要求抽出の時期を考慮した
プロジェクトマネジメント・パターン
に関する研究

２０１２年　３月

堀　　昭　三

目次

図目次

表目次

1. はじめに

1.1. 研究の背景と動機

近年，インターネントネット，携帯電話や携帯端末などに代表されるようにＩＴ技術の進化は著しいものがある．それに伴い，企業の情報システムや製品開発への投資はうなぎのぼりに増大傾向にある[1]．その情報化投資の増大傾向の中で，ある企業では本番稼動後のシステムに深刻な障害が発生し，業務に多大な損害が発生したとしてユーザがベンダーを訴えるケースや，開発スケジュール遅れによる製品リリース時期の延期およびシステムのソフトウェア品質などに対する不満から開発費の支払いを元請けが拒否した際，支払いを求めて下請けが元請けを訴えるケースなど，社内的な問題にもなっている[2]．その問題の要因は，需要の増大に伴い，生産能力が追いついていないため品質低下になっているからである．また，ソフトウェア開発への要求は，高機能で複雑化する（大規模化する）反面，高品質，短納期，低コストの開発が要求され，ほとんどの開発プロジェクトは計画どおりに進んでいない. 1995年から2009年までの米国のStandish Group Report[3]によると，1994年のソフトウェア開発プロジェクトの31%がFailed（開発途中で中断），53%がChallenged（問題はあるが最後まで継続）であり，Successful（成功）はたったの16%であったが, 2008年のソフトウェア開発プロジェクトでは, 24%がFailed, 44%がChallenged, Successfulは32%であり，Successfulが2倍に改善している．しかし，依然としてソフトウェア開発プロジェクトの失敗（FailedとChallengedを加えたもの）は約70%で高い数値を示している．その12年間のソフトウェア開発プロジェクトの成否状況の推移を図1.1に示す. このReportでのプロジェクトの成功の判断基準は，プロジェクトの目標を予定通りの期限と予算内で達成したかどうかで判断されている．また，Standish Group Report[3]からプロジェクト失敗要因を抽出し表1.1に示す．この表では，プロジェクト失敗の要因の中に，Incomplete Requirements（要求の不備）が13.1%，Lack of User Involvement（顧客関与の欠落）が12.4%，Changing Requirements & Specifications（要求仕様の変更）が8.7%であり，要求に関連する失敗要因が全体の約35%を占める．つまり，要求の不備や開発途中での要求変更はプロジェクトに大きな影響を与えていることになる．ソフトウェア開発の要求定義のアウトプットである要求仕様の完成度の良し悪しが，見積もり精度，計画精度，成果物の品質，納期，コストに影響を与えていることは言うまでもない事実である．

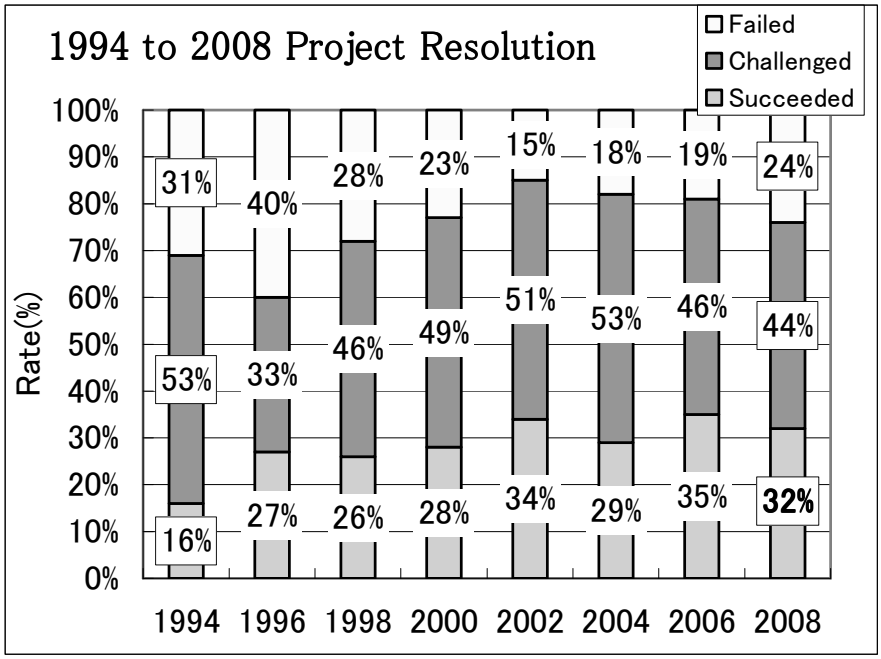


図 1.1　ソフトウェア開発プロジェクトの成否状況（1994-2008）

更には, Lack of Planning（プロジェクト計画の不備）が8.1%, Lack of Executive Support（管理職の支援不足)が9.3%，Lack of Resources（開発要員の不足)が10.6%など，プロジェクトマネジメントの不十分さも全体の約31%あり大きな割合を占めている.

表 1.1　プロジェクト失敗の要因

No	Projtct Impaired Factors	% of the Responses
1	Incomplete Requirements	13.1%
2	Lack of User Involvement	12.4%
3	Lack of Resources	10.6%
4	Unrealistic Expectations	9.9%
5	Lack of Executive Support	9.3%
6	Changing Requirements & Specifications	8.7%
7	Lack of Planning	8.1%
8	Didn't Need IT Any Longer	7.5%
9	Lack of IT Management	6.2%
10	Technology Illiterracy	4.3%
11	Other	9.9%
Total		100.0%

多くのソフトウェア開発のプロジェクトにおいて，要求分析工程が完了した後，設計，コーディングおよびテスト工程においても，ソフトウェアの要求変更や追加が抽出されることがある．“ソフトウェア開発データ白書2009”[4]によると，2,230件のプロジェクトの内，約90%強のプロジェクトにおいて，ウォータフォール型開発プロセスが実践され，開発工程の早い時期に要求抽出することが望ましいと考えられている．しかし，多くのプロジェクトにおいて，全ての要求が開発の早い時期に決まらず，開発途中での突発的な要求抽出や開発の遅い時期の要求抽出が生じている．プロジェクトの開発途中の要求変更および追加は，プロジェクトのスケジュール遅れの最も重要な原因の１つであり[3]，プロジェクトの担当者は受け入れたくないものである．しかし，顧客の目標および満足度を達成するためには，より品質の高いシステム構築を目指し，要求分析工程が完了した後，開発途中であっても要求を積極的に取りにいかなければならない[5]．

要求の追加や変更も，それらを受け取る時期を誤らなければプロジェクトが遅延することを避けることが可能であるはずである．そこで，本論文において，スケジュールおよび品質に関して比較的うまくいった実際のソフトウェア開発プロジェクトの要求抽出プロセスやプロジェクトマネジメントがどのように実施されていたかを明らかにすることは，スケジュール遅延を防ぐための解決策の一つに繋がると期待している．ソフトウェア開発のプロジェクトは全く同じものはないが，効率的，かつ効果的なプロジェクトマネジメントを実践する上で要求抽出の進め方，抽出の時期は重要なポイントである．

本論文では，その要求抽出に伴うプロジェクトマネジメント技術を再利用できるようにプロジェクトマネジメント・パターンとして整理し，適用方法を提案する．今後のソフトウェア開発プロジェクトを上手くマネジメントするためにプロジェクトマネジメント・パターンを提案することがこの研究の目的である．

1.2. 関連研究

本節では，従来の開発プロセスにおける要求抽出の問題・課題を述べる．これらの問題・課題をすべて解決できるわけではないが，ひとつの解決策として，本論文では，要求抽出プロセスと開発プロセスを統合した統合型要求工学に基づく開発プロセスを推奨する．その統合型要求工学の開発プロセスの研究の一例をこの節で述べる．

1.2.1. 従来の開発プロセスの概要と課題

従来のソフトウェア開発の開発プロセスモデルの中から，5 つの開発プロセスモデルの特徴と問題・課題を以下に述べる．

1）ウォータフォール型開発プロセス

プロジェクト全体を「要求定義」「外部設計（概要設計)」「内部設計（詳細設計)」「開発（プログラミング)」「テスト」「運用」などの作業工程に分割し，原則として前工程が完了しないと次工程に進まないことが前提であり，前工程の成果物の品質を確保し、前工程への後戻り（手戻り）を最小限にする開発プロセスである[6].

問題点は，工程の後戻りができない（つまり，前工程の完全性が求められる)，開発途中での要求は著しく変わらないことが前提であるため，要求抽出に失敗した場合，開発作業の手戻りが発生し，プロジェクトの遂行に多大な影響を与えることである．現実的に，システム化が初めての場合やシステムが複雑な場合，要求定義フェーズですべての要求を洗い出すことは困難である．

2）プロトタイプ型開発プロセス

開発の早い段階で試作品（プロトタイプ）を作成し，その試作品をエンドユーザが確認，評価することにより，システムの仕様を確定していく開発プロセスである[7].

問題点は，試作品の機能の絞込みを小さくしなければ，開発コストが高くなることである．そのため，どの機能範囲に絞り込んで試作するかが重要ポイントである．

3）スパイラル型開発プロセス

独立性の高いサブシステムごとにシステム設計，プログラミング，テストの工程を繰り返しながら，徐々に開発範囲を拡大し，改良を加えて全体を完成させる開発プロセスである[8]．つまり，ウォータフォール型開発プロセスと反復型のプロトタイプを結合させたリスク中心型の開発プロセスモデルである．

問題点は，要求抽出の「誤り」による後工程での手戻り作業を防止するようにリスク対策するが，開発途中での要求変更，追加など，想定外である．そのため，要求変更が発生するとスパイラルでも多大な手戻りになることである．

4）アジャイル開発プロセス

最初に要求を確定して開発を進めるのではなく，実際に機能するソフトウェアを早期に検証しながら要求仕様をつめ，その後の要求変更を積極的に取り込み，実装を繰り返し，良いものを素早く無駄なく作る開発手法である[9].

問題点は，要求仕様が開発開始時に決まっていないため，費用とスケジュールの確定が困難であり，プロジェクトの進捗状況を把握しにくいことである．また，この開発手法は高い開発力が求められるため，技術者の確保が困難である．

５）統一プロセス（Rational Unified Process：ＲＵＰ）

ユースケース駆動，アーキテクチャ中心，反復型およびインクリメンタル型などを考慮した開発プロセスであり，大きくは４つの開発フェーズ(即ち，方向付け，推敲，構築，移行)からなる[10]．繰り返しの中で動くプログラムを開発し，その過程で，技術的な不確定要素，要求の変更，不明瞭さを順次解消していくというアプローチである．

ウォータフォール型やスパイラル型の開発プロセスのような開発途中で頻繁に発生する要求変更や技術リスクに対処するために，アジャイル開発や統一プロセスのような繰り返し型の開発プロセスが提唱されている．しかし，アジャイル開発や統一プロセスなど文献に書かれているような開発プロセスがあるが，実際の開発現場では混沌として使用されている．このように従来の開発プロセスには利点，欠点があるが，第 1.1 節で述べたように，日本国内では，未だに 90%強のソフトウェア開発プロジェクトがウォータフォール型開発プロセスを適用し，大半のプロジェクトは失敗している[4]．今でもウォータフォール型開発が推奨される理由として，定義が簡単で理解しやすい，進捗管理が簡単という幻想（例えば,「要求抽出か完了した」と言い切れる単純さ）などがあるが，理想的ではないという根拠が集まっているということに気づいていない[11]．本論文では，これを解決するために，次の第 1.2.2.項で記述する統合型要求工学に基づく開発プロセスを推奨する．

1.2.2. 統合型要求工学による開発プロセス

第 1.1 節の研究の背景で述べたように，ソフトウェア開発プロジェクトを失敗させる要因の上位に要求仕様の不備，要求変更が挙げられている．開発中のソフトウェアに対する要求は，開発の初期工程にだけに閉じたものではなく開発プロセス全体に渡って，ダイナミックに変化，発展しつづけると理解されている．したがって，これらの変化，発展しつづける要求を獲得，管理し，進行中のソフトウェア開発に速やかにかつ適切に反映させることが，要求工学における重要な課題の一つになっている[12]．Ian Sommerville は，統合型要求工学において，要求プロセスとそれ以降の開発プロセスとを分離するのではなく，これらのプロセスを統合すべきだと主張している[13]．つまり，統合型要求プロセスは，要求抽出プロセ

スと，設計，実装，テストからなる開発プロセスを統合した開発プロセスである．しかし，文献では，具体的な手法までは言及していない．第 1.2.1 項で述べた従来の開発プロセスでは，要求抽出プロセスと開発プロセスが分離され，いろいろな課題があった．本論文では，それらの課題を解決する策として統合型要求プロセスの実践を提案する[14].

本論文で提案するひとつの統合型要求プロセスは，比較的うまくいった実際の開発プロジェクトの実態調査結果から得られた知見を基に，アジャイルや統一プロセスを取り入れ，統合型要求工学による要求抽出プロセスとプロジェクトマネジメント技術を体系化したものである．具体的には，以下のような開発プロセスであろうと定義した（図 1.2）.

- ソフトウェアの構成要素の特徴，品質特性に応じて，適切な開発プロセスを取り込むことが可能な統合型の開発プロセス．（アジャイル，RUP などを取り込む）
- 顧客とのコミュニケーションを重要視した顧客参加型の開発プロセス．（JAD(Joint Application Development), PD(Participately Design)[15]を取り込む）.
- 要求変更に耐え得るソフトウェアアーキテクチャを意識する．つまり，アーキテクチャ重視の開発プロセス.
- 開発者主導で必要な要求を適切な時期に計画的に抽出する開発プロセス.

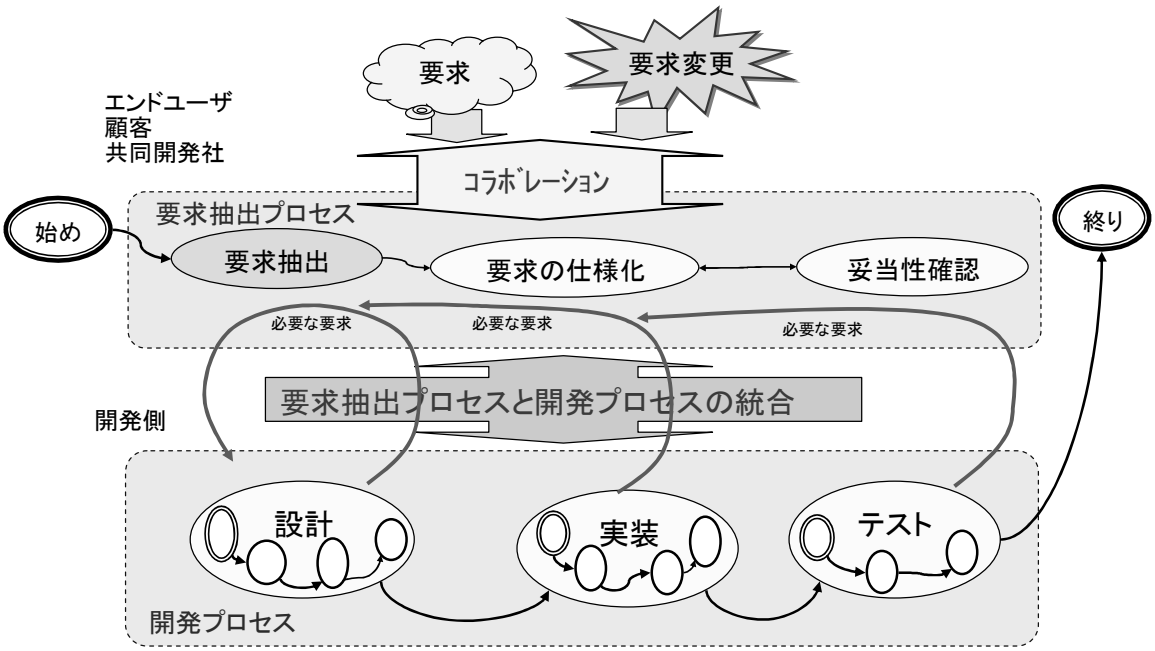


図 1.2　統合型要求工学による開発プロセス

このような統合型要求プロセスが，実際にどのように管理されるのか，また，どのようなリスクがあるのか，抽出する要求の種別と，それを抽出すべき時期との関係，あるいはソフトウェアアーキテクチャとの関係は，まだ明らかにされていない．そこで，本研究では，実際の開発現場で要求がどのように抽出されているかを明らかにするために，ある会社が開発した“レストラン注文管理システム”を実例プロジェクトとして採り上げ，その開発の要求抽出の実態を調査した[14][16][17]．その調査結果は，第 2 章にて述べる．

統合型要求工学プロセスの研究の一種として，中谷等の研究グループによって，「The PRINCE（Pre Requirements Intelligence Net Consideration and Evaluation）モデル（以降，PRINCE モデル）」が，産学戦略的研究フォーラムから支援を請け，2007 年度から 2009 年度までの 3 年間の研究成果として開発された[18]．PRINCE モデルを構成する要素は以下の３つである．

① 要求抽出プロセスを観測するためのガイドライン

② プロジェクトマネジメントのためのパターン

③ 要求の観測と評価のためのツール

の構成からなる．本論文の研究成果はその PRINCE モデルの①，②に取り入れられている．

1.2.3. プロジェクトマネジメント・パターン

本論文におけるプロジェクトマネジメント・パターン（以降，Project Management パターン：PM パターンと呼ぶ）の研究は，全ての建物はユニークであるが個々は一般的なパターンの集まりからなるという Alexander のパターンの概念[24]をベースにしている[19][21][22][23]．ソフトウェア開発ではデザインパターンの文献[25][26]が多く出ているが，プロジェクトマネジメントに関する文献はプロジェクトマネジメント知識体系（A Guide to the Project Management Body of Knowledge：PMBOK）[27]に代表されるように知識体系はあるがベストプラクティスを提案している訳ではない．それ故に，PMBOK をどのように活用してプロジェクトの問題を解決すればよいのか，あるいは，解決できるのかは不明である．また，IT プロジェクトの失敗学から教訓を整理した文献もある[28]がパターンの概念を取り入れたものは多くない．強いてあげるならば，ソフトウェア技術やソフトウェア開発プロジェクトが実際に抱える問題を摘出し，その問題と解決策をアンチパターンとして示した書物はある[29]．これは失敗から得られたソフトウェア開発の教訓

をパターン化している．第 1.3 節で述べるが，本研究の目的は，ソフトウェア開発途中での要求抽出を開発者主導で計画的にコントロールしスケジュール遅延を防ぐためのプロジェクトマネジメント技術を再利用できるように PM パターンとして整理しその活用方法を提案することである．しかし，そのような PM パターンは主に大学において研究されているが実践向きではない．産業界では，プロジェクトマネジメント技術がベストプラクティスや教訓として整理され，再利用が推進されているがうまくいっていない．その原因は，問題や解決策に対する背景や適用事例の整理不足，適用手順の不備にある．以下に，大学関係および産業界の研究事例と本論文の研究との関連性を述べる．

古市等の研究グループはプロジェクトマネジメント分野に Alexander が提唱する街づくりや建築デザインなどの創造過程で暗黙的に使われてきたノウハウや特定の価値観に基づく指針を知的体系として形式化したパターン・ランゲージ手法の適用を提案している[30][31]．彼等の研究では，プロジェクトマネジメント関連の文献からプロジェクトマネジメントのノウハウを抽出し，問題と解決策を PM パターンとしている．しかし，その有効性の評価は大学生へのアンケートによって行われており，開発現場で活用できる実践向きの PM パターンではない．本論文では，実際のプロジェクトの開発データからクリティカルな問題を抽出し，要求抽出に関連するプロジェクトマネジメントのノウハウを再利用できるように PM パターンとして整理し，他の実際のプロジェクトでその有効性を評価する. 更に, PM パターンをプロジェクトに適用する上で効果的なフレームワークを開発し，それを使って PM パターンの妥当性を評価する[23].

Andrew Hatch 等は大学生の問題に基づく学習（PBL Problem Based Learning）から得られたデータを使って研究している[32]．学生にプロジェクトマネジャの活動を理解させるために PM パターンは適切な方法であるが，PM パターンの抽出というより，PM パターンを用いて討議することによって，より理解を深めることに主眼をおいている．

Stamelos は, ソフトウェア開発のプロジェクトマネジメントを上手く行うためのリーダシップに着目し，PM パターンやアンチパターンについて調査研究をしている[33]．アンチパターンは，現実に存在する問題に対する不適切な解決策を分類したものであり，主に失敗した開発プロセスに焦点を当てて失敗に陥るパターンを整理している[34].

山本は実際のプロジェクト活動を通じて学んだことを教訓としてまとめており，中小規模のプロジェクト教訓や進め方を提案している[35]．プロジェクトマネジメント知識体系（A Guide to the Project Management Body of Knowledge：PMBOK）[27]をはじめとする文献において，“教訓”を纏めることが薦められているが，その手法・進め方を具体的に示したものが少ないため，開発現場のプロジェクトマネジャにとっては参考になる提案で

ある．しかし，山本が提案した教訓は PM パターンに近いものであるが，プロジェクトマネジメントの観点から教訓の客観性を評価しておらず，単に教訓を集めただけになっている．これでは利用者は活用できない．利用者が活用する時に必要なキー項目が何か，何故，この解決策が妥当なのかを整理し，その項目に基づいて整備されるべきである．本論文では，PMBOK の 9 つの知識体系と 5 つのプロジェクトマネジメント・グループを取り入れたフレームワークを定義し，そのフレームワークを使って，抽出した PM パターンの有用性を評価する．

村上等の研究グループはプロジェクトの成功体験から PMO(Project Management Office)やプロジェクトマネジャの立場で役に立つ情報を「ベストプラクティス」として，SECI(Socialization, Externalization, Combination, Internalization)モデルと比較・評価している[36]．SECI モデルは知識変換の 4 種類のモード“共同化(Socialization)”，“表出化(Externalization)”，“連結化(Combination)”，“内面化(Internalization)”を通じ，個人の知識からプロジェクトの知識へとスパイラルしながら増幅していくものである[37]．その SECI モデルとベストプラクティスを比較評価しているが，ベストプラクティスの整理の仕方，開発現場での活用の仕方が明示されていないため，開発現場のプロジェクトマネジャには殆ど活用されていない．研究としては，まだまだ発展途上の段階である．

以上の関連研究の状況から，本論文では，要求抽出によるスケジュール遅延を防ぐために，実践的で且つ実用的な産業界のプロジェクトマネジメント技術を再利用できるように PM パターンとして整理し，これらが実践的で有効なものであることを検証する[19][21][22][23]．更に，PM パターンを適切に活用するためのフレームワークを開発，提案しプロジェクトの重要な問題を解決する手順を提案する．

1.3. 本研究の目的とゴール

第 1.1 節で述べたように，ソフトウェア開発のプロジェクトにおいて，開発途中の要求変更や追加はスケジュール遅延の要因となることが多く，その要求変更・追加の抽出時期と内容によっては，プロジェクトへの影響度合いも変わってくる．顧客の要求を満足するためには開発途中の要求変更の受け入れを避けて通ることはできない[5]．それならば，要求変更の受け入れ時期，その要求変更への対処方針決め，方法など，計画的かつ積極的に要求を取り込むプロセスとそのプロジェクトマネジメントが解決へのポイントと考える．そこで，本論文の研究の目的は，実際のソフトウェア開発現場のプロジェクトにおいて，

どのような要求がどのようなプロセスで抽出され，どの時期でどのようなプロジェクトマネジメントが実施されているのかを明らかにすることである．さらに，その調査結果から得られた知見を整理し，プロジェクトに活用する方法，仕組みを提案することによって，開発途中での要求変更によるプロジェクトのスケジュール遅延を防止することに貢献できると期待する．具体的には，それらの知見をベースに要求抽出に伴うプロジェクトマネジメント技術を再利用できるように PM パターンとして整理し，それらを活用する仕組みとしてプロジェクトマネジメントのフレームワークを開発し提案することである．開発現場のプロジェクトマネジャが，このフレームワークを用いることによって，要求抽出する時期を計画し，要求の抽出を監視し，制御するための PM パターンを適切に選択でき，プロジェクトの問題を適切に解決できるようになると期待する．

1.4. 本論文の構成と概要

本論文の構成と概要は以下の通りである．

第 2 章では，本研究活動のきっかけとなった実例プロジェクトの背景や概要を説明し，開発現場で実践されている要求抽出プロセスの実態調査の結果を述べる．この実態調査では，プロジェクトの開発履歴からソフトウェア品質特性や機能構成要素ごとの要求抽出の件数や内容を調査し分析する．その上で，開発のプロジェクトマネジャやリーダへの直接インタビューを行うことによって，要求抽出プロセスを明らかにし，エンジニアリング技術，プロジェクトマネジメント技術などの知見を整理する．また，関連研究で述べた統合型要求抽出プロセスとの関連性や合理性を考察する．

第 3 章では, 第 2 章での要求抽出の分析視点を変えて開発経過日数と要求抽出の割合（即ち，成熟度）から分析する．その上で，ソフトウェアの品質特性や機能構成要素によって要求抽出プロセスの成熟度が異なり，大別して 3 つの要求抽出プロセスの成熟度タイプがあることを明らかにする．その 3 つの成熟度タイプの定義付けを述べ，要求変更によるスケジュール遅れを防止するためのプロジェクトマネジメントとの関連を考察する．

第 4 章では，第 3 章の要求抽出プロセスの分類の結果から，スケジュール遅延を防ぐためには，プロジェクトマネジャの役割の中で，ソフトウェア構成要素毎に要求抽出の時期を計画し，必要な時期に適切に要求抽出をすることが重要であることを明らかにする．その成熟度タイプに伴う典型的なプロジェクトマネジメント技術を整理する．

第 5 章では，第 4 章で得られた要求変更が原因でプロジェクトのスケジュール遅れを防止するための知見，すなわち，プロジェクトマネジメント技術を Alexander のパターンの概念[24]をベースに PM パターンとして整理する．また，これらの PM パターンが他の実際のプロジェクトでも普遍的に使用されているかを検証し，有用性について考察する．

第 6 章では，新たな PM パターンの抽出や PM パターンをプロジェクトに適用するための手段として，PM パターンのフレームワークを定義し，このフレームワークの構成を述べる．プロジェクト目標を問題に分解後の PM パターンの選択ならびにフレームワークの活用法について述べる．

最後の第 7 章では，本論文の全体のまとめと今後の研究課題と方向性について述べる．残された研究課題は多々あるが，PM パターンの拡充や PM パターンのフレームワークの改善並びに PM パターンの定量的評価と継続的な開発に伴う要求抽出プロセスのモデル化が主なテーマと認識している．その課題認識と方向性について整理し，本論文のまとめとする．

2. 要求抽出プロセスの実態調査

開発途中の要求変更はスケジュール遅延の大きな原因として指摘されることが多い．しかし，技術革新の速度やシステムの環境変化の速度は早く，顧客の要求を満足するよいシステムを作るためにはプロジェクトの進行過程でも要求の追加や変更を積極的に受け入れなければならない[5]．要求の追加や変更も，それらを受け取る時期を誤らなければプロジェクトが遅延することを避けることが可能である．そこで本研究では，期間内で終了したひとつの実例プロジェクトに着目し，開発期間中に発生した要求の追加や変更を調査することで，一つの要求抽出プロセスの実態を明らかにする．調査対象は，レストランの注文管理システムの半年間に及ぶ開発履歴である．本章では，調査結果を示すと共に，その調査結果から得られた要求抽出プロセスに対する知見を考察する[14][16][17].

2.1. 実例プロジェクトの概要

本節では，要求抽出の実態調査の対象である実例プロジェクトの特徴として，システム開発の背景，システム概要，開発プロセスとスケジュール，開発規模，プロジェクト関係者間の依存関係などを記述する．

2.1.1. 開発の背景

調査対象とした実例プロジェクトが開発したシステムはレストラン店舗におけるオーダ注文管理システムである．このシステムは，既存システムが抱えていた課題を解決するために，新たな要求を取り入れ，再開発を行ったものである．再開発の改善課題は以下の４つである．

１） システム導入，運用効率の向上
２） 障害回復可能性の向上
３） 外部システムとの相互運用性向上
４） 拡張性の向上

これらの課題を達成するために，プロジェクトの元請けと共同開発者が連携し情報共有を図りながら，開発が進められた．その結果，上記１）から４）の課題を達成し，期間内にプロジェクトを完了することができていた．

筆者は，開発当初からこのプロジェクトの品質レビューに参加しており，開発途中に多

くの要求の追加や変更が発生していたことがわかっていた．そこで，このプロジェクトの要求抽出プロセスを実践例として採り上げ，通常はプロジェクトを遅延させる原因となっている開発途中の要求の追加や変更が，このプロジェクトでは納期遅延の原因となっていなかった理由を解析することにした．

2.1.2. システム構成と概要

図 2.1 にレストラン店舗の注文管理システムの機器構成を示す．太い点線枠内が実例プロジェクトの開発範囲である．注文管理システムは，管理端末／システムサーバ，座席端末，センター連携サーバによって構成され，地域情報などを配信するコンテンツセンタとインターネット経由で繋がる．また，注文管理システムは，複数の他メーカが開発・販売している外部システム（注文受付システム）との連携も可能である．なお，一つの店舗には複数の注文受付システムが存在することはなく，何れか一社の製品が存在する．

運用面の特徴として，オーダ受付業務の省力化を目指し，お客様自身が客席の座席端末（液晶タッチパネル式端末）を簡単に操作し，自由にメニューを注文できるセルフオーダリングシステムであり，他メーカが開発・販売している注文受付システムとの並行運用を可能にしたシステムである．

以下に，それぞれの構成機能の概要を示す．

1）システムサーバ：　レストラン注文管理システムの全体を制御し，システム情報の設定変更機能，注文受付データの管理機能を持つ．また，センタサーバや注文受付システムとの連携機能を持ち，コンテンツ情報やメニューデータの入れ替え機能を持つ．

2）管理端末：　システムサーバの管理端末であり，システム全体の稼動監視，及び保守用のシステム情報の設定／変更の機能を持つ．

3）座席端末：　顧客座席の端末（タッチパネル式端末）から注文受付け，注文確認や精算情報確認の機能を持つ．

4）センタサーバ：　コンテンツセンタとインターネット経由で連携し，注文管理システムが扱う注文メニュー，娯楽，地域情報などのコンテンツ情報の配信機能を持つ．

5）注文受付システム：　レストランの接客係が持つハンディ端末（注文受付端末）から入力された注文データを注文受付システム（Order Entry System：以降，OES と呼ぶ）が受信し，注文受付，調理指示，伝票発行，会計精算までの一連のレジ機能を持つ．注文受付システムは複数メーカが開発・販売している製品であり，メーカ毎にインタフェース仕様が異なる．

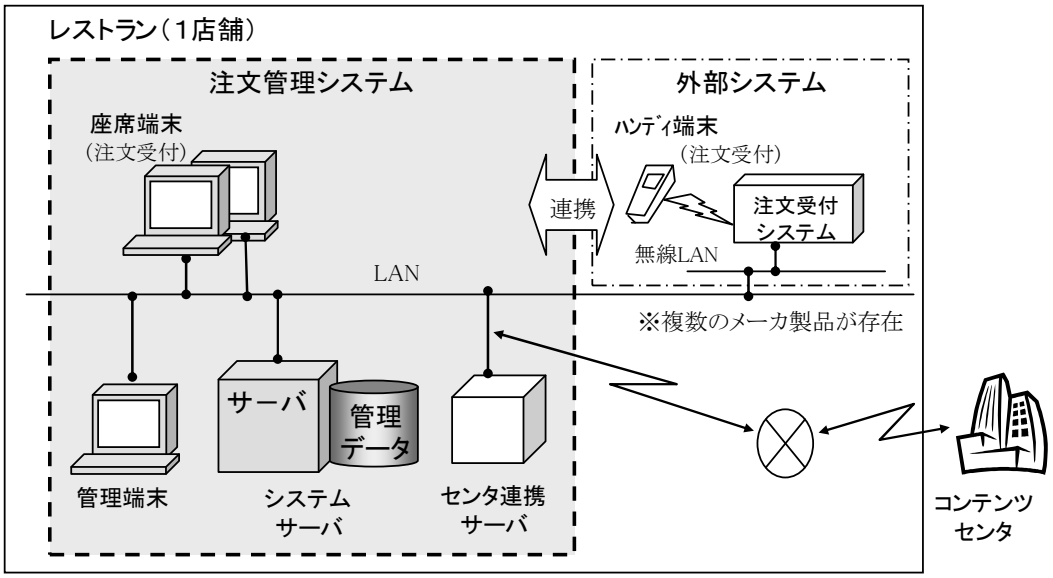


図 2.1　レストラン注文管理システムのシステム構成

2.1.3. ソフトウェア機能構成

本項では，レストラン注文管理システムのソフトウェアの構成要素を述べる．第 2.1.2 項の図 2.1 のシステム構成に基づいて，ソフトウェア構成は，主に以下の 8 つのソフトウェアコンポーネント機能で構成される（図 2.2）．

