pp.166-175.

[140] Tarr, P., Ossher, H., Harrison, W. and Sutton, S. N degrees of separation: Multi-dimensional separation of concerns. *Proc of 21st International Conference on Software Engineering*, 1999.

[141] Tsumaki, T. and Morisawa, Y. A framework of requirements tracing using UML. *Proc. of 7th Asia-Pacific Software Engineering Conference*, 2000, pp. 206-213.

[142] Tsumaki, T. Requirements engineering pattern structure. *Proc. of 11th Asia-Pacific Software Engineering Conference*, 2004, pp. 502-509.

[143] Tsumaki, T. and Tamai, T. A framework for matching requirements engineering techniques to project characteristics. *Software Process: Improvement and Practice* 11(5): 505-519, 2006.

[144] Verner, J., Cox, K., Blerstein, S. and Cerpa, N. Requirements engineering and software project success: An industrial survey in Australia and the U.S. *Proc. of 9th Australian Workshop on Requirements Engineering*, 2004, pp. 225-238.

[145] Viller, S. and Sommerville, I. Social analysis in the requirements

engineering process: From ethnography to method. *Proc. of 4th International Symposium on Requirements Engineering*, 1999, pp. 6-13.

[146] Vlasblom, G., Rijsenbrij, D. and Gastra, M. Flexibilization of the methodology of system development. *Information and Software Technology: Elsevier-Science B.V.* 37 (11): 595-607, 1995.

[147] Wooldridge, M., Jennings, N. and Kenny, D. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agent and Multi-Agent Systems* 3: 285-312, 2000.

[148] Yadav, S.B., Bravoco, R.R., Chatfield, A.T. and Rajikumar, T.M. Comparison of analysis techniques for information requirement determination. *Communication of ACM* 31 (9): 1090-1097, 1988.

[149] Yu, E. Towards modelling and reasoning support for early-phase requirements engineering. *Proc. of 3rd International Symposium on Requirements Engineering*, 1997, pp. 226-235.

[150] Zachman, J.A. A framework for information systems architecture. *IBM System-J* 26(3): 276-292, 1987.

[151] Zahniser, R.A. How to speed development with group sessions. *IEEE Software* 7(3): 109-110, May 1990.

[152] Zave, P. An operational approach to requirements specification for embedded systems. *IEEE Transactions on Software Engineering* 8(3): 250-269, 1982.

[153] Zeroual, K. An approach for automating the specification - acquisition process. *Proc. of 2nd International Workshop on Software Engineering and its Applications*, 1989, pp. 349-355.

[154] Zhao, J. Slicing aspect-oriented software. *Proc. of 10th International Workshop on Program Comprehension*, 2002, pp. 251-260.

[155] Zucconi, L. Techniques and experiences capturing requirements for several real-time applications. *ACM SIGSOFT Software Engineering Notes* 14(6): 51-55, 1989.