受注開発型ソフトウェア開発における誤りの除去と管理に関する研究

Finding and Managing Defects in Contract-based Software Development

2008 年 7 月

早稲田大学大学院　理工学研究科
経営システム工学専攻　生産システムデザイン研究

中島毅

目次

第1章　　序論

1.1　研究の背景

近年，社会経済活動における IT 利用度は，かつてないほど高まり，IT システムの障害による業務・サービスの停止や機能低下の社会的影響は日々，深刻化してきている．これらの中で特に，受注開発で構築されたシステムに，問題が発生するケースが多い．

2003 年 4 月のみずほ銀行再編に伴う大規模システム障害では，自動現金預入払出機の障害と口座振替の遅延が生じ，連鎖的に生じたトラブルの収拾に約 1 か月を要した．これは，顧客と請負会社の間のシステム統合上の要求決定の遅れが，開発期間を圧迫し，十分なシステム検証が行えなかったことが原因である．2007 年 10 月の PASMO/Suica 窓口処理の停止は，JR 東日本や東京メトロなどの 8 都県 662 駅で自動改札機が起動しない事態を引き起こし、早朝通勤時の首都圏で約 260 万人の足を直撃した．これは，請負会社がインタフェース仕様を誤解し，またプログラム検証が不十分であったことにより，製品プログラムに誤りが紛れ込み，その誤りがある条件下で実行されたことにより発見されたことによる．こうした社会的影響に鑑み，受注開発型システムの効率性・信頼性・安全性の向上，特にその心臓部とも言えるソフトウェア品質の向上は，喫緊の課題となっている．

しかし，その一方で，IT システム開発をめぐる企業間競争は，顧客側と請負側の両方で，その激しさを増してきている．そのため，受注開発型のソフトウェア開発プロジェクトは，常に低コスト・短納期達成の強い圧力に晒されている．開発の各段階では，納期とコストの圧力ゆえに，混入された誤りを十分に除去することができずに，後段階に誤りを流出させている．**要求分析段階**では，顧客ニーズを適切に取り込めず，要求決定の遅延や，要求仕様上の誤りの混入を許し，その後の頻繁な要求変更の原因となってしまう．製品を開発する**エンジニアリング段階**では，時間とコストの制約から，検証作業を十分に行えずに，設計の過程で混入した誤りをに除去しきれずに製品確認段階へ流出させてしまう．そして，最終的に製品を確認する**製品確認段階**では，顧客から要請される頻繁な要求変更に対応するため，本来実施すべき製品検査を十分に実施できず，多くの誤りを製品中に残したまま出荷してしまう．

本研究は，受注開発型のソフトウェア開発を対象とし，その開発で混入するソフトウェア誤りを，開発各段階において，時間・コスト制約を満たしつつ確実に除去することを課題とする． この課題を解決するために，以下の 3 つの研究目的とアプローチを設定する．

【研究 1】

顧客要求分析を，「効果的に」行うことを目的とする．ここで「効果的に」は，開発規模の抑制，要求決定遅延の防止，要求仕様誤りの確実な除去，及び要求変更の抑制を指す. 顧客とのインタラクションを支援するプロトタイピングシステムの提案と評価を行う．

【研究 2】
　製品の最終検査を，効率的にかつ網羅性高く行うことを目的とする．テスト自動化システムの提案と評価を行う．

【研究 3】
　エンジニアリング段階における誤り除去活動を，コスト効率良く確実に管理することを目的とする．テストとレビュー実施のコスト対効果を最適化する品質計画支援システムの提案と評価を行う．

なお，本論文で用いる「誤り(defect)」は，要求仕様を満たさない状態（failure）を引き起こすソフトウェア（ドキュメント及びプログラム）内の欠陥(fault)，及び要求仕様そのものが顧客ニーズを満足できないあるいは適切に実現できないという要求仕様上の欠陥(requirements error)，から成るものと定義する．

1.2　研究対象と範囲

本研究で扱う研究対象と範囲を明確化する．

対象製品分野

表 1-1 に，本研究で対象とする製品分野とその定義，及び研究で注目した製品開発上の特性を示す．

表 1-1　製品分野の定義と本研究での対象製品分野

製品分野		研究の対象（○は直接対象範囲としたもの）			
カテゴリ	定義	研究1	研究2	研究3	研究で注目した特性
受注開発型システム	顧客依頼に基づく開発製品			○	・受注開発型システム開発プロセス
組込みシステム	製品に特定機能を実現する目的で組込まれるコンピュータシステム				
大規模組込みシステム	複数コンピュータとOSの組合せ使用．携帯電話，デジタル家電，車載ナビなど．				
小規模組込みシステム	1チップマイコン使用．家電品，玩具，車載制御機器など．		○		・反応的仕様が多 ・実機テストが重要
非組込みシステム	汎用コンピュータとネットワークで構成されるシステム				
情報通信システム	情報の収集・蓄積・処理・伝達・利用を行うシステム．企業情報・インターネット利用・通信制御システムなど．				
監視・制御システム	外部オブジェクトの監視と制御を行うシステム．プラント制御・交通管制システムなど．	○			・再利用を主とする開発 ・物理オブジェクトの存在

マイコン：マイクロコンピュータ
OS: オペレーティングシステム，家電: 家庭電気製品，ナビ：ナビゲーション

本研究は，受注開発型システムを対象とする．受注開発型システムは，特定顧客からの開発依頼に応じて開発する製品である．受注開発型システムは，組込みシステムと非組込みシステムとに分類できる．

組込みシステムは，産業機器や家庭電気（家電）製品などにおいて，特定の機能を実現する目的で組み込まれるコンピュータシステムである．組込みシステムは，大規模組込みシステムと小規模組込みシステムに分類できる．大規模組込みシステムは，複数のコンピュータと複数のオペレーティングシステムの組み合わせを用い，多機能かつ大規模なソフトウェアをもつ．代表例として，携帯電話及びデジタル家電製品がある．小規模組込みシステムは，1 チップ・マイクロコンピュータ（マイコン）を用い，機器制御や表示・操作系制御など少機能かつ小規模なソフトウェアをもつ．代表例として，家電製品，玩具，及び車載制御機器がある．

非組込みシステムは，汎用コンピュータとネットワークで構成されるシステムである．非組込みシステムは，情報通信システムと監視・制御システムに分類できる．情報通信システムは，情報の収集・蓄積・処理・伝達・利用を行うシステムである．代表例として，企業情報システム，インターネット利用システム，シミュレーションシステム，及び通信制御システムがある．監視・制御システムは，外部オブジェクトの監視と制御を目的としたシステムである．代表例として，原子力や鉄鋼などのプラント制御システム，及び交通管制システムがある．

研究 1 及び研究 2 に関しては，製品分野を絞り，その分野の開発上の特性に注目することで，実用性の高い技術開発を行った後，各研究のまとめの項で他の製品分野への適用可能性について議論する．

研究 1 は，監視・制御システムを対象とする．注目した特性は，以下の 2 点である．

- 要求仕様は，監視・制御システムと実世界の物理オブジェクトとのやりとりを取り扱う．
- システム構成と提供する機能仕様に大きな変更はなく，既存のシステム構成部品を変更して組み立てるイージーオーダ型の開発形態をとることが多い．

研究 2 は，小規模な組込みシステムを対象とする．注目した特性は，以下の 2 点である．

- 要求仕様の多くが，システムに対する刺激とそれに対する応答を記述する反応的仕様である．
- 製品確認段階では，製品に組み込まれた状態で，実機テストが行われる．その段階にならなければ妥当性を確認できない仕様もあり，要求仕様変更が頻繁に発生する傾向がある．

研究 3 は，受注開発型システム全般を扱う．

対象プロセス

図 1-1 に，本研究が対象とする**受注開発型のソフトウェア開発プロセス**を示す．

受注開発型のソフトウェア開発プロセスは，開発各段階に対応して，**要求分析プロセス**，**エンジニアリングプロセス**，及び**製品確認プロセス**からなる．

要求分析プロセスは，顧客からの開発依頼に基づいて開始し，顧客ニーズを抽出し，それを要求仕様にまとめるプロセスである．顧客のニーズを効果的に取り込むこと，及び要求仕様に誤りが少ないことが望ましい状態である．エンジニアリングプロセスは，要求仕様に基づきソフトウェア製品を製造していくプロセスである．要求仕様をプログラムに変換する過程で混入する誤りを，適切なタイミングで確実に除去すること，及び開発で達成すべき品質(Q: Quality)，コスト(C: Cost)，及び期間(D: Delivery)の目標を達成することが望ましい状態である．製品確認プロセスは，製品が要求通りであることを確認するプロセスである．製品が，顧客ニーズを満足し，要求仕様に適合していることが，望ましい状態である．

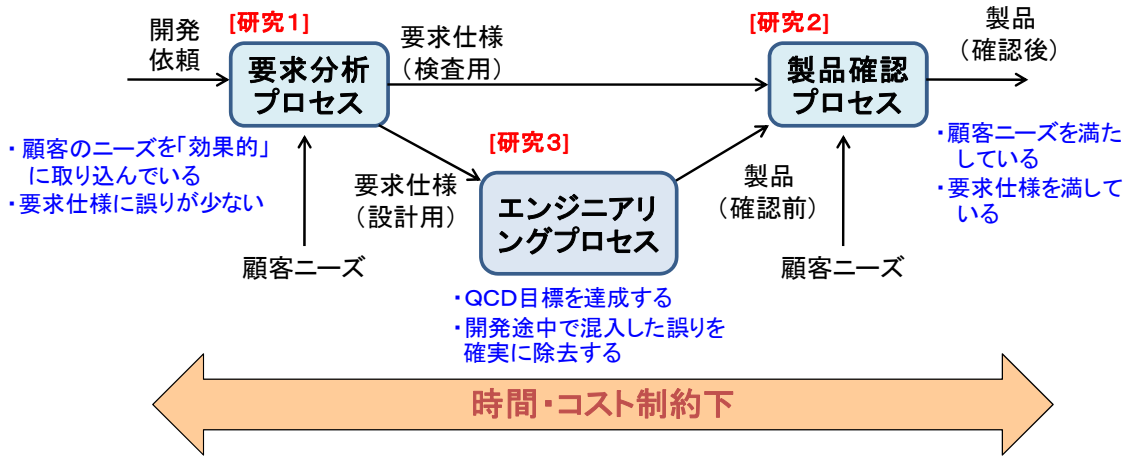


図 1-1　受注開発型のソフトウェア開発プロセス

研究 1 は要求分析プロセス，研究 2 は製品確認プロセス，研究 3 はエンジニアリングプロセスを対象とする．以下で，研究対象の 3 つのプロセスについて，プロセスを詳細化して記述するとともに，対象製品分野の特性を考慮し，各研究の目標状態を定義する．

(1) 要求分析プロセス

図 1-2 に示すように，要求分析プロセスは，顧客ニーズを抽出する要求抽出プロセス，顧客ニーズを分析し要求仕様として記述する分析・仕様化プロセス，及び要求仕様が顧客ニーズを満足していることを確認する要求妥当性確認プロセスからなる．

研究 1 は，製品分野として監視・制御システムを対象とし，要求分析者のイージーオーダ型の開発形態を支援するプロトタイピングシステムの技術開発を行う．このため，要求分析内の要求抽出，分析・仕様化，及び要求妥当性確認の 3 つのプロセスのサイクルにおいて，以下を達成することを目指す．

- 既存要求仕様へ顧客を誘導する．（要求抽出プロセス）
- 要求仕様記述言語を用いて，誤りの少ない要求仕様を作成する．（分析・仕様化プロセス）

- 顧客のシステム理解を高め，要求仕様の妥当性を確実に確認する．（要空妥当性確認プロセス）

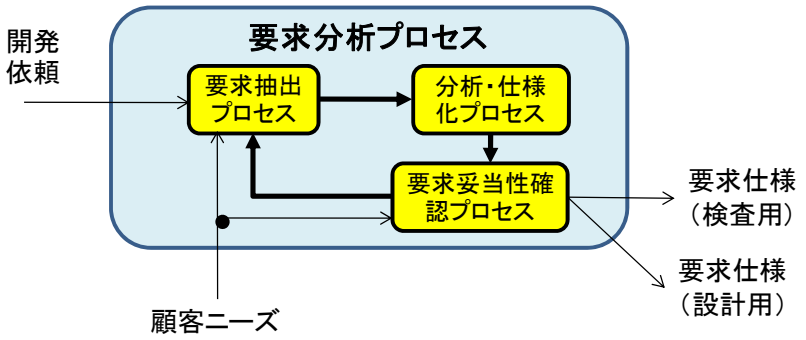


図 1-2　要求分析プロセス

(2) 製品確認プロセス

図 1-3 に示すように，製品確認プロセスは，製品が要求仕様を満足していることを検査する製品検査プロセス，製品が顧客ニーズを満足していることを確認する製品妥当性確認プロセス，及び検知誤りあるいは要求変更に応じてソフトウェアの修正を行うリワークプロセスからなる．リワークプロセスが実施されると，製品検査プロセスを再実施（再検査）する必要がある．

研究 2 は，製品分野として小規模な組込みシステムを対象とする．小規模な組込みシステムでは，製品検査が実機を使用したテスト（実機テスト）となるため実施コストが高く，また製品の性格上妥当性確認で要求変更が多数発生するため，製品検査の再実施を効率化することは特に重要となる．研究 2 は，製品検査者が検査（及び再検査）を短時間で網羅的にできることを目指して，テスト自動化システムの技術開発を行う．

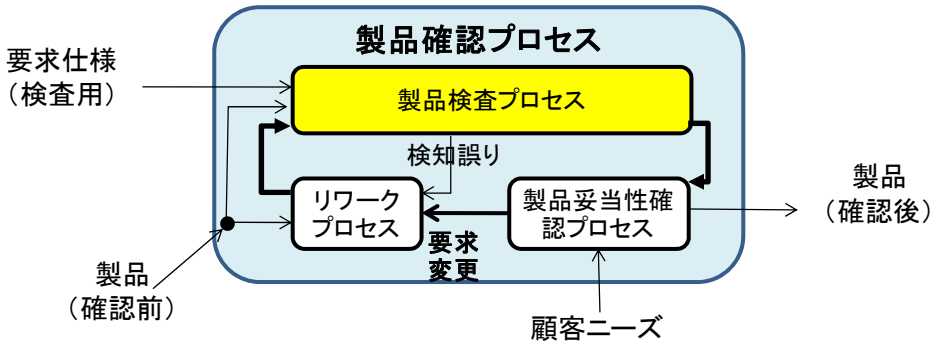


図 1-3　製品確認プロセス

(3)エンジニアリングプロセス

図 1-4 に示すように，エンジニアリングプロセスは，段階的に設計成果物を詳細化していく設計プロセスの連鎖，及び設計プロセスに対応する検査を行うテストプロセスの連鎖からなる．設計プロセスは，設計成果物を作成する技術プロセス，設計成果物を検証する品質プロセス，及び品質プロセスで検知された誤りを修正するリワークプロセスからなる．テストプロセスは，品質プロセス及びリワークプロセスからなる．品質プロセスにおける誤り除去活動は，主として，設計プロセスにおいてはレビュー，テストプロセスにおいてはテストである．

研究 3 は，受注開発型のシステム開発全般を対象とする．この研究は，エンジニアリングプロセスにおけるレビューとテストの実施を，品質プロセスとリワークコストの見積りに基づき適切に計画できることを目標状態とし，品質とコストの見積りモデルと，品質計画支援システムの技術開発を行う．

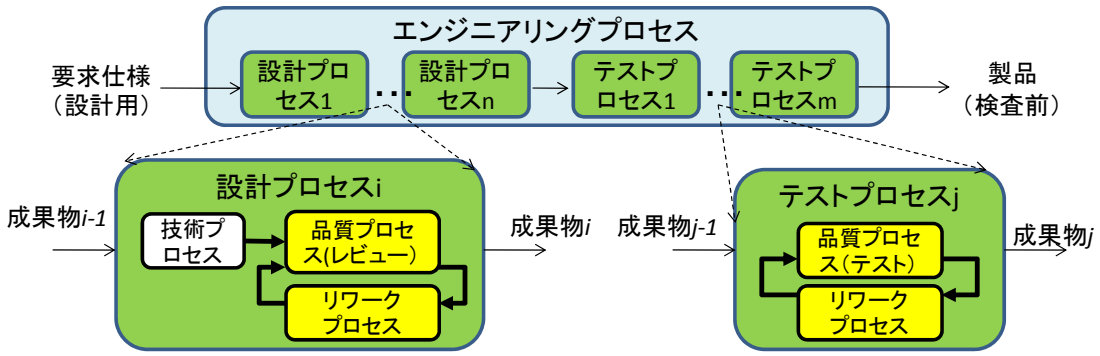


図 1-4　エンジニアリングプロセス

なお，研究 3 においては，ライフサイクルモデルとしてウォータフォールモデルを採用し，エンジニアリングプロセスを m+n 個の直列なプロセスからなるものとする．以後これらの直列に連鎖したプロセスを**フェーズ**と呼ぶこととする．

1.3　研究成果と概要

以下に本研究の成果と概要を示す．

［研究 1］監視・制御システムの要求分析を効果的に支援するプロトタイピングシステムの提案と適用評価[82]

オブジェクト指向モデルを要求仕様言語とし，既存要求仕様及び GUI(Graphical User Interface)仕様の部品化，直接操作によるプロトタイプ構築，物理オブジェクトのダイナミズムを含む要求仕様のインタプリタ実行，を実現するプロトタイピングシステムを提案し，適用評価を行った．

［研究 2］小規模な組込みシステムの最終製品検査を効率化する自動テストシステムの提案と適用評価[84]

状態遷移仕様に基づき，ICE(In-Circuit Emulator)を用いたメモリ設定と照合によって実機テストを自動化するシステムを提案し，適用評価を行った．

［研究 3］品質とコスト目標を達成する品質計画支援システムと，その定量的基盤を提供する品質とコストの見積りモデルの提案と適用評価[85]

エンジニアリングプロセスの各フェーズの品質プロセスへの投資コストから検知誤り量，及び検知誤り量から各フェーズのリワークプロセスのコストを推定することにより，エンジニアリングプロセス全体での作業コストを見積るモデルを提案・評価した．この見積りモデルを用い，品質マネジメントの基本戦略に沿って，品質計画を策定する手順と，その手順を支援する試作システムを開発し，適用評価を行った．

研究 1 は本論文の第 3 章，研究 2 は本論文の第 4 章，研究 3 は本論文の第 5 章にそれぞれ対応して述べる．

なお，本論文の各研究の実証実験では，なるべく定量的かつ客観的に評価を行う姿勢をとりながら，少ない適用例から，（定性的な評価も併用して）システム適用の有効性を評価するアプローチをとっている．一般に，ソフトウェアに関するシステム提案では，適用対象であるソフトウェアプロジェクトの独自性が強く再現性のある実適用を実施しにくいこと，また実適用そのものの実施期間が長くコストも高いことにより，実適用を通じて有効性の完全な検証を実施することが困難である．そのため，この分野の多くの研究[98][99][100]が，限定された適用結果に基づいて評価を実施している．本研究もそうした評価方法を採用している．

1.4 本研究の構成

以下に本研究の構成を示す．

第 2 章では，背景知識と関連研究について述べる．要求分析プロセス，製品確認プロセス，及びエンジニアリングプロセスの誤り除去活動について，それぞれ背景知識と，その中での本研究の位置づけと関連研究について明確にする．

第 3 章から第 5 章は，3 つの研究成果に対応している．第 3 章は，研究 1 の成果に対応して，オブジェクト指向仕様に基づくプロトタイピングシステムの提案と適用評価について述べる．第 4 章は，研究 2 の成果に対応して，状態遷移仕様に基づくテスト自動化システムの提案と適用評価について述べる． 第 5 章では，研究 3 の成果に対応して，品質とコストの見積りに基づく品質計画支援システムの提案と適用評価について述べる．各章は，目的，従来システムと問題点，提案システムと技術課題，提案方式（あるいはモデル），実証実験，及びまとめの構成をとる．

最後に，第 6 章において，結論と今後の展望について述べる．

第2章　　背景知識と関連研究

2.1　要求分析プロセスの誤り除去活動

要求分析プロセスの誤り除去の技法とシステム

要求分析プロセスにおいて混入する誤りは，要求決定の遅延や要求変更となって，エンジニアリングプロセスに多大な影響を与える．要求分析プロセスの品質確保は，受注開発型のソフトウェア開発の成否を決める重要な事項である．

JIS X0129-1[2]にソフトウェア製品がもつべき品質特性として，機能性，効率性，信頼性，使用性，保守性，及び移植性が定義されている．これらの品質特性に対する要求は，品質要求と呼ばれ，機能要求とともに要求仕様として記述される．要求仕様に記述された機能要求と品質要求は，エンジニアリングプロセスの中でその実現が設計・検証される．IEEE Std. 830[3]は，要求仕様に含まれる誤りとして，間違い，あいまい，矛盾，抜け，実現不可能，及び検証不可能を定義している．

要求プロセスにおける誤り除去作業は，要求の妥当性確認と呼ばれる[1]．妥当性確認は，顧客の運用環境においてその目的を達するかどうかを確認する作業である[4]．要求プロセスにおける妥当性確認の技法は，要求レビュー，モデルの妥当性確認，及びプロトタイピングの 3 つのカテゴリに分類される[1]．以下でそれぞれについて述べる．

(1) 要求レビュー

レビューは，レビュー対象を直接人間の目で確認する作業であり，要求の妥当性確認の手段として最も多く使用されている．要求レビューの対象は，要求仕様書あるいは要求モデルである．レビューの効果と効率を上げる方法として，レビュープロセスの形式化及びレビューポイントの設定がある．

レビュープロセスの形式化は，レビューのプロセスを形式化することで，レビューアの属人性を極力排除しレビューのプロセス品質の底上げを行うものである．インスペクション[21] [22]は，レビュープロセスの形式化に属する技法のひとつである．インスペクションは，参加者の役割，手順，データ収集法などを明示的に定義している．役割には，モデレータ，記録者，リーダ，及び開発者がある．手順は，計画，概観，準備，ミーティング，再作業，及びフォローアップからなる．ミーティングの結果は，あらかじめ定義されたフォーマットに従って報告書としてまとめられ，管理者に通知される．インスペクションでは， 開始基準と終了基準，エラー検知を支援するためのチェックリストなどのツールの整備が求められる．

レビューポイントの設定は，レビュー参加者が何に従い，何に注意を向けて，誤りを見つけ出すかを，事前に設定しておくことことで，レビューの効果と効率の向上をねらう方法である．レビューポイントの設定における代表的な技法は，チェックリストに基づくレビュー（Checklist-based Reading），及びユーザやテスト担当者などの観点別に行うレビュー(Perspective-based Reading,あるいは Usage-based Reading)である．これらの技法に

関する効果と効率は，種々の条件下で比較されている[23][24][25][75].

レビューは，エンジニアリングプロセスの検証の手段としても使用できる．要求の妥当性確認としてレビューを用いる場合，レビュー観点を要求仕様の誤り種別と対応付ける方法，顧客の参加の方法などが，技法の効果と効率に大きく影響することが報告されている[5][6].

(2) モデルの妥当性確認

モデルの妥当性確認は，要求の分析・仕様化で作成した要求モデルに対して，そのモデルの特性を利用して，妥当性確認を実施する方法である．要求モデルとその確認方法の組み合わせが，妥当性確認の技法となる．

モデルの妥当性確認には，顧客による理解容易性向上に重点をおくことにより，レビュー効果と効率を上げる技法と，記述言語の推論機構を利用することにより，要求の誤りを検出する技法がある．

前者の代表的な技法は，システムの動作をシナリオとして表現し，顧客との対話を促進する方法である[13][14]．シナリオは，顧客にとって比較的理解しやすい要求モデルであり，また作成と修正に時間がかからないため要求抽出にも利用しやすい。オブジェクト指向モデリング言語 UML® (Unified Modeling Language)[15][16]は，タイプの異なる図的な要求仕様記述言語の集合とそれらの間の関係を定義している。UML においてシナリオは，ユースケースとして表現され，他のモデル（クラス図，状態図，シーケンス図など）との関係が明確に定義されているため，要求の分析・仕様化及び妥当性確認に用いる記述形式として実用性が高い。顧客にシステムイメージを伝えるため，アニメーションなどの補助的な手段が併用されることがある[17]．3 章の研究では，UML の前身であるオブジェクト指向モデリング言語 OMT(Object Modeling Technique)[56]を，要求仕様記述言語として利用している．

後者の技法は，形式的仕様記述言語及びそのモデル検証である．数学的なバックボーンを持つ形式的言語を用いて要求仕様を厳密に記述することにより，形式的推論を利用して，要求のあいまいさ，矛盾，及び抜けを発見する技法である[18][19]。形式的仕様記述言語は，読み方について訓練を受けていない顧客にとって難解となるため，通常，顧客が直接レビューするための要求モデルとしては利用しにくい.

(3) プロトタイピング

プロトタイピング[7][8]は，システムの見た目とふるまいをプロトタイプとして実装し，妥当性確認のための顧客との対話を促進する方法である。作成されたプロトタイプは，顧客にとって直観的に理解し易いので，要求の妥当性確認を効果的に行うことができる．しかし一般に，プロトタイプ作成には多くの作業コストと時間がかかるため，開発のコストと期間の制約を満たせない場合には，プロトタイピングは利用できない.

この作業コストと時間の問題を軽減するために，種々のプロトタイピングシステムが提案されている．要求分析用プロトタイピングシステムは，大きく２つのカテゴリに分類できる．一つは，強力な GUI 構築機能と専用プログラミング言語環境を併せ持つタイプのシステムである（**GUI 指向言語システム**と呼ぶ[82]）. GUI 指向言語システムとして，

Visual Works®や Visual Basic® for Application がこのカテゴリに入る．もう一つは，要求仕様記述言語でシステムのふるまいを記述し，その仕様と GUI 仕様とを関連付けてプロトタイプを作成するタイプのシステムである（**仕様プロトタイパ**と呼ぶ[82]）．拡張状態遷移図に基づく Statemate® [9]，オブジェクト指向の仕様に基づく Rhapsody® [10]，ObjecTime® [11]，ペトリネットに基づく PROT™ [12]等がこのカテゴリに入る．3 章の研究では，オブジェクト指向仕様に基づく仕様プロトタイパを研究のベースとして利用している．

要求分析の作業サイクル

本研究が対象とする受注開発型ソフトウェア開発において，要求分析プロセスは顧客と共同で進める作業である．この視点から捉えると，図 2-1 に示すような 2 つの作業形態と作業サイクルがある [44].

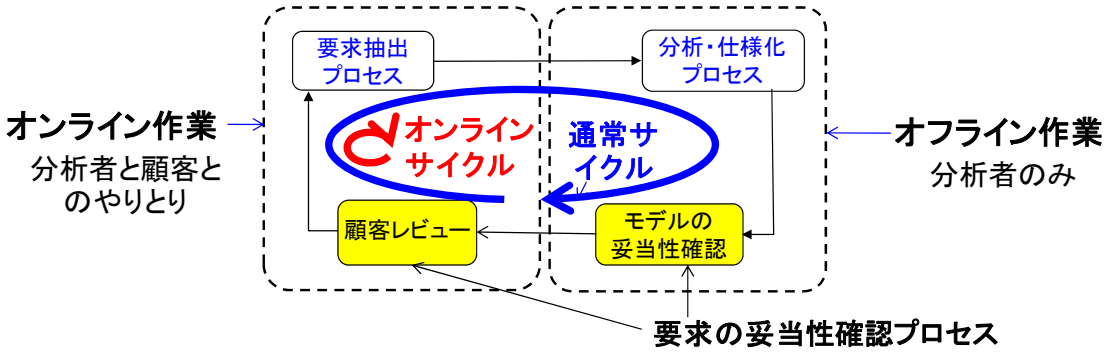


図 2-1　要求分析プロセスにおける 2 つの作業形態とサイクル

2 つの作業形態とは，オンライン作業とオフライン作業である．**オンライン作業**とは，分析者が要求の抽出と妥当性確認を目的に直接顧客とやり取りする作業であり，**オフライン作業**とは，（顧客が参加せず）分析者のみで分析・仕様化及びモデルの妥当性確認を行う作業である．2 つの作業サイクルとは，通常サイクルとオンラインサイクルである．**通常サイクル**とは，オフライン作業である分析・仕様化及びモデルの妥当性確認と，オンライン作業である要求抽出及び妥当性確認（顧客レビュー）の繰り返しである．**オンラインサイクル**は，オンライン作業における要求抽出と顧客レビューの繰り返しである．

プロトタイピングシステムには，以下のことが求められる．

- 顧客との対話は限られた時間と頻度でしか実施できないため，顧客の参加を必要とするオンライン作業には，高い効率が求められる[55]．オンラインサイクルのサポートにより，要求抽出及び妥当性確認を素早く回すことが重要となる．

- 要求の妥当性確認のプロセスの効果と効率を向上させるためには，要求抽出及び分析・仕様化のプロセスと連携が重要である[1][52]．例えば，要求のレビューは，記述された要求仕様の妥当性確認を目的とするが，その作業を通じて新たな要求を抽出する機会ともなる[6]．モデルの妥当性確認は，分析・仕様化で用いる要求モデルを入力とすることが前提である[1]．そして，プロトタイピングは，既に抽出された要求に対する妥当性確認，及び新たな要求の抽出の 2 つの目的をもって実施される場合が多い[53]．従って，妥当性確認の 3 つのカテゴリに属する技法・システムを適切に組合せることが必要である．
- 単に顧客が望むシステム要求を抽出するのではなく，顧客の要求を既存システムの仕様へと積極的に誘導することが求められるようになってきている[44]．この要求に対して，業務分野のオントロジーと要求仕様を対応付けることにより，要求の誘導を効率的に行う技法 [48][49]，及び既存のシステムをベースとしてビジネスゴール及び利用シナリオを構築する支援環境に関する研究[50]がなされている．

研究の位置づけと関連研究

本論文の第 3 章では，監視・制御システムを対象製品分野とし，要求の妥当性確認プロセスの技法として**プロトタイピング**を対象とする．本論文第 3 章の位置づけと関連研究を表 2-1 に示す．

一般に，顧客との対話の時間（オンライン作業時間）が限られる場合が多い．また，監視・制御システムの製品分野では，既存製品群から顧客要求に近いものを選択しカスタマイズする開発形態への変化が顕著である [20]．

本研究で提案するプロトタイピングシステムは，**仕様プロトタイパ**をベースとしている．これによって，要求分析者の分析・仕様化作業の支援を行うことができる．提案システムは，従来の仕様プロトタイパに対して，オンラインサイクルのサポートと，監視・制御システム特有の事項を考慮し，既存システム仕様への誘導を行えるように改良している．

表 2-1　本論文第 3 章の位置づけと関連研究

要求の妥当性確認プロセスに求められること ＼ プロトタイピングツールのカテゴリ		受注型システム開発全般: 要求抽出および分析・仕様化のプロセスとの連携			監視・制御システム特有: 既存システム仕様への誘導
		モデルの妥当性確認のサポート	要求モデルを使用（分析・仕様化のサポート）	オンラインサイクルのサポート	
GUI指向言語システム		無	無	VisualWorks VBA	困難
仕様プロトタイパ	従来型	一部（文法と，モデル間の整合性チェック）	オブジェクト指向モデル+Rhapsody[10] オブジェクト指向モデル+ObjecTime[11] Statecharts[57]+Statemate[9] Petri-net+PROT[12]	困難	困難
	提案方式	一部（文法と，モデル間の整合性チェック）	本論文第3章 オブジェクト指向モデル+ROAD/EE[82][83]		

2.2　製品確認プロセスの誤り除去活動

製品確認プロセスの誤り除去の技法とシステム

製品確認プロセスにおける誤り除去は，製品の妥当性確認プロセスと製品検査プロセスにおいて行われる．

(1) 製品妥当性確認プロセス

製品妥当性確認プロセスの技法の代表的なものは，受入テストである．受入テストは，システムが組み立てられた状態で，運用形態もしくはそれに近いテスト環境で，顧客がその製品を受け入れ可能であるかどうかを判断するテストである[1][4]．受入テストでは，その製品が，有効性，生産性，安全性，及び満足性の 4 つの使用時品質[2]を満たしていることが確認される．

(2) 製品検査プロセス

製品検査プロセスでは，製品妥当性確認プロセスにより生じる要求変更，及び製品検査プロセス自体が検知した誤りを，リワークプロセスにおいてプログラム修正した後に，製品を再度テストしなければならない．この段階では以下の問題がある．

[P1] 製品としての全貌が初めて顧客の目に晒されるため，要求変更の要請を受け易い．要求変更を受け入れれば，プログラムを修正する必要が生じる．

[P2] 要求変更及び要求の誤りに伴うプログラムへの修正は，プログラムの他の部分に予想外の悪影響を及ぼす可能性がある．こうした悪影響がないことを保証するテストが回帰テストである．回帰テストは，変更の影響範囲が特定できない場合，実施済みテストの全面的なやり直しを必要とする．

[P3] テスト環境に，実運用環境もしくはそれに近いテスト環境でテストを実施しなければならないため，テストデータの入力と結果照合を自動化することが難しい．

回帰テストの効率化の技法・システムは，テスト範囲の限定及びテスト自動化という２つのカテゴリに大別される．**テスト範囲の限定**は，プログラムに加えた変更が，①プログラムの他の部位にどのように影響を与えるか，また，②要求機能にどのような影響を与えるか，を把握することで，合理的にテスト範囲を削減するための技法・システム群である．このカテゴリ属する技法・システムとして，要求から設計コンポーネントへの双方向の追跡性の定義[3]，追跡性管理のための支援システムなどが提案されている[54]．

テスト自動化には，要求仕様からテストシナリオを生成するシステム（テストシナリオジェネレータ），設定されたテストデータに基づき回帰テストを自動化するシステム（自動テスタ）などがある．テストシナリオジェネレータでは，反応的な仕様（状態遷移仕様）に基づくものが実用化されているが[67][68][69][72]，変換的な仕様及び品質要求についての研究はほとんど見られない．自動テスタには，テスト用データを手動によって定義するものとして，テストの操作記録を再実行する方式[70][71][87]，及びスクリ

プト言語によりテスト内容を記述する方式[87]がある．自動テスタでは，被テストプログラムに入力を与え，プログラムの出力を参照して期待値と照合するためのインタフェースの設定が重要となる（本論文ではこれを**テスト界面**と呼ぶ）．非組込みシステムでは，Web ページや Active-X，CORBA など入出力インタフェースが標準化されたコンポーネントが用意されているケースが多いため，このインタフェースをテスト界面とする自動テスタが多い[87]．これに対して，組込みシステムでは，ハードウェアと組み合わせた状態での実機テストとなるため，テスト界面を適切に設定することが難しく，それがテストの自動化を困難にしている．実用化された数少ない例として，マイコンポートをテスト界面とする自動テスタがある[70]．

研究の位置づけと関連研究

本論文の第 4 章では，小規模な組込みシステムを対象製品分野とし，製品検査プロセスにおける**テスト自動化**を対象とする．本論文第 4 章の位置づけと関連研究を表 2-2 に示す．

小規模な組込みシステムの製品分野では，製品検査プロセスは，ハードウェアと組み合わせた実機テストとして実施されること，及び要求仕様に占める反応的な仕様の割合が多いこと，という 2 つの特性に注目している．

本研究では，テスト自動化システムのテスト用データ設定方法として，要求仕様変換方式を提案する．この方式によって，要求仕様の変更に伴う再テストの実行を，一貫して自動化することが可能になる．提案システムは，状態遷移仕様からテストシナリオを生成する部分に従来のアルゴリズム[67][68][69][72]を，テスト界面にデバイスドライバインタフェースのメモリインタフェースを採用している．

表 2-2　本論文第 4 章の位置づけと関連研究

製品分野 / テスト用データ設定方法		組込みシステム（実機テスト）	非組込みシステム（汎用コンピュータ上のテスト）
手動定義方式	操作記録方式	ADO[22] テスト界面： マイコンポート	TestComplete[28] テスト界面： 標準コンポーネントI/F
	スクリプト方式	無	
要求仕様変換方式	反応的仕様	本論文第4章 testCASE[26] 状態遷移仕様→テストシナリオ[19][20][21] テスト界面： デバイスドライバのメモリインタフェース	無
	変換的仕様	無	無
	非機能的仕様（品質要求）	無	無

2.3　エンジニアリングプロセスの誤り除去活動

エンジニアリングプロセスの誤り除去活動の見積りモデル

エンジニアリングプロセスにおける誤り除去は，品質プロセスで行われる．品質プロセスは，設計プロセスではレビュー，テストプロセスではテストが活動の主体である．

(1) 品質プロセスの効果と効率の定量的把握

レビューとテストの効果を表す指標は，混入誤り量 R に対する検知誤り量 D から計算する検知率 D/R，また効率を表す指標は，技法の適用にかけた作業コストあたりの検知誤り量で表現することが多い[36][37][38].

