仕様変更に柔軟に対応できる開発手法の研究

－原理・原則（拡張性、柔軟性）に基づく開発手法の確立－

FUJITSU ファミリ会 LS 研究委員会「仕様変更に柔軟に対応できる開発手法の研究」分科会

■ 執筆者Ｐｒｏｆｉｌｅ ■

2010 年 4 月 FUJITSU ファミリ会 LS 研究委員会の研究分科会にて、16 名のメンバーと富士通テクニカルアドバイザー2 名の合計 18 名で研究活動を実施。

2011 年 7 月 総合発表会にて優秀賞を受賞。現在、メンバー各社にて研究成果を適用し、継続的な改善に取り組んでいる。

共同執筆メンバー

氏　　名	団　体　名（順不同）
神林　昭	（株）ＴＫＣ
嘉山　裕晶	ハマゴムエイコム（株）
小林　裕	第一生命情報システム（株）

■ 論文要旨 ■

近年のビジネス環境の激変に伴いシステムのリリース後における仕様変更は頻発し、避けられなくなっている。これに迅速かつ低コストで対応するには、開発プロセスの改善だけでなく、システムに今後の仕様変更に対応できるだけの保守性を持たせる必要がある。当分科会では、仕様変更を事前に想定するために必要な「拡張性」と、事前に想定できない仕様変更に対応するために必要な「柔軟性」が、今後の仕様変更における保守性の低下を防ぐ鍵になると考えた。事例分析や議論を重ねた末、拡張性を満たす手段として原理・原則に基づく設計手法、柔軟性を満たす手段としてグレーゾーンを最小化する影響調査手法を確立した。過去事例による検証では、開発コストや影響調査時間の大幅な削減効果を確認し、開発量の削減に有用であることを実証した。

研究期間終了後の現在は、研究成果をメンバー各社へ展開中である。そこで見つかった課題と考察についても取り上げる。

■ 論文目次 ■

１．はじめに ……《 4》
１．１　背景
１．２　結論

２．現状分析と課題認識 ……《 5》
２．１　仕様変更の原因
２．２　現状分析
２．３　解決すべき課題
２．４　柔軟な対応に求められる特性

３．拡張性の確保 ……《 8》
３．１　拡張性とは？
３．２　拡張性を確保する手段の追究
３．３　検証

４．柔軟性の確保 ……《 12》
４．１　柔軟性とは？
４．２　柔軟性を確保する手段の追究
４．３　検証

５．新・開発ガイドライン ……《 17》
５．１　新・開発ガイドラインの構成と役割
５．２　新・開発ガイドラインの評価

６．適用に向けて ……《 20》
６．１　研究終了後の成果の適用状況
６．２　残された課題

７．おわりに ……《 23》

■ 図表一覧 ■

図１ 釣り場所の変更における二つの判断基準 ……《 4》
図２ 研究の進め方 ……《 5》
図３ 仕様変更の原因 ……《 5》
図４ 仕様変更に関する現状分析 ……《 6》
図５ 解決すべき課題 ……《 6》
図６ 燃料警告ランプ点灯の原理・原則 ……《 8》
図７ 燃料警告ランプ点灯機能の設計例 ……《 8》
図８ 概念モデリング ……《 9》
図９ 原理・原則導出モデル ……《 10》
図10 原理・原則の発見と拡張性の確保 ……《 10》
図11 担当案件の掘り起こしとシミュレーション ……《 11》
図12 仕様変更への対応が不要となる例 ……《 12》
図13 柔軟性の概念図 ……《 13》
図14 影響調査の現状に関する調査結果 ……《 13》
図15 検索キーワード決定の前提条件 ……《 14》
図16 開発工程での打ち手 ……《 15》
図17 あるべき影響調査の流れ ……《 15》
図18 影響調査手法の検証結果 ……《 16》
図19 新・開発ガイドラインの構成 ……《 17》
図20 原理・原則導出ガイド（サンプル） ……《 18》
図21 拡張性編の活用イメージ ……《 18》
図22 柔軟性編の活用イメージ ……《 19》
図23 新・開発ガイドラインWBS ……《 19》
図24 アンケート質問項目及び回答結果 ……《 20》
図25 原理・原則導出モデルの活用 ……《 21》
図26 キャッチアップ資料の活用 ……《 21》
図27 拡張性と柔軟性の融合 ……《 22》

表１ 保守性をもたらすポイントの抽出と分類 ……《 7》
表２ 原理・原則に基づく設計の適用シミュレーション結果 ……《 12》
表３ 影響調査の問題点と対応案 ……《 14》
表４ 影響調査手法の検証で見つかった問題点とその対応 ……《 17》

１．はじめに

１．１　背景

開発者にとって、仕様変更はなくなるに越したことはない。仕様変更の発生がコストや納期の超過につながり、その場限りの対応がもとで品質の低下を招いた経験は誰もが持っている。しかし、近年のビジネス環境の急激な変化によってユーザーの要求は日々変化しており、開発中のみならず情報システムのリリース後も仕様変更は頻発する。したがって、仕様変更をなくすことよりもどう対処するか考えることが、今向き合うべき課題といえる。

このような課題に対して、これまで開発プロセスの改善による解決が研究されてきた。現在もなお主流といえるウォーターフォール型開発では、仕様変更発生時の手戻りのコストが高く、一般的には仕様変更に弱いと言われている。一方、近年注目されるアジャイル型開発の多くは、開発対象を機能単位に分割し、イテレーション（反復）という短い期間単位で開発を進めていく。これによって、前イテレーションで発生した仕様変更を次イテレーションで吸収でき、仕様変更に対応しやすくなる。

しかし、開発プロセスの改善だけでビジネス環境の変化によって繰り返し発生する仕様変更に耐えられるだろうか？

ここで、身近な例として釣り場所の変更を取り上げる。まずは図１をご覧いただきたい。

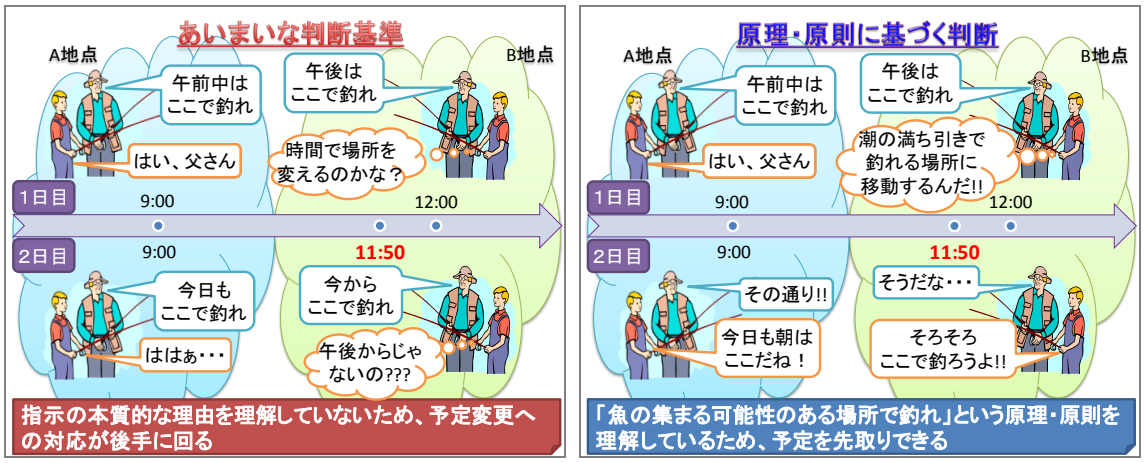


図１　釣り場所の変更における二つの判断基準

ある日、釣りの達人である父とその息子が一緒に釣りをしていた。朝、「午前中は A 地点で釣れ」と父は言い、正午には「B 地点で釣れ」と言う。このとき息子は、場所を変える判断基準を「時間」だと考えていた。ところが、翌日父は 11 時 50 分に「B 地点で釣れ」と指示した。息子は前日より 10 分早く移動するという指示の本質的な理由を理解できず、ただ従うだけだった。これが左の図の状況である。

一方、右の図で息子は、最初に B 地点に移動した時に指示の本質的な理由を考える。すると、「魚の集まる可能性のある場所で釣れ」という原理・原則に基づいて釣り場所を移動していることに気づく。この時点で息子は本当の判断基準を理解したため、魚の動きに影響する潮の満ち引きを考慮する。その結果、翌日 B 地点に移動するのは 11 時 50 分だとあらかじめ想定し、父に進言する。

この例を仕様変更への対応に当てはめてみると、左の図は仕様に盲目的に従うのみで常に対応が後手に回る状況、右の図は仕様の背後にある理由を捉えて先取りして対応する状況を表している。どちらが柔軟な対応であるかは一目瞭然ではないだろうか。開発プロセスの改善だけでも仕様変更への対応の生産性は上がるだろうが、仕様の背後にある本質的な理由を捉えれば、その後発生する仕様変更への対応の仕方は劇的に変わるはずである。