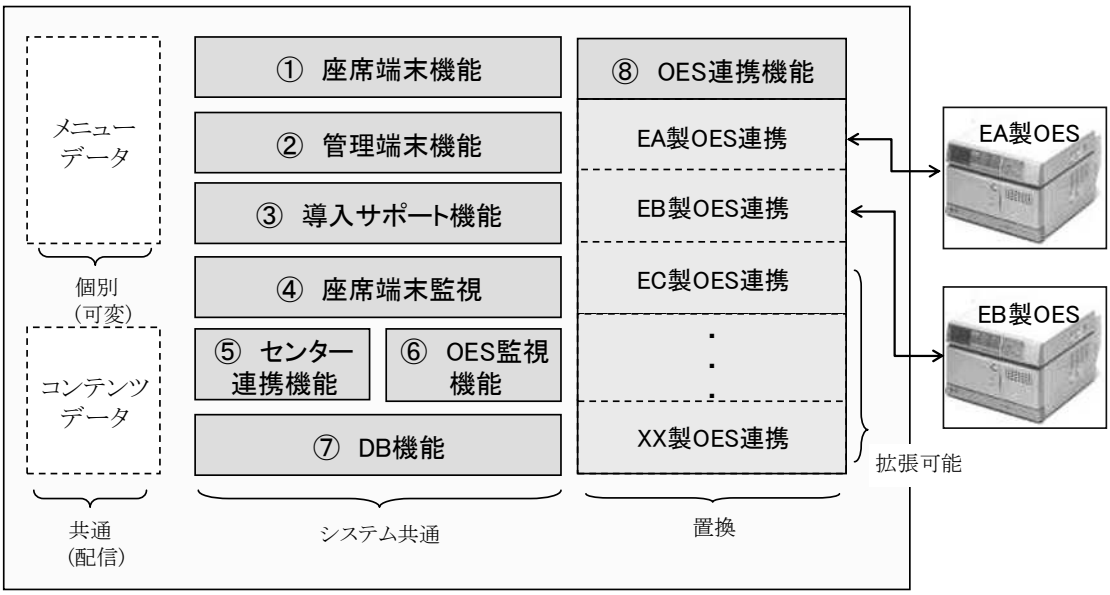


図 2.2　注文管理システムのソフトウェア構造（概略）

以下に，各々の機能概要を簡単に述べる．

①座席端末機能

レストランの各座席にある座席端末によるサービス開始，注文メニュー受付，店員呼び出し，コンテンツ情報表示，追加注文，精算情報の取得などの機能を持つ．開発時に操作性の変更要求を受け易いコンポーネントである．

②管理端末機能

システムサーバの管理端末機能であり，店舗の開店閉店，座席端末の制御，座席のグルーピング／解除，注文開始／終了，センター連携監視，OES 連携監視などシステム全体の稼働監視及び保守用のシステム情報の設定／変更の機能を持つ．

③システム導入サポート機能

座席端末のメニューデータチェック，管理端末インストール機能などをもつ．

④座席端末監視機能

座席端末がシステムサーバとオンライン状態であるか否かを定期的に監視する機能である（死活監視機能）．オフライン状態であれば，管理端末機能にアラームを通知する．

⑤センター連携機能

レストランの各店舗の開店／閉店処理連携，地域情報や娯楽情報などのコンテンツ情報の配信開始／停止の機能を持つ．

⑥OES 監視機能

商品品切れ情報取得機能，精算情報取得機能や OES と注文管理システムとがオンライン状態であるか否かを定期的に監視する機能である（死活監視機能）．オフライン状態であれば，管理端末機能にアラームを通知する．

⑦DB 機能

システムサーバのデータベース（DB）を制御する機能．

⑧OES 連携機能

OES 連携機能は，複数メーカが開発している OES と連携する機能である．これはシステムサーバのアプリケーション部とは独立し，各メーカの OES と連携ができ，置換可

能なソフトウェア構造としている．また，将来的に，追加可能なように拡張性を持たせている．

2.1.4. プロジェクト関係者（ステークホルダー）の依存関係

図 2.3 に示すように，注文管理システムの開発プロジェクトに関わるステークホルダーは，元請け（全体システムの取りまとめ，かつスポンサー）と，共同開発グループ（A 社：新システムサーバ開発，B 社：既存システムサーバ開発，C 社：座席端末のソフトウェア開発）の 4 社で構成していた．それぞれの会社間には依存関係があり，元請けを中心に情報共有を実施していたが，B 社が担当していた既存システムサーバ開発を A 社が担当することになり下請け業者間の競合があった．そのために，A 社は既存システムサーバの詳細仕様を短期間で把握することが困難であり，苦戦を強いられた．従って，A 社はドメイン知識を熟知している元請けへのインタビューを通じて既存システムを把握した．

注文受付システム（OES）を開発しているメーカは複数あり，注文管理システムの開発によるシステム上の変更はなく，プロジェクト範囲外である．注文管理システムとのシステム連携によるインタフェース仕様の開示だけである．元請けと複数のメーカ間には営業戦略上において競合関係にあり，そのため，システム連携におけるインタフェース仕様の開示が遅れ，開発の進め方やスケジュールに影響を与えていた．

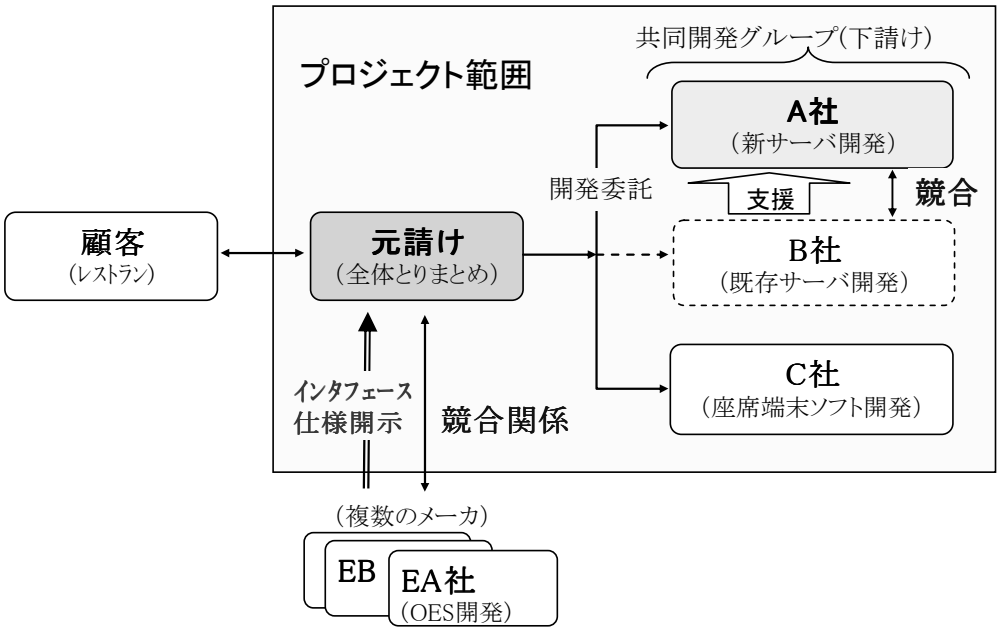


図 2.3　プロジェクト関係者

2.1.5. 開発プロセス，スケジュール，開発期間

図 2.4 に，レストラン注文管理システムのメイン機能であるシステムサーバの開発プロセスとスケジュールを示す．開発プロセスは，ウォータフォール型開発モデルを基本として適用している．しかし，開発期間が 5 ヶ月という短い期間のもとに，設計工程，製造工程，テスト工程がかなり重なり合った並行開発を行っていた．つまり，開発を進めながら要求抽出と，設計，実装，テストの開発プロセスが並行に進められていた．図 2.4 の横軸は開発開始からの相対月を示し，縦軸に開発プロセスの計画と実績を示す．

開発途中の工程では，要求抽出の遅れに伴う品質低下，納期遅れなどのリスクを回避するための対策を実施していた．以下に，開発途中の重要な意思決定の内容に沿って，開発の経過を以下に示す．

1) 既存のシステムサーバの詳細仕様が把握できないという問題があった．その理由は既存システムが他社によって開発されたために開発ドキュメントが全く残っていなかったからである．これを解決するために，要求定義に先駆けて，現場視察による既存システムの運用業務フローの把握を実施していた．また，将来的に，システム拡張，変更に耐え得る設計にするために，業務サービス内容や要件ヒヤリングなどの事前調査を実施していた．（図 2.4 の対策 1）
2) 未経験分野の業務システムであるため，要求仕様の妥当性の判断が難しく，短期間では開発要員が立ち上がらないという問題があった. 要求を継続的に抽出するためには，分野に精通した元請けの協力は不可欠である．そのために，契約段階で，元請けへの全面的な協力要請と OES メーカへの協力支援を元請け経由で依頼していた．また，要求定義，仕様設計段階での要求の妥当性確認への顧客参加の開発プロセスを取り入れていた．（図 2.4 の対策 2）
3) OES は複数のメーカが開発販売しており，元請けとは競合関係にあり，そのため，OES とのインタフェース仕様の開示遅れが品質, 納期に影響する可能性が高いという問題があった．そのため，元請け経由で OES 開発メーカにインタフェースの開示を依頼していた．また，次の策として，OES 導入研修への参加によるインタフェース仕様の事前調査を実施していた．（図 2.4 の対策 3）

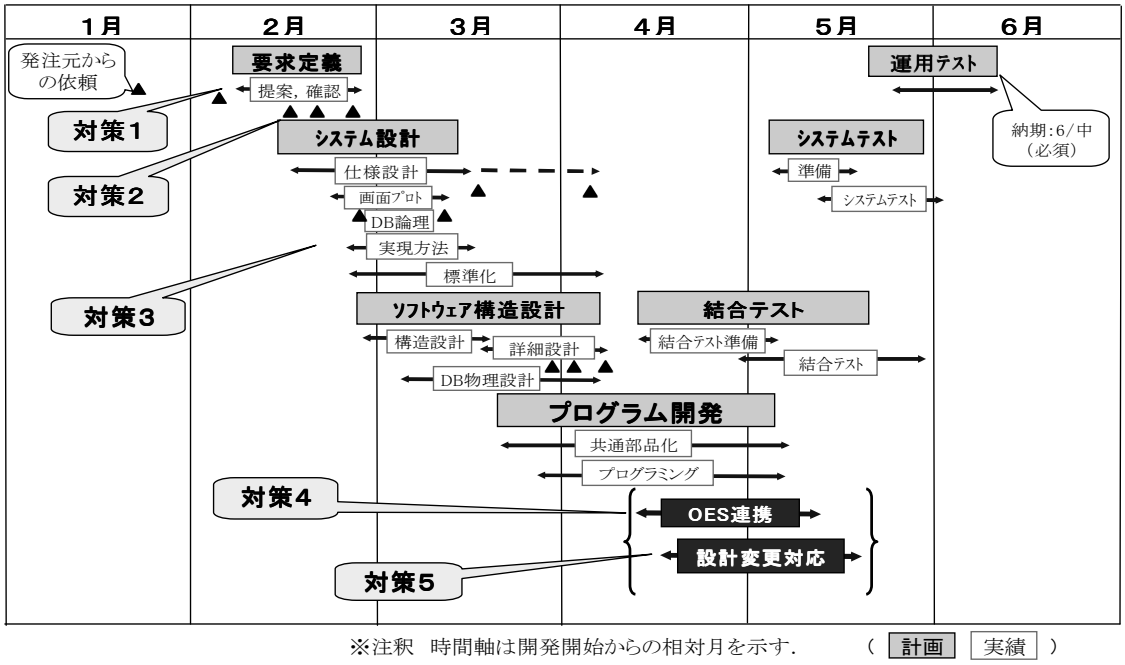


図 2.4　開発プロセス，スケジュール

4) OES とのシステム連携は初めてであり，結合テストにおいて，どのような問題が発生するか予測できない．また，メーカ毎に OES インタフェース仕様が異なり，苦戦が想定された．そのため，プログラム開発段階で OES の実機調査を実施していた．（図 2.4 の対策 4）

5) メーカからの OES インタフェース仕様の開示遅れたとしても，元請けのへ納期は守らなければならない．今回の再開発の狙いの一つであるシステムの拡張性は，複数のメーカの注文受付システムとの接続を可能にすることである．開発当初より複数のメーカとの協力関係の調整遅れを懸念していた．そのため，事前調査不足による要求変更を吸収するために，プログラミング工程時の正式仕様開示をデットラインとし，設計変更対応プロセスを開発スケジュールに盛り込んでいた．（図 2.4 の対策 5）

2.1.6. 開発規模，工数，開発環境

注文管理システムの開発言語はＣ＃，Ｊａｖａで記述され，開発規模はソースコードで約５０千行であった．開発工数は約４０人月であった．実行環境のプラットフォームは Windows2000Server または Windows2003Server を可能とし，開発では Windows2003Server を使用していた．また，ＤＢとしては，MSDE2000(SQLServer2000)を使用していた．

2.2. 要求抽出に関する調査

本節では，レストラン注文管理システムの要求抽出プロセスを明らかにするため，調査対象となる開発途中のエビデンス，およびその調査方法について記述する．

2.2.1. 調査対象と方法

第 2.1.4 項のプロジェクト関係者の依存関係より，実例プロジェクトでは複数の会社が開発作業を分担していたが,その中で要求定義からシステムテストまでのアウトプットである作業成果物（開発履歴）が比較的しっかり残っていた A 社の開発履歴を調査対象にした．調査の進め方は，先ず，A 社の開発履歴をすべて調査し，プロジェクトマネジメント関連，エンジニアリング関連，品質管理関連，開発支援関連と分類した（表 2.1）．次に，開発履歴の中から開発期間中に要求追加や変更の内容および件数と,それが行われた開発プロセス上の時期を調査した．更に，調査結果の裏づけを取るために，開発のプロジェクトリーダへのインタビューも実施した．

特に，開発履歴の中で要求抽出が記録されていた以下の資料に注目した．

- 要求仕様提案書：元請けの要求に対する提案確認内容
- QA 管理表：QA（Question And Answer）票の管理表であり，要求機能に関する質問事項，問合せ先，期限，優先順位，回答内容を含む
- レビュー指摘事項：元請けのレビューによる指摘事項
- 打合せ議事録：進捗会議や技術会議にて決定された事項，担当，期限を含む
- 課題管理表：残作業，要求変更履歴，実施すべき作業，期限，優先順位，担当を含む
- 障害一覧表：障害内容，対処方法，担当，優先順位，期限，承認者を含む
- 要望書一覧：要望内容，重要度，指摘者，対応時期を含む

これらの作業成果物は，要求分析，仕様設計，構造設計，実装，テストに至る各開発フェーズにおいて作成され，管理されていた．また，開発作業プロセスの内容調査から要求の追加および変更を開発に取り込むか否かは，システムへの実現可能性，対応に要するコスト，他への影響度，緊急度，優先順位付けなどを元請けとの調整および合意によって決定していた．これらの要求抽出に対するプロジェクトマネジメント情報も各開発履歴に記録していた．したがって，要求変更の内容や頻度，および開発プロセスと要求変更のタイミングとの関係を調査するために，これらの開発履歴を参照し分析することは本研究に有効であると確信している．

表 2.1　作業成果物一覧

	ﾌﾟﾛｼﾞｪｸﾄﾏﾈｼﾞﾒﾝﾄ関連	ｴﾝｼﾞﾆｱﾘﾝｸﾞ関連	品質管理関連	開発支援
作業成果物	・プロジェクト計画書 ・進捗報告 **・課題管理表** ・問題管理表 **・QA管理表(QA票)** **・打合せ議事録** ・ペンディング事項 ・申し送り事項一覧	**・要求仕様提案書(RFP)** ・仕様設計書 ・構造設計書 ・プログラム設計書 ・ソースコード ・テスト仕様書 ・レビューチェックシート **・レビュー指摘事項**	**・障害一覧表** ・故障票 ・障害分析結果 **・要望書一覧**	・開発手順 ・様式

2.2.2. 要求の定義と計測方法

ソフトウェア工学の国際規格 IEEE Std 610.12 では“要求”を以下のように定義している[38].

① 問題を解決したり，目標を達成するために，ユーザが必要とする条件や能力.

② 契約，標準，仕様，あるいは，その他の正式に要請された文書を満たすために，システムやシステムコンポーネントが満たすべき条件，あるいは，持つべき能力.

③ 上記の①，②の条件や能力を記述した文書.

本論文における要求の定義は，開発期間中に発生した要求の追加変更削除の実態を調査することに主眼をおいているため,“開発するシステムや製品を実現する上での必要な条件や情報”を要求と定義する．具体的には，顧客から得られる要求だけでなく，開発者自ら得られる要求，共同開発者から得られる要求，プロジェクト外や市場から得られる要求など，システムに求められる要求を開発履歴から全て計測している．この計測の中には，開発途中の仕様変更も要求の変更として含めているが，障害は含めていない.

また，要求数の計測方法は，ソフトウェアの開発工程において，要求定義後の仕様設計以降からシステムテストまでの要求変更，追加，削除の個々を１件として計測している．数式では以下のとおりとする.

要求抽出数　＝　要求追加の数　＋　要求変更の数　＋要求削除の数

とし，累積抽出件数で示している．この計測方法は，後に記述する PRINCE モデル[18]の計測方法に取り入れられ,「要求抽出プロセスを観測するための要求計測ガイドライン」

の作成に貢献している.

2.2.3. 仕様設計工程の要求抽出プロセス

第 2.2.1.項の A 社の要求抽出に関連する資料を一つ一つ紐解きながら,資料の位置付け,記載項目と内容から開発プロセスとの関連を調査した．ここでは仕様設計工程の要求抽出を例として採り上げ，要求仕様の確認，仕様設計作業，元請けによるレビュー，内部レビュー，仕様書修正，および承認までの開発作業プロセスと作業成果物の関連を整理すると図 2.5 のようになる.

以下に，その流れを述べる.

1) 元請けの要求仕様をベースに仕様設計を進め（図 2.5 の①），疑問点および不明点があれば，元請けに質問（QA 票（Question））を投げる．（図 2.5 の②）
   また，並行して共同開発先への質問事項があれば QA 票を共同開発先へ投げる．（図 2.5 の⑧）
2) 元請けからの回答（QA 票（Answer））を待つ余裕がないため，出来上がった仕様書から内部レビューをかける．（図 2.5 の③）
3) 内部レビューが完了し，出来上がった仕様書から元請けに提出し，レビューを依頼する(図 2.5 の④)．この時，元請けのレビューアに対して，内部レビューでの申し送り事項として要検討事項やペンディング事項を仕様設計書に添付する.
4) 元請けは受け取った仕様設計書からレビューし（図 2.5 の⑤），A 社からの質問（QA 票（Question））への回答（QA 票（Answer））と設計上の指摘事項，確認事項を A 社に返す（図 2.5 の⑥）
5) A 社は共同開発先からの回答や元請けから受け取った回答（QA 票（Answer））および指摘事項を確認し，仕様設計書に反映し，再レビュー後，承認依頼をする（図 2.5 の⑦）

上記の要求抽出プロセスは仕様設計に限らず，画面仕様および操作設計作業，結合テスト仕様作成，システムテスト仕様作成においても同様のプロセスを実施していた.

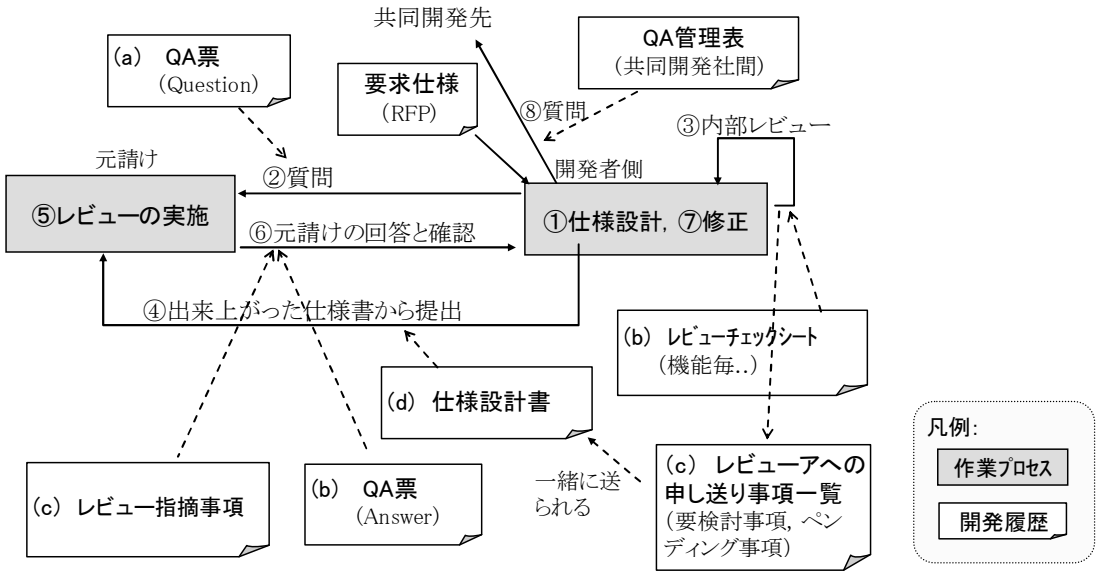


図 2.5　仕様設計工程での要求抽出プロセス

上記の調査より，以下の２つの知見を得た．

1）出来上がったもの順に提出効果（逐次提出）

出来上がったものから逐次提出し，要求抽出，仕様化，レビュー，合意，開発，納品というアジャイル開発[9]に近い開発プロセスを踏むことを契約段階で元請けと合意し，元請けの協力を得ていた．図 2.4 の開発スケジュール上のシステム設計において，「出来上がったものから提出」するプロセスは，元請けの作業負荷の手間を取り，更に開発のトータル工数が増加する可能性があり通常の請負契約では問題になる．しかし，実際には問題になっていない．何故ならば，この実例プロジェクトの課題は開発期間が 5 ヶ月という短い期間でサーバシステムを再構築し新製品を市場リリースすることであり，元請けと共同開発各社が協力し合い，如何に早く，既存システムを把握し，品質と納期を守るかに重点が置かれていたからである．不明点は申し送り事項やペンディング事項として情報を添付して元請けの担当者にレビューを実施して貰うか，あるいは，毎週の定例会議にて確認するという進め方を取っており，問題解決のスピードアップを図っていた．プロジェクトの立ち上げ時に，そういうルールを提案し，契約を交わし，合意しておけば，この進め方は有効である．

2）文書による記録効果

開発作業，レビュー，仕様変更の履歴を文書で管理することによって，要求確認，及び回答，ペンディング事項などの検討項目が明確に文書として残り，プロジェクト関係者間で共通認識と合意が得られたことである．また，前工程の要求定義段階での残課題や，システム設計での新たな要求や変更を始めとした以下の項目を課題管理表によって整理していた．

・残課題
・要求変更履歴
・打合せの決定事項

課題管理表によって整理していたこれらの内容は，ステークホルダー間で課題の共通認識ができ，必ず実施するべき事項である．これらを記録として残していたことによって，誰がいつまでに何を実施するべきかが明確になり，実施すべき事項を取りこぼさないように管理ができ，効果的であったと判断する．つまり，短期間での開発では，開発プロセスに管理プロセスを取り込んだコンカレントな開発が現実的である．

2.3. 要求抽出の調査結果

本節では，第 2.2 節で示したA社の開発履歴から，新たな要求，および要求変更に関する内容をひとつずつ解析し，ソフトウェア品質特性とソフトウェア構成要素の二つの視点によって，要求内容を分類し，開発途中での要求抽出傾向を調査した結果を述べる．

2.3.1. 品質特性別の要求抽出調査

実例プロジェクトの開発履歴から抽出した要求をソフトウェア品質特性（JIS X 0129-1:2003）[39]の 6 つの品質特性，即ち，機能性，信頼性，使用性，効率性，保守性，移植性別に分類し，要求抽出の時期を開発開始からの経過日数で整理した．ソフトウェア品質特性は表 2.2 に示すように 6 個の品質特性と 21 個の品質副特性からなる．本論文では，要求抽出の概略傾向を調査することに主眼をおき，6 つの品質特性毎に整理した．

表 2.2　ソフトウェア品質特性(JIS X 0129-1:2003)

品質特性	品質副特性	主な内容
機能性	合目的性、正確性、相互運用性、セキュリティ, 標準適合性	目的から求められる必要な機能の実装の度合い
信頼性	成熟性、障害許容性、回復性	機能が正常動作し続ける度合い
使用性	理解性、習得性、運用性、魅力性	わかりやすさ、使いやすさの度合い
効率性	時間効率性、資源効率性	目的のために使用する資源の度合い
保守性	解析性、変更性、安定性、試験性	保守(改訂)作業に必要とする努力の度合い
移植性	環境適応性、設置性、共存性、置換	別環境へ移した際、そのまま動作する度合い

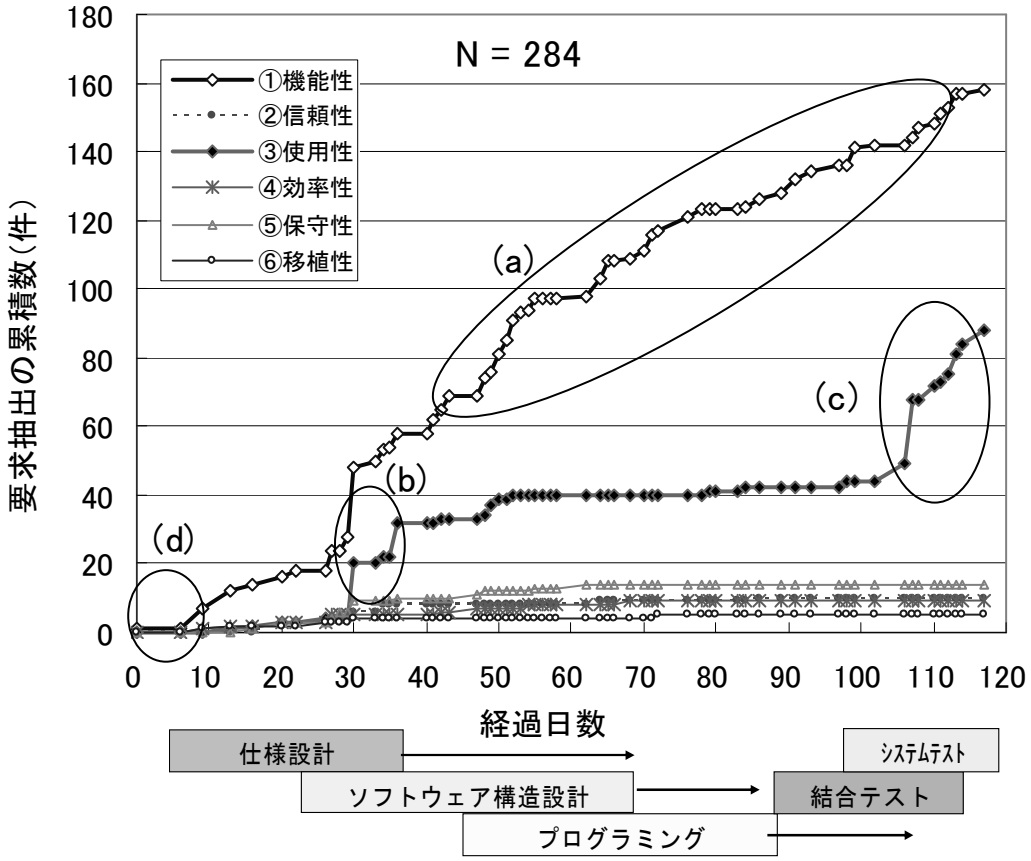


図 2.6　品質特性別の要求抽出累積グラフ

表 2.3　品質特性別の要求抽出件数

品質特性	抽出数	比率
① 機能性	**158**	**56%**
② 信頼性	10	4%
③ 使用性	**88**	**31%**
④ 効率性	9	3%
⑤ 保守性	14	5%
⑥ 移植性	5	2%
合　計	284	100%

図 2.6 に，その調査結果を示す．横軸は開発開始からの経過日数を示し，縦軸は品質特性別の要求抽出の累積件数を示す．図 2.6 から，実例プロジェクトでは開発開始から終了に至るまで 120 日間の開発プロセス全体で継続的に要求抽出していたことが分かる．しかし，仕様設計工程以前の要求定義工程での要求抽出がないのは調査段階で除外したからである（図 2.6 の(d)）．何故ならば，本研究の調査の狙いが要求定義工程以降の要求変更，追加の実態調査に焦点を当てているからである．実際には，要求定義工程で 133 件の要求を抽出していた．

また，表 2.3 に，品質特性別の要求抽出数と比率を示す．特に，機能性，使用性に関する要求抽出が多い．全体の要求抽出件数 284 件に対し，機能性の要求抽出数は 158 件で，全体の 56%を占める．仕様設計工程以降においても，継続して要求抽出している点が注目点である（図 2.6 の(a)）．また，使用性の要求抽出数は 88 件で，全体の 31%を占める．仕様設計工程でプロトタイピング[7]を適用し顧客参加型[15]の要求抽出を実践していた（図 2.6 の（b)）．更に，システムテスト工程では，画面表示，操作性の改善要望 45 件を抽出していた（図 2.6 の(c)）．

個々の品質特性別の要求抽出履歴を調査した結果, 様々なことが明らかになった. 以降に, 機能性，信頼性，使用性，効率性，保守性，移植性毎の要求抽出状況を述べ，考察する．

1）機能性に関する要求抽出

図 2.7 に機能性に関する要求抽出状況を示す．縦軸に要求抽出件数，横軸に開発開始からの経過日数を示す．以下に，調査結果を記述する．

- 機能性の要求抽出数 158 件に対して，仕様設計／構造設計の工程で，62%の要求抽出を実施していた（図 2.7 の①）．ソフトウェアの機能からみると，管理端末機能，注文受付システムとの連携機能を抽出していた．
- 機能性の要求抽出は開発開始から終了まで常に抽出し続けていた（図 2.7 の①, ②）．その理由は，システムサーバの基幹部分が管理端末であり，注文受付システムとの連携およびシステム全体を監視する機能であるが，既存システムの開発ドキュメントが殆どなく，A 社が仕様設計工程から元請けと一緒に開発を進め，要求抽出を行ったからである．
- 注文受付システムのインタフェース仕様の開示がプログラミング工程まで遅れ，結合テスト工程以降も要求抽出していたにもかかわらず，プロジェクトの遅延とはならなかった（図 2.7 の③）．その理由は，元請と注文受付システムの開発販売メーカとは競合関係にあり，開発当初からインタフェース仕様の開示が遅れると予測し，要求を繰り返し取り込むインクリメンタル型の開発プロセス[8]を行っていたためである．開発者へのインタビューにより，注文受付システムとの連携部分はコンポーネント化し，他の機能に影響しない設計であることを確認できた．

開発を進めながら要求抽出するには，単に，要求を段階的に抽出すればよいと言うのではなく，要求抽出の進め方に適合した設計を行う必要があることがわかった．

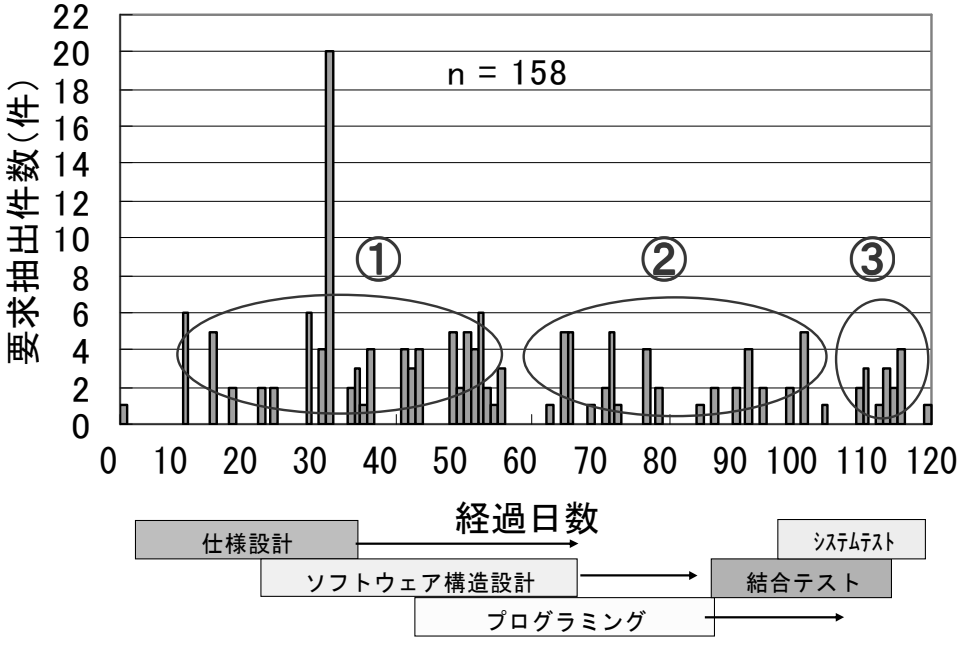


図 2.7　機能性に関する要求抽出

2）信頼性に関する要求抽出

図 2.8 に信頼性に関する要求抽出状況を示す．縦軸に要求抽出件数，横軸に開発開始からの経過日数を示す．以下に，調査結果を記述する．

- 信頼性の要求抽出数 10 件に対して，仕様設計／構造設計の工程で 90%の要求抽出を実施していた（図 2.8 の①，②）．
- 仕様設計工程では，システムダウン時の調査方法（エラー発生時のアクセスログ，操作ログ）について要求抽出を実施していた（図 2.8 の①）．
- ソフトウェア構造設計工程では，過去データの保持，システムの復旧方法について要求抽出を実施していた（図 2.8 の②）．
- プログラミング工程では，注文受付システムとの外部インタフェース接続による要求抽出を実施していた（図 2.8 の③）．これは，インタフェース仕様マニュアルには明記されていなかった仕様であり，プログラミング工程において，注文受付システムと接続して初めて分かった要求であるからである．

