一般
投稿論文

設計項目間の不整合を検出するためのシステム開発と評価

元山 厚[†1]　中谷 多哉子[†2]

[†1]（株）日立製作所　[†2]筑波大学大学院ビジネス科学研究科

ソフトウェアの品質を向上させるために開発の各工程で，欠陥を検出し修正する必要がある．我々の調査によると設計工程で見落とされていた設計書の欠陥のうち，テスト工程まで発見できなかったものは，設計書の設計項目間の不整合が多い傾向があった．このような設計書の欠陥の見落としを防止するための仕組みがあれば，設計レビューを通して設計項目間の不整合の検出が可能である．本論文では，設計項目間の不整合を自動的に検出する支援システムを提案する．支援システムは，設計書のテンプレートと整合チェックツールで構成する．設計書のテンプレートは，設計書に記述すべき事項と構造を定義し，設計項目間の対応関係を明確にしたものである．整合チェックツールは，設計項目間の内容を自動的に比較し，不整合を検出する．提案した支援システムをソフトウェア開発プロジェクトでの設計レビューに適用した．その結果，設計項目間の不整合の見落としを防止することができた．

1. はじめに

ソフトウェアの品質を向上させるために開発の各工程で，欠陥を発見し修正する必要がある[2]．欠陥の原因はさまざまであり，それらを発見する方法が提案されている．本論文では，設計工程に着目し，設計工程で設計上の欠陥を発見する方法を提案する．設計工程において，設計の内容を示す文書である設計書の品質を高めることが重要である．なぜならば，設計工程はソフトウェアプロセス全体に影響する工程となっているためである．設計書の品質が高くなければ，テスト工程やリリース後に欠陥が多発してしまうからである．

設計レビューは，設計書の欠陥を検出する効果的な方法の1つである[20]．設計レビューは，まだプログラムが動作していない段階でソフトウェアの潜在的な欠陥を発見する静的解析の1つであり，テストに先立って実施される．プログラミングやテストよりも早い段階で，設計書の誤りや矛盾の検出を目的としている．設計工程でのV&V活動[3],[15]の検証の手段の1つであり，設計書の品質向上に必須のものである．V&VはVerification & Validationの略であり，検証と妥当性確認を意味している．

設計工程で設計書の欠陥を，可能な限り検出する必要がある．検出されなかった設計書の欠陥は，設計工程の次工程以降で何倍にも増幅される[9]．設計レビューを実施するためには，開発者は時間と労力を，開発組織は費用をかけなければならない．設計工程でコストをかけなければ，あとで多くのコストがかかることとなる．しかし，設計書のレビューは容易ではない．設計レビューを経ているにもかかわらず，設計書の欠陥がテスト工程で発見されることが少なくない[5]．

テスト工程で発見される設計工程の欠陥にはさまざまな種類がある．本研究に先立ち我々は，テスト工程まで発見できなかった設計書の欠陥にどのようなものがあるかを調査した．その結果，画面設計書やデータベース設計書などの複数の設計書に記述された項目（以下，設計項目とする）間の不一致といった，設計項目間の不整合が多いことを発見した[18]．この不整合は，開発対象のソフトウェア規模の大小にかかわらず発生し，スケジュール遅延や品質の低下の原因となっていた．基本設計書や詳細設計書（以下，これらの文書を本論文では単に設計書という用語を用いる）のレビューにおいて，技術者達が，手動で設計書の設計項目の整合を1つ1つ検証していた．手動での設計項目の整合の検証では，不整合を検出しきれていないことを示している．

設計項目間の対応関係に基づいて，設計項目間の不整合を正確に検出する仕組みが必要である．設計レビューで設計項目間の不整合を検出するためには，レビューアが複数の設計書を比較し，対応する項目の内容に矛盾がないことを1つ1つ確かめる必要がある[14],[19]．このとき，レビューアはレビュー対象となる複数の設計書を選択し，関係する設計項目を洗い出さなくてはならない．

そこで本論文では，設計項目間の不整合を自動的に検出する支援システムを提案する．支援システムは，設計

書の設計項目間の対応関係を定義した設計書のテンプレートと，設計項目間の内容を自動的に比較するツールで構成する．まず設計者が，設計書のテンプレートに基づいて設計書を作成する．次に，レビューアが設計レビュー時に, ツールを適用して設計項目間の対応関係を辿り，設計項目間の不整合の検出を行う．支援システムを適用した設計レビューにより，不整合の検出の正確性を向上できると考える．

ただし，支援システムの適用は，提案する設計書のテンプレートが適用可能なシステム開発に限定される．設計書のテンプレートを使用して記述された設計書だけが，提案するツールにより不整合が検出可能となるからである．

ここで，設計項目間の不整合とは，以下のように定義する．

- 名称，桁数，型および属性が一致していない．
- 設計項目間で対応関係を持つべき一方の要素が存在しない．
- 大小関係や順序が同等でない．

内容が正しく詳細化されているかは，不整合の検出の対象としない．なぜならば，レビューアの判断が必要であり，自動化が困難であるからである．

本論文では，提案する支援システムの有効性を評価する．この評価には，複数のソフトウェア開発プロジェクトでの設計レビューに支援システムを適用した．

本論文の構成は以下の通りである．第2章では，関連研究を紹介する．第3章では，支援システムを構成する設計書のテンプレートと，それに基づき設計項目間の不整合を検出するツールの説明を行う．第4章では，適用実験の方法とその結果を示し評価する．支援システムにより，ソフトウェア開発プロジェクトで設計工程での設計項目間の不整合を検出することができ，設計書のレビューで見逃した設計項目間の不整合の検出に有効であることを示す．