効果＝検知率 D/R
効率＝検知誤り量/活動にかけた作業コスト　　（Gilb ら[88]）

レビューとテストの効果及び効率は，プロジェクトの特性，組織や個人の能力差，及び設計の技法との組み合わせに大きく依存するため，ばらつきが大きい．しかし，その影響は統計的なばらつきと考え，誤り検知効率一定など比較的単純なモデルを用いて，レビューやテストの作業コストの見積りを行う場合が多い[24].

レビューへの投資対効果を，同一の誤りをテストで検知した場合とのコスト差異として表現する指標の提案がある [92][93][94]．本論文の第 5 章で提案する見積りモデルは，レビューとテストを対峙したものと捉えず，エンジニアリングプロセス全体でのコスト最適化を扱うという意味で，これらの研究を一般化したものである．

(2) 品質とコストの見積りモデル

品質の見積りとは，ソフトウェアに含まれる残存誤り量を推定することに相当する．残存誤り量を推定するモデルは，係数法，外挿法，及び総量管理法に大別される．**係数法**は，プロジェクトの特性を表すパラメータ，及び過去のプロジェクトの実績データで調整した係数を用いて，エンジニアリングプロセス全体での混入誤り量及び検知誤り量を見積る方法である．例えば，COCOMOⅡの COQUALMO モデル[43][45]などがある．**外挿法**は，対象プロジェクトの検知誤り量の測定履歴をモデル曲線にフィッティングさせ，将来の検知誤り量及び残存誤り量を推定する方法である．単一フェーズ内の外挿法として，成長曲線モデル（横軸：テスト実施コスト，縦軸：累積検知誤り量）があり，フィッティングするモデル曲線として，指数型モデル[74][89]，S 字型モデル[90]などが提案されている．エンジニアリングプロセス全体の外装法として，Rayleigh 曲線に基づく推定法[39]がある．この場合，グラフの横軸は開発フェーズ，縦軸は各フェーズの検知誤り量で表現する．

総量管理法は，フェーズ毎の混入誤り量は所与のものとして，各フェーズの検知誤り量の見積りに基づき，順次各フェーズの残存誤り量を計算していく方法である．例えば，総量管理モデル[42]がある．総量管理モデルでは，各フェーズにおける混入誤り量が組織能力値とプロジェクト特性から決定される定数値であるとして，検知誤り量を，品質プロセスコスト×ヒット率（残存誤り量に依存せず効率一定と考える方法）で推定する．

検知誤り量からリワークコストを推定する方法として，Kan[40]のモデルが提案されている．このモデルでは，誤り 1 件あたりのリワークコストが，検知されたフェーズ毎で一定としている．

総量管理モデルと Kan のモデルでは，誤りデータ種別として検知フェーズのみが考慮されている．また，これらのモデルを連結し，各フェーズの品質プロセスコストから，各フェーズの検知誤り量と，そのリワークコストを推定し，それらを総計してエンジニアリングプロセス全体のコストを見積るモデルは，提案されていない．

研究の位置づけと関連研究

本研究の第 5 章では，受注開発型システム全般を対象製品分野とし，エンジニアリングプロセスにおける品質計画支援を目的にして，各フェーズでの品質プロセスの実施コストからエンジニアリングプロセスの品質とコストを見積るモデル及びシステムを提案する．本論文第 5 章の位置づけと関連研究を表 2-3 に示す．

本研究で提案する見積りモデルは，総量管理法をベースとして，誤りデータ種別の考慮範囲を混入及び検知フェーズにした上で，見積り範囲をリワークコストの推定まで含めている．

表 2-3　本論文第 5 章の位置づけと関連研究

<table>
<tr><td colspan="2" rowspan="2">誤りに関する
見積り項目

誤りデータ種別
の考慮範囲</td><td colspan="4">品質見積り</td><td rowspan="2">コスト見積り</td></tr>
<tr><td colspan="2">単一フェーズ内</td><td colspan="2">エンジニアリング・プロセス全体</td></tr>
<tr><td colspan="2"></td><td>検知誤り量と残存誤り量の推定</td><td>混入誤り量の推定</td><td>検知誤り量と残存誤り量の推定</td><td>混入誤り量の推定</td><td>リワークコストの推定</td></tr>
<tr><td colspan="2">エンジニアリングプロセス全体</td><td colspan="2"></td><td colspan="2">COCOMOII・COQUALMOモデル[45]</td><td>無</td></tr>
<tr><td rowspan="2">検知フェーズ</td><td>外挿法</td><td colspan="2">指数型モデル[74][89]
S字型モデル[90]</td><td colspan="2">Rayleighモデル[39]</td><td rowspan="2">Kan[40]</td></tr>
<tr><td>総量管理法</td><td colspan="4">総量管理モデル[42]</td></tr>
<tr><td>混入及び検知フェーズ</td><td>拡張総量管理法</td><td colspan="5">本論文第5章[85]
提案モデル</td></tr>
</table>

第3章　　オブジェクト指向仕様に基づくプロトタイピングシステム

3.1　本章の研究の目的と位置づけ

本章の研究目的は，監視・制御システムを対象として，要求分析を効果的に支援するオブジェクト指向仕様に基づく仕様プロトタイパの提案と適用評価である．

監視・制御システムの製品としての特性は，実世界オブジェクトの監視と制御を扱うことである．また，この製品分野は，事業的に成熟しており，海外市場の新規顧客の獲得を狙う一方，開発コスト抑制のため，既存システムをベースにカスタマイズ開発するイージーオーダ型の開発スタイルを志向している．この場合，要求分析プロセスにおいて以下のような特性をもつ．

- 顧客が，同種のシステムに対する利用経験をもたない．
- 要求分析者と顧客との対話が，限られた時間と頻度でしか行えない．
- 請負側企業は，実績のある既存のシステムを持ち，再利用率向上のため既存のシステム仕様へと誘導したいと考えている．

図 3-1 に本章の研究の対象プロセスに沿って，このような製品分野の特性から生じるプロトタイピング実施上の目標を示す．

[目標1]　既存要求仕様への誘導
[目標 2]　オンライン作業の効率的実施
[目標 3]　監視･制御システムのふるまいの理解性向上

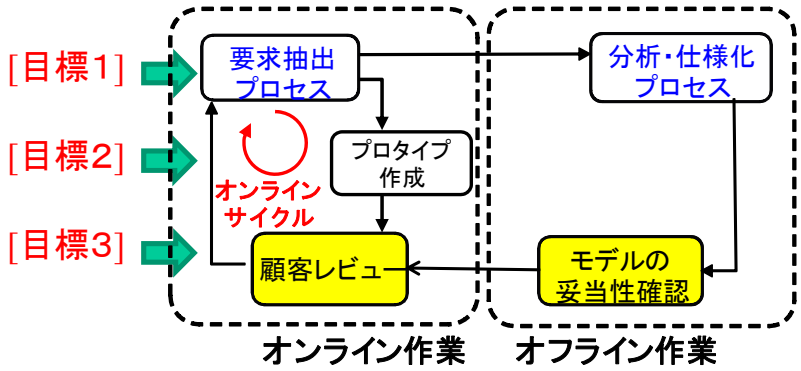


図 3-1　本章の研究対象プロセスと目標

3.2　従来システムと問題点

オブジェクト指向仕様に基づく仕様プロトタイパの従来システムとして, Rhapsody[10]と ObjecTime[11]がある．このシステムは，オブジェクト指向要求仕様記述モデルを用いてシステムのふるまいの仕様を記述し，そのふるまい仕様を GUI 仕様と関連付けてプロトタイプを作成するシステムである．

図 3-2 に，オブジェクト指向仕様に基づく仕様プロトタイパ（以下従来システム）のシステム構成を示す. 従来システムは, ふるまい仕様エディタ, インスタンスエディタ, プロトタイプ定義エディタ，及びプロトタイプジェネレータの４つの部分からなる．従来システムの特徴は，以下の 2 点である．

- ふるまい仕様と GUI 仕様との関連付けがインスタンスレベルで行われること．
- プロトタイプデータを作成した後，プロトタイプジェネレータがプログラムコードを生成し，そのコードをコンパイル・リンクすることで，プロトタイプをビルドすること．

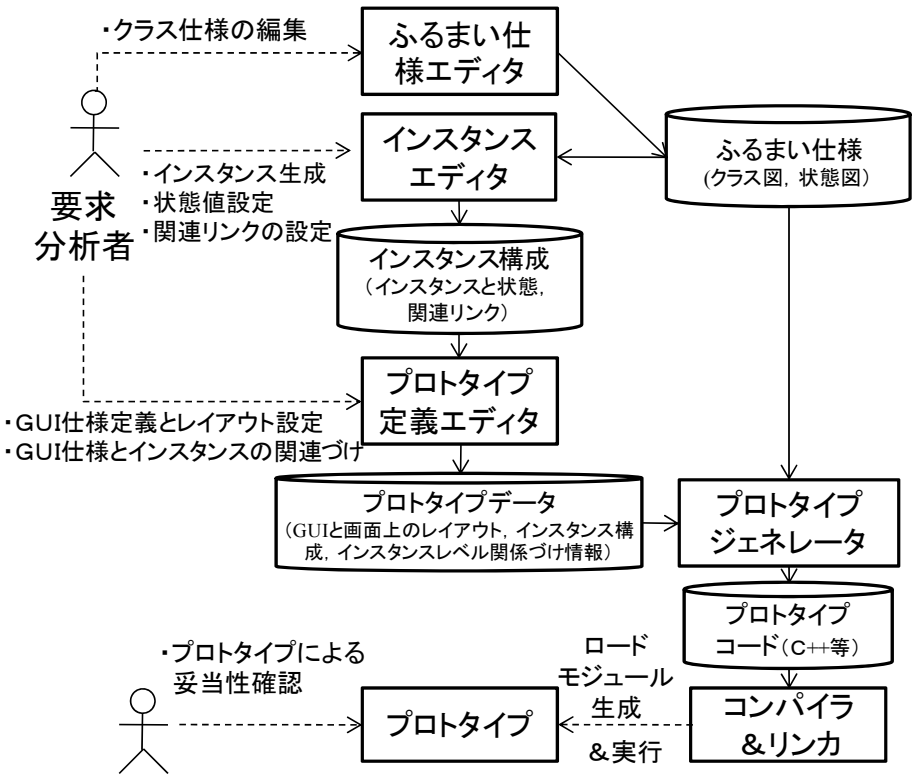


図 3-2　仕様プロトタイパのシステム構成

以下に，プロトタイプ構築手順を記述する．

(1) **ふるまい仕様の記述**：ふるまい仕様エディタを用いて，一つのクラスのふるまいとして一つの状態図を割り当ててふるまい仕様の記述を行う．

(2) **インスタンス構成の記述**：インスタンスエディタを用いて，プロトタイプを構成するインスタンスを記述する．

(3) **プロトタイプ定義**

(3-1) **GUI 仕様定義**： プロトタイプ定義エディタを用いて，プロトタイプのもつべき画面（プロトタイプ画面）の GUI 仕様を定義する．プロトタイプ定義エディタ上の GUI パレットには，円，矩形，多角形，折れ線などの基本描画要素，基本描画要素をグループ化してあらかじめ登録しておくグループ描画要素，画面レイアウト用のフレーム，及びボタンやシリンダのような操作入力用 GUI 部品があり，要求分析者はこれらを GUI パレットからプロトタイプ画面上に直接張り付け，さらに変形・移動させることでプロトタイプ画面のレイアウトを作成する．

(3-2) **インスタンスと GUI 仕様の関連付け**： プロトタイプ定義エディタを用いて，インスタンス図上のインスタンスの状態と，プロトタイプ画面上のアイコンとの対応づけを行う．例えば，o1 の「ON」状態を「青色のアイコン」に，o1 の「OFF」状態を「無色のアイコン」に対応付ける．アイコンとして，単純な繰り返し動作を表現するアニメーションを定義することもできる．この対応付けの操作は，クラス毎で，インスタンス数と表示アイコンを関連づける状態数を掛けた数だけ必要となる．さらに，GUI 部品への入力と，それによって生じるイベントの解釈を対応付ける．例えば，図 3-4 で「ON」ボタンを，インスタンス o1 に対する「on」イベント（クラス C2 の状態図中のイベント）へと対応付ける．この対応付けの操作は，クラス毎で，インスタンス数と入力イベント数を掛けた数だけ必要となる．

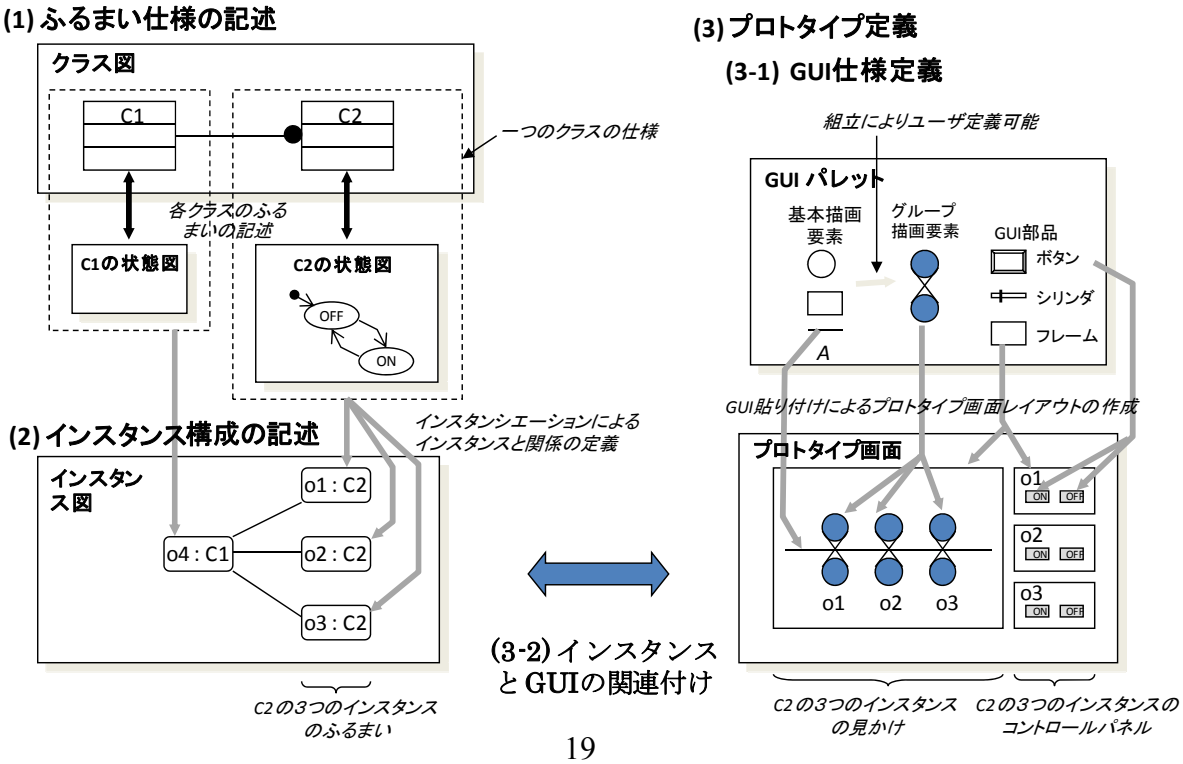


図 3-3　従来システムにおけるふるまい仕様とプロトタイプデータ，及びその作成手順

(4) **プロトタイプのビルド**：プロトタイプジェネレータを用いて，プロトタイプデータからプロトタイプのコードを生成し，その生成コードをコンパイル・リンクすることで，プロトタイプのロードモジュールを作る．

図 3-3 は，従来システムによるふるまい仕様とプロトタイプデータ，及びその作成手順（上記 (1)-(3)）を示す．

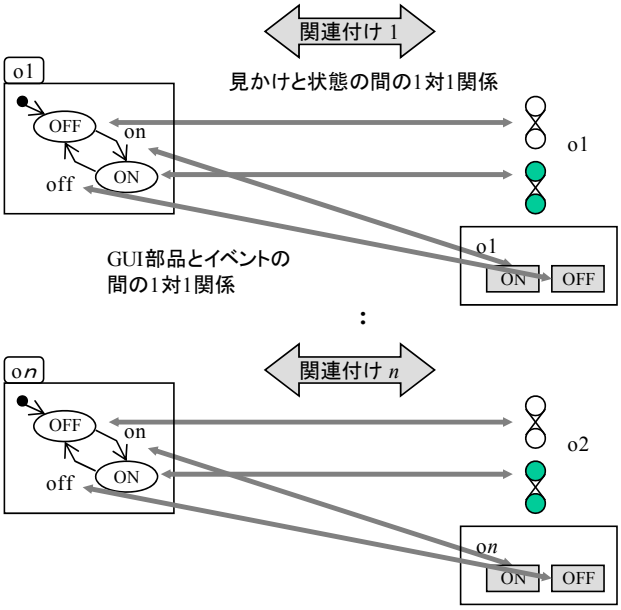


図 3-4　仕様プロトタイパにおけるインスタンスとGUI仕様の関連付け

次に，従来システムの問題点を明確にする．

[問題 1]　要求仕様（ふるまいと GUI）を部品化できない．このため，目標 1 の既存要求仕様への誘導が阻害されている．

[問題 2]　プロトタイプ構築に時間がかかる．プロトタイプ定義においてインスタンスレベルで GUI 仕様と関連づけているため操作数が多くなることと，コード生成を行うためコンパイル・リンクが必要であることによる．このため，目標 2 のオンライン作業の効率的実施が阻害されている．

[問題 3]　監視・制御オブジェクトは物理オブジェクトであり，通常のクラス図と状態図の範囲では，そのダイナミズムを記述・表現することができない．このため，目標 3 の顧客にとって監視・制御システムの動きを理解しにくくしている．

3.3　提案システムと技術課題

システム要件

監視・制御システムにおけるプロトタイピングの目標を達成するためのシステムを提案する．提案システムは，オブジェクト指向仕様に基づく仕様プロトタイパをベースとし，従来の仕様プロトタイパがもつ前述の 3 つの問題を解消する．提案システムが満足すべきシステム要件を以下に示す.

[要件 1] 要求仕様の部品化と組み立て （問題 1 と問題 2 の解決）
監視・制御オブジェクトの要求仕様（ふるまい仕様と GUI 仕様）を部品化し，それらを組み立てることでプロトタイプを定義できること.

[要件 2] プロトタイプデータのインタプリタ実行（問題 2 の解決）
コード生成及びコンパイル・リンクを経ずに，実行可能なプロトタイプを構築できるように，プロトタイプデータを，直接インタプリタ実行できること.

[要件 3] 監視・制御対象オブジェクトのダイナミズムの表現・実行（問題 3 の解決）
監視・制御システムで扱う物理オブジェクトのダイナミズムを，ふるまい仕様中に記述し，実行し，プロトタイプ上で表現できること.

提案システムの構成と解決すべき技術課題

図 3-5 に提案システムのシステム構成と，解決すべき技術課題を示す.

図 3-5 に示すように，提案システムは，クラスレベルでふるまい仕様と GUI 仕様を関連づけた**要求仕様部品**をデータベースとしてもち，要求仕様部品の作成と登録を行う**要求仕様部品エディタ**，及び登録した要求仕様部品を組み立ててプロトタイプデータを作成する**部品組立型プロトタイプ定義エディタ**をもつ. これによって, 要件 1 を満足する．また，プロトタイプデータを直接インタプリタ実行する**実行エンジン**と，プロトタイプの実行をディスプレイする**プロトタイパ**をもつ．これによって要件 2 を満足する．最後に，要件 3 に対応して，ふるまい仕様に，物理法則を表現するダイナミズムを定義できるような拡張を行い，それを実行エンジンが実行し，その結果をプロトタイパが表示できるようにする.

提案システムにおいて解決すべき技術課題は，以下の 2 つである.

[技術課題 1] 要求仕様（ふるまいと GUI）をクラス単位でパッケージ化する方式
[技術課題 2] ダイナミクスを含むふるまい仕様の直接実行方式

これら 2 つの技術課題を解決する方式の提案と評価を次節で行う.

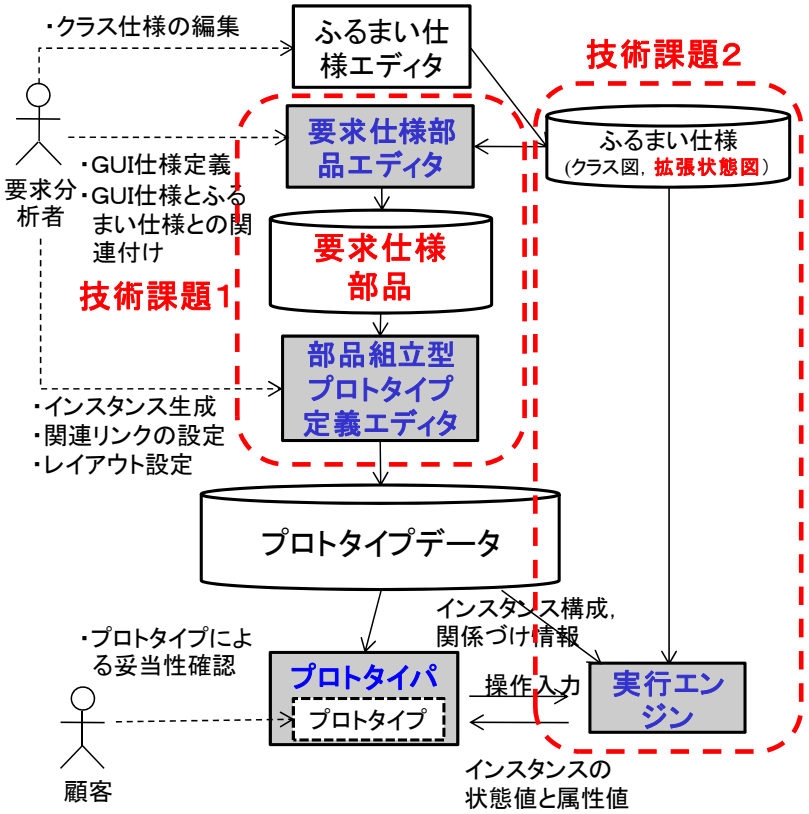


図 3-5　提案システムの構成と技術課題

3.4　提案方式とその評価

要求をクラス単位でパッケージ化する方式

(1) 従来の問題点・課題

従来方式[10][11]は，GUI 仕様との関連づけをインスタンスレベルで定義している．このため，プロトタイプ定義時における操作が多い．特に，顧客との対話を行うオンライン作業においては，煩雑な操作の数を減らす必要がある．

(2) 提案方式

提案方式では，インスタンスレベルのふるまい仕様と GUI 仕様間の関連づけを，クラスレベルで行う．図 3-6 に，要求仕様部品内でのふるまい仕様と GUI 仕様との対応関係を示す．

提案方式では，要求仕様部品定義における「関連づけ情報」を，以下の 4 種類に整理した（図 3-6 の No.7-No.10）．

① クラス⇔制御パネル
② クラス属性⇔値表示・設定用 GUI 部品
③ イベント⇔操作入力用 GUI 部品
④ 状態⇔アイコン表示

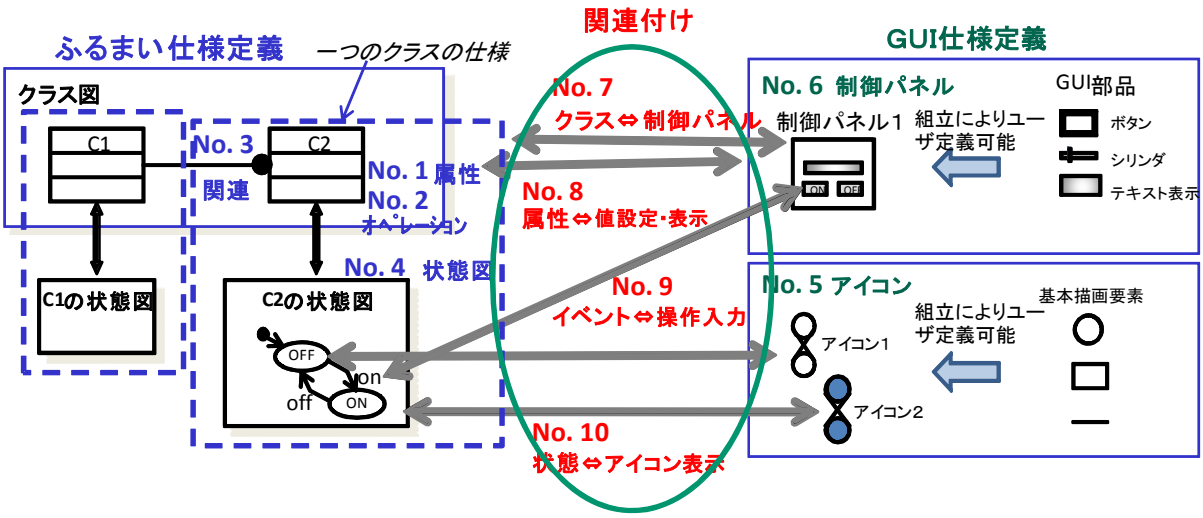


図 3-6　要求仕様部品内でのふるまい仕様とGUI仕様との対応関係

これらのクラスレベルの関連付けは，プロトタイプ定義において，インスタンスレベルの関連に自動的に展開される．図 3-7 に，プロトタイプ定義を行う際の，部品組立型プロトタイプ定義エディタ上での要求仕様部品の組み立て操作を示す．

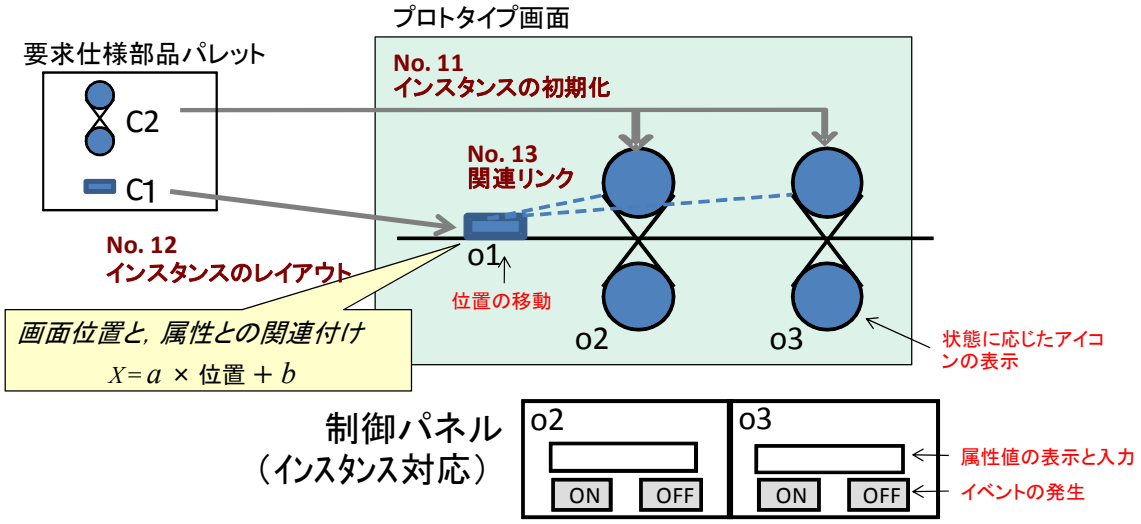


図 3-7　部品組立型プロトタイプ定義エディタ上での要求仕様部品の組み立て操作

図 3-7 に示すように，要求仕様部品をプロトタイプ画面に直接貼り付けることで，以下のことが部品組立型プロトタイプ定義エディタ上で自動的に展開される．

① プロトタイプ上へのアイコンの生成
② インスタンスの生成と初期化
③ 制御パネルの生成
④ 状態対応のアイコンと状態値との関連づけ
⑤ 制御パネル上での値表示･設定用 GUI 部品とインスタンスの属性値との関連づけ
⑥ 制御パネル上での操作入力用 GUI 部品とインスタンスへのイベント送付との関連づけ

④～⑥の関連づけが展開されることによって，プロトタイプ画面とインスタンスの動的ふるまいの間で，以下のインタラクションを行うことができるようになる．

- インスタンスの状態変化から，関係づけられたアイコンへの表示切り替え
- インスタンスの属性値変化から，値表示・設定用 GUI 部品での「値」の表示変化
- 値表示・設定用 GUI 部品での「制御パラメータ」の設定から，インスタンスの属性値変化
- 制御パネル上での操作から，インスタンスへのイベント送付

(3) 評価

従来方式[9][10]と比べて，プロトタイプ定義時の操作数が減ることを示す．

図 3-8 は，従来方式によるプロトタイプ定義時における生成操作と関連づけ操作と，各々の操作数を示している．従来方式では，インスタンス図上でインスタンスを生成し，プロトタイパ上で GUI 部品を編集・生成し，その後，インスタンスと GUI 部品との関連付けを行う．

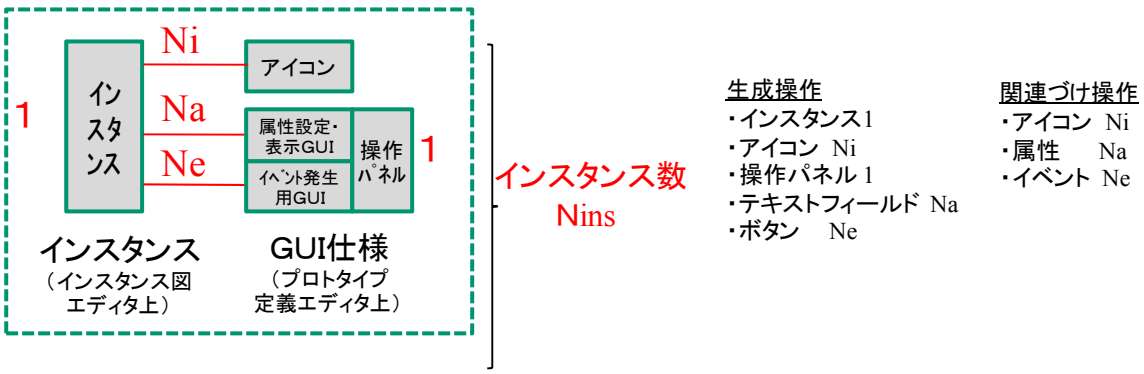


図 3-8　従来方式による生成操作と関連づけ操作

図 3-9 は，提案方式による要求仕様部品の定義時と，プロトタイプ定義時における生成操作と関連づけ操作と，各々の操作数を示している．提案方式では，要求仕様部品エ

ディタ上で, ふるまい仕様と GUI 仕様との関連付けを行う. その後. プロトタイパ上で, 要求仕様部品を直接操作により具現化する.

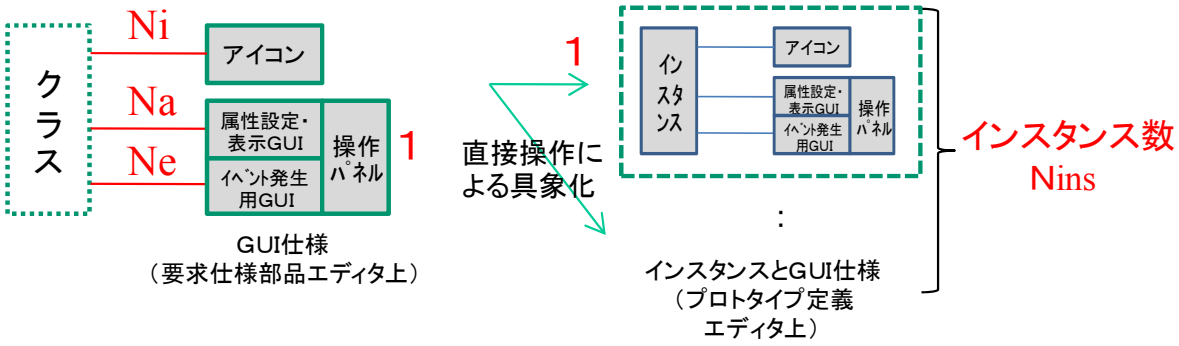


図 3-9　提案方式による生成操作と関連づけ操作

表 3-1 に，従来方式と提案方式による生成操作及び関連づけ操作に関する操作数の総和を示す.

表 3-1 従来方式と提案方式による生成操作と関連づけ操作の操作数比較

	要求仕様部品定義		プロトタイプ定義	
	生成	関連付け	生成	関連付け
従来方式			Nins（2+Ni+Na+Ne）	Nins（Ni+Na+Ne）
提案方式	1+Ni+Na+Ne	Ni+Na+Ne	Nins	

操作数は，プロトタイプ定義時（オンライン作業に相当）で Nins (1+2Ni+2Na+2Ne)の減少，要求仕様部品エディタ上の操作を含めても Nins -1+2(Nins-1)(Ni+Na+Ne)の減少，となる.

提案方式を用いることで，オンライン作業量を削減することができ，インスタンス数に比例してその削減効果も大きくなることがわかる.

ダイナミクスを含むふるまい仕様の直接実行方式

(1) 従来の問題点・課題

ふるまい記述に用いる状態図は，Statecharts[10]の記法に基づいている. Statecharts は，対象システムを，タイムアウト遷移をもつ並行な状態機械としてモデル化する．Harel ら[57]は，Statecharts の実行意味論を単位時間経過イベント(Tick)により時間経過するリアクティブシステム（Timed System）として与えている．Timed System は離散システムのための計算モデルであり，従来方式[9][10]はその実装方式を提供している.