信頼性に関する要求抽出は，既存システムが存在するため仕様設計／構造設計までに殆ど抽出していたのは適切と言える．ただ，外部システムの死活状態のインタフェース仕様がマニュアルと異なるというのは実際に接続してみないと分からない要求であり，この時期での抽出は妥当と言える．

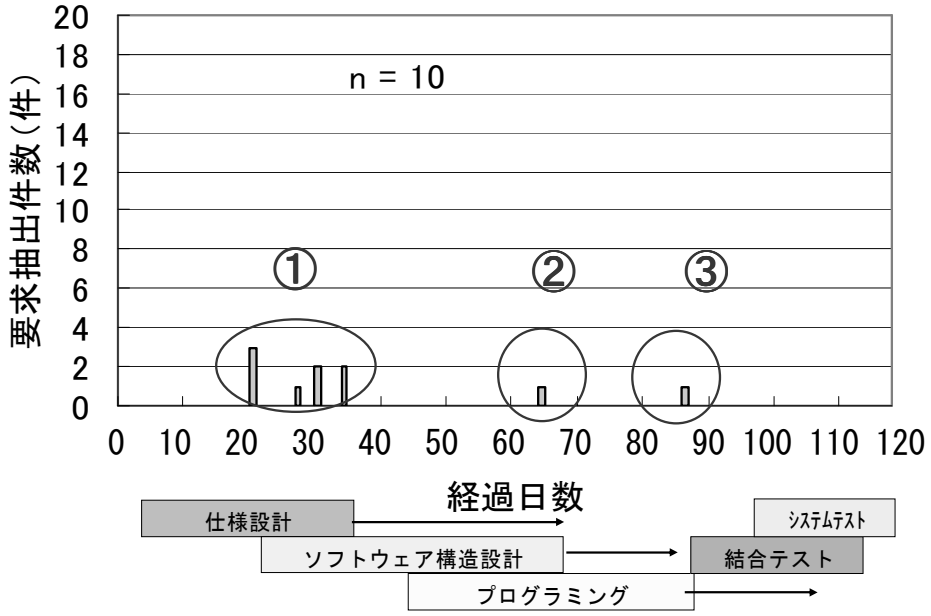


図 2.8　信頼性に関する要求抽出

3）使用性に関する要求抽出

図 2.9 に使用性に関する要求抽出状況を示す．縦軸に要求抽出件数，横軸に開発開始からの経過日数を示す．以下に，調査結果を記述する．

- 使用性の要求抽出数 88 件に対して，仕様設計／構造設計の工程で 50%の要求抽出を実施していた（図 2.9 の①）．
- 注目すべ点は，構造設計からプログラミング工程の間では殆ど要求抽出していないが（図 2.9 の②），システムテスト工程では画面表示や操作性に関する要望事項を 45 件（約 50%）も抽出していたことである（図 2.9 の③）．その理由は，後工程での手戻り作業を回避するために仕様設計工程でプロトタイプの画面を作成し，元請けと密接に打合せしながら要求抽出を行っていたが，システムテスト工程では，元請けが自身の目で確認し操作することで新たな改善要求を抽出したからである．
- このように新たな改善要求を受入れても，プロジェクトの遅延を招くことはなかった．その理由は，開発者が使用性の改善をプロジェクトの後期に取り込めるような設計を行っていたことにある．開発者へのインタビューにより，過去の経験から，ユーザインタフェースを置換可能となるようにアーキテクチャを設計していた．

使用性に関する要求抽出は，実際に出来上がって，顧客が自分自身で確認し操作することによって新たな要求が発生し易いものである．そのため，要求変更を想定し，開発途中での要求変更に耐え得るソフトウェア構造を設計工程で実践するべきである．

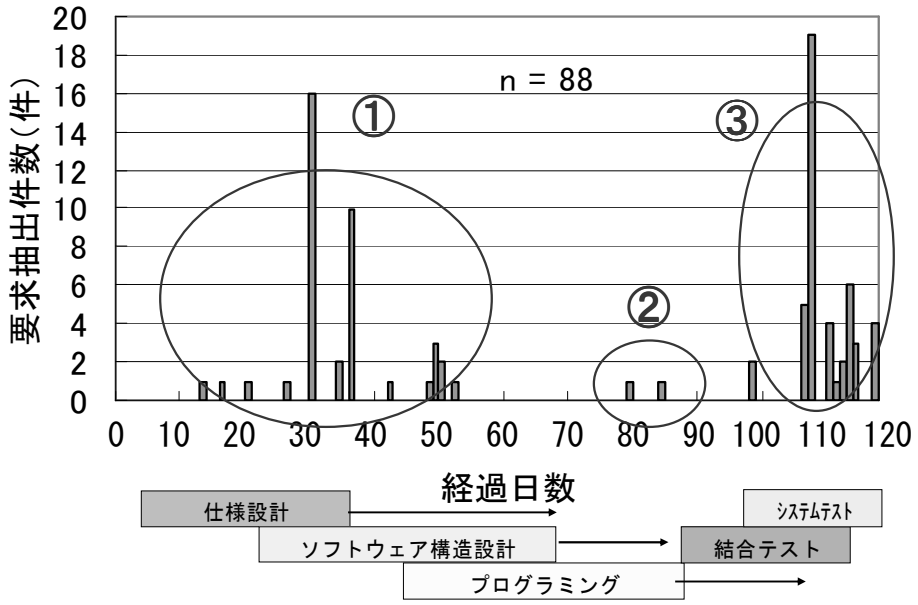


図 2.9　使用性に関する要求抽出

4）効率性に関する要求抽出

図 2.10 に効率性に関する要求抽出状況を示す．縦軸に要求抽出件数，横軸に開発開始からの経過日数を示す．以下に，調査結果を記述する．

- 効率性の要求抽出数 9 件に対して，仕様設計／構造設計の工程で 90%の要求抽出を実施していた（図 2.10 の①，②，③）．
- 仕様設計工程では，注文受付システムとの外部インタフェース性能，端末の画面表示性能など，性能に関する要求抽出を実施していた（図 2.10 の①）．また，データ量に伴うＤＢ構造見直しなどの資源効率についても要求抽出を実施していた．
- 構造設計工程では，ＤＢ性能改善（図 2.10 の②），メニュー性能改善（図 2.10 の③）など性能に関する要求抽出を実施していた．構造設計工程での要求抽出は適切な時期であったと言える．何故ならば，DB の論理構造を定義するプロセスが，ソフトウェア構造設計であり，処理スピードの性能を考慮する必要があるからである．
- プログラミング工程では，注文受付システムとの外部インタフェース性能改善に関する要求抽出を実施していた（図 2.10 の④）．

効率性に関する要求抽出は，既存システムが存在するため仕様設計／構造設計までに殆ど抽出していたのは適切と言える．

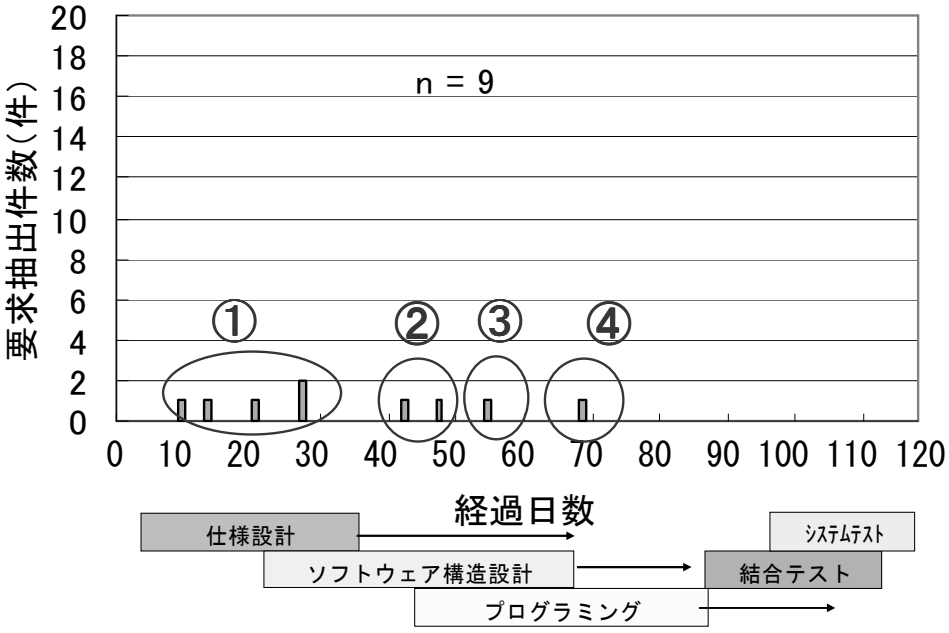


図 2.10　効率性に関する要求抽出

5）保守性に関する要求抽出

図 2.11 に保守性に関する要求抽出状況を示す．縦軸に要求抽出件数，横軸に開発開始からの経過日数を示す．以下に，調査結果を記述する．

- 保守性の要求抽出数 14 件に対して，仕様設計／構造設計の工程で 100%の要求抽出を実施していた（図 2.11 の①，②）．
- 仕様設計工程では，設計書の用語統一やエラーコードの定義，メニュー切替方式，フレームワークの使用方式など設計ドキュメントの見易さ，構造化設計などの要求抽出を実施していた（図 2.11 の①）．
- ソフトウェア構造設計工程でも同様に設計書の様式，コード定義様式など設計ドキュメントの見易さの要求抽出を実施していた（図 2.11 の②）．また，ユーザインタフェースは要求変更を受け易いため変更に耐え得る構造設計を実践し，外部インタフェース仕様はメーカ毎に異なるためコンポーネント化し置換可能な設計をする要求を開発者自身で要求抽出していた．

保守性に関する要求抽出は，顧客から要求抽出と言うより，開発者の経験から仕様設計／構造設計までに殆ど抽出可能と言える．

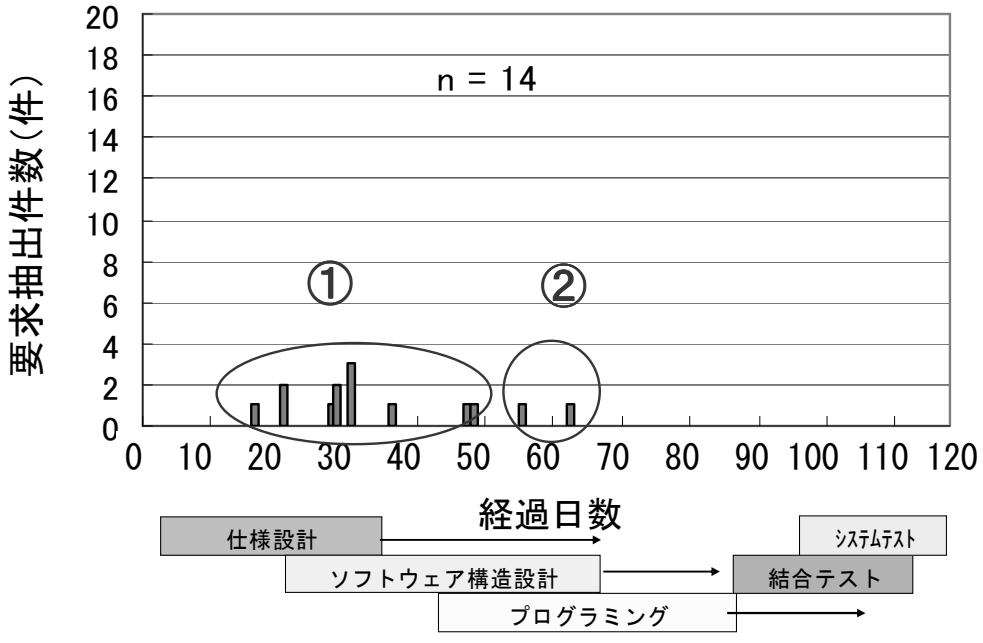


図 2.11　保守性に関する要求抽出

6）移植性に関する要求抽出

図 2.12 に移植性に関する要求抽出状況を示す．縦軸に要求抽出件数，横軸に開発開始からの経過日数を示す．以下に，調査結果を記述する．

- 移植性の要求抽出数 5 件に対して，仕様設計／構造設計の工程で 80%の要求抽出を実施していた（図 2.12 の①）．
- 仕様設計工程では，インストーラの設定，ＩＰアドレスの設定など，システム導入時のシステム環境設定の要求抽出を実施していた（図 2.12 の①）．
- プログラミング工程では，システム導入時の端末の初期設定情報の不足分の要求抽出を実施していた（図 2.12 の②）．この不足分は，端末のインストール情報であるため，ソフトウェアのアーキテクチャに影響することもないため，システムテスト工程までに決めれば開発上の問題はない．

移植性に関する要求抽出は，既存システムが存在するため仕様設計／構造設計までに殆ど抽出していたのは適切と言える．

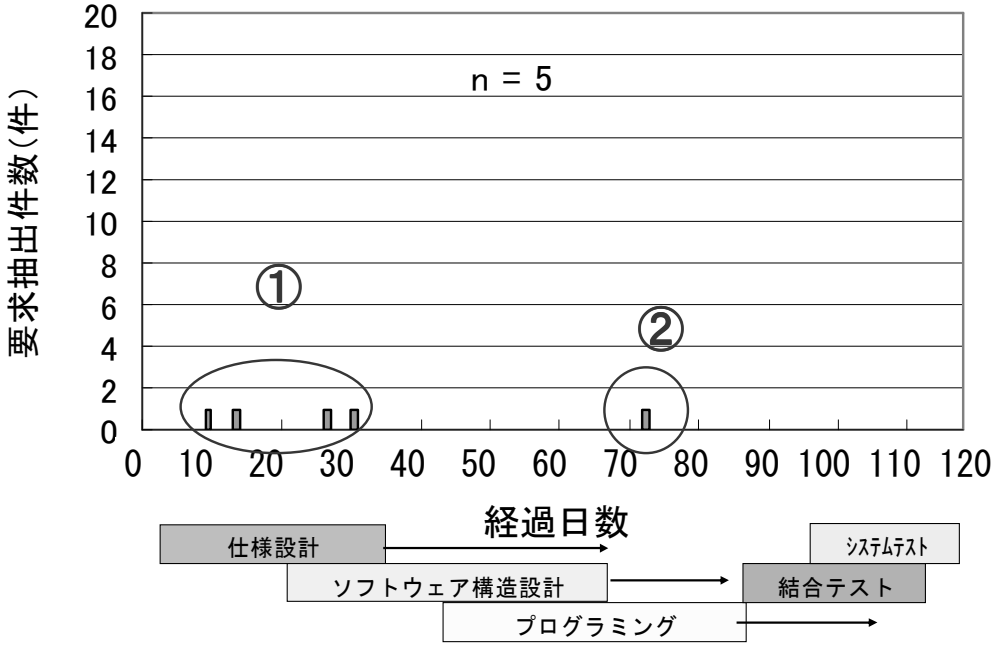


図 2.12　移植性に関する要求抽出

2.3.2. 機能構成要素別の要求抽出調査

本項では，第 2.1.3 項で述べた注文管理システムのソフトウェア機能構成要素からみた要求抽出を分析する．この分析の目的は，要求抽出する相手，機能などの特徴によって要求抽出の内容，タイミングの傾向を調査するためである．第 2.3.1 項で述べたソフトウェア品質特性別の要求抽出の分析と同様に，図 2.13 に，ソフトウェアの機能構成要素別の要求抽出件数を示す．横軸は開発開始からの経過日数を示し，縦軸は機能構成要素別の要求抽出累積件数を示す．図 2.13 から，実例プロジェクトでは開発開始から終了に至るまでの 120 日間の開発工程全体で継続的に要求を抽出していたことが分かる．しかし，調査段階で要求定義工程での要求抽出を除外した．何故ならば，本研究の調査の狙いが要求定義工程以降の要求変更，追加の実態調査に焦点を当てているからである．

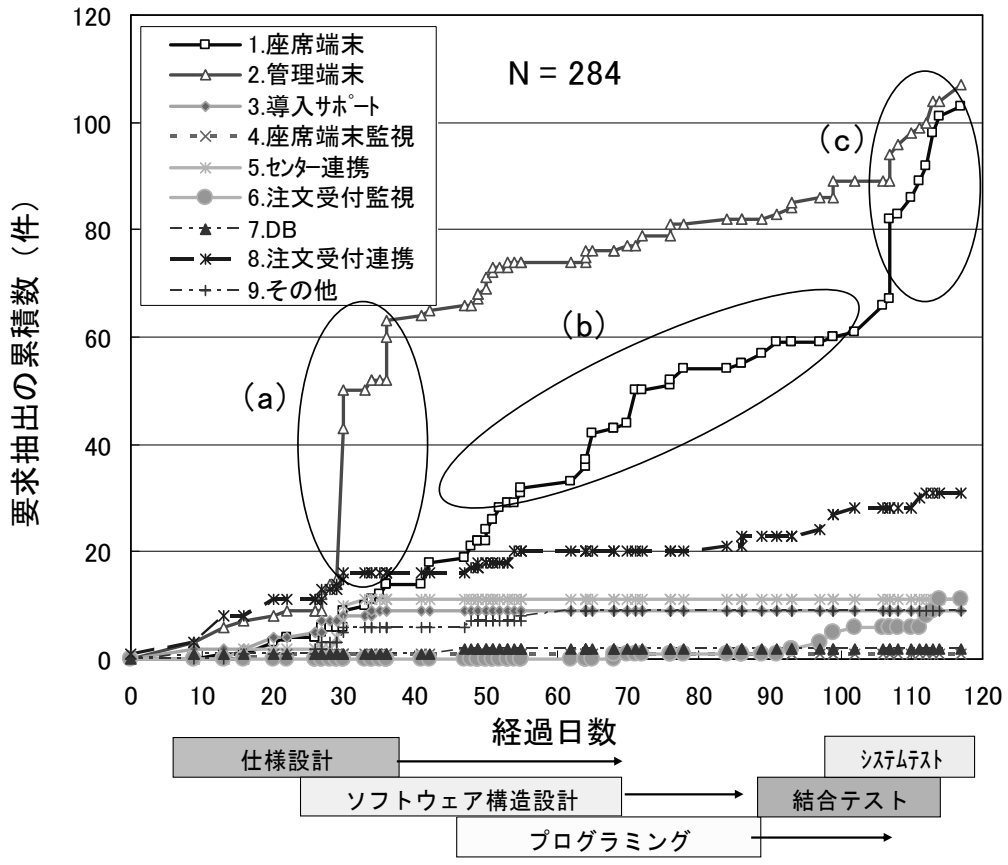


図 2.13　ソフトウェア機能構成要素別の要求抽出累積グラフ

表 2.4　ソフトウェア機能構成要素別の要求抽出件数

機能構成要素	抽出数	比率
1．座席端末	**103**	**36%**
2．管理端末	**107**	**38%**
3．導入ｻﾎﾟｰﾄ	9	3%
4．座席端末監視	1	0%
5．ｾﾝﾀｰ連携	11	4%
6．注文受付監視	11	4%
7．DB	2	1%
8．注文受付連携	31	11%
9．その他	9	3%
合　計	284	100%

また，表 2.4 に，ソフトウェア機能構成要素別の要求抽出数と比率を示す．特に，管理端末機能，座席端末機能に関する要求抽出が目立つ．全体の要求抽出件数 284 件に対し，管理端末機能の要求抽出数は 107 件で，全体の 38%を占める．仕様設計工程でプロトタイピングを活用し顧客参加型[15]の要求抽出を実践していた（図 2.13 の(a)）．また，座席端末機能の要求抽出数は 103 件で，全体の 36%を占める．仕様設計工程以降においても，継続して要求抽出している点が注目点である（図 2.13 の(b)）．更に，経過日数 105 日目のシステムテスト工程では，画面表示，操作性の改善要望 45 件を抽出していた（図 2.13 の(c)）．これは画面操作など人による運用であるため要求変更を受け易いためである．

この図 2.13 の中で，要求抽出の変化が著しい管理端末，座席端末，注文受付連携，注文受付監視機能に関して，以降に分析した結果を述べる．

1）管理端末に関する要求抽出

図 2.14 に管理端末に関する要求抽出状況を示す．縦軸に要求抽出件数，横軸に開発開始からの経過日数を示す．以下に，調査結果を記述する．

- 管理端末の要求抽出数 107 件に対して，仕様設計工程で画面や機能仕様など 66 件（62%），構造設計工程で 9 件（8%），設計工程での合計で約 70%を要求抽出していた（図 2.14 の①）．プログラミング工程で 12 件（11%），結合テスト工程で 4 件（4%）とやや少なく（図 2.14 の②），システムテスト工程では画面操作の変更や注文受付連携に関連する 16 件（15%）の新たな要求を抽出していた（図 2.14 の③）．
- 設計工程で約 70%の要求抽出ができたのは，管理端末機能は，第 2.1.3 項のソフトウェア機能構成で記述しているように，システム全体の機能をコントロールするシステムサーバそのものであり，既存システムがあるものの，元請けの協力度が高かったからである．

第 2.1.4 項のプロジェクト関係者の依存関係で記述したように，下請け業者間に対立関係があっても既存システムを熟知している元請けの協力度合いによっては，設計工程において約 70%要求抽出できることが分かった．つまり，顧客参加型の開発を推進する[15]ことによって要求抽出を早くすることが可能である．

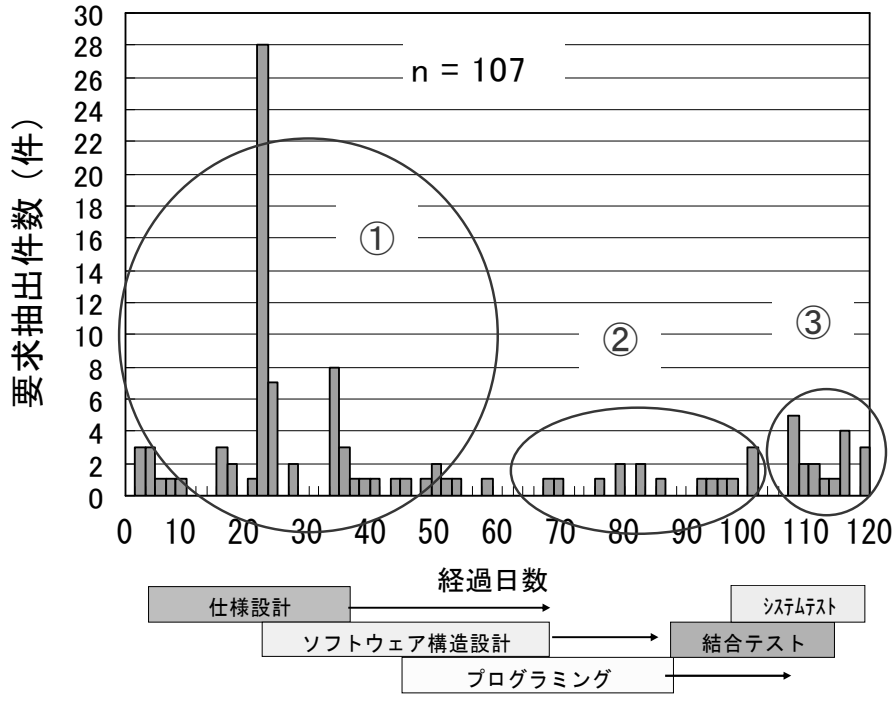


図 2.14　管理端末の要求抽出

2）座席端末に関する要求抽出

図 2.15 に座席端末に関する要求抽出状況を示す．縦軸に要求抽出件数，横軸に開発開始からの経過日数を示す．以下に，調査結果を記述する．

- 座席端末の要求抽出数 103 件に対して，仕様設計工程で 22 件（21%），構造設計工程で 28 件(27%)，設計工程での合計で約 50%を要求抽出していた(図 2.15 の①)．プログラミング工程では 13 件（13%）と少ないが（図 2.15 の②），システムテスト工程では 40 件（39%）も抽出していた（図 2.15 の③）．システムテスト工程で要求抽出が多いのは，元請けが自身の目で確認し操作することで新たな改善要求を抽出したからである．

システムテスト工程で多くの要望を受け入れることはプロジェクトの遅延を招く危険性があるが，このような新たな改善要求を受入れても，プロジェクトの遅延を招くことはなかった．その理由は，開発者の過去の経験から，ユーザインタフェースを置換可能となるようにアーキテクチャを設計しており，比較的容易な修正で済んだからである．座席端末のようなユーザフレンドリーな機能は，実際に出来上がって，顧客が自分自身で確認し操作することによって新たな要求が発生し易く動的なものである．そのため，要求変更を想定し，開発途中での要求変更に耐え得るソフトウェア構造を設計工程で実践するべきである．

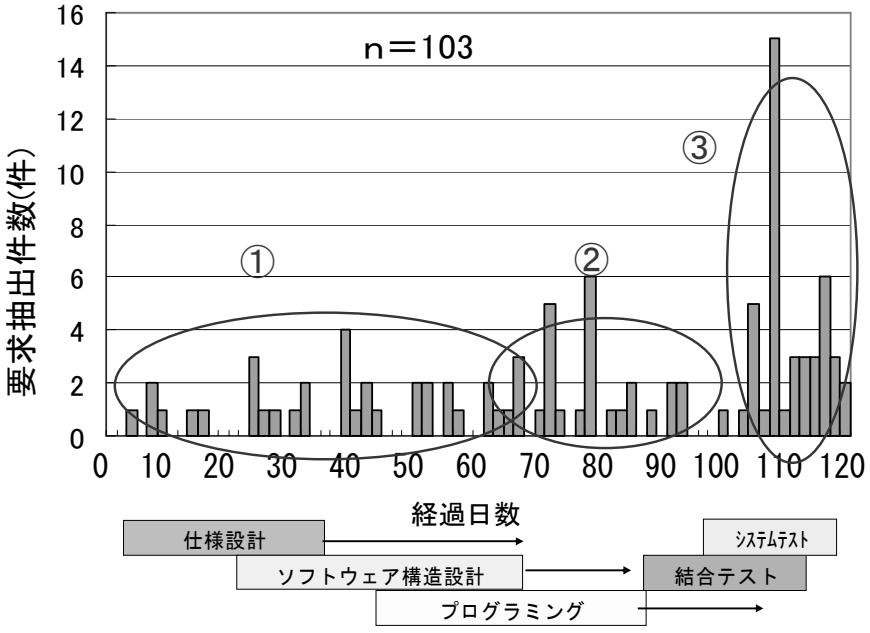


図 2.15　座席端末の要求抽出

3）注文受付連携に関する要求抽出

図 2.16 に注文受付連携に関する要求抽出状況を示す．縦軸に要求抽出件数，横軸に開発開始からの経過日数を示す．以下に，調査結果を記述する．

- 注文受付連携の要求抽出数 31 件に対して，仕様設計／構造設計工程で 58%(18 件)の要求抽出を実施していた（図 2.16 の①）．これは，第 2.1.5 項の開発プロセスで述べているとおり，元請けと注文受付システムを開発しているメーカは競合関係にあるため，注文受付システムとのインタフェース仕様の開示が遅れたが，メーカが一般向けに公開している導入研修へ内密で参加し，インタフェース仕様の事前調査を実施していたからである．
- 経過日数 70 日以降のプログラミング工程途中からシステムテストまで要求抽出を実施していた（図 2.16 の②）．これは，第 2.3.1 項の品質特性別の機能性や信頼性に関する要求抽出で述べているとおり，注文受付システムとの接続試験がプログラミング工程からであり，一般向けのインタフェース仕様マニュアルとインタフェースが異なる可能性があることは計画段階で想定しており，プログラミング工程においてリカバリする開発プロセスをスケジュールに盛り込んでいたため，納期的には問題がなかった．

このように，要求抽出する時期を計画し，必要な時に適切に要求抽出することは効率的な進め方と言える．

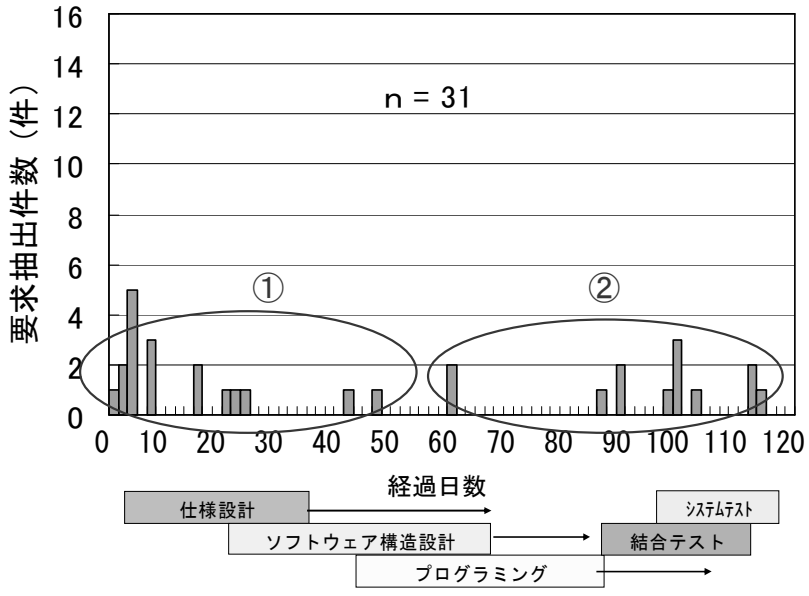


図 2.16　注文受付連携の要求抽出

4）注文受付監視に関する要求抽出

図 2.17 に注文受付監視に関する要求抽出状況を示す．縦軸に要求抽出件数，横軸に開発開始からの経過日数を示す．以下に，調査結果を記述する．

- 注文受付監視の要求抽出数 11 件であるが，仕様設計／構造設計工程での要求抽出はなく，経過日数 70 日以降のプログラミング工程途中から要求抽出を実施していた（図 2.17 の①）．これは，第 2.3.1 項の品質特性別の機能性や信頼性で述べているとおり，外部システムの注文受付システムとの接続試験がプログラミング工程からであり，インタフェース仕様マニュアルには明記されていなかった要求が接続して初めて分かったためである．
- 結合テストにおいても注文受付監視機能に関する不備を 5 件抽出していた（図 2.17 の②）．開発担当者へのインタビューによると，開発当初から注文受付システム監視機能を考慮していたが，2 番目以降のメーカが提供するインタフェース仕様には 1 番目のメーカのインタフェース仕様にない情報があることが判明し，運用上，注文受付システムの監視機能を変更せざるを得なかったからである．

インタフェース仕様マニュアルに明記されていない要求は予測不可能であり，ソフトウェアに組み込むことはできない．しかし，システムを動かすためにはこの要求を受け入れ，監視機能を変更するのは合理的な対処である．

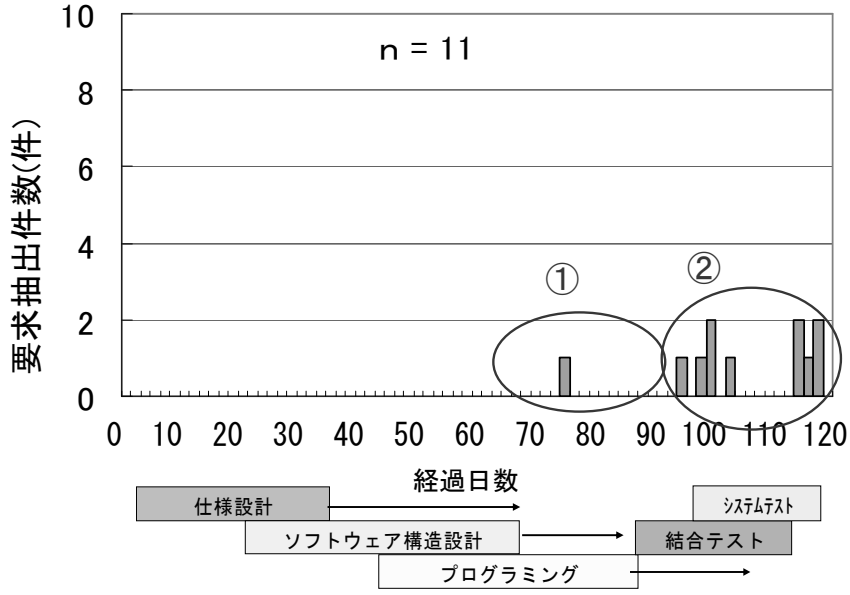


図 2.17　注文受付監視の要求抽出

2.4. 考察

本節では，第 2.3 節におけるソフトウェア品質特性とソフトウェア機能構成要素の二つの視点による要求抽出の調査結果から得られた知見について以下に考察する．考察は，開発プロセスと設計の視点，品質特性別の視点，ソフトウェア機能構成要素別の視点の 3 つの視点で整理する．また，第 1.2.2 項で述べた統合型要求工学による要求抽出プロセスの視点からも考察する．

1）開発プロセスと設計の視点

- **要求抽出の相手の特性を考慮した開発プロセスを実行するべき**

  他システムの外部インタフェース仕様の正式開示がプログラミング工程まで遅れたが，これは開発のスケジュール遅延を招くことはなかった．その理由は，第 2.1.4 項のプロジェクト関係者の依存関係で述べたように，顧客と他システムのメーカ間には営業戦略上において競合関係にあるため，開発当初から，外部インタフェース仕様の開示が遅れることを予測していた．そのため，開発の早期にメーカ公開のユーザ導入教育への参加や，開発の後期に外部インタフェース仕様が開示された段階で要求を取り込む開発プロセスを計画に盛り込こみ適切に実践していたためである．言わば，要求抽出の時期を計画し開発への影響を最小限にするためのリスク管理を実践していた．

- **要求変更に耐え得るソフトウェア構造設計を実行するべき**

  利用者インタフェースは，仕様設計工程において，プロトタイプの画面を作成し，顧客と確認しながら要求抽出を行っていたにも関わらず，システムテスト工程においても，画面表示や操作性に関する要望事項を数多く抽出していた．つまり，ソフトウェアが完成に近い形で出来上がった後，実際に，顧客自身が目で確認し，操作することで使用性に関する改善要求を抽出していた．このような改善要求を受け入れたにも関わらず，プロジェクトの遅延を招くことはなかった．この理由は，開発者が使用性の改善要求をプロジェクトの後期に取り込めるようなソフトウェア構造設計を行っていたからである．つまり，開発者自身の過去の経験から，利用者インタフェースを置換可能となるようにソフトウェアのアーキテクチャを設計していたからである．

  また，他システム連携のための接続部分はコンポーネント化し，他のアプリケーション設計部分に影響を与えないソフトウェア構造設計を選択していたことを開発リーダへのインタビューや設計資料から確認することができた．

  これらのように，開発途中での要求変更や追加を受け入れ易い機能は変更に耐え得る

ソフトウェア構造を設計段階で実践するべきである．

- **適切な時期に要求抽出を実行するべき**

ソフトウェア構造設計工程では DB の構造や性能に関する要求を抽出し，DB への要求は殆ど完了していた．既存システムが存在しており，開発スケジュール上，この時期での DB の要求抽出は適切な時期であったと言える．何故ならば，DB の論理構造を定義するプロセスが，ソフトウェア構造設計工程であり，DB の論理設計では，性能の要求を考慮する必要があるからである．また，このソフトウェア構造設計工程以前に，これらの要求を参照することはない．

2）品質特性の視点

- **保守性への要求抽出は早期に実行可能**

保守性に関しては，仕様設計工程で必要な要求を抽出し，他システムとの外部インタフェースのコンポーネント化，アプリケーションの構造化，生成文書の標準化を検討し実施していた．保守性に関する要求抽出は，顧客から要求抽出すると言うより，開発者の経験から仕様設計／構造設計までに殆ど抽出可能と言える．つまり，ソフトウェア開発の観点から開発者主導で早期に要求抽出が可能である．

- **移植性への要求抽出は早期に実行可能**

移植性に関しては，保守性と同様に要求定義工程でほとんど要求抽出が可能であり，ソフトウェア開発の観点から開発者主導で早期に要求抽出するべきである．ただし，分析した実例プロジェクトでは，プログラミング工程においてシステム導入時の端末の初期設定情報の不足分を要求抽出していた．この不足分は，端末のインストール情報であり，ソフトウェアのアーキテクチャに影響することもないため，システムテスト工程までに決めれば開発上の問題はない．開発者へのインタビューによってそのことを確認することができた．

- **信頼性への要求抽出は早期に実行するべき**

信頼性に関しては，保守性と同様に要求定義工程でほとんど要求抽出が可能であるが，その実現性に関して仕様設計や構造設計工程において十分に検討するべきである．つまり，ソフトウェア開発の観点から開発者主導で早期に要求抽出するべきである．実例プロジェクトでは，第 2.1.1 項で述べたようにシステム改善課題のひとつであった“障害回復可能性の向上”を解決するために，要求定義工程で要求抽出し，仕様設計工程でエラー発生時のアクセスログ，操作ログなど，システムの復旧時に必要な解析方法を具体化しながら要求抽出していた．要求機能の仕様化を検討するのが仕様設計工程であり，

“障害回復可能性の向上”という曖昧な要求を具体化したのは適切な時期である. ただ, プログラミング工程において，外部システムの死活状態のインタフェース仕様がマニュアルと異なるという要求抽出は実際に接続してみないと分からない予測不可能な要求であり，この時期での抽出は妥当と言える.