仕様変更に対応する開発方法論、プロセス、設計手法は多々あるが、「ビジネス環境の変化による仕様変更」に対して、未だ決定打となる手法は出てきていない。

１．２　結論

仕様変更に柔軟に対応できる開発手法とは、「原理・原則に基づく開発手法」である。

図１では、潮の満ち引きという環境変化に対して、釣り場所を変えるタイミングの判断が変わっている。しかし、「魚の集まる可能性のある場所で釣れ」という原理・原則は何ら変わっていない。ビジネス環境の変化に対しても、この例のように原理・原則を捉えて設計に反映させれば、仕様変更に強いシステムを構築できると考えた。

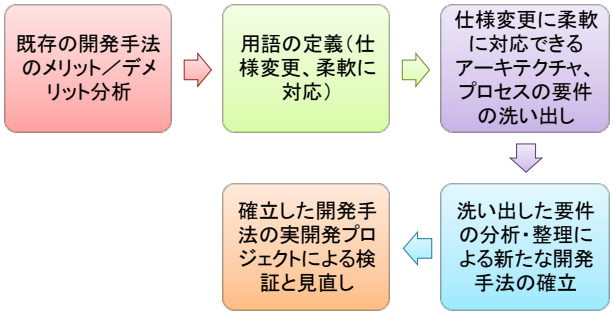


図２　研究の進め方

当分科会では、図２の研究を経てこの結論に至った。その過程、及び「原理・原則に基づく開発手法」の詳細について、次章以降で明らかにする。

２．現状分析と課題認識

２．１　仕様変更の原因

図３は、一般的に考えられる仕様変更の原因である。

このうち①から④までは「あってはならない仕様変更」であり、反復型開発やプロトタイピングの活用といった開発プロセスの改善による対処が有効である。ただ、当分科会メンバー各社ではいまだにウォーターフォール型が主流であり、社内規定などの問題から全面的な導入は難しい。

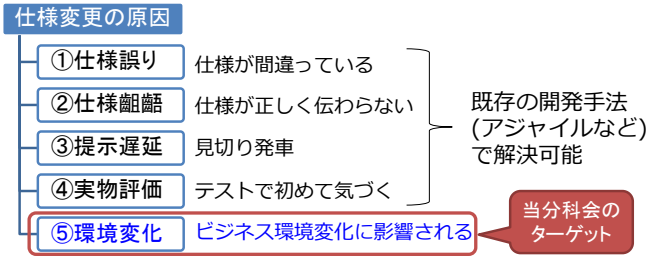


図３　仕様変更の原因

一方、⑤の環境変化は法改正や新サービスの追加といった「避けられない仕様変更」であり、その頻度は近年のビジネス環境の激変によって上昇している。情報システムがビジネスに及ぼす影響の大きさを鑑みると、ユーザーにとってシステム側の対応は不可欠である。その対応において、開発中の仕様変更ではアーキテクチャの見直しは非現実的であり、開発プロセスの改善で対応するしかない。しかし、このような緊急的な対応を繰り返すとシステムの保守性は低下する。その結果、リリース後の環境変化による仕様変更では開発量が増加し、開発プロセスを改善しても品質低下、コスト超過、納期遅延を防げなくなる。

したがって、開発プロセスの改善だけではリリース後の環境変化による仕様変更には対応しきれない。このことは、メンバー各社の仕様変更対応事例を用いた反復型開発のシミュレーションにおいて、手戻りを始めとするウォーターフォール型開発の問題点を解決できない事例が過半数にのぼったことからも窺える。

このような考察を経て、当分科会ではウォーターフォール型や反復型といった開発プロセスの違いにこだわらず、「ビジネス環境の変化によるリリース後の仕様変更」に焦点を当てて研究を進めることとした。

２．２　現状分析

図４は、仕様変更に関するユーザー及び開発者の現状を分析したものである。

ユーザー側において重要なのは、情報システムの位置づけの変化である。今や情報システムは業務遂行に不可欠であり、経営競争力を左右するものである。ビジネス環境の変化による他社との競争が激化すると、開発者への要求が高度化・複雑化するだけでなく、短納期での対応を求めることとなる。

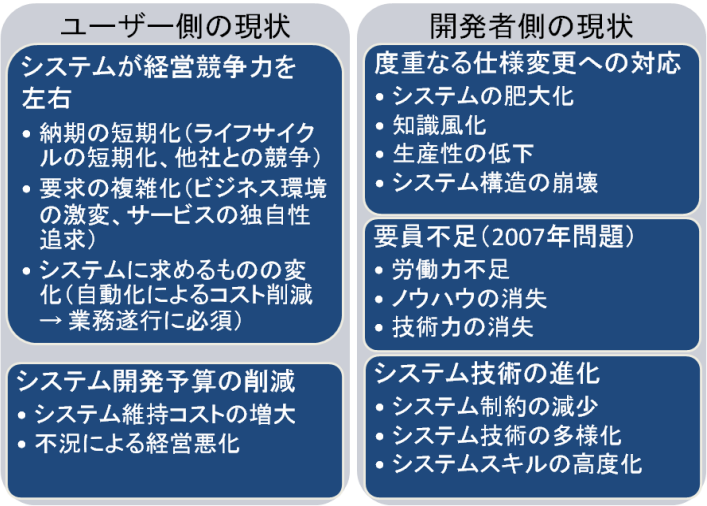


図４　仕様変更に関する現状分析

また、コスト面の変化も見逃せない。「企業 IT 動向調査 2010（JUAS）」によれば、2009 年度における企業の開発費の伸び率はマイナス 20%を超えている。これは、仕様変更への対応にもコスト削減圧力がかかることを意味する。このように、仕様変更への対応に設けられたハードルは高くなってきている。

一方の開発者側においては、技術の高度化や技術力の消失といった問題もさることながら、やはり度重なる仕様変更への対応に伴う様々な問題が、その後の仕様変更への対応における最大の阻害要因となっている。ユーザーの設けるハードルの高さに逆行して、開発者のハードルを越える力は低下している。

２．３　解決すべき課題

図５は、現状分析を受けて、解決すべき課題を QCD（品質・コスト・納期）の側面から捉えたものである。

リリース後の仕様変更においては、多様な要求に対して一定の品質を確保しながら、低コストかつ短納期で対応しなければならない。そのために解決すべき最重要課題は「開発量の削減」である。

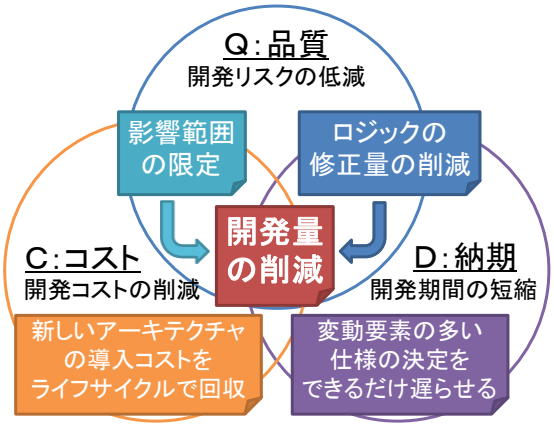


図５　解決すべき課題

開発量の削減とは、開発生産性の向上ではなく開発作業（設計・テストなどのすべての

作業）自体が不要になることを意味する。開発作業が発生しないため、おのずから期間短縮、コスト削減を達成でき、開発リスクも低減できる。この開発量の削減を可能にするのが、「ロジックの修正量の削減」と「影響範囲の限定」である。

ロジックの修正量の削減では、ロジックの修正ではなく設定変更による対応を可能にするといった新たなアーキテクチャが必要となり、一定の導入コストがかかる。しかし、実現すれば開発リスクの低減、及びシステムライフサイクルにおけるコスト削減が見込まれる。一方、影響範囲を限定する仕組みがあれば、仕様変更の内容を問わず影響分析すべき範囲を狭められ、対応後のリグレッションテストなどにかかる工数を抑えられる。

当分科会は、開発量の削減を中心とした QCD の課題を解決すれば、仕様変更に柔軟に対応できると考えた。なお、仕様の決定をできるだけ遅らせる取り組みについては、既存の反復型開発の守備範囲と考えられるため、本研究の対象外とする。

２．４　柔軟な対応に求められる特性

「柔軟な対応」とは、今後の仕様変更に対する保守性を確保し、最小のコスト、納期でユーザーの求める品質を実現することである。仕様変更が繰り返し発生しても保守性を低下させないことが、柔軟な対応につながる。開発量の削減は、今後の仕様変更に対する保守性を確保するために解決すべき課題である。

当分科会では、開発量の削減の実現方法を具体化するため、変化に強いと言われる既存の設計技法を調査・分析し、その結果をもとにアーキテクチャの保守性の評価指標を検討した。しかし、既存の設計技法は開発生産性に重きを置いたものが多く、また開発者のスキルへの依存度も高い。保守性の評価指標については、既に定量的に評価するサービスが実在することが判明した。それでも繰り返し発生する仕様変更への対応に課題が残る現状から、既存の設計技法や保守性の評価指標とは違った新たな切り口が必要と考えられる。

