テスト視点による上流工程での予防活動と検知活動の成熟度向上

羽田 裕† 青木 教之†

ソフトウェア開発の上流工程における欠陥の予防活動と検知活動(設計レビュー)は，フロントローディングを実現するための重要な活動である．筆者らは，上流工程の予防活動と検知活動の質の向上を図るために，ドメイン分析を適用してテストの概念を整理したテスト観点ツリーを作成し，実際のプロジェクトに適用した．これによって欠陥混入の予防，欠陥の発見，製品に対する開発チームメンバの教育に有効であることの見通しを得た．

Maturity improvement of preventive activity and detective activity in upper process by viewpoint of testing

Yutaka Hada† Noriyuki Aoki†

In upper process about software development, the preventive activity of defect and the detective activity of defect (design review) that are important activity to achieve front-loading. We made testing viewpoint tree from testing concept by domain-analysis and used it for real projects. By this, we got the prospects that become effective for the prevention of defect, the detection of defect, the education of development team member about product.

1. はじめに

ソフトウェア開発の短期化に対応するため，Wモデルのような，開発の早い段階から品質を作り込むフロントローディングへの取り組みが盛んである[1]．当社でもWモデルに取り組んではいるものの，開発プロジェクト全体の工数に対するテスト工数の比率は顕著に低くなっていない．

フロントローディングへの取組みを確実に進め，品質目標を達成し，かつ高効率な開発をするためには，品質保証活動にかかわる予防活動，検知活動，修正活動の具体的な内容や相互関係を品質保証計画として立案しておくことが必要である[1][2]．すなわち，品質保証にかかわる活動を計画し，その活動を着実に遂行することが品質目標の達成をより確実にするのである．品質保証活動のうち，予防活動は各種成果物の作成段階において，そもそも欠陥を混入させない活動であり，開発標準の適用はその一例である．検知活動はレビューとテストに代表される活動である．特に上流工程の設計作業における予防活動と検知活動がフロントローディングのポイントとなり，後者では成果物そのものの品質に主眼を置いた設計レビューがポイントとなる．

本稿では，上流工程の欠陥の予防活動と検知活動(設計レビュー)の質の向上を図るために，ドメイン分析を適用してテストの概念を整理したテスト観点ツリーを作成し，実際のプロジェクトに適用した活動について述べる．第 2 章でテスト観点ツリー作成の動機とねらいについて延べる．第 3 章でドメイン分析を行い，テスト観点ツリーを作成するまでの活動について述べる．第 4 章で設計作業と設計レビューへのテスト観点ツリーの適用方法を説明する．第5章で実際の開発プロジェクトへ適用した結果を述べ，第6章でその結果について考察する．

2. 動機・ねらい

2.1. 動機

従来の設計レビューにおいては，仕様や開発標準，過去の障害事例，開発のベースとなるシステムあるいは類似システムの成果物，レビューア個人の経験やスキル・ノウハウ，などがレビューアの指摘の拠り所となっている．これら指摘の拠り所のうち，暗黙知を形式知にすることが，予防活動や設計レビューを効果的な活動

† 日本電気通信システム株式会社
NEC Communication Systems Ltd.

にするとされ，従来から開発標準やチェックリストなどで形式知として利用されている．すなわち，暗黙知を形式知とし，予防活動と設計レビューの道具とすることで，より多くの欠陥，あるいは重大な欠陥について，設計時の混入を防ぎ，設計レビューで発見できるのである．

しかし，設計レビューでは，まだ幾つかの暗黙知がレビューアの指摘の拠り所となっている．以下にいくつかの例を示すが，これらの暗黙知も形式知とすることで，予防活動や設計レビューをより効果的な活動とすることが期待できる．

- 仕様では，システムの利用条件や非機能要件などが暗黙の要求となっており，すべてが形式知になっていることが少ない．
- 過去の障害事例では，開発に対する影響の小さな障害の対策は，開発者個人の経験やスキル・ノウハウとなっており，暗黙知になっていることが多い．
- 開発のベースとなるシステムあるいは類似システムの成果物では，暗黙の要求と仕様との関係は明記されていることが少なく，暗黙知となっていることが多い．またそれらの仕様ですら，テストなどの開発の下流工程で変更・追加された場合などは，暗黙知となっていることがある．

これらの暗黙知を形式知にするため，筆者らは設計レビューと同じ検知活動であるテストに着目した．設計レビューは上流工程になるほど，テストは下流工程になるほど妥当性確認の要素が強くなる[1]．このような関係があるので，暗黙の要求や過去の障害事例の対策も含めた妥当性確認である下流工程のテストの概念は，上流工程の予防活動と設計レビューでも利用できるはずである．そこで，筆者らはソフトウェア開発者のテスト観点に関する知見を整理して，テスト観点ツリーを作成した．暗黙知を形式知とし，予防活動と設計レビューの道具としようというのである．

