ソフトウェア変更影響の可視化手順書

日科技連 第 20 年度ソフトウェア品質管理研究会
第 7 分科会 C グループ

目　次

1. はじめに

1.1. 目的

近年、ソフトウェアはハードウェアの性能向上や企業間競争の激化から、多機能化、多品種化、短納期を求められている。それらの要求を満足するために共通のソースコードを再利用した派生機種開発が盛んに行われているが、ソフトウェアのテスト工数は増大の一途を辿っている。

再利用型ソフトウェア開発においては、変更点に着目してソフトウェア開発を効率化することが重要となる。テストにおいても、変更点、およびその影響範囲に着目し、適切なテスト範囲決定や変更の種類に応じた適切なテスト方法選択が重要となる。

本手順書は、ソフトウェア変更影響の可視化手順と、その可視化した情報を利用してテストすべき項目を選定し効率的にテストする方法を、明確にすることを目的としている。

本手順書では、３章でソフトウェア開発の設計情報を用いて変更影響を可視化する手順を示す。また、４章で可視化された情報を用いての影響範囲の特定方法およびテスト範囲の決定方法についての考え方を示す。変更影響の可視化は、品質機能展開の二元表を用いることで実現している。

1.2. 適用

本手順書が適用できるプロジェクトは、オブジェクト指向設計手法または構造化設計手法を用いて設計されているソフトウェアを再利用して、機能の追加，変更，削除などを行い開発しているプロジェクトとする。

また適用範囲としては、単体テスト工程、結合テスト工程、およびシステムテスト工程とし、対象者としては、上記工程のテスト担当者とする。

1.2.1. 必要な資料

本手順書に必要な資料を表 1-1 に示す。

表 1-1 資料分類と成果物の例

資料分類	設計手法	成果物の例
顧客要求を分析した資料	オブジェクト指向設計	ユースケース図（ユースケース記述）
	構造化設計	機能仕様書，要求分析表
システム内部の設計情報を表した資料	オブジェクト指向設計	パッケージ図，クラス図，シーケンス図，ステートチャート等の UML ダイアグラム
	構造化設計	CD，DFD，部品リスト，部品説明書，関数一覧
顧客要求と設計情報の関連を表した資料	オブジェクト指向設計	コラボレーション（ユースケースの実現），機能仕様書
	構造化設計	機能仕様書，使用部品一覧表
設計情報と設計情報の関連を表した資料	オブジェクト指向設計	関係(関連，集約等)，メッセージ
	構造化設計	関数一覧，関数設計書，関数コールグラフ

1.2.2. 用語説明

本手順書で使用している用語の定義は、以下の通りである。

- 単体テスト
  モジュール単体で行うテスト。
- 結合テスト
  複数のモジュールが結合された状態で行うテスト。
- システムテスト
  システムとして統合された状態で行うテスト。
- 二元表
  QFD（Quality Function Deployment：品質機能展開）における二元表のことで、本手順書では R-C(Requirement -Component：顧客要求ー システム内部の構造)、および C-C(Component-Component)の関係を抽出している。
  二元表の例を図 1-1 に示す。

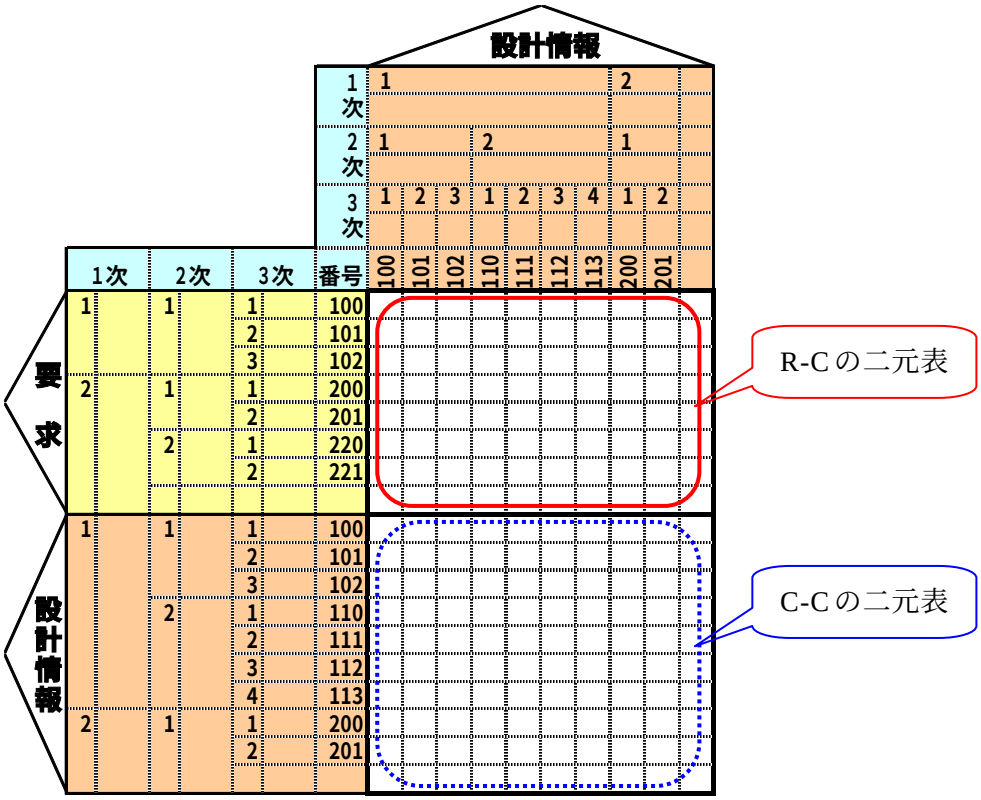

図 1-1 「二元表」の例（３次ｘ３次）

- 機能要求
  顧客要求のうち、ユーザから見える要求であり、システム間の相互作用（何をするかなど）を記述したものである。反対に、非機能要求とは、システム上の制約であり、システム化に対する要求（制約条件など）を記述したものである。

- オブジェクト指向設計
  顧客要求を、データ「属性」と処理「操作」を一体化（カプセル化）したオブジェクトとして設計していく手法。
- 構造化設計
  顧客要求を、DFD などを利用して図示することにより、段階的に詳細化（モジュール分割）していく設計手法。ボトムアップにより段階的に合成していくこともある。
- DFD（Data Flow Diagram：データフローダイアグラム）
  システム間のデータの流れを示す図。データ項目，処理，データストア，外部エンティティなどを表す図形で構成され、データがどのような処理に使用されているかを示す。
- CD（Context Diagram：コンテキストダイアグラム）
  構造化設計の段階的詳細化における、第 0 レベルの DFD のこと。
- 重みづけ
  定量的な情報で影響の大きさを表現すること（定性的な情報でも可）。
- ゴール分解
  顧客要求を、具体的な要求項目へと徐々に分解し詳細化していく要件獲得手法。

2. 変更影響可視化の考え方

2.1. 変更影響可視化の方針

一般的に変更影響は、DFD やクラス図, 関数コールグラフなどの依存関係をグラフとして表現している文書情報から得ることができる。変更影響の伝播はグラフ上の変更点を起点として、辺を辿っていくことで認識できる。一方、グラフの読解には、それぞれの分野の表記方法に関する専門的なスキルが必要となる。また、グラフを直接比較し変更点を検出することは困難である。

一般に、グラフによる依存関係は表にマップすることが可能である。個々の専門的なグラフによる表現から、依存関係のみに着目した統一的な表による表現に変換する方法を定義することにより、グラフの表記方法に専門的なスキルを持たない人でも、依存関係にもとづいた変更影響を認識することが可能となる。

依存関係を表す表の一つに QFD の二元表がある。二元表には以下のような特徴がある。

- 表のある軸に着目して別の表に展開することにより、表をまたがる依存関係を保持しつつ様々な視点での分析が可能である。
- 表の各軸は階層構造を持っており、各軸項目の粒度を変えた分析が可能であり、直感的に理解できかつ特別な手法を覚える必要がない。

ある文書にグラフ表現されている依存関係は、一つの二元表として表現することができ、文書間の依存関係の関連は、二元表を展開することで表現することができる。また、変更点は変更前後の二元表で内容が変化したセルとして認識できる。

したがって、本手順書では依存関係を QFD の二元表を用いて表現し、変更前後の二元表を比較することにより、変更影響を可視化する。

2.2. 二元表における依存関係の表現

本手順書では、変更影響の可視化において、顧客要求をシステム内部の構造に割り付ける R-C の二元表とシステム内部の依存関係を表現する C-C の二元表の２つを基本構成と考える。

これにより、顧客要求からのシステムに対する変更影響および、システム内部の影響範囲の伝播を把握する。

単純に依存関係の有無を表現する場合には、依存関係の方向に応じて横軸から縦軸（もしくはその逆）の交点に○をつけることにより表現する。通常は対象物により依存の種類を依存記号として定義し、その強さを数値化して影響の大きさを表現することになる。図 2-1 に R-C, C-C の二元表のイメージを示す。図 2-1 では「R3→C3」,「C3→（C2, C4)」,「C4→(C5, C6)」の依存関係を示している。