2. 関連研究

設計項目の対応関係を管理する手法は，これまでに多くの研究がある．Ramesh ら[8]や Letelier ら[4]は，要求仕様書の構成要素の間で, 依存関係を構成する要素の型, およびこれらの要素間に存在する関係の型をトレーサビリティ情報として表現するためのモデルを，参照モデルとして使用することを提案している．大橋ら[17]は，業務アプリケーション開発における要求定義工程と基本設計工程を対象に，モデルを利用したトレーサビリティの構築と，それを利用した完全性検証を行うための取り組みを示した．これらの研究は，設計項目の対応関係を管理するためのモデルを定義したものである．我々は，設計項目間の不整合を検出する支援システムを提案するにあたり，設計項目の対応関係の定義に，設計書のテンプレートを適用する．

設計レビューでは，系統立てた方法を適用するのが効果的である[16],[20]．レビューで使用される設計書を調べる方法には，さまざまなものが提案されており，その有効性が研究されている[12]．たとえば，チェックリストに記載された質問を順番に参照する方法，発見済みの欠陥の特徴に注目して，それと同様の欠陥を検出する方法[7]，ユースケース記述など利用者とシステムとの対話手順に基づいてレビューを行う方法[13]，アクターの観点（利用者，設計者など）や品質特性の観点でレビューする方法[10]がある．また，Travassos ら[14]は，オブジェクト設計での要求定義書と設計書の関連，および異なる種類の設計書の関連に基づくレビュー方法（Traceability-Based Reading（TBR））を提案している．しかし，これらの設計レビューの方法には，設計書の設計項目間の不整合を検出する方法は示されていない. 本研究では, 設計項目間の整合を検証する方法を明らかにし，その支援システムで自動化を実現する．

設計項目間の対応関係を管理するためのツールはいくつか存在する．DOORS[1]，TOOR[6]や RequisitePro[11]が，有名なツールである．これらのツールを用いることで，ある設計項目の内容の変更が発生した場合，トレーサビリティ情報に基づき影響範囲を検出することができるが, 設計項目間の不整合を検出するものではない. 我々は，設計項目間の不整合の見逃しを防止する支援システムを提案する．

3. 設計項目間の不整合の検出のための支援システム

この章では，支援システムを構成する要素を示す．

3.1 支援システムの構成要素

図1に本論文で提案する支援システムの構成要素を示す．図1では，実線の四角形で支援システムを構成する要素，実線矢印で入出力，点線矢印で参照，および白抜きの矢印で支援システムの適用プロセスが示されている．支援システムは，設計書のテンプレートと整合チェ

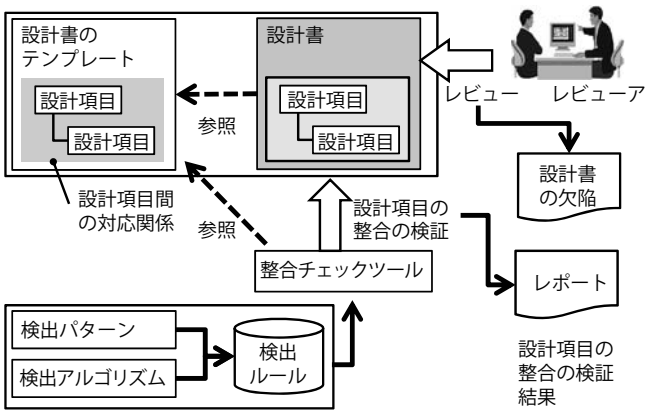

図 1　設計項目間の不整合検出のための支援システムの構成要素

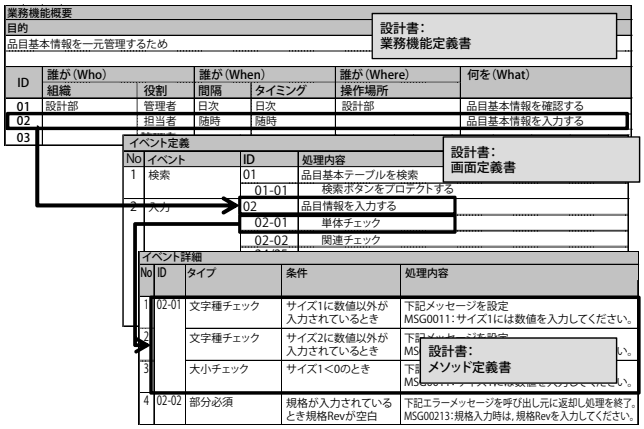

図 2　設計書の設計項目間の対応関係の例

ックツールで構成する．以下に，設計書のテンプレートと整合チェックツールの詳細を説明する．

3.1.1　設計書のテンプレート

設計書のテンプレートは，整合チェックツールにより，あらかじめ定義された設計項目間の対応関係を辿り，自動的に設計項目間の不整合を検出することを目的にしている．設計書のテンプレートは，設計書に記述すべき事項と構造を定義し，設計項目間の対応関係を定義している．各々の設計項目に識別子が付加されており，設計内容が一致すべき設計項目に対して，同じ識別子が設定されている．整合チェックツールが，同じ識別子を持った設計項目間を比較し，不整合を検出する．設計書のテンプレートには，基本設計と詳細設計で使用する各々の設計書向けのテンプレートがある．

設計書の設計項目間の対応関係の例を**図2**に示す．図2では，基本設計書である業務機能定義書の「業務機能概要」が記述してある設計項目を，基本設計書の画面定義書の設計項目「イベント定義」に，対応付けている．さらに前述の設計項目を，詳細設計書のメソッド定義書の設計項目「イベント詳細」に対応付けている．