3）ソフトウェア機能構成要素別の視点

- **ドメイン知識の有識者参加を実践するべき**

開発するべきシステムのメイン機能が多々あり，開発者にドメイン知識がなければ，ドメイン知識のある有識者を参加させ，開発開始から終了まで要求抽出を行うべきである．実例プロジェクトでは，第 2.1.4 項のプロジェクト関係者で述べたように，開発するべき機能が多々あるが，既存システムの設計ドキュメントがなく，請負会社間に対立関係があっても，既存システムを熟知している顧客の協力度合いによっては，設計工程において約 70%要求抽出できていた. つまり, 顧客参加型の開発を積極的に推進する[15]ことによって要求抽出を早くすることができ，かつ顧客の要求を満足する品質のよいシステムを開発できる.

- **ユーザフレンドリーな機能は置換可能なソフトウェア構造設計をするべき**

画面操作がある端末のようなユーザフレンドリーな機能は，実際に出来上がって，顧客が自分自身で画面を確認し操作することによって新たな要求が発生し易いものである. そのため，予め，開発者は要求変更を想定し，開発途中での要求変更に耐え得るソフトウェア構造を設計工程で実践するべきである．また，開発工程の後半に要求変更や追加が発生した場合，納期と品質を確保するためには，それらの要求を受け入れるべきか，次の開発で対応するべきかを適切に判断しマネジメントするべきである[5]．実例プロジェクトではそのことを実践していたことを開発者へのインタビューによって確認することができた.

- **他システムとの連携は要求抽出のリスクを考慮するべき**

開発するべきシステムが他システムと連携する場合，インタフェース仕様に関する要求抽出が遅れる可能性があり，そのことを想定したリスク対策を考慮し実践するべきである．例えば，他システムを担当する会社と競合関係である場合や，他システムが新規開発である場合などが考えられる．実例プロジェクトでは，第 2.1.4 項や第 2.3.2 項で述べたように，競合関係にある外部システムとの連携ではインタフェース仕様の要求抽出する時期が遅れた．そのため，競合メーカが主催するユーザ導入教育への内密で参加や契約の早期締結に向けたネゴシエーションの強化など，特殊な要求抽出のプロセスを踏

み，スケジュール遅延を防ぐために，リスクを考慮しプロジェクトマネジメントを実践していた．

4）統合型要求工学による要求抽出プロセスの視点

第 1.2.2 項で述べた統合型要求工学による要求抽出プロセスの視点から得られた知見を以下に示す．

- **ソフトウェアの構成要素の特性に応じた要求抽出を計画的に実行するべき**

  第 2.3 節において，ソフトウェアの品質特性や機能構成要素の特性に応じて，急がなくてもよい要求は必要となる時期までに開発者主導で要求抽出し決めればよいということが分かった．つまり，統合型要求プロセスでは，単に，開発プロセスと要求抽出のプロセスを統合し要求を段階的に抽出すればよいと言うのではなく，ソフトウェアの構成要素の内外的な特性に応じて適切な時期に要求抽出を行う開発プロセスを計画的に実践するべきである．内外的な特性というのは，例えば，要求抽出をする相手の協力度合いやドメイン知識の度合い，また，開発者のスキルレベルの高低などである．要求を抽出する相手の特徴や事情による外的要因とエンジニアリングを実践する開発者側の内的要因によって要求抽出プロセスやタイミングが異なるのである．

- **要求変更に耐え得るソフトウェア構成設計を実践するべき**

  統合型要求プロセスでは，単に，開発途中での要求変更や追加を受け入れればよいというわけではなく，開発への影響を最小化するために，要求変更に耐え得るソフトウェアアーキテクチャの設計を計画的かつ積極的に実践するべきである．例えば，コンポーネント化や置換可能なアーキテクチャ設計を実践することによって他機能への影響を最小化することが可能である．

2.5. まとめ

第 2 章では，本研究の活動のきっかけとなった実例プロジェクト（レストランの注文管理システム）の開発背景および開発履歴の分析や開発者へのインタビュー結果から，開発現場で実践されている要求抽出プロセスや開発プロセス上の知見，ソフトウェア機能構成別の知見，マネジメントの知見などを得ることができた．また，統合型要求工学による要求抽出プロセスの視点においても，単に，開発プロセスと要求抽出のプロセスを統合し要求を段階的に抽出すればよいと言うのではなく，ソフトウェアの構成要素の内外的な特性に応じて適切な時期に要求抽出を行う開発プロセスを計画的に実践するべきであるという

知見を得ることができた．つまり，要求抽出の時期は，ソフトウェア品質特性やソフトウェア機能構成要素毎に要求抽出する相手や要求抽出する側の内外的な特性や事情によって，適切な時期が存在する．また，開発途中での要求変更や追加に対してスケジュール遅延なく対応するためにはソフトウェア構造設計への配慮やプロジェクトマネジメントが必要であることを明らかにした．

次章では，ソフトウェア品質特性およびソフトウェア機能構成要素毎の要求抽出の時期と要求抽出度合いとの因果関係を分析し，要求抽出と開発プロセスおよびプロジェクトマネジメントとの関連について提案する．

3. 要求抽出プロセスの成熟度からみた分類

第 2 章では，実例プロジェクトのレストラン注文管理システムにおける要求抽出プロセスの実態調査を実施した．その結果，ソフトウェア品質特性別，あるいはソフトウェア機能構成要素別に要求を分類すると，要求を抽出する相手の特徴や事情による外的要因とエンジニアリングを実践する開発者側の内的要因による要求抽出プロセスの種類があることが判明した．また，それらの要求抽出プロセスの種類によっては適切な要求抽出の時期があり，開発途中での要求変更に対してスケジュール遅延なく対応するためには，ソフトウェア構造の設計へのエンジニリングとプロジェクトマネジメントが重要であることを明らかにした．

本章では，開発経過に沿った要求抽出の時期と要求抽出の割合（即ち，成熟度）から見た分類を行い，要求抽出プロセスの成熟度モデルについて提案する．

3.1. 要求抽出の分類の観点

本節では，第 2 章の調査結果をベースに要求抽出プロセスの種類と特徴を分類する観点を述べる．分類の観点は，ソフトウェア品質特性別，およびソフトウェア機能構成要素別の要求抽出の成熟度がどのようになるかを確認し整理することである．成熟度に着眼した理由は，第 2.3 節の分析では要求抽出の件数に着眼したが，もともと要求抽出件数の少ない構成要素の場合，1 件の要求抽出が全体の要求に占める重みは大きくなるからである．つまり，要求抽出件数ではどの時点で要求抽出が完了したのかが判断しにくいため，実際の件数よりも，どの程度の割合の要求がどの時期に抽出されたかという相対的な要求抽出度数の方がより重要となる．この要求抽出度数の割合を成熟度と定義する．成熟度の考え方を適用することによって，要求抽出件数が少ない品質要求や機能要求における要求抽出のタイミングと変化が分かり易くなると期待する．

3.2. 要求抽出の成熟度分析

本節では，ソフトウェア品質特性別およびソフトウェア機能構成要素別の要求抽出の成熟度を分析した結果をそれぞれ図 3.1，図 3.2 に示す．

3.2.1. ソフトウェア品質特性別の要求抽出

図 3.1 に，ソフトウェアの品質特性毎に要求抽出の成熟度を分析した結果を示す．図 3.1 の縦軸は品質特性別の要求抽出の割合（成熟度）を示す[16][17]．横軸はプロジェクトの開始日からの相対的な経過日数を示す．その要求抽出の割合を以下のように定義する．:

（経過日までの要求抽出の割合：成熟度）＝　cumReq　／　allReq　×　100

allReq はプロジェクトの開始から終了までの品質特性別に対して抽出された要求の総数であり，cumReq は経過日までに抽出した要求の累積数である．図 3.1 の分析元データは第 2.3 節の図 2.6 の分析データと同じであり，要求抽出件数が少なく変化が分かりにくかった信頼性，効率性，移植性，保守性の要求抽出プロセスの変化が分かり易くなっている．この図 3.1 から，移植性や保守性に関する要求抽出はソフトウェア構造設計工程までに成熟しており，また，機能性や使用性に関する要求抽出は開発の完了まで段階的に要求を抽出し続けていることを確認できる．

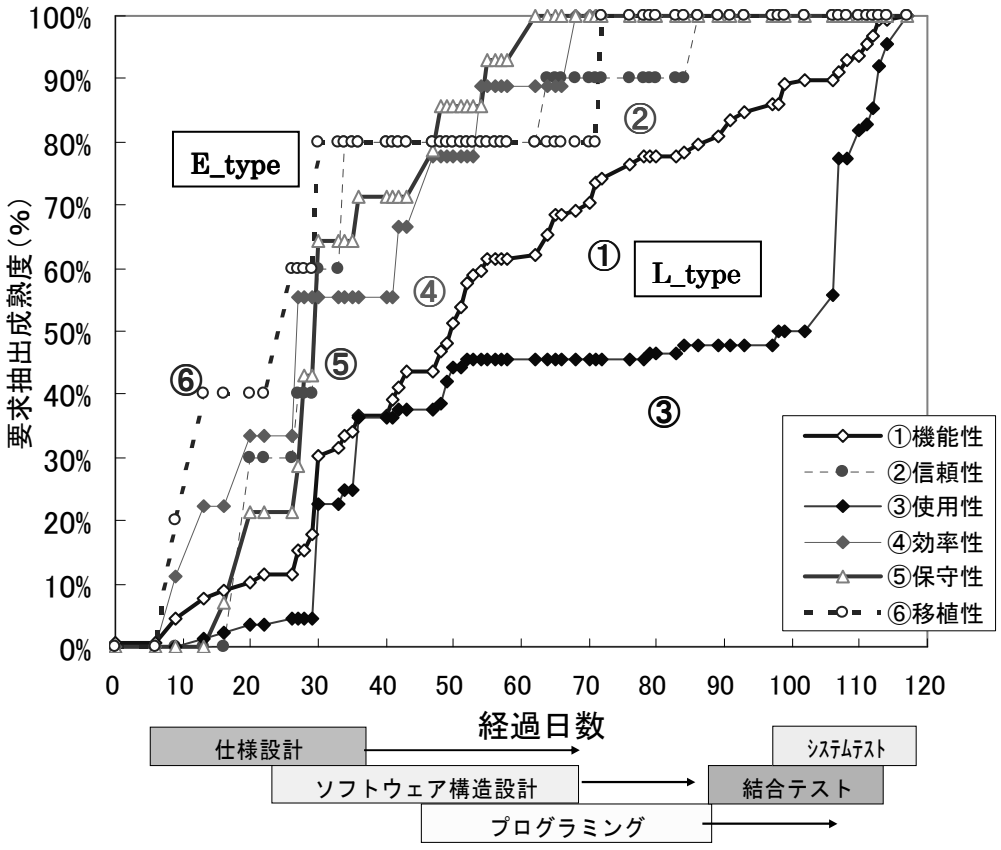


図 3.1　ソフトウェア品質特性別の要求抽出成熟度

3.2.2. ソフトウェア機能構成要素別の要求抽出

第 3.2.1 項のソフトウェア品質特性の場合と同様に，図 3.2 に，ソフトウェアの機能構成要素毎に要求抽出の成熟度を分析した結果を示す[16][17]．機能構成要素単位の分類の理由は，プロジェクトの管理単位がソフトウェア構成要素であるからである．図 3.2 の縦軸はソフトウェアの構成要素毎の要求抽出の割合（成熟度）を示す．横軸はプロジェクトの開始日からの相対的な経過日数を示す．その要求抽出の割合は，第 3.2.1 項の成熟度の計算式と同様に定義できる．第 2.3 節の図 2.13 では，要求抽出件数が少なく要求抽出の変化が分かりにくかった導入サポート機能，座席端末監視機能，センター連携機能，DB 機能が図 3.2 では要求抽出プロセスの変化が分かり易くなっている．図 3.2 から，導入サポート機能，座席端末監視機能，センター連携機能，DB 機能に関する要求抽出はソフトウェア構造設計の比較的に早い時期までに成熟している．また，座席端末機能，管理端末機能，注文受付連携機能に関する要求抽出は開発の完了まで段階的に要求を抽出し続けている．注文受付監視機能に関する要求抽出は開発の遅い時期に突発的に成熟している．

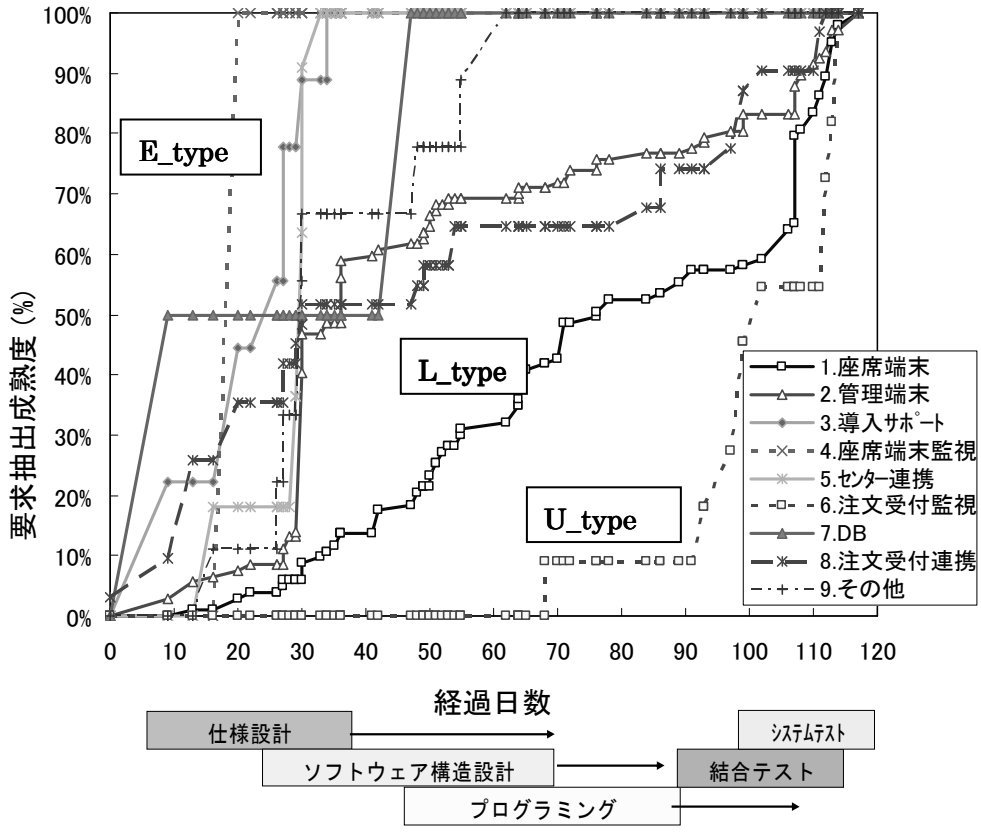


図 3.2　機能構成要素別の要求抽出成熟度

3.3. 要求抽出の成熟度タイプの定義

第 3.2 節において，ソフトウェア品質特性および機能構成要素別の要求抽出プロセスの成熟度の分類から，要求抽出プロセスには，開発の早い時期に成熟する，開発の開始から終了まで段階的に成熟する，開発途中で突発的に成熟するなど，大きく 3 つの成熟度タイプがあることを明らかにした（図 3.1 および図 3.2）．本節では，この要求抽出プロセスの 3 つの成熟度タイプを以下のように定義する．

- **早期成熟タイプ(E_type)**：

  開発の早い時期で要求抽出を完了するプロセスである．このタイプは，既存システムを再利用し要求変更が少ない構成要素や，要求定義において殆ど抽出されたシステム拡張および導入サポートなどの構成要素，および共同開発先から提供されるデータ依存型の構成要素の場合に該当する（図 3.2）．品質特性からみると，要求定義において殆ど抽出された保守性，移植性などはこのタイプに該当する（図 3.1）．

- **段階的成熟タイプ(L_type)**：

  開発の始めから終わりまで継続的かつ段階的に要求抽出するプロセスである．このタイプは，要求変更を受け易いユーザインタフェースを提供する構成要素や，機能の多様性を持つ構成要素，他社のサービス動向に影響を受ける構成要素の場合に該当する（図 3.2）．品質特性からみると，多様な機能をもつ機能性，顧客の要求変更を受け易いユーザインタフェースなどの使用性はこのタイプに該当する（図 3.1）．

- **突発的な成熟タイプ(U_type)**：

  開発途中に予期せずして起こる突発的な（想定外の）要求抽出のプロセスである．このタイプは，他システムとの連携する外部インタフェースのような構成要素，つまり，要求抽出の相手や開発資源に依存し予測不可能な要求をもつ構成要素の場合に該当する（図 3.2）．

3.4. 要求抽出の成熟度タイプ毎の分類

本節では，実例プロジェクトにおける品質特性および機能構成要素から見た要求抽出の成熟度タイプ毎の分類を述べる．

3.4.1. 品質特性から見た要求抽出

1）E_type の要求抽出

- **保守性**

  経過日数の 37 日目までに 70%，62 日まで 100%，つまり仕様設計工程において要求抽出が行われており，E_type に該当する．仕様設計では，エラーコードの定義，メニュー切替方式，構造化設計などの要求を抽出し，構造設計では，外部インタフェース機能のコンポーネント化による置換可能な設計など開発者の経験から要求抽出を実施していた．

- **移植性**

  経過日数の 30 日目までに 80%，つまり仕様設計工程において要求抽出が行われており，E_type に該当する．仕様設計では，システム導入時のシステム環境設定の要求を抽出したが，72 日目のプログラミング工程において，端末の初期設定情報の不足分の要求抽出を行っていた．

2）L_type の要求抽出

- **機能性**

  経過日数の 30 日目までに 30%，56 日まで 60%，86 日まで 80%，つまり，開発の開始から終了まで常に要求を抽出し続けていた．多様な機能を開発する必要があるが，既存システムの開発ドキュメントが殆どなく，また，ドメイン知識もないため，一気に開発することを避け，顧客参加による開発を推進し，要求抽出を行っており，L_type に該当する．

- **使用性**

  経過日数の 52 日目までに 50%，107 日まで 77%，つまり，開発の開始から終了まで常に要求を抽出し続けていた．多様な機能と連動し，画面操作などはユーザフレンドリーの機能運用であるため顧客の要求変更を受けやすく，ドメイン知識もないため，顧客参加による開発を推進し，要求抽出を行っており，L_type に該当する．特に，経過日数 105 日目のシステムテスト工程では要望事項として新たに要求を抽出した．

3）U_type の要求抽出

図 3.1 では，U_type の要求抽出プロセスが品質特性からは把握できない．これは，後述の第 3.4.2 項で述べる U_type に該当する注文受付監視機能が機能性の要求抽出に

吸収されているからである.

3.4.2. 機能構成要素から見た要求抽出

1）E_type の要求抽出

- **DB 機能**

  経過日数の 10 日目までに仕様が固まっていたが，45 日目で，データ量に伴う DB 構造の見直しなど資源効率の要求抽出を行った. 開発リーダへのインタビューによると，本来，ＤＢの論理構造を定義するプロセスが，構造設計工程であり，処理スピードの性能を考慮する必要があるからである．また，構造設計工程以前では，これらの要求が参照されることはなく，適切に意思決定していた．この DB 機能は，既存システムを再利用し拡張する構成要素であり，E_type に該当する．

- **導入サポート機能，座席端末監視機能，センター連携機能**

  経過日数の 10 日目から 30 日目まで，即ち，仕様設計工程でほぼ要求抽出が完了している．開発リーダへのインタビューによると，第 2 章の図 2.4 の対策 1，対策 2 で挙げた運用現場の視察および業務サービス内容の事前調査や, 元請けの開発への積極的な参画の効果もあり, この仕様設計の段階で抽出すべき要求を確実に取り込んでいた．これらの導入サポート機能，座席端末監視機能は，既存システムを再利用し拡張する構成要素であり，E_type に該当する．また，センター連携機能は，共同開発先から提供されるデータ依存型の外部の構成要素であり，E_type に該当する．

2）L_type の要求抽出

- **座席端末機能，管理端末機能**

  第 2.3 節で述べたように，開発の開始から終了まで常に要求を抽出し続けている．特に, 画面操作などはユーザフレンドリーの機能運用であるため要求変更を受けやすく，経過日数 105 日目のシステムテスト工程では要望事項として新たに要求を抽出した. 座席端末機能は, 他社の製品と競争するユーザインタフェースを提供する構成要素であり，L_type に該当する．管理端末機能は，システム全体を制御するため多様な機能を持つ構成要素であり，L_type に該当する．

- **OES 連携機能**

経過日数 85 日目と 95 日目のプログラミング工程と結合テスト工程に要求抽出の変化が見られる．開発リーダへのインタビューによると，複数メーカの内，2 番目の OES メーカとのシステム連携を始めたタイミングである．OES 連携は売れ筋の OES メーカを優先的に開発し流用可能となるように標準コンポーネントを意識して開発したが,接続してみて初めて得られるインタフェース仕様の違いがあった.結局は,その要求変更を受け入れ，個別にコンポーネントを開発する意思決定をしていた．また，個別にコンポーネントを開発する意思決定をしたが，プロジェクトの納期に影響しなかった理由は，複数の OES との連携を想定しており，OES 連携の開発担当を通常の 2 倍（2 名）を確保し，納期遅延が発生しないように開発当初から担当させるリスク対策を行っていた．この OES 連携機能は，競合他社のサービス動向に影響を受ける構成要素であり，L_type に該当する．

3）U_type の要求抽出

- **OES 監視機能**

経過日数 70 日目のプログラミング工程に要求抽出の変化が見られる．開発リーダへのインタビューによると,開発当初から注文受付システム監視機能を考慮していた.しかし，ある OES メーカが提供するインタフェース仕様にはない情報がわかり，運用上，注文受付システムの監視機能を変更せざるを得なかったからである．この要求抽出は仕様設計工程での OES のマニュアル調査では抽出できないインタフェース仕様であった．マニュアルに書いていないインタフェース仕様であり，OES メーカによってはその機能がない．これは接続してみて初めて得られる情報である．これらの情報を如何に早く抽出するかが課題である.実例プロジェクトではプログラミング工程以降に OES メーカとの接続を実施しているが，元請けによる営業戦略上の都合で契約が遅れたのも事実である．この OES 監視機能は，他社の構成要素に依存し予測不可能な要求をもつ構成要素であり，U_type に該当する．開発者からすると初物との接続は構造設計時点でプロトタイプ的に開発したいところである.実例プロジェクトでは，この U_type の要求抽出が少なかったことが幸いしていた.

3.5. 調査結果から得られた知見の総括と考察

本節では，第 2 章の実例プロジェクトの調査結果から得られた知見と本章で得られた知見を総括し，要求抽出とリスクに対するプロジェクトマネジメントの視点で考察する．

1）要求抽出は開発者側から積極的に働きかけるべき

第 2.4 節の考察で述べたように，要求抽出のタイミングは，単に，発注者側から要求を受けるのではなく，開発者側の作業分担，開発資源（要員，資源，設備環境など）やソフトウェア構造の特性に応じて，必要な時期をコントロールするべきである．このことによって，適切な時期に，必要な要求を積極的かつ効率的に抽出ができ，開発作業の手戻り発生を軽減することができるため，プロジェクトマネジメント上のリスク対策として有効である．近年のシステム開発は複雑化しており，開発の初期で全ての要求を決めることはできない．その状況下でウォータフォール型の開発プロセスモデルを適用して開発の初期段階で無理やり要求を定義するのは危険である．

2）要求抽出の計画を立て効率的なプロジェクト運営を実践すべき

開発途中での要求変更，追加，削除などの要求抽出は，通常，プロジェクトのスコープ，タイム，品質に大きな影響を与えるリスクとなる[27]．第 2.4 節の考察で述べたように，開発者は無闇に要求を抽出するのではなく，開発作業に必要な要求をどの時期で抽出するかを計画するべきである．それによって，効率的なプロジェクト運営が期待できる．また，計画的な要求抽出は，想定内の事象であり，必ずしもプロジェクトの納期遅れ，品質低下の要因にはならない．何故ならば，第 2 章で調査した実例プロジェクトのように，要求抽出ができなかった場合を想定して，対応プロセスと対応計画をスケジュールに組み込むことができるからである．経験を積んだプロジェクトマネジャであれば実践可能である．

3）機能が多いサブシステムはインクリメンタルな開発を実践すべき

実例プロジェクトのサーバ機能，管理端末は，注文管理システムの全体をコントロールしており，サブシステム間のインタフェースが多く，全ての機能を一気に開発することは困難であり，かつ危険である．従って，開発の優先順位を考慮し，開発開始から完了するまで，常に，要求を抽出し続けながら開発を進めていた．つまり，全体の開発プロセスモデルはウォータフォール型[6]というより，リスクを回避するため，実際は，新

たな要求を追加しつつも，段階的に機能追加していくインクリメンタルな開発[8]を実施しており，適切な開発プロセスを選んでいると言える．

4）既存の外部システムとのインタフェースは上流工程で解決すべき

実例プロジェクトでは,外部システムとの連携接続がプログラミング工程以降であり，マニュアルには書かれていないインタフェース仕様が接続して初めて判明する要求があった．外部システムのインタフェース仕様は既存であり，画面操作のように人的要因によって要求変更は発生しない．従って，開発の初期の段階でプロトタイプ的に接続検証を行い，リスクを軽減するプロジェクトマネジメントを実践すべきである．それによって，既存の外部システムとの連携では品質が確保でき，結合テスト以降の作業を更に効率的に実施できる．ただし，実例プロジェクトのように外部システムのメーカと競合関係にある場合，営業戦略上の共通ゴールの設定と早期契約が前提となる．

5）ユーザインタフェースは置換可能なソフトウェア構造にすべき

実例プロジェクトでは，仕様設計工程でユーザインタフェースの画面操作をプロトタイプ的に開発し，元請けの承認を得ていた．しかし，システムが見える形にできあがると，より良いシステムへの新たな要求として約 50 件の要望事項を抽出していた．その理由は，ユーザフレンドリーな画面操作は人的な要求変更を受け易いからである．この場合，置換可能なソフトウェア構造としておくことによって，手戻り作業を最小限にすることが望ましい．そうすることで，コスト超過や納期遅れといったリスクを軽減可能である．実例プロジェクトでも開発者自身の過去の経験から必然的に実践していた．

以下に，置換可能なソフトウェア構造の概略を述べる．図 3.3 は実例プロジェクトのユーザインタフェース部分のソフトウェア構造を示している．左側は既存システムのソフトウェア構造であり，コンテンツ機能は共通であるが，アプリケーション機能とメニュー機能は一体化している．そのため，ユーザ個別のソフトウェアになっており，メニューの変更が発生すれば，アプリケーションも変更しなければならない．ソフトウェアの構造上，要求変更に弱く，維持管理が大変である．それに比べ，右側の新システムのソフトウェア構造では，アプリケーション機能とメニュー機能が分離し独立している．メニューの変更が発生しても，アプリケーション機能には影響がなく，変更する必要がない．つまり，要求変更に強いソフトウェア構造となっている．

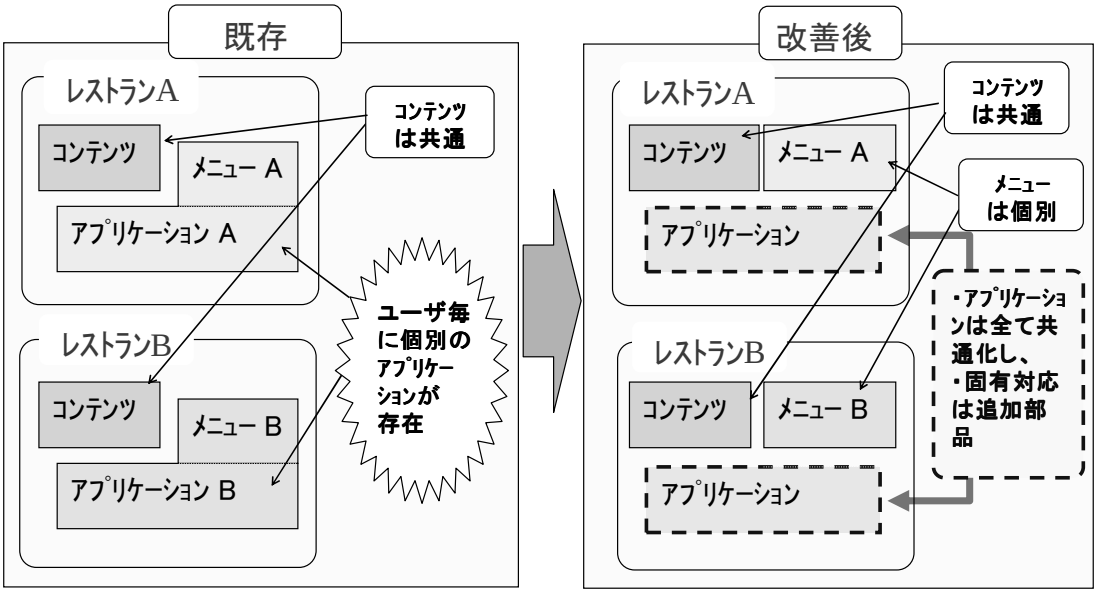


図 3.3　置換可能なソフトウェア構造

3.6. まとめ

本章では，実際のプロジェクトにおける要求抽出プロセスをソフトウェア品質特性および機能構成要素毎に分類した結果，要求抽出プロセスには 3 つの成熟度タイプがあることを明らかにした．実際のプロジェクトにおいて，品質特性または機能構成要素別にどの成熟度タイプの要求抽出プロセスを採用するかを分類し，計画するのはプロジェクトマネジャの役割である．プロジェクトマネジャがどの要求抽出プロセスの成熟度タイプに分類するかを判断する基準を明らかにする必要がある．実際のプロジェクトの調査から得られた知見から,開発するシステムの機能の規模,開発要員の業務スキルやドメイン知識レベル,顧客を含めたステークホルダーの協力関係の度合い，開発環境の準備レベルなどの開発資源に依存していることが分かった．これらの開発資源は要求抽出プロセスの成熟度タイプを分類するための判断要素であり，何れかの要素の内容が不足していれば，L_type や U_type の要求抽出プロセスが選択される．その場合，プロジェクトの納期，品質，コストへの影響を最小限にするようなリスクマネジメントが必要となる．次章では，要求抽出プロセスの成熟度タイプに伴うプロジェクトマネジメント技術を整理し，述べる．