2.2. ねらい

テスト観点ツリーを上流工程の予防活動と検知活動（設計レビュー）に適用し，それぞれの活動の質の向上を図るためのねらいについて述べる．

(1) 予防活動

① 欠陥混入の予防

設計作業における欠陥の混入防止をねらう．設計作業に着手する前に，テスト観点ツリーから設計者が設計時の留意点（設計留意点）を選定し，それに留意して設計することで欠陥の混入防止を図る．

(2) 検知活動（設計レビュー）

設計レビューでは，参考文献[1]のレビュー手法に期待される 14 の効果（表 1）のうちの，以下の②と③の効果をねらう．

表 1 レビュー手法に期待される主な効果（参考文献[1]）

種別＼効果	欠陥製品の発見	仕様との適合性のチェック	標準との適合性チェック	製品の完全性と正しさの検証	理解性と保守性の評価	クリティカルまたは高リスクのコンポーネントの品質の実証	プロセス改善のためのデータ収集	文書品質の測定	製品に関する他のチームメンバの教育	対処法についてのコンセンサスの決定	変更または欠陥修正の正しさの確認	別の対処法の検討	プログラム実行のシミュレーション	レビューコストの最小化
アドホック・レビュー	●													●
ペア・プログラミング	●				●				●	●		●		
ピア・デスク・チェック	●	●	●									●		●
ピア・レビュー	●	●	●	●					●	●	●			
パス・アラウンド	●	●	●		●				●					
チーム・レビュー	●	●	●				●		●	●	●			
ウォークスルー	●			●	●				●	●	●	●	●	
インスペクション	●	●	●	●	●	●	●	●						

② 欠陥の発見

従来手法では発見できない欠陥の発見をねらう．設計レビューにおいて，レビューアの指摘の拠り所にテスト観点ツリーを加えることで，従来であれば総合テスト，あるいはそれ以降に流出する恐れのある欠陥を設計レビューで発見する．

③ 製品に対するチームメンバの教育

製品に対するチームメンバの設計能力向上をねらう．設計レビューを通して，テスト観点を用いた設計法やテスト観点の設計へのフィードバック法をチームメンバで共有し，チームメンバの設計における多角的な視点を育成する．また，チームメンバ全員による，開発対象システム用のテスト観点ツリー作成を通して，製品に関する知見を共有し，設計に必要な製品知識を習得する．

3. テスト観点ツリーの作成

テスト観点とは，何をテストすべきかを示す概念である[3]．テスト観点ツリーは，テストやレビューの際に確認すべき観点をツリー状に整理したものである．筆者らはテスト観点ツリーを，当社の主要な業務である，通信機器の組込みソフトウェア開発の分野を対象とし，ドメイン分析[4]の手法を適用して作成した．

ドメイン分析は，対象システム自身が本来持つ各種の性質や開発上の様々な知識を十分に分析し，認識して組織化し，システムの開発に有効な，共通の対象領域（ドメイン）に属する，用語，問題のとらえ方，シス

テムの構造，システムの作り方などの，固有な概念構造を得るプロセスである．この概念構造をドメインモデルとよぶ．このドメインモデルを用いることによって，個別システムの開発生産性向上と再利用促進を図るものである[4]．

整理の方法として，JIS X 0129-1 の品質特性[5]による分類法を適用したツリー構造を用いた．具体的には，ひとつずつの品質特性について，その品質特性・品質副特性の下位に当たる特性を列挙し，階層的に表現した．階層的な表現は，マインドマップを用いたツリー構造とした．列挙にはブレインストーミングを用い，通信機器に求められる特性やテストすべき特性を，次々とテスト観点としてリストアップし，階層的に配置していった．品質特性に分類することが困難なものは，残しておいて最後に見直すこととした．結果的に，第 1 階層に 6 つの品質特性と動作環境，システム機能とを配置し，第 2 階層以下に品質副特性や詳細な条件を配置したツリー構造としてテスト観点を整理した（図 1,2）．