3.1.2　整合チェックツール

設計項目間の不整合を検出する精度を向上するために，不整合を整合チェックツールを用いて検出する．設計項目間の整合を手動で検証する設計レビューでは，整合の検証の精度を高める必要がある．なぜならば，レビュー対象を漏れなく検証できるかは，レビューアの能力に依存するからである．さらに，レビュー時間の制約があり，レビュー対象物すべてに対しては，検証を実施しきれないこともある．

整合チェックツールは，設計書の設計項目間の不整合を検出する対象とする．整合チェックツールが，設計書のテンプレートに定義されている設計項目の対応関係を辿り，該当する設計書の設計項目の内容が記述されているセルの内容から抽出する．設計項目の内容を比較し，整合を検証する．不整合の検出結果をレポートとして出力する．整合チェックツールは，検出ルールに基づき，不整合を検出する．

表 1　検出パターンの説明

項	検出パターン	説明
1	個別間型	1 対 1 の個別の設計書間において，設計項目間の整合を一方向に検証し不整合を検出する．
2	一覧間型	1 対 1 の設計書一覧の間において，設計項目間の整合を一方向に検証し不整合を検出する．
3	複数個別間型	複数の個別の設計書間において，設計項目間の整合を双方向に検証し不整合を検出する．
4	双方向型	1 対 1 の個別の設計書間において，設計項目間の整合を双方向に検証し不整合を検出する．
5	単数一覧・個別間型	単数の設計書一覧の設計項目に対する，個別の設計書の設計項目との不整合を検出する．
6	複数一覧・個別間型	複数の設計書一覧に対していずれかの設計書の設計項目に対する，個別の設計書の設計項目との不整合を検出する．

設計項目間の不整合の検出パターンは，**表1**に示す6種類がある．設計書は，設計情報の内容を記述した個別の設計書と，複数の設計情報を目次として一覧化した設計書一覧に分類できる．設計書の比較は，1つの設計書と他の1つの設計書を1対1に比較する場合と，複数の設計書間で比較する場合がある．設計書の設計項目間の比較は，ある設計項目の内容が他の設計項目の内容に対して一方向で実施する場合と，双方向で実施する場合がある．以上の分類の組合せによりに，設計項目間の不整合の検出パターンは分類される．表1に検出パターンの説明を示し，検出パターンの例を**図3**に示す．

本論文で提案する整合チェックツールが不整合を検出するアルゴリズムは5種類がある．**表2**に検出アルゴリズムの説明を示し，**表3**に検出ルールの定義形式を示す．

表3の「ルールの項目」の列に示した，項番1から15

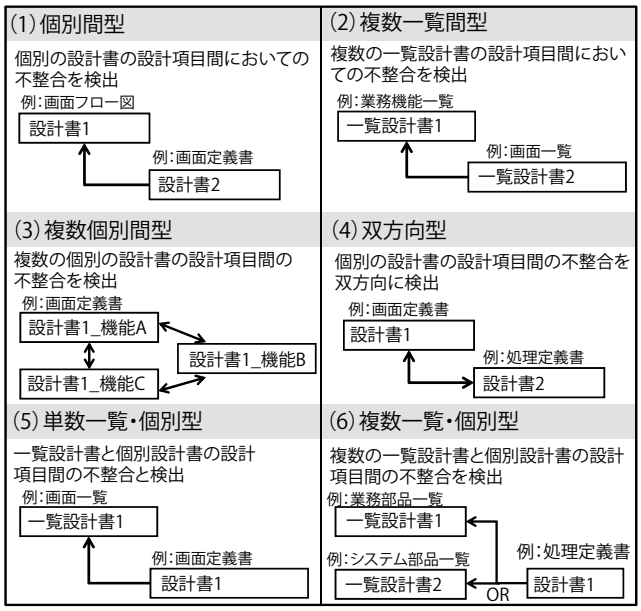

図 3　検出パターンの例

までに示した15項目を1つの定義体とし，検証対象ごとに設定する．定義体は，第1から7項目の検証対象の比較元の設計書に関する情報，第8から13項目の検証対象の比較先の設計書に関する情報，第14から15項目の検証ルールに関する情報で構成される．各項目の設定における必須または任意の条件を，「必須／任意」列に示す．

- 第1項目：検証を実施する単位の識別子を指定する．
- 第2項目：検証を実施する比較元の設計書の名称を設定する．
- 第3項目：共通で設計する設計書か機能別に設計する設計書かを示す区分（共通／機能別）を設定する．
- 第4項目：検証を実施する比較元の設計項目名を設定する．
- 第5項目：比較を実施する設計項目の内容が記述されている範囲の，起点のセル位置から値を読み込む方向として，縦または横を設定する．
- 第6項目：読込みの範囲として，検証を実施する項目名が記述されている範囲の起点から終点までの項目数を設定する．
- 第7項目：設計項目の内容の読込みを終了する条件を指定（項番／空白／罫線）を設定する．
- 第8 ～ 13項目：検証を実施する比較先の情報を設定する．比較元の設計項目と比較する比較先の設計項目には，設計書のテンプレートにおいて，比較元の設計項目の識別子と同じ識別子が設定されている設計項目を設定する．
- 第14項目：検出パターンを設定する．
- 第15項目：検出アルゴリズムを設定する．

表 2　検出アルゴリズムの説明

項	検出アルゴリズム	説明
1	存在型	一方の設計項目の内容が，他方の設計項目の内容に存在するかを検証する．
2	一致型	一方の設計項目の内容が，他方の設計項目の内容に一致するかを検証する．
3	複合型	一方の設計項目の内容が，他方の設計項目に存在し，かつ一致するかを検証する．
4	大小型	設計項目間の内容での大小関係が同等であるかを検証する．
5	順序型	設計項目間の内容での記載の順序が同等であるかを検証する．