4. プロジェクトマネジメント技術への展開

第 3 章では, 要求抽出プロセスの成熟度からみた分類の結果から, スケジュール遅延を防ぐためには，プロジェクトマネジャの役割の中で，ソフトウェアの品質特性や機能構成要素毎に要求抽出の時期を計画し，必要な時期に適切に要求抽出をすることが重要であることを明らかにした．また，要求抽出プロセスの成熟度タイプには３つのタイプがあり，どの要求抽出プロセスの成熟度タイプに分類できるかの判断要素が開発者のスキルや知識レベル，顧客を含めたステークホルダー間の協力度合いなどに依存することも分かった．

本章では，ソフトウェア機能構成要素や品質特性毎に，どの要求抽出の成熟度タイプに当てはまるか，また，どのようなマネジメントを行うのかを第 2 章で述べた実例プロジェクトのデータを基にプロジェクトマネジメント技術を整理する．

4.1. 要求抽出の成熟度モデル化と期待

第 3.5 節で述べた要求抽出に関するプロジェクトマネジメントの知見から，プロジェクトマネジャが要求抽出の計画をたて効率的なプロジェクト運営を実践するためには，そのメジャメントとなるべき指針やモデルが必要である．本節では，その概要を述べ，適用の考え方を示す．

4.1.1. 成熟度モデルの概要

第 3.3 節で定義した要求抽出プロセスの成熟度タイプは中谷等研究グループが進める PRINCE モデルに取り入れられ，進化を遂げている[18]. PRINCE モデルは「The PRINCE (Pre Requirements Intelligence Net Consideration and Evaluation）モデル」の略称であり，産学戦略的研究フォーラムから支援を請け，2007 年度から 2009 年度までの 3 年間における統合型要求工学の研究成果であり, 要求抽出プロセスのモデルである. 本項では, その PRINCE モデルの先駆けとなった要求抽出の成熟度モデルを図 4.1 に示す[19][20]. 縦軸は，要求抽出の割合（成熟度）を示し，横軸は開発開始からの相対日数，つまり経過日数を定義した．経過日数は，プロジェクトの開発工程を早い時期，中期，遅い時期と 3 つに分けている．プロジェクトの工程を分割すると早い時期，中期，遅い時期は，ウォータフォール型開発，繰り返し型開発などの開発プロセスによって異なる．

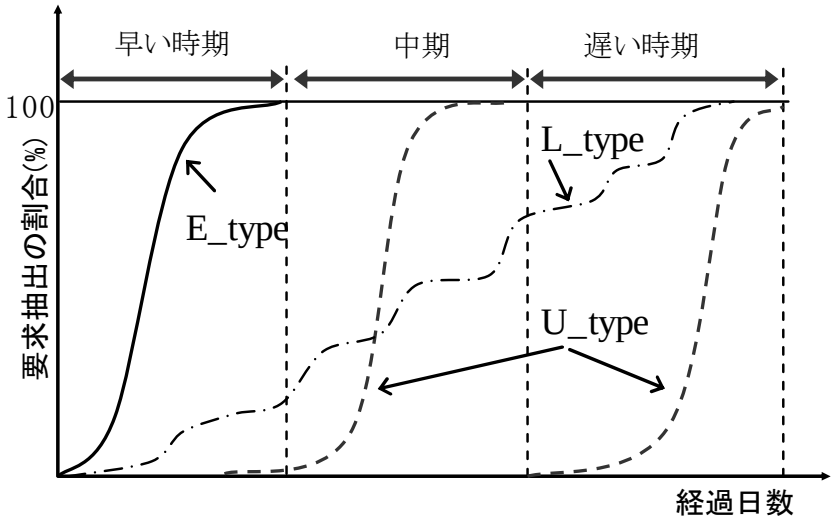


図 4.1　要求抽出プロセスの成熟度モデル

例えば，ウォータフォール型開発では，早い時期とは要求分析から仕様設計まで，中期とはプログラミングが完了するまで,遅い時期とはシステムテストが完了するまでとする．また，繰り返し開発では，早い時期とはシステム全体の機能構成の推敲まで，中期とは作り込み期間まで，遅い時期とはそれ以降に対応している．要求抽出の成熟度タイプの定義は第 3.3 節で述べたように，以下のようになる．

1)　E_type：　開発の早い時期で要求抽出を完了する成熟度タイプである．

2)　L_type：　開発の始めから終わりまで継続的かつ段階的に要求抽出する成熟度タイプである．

3)　U_type：　開発の途中に予期せずして起こる突発的な（想定外の）要求抽出の成熟度タイプである．

4.1.2. 成熟度モデルの適用への期待

本項では，プロジェクトにおいて要求抽出の成熟度モデルをどのように適用するのかについて述べる．この成熟度モデルに沿ってプロジェクト運営をする場合，ソフトウェア構成要素（例えば，品質特性あるいはソフトウェア機能構成要素）の各々を，プロジェクトの開発工程上で E_type，L_type，U_type のどの成熟度タイプに割り当てるかを計画し，実行，監視コントロールするのは，プロジェクトマネジャの仕事である．この成熟度モデルでは，すべての要求が早期成熟タイプ(E_type)で抽出できることを前提にしてプロジェ

クト計画を立案するのではなく，早期に抽出できない要求があることを前提にしてプロジェクト計画を立案し運用を行う．そして，要求の抽出時期に応じて，協力関係の会社との契約時期，要員配置の計画，テストの計画などの開発計画を立てることを重視している．

実際のプロジェクトでは，ウォータフォール型の開発プロセスのように，開発の初期段階でシステムの全ての要求を抽出するような E_type の要求抽出プロセスが望ましいと言われているにもかかわらず，L_type と U_type のような要求抽出プロセスも受入れられていた．その L_type と U_type へのマネジメントの良し悪しが影響して，プロジェクトは成功したり失敗したりしていた．従って，L_type と U_type の要求抽出プロセスをどのようにコントロールするかがプロジェクトの成否への重要な課題と捉える．開発プロセスだけでなく要求抽出プロセスのリスクを考慮したプロジェクトマネジメントの実践およびそのマネジメント技術が重要である．そのプロジェクトマネジメント技術を整理することはプロジェクトを成功に導くためのマネジメントを実践すべき開発現場のマネジャやリーダにとっては有用になると期待する．

4.2. 成熟度タイプに関連するプロジェクトマネジメント技術

本節では，実際のプロジェクトにおける要求抽出プロセスの実態調査から得られた要求抽出の成熟度タイプに伴うプロジェクトマネジメント技術を以下に整理する．

1）早期成熟タイプ（E_type）の場合

- 開発者側に既存システムのドメイン知識がなくても開発業務の経験と知識があれば，既存システムの事前調査によってソフトウェア部品の再利用や品質特性の保守性や移植性などアプリケーションに影響しない要求を開発者主体で抽出できる．また，ドメイン知識の熟知者を開発に参画させることによって開発初期からシステムへの要求を頻繁に抽出できる．顧客参画型の開発プロセス[15]によって開発者の業務ノウハウ不足を補うことが可能となる．
- 共同開発者間に利害対立がある場合，対立の解消のための顧客による早期の契約締結や協力関係の構築によって，早期にインタフェース仕様の要求を抽出可能となる．何もしなければ，開発の遅い時期での要求抽出になる可能性がでてくる．

- 複数の他システムと連携する場合,早期の契約によってインタフェースが開示されれば，そのインタフェースを理解した上で，システム間の相互運用に関する要求を開発の初期段階で抽出できる．複数社との契約締結が同時にできず段階的になったり，難航し遅い時期の要求抽出になったりした場合,契約の時期に合わせた段階的な要求抽出になる．
- 既存システムのドメイン知識や課題認識の情報があれば，要求は初期に抽出される．一方，その情報がなければ，要求を抽出する時期が遅れる．それに伴い，プロジェクトのリスクは軽減したり，増加したりする．それを解決するためには，顧客を含めたステークホルダー間の相互協力や意思疎通を図るためのコミュニケーションが重要である．

2）段階的成熟タイプ（L_type）の場合

- 開発者側に開発業務の経験があれば,急がなくてもよい機能は必要となる時期までに開発者主導で計画的に要求抽出ができる．
- 市場競争や人的要因によって要求変更を受け易い画面表示や操作系などのユーザインタフェースでは，開発の遅い時期（例えば，結合テスト工程以降）に実際にソフトウェアが出来上がって目に見えるようになれば，新たな要求が出てくるものである．従って，要求変更を受け易いユーザインタフェースは，顧客満足を満たすために，開発初期にプロトタイプをつくり，基本的な要求の認識合わせを実施し，顧客の有識者と密着しながらインクリメンタルな開発（顧客参加型開発）を行うことにより，開発作業の手戻りリスクを最小限にすることが重要である．また，画面の表示などは，要求変更に柔軟に対応できる置換可能なソフトウェア構造にするべきである.それによって,開発の後期に新たな要求変更や追加を受け入れても開発作業の手戻りリスクを軽減可能である．従って，顧客参加型の開発や置換可能なソフトウェア構造設計はプロジェクトの計画段階で計画に組み入れるべきである．
- 機能の多様性を持つ構成要素の場合，段階的な要求抽出となる．何故ならば，開発者にドメイン知識がなく，開発途中での要求変更が多々予想される中，一気に開発するのは開発作業の手戻りが多くスケジュール遅延を起こす危険性が高い．したがって，ドメイン知識を有する顧客参加による段階的な要求抽出を選択し,開発作業の手戻りによる品質低下やスケジュール遅れを防ぐためのリスクマネジメントとして L_type

となるのは合理的な判断である.

- 共同開発者間に利害対立がある場合,対立の解消のための顧客による早期の契約締結や協力関係の構築が遅れれば，契約の時期に合わせ，設備機器のインタフェース仕様の要求を抽出となる．何もしなければ，開発の遅い時期での要求抽出になる可能性がでてくる.
- 複数の他システムと連携する場合,早期契約の遅れによってインタフェースが開示されなければ，契約締結が同時にできず段階的になり，契約の時期に合わせた段階的な要求抽出になる.

3）突発的な成熟タイプ（U_type）の場合

- 開発途中や遅い時期に流用ソフトウェア部品の品質不良や既存の設備機器のインタフェース仕様に予測不可能な要求が追加されている場合などがある.これらは開発工程の結合テストなどの遅い時期に結合してみないと抽出できない要求である.要求抽出の可能な時期を予測できても具体的な要求内容までは予測できない．また，予測可能な要求には対応策が立てられるが,突発的に起こる予測不可能な要求には対応策を立てようがない．要求の抽出後，対応策の検討，立案が必要となるので，リスク受容型とも言える.
- 共同開発者間に利害対立がある場合,対立の解消のための顧客による契約締結や協力関係の構築が開発の遅い時期になれば，設計期間が短くなり，以降で予測不可能なインタフェース漏れなどの要求抽出になる可能性がでてくる．また，他の機能への影響もでてくる.経験のある開発リーダやプロジェクトマネジャはこの時期で新たな要求が発生することを想定できても，要求内容までは想定できない．従って，この時期での突発的な要求への対応は開発への影響や顧客満足を考慮し,顧客や共同開発者間で調整することになる.
- 複数の他システムと連携する場合，営業戦略上，契約締結遅れにより開発の遅い時期にインタフェース仕様が開示されれば，設計期間が短く調査が不十分となり，予測不可能なインタフェース漏れなどの要求抽出になる可能性がでてくる．また，他の機能への影響もでてくる．例えば，実例プロジェクトでは，顧客と他社が競合関係にあるため外部インタフェースの仕様開示が遅れることを想定し,仕様設計の早い時期に他社製品の購入や導入教育への参加による要求抽出を実施したにもかかわらず,結合テ

ストではマニュアルにないインタフェース仕様を新たに抽出していた．したがって，外部インタフェース仕様の開示遅れを想定した対応プロセスを計画に盛込むリスクマネジメントを実施していたのは適切であったと言える．実例プロジェクトのように，開発工程のどの時期において要求抽出すべきかを設定できれば，対応するプロセスを計画段階でスケジュールに盛り込むことは可能である．

以上から，要求抽出の内容がシステム構成上，予測できないような致命的なものであった場合，開発の手戻り作業が多大に発生し，品質や納期に影響することもある．想定できないリスクをどのように管理すべきかが課題である．リスク管理はそもそもリスクとして識別されたものに対して実施されるが，想定できないリスクは管理できない．プロジェクトマネジャは，実際にやってみないと分からないことを先行的に実施する手段がないかをステークホルダー間で検討し，問題があるかないかを早く判断する必要がある．その判断のための対応を遅らせれば，問題の先送りとなりプロジェクトのリスクは拡大するし，早めればリスクは軽減可能である．

4.3. 要求抽出元の特性による成熟度タイプへの影響

本節では，第 4.2 節のプロジェクトマネジメント技術の整理から，要求抽出プロセスの成熟度タイプの決定に影響を与える要因について記述する．プロジェクトマネジャは，要求抽出を効率的に行うために，プロジェクト計画において，ソフトウェア構成要素毎にどの要求抽出の成熟度タイプを選択するかを決定しなければならない．しかし，実際の要求抽出の時期は，要求抽出の 3 つの成熟度モデルに該当する要求抽出プロセスだけで決まるのではなく，開発者の業務経験とドメイン知識レベル，要求抽出する相手のドメイン知識度と協力度合い，共同開発者の競合関係，他社システム連携時の契約の時期などの要求抽出元の特性に影響を受け，早くなったり，遅くなったりする（図 4.2）．プロジェクトマネジャはこれらの要求抽出元の特性を把握した上で要求抽出を確実に実施するためのマネジメントを実施する必要がある．これらを整理すると以下のようになる．

- 開発者に業務知識やドメイン知識があれば，要求抽出は早くなり，知識がなければ要求抽出が遅くなる（図 4.2 の①）．しかし，開発者にドメイン知識がなくても，ドメイン知識の熟知者の協力があれば，要求抽出の時期を早くすることが可能である．
- ドメイン知識の熟知者の協力度が高ければ，早期にシステム要求を抽出できるが，協力度が低ければ要求抽出が遅れることになり，プロジェクト遅延のリスクが高くなる

（図 4.2 の②）.

- 共同開発者との競合がある場合,対立関係の解消時期によってはインタフェース仕様の入手時期が早くなったり遅くなったりする（図 4.2 の③）．対立関係の解消策は，プロジェクトスポンサーとの契約締結による縛りの強化やプロジェクト立ち上げ時のプロジェクトゴールや情報の共有とネゴシエーションである.
- 他社システム連携時の契約時期によってはインタフェース仕様の入手時期が早くなったり遅くなったりする（図 4.2 の④）．顧客と他社が営業戦略上の競合があれば，業務提携の契約が遅れる可能性が高く，インタフェース仕様の開示は遅れる．これはプロジェクト内ではコントロールできないため,顧客と他社の間で早期に解決して貰うしかない.

以上より，これらの要求抽出元の特性に応じたマネジメントによって要求抽出の成熟度タイプが調整されることになる．即ち，ソフトウェア構成要素の性質による要求抽出プロセスと要求抽出元の特性によって成熟度タイプが決定され，スケジュール遅延を防止するためのプロジェクトマネジメント技術がより効果を発揮する．また，これらの要求抽出元の特性は開発プロセスのスタイルにも影響を与え，早期に要求抽出が可能ならばウォータフォール型の開発プロセスが適用され，そうでなければ，インクリメンタル型の開発プロセスが適用されることになる.

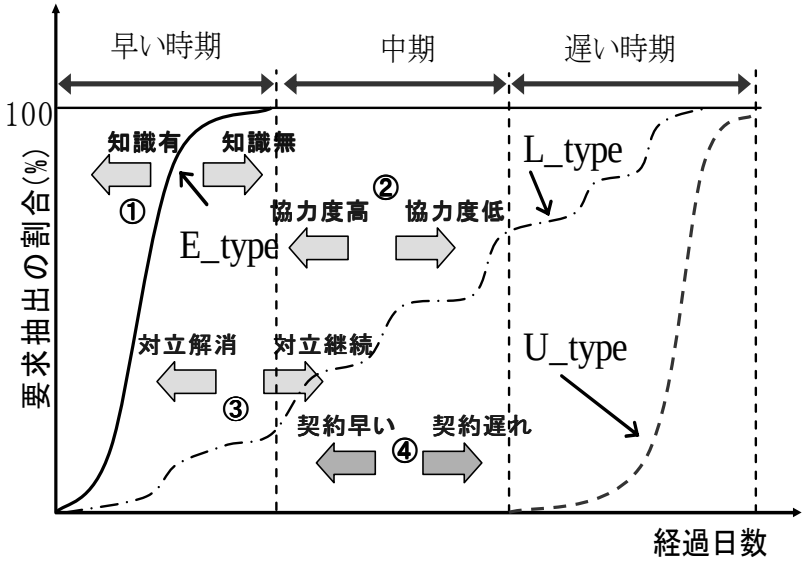


図 4.2　要求抽出の時期への影響

例えば,第 2 章で述べた実例プロジェクトにおいて,他社システム連携の構成要素の場合，顧客と他社間の営業戦略上の契約が早期に締結され，外部インタフェース仕様が入手できていれば，早期成熟度タイプ（E_type）に近づけたと考える．あるいは，過去に他社システム連携の接続実績があれば，そのまま流用できるため，早期成熟度タイプ（E_type）になる．逆に，顧客と他社間の営業戦略上の契約が遅れ，開発の遅い時期に締結されると，要求抽出プロセスの成熟度タイプは突発的な成熟度タイプ(U_type)となる可能性がある．つまり，要求抽出の相手をコントロールできれば要求抽出を早めたり，遅くしたりすることが可能となる．ただし，要求抽出元の準備ができていない時点での無理な要求抽出は却って品質低下によるスケジュール遅れを招くこともある．

4.4. 要求抽出の典型的なマネジメント技術

本節では，プロジェクトマネジャが，要求抽出プロセスの成熟度モデルを使って，要求抽出を計画し，実行，監視コントロールするための典型的なプロジェクトマネジメント技術を整理する．

第4.2節で述べたように，要求抽出の成熟度モデルに沿って要求抽出の成熟度の時期をコントロールするプロジェクトマネジメント技術は，以下の３つである．

- 早い時期の要求抽出
- 段階的な要求抽出
- 開発途中の突発的な要求抽出

また，第4.3節で述べたように，要求抽出の成熟度の時期は，開発者の業務経験とドメイン知識レベル，要求抽出する相手のドメイン知識度と協力度合い，共同開発者の競合関係，他社システム連携時の契約の時期などの要求抽出元の特性に影響を受け，早くなったり，遅くなったりする．これらの要求抽出元をコントロールするプロジェクトマネジメント技術は，以下の４つである．

- 開発者自身からの要求抽出
- ドメイン知識の熟知者からの要求抽出
- 利害対立する共同開発者からの要求抽出
- プロジェクト外からの要求抽出

その他に，エンドユーザ，市場ニーズ，法律などの要求抽出元があるが，本論文では，実際のプロジェクトの調査では該当しなかったため，対象外としている．

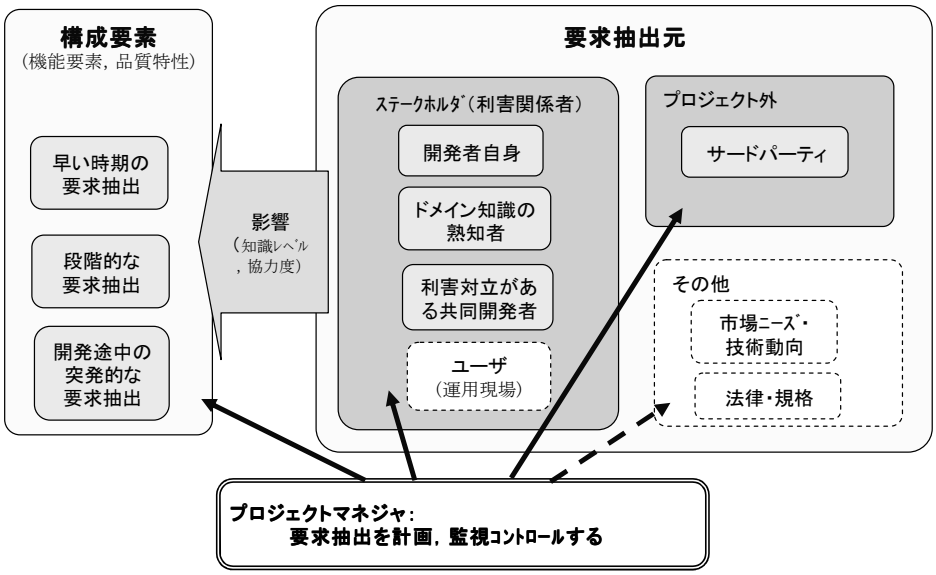


図 4.3　典型的な要求抽出のマネジメント技術

図 4.3 に，プロジェクトマネジャが，ソフトウェア構成要素の性質からの要求抽出と要求抽出元からの要求抽出をコントロールするマネジメント技術の関係を示す.

以上より，ソフトウェアの構成要素や品質特性の要素の性質によって要求抽出の成熟度の時期を計画コントロールする３つと要求抽出元の知識レベル・協力度によって要求抽出の成熟度の時期を計画コントロールする４つとを合計して７つを典型的なプロジェクトマネジメント技術とする．ここでいう典型的とは，実際の開発現場でもっとも頻繁的に使用されるという意味である．これらの有用性や適用性については後述の第 5.3 節で検証する.

4.5. まとめ

本章では，要求抽出の時期がソフトウェア機能構成要素や品質特性要素の性質から決まる成熟度タイプだけでなく,要求抽出元の特性によっては要求抽出の時期が早くなったり，遅くなったりすることを明らかにした．プロジェクトマネジャは，要求抽出に伴うスケジュール遅れを防止するために，その要求抽出元の特性を把握し，要求抽出プロセスの成熟度モデルを使って，要求抽出を計画し，実行，監視コントロールしなければならない．そのための典型的なプロジェクトマネジメント技術を 7 つ整理した．次章では，経験の浅いプロジェクトマネジャやリーダでも，整理した 7 つの典型的なプロジェクトマネジメント技術を再利用できるようにするために，Alexander のパターンの概念[24]を取り入れ，プロジェクトマネジメント・パターンとして整理し, その適用性について検証し, 提案する.

5. プロジェクトマネジメント・パターンの提案

第 4 章では，実例プロジェクトのデータを基に得られた知見から，要求変更が原因でスケジュール遅延を防止するための典型的な 7 つのプロジェクトマネジメント技術を整理した．これらは，従来，開発現場において経験と勘で実施されていたプロジェクトマネジメント技術である．

本章では，経験の浅いプロジェクトマネジャやリーダでも，抽出した 7 つの典型的なプロジェクトマネジメント技術を再利用できるようにするために，プロジェクトマネジメント・パターン（以降，Project Management パターン：PM パターンと呼ぶ）として整理する．これらの PM パターンは，全ての建物はユニークであるが個々は一般的なパターンの集まりからなるという Alexander のパターンの概念[24]を取り入れている．また，これらの PM パターンが他のソフトウェア開発プロジェクトにおいても同様に使用できるプロジェクトマネジメント技術であるか否か，その適用性について検証し，典型的な 7 つの PM パターンとして提案する[19][21][22][23]．

5.1. ＰＭパターンの研究の目的

ソフトウェア開発の要求抽出は第3章で述べたE_typeの要求抽出プロセスが望ましい．しかし，実際のプロジェクトでは，L_typeとU_typeの要求抽出プロセスがスケジュール遅れやプロジェクト崩壊を引き起こす可能性があるにもかかわらず，受入れられている．その結果，プロジェクトは成功したり失敗したりしている．従って，L_typeやU_typeの要求抽出に対するプロジェクトマネジメント方法や技術を明らかにすることは開発現場の経験の浅いプロジェクトマネジャやリーダにとっては効果があると期待する．開発途中の要求の追加や変更も，それらを受け取る時期を誤らなければプロジェクトのスケジュール遅延を避けることが可能である．実際のソフトウェア開発プロジェクトにおいて，どのように要求が抽出され，マネジメントされているのかを明らかにすることによって，効率的，かつ効果的な要求抽出のマネジメントが実施できると考える．

本章では，その要求抽出に伴うプロジェクトマネジメント技術を再利用できるようにPMパターンとして整理し，他のプロジェクトでも適用可能か否かを検証する．

5.2. ＰＭパターン

本節では，PM パターンの定義を述べ，第 4 章で整理した要求抽出に起因したスケジュール遅れを防止するための典型的なプロジェクトマネジメント技術を他のプロジェクトでも再利用できるように PM パターンとして定義し，提案する．

5.2.1. ＰＭパターンの項目定義

PMパターンを定義するにあたり，全ての建物はユニークであるが個々は一般的なパターンの集まりからなるというAlexander のパターンの概念[24]をベースにHillside Group が提唱しているパターンテンプレート[40]の項目の中から，以下の6項目を適用してパターン構成を記述した．

- パターン名： パターンを識別する名前．
- 背景 ： パターンを使うことが想定されている利用者を限定する背景の説明．これで，問題領域を特定する．
- 問題 ： パターンが解決する問題の記述．
- フォース ： このパターンの解決策が有効となる状況や制約．問題の解決策を選択するために，さらに問題領域を限定する．
- 解決策 ： 問題に対する解決策．
- 問題解決後の状況： パターンを適用して問題を解決した後に，新たに発生する可能性のある問題．

ここで定義した PM パターンの構成要素以外に，文献[40][41]には，“合理性”，“例”，“関連”などがあるが，上記の 6 項目にしたのは，プロジェクトマネジメントの経験がある技術者であれば，これらの 6 項目のみで PM パターンを適用することが可能であると考えたからである．

5.2.2. ＰＭパターンの提案

第4章ではプロジェクトを成功させるために様々な要求抽出のプロジェクトマネジメント技術を整理した．これらの結果をベースに，要求変更によるスケジュール遅延を防止するための典型的な７つのプロジェクトマネジメント技術を他でも再利用できるようにPM パターンとして整理し提案することは，開発現場のプロジェクトマネジャにとって役に立つと考えている．そのPMパターンは要求抽出の特徴から２つのPMパターン群に分類する

ことができる．１つ目は要求抽出の時期をコントロールし要求抽出するPMパターン群であり，２つ目は要求抽出元をコントロールし要求抽出するPMパターン群である．それぞれ，要求抽出時期PMパターン群と要求抽出元PMパターン群と定義する．以降のT-i，S-iの記号はPMパターン番号である．

1）要求抽出時期PMパターン群

このPMパターン群には以下の3つのPMパターンがある．

T-1）早い時期の要求抽出

T-2）段階的な要求抽出

T-3）開発途中の突発的な要求抽出

2）要求抽出元PMパターン群

このPMパターン群には以下の4つのPMパターンがある．

S-1）開発者自身からの要求抽出

S-2）ドメイン知識の熟知者からの要求抽出

S-3）利害対立する共同開発者からの要求抽出

S-4）プロジェクト外からの要求抽出

要求抽出時期PMパターン群内のパターン間はそれぞれ独立した関係であり，同時期に実行されることはない．要求抽出元PMパターン群内のパターン間もそれぞれ独立した関係ではあるが，それぞれのPMパターンは同時期に実施されることがある．

5.2.3. ＰＭパターンの記述

本項では，第 5.2.2 項で示した要求抽出に伴う PM パターンを第 5.2.1 項で定義したパターンテンプレートに従って記述する．

T-1）早い時期の要求抽出

–背景：短い開発期間で顧客および市場ニーズの変化に対応したシステムを開発する．顧客の要求やシステムへの要求は様々であり，早期に要求抽出できるもの，できないものがある．

–問題：早期に全ての要求を抽出し，一気に開発できるものではない．

– フォース：全ての要求が出揃うまで，開発着手を遅らせるわけにはいかない．納期を守るために，進められるところは進めなければならない．
– 解決策：早い時期に抽出可能な要求，そうでない要求を分類しながら，早い時期に流用可能なソフトウェアコンポーネントやアプリケーションに影響しないソフトウェア機能構造の要求を抽出する．また，アプリケーションに関係ない保守性や移植性などの品質要求は開発者主導で要求抽出する．
– 問題解決後の状況：流用可能なソフトウェアコンポーネントの品質が悪ければ，開発の最終段階で問題が生じる可能性がある．

T-2）段階的な要求抽出

– 背景：顧客および市場ニーズの変化に対応した多様な機能を持つシステムを開発する．しかし，ソフトウェアの機能構成要素の特性によっては早期に要求が決まらず抽出できないものがある．
– 問題：要求が曖昧で且つ不適切な時期の要求抽出は却って品質低下およびスケジュール遅れのリスクがある．
– フォース：全ての要求が出揃うまで，開発着手を遅らせるわけにはいかない．品質低下およびスケジュール遅れのリスクが大きい場合，多様な機能を一気に開発することを回避しなければならない．
– 解決策：ドメイン知識を持つ熟知者を開発に巻き込み，熟知者とタイムリーにコミュニケーションが取れる状態を作り，適切な時期に段階的に要求を抽出する[9][15]．急がなくてよい機能は必要となる時期までに開発者主導で計画的に要求抽出する．
– 問題解決後の状況：熟知者を適切にコントロールし要求抽出しなければ，品質低下およびスケジュール遅れのリスクが増大する．

T-3）開発途中の突発的な要求抽出

– 背景：開発途中や遅い時期での要求抽出となるソフトウェア構成要素があるシステムを開発する．開発途中や遅い時期での予測不可能な要求変更は納期および品質を約束できない可能性がある．例えば，他社システムとの連携や新しい機器との接続があるシステムを開発する場合，システム連携や機器接続はシステムテスト工程のように開発の遅い時期において，予測不可能な要求を抽出することがある．

– 問題 : 要求抽出の時期は想定できても突発的に起こる予測不可能な要求内容までは想定できない．開発の最終段階での要求変更は納期を守ることができない可能性がある．

– フォース : 契約上，約束した納期は守らなければならない．

– 解決策 : 開発途中で抽出した予測不可能な要求を分析し，開発責任者を交えて受入れるべきか次のバージョンに残すべきかを判断する．また，要求変更が発生することを予測し，変更に耐え得る置換可能なソフトウェア構造やコンポーネント化を実践する．

– 問題解決後の状況 : 要求抽出による変更範囲や変更規模などの影響を充分に検討しなければ，品質低下およびスケジュール遅れのリスクが発生する．

S-1）開発者自身からの要求抽出

– 背景 : 顧客および市場ニーズの変化に対応した多様な機能を持つシステムを開発する．開発者にはシステムのドメイン知識はないが，開発の業務経験や知識はある．システム開発にはドメイン知識および業務知識が不可欠である．

– 問題 : ソフトウェアの観点から要求抽出，分析しなければ，顧客要求を実現可能かどうか約束できない．

– フォース : 無闇に顧客要求を受け入れるのは危険であり，品質低下およびスケジュール遅れを招く危険があり，要求の分析は不可欠である．

– 解決策 : 開発の業務経験や知識を活かし，ソフトウェアの観点からの要求抽出，分析を行う．例えば，流用可能なソフトウェア部品，ソフトウェア構造を決める要求を抽出する．

– 問題解決後の状況 : 開発者自身の思い込みにより的外れの要求抽出の可能性がある．顧客を含めたレビューを必ず実施する必要がある．

S-2）ドメイン知識の熟知者からの要求抽出

– 背景 : 顧客および市場ニーズの変化に対応した多様な機能を持つシステムを開発する．開発者には開発の業務経験や知識はあるがシステムのドメイン知識はない．ドメイン知識を有し既存システムを熟知しているのは顧客のみである．

– 問題 : システムの仕様およびドメイン知識がなければシステム機能構築や改善は困難であり，品質および納期を約束できない．

–フォース：システム要求の品質を確保するためにはドメイン知識を持った熟知者の協力が不可欠である．

–解決策：ドメイン知識の熟知者の協力を得てシステムへの要求を抽出する．

–問題解決後の状況：顧客の作業負荷が上がり，クレームが生じる可能性がある．顧客との契約段階で顧客の全面的な協力を得ておく必要がある．

S-3）利害対立する共同開発者からの要求抽出

–背景：顧客を中心に複数の請負業者が相互協力してシステムを共同開発する．しかし，請負業者間には競合関係にあり，利害対立する．契約上は顧客と個々の請負業者間で結ばれ，請負業者間での直接的な契約はない．

–問題：請負業者間の対立により意思疎通が悪ければ，開発に必要な要求が抽出できず，また，開発効率が悪く，スケジュール遅れや品質低下を引き起こす可能性がある．

–フォース：開発の初期段階で請負業者間の対立関係を解決しなければならない．

–解決策：スポンサーである顧客がビジネスゴールや請負業者間の協力メリットを明確にし，早期に協力関係を築く．請負業者間でプロジェクトの問題解決や進捗状況を共有し，意思疎通を図る．

–問題解決後の状況：共有情報に関してセキュリティ問題が生じる可能性があり，リスクを考慮しておく必要がある．

S-4）プロジェクト外からの要求抽出

–背景：複数の他社システムとの連携や新しい機器との接続があるシステムを開発する．あるいは，サードパーティのソフトウェア製品などを使用しシステム構築する．開発プロジェクト内には複数の他社システムとの連携や機器のインタフェース仕様の情報がない．

–問題：インタフェース仕様がなければシステム連携や機器接続はできない．

–フォース：何とかして確実なインタフェース仕様を抽出しなければならない．

–解決策：インタフェース仕様の開示を顧客経由で依頼し（ネゴシエーション），何とか要求仕様を抽出する．また，インタフェース仕様の開示遅れを想定し，その対策を計画に盛込む．

–問題解決後の状況：抽出したインタフェース仕様が不十分であれば，インタフェース・ミスの問題が結合テストで生じ，スケジュール遅れになる可能性がある．

5.3. ＰＭパターンの適用性の検証

本節では，第 5.2.2 項で提案した 7 つの典型的な PM パターンが他のソフトウェア開発プロジェクトにおいても同様のプロジェクトマネジメント技術として使われているかどうか，つまり，PM パターンの適用性があるかどうか，あるいは典型的な PM パターンであるかを検証する．以降において，３つの分野から検証対象にした実際のプロジェクトの概要を述べ，それぞれのプロジェクトの問題，その解決策を挙げ，どの PM パターンに該当するかを検証する．（文中の T-1，T-2 などの記号は問題と解決策に対するＰＭパターン番号を表す．）

第 1 番目はエンジニアリングシステム分野，第 2 番目はビジネスシステム分野，第 3 番目は組み込みシステム分野の開発プロジェクトである．このように，異なる分野の開発プロジェクトにおいても PM パターンが当てはまることを検証する．

1) 自動倉庫システム

エンジニリングシステム分野の一例として，ＦＡ（ファクトリ・オートメション）の自動倉庫システムを採り上げる．これは，文書類の保管効率を向上させるための立体自動倉庫の運用制御システムである．全体管理サーバ，管理端末，事務所端末，およびバーコードハンディターミナルのソフトウェア開発であった（図 5.1 の点線枠内）．機能的には文書類の入出荷指示を下位設備機器に投げ，監視コントロールする．また，文書類の入出荷情報を定期的に上位の情報システムに送信する．開発規模は，標準の自動倉庫システム（ソフトウェアパッケージ)を顧客要求である文書類の保管システムにカスタマイズする比較的大きな開発であり，ソースコードで 202 千行，開発期間は 12 ヶ月であった．

以下，開発上の 3 つの問題と解決策を述べる．

- 問題１：開発開始時，顧客内でシステムの運用イメージ，業務フローが整理されていないため，標準システムをどのようにカスタマイズするか要求仕様がまとまらない．

  解決策：現場の運用調査後，運用イメージ，業務フローを整理しながら，段階的に顧客要求を抽出していた(T-2)．要求仕様を決定する仕様設計の段階から顧客を巻き込んで一緒に設計を進めていた (T-2) (S-1) (S-2).