この新たな切り口を明らかにするため、メンバー各社の仕様変更対応事例における成功／失敗要因を掘り下げ、保守性をもたらすポイントを表１のとおり抽出した。

主な事例（○：成功事例、×：失敗事例）	ポイント	分類
○ 帳票申請システムへの個別処理の追加 × 給付金請求における書類選択機能の追加 ⇒ 変更箇所を限定できる設計にすべきだった	業務の概念をモデリングし、その構造を設計に反映する	拡張性
○ 銀行業務の取引登録における日付条件の変更 × 書類管理ステータスの追加・変更 ⇒ ステータス遷移をルール化すべきだった	仕様や要求の背後にある業務の原理・原則を導いてルール化し、設計に反映する	
× データ参照専用のシステムへの更新機能の追加 × 社名変更に伴う文言変更 ⇒ 影響箇所を特定しやすい仕組みにすべきだった	影響調査の仕組みを確立し、開発量を削減させる	柔軟性
× 予実管理における翌年度予算入力機能の追加 × インターネットバンキング新規サービスの追加 ⇒ 業務的なチェック処理を共通化すべきだった	設計と実装の構造を一致させ、機能の実装箇所不明といった構造崩壊を防ぐ	
× ワークフロー作成書類のカスタマイズ ⇒ 変更管理手順を定め、周知徹底すべきだった	担当者交代やドキュメント不備による追跡可能性消失を防ぐ	

表１　保守性をもたらすポイントの抽出と分類

最初の二つのポイント（モデリング、原理・原則）は、システムの扱う業務を設計に反映するものである。業務を正しく捉えれば事前に想定できる仕様の範囲を広げられるため、

「ロジックの修正量の削減」を実現でき、保守性をもたらす。残りの三つのポイント（影響調査、構造、追跡可能性）は、「影響範囲の限定」を実現しやすくする。影響範囲を限定すれば、設計や仕様変更の内容を問わず対応コストや期間を削減できる。したがって、事前に想定できない仕様変更にも一定の保守性をもたらす。

このように、ロジックの修正量の削減、影響範囲の限定という二つの課題を解決すれば、開発量を削減して今後の仕様変更に対する保守性を確保できることが、仕様変更対応事例からも明らかになった。これら二つの課題のうち前者に求められる特性を「拡張性」、後者に求められる特性を「柔軟性」と呼ぶこととし、次章以降で具体的な定義や手法の追究プロセスを述べる。

３．拡張性の確保

３．１　拡張性とは？

当分科会の考える拡張性とは、「業務の本質が変わらない限り変わらない『原理・原則』を発見し、変わらない部分に影響しないよう、仕様変更を事前に想定して設計に反映させるために必要な特性」である。この定義に至った経緯、及び「原理・原則」の実体を明らかにするため、自動車の燃料警告ランプ点灯の事例を挙げる。

燃料警告ランプを点灯する機能を開発する場合、通常はガソリンの残量と燃費で判断するロジックを想定する。車種ごとの燃費テーブルを定義すれば、車種が追加されても柔軟に対応できる。しかし、ハイブリッドカーの車種が追加されると同じテーブルでは計算できず、ロジックの修正が必要となる。

ここで、仕様、要件やその背後にある理由に注目すると、「国内で一番離れたガソリンスタンドに、燃料が切れる前に到着できれば良い」という本質に気づく。国内でガソリンスタンド間の最長距離は 140km であるため、あるべき仕様は「一番離れたガソリンスタンド（140km）に到着できるうちに燃料警告ランプを点灯すること」となる。これが、本質が変わるまで変わらない「原理・原則」である（図６）。

図６　燃料警告ランプ点灯の原理・原則

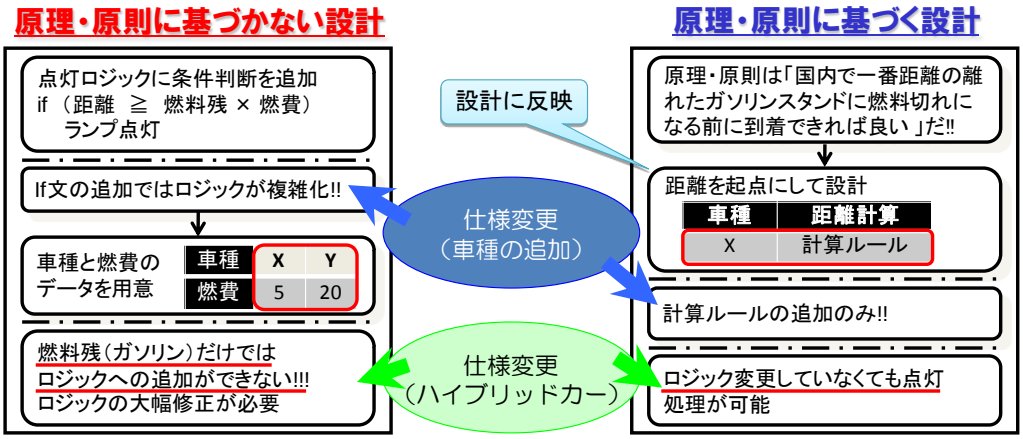


図７　燃料警告ランプ点灯機能の設計例

前ページの図７は、原理・原則に基づかない設計（燃料残と燃費で判断）と、原理・原則に基づく設計の対比を表している。原理・原則をシステム構造に反映させると、距離を起点にした計算ルールを車種ごとに定義する設計となる。計算ルールの異なる車種の追加を事前に想定することによって、ハイブリッドカーの車種に対してもルールコンテンツの追加のみで対応できる。ロジックの修正を抑えられるため、最小のコスト・期間で対応できる。これが、拡張性を持った設計である。

３．２　拡張性を確保する手段の追究

原理・原則を発見し、それに基づく設計を実践すれば、拡張性を確保できる。その手段を追究するためにまず取り組んだのが「概念モデリング」である。

対象業務の概念をモデリングすることによって、仕様や要件の背後にある理由を可視化してその関連性を整理でき、システムに反映しやすくなる。このため、メンバー各社の仕様変更対応事例、及び予約やネットショッピングといった代表的なビジネスパターンを題材に、概念モデリングに取り組んだ。その一例を図８に示す。

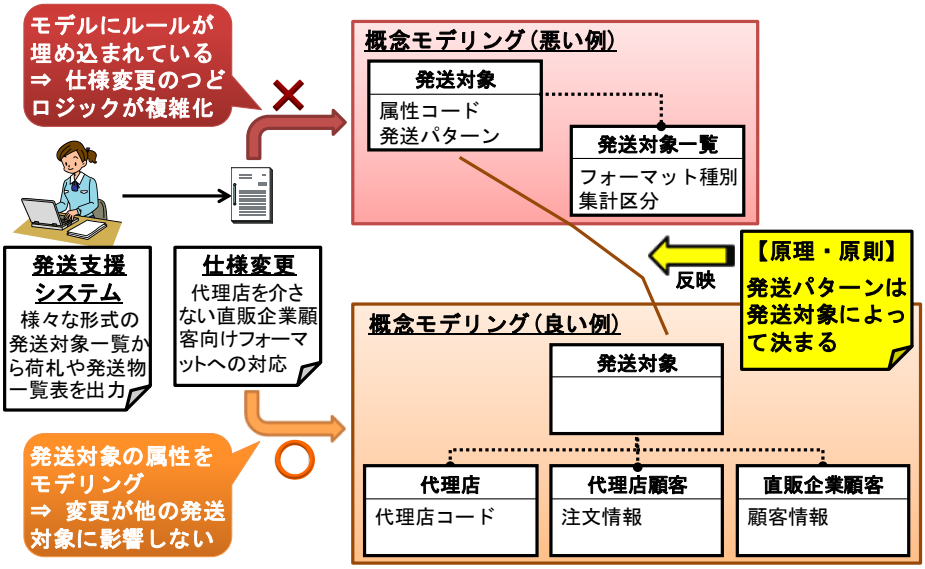


図８　概念モデリング

ソフトウェアベンダーの Ｚ 社では、代理店経由でソフトウェアを販売するほか、代理店自身が利用するソフトウェアも販売している。インストールメディアやマニュアルなどを希望する顧客に発送する際は、データベースやスプレッドシートといった、様々な入力形式から荷札や発送物一覧表を出力し、発送作業に利用している。

今回、代理店を介さない直販企業顧客向け発送への対応が必要となった。この仕様変更に対して、図８の上のモデルは発送対象を属性コードで管理しており、一見すると影響が小さいように見える。しかし、実際は発送対象と入力形式を個別ロジックで逐一関連づけており、モデルにルールが埋め込まれた状態になっていた。その中で、直販企業顧客向けに必要な入力形式が既に代理店向けとして使用されていたため、条件分岐が多数発生した。したがって、これは仕様変更に柔軟に対応できない概念モデルである。

一方、図８の下のモデルは発送対象を一つの概念に集約せず、「発送パターンは発送対象によって決まる」という原理・原則に基づいて個々にモデリングしている。これによっ

て、入力形式との関連づけのルールが発送対象から切り離され、発送対象の追加におけるロジックの修正を抑えられる。これが、仕様変更に柔軟に対応できる概念モデルである。

この例では、発送対象の追加という仕様変更だけでなく、その背後にある業務を対象に概念モデリングしたことで、原理・原則を発見して設計に反映させることができた。しかし、研究過程でこのようにモデリングできた例はごくわずかである。それらに共通する前提条件として、少なくとも対象業務にある程度精通していることが求められる。その上で、モデリングのスキルに関係なく原理・原則を発見する方法を確立するため、次の三つのプロセスを繰り返した。