Maler ら[58]は，自然界のダイナミクスを含むハイブリッドシステムをモデル化するために，Statecharts に対して，変数の連続的な変化を表す微分方程式を，OR 状態に記述できるように，拡張を加えた上で，その実行意味論を Phased Transition System (PTS)として与えている．一般に，連続系のダイナミクスは n 個の連続変数ベクトル $x=[x_1,x_2,...x_n]^T$ に関する連立線形微分方程式で表現されることが多い．

$$\frac{d}{dt}x = Ax + f(t) \qquad (3\text{-}1)$$

ここで，A は $n \times n$ の定数行列，$f(t)$は n 個の時間関数のベクトル$[f_1(t),f_2(t)...f_n(t)]^T$ である．

図 3-10 に，PTS の記述例を示す．Moving という状態に対して微分方程式を割り当てている．

PTS の計算は，（時刻進行がなく状態遷移の連鎖とそれに伴う離散変数の更新が起こる）離散フェーズと，（時刻進行があり微分方程式に従って連続変数の更新が起こる）連続フェーズを繰り返すものである．

提案方式は，PTS の実装方式を提供する．この際の課題は，提供する実装方式が，プロトタイピングの目的に対して，合目的性と近似性をもったものとなることである．

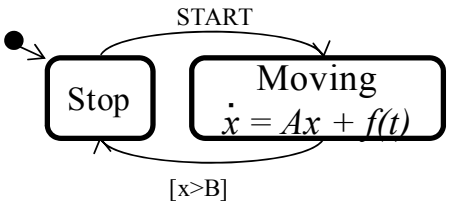


図 3-10　PTSの記述例

(2) 提案方式

提案方式は， PTS をコンピュータ上で近似的に模擬する．つまり，ふるまい仕様記述に用いる状態図において，OR 状態が変数の連続的な変化の近似を表す差分方程式をもてるように記法を拡張する．実行にあたっては，PTS の連続フェーズを，最小経過時間 ΔT 幅で離散時間サンプルして連続変数の更新を行う．

提案方式では，(3-1)式で示す微分方程式を，差分方程式として定義する．このとき，差分方程式は，微分方程式の数値計算等で用いる近似計算式，例えば ΔT が十分小さいとして(3-2)式で示される漸化式，で近似する．

$$x_{n+1} = (I + A\Delta T)x_n + f(t_n)\Delta T \qquad (3\text{-}2)$$

図 3-11 に，提案方式における PTS の近似的な記述例を示す.

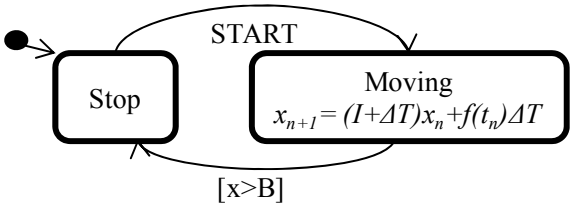


図 3-11 提案方式におけるPTSの近似的な記述例

図 3-12 に，提案方式におけるふるまい仕様の実行時における実行の制御手順を示す.

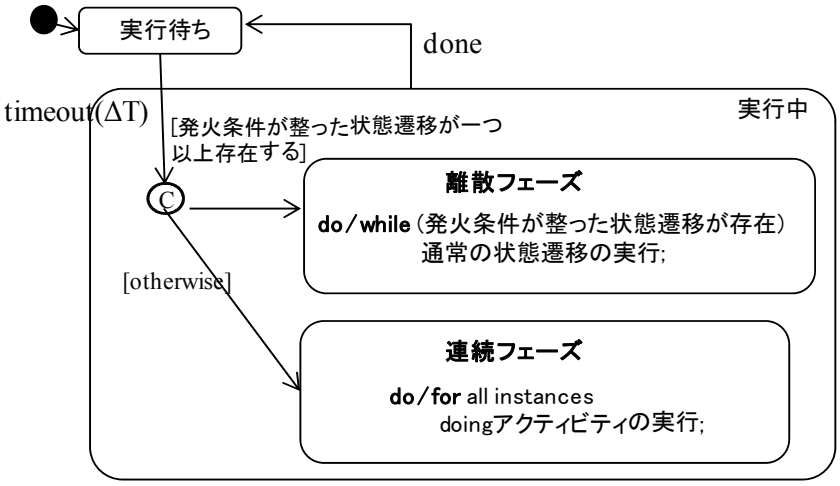


図 3-12　提案方式における実行の制御手順

図 3-12 において，プロトタイプは，初期状態として「実行待ち」状態にいる．単位経過時間ΔT が経過した時点で，発火条件を満たすトリガが存在する場合，すなわち，イベント発生，あるいは連続時間変数を含む条件式の真への変化により，発火条件が整った状態遷移が一つ以上存在する場合，離散フェーズへ移行する．離散フェーズでは，発火条件を満たすトリガが存在するインスタンスがある限り，そのインスタンスについて状態遷移に伴う計算が実行される．単位経過時間ΔT が経過した時点で，すべてのインスタンスに発火条件を満たすトリガが存在しない場合には，連続フェーズへ移行する．連続フェーズでは，現在状態に doing アクティビティが存在するすべてのインスタンスについて，そのアクティビティに記述された差分方程式を 1 ステップ実行する．連続フェーズあるいは離散フェーズの計算が終了すると，実行待ち状態へ戻る．

なお，Maler らのモデルでは，状態遷移に最小遅延時間λと最大遅延時間μを設定することで遷移の可能状態と実際の発火との関係に一定の制約を記述することができるが,

この記述方法は仕様に非決定性（どの遷移が発火するかは一意に決まらない）を生じるため，提案方式の状態図ではタイムアウト遷移（λ＝μ）のみの追加に留めている．

(3) 評価

まず，プロトタイピングの目的に対する提案方式の合目的性について評価する．

監視・制御システムのプロトタイピングでは，提案されたシステムに対する要求仕様を，顧客にとって理解し易くするという目的に対して，以下の２つのことを表現・模擬できなければならない．

- 外部オブジェクトの動きをビジュアル化して模擬
- 外部オブジェクトの属性値（制御状態値）が閾値を超えたことをトリガとした制御の開始

提案方式で，どのように，上記 2 点を満たすかを，図 3-13 のエレベータの例を使って示す．

図 3-13 において，エレベータの Moving 状態で，オブジェクトの属性「高さ」の更新が行われる．この「高さ」をプロトタイプ画面上のオブジェクトの位置と対応付けることで，オブジェクトの動きをビジュアルに表現することができる．また，エレベータ制御の状態図中に「高さ」をモニタする遷移条件を記述することで， 例えば，ある階への到達を検知したことをトリガとして制動動作を開始することを模擬できる．

すなわち，上記 2 点は，提案方式により以下のように表現可能である．

① オブジェクトの属性値をプロトタイプ画面上の位置と関連づけることで, 動きとして表現

② オブジェクトの属性値をモニタするトリガ記述により, 閾値を超えた際の制御開始イベントを表現

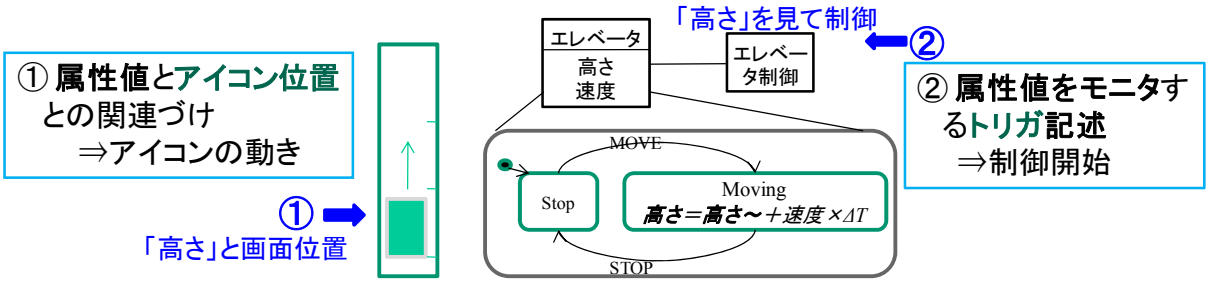


図 3-13　エレベータの記述例と要件の満足

次に，提案方式の要求仕様としての近似性について議論する．

(3-2)式の近似による誤差として，微分方程式を差分方程式で近似することによる数値計算上の誤差の累積と，状態遷移条件の発火がΔT 毎でしか評価できないことによる最大ΔT の発火タイミングの遅延がある。しかし，これらは，外部環境の経時変動の速さに比してΔT を十分に小さくすることで，その誤差範囲を抑えることが可能である．ま

た，提案方式を利用した要求の妥当性確認では，離散システムが外部環境を計測・制御する動作をビジュアル化することでプロトタイプをよりリアルに見せることが目的であり，離散システムと環境の時間制約を厳密に検証することを目的としたものではないため，これらの近似誤差は大きな問題とはならない．

このように，提案方式は，プロトタイピングの目的に対して，合目的性と近似性をもった実装方式を提供していることがわかる．

3.5　実証実験

本章の研究の目標状態の確認のために実証実験を行った．

試作システム

提案システムの試作（ROAD/EE と呼ぶ）を行った．ROAD/EE は，UNIX ワークステーション上に，X Window/Motif，C++言語を用いて実装している．そのふるまい仕様の記述モデルとしてオブジェクト指向分析設計手法 OMT(Object-Oriented Modeling Technique)[56]を採用した．

図 3-14 に試作システムの構成を示す．

ROAD/EE のふるまい仕様エディタは，クラス図エディタと状態図エディタからなり，OMT のふるまい仕様の作成を行うことができる．ROAD/EE では，図 3-5 におけるプロトタイパとプロトタイプ定義エディタを，一体化して実装している．これにより，プロトタイパ上で，プロトタイプデータを定義し，その画面上でプロトタイプ実行を行うことができる．実行エンジンは，要求仕様とプロトタイプデータを読み込み，インタプリタ実行する．ツールマネージャは，これらのコンポーネントの起動と終了の管理，及びコンポーネント間の通信の制御を行う．

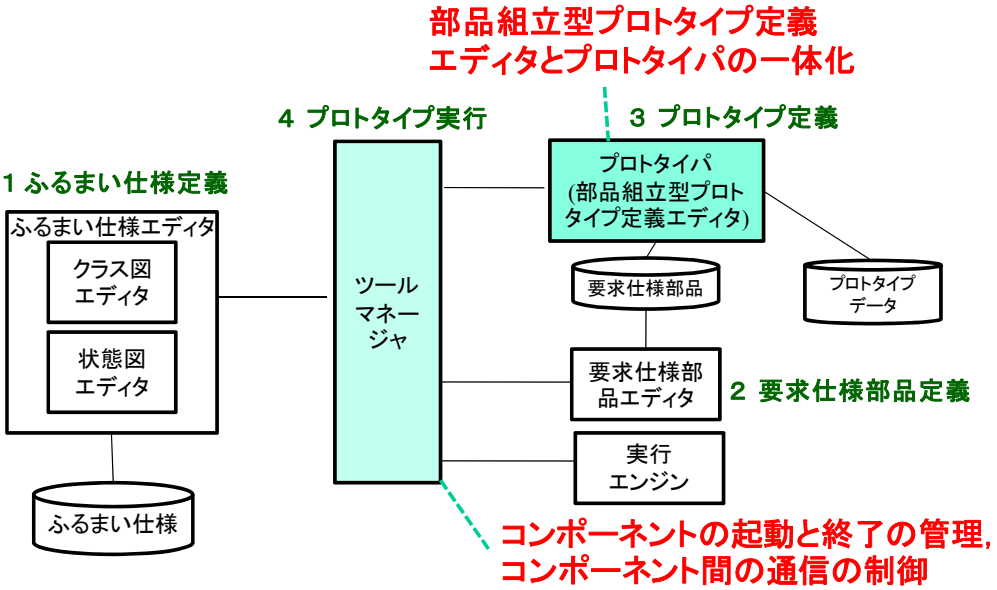


図 3-14　試作システムの構成

要求仕様の再利用によるプロトタイプ構築機能

表 3-2 に ROAD/EE を用いたプロトタイプ構築の作業項目と設定項目の概要を示す.

表 3-2　プロトタイプ構築における定義項目

作業項目		設定項目		No.
要求仕様部品定義	ふるまい仕様定義	クラス図	属性	1
			オペレーション	2
			関連	3
		状態図		4
	GUI仕様定義	アイコン		5
		制御パネル		6
	関係づけ	クラス⇔制御パネル		7
		属性⇔値設定・表示		8
		イベント⇔操作入力		9
		状態⇔アイコン表示		10
プロトタイプ定義		インスタンス	初期化	11
			レイアウト	12
		関連リンク		13

表 3-2 に示すように，ROAD/EE におけるプロトタイプ構築プロセスは，大きく要求仕様部品定義とプロトタイプ定義に分かれる．以下にそれらの詳細を記述する．

(1) 要求仕様部品定義

要求仕様部品定義は，クラス図と状態図を定義するふるまい仕様定義と，アイコンと制御パネルを定義する GUI 仕様定義，及びそれらの間の関連付けからなる．図 3-6 上の No は，表 3-2 の要求仕様部品定義における設定項目に対応している．

要求仕様部品は，ふるまい仕様，GUI 仕様，及びそれらの間の関連付けをパッケージ化し登録したものである．要求仕様部品は，クラス図内の各クラスに対して定義できるもので，次の構成要素からなる．それらは，矢印の先で示す専用エディタによって編集できる．

・ふるまい仕様定義(クラス定義と対応する状態図)　⇒ ふるまい仕様エディタ
・1 つの制御パネル　⇒ 制御パネルエディタ
・1 組のアイコン集合　⇒ アイコンエディタ
・ふるまい仕様と GUI 仕様の間の関係付け
　⇒ 制御パネルエディタ及びアイコンエディタ

ふるまい仕様定義において，クラス図とそのふるまい仕様を記述する状態図を定義する．クラスのオペレーションでは，そのインスタンス属性値への参照と代入に加え，関連によりたどることができる他クラスのインスタンス属性値への参照，オペレーションの呼び出しを記述することができる．状態図の中で，イベントに対するアクション，及びアクティビティを記述できる．アクション及びアクティビティには，クラスのオペレーションの呼び出しと，クラスの属性名への代入を記述することができる．

状態図上の状態遷移と，状態中に記述するアクティビティの構文を BNF で以下に記述する．

状態遷移の構文

state-transition := *trigger* | *trigger* '/' *actions*
trigger := *event-name* | '[' *condition* ']' |
event-name '[' *condition*']' | **timeout** '(' *literal-number* ')' | **done**
condition := *boolean-expression*
actions := *action* | *actions* ';' *action*
action := *assignment* | *event-sending* | *messaging*
assignment := *attribute-name* '=' *expression*
event- sending := *instance* '!' *event-name* // インスタンスへのイベント送付
messaging := *instance* '.' *operation-name* '(' *argument-list* ')'
// インスタンスのオペレーションの実行
argument-list := *expression* | *argument-list* ',' *expression*
instance := *association-name* |**SELF** // 関連名により関係リンク先を参照
// SELF はそのインスタンス自身を参照

アクティビティの構文

activities := *activity* | *activities* ';' *activity*
activity := *activity-type-name* '/' *operation-name*
activity-type-name := **entry** | **exit** | **do** | **doing** // entry, exit, do は UML の定義と同じ
// doing は連続変数の単位時間更新のため追加

制御パネルは，ボタン，テキストボックス，スライダなどの GUI 部品を配置したウィンドウであり，配置された各 GUI 部品は，クラスの属性及び状態図上のイベントと関連付けて定義する．実行時において，GUI 部品を操作することにより，対応するインスタンスに対して，属性値の設定とモニタ及びイベントの発生を行うことができる．制御パネルは，個々のインスタンスへの操作インタフェースを与える．

アイコンは，インスタンスの視覚的イメージを与える． 1 つのアイコンは，状態図の任意の OR 状態に対して関連付けることができる．実行時において，各インスタンスは，その現在状態に対応したアイコンとして表示される．

要求仕様部品をクラス毎に定義するのは，従来の仕様プロトタイパにおいて課題であったふるまい仕様と GUI 仕様とを関連付ける手間を減らすことと，既存の要求仕様をパ

ッケージ化する単位としてクラスを用い，要求仕様の再利用と顧客要求の誘導を図るためである．ROAD/EE の設計思想として，要求仕様部品定義の作業のほとんどはオフライン作業で行い，オンライン作業となる顧客の要求に応じたカスタマイズ作業は，プロトタイプ定義で行うことを想定している．

(2) プロトタイプ定義

プロトタイプ定義は，インスタンスの初期化とレイアウト，及び関連リンクの設定からなる．この作業は，ROAD/EE が提供するプロトタイパにより行う．

図 3-7 上の No は，表 3-2 の「プロトタイプ定義」における設定項目に対応している．

プロトタイパには，「要求仕様部品定義」であらかじめ定義されている要求仕様部品が登録されている．インスタンスの初期化では，要求仕様部品を表す「クラスアイコン」を Drug ＆ Drop 操作によりプロトタイプ画面上に直接貼り付けることで，インスタンスを生成する．その操作により，そのクラスに関連付けられている制御パネルウィンドウも同時に生成される．

プロトタイプ画面上では，アイコンを通常の描画要素と同様にレイアウトすること，及びアイコン表示位置と属性値とを関連づけることができる．制御パネル上では，インスタンスの属性値を設定することができる．アイコン表示位置（X,Y）を，属性値から計算する式を定義することで，属性値の変化をアイコン表示位置の移動として見せることができる．また，プロトタイプ画面上の直接操作により，インスタンス間に関係リンクを設定することができる．その際，定義済みのクラス間の関連のみ関係リンクを設定することが許される．

プロトタイプ定義の作業において，プロトタイプ画面の GUI 仕様とふるまい仕様との関係付けを改めて設定する必要はない．これによって，オンライン作業に対応するプロトタイプ定義の設定作業の負荷を小さくしている．

ふるまい仕様のインタプリタ実行機能

ROAD/EE では，プロトタイプを実行すると，オブジェクト指向仕様を直接解釈実行し，その実行結果をプロトタイプ画面上に表示する．

プロトタイプ定義後，プロトタイパ上で実行を開始するメニューを選択することにより，実行が開始する．プロトタイプデータは，インスタンス，その属性値，及びインスタンス間の関連リンクからなる．プロトタイプデータ内のインスタンスは，その所属クラスの状態図に従う状態機械として，並列に実行される．

制御パネルに対して操作を行うと，インスタンスの属性値の変更を含むイベントが発生する．図 3-2 に示す実行制御手順に従って，インスタンス毎にイベントの処理を行い，状態遷移が起こる．状態遷移が起きると，状態遷移元の exit アクション，状態遷移中のアクション，及び状態遷移先の entry アクションが実行される．

アクションでは，関係リンクを使った関係インスタンスの参照が可能であり，これを使って，参照先の属性値の参照，参照先へのイベント送付，及び参照先のオペレーションの呼び出し（メッセージ送付）を記述することが可能である．つまり，一つのインス

タンスへの一つのイベントからスタートした実行を，アクションとアクティビティを通じて，他のインスタンスへと連鎖させていくことができる．

こうした実行の連鎖は, シミュレーション時間としては同時に, しかし実際の計算は, 逐次的に現在状態及び属性値に副作用を及ぼしながら進む．ROAD/EE では，属性値に対する以下の 2 つの参照法を許している．

- 「属性名」:　　ステップ実行中の副作用結果を反映した現在値
- 「属性名～」:　ステップ実行前の副作用結果を反映しない値

実行順序により生じる非決定性を明示的に回避したい場合には,「属性名～」を使用することを推奨している．

プロトタイプ画面上では，実行結果を表示する．インスタンスの状態遷移により，遷移先の状態と関連付けられたアイコンへと表示が切り替わる．インスタンスの属性値に変化があると，制御パネル上の属性値の表示が更新され，その属性値が計算式によってプロトタイプ画面上のインスタンス表示位置と関連づけらている場合には，アイコン表示が計算式によって決まる画面位置へと移動する．

適用評価

ROAD/EE を監視・制御システム向けプロトタイピングシステムとして業務へ適用することにより，その有効性を評価した．

(1) 適用対象

本適用評価におけるプロトタイピングを行う対象は，鉄鋼圧延プラントの監視制御システムである．図 3-15 に鉄鋼圧延システムの概念図を示す．

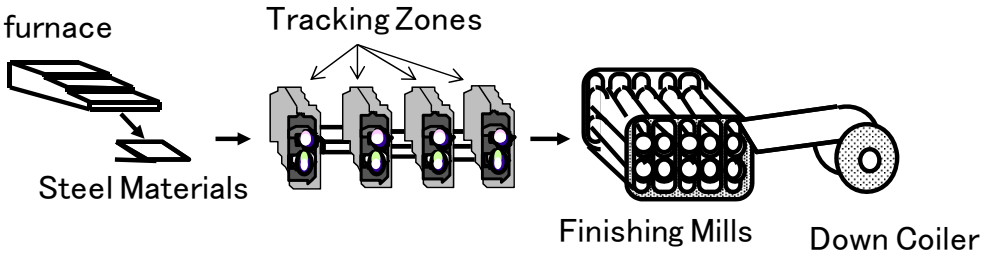


図 3-15　鉄鋼圧延プラント概念図

本システムは，鉄鋼圧延プラントにおける鋼材(Steel Materials)の位置と状態のモニタ，異常時における監視者による制御を行う．プロトタイピングを行う目的は，監視画面に対する顧客要求の抽出と妥当性確認を行うことであり，顧客に監視画面上で行う作業の流れをイメージしてもらい，要求仕様をスムーズに決定することである．この際，顧客は同種のシステムについての経験がない海外ユーザを想定しており，要求分析者にとって限られた頻度と時間の打ち合わせしか許されていない．また，要求分析者は，同種のシステム開発を数多く経験してきており，顧客の満足を得ながらなるべく既存のシステ

ムを流用しコストを抑えることを望んでいる.

(2) 評価手順

適用評価は，以下の手順に従って実施した.

① SE 部門の要求分析者(2 名)と協力して, ふるまい仕様記述及びプロトタイプを作成する.

② 要求分析者が, 想定顧客との打ち合わせの際に生じる可能性の高い要求変更を, ブレンストーミングにより抽出する.

③ プロトタイプに対して，洗い出した要求変更を採り入れる改修作業を実施する.

④ 評価項目を設定し，要求分析者が上記適用結果を評価する.

(3) 評価項目

評価項目は，監視・制御システムのプロトタイピングの目標状態の達成を評価できるように，以下のように設定した.

R1) システム記述力の十分さ [目標 3]

R1.1) システムの見かけ

R1.2) システムのふるまい(外部刺激に対するシステム応答)

R1.3) システムの外部環境及びそれによる制約

R2) 要求の反映と確認作業の短縮化 [目標 2]

R2.1) 要求の追加・修正作業が顧客にもわかりやすい簡単な操作でできること

R2.2) プロトタイプのビルド処理で待たされないこと

R3) オフライン作業とのデータ共有 [目標 1]

R4) 要求仕様の部品化と組立機能 [目標 1]

(4) プロトタイプ構築結果

プロトタイプを構築した際に作成したふるまい仕様記述のうちクラス図を図 3-16 に示す.

オブジェクト指向分析でふるまい仕様記述を行ったところ, ほぼ図 3-15 に示した装置ごとにクラスを割り当てる結果となった. その中で TrackingZone (トラッキングゾーン), FinishingMill (圧延機), 及び DownCoiler (巻取機)クラスに対しては，Zone という共通の親クラスを設けた. Zone は「鋼材がその上に存在する領域」としてモデル化し，それらに共通なオペレーションを共有させるようにした. これらのクラスはイベントに対するふるまいがそれぞれ異なるため, 状態図は個々のクラスに対して別々に定義した. また, Zone のサブクラスにおいて，表示アイコン，表示属性，及び操作が異なるので，アイコン及び制御パネルをクラスに対応して作成した.

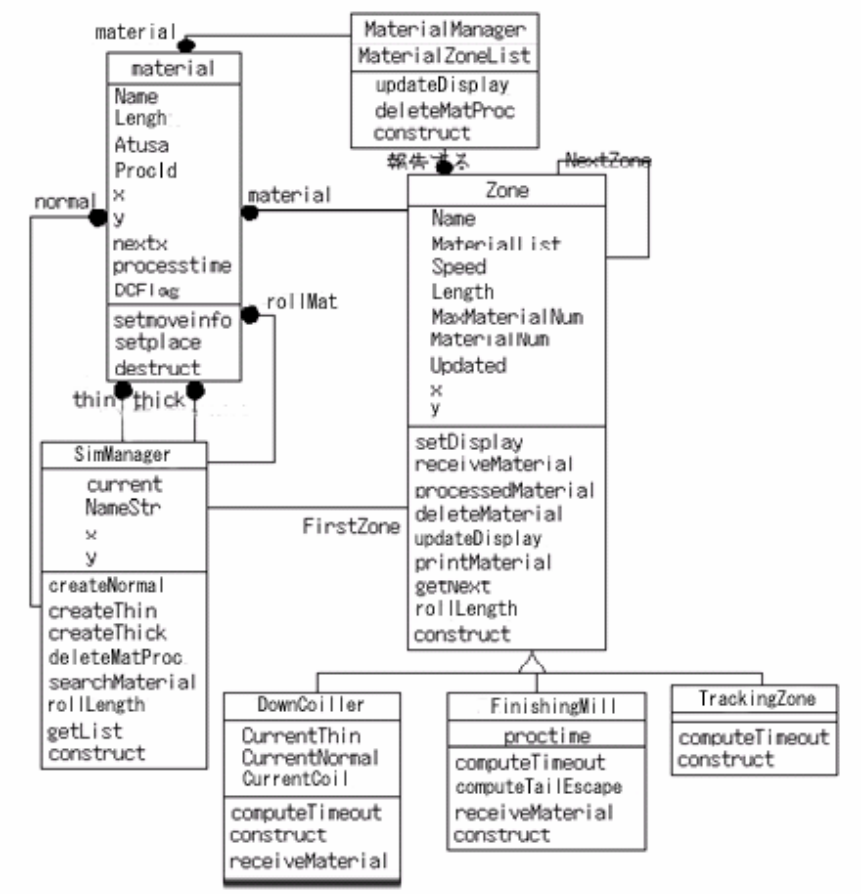


図 3-16　鉄鋼圧延プラント監視・制御システムのクラス図

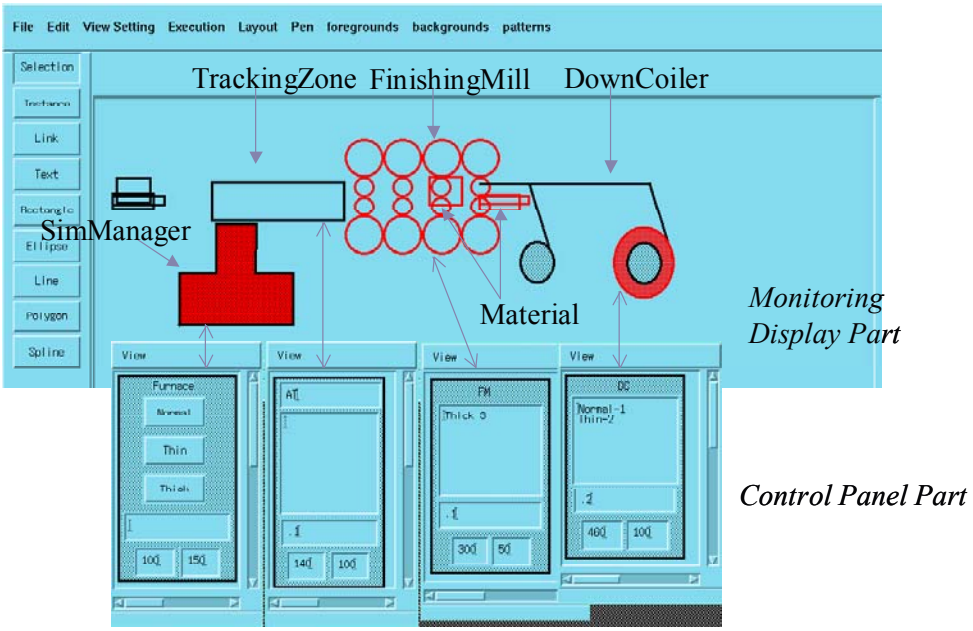


図 3-17 監視画面プロトタイプ

図 3-17 は，ROAD/EE で作成した監視システムのプロトタイプである．図 3-17 上部はセンサからの情報を基に装置の状態を表示する監視画面部分 (Monitoring Display Part) であり，下部は監視者が制御パネルを用いてプラントを制御する制御パネル部分 (Control Panel Part) である．

(5) 要求分析者による評価

要求分析者との対話により，顧客との打合せで起こりやすい主な要求変更を洗い出した．それらの要求変更を ROAD/EE 上で実装した．その際，要求変更と修正を行った設定項目との対応を表 3-3 に示す. ここで, 設定項目の欄には表 3-2 右端の No を使用した.

このプロトタイプのオンライン作業における有効性について要求分析者 2 名に評価をしてもらった. 評価は, (3)の評価項目に沿って行う. まず, システム記述力の十分さ（R1）については, ROAD/EE が提供するプロトタイプは, 単に GUI 仕様を見せるのではなく，鋼材がどのように動き，監視画面上でユーザは何を監視し，何をオペレーションするかというイメージを十分に伝えることができるという評価を得た（R1.3 の満足）.

要求の反映と確認の短縮化（R2）について，次のような評価結果となった．要求変更 a～e は，他の設定項目への修正を伴わないため，速やかな修正が可能である．f，g はアイコンやオペレーションの新規追加が必要であるが, 単一クラス内に閉じた修正のため, OMT に習熟している使用者であれば一連の手順で修正が可能である．従って，要求変更 a～g は，顧客対話時に修正可能な範囲である (R2.1 を満足する要求範囲)．これに対して，複数クラスの修正や他の設定項目への新規追加が必要な h～j については，クラス図の再設計になるため，大きな作業量とともに要求分析者に特殊な訓練が必要となるという評価を受けた（R2.1 を満足できない要求範囲）．また，プロトタイプ再構築に要する時間（R2.2）については，要求分析者に特に比較の対象がなく評価を受けなかった.

オンライン作業とのデータ共有（R3）については，オブジェクトモデルの各クラスの属性は最終製品における監視すべき状態を表し，データモデル作成上有用であるとの評価を受けた．しかし，現場にはオブジェクト指向の考え方と OMT が普及していないため，要求仕様書としてそのまま使える現状にはないという意見もあった.

要求仕様の部品化と組立機能（R4）については，要求変更 a～c の範囲が顧客別のプラント構成の違いに相当し，定義済みの仕様部品を用いることで，プロトタイプ定義のみの簡単な設定のみで対処できる．このため，顧客のプラント構成に対応した監視画面イメージを既存仕様の組合せで構成でき，そこからの要求誘導（要求変更 d～j に対応）に便利であるとの評価を受けた.

表 3-4 は，表 3-1 の計算方法を用いて，図 3-17 監視画面プロトタイプにおける従来方式と提案方式でのプロトタイパ上の操作数の差異を求めた結果である．従来方式は，操作数 96 回を必要とするのに対し，提案方式は生成操作数 9 回で済み，操作数 87 回の削減となっている.

この適用評価によって，(R1) システム記述力の十分さ，(R2) 要求の反映と確認作業の短縮化，(R3) オフライン作業とのデータ共有，(R4) 仕様の部品化と組立機能を満たすことを確認し，この方式が目標状態の達成に有効性であることを示した.