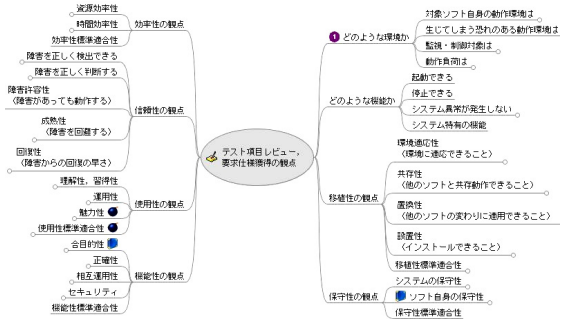

図 1 テスト観点ツリー（第 2 階層まで展開）

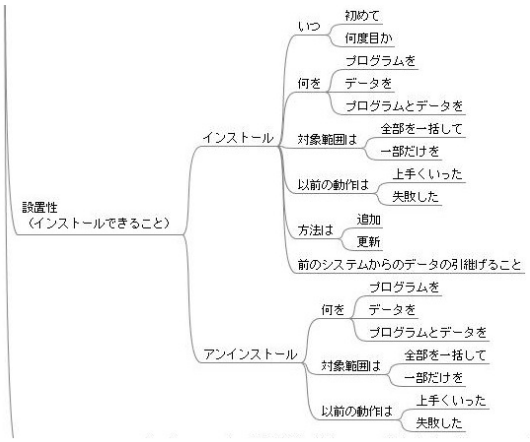

図 2 テスト観点ツリー（設置性のノード）

筆者らのテスト観点ツリーは，一番上に上位水準（要件仕様獲得，テスト項目レビューの観点），その下に基本水準（品質特性の観点），さらにその下に下位水準（品質副特性の観点）の 3 つの水準を持つ階層構造であり，認知におけるカテゴリーと類似した構造[6]である．従って，ツリーの利用者にカテゴリーが何を表現しているかの知識があれば，ツリーを使ってカテゴリー帰納推論[6]を促し，ツリーを適用する開発対象システムのテスト観点の導出を期待できる．
カテゴリー帰納推論には，一般帰納と特殊帰納がある．一般帰納は注目するカテゴリーの特徴を，それを包含する上位のカテゴリーに適用する推論である．特殊帰納は注目するカテゴリーの特徴を，同じレベルにあるカテゴリーに適用する推論である[6]．

4. テスト観点ツリーの適用方法

本章では，上流工程の設計作業と設計レビューにテスト観点ツリーを適用する方法について説明する．

4.1. 適用手順

テスト観点ツリーの適用の手順を以下の①～⑧に示す．

① テスト観点ツリーのテーラリング
② テスト観点ツリーのリスト化（テスト観点リスト作成）
③ リストから設計留意点の選定
④ リストのレビュー
⑤ リストのレビューの指摘結果をリストへ反映
⑥ 設計留意点に留意しつつ，設計作業を実施
⑦ 成果物の設計レビュー
⑧ 設計レビューの指摘結果を成果物へ反映

4.2. 適用手順の詳細

前節で述べた適用手順の詳細を説明する．

(1) テスト観点ツリーのテーラリング

開発プロジェクト着手時点に，テスト観点ツリーを開発対象システムに合せてテーラリングし，開発対象システム用のテスト観点ツリーを作成する．
ドメインモデルのテスト観点ツリーに対して，開発メンバ全員でブレインストーミングを用い，開発対象システムの上位要求から「不要観点の削除」，「不足観点の追加」，「対象システムの主要な機能部分の観点の詳細化」などを行って，ツリーを再構築する（図 3）．

ブレインストーミングで，ドメインモデルのテスト観点ツリーの各ノードから，開発メンバに代表性ヒューリスティック[6]やカテゴリー帰納推論を起こさせ，開発対象システムに合致したノードを表出させやすくし，個人の経験やスキル・ノウハウといった暗黙知の表出を促す．