- ビジネスルールの分類などの関連知識・技術情報の収集
- 仕様変更対応事例を対象とした原理・原則発見への取り組み
- 結果の分析と討議

図９は、その成果として当分科会が確立した「原理・原則導出モデル」である。このモデルは、開発工程の中で要件、理由、仕様といった要素ごとに内容を記述して可視化することによって、どの要素からも原理・原則が発見できることを表している。発見した原理・原則をデータモデルやルールコンテンツに反映する一方、既存システムの分析によって両者から原理・原則を発見することも可能である。

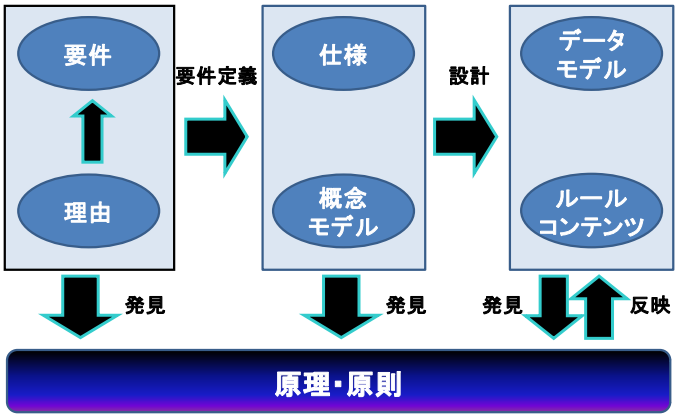


図９　原理・原則導出モデル

これら一連の研究過程で得たノウハウについては、次の二つのドキュメントにまとめた。

- 原理・原則導出ガイド（原理・原則導出モデルを用いた原理・原則発見の手順書）
- 原理・原則書（発見した原理・原則を記録し、活用するための様式）

当分科会は、「『原理・原則に基づく設計』は仕様変更に柔軟に対応できる」と考える。ここまでの研究過程をまとめると、原理・原則の発見は図 10 のステップをたどる。

まず、要件を判断、行動、データに分類する。次に、要件の成立に必要な判断からルールを導き出し、そのルールのうち業務の本質として今後変わらないこと、変わることを明確に区別する。この「今後変わらないこと」が原理・原則となる。

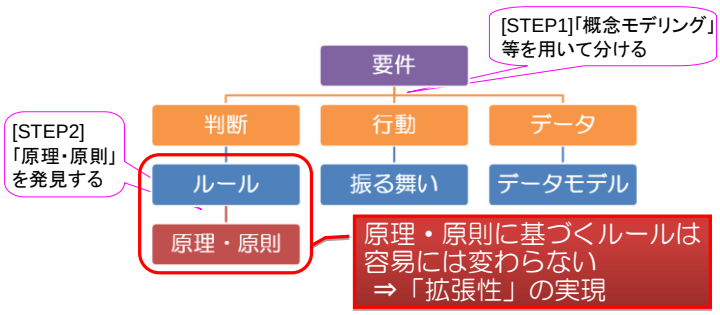


図 10　原理・原則の発見と拡張性の確保

このように業務の本質を意識することで、今まで見えていなかった原理・原則を見つけやすくなる。振る舞いやデータモデルに原理・原則を散在させるのではなく、ルールのみが原理・原則に基づいたシステム構造としておくことで、振る舞いやデータモデルに対する仕様変更を事前に想定できる。これが、拡張性を確保する手段である。

３．３　検証

本節では、拡張性を確保する手段として確立した、原理・原則に基づく設計手法の検証過程を述べる。

３．３．１　方法

メンバーの直近３年分の担当案件を対象に、原理・原則に基づく設計による開発コストの削減効果をシミュレーションした。その手順は次のとおり。

（１）シミュレーション対象案件の選定

- ３年以上稼働している業務系システム
- 仕様変更による作り込みが多いシステム
- 直近３年分の案件

（２）選定案件の分析（案件ごとに以下の手順を繰り返す）

- 機能開発当時を振り返り、WBS や要件定義書、基本設計書、画面定義書、帳票定義書、テーブル定義書、ER 図、詳細設計書、ソースコードなどを掘り起こす
- 原理・原則導出ガイド、原理・原則書を用いて原理・原則を発見する

（３）コスト削減効果のシミュレーション（原理・原則を発見した場合のみ）

- 機能開発当時から原理・原則に気づいて設計に反映したと仮定し、リリース後の仕様変更への対応にかかる開発コストの変化をシミュレーションする
- シミュレーション結果（開発コスト、作業ログなど）をまとめる

３．３．２　結果

メンバー４名が PM として携わった 201 案件がシミュレーション対象の要件を満たし、これらを対象にシミュレーションを実施した。そのサンプルを図 11 に示す。

過去案件掘り起こし					原理原則が適用できた場合のAFTER			
案件名	着手時期	振舞いとルール	工数（人月）	期間	原理・原則を導出できたか？	工数（人月）	期間	どういう拡張性があったらコスト削減できたか？
案件1	21年6月	【ルール】 ・対象サービスは、総合〇〇と〇〇 ・顧客の〇〇が〇〇の場合は、〇〇チェックを行う ・一般〇〇の場合は、自システムでエラーチェックを行わない ・大量〇〇の場合は、自システムでエラーチェックを行う ・ホストで〇〇超過の際は、データのみ送信 【振舞い】 ・〇〇チェック　又は　チェックせずにホスト送信 ・〇〇超過時、データのみ送信（一般〇〇） ・〇〇超過時、エラー返信（大量〇〇）	4.20	3ヶ月	〇	2.00	3ヶ月	限度額チェック処理と条件を共通化していれば、当時の対応はルール改定とホスト側からの限度額超過エラー時の判定追加のみで対応可能
案件2	19年11月	【ルール】 ・対象サービスは、〇〇 ・本部〇〇は、即時〇〇と予約〇〇を実施 ・支部〇〇は、即時〇〇のみ実施 ・〇〇照会結果で相違した場合、マスタ反映有無確認を行う 【振舞い】 ・〇〇照会実施有無 ・〇〇マスタへの反映ボタンを表示	3.10	3ヶ月	×	－	－	－
案件3	20年	【ルール】						

図 11 担当案件の掘り起こしとシミュレーション

前述の手順からもわかるとおり、201 案件の洗い出しと分析にかかる作業量は膨大であり、多くの時間を費やした。その甲斐あって、次のとおりの結果を出すことができた。

コストベースで見ると、原理・原則に基づく設計の適用によって４社で計約８％（約 123 人月）の開発コスト削減となった。要員単価を１人月 100 万円で試算すると、約 1.2

億円もの削減となる。

案件ベースで見ると、原理・原則を発見できた案件は 14 件となった（表２）。中でも、定期改訂案件、同種の仕様変更の対応実績がある案件において、高い確率で原理・原則を導出できる傾向が見られる。

事例	担当案件（直近３年）		シミュレーション結果			
	総工数（人月）	総案件数（件）	原理・原則導出（件）	対応不要（件）	削減工数（人月）	削減率（%）
A 社	160.72	29	2	5	26.50	17
B 社	799.76	31	6	4	85.45	11
C 社	535.20	112	2	1	7.00	1
D 社	69.50	29	4	5	4.00	6
計	1,565.18	201	14	15	122.95	8

表２ 原理・原則に基づく設計の適用シミュレーション結果

更に、機能開発当時に拡張性を作り込むことによって後続の対応を不要にできる案件が 15 件存在した。一例を図 12 に示す。

この例において「顧客への返却が不要になった書類は、保管拠点から原本を倉庫へ移動する」という原理・原則を発見して設計・実装に反映した場合、倉庫への移動対象書類や移動条件の追加・変更に関する後続２案件（9.2 人月）への対応がほぼ不要となった。対応不要という結果は、究極の開発量削減だといえる。

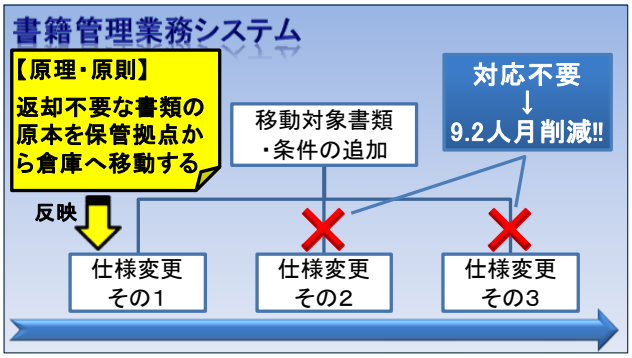


図 12 仕様変更への対応が不要となる例

このようなコスト削減効果がもたらされた要因を考察すると、やはりロジック修正量の削減によるところが大きい。なぜなら、原理・原則に基づいて設計し、設定変更のみで対応できるようなシステム構造としておくことで、リリース後の仕様変更への対応に必要な開発ステップ数、テストケース数が格段に少なくなるからである。