- 問題２：このシステムには様々な設備機器とのインタフェースがあるが，初物の設備機器（例えば，図 5.1 のモービルソータ）との接続もあり，接続制御の経験者もいない．（モービルソータとは自動搬送コンベア間の無人搬送を行う機器である．）

  解決策：設備機器の担当会社に，仕様設計段階からインタフェース仕様の開示を依頼した(S-4)．インタフェース仕様書は，開発初期の仕様設計段階でほとんど決定していた(T-1)が，初物の設備機器との接続であるためインタフェース仕様に漏れがないかを熟練者の確認によって補っていた(S-5).
- 問題３：設備機器との結合テスト段階で，設備機器側はインタフェース仕様どおりに開発されていないことが判明．

  解決策：遅い時期での予測不可能な要求抽出となったが(T-3)，この時期での設備機器側の改修では，システムの本番稼動開始が遅れ，顧客が手配している運用機材，要員の調整に影響するだけでなく，機会損失にもなりかねない．その問題を解決するために，アプリケーション側のソフトウェアの変更の影響範囲調査し，その結果，開発期間内で対応可能と判断し，早急に対応していた(T-3).
- 結果：上記の問題 1 から問題 3 に関係なく，顧客都合によって納期が 2 ヶ月遅れたが，開発者側の責任によるスケジュールの遅れはほとんどなかった．
- 考察:過去に接続実績のある設備機器とのインタフェース仕様は開発の早い段階で要求抽出ができる（T-1）のは当然だが，初物の設備機器との接続には予測できない問題が発生する可能性があるため,開発リーダが早めのインタフェース仕様の開示要求を行っていたのはリスクの極小化として合理性がある．また，顧客が運用イメージ，流れを整理できないことに対し，顧客と一緒に運用イメージ，業務フローを整理することによって，顧客との信頼関係を築いていた．これは，顧客満足度を向上させるために，より品質の高いシステムを構築する活動であり，まさに，第 1 章で定義した顧客参加型の開発プロセスを重視した統合型要求プロセスの実践である．

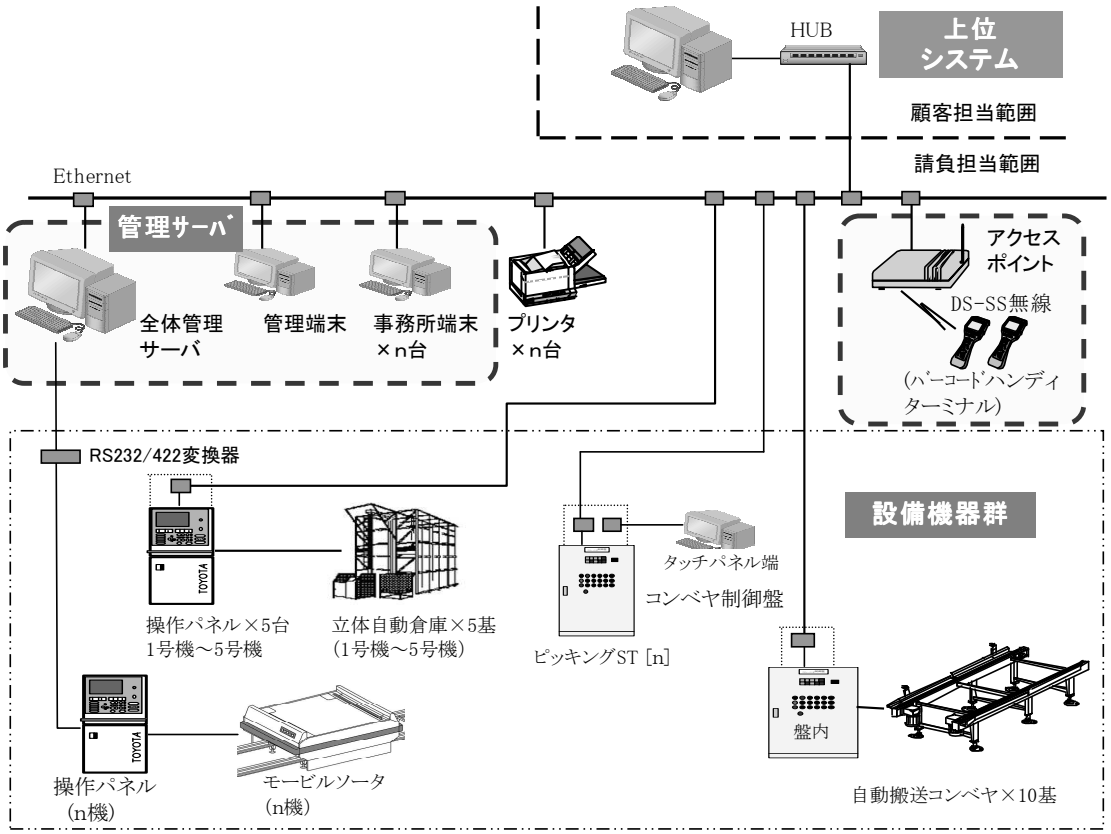


図 5.1　自動倉庫システム構成図

2) 統合医療事務システム

ビジネスシステム分野の一例として，統合医療事務システムを採り上げる．これは総合病院内の様々な既存医療システムと新たなシステムを統合する再構築の開発であった(図5.2)．新システムの基本的な部分は医療パッケージ・ソフトを使用した開発であり，個々の医療システムをどのように接続し再開発するかがプロジェクトの課題であった.個々の医療システムは様々なメーカの製品があり，既存システムをそのまま使用，他社製品への切り替え，同じメーカ製品のバージョンアップなどを行い，これらを病院内ネットワークで結ぶものであった．この開発プロジェクトでの役割は元請けの位置づけであった．開発規模は 100 千行，開発期間は 5 ヶ月であった.

以下，開発上の 2 つの問題と解決策を述べる.

- 問題 1 ：開発担当者に医療パッケージ・ソフトについての知識が全くなかった.

解決策：顧客および医療パッケージ・ソフト・メーカと一緒に設計作業グループを立上げ（分科会の設置），利害関係者間の調整を行い，知識を補った(S-2)(S-3)．標準パケ

ージを活用した運用仕様を顧客に確認しながら，段階的に要求を抽出していた(T-2) (S-2). 一方，開発の早い段階で，顧客への事前インタビューと運用調査によって，既存の医療事務システムの要求を抽出していた(T-1).

- 問題 2 : 個々の医療システム・メーカと顧客との契約時期が遅れ，医療パッケージとのインタフェース仕様の要求抽出ができない.

  解決策 : 元請けの権限を活かし，医療システム・メーカへの強い協力要請により迅速に要求を抽出した（S-3）(S-4）が，契約が完了したメーカの医療システムから段階的にインタフェース仕様の要求抽出していた（T-2）．その中には，契約が想定以上に遅れ，要求抽出が開発期間の後半になった (T-3)が，顧客とのスケジュール調整を随意実施していた.

- 結果： 納期が 2 か月遅延したが，顧客が開発の経緯を理解しており，大きな問題にはならなかった.

- 考察 : 複数の医療システム・メーカとの契約は一気にできないため，契約が完了したメーカから段階的にインタフェース仕様を抽出するのは妥当である. 開発開始時点から分科会を立ち上げ，顧客や医療パッケージ・メーカを巻き込んでの情報共有は要求仕様の食い違いを避けるための解決策として適切である.

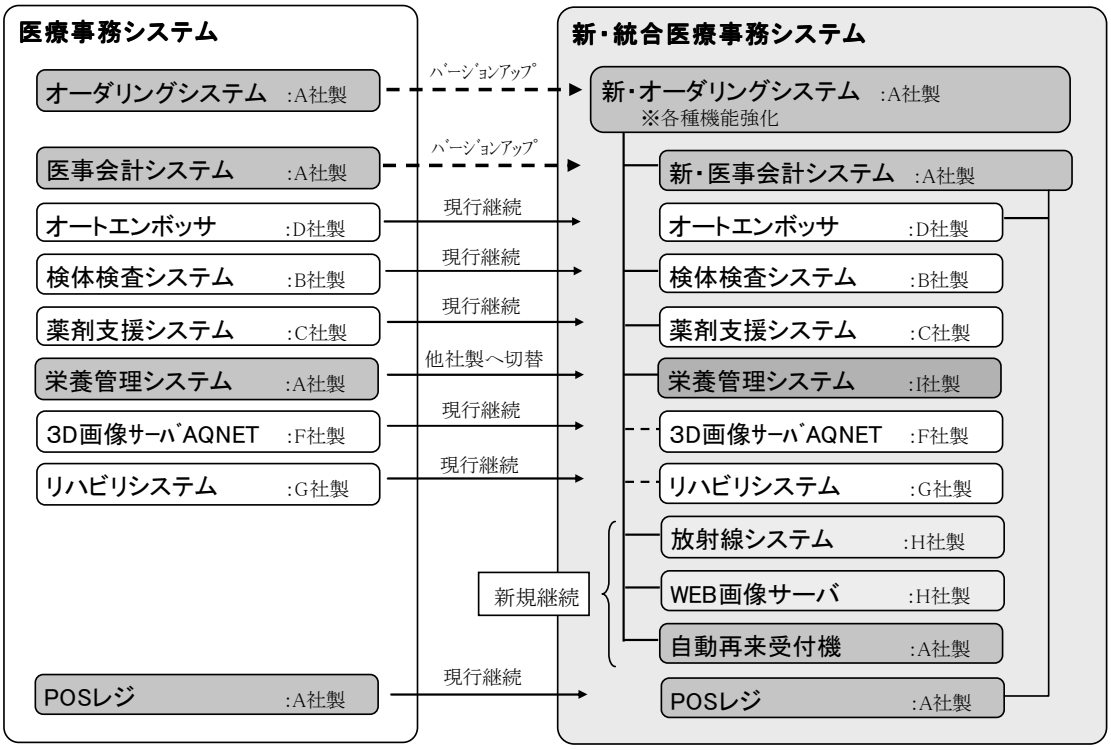


図 5.2　統合医療事務システム構成図

3) ビル消費電力監視システム

組み込みシステム分野の一例として，ビル消費電力監視システムを採り上げる．これは Echonet 機器（家庭用エアコンなど，Echonet 規格に対応した家電機器．）の消費電力データを収集し，ビルセンターで空調，照明，火気検出などを制御する総合制御システムに BACnet プロトコル(Building Automation and Control Networking protocol の略)で送受信する Gateway システムである（図 5.3 の点線枠内）．また，エコキュート(ヒートポンプ技術を利用し空気の熱で湯を沸かす電気給湯器)とも接続する．開発規模は 8 千行，開発期間は 5 ヶ月であった．別の会社が開発した装置コントローラと Echonet 接続が必要であった．

以下，開発上の 3 つの問題と解決策を述べる．

- 問題 1 ：装置コントローラのインタフェース仕様が開示されないため，Gateway システムへの要求が実現可能かどうか判断できない．

  解決策 : 開発の初期段階で，開発環境の早期準備として装置コントローラの購入を元請けに要請していた(T-1). その上で，装置コントローラの開発会社へ協力要請し(S-4), 元請けとの合同合宿によりインタフェース仕様を入手し製品実現性を迅速に確認していた(S-2).

- 問題 2 : エコキュートは Echonet 機器ではなく，JEM-A 仕様（日本電機工業会規格で定められた家電仕様）であり，この担当会社が決まっていないため，設計開始が全体工程の後半になる．予測不可能なインタフェース仕様が発生する可能性がある．

  解決策 : エコキュート制御処理部のみ開発スケジュールを開発後半にシフトし，問題が発生しても全体に影響がないような独立したソフトウェア構造にしていた(T-3, S-1)．

- 問題 3 : 開発者に装置接続の開発経験がないため，インタフェース仕様の要求を的確に抽出できない．

  解決策 : 経験を積んだ知見者による技術支援および元請レビューを計画的に行い，知識を補っていた(S-2)．更に，維持開発要員として計画的に育成していた．

- 結果： この問題 1，問題 2 に対し計画的に解決策が実行され，毎日の進捗会議で監視されていた．開発プロジェクトは品質低下や納期遅れすることなく，計画どおりに終わることができた．

- 考察 : 初物の設備機器接続には問題が発生する可能性があるため，開発リーダが早めのインタフェース仕様の開示要求をし，元請との合宿による要求抽出を実施していた（S-2）のはリスクの極小化として合理性がある．

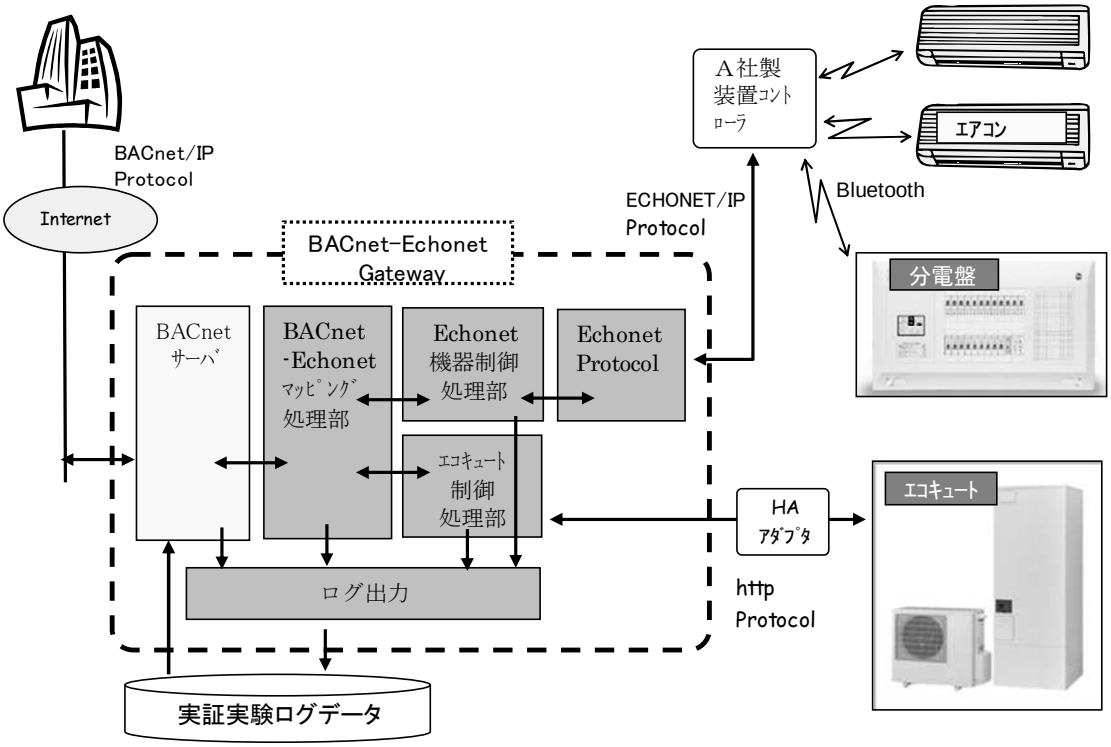


図 5.3　ビル消費電力監視システム構成図

また，エコキュート機器の担当会社が決まらず開発スケジュールを開発後半にシフトし計画的に実行したのは，リスクはあるが止むを得ない対応である（T-3）．

以上の 3 つの実例プロジェクトと第 2 章で述べたレストラン注文管理システムを含めて，4 つの実例プロジェクトへの PM パターンの適用を表 5.1 にマッピングした．左の垂直軸は，PM パターンの番号とパターン名を示す．また，水平軸は検証した 4 つのプロジェクト名，および，システム分野を示す． 表 5.1 の枠内には，その解決策をマッピングしており，空白部分は“適用なし”を指している．表 5.1 の 4 つの実例から，開発対象とするシステム或いは製品への要求は開発スケジュールの初期に全て決まるのではなく，開発期間を通じて継続的に要求抽出が実施され，7 つの PM パターンが適用されていることを確認できる．これは良いシステムを作るためには開発途中でも積極的に要求変更を取り入れ，しかも，当初の契約であるプロジェクトのスケジュールを守るために，様々なプロジェクトマネジメント技術を実施しているからである．4 つの実例プロジェクトの要求抽出に共通しているのは，U_type の要求抽出もあるが，どの時期に必要な要求を抽出するかをプロジェクト計画の段階で計画していたことである．

表 5.1　PM パターンの適用検証（例）

プロジェクト名 / 分野 / PMパターン	レストラン注文管理システム	自動倉庫システム	統合医療事務システム	ビル消費電力監視システム
	ビジネスシステム分野	エンジニアリングシステム分野	ビジネスシステム分野	組み込みシステム分野
T-1 早い時期の要求抽出	・事前調査により，再利用ｺﾝﾎﾟｰﾈﾝﾄ，アプリに影響しない機能の要求抽出.	・既存の様々な設備インタフェース仕様の抽出.	・事前ヒヤリング，運用調査により，既存システムの流用部分など，要求抽出.	・元請けとの合宿による集中的に要求抽出.
T-2 段階的な要求抽出	・多様な機能やﾕｰｻﾞｲﾝﾀﾌｪｰｽなど変化し易い機能は設計段階から段階的に要求抽出.	・運用イメージを整理しながら，段階的に顧客要求を抽出.	・標準パケージを活用した運用仕様を顧客に確認しながら，段階的に要求抽出.	
T-3 開発途中の突発的な要求抽出	・他社システムとの結合時，予測不可能なi/f仕様の要求抽出.	・結合試験時，設備が仕様どおり出来ていなく新たな要求抽出.	・他の医療システムメーカとの契約が遅れ，開発後半に予測不可能なI/F仕様の抽出.	設備機器の担当会社が未決であり，開発後半での要求抽出に変更.
S-1 開発者自身からの要求抽出	・再利用ｺﾝﾎﾟｰﾈﾝﾄ，アプリに影響しない機能の要求抽出.	・標準のソフトウェアパッケージ. ・既存の様々な設備インタフェース仕様を抽出.	・既存システムの流用部分を要求抽出.	・全体に影響を与えないソフトウェア構造の要求抽出.
S-2 ドメイン知識の熟知者からの要求抽出	・ドメイン知識を持つ元請から要求を抽出し，知識を補う.	・既存運用を顧客に確認しながら要求抽出し，顧客と一緒に設計を進める.	・設計時に，顧客および医療パッケージ・メーカから知識を補う.	・元請との合宿による装置コントローラのI/F仕様による実現性の確認.
S-3 利害対立する共同開発者からの要求抽出	・業者間でタイムリに問題や進捗状況を共有し，意思疎通を図る.（協力要請の根回し）		・顧客および医療パッケージメーカと一緒に設計し，知識を補う（分科会の設置）	
S-4 プロジェクト外からの要求抽出	・他社システムのI/F仕様の開示を元請経由で依頼し，要求抽出.	・初物の設備機器側（他社）にI/F仕様の開示依頼，要求抽出.	・元請の権限を活かし複数メーカへの強い協力要請（ネゴシエーション）により要求抽出.	・装置コントローラの開発会社への協力要請，元請けとの合宿.

つまり，第 5.2.3 項で提案した PM パターンを使い要求抽出計画がされ，開発途中での要求抽出へのプロジェクトマネジメントが実施されるのは，経験のあるプロジェクトマネジャであれば必然的なことである.

5.4. 高次ＰＭパターン

本節では，第 5.2 節で提案した要求抽出時期 PM パターン群と要求抽出元 PM パターン群との PM パターンを組み合わせて生成される高次の PM パターンについて述べる．

第 5.3 節の PM パターンの適用検証の整理結果（表 5.1）から，要求抽出時期 PM パターン群と要求抽出元 PM パターン群の具体的な解決策には類似したものがあり，要求抽出時期と要求抽出元の PM パターンを組み合わせて新たな PM パターンができることが分かる．例えば，表 5.1 の統合医療事務システムにおいて，T-1 と S-1 では，“既存システムの流用部分を要求抽出する”という解決策が共通であり，要求抽出時期の“早い時期に”と要求抽出元の“開発者自身から”を組み合わせると，“開発の早い時期に開発者自身で既存システムの流用部分を要求抽出する”解決策ができる．このように，要求抽出時期 PM パターン群と要求抽出元 PM パターン群との PM パターンの組み合わせは表 5.2 に示す２次元のマトリクスに展開できる．

表 5.2　PM パターンの組合せによる２次元マトリクス表

要求抽出元 ＼ 要求抽出時期	**開発者自身から要求抽出** [S-1]	**ﾄﾞﾒｲﾝ知識の熟知者からの要求抽出** [S-2]	**利害対立する共同開発者からの要求抽出** [S-3]	**プロジェクト外からの要求抽出** [S-4]
早い時期の要求抽出 [T-1]	・流用可能なソフトウェア部品，アプリケーションに影響しない構成要素や品質特性（保守性，移植性） ・既存の機器・設備接続（接続実績有）	・開発者にドメイン知識がなく，既存システムの仕様把握.	・共同開発者間の対立を解消し，i/f仕様の開示.（早期に契約締結） ・新規の機器・設備接続（早期の契約締結、i/f開示）	・複数の他システム連携のi/f仕様開示.（営業戦略上の競合がなく，根回しによる早期の業務提携）
段階的な要求抽出 [T-2]	・急がなくてよい機能は必要となる時期までに開発者主導で計画的に要求抽出. ・標準ﾊﾟｯｹｰｼﾞを活用した多様な運用要求の抽出.	・開発者にドメイン知識がなく，顧客および市場ニーズの変化に対応した多様な機能.（市場競争の影響による要求変更を受け易いﾕｰｻﾞi/f. ）	・コミュニケーションが取れる状態を作り，段階的に要求抽出.（新規の機器，設備が同時並行開発）	・複数の他システム連携があるが，全ての契約が一気に進まず，段階的にi/f仕様開示.
開発途中の突発的な要求抽出 [T-3]	・流用ソフトウェア部品の品質不良（完全に流用不可） ・既存の機器・設備接続（非通知のi/f変更）	・出来上がってからの新たな要求抽出（要求変更を受け易い構成要素） ・熟知者も予測不可能な要求抽出.	・遅い時期でのi/f仕様の抽出.（契約遅れ） ・新規の機器・設備接続（i/f仕様通りに未完成）	・契約遅れによるi/f仕様開示遅れ.（営業戦略上の競合） ・i/f仕様にない機能仕様を抽出（ﾒｰｶ毎にi/f仕様が異なる）

表 5.2 において，縦軸は要求抽出時期の PM パターン群，横軸は要求抽出元の PM パターン群で構成され，12 個（縦 3 個×横 4 個）のセルがある．この 2 次元マトリクス上で交差した部分の各セルが新たな PM パターンとなり，その内容が具体的に実行される解決策，即ち，アクティビティである．表 5.2 の各セルの記述は具体的な解決策を示している．

本論文では，高次 PM パターンの識別として下記の PM パターン番号の表現を定義する．

[T-m, S-n] ： mは要求抽出時期 PM パターン群の数（1 から 3 まで）
nは要求抽出元 PM パターン群の数（1 から 4 まで）

高次 PM パターンのパターン番号とパターン名の記述は以下のようになる．

[T-1,S-1] 早期に開発者自身で要求抽出を行う．
[T-1,S-2] 早期にドメイン知識の熟知者から要求抽出を行う．
[T-1,S-3] 早期に利害対立する共同開発者から要求抽出を行う．
[T-1,S-4] 早期にプロジェクト外から要求抽出を行う．
[T-2,S-1] 開発者自身で段階的に要求抽出を行う．
[T-2,S-2] ドメイン知識の熟知者から段階的に要求抽出を行う．
[T-2,S-3] 利害対立する共同開発者から段階的に要求抽出を行う．
[T-2,S-4] プロジェクト外から段階的に要求抽出を行う．
[T-3,S-1] 開発者自身で開発途中に突発的な要求抽出を行う．
[T-3,S-2] ドメイン知識の熟知者から開発途中に突発的な要求抽出を行う．
[T-3,S-3] 利害対立する共同開発者から開発途中に突発的な要求抽出を行う．
[T-3,S-4] プロジェクト外から開発途中に突発的な要求抽出を行う．

例えば，表 5.2 の[T-2,S-2] の高次 PM パターンを PM パターン項目定義に従って記述すると以下のようになる．

[T-2,S-2] ドメイン知識の熟知者から段階的に要求抽出を行う．

–背景：顧客および市場ニーズの変化に対応した多様な機能を持つシステムを開発する．ソフトウェアの構成要素の特性によっては早期に要求が決まらず抽出できないものがある．開発者には開発の業務経験や知識はあるがシステムのドメイン知識はない．ドメイン知識を有し既存システムを熟知しているのは顧客のみである．

– 問題 : 要求が曖昧で且つ不適切な時期の要求抽出は却って品質低下およびスケジュール遅れのリスクがある．また，システムの仕様およびドメイン知識がなければシステム機能構築や改善は困難であり，品質および納期を約束できない．
– フォース : 全ての要求が出揃うまで，開発着手を遅らせるわけにはいかない．開発のリスクが大きい場合，一気に開発することを回避しなければならない．また，システム要求の品質を確保するためにはドメイン知識の熟知者の協力が不可欠である．
– 解決策 : ドメイン知識を持つ熟知者を開発に巻き込み，熟知者とタイムリーにコミュニケーションが取れる状態を作り，適切な時期に段階的に要求を抽出する．
– 問題解決後の状況 : 顧客を適切にコントロールし要求抽出しなければ，品質低下およびスケジュール遅れのリスクが増大する．

ここでは，PM パターン群の組合せによって生成される高次 PM パターンの考え方に主眼をおいているため, 表 5.2 の 12 個のセル全ての高次 PM パターンの詳細を記述はしない．表 5.2 では，縦 3 個×横 4 個の 12 個のセルとしたが，横軸の要求抽出元のセルが 1 つ増えれば，2 次元のマトリクスのセルは 3 つ増える．例えば，市場ニーズから要求抽出，法律から要求抽出などの要求抽出元のＰMパターン群が 2 つ増えれば，6 つのセルが増えることになる．ただし，すべてのセルが高次ＰMパターンとして埋まるわけではない．組み合わせにおいて起こり得ない場合もある．PM パターンの組み合わせによる高次 PM パターンの整理はまだまだ研究の余地があり，今後の研究課題と捉えている．

5.5. 考察

第 5.2 節で述べた PM パターンに類似したプロジェクトマネジメント技術は，第 5.3 節で検証した以外の多くのプロジェクトでも使われていた．その結果，幾つかのプロジェクトは成功裏に完了しており，本論文で提案している PM パターンは普遍的なものであり，かつ有効であるといえる．

しかし，いくつかのプロジェクトは失敗していた．失敗の主な要因の一つは，予測不可能な品質要求が U_type プロセスのような遅い段階で数多く発生していた．例えば，ある都市の小学校の全児童を犯罪や危険から守るための防災ブザーや GPS(Global Positioning System)機能や PHS(Personal Handy-phone System)機能を備えた携帯端末の開発プ

ロジェクトである．この開発プロジェクトはハードウェアとソフトウェアを同時に開発する組み込みソフトウェア開発プロジェクトであり，約 6 ヶ月遅れの 16 ヶ月の開発期間を要した．筐体とハードウェアとの振動の問題，筐体の水漏れの問題，電波状況による通信不能の問題，内臓電池の消費による使用時間の問題など，様々な問題が発生した．ハードウェア内の問題をソフトウェアによって解決すると決定された時に，予想外のソフトウェア要求が数多く発生し設計変更を幾度も行っていた．これらの問題は，正常な処理の時には発生せず，異常処理または特殊なケースの時に発生している．その理由は，ハードウェア仕様には異常系の細かな仕様が記述されていないため，接続して初めて抽出できる要求であるからである．これがプロジェクトのスケジュールを大きく遅らせていた．このことは本論文で提案した要求抽出によるスケジュール遅れを防止するための PM パターンを適用する上で不十分なところである．U_type プロセスに対する PM パターンは問題が発生した時のことを想定し，対処方針，対応期間や費用を最低限準備しておくなどのリスクマネジメントを計画段階から慎重に実施する必要がある．そのためには，開発対象の製品が使用される環境（地域，通信など），使用者の年齢やレベル，防災上の法律，筐体の材質・強度・重量など，様々な観点からの要求分析を開発の初期で実施する必要がある．経験がなければ予測できない要求が,製品が出来上がって使ってみて初めて抽出されるのである．このような予測不可能な機能は非正常系要求と呼ばれ，分析手法がいろいろと研究されている[42][43]．従って，L_type や U_type の要求抽出に対するプロジェクトマネジメントの方法や技術を明らかにすることは開発現場の経験の浅いプロジェクトマネジャやリーダにとってはとても効果があると期待する．

5.6. まとめ

本章では，要求抽出に起因するスケジュール遅延を防ぐための典型的な 7 つの PM パターンを提案し，3 つの異なる分野（エンジニアリングシステム分野，ビジネスシステム分野，組み込みシステム分野）の実際のプロジェクトにおいて 7 つの典型的な PM パターンが有用であり効果的であることを検証した．また，本研究で抽出した PM パターンは大きく２つに分類でき，１つ目は要求抽出の時期をコントロールする PM パターン群であり，２つ目は要求抽出元をコントロールする PM パターン群である．これらの PM パターンの組み合わせを 2 次元のマトリクス上で示し，その交差した各セルが新たな PM パターンとなり，12 個の PM パターン（高次 PM パターン）を整理できることを明らかにした．

しかし，本章で提案した PM パターンは有用ではあるが，プロジェクトマネジメント経験の浅い技術者には実際のプロジェクトにその PM パターンを自分で適用することが難しい．経験の浅い技術者は，自分が担当するプロジェクトの全体像を理解し，プロジェクトの制約や問題を整理し，問題解決に漕ぎ着けるようになるまで，経験と知識の蓄積が必要となる．したがって，経験の浅い技術者でも PM パターンを適用できる仕組みを定義し，次章で述べる．