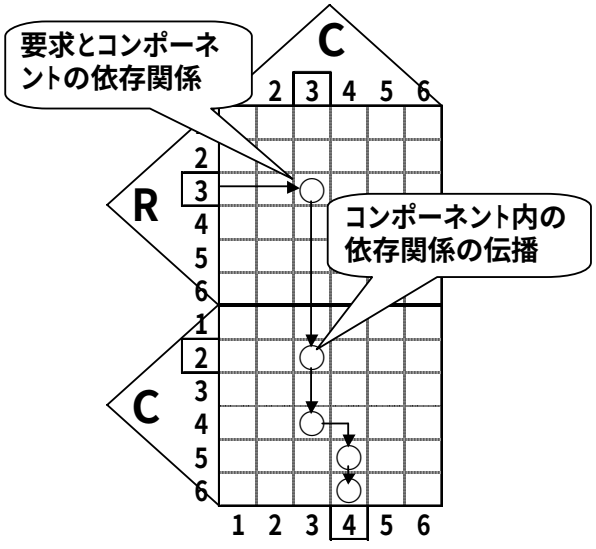

図 2-1 二元表のイメージ

2.3. 二元表における変更影響の特定

変更影響は、変更前後の二元表をセルごとに差分表示することにより認識する。

「R-C」の二元表では、顧客要求の変化にもとづき、再利用システムの中から変更が必要な設計要素を見つけ出し、直接的な影響範囲（１次の影響範囲）を特定する。

「C-C」の二元表では、システム内の設計要素の依存関係を再帰的に辿ることにより、間接的な影響範囲（ｎ次の影響範囲）を特定する。影響の強さは、依存記号から得られる影響の強さから影響範囲として特定するしきい値を設けることが可能である。これにより、KKD（勘、経験、度胸）に依存しない変更影響範囲の特定手順を定義することができる。

変更影響の特定手法の詳細については、4.2.1.二元表を用いた変更影響の可視化の項で述べる。

3. 二元表の作成手順

3.1. オブジェクト指向設計

3.1.1. 顧客要求の階層化

オブジェクト指向では、顧客要求の表現方法として UML のユースケースがある。1 つのユースケースは、利用者の視点での 1 つのシステムの機能を表している。ユースケースでは、要求の階層化はステレオタイプ「≪include≫」をつけた依存関係として表現される。矢印の先は包含されるユースケースを示すため、これを二元表の要求階層として転記する。

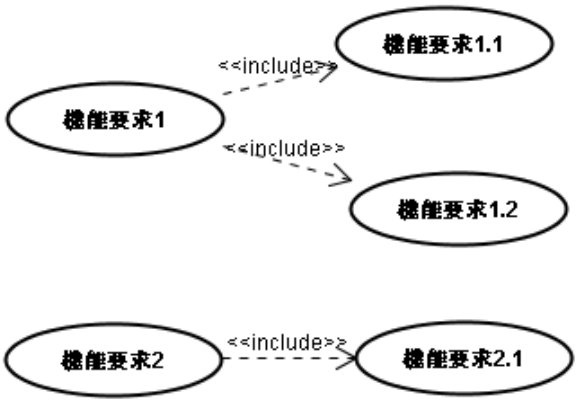

図 3-1 階層化されたユースケースによる機能要求の抽出例

ユースケースでは、前述した通り、顧客要求のうち機能要求にあたる部分を表現する。したがって、顧客要求を二元表にマップするには、オブジェクト指向に特化しない非機能要求も整理し、要求階層に含める必要がある。これらの方法には、例えばゴール分解などがある。

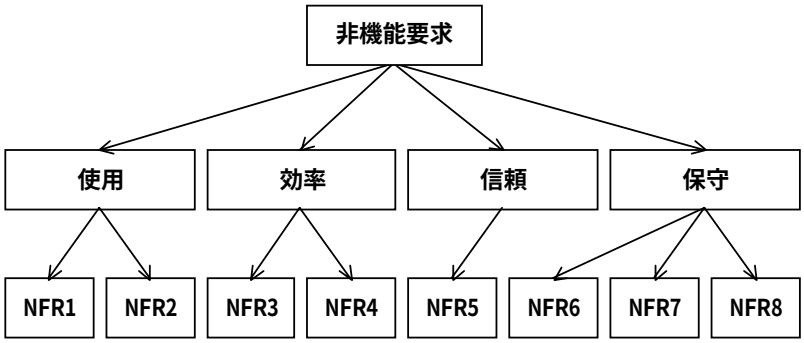

図 3-2 ゴール分解による非機能要求の抽出例

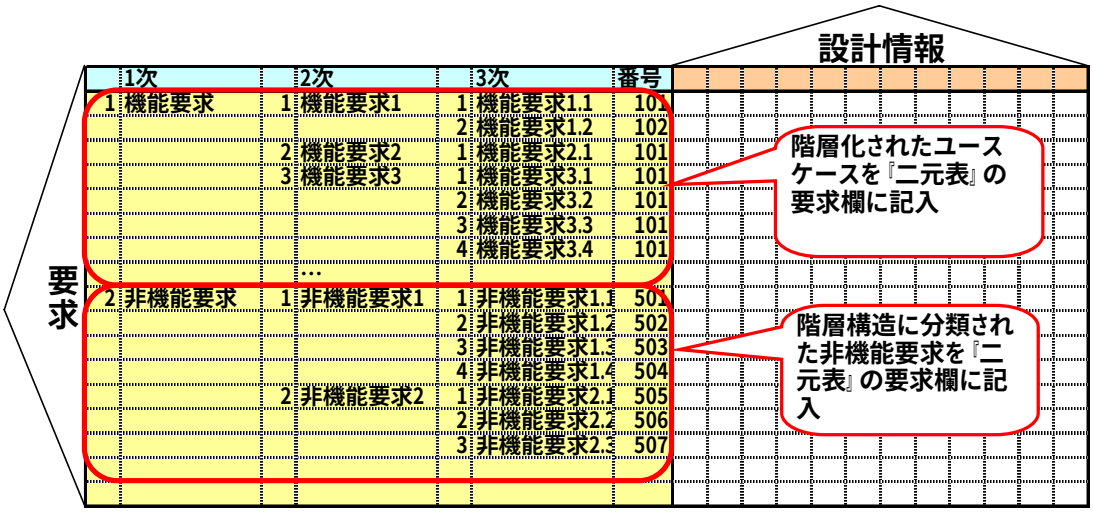

図 3-3 「二元表」の要求欄への記入例

3.1.2. 設計情報の階層化

オブジェクト指向では、システムの内部構造の表現方法としては UML のパッケージ図，クラス図などの静的モデルがある。パッケージ図はクラス図等の UML の要素をまとめるために用いられる。これを二元表の要求階層として転記する。

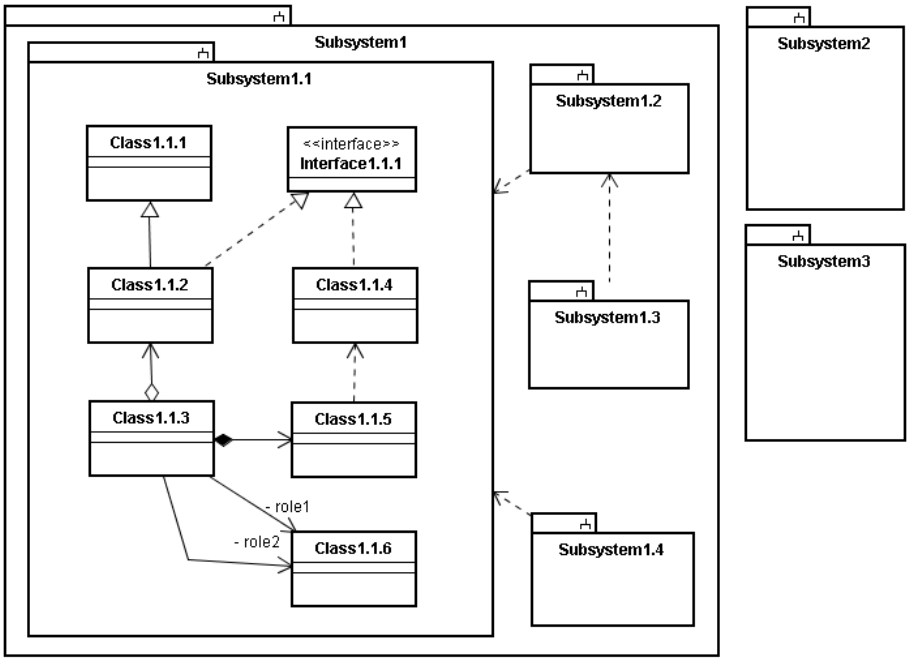

図 3-4 「パッケージ(サブシステム)図，クラス図」の例

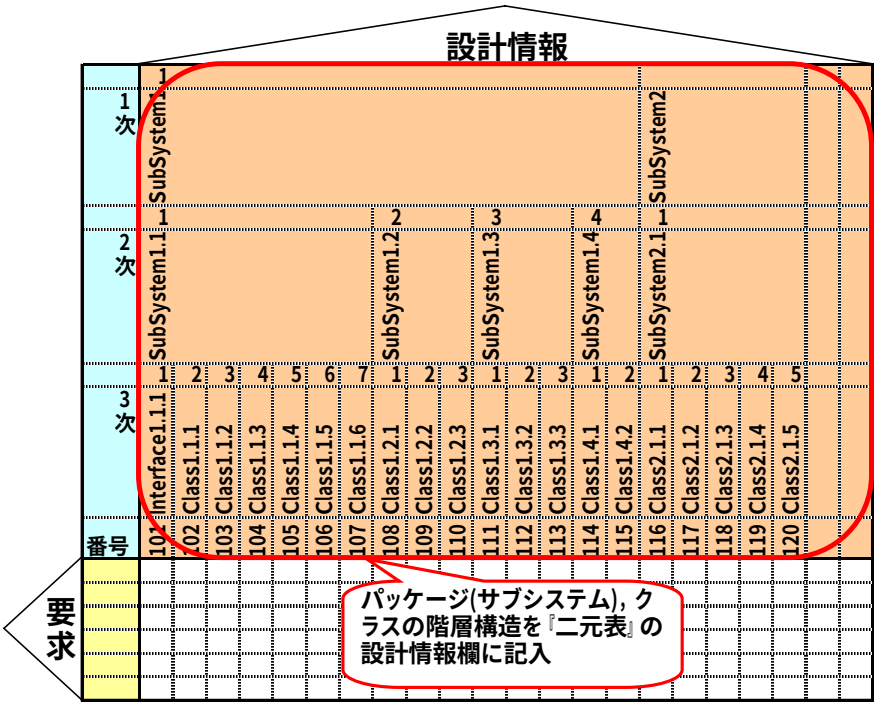

図 3-5 「二元表」の設計情報欄への記入例

3.1.3. 関連の割り付け

3.1.3.1. 要求仕様と設計情報の関連を入力

要求仕様と設計情報の関連は、UML のユースケースとコラボレーションを用いた「ユースケースの実現」により表現される。実際の仕様書としては「ユースケースの実現」として表現されていない場合もあり、その場合には、ユースケース、およびユースケース記述の説明として機能仕様書などに含まれるある 1 つのビューとしてのクラス図やそのオブジェクトの振る舞いを示したシーケンス図等から、ユースケースと個々のクラスを関連づける。