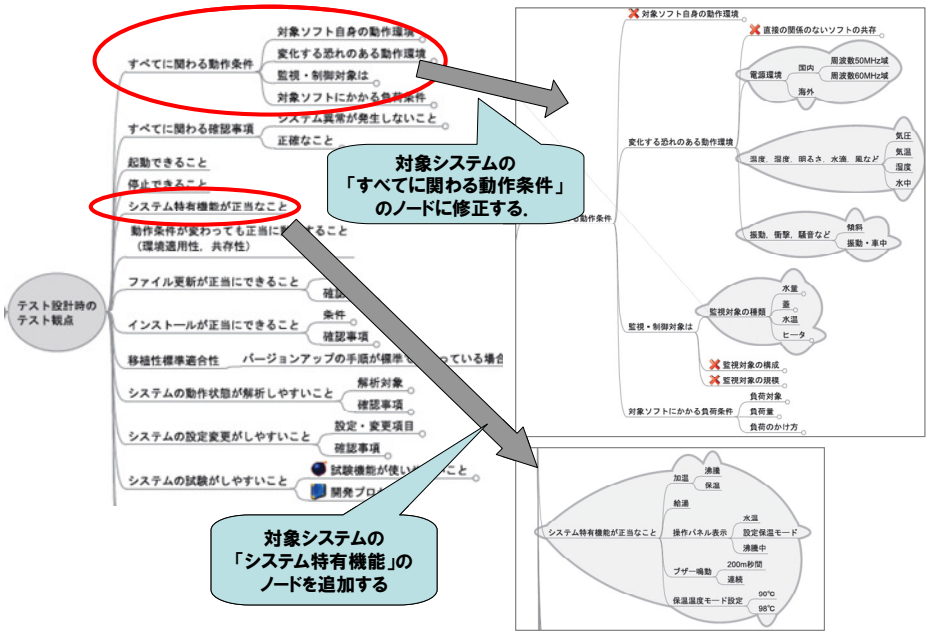

図 3 テスト観点ツリーのテーラリング例

また，開発メンバ全員でテーラリングすることで，開発対象システムと，ドメインモデルや既存システムモデルとの「差」を共有し，後の設計作業において，設計者に対する欠陥混入の予防効果を与える．同時に，後の設計レビューにおいて，レビューアに欠陥を発見しやすくさせる．DRBFM[7]と同様に，差を見て問題に気づく効果をねらうのである．

(2) テスト観点ツリーのリスト化（テスト観点リスト作成）

テーラリングしたテスト観点ツリーを，表形式に変換し，テスト観点リストとする（図 4）．

表の 1 行を 1 つのテスト観点とする表形式にして，表の 1 行（テスト観点）ごとに，設計留意点のチェック，レビュー結果，設計への反映有無，設計留意点を選定した根拠，それぞれを記録する欄を設ける．表形式にして，これらの欄を設けたのは，プロジェクト適用時の運用・管理を容易にし，テスト観点ごとに適用効果を確認するためである．

- 設計留意点のチェック
  設計者が該当テスト観点を設計留意点の対象とするかしないかを記録する．
- レビュー結果
  リストのレビュー後，設計者が該当テスト観点を設計留意点の対象とするかしないか記録する．対象としたテスト観点が，設計時の留意点となる．
- 設計への反映の有無
  設計留意点を設計に反映させたかどうかを記録する．要求や条件の新たな導出の元となった，あるいは，上位要求に対する確認の元となった設計留意点を，設計に反映させた設計留意点とする．設計レビュー前に設計に反映させた場合は「設計時」と記録する．設計レビュー時に設計留意点による指摘で設計に反映させた場合は「レビュー後」と記録する．
- 設計留意点を選定した根拠
  設計者が該当テスト観点を設計留意点の対象とした根拠を文章で記録する．

(3) リストから設計留意点の選定

設計者は，リストのテスト観点のうち，設計作業で留意すべきテスト観点を設計留意点として選定し，リストの「チェック」欄に記録する．また，選定した根拠を「設計留意点を選定した根拠」欄に記録する．

設計留意点は，「初めて」「変化」「久しぶり」の 3H[8]も考慮して選定する．

(4) リストのレビュー

リストをレビューする．リストのレビューのポイントは，設計者が選定した設計留意点の妥当性である．レビュー手法としては，チーム・レビューを基本とし，設計留意点を選定した根拠についても議論する．

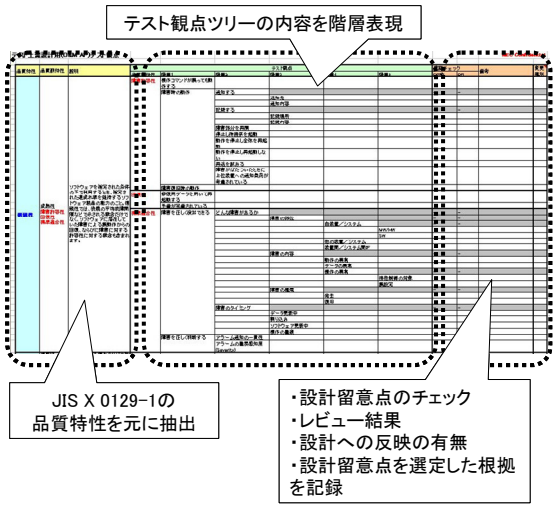

図 4 テスト観点リスト

(5) リストのレビューの指摘結果をリストへ反映

設計者は，リストのレビューの指摘結果をリストへ反映させる．リストのレビューでの指摘を勘案し，再び設計作業で留意すべきテスト観点を設計留意点として選定し，リストの「レビュー結果」欄に記録する．

(6) 設計留意点に留意しつつ，設計作業を実施

設計者は，「レビュー結果」欄で選定した設計留意点を確認しながら，設計作業を実施する．

設計留意点から，要求や条件などが新たに導出された，あるいは，上位要求に対して確認すべき事柄が発生した場合には，リストの該当テスト観点の「設計への反映の有無」欄に「設計時」として記録する．それらの対応がなかった場合は「反映無」として記録する．