表 3-3　要求変更に対して必要な設定項目

	要求変更	ROAD/EE修正の概要	修正する設定項目			
			ふるまい仕様定義	ＧＵＩ仕様定義	関係付け	プロトタイプ定義
a	圧延機の鋼材許容枚数の増減	初期値の修正				11
b	装置の順番入れ替え	レイアウト、関連リンクの追加				12, 13
c	圧延機の追加	インスタンス、関連リンクの追加				11, 12, 13
d	鋼材番号の表示	制御パネルで既存の属性を表示		6	8	
e	異常時に圧延機アイコンを警告表示する	アイコンを既存の状態に追加		5	10	
f	巻取機が鋼材を巻き取るアニメーションの追加	状態、アイコンの追加	2, 4	5	10	
g	鋼材許容枚数超過時の、圧延機の対応処理追加		2, 4	5	10	
h	圧延機が鋼材を圧延するアニメーションの追加	クラス、状態の追加及び，既存クラスへの修正	1, 2, 3, 4	5	10	11
i	異常鋼材の取り除き操作の追加		1, 2, 3, 4	6	9	11, 13
j	鋼材の移動アニメーションの追加		1, 2, 3, 4	5, 6	8, 9, 10	11, 12, 13

表 3-4 従来方式と提案方式でのプロトタイパ上の操作数比較

項目			SimManager	Material	TrackingZone	FinishingMill	DownCoiler	計
GUI構成部品数		アイコン Ni	2	2	2	2	4	12
		表示用属性 Na	3	0	5	4	4	16
		操作ボタン Ne	3	0	0	0	0	3
インスタンス数		Nins	1	5	1	1	1	9
操作数	従来方式	生成　A Nins(2+Ni+Na+Ne)	10	20	9	8	10	57
		関連付け　B Nins(Ni+Na+Ne)	8	10	7	6	8	39
	提案方式	生成　A' Nins	1	5	1	1	1	9
操作数の削減数		A+B−A'	17	25	15	13	17	87

(6) 要求変更に対する作業規模に関する評価

表 3-3 に示した ROAD/EE 上での要求変更は，作業規模の面から次の 4 つに分類できる．そのそれぞれについて，従来の仕様プロトタイパを使用した場合との比較を行う．

① インスタンス構成の変更

表 3-3 における a～c は，圧延プラントの構成など運用環境の変更に相当する．この変更を従来の仕様プロトタイパで行うと，インスタンス図(インスタンスの構成を定義するモデル図)と，GUI 仕様を別々に書き直し，GUI 仕様とインスタンスの関連付けを行った後，コード生成とビルドを行う作業を経なければならない．一方，ROAD/EE では，モデルを直接解釈実行する方式のため，プロトタイパ上でのインスタンス生成とリンク生成操作（プロトタイプ定義の設定）のみでプロトタイプの再構成が完了し，短時間で変更を反映することができる．

この分類の変更に対しては，**プロトタイプデータのインタプリタ実行（システム要件 2）**による作業の軽減効果が現われている．

② 個別クラス仕様の変更

表 3-3 の d, e, f, g はクラス内で閉じた変更である．d, e は既に定義済みの属性あるいは状態を画面表示に反映する変更であり，f, g は新しい状態を追加し画面表示に反映する変更である．これらの変更を行う場合，従来の仕様プロトタイパでは GUI 仕様を含む要求仕様部品の登録はできないため，インスタンス毎に GUI 仕様とふるまい仕様との関連付けを行う必要がある．これに対して，ROAD/EE ではクラス定義で GUI 仕様との関連付けを行うことができるので, インスタンス毎の関連付け操作を省略できる．

この分類の変更に対しては，**要求仕様の部品化と組み立て（システム要件 1）**による作業軽減効果が現われている．

③ システム仕様全体に渡る変更

表 3-3 の h, i は，システム全体に渡る要求仕様の見直しとなる新規機能の追加である．変更は複数のクラスに渡り，属性とオペレーションの追加と変更及び状態図の大幅な変更が必要である. 変更の大きさは従来の仕様プロトタイパにおいても同じである．どちらの場合でも，オンライン作業のターンアラウンド時間の許容範囲を越えていると考える．

④ 環境のダイナミクスの追加

表 3-2 の j は，鋼材を画面上にアイコンとして表示し，時間の経過とともに，それが Zone 上を移動するという動作表示の追加である．変更以前は，鋼材処理中の Zone の色が変化する要求仕様であった．この変更は，ROAD/EE 上では，鋼材の「移動中」状態に対して，鋼材の位置属性 x のダイナミクスを表現する差分方程式 :

$$x = x\sim + v\,\Delta T \qquad (3\text{-}3)$$

を定義し, プロトタイプ画面上の座標と鋼材の位置属性と結び付けることで実現できる．ここで，v は速度を表す鋼材の属性である．

従来の仕様プロトタイパでは, 採用している状態モデルが離散的な仕様しか表現できないため，連続的な材の動きと材が各 Zone を通過するタイミングを視覚的に表現

することはできない．こうしたプラント監視システムでは，監視対象となるオブジェクトが連続時間系のダイナミクスをもっているケースが多いので，連続的な動作の記述と実行は顧客のシステム理解のために重要である．ROAD/EE では，外部環境のダイナミクスを差分方程式で，外部環境の変化を監視するシステム仕様をオブジェクトの属性値に関する状態遷移部の条件式で表現することで，プロトタイプ実行に反映することができる．

この分類の変更に対しては，**監視・制御対象オブジェクトのダイナミズムの実行・表現機能（システム要件 3）**によりはじめて対処可能となった．

(7) UML との関係

ROAD/EE がふるまい仕様記述モデルとして用いている OMT は，UML の母胎となった仕様記述モデルである．ROAD/EE がプロトタイピングで使用するクラス図及び状態図の範囲では，OMT 仕様は UML 仕様に意味を変えずに変換可能である．ROAD/EE で拡張したダイナミクスの記述を，UML における状態図に対して加えることも可能である．従って，ROAD/EE の要求モデルとして UML を採用した場合でもその実装は容易である．

(8) 実証実験により明確となった課題

ROAD/EE を用いてオンライン作業をさらに効率よく支援するためには，次の課題を解決していく必要であると考えている．

- プロトタイプのカタログ化

  プロトタイプデータをカタログ化し蓄積・検索できるようにしておくことにより，良く似たシステム構成を素早く検索し，そこからオンライン作業を開始することができる．

- ふるまい仕様と GUI 仕様の関連付けの柔軟化

  ROAD/EE は，現在 1 クラスに 1 組の GUI 仕様群（アイコンセットと制御パネル）しか関連付けることを許していない．しかし，プロトタイプ画面上に置かれたインスタンスに対して，GUI 仕様が同じでクラスを変えたい場合や，逆にクラスが同じで GUI 仕様だけ変えたい場合が多い．その場合，その関連付けをプロトタイプ画面上で容易に切り替えることができれば，さらに多くの要求変更への素早い対処が可能になる．

3.6 本章のまとめ

適用範囲

提案システムによるオンライン作業支援は，比較的大規模な監視・制御システムの要求分析を対象とし，プロトタイプを構築する以前に必要な要求仕様部品がある程度準備されていることを前提としている．

開発システムの規模が小さいと，オンライン作業にプロトタイピングを使うこと自体の意味が小さい．また，**情報通信システム**では，画面そのものが単純で，画面切り換え以外に動きを持たない場合が多く，ふるまい仕様に，状態図を使うほど，複雑でリアルタイム性をもつ仕様を表現する必要が少ないことから，仕様プロトタイパより GUI 指向言語システムによるプロトタイピングの方がより適切であろう．

新規システムの初期打ち合わせ作業を想定した場合，プロトタイプ定義に必要な要求仕様部品が予め存在していることは仮定できない．この場合，提案システムを用いたプロトタイプ構築は，要求仕様部品を作るところから開始しなければならず，オンライン作業の時間制約上不適切である．新規システムの初期打ち合わせでは，要求会議[59]やシナリオ分析[14] など他の要求抽出の技法を併用する必要がある．特に，UML[15]は，ユースケースあるいはシーケンス図を用いて，顧客がシステムを使用するシナリオを時間順に記述するもので，顧客にとって直観的である．シーケンス図は，それを入力として状態図を導く手順[60]が報告されているので，提案システムのモデル記述の初期入力として併用することも可能である．

提案システムのふるまい仕様記述は，機能的な仕様を表現し，実行させることによって，機能と使用性の妥当性を確認することにその用途が限定される．効率性や信頼性などの他の品質要求については，別の手段により，要求の抽出と妥当性確認を行う必要がある．

まとめ

本章では，監視・制御システムの要求分析を効果的に支援するオブジェクト指向仕様に基づく仕様プロタイパの提案と適用評価を行った．

まず，従来の仕様プロトタイパが，監視・制御オブジェクトの要求仕様（ふるまいと GUI）をばらばらに作るため部品化できないこと，プロトタイプ構築に時間がかかること，オブジェクトのダイナミズムを表現できないこと，の 3 つの面で問題があることを示した．

次に，従来の仕様プロトタイパの弱点をカバーする提案システムを提案し，2 つの技術課題，すなわち要求仕様をクラス単位でパッケージ化する方式及びふるまい仕様の直接実行方式，について，提案及び評価を行った．

最後に，目標状態の達成の確認を行う目的で実証実験を行った．提案システムを実装する試作システムを開発し，鉄鋼圧延プラントの監視画面プロトタイピングに適用する実験と評価を行った．その結果，提案システムが監視・制御システム向けプロトタイピングシステムとしての有効性であること示した．

第4章　　状態遷移仕様に基づくテスト自動化システム

4.1　本章の研究の目的と位置づけ

本章の研究目的は，小規模な組込みシステムを対象製品分野として，製品検査プロセスを効率化するための，状態遷移仕様に基づくテスト自動化システムの提案と適用評価である．

組込みソフトウェアは，マイコン内に組込まれ，入力機器からの信号を出力機器への信号へ変換する役割をもつ．小規模な組込みソフトウェア[73]は，デジタル信号処理を行うものを除きそのほとんどが表示系・操作系を含む機器制御用であり，要求仕様記述に占める**反応的な要求仕様**の割合は 50%以上である．反応的な要求仕様とは，入力と内部状態から出力を決定する仕様を指す．状態遷移仕様は，反応的な要求仕様を記述できる．

組込みシステムは，周辺機器とのインタフェースをとるため，機器依存の信号入出力を扱わなければならない．従って，製品確認プロセスでは**実機テスト**が欠かせない．そして，小規模な組込みシステムの開発では，機器依存の部分を局所化する目的で，プログラム構造として, 図 4-1 に示すような入力-処理-出力の 3 層のモジュール構造で実装されている場合が多い[73]．3 層のモジュール構造とは，マイコンポート上の入力信号を処理し入力データをメモリ上に値として書き込む部分(入力デバイスドライバ)，入力メモリ上の変化を読み，要求仕様に基づき出力データを作成し，それをメモリ上の値として書き込む部分（ロジック処理)，及び出力データをメモリ上から読みマイコンポートへの制御出力信号として書き込む部分(出力デバイスドライバ)の 3 つの部分に分かれたモジュール構造を指す．

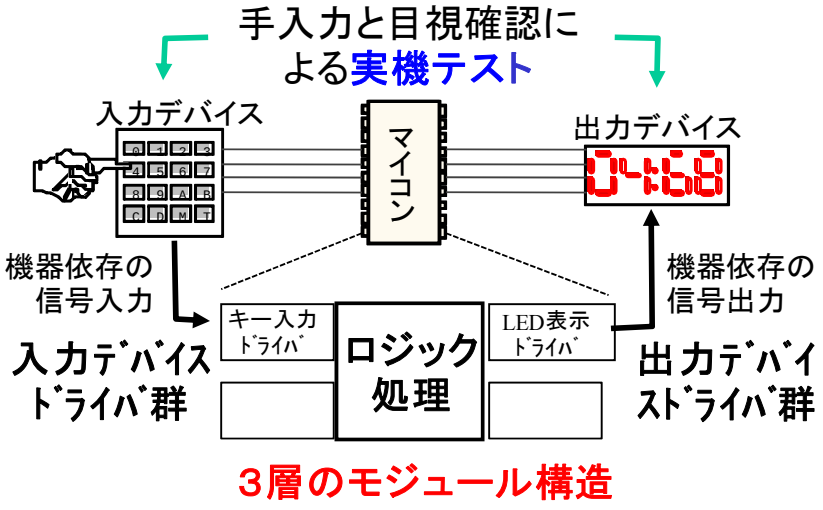


図 4-1　小規模組込みソフトウェアの実機テストと 3 層のモジュール構造

小規模な組込みシステム開発では，実機上での動作の確認ができる製品確認プロセスになって初めて，顧客が製品としてのふるまいを確認することができるようになるため，製品の機能と品質をエンドユーザの嗜好に合わせる調整や，他社製品との比較から必要と判断した機能の追加等を目的に，要求変更依頼を受ける場合が多い．要求変更依頼に対応すれば，製品検査プロセスにおいて，製品の要求仕様への適合を確認するために再テストを行う必要が生じる．製品の高機能化によりプログラム規模の増大化が進み，ビジネス上の要求から開発期間の短縮化圧力が強まる中，この再テストにかかる時間とコストの削減は，システム開発上の大きな課題となっている．

本章の研究は，こうした課題を解決することを狙っており，その目標状態は，**製品検査プロセスにおける実機上のテスト及び再テストを，検査の網羅性を確保しつつ効率的に実施できること**においている．

4.2　従来システムと問題点

テスト自動化技術の現状

図 4-2 は，製品検査において，要求仕様に基づきテストを実行するまでの作業フローである．

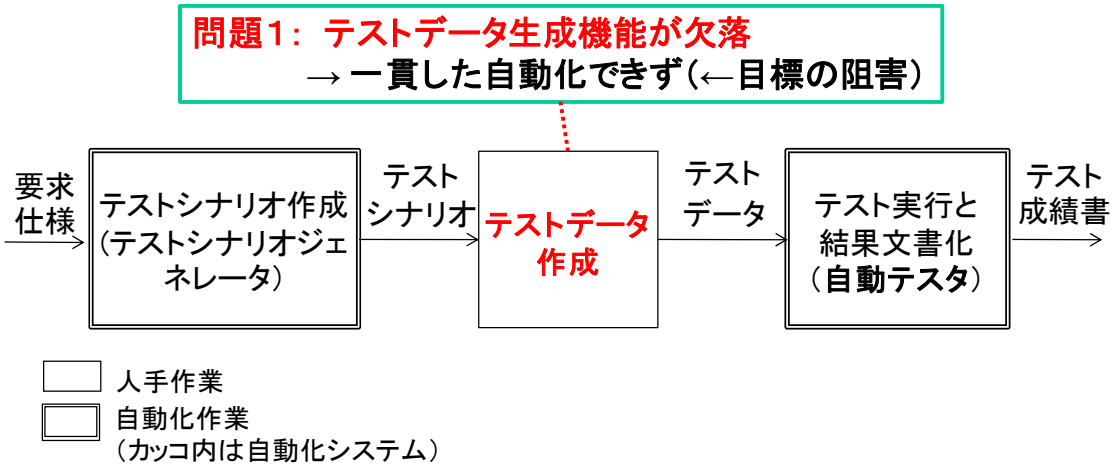


図 4-2 要求仕様からテスト実行までの作業フローと（従来）自動化が可能な部分

最初に，**テストシナリオ作成**では，要求仕様に基づき，テスト項目を洗い出し，テストシナリオを作成する．テストシナリオは，被テストプログラムに与えるべき入力及び照合の系列である．次に，**テストデータの作成**では，具体的な入力及び照合のやり方を決定するために，テストシナリオからテストデータを作成する．ここで，テストデータとは，テストシナリオの入力及び照合を，具体的に被テストプログラムに入力し，その

出力と期待値とを照合できる物理的なデータ形式もしくは自動テスタへ入力及び照合を指示するコマンド列である．最後に，**テスト実行と結果文書化**では，テストデータを入力し，被テストプログラムを実行し，テスト結果を評価し，テスト成績書を作成する．

図 4-2 において，二重線の箱は，従来技術により自動化が可能な部分を表している．テストシナリオ作成作業に対しては，有限状態モデルで記述された要求仕様（以下，状態遷移仕様と呼ぶ）からテストシナリオを自動生成する方式が提案されている．代表的なものとして，状態遷移仕様内の状態遷移を網羅するトランジションツアー法[61]，状態判定系列を生成するアルゴリズム[62][63][64][65][66]，有効グラフの簡略化アルゴリズムを用いて状態遷移の全組合せパスを網羅する手順[27]などがあり，それらの実装も報告されている[67][68][69]．テスト実行と結果文書化作業については，テスト実行の自動化を一部実現する自動テスタが実用化されている [70][71]．これらの実用化された自動テスタは，マイコン（とそれに載ったソフトウェア）を完全なブラックボックスと考え，マイコンポートに現れる入出力信号レベルで，テスト実行を自動化する方式をとっている．

しかし，従来技術では，テストデータ作成作業部分の自動化が欠落しており，要求仕様からテスト実行までの作業を一貫して支援することができない．このため，自動テスタによりテスト実行するためには，テストシナリオから手作業でテストデータを設定するしか手段がなく，この設定に要する作業コストが，システム全体の効率性を著しく低下させている．テストデータの自動生成は，要求変更から再テストまでのターンアラウンド時間を短縮する上で不可欠な機能と言える．

従来の自動テスタと問題点

組込みシステムにおいて，テストデータの自動生成が困難な理由を，テストデータの手動設定方式をとる自動テスタ ADO[70]を例にとり考察する．

ADO は，マイコンを完全なブラックボックスと考え，マイコンポートに現れる入出力信号レベルで，テスト実行を自動化するシステムである．図 4-3 に ADO のシステム構成を示す．

図 4-3 で，テスト環境は，マイコンを含む実機環境，信号設定・照合装置，及び PC から構成される．テスト者は，PC からマニュアル操作で，入力値とそれに対する期待値のペアの系列を設定する．信号設定・照合装置は，マイコンの入出力ポートに直接結線されており，設定された系列に従って，入力値から入力ポートへ擬似入力信号を発生し，出力ポート上に現れる出力信号を期待値と照合する．その結果を PC で加工しテスト成績書を出力する．ADO は，実時間性を保った完全な実機環境下でのテストが可能であり，マイコンポート上の入出力信号の仕様に変更が無い場合，プログラムの構造が変化してもそのままの設定で再テストできる利点をもつ．

ADO の問題点は，以下の 2 つである．

[問題 1] テストデータを自動生成できる適切なテスト界面を設定できない．
[問題 2] 自動テスタに専用ハードウェアを使っているため，移植性・保守性が悪い．

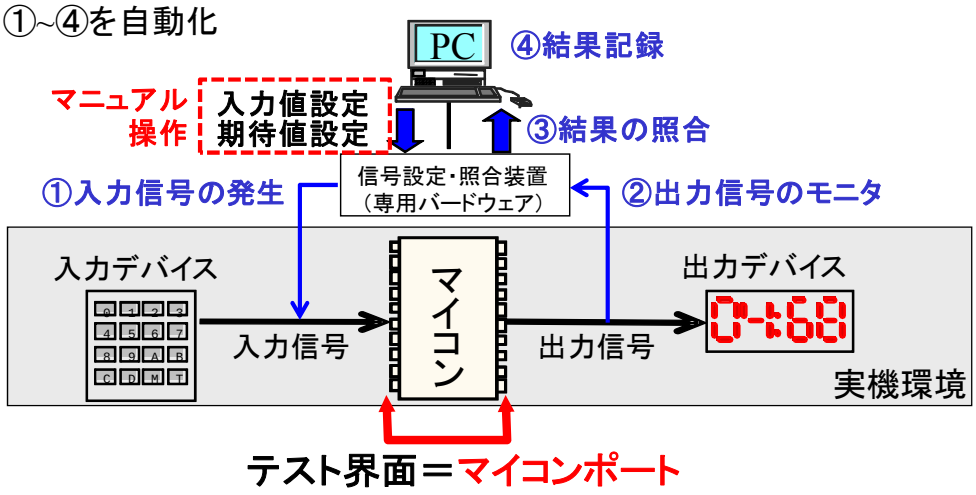


図 4-3　ADOのシステム構成

一般に自動テスタは，プログラムへの(擬似)入力を行い，その入力に対するプログラムの出力の参照と，期待値との照合を繰り返すことによって，テスト実行を自動化する．このとき，自動テスタが，プログラムとやり取りする物理インタフェースをプログラムの**テスト界面**と呼ぶ．

ADO は，テスト界面としてマイコンポートを用いている．マイコンポートを用いることの欠点は，要求仕様レベルの入力と期待値に比べて，マイコンポートにおける入出力信号が低レベルであるため，テストデータの作成と設定が困難なことと，テストできる要求仕様の範囲が制限されることである．

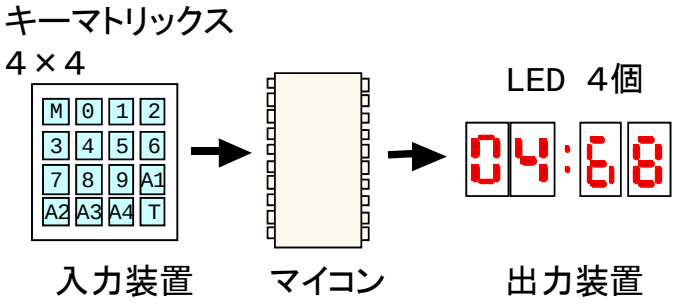


図 4-4 要求仕様例の入出力装置

例えば，図 4-4 に示すような時刻を表示する 4 個の LED(Light Emitting Diode)に対して，キーマトリックスを操作して，時刻設定をする機能を想定する．この機能に対して，「KEY0～9 を押したら 1 個目の LED にその数字が表示される」という要求仕様を考える．この場合，人手によるテストでは，「数字キー1 を押す」という操作に対して，「1

個目のLEDに1が表示されている」というシステム応答を目視確認すればよい. しかし, 通常組込みソフトウェアでは，次のような入出力仕様が加わる.

(1)キーの入力判定を行うために 20ms 毎の入力検査を行う．チャタリング防止のため 2 回連続して同一キーを確認してはじめてキーインと見なす.

(2) LED の残像効果を利用するために，20ms 毎に 4 つの LED を巡回的に点灯する制御信号を出力する.

この場合，マイコンポート上に現れる入出力信号は，テストすべき要求仕様と入出力仕様が絡み合ったものとなるため，マイコンポート上でのテストデータ（入力と期待値）の設定は実時間仕様を含む複雑なものとなる．特に(2)に対する照合では，プログラムが出力を開始する時点と，出力のモニタ時間幅を一意には決定することができないため，照合データの設定が不可能である．このように，要求仕様をマイコンポート上の入出力信号へ自動変換することは，一般には簡単ではない.

ADO におけるテストデータ作成の困難さは, テスト界面をマイコンポートに設定したことに起因する．要求仕様からテスト実行までを自動化するためには，異なるテスト界面の設定が必要である.

4.3 提案システムと技術課題

システム要件

小規模な組込みシステムにおけるテスト自動化の課題を解決するシステムを提案する. 提案システムは．要求仕様として状態遷移仕様を用い，要求仕様に基づきテストケース及びテストデータを自動生成する機能を実現することにより, 実機テストを自動化する. 提案システムが満足すべきシステム要件を，以下に示す．これらは，ADO の問題点を解決するものである.

[要件 1] 状態遷移表仕様に基づくテスト自動化機能を提供すること
[要件 2] テスト界面への入出力に専用ハードウェアを使わないこと

提案システムの構成と解決すべき技術課題

図 4-5 に，提案するテスト自動化システムのシステム構成を示す.
提案システムは，上記 2 つの要件を満たすべく以下のように構築する.

- 被テストプログラムのデバイスドライバのメモリインタフェースをテスト界面とし, 状態遷移仕様のテストシナリオを，変換テーブル（ブレーク定義）を用いて，テストデータに変換する.
- 実機とのやり取りに**ICE®**[1]を利用する.

---

1 ICE: In Circuit Emulator,マイコン基板を開発する際に使うデバッガ。ソフトウェアのデバッグとハードウェアの動作確認を行なうことが可能。CPU のソケットに接続用コネクタを挿入し、マイコン機

提案システムを実現する上での技術課題は以下である．

[技術課題] テスト自動化方式を実現するためのテスト界面の設定とデータ変換方式

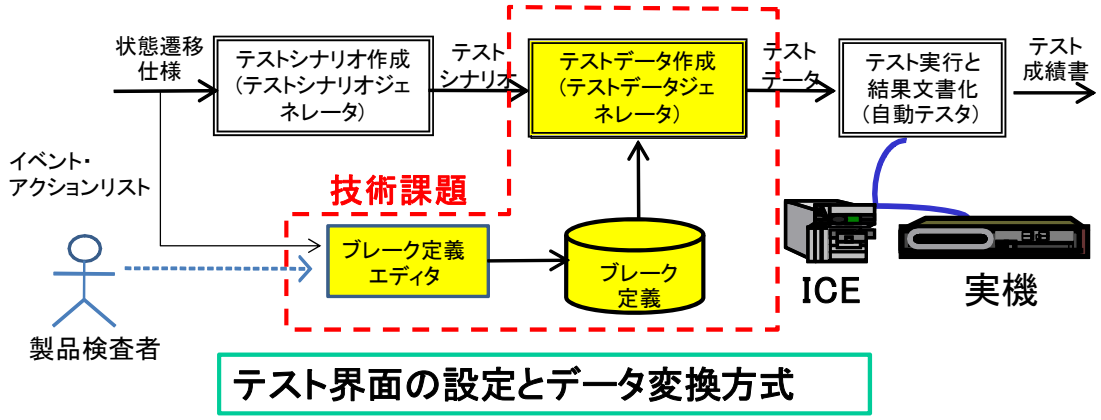


図 4-5　提案システムの構成と技術課題

4.4　提案方式とその評価

テスト界面の設定とデータ変換方式

(1) 従来の問題点・課題

従来方式[70]は，テスト界面として，マイコンポートを用いている．先に述べたように，従来方式の問題点は，マイコンポート上の信号レベルの仕様は，実時間仕様を含むため，要求仕様とテストデータの間の対応関係を取るのが困難なことである．従って，テスト界面は，要求仕様との対応の取り易さと，広範囲の被テストプログラムに対するテスト可能性をもたなければならない．また，要求変更に伴う再テストを容易にするためには，要求変更に伴う再テスト用の設定作業量が少ないことが必要となる．

(2) 提案方式

本研究で提案する方式では，テスト界面として，デバイスドライバのメモリインタフェースを用いる．そして，ICE は，このテスト界面における模擬入力と，結果出力の参照を行う．ICE は，周辺機器を接続した実機環境においてマイコンをエミュレートするデバッグ環境であり，マイコン上で動作するプログラムのメモリ値を参照・設定できる機能を持っている．

図 4-5 のブレーク定義には，図 4-6 に示すように，入力イベント及び出力アクションを，ブレーク時及びコマンドの組合せに対応付けるデータを定義する．入力イベントに対する対応付けを**入力ブレーク**，出力アクションに対する対応付けを**照合ブレーク**と呼ぶ．ここで，**ブレーク時**とは，入力及び照合を行うタイミングの記述であり，被テスト

能をエミュレートすることでデバックを行う。

プログラムの指定アドレスへの到達時，モジュールの入り口と出口，指定変数の変更時などを指定することができる．**コマンド**とは，入力及び照合方法の記述である．入力と照合の対象として変数値（メモリ上の値）を選ぶことができる．コマンドには 2 種類あり，変数値の設定を行うコマンドを**入力コマンド**，変数値を参照し期待値と照合するコマンドを**照合コマンド**と呼ぶ．

テストデータジェネレータは，入力ブレーク及び照合ブレークの定義を用いて，イベント及びアクションの系列(状態遷移仕様から生成されたテストシナリオ)を，被テストプログラム上での擬似入力及び期待値照合の系列(ICE 制御コマンド列)に自動的に変換する．この変換によって，ICE を用いたテスト実行の自動化が可能になる．

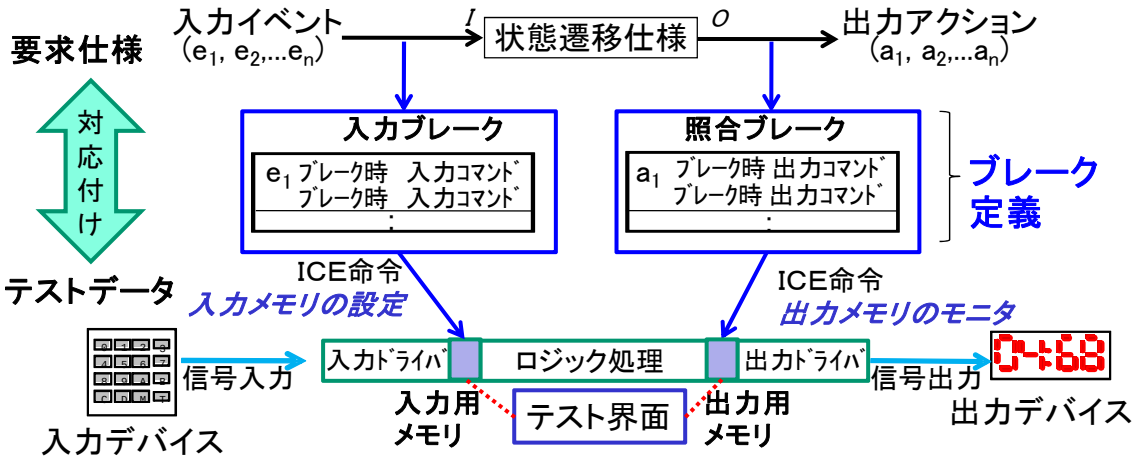


図 4-6 状態遷移仕様からのテストデータの生成

(3) 評価

提案方式に関して，要求要求仕様との対応の取り易さ，被テストプログラムの網羅度，及び要求変更に伴う再テスト用の設定作業量の 3 点について評価を行った．

(3-1) 要求仕様との対応の取り易さの評価

要求仕様との対応の取り易さについて，以下の 2 点から評価する．

① プログラム調査及びヒアリング調査[72]

提案システムは，小規模な組込み製品分野を対象とする．事前に行ったプログラム調査及びヒアリング調査の結果から，この製品分野の被テストプログラムは，プログラムが 3 層のモジュール構造をとるケースが多いことがわかっている．

② ソフトウェア工学上の知見

実績のある設計手法は，3 層のモジュール分解法を採用している．GUI プログラムの設計指針である Model-View-Controller モデル[96]，複合／構造化設計法[97]がそれにあたる．複合／構造化分析法における Source-Transform-Sink 分解は，図 4-7 に示すように，「3

層の継ぎ目において，入出力データの最大抽象化点が現れるように設計する.」ことを薦めている.

①と②から， 対象被テストプログラムについて，3 層のモジュール構造をとり，その最大抽象化点であるモジュール境界上で，要求仕様レベル記述と対応が取りやすいことが推論できる.

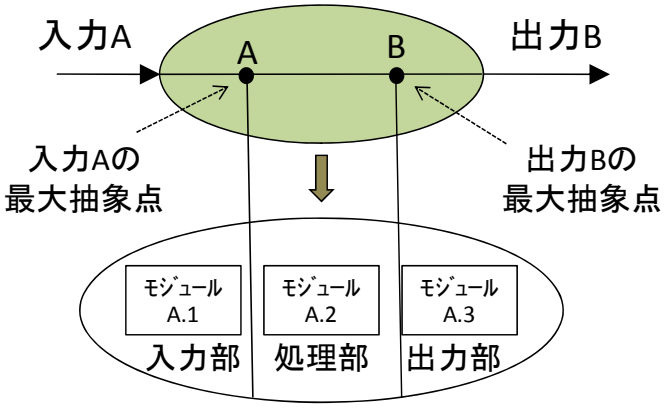


図 4-7　Source-Transform-Sink分解における最大抽象化点

例を使って，状態遷移仕様と被テストプログラムとの対応関係を説明する．図 4-8 は図 4-4 で示した例における状態遷移表（の一部）と被テストプログラムの実装の例である.

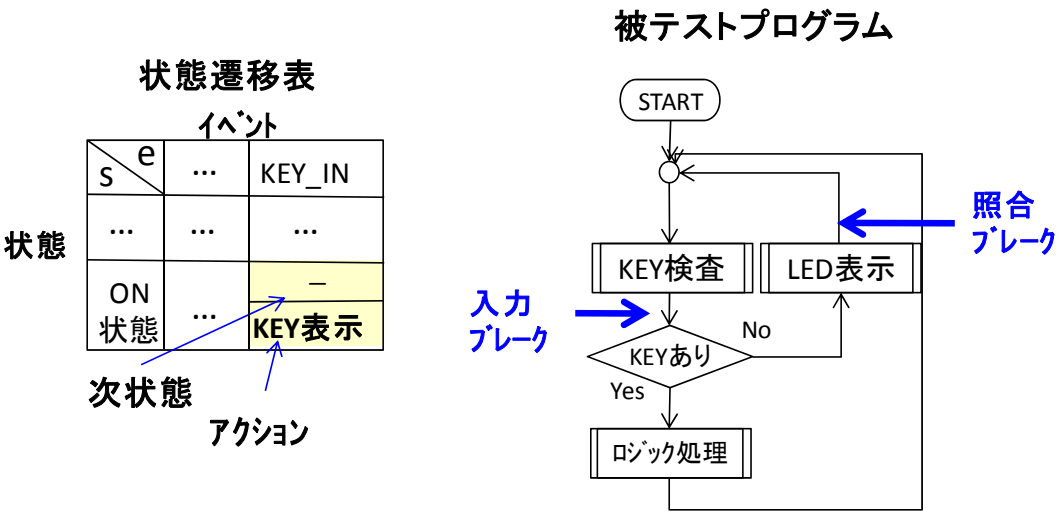


図 4-8　状態遷移表（左図）と被テストプログラム実装（右図）の例

被テストプログラムが 3 層のモジュール構造をとるならば，入出力デバイスドライバはそれぞれ，KEY 検査のチャタリング防止処理と LED のダイナミック点灯処理を受けもち，例えば，次のようにロジック部分の入出力が設計される．

・入力 :「KEY0～9 を押した」という入力があった状態は，「KEY 検査モジュール(入力デバイスドライバ)を通過した後，有効な入力があることを示すフラグを立て，その KEY 値をあるメモリに格納する」に相当する．

・出力 : (KEY0～9 が押された結果)「n 番目の LED の表示がその KEY 値になる」は，LED 表示モジュール(出力デバイスドライバ)に受け渡す前に，LED の値を格納するメモリ配列の n 番目に KEY 値を設定する」に相当する．

この場合，要求仕様「KEY0～9 を押したら 1 個目の LED にその数字を表示する」に対して，図 4-8 右図の矢印のプログラム位置をブレーク時とし，疑似入力の発生，及び結果の照合を行うブレーク定義を，次のように設定することができる．

【入力ブレーク】
イベント： KEY0～9 を押した
・ブレーク時　　KEY 検査モジュール通過後
・入力コマンド　　有効入力フラグ=ON
　　KEY 値保存メモリ=KEY 値
【照合ブレーク】
アクション： 1 番目の LED の表示がその KEY 値になる
・ブレーク時　　LED 表示モジュールの出口
・照合コマンド　　LED 値の格納配列 1 番目==KEY 値

このように，入力ブレークと照合ブレークにより，テストしたい要求仕様とテストデータとを直接対応付けることが可能になる．通常入出力デバイスドライバが実時間仕様を吸収するので，マイコンポートがテスト界面の場合には自動テストできない複雑なタイミングの絡んだ仕様でも，入力ブレークと照合ブレーク上のテストデータとして設定することが可能となる．

(3-2) 被テストプログラムの網羅度の評価

次に，被テストプログラムの網羅度について評価する．本方式が被テストプログラムに対して課す制約は次の 3 つである．

① **可制御制約**: イベントに相当する擬似入力状態を，ICE による複数回のメモリ値設定(イベントに対して 1 組定義)で行えなければならない．

② **処理タイミング制約**: イベントに相当する入力処理がアクションに相当する出力処理の開始前に終了しなければならない．

③ **可観測制約**: ICE による複数回のメモリ参照(アクションに対して 1 組定義)で期待する出力の発生を検査できなければならない．

各制約に対応して，被テストプログラムの構造あるいは要求仕様の記述に対して，場合分けを行い，ブレーク定義が，その場合分けすべてで網羅的に設定できることを確認した．表 4-1 にその結果を示す．可制御制約がイベント発生タイミング，処理タイミング制約が出力照合タイミング，可観測制約が期待値設定に対応し，それぞれに対して場合分けを行っている．

結論として，提案方式によるテスト界面は，要求仕様との対応関係をもっていること，被テストプログラムに対する網羅性が高いことがわかる．

表 4-1　被プログラムの構造・要求仕様の記述パターンに対するブレーク定義の記述法

項目	場合分け		設定法
イベント発生タイミング	ポーリング処理		ブレークポイント＝メモリ検査前のプログラム位置
	割込み処理		割込み発生（含マイコンポート直接書込み）
出力照合タイミング	アクション処理の完了が，プログラム位置で特定できる場合		ブレークポイント＝ロジック処理モジュールの入り口 （含アクションなしのチェック：関心のメモリをチェック）
	それ以外	出力時にメモリ書込み有	ブレークポイント＝メモリ書き込み時 （未到達の場合，2秒タイムアウト→NG）
		出力ドライバにモジュール境界有	ブレークポイント＝出力ドライバモジュール出口 （未到達の場合，2秒タイムアウト→NG）
		それ以外	アクションなしとして常に成功
期待値設定	定数値		照合対象のメモリ値と直接照合
	他メモリ値と前メモリ値を利用する演算結果		当該メモリ値を変数として保存，照合に利用 （機能拡張：ブレーク定義内，四則演算を可）

(3-3) 要求変更に伴う再テスト用の設定作業量の評価

最後に，再テストの容易さを評価する．

提案方式では，状態遷移表の要求変更による再テスト作業への影響は，ブレーク定義のみに限定される．当然，プログラムの制御構造自体が大きく変わった場合には，ほとんどのブレーク定義を再設定する必要が生じるが，そうでない場合には，表 4-2 に示すように，状態遷移表内の要求仕様の変更に伴うブレーク定義の変更は，アクション内容の変更時，及びその他の要素についての追加時に限定される．

イベント数を m，状態数を n とすると，従来方式では，1 つのイベントの追加は状態遷移セル n 個分，1 つの状態の追加は状態遷移セル m 個分のテストデータの変更に相当する．また，出力アクションが同じケースではその重複分だけの変更に相当する．提案方式では，イベントあるいは状態の追加とも，1 つの定義追加を行うだけで済む．

提案方式では，要求仕様変更があった場合でも，イベント，状態，あるいはアクションの定義に変更がないならば，ブレーク定義を編集せずに，再テストが実施できる．また，イベント，状態，あるいはアクションの定義に変更がある場合でも，必要なブレーク定義の追加や変更を行えば，状態遷移セルに相当するテストデータを自動生成できる．これに対して，従来システムでは，要求仕様からテストデータへの自動変換機能がない

ため，要求仕様変更に関わる状態遷移セル分のテストデータの再設定をすべて手作業で行わなければならない．

表 4-2 要求仕様変更が与えるブレーク定義への影響

要求仕様変更部分		要求仕様変更部分
アクション内容（出力内容）変更		照合ブレーク定義内容の変更
状態変更	追加（アクション追加）	照合ブレーク定義の追加
	削除（アクション削除）	変更なし
イベント変更	追加（イベント・アクション追加）	入力・照合ブレーク定義の追加
	削除（アクション削除）	変更なし
遷移先変更		変更なし
条件分岐変更	条件式変更	変更なし
	追加（アクション追加）	照合ブレーク定義の追加
	削除（アクション削除）	変更なし

4.5 実証実験

提案システムの有効性を評価することを目的として，実証実験を行った．

試作システム

提案システムの試作（testCASE と呼ぶ）を行った．testCASE は，PC 上に C，C++，Visual Basic を用いて実装している．図 4-9 に testCASE のシステム構成を示す．