表 3　検出ルールの定義形式

項	設計書情報	分類	ルールの項目	必須
1	定義情報		検証識別子	○
2	比較元	設計書	設計書名	○
3			設計書区分	○
4		設計項目	設計項目名	○
5			読込方向	○
6			読込位置	×
7			読込終了条件	×
8	比較先	設計書	設計書名	○
9			設計書区分	○
10		設計項目	設計項目名	○
11			読込方向	○
12			読込位置	×
13			読込終了条件	×
14	検出情報		検証パターン	○
15			検出アルゴリズム	○

整合チェックツールの検出結果レポートの例を**図4**に示す．図4は，基本設計書である画面一覧と画面定義書の設計項目の整合の検証結果の例である．画面一覧の設計項目である画面名と，画面定義書の設計項目である画面名の整合を，検出アルゴリズムの「一致型」で検証したものである．「検証結果」列と「メッセージ」列に検証結果が示されている．

設計書のテンプレートは，マイクロソフト社のMS-Excelのファイル形式となっている．また，検出ルールはXML形式のファイルとなっている．支援システムの開発には，整合チェックツールに6人月，設計書のテンプレートに2人月の工数を要した．開発期間は3カ月であった．

4. 適用プロジェクトでの評価

4.1　評価方法

提案した支援システムによる設計項目間の不整合の検出における有効性を評価するために，複数のプロジェク

No	比較元の設計書					比較先の設計書					検証アルゴリズム	メッセージ
	設計書名	シート名	項目名	値	セル位置	設計書名	シート名	項目名	値	セル位置		
1	BA05-01#02_画面一覧.xls	画面一覧	画面ID, 画面名	-	-	BA05-02#01_画面定義書_一覧検索.xls	画面定義	画面ID, 画面名	BSK001, 一覧検索	E2, Q2	一致型	BA05-01#02_画面一覧.xlsに「BSK001」は存在しますが「一覧検索」ではありません.
2	BA05-01#02_画面一覧.xls	画面一覧	画面ID, 画面名	-	-	BA05-02#01_画面定義書_変更・削除.xls	画面定義	画面ID, 画面名	BSK0003, マスタ変更・削除	E2, Q2	存在型	BA05-01#02_画面一覧.xlsに「BSK0003, マスタ変更・削除」が存在しません.
3	BA05-01#02_画面一覧.xls	画面一覧	画面ID, 画面名	BSK002, マスタ新規登録	B5, C5	BA05-02#01_画面定義書_新規登録.xls	画面定義	画面ID, 画面名	BSK002, マスタ新規登録	E2, Q2	一致型	整合が確保されています.
	:	:	:	:	:	:	:	:	:	:	:	:

図 4　整合チェックツールの検出結果レポートの例

表 4　適用結果

項	プロジェクト名	対象業務	見積規模（KS）	適用した設計工程	適用時期	対象設計書数（件）	確認対象の設計項目数（件）	不整合の検出	検出密度（件／ KS）	検出密度（件／設計数）
1	A	商品管理	1,553	基本	完了時	3,205	13,407	909	0.59	0.28
2				詳細	完了時	660	2,327	200	0.13	0.30
3	B	保守サービス	259	詳細	レビュー前	143	2,663	238	0.92	1.66
4	C	販売管理	589	基本	完了時	40	1,048	88	0.15	2.20
5				詳細	レビュー前	338	9,678	1,937	3.29	5.73
6	D	受注管理	1,411	基本	完了時	296	24,516	1,764	1.25	5.96
7				詳細	完了時	495	115,276	27,684	19.62	55.93
8	E	生産管理	450	詳細	完了時	18	1,197	8	0.02	0.44
9	F	口座管理	82	詳細	完了時	47	2,713	153	1.87	3.26

トに適用した．設計項目間の不整合の検出の有効性は，不整合の検出が自動化できるか，設計レビューでの不整合の見落としを防止することができるか，不整合の検出により開発工数を減少させることができるかで評価することとした．設計項目間の不整合の検出の有効性の定義を以下に示す．

(1) 不整合の検出の自動化
設計工程で作成された設計書に対して，支援システムにより設計項目間の不整合を検出できれば，支援システムは不整合の検出に有効である．

(2) 不整合の見落としの防止
テスト工程で発見された欠陥において，設計項目間の不整合を原因としたものがなければ，設計項目間の不整合の見落としの防止に有効である．

(3) 不整合の検出の効率
設計からテスト工程で要した工数が計画工数より減少すれば，支援システムは不整合の検出の効率向上に，有効である．

上記（1）の有効性を評価するために，各プロジェクトで検出した設計項目間の不整合数とその内容を比較した．(2) の有効性を評価するために，設計レビューとテストで検出した欠陥の件数の予定と実績を比較した．この予定は，各プロジェクトで，過去のプロジェクト実績より設定したものである．(3) の有効性を評価するために，設計からテスト工程で要した実績工数を取得し，計画工数と比較した．

適用した6プロジェクトの概要を適用結果と合わせて，**表4**に示す．全適用プロジェクトは，Javaを開発言語とし，提案した設計書のテンプレートを用いて，設計書を作成した．全適用プロジェクトは，業務アプリケーションをドメインとした．表4での適用した設計工程の列には，整合チェックツールを各プロジェクトで適用した工程を示している．適用時期の列は，各設計工程での整合チェックツールを適用した時期を示している．完了時とは，設計工程の完了時に適用したことを示している．また，レビュー前とは，各設計者が設計レビューの前に，支援システムの適用により不整合を検出したことを示す．整合チェックツールを適用した，設計工程と適用時期における効果を比較した．有効性の定義に基づいた評価を実施するための該当プロジェクトを下記に示す．