(7) 成果物の設計レビュー

リストを参照しつつ，設計作業の成果物をレビューする．

(8) 設計レビューの指摘結果を成果物へ反映

設計者は，設計レビューの指摘結果を成果物とリストへ反映させる．

設計レビューでの指摘を勘案し，設計留意点による指摘で成果物を変更した，あるいは，上位要求に対して確認すべき事柄が発生した場合には，リストの該当テスト観点の「設計への反映の有無」欄に「レビュー後」として記録する．それらの対応がなかった場合は「反映無」として記録する．

5. 開発プロジェクトへの適用結果

テスト観点ツリーを 2 つの開発プロジェクト α, β に適用した結果を示す．適用した順序は α, β の順である．

5.1. 適用した開発プロジェクトの条件

(1) 開発チームのメンバ構成

メンバ構成は，β で開発メンバを 1 名追加した以外は，α と β は同一である．リーダも同じである．

(2) チームの開発対象

開発対象は，α，β とも，同じ機能ユニットである．

(3) 適用した設計作業

適用した設計作業は，α，β とも，参考資料[2]のソフトウェア要求仕様書の作成と，ソフトウェア・アーキテクチャ設計仕様書の作成にあたる作業である．

(4) テスト観点の数

テスト観点リストのテスト観点の総数は，α，β とも，140 個である．

5.2. 開発プロジェクトの設計留意点の数

α と β の設計アイテムごとの設計留意点の数をそれぞれ表 2, 3 に示す．

表中の分類は以下を表す．

- 設計アイテム
  ソフトウェアの開発計画時の開発アイテム．
- 担当

設計アイテムの設計者．担当 A は初級者（経験 2 年未満），担当 B と C は中級者（経験 2〜5 年），担当 D と E は上級者（経験 5 年以上）である．

- 設計留意点の数
  - 設計者選定：4.2 節(3)で選定した留意点の数．
  - レビュー結果：4.2 節(5)で選定した留意点の数．
  - 差分：上記の差分．4.2 節(5)において 4.2 節(3)から増加／削減した数と，その絶対値の合計．
- 反映した設計留意点の数
  - 設計時：4.2 節(6)で設計に反映させた留意点の数．
  - レビュー後：4.2 節(8)で設計に反映させた留意点の数．

表 2 開発プロジェクト α の設計留意点の数

設計アイテム	担当	設計留意点の数					反映した設計留意点の数	
		設計者選定	レビュー結果	差分			設計時	レビュー後
				増加	削減	合計		
α-1	A	17	24	11	4	15	0	0
α-2	A	25	9	1	17	18	0	1
α-3	B	9	10	1	0	1	0	2
α-4	B	14	14	0	0	0	0	0
α-5	B	10	11	1	0	1	0	0
α-6	D	6	6	0	0	0	0	0
α-7	E	2	2	0	0	0	0	0
α-8	E	11	11	0	0	0	0	0

表 3 開発プロジェクトβの設計留意点の数

設計アイテム	担当	設計留意点の数					反映した設計留意点の数	
		設計者選定	レビュー結果	差分			設計時	レビュー後
				増加	削減	合計		
β-1	A	20	18	2	4	6	0	0
β-2	A	29	29	0	0	0	0	0
β-3	A	27	27	1	1	2	0	0
β-4	B	23	23	0	0	0	0	0
β-5	B	28	30	2	0	2	1	2
β-6	B	8	8	0	0	0	1	1
β-7	C	15	15	0	0	0	0	0
β-8	D	24	24	0	0	0	0	0
β-9	D	16	17	1	0	1	0	0
β-10	D	3	3	0	0	0	0	0
β-11	D	33	32	1	2	3	0	0
β-12	D	9	9	0	0	0	0	0
β-13	E	32	32	0	0	0	0	0

5.3. 開発プロジェクトへの効果

- 開発生産性(開発規模あたりの工数)
  - α=適用前より 5%向上
  - β=αと同じ生産性
- 総合テスト以降に発見された欠陥(開発規模あたりの欠陥件数)
  - α=適用前の 1/4 に減少
  - β=適用前の 1/6 に減少

テスト観点ツリー適用による作業負荷の影響はみられない. 総合テスト以降に流出した欠陥は適用により減少し, 適用の繰り返しでさらに減少した.

5.4. 「欠陥混入の予防」に対する効果

- 設計時に反映した設計留意点
  - ✓ 設計留意点の数　　α=0, β=2

設計レビュー前の設計時に, 要求や条件の新たな導出, あるいは, 上位要求に対する確認の元となった設計留意点は, αではなし, βでは 2 件あった. 2 件は, それぞれ機能性(正確性)と移植性(環境適応性)のテスト観点を設計留意点としたものである.