「≪include≫」関係を用いたユースケースの要求階層は、上位の階層のサブセットが下位の階層にあらわれるため、上位の階層と直接結びつける場合は、対応階層ごとにマーキングの種類を変える。例えばユースケースが最大 3 階層となっている場合は、1 階層目と関連するコンポーネントは「△」、2 階層目と関連するコンポーネントは「○」、3 階層目と関連するコンポーネントは「◎」とする。あるパッケージと 1 階層目のユースケースが対応づけられるような場合は、ユースケースとそのパッケージ全体が「△」に対応づけられることになる。

非機能要求要求部分についても同様のマーキングをする。非機能要求については横断的（アスペクト）に対応づけられるため、マーキングが散在することになる。

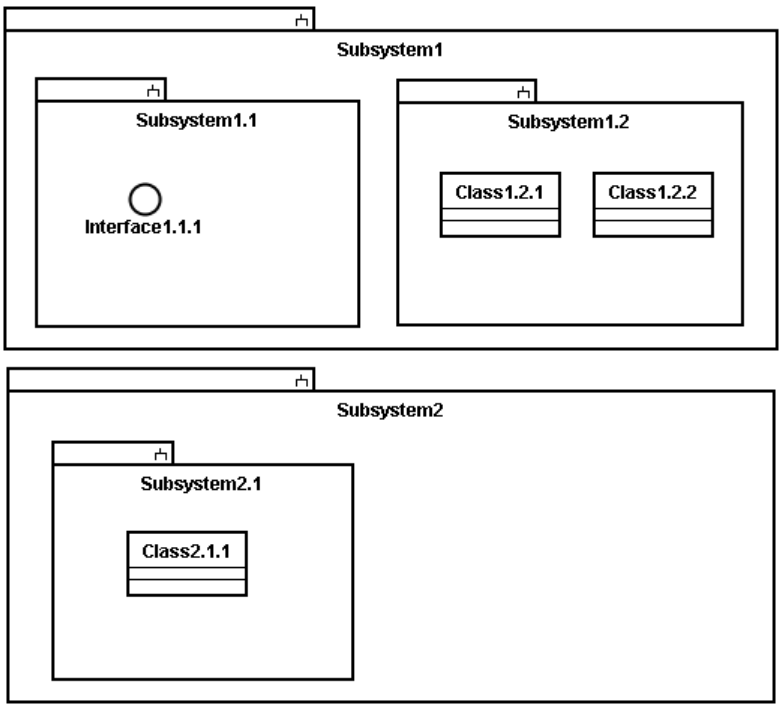

図 3-6 機能仕様書に含まれるダイアグラムから抽出されたクラス群

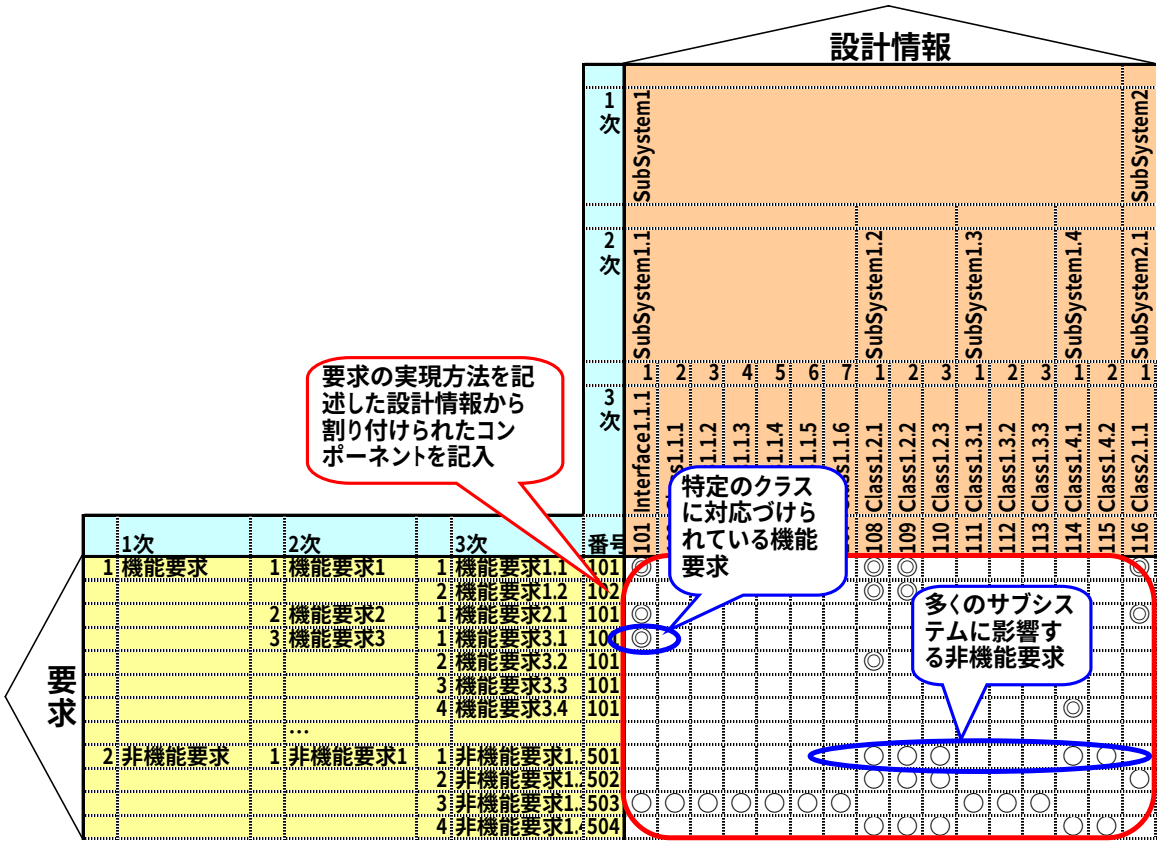

図 3-7 「二元表」要求と設計情報の関連の記入例

3.1.3.2. 設計情報と設計情報の関連を入力

設計情報同士の関連の 1 つに、UML の「関係」がある。「関係」には表 3-1 に示す種類があるため、それぞれに略号を割り当てる。表 3-1 に例を示す。

表 3-1 UML での「関係」の種類と「二元表」での表現方法例

関係の種類(日本語)	依存記号
Association（関連）	a
Aggregation（集約）	b
Composition（コンポジション）	c
Dependency（依存）	d
Generalization（汎化）	g
Realization（実現）	r