(1) 不整合の検出の自動化における有効性の評価
- 基本設計工程と詳細設計工程における不整合の検出数の比較：A，C，Dプロジェクト
- 詳細設計工程の適用時期（レビュー前／完了時）における不整合の検出数の比較：B（レビュー前），C（レビュー前），E（完了時），F（完了時）．プロジ

表5　BおよびFプロジェクトにおけるテスト工程での欠陥検出の結果

項	プロジェクト名	工程	テスト対象規模（KS）	チェックリスト					欠陥				
				件数			密度（件／KS）		件数			密度（件／KS）	
				予定	実績	差	予定	実績	予定	実績	差	予定	実績
1	B	UT	84.6	8,458	9,273	815	100	110	846	709	−137	10.0	8.4
2		PT	94.8	1,991	2,224	233	21	23	199	216	17	2.1	2.3
3		計	94.8	10,449	11,497	1,048	110	121	1,045	925	−120	11.0	9.8
4	F	UT	66.1	6,610	6,780	315	100	103	198	140	−58	3.0	2.1
5		PT	70.0	291	291	0	4	4	167	28	−139	2.4	0.4
6		計	70.0	6,901	7,071	315	99	101	365	168	−197	5.2	2.4

ェクト基本設計書のレビュー前への整合チェックツールの適用は，プロジェクト都合により実現できなかった．

(2) 不整合の見落としの防止の有効性の検証として，テスト工程で検出した欠陥を比較：BとFプロジェクト

(3) 不整合の検出の効率の検証として，設計からテスト工程で要した工数を比較：BとFプロジェクト

4.2　適用結果

(1) 不整合の検出の自動化における有効性

適用結果を表4に示し，以下にまとめる．

- 基本設計工程の完了時に整合チェックツールを適用したプロジェクトである，A，CおよびDにおいて設計項目間の不整合が検出された．
- 詳細設計工程の完了時に整合チェックツールを適用したプロジェクトである，A，D，EおよびFにおいて設計項目間の不整合が検出された．
- 詳細設計工程での設計レビュー前に整合チェックツールを適用したプロジェクトであるBとCは，詳細設計工程の完了時に適用したプロジェクトAとEに比べ，設計項目間の不整合の検出密度が高い．ただし，プロジェクトDとFは，プロジェクトBとCより，設計項目間の不整合の検出密度が高い．
- プロジェクトDは，その他のプロジェクトと比較すると設計項目間の不整合の検出密度が10倍から100倍以上高い．

(2) 不整合の見落としの防止

- BとFプロジェクトのテスト工程での欠陥検出の結果を**表5**に示す．表5中でのUTとは，プログラムのクラス別の単体テスト，PTとは，複数のクラスを結合し，1つの機能を持つプログラムのテストのことである．UT，PTの各テストのチェックリスト数，プログラム規模1,000ステップ（KS）当たりのチェックリストを密度，検出した欠陥の件数，プログラム規模KS当たりの欠陥の件数について，テスト計画時に設定した予定との比較として示している．テストの実施量の妥当性を確認するために，チェックリスト数の予定と実績を比較した．
- UTのチェックリストは予定より多くなっていたが，欠陥は予定より減少した結果となった．PTのチェックリストも同様であるが，Bプロジェクトの欠陥は予定より多くなった．両プロジェクトにおいてUTとPTの合計は，チェックリストと検出した欠陥件数共に，UTと同じ傾向となった．
- BおよびFプロジェクトでの設計からテストで検出した欠陥の件数の予定と実績を比較した結果を，**図5**および**図6**に示す．図5および図6のグラフの

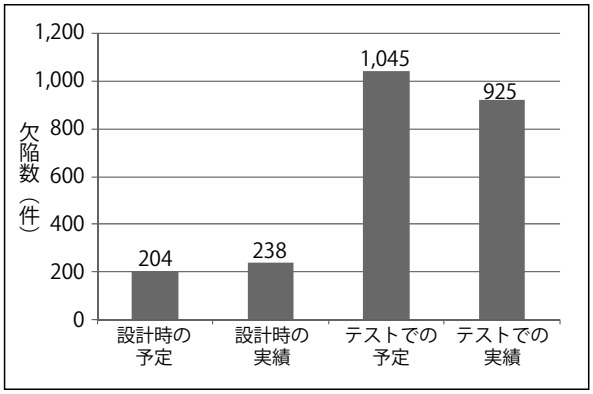

図5　Bプロジェクトでの欠陥検出の予定と実績

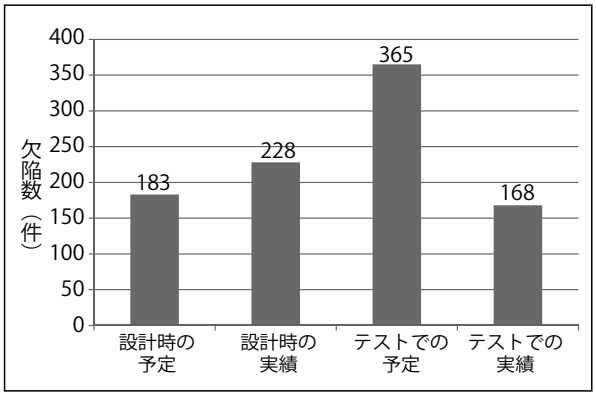

図6　Fプロジェクトでの欠陥検出の予定と実績

横軸の，設計時の予定／実績とは，設計レビューでの欠陥検出数の予定と実績，テストでの予定／実績とは，UTからPTのテストでの欠陥検出の予定／実績を示している．縦軸は，欠陥検出数を示している．両プロジェクトにおいて，設計レビューで検出した欠陥件数は増加したが，テストで検出した件数は減少し，設計レビューとテストでの欠陥検出数の合計は予定よりBプロジェクトで6.9%，Fプロジェクトで27.7%減少した．