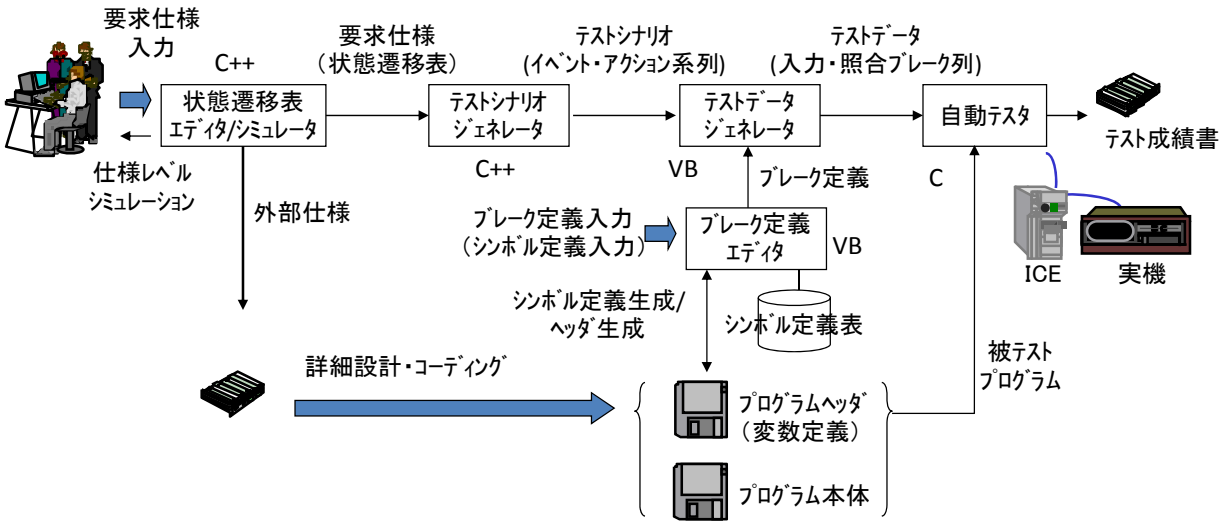


図 4-9 試作システムの構成

testCASE は，状態遷移表エディタ/シミュレータ，テストシナリオジェネレータ，ブレーク定義エディタ，テストデータジェネレータ，及び自動テスタの 5 つのコンポーネントからなる．以下，各コンポーネントの実現方法について説明する．

(1) 状態遷移表エディタ／シミュレータ

状態遷移表エディタ／シミュレータは，要求仕様の入力支援のための状態遷移表編集機能と，要求の妥当性を検証するためのシミュレーション機能を提供する．

図 4-10 に，状態エディタ／シミュレータの画面例を示す．図 4-10 において，左側のイベントボタンによって操作イベント入力を行うことで実行が進む．シミュレーション実行結果として，右側の出力ウィンドウに，アクション出力，及びアクション実行後の内部変数の変化と現在値を表示し，中央の状態遷移表上に，現在状態及び直前に実行された遷移セルをハイライトする．

シミュレーション機能は，顧客との要求の妥当性確認に用いることができる．これによって，早期に要求の品質を向上させることを支援する．さらに，シミュレーション実行時に，入力したイベント及び実行したアクションを記録することで，テストシナリオを作成する機能も提供している．この機能は，テストシナリオジェネレータにより自動生成されるテストシナリオを補強するために用いる．

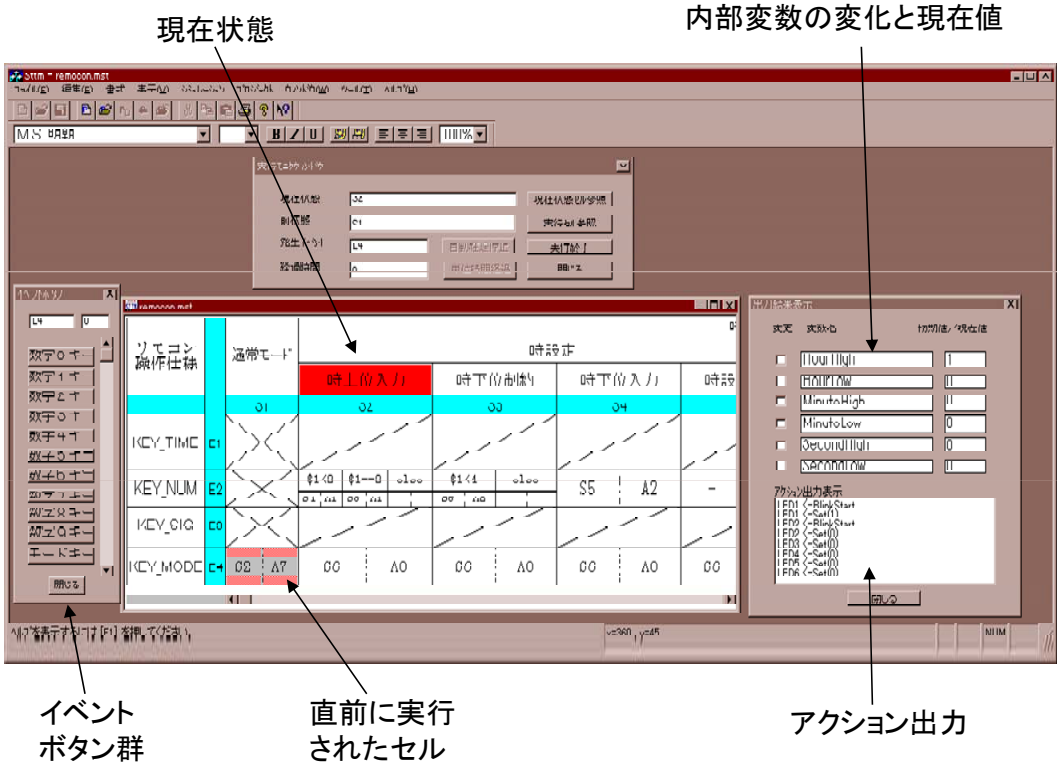


図 4-10 状態遷移表エディタ／シミュレータの画面例

(2) テストシナリオジェネレータ

テストシナリオジェネレータは，状態遷移表からテスト手順を生成する．testCASE では, 実用性に考慮して, 次の 4 つのテストシナリオ生成アルゴリズムを提供している[72].

① 状態遷移網羅パス生成法
すべての状態遷移を少なくとも 1 回は通るテストシナリオを生成するアルゴリズムである．入力イベントの生成をランダムではなく一定の順序で行う以外は，トランジッションツアー法[61]と同じである．

② 状態指定パス結合法
開始状態と終了状態及びパスの長さを指定し，開始状態及び終了状態の間の指定長(あるいはその指定長以内)の全パスを計算する．なるべく網羅的にテストを行いたいが生成シナリオの数を制限したい場合に有効なアルゴリズムである．

③ レイヤ指定パス結合法
ある状態を指定しその後ろ(あるいは前)の網羅的パスを指定する方法である．後ろ(あるいは前)に何ステップをとるか(レイヤ数)を指定することができる．このアルゴリズムは状態の変更や追加に伴う再テストに対して有効なアルゴリズムである．

④ グラフパス展開法
状態遷移表の一部分を指定し部分グラフを計算することで, その部分グラフの全パスを展開する．グラフを全パスへ展開するアルゴリズムは Beizer[27]によるものを用いる．Beizer のアルゴリズムは，状態遷移表の規模が大きくなると計算量及びメモリ使用量が爆発的に増加してしまうが, 対象範囲を状態遷移表の部分に限定することで，計算量を抑えつつ，パス網羅的なテストシナリオを生成する．

状態遷移表は，内部変数（あるいはイベント値）を用いた論理式により条件分岐をもつように拡張している. 従ってテストシナリオ生成の対象となる状態遷移表のクラスは, 変数付き有限状態機械である.

展開では，代表値テスト及び境界値テストを考慮して，各分岐に対応した内部変数の値域から中央値，最小値，及び最大値をとり，別々の状態遷移として分解する方法をとっている．また，ICE で設定可能なテストデータを生成するために，条件分岐内の論理式を次のように制限している.

- 条件分岐では論理オペレーション AND，OR，及び NOT を使わない．算術論理式の左辺は必ず内部変数あるいはイベント値とする.
- 内部変数はテスト対象プログラム変数と関連付けられている.
- 内部変数には上下限値を設定する.

テストシナリオジェネレータでは，アルゴリズムの選択及びそのアルゴリズムが必要とするパラメータの入力を行うことができる．さらに，生成に先立って，各アルゴリズムが生成するテスト項目数及びテスト実行に要する時間の予想を行うことができる．こ

れによって，アルゴリズム選択のための情報をテスト者に提供する．

(3) ブレーク定義エディタとテストデータジェネレータ

テストデータジェネレータは，テストシナリオジェネレータで生成したテストシナリオを，ICE を制御するコマンド列(テストデータ)へ変換する機能を提供する．これは，4.4.1 で述べたテスト界面の設定とデータ変換方式の実装である．

テストデータへの変換に先立って，要求仕様におけるイベント及びアクションをプログラムと対応付けるために，入力及び照合ブレークと，ブレーク定義内で用いるプログラムシンボルの定義が必要である．testCASE では，ブレーク定義エディタがその編集機能を提供する．ブレーク定義では，ブレーク時とコマンドを，プログラムラベルと変数名を直接用いて定義する．このためのシンボル定義表は，直接編集して入力する方法，及びプログラムのヘッダファイルから生成する方法の 2 通りのやり方で定義できる．さらに，シンボル定義表とヘッダファイルを常に同一に保つ機能を提供しており，イベント及びアクションとプログラム間の関連付けを容易に行うことができ，誤りの混入も防止できる．

(4) 自動テスタ

自動テスタの動作を図 4-11 に示す．自動テスタは，ICE との通信の設定，テスト対象プログラムの ICE へのロード，及びテストデータの読み込みを行った後，ICE を制御してテストを自動実行する．

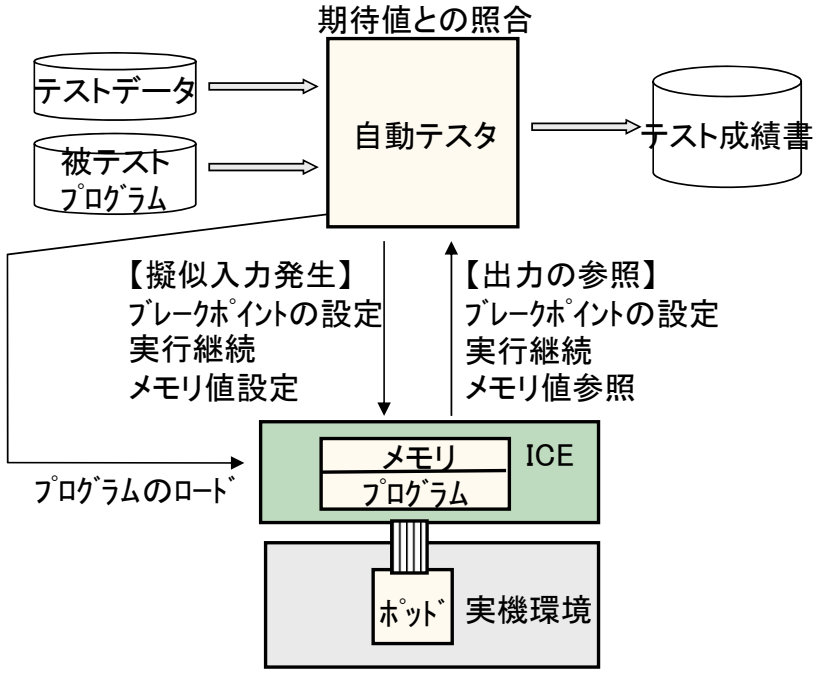


図 4-11 自動テスタの動作

テスト実行では，テストデータジェネレータが生成したテストデータに従って，被テストプログラムに対して，入力ブレーク及び照合ブレークを記述順に実行する．このとき，自動テスタは，入力ブレーク時を契機として，入力コマンドで定義された擬似入力の発生及び指定出力メモリ値の参照を ICE に指示し, 次に照合ブレーク時を契機として，照合コマンドの定義に従い，ICE から出力メモリ値を取得し，期待値との照合を行う．最後に照合結果を記録する．

テスト実行には，テスト列とテスト項目と呼ぶ単位がある．テスト項目は，一対の入力ブレークと照合ブレークに対応する．テスト列は，テストシナリオに対応するテスト項目の列である．一つのテスト列が実行される際， ICE をリセットして初期状態からスタートし，テスト項目を順に実行する．

テスト実行の結果は，テスト項目毎に報告する．エラーの種類には，タイムアウトエラー及び照合エラーがある．タイムアウトエラーは，入力あるいは照合ブレークで指定されたブレーク時(プログラムアドレスあるいは変数への書き込み)が起きなかったときのエラーである．照合エラーは，メモリ値を期待値と照合して異なる場合のエラーである．照合エラーでは，一致しないメモリ値及び期待値を報告する．テスト結果は，テスト成績書としてドキュメント出力する．

自動テスタは，このように，一連のテスト作業，すなわちテストシナリオ作成，テストデータ作成，テスト実行，及び結果ドキュメント化を自動化する．

適用評価

テストの自動化による製品検査プロセスの作業効率向上について評価を行った．評価対象分野は，家庭電気製品などの小規模組込みシステムとし，評価はその調査データに基づいて実施した．

(1) 評価方法

まず，基本データの収集を次のように行った．対象分野の作業現場(ソフトウェアハウス 10 社)を対象として，ヒアリング調査を行い，日常的な要求仕様の量と，項目当たりのテスト作業時間を得る．実験用データを使って，本システムを利用した場合のテスト実施時間を計測し，テスト項目あたりの平均テスト実施時間を計算する．

次に，収集した基本データを用いて，典型的な作業形態を想定し，従来の手作業の場合，及び本システムを利用した場合におけるテストの総作業量を比較する．

(2) 評価結果

対象分野の機能仕様では, 状態遷移表の大きさがイベント数 50×状態数 100 以上となる．手作業で実機テストを行う場合，テストシナリオの作成を行い，実際にテストを実施し，その結果を記録する．一方，testCASE を利用した場合では，ブレーク情報の設定

を行う作業以外は，テストシナリオ生成，テストデータ生成，及びテスト実行と結果文書化を，一貫して自動化できる．

テスト実施時間

60×120 の表で単純に 7200 個のテスト項目がある．手作業では，1 項目あたり 20 秒かかるとすると，40 時間（8 時間×5 日間）の作業量となる．testCASE を利用した場合では，1 項目の処理時間は，タイムアウトで照合エラーになる場合(2.0 秒)も含めて平均 1.0 秒かかるものとすると，2 時間でテストが完了する．しかも，この間作業者の時間を拘束しない．

テストシナリオ生成

手作業では，テストシナリオは作成に時間がかかり過ぎるため，通常は作成せず状態遷移表で代用している．このため，テスト手順が，非効率でパス網羅性の低いものになりやすい．この問題は，testCASE が自動生成することで解決している．

(3) 結論

評価結果より，testCASE を用いることにより，検査網羅性の確保とテスト作業時間の大幅な短縮が図ることが可能であることを確認できた．

また，ソフトウェアハウス 3 社で試使用を実施してもらい，参加したテスト作業者により，テスト実行時に作業者が拘束されないため効率的であるという評価を受けた．また，プログラム内のシンボル情報と，ブレーク定義で利用するシンボル情報の間で双方向に保守する機能についても評価が高かった．しかし，ブレーク定義自体の誤りによってテストが失敗するケースが多いことも指摘された．プログラム修正に対応したブレーク定義の正しい修正のために，何らかの支援機能が必要であることがわかった．

4.6 本章のまとめ

適用範囲

(1) 適用分野

提案システムは，小規模な組込みシステムの実機テストを自動化する．別の製品分野である**非組込みシステム**は，ICE を用いない開発であるため，そのまま適用することはできない．**大規模な組込みシステム**も，マルチプロセッサ構成をとることが多く，この場合適用不可となる．また，非組込みシステムは，要求仕様が反応的なものの比率が低く，その場合，状態遷移に基づくテスト自動化のアプローチ自体の適用効果は薄い．

(2) 実機テストとしての有効性

実機テストでは，ソフトウェアが実機に組込まれた状態で機能することを確かめなければならない．提案システムは，実機に載った状態でのテストを自動化している．しかし，最終的なマイコンポート入出力を検査するわけではなく，プログラム内部の変数値を設定･参照する方法を採っているので, 実機テストとして有効性を保証するためには,

入出力デバイスドライバがそれぞれ正常に機能していることを別途検査する必要がある．これらは，入力デバイスドライバに対しては，手作業による入力及びメモリ状態のチェック，出力デバイスドライバに対しては，メモリへの値設定及び出力の目視確認を，一通りの入出力について，検査することで十分である．これらは短時間で実施できる．

(3) テスト可能性

提案システムが被テストプログラムに課す制約は，可制御制約，処理タイミング制約，及び可観測制約であるが，これらは，処理の実行順序関係と処理間のデータの受け渡し方法のみを規定しており，制御構造を柔軟に選択する自由度を残している．例えば，

- 各処理は，異なるタスクにまたがってもよい．例えば，割込みを利用して他の処理へジャンプするタイプの制御構造も適用可能である．
- 各処理群は明確なモジュールの形式をとっていなくてもよい．また，入力処理群と出力処理群の前後関係さえ守られるならば，ロジック処理が出力処理群を順にサブルーチン呼出するような構造も可能である．
- ICE がポート値をレジスタ値として扱うことを許す場合，テスト界面はマイコンポートも含んだものとなる．つまり，ADO のように直接マイコンポートへの信号入力と参照を行うこともできる．

このように，提案システムは反応的な機能仕様を持つ小規模な組込みプログラムへの適合性が高い．今後，上記制約を設計ガイドライン化することで，ほぼ完全な適合性を確保することができると考えている．

(4) 対象仕様種別

提案システムは，要求仕様全体に対して状態遷移表で記述可能な範囲を扱っている．小規模な組込みソフトウェアでは，その割合は 50%強を占めており[73]，その部分について，要求分析プロセスの要求妥当性確認，及び製品検査プロセスの再テスト作業を効率化できることの効果は大きい．

状態遷移表では扱えない要求として，性能仕様（任意の２時点間の所用時間に関する言明）がある．提案システムでは，ICE のエミュレーション機能を使用するため厳密な意味で実時間実行はできないことから，性能仕様を検査することはできない．また，信頼性・使用性などの品質要求も対象外である．

まとめ

小規模な組込みシステムの製品検査プロセスにおけるテスト自動化システムを提案し適用評価した．

従来システムでのテスト自動化が，テストシナリオ作成，テストデータ作成，及びテスト実行と結果文書化を一貫して支援できていないこと，そしてその原因がテスト界面の設定にあることを示した．

提案システムは，小規模な組込みソフトウェアの典型的モジュール構造に注目し，ICE

を制御することで，プログラムへの擬似入力及びプログラムからの出力参照を行うことができるテスト界面を定義し，その界面を用いてテスト自動化方式を実現する．この方式により，要求仕様レベルのテストシナリオと，テスト界面上のテストデータとの間の直接的対応関係が定義可能となり，その定義を用いることによって要求仕様に基づくテスト自動化が可能となった．実証実験によって，提案システムが，製品検査プロセスにおける検査網羅性の確保及び作業量の削減に効果があることがわかった．

第5章　　品質とコストの見積りに基づく品質計画支援システム

5.1　本章の研究の目的と位置づけ

本章の研究の目的は，エンジニアリングプロセスの誤り除去活動の効果と効率の向上を狙いとし，品質とコスト目標を達成する品質計画支援システム，及びその定量的基盤を提供する品質とコストの見積りモデルを提案・適用評価することである．品質計画支援システムは，プロジェクトマネージャ（以下ＰＭと略す）が，品質マネジメントの基本戦略に沿い，コスト最適な品質計画を策定することを支援する．ここで，品質マネジメントの基本戦略とは，以下に示す戦略，施策，及びコスト最適化からなるものである．

(1) 戦略

品質マネジメントの狙いは，出荷後残存誤り量の抑制にある．Gaffney[39]は，エンジニアリングプロセスの各フェーズの検知誤り量から，図 5-1 に示すような Rayleigh 曲線を外挿することで，出荷後残存誤り量を推定できることを示した．これによれば，出荷後残存誤り量の抑制は，次の２つの戦略に集約できる．

〔戦略 1〕　混入誤り量が同じなら，誤りをより早期に検知
〔戦略 2〕　総混入誤り量を抑制

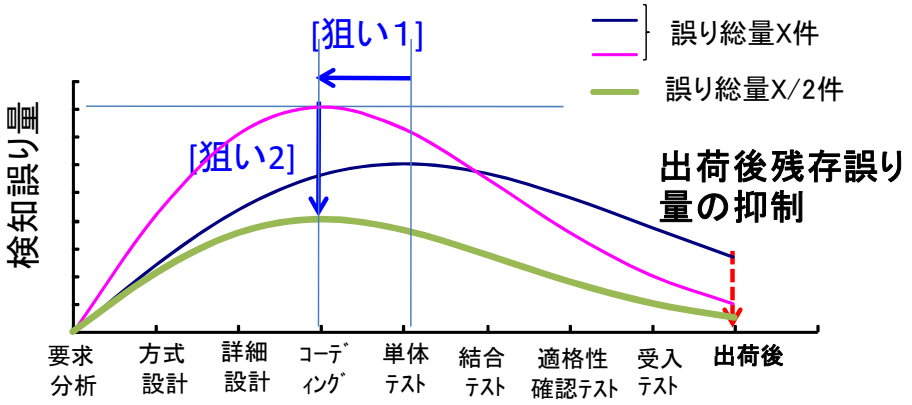


図 5-1　エンジニアリングプロセスの検知誤り量と品質マネジメントの狙い

(2) 施策

以下に示す施策 1～3 を計画し実施することで，上記戦略を達成する．以下，レビューとテストによってソフトウェアの誤りを発見するためのプロセスを，**品質プロセス**と呼び，品質プロセスに対する投資コストを**品質プロセスコスト**と呼ぶこととする．

［施策 1］　品質プロセスコストの早期投入

品質プロセスコストを，エンジニアリングプロセスの早い時期での投入することにより，戦略 1 を実現する．

次の 2 つの施策は，それぞれ戦略 1 と戦略 2 の達成に寄与する改善事項の盛り込みである．

［施策 2］　品質プロセスの誤り検知効率の向上

［施策 3］　技術プロセスの質の向上

(3) コスト最適化

誤りが後フェーズに流出した場合，誤りの検出が後フェーズになるほど，リワークコストが増大する[41]．施策 1 は，品質プロセスコストを早期に消費することで，リワークコスト削減の効果を得るものである．しかし，品質プロセスコストへの投資は，多すぎるときにその投資に見合うだけのリワークコスト削減の効果を生みださない．

つまり，施策 1 の効率のよい実施は，目標とする出荷後残存誤り量の達成を制約事項とし，各フェーズへの品質プロセスコストの投入量を制御量とし，その結果としての品質プロセスコストとリワークコストの総和を最小化する最適化問題と考えることができる．図 5-2 に，施策 1 の実施による開発コスト削減達成のイメージを示す．

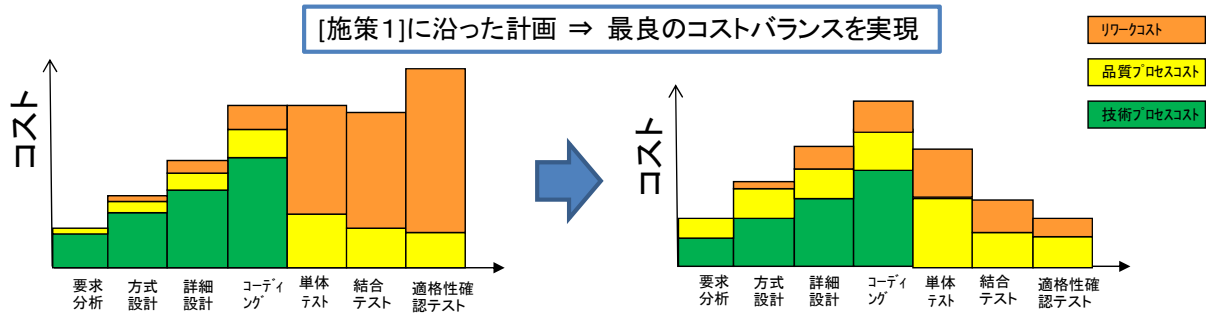


図 5-2　施策 1 よる開発コスト削減の達成

本章の研究の目標状態は，**プロジェクトマネージャが，品質マネジメントの基本戦略に沿い，出荷後残存誤り量の目標を達成するコスト最適な品質計画を策定できるようになること**である．ここで，**品質計画**とは，施策 1，施策 2，及び施策 3 の計画を指すものとする．品質とコストの見積りモデルは，施策 1 のための定量的基盤を提供する．

なお，本章においては，作業コストを工数によって評価する．すなわち，コストと工数を同義として扱い，「人時間（Mh）」を単位とする．以下，断りのない限り「コスト」を用いる．

5.2　従来システムと問題点

従来の品質計画支援システムは，総量管理モデル[42]に基づく品質見積り機能を提供している．図 5-3 に，総量管理モデルに基づく品質見積り機能をベースとした従来の品質計画支援システム示す. ここで, 品質プロセス密度[Mh/KSLOC]と誤り密度[件/KSLOC]は，それぞれプログラム開発規模 1 KSLOC 当たりの品質プロセスコストと誤り量を表すものとする．

従来の品質計画支援システムは，品質見積り機能により，フェーズ毎の品質プロセス密度を入力し，各フェーズの検知誤り量を見積る．これによって，ＰＭは，フェーズ毎の検知誤り密度及び品質プロセス密度が組織で定める品質基準値を満たしているか，及び最終残存誤り量の見積り値が目標値を満たしているか，を確認できる．しかし，従来システムは，品質計画支援を行う上で 2 つの問題をもつ．

[問題 1] コストの見積り結果を出力しない．
[問題 2] 誤り検知効率を決定する誤り種別に，誤り混入フェーズを考慮していない．

このため，問題 1 により，品質計画のコスト最適性を確認することができない．問題 2 により，どの検知フェーズでどの誤り種別をどれだけ検知すべきかの的確な判断ができない．

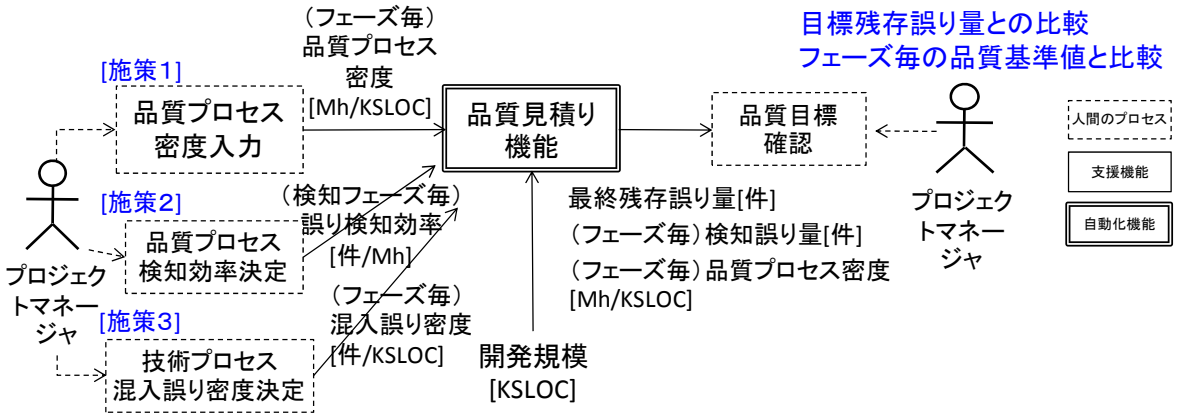


図 5-3　総量管理モデルに基づく品質計画支援システム

5.3　提案システムと技術課題

システム要件

「プロジェクトマネージャが，品質マネジメントの基本戦略に沿い，コスト最適な品質計画を立てられるようにすること」を目標状態として，従来システムがもつ 2 つの問題を解決するような，品質計画支援システムを提案する．提案システムの要件は，以下の 2 点である．

[要件 1]　品質プロセスコストとリワークコストの見積り機能をもつこと
[要件 2]　誤り種別に混入フェーズを考慮すること

提案システムの構成と解決すべき技術課題

図 5-4 に，提案する品質計画支援システムのシステム構成を示す.

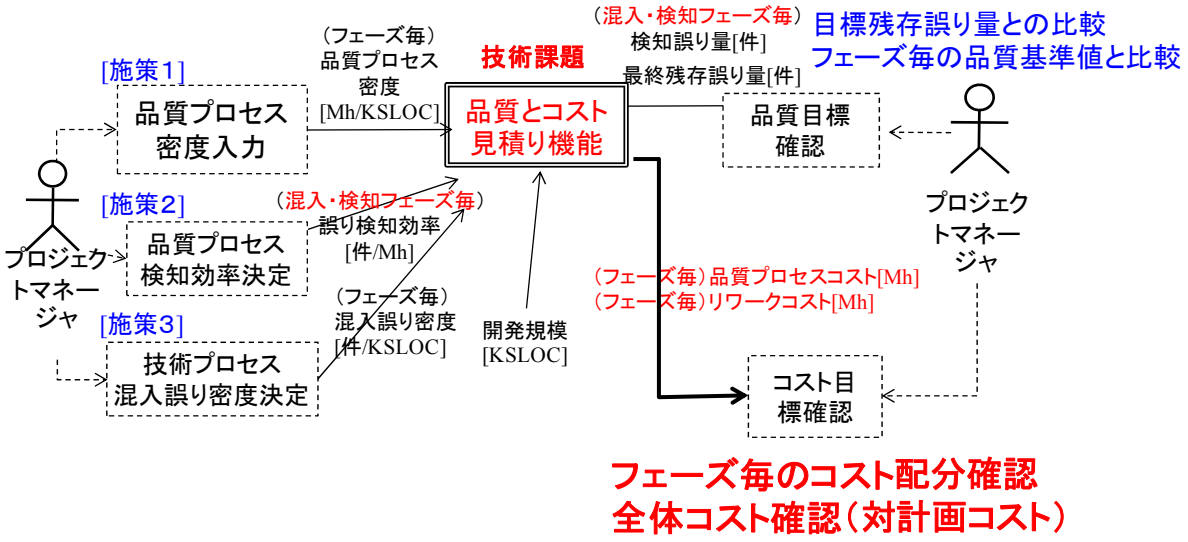


図 5-4　提案システムの構成と技術課題

従来システムの品質見積り機能を，品質とコスト見積り機能に拡張し，誤り検知効率と誤り 1 件あたりのリワークコストについて，混入フェーズと検知フェーズの両方を考慮できるようにしている．提案システムを実現するための技術課題は，以下である.

[技術課題] 誤りの混入及び検知のフェーズを考慮した，検知誤り量とリワークコストの見積りモデル

5.4　提案モデルとその評価

誤り量とコストの関係の定式化

品質プロセスの実施とその効果に関する問題を，ビジネスモデリング手法である IDEF0[46]を用いて定式化するとともに，本章で用いる用語の定義を行う．図 5-5 に，この目的での IDEF0 の記法の用い方を示す.