要素間に複数の関係がある場合は、1 つのセルへ複数の略号を入力する。これにより、クラス間がどのような関連で結ばれているかを二元表上に表現することができる。

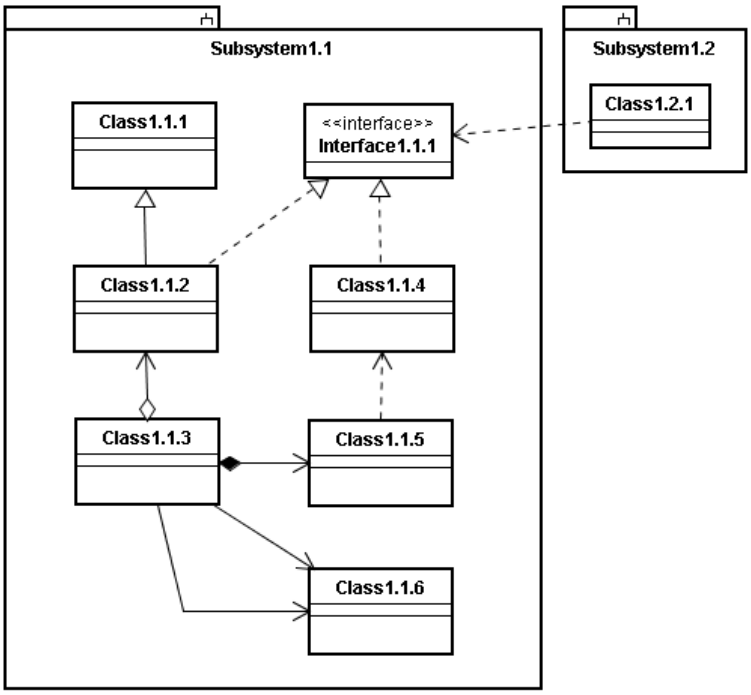

図 3-8「クラス図」に記述された関係の例

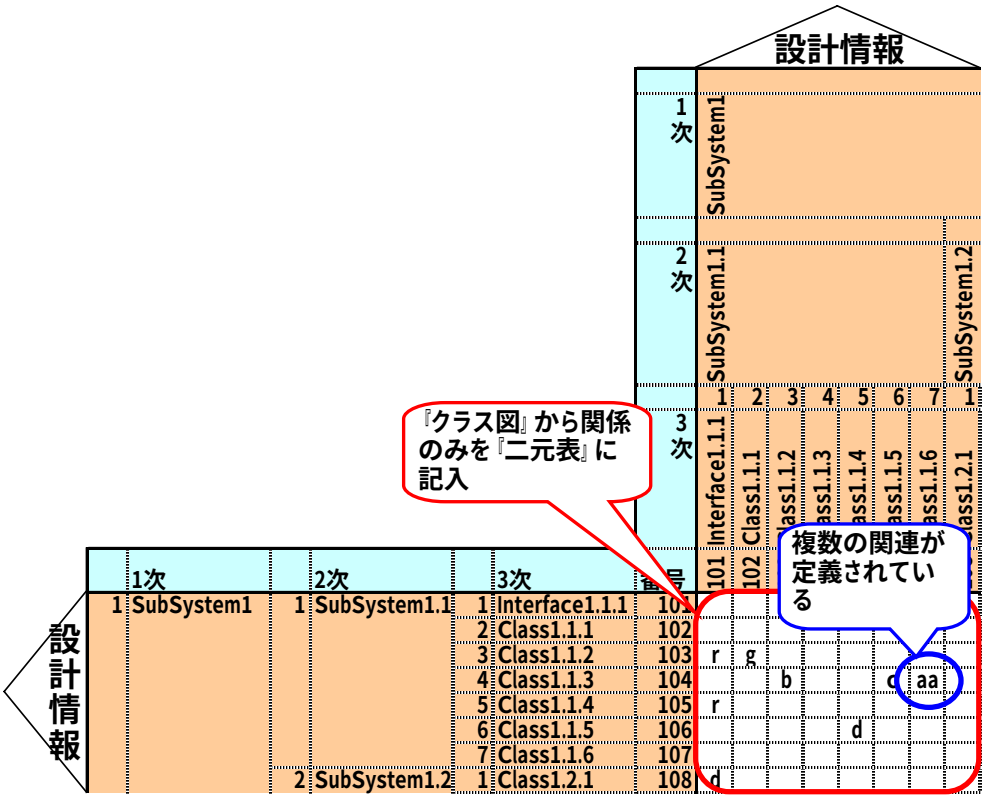

図 3-9「二元表」設計情報と設計情報の関連の記入例

3.2. 構造化設計

3.2.1. 顧客要求の階層化

構造化設計の例として、標準のフルセットシステムから必要な機能をカスタマイズしていくような場合に使用している「要求分析表」を、顧客要求を階層化するための入力情報とする。要求分析表とは、要求分析工程において機能要求と非機能要求を分類し階層化した資料のことである。「要求分析表」の例を表 3-2 に示す。

表 3-2「要求分析表」の例

	大分類		中分類		小分類	番号	有無	入力	出力	
1	グループA	1	サブグループA-1	1	要求A-1-1	100				
				2	要求A-1-2	101				
				3	要求A-1-3	102				
				4	要求A-1-4	103				
				5	要求A-1-5	104				
2	グループB	1	サブグループB-1	1	要求B-1-1	200				
				2	要求B-1-2	201				
				3	要求B-1-3	202				
		2	サブグループB-2	1	要求B-2-1	220				
				2	要求B-2-2	221				
				3	要求B-2-3	222				
				4	要求B-2-4	223				

表 3-2 の大分類，中分類，小分類を、「二元表」の要求欄（図 3-10 の赤枠）の 1 次から 3 次に順次記入することで、顧客要求を階層化することができる。

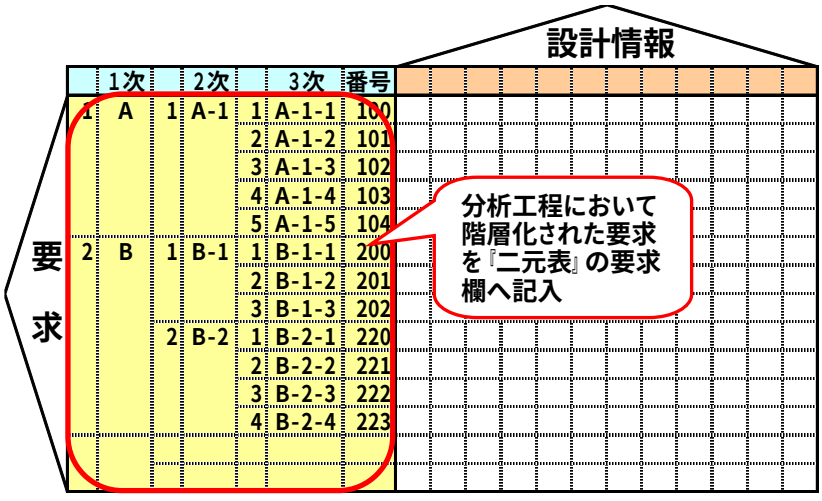

図 3-10「二元表」の要求欄への記入例

このように、ソフトウェアに要求される機能を分類した情報を利用して、顧客要求を階層化することができる。

3.2.2. 設計情報の階層化

構造化設計の例として、設計の各工程で生成された各種の情報（顧客要求を機能分割した「CD」,「DFD」や、各部品の持つ関数を示す「関数一覧」など）を、設計情報を階層化するための入力情報とする。設計情報を用いて「二元表」の設計情報欄（図 3-12 および図 3-13 の赤枠）の 1 次から 3 次に記入する方法を記す。

① 設計情報欄の 1 次：機能名の記入例

図 3-11 に示す「DFD」 を入力情報とし、機能 A や機能 B などで示される機能名を抽出し、その名称を「二元表」の設計情報欄の「１次」の項目へ記入する。

② 設計情報欄の 2 次：部品名の記入例

同様に、図 3-11 の部品 A や部品 B などで示される部品名を抽出し、その名称を「二元表」の設計情報欄の「２次」へ記入する。

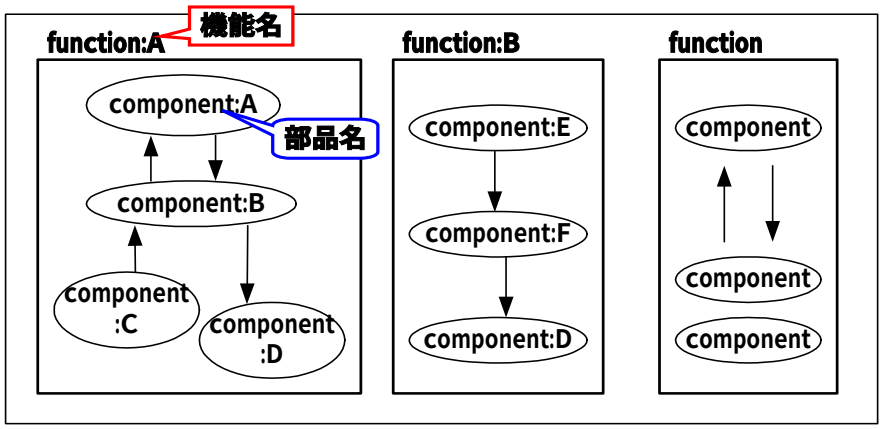

図 3-11「DFD」のイメージ

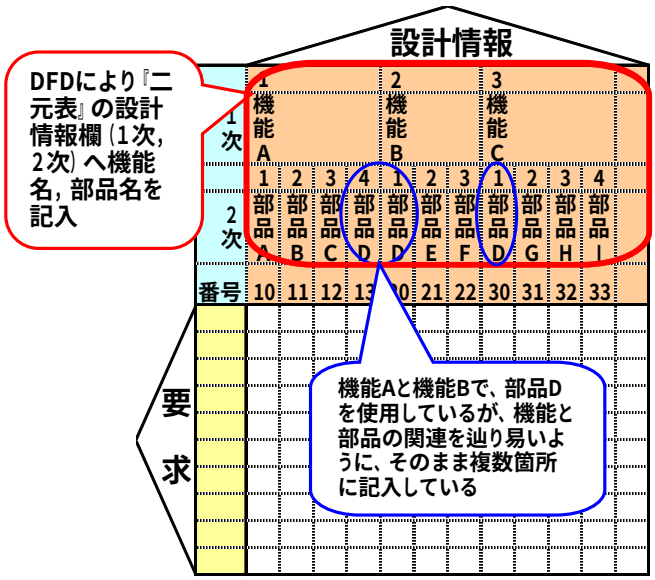

図 3-12「二元表」の設計情報欄への記入例

③－１　設計情報欄の３次：関数名の記入例

表 3-3 に示す「関数一覧」を入力情報とし、②で抽出した部品に含まれる関数名を、「二元表」の設計情報欄の「関数名」へ記入する。

③－２　設計情報欄の３次：関数種別の入力例（関数名セルの色づけ）

同様に、表 3-3 の関数名からデータ渡しのある関数を見つけ、関数種別とデータ渡しの有無により関数名のセルを色づけしておく。ここでデータ渡しのある関数とは、引数または戻り値を持つ関数（「void 関数名(void)」の型でない関数）とする。この色づけは、関数同士の関連を記入するときに利用する。（詳細は 3.2.3.2 を参照）

表 3-3「関数一覧」の例

部品名	関数名	処理名	関数種別
部品A	void func_A1(void)	部品Aメイン処理	グローバル
	unsigned int func_A2(void)	部品A変数獲得関数	グローバル
	void func_A3(void)		ティック
	void func_A4(void)		ティック
部品B	void func_B1(void)		ローバル
	signed int func_B2(signed int)	部品B変数設定関数	スタティック
	unsigned int func_B3(void)		ローバル
	‥	‥	
‥			

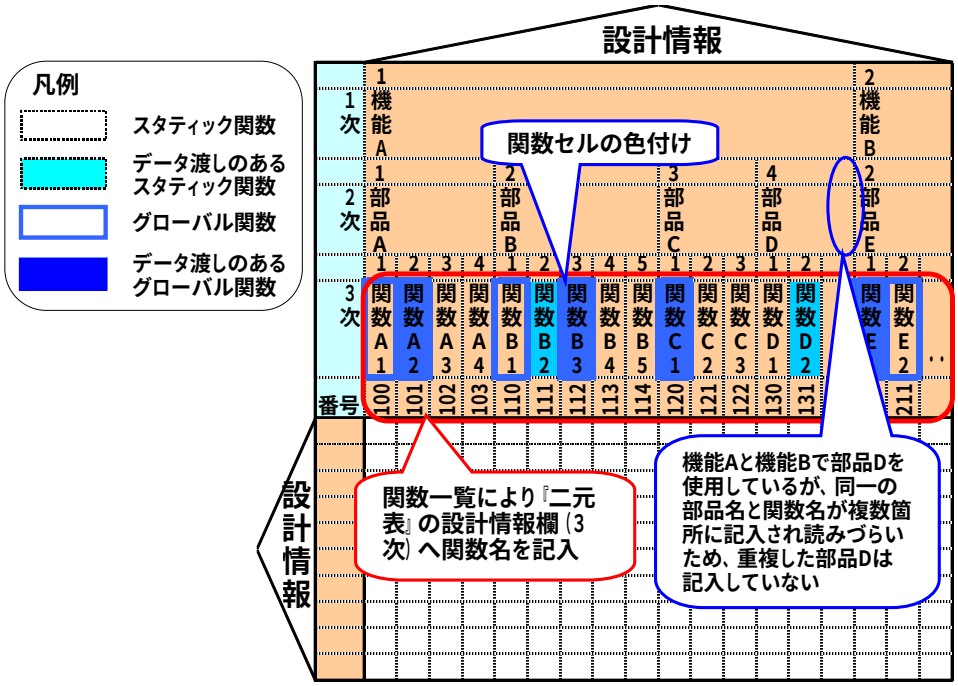

図 3-13「二元表」の設計情報欄への記入例

このように、「DFD」の機能名，部品名を「二元表」の設計情報欄（図 3-12 の赤枠）の１次と２次に、そして「関数一覧」の関数名を「二元表」の設計情報欄（図 3-13 の赤枠）の３次に記入することで、設計情報を階層化することができる。