なお、検証過程で原理・原則書の使い勝手の面を中心に次の見直しを実施した。これによって、原理・原則に基づく設計手法の有用性を更に高めることができた。

- どのような拡張性を作り込めば良いかを記録できるようにする
- 拡張性の作り込みや現状の仕組みにおける課題を記録できるようにする
- 原理・原則書に記録した履歴を管理できるようにする

このように、当分科会の確立した原理・原則に基づく設計手法を用いて拡張性を作り込めば、リリース後の仕様変更では開発コストを削減する効果が得られる。拡張性を作り込む際にはコストがかかるが、システムライフサイクルで得られる効果はそのコストをはるかに上回る。したがって、２章で挙げた「ロジック修正量の削減」とそれによる「開発量の削減」という課題を解決し、仕様変更への柔軟な対応を可能にする。

４．柔軟性の確保

４．１ 柔軟性とは？

当分科会の考える柔軟性とは、「グレーゾーンを小さくするために必要な特性」である。この定義に至った経緯を明らかにするため、ある仕様変更対応事例を取り上げる。

図 13 の例では、「ボーナスポイントサービスの全店舗展開」という仕様変更に対し、「影響調査」「設計」「改修」の各プロセスで対応する。この中の「影響調査」において、仕様書が見つからないなどの理由で影響範囲を特定できない「グレーゾーン」が大きくなると、影響分析が困難となる。その結果、改修工数の増大や、膨大なリグレッションテストといった事態に陥る。これに対して、必要なドキュメントや手順を整備しておけば、影響調査におけるグレーゾーンを小さくでき、迅速かつ正確に進められる。これによって、事前に想定できない仕様変更への対応であっても一定の保守性を確保できる。

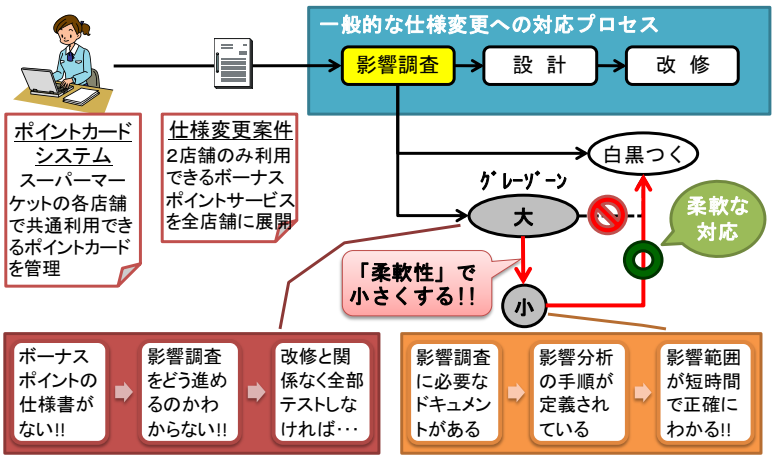


図 13 柔軟性の概念図

このことから、当分科会はグレーゾーンを小さくするために必要な特性を「柔軟性」と定義した。グレーゾーンを小さくすることは変わることのない本質であり、いわば柔軟性の「原理・原則」と呼ぶことができる。なお、事前に想定できない仕様変更への対応において、設計や改修に画一的な方法を強制するのは現実的ではない。このため、柔軟性を確保するプロセスとして、設計や改修ではなく影響調査に着目する。

４．２　柔軟性を確保する手段の追究

図 14 は、仕様変更発生時における影響調査の対象及び問題点に関する、当分科会メンバー各社を対象とした現状調査の集計結果である。

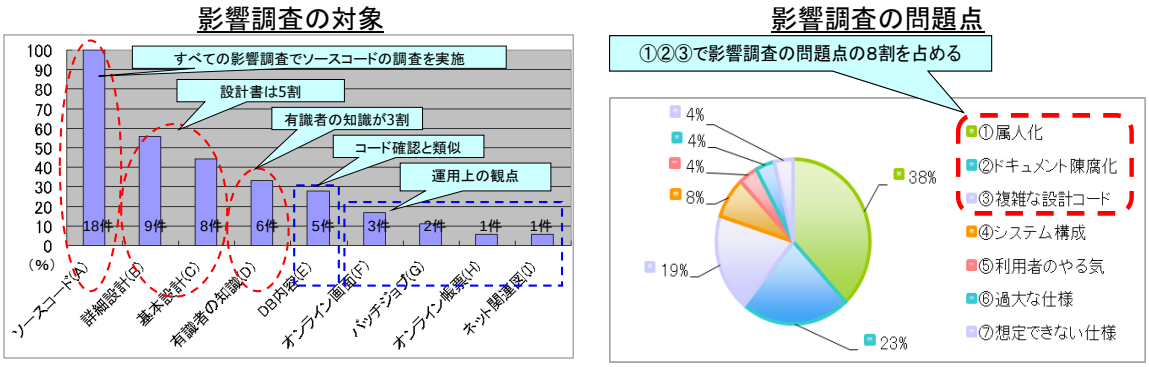


図 14 影響調査の現状に関する調査結果

これを見ると、影響調査の対象、問題点の各上位３件の関連性が浮かび上がってくる。まず、全メンバーが対象としたソースコードについては、コードが複雑で調査にコストがかかる問題が関連づけられる。同様に、設計書についてはドキュメントの陳腐化、有識者の知識については属人化の問題が関連づけられる。

これらの問題は、ツールによる生産性向上だけでは解決できない。なぜなら、仮にツールの機能でドキュメントに記載された設計情報を蓄積・管理する仕組みがあっても、その情報の質はそこに携わる人に依存するためである。現に、現状調査では開発ツールの機能によるソースコード検索の効率化事例や、ソースコード検索後の分析、設計、テスト工程までカバーする検索ツールの活用事例が見られたが、そこでも属人化やドキュメントの陳腐化といった問題はなくなっていない。

したがって、影響調査におけるグレーゾーンを小さくするためには、ツールではなく「ヒト」の行動を変える何らかの打ち手が必要である。当分科会では、影響調査においてグレーゾーンを小さくし、かかるコストと期間を最小化するための方法を明らかにするため、表３のとおり調査対象と問題点の各上位３件に絞って検討した。

対象	問題点	対応案	検討事項
ソースコード	複雑な設計コード	検索結果を少なくし、影響分析しやすくすることで調査時間を短縮する	検索キーワードの決定基準
設計書	ドキュメントの陳腐化	設計書に限らず、影響調査に使えるドキュメントを事前に準備する	開発工程で実施すべきこと
有識者の知識	属人化	有識者以外が適切に判断するための材料を揃える	有識者の持つノウハウの実体

表３ 影響調査の問題点と対応案

（１） 検索キーワードの決定基準

検索キーワードの決定基準に則って適切な検索キーワードを選定できれば、検索結果のヒット率（真の影響範囲が含まれる割合）が向上し、影響調査の時間短縮と精度向上の両方を実現できる。このため、当分科会ではデータベーステーブルなどの具体的な対象について、一定の検索キーワード決定基準を定めた。しかし、検索キーワード以上に重要なのは、適切な検索キーワードの決定と影響範囲の特定を実現する、図 15 の前提条件である。

ドキュメントが整備されていれば検索範囲を狭めることができ、用語が統一されていれば検索キーワードにブレがなくなることから、検索結果を少なくできる。更に、調査・分析プロセスが定義されていれば、有識者以外でも調査・分析できる。したがって、これらの前提条件を満たせば、誰でも影響調査におけるグレーゾーンを小さくできる。

✓設計書などのドキュメントが必ず存在し、追跡に必要な情報を確認できる
✓開発規約によってソースコードの変数名などに規則性があり、またドキュメントで使用する用語が統一されている
✓検索キーワード決定後の調査・分析プロセスが定義されている

図 15 検索キーワード決定の前提条件

（２） 開発工程で実施すべきこと

ドキュメントの陳腐化の問題に対して、設計書の陳腐化を防止するという対処は、必ずしも適切ではない。これは、次のような理由による。

- 現状調査において、保守開発時に改訂部分の設計書しか作成しておらず、新規開発時の設計書のメンテナンスが開発作業の実態から乖離しているケースが多く見られた。
- 対処にかかる工数が従来の開発工数にプラスされるだけでは、目立った効果が見込まれない限り定着しない。

このため、設計書の陳腐化防止という対処にこだわらず、「日々の開発作業の中でできること」の観点で、過去の影響調査漏れ事例などをもとに広く打ち手を洗い出した。

数多く挙がった打ち手の中には、テキストマイニング（出現頻度の高いものを検索キー

ワードとする）といった独創的なものも見られた。しかし、その精度を高めるためには事前の標準化が不可欠であり、それは他の打ち手にも共通して当てはまる。一方、有識者の立場からは、標準化のために整備されたドキュメントやルールを逐一確認するのは現実的でない。このため、ターゲットを「有識者以外による影響調査」とし、挙がった打ち手の中から、事例検証において影響調査の課題解決、導入負荷の軽減の面で効果の大きかった４つの打ち手に絞り込んだ（図 16）。