これらは, 従来の方法で開発していれば総合テスト, あるいはそれ以降に流出する恐れのある欠陥となりうるものであったが, 流出を防止した.

5.5. 「欠陥の発見」に対する効果

- 設計レビュー後に設計に反映した設計留意点
  - ✓ 設計留意点の数　　α=3, β=3

設計レビューにおいて欠陥発見の元となった設計留意点は, αおよびβとも 3 件あった. αでは, 保守性(解析性), 機能性(正確性)(2 件)のテスト観点を設計留意点としたものである. βでは, 信頼性(回復性), 保守性(変更性), 移植性(置換性)のテスト観点を設計留意点としたものである.

これらは, 従来の方法で開発していれば総合テスト, あるいはそれ以降に流出する恐れのある欠陥となりうるものであったが, 流出を防止した.

5.6. 「製品に対するチームメンバの教育」に対する効果

- 設計留意点のレビュー前後の数の変化
  - ✓ 設計初級者の差分
    - α(2 つの設計アイテム):15 個, 18 個
    - β(3 つの設計アイテム):6 個, 0 個, 2 個
  - ✓ 上記以外の担当者の差分
    - αとβで大きな差はなし

設計の初級者(担当 A)は, リストのレビュー前後の差分が, αでは 2 つの設計アイテムで各々15, 18 個と, 他の開発メンバと比較すると多かった. βでは 3 つの設計アイテムで各々6, 0, 2 個となり, 減少傾向となった.

設計の初級者は, 設計対象の望ましい性質を推定する能力が十分でなく, 最初に選定した設計留意点に対して, リストのレビューで多くの過不足数が発生したと考えている. 適用の繰り返しによって過不足数が減少していることから, 担当者の知識・設計力が向上していることを示していると考えられる.

5.7. 改善のための設計者へのヒアリング結果

a) 設計留意点を選定する時間の変化
  - ✓ 設計アイテムあたりの平均時間
    - α=0.5 時間, β=0.25 時間

適用当初は工数の増加が見られたが, 適用に慣れてから工数の増加は見られなかった.

b) 設計留意点のレビュー時間の変化
  - ✓ 設計アイテムあたりの平均時間
    - α=0.5 時間, β=0.25 時間

適用当初は工数の増加が見られたが, 適用に慣れてから工数の増加は見られなかった. また, テスト観点ツリー適用により, 従来のレビュー時間にリストのレビュー時間が加わったが, 適用前後の開発規模あたりのレビュー時間に大きな変化は見られなかった.

c) 選定する設計留意点の傾向

設計留意点として選定されるテスト観点は, テスト観点ツリーの第 1 階層別で見ると偏りが見られた(図 5).

機能性(セキュリティ), 効率性(資源効率性), 移植性(環境適応性)の一部は, 留意点として選択されていない(図 5 の①). 信頼性(障害許容性, 信頼性標準適合性)は, ソフトウェア要求仕様書作成時に留意点として選定されることが多い(図 5 の②). 移植性(環境適

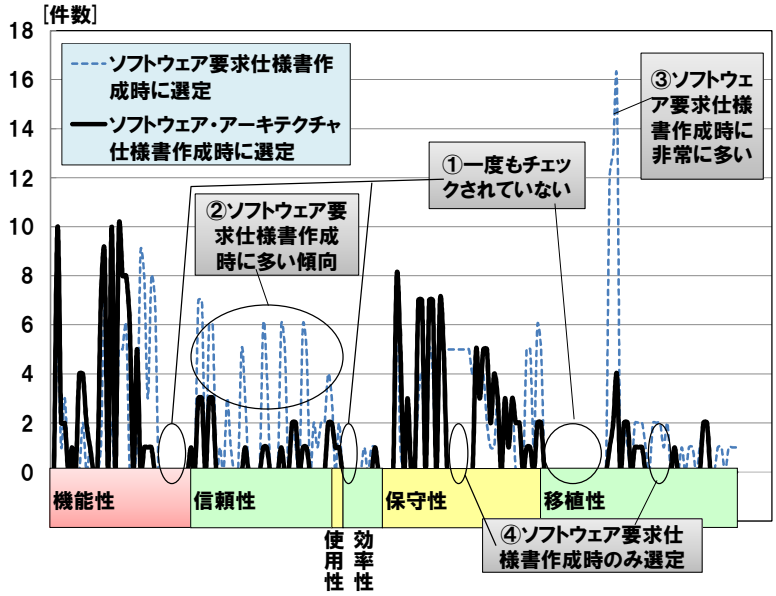

図 5 設計留意点としたテスト観点の件数分布

応性)の一部は，ソフトウェア要求仕様書作成時に留意点として選定されることが非常に多い(図 5 の③)．保守性(解析性)の一部や移植性(環境適応性)の一部は，ソフトウェア要求仕様書作成時のみに選定されていた(図 5 の④)．