3.2.3. 関連の割り付け

3.2.3.1. 要求と設計情報の関連を入力

顧客要求の変化に付随して変更が必要となる設計要素を見出し、「R-C」の二元表に関連記号を入力する方法を記す。但し、顧客要求と関連づける設計情報は 2 次の部品名とする。

構造化設計の例として、要求に対する使用部品を示す「使用部品一覧表」と DFD を、顧客要求と設計情報の 2 次（部品）との関連を特定するための入力情報とする。

DFD と使用部品一覧表により、ある要求で使用している部品を特定できるため、「二元表」の要求欄と設計情報欄の 2 次（部品）との交点セルに関連記号（○印）を割り付ける。重複した部品の間接的な使用の場合は、別の関連記号（△印）を割り付ける。例えば、要求 A-1 で機能 A にある部品 D を使用しているため交点セルには「○」を割り付けるが、機能 B や機能 C にある部品 D も間接的に使用されるため「△」を割り付ける。

表 3-4「使用部品一覧表」の例

No.	要求 / 部品名	要求A-1					要求B-1			要求B-2				
		A-1-1	A-1-2	A-1-3	A-1-4	A-1-5	B-1-1	B-1-2	B-1-3	B-2-1	B-2-2	B-2-3	B-2-4	
1	component:A	○	－	－	－	○	－	－	－	－	－	－	－	
2	component:B	－	○	－	－	－	－	－	－	－	－	－	－	
3	component:C	○	－	○	－	－	○	－	－	－	－	－	－	
4	component:D	－	○	○	○	○	－	－	○	○	○	－	－	
5	component:E	－	－	－	－	－		○	－	○	－	○	－	
6	component:F	－	－	－	－	－	○						○	
7	component:G	－	－	－	－	－	－	－					－	
8	component:H	－	－	－	－	－	－	－					－	
9	component:I	－	－	－	－	－	－	－					－	

異なる要求で同一の部品を使用している場合、『二元表』へ転記するときに△が割り付けられる

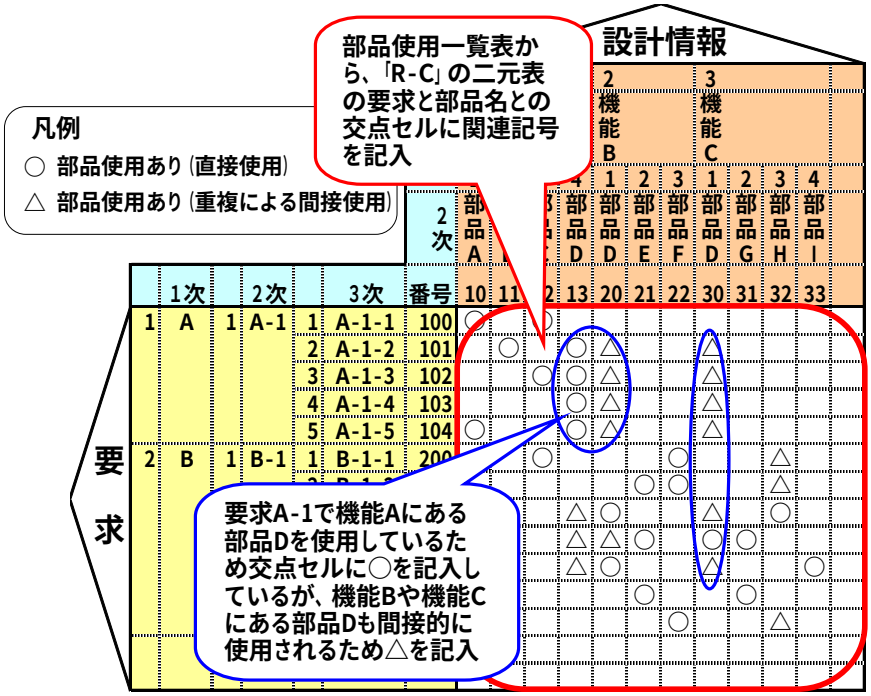

図 3-14「二元表」要求と設計情報の関連の記入例

「R-C」の二元表により、顧客要求の変化に付随して変更が必要となる設計要素（機能名，部品名）を特定することができる。これにより顧客要求からシステム内部の設計情報に対する変更影響の伝播を把握することができる。

3.2.3.2. 設計情報と設計情報の関連を入力

設計情報から関数と関数の関連を見出し、設計情報間の依存関係を表現した「C-C」の二元表に関連記号を入力する方法を記す。

構造化設計の例として、設計工程で作成した関数設計書に記載されている（或いはソースコードからツール等により逆生成した）「関数コールグラフ」を、設計情報の 3 次（関数）同士の関連を特定するための入力情報とする。

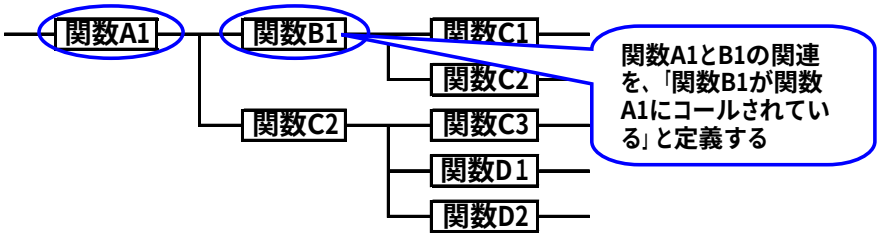

図 3-15「関数コールグラフ」の例

「関数コールグラフ」に示された関数と関数の関係により、二元表の設計情報欄の関数名と関数名との交点セル（横軸にあるコールされる関数名と縦軸にあるコールする関数名との交点セル）が特定されるため、その交点セルに表 3-5 に従った依存記号を割り付ける。

表 3-5　関数の種類と二元表での表現方法例

関数の種類	関数名セルの状態	依存記号
グローバル関数（データ渡しなし）	セルが青色で囲まれている	○
データ渡しのあるグローバル関数	セルが青色で塗られている	◎
スタティック関数（データ渡しなし）	セルが色づけされていない	△
データ渡しのあるスタティック関数	セルが水色で塗られている	▲

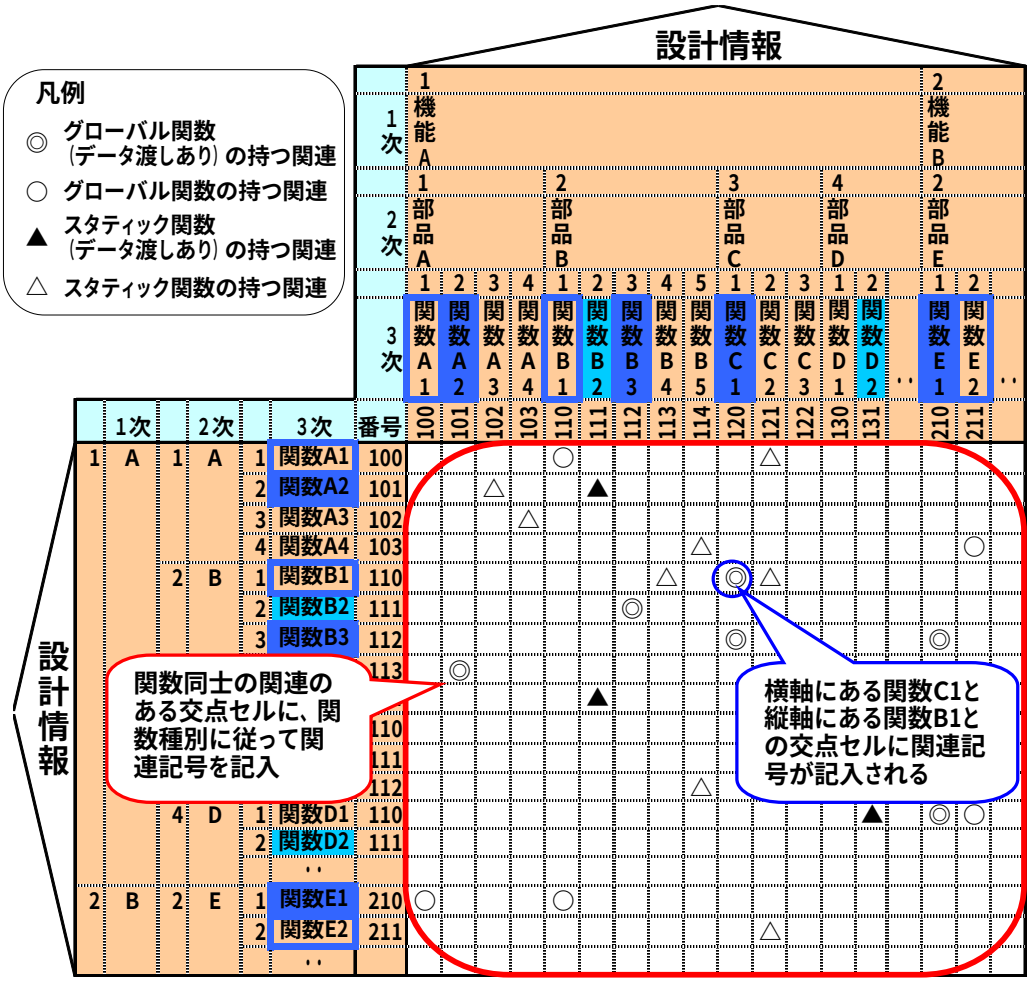

図 3-16「二元表」設計情報と設計情報の関連の記入例

「C-C」の二元表により関数同士の関連を特定することができる。これにより、システム内部の設計要素同士の依存関係を再帰的に辿ることができ、システム内部の影響の伝播を把握することができる。