図 5-5 において，ボックスはエンジニアリングプロセス，左から入るアローはプロセスへの入力，右へ出て行くアローはプロセスの出力，上から入るアローはプロセスの効率を制御する施策，下から入るアローはプロセスの実施にかかる作業コストを指すもの

とする．図 5-5 は，「プロセス(*P*)は，施策(*V*)の下に，作業コスト(*C*)を費やして，誤り量(*DI*)を含む入力成果物(*AI*)を，誤り量(*DO*)を含む出力成果物(*AO*)に変換する」と読む．施策には，このプロセスを遂行するにあたってのマネジメントの方針，指示，及び作業マニュアルが含まれる．

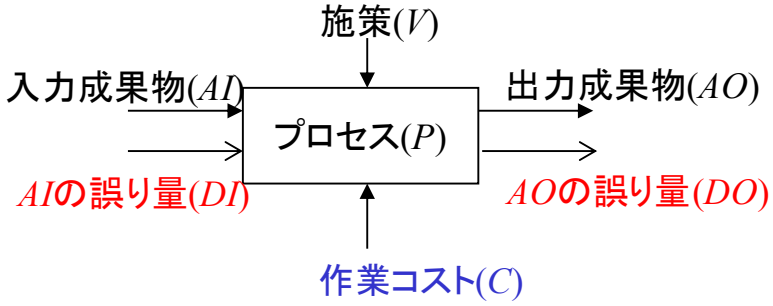


図 5-5　エンジニアリングプロセスに対するIDEF0 の記法

図 5-5 の記法を用いて，図 5-6 にエンジニアリングプロセス全体を表現する．

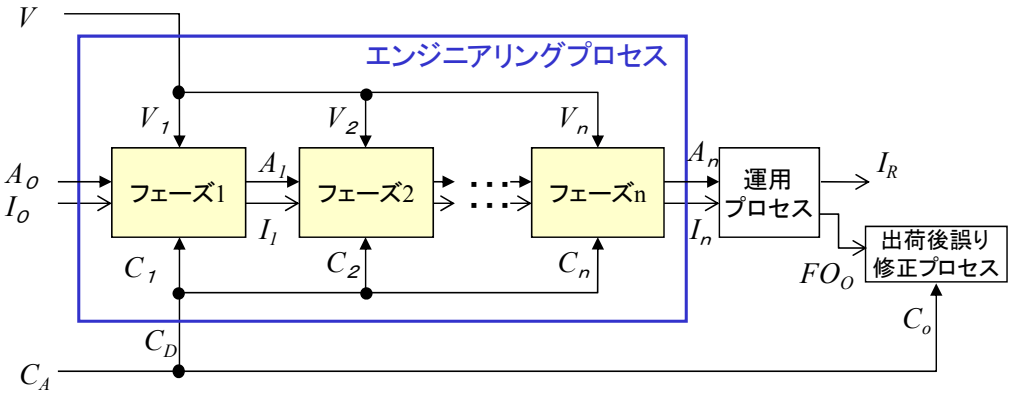


図 5-6　エンジニアリングプロセス全体の誤りとコストの影響関係

一般に，エンジニアリングプロセスは，より詳細な複数プロセスのネットワークとして表現される．ここでは，ライフサイクルモデルとしてウォータフォールモデルを採用し，エンジニアリングプロセスを n 個の直列なプロセスからなるものとする．これらのプロセスを**フェーズ**と呼ぶ．フェーズ i に関し，A_i はフェーズ成果物，I_i は残存誤り量とする．エンジニアリングプロセス全体は，誤り量 I_o を含む初期仕様 A_o を入力として，施策 $V(=\Sigma V_i)$の下に，開発コスト C_D を費やして，残存誤り量 I_n を含む製品 A_n を出力する．

運用プロセスは，残存誤り量 I_n を含む製品 A_n を運用する．運用プロセスにおいて，残存誤り量 I_n の一部が発見される．このとき，FO_o を運用プロセスで発見される誤り量，

I_R を運用で顕在化しない誤り量とする（$I_R = I_n - FO_o$）．出荷後誤り修正プロセスで FO_o を修正するコスト C_o を，**出荷後ロスコスト**と定義する．開発コスト C_D と出荷後ロスコスト C_o の和 C_A を，**製品コスト**と定義する．製品コスト C_A は，次式となる．

$$C_A = C_D + C_o \tag{5-1}$$

図 5-7 は，図 5-6 の開発各フェーズを詳細化したものである．

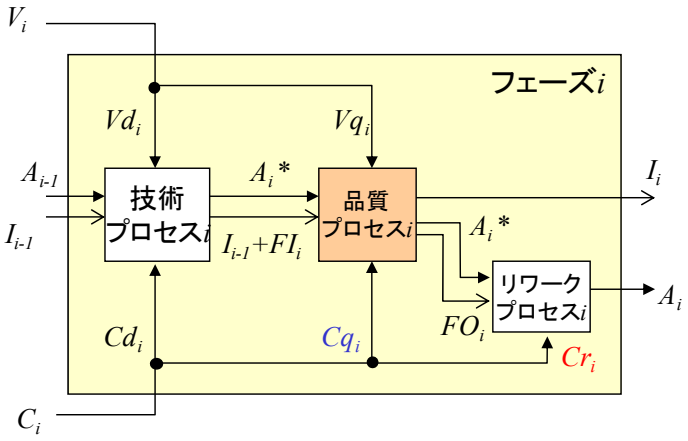


図 5-7　エンジニアリング各フェーズの詳細化

フェーズ i は，技術プロセス i，品質プロセス i，及びリワークプロセス i の 3 つのプロセスからなる．**技術プロセス i** は，前フェーズの成果物 A_{i-1} から成果物 A_i*を作成するプロセスであり，前フェーズから流出してきた誤り量 I_{i-1} 及び技術プロセスの実施に伴う新たな誤り量 FI_i を成果物 A_i*に混入させる．**品質プロセス i** は，成果物 A_i*に含まれる誤りを検知するプロセスであり，$I_{i-1}+FI_i$ を検知誤り量 FO_i と残存誤り量 I_i とに分離する．**リワークプロセス i** は，検知誤り量 FO_i を含む成果物 A_i*を修正し，成果物 A_i を出力するプロセスである．Vd_i は技術プロセス i，Vq_i は品質プロセス i に対する施策であり，それぞれ混入誤り量 FI_i の抑制と，誤り除去率 b_i の増大に寄与する．ここで，フェーズ i での**誤り除去率** b_i を次式で定義する．

$$b_i = FO_i/(I_{i-1}+FI_i) \tag{5-2}$$

誤り除去率は，フェーズ i に混入あるいは流入した誤り量の内，そのフェーズ i で検知される誤り量の割合である．

フェーズ i の作業コスト C_i は，次式となる．

$$C_i = Cd_i + Cq_i + Cr_i \tag{5-3}$$

ここで，Cd_i ，Cq_i ，Cr_i は，それぞれ技術プロセス i，品質プロセス i，リワークプロセス i に対する作業コストを表す．

開発コスト C_D は，次式で定義する．

$$C_D = \Sigma C_i \quad (5\text{-}4)$$

また，エンジニアリングプロセス全体で各作業コストを総和したものについて，$Cd(=\Sigma Cd_i)$ を**技術プロセスコスト**，$Cq(=\Sigma Cq_i)$ を**品質プロセスコスト**，$Cr(=\Sigma Cr_i)$ を**開発リワークコスト**と呼ぶこととする．品質プロセスコスト Cq と開発リワークコスト Cr の間の依存関係を直接扱うための指標として，**開発品質コスト** C_{DQ} を以下のように定義する．

$$C_{DQ} = Cq + Cr \quad (5\text{-}5)$$

さらに，**製品品質コスト** C_{AQ} を，開発品質コスト C_{DQ} に出荷後ロスコスト C_o を加えたものとして定義する．

$$C_{AQ} = C_{DQ} + C_o \quad (5\text{-}6)$$

フェーズ i の品質プロセスコスト Cq_i が増えれば，品質プロセス i の誤り除去率 b_i が高くなる．それによって，その検知誤り量 FO_i が増え，後フェーズへの流出誤り量 I_i が減少する．

$$b_i = f(I_{i-1}+FI_i, \mathbf{Cq_i}, Vq_i) \quad (5\text{-}7)$$

$$FO_i = (I_{i-1}+FI_i) \times b_i \quad (5\text{-}8)$$

$$I_i = I_{i-1} + FI_i - \mathbf{FO_i} \quad (5\text{-}9)$$

一方，検知誤り量 FO_i が増加すれば，その修正を行うためにリワークコスト Cr_i が増大する．

$$Cr_i = g(FO_i) \quad (5\text{-}10)$$

フェーズ i における Cq_i への投資は，Cq_i と Cr_i の増大をもたらし，(5-3)式により，そのフェーズの作業コスト C_i を増大させる．

エンジニアリングプロセス全体では，Cq_i の増加は，直接 C_{DQ} 及び C_{AQ} の増加をもたらす．しかし，その効果として，後フェーズへ流出する誤り量 I_i が減少する．この減少は Cr 及び C_o を減らす効果をもち，結果として開発品質コスト C_{DQ} 及び製品品質コスト C_{AQ} を減らす．このように，Cq_i の増加は，(5-3)式の指標に対して悪影響，(5-5)式及び(5-6)式の指標に対して好影響と悪影響の両方をもつことがわかる．

従来研究

(1) 品質プロセスコストから検知誤り量の見積り

各フェーズの品質プロセスコスト Cq_j が，各フェーズの誤り検知量 FO_j にどれだけ貢献するかを見積るモデルについて，従来技術を以下に述べる．

誤り総量管理モデル[42]は，フェーズ i の混入誤り量 FI_i を所与のものとして，フェーズ j の検知誤り量 FO_j から，後フェーズへの流出誤り量 I_j を，(5-9)式に従い順次計算す

るモデルである．フェーズ j の品質目標としては，(5-2)式で表わされる誤り検出率 b_j を用いる．この目標を実現するのに必要な品質プロセスコスト Cq_j の見積り値は，フェーズ j の誤り検知効率の組織能力値 k_j^0 [件/ Mh]を利用して，次式により計算する．

$$Cq_j = FO_j / k_j^0 = b_j (I_{j-1}+FI_j) / k_j^0 \qquad (5\text{-}11)$$

誤り総量管理モデルは, 同一の組織において組織の標準的な開発スタイルに従う限り，各フェーズで：

・設計プロセスは，作成ドキュメント及びプログラムに一定の密度で誤りを作りこむ.
・品質プロセスは，それに費やしたコストに比例した誤り量を検出する.

という前提に基づいている.

表 5-1 に示す**誤りデータマトリクス**[40]は，誤り総量管理モデルにおいて，誤り検出率の実績データを得るために用いる．表 5- 1 では，その混入フェーズ i と検知フェーズ j 別に検知誤り件数を, 検知誤り量の実績値 D_{ij} としてカウントする. をこの実績データより，混入誤り量のプロジェクト実績値 FI_i^p と検知誤りのプロジェクト実績値 FO_j^p を求める．また，次式を使って，各フェーズの誤り検知効率のプロジェクト実績値 k_j^p を算出する．

$$k_j^p = FO_j^p / Cq_j^p \qquad (5\text{-}12)$$

組織能力値 FO_j^0 及び k_j^0 は，複数のプロジェクト実績値からスクリーニング等をかけ決定する.

表 5-1　誤りデータマトリクスの例

混入フェーズ i / 検知フェーズ j		ソフトウェア要求分析 1		ソフトウェア方式設計 2		ソフトウェア詳細設計 3		コーディング 4		計
ソフトウェア要求分析（レビュー）	1	D_{11}	62							62
ソフトウェア方式設計（レビュー）	2	D_{12}	10	D_{22}	98					108
ソフトウェア詳細設計（レビュー）	3	D_{13}	3	D_{23}	57	D_{33}	120			180
コーディング（レビュー）	4	D_{14}	1	D_{24}	5	D_{34}	15	D_{44}	340	361
単体テスト	5	D_{15}	0	D_{25}	3	D_{35}	30	D_{45}	140	173
ソフトウェア結合テスト	6	D_{16}	0	D_{26}	17	D_{36}	2	D_{46}	12	31
ソフトウェア適格性確認テスト	7	D_{17}	2	D_{27}	2	D_{37}	1	D_{47}	5	10
受け入れテスト	8	D_{18}	1	D_{28}	0	D_{38}	1	D_{48}	1	3
出荷後	9	D_{19}	0	D_{29}	0	D_{39}	1	D_{49}	2	3
計		R_{11}	79	R_{22}	182	R_{33}	170	R_{44}	500	931

誤り総量管理モデルは，混入フェーズの異なる誤りをすべて同様に扱っている．例えば，ソフトウェア要求分析フェーズで混入した誤りと，ソフトウェア詳細設計フェーズで混入した誤りは，ソフトウェア結合テストにおいて，同一効率で検知される.

V 字モデルは, ソフトウェア品質保証の枠組みとしてよく使用される. V 字モデルは,

あるテストフェーズを,「プログラムが，V 字の対応設計フェーズの設計を満たすことを確認する検査」と位置づけている．従って，V 字モデルに従う場合，ある設計フェーズを混入フェーズとする誤りは，対応するテストフェーズで多く見つけ出されるはずである．誤りの総量管理モデルは，この点を考慮していない．

総量管理モデルの技術的問題を以下にまとめる．

[P1-1] 誤りの扱いが，V 字モデルに適合しない ．あるテストフェーズは，対応する設計フェーズの出力に基づく検証であるという作業目的で実施されているのに，その事実を反映していない．

[P1-2] 誤り検知効率を，成果物に含まれる残存誤り量によらず，定数値として扱うことは，合理的でない．一定時間品質プロセスコストを投入したら，すべての誤りを取りきることが可能となってしまうからである．

なお，表 5- 1 では，具体的な開発フェーズとして，ソフトウェア・ライフサイクル・プロセス [47]を用いているが，以降もこれに従うものとする．

(2) 検知誤り量からリワークコストの見積り

次に，フェーズ毎の検知誤り量の推定値 FO_i から，リワークコスト Cr 及び出荷後ロスコスト C_o を求める従来技術について述べる．

Kan[40]は，誤りを検知したフェーズ別に，修正にかかるコスト実績を集積し，フェーズ別に誤り 1 件あたりのリワークコストの組織能力値 cr_i^0 を求め，誤り量の推定値 FO_i と掛けることで，リワークコストの見積り値を求める方法を用いている．

$$Cr = \qquad \times \left\langle cr \right\rangle_{i}(0) \tag{5-13}$$

要求段階や方式設計段階のような上流フェーズで混入した誤りは，下流フェーズで混入した誤りに比べて，より多くのリワークコストを要する[41]と言われており，この性質を用いていない．Kan の方法の問題点を以下に示す．

[P2-1] 誤り修正コストが誤りの混入フェーズに依存する性質を利用していない．

提案モデルの構成と課題

提案する見積りモデルは，(5-5)式に示す開発品質コスト C_{DQ} を，各フェーズに投資する品質プロセスコスト Cq_i から見積る．図 5-8 に，提案モデルの全体像を示す．

提案する見積りモデルは，検知誤り量の見積りモデル及びリワークコストの見積りモデルの 2 つのモデル要素からなる．従って，提案モデルを実現するために解決すべき課題は，以下の 2 点である．

[課題 1] 検知誤り量の見積りモデル，及びそのパラメータの決定方法

[課題 2] リワークコストの見積りモデルと，及びそのパラメータの決定方法

なお，提案する見積りモデルでは，フェーズ i の混入誤り密度 r_{ii} の見積り値は所与であるとする．この見積りには，総量管理モデル[42]や COQUALMO[43][45]の誤り混入モデルを利用することができる．

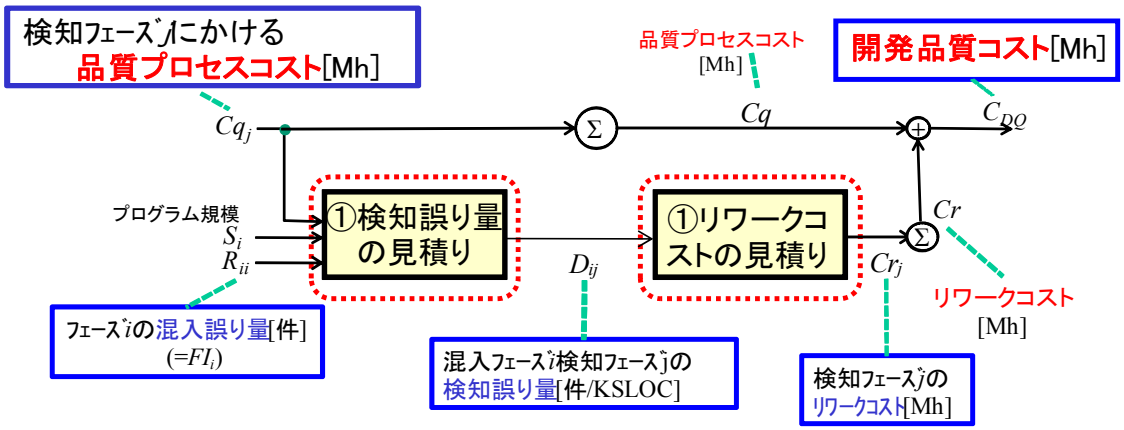


図 5-8　提案見積りモデルの全体像

課題を解決する上での研究アプローチは，以下の 2 点である．

- ソフトウェア工学の知見の活用し，従来研究の問題を解決する．
- 2 つのモデル要素について，問題点を解決するモデル要件と，それを満たす採用モデルとパラメータ算出法の組み合わせを分離し，実現可能な組合せを選択する．

以下 2 つのモデル要素について，提案と評価を行う．

検知誤り量の見積りモデル

(1) モデル要件

混入フェーズ別の残存誤り量に対して，各フェーズの品質プロセスコスト投入の結果として期待できる検知誤り量を推定するモデルを定式化する．これは，フェーズ i で混入した誤りをフェーズ j で検知する量を D_{ij} として，D_{ij} を推定する関数 f を決定することである．

$$D_{ij} = f(R_{ij}, Cq_j, S_j) \qquad (5\text{-}14)$$

ここで，R_{ij} はフェーズ i で混入した誤りの内フェーズ j に流入した誤り量($i=j$ のときはフェーズ内で混入する誤り量)，Cq_j は投入する品質プロセスコスト，S_j は対象ソフトウェアの規模ファクタとする．　R_{ij} 及び D_{ij} は，流出誤り量 I_j，検知誤り量 FO_j，及び混入誤り量 FI_j と以下の関係にある．

$$I_{j-1} = \sum_{i=1}^{j-1} R_{ij} \qquad FO_j = \sum_{i=1}^{j} D_{ij} \qquad FI_j = R_{jj} \tag{5-15}$$

モデル要件は以下の 2 つである.

要件 1： f は Cq_j に関して，Cq_j=0 で D_{ij}=0，かつ Cq_j の増加に伴って D_{ij}=R_{ij} に漸近する単調増加関数であること.

要件 2： D_{ij} は R_{ij} と強い正の相関をもつこと.

(2) 採用モデル

我々は，上記の要件を満たすモデルとして以下を採用した.

$$D_{ij} = r_{ij} \times S \times E_{ij} \tag{5-16}$$

ここで， S は開発プログラムの規模[KSLOC]である．r_{ij} は，各フェーズの規模ファクタ S_j を S としたとき，R_{ij} を S で正規化した流入誤り密度[件/KSLOC]である．E_{ij} は r_{ij} に対する誤り検知率（値域 0～100%）であり，次式で表す.

$$E_{ij} = 1 - e^{-k_{ij} cq_j} \tag{5-17}$$

ここで，cq_j は，Cq_j を S で正規化した品質プロセスコストの密度[Mh/KSLOC]であり，これ以降，**品質プロセス密度**と呼ぶこととする．k_{ij} は，混入フェーズ i の誤りに対する検知フェーズ j における検知能力を表す**誤り検知効率係数**である.

(5-17)式は，指数型信頼度モデル[54]と呼ばれ，検知誤り量をこの曲線に外挿することで，残存誤り量を推定するのに用いるモデルである．指数型信頼度モデルは，瞬間的な誤り検出効率が，残存誤り量に（期待値の意味で）比例することを前提にしたモデルである．図 5-9 に，品質プロセス密度 cq_j，誤り検知効率係数 k_{ij} における誤り検知率 E_{ij} の形状を示す.

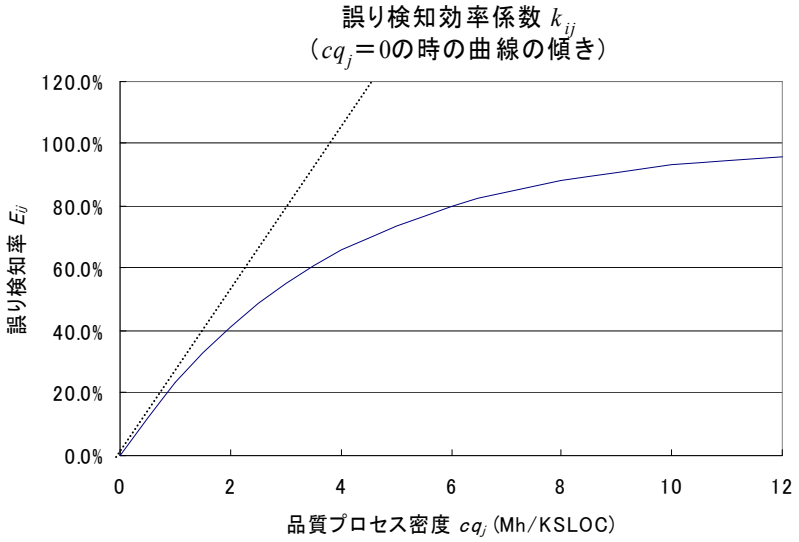


図 5-9　誤り検知率のグラフ形状

r_{ij}（$j>i$）は，フェーズ i の混入誤り密度 r_{ii} [件/KSLOC] 及びフェーズ i 以降の誤り検知率 E_{ij} から，次式で順に計算することができる．

$$\begin{aligned} d_{ij} &= r_{ij} \times E_{ij} \\ r_{ij+1} &= r_{ij} - d_{ij} \end{aligned} \qquad (5\text{-}18)$$

ここで，d_{ij} は検知誤り密度[件/KSLOC]である．

採用モデルでは，以下の２つの仮定を採用している．これらは，総量管理モデルと同じ仮定である．

- 規模ファクタは，フェーズによらず一定である．
- 誤り検知率 E_{ij} は，残存誤り密度の初期値 r_{ij} には依存せず，品質プロセス密度 cq_j の関数である．

(5-16)-(5-18)式で表される採用モデルは，誤り種別に誤りデータマトリクス以上の分解能をもたない場合において，要件 1 と 2 を満たす最も単純なモデルの一つである．

(3) パラメータ算出

(5-16)-(5-18)式により D_{ij} を計算するために必要なパラメータは，フェーズ毎混入誤り密度 r_{ii} と誤り検知効率係数 k_{ij} の 2 つである．

まず r_{ii} についての算出方法を示す．組織内の n 個のプロジェクト(プロジェクト番号 p=1,…,n)で，プログラム規模の実績値 S^p[KSLOC]，及び誤りデータマトリクスによる誤り検知量の実績値 D_{ij}^p[件]を収集する．混入誤り密度の組織能力値 r_{ii}^0[件/KSLOC]は，(5-19)式によりプロジェクトの実績値を平均して求める．

$$\begin{aligned} r_{ii}^0 &= \frac{1}{n}\sum_{p=1}^{n}(R_{ii}^p / S^p) \\ R_{ii}^p &= \sum_{j\geq i} D_{ij}^p \end{aligned} \qquad (5\text{-}19)$$

品質計画策定時に用いる個別プロジェクトの r_{ii} は，組織能力値 r_{ii}^0 に，個々のプロジェクト特性（問題の難しさ，設計技法，要員能力等）を考慮して決定する．

次に k_{ij} についての算出方法を示す．組織内の n 個のプロジェクト(プロジェクト番号 p=1,…,n)で，誤り検知量の実績値 D_{ij}^p [件]に加え，品質プロセス密度の実績値 cq_j^p[Mh/KSLOC]を収集する．組織能力値 k_{ij}^0 は，(5-17)式に基づき(5-20)式を用いることにより，プロジェクトの実績を平均して求める．

$$k_{ij}^{0} = -\frac{1}{n}\sum_{p=1}^{n}\frac{\ln(1-E_{ij}^{p})}{cq_{j}^{p}}$$
$$E_{ij}^{p} = D_{ij}^{p} / R_{ij}^{p} \qquad (5\text{-}20)$$
$$R_{ij}^{p} = \sum_{l\geq j} D_{il}^{p}$$

品質計画策定時に用いる個別プロジェクトの k_{ij} は，k_{ij}^{0} に個々のプロジェクト特性（問題の難しさ，テストあるいはレビュー技法，要員能力等）を考慮して決定する．

実際には，レビューではそのフェーズ内の混入誤り，テストではV字モデルで対応する設計フェーズ内の混入誤りに対する検知以外では，D_{ij}^{p} の値そのものが小さくまたデータの信頼性も低いため，すべての i と j に対して誤り検知率の実績値 E_{ij}^{p} から k_{ij}^{0} を求めても，信頼性と精度の面に問題がある．このため，以下のような簡略化したパラメータ算出法を導入している．

<u>レビューに関する簡略化</u>

レビューにおける誤り検知効率係数の組織能力値としては，(5-20)式を使って k_{jj}^{0} のみを求めておく．品質計画策定時において， k_{jj}^{0} を参考に k_{jj} を決定し，その他の k_{ij} は，k_{jj} の値を使って次式で計算するものとした．

$$k_{ij} = k_{jj} \times \alpha_{ij} \qquad (5\text{-}21)$$

ここで， α_{ij} は フェーズ i で混入した誤りに対してフェーズ j のレビューを実施した場合の減衰率($0 \leqq \alpha_{ij} \leqq 1$)である．これは，あるフェーズ j のレビューは，より上流フェーズ i の混入誤り量に対して検知効率が定数倍減衰するという仮定に立った簡易化である．α_{ij} としては，複数の有識者の意見に基づく経験値を利用している． α_{ij} は組織定数値として与えた上で，品質計画策定時に設定できる数値とした．

<u>テストに関する簡略化</u>

ソフトウェア・ライフサイクル・プロセス[47]に従い，テストと設計とのフェーズ対応関係を，単体テストは詳細設計とコーディング，結合テストは方式設計，適格性確認テストと受入れテストは要求分析に対応づける．このときテストフェーズ j に対する設計フェーズを j'とする．テストの誤り検知効率係数の組織能力値としては，(5-20)式を使って $k_{j'j'}^{0}$ のみを求めておく．品質計画策定時において， $k_{j'j'}^{0}$ を参考に $k_{j'j}$ を決定し，その他の k_{ij} は，$k_{j'j}$ の値を使って次式で計算するものとした．

$$k_{ij} = \begin{cases} 0 & (i < j') \\ k_{jj} & (i = j') \\ k_{jj} \times \beta_{ij} & (i > j') \end{cases} \qquad (5\text{-}22)$$

ここで，β_{ij} はフェーズ i で混入した誤りに対してフェーズ j のテストを実施した場合の減衰率($0 \leqq \beta_{ij} \leqq 1$)である．(5-22)式の簡易化は，テストが V 字モデルで対応する設計フェーズ内の混入誤りに対して最も検知効率がよく，それ以前のテストでは上流設計での混入誤りを検知ことが困難であり，それ以降のテストが下流設計の混入誤りも検知できるという性質を考慮している．β_{ij} としては，複数の有識者の意見に基づく経験値を利用している．β_{ij} は組織定数値として与えた上で，品質計画策定時に設定できる数値とした．

(5-21)(5-22)式による簡略化の効果は，レビュー及びテストの誤り検知率の実績値として比較的データの質が良い E_{jj}^{p} 及び $E_{jj}{}^{p}$ を利用することによって，推定するパラメータ数を減らすことができること，個々のプロジェクトの品質計画策定時に，誤り検知効率係数を向上させる施策を考える場合，混入フェーズ i に対して，本来焦点をあてるべき同一フェーズのレビュー，及びそのフェーズと対応するテストフェーズでのテストに，施策を集中できることが挙げられる．

フェーズ j における品質プロセス密度 cq_j は，規模あたりの作業コスト[Mh/KSLOC]の次元を持っている．便宜上，テストについての品質計画策定時に，直接設定する値として，規模あたりのテスト項目数 t_j [項目/KSLOC]を用い（これを**テスト密度**と呼ぶ），テスト密度にテスト項目あたりの作業コスト tc_j[Mh/項目]をかけることによって，品質プロセス密度の次元に合わせるようにした．レビューの場合には，レビューにかける規模あたりの作業コスト（**レビュー密度**と呼ぶ）を直接用いている．

リワークコストの見積りモデル

(1) モデル要件

誤り量から各フェーズのリワークコストを見積るモデルを定式化する．これは，混入フェーズ i，検知フェーズ j における検知誤り量 D_{ij} が与えられたとき，リワークコスト Cr_j を推定する関数 g を決定することである．

$$Cr_j = g(D_{1j}, D_{2j}, .., D_{Nj}) \qquad (5\text{-}23)$$

ここで，N は誤り混入フェーズの最後のフェーズとする(表 5-1 では「コーディング」フェーズで N=4)．Cr_j が決定できれば，開発リワークコスト Cr はすべての開発フェーズについて Cr_j を合計したもの，出荷後ロスコスト C_o はフェーズ j を M（出荷後：表 5-1 では M=9)であるとしたときの Cr_j（$=Cr_M$ ）として計算することができる．

誤りの原因分析にかかるコストは検知フェーズに，誤りの修正にかかるコストは誤り混入フェーズに強く依存することを考慮して，以下のモデル要件をおく．

要件： 誤り１件あたりのリワークコストは，その誤りの混入フェーズ及び検知フェーズに依存して異なる統計値をもつ.

(2) 採用モデル

我々は，上記の要件を満たすモデルとして，以下を採用した.

$$Cr_j = \sum_{i=1}^{N} Cr_{ij}$$
$$Cr_{ij} = cr_{ij} \times D_{ij} \qquad (5\text{-}24)$$

ここで，cr_{ij} は誤り混入フェーズ i，検知フェーズ j における誤り 1 件当りのリワークコスト [Mh/件]の見積り値である．(5-24)式で表される採用モデルは，誤り種別に誤りデータマトリクス以上の分解能をもたない場合において，上記要件を満たす最も単純なモデルである.

誤り 1 件の修正に要するリワークコストは，統計的なばらつきが大きい．しかし，開発規模が大きくなるほど検知件数 D_{ij} が増えるので，大数の法則により，開発規模が増大するに従い，リワークコストの総和はその推定値 Cr_{ij} の周りでばらつき度合が小さくなる．このため，品質計画策定時においてプロジェクト全体のコスト見積りに利用する範囲であれば，推定値 Cr_{ij} は十分な精度を得られると考える.

(3) パラメータ算出

(5-24)式の Cr_j の計算に必要なパラメータは cr_{ij} である.

cr_{ij} の組織能力値 cr_{ij}^0 は，組織内の複数プロジェクトで，混入フェーズ i，検知フェーズ j の誤り１件に対するリワークコスト実績値 cr_{ij}^p を収集し，それらを平均して求める.

表 5-2　誤り１件あたりのリワークコストの組織能力値例

検知フェーズ j ＼ 混入フェーズ i		ソフトウェア要求分析 1	ソフトウェア方式設計 2	ソフトウェア詳細設計 3	コーディング 4
ソフトウェア要求分析（レビュー）	1	cr_{11}^0 0.00			
ソフトウェア方式設計（レビュー）	2	cr_{12}^0 0.40	cr_{22}^0 0.00		
ソフトウェア詳細設計（レビュー）	3	cr_{13}^0 1.60	cr_{23}^0 0.80	cr_{33}^0 0.00	
コーディング（レビュー）	4	cr_{14}^0 3.20	cr_{24}^0 1.60	cr_{34}^0 0.30	cr_{44}^0 0.00
単体テスト	5	cr_{15}^0 3.30	cr_{25}^0 1.70	cr_{35}^0 0.50	cr_{45}^0 0.10
ソフトウェア結合テスト	6	cr_{16}^0 9.60	cr_{26}^0 6.00	cr_{36}^0 2.50	cr_{46}^0 2.00
ソフトウェア適格性確認テスト	7	cr_{17}^0 29.00	cr_{27}^0 13.00	cr_{37}^0 7.00	cr_{47}^0 5.00
受け入れテスト	8	cr_{18}^0 90.00	cr_{28}^0 28.00	cr_{38}^0 20.00	cr_{48}^0 16.00
出荷後	9	cr_{19}^0 200.00	cr_{29}^0 100.00	cr_{39}^0 70.00	cr_{49}^0 50.00

品質計画策定時における個別のプロジェクトの cr_{ij} の値には，cr_{ij}^{0} をそのまま用いている． 一般に cr_{ij} は，開発チームの能力，対象ソフトウェア種別など多くの要因によって影響を受ける[43]. そのため，実績値とともにそうした要因をデータとして蓄積して，組織に合う形で提供していくことが望ましい．例えば，cr_{ij} として組織平均値を採用する代わりに，同一製品群のプロジェクトの平均値を使う方法もある．また，リワークプロセス自体の生産性向上施策を計画する場合には，cr_{ij} を固定値とせず調整可能なパラメータとすることも考えるべきであろう．

表 5-2 は，組織能力値としての cr_{ij}^{0} を格納したマトリクスの例である．例えば，ソフトウェア要求分析フェーズ(i=1)で混入した誤りがソフトウェア結合テストフェーズ(j=6)で検知された場合， 1 件あたり 9.6Mh のリワークコスト（cr_{16}^{0}）がかかると読む．

モデル評価

提案モデルに関して，V 字モデルへの適合性，モデルとパラメータ算出法の組み合わせの実装可能性，及び定量的な基盤としての適切性の 3 点について評価を行った．

(1) V 字モデルへの適合性

受注開発型ソフトウェア開発においては，V 字モデルが品質保証の枠組みとしてよく使用される．図 5-10 に示すように，V 字モデルでは，レビューとテストを品質保証の観点から，以下のように位置づけている．

- 各設計フェーズのレビューは，そのフェーズの設計が上位の設計の要求事項を満足していることを確認する．
- 各テストフェーズは，対応する設計フェーズの設計を満足していることを確認する．

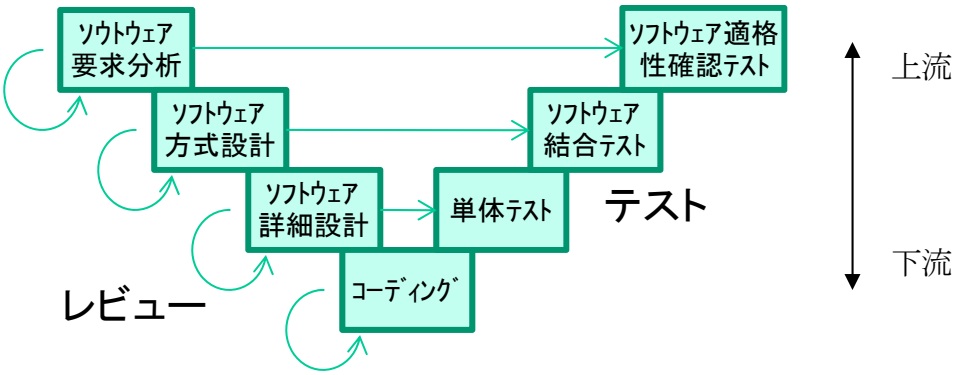


図 5-10　V字モデルにおけるレビューとテストの位置づけ

V 字モデルに従った開発では，誤り検知効率は，以下のような性質をもつ．

① あるフェーズのレビューは、本来そのフェーズで混入した誤りを検知する効率がよい。下流のレビューは，より上流で混入した誤りを検知する効率が落ちる．
② あるテストフェーズは対応する設計フェーズの誤りを検知する効率がよい．
③ 下流テストは上流で混入した誤りを検知することが困難である．
④ 上流テストは、下流で混入した誤りを検知することもできる．

表 5-3 は，提案モデルで用いる誤り検知効率係数のデータである． 提案モデルでは，誤りが混入したフェーズ別に，検知フェーズ毎の誤り検知効率を設定できる．V 字モデルへの適合は，この係数表に組織能力値を設定することで表現できる．すなわち，係数表では，①レビューの検知効率が混入フェーズ以降のフェーズで減衰する，②テストの検知効率は対応する設計フェーズの誤りに対して最大で，③それ以前は検知効率が 0，④それ以後は相応の検知効率をもつこと，などを表現することができる．

表 5-3　誤り検知効率係数表

検知フェーズ j ＼ 混入フェーズ i	要求分析 1	方式設計 2	詳細設計 3	ｺｰﾃﾞｨﾝｸﾞ 4
要求分析ﾚﾋﾞｭｰ 1	k_{11}			
方式設計ﾚﾋﾞｭｰ 2	k_{12}	k_{22}		
詳細設計ﾚﾋﾞｭｰ 3	k_{13}	k_{23}	k_{33}	
ｺｰﾃﾞｨﾝｸﾞﾚﾋﾞｭｰ 4	k_{14}	k_{24}	k_{34}	k_{44}
単体ﾃｽﾄ 5	$k_{15}=0$	$k_{25}=0$	k_{35}	k_{45}
結合ﾃｽﾄ 6	$k_{16}=0$	k_{26}	k_{36}	k_{46}
適格性確認ﾃｽﾄ 7	k_{17}	k_{27}	k_{37}	k_{47}
受入れﾃｽﾄ 8	k_{18}	k_{28}	k_{38}	k_{48}

(2) モデルとパラメータ算出法の組み合わせの実装可能性

提案モデルは，プロジェクト毎に，各フェーズに投入した品質プロセスコスト，及び誤り 1 件に対して以下の記録を収集することで，パラメータを算出することが可能である．

・誤りが検出されたフェーズ
・誤りが混入したフェーズ
・誤りの修正にかかった作業コスト

これらは，誤り総量管理モデルが収集しているのと同じデータである．つまり，提案モデルは, そのパラメータ算出に, 従来モデルと同じデータ収集コストしか必要とせず，実装可能性が保証されていると言える．

(3) 定量的な基盤としての適切性

開発品質コスト C_{DQ} と製品品質コスト C_{AQ} の最小化はどのように達成されるかを，提案見積りモデルを用いてシミュレーションを実施することで確認した．

図 5-11 は，コードレビュー密度を変化させたときの C_{DQ} 及び C_{AQ} の変化を示している．他のフェーズのレビュー密度及びテスト密度は，誤り検知率がレビュー80%，テスト90%以上になるように設定した．

図 5-11 において興味深いことは，C_{DQ} と C_{AQ} の最適点に大きな差異があることである．C_{DQ} の最適点は 3.0[Mh/KSLOC]，C_{AQ} は 8.0[Mh/KSLOC]の付近に現れている．PMにとって，技術プロセスコストを確保し成果物の作成を完了することが最重要ならば，開発品質コスト C_{DQ} の最適点で最善の選択となる．しかし，その選択では，製品品質コスト C_{AQ} における出荷後ロスコスト C_0 が増大し，計画コストの 50%近くまで増えてしまう．製品品質コスト C_{AQ} の最適点に到達しようとすれば，開発品質コスト C_{DQ} の計画コスト比率で 2～3%程度の上昇を受け入れなければならない．

この最適点の差異は，PMが低めの品質プロセス密度を受け入れてしまう原因の一つと考えられる．

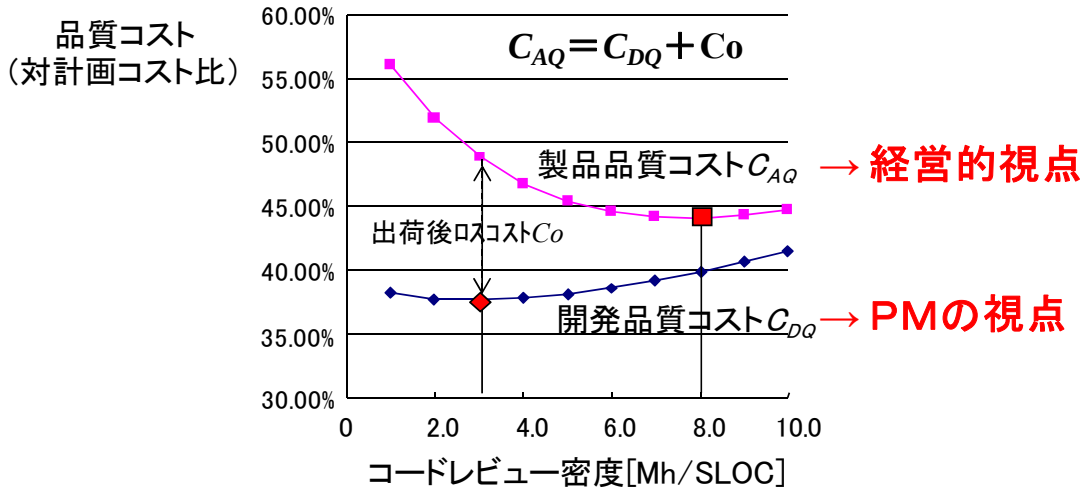


図 5-11　コードレビュー密度に対する開発品質コスト及び製品品質コストの変化

図 5-12 は，単体テストの品質プロセスが，コードレビュー密度を変化させたときの製品品質コスト C_{AQ} のグラフである．図 5-12 において，N は単体テスト密度であり，25, 35, 45[項目/KSLOC]（誤り検知率は，それぞれ 79.8%，89.3%，94.4%）の３つのケースで，C_{AQ} のグラフを描いている．

図 5-12 で，単体テスト密度が高いほど，C_{AQ} のコスト最適点が左に移動し，かつグラフが横に寝た形状となる．N=45 では，コードレビュー密度 3～10[Mh/KSLOC]の範囲で，

C_{AQ}の計画コストに占める比率の差が 2%以内にとどまっている．シミュレーション結果は，単体テスト密度が高い状況では，コードレビュー密度が製品品質コストC_{AQ}に与える影響は顕著ではなく，逆に低い状況では，影響が顕著となることを示している．言い換えると，単体テスト密度が高い場合，コードレビュー密度の増減は製品品質コストに大きな影響を与えないが，逆に単体テスト密度が低い場合，コードレビューを怠ると製品品質コストが急激に増大する．この現象は，コードレビューと単体テストの相補的な関係を示す報告[38]に一致するものである．V 字モデルの他の対応フェーズ間については，コードレビューと単体テストほど顕著ではないが，同様の傾向を確認した．

2 つのシミュレーション結果は，提案モデルが，品質プロセスのコスト対効果を評価する上での定量的基盤を与え，かつこのケースでソフトウェア開発現場に存在する状況を的確に表現していることを示している．

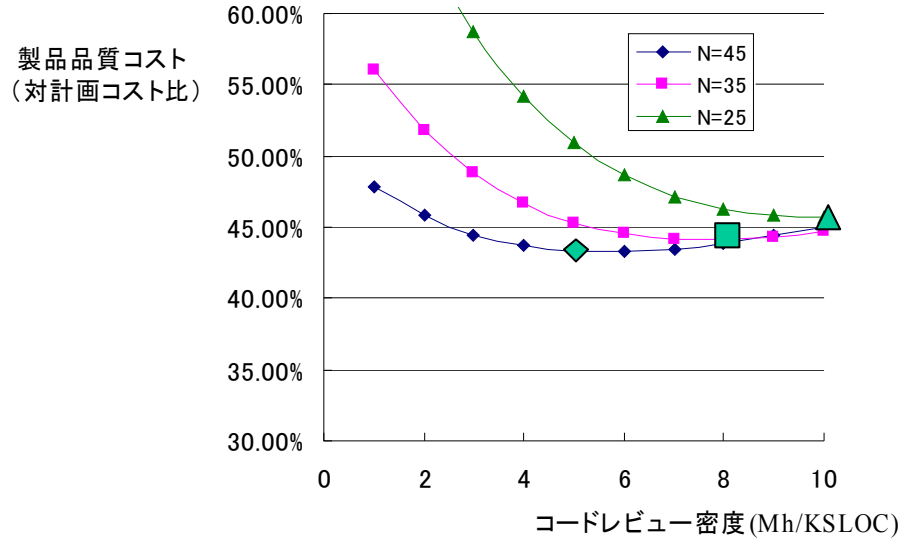


図 5-12　単体テスト密度を変化させたときのコードレビューの製品品質コスト曲線

5.5　実証実験

提案システムを試作し，プロジェクトマネージャが行う品質計画策定作業の支援に適用することで，目標状態の達成を評価する実証実験を行った．

試作システム

5.4 項の提案モデルに基づき，プロジェクト計画のための品質計画支援システムの試作を行った．試作システムは，QualityPlanner と呼ぶ．QualityPlanner は，品質とコストの見積りモデルに基づく施策 1 の計画だけに留まらず，施策 2 及び施策 3 の計画支援機能も実装している．以下，QualityPlanner について記述する．

(1) ユースケース

QualityPlanner のユースケース図を図 5-13 に示す.
ユースケースは以下の２つである.