d) 設計留意点の表現の影響の有無

テスト観点ツリーのノードを，もう少し設計対象システムに近い表現にした方が良いという開発メンバがいた．一方で，効率的なテーラリングのために抽象度の高い表現が良いという開発メンバがいた．しかし，設計時に反映した設計留意点の数(5.4 節)や，設計レビュー後に設計に反映した設計留意点の数(5.5 節)に対する影響は見られなかった．

e) 設計時における設計留意点の使い方

選定した設計留意点は，設計者により使い方が異なった．事前に留意点を確認してから設計する，設計と留意点の確認を逐次繰り返す，設計の終わりに留意点を確認する，など様々な使い方がある．これらの使い方は，設計者の経験度との関連は見られなかった．

6. 考察

6.1. 適用結果に対する考察

2 つの開発プロジェクトにテスト観点ツリーを適用し，いずれのプロジェクトでも，従来では発見できなかったと思われる欠陥を発見することができた(5.4 節，5.5 節)．これまでの開発標準に加えて，設計対象ごとの設計留意点を提示したことによって設計品質が向上したことを意味する．これまでは設計留意点を明示的には示しておらず，設計者自身の知見に頼っていた．これに対して，開発チームの知見を総合して共有して適用したことが有効だったと言える．

適用の繰り返しに従って観点の理解が進んでいることが観測された(5.6 節，5.7 節 a, b)．整理のしかたが間違っていないことを示しているものとみている．

選定された設計留意点を，テスト観点ツリーの第 1 階層別で見ると偏りがあることが観測された(5.7 節 c)．まったく選択されていない留意点については，あらためて以下のことを吟味する必要があろう．

- 開発対象のシステムで留意すべきものか
- 開発対象の機能ユニットで留意すべきものか
- 適用した設計作業(ソフトウェア要求仕様書の作成，ソフトウェア・アーキテクチャ設計仕様書の作成)で留意すべきものか

この吟味は，テスト観点ツリーのテーラリングの方法やテーラリングの妥当性を検証する方法につながる可能性があり，今後の課題と考えている．

α，β とも，テスト観点ツリーのテーラリング状況について観測を行わなかった．この状況を観測することで，設計者の暗黙知の傾向や，設計者の観点理解の進み方を把握できる可能性がある．状況によっては，

テーラリング作業が設計レビュー以上の効果を生んでいることも考えられる. テーラリング状況の観測については, 今後の課題としたい. さらに, 観測結果の分析は, 開発チームの弱点の発見に活用できる可能性があり, チーム力の改善に有効かもしれない.

設計者によって, より具体的な観点を好むものと, より抽象的な観点を好むものがいることが分かった(5.7 節 d). 観点の表現については, β への適用時に, テスト観点ツリーのテーラリングにおいて, 該当の開発プロジェクトでポイントとなるノードを選定し具体的な表現の観点とした. このような部分的な表現の使い分けは, まだ直接効果に結び付いていないため, 今後効果を見定めることが必要である.

設計者によって, 設計時の設計留意点の使い方が多様であることが分かった(5.7 節 e). 「留意」は, 認知心理学では「注意」と同類とされ, 集中的注意, 選択的注意, 分割的注意, 予期・期待の種類がある[6]. 注意が働く状況や注意それぞれの強さは, 設計者の作業条件や個人差があり, 使い方に制約を設けることは適当ではなかろう.

設計時に反映した設計留意点の数が, α では 0 件, β では 2 件と, それほど多くないにも関わらず(5.4 節), 開発プロジェクトにおいて, 総合テスト以降に流出した欠陥数の減少が見られた. このことから, 設計時に反映させたと記録された件数以上に, 留意点を考慮したことによって品質を作りこんでいるものと考えられる. これは開発プロジェクトの生産性の向上(5.1 節)からも推定される. 直接計測できない留意点の効果については, さらに多くの開発プロジェクトで監視したい.

6.2. テスト観点ツリーの表現と構造に対する考察

本節では, より効果的に適用するためのテスト観点ツリーの表現と構造について考察する.

(1) 認知行動をより活性化させるための表現と構造

筆者らのテスト観点ツリーは, 利用者にカテゴリー帰納推論を促すことで, 設計対象システムのテスト観点の表出を期待するものである. 従って, 認知におけるカテゴリー構造に近づけることが有効である. そのためには, 以下についての検討が必要となる.