4. 二元表の活用方法

4.1. 二元表によるテスト効率化の適用範囲

本手法は、顧客要求や設計情報の関連を辿ることで影響範囲を特定しているため、関連情報が得られない場合、または不完全な場合は影響範囲の特定が困難となる。このような場合、何らかの手段による情報の補完が必要となるが、補完により変更影響の特定精度の低下や可視化のための工数の増大が発生する可能性がある。したがって本手法には二元表に含まれる関連情報による適用限界が存在する。本手法の適用範囲を表 4-1 に示す。

表 4-1 二元表によるテスト効率化の適用範囲

テスト工程 / 二元表に含まれる情報	システムテスト	結合テスト	単体テスト
要求仕様，外部設計，内部設計の関連	○	○	○
要求仕様，外部設計の関連	○	△	×
外部設計同士の関連	△	○	×
外部設計と内部設計の関連	×	△	△
内部設計同士の関連	×	△	○

○=効率化可能

△=条件付で効率化可能（補足する資料を作成すれば可）

×=効率化が非常に困難

4.2. 標準的な活用方法

既存のソフトウェアシステムをベースとした再利用型ソフトウェア開発において､1.2.1 に示すドキュメントが整備され保守されている場合の活用方法を解説する。

4.2.1. 二元表を用いた変更影響の可視化

3 章で示した手順により、変更前、および変更後で二元表を作成する。変更前と変更後の関係は､「前バージョンと現バージョン」,「基底バージョンと派生バージョン」など、再利用が行われる製品であれば特に限定しない。

二元表の行、および列の名称が一致しているセル間の比較を行う。セル内容の比較結果を変更分類（表 4-2）にしたがって表現し、変更を可視化する。セル間の比較にあたっては、変更前後の二元表の各軸にある要素名が同一となるように位置関係を揃えておくとよい。

表 4-2 変更分類

セルの変化（変更前→変更後）	変更分類	セルの可視化
空欄　→　何らかの依存記号	追加	赤表示
依存記号　→　異なる依存記号	変更	緑表示
依存記号　→　空欄	削除	青表示
変化なし	変更なし	（白表示）

追加，変更，削除として認識されたセルから縦軸（もしくは横軸）に依存関係にある要素を辿ることにより、変更影響を認識することができるようになる。

図 4-1 では、R9 の追加に対して C3 に依存関係があるとすると、R9 に直接対応関係のある C3 に 1 次の影響、C4，C6 に 2 次の影響、そして C7，C8 に 3 次の影響があることがわかる。

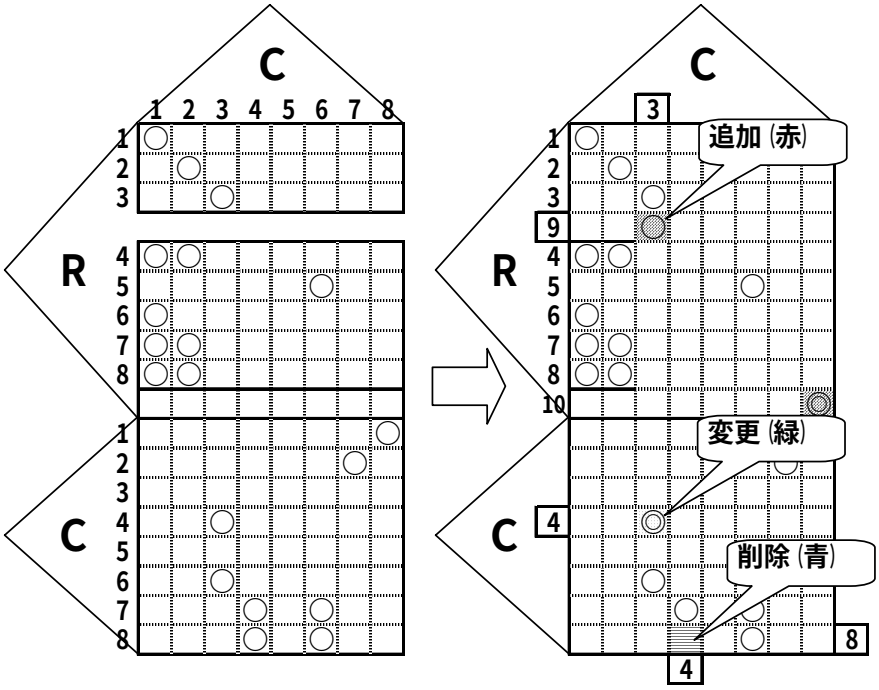

図 4-1 変更影響の可視化

4.2.2. 二元表を用いた変更影響の追跡

変更影響は、二元表の各軸で何らかの関係が存在している個所を順方向、または逆方向に辿っていくことで把握する。

展開された二元表も展開元，展開先間で同様に辿ることにより、異なる表記法であった文書間でも依存関係を機械的に辿ることが可能となるため、任意の着目点を起点とし、ツール等によって影響範囲を特定することが可能になる。

図 4-1 における「R9-C3」の対応関係からの変更影響の追跡を、図 4-2 に示す。

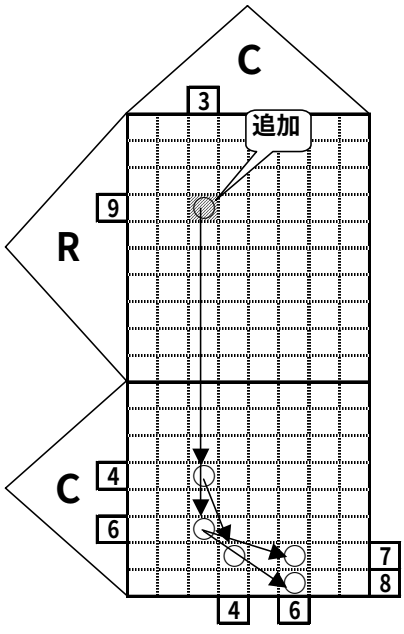

図 4-2 追加に対する変更影響の追跡

4.2.3. 変更影響によるテスト効率化

変更に対して以下の着眼点を考慮することで、テスト効率化を実施する。

4.2.3.1. テスト範囲重みづけの着眼点

変更影響に推移関係がない場合は 1 次のみを追跡すればよいが一般にはその影響は伝播していくことになる。

影響範囲の特定には二元表間および二元表内において依存関係を辿る回数を設定し、変更の種類に応じて順方向、および逆方向にその回数分辿ることにより影響範囲を特定する。辿る回数は二元表のもととなる顧客要求や設計情報の特性に応じて決定する。

また 1 次の影響は R-C，C-C の表から直感的に読み取ることが可能である。

4.2.3.2. 変更種別によるテスト方針決定時の着眼点

以下の変更分類に着目しテスト方針を決定する。

① 追加では、機能提供側の使われ方が変化すると考えられる。したがって、依存関係を順方向に走査して、対象となる機能提供側が追加された機能利用側の要求する機能を提供できるかを検証するテスト項目を追加する。
追加の影響範囲にあるテスト項目は、基本的にそれぞれの製品ドメインで実施していた「新規」に相当するテスト方針が適用できる。

② 変更では、機能提供側，機能利用側の双方が変化すると考えられる。したがって、依存関係を順方向、および逆方向に走査して、それぞれの変更点に対するテスト項目を追加する。
変更の影響範囲にあるテスト項目は、「改造・修正」に相当するテスト方針が適用できる。

③ 削除では、全ての機能利用側で利用する必要がなくなったと考えられる。依存関係を逆方向に走査して、削除された機能の呼び出しが不要になったことを検証する、もしくは、代替の手段が提供されているかを検証するテスト項目を追加する。
不要や代替手段の検証は、実際にはコンポーネントの無効化設定やパッケージ移動の確認といったテスト項目となる。

④ 未変更部分では、回帰テストとして用意されたテストセットを実施する。

4.3. 実践的な活用方法

実プロジェクトに適用する際に必要となる活用方法の詳細について、より詳細なデータがある場合、もしくはデータが欠落している場合の変更影響範囲を特定する手段を述べる。

4.3.1. より詳細な設計情報がある場合のテスト範囲決定方法

4.3.1.1. n 階層化された二元表の展開

本手法では顧客要求や CD，DFD，関数コールグラフなど依存関係を表現するための表記法が変わる（多くは別の文書となる）と、その表記方法「間」の依存関係を表す二元表（例えば R-C）と表記方法「内」の依存関係（例えば C-C）を表す二元表を 2 つ組にして用いる。

したがって、図 4-3 に示す二元表の構成に一般化することにより様々な開発手法で利用される複数の依存関係を表す表記方法に対して適用することが可能になる。

例えば、オブジェクト指向設計で、クラスの粒度ではなくより詳細なメソッド呼び出しの粒度などまで得られている場合は二元表の組を増やすことに厳密な変更影響を特定することが可能になる。