6. ＰＭパターン適用のためのフレームワーク

第 5 章では，開発途中での要求抽出をコントロールするため 7 つの典型的な PM パターンを抽出した．更に，その PM パターンが有用であることを幾つかの実際のプロジェクトにおいて検証した．また，7 つの典型的な PM パターンの組み合わせによる高次 PM パターンを定義した．

本章では，新たな PM パターンの抽出や PM パターンをプロジェクトに適用するための検証手段としてフレームワークを提案する．本研究において開発したフレームワークは二次元のマトリクスからなり，最初の次元は PMBOK[27]の 9 つの知識エリアで構成し，第 2 番目の次元は，計画，実行，監視コントロールのようにプロジェクトマネジメント・プロセス群で構成している．また，プロジェクトマネジャが，開発期間中の適切な時期に，そのフレームワークをどのように活用し，プロジェクトの問題を解決するための PM パターンを選択すればよいかについて述べる．

6.1. フレームワークの定義と目的

本論文で提案するフレームワークの定義は，経験の浅い人でもフレームワークを活用することによって同等のアウトプットが出せるアプリケーションフレームワークのように，PM パターンの適用方法などの枠組みと業務プロセスを手順化したものとする．

フレームワークの目的は，プロジェクトマネジャが対象となるプロジェクトの問題を適切に解決するために，たとえ経験の浅いプロジェクトマネジャでも，悩むことなく，効率的に PM パターンを選定し，問題解決に取り組むことができる枠組みと業務プロセスの手順を提供することである．問題解決のアプローチ方法は新 QC 七つ道具[44]や TOC[45]など様々あるが，この PM パターンのフレームワークを活用すれば，現場のプロジェクトマネジャやリーダが自分の経験に基づき我流で考えた場合に比べて，重要な検討事項を見過ごさずに比較的簡単に PM パターンを選択でき，問題を解決できることを目指している．また，新たな PM パターンを抽出した場合，このフレームワークを活用して，プロジェクトへの影響範囲をシミュレーションすることができることも目的のひとつである．

6.2. フレームワークの構成

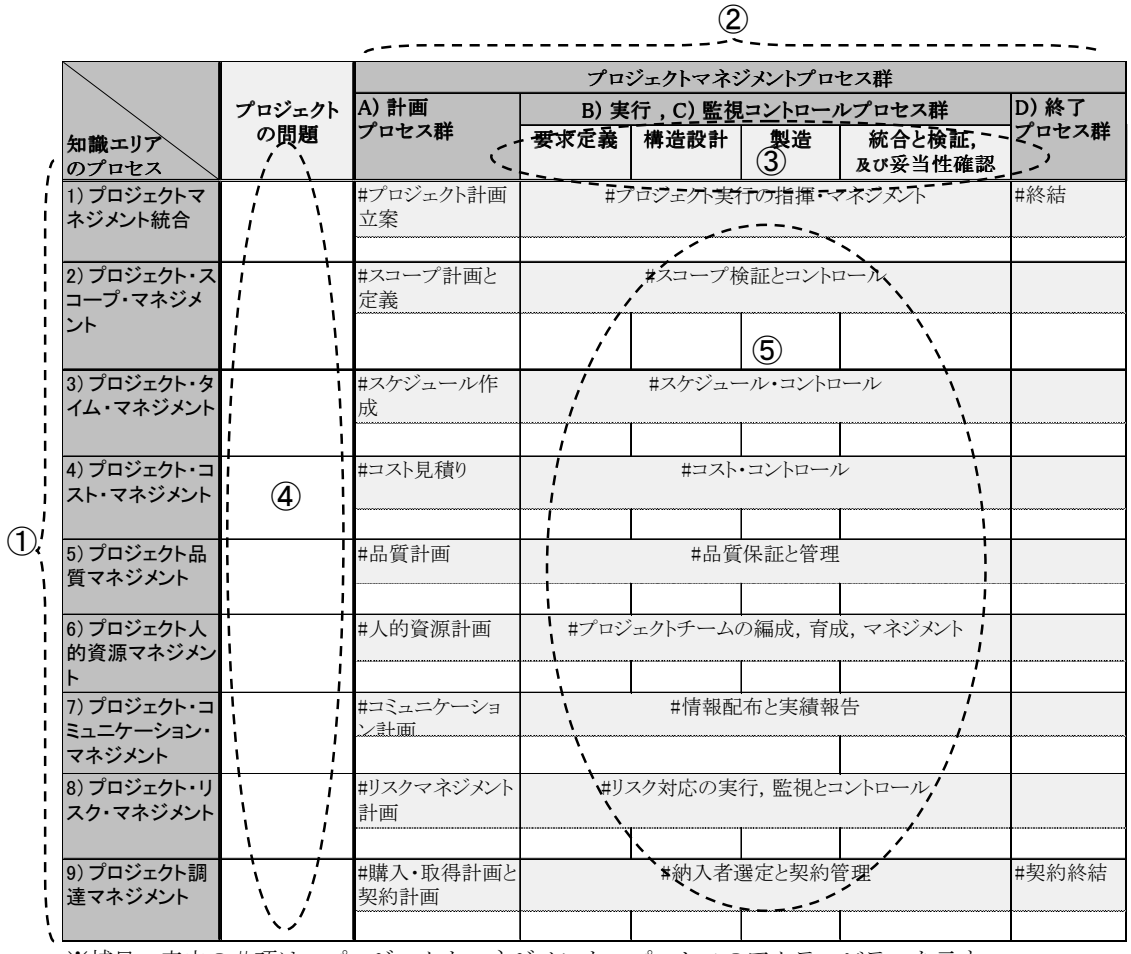
本節では，PM パターンのフレームワークの構成として，二次元のマトリックス構造からなる表を提案する（表 6.1）.

1 次元目の縦軸はプロジェクトマネジメント知識の 9 つのエリアである．最左側欄にその知識エリアを指定する（表 6.1 の①）．2 次元目の横軸はプロジェクトマネジメント・プロセスの 5 つのプロセス群である．最上位列の右部分に 5 つのプロセス群を指定する（表 6.1 の②）．フレームワーク構造の知識エリアとプロジェクトマネジメント・プロセス群の各項目は，業界で広く受入れられている“プロジェクトマネジメント知識体系（PMBOK）ガイド[27]”に準拠している．表 6.1 の各行は，

- **プロジェクトマネジメント統合**：プロジェクトマネジメント・プロセス群内で識別、定義、結合、統一、調整された、プロジェクトマネジメントのさまざまな要素を統合するプロセス.
- **プロジェクト・スコープ・マネジメント**：プロジェクトの成功に必要な作業を過不足なく含めることを確実にするために必要なプロセス.
- **プロジェクト・タイム・マネジメント**：プロジェクトを所定の時期に完了させるために必要なプロセス.
- **プロジェクト・コスト・マネジメント**：プロジェクトを承認された予算内で完了させるために必要なコストの計画、見積り、予算化、コントロールのプロセス.
- **プロジェクト品質マネジメント**：プロジェクトが所定の目標を満たすことを確実にするためのプロセス.
- **プロジェクト人的資源マネジメント**：プロジェクト・チームを組織化し、マネジメントするためのプロセス.
- **プロジェクト・コミュニケーション・マネジメント**：プロジェクト情報の生成、収集、配布、保管、廃棄をタイムリーかつ適切に行うために必要なプロセス.
- **プロジェクト・リスク・マネジメント**：プロジェクトのリスクマネジメントを行うプロセス.
- **プロジェクト調達マネジメント**：プロダクト、サービス、所産の購入または取得のプロセス、および契約のプロジェクトマネジメント・プロセス.

から構成される.

表 6.1　PM パターンのフレームワーク構造



知識エリアのプロセス	プロジェクトの問題	プロジェクトマネジメントプロセス群				
		A) 計画プロセス群	B) 実行，C) 監視コントロールプロセス群			D) 終了プロセス群
			要求定義	構造設計	製造	統合と検証，及び妥当性確認
1) プロジェクトマネジメント統合		#プロジェクト計画立案	#プロジェクト実行の指揮・マネジメント			#終結
2) プロジェクト・スコープ・マネジメント		#スコープ計画と定義	#スコープ検証とコントロール			
3) プロジェクト・タイム・マネジメント		#スケジュール作成	#スケジュール・コントロール			
4) プロジェクト・コスト・マネジメント		#コスト見積り	#コスト・コントロール			
5) プロジェクト品質マネジメント		#品質計画	#品質保証と管理			
6) プロジェクト人的資源マネジメント		#人的資源計画	#プロジェクトチームの編成，育成，マネジメント			
7) プロジェクト・コミュニケーション・マネジメント		#コミュニケーション計画	#情報配布と実績報告			
8) プロジェクト・リスク・マネジメント		#リスクマネジメント計画	#リスク対応の実行，監視とコントロール			
9) プロジェクト調達マネジメント		#購入・取得計画と契約計画	#納入者選定と契約管理			#契約終結

※補足　表中の＃項は，プロジェクトマネジメント・プロセスのアクティビティを示す．

また，表 6.1 の最上位欄の各列は，

- 計画プロセス群
- 実行，監視コントロールプロセス群
  - 要求定義工程
  - 構造設計工程
  - 製造工程
  - 統合と検証工程，および妥当性確認工程
- 終了プロセス群

から構成される．

プロジェクトマネジメント知識体系（PMBOK）によると，プロジェクトマネジメントの全てのアクティビティは，5 つのプロジェクトマネジメント・プロセスグループと 9 つの

プロジェクトマネジメント知識エリアに分けて考えることができる．ただし，表 6.1 では PMBOK に定義されているプロジェクトマネジメント・プロセスグループのうち，立上げプロセス群を省略している．これは，PM パターンが，プロジェクトマネジメントの計画，実行，監視制御プロセスに焦点を当てているからである．さらに，PM パターンのフレームワークは，実行，監視コントロールプロセス群の中に，システムエンジニアリングの要求定義，構造設計，製造，統合と検証，および妥当性確認の 5 つの開発プロセス[46]を導入している（表 6.1 の③）．これらの開発プロセスを導入することによって，プロジェクトマネジャが開発工程のどの段階でプロジェクトの問題を解決するべきかを計画し，実行するかが分かる[22]．表 6.1 の第 2 列目は PM パターンが対処するプロジェクトの問題を記述する欄である（表 6.1 の④）．したがって，これらの項目によって構成されるフレームワークを使用すれば，どの時期にプロジェクトの問題を解決するかを計画し，第 5.4 節で示した 12 個の高次 PM パターンのひとつの解決策をフレームワーク上に対応づけることが可能となる（表 6.1 の⑤）．

6.3. 実際のプロジェクトから抽出した問題と解決策

本節では，第5.3節でPM パターンの有用性を検証した実際の複数プロジェクトから共通的に使用できる要求変更に伴うプロジェクトのスケジュールの遅延を防ぐための解決すべき問題や制約，解決策を抽出しフレームワーク上に整理した（表6.2）．以下に記述する本文の［P-i］やプレームワーク上の［P-i］は，問題の番号である．

- プロジェクト上の制約や問題

  他の会社によって開発された既存システムを短い期間で再開発しなければならない．

  P-1 納期や品質を損なうリスクがある．

  P-2 不適切な時期での要求抽出は却ってスケジュール遅れとなる．

  P-3 様々な機能と複数の他システム連携がある．

  （早期に全ての要求を抽出できない．）

- 開発環境上の制約や問題

  P-4 詳細なシステム仕様のドキュメントがない．

  P-5 類似システムの開発経験がない．

  P-6 他システムインタフェース仕様の抽出のためのコネクションがない．

  P-7 ステークホルダー間のコミュニケーションが十分ではない．

表 6.2　スケジュール遅れを防ぐための要求抽出に関する問題と解決策

知識エリアのプロセス	プロジェクトの問題	プロジェクトマネジメントプロセス群					
		A) 計画プロセス群	B) 実行，C) 監視コントロールプロセス群				D) 終了プロセス群
			要求定義工程	構造設計工程	製造工程	統合と検証工程，及び妥当性確認工程	
1) プロジェクトマネジメント統合	プロジェクトの様々な問題 (P-1～P-7を統合)	#プロジェクト計画立案 要求の抽出方法，時期，コスト，コミュニケーション，要員，リスク，契約など，総合的に計画する.	#プロジェクト実行の指揮・マネジメント 計画の実行，および変化を監視し，コントロールする.				#終結
2) プロジェクト・スコープ・マネジメント	(P-3) 様々な機能と複数の他システム連携がある 早期に全ての要求を抽出できない.	#スコープ計画と定義 (1)多機能に対する手戻りﾘｽｸ軽減のため，ｲﾝｸﾘﾒﾝﾀﾙ開発を計画:[*,*] (2)他システムI/F仕様の抽出を計画:[*,S-4]	#スコープ検証とコントロール (1)事前調査，現場視察の実施による既存システムの流用機能や運用の把握:[T-1,*]	(1)設計への顧客参加による要求の抽出:[T-2,S-2] (画面操作周りのプロトタイプによる要求抽出) (2)他メーカの一般ユーザ研修へ参加，マニュアル調査によるI／F仕様の抽出:[T-2,S-4]	(1)画面など出来上がったものから顧客ﾚﾋﾞｭｰによる要求抽出:[T-2,S-2] (2)他ｼｽﾃﾑの借用による基本的なI/F仕様の動作確認:[T-2,S-4]	(1)性能，信頼性などの要求抽出:[T-3,S-2] 画面操作周りのなどの改善および新たな機能要求の抽出: (2)他システム接続による予測不可能な新たな要求抽出:[T-3,S-4]	
3) プロジェクト・タイム・マネジメント	(P-2) 不適切な時期での要求抽出は却ってスケジュール遅れる	#スケジュール作成 (1)品質特性，構成要素ごとの要求抽出の時期を立案:[*,S-2] (2)他システムI/F仕様抽出の時期設定:[*,S-4]	#スケジュール・コントロール (2)他メーカのi/f仕様開示時期の確認:[T-1,S-4]	(1)タイムリーに顧客レビューを調整:[T-2,S-2] (2)他メーカのi/f仕様開示の督促:[T-2,S-4]	(1)タイムリーに顧客レビューを調整:[T-2,S-2] (2)他メーカのi/f仕様開示の督促:[T-2,S-4]	(1)新たな要求への対応ｽｹｼﾞｭｰﾙを迅速に調整:[T-3,S-2] (2)他ｼｽﾃﾑ接続による予測不可能な要求の抽出に対して，対応ｽｹｼﾞｭｰﾙを迅速に調整:[T-3,S-4]	
4) プロジェクト・コスト・マネジメント		#コスト見積り	#コスト・コントロール				
5) プロジェクト品質マネジメント	(P-4) 詳細なシステム仕様のドキュメントがない	#品質計画 (1)要求抽出漏れを防ぐための方策を計画 ・要求仕様，設計レビューへの顧客の参加要請:[*,*]	#品質保証と管理 (1)顧客による要求仕様のレビュー実施と評価:[T-1,S-2]	(1)設計レビューに顧客参加:[T-2,S-2] (画面および操作周り仕様漏れチェック) ・ﾕｰｻﾞI／Fの置換可能な設計:[*,S-1] ・外部機器I／Fのｺﾝﾎﾟｰﾈﾝﾄ化:[*,S-1]	(1)画面など出来上がったものから顧客レビュー実施:[T-2][S-2]	(1)画面，および操作周りの動作検証での顧客参加:[T-3,S-2] (2)システム全体の妥当性確認での顧客参加:[T-3,S-2]	
6) プロジェクト人的資源マネジメント	(P-5) 類似システムの開発経験がない	#人的資源計画 (1)開発への顧客参加と協力を要請（役割の明確化）:[*,S-2]	#プロジェクトチームの編成，育成，マネジメント (1)現場視察および顧客へのヒヤリングによる知識の補充:[T-1,S-2]	(1)タイムリーな顧客レビューの実施による知識アップ:[T-2,S-2]	(1)タイムリーな顧客レビューの実施による知識アップ:[T-2,S-2]	(1)予測不可能な要求に対する技術支援の増員:[T-3,*]	
7) プロジェクト・コミュニケーション・マネジメント	(P-7) ステークホルダ間のコミュニケーションが十分でない （対立関係）	#コミュニケーション計画 (1)進捗，問題を共有するための手段の計画（定例Meeting，ｲﾝﾌﾗ）:[*,S-3]	#情報配布と実績報告 (1)定例Meeting実施，ｲﾝﾌﾗ共有による早期の問題解決:[*,S-3] ・責任者出席による問題解決のｽﾋﾟｰﾄﾞｱｯﾌﾟ			(1)緊急の技術Meeting実施による早期の問題解決:[T-3,S-3]	
8) プロジェクト・リスク・マネジメント	(P-1) 納期や品質を損なうリスクがある	#リスクマネジメント計画 (1)他システムI／Fの開示遅れを想定し，リスク対策を計画（ネゴシエーション，製造工程に対応プロセスを確保）:[*,S-4]	#リスク対応の実行，監視とコントロール (1)計画していた顧客の責任者へのネゴシエーション実施:[T-1,S-4]	(1)計画していた一般ユーザ研修へ参加し，マニュアル調査を実施:[T-2,S-4]	(2)他システムI/F仕様開示が遅れた場合，計画していた対応プロセスを実施:[T-2,S-4]	(1)他システム接続時に，予測不可能な新たな要求の抽出に対して，支援体制の増強:[T-3,S-4]	
9) プロジェクト調達マネジメント	(P-6) 他システムI/F仕様の抽出のためのコネクションがない （プロジェクト外）	#購入・取得計画と契約計画 (1)他メーカと早期の契約締結を計画:[*,S-4]	#納入者選定と契約管理 (1)他メーカとの契約締結交渉:[T-1,S-4]	(1)他メーカとの契約締結遅れへ督促:[T-2,S-4] ・強力なネゴシエーション実施	(1)他システムの借用:[T-2,S-4]		#契約終結

※用語説明　　WBS: Work Breakdown Structure　　I/F: インタフェース

これらの問題を表6.2の第2列目に示した．個々の問題の先頭に付与した番号は，前述の問題番号に対応している．表6.2の第2列目に示した個々の問題を解決するための解決策が，プロジェクトマネジメント・プロセスグループの各セルに分解され示されている．表6. 2の中の［T-m, S-n］記号は，第5.4節で定義した高次PM パターンの番号である．複数のPM パターンが一つのプロジェクトにおいて同時に適用される可能性がある．そこで，相互の問題解決策が矛盾を起こさないか否かをフレームワーク上でプロジェクトのアクティビティのシナリオによってシミュレーションする必要がある．相互の問題解決策の間に矛盾がないというのは，解決策の全てのアクティビティが他のマネジメントのアクティビティと相反することなく一緒に実行できるという意味である．PM パターン間の矛盾については後述の第6. 6節で述べる．

本論文では，実際のプロジェクトから共通的な要求抽出に伴う問題，解決策とPMパターンを関連付けて整理した（表6.2）が，今後は，ドメイン分野ごとに整理するとドメイン特有の問題，課題とPMパターンからなるフレームワークを提案できると考えている．

6.4. フレームワークの活用プロセス

本節では,プロジェクトマネジャがフレームワークを活用するためのプロセスを述べる．図 6.1 に示すように，先ず，プロジェクト目標を問題や課題に分解し，次に，その問題や課題に対する要因分析を行う．3 番目に，問題をフレームワーク上に配置し，4 番目にどの PM パターンに該当するか選択する．5 番目に選択された PM パターン間に矛盾や補完がないかをシミュレーションし適切な PM パターンであるかを確認する．最後に，プロジェクトの全体スケジュール上のどの時期に実施するかをスケジューリングする．以下に，各々の処理概要を示す.

1) プロジェクト目標の分解

プロジェクトにはいくつかの目標がある．それらの目標を達成するためには制約となる問題や課題を解決しなければならない．プロジェクトの問題と PM パターンの問題記述レベルがほぼ一致するレベルまでプロジェクト目標を分解する．

2) 問題の要因分析

分解された問題や課題に対して，それらの問題の制約になっている要因を段階的に詳細化する．

3) フレームワークへの割当て

分解した問題がフレームワークのどの知識エリアに該当するかを分類し，フレームワーク上の問題欄に割り当て，他の知識エリアへの関連性を確認する．

4) PM パターンの選択

プロジェクトの問題記述レベルとほぼ合致する PM パターンを PM パターン群の中から抽出する．PM パターン群とは様々な PM パターンを集めた知識ベースである．現時点では第 5 章で定義した PM パターンであるが，将来的には様々な分野やプロジェクトから抽出した PM パターン群となる．

5) PM パターン間の矛盾・補完

選択した複数の PM パターンを同時に実行する場合，PM パターン同士が相矛盾するアクティビティになっていないか，または，複数の PM パターンを組み合わせて（補完）プロジェクトの問題を解決する PM パターンを抽出するなどのシミュレーションを行う．

6) スケジューリング

選択した PM パターンがプロジェクトの全体スケジュール上のどの時期に実施するかを調整しスケジューリングする．

フレームワークの活用プロセスの中で重要と捉えているプロジェクト目標，および PM パターン間の矛盾・補完に関しては，第 6.5 節以降に，事例を挙げて説明する．

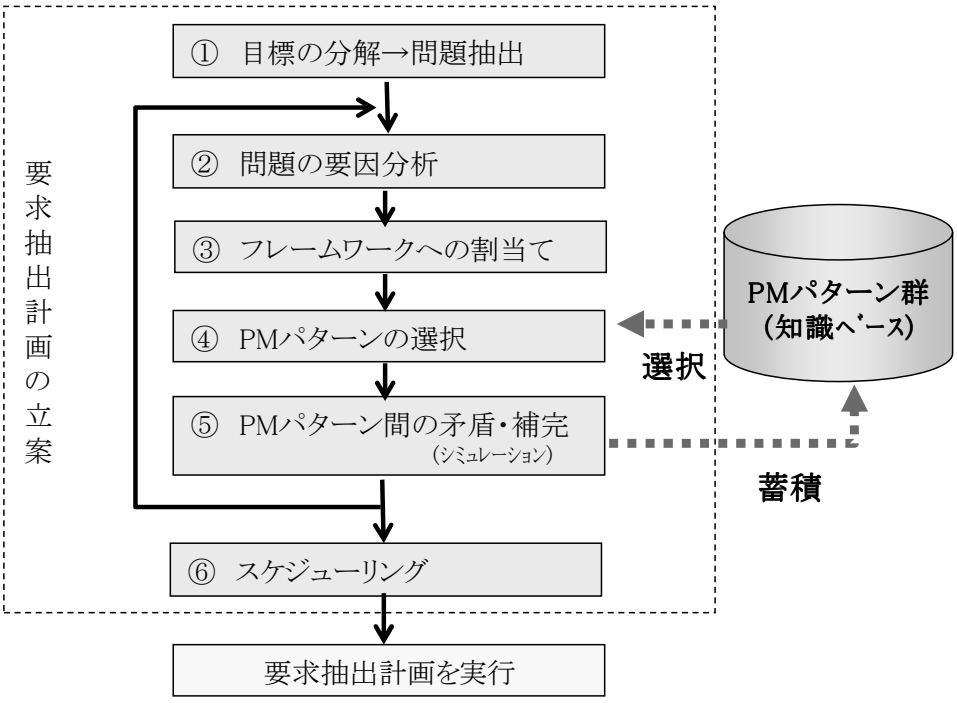
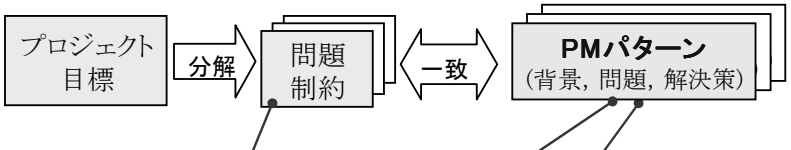


図 6.1　フレームワークの活用プロセス

6.5. プロジェクト目標の分解

本節では，フレームワークの活用プロセスにおけるプロジェクト目標の分解中を説明する．プロジェクトを成功に導くためには，目標の達成を阻害する制約や想定される様々な問題を解決して，プロジェクト目標を達成しなければならない．本論文の目的に対応づけるならば，要求変更によるプロジェクトのスケジュールの遅延を防ぐためには，解決すべき問題と制約に対する処置をPMパターンと関連づける必要がある．即ちPMパターンを選択するために，プロジェクトの問題とPMパターンに記述した問題とがほぼ一致する記述レベルまでプロジェクト目標を制約と問題に分解する必要がある．図6.2に示すように，フレームワークを適用すると，解決すべき問題と制約に対する処置は，フレームワーク上の各セルに対応する問題に分解される．

例えば，第2章で説明した実例プロジェクトでは，システムの拡張性，導入の効率化など改善要求を取り込み，システムを再開発するというプロジェクト目標がある．この目標を阻害する問題のひとつに，要求変更を受け易いユーザインタフェースがあげられる（図6. 2）.



フレームワーク

知識エリアのプロセス	プロジェクトの問題	プロジェクトマネジメントプロセス群					
		A) 計画プロセス群	B) 実行，C) 監視コントロールプロセス群				D) 終了プロセス群
			要求定義	構造設計	製造	統合と検証，及び妥当性確認	
1) プロジェクトマネジメント統合		#プロジェクト計画立案	#プロジェクト実行の指揮・マネジメント				#終結
2) プロジェクト・スコープ・マネジメント	(P-3) 要求変更を受け易いユーザI/Fがある	#スコープ計画と定義	#スコープ検証とコントロール				
		手戻りリスク軽減の為，インクリメンタル開発を計画 :[*,S-2]	事前調査よる流用機能や運用の把握 :[T-1,*]	プロトタイプなど，顧客参加による要求抽出:[T-2,S-2]	出来たものから顧客確認による要求抽出: [T-2,S-2]	統合および検証による新たな要求抽出 :[T-3,S-2]	
3) プロジェクト・タイム・マネジメント		#スケジュール作成	#スケジュール・コントロール				
中略							
9) プロジェクト調達マネジメント		#購入・取得計画と契約計画	#納入者選定と契約管理				#契約終結

図 6.2　プロジェクト目標の問題分解

6.6. ＰＭパターン間の矛盾

プロジェクトには開発開始から終了にいたるまで様々な問題や課題が発生する．それらの問題をスピーティに且つ適切に解決するために，PM パターンの適用は有効であるが，時として選択した PM パターン間同士で相矛盾することがある．その矛盾を解消するために，プロジェクトマネジャは，PM パターンをプロジェクトに適用する前に，選択したすべての PM パターンが同じプロジェクト内で同時に使用できるかどうか，矛盾がないか，プロジェクトにとって効率的な要求抽出計画を作成するための PM パターンであるかを検証しなければならない．最初にフレームワーク内のマトリクスのセルの各々で，２番目にＰM知識エリアの各々で，３番目にプロジェクト全体で PM パターンによって決定されるすべての解決策のアクティビティを段階的にシミュレートすることになる．矛盾がある場合，プロジェクトマネジャはプロジェクト目標の QCD，即ち，品質（Q），コスト（C），納期（D）への影響を考慮した上で，顧客の満足を損なわないように対応の優先順位づけや対応への重み付けなどのバランスをとる必要がある．

例えば，第 2 章で採り上げた実例プロジェクトでは，プロジェクト目標の一つに，「競合他社システムと連携を実現する」がある．表 6.3 に示すように，フレームワーク上に，この目標を阻害する問題，課題に分解すると，「競合関係であるため I/F 仕様の開示が遅れる可能性がある」というリスクと「競合関係であるが開発は進めなければならない」という問題とに分解される．それぞれに対応策を検討すると，「遅い時期での要求抽出への対応プロセスの準備」というリスク軽減の策（[T-3,S-4]）と「早期の業務提携による I/F 仕様開示」とう解決策（[T-1,S-4]）がある．つまり，計画時点では，早い時期に要求抽出するという策と遅い時期に要求抽出するという策が相矛盾することになる．しかし，要求抽出によるスケジュール遅れを防止するためには，片方の策だけでなく両方の策を取り入れるか，段階的に契約を結びつつ要求抽出するような策にするかなど，リスクを考慮した解決策を選択し実施することになる．表 6.3 を分かりやすく示したのが図 6.3 である．

表 6.3　PM パターン間の矛盾例

知識エリアのプロセス	プロジェクトの問題	プロジェクトマネジメントプロセス群					
		A) 計画プロセス群	B) 実行，C) 監視コントロールプロセス群				D) 終了プロセス群
			要求定義	構造設計	製造	統合と検証，及び妥当性確認	
1) プロジェクトマネジメント統合		#プロジェクト計画立案	#プロジェクト実行の指揮・マネジメント				#終結
2) プロジェクト・スコープ・マネジメント		#スコープ計画と定義	#スコープ検証とコントロール				
3) プロジェクト・タイム・マネジメント		#スケジュール作成	#スケジュール・コントロール				
	中略						
8) プロジェクト・リスク・マネジメント	I/F仕様の開示が遅れる可能性がある	#リスクマネジメント計画	#リスク対応の実行，監視とコントロール				
		遅い時期にI/F仕様の要求抽出を計画．:[＊,S-4]			製造工程で要求抽出への対応プロセスの準備:[T-2,S-4]	検証工程で要求抽出への対応プロセスの準備:[T-3,S-4]	
9) プロジェクト調達マネジメント	競合関係にあるが，開発は進めなければならない	#購入・取得計画と契約計画	要求抽出時期が相矛盾！		者選定と契約管理		#契約終結
		早期の業務提携を計画．:[T-1,S-4]	早期の業務提携によるI/F仕様開示．:[T-1,S-4]				

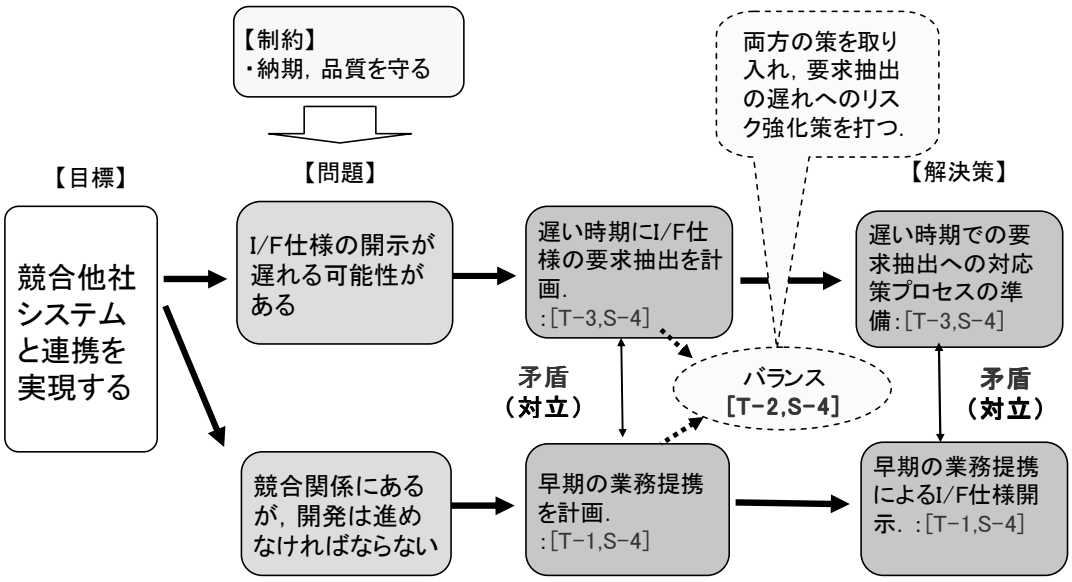


図 6.3　PM パターン間の矛盾図

また，別の例として，開発期間の遅い時期（結合テスト工程やシステムテスト工程など）に多くの要求変更を抽出した場合を採り上げる．顧客の満足を満たすためにはこれらの新たな要求を受け入れなければならない．しかし，この時期での要求変更は，開発作業の手戻りに伴う対応工数や費用を増大させるだけでなく，品質の低下やスケジュール遅れに繋がりかねない．顧客との契約上，納期や品質は守らなければならないという制約がある．ここに，多少品質は落としてでも納期を厳守するか，品質を確保するが多少納期を遅らせるか，あるいは，要求変更を多少受け入れず納期を厳守するか，要求変更を受け入れて納期を遅らせるなど，フレームワークの格セル内の解決策の同士の対立の関係（つまり，矛盾）が生じる（図 6.4）．これらの問題は，契約上の納期厳守，品質確保が制約になっている．これらの問題を解決するために，プロジェクトマネジャは，顧客の満足や運用への影響をベースに，プロジェクトへの影響を最小化することを考慮し，どれに重点をおき，優先順位を決め，対処するかなど，フレームワークを活用して，シミュレーションすることができる．