✓規約を整備し、用語を統一する
✓変更管理指針を標準化する
✓CRUD表を作成する
✓キャッチアップ資料（虎の巻）を作成する

図 16 開発工程での打ち手

追加工数の発生を考慮すると、これらのドキュメントや規約は開発を進めながら整備するのが理想である。しかし、それが難しければ導入負荷を抑える工夫が必要となる。例えばキャッチアップ資料（保守引き継ぎ資料）については、事後的にシステムを分析して作成することが可能であり、有識者自身ではなく有識者以外がヒアリングして作成することも可能である。これによって、最小限のコストで最大の効果を得られる。

（３） 有識者の持つノウハウの実体

有識者と一口に言っても、システム開発における有識者には様々な側面がある。メンバー自身が有識者である事例を振り返って特徴を列挙すると、次のようになる。

- システム構造を例外的な部分も含めて熟知している
- 独自の影響調査方法を持っている
- 開発時のドキュメントの内容や変更の経緯を熟知している
- 最適な検索キーワードを選定できる
- その他多くの暗黙知（人が経験や勘として蓄えている知識）を持っている

これらの特徴から、有識者は「特定のシステム開発に従事した経験からそのシステム構造に精通し、かつ対象業務領域の知識と関連づけられる人」と定義できる。これらの経験や知識が、有識者の持つノウハウの実体といえる。

有識者以外が影響調査を実施する場合、このノウハウがないことから効率的な情報収集や適切な判断ができず、有識者に依存せざるを得ない。これを解決するためには、有識者の持つノウハウの可視化、つまり「暗黙知の形式知化」が欠かせない。「キャッチアップ資料」は暗黙知の形式知化の一例であり、有識者の持つノウハウを引き継ぐ有効な手段である。ただ、引き継ぎの際は前述のような導入負荷を抑える工夫が求められる。

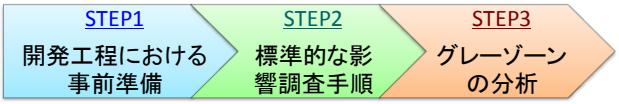


図 17 あるべき影響調査の流れ

当分科会は、「『誰がやっても同じ結果になる影響調査』は仕様変更に柔軟に対応できる」と考える。ここまでの検討過程をまとめると、このような影響調査は図 17 のステップをたどる。このうち STEP１と STEP２については、本研究で次の２点からなる影響調査手法を確立した。

- 仕様変更発生前の事前準備
- 仕様変更発生後の影響調査手順

一方の STEP３については、従来通り有識者が活躍する場となる。

４．３　検証

本節では、柔軟性を確保する手段として確立した影響調査手法の検証過程を述べる。

４．３．１　方法

メンバーの自社事例を用いた机上検証において、仕様変更発生前の事前準備を実施した場合としない場合を比較することで、影響調査のコスト削減と精度向上の効果を確認した。

（１）　事例の選定

対象が有識者以外であることから、メンバー自身が有識者でないものを優先的に選定し、メンバーが有識者の場合は社内の第三者に影響調査の実施を依頼することとした。

（２）　チェック項目

影響調査にかかる時間を始めとした定量的な比較に重点を置き、属人性の排除に関する定性的なチェック項目が補完する形で「検証チェックリスト」を作成した。仕様変更発生前の事前準備については、同チェックリストで打ち手ごとに評価できるようにした。

（３）　事前準備の適用方法

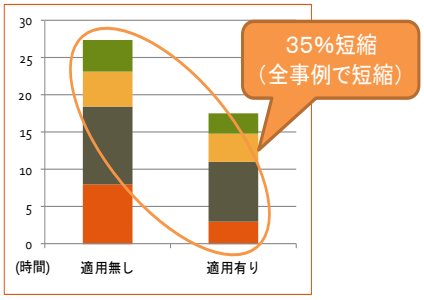
事前準備の打ち手については、検証事例の中で既に適用済みの場合もあれば、適用困難な場合もある。これを踏まえて、以下の方針を定めた。

- CRUD 表やキャッチアップ資料などの保守ドキュメントが既にある場合、事前準備を適用しないケースでは当該ドキュメントを参照せずに影響調査を実施する。
- 命名やコーディングなどの規約の整備、用語の統一ができていない場合、事前準備を適用するケースの確認に入る前に規約の整備及びプログラム修正を実施する。ただし、メンバー自身が有識者であり、かつシステム規模が大きくない場合に限る。
- 適用困難な打ち手については検証チェックリストにその旨記載し、全メンバーの検証結果を集約した時点で網羅されていれば検証済みとみなす。

４．３．２　結果

前述のチェック項目に対する結果を図 18 に示す。

影響調査にかかる時間の比較



属人性排除のチェック結果

チェック項目	YES	NO
１．手戻りしなかったか	80%	20%
２．効率化できたか		
（１）キーワードのヒット率が向上したか	80%	20%
（２）キーワードのブレをなくせたか	100%	0%
（３）対象範囲を限定できたか	100%	0%
（４）設計書を影響調査に使用したか	100%	0%
３．非有識者でも問題なく調査できたか	100%	0%

図 18 影響調査手法の検証結果

（１）　影響調査にかかる時間の短縮

仕様変更発生前の事前準備を適用した場合、すべての検証事例において影響調査にかかる時間の短縮が見られ、適用しない場合と比べて約 35%短縮された。これは、事前準備

の実施によってグレーゾーンが小さくなった結果と考えられる。

（２） 属人性の排除

すべてのチェック項目について一様に効果を確認でき、有用なドキュメント、有識者によるレビューの少なくとも一方があれば、影響範囲の限定や影響調査の品質向上を実現できることがわかった。中でも、保守ドキュメントが影響調査に極めて有用であることが、打ち手ごとの評価結果からわかった。なお、検証に際してこれらのドキュメントを新たに作成する負荷はそれほどかからず、導入負荷の面でも特に問題はないといえる。

（３） 問題点と対応

表４は、メンバーが検証中に気づいた問題点とその対応の検討内容である。これによって、影響調査手法の内容を更に充実させることができた。

主な問題点	対応
検索キーワードのヒット率が上昇したが、実施者が誤って影響範囲外と判定した	影響調査手順の影響分析フェーズの内容を具体化
キャッチアップ資料は便利だが、記載すべき項目の見極め方が難しい	キャッチアップ資料のサンプルを作成し、記載すべき項目を具体化
事前準備の打ち手が多く、中途半端に適用するとかえって影響調査漏れのおそれがある	事前準備を実施しても救えないケースを洗い出し、その結果をもとに打ち手の内容を具体化

表４ 影響調査手法の検証で見つかった問題点とその対応

このように、当分科会の確立した影響調査手法は影響調査の時間短縮、属人性排除に役立ち、有識者以外による精度の高い影響調査を実現する。したがって、２章で挙げた「影響範囲の限定」とそれによる「開発量の削減」という課題を解決し、仕様変更への柔軟な対応を可能にする。

５．新・開発ガイドライン

５．１ 新・開発ガイドラインの構成と役割

当分科会では、前章までに述べた拡張性、柔軟性の研究成果を「新・開発ガイドライン」にまとめた（図 19）。

新・開発ガイドラインは、仕様変更への柔軟な対応のために必要な取り組みのうち、一般的な開発標準に則って従来通り開発を進める中で実施可能なプラクティスに特化している。したがって、企画から運用までを包括的に扱う SDEM などの一般的な開発標準に取って代わるものではなく、補完するものである。

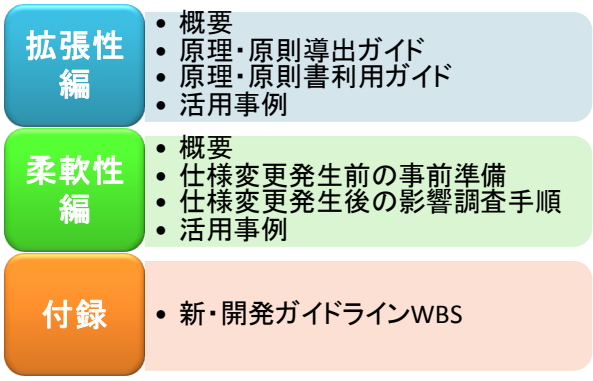


図 19 新・開発ガイドラインの構成

５．１．１ 拡張性編

システムに拡張性を持たせるため、開発工程において原理・原則を発見して設計に反映させるとともに原理・原則書に記載し、繰り返し発生する仕様変更において活用する手順をまとめたのが、「新・開発ガイドライン（拡張性編）」である（以下拡張性編という）。

その内容は次のとおり。

（1） 原理・原則導出ガイド

原理・原則導出モデル（3章参照）の「要件」「理由」「仕様」「概念モデル」「データモデル」「ルールコンテンツ」の各要素から原理・原則を発見し、原理・原則書に記載するまでの手順、具体例、実施すべき工程などを記載している（図 20）。

図 20 原理・原則導出ガイド（サンプル）

（2） 原理・原則書利用ガイド

原理・原則書の雛形を掲載し、作成後の活用方法を記載している。更に、繰り返し発生する仕様変更における活用と見直し、開発プロジェクトの特性に応じたカスタマイズにも言及している。