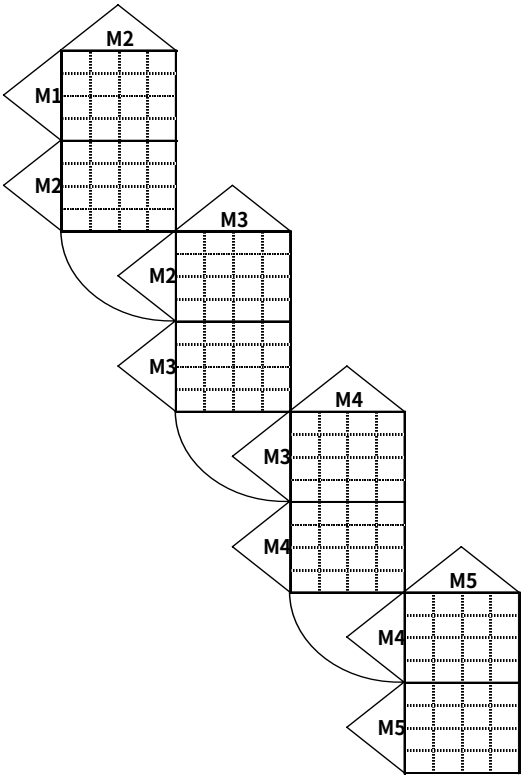

図 4-3 階層化された二元表

4.3.1.2. 関係の重みづけにもとづくテスト範囲の特定

4.2.3.1 節では依存関係を辿る回数により影響範囲を特定したが、オブジェクト指向設計，構造化設計で示した手順に見られる関係の重みづけを利用することにより厳密な影響範囲の特定が可能となる。すなわち関係の重みづけにより影響を数値化し、あるしきい値までを影響範囲とすることで、より定量的な影響範囲の特定が可能となる。例えば「◎＝0.5，○＝0.3，△＝0.1」といった対応関係でしきい値が 0.1，影響の伝播が「◎→○→△→△…」となっている場合は 0.5×0.3×0.1＝0.015＜0.1 となるため、「◎→○」を影響範囲と特定することができる。図 4-4 に例を示す。

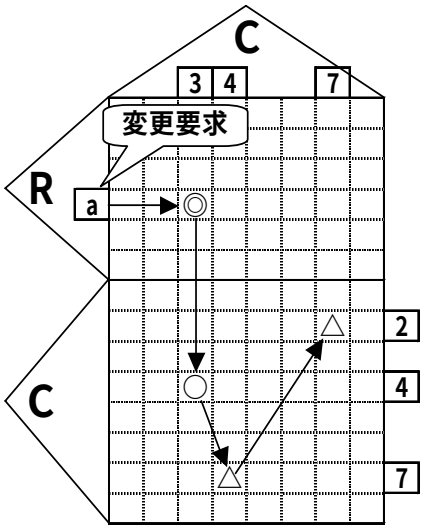

図 4-4 影響範囲の特定方法

4.3.1.3. 優先度にもとづくテスト優先順位の決定

4.3.1.2 節で、一つの変更に対する各交点となるセルの影響度を定量化することを示した。全ての変更に対して影響度の総和を求めることにより、二元表の対応関係毎や各軸の項目毎の変更影響を把握することが可能となる。これによりテスト対象の優先度づけが行えるようになる。

例えば、二元表が関数-関数であり S1 網羅を 100%とするテスト工数がない場合、変更影響度の高い関数からコールペアの網羅となるようにテストケースを設計することにより、実質的な網羅度を上げることになる。

4.3.1.4. C 言語で関数まで詳細化されている場合の活用例

標準のフルセットシステムから必要な機能をカスタマイズしていくような場合では、以下のような活用方法が考えられる。

① カスタマイズすべき要求から影響を受ける機能が「R-C」の二元表により特定でき、順次、部品名，関数名まで特定することができる。
② 特定された関数と関連を持つ関数が「C-C」の二元表により特定できる。
③ 特定された関数から逆追いで影響を受ける部品，機能が特定される。
④ 要求Ａがカスタマイズされるとした場合、要求Ａやその他の関連する要求が特定されるため、それらの要求に対する評価や、関連する部品に対する評価を実施する。
⑤ 関数の種別（グローバル／スタティック，データ渡しの有／無）や関連の深さ等により、評価の優先順位を付けて実施する。

4.3.2. 設計情報が欠落している場合のテスト範囲特定方法

既存のソフトウェアシステムをベースとした再利用型ソフトウェア開発では、既存のドキュメントの保守が十分に行われていないケースも多い。特に小規模のソフトウェア開発では、コストや期間の関係で、ドキュメントの保守は軽視される傾向にある。また、開発担当者が少人数で、かつ人材が流動的でない場合もドキュメントの保守は軽視される傾向にある。

このようなケースでは、本手順書が前提としている設計情報が欠落している場合がある。設計情報が欠落している場合、欠落した情報を対象にテストを行っている工程では、テスト項目を効率的、かつ安全に絞り込むことができず、安全を優先するならばできる限り多くのテスト項目を行い、コストや日程を優先するならばリスクが増加することを許容してテスト項目の削減を行わなくてはならない。そこで、本項では設計情報が欠落しているケースでのテスト範囲特定方法について解説する。

設計情報が欠落している場合の対処方法は、基本的に変更部分とその影響を受ける部分を特定し、その変更および影響を受ける部分についての上位（または下位）の設計情報のみを得ることで、最小の工数でテスト範囲を特定するということである。そこで本項では「コンポーネント同士の関連のみの関連情報が含まれている時にシステムテストを実施する場合」を例に解説する。

本ケースでは、コンポーネント同士の関連を示す設計情報があるが、その上位に位置する顧客要求とコンポーネントの関連を表す情報がない場合に用いることを想定している。

コンポーネント同士の関連情報のみが明らかな場合、二元表を作成すると図 4-5 のようになる。

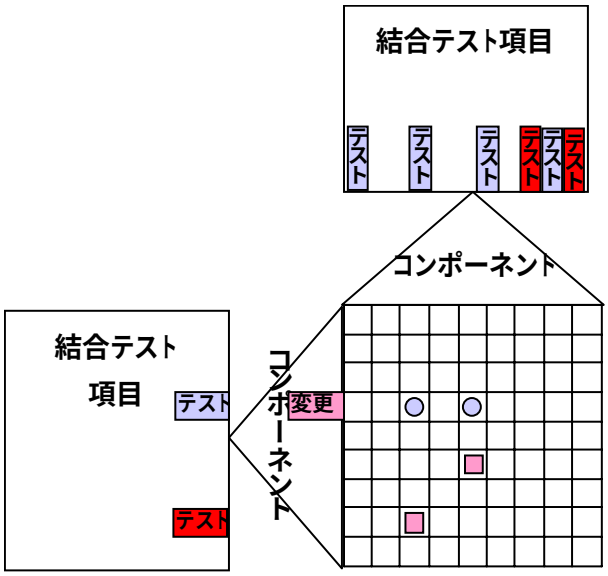

図 4-5 情報がコンポーネント同士の関連情報のみのケースでの二元表

本ケースの場合を例に、必要最小限の機能要件とコンポーネントの情報を得るだけで、機能要件への変更影響範囲を特定する方法を説明する。

まず、図 4-5 のコンポーネント同士の二元表を作成し、変更されたコンポーネントとそれに

影響を受けるコンポーネントを可視化する。これを結合テスト項目と関連づけることによって、結合テストレベルでのテストの効率化を行うことができる。

次に、変更されたコンポーネントがどの機能要件の変更によって変更に至ったのかを調査し、さらに、変更影響を受けるモジュールが関係する機能要件の絞込みを行う。この変更影響を受けるモジュールが実現している機能と、その機能に対して関連がある機能を完全に特定できた場合は、その特定された機能要件のテスト項目をテスト対象とすればよい。しかし、十分な追跡が行えず、変更影響を受ける機能のみを特定でき、その機能に関連する機能が特定できなかった場合には、機能ツリーを使用してテスト範囲を決定する。

4.3.2.1. 機能ツリー

機能ツリーとは、富士通株式会社で使用されているテスト範囲の決定手法であり、機能同士の関連をツリー上に表現したものである。機能が変更された時に、その変更された機能が機能ツリーの何処に位置するかを確認し、テストすべき機能を決定するために用いられている。

本項の機能ツリーの解説は 2002/06/28 の ソフトウェア開発環境展 専門セミナー 「テスト技術と品質保証」セッションにて、「第三者テスト手法と実際」というテーマで、富士通株式会社 有村 雄二氏が講演した内容の一部を引用している。

4.3.2.2. 機能ツリーとは

図 4-6 に示すように、機能を大きいものから小さいものへツリー上に分割したものである。機能ツリーは外部仕様（機能仕様）を元に作成する。

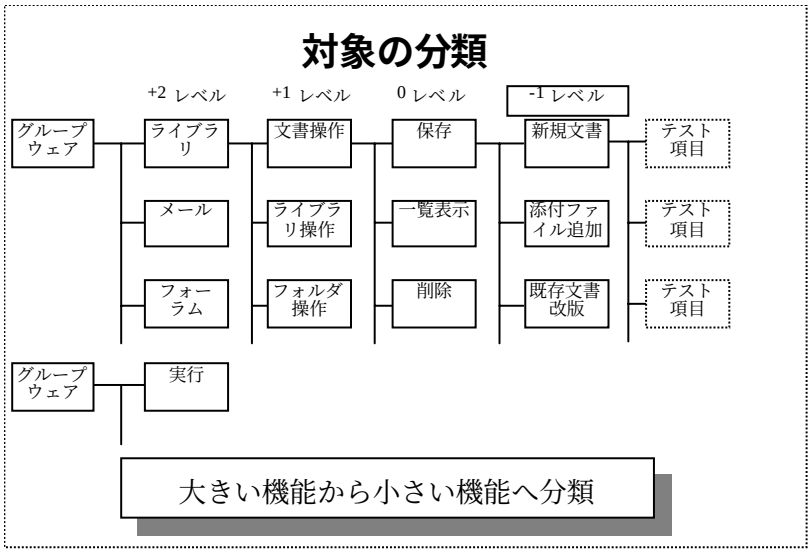

図 4-6 機能ツリー（対象の分類）（出典:第 11 回 SODEC 専門セミナー 「第三者テスト手法と実際」）

機能ツリーを用いたテスト設計では、機能ツリーで分解できないレベルとなった機能に対して、テスト項目を設定していく。テスト項目を作成する場合、図 4-7 に示す「因子抽出の観点」にもとづき、テスト項目を設計する。