(3) 不整合の検出の効率への有効性

BとFプロジェクトでの設計からテストの開発工数の計画と実績の比較結果を示す．Bプロジェクトでは，設計からテストまでの計画工数は42.2人月，実績は38.1人月となり，予定に比較して9.7％減少となった．Fプロジェクトでは，設計からテストまでの計画工数は53.2人月，実績は52.0人月となり，予定に比較して2.3％減少となった．

(4) 整合チェックツールの適用時に要した工数

BおよびFプロジェクトへの整合チェックツールの適用時に要した工数を**表6**に示す．整合チェックツールの適用プロセスは，適用プロジェクトにおける整合チェックツールの運用設計，整合チェックツールのためのソフトウェア環境の構築や検出ルールの設計と設定，整合チェックツールの実行確認，適用プロジェクトへの運用の引き継ぎとした．実行確認時には，結果を確認し設定ミスを修正した．実行確認の範囲は，一部に限定して実施した．合わせて，整合チェックツールの改修の必要有無を判断するために，適応プロジェクトの設計書に対する整合チェックツールのFit&Gapを整理した．BとFプロジェクトへの適用においては，整合チェックツールの改修の必要はなかった．適用に要した工数の中で，検出ルールの設定と実行確認の工数は，検証の対象とした設計書の数に比例した結果となった．

4.3 考察

実験の結果より，本論文で提案した支援システムの有効性の評価を以下に示す．

4.3.1 不整合の検出の自動化

不整合の検出の自動化への有効性については，支援システムにより，各プロジェクトで設計工程での設計項目間の不整合を検出することができ，設計レビューで見逃した設計項目間の不整合の検出に有効であることが明らかとなった．また，設計レビュー前に整合チェックツールを適用した場合は，設計レビューで検出する設計項目間の不整合を事前に検出できていることが判明した．これは，支援システムが設計レビューの品質の向上に寄与したと考える．

表6 適用に要した工数

項	適用	プロセスの内訳	プロジェクト	
			B(人日)	F(人日)
1	運用設計	運用プロセスの検討と プロセス図作成	2.0	1.5
2	環境構築	前提ソフトウェアの準備	0.2	0.2
3		整合チェックツールの準備	0.1	0.1
4		検出ルールの設計	4	1
5		整合チェックツールの環境構築	0.3	0.3
6		検出ルールの設定と実行確認	3	1.5
7		Fit&Gap 結果の整理	0.5	0.3
8	引き継ぎ	整合チェックツールの実行 手順書の作成	0.5	0.5
9		引き継ぎ作業	0.5	0.3
10	合計	-	11.1	5.7

DとFプロジェクトでは，基本設計，詳細設計での支援システムの適用の結果，欠陥の検出密度が他のプロジェクトに比べて高い．特に詳細設計工程時の検出数が他のプロジェクトと比較して高い．設計レビューでの検証の不足を示している．DとFプロジェクトでは，詳細設計工程にオフショア開発を適用しており，その作業品質に問題があった．支援システムは，設計レビューの不足を補うことが可能である．

BとFプロジェクトにおいての欠陥件数の比較から，設計からテスト工程において検出する欠陥の全数を減少させることを確認できた．設計書の作成時に，提案した支援システムにより設計項目間の不整合を取り除くことができ，設計書への欠陥の作り込みを減少させることを確認できた．また，設計レビュー時には，設計項目間の不整合は取り除かれている．処理ロジックなどの設計仕様の内容のレビューに時間を割くことができるようになり，設計工程で欠陥を取り除く割合が従来より増加したためである．

4.3.2 不整合の見落としの防止

不整合の見落としの防止への有効性を評価するために，PTで検出された欠陥の原因の分析し，設計工程での設計項目間の不整合の見落としをどの程度防止できたかを検証した．分析結果を**表7**に示す．設計項目間の不整合を原因とした欠陥は，表7の項番2に示すように，基本設計書の設計項目に対してはBプロジェクトでは12件，Fプロジェクトでは0件となっている．詳細設計書

表 7　PT で検出された欠陥の原因

項	原因の分類	原因	プロジェクト	
			B（件）	F（件）
1	要求仕様書の誤り	要求仕様誤り	0	1
2	基本設計書の誤り	設計時の検討不足	13	7
3		設計書間の整合の確認不足	12	0
4		要求仕様書の理解不足	1	2
5		修正漏れ，修正誤り	1	0
6		誤記・脱字，表現上の問題	1	0
7		その他	1	2
8	詳細設計書の誤り	詳細設計時の検討不足	27	3
9		設計書間の整合の確認不足	0	0
10		基本設計書の記述見落とし	9	0
11		基本設計書の理解不足	4	0
12		設計基準の不備	2	0
13		設計変更の指示漏れ	1	0
14		設計基準の違反	1	0
15		その他	10	0
16	コーディングの誤り	処理ロジックの誤り	79	8
17		詳細設計書の記述見落とし	18	0
18		修正漏れ，修正誤り	13	0
19		コーディング時の誤り	5	3
20		言語仕様の理解不足	1	4
21		関連基準の不備	1	0
22		その他	16	0
23	合計	-	216	28