（3） 活用事例

モデルプロジェクトを用いて、原理・原則を導出する一連の作業手順を示している。

図 21 に拡張性編の活用イメージを示す。

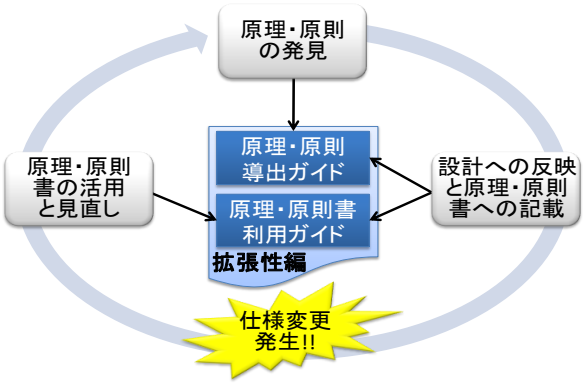


図 21 拡張性編の活用イメージ

原理・原則は誰でも発見できるものではない。導入当初は、業務に精通した有識者が拡張性編を用いて、強い意志を持って原理・原則の発見に取り組むべきである。その後、プロジェクト全体に浸透させる過程において、拡張性編が更に威力を発揮することになる。

５．１．２　柔軟性編

事前に想定できない仕様変更に対応する上で必要な柔軟性を確保するため、その鍵を握る影響調査プロセスに有用な打ち手をまとめたものが、「新・開発ガイドライン（柔軟性編）」である（以下柔軟性編という）。その内容は次のとおり。

（１）　仕様変更発生前の事前準備

次の三つの観点で有用な打ち手を、拡張性編と同じガイド形式（図 20）で解説している。

- 影響調査漏れによる手戻りの防止
- 影響調査の効率化
- 影響調査の属人化の防止

このガイドでは、手順、具体例、実施すべき工程のほか、実施担当者及び主な適用対象（新規／既存）を明示している。

（２）　仕様変更発生後の影響調査手順

標準的な影響調査手順をフローチャートで示している。各手順を選択すると、作業内容や具体例の詳細を確認できる。また、事前準備のうち各手順への効果の大きいものを「推奨する事前準備」に記載している。これによって、ドキュメントや規約の影響調査における活用のタイミングがわかるほか、事前準備の段階的な適用の検討材料にもできる。

（３）　活用事例

モデルプロジェクトを用いて、適用範囲の異なるシナリオ（影響調査手順のみ、影響調査手順と事前準備の両方）や既存システムへの適用シナリオを示している。

図 22 に柔軟性編の活用イメージを示す。

事前準備の打ち手は、各社で類似した取り組みがあるものがほとんどである。しかし、その打ち手が適切な影響調査の実施とそれによる柔軟性の確保に有用であることを認識し、この柔軟性編を各社の実状に合わせてカスタマイズしながら適用することが望ましい。

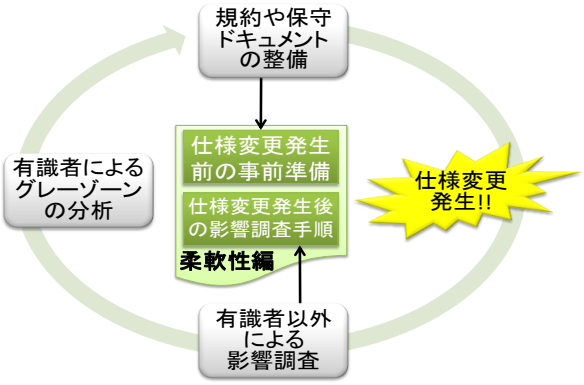


図 22 柔軟性編の活用イメージ

５．１．３　新・開発ガイドラインWBS

新・開発ガイドラインでは、付録として図 23 の標準 WBS（Work Breakdown Structure）

ID		タスク名	期間	開始日	終了日	先行タスク
1		VP	10日	10/04/01 (木)	10/04/14 (水)	
2		原理・原則書の作成	10日	10/04/01 (木)	10/04/14 (水)	
3		要件欄の記載	2日	10/04/01 (木)	10/04/02 (金)	
4		理由欄の記載	2日	10/04/05 (月)	10/04/06 (火)	3
5		原理・原則欄の記載	2日	10/04/08 (木)	10/04/09 (金)	4
6		課題の検討	3日	10/04/12 (月)	10/04/14 (水)	5
7		SP	10日	10/04/15 (木)	10/04/28 (水)	1
8		原理・原則書の作成	10日	10/04/15 (木)	10/04/28 (水)	
9			2日	10/04/15 (木)	10/04/16 (金)	6
55		IT	15日	[illegible]	[illegible]	
56		ST	20日	10/09/06 (月)	10/10/01 (金)	
57		キャッチアップ資料(虎の巻)の作成	20日	10/09/06 (月)	10/10/01 (金)	55

図 23 新・開発ガイドライン WBS

を掲載している。拡張性編や柔軟性編の有用性は理解できても、実際の開発では追加工数の懸念から導入が進まないケースが想定される。新・開発ガイドラインの各プラクティスを図 23 のような WBS のタスクに落とし込み、開発プロジェクトの WBS と併せてスケジューリングすれば、追加工数の発生を最小限に抑え、導入しやすくなると考えられる。

５．２　新・開発ガイドラインの評価

「新・開発ガイドライン」の有用性を客観的に評価するため、メンバー各社においてアンケートを実施し、延べ 50 名超の方にご回答いただいた。回答者は入社後数年の若手から 30 年前後のベテランまで幅広く分布し、担当開発工程も網羅している。

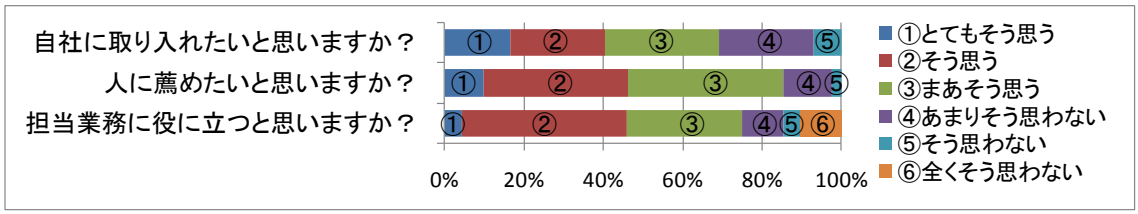


図 24 アンケート質問項目及び回答結果

図 24 の回答結果から、回答者の 74%が新・開発ガイドラインを担当業務に役立つと考えており、85%が人に薦めたいと考えていることがわかった。しかし、自社への適用については、適用したいとの回答は 68%に留まった。ただし、その主な理由は、社内標準としての採用には厳密な検討が必要であり、すぐには適用できないためであった。したがって、新・開発ガイドラインの有用性を否定するものではないと考えられる。

このように有用性に関して評価を得られた一方で、個々の内容への指摘もあった。各指摘事項については分類・整理し、活用事例を増やす、文章表現を平易にするといった見直しを実施した。これによって、更に適用しやすい研究成果とすることができた。

６．適用に向けて

６．１　研究終了後の成果の適用状況

研究成果の実際の開発プロジェクトへの適用については、研究期間中も常に念頭に置いて議論を重ねた。一般的に、新たな開発手法を提唱しても既存の開発プロジェクトへの適用は困難と受け取られる。しかし、一つの案件で扱う機能単位での段階的な適用は可能と当分科会は結論づけた。この結論を現実のものとするため、新・開発ガイドラインには標準 WBS を掲載し、開発プロジェクトの特性に合わせたカスタマイズを推奨している。

研究終了後の現在は、原理・原則（拡張性、柔軟性）に基づく開発手法のメンバー各社への展開フェーズに入っている。メンバーの A 氏が PM を務める開発プロジェクトでは、「原理・原則」が既にプロジェクトの共通用語として浸透している。以下、拡張性、柔軟性それぞれの適用事例を紹介する。

６．１．１　拡張性の適用事例

３章で例として取り上げた発送支援システムは、メンバーの K 氏が実際に担当している

案件である。研究期間中はホストコンピュータで動作する既存システムを用いてシミュレーションしたが、現在は PC で動作するシステム（以下新システムという）への移行のための開発を実際に進めている。仕様変更の要件である「直販企業顧客対応」については、現行では手作業となっているものを、新システムでは機能としてサポートする。図 25 のように原理・原則導出モデルを活用して原理・原則を発見し、設計に反映させることによって、この機能追加に対しても必要最低限の開発量で対応できている。

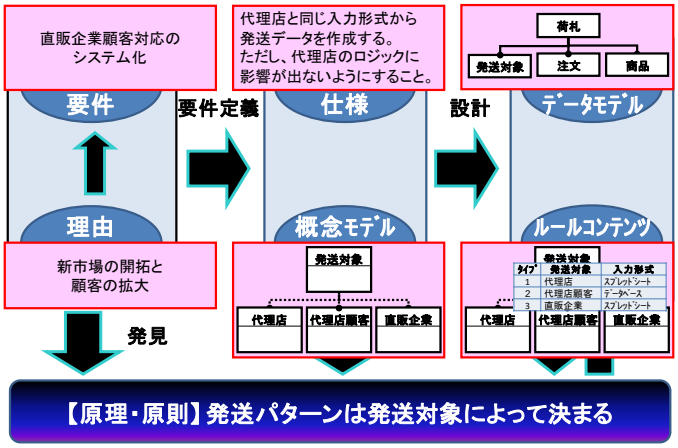


図 25 原理・原則導出モデルの活用

現在は拡張性を作り込む段階のため、コスト削減効果を定量的に示すことはできない。ただ、本研究のターゲットであるリリース後の仕様変更だけでなく、開発中の仕様変更にも短時間で対応できており、既にその効果を実感しつつある。今後の新システムへの仕様変更において、更に目に見える効果が現れることを期待している。