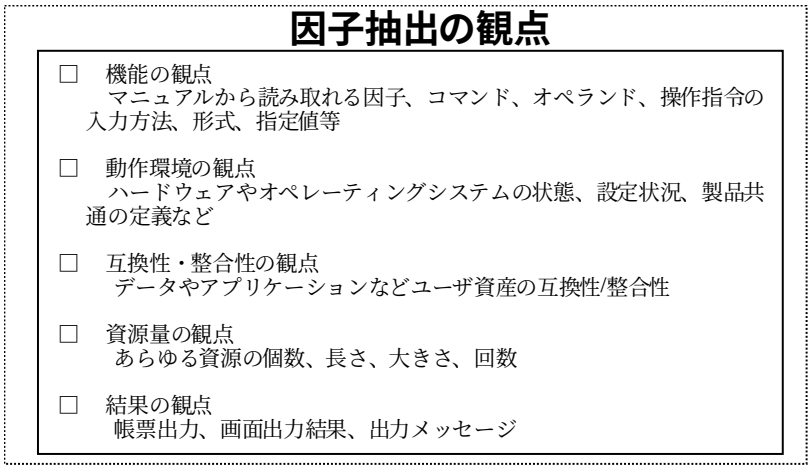

図 4-7 機能ツリー（因子抽出の観点）（出典:第 11 回 SODEC 専門セミナー 「第三者テスト手法と実際」）

4.3.2.3. 機能ツリーを使用した影響範囲の特定方法

富士通では、デグレード（degrade）防止テストのテスト範囲決定するのために機能ツリーを使用している。機能ツリーの使用方法としては、以下の 2 通りがある。

- 機能使用側トリガ
  - 製品自身のアップグレード，保守
- 機能提供側トリガ
  - 下位レイヤ（ハード，OS，JDK など）のエンハンス，保守
  - 連携ソフト（ブラウザ，データベースなど）のエンハンス，保守
  - 環境の定義，設定値の変更

4.3.2.4. 機能使用側トリガでの機能ツリー使用方法

製品自身の機能に対して変更が行われた場合、機能ツリー上の該当機能を探し出し、該当機能の一段階上の機能レベルに上がり、その機能の配下となっている全ての機能に対して再テスト（回帰テスト）を行う。

例えば、図 4-8 機能使用側トリガで示すように、+2 レベルが A、+1 レベルが 01、0 レベルが 01、-1 レベルが 01、-2 レベルが 01 の機能で変更が発生した場合、+2 レベルが A、+1 レベルが 01、0 レベルが 01、-1 レベルが 01 の配下にある機能全てに対して再テスト（リグレッションテスト）を行う。

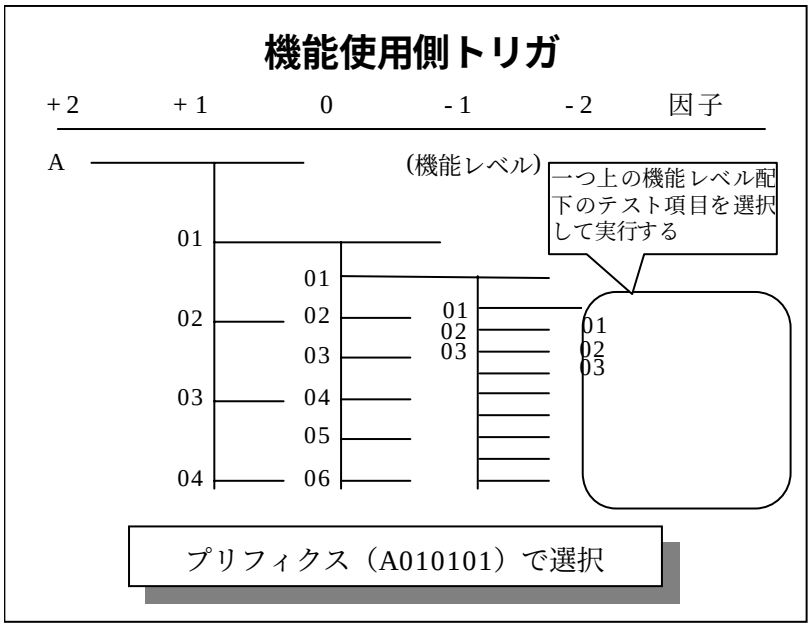

図 4-8 機能ツリー（機能仕様側トリガ）（出典:第 11 回 SODEC 専門セミナー「第三者テスト手法と実際」）

4.3.2.5. 機能提供側トリガでの機能ツリー使用方法

製品が使用される環境や、間接的に使用している OS やライブラリなどに変更が発生した場合、機能提供側トリガとして扱う。この場合の機能ツリーの使用方法は、図ｘｘに示すように、機能ツリー上でトリガとなった因子と共通の因子を持つ機能を洗い出し、再テスト（回帰テスト）を行う。例えば、ファイルを呼び出すライブラリに変更があった場合、ファイル呼び出し機能を持った全ての機能に対して再テストを行う。

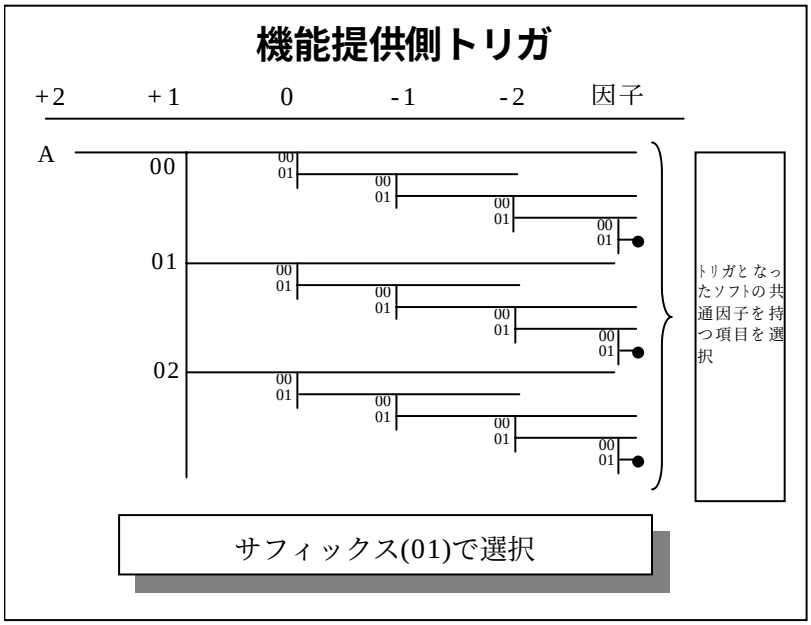

図 4-9 機能ツリー（機能提供側トリガ）（出典:第 11 回 SODEC 専門セミナー「第三者テスト手法と実際」）

5. まとめ

このように、顧客要求に変更が発生した場合に、要求仕様やソフトウェアの設計情報を用いて二元表を作成し、その関連を視覚的に表すことで、開発方法論に依存しない手法でソフトウェアへの変更影響を可視化することができる。また、変更影響の可視化結果から、顧客要求の変更によるソフトウェアへの影響範囲を直接的または間接的に特定し、テスト項目を効率良く抽出することができる。

本手順書のように顧客要求の変更がソフトウェアに与える影響を可視化することができれば、経験の浅いテスト担当者や設計情報を持たないテスト担当者でも、従来のような経験や勘に依存しない方法でテスト効率を向上させることができる。

6. 出典／参考文献

(1) 21 世紀へのソフトウェア品質保証技術：
菅野 文友, 吉澤 正(監修)：日科技連出版社：1994/09/21
(2) 実践的 QFD の活用―新しい価値の創造―：
新藤 久和 (編集), 赤尾 洋二, 吉沢 正 (著)：日科技連出版社：1998/06/30
(3) 品質展開入門：赤尾 洋二 (著)：日科技連出版社：1990/11
(4) UML モデリングのエッセンス―標準オブジェクトモデリング言語入門：
Martin Fowler (著), Kendall Scott (著), 羽生田 栄一 (翻訳)：翔泳社：2000/04/15
(5) ユースケース実践ガイド―効果的なユースケースの書き方：
Alisteir Cockburn (著), ウルシステムズ（株） (翻訳)：翔泳社：2001/11/20
(6) はじめて学ぶ UML：竹政 昭利 (著)：ナツメ社：2002/12/27
(7) 図解でわかるソフトウェア開発のすべて：
Mint（経営情報研究会）（著）：日本実業出版社：2000/07
(8) Unified Modeling Language Specification：http://www.omg.org/
(9) SESSAME－組込みソフトウェア管理者・技術者向け用語集－：http://www.sessame.jp/
(10) 品質機能展開による高品質ソフトウェアの開発手法（解説編）：
情報処理進行事業協会技術センター（編）：コンピュータエージ社：1989/03
(11) 品質機能展開による高品質ソフトウェアの開発手法（活用事例編）：
情報処理進行事業協会技術センター（編）：コンピュータエージ社：1989/03
(12) 2002 年ソフトウェア開発環境展専門セミナー「第三者テスト手法と実際」：
富士通株式会社 有村 雄二氏：2002/06/28
(13) 属性つきゴール指向要求分析法：電子情報通信学会技術研究報告 Vol. 101, Mar. 2002：
海谷治彦, 佐伯元司, 海尻賢ニ
(14) ”An Application of QFD to System Integration Test ”：
品質, Vol.30,No.1,pp.113-125 received on 2000-10-28：Takami Kihara, Charles E. Hutchinson, Dan Dimancescu, Hisakazu Shindo and Tadashi Yoshizawa
(15) ”A new approach to software design with QFD-like tabulations of UML diagrams”：
10th International QFD Symposium 2004：Yoshimichi Watanabe, Yunarso Anang, Masanobu Yoshikawa, Yujiro Kasai and Hisakazu Shindo