名称：　品質計画を策定する
アクタ：PM，SQA(ソフトウェア品質保証) [81]
主フロー：１．PM が，開発規模，計画コスト，及びフェーズ別要求変更発生予測を入力する.
２．PM が，品質計画を入力する.
３．システムが，品質とコストの推定値を出力する.
４．PM が，品質とコストの推定値から，品質計画の妥当性を確認する.
５．SQA が，品質計画を品質基準値の遵守性の面から確認する.

名称：　組織値を更新する
アクタ：SEPG(ソフトウェアエンジニアリングプロセスグループ)[81]，SQA
主フロー：１．SEPG が，プロセス実績 DB より，組織能力値を更新する.
２．SEPG と SQA が，組織目標と実績に基づき，品質基準値を更新する.

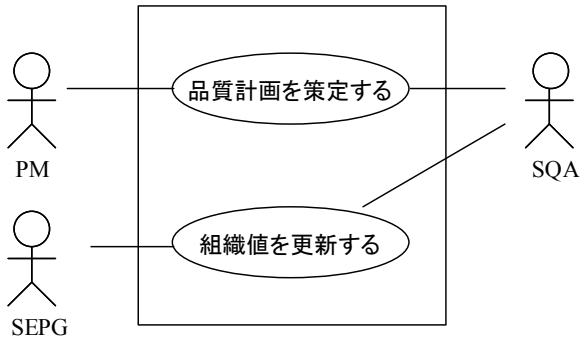


図 5-13　QualityPlannerのユースケース図

ユースケース「組織値を更新する」では，QualrityPlanner がもつ組織能力値と品質基準値を更新する．組織能力値は，組織の標準的な作業能率を表し，QualityPlanner の設定パラメータのデフォルト値となる．組織能力値は，5.4.4 及び 5.4.5 の(3)パラメータ算出法により求める．品質基準値は，組織の実績値をベースにして，組織としての改善事項を織り込んで設定する．QualityPlanner の組織能力値及び品質基準値には以下がある.

組織能力値：

- （フェーズ別）標準混入誤り密度，標準テスト時間効率，標準総コスト比率，標準ドキュメント・コード規模比
- （混入・検知フェーズ別）標準レビュー誤り検知効率係数（あるいは標準テスト誤り検知効率係数），誤り 1 件の標準リワークコスト

・標準生産性，標準管理コスト比率

品質基準値：

・（フェーズ別）基準誤り除去率，基準誤り検知率，基準レビュー密度，基準テスト密度

ユースケース「品質計画を策定する」の主フロー3 番目で，QualityPlanner は，開発規模と計画コスト，フェーズ別要求変更発生予測，及び品質計画から，品質とコストの推定値を出力する．プロジェクトの計画コストに対して，開発各フェーズにおいて，品質プロセス密度及び作業能率を設定し，5.4.4 の検知誤り量の見積りモデルを用いて，検知誤り密度の推定する．さらに，各フェーズでの検知誤り量及び出荷後誤り量から，5.4.5 のリワークコストの見積りモデルを用いて，開発リワークコスト及び出荷後ロスコストを推定する．P M は，作業能率に組織能力値をそのまま使うのではなく，プロジェクト特性や施策の効果を加味し，チューニングして用いる．

(2) システム構成

図 5-14 は，QualityPlanner のシステム構成である．

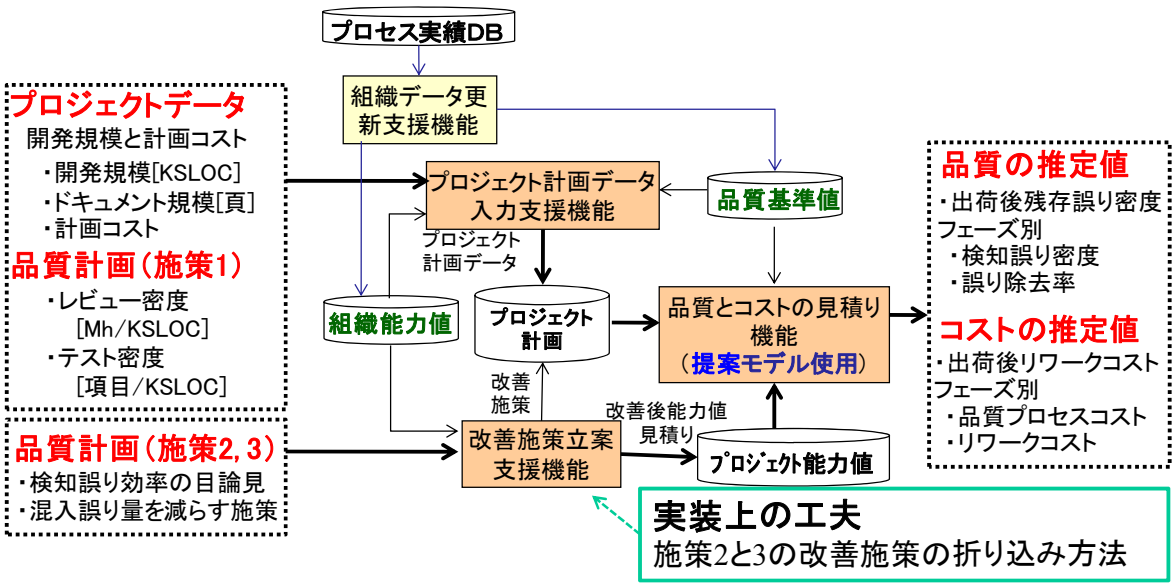


図 5-14　QualityPlannerのシステム構成

ユースケース「組織値を更新する」は，組織データ更新支援機能に対応し，ユースケース「品質計画を策定する」は，プロジェクト計画データ入力支援機能，品質とコストの見積り機能，及び改善施策立案機能が対応する．施策 2 及び施策 3 の計画は，改善施策立案機能が対応する．

(3) 実現方式とツール概観

QualityPlanner は，Microsoft Excel のベーシックな表計算機能を用いて開発しており，次の役割をもつ 3 つのシートから構成されている．図 5-15 に Plan シートの概観を示す．

図 5-15 QualityPlanner のPlanシートの概観

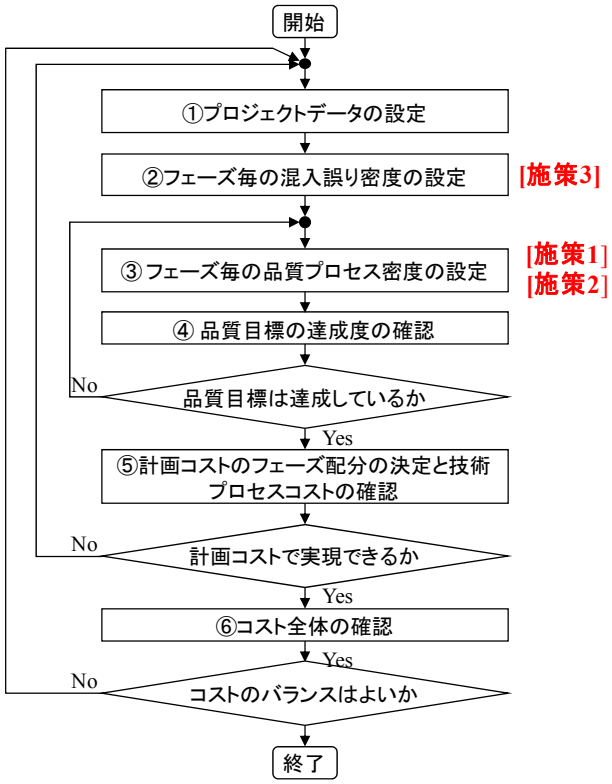


図 5-16　品質計画策定手順

Plan シート:　品質計画策定上の主なパラメータの設定(図 5-17, 図 5-18, 図 5-20, 図 5-23)，及び見積り結果のグラフ表示(図 5-22 図 5-24, 図 5-25)

Design シート: 設計各フェーズにおける混入誤り密度(図 5-19)及びレビュー検知効率係数を決定する上での目論見の設定

Test シート:　テスト各フェーズにおけるテスト検知効率係数及びテスト時間効率を決定する上での目論見の設定

(4) 品質計画策定の手順

図 5-16 は，PM が，QualityPlanner を用いて，施策 1，施策 2，及び施策 3 の計画を行う際の品質計画策定手順である.

品質計画策定手順と支援機能

図 5-16 中に現れる 6 つのステップを順に説明し, そのプロセスに従った QualityPlanner の支援機能について記述する.

(1) プロジェクトデータの設定

図 5-17 に示すように、PMは，開発するシステムの規模見積り[KSLOC]と，標準生産性と標準管理コスト比率を参考にしながら, 計画コスト及び管理コスト比率を設定する.

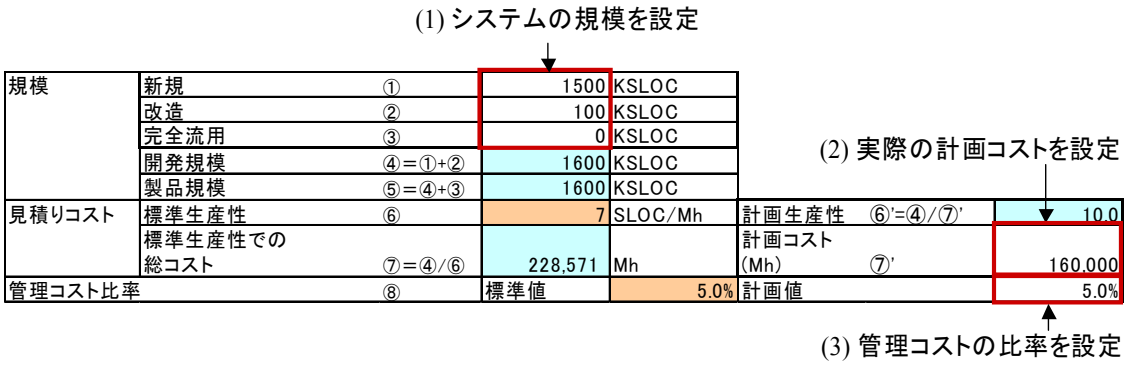
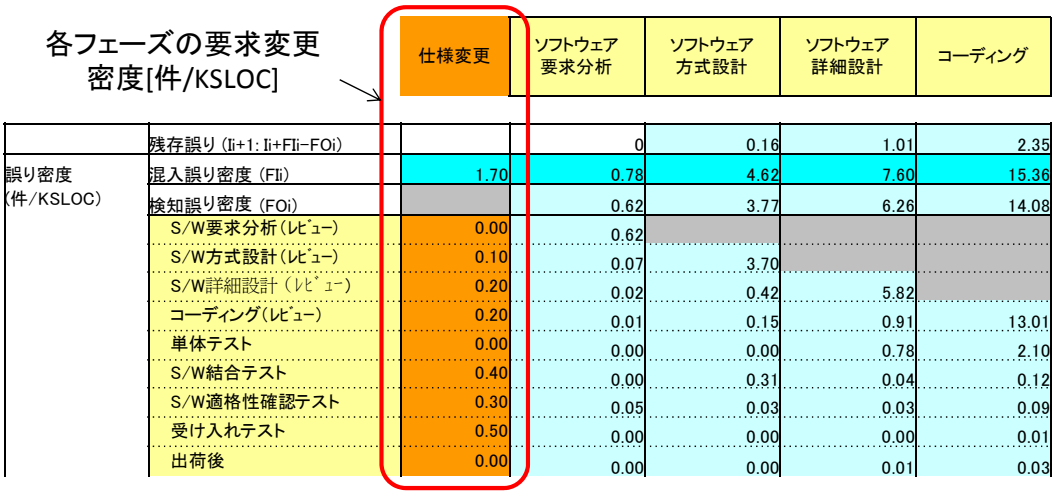


図 5-17　規模とコストの見積りの設定(Planシート)

図 5-18 は，混入誤りの表示画面（Plan シート）である．この画面上では，フェーズ別要求変更発生予測を, 要求変更密度として設定することもできる. シミュレーション上，要求変更密度分の要求変更量が，指定したフェーズで必ず生じるものとして扱い，要求変更 1 件あたりのリワークコストは，検知フェーズ別に設定している要求誤り 1 件に対するものと同じとしている.



		仕様変更	ソフトウェア要求分析	ソフトウェア方式設計	ソフトウェア詳細設計	コーディング
	残存誤り (Ii+1: Ii+FIi−FOi)		0	0.16	1.01	2.35
誤り密度 (件/KSLOC)	混入誤り密度 (FIi)	1.70	0.78	4.62	7.60	15.36
	検知誤り密度 (FOi)		0.62	3.77	6.26	14.08
	S/W要求分析(ﾚﾋﾞｭｰ)	0.00	0.62			
	S/W方式設計(ﾚﾋﾞｭｰ)	0.10	0.07	3.70		
	S/W詳細設計(ﾚﾋﾞｭｰ)	0.20	0.02	0.42	5.82	
	コーディング(ﾚﾋﾞｭｰ)	0.20	0.01	0.15	0.91	13.01
	単体テスト	0.00	0.00	0.00	0.78	2.10
	S/W結合テスト	0.40	0.00	0.31	0.04	0.12
	S/W適格性確認テスト	0.30	0.05	0.03	0.03	0.09
	受け入れテスト	0.50	0.00	0.00	0.00	0.01
	出荷後	0.00	0.00	0.00	0.01	0.03

図 5-18　混入誤りの表示画面（Planシート）とフェーズ別要求変更密度の設定

(2) フェーズ毎の混入誤り密度の設定

PM は，標準フェーズ別混入誤り密度を参考にして，プロジェクト特性や施策の効果を加味して，フェーズ別混入誤り密度の目論見値を設定する．

図 5-19 は要求分析フェーズに関する目論見の設定例である．図 5-19 では，混入誤り密度の組織能力値 $r_{II}{}^{0}$ を 1.19 件/KSLOC と表示している．その上部に，シートの別の部分で設定した規模あたりの要求仕様書の頁数を用いて換算した，頁あたりの混入誤り密度（0.60 件/頁）を表示している．PM は，この値を参考にして，目論見として 1 頁あたりの混入誤り件数を設定する．この例ではそのままコピーして用いている．

次にその値に影響を与える要因として，プロジェクト特性やこれから実施する品質向上施策をリストアップし，その影響度を設定する．この例では，プロジェクト特性による増大要因として，流用の悪影響及び要員のスキル，品質向上施策による減少要因として，類似システムの経験及び UML の利用・教育を設定し，それぞれ 20%,20%,-50%,-10% の係数を与えている．これらの影響を掛け合わせて 0.39 件/頁及び，このプロジェクトで採用する混入誤り密度 r_{II} を 0.78 件/KSLOC と計算している．

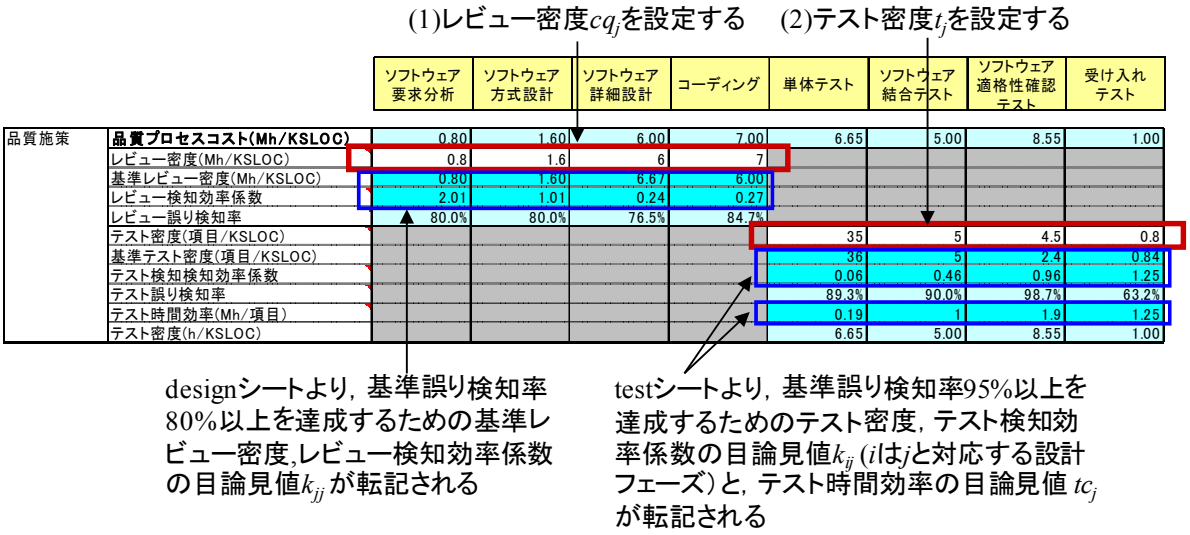
項目	組織能力値		目論見		
要求分析	混入誤り密度	0.60 件/頁	0.39 件/頁	1 頁中の誤り	0.60 件
	(新規)	1.19 件/kline	0.78 件/kline	流用の悪影響	20%
		$r_{II}{}^{0}$	r_{II}	要員のスキル	20%
				類似システムの経験	−50%
				UMLの利用・教育	−10%

図 5-19　要求分析中の混入誤り密度の設定と目論見(Designシート)

(3) フェーズ毎の品質プロセス密度の設定

図 5-20 に示すように，QualityPlanner は，目標とする基準誤り検知率(この例ではレビューは 80%，テストは 95%)と，それを達成するためのレビュー密度及びテスト密度を提示する．

PM は，これらを参考に，システムの特性と開発チームの実力を加味して，フェーズ毎のレビュー密度（図 5-20 の 2 行目）及びテスト密度（図 5-20 の 6 行目）を設定する．PM は，レビューについては design シート，テストについては test シートを用いて，組織能力値をベースに，プロジェクト特性及び計画した施策の効果を加味して，誤り検知効率係数やテスト時間効率[Mh/項目]の目論見値を設定する．



		ソフトウェア要求分析	ソフトウェア方式設計	ソフトウェア詳細設計	コーディング	単体テスト	ソフトウェア結合テスト	ソフトウェア適格性確認テスト	受け入れテスト
品質施策	品質プロセスコスト(Mh/KSLOC)	0.80	1.60	6.00	7.00	6.65	5.00	8.55	1.00
	レビュー密度(Mh/KSLOC)	0.8	1.6	6	7				
	基準レビュー密度(Mh/KSLOC)	0.80	1.60	6.67	6.00				
	レビュー検知効率係数	2.01	1.01	0.24	0.27				
	レビュー誤り検知率	80.0%	80.0%	76.5%	84.7%				
	テスト密度(項目/KSLOC)					35	5	4.5	0.8
	基準テスト密度(項目/KSLOC)					36	5	2.4	0.84
	テスト検知検知効率係数					0.06	0.46	0.96	1.25
	テスト誤り検知率					89.3%	90.0%	98.7%	63.2%
	テスト時間効率(Mh/項目)					0.19	1	1.9	1.25
	テスト密度(h/KSLOC)					6.65	5.00	8.55	1.00

図 5-20　フェーズ毎の品質プロセスの設定(Planシート)

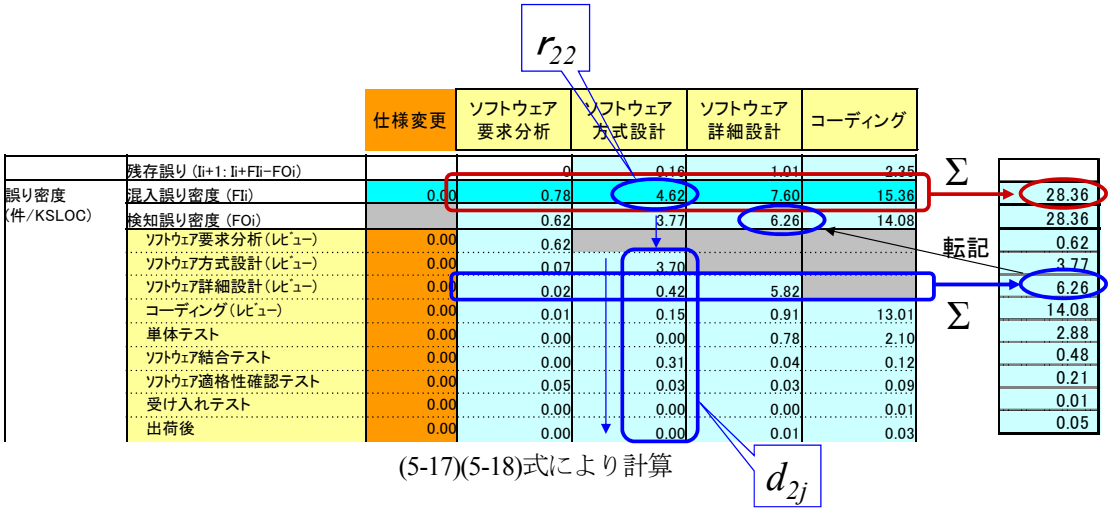


図 5-21　フェーズ毎混入誤り密度の表示(Planシート)

図 5-19 及び図 5-20 の設定を行うと，QualityPlanner は，図 5-21 に示す誤りデータマトリクスに，フェーズ毎混入誤り密度の目論見値を転記し，検知フェーズ j における流入誤り密度 r_{ij} 及び検知誤り密度 d_{ij} の推定値を自動計算する．

(4) 品質目標の達成度の確認

QyalityPlanner は，(1)-(3)の計画値の設定から，図 5-22 に示すように，各フェーズの検知誤り密度及び誤り除去率をグラフ表示する．

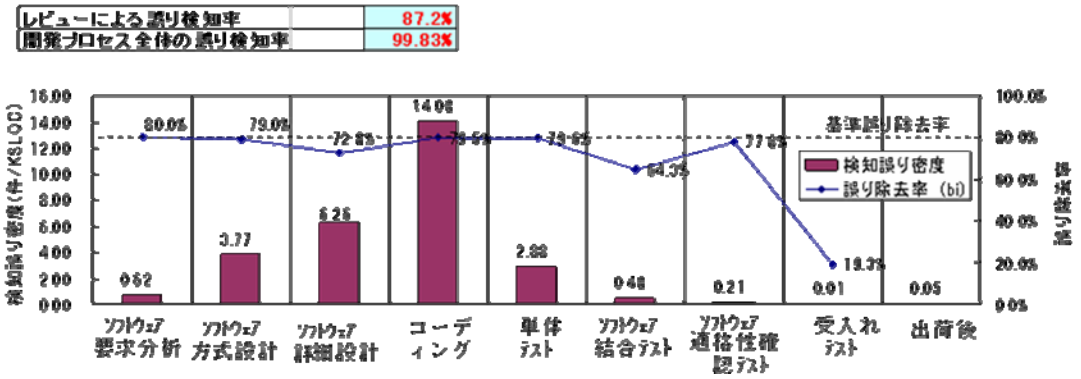


図 5-22　品質目標の達成度の確認(Planシート)

PM は，このグラフを見て，(1)-(3)で設定した計画値が，最終的な製品品質目標である出荷後残存誤り密度を達成するのに十分であるかを確認する．また，各フェーズの誤り除去率が，品質基準値である基準誤り除去率（図 5-22 では 80%に点線表示されている）を満たしているかも確認する．いずれかが満たされていない場合には，(3)での品質プロセスコストの設定を再調整する．

計画コストのフェーズ配分を設定

		ソフトウェア要求分析	ソフトウェア方式設計	ソフトウェア詳細設計	コーディング	単体テスト	ソフトウェア結合テスト	ソフトウェア適格性確認テスト	受け入れテスト
計画コスト	コスト比率(比率配分を入力)	1	2.5	4.5	7	2.5	1.2	1	0.5
		5.0%	12.4%	22.3%	34.7%	12.4%	5.9%	5.0%	2.5%
	標準総コスト比率	10.0%	20.0%	20.0%	20.0%	10.0%	10.0%	5.0%	5.0%
	①フェーズ別コスト(総コストを配分)	7,921	19,802	35,644	55,446	19,802	9,505	7,921	3,960
管理	②管理コスト(Mh)　①×管理配分	396	990	1,782	2,772	990	475	396	198
品質プロセス	③品質プロセスコスト(Mh)	1,280	2,560	9,600	11,200	10,640	8,000	13,680	1,600
リワーク	④リワークコスト(Mh)	0	48	600	881	964	3,560	4,004	365
技術プロセス	⑤技術プロセスコスト ①-(②+③+④)	6,245	16,204	23,661	40,592	7,208	0	0	1,798
	標準作成ドキュメント頁数	3,200 頁	6,400 頁	32,000 頁					
	作成ドキュメント頁数 A	3,200 頁	6,400 頁	32,000 頁					
	純粋作成効率 A/④	0.51 頁/h	0.39 頁/h	1.35 頁/h	39.4 line/h				
累計	計画コスト累積(Mh)	7,921	27,723	63,366	118,812	138,614	148,119	156,040	160,000
	リワークコスト累積(Mh)	0	48	648	1,529	2,493	6,053	10,057	10,422
	品質プロセスコスト累積(Mh)	1,280	3,840	13,440	24,640	35,280	43,280	56,960	58,560

ドキュメント頁数の入力と作業効率の確認

図 5-23　計画コストのフェーズ配分の設定と技術プロセスコストの確認(Planシート)

(5) 計画コストのフェーズ配分の決定と技術プロセスコストの確認

PM は，(1)で決定した投入コストを，標準総コスト比率を参考に各フェーズに振り分ける．図 5-23 の例では，3 行目が標準コスト比率を参考に，1 行目でコスト比 1, 2.5, 4.5, …, 0.5 と入力することで，2 行目のフェーズ毎のコスト比率が決定される．次に目論見で設定するドキュメント・コード比より計算した標準作成ドキュメント頁数を参考に，作成するドキュメント頁数の計画値 A を設定する．図 5-23 の例では，10 行目で計画値 A に標準値をそのままコピーして用いている．

(5-3)式の作業コスト C_i を，計画コスト Cp_i から管理コストを引いたものとすると，技術プロセスコスト Cd_i は，以下のように計算できる．

$$Cd_i = Cp_i \times (1 - \gamma_m) - Cq_i - Cr_i \qquad (5\text{-}25)$$

ここで，γ_m は図 5-17 で設定した管理コスト比率である．

リワークコストは(5-24)式を用いて計算される．図 5-23 において，技術プロセスの生産効率として，ドキュメント作成効率及びソースコード作成効率が計算されるので，技術プロセスに割くコストが十分であるかどうかを確認する．

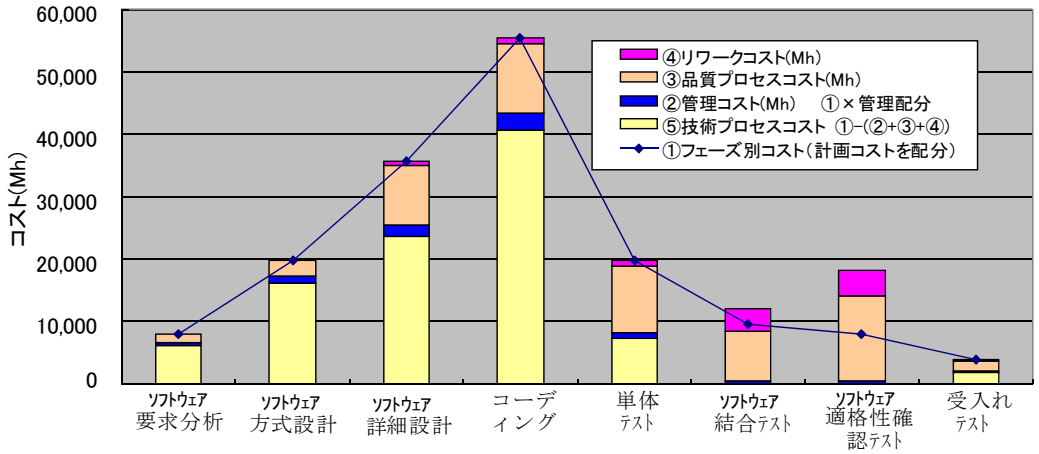


図 5-24　フェーズ毎コスト配分(Planシート)

QualityPlanner は，図 5-24 に示すように，フェーズ別に，技術プロセスコスト，管理コスト，品質プロセスコスト，及びリワークコストの比率をグラフ表示する．ＰMは，このグラフを見て，計画コストのフェーズ配分及び各フェーズの品質プロセスコストのコスト的な妥当性を確認する．図 5-24 で折れ線がフェーズ別計画コストである．リワークコストが増え過ぎると，積み上げた開発コストが計画コストの折れ線を越える場合が

ある．この場合には，計画上の矛盾と捉え(1)-(5)の見直しを行う．

(6) コスト全体のバランスの確認

各フェーズの品質プロセスコストを変えることによって，計画コスト内の各コスト種別のバランス，及び出荷後ロスコストが変化する．図 5-25 の左の棒グラフは，計画コスト内に占める，開発リワークコスト，品質プロセスコスト，管理コスト，及び技術プロセスコストの比率，右の棒グラフは計画コストに対する出荷後ロスコストの推定値を示している．技術プロセスコスト *Cd* は，開発フェーズ全体に関して(5-25)式の Cd_i を総計したものである．技術プロセスコストが実施可能な範囲を下回っているならば，開発コストは計画コストを超過する可能性が高くなる．

図 5-25 を見て，PM は，次の２つの視点から品質プロセスの計画を評価する．

① 開発品質コスト C_{DQ} の最小化

計画コスト一定の条件下において，(5-5)式の開発品質コスト $C_{DQ}(=Cq+Cr)$ を最小化することで，技術プロセスコスト *Cd* を最大化する視点である．

② 製品品質コスト C_{AQ} の最小化

技術プロセスコスト *Cd* 一定の条件下において，(5-6)式の製品品質コスト $C_{AQ}(=Cq+Cr+C_o)$ を最小化することで，製品コスト C_A を最小化する視点である．

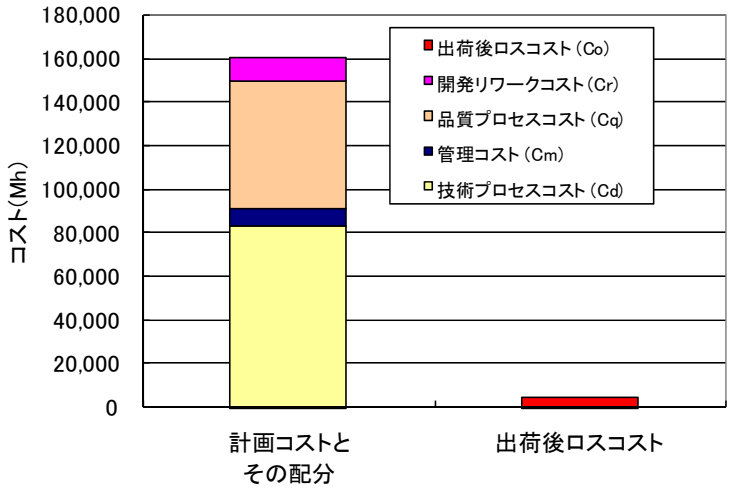


図 5-25　コスト全体の確認(Planシート)

適用評価

(1) 適用評価の結果と分析

開発規模 50KSLOC 以上の防衛関連システム 5 プロジェクトを対象に，次の適用・評価を実施した．従来から組織内で利用してきた誤り総量管理モデルの管理手順に従い，

PM がプロジェクト管理計画書作成時に設定した品質関連の計画値を対象とした．計画値の設定方法は以下の通りである．

① 出荷後残存誤り密度：　各プロジェクトの目標値を設定
② 混入誤り密度：　組織能力値を調整せず設定
③ 誤り検知効率：　検知フェーズでの固定値[件/KSLOC・Mh]（組織提供）を調整せず設定
④ 品質プロセス密度：　各フェーズの下限値として設定されている品質基準値に基づき，①の目標値を満たすように調整して設定

これらの計画値の内①②④を QualityPlanner に入力した．この際，QualityPlanner がドキュメント作成効率及びコーディング効率から計算した各設計フェーズに必要な技術プロセスコストに，計画コストを優先的に割り当てるようにした．

表 5-4 は，各プロジェクトの開発規模と，提案システムで再評価した結果としての計画コストに対する開発コストの見積り値の超過率を示している．表 5-4 で，A-D すべてのプロジェクトで超過している．これは，レビューと単体テストの比率を低く抑え，結合テストに比重を置いた設定をしていたためである．

表 5-4　5 つの適用プロジェクトの開発規模と計画コスト超過率

ﾌﾟﾛｼﾞｪｸﾄ	A	B	C	D	E
開発規模[KSLOC]	350	170	110	92	76
計画コスト超過率	+11%	+33%	+49%	+19%	+13%

次に，各プロジェクトの PM を支援し QualityPlanner 用いて，品質計画の再策定を行った．

まず，施策 1 に従い，かつコスト最適化を求めるように，各フェーズにおける品質プロセス密度の再設定を行った．その結果，A-E すべてのプロジェクトについて，レビューによる誤り除去率が 80%以上に変化した．

次に，設計品質向上の施策 2 と，レビュー効率向上の施策 3 を，それぞれ QualityPlanner 上の目論見として計画した．この際，施策効果が現実的でない計画とならないように，施策種別毎にその改善効果が，組織能力値の標準偏差の範囲に収まるべく，以下の限度幅を設定した．

- 設計品質向上施策：混入誤り密度で-50%
- レビュー効率向上施策：誤り検知効率係数で+40%
- 生産性向上施策：各ドキュメント作成効率とコーディング効率を+10%

結果として， C を除く 4 つのプロジェクトで，上記の施策効果の限度幅内で計画コスト内に入る計画，すなわち，コスト的に整合する技術プロセス及び品質プロセスへのコ

スト配分計画と，その達成のために必要な生産性向上施策及び品質向上施策，を立案することができた.