６．１．２　柔軟性の適用事例

図 26 は、メンバーの B 氏が参画した製品 F プロジェクトの問題解決を表したものである。

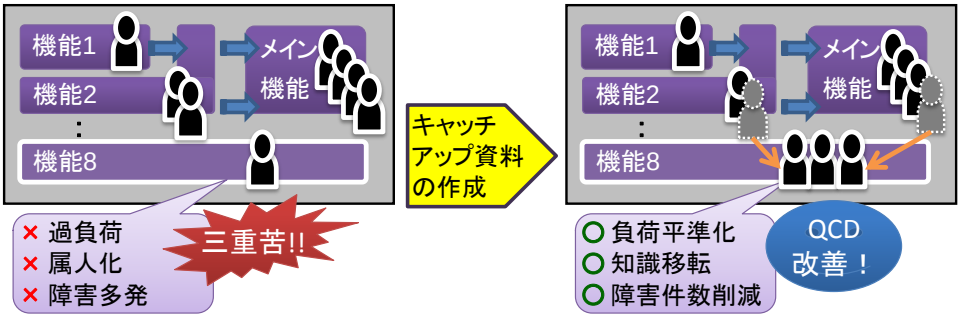


図 26 キャッチアップ資料の活用

製品 F は一般に販売されており、エンドユーザーからの要望による仕様変更がある。メイン機能と機能１～８のうち、機能８は独立したツールの位置づけとなっており、改善要望が多いことから仕様変更が頻発する。これに対して機能８の担当が１名であったため、次の問題を認識しながら改善できない状況が続いていた。

- 常に作業負荷が高く、残業が月 35 時間（他のメンバーは月 15 時間）と突出している。
- 対象機能に関する知識が属人化し、他のメンバーには分担できない。
- 障害件数が多く、その根本原因の７割は影響調査漏れからくる考慮漏れである。

しかし、このままでは QCD の低下が深刻化してしまう。このため、機能８について影響調査の事前準備の一つであるキャッチアップ資料を作成し、他のメンバーによる分担に活用した。この打ち手によって、以下の改善が見られた。

- 残業時間が平準化した（機能８の担当が月 25 時間、他のメンバーが月 17 時間）。

● キャッチアップ資料によって、機能８に関する暗黙知の形式知化と共有が進んだ。
● 影響調査漏れからくる考慮漏れによる障害を全体の３割まで削減できた。

この事例から、柔軟性の研究成果である影響調査手法は、影響調査の精度向上だけでなく、作業負荷の軽減や知識の移転にも有用であることがわかる。

６．２　残された課題

２章で掲げた課題である「ロジック修正量の削減」「影響範囲の限定」とそれによる「開発量の削減」は、本研究成果によって解決できる。仕様変更対応事例を用いた検証においても、このことは実証されている。新・開発ガイドラインに関するアンケートでは導入の難しさを指摘されたが、回答として標準 WBS の活用や段階的な導入を提案している。したがって、当分科会は仕様変更に柔軟に対応できる開発手法を確立したと認識している。

それでもなお、研究終了後にいただいた評価では実際の開発プロジェクトへの適用に関する指摘が散見された。これに対しては、前述のようなメンバー各社への適用実績を積み上げていくしかないと思われる。ただ、メンバー各社へ展開する中で新・開発ガイドラインについて改善すべき点が見えてきていることも事実である。この「残された課題」は、主に次の２点である。

（１）作業内容の具体化

メンバーの A 氏のプロジェクトで新・開発ガイドラインを活用したところ、「教育資料にはなるが開発の実践的な部分が不足している」という意見が挙がった。新・開発ガイドラインでは、原理・原則（拡張性、柔軟性）に基づく開発手法の根本的な考え方を伝えることに重点を置いており、開発作業レベルの詳細はプロジェクトの特性に合わせたカスタマイズに委ねている。これはプロジェクトの多様性に配慮したものであるが、更に開発作業レベルに踏み込んだ内容に改善していけば、より適用しやすいガイドラインになる。

（２）拡張性と柔軟性の融合

本章においても適用事例を拡張性と柔軟性に分けて紹介しているように、当分科会では拡張性と柔軟性を常に「別物」として扱ってきた。しかし、拡張性と柔軟性は本来密接に関連する概念である。

図 27 に示すとおり、原理・原則を反映したシステム構造としておけば、仕様変更へ対応する際に影響範囲を限定しやすくなる。つまり、拡張性を確保することで柔軟性ももたらされる。両者を融合させることによって、仕様変更への対応における QCD の課題をより高い次元で解決することが期待できる。

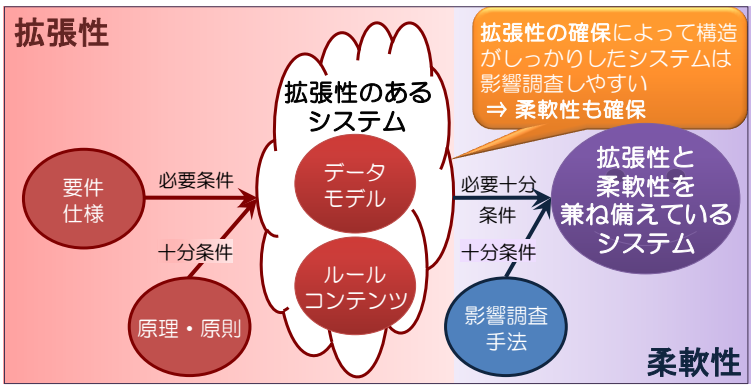


図 27 拡張性と柔軟性の融合

新・開発ガイドラインについても、拡張性編と柔軟性編を独立させるのではなく、両者の関連性を盛り込んだ内容に改善していけば、より有用性の高いガイドラインになる。

７．おわりに

「仕様変更」は深遠なテーマであり、１年余りの研究ですべてを見通すことは不可能である。ただ、原理・原則に基づく開発手法を確立する過程で「本質を見抜こうとする心」の重要性を実感できた。これは、変化の激しい現代において、あらゆるシーンに当てはまる「原理・原則」ではないだろうか？今後も物事の本質を見誤ることなく、本研究成果を活用しながら継続的に改善し、成長させていきたい。

参考文献

［１］調査研究｜JUAS 社団法人 日本情報システム・ユーザー協会、企業IT動向調査2010：
http://www.juas.or.jp/servey/it10/index.html

［２］非ウォーターフォール型開発に関する調査　調査報告書：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、2010年3月

［３］反復型開発における見積もりの実際 - 上流工程-見積もり：ITpro：
http://itpro.nikkeibp.co.jp/article/COLUMN/20060811/245759/

［４］アジャイル開発と反復開発の落とし穴 ― @IT自分戦略研究所：
http://jibun.atmarkit.co.jp/lskill01/rensai/takumi/04/01.html

［５］Walker Royce、Kurt Bittner、Mike Perrow著、藤井 拓監訳：反復型開発のエコノミクス 業績を改善するソフトウェアプロジェクト管理、ピアソン・エデュケーション、初版１刷、2009年12月

［６］アーキテクチャ ジャーナル - 第 15 号　アーキテクトの役割 -：
http://msdn.microsoft.com/ja-jp/architecture/cc811463.aspx

［７］APMサービス（アプリケーション運用・保守）　富士通：
http://fenics.fujitsu.com/outsourcingservice/apm/

［８］Jimmy Nilsson、長尾 高弘訳：ドメイン駆動 デザインパターンへの適用法、翔泳社、初版１刷、　2008年3月

［９］要求開発入門：要求開発アライアンス、2007年1月

［１０］Pavel Hruby、依田 智夫監修、溝口 真理子・依田 光江訳：ビジネスパターンによるモデル駆動設計、日経BPソフトプレス、初版、2007年8月

［１１］Business Rules Group、Defining Business Rules ~ What Are They Really?：
http://www.businessrulesgroup.org/first_paper/br01c0.htm

［１２］Karl E.Wiegers、渡部 洋子監訳：ソフトウェア要求 顧客が望むシステムとは、日経BPソフトプレス、初版４刷、2008年7月

［１３］SDEM標準プロセス説明書 1.0.1版、富士通、2007年9月

［１４］C.Alexander著、宮本 雅明訳：オレゴン大学の実験、鹿島出版会、1977年

［１５］INTARFRM　富士通：
http://jp.fujitsu.com/solutions/afw/

［１６］野中 郁次郎、竹内 弘高著、梅本 勝博訳：知識創造企業、東洋経済新報社、初版20刷、2010年6月