① 3 つの階層からなる構造にする
② 第 1 階層ノードの名称を短くする
③ 第 1 階層ノードは同じ階層相互の区別が容易なものにする
④ 第 1 階層ノードは日常頻繁に利用する名称にする
⑤ 第 1 階層ノードは設計初級者でも理解できる名称にする

現在のテスト観点ツリーは, 階層数が最高 5 つである. また, 第 1 階層ノードとした品質特性の認知度は社内では高いとは言えない. 今後の検討課題としたい.

さらに, 各テスト観点の典型性は, 代表性ヒューリスティクの起こりやすさに大きな影響を与える[6]. ドメインモデルのテスト観点ツリーのテスト観点は, 必ずしも典型性が高いものばかりではない. テスト観点を表出した人にとって最近発生したものや, 印象的なものが含まれる, 利用可能性ヒューリスティック[6]など, 代表性以外の性質のヒューリスティックによって表出したものである可能性がある. 従って, テスト観点の典型性を高めるための工夫も検討すべきであろう. 例えば, 第 1 階層の観点の特徴を幾つか列挙し, その特徴と一致した合計の数で典型性を測る, 家族的類似性[6]の利用も 1 つの案である.

(2) テスト観点の多面性への対応

今回報告したプロジェクトでは, ドメイン分析で作成したテスト観点ツリーをテーラリングし, 品質特性のみを第 1 階層に配置したテスト観点ツリーとした. テスト観点とは, 詳細化するとテストケースになるような抽象概念である. 例えば, 負荷といった入力条件, インストールといった機能, USB 接続といったテスト対象の構成要素, セキュリティといった品質特性, バッファオーバーフローといった欠陥のパターンなどがテスト観点として挙げられる[3]. 筆者らがドメイン分析で作成したテスト観点ツリーでも, 品質特性以外のこれらの特性を配置しているが, それでも導出しにくい特性があった. そのため, 以降のプロジェクトでは, 経験上仕様書の備考欄の記載箇所から欠陥が発生しやすいことから「備考テスト」という特性を第 1 階層に加えた.

欠陥混入の防止と欠陥の発見というねらいに対して, テスト観点ツリーをさらに効果的なものにするためには, テスト対象とするコンポーネントやシステムにおける, 機能, トランザクション, フィーチャー, 品質特性, 構成要素のようなテストの基盤となるものを特定して[10], その組み合わせを検討し, 網羅的かつ排他的になるよう整理することが必要であろう.

7. おわりに

筆者らのテスト観点ツリーは, 通信機器のソフトウェア開発のテストに関する暗黙知を形式知とするもので, SECI モデルの表出化[9]を促すものである. 設計者は自身の知識・経験をこれに追加することによって, 開発対象システムのテスト観点として整理することができる. 適用の効果は設計者のスキル・暗黙知に依存することになるが, これを開発チームで見直し共有することによ

って，形式知を組み合わせて新たな知識を創る連結化[9]を行い，改善することができる．

実際のプロジェクトに適用して残存する品質問題の削減に有効なことが確認できた．また，適用時のテーラリングや設計レビューを通した，設計者の知識教育，設計力育成にも有効であることの見通しを得た．今後は，適用プロジェクトから示されたテスト観点ツリーの課題について解決するとともに，テスト観点ツリーの適用を拡大することで，用途の多様化を図る．

謝辞

本研究を進めるにあたり，多くの助言を頂いた電気通信大学　西康晴先生に感謝する．また，本稿執筆にあたり，助言を頂いた国立情報学研究所　石川冬樹先生に感謝する．

参考文献

[1] IPASEC，高信頼化ソフトウェアのための開発手法ガイドブック, IPASEC, 2010
[2] IPASEC, [改訂版]組込みソフトウェア向け開発プロセスガイド，翔泳社, 2007
[3] 吉澤智美，西康晴，ソフトウェアテストの最新動向とフロントローディング，日本品質管理学会誌論文, Vol.42, No.4, pp467-477, 2011
[4] 伊藤潔，田村恭久，吉田裕之，杵嶋修三，広田豊彦，ドメイン分析・モデリング―これからのソフトウェア開発・再利用基幹技術，共立出版, 1996
[5] JIS X 0129-1:2003(ISO/IEC 9126-1:2001): ソフトウェア製品の品質 第1部: 品質モデル，日本規格協会, 2003
[6] 箱田裕司，都築誉史，川畑秀明，萩原滋，認知心理学，有斐閣, 2010
[7] 吉村達彦，想定外を想定する未然防止手法GD3，日科技連, 2011
[8] 工場管理増刊 3H(初めて,変更,久しぶり)の理論と実践 2011年12月号，日刊工業新聞社, 2011
[9] 野中郁次郎，紺野登，知識創造の方法論―ナレッジワーカーの作法，東洋経済新報社, 2003
[10] ISO/IEC/IEEE FDIS 29119, Software and systems engineering‐Software testing‐ Part 1:Concepts and definitions, 2013