この適用結果により，QualityPlanner が，プロジェクト管理計画における品質とコストの計画間の整合性確認，及び品質マネジメントの基本戦略に基づく合理的な施策立案の支援に効果があることが確認できた.

(2) 定性的評価

上記適用プロジェクトの PM から，ツールの導入上の課題として，以下の 2 点の指摘を受けた.

- プロジェクトの特性と改善効果を加味した目論見を作成するためには，ツールの背後にある品質とコストの関係の理解が必須であるので，その習得が容易な PM 教育がほしい
- ツールが，品質と開発期間の直接的な関係を明確にする機能を持たないため，計画において開発期間の短縮化を計画する場合には，別の視点が必要となる.

SQA 担当者からは,「PM 自らが納得する合理的な品質計画を策定できることは，組織の品質保証の観点からみて望ましい.」との意見があった．SQA 部門が，画一的な品質基準値セットを提供し，開発のホールドポイントでその逸脱を指摘し，一方的に是正処置を求めるだけでは，組織間に軋轢を生み，反って生産活動にプラスに働かないことが多い.「PM 自らがコストと品質を整合させる品質計画を策定し説明し実行するなら，SQA が行う計画逸脱の指摘と是正措置の勧告は遥かに利用価値の高いものとなる.」との意見である.

5.6 本章のまとめ

適用範囲

提案モデルとシステムは，現状，ウォータフォール型の開発モデルにしか対応していない．誤りの総量管理モデル自体は，インクリメンタル型やエボリューショナル型などの他の開発モデル[79]にも適用可能であるので，他の開発モデルへとモデルとシステムを拡張することは可能と考える.

採用モデルは簡略化のため，規模ファクタを見積り開発量 SLOC で表現している．各フェーズでの品質プロセスを考える上で規模ファクタはベースとなるので，若干の考慮が必要である．将来的には，要求分析・方式設計フェーズとその対応テストフェーズではファンクションポイント[80]を，下流の開発フェーズでは SLOC を利用することがよいと考える.

提案モデルとシステムは，品質とコストの関係のみを捉えており，開発期間への影響は表現していない．例えば，リワークコストは個々の誤りに対する平均修正工数を使っており，誤り修正に関するリソース競合（修正担当者，テスト環境等）による待ち時間は考慮していない．開発終盤でのリワークコストの増大は，単純な修正工数の積み上げによる遅れにとどまらず，開発進捗に破壊的な影響を及ぼすことがある．開発期間への

影響を表現するためには，プロセスシミュレーションのアプローチ[77]が必要であると考える．

まとめ

本章は，品質とコスト目標を達成する品質計画策定を支援するために，エンジニアリングプロセスの各フェーズでの品質プロセスの実施コストが，エンジニアリングプロセス全体のコストにどのように影響するかを定量化するための見積りモデル，及びそれを用いた品質計画支援システムを提案し評価した．

提案モデルは，以下の２つのモデルを組み合わせて構成している．

- 検知誤り量の見積りモデル
- リワークコストの見積りモデル

上記２つのモデルは，それぞれ誤りの総量管理モデル及び Kan のモデルをベースにして，その技術的問題点を解決し，かつ実現可能なモデルとパラメータ算出法を提供している．提案モデルを用いたシミュレーションにより，品質プロセスのコスト対効果が定量的に把握できることを示すとともに，ソフトウェア開発現場の状況を的確に表現することを示した．提案モデルを実装した試作システムを開発し，適用評価を行い，品質とコストの整合性確認，及び戦略に基づく合理的品質計画策定を支援できることを確認した．

第6章　　結論と今後の課題

6.1　結論

本研究の結論を以下にまとめる.

本研究は，受注開発型ソフトウェア開発の品質向上のために，ソフトウェア開発ライフサイクルを通して混入する誤りを，時間及びコストの制約下で，確実に除去することを狙いとした技術開発に関するものである．そのために，受注開発型ソフトウェア開発の 3 つのプロセスを対象として，それぞれの誤り除去活動と管理の効率と効果の向上を目指した研究を行った.

第 3 章では，監視・制御システムの要求分析を効果的に支援するプロトタイピングシステムの提案と評価を行った．実証実験を通じて，オンラインサイクルの短縮化に効果があることを確認した.

第 4 章では，小規模組込みシステムの製品検査プロセスを効率化する自動テスト方式を提案し評価した．実証実験を通じて，設定したテスト界面が反応的な仕様を持つプログラムに対応できること，テスト実施作業の大幅な削減が図れることを示した.

第 5 章では，品質とコスト目標を達成する品質計画策定を支援する定量的基盤としての見積りモデルと，そのモデルを用いた品質計画策定支援システムの提案と評価を行った．実証実験を通じて，プロジェクト計画における品質とコストの計画間の整合性確認と，整合性を保つための合理的な施策立案の支援に効果があることを示した.

6.2　今後の課題と展望

本論文第 3 章の研究について，製品分野として，監視・制御システムに特化しているが，情報通信システムについても，製品開発が円熟していくに従い，ビジネス形態が次第にイージーオーダ型開発へ移行していくと考えられるため，本研究の拡張と応用が広まっていく可能性が高い．要求分析プロセスは，製品分野の性質を見極めながら最良のプロトタイピングシステムを考えていく必要がある．また，当該研究は，モデルの妥当性確認はオブジェクト指向要求記述言語の文法チェックに負うところが多いが，モデル検証技術などへの発展が考えられる.

本論文第 4 章の研究について，製品分野として，小規模な組込みシステムに特化している．これは，この分野が反応的な仕様の割合が高く，テスト界面を ICE で設定・参照するメモリに設定したためである．他の分野においても，仕様に基づくテスト自動化を進める場合には，要求仕様記述として何を選択するか，及び適切なテスト界面をどのように設定するか，の２つの点に注意すべきである．例えば，非組込みシステムにおいて反応的な仕様を対象にテスト自動化を行うなら，要求仕様全体に占める反応的な仕様の割合が高い応用分野かどうかを考慮すべきである．変換的仕様や品質仕様からのテスト自動化については，仕様からテストデータへの変換法は未確立である．効果のある仕様範囲を限定して仕様記述法とテストデータへの変換法を考えていくアプローチが必要で

あろう．

本論文第 5 章の研究については，モデリング対象は，エンジニアリングプロセスに限っているが，要求分析プロセス及び製品確認プロセスについても，誤り除去の活動のコスト最適化問題はある．この部分は研究がなされていない領域である．また，提案モデルは，品質計画策定を支援する提案システムに用途が限定されるものではない．レビューのテストに対するコスト対効果を見積ること，フェーズ毎品質プロセスコストの最適点の計算などにも利用できると考えている．

誤り除去のための活動は，個別のプロジェクトがおかれた環境下で，それらをどう組み合わせることが可能か，それによって，プロジェクト全体の QCD にどのような影響を及ぼすかを正しく理解することにより，計画されるべきである．この実現ためには，本研究に加えて，下記の課題を克服していく必要がある．

(1) 開発ライフサイクルのダイナミズム，すなわちウォータフォール以外のライフサイクルでの誤り除去活動の分析
(2) 有用な誤り除去技法・システムに関する経験の蓄積とカタログ化
(3) 不確定性を加味した，高度なプロジェクトシミュレーションへの発展

謝辞

早稲田大学理工学術院創造理工学研究科 東基衞教授には, 本研究をまとめる全過程において親身のご指導と適切なご助言を賜り，さらに更新わたり数々の貴重なご教示，ご激励を戴きました．このことに深く感謝し，心から御礼申し上げます.

本研究をまとめる過程において，数々のご指導とご助言を頂きました大成尚教授，平澤茂一教授，高橋真吾教授に深く感謝するとともに厚く御礼申し上げます.

本論文は，筆者が 1996 年から 1998 年にかけて三菱電機（株）情報技術総合研究所（旧情報電子研究所）において行った研究と，2005 年から 2006 年にかけて鎌倉製作所において行った研究に基づいています．研究の機会を与えてくださった情報技術総合研究所坂下善彦元部長（現在，湘南工科大学教授），鎌倉製作所情報システム部三堀隆元部長，村田芳夫元部長に感謝致します.

本研究を進めるにあたっては，情報電子研究所システム部の皆様のご協力を賜りました．特に，オブジェクト指向仕様に基づくプロトタイピングツールの開発では，上原憲二氏，田村直樹氏，岡田和久氏，状態遷移仕様に基づく自動テストツールの開発では，別所雄三氏，山中弘氏，と共同して研究開発と適用評価を進め，実りある成果をあげることができました. また, コロラド大学コロラドスプリングス校 Alan M. Davis 教授には，2 つの研究の初期のアイデアにご助言を頂きました．品質計画支援システムの開発では，電子システム事業本部 藤山直樹氏，糸谷友良氏，鎌倉製作所の吉田見岳氏，高瀬康弘氏に，研究に対するご協力とご助言を頂きました．心から感謝致します.

本研究をまとめる過程で，様々なご協力を頂いた早稲田大学創造理工学部経営システム工学科東研究室の諸氏に感謝致します. また, 研究のまとめのため時間を割くことを，快く認め応援していただいた生産技術部の皆様，特に，山口隆一副本部長（元部長)，杉山武史部長，太田太次長，佐々木誠主任技師長，江頭誠グループマネージャ，森本幸博グループマネージャに，心より感謝致します.

参考文献

[1] Guide to the Software Engineering Body of Knowledge, IEEE Computer Society (2004).

[2] JIS X 0129-1 ソフトウェア製品の品質　第１部 品質モデル.

[3] IEEE Std. 830-1998 Recommended Practice for Software Requirements Specifications.

[4] ISO 9000:2005 品質マネジメントシステムー基本および用語

[5] 岡本, 鈴木, 野中 : SRS 特性に着目したレビュー手法の評価および現場適用に関する一考察第 24 回ソフトウェア品質シンポジウム 発表報文集, pp.309-316 (2005).

[6] Tsuyoshi Nakajima and Alan M. Davis: Classifying Requirements Errors for Improving SRS Reviews, 1st International Workshop on Requirements Engineering: Foundation for Software Quality (1994).

[7] Hassan Gomaa: The Impact of Prototyping on Software System Engineering, In System and Software Requirements Engineering, R. Thayer and M. Dorfman eds., pp. 543-52, IEEE Computer Press, Washington D.C. (1990).

[8] 古宮 誠一：プロタイピング，情報処理ハンドブック，pp. 719-724 (1995).

[9] Statemate 4.0: Prototyper User and Reference Manual, i-Logix (1991).

[10] David Harel and Eran Gery: Executable Object Modeling with Statecharts, In Proc. of ICSE18, pp. 246-257, (1996).

[11] Bran Selic: Real-Time Object-Oriented Modeling (ROOM), Proceedings of the 2nd IEEE Real-Time Technology and Applications Symposium (1996).

[12] Giorgio Bruno and Giuseppe Marchetto: Process-Translatable Petrinets for the Rapid Prototyping of Process Control System, IEEE Transactions on Software Engineering, Vol. 12, No. 12, pp. 346-357 (1986).

[13] Alistair Sutcliffe: Scenario-based Requirements Engineering, 11th IEEE International Requirements Engineering Conference (2003).

[14] Pei Hsia, Jayarajan Samuel, Jerry Gao, David Kung, Yasafumi Toyoshima, and Cris Chen: Formal Approach to Scenario Analysis, IEEE Software, Vol. 11, No. 2, pp. 33-41 (1994).

[15] James Rumbaugh, Ivar Jacobson, and Grady Booch: The Unified Modeling Language Reference Manual, Addison-Wesley (1998).

[16] Bruce P. Douglass: Real-Time UML: Developing Efficient Objects for Embedded Systems, 2nd Edition, Addison Wesley, 2000.

[17] 大西淳： ビジュアルなソフトウェア要求仕様化技法, 情報処理学会論文誌, Vol.36, No.5 (1995).

[18] Jim Woodcock and Martin Loomes: Software Engineering Mathematics, SEI Series in Software Engineering, Addison Wesley (1988).

[19] Wang Yi, Paul Pettersson, and Mats Daniels: Automatic Verification of

Real-Time Communicating Systems by Constraint-Solving, In Proc. of the 7th International Conference on Formal Description Techniques (1994).

[20] Klaus Pohl, Gunter Bockle, and Frank Van Der Linden: Software Product Line Engineering: Foundations, Principles and Techniques, Springer-Verlag (2005).

[21] Michael. E. Fagan: Advances in Software Inspections, IEEE Transactions of Software Engineering, Vol. 12, No. 7, pp. 744-751 (1986).

[22] A. Frank. Ackerman, L. S. Buchwald, and Frank. H. Lewski: Software Inspections: An Effective Verification Process, IEEE Software, Vol. 6, No. 3, pp. 31-36 (1989).

[23] 岡本博幸, 内山敬太, 鈴木彩子, 高橋一仁, 西山茂, 野中誠： SRS 特性に着目したレビュー手法の評価および現場適用に関する一考察, 第 24 回ソフトウェア品質シンポジウム 発表報文集, pp.309-316 (2005).

[24] 野中誠： 設計・ソースコードレビューを対象とした個人レビュー法の比較実験, 情報処理学会研究報告 SE-146-4 (2004).

[25] B. Bern´ardez, M. Genero, A. Dur´an, M. Toro: A Controlled Experiment for Evaluating a Metric–Based Reading Technique for Requirements Inspection, Proceedings of the 10th International Symposium on Software Metrics (METRICS'04).

[26] Glenford J. Myers: Art of Software Testing John Wiley & Sons (1979) .

[27] Boris Beizer : Software Testing Techniques, Van Nostrand Reinhold (1990).

[28] 山本訓稔, 秋山浩一 : 直交表を利用したソフトウェアテスト ーHAYST 法ー, ソフトウェアテストシンポジウム (2004).

[29] Stewart Gardiner (Ed.), Testing Safety-Related Software: A Practical Handbook, Springer-Verlag (1999).

[30] Richard Hamlet: Random testing, Encyclopedia of Software Engineering, John Wiley and Sons (1994).

[31] テストの自動化, ソフトウェア・テスト PRESS Vol.3 (2006).

[32] Luigi Lavazza and Giuseppe Valetto, Enhancing Requirements and Change Management through Process Modelling and Measurement, 4th International Conference on Requirements Engineering, (2000).

[33] 中島毅, 東基衛： 要求生成のメカニズムのモデルに関する一考察, 情報処理学会ソフトウェア工学研究会 SE146-11 (2005).

[34] 井上克郎, 松本健一, 鶴保征城, 鳥居宏次： 実証的ソフトウェア工学環境への取り組み, 情報処理学会誌, Vol. 45, No. 7, pp. 722-728 (2004).

[35] Dag I. K. Sjoberg , Tore Dyba , Magne Jorgensen: The Future of Empirical Methods in Software Engineering Research, Future of Software Engineering (FOSE '07)

[36] Bernd Freimut and Lionel C. Briand: Determining Inspection Cost-Effectiveness by Combining Project Data and Expert Opinion, IEEE Trans. on Software Engineering, Vol. 31, No. 12 (2005).

[37] 小室睦，男澤康，木村好秀： 開発現場の実態に基づいたピアレビュー手法改善と改善効果の定量的分析， SEC journal No.4 (2005).

[38] Per Runeson, Carina Andersson, Thomas Thelin, Anneliese Andrews, and Tomas Berling,: What Do We Know about Defect Detection Methods? IEEE Software May/July (2006).

[39] Gaffney, Jr., J. E., "On Predicting Software Related Performance of Large-Scale Systems," CMG XV, San Francisco, December 1984.

[40] Stephen H. Kan, Metrics and Models in Software Quality Engineering, 2nd Edition, Addison-Wesley, 2005.

[41] Barry Boehm: Software Engineering, IEEE Transaction on Computers, Vol. 25, No. 12, pp. 1226-1241 (1976).

[42] 竹山寛：実践的ソフトウェア開発工程管理，技術評論社 (2000).

[43] Ellis Horowitz, Ray Madachy, Donald Reifer, Bert Steece, Bradford K. Clark, and Barry W. Boehm: Software Cost Estimation with COCOMO II, Englewood Cliffs, NJ:Prentice-Hall (2000).

[44] 中島 毅，岡田 和久：プロトタイピング再考，情報処理学会ソフトウェア工学研究会 ウィンターワークショップ・イン・恵那 (1998).

[45] LiGuo Huang and Barry Boehm: How Much Software Quality Investment Is Enough: A Value-Based Approach, IEEE Software, Vol. 23, No. 5, pp. 88-95 (2006).

[46] IEEE Std 1320.1-1998. IEEE Standard for Functional Modeling Language—Syntax and Semantics for IDEF0, New York: IEEE, 1998.

[47] JIS X 0160 ソフトウェア・ライフサイクル・プロセス

[48] 海谷治彦，佐伯元司： 要求分析におけるオントロジーの活用法，電子情報通信学会技術研究報告 SS Vol.105, No.25, pp. 7-12 (2005).

[49] LI Zong-yong , WANG Zhi-xue , YANG Ying-ying , WU Yue , and LIU Ying: Towards a Multiple Ontology Framework for Requirements Elicitation and Reuse, 31st Annual International Computer Software and Applications Conference, Vol. 1 (COMPSAC 2007).

[50] Lawrence Chung, Weimin Ma, and Kendra Cooper: Requirements Elicitation through Model-Driven Evaluation of Software Components, Fifth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'06).

[51] Pericies Loucopoulos and Vassilios Karakostas: System Requirements Engineering, McGraw-Hill (1995). (富野 壽監訳，「要求定義工学入門」，共立出版)

[52] Ann M. Hickey and Alan M. Davis: Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes, 36th Annual Hawaii International Conference on System Sciences (HICSS'03)

[53] Alan M. Davis: Software Requirements: Objects, Functions, and States,

Prentice Hall (1993).

[54] Merlin Dorfman and Richard F. Flynn : ART-an automated requirements traceability system, Journal of Systems and Software, Vol. 4 , No. 1, pp. 63-74 (1984)

[55] Kurt Schneider: Prototypes as Assets, not Toys: Why and How Extract Knowledge and Prototypes, Proceedings of the 18th International Conference on Software Engineering, pp. 522-531 (1996).

[56] James Rumbaugh, Michael Blaha, and William Premerlani: Object-Oriented Modeling and Design, Prentice Hall (1991).

[57] David Harel, Amir Pnueli, Jeanette P. Schmidt, and Ravi Sherman: On the Formal Semantics of Statecharts, In Proceedings of Symposium on Logic in Computer Science, pp.54-64 (1987).

[58] Oded Malar, Zohar Manna, and Amir Pnueli: From Timed to Hybrid Systems, Real-Time: Theory in Practice, LN-CS 600, Springer-Verlag, pp. 447-484 (1991).

[59] 蓬莱 尚幸: 問題点と解決策に注目した要求会議分析における機能単位分析の適用と改良, 情報処理学会ソフトウェア工学研究会 SE-115 (1997).

[60] Kai Koskimies and Erkki makinen: Automatic Synthesis of State Machines from Trace Diagrams, Software-Practice and Experience, Vol. 24(7), pp. 643-658 (1994).

[61] Naito, S. and Tsunoyama, M.: Fault Detection for Sequential Machines by Transition-Tour, Proceedings of IEEE Computing Conference, pp. 238-243 (1981).

[62] Tsun S. Chow: Testing Software Design Modeled by Finite-State Machines, IEEE Transactions on Software Engineering, Vol. 4, No. 3, pp.178-186 (1978).

[63] Guney Gonenc: A Method for the Design of Fault Detection Experiments, IEEE Transactions on Computer, Vol. 19, No. 6, pp. 551-558 (1970).

[64] Alfred V. Aho, Anton T. Dahbura, David Lee, and M. Umit Uyar: An Optimization Technique for Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours, IEEE Transactions on Communication, Vol. 39, No.11, pp. 1604-1615 (1991).

[65] Susumu Fujiwara, Gregor V. Bochmann, Ferhat Kbendek, Mokhtar Amalou, and Abderrazak Ghedamsi: Test Selection Based on Finite State Models, IEEE Transactions on Software Engineering, Vol. 17, No. 6, pp. 591-603 (1991).

[66] 佐藤 文明 宗森 純 井手口 哲夫 水野 忠則 : 有限オートマトンに基づくシステムの試験系列自動生成手法の提案—単一遷移検査系列法—, 電子情報通信学会論文誌 B, Vol.J72-B1 No. 3, pp. 183-192, (1989)

[67] Barry S. Bosik and M. Umit Uyar: Finite state machine based formal methods in protocol conformance testing: from theory to implementation, Computer Networks and ISDN Systems, Vol. 22, No. 1, pp. 7-33 (1991).

[68] Peter Savage, Steve Walters, and Mark Stephenson: Automated Test Methodology for Operational Flight Programs, IEEE Aerospace Conference in February (1997).

[69] 古川善吾，野木兼六，徳永健司： AGENT：機能テストのためのテスト項目作成の一手法，情報処理学会論文誌，Vol. 25, No. 5, pp. 736-744 (1984).

[70] 橋本:ADO(自動テスト実行ツール)のテストレポート，MITECH 会第 8 会 SE FORUM 分科会，(1996).

[71] 堀江誠一： 組み込みソフトの自動テストシステム, GAIO CLUB 9 月号, (2004).

[72] 山中弘，中島毅，別所雄三，広田和洋: ソフトウェア機能試験手順の状態遷移表に基づいた生成法, SE-119, (1997).

[73] 中島毅, 別所雄三, 山中弘, 広田和洋: シングルチップマイコン用 S/W 開発における問題点と一解決法, 情報処理学会ソフトウェア工学研究会, SE-115, pp.17-24 (1997).

[74] Amrit L. Goel: Software Reliability Models: Assumptions, Limitations, and Applicability, IEEE Transactions on Software Engineering, Vol. 11, No. 12, pp. 1411-1423 (1985).

[75] Forrest Shull, Vic Basili, Barry Boehm, A.Winsor Brown, Patricia Costa, Mikael Lindvall, Dan Port, Ioana Rus, Roseanne Tesoriero, and Marvin Zelkowitz: What We Have Learned About Fighting Defects, 8th IEEE International Symposium on Software Metrics (METRICS'02) , pp. 249-258 (2002).

[76] Ioana Rus, Forrest Shull, and Paolo Donzelli: Decision Support for Using Software Inspection, Proceedings of the 28th Annual NASA Goddard Software Engineering Workshop, (2003).

[77] Paolo Donzelli: A Decision Support System for Software Project Management, IEEE Software, Vol. 23, No. 4, pp. 67-75 (2006) .

[78] Jürgen Münch and Ove Armbrust, "Using Empirical Knowledge from Replicated Experiments for Software Process Simulation: A Practical Example." ISESE'03, 2003.

[79] 独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター編: 共通フレーム 2007, SLCP-JCF2007 国際規格適合 (2007).

[80] Capers Jones: Applied Software Measurement: Assuring Productivity and Quality, Mcgraw-Hill (1991).

[81] Watts S. Humphrey: Managing the Software Process, SEI Series in Software Engineering (1991).

[82] 中島毅，田村直樹，岡田和久，和田雄次： 顧客対話時の要求獲得と確認を支援するプロトタイピングツール：ROAD/EE, 電子情報通信学会論文誌 D-I, Vol.J85-D-I, No.8, pp.725-740 (2002).

[83] Naoki Tamura and Tsuyoshi Nakajima: ROAD/EE: A Prototyping Environment

for Object-Oriented Specifications, TOOLS USA'97, pp. 176-189 (1997).

[84] 中島毅，別所雄三，山中弘，広田和洋: 状態遷移モデルで記述された要求仕様に基づく組込みソフトウェアの自動試験法，電子情報通信学会論文誌 D-I，Vol.J84-D-I, No.6, pp.682-692 (2001).

[85] 中島毅，東基衛: ソフトウェア開発における品質プロセスのコスト最適化のためのモデルとシミュレーションツール，電子情報通信学会 D，Vol.J-D91，No.5，pp. 1216-1230 (2008).

[86] Alan. M. Davis, K. Jordan, and Tsuyoshi Nakajima: Elements Underlying the Specification of Requirements, Annals of Software Engineering, Vol. 3, pp.63-100 (1997).

[87] TestComplete 6, http://automatedqa.com/products/testcomplete/index.asp, AutomatedQA 社 HP.

[88] Tom Gilb, Dorothy Graham, and Susannah Finzi: Software Inspection, Addison-Wesley (1993).

[89] Kazuhira Okumoto: A Statistical Method for Software Quality Control, IEEE Transactions on Software Engineering, Vol. SE-11, No. 12, pp. 1424-1430 (1985).

[90] Shigeru Yamada and Shunji Osaki: Software Reliability Growth Modeling: Models and Applications, IEEE Transactions on Software Engineering, Vol. SE-11, No. 12 , pp. 1431-1437 (1985).

[91] Bernd Freimut, Lionel C. Briand, and Ferdinand Vollei: Determining Inspection Cost-Effectiveness by Combining Project Data and Expert Opinion, IEEE Transactions on Software Engineering, Vol. SE-31, No. 12, pp.1074-1092 (2005).

[92] James S. Collofello and Scott N. Woodfield: Evaluating the effectiveness of reliability- assurance techniques, Journal of Systems and Software, Vol. 9, No. 3, pp. 191-195 (1989).

[93] Robert B. Grady and Tom van Slack: Key Lessons In Achieving Widespread Inspection Use, IEEE Software, Vol. 11, No. 4, pp. 45-57 (1994).

[94] Shinji Kusumoto, Ken-ichi Matsumoto, Tohru Kikuno, and Koji Torii: A New Metric for Cost Effectiveness of Software Reviews, IEICE Transactions on Information and Systems Vol. E75-D, No. 5, pp. 674-680 (1992).

[95] Mark H. Klein, Thomas Ralya, Bill Pollak, and Ray Obenza: A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems, Kluwer Academic (1993).

[96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal: Pattern-Oriented Software Architecture. John Wiley and Sons (1996).

[97] Glenford J. Myers: Composite/Structured Design, Van Nostrand Reinhold (1978).

[98] 長井栄吾，牧寺彩，岡野浩三，谷口健一: 時間制約を保証する UML/OCL を用いた分散実時間アプロケーション開発手法，Vol. 89, No. 4 (2006).

[99] 高木智彦，古川善吾，山崎敏範: ソフトウェア保守のための統計的回帰テスト法の概要とそのケーススタディ, Vol. 89, No. 8 (2006).

[100] 津田道夫，楠本真二，松川文一，山村知弘，井上克郎，英繁雄，前川祐介: ユースケースポイント計測におけるアクタとユースケースの自動分類の試みと支援ツールの試作, Vol. 91, No. 4 (2008).

研究業績

種 類 別	題名、　　発表・発行掲載誌名、　　発表・発行年月、　　連名者
論文	
○	状態遷移モデルで記述された要求仕様に基づく組込みソフトウェアの自動試験法 電子情報通信学会論文誌 D-I，Vol.J84-D-I, No.6, pp.682-692 (2001-6) 中島毅，別所雄三，山中弘，広田和洋
○	顧客対話時の要求獲得と確認を支援するプロトタイピングツール：ROAD/EE, 電子情報通信学会論文誌 D-I，Vol.J85-D-I, No.8, pp.725-740 (2002-8). 中島毅，田村直樹，岡田和久，和田雄次
○	ソフトウェア開発における品質プロセスのコスト最適化のためのモデルとシミュレーションツール, 電子情報通信学会 D, Vol.J-D91, No.5, pp.1216-1230 (2008- 5). 中島毅，東基衛
	Elements Underlying the Specification of Requirements Annals of Software Engineering, Vol. 3, pp.63-100 (1997-1) Alan. M. Davis, Kathleen Jordan, and Tsuyoshi Nakajima
講演	(国際会議) Classifying Requirements Errors for Improving SRS Reviews 1st International Workshop on Requirements Engineering: Foundation for Software Quality, Utrecht(The Netherlands), June 1994, pp.88-100. Alan M. Davis and Tsuyoshi Nakajima
	ROAD/EE: A Prototyping Environment for Object-Oriented Specifications Technology of Object-Oriented Languages and Systems, Santa Barbara(USA), Aug 1997 pp.176 - 189 Naoki Tamura, and Tsuyoshi Nakajima
	(講演) オブジェクト指向仕様記述の実行・検証系 ROAD/EE 情報処理学会ソフトウェア工学研究会 SE102-11, 1995 年 1 月 pp. 59-64 田村直樹，柳生理子，萩原正敏
	新しい要求分析スタイルの模索～オブジェクト指向仕様の実行検証系：ROAD/EE～ 情報処理学会ソフトウェア工学研究会サマーワークショップイン立山，1995 年 7 月 中島毅，田村直樹，柳生理子
	オブジェクト指向プロトタイピングツール ROAD/EE　(ソフトウェアツール/環境・デモンストレーション) 情報処理学会ソフトウェア工学研究会 SE-106，1995 年 11 月，pp.17-24 田村直樹，中島毅，柳生理子，萩原正敏

種 類 別	題名、　　発表・発行掲載誌名、　　発表・発行年月、　　連名者
講演	(講演) シングルチップマイコン S/W 開発における問題点と一解決法 情報処理学会ソフトウェア工学研究会 SE-115-3, 1997 年 7 月，pp.17-24 中島毅，別所雄三，山中弘，広田和洋 ソフトウェア機能試験手順の状態遷移表に基づいた生成法 情報処理学会ソフトウェア工学研究会 SE-118-16，1998 年 3 月，pp.119-126 山中弘，中島毅，別所雄三，広田和洋 プロトタイピング再考 情報処理学会ソフトウェア工学研究会ウィンターワークショップ・イン・恵那，1998 年 1 月 中島毅，岡田和久 要求生成のメカニズムのモデルに関する一考 情報処理学会ソフトウェア工学研究会 SE146-11，2005 年 11 月，pp 79-86 中島毅，東基衛 コストを考慮した品質計画立案支援ツール プロジェクトマネジメント学会 2007 年度秋季研究発表大会，2007 年 9 月 中島毅，吉田見岳
その他	(国際会議) A Method for Recording and Analyzing Design Processes 5th International Workshop on Software Process, Kenebunkport(USA), October 1989, pp.106-108 Tsuyoshi Nakajima PPK: A Method for Recording and Analyzing Software Processes IEEE COMPSAC ’90, Chicago(USA), October 1990, pp.555-563 Tsuyoshi Nakajma, Naoki Tamura, and Kenji Uehara Domain Specific Design Method Based on Software Design Processes Fourth International Conference on Software Engineering & Its Apprications (TOULOUSE ’91), Toulouse(France) December 1991, pp.575-586 Norihiko Suhara, Tsuyoshi Nakajima,Kenji Uehara (講演) プログラミングデータベース:SODA 情報処理学会ソフトウェア工学研究会 SE-40-20，1985 年 2 月，pp.115-120 中島毅，上原憲二，石川由美子，高野彰，春原猛

種 類 別	題名、　　発表・発行掲載誌名、　　発表・発行年月、　　連名者
その他	(講演) ソフトウェア開発用インタフェース:sif 情報処理学会ソフトウェア工学研究会 SE-58-17, 1988 年 2 月，pp.131-138 藤岡卓，中島毅，上原憲二，高野彰 ソフトウェア開発用インタフェース sif における作業記述について 情報処理学会ソフトウェア工学研究会 SE-64-21，1989 年 2 月，pp.161-168 中島毅，藤岡卓，上原憲二，高野彰 PPK 法:ソフトウェア設計プロセスの記録と分析の手法 情報処理学会ソフトウェア工学研究会 SE-67-2，1989 年 7 月，pp.1-8 中島毅，田村直樹，藤岡卓，上原憲二，高野彰 ハイパテキストを用いた設計プロセス支援ツールの試作 情報処理学会ソフトウェア工学研究会 SE-68-7，1989 年 9 月，pp.1-8 田村直樹，中島毅，上原憲二 ＪＳＰ法を用いた設計プロセスの記録と分析 情報処理学会ソフトウェア工学研究会 SE71-4，1990 年 2 月，pp.25-32 中島毅，田村直樹，上原憲二 設計プロセス支援ツールにおける作業の記録、モニタリング、再利用 情報処理学会ソフトウェア工学研究会 77-18，1991 年 2 月，pp.107-112 田村直樹，中島毅，上原憲二 グループ企業連携でのソフトウェア・プロセス改革 プロジェクトマネジメント学会 2007 年度春季研究発表大会 2209 吉田見岳，佐々木俊昌，渡辺晴彦，大野俊樹，近藤聖久，味岡学，藤山直樹，糸谷友良 (著作) ソフトウェア開発 オーム社（情報処理学会編集），2003 年 小泉寿男，辻秀一，吉田幸二，中島毅