の設計項目に対しては，両プロジェクト共に0件であった．この結果は，支援システムにより，詳細設計書の設計項目間の不整合が網羅的に検出できたこと示している．Bプロジェクトについては，基本設計書の設計項目間の不整合が，設計レビューで検出しきれず，詳細設計工程で検出されていた．基本設計書に対しても，本論文で提案した検出方法の適用が有効であると考える．

ただし，提案した支援システムを用いた設計項目の整合の検証対象は，自動的に検出可能なものに限定されている．設計項目間の対応関係における内容の詳細化の場合は，支援システムは整合を検証する方法を提供していない．依然，設計レビューでそのような整合を確認する必要があり，そのための方法はレビューアに依存している．設計項目の内容の詳細化をした対応関係における不整合を検出する新たな方法が必要である．

4.3.3　不整合の検出の効率への有効性

不整合の検出の効率への有効性については，支援システムを適用したBとFプロジェクトにおいて，設計とテストを合わせた実績工数が, 計画工数より減少したため, 効率の向上に有効であることが分かった．BとFプロジェクトにおいて，設計レビューとテストで検出した欠陥の件数の予定と実績を比較した結果, UTとPTの合計は，チェックリストは予定より多くなっていたが，欠陥は予定より減少していた．設計レビューで欠陥が検出され，テスト工程まで検出されない欠陥が減少したため，予定よりテスト量を増やしたにも関わらず，テスト工程での欠陥が少なくなった．テスト工程での欠陥が少なくなり欠陥の修正工数が減少したため，開発工数の実績が計画に対して減少したと考える．

4.3.4　適用コスト

整合チェックツールの適用コストは，Bプロジェクトで11.1人日（0.6人月），Fプロジェクトで5.7人日（0.3人月）であった．これらはプロジェクトにおいて追加工数となり，両プロジェクトの設計からテストの開発の実績工数に対して，Bプロジェクトにおいて1.6%，Fプロジェクトにおいて0.6%相当の工数となった．整合チェックツールによる，開発工数の計画に対する実績の減少の割合と比較して下回っており，整合チェックツールの適用の追加工数を投入しても，開発工数全体での工数の削減の効果があると考える．

4.3.5　支援システムの適用プロセス案

表4のDプロジェクトでは，設計項目間の不整合の修正に，多くの時間を設計工程の最後で費やしたことになった．スケジュールに与える影響が大きく計画の変更が必要であった. 設計工程の完了時に欠陥が検出されても，修正のスケジュールを追加することは困難である．設計レビュー前と設計工程の完了時の両方で使用することにより，整合の検証，欠陥の修正の精度と効率の向上に繋がる．支援システムの適用プロセス案を以下に示す．

(1) テンプレートを用いた設計書の作成
　各設計者が設計書のテンプレートに基づき，設計書を作成する．設計書のテンプレートに基づき，設計項目間の対応関係と整合を意識した設計を行う．

(2) 設計者による検出ルールの設定
　各設計者が，設計項目の対応関係の整合を検証したい対象に，検出パターンと検出アルゴリズムを割り当て，検出ルールを設定する．

(3) 設計者による設計項目の整合の検証
　各設計者が整合チェックツールを用いて，各自が作成した設計書間の設計項目の整合を検証する．検出した不整合を修正する．

(4) 設計レビューアによる検出ルールの設定
　設計レビューアが，複数の設計者が作成した設計書間の設計項目の対応関係において，整合を検証したい対象に対して，検出パターンと検出アルゴリズムを割り当て，検出ルールを設定する．

(5) 設計レビューアによる設計項目の整合の検証
設計レビューアが整合チェックツールを用いて，設計書間の設計項目の整合を検証する．検出した不整合を設計者にフィードバックし，各設計者が修正する．

(6) 設計レビューアによる設計項目の整合の検証
設計レビューアが，設計項目の内容の検証など整合の検証以外のレビュー観点で，設計レビューを実施する．検出された欠陥を設計者が修正する．

(7) 品質管理者による設計項目の整合の最終検証
品質管理者が設計工程完了時に，整合チェックツールを用いて，最終検証を実施する．検出した不整合を設計者にフィードバックし，各設計者が修正する．

本案の有効性の検証は，今後の課題である．

5. まとめ

本論文では，ソフトウェア開発プロジェクトの設計工程で，設計書の設計項目間の不整合の検出するための，設計書のテンプレートと整合チェックツールで構成した支援システムを提案した．

設計書のテンプレートには，設計書に記述すべき事項と構造が定義されている．このテンプレートを用いて作成した設計書に対して，整合チェックツールによって，設計項目間を自動的に比較する．その結果，設計項目間の不整合を検出することが可能となった．この支援システムをソフトウェア開発プロジェクトの設計レビューに適用した．その結果，設計項目間の不整合の見落としを防止できた．

本論文で提案した支援システムの評価結果は，特定の企業のプロジェクトを対象としたものである．設計書のテンプレートは，各企業が独自に設定することが多い．設計項目間の対応関係も独自なものとなり，本論文で提案した設計項目の対応関係と異なる．そのために，支援システムをそのまま他の企業のプロジェクトに適用することはできない．ただし，整合チェックツールの検出ルールは，ソフトウェア開発で共通的なものである．各企業の設計書のテンプレートでの設計項目間の対応関係に基づき検出ルールを対応づければ，整合チェックツールの適用が可能になると考える．本論文で提案した支援システムを，さらに複数のプロジェクトに適用し，設計項目間の不整合の検出における有効性，および開発工数の削減効果の精緻化が必要である．

支援システムは，提案した設計書のテンプレートが適用可能なシステム開発において活用できる．設計書のテンプレートを使用して記述された設計書だけが，提案した整合チェックツールにより不整合が検出可能となるからである．ソフトウェア開発プロセスの計画時に，設計書のテンプレートの開発，または本論文で提案した設計書のテンプレートの適用の検討を実施することにより，支援システムの活用が可能となる．