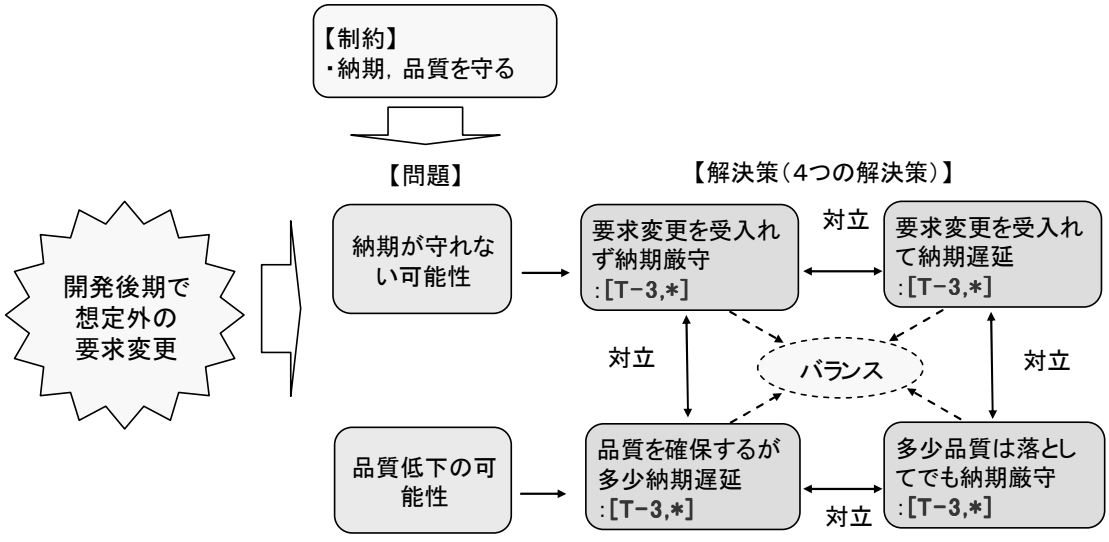


図 6.4　フレームワークの格セル内の解決策間の矛盾

6.7. ＰＭパターン間の補完

ある目標を達成するために，1 つの PM パターンだけでなく，2 つ以上の PM パターンを使用することによって問題を解決し，目標を達成できる場合がある．これを PM パターンの補完の関係と定義する．

例えば，第 2 章で採り上げた実例プロジェクトでは，プロジェクト目標の一つに，「多様なユーザ I/F を持つシステムを開発する」がある．表 6.4 で示すように，フレームワーク上に，この目標を阻害する問題，課題に分解すると，「要求変更を受け易いユーザ I/F がある」という課題と「開発者にドメイン知識がない」という問題とに分解される．それぞれに対応策を検討すると，「手戻りを避けるため，プロトタイプによる要求抽出」という手戻りリスクを軽減する解決策（[T-2,*]）と「熟知者からドメイン知識を補いながら開発する」とう解決策（[*,S-2]）がある．開発者にドメイン知識がなければ顧客が満足するユーザフレンドリーなシステムを開発できない．そのため，システムの熟知者から知識を補いながら,プロトタイプによる要求抽出する組み合わせで問題を解決し,目標を達成できる．第 5.4 節で定義した高次 PM パターンの［T-2,S-2］,［T-2,S-3］と［T-1,S-2］,［T-2,S-2］との補完となる．つまり，一つの PM パターンに関連して，他の PM パターンが相互に補いながら問題を解決している（表 6.4）．表 6.4 を分かりやすく示したのが図 6.5 である．

表 6.4　PM パターン間の補完例

知識エリアのプロセス	プロジェクトの問題	プロジェクトマネジメントプロセス群					
		A) 計画プロセス群	B) 実行 , C) 監視コントロールプロセス群				D) 終了プロセス群
			要求定義	構造設計	製造	統合と検証，及び妥当性確認	
1) プロジェクトマネジメント統合		#プロジェクト計画立案	#プロジェクト実行の指揮・マネジメント				#終結
2) プロジェクト・スコープ・マネジメント	**要求変更を受け易いユーザI/Fがある**	#スコープ計画と定義	#スコープ検証とコントロール				
		手戻りリスク軽減の為，インクリメンタル開発を計画 :[T-2,*]		**手戻りを避けるため，プロトタイプによる要求抽出 :[T-2,*]**		**統合および検証による新たな要求抽出 :[T-3,S-2]**	
		中略		補完関係			
6) プロジェクト人的資源マネジメント	**開発者にドメイン知識がない**	#人的資源計画	#プロジェクト実行の指揮・マネジメント				
		ドメイン知識の熟知者から要求抽出計画 :[* S-2]		**熟知者から知識を補い，要求抽出 :[*,S-2]**			
8) プロジェクト・リスク・マネジメント		#リスクマネジメント計画	#リスク対応の実行，監視とコントロール				
9) プロジェクト調達マネジメント		#購入・取得計画と契約計画	#納入者選定と契約管理				#契約終結

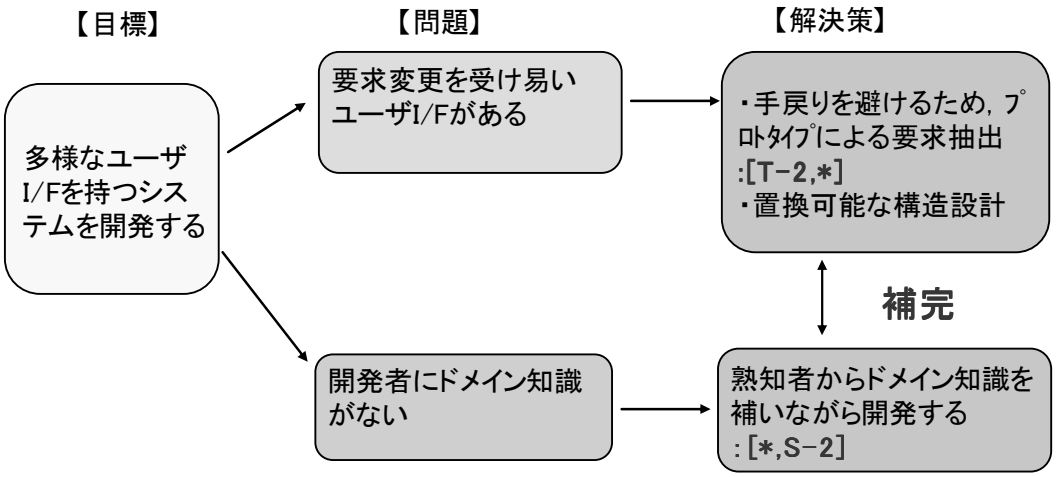


図 6.5　PM パターン間の補完図

何故，このように補完の関係が発生するのかを以下に考察する．パターンでは，原則的には問題に対する解決策は１つである．補完の要因は，ひとつのプロジェクトの問題に対して要因が複数あり，それらの一つ一つを解決する PM パターンが必要となるからである．第 6.5 節で述べたように，問題の分解レベルが不充分な場合，選択した PM パターンだけでは問題が解決せず，別の PM パターンと組み合わせて使用することによって初めて問題を解決できることになる．即ち，問題を解決するためには一つの解決策ではなく，２つ以上の策が必要なのである．つまり，問題（現象）を引き起こしている要因が２つ以上あるということである．例えば，スケジュールが遅れているという問題が発生した時，開発メンバーのスキル不足からくる遅れ，仕様が曖昧なために遅れるなど，２つの要因が重なっている場合がある．このような場合，両方に対策を打たなければ問題は解決しない．逆に，問題を２つに分ける必要がある．しかし，表面上の現象は同じであれば２つに分けることは難しい．その場合，パターンの背景を詳細に記述することになる．

6.8. 考察

PM パターンをプロジェクトに適用するために選択した後，選択したすべての PM パターンが同じプロジェクト内で同時に使用できるかどうか，矛盾がないか，プロジェクトにとって効率的な要求抽出計画を作成するための PM パターンであるかを証明しなければならない．最初にフレームワーク内のマトリクスのセルの各々で，２番目に PM 知識エリアの各々で，３番目にプロジェクト全体で PM パターンによって決定されるすべての解決策のアクティビティを段階的にシミュレートすることによって矛盾がないことを証明できる．例えば，大規模のシステムを開発するプロジェクトではシステムの構成要素を開発するサブプロジェクト毎に分けてシミュレートする．その場合，サブプロジェクトの各々内で，次にプロジェクト全体で PM パターンを段階的にシミュレートすることによって矛盾がないことを証明する必要がある．矛盾が発生した場合，プロジェクトマネジャは，プロジェクト目標の QCD，即ち，品質（Q），コスト（C），納期（D）への影響を考慮した上で，顧客の満足を損なわないように対応の優先順位づけや対応への重み付けなどのバランスを調整する必要がある．

第 5 章でも述べたが，経験の浅い技術者が，自分が担当するプロジェクトの全体像を理解し，プロジェクトの制約や問題を整理し，問題解決に漕ぎ着けるようになるまで経験と知識が必要となる．それを補うために開発した PM パターンのフレームワークは，選択された複数の PM パターンを同時に実行する場合，PM パターン間の矛盾がないかのシミュレーションに効果を発揮することができる．また，このフレームワークを活用することによって，プロジェクトマネジャが，要求を抽出する時期を計画し，要求の抽出を監視し，制御するための PM パターンを適切に選択できるようになり，プロジェクトの問題を適切に解決できるようになると期待する．しかし，プレームワークを活用するに当たって，業界で広く受入れられている“プロジェクトマネジメント知識体系（PMBOK）ガイド[27]”の 9 つの知識エリアとプロジェクトマネジメント・プロセス群は最低限理解しておく必要がある．これは PM パターンとフレームワークの導入教育によって説明し，理解させる必要がある．

また，本論文で提案したフレームワークの適用研究を進め，経験の浅いプロジェクトマネジャでも，プロジェクトの問題をスピーディかつ適切に解決するために，適用すべき PM パターンの妥当性が判断できるフレームワーク構造に進化させる必要があり，今後の研究課題と捉えている．

6.9. まとめ

本章では，プロジェクトマネジャがプロジェクト目標を達成するために，プロジェクトの目標を制約や問題に分解し，その解決策としてPMパターンをプロジェクトに適用する手段であるフレームワークを開発し，その活用の仕方について提案した．特に，複数の問題に対するそれぞれのPMパターンが同時に使用できるかどうか， PMパターン間で実行の矛盾がないかをシミュレーションするためにフレームワークは有効であることを示した．また，経験の浅いプロジェクトマネジャや技術者にも比較的簡単にフレームワークを活用できることを提案したが，実証は今後の課題である．フレームワークを活用する前提として“プロジェクトマネジメント知識体系（PMBOK）ガイド[27]”の9つの知識エリアとプロジェクトマネジメント・プロセス群は最低限理解しておく必要がある．これはフレームワークの運用上のトレーニング課題と捉え，本論文ではトレーニング内容やレベルまでは記述しない．むしろ，経験の浅いプロジェクトマネジャや技術者でも比較的簡単にフレームワークを活用できるようなフレームワーク構造を検討すべきであると考えている．

また，本論文では，実際のプロジェクトから共通的な要求抽出に伴う問題，解決策とPMパターンを関連付けて整理したが，今後は，ドメイン分野ごとに整理するとドメイン特有の問題，課題とPMパターンからなる知識ベースのフレームワークを提案し，更に実用的なものになると期待する．

7. 全体まとめと今後の課題

以下に，本論文全体のまとめと今後の課題を述べる．

7.1. 全体まとめ

本論文では，統合型要求工学による要求プロセスの研究の一環として，開発途中で多くの要求変更を受け入れたにも関わらず，スケジュール遅延することなく，顧客満足度も落とさず，比較的うまくいった実際のプロジェクト（レストラン注文管理システム）の開発履歴から要求抽出プロセスおよびプロジェクトマネジメント技術の実態調査，分析を行った．その結果，要求抽出プロセスには３つの成熟度タイプ，即ち，早期成熟度タイプ，段階的成熟度タイプ，突発的な成熟度タイプがあることを明らかにした．

また，ソフトウェア開発のプロジェクトにおいて，開発途中での要求変更に伴うプロジェクトのスケジュール遅延を防ぐためには，開発現場のプロジェクトマネジャがこれら 3 つの成熟度タイプの要求抽出プロセスを適切にかつ計画的に実行しマネジメントすることが重要であることも明らかにした．そのプロジェクトマネジメント技術を再利用できるようにプロジェクトマネジメント・パターン（以降，Project Management パターン：PM パターンと呼ぶ）として整理し提案した．この PM パターンは，実際のプロジェクトの開発履歴を調査分析し，要求抽出のプロセス，要求の内容と抽出時期や様々な知見を整理し，Alexander のパターンの概念を取り入れ，整理したものである．また，整理した PM パターンが他のプロジェクトにおいても普遍的に使用されるべきものかどうかを確認するために，エンジニアリング系，ビジネス系，組み込み系の 3 つの分野の実際のプロジェクトをピックアップし，それぞれのプロジェクトの問題と解決策がどの PM パターンと一致するかを分析した結果，広く有用であることを確認することができた．本論文では，整理した 7 つの PM パターンが要求抽出をコントロールする典型的なパターンであり，これらを組み合わせて高次 PM パターンも生成できることも明らかにした．

さらに，これらの PM パターンを効果的に活用するための仕組みとして，PMBOK の 9 つの知識エリアと 5 つのプロジェクトマネジメント・プロセスを取り入れ，プロジェクトの問題と解決策を整理する二次元のフレームワークを開発し，提案することができた．

7.2. 考察

ソフトウェア開発において開発途中の要求変更はスケジュール遅延の主な要因のひとつに挙げられている．要求工学の分野においては要求抽出プロセスと開発プロセスとを統合すべきであると提言され，統合型要求工学として要求抽出プロセスと開発プロセスの関連性について研究されている．その研究のひとつに筑波大学の中谷等の研究グループの PRINCE モデルがある．本研究のテーマである“要求抽出の時期を考慮したプロジェクトマネジメント・パターンに関する研究”は，要求工学とプロジェクトマネジメントの両分野を応用した研究であり，その PRINCE モデルに取り入れられている．一方，PM パターンの研究は，大学関係において行われているが，単に文献から抽出した問題と解決策の整理であり，実践向きではない．また，産業界ではプロジェクトマネジメントオフィス(PMO)経験からプロジェクトマネジメントの教訓やベストプラクティスを集めているが，単に集めただけであり，その有用性や活用の仕組みが整備されておらず，うまくいっていない．研究としては発展段階である．そこで，本論文における研究の目的は，従来の研究の課題を解決し，開発途中における要求抽出によるスケジュール遅延を防ぐために，実際のプロジェクトの開発履歴データを調査，分析した結果から得られた知見，即ち，要求抽出プロセスをマネジメントする技術とその活用の仕組みを提案することである.その成果として，実践的で且つ実用的な産業界のプロジェクトマネジメント技術を再利用できるように PM パターンとして整理し，PM パターンとフレームワークによって重要な問題を適切かつスピーディに解決する手法を提案した．更に，開発現場のプロジェクトマネジャがこの PM パターンとフレームワークを活用することによって要求抽出を計画的にコントロールしながらマネジメントできることを期待している．従って，要求工学とプロジェクトマネジメントの両分野に多少は貢献できると期待している．今後，本論文での研究を継続することによって，様々な分野のソフトウェア開発のプロジェクトにおいて，プロジェクトのリスクが高く突発的な要求抽出プロセスである成熟度タイプ（U_type）の PM パターンの拡充とフレームワークの改善を繰り返し，更に実践的で有効なものに発展していくと期待している．

以上を整理すると，実際のソフトウェア開発現場のプロジェクトの実態調査による知見から，要求抽出プロセスに関連するプロジェクトマネジメント技術を再利用できるように PM パターンとして整理し，その適用する仕組みであるフレームワークを開発し，提案できたことにより，本研究の目標は達成できたと考える．

7.3. 今後の課題

本論文において，要求抽出に起因するスケジュール遅延を防止するための PM パターンと，PMBOK の 9 つの知識エリアとマネジメントプロセス群からなる二次元のフレームワークを提案したが，研究および改善の余地はまだある．今後は，本論文で明らかにした要求抽出に伴うスケジュール遅延を防止するための PM パターンとその適用の仕組みであるフレームワークの研究を継続し，開発現場のプロジェクトマネジャやリーダに更に分かり易く適用可能なドメイン分野を広げる研究活動を展開していく．その研究活動の中で，当面の課題として以下の 4 点を挙げる．

1）PM パターンの補充

本論文で提案した典型的な 7 つの PM パターンと同様のマネジメント技術が他の複数のプロジェクトでも使われており，幾つかのプロジェクトでは，PM パターンを適用することによって，要求の抽出が遅延してもプロジェクトのスケジュールを遅らせることなく完了できていた．しかし，スケジュール遅延が生じていたプロジェクトもあった．その要因は，プロジェクトの後半に突発的に起こる予測不可能な要求抽出（U_type）への対処が十分にできていなかったことにある．その失敗は組み込みソフトウェア開発のプロジェクトで観測することができた．たとえば，ハードウェア内の問題をソフトウェアによって解決することが，プロジェクトの後期に決定された例などは，将に突発的に起こる予測不可能な要求抽出となる．このような突発的な要求抽出の問題に対処するために，複雑なプロジェクト環境での短期間の要求抽出へのマネジメント技術はリスクマネジメントと関連し，実際のソフトウェアエンジニアリングの最も重要な関心事の一つである．本論文で提案した 7 つの PM パターンはプロジェクトマネジメント技術の一部であり，更に有用度を向上させるための研究課題があると認識している．従って，そのような要求抽出によるスケジュール遅延を防ぐための PM パターンの研究を継続し，PM パターンを補充することを今後の課題のひとつと捉えている．加えて，これらの PM パターンを組み合わせて生成される高次 PM パターンの整理も残っている．

また，プロジェクトマネジメント経験の浅いプロジェクトマネジャやリーダにとって更に理解し易く且つ活用し易い PM パターンを提供するために，PM パターン項目の追加による構造改善や記述レベルを詳細化する必要があると認識している．例えば，解決策の合理性や事例の追加などを検討している．更には，アプリケーション分野やプロジ

ェクト特性に応じた PM パターンを整理し，PM パターンの知識ベースのフレームワーク提供に発展させていく所存である．

2） PM パターンのフレームワークの拡充

本論文では，プロジェクトマネジャがプロジェクトの問題をスピーティにかつ適切に解決するための PM パターンとそのフレームワークを提案した．本論文で提案したフレームワークは PMBOK の 9 つの知識エリアと 5 つのプロジェクトマネジメント･プロセスをベースにシステムエンジニアリングの開発工程を取り入れた二次元のマトリクス表である．これらを使いこなすには PMBOK の基礎知識が必要であり，プロジェクトマネジメントの経験があるマネジャであればフレームワークを PM パターンの選択や PM パターン間の矛盾や補完といったシミュレーションに上手く活用できるかもしれないが，基礎知識がなく未熟なプロジェクトマネジャでは活用が幾分か難しいかもしれない．今後，本論文で提案したフレームワークの適用研究を進め，経験の浅いプロジェクトマネジャでも，プロジェクトの問題を発見し，プロジェクトの計画，実行，監視コントロールに沿って問題と解決策を整理し，適用すべき PM パターンの妥当性が判断できる構造に進化させたいと考えている．また，プロジェクトの問題の大小によってプロジェクトマネジメント技術は様々であり，PM パターン間の矛盾や補完の考え方をさらに整理し，改善すべきであると考えている．今後，PM パターンの構成要素を取り入れたフレームワークを考案し，シンプルで使いやすく，効果がでる仕組みを提案していく所存である．また，PM パターンの補充と連動しドメイン分野ごとに整理することにより，ドメイン特有の問題と解決策を集めた PM パターン群からなる知識ベースのフレームワークを提案できると考えている．

3） PM パターンとフレームワークの客観的評価（定量的評価）

本論文で提案した PM パターンやフレームワークは比較的うまくいった実際のプロジェクトをベースにしている．これらを活用することにより実際のプロジェクトがどれだけ良くなったかを客観的に評価するためには定量的に測る指標および基準が必要である．そのためには，成功したプロジェクトと失敗したプロジェクトの要求抽出プロセス，PM パターンを活用した場合の比較や，適用前と適用後のデータ比較などによって評価基準の設定ができると考えている．例えば，評価基準として，開発規模，開発工数，リー

ドタイムなどの指標を定義し，PM パターンを適用するプロジェクトと適用しないプロジェクトの計画と実績の差分を統計処理し，分類評価するなどを考えている．また，ドメイン分野，プロジェクト毎に評価基準値が異なると想定しており，多くの実際のプロジェクトのデータを統計的に分析し，評価基準を設定することを考えている．しかし．これらの実証，分析にはかなりの時間がかかり，今後の研究課題のひとつと捉え，研究を継続する所存である．

４） 継続的な開発に伴う要求抽出プロセスのモデル化

本論文では，実際のプロジェクトの開発開始からシステムテスト完了までの１サイクルの要求抽出プロセスとそれに伴うプロジェクトマネジメント技術に留まっている．しかし，現実のソフトウェアの開発現場では，1 次リリース後，新たな要求が発生し継続的に開発が進められている．例えば，RUP のように，１次開発（Step1），２次開発（Step2），・・・とインクリメンタルな開発が実施されている場合もある．このインクリメンタルな開発の場合，要求抽出プロセスの成熟度タイプも異なるのではないかと想定している．インクリメンタルな開発の場合の要求抽出プロセスがどのようになるかは学会においてまだ明らかにされていない．それ故に，本研究の延長として，インクリメンタルな開発の場合の要求抽出プロセスのモデル化とそのプロジェクトマネジメント技術のパターン化は興味深く，有意義な研究になるのではないかと期待している．今後の研究課題のひとつと捉え，研究を継続する所存である．

謝辞

本研究を進める上でお世話になった多くの方々に深く謝意を表します．
問題解決へのアプローチを学ぶために本学の大学院に入学し，その甲斐あって，開発プロジェクトへのインプットである要求の抽出，検討，仕様化，さらに要求変更管理といった要求工学に興味をもち，研究活動の楽しさ，面白さ，厳しさを学ぶことができました．研究の進め方では，研究中盤に暗礁に乗り上げ，中座することが幾度もありましたが，なんとか博士論文を執筆できるところまでこぎつけました．このことは，私自身にとって，今後の研究活動の肥やしとなると期待しており，私を叱咤激励して下さった多くの先生方，研究グループの方々，会社の同僚，および家族に深く感謝しております．

九州工業大学大学院情報工学研究院情報創成工学研究系　橋本正明教授には，大学院入学当初から，問題解決へのアプローチ方法，モデリングの考え方など，研究全般にわたるご指導とご助言を賜るとともに，日頃より研究者としての考え方など様々な点においてご指導とご鞭撻を賜り，深く感謝申し上げます．

筑波大学大学院ビジネス科学研究科　中谷多哉子准教授には，要求工学の位置づけおよび重要性や本研究の起点となった実例プロジェクトによる要求抽出の実証研究へのアプローチ，プロジェクトマネジメントとの関連，パターンの概念など，ゼミおよび SSR（Joint Forum for Strategic Software Research）の研究会などで叱咤激励や暖かいご指導を賜り，深く感謝申し上げます．

九州大学システム情報科学研究院　鵜林尚靖教授，九州工業大学大学院情報工学研究院情報創成工学研究系　片峯恵一助教には，日頃から研究全般にわたり，関連研究へのアドバイス，客観的なご意見，課題へのご討論を賜り，深く感謝申し上げます．

九州工業大学大学院情報工学研究院　秋山義博教授には，PMBOK の基礎知識およびプロジェクトマネジメメントの重要性, PSP/TSP をはじめとする自律型プロセスとチーム開発および実際の開発現場におけるプロジェクトの厳しさなどご指導を賜り，深く感謝申し上げます．

九州工業大学大学院情報工学研究院情報創成工学研究系　橋本正明教授のゼミに参加いただいたパナソニック電工株式会社　三瀬敏朗氏，新屋敷泰史氏には，非正常系の要求抽出の考え方の伝授，また私の研究へのご意見を賜り，深く感謝申し上げます．

九州工業大学大学院の非常勤講師である　隅田重信氏には，隅田氏が安川情報システム

株式会社に勤務されている期間，公私ともどもご指導を賜り深く感謝しております．私が本学の大学院に進むきっかけを提供していただいたことに改めて感謝申し上げます．

安川情報システム株式会社　下司正雄氏には，本実証研究の背景となった実例開発プロジェクトのデータの提供，並びにプロジェクトマネジャへのインタビューなど貴重なご意見をいただくとともに，本研究の実現にご協力していただき，深く感謝申し上げます．

SSR（Joint Forum for Strategic Software Research）にご参加いただいておりました大阪大学大学院　津田道夫氏，東芝ソリューション株式会社　位野木万里氏，瓜田昌義氏，北川貴之氏，日本電気株式会社　藤原由希子氏，株式会社富士通研究所　栗原英俊氏，名古屋経営短期大学　近藤城史氏，東京女子大学　白銀純子准教授には，要求抽出の成熟度タイプとプロジェクトマネジメント・パターンへの貴重なるご意見を賜り，深く感謝申し上げます．

最後に，本論文をまとめるにあたり，論文の構成，要求抽出の成熟度タイプとプロジェクトマネジメント・パターンの関連性，論理性，整合性など，辛抱強くご指導，ご助言を賜るとともに，日頃より研究者としての考え方など様々な点においてご指導を賜りました九州工業大学大学院情報工学研究院情報創成工学研究系　吉田隆一教授に厚く御礼申し上げます．

参考文献

[1] 財団法人　日本情報システム・ユーザー協会(JUAS),” 企業 IT 動向調査 2011 報告書”

[2] 日経コンピュータ, ”法廷に出た動かないコンピュータ”, 2004.10.18, No.611.

[3] The Standish Group report, "CHAOS Summary 2009", Boston, Massachusetts, 23 April, 2009, http://www1.standishgroup.com/newsroom/chaos_2009.php.

[4] 独立行政法人　情報処理推進機構（IPA）, ソフトウェア・エンジニリング・センター(SEC)　著作・監修, ”ソフトウェア開発データ白書 2009”,日経 BP 社.

[5] Alan M. Davis, Just Enough Requirements Management: Where Software Development Meets Marketing, Dorset House, 2005.

[6] Barry W. Boehm: Software Engineering Economies, Prentice Hall, 1981.

[7] Alan M. Davis: Software Requirements　Revision　Objects, Functions, & States, Prentice Hall, 1993.

[8] Barry W. Boehm: “Developing Small-Scale Application Software Products: Some Experimental Results,” Proc. of the IFIP 8th World Computer Congress, 1980, pp.321-326.

[9] K. Beck, et al., Manifesto for Agile Software Development, http://agilemanifesto.org/, 2001.

[10] I. Jacobson, G. Booch, and J. Rumbaugh: The Unified Software Development Process, Addison-Wesley, 1999. （藤井拓監修, UML による統一ソフトウェア開発プロセス, 翔泳社, 2000.）

[11] G. Larman: Agile and Iterative Development A Manager’s Guide. （児高慎治郎,松田直樹監訳, 越智典子訳,初めてのアジャイル開発, 日経 BP 社, 2004.）

[12] 大西淳, 郷健太郎(著), ”要求工学―プロセスと環境トラック（ソフトウェアテクノロジー)”, 共立出版, 2004.

[13] I. Sommerville: ”Integrated Requirements Engineering”, IEEE　SOFTWARE Published by the IEEE Computer Society, pades 16,January/February 2005 IEEE SOFTWARE, pp. 16-23.

[14] 堀昭三, 中谷多哉子, 片峯恵一, 鵜林尚靖, 橋本正明,"レストラン注文管理システムにおける統合型要求プロセスの調査”, 信学技報(IEICE Technical Report), Vol. 107 No.331, pp.13-18, 2007.

[15] E. Carnel, R. Whitaker, and J. George, “PD and joint application design. A transatlantic comparison”, CACM, 36(4), pp. 40-47, 1993.

[16] 堀昭三, 中谷多哉子, 片峯恵一, 鵜林尚靖, 橋本正明, ”効率的な要求獲得によるリスク回避の実証研究”,プロジェクトマネジメント学会 2008 年度春季研究発表大会予稿集, pp.470-475, 2008.

[17] T. Nakatani, S. Hori, N. Ubayashi, K. Katamine, and M. Hashimoto, “A case study: Requirements elicitation processes throughout a project”, in The 16th International Requirements Engineering Conference (RE’08), pp. 241–246, IEEE, 2008.

[18] http://www2.gssm.otsuka.tsukuba.ac.jp/staff/nakatani/SSR09/2009ssr_RE.pdf

[19] S. Hori, T. Nakatani, K. Katamine, N. Ubayashi, and M. Hashimoto, “Project Management Patterns to Prevent Schedule Delay Caused by Requirements Changes,” in *The 4th International Conference on Software and Data Technologies (ICSOFT 2009)*, pp.115-120, INSTICC, 2009.

[20] T. Nakatani, S. Hori, M. Tsuda, M. Inoki, K. Katamine, and M. Hashimoto, “Towards a strategic requirements elicitation: A proposal of the PRINCE model,” in The 4th International Conference on Software and Data Technologies (ICSOFT2009), pp.145-150, INSTICC, 2009.

[21] 堀昭三, 中谷多哉子, 鵜林尚靖, 片峯恵一, 橋本正明, ”要求抽出に起因するスケジュール遅れを防ぐための PM パターンの研究について”, 信学技報(IEICE Technical Report), Vol. 109 , pp.55-60, 2009.

[22] S. Hori, T. Nakatani, K. Katamine, N. Ubayashi, and M. Hashimoto, “A DISCUSSION ON A FRAMEWORK OF PROJECT MANAGEMENT PATTERNS TO PREVENT SCHEDULE DELAY CAUSED BY REQUIREMENT ELICITATION”, Proc. IADIS Information Systems 2010 (Porto, Portugal), pp.414-418, 2010.

[23] S. Hori, T. Nakatani, K. Katamine, N. Ubayashi, and M. Hashimoto, “Project management patterns to prevent schedule delay caused by requirement elicitation,” *IEICE Transaction of Information and System,* pp.745-753, 2010.

[24] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series). Oxford University Press, 1977.

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

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

[27] Project Management Institute, "A Guide to the Project Management Body of Knowledge (PMBOK Guide) Third Edition", PMI, 2004.

[28] 不条理なコンピュータ研究会 (著),"IT 失敗学の研究",日経コンピュータ.

[29] W. Brown, H. McCormick, R. Malveau, T. Mowbray, "Anti Patterns: Refactoring Software, Architectures, and Project in Crisis", 1999.

[30] 古市奏文, 若松孝次, 湯村洋平, 井庭崇, "プロジェクト推進のためのパターンの提案" , http://ilab.sfc.keio.ac.jp/2007/output/files/mps64-furuichi.pdf, pp. 37-40,2007

[31] 井庭崇, 湯村洋平, 若松孝次, 古市奏文, "プロジェクト推進のパターン・ランゲージとその評価", http://ilab.sfc.keio.ac.jp/2007/output/files/jwein2007-pattern.pdf, 2007.

[32] H. Andrew, H. et al., Project Management Patterns And The Research-Teaching Nexus, http://www.ics.heacademy.ac.uk/events/8th-annual-conf/Papers/Andrew%20Hatch%20final.pdf, 2007.

[33] I. Stamelos, "Software project management anti-patterns, Journal of Systems and Software", In Press, Corrected Proof, Available online 9 September 2009.

[34] Scott W. Ambler, "Process patterns: building large-scale systems using object technology", Cambridge, UK: Cambridge University Press, p. 4. , 1998.

[35] K. Yamamoto, "Proposal for practical use of project lessons learned", In Proc. of the Society of Project Management (in Japanese), pp. 395-400, 2007.

[36] K. Murakami, K. Watanabe, S. Satho, Y. Akasaka, "Extraction and development activities of the "best practices" based on SECI model.", In Proc. of the Society of Project Management (in Japanese), pp. 390-394, 2007.

[37] 野中郁次郎, 梅本勝博, "知識管理から知識経営へ　ーナレッジマネジメントの最新動向ー", 人口知能学会, 2007.

[38] IEEE Std 610.12-1990, IEEE Standard Glossary of Software Enginnering Terminology, IEEE, 1990.

[39] ISO/IEC9126-1 Software engineering - Product quality - Part1: Quality model.

(JIS X 0129-1:2003) ソフトウェア品質特性（Quality Characteristics）.

[40] The Hillside Group. Ag's html template, http://www.hillside.net/index.php/ag-template.

[41] Linda Rising, Customer interaction Patterns, http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions/P11/P11.htm, (Pattern Languages of Programs'98), 1998.

[42] 三瀬敏朗, 新屋敷泰史, 橋本正明, 鵜林尚靖, 片峯恵一, 中谷多哉子：組み込みソフトウェア仕様抽出のための非正常系分析マトリクス, 情報処理学会組み込みソフトウェアシンポジウム 2004 論文集, pp.12-19, 2004.

[43] Y. Shinyashiki, T. Mise, M. Hashimoto, K. Katamine, N. Ubayashi, and T. Nakatani: *Enhancing the ESIM (Embedded Systems Improving Method) by Combining Information Flow Diagram with Analysis Matrix for Efficient Analysis of Unexpected Obstacles in Embedded Software,* Proceedings of the 14th Asia-Pacific Software Engineering Conference (APSEC'07), pp.326-333, 2007.

[44] 水野滋　監修, "管理者スタッフの新 QC 七つ道具", 日科技連.

[45] Robert E. Stein (著), 川辺 恭寛 (翻訳), 竹之内 隆 (翻訳), 椎名 茂 (翻訳), TOC 研究会 (翻訳), " TOC ハンドブック―制約条件の理論",日刊工業新聞社.

[46] R. Stevens, P. Brook, K. Jackson, and S. Arnold. *Systems Engineering: coping with complexity.* Prentice Hall, 1998.