設計項目間の整合を検証する個所やパターンを，設計書の作成前に，設計書のテンプレートを参照して設定すれば，設計項目間の整合を確保した設計書の自動生成が可能になる．設計項目間の不整合を防止し，設計書の作成の作業品質と効率を向上することができる．設計項目の整合を確保した設計書を自動生成するシステムは，次の研究課題とする．

参考文献

1) Avanthi, R. and Sreenivasan, P.: Managing Requirements Across Analysis and Design Phases using IBM Rational System Architect & IBM Rational DOORS, Technical Report, IBM (2010).
2) Boehm, B. and Basili, V. R.: Software Defect Reduction Top 10 List, *Computer*, Vol.34, pp.135-137 (2001).
3) Boehm, B. W.: Verifying and Validating Software Requirements and Design Specifications, *Software*, IEEE, Vol.1, No.1, pp.75-88 (1984).
4) Letelier, P.: A Framework for Requirements Traceability in UML-based Projects, *In Proc. of the 1st Intl. Workshop on Traceability in Emerging Forms of Softw. Eng.*, pp.32-41 (2002).
5) Perry, D.: Where do Most Software Flaws Come from?, *Making Software: What Really Works, and Why We Believe It* (Oram, A. and Wilson, G., eds.), O'Reilly Media, Chapter25, pp.453-494 (2010).
6) Pinheiro, F. A. and Goguen, J. A.: An Object-Oriented Tool for Tracing Requirements, *IEEE Software*, Vol.13, No.2, pp.52-64 (1996).
7) Porter, A. A., Lawrence G., Votta, J. and Basili, V. R.: Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment, *IEEE Transactions on Software Engineering*, Vol.21, pp.563-575 (1995).
8) Ramesh, B. and Jarke, M.: Toward Reference Models for Requirements Traceability, *IEEE Transactions on Software Engineering*, Vol.27, pp.58-93 (2001).
9) Roger, S. P.: Review Techniques, *Software Engineering: A Practitioner's Approach, 7th International edition*, McGraw-Hil, Chapter15, pp.416-431 (2009).
10) Shull, F., Rus, I. and Basili, V.: How Perspective-Based Reading Can Improve Requirements Inspections, *Computer*, Vol.33, pp.73-79 (2000).
11) Spence, I. and Probasco, L.: Traceability Strategies for Managing Requirements with Use Cases, Technical Report, Rational Software (2000).
12) Thelin, T., Runeson, P. and Wohlin, C.: An Experimental Comparison of Usage-Based and Checklist-Based Reading, *IEEE Transactions on Software Engineering*, Vol.29, pp.687-704 (2003).
13) Thelin, T., Runeson, P. and Wohlin, C.: Prioritized Use Cases as a Vehicle for Software Inspections, *IEEE Software*, Vol.20, pp.30-33 (2003).
14) Travassos, G., Shull, F., Fredericks, M. and Basili, V. R.: Detecting Defects in Object-oriented Designs: Using Reading Techniques to Increase

Software Quality, *OOPSLA'99: Proc. of the 14th ACM SIGPLAN conference on Object-oriented Programming, Systems, Languages, and Applications*, pp.47-56 (1999).
15) Wallace, D. R. and Fujii, R. U.: Software Verification and Validation: An Overview, *Software, IEEE*, Vol.6, No.3, pp.10-17 (1989).
16) Wiegers, K.: *Peer Reviews in Software: A Practical Guide*, The Addison-Wesley Information Technology Series, Addison-Wesley (2002).
17) 大橋恭子，栗原英俊，田中ユカ，野津昌弘，山本里枝子：ビジネスアプリケーション開発における追跡可能性構築の取り組み，ソフトウェアエンジニアリング最前線 2010 ―ソフトウェアエンジニアリングシンポジウム 2010（SES 2010）予稿集，近代科学社，pp.155-160 (2010).
18) 元山　厚，中谷多哉子：ソフトウェアレビューのための設計仕様メタモデルの提案，電子情報通信学会技術研究報告，KBSE，知能ソフトウェア工学，Vol.110, No.305, pp.1-6 (2010).
19) 元山　厚，中谷多哉子：設計項目間の不整合を検出するためのレビューチェックリストの開発手法の提案，電子情報通信学会論文誌 D，Vol.J96-D, No.11, pp.2692-2704 (2013).
20) 森崎修司，野中　誠，込山俊博，清田辰巳，細川宣啓，永田　敦：ソフトウェアレビュー／ソフトウェアインスペクションと欠陥予防の現在，情報処理，Vol.50, No.5, pp.375-417 (May 2009).

元山 厚（正会員）at.motoyama@gmail.com
1992 年東北大学農学部農芸化学科卒業後，（株）日立製作所に入社．博士（システムズ・マネジメント）．現在，エンタープライズ系アプリケーションのソフトウェア開発とサービス展開に従事中．電子情報通信学会，プロジェクトマネジメント学会各会員．

中谷 多哉子（正会員）nakatani@gssm.otsuka.tsukuba.ac.jp
東京理科大学理学部応用物理学科卒業後，1995 年にエス・ラグーンを起業．博士（学術）．現在，筑波大学大学院ビジネスサイエンス系准教授．要求工学，ソフトウェア工学の研究に従事．電子情報通信学会，日本ソフトウェア科学会，プロジェクトマネジメント学会，IEEE-CS，ACM 各会員．

投稿受付：2013 年 8 月 19 日
採録決定：2014 年 2 月 13 日
編集担当：藤瀬哲朗（（株）三菱総合研究所）