自律した品質改善活動に寄与する欠陥特性の提案

The important Defect Characteristics for suitable improvement activities on software quality

主査　：　細川　宣啓　日本アイ・ビー・エム（株）
副主査　：　永田　敦　　ソニー（株）
リーダー：　太田　範　　関電システムソリューションズ（株）
研究員　：　奥山　剛　　アズビル（株）　　高橋　功　　ソーバル（株）
　　松本　達平　（株）インテック　　森　龍二　　（株）エクサ
　　渡邊　孝志　テックスエンジソリューションズ（株）

研究概要

企業のソフトウェア開発において，品質管理基準などの活用により達成すべき品質目標は守られている．一方で，達成を阻害する具体的なソフトウェア欠陥に対して，その基準や回避方法が明確に示されないことにより，品質の確保や改善を難しくする例も多い．我々は，欠陥の定義や性質の不十分な理解が，品質に対する活動を阻害するものと捉え，欠陥に関する知見の整理を試みた．

本研究では，欠陥の定義を明確化する 3 つの基本特性と，欠陥の混入過程から導いた 4 つの混入特性からなる「欠陥特性」を提案する．欠陥の性質を踏まえた品質改善活動は，欠陥の定義などの曖昧さに起因する問題を抑制し，より効果的に機能することが期待できる．欠陥特性の有効性検証として，欠陥に対する認識共有の改善と，特性を利用した改善施策の評価を行った結果，提案する基本特性と混入特性が有効であることを確認した．

Abstract On the enterprise software development, quality control standard gives us control metrics and guarantee of good software quality. But the standard does not show us what to avoid or how to prevent defect injection. In some case, this makes it difficult to keep and improve quality. We regard this difficulty happens as a result of insufficient understandings of property and behavior in defects. In this study, we try to organize the knowledge as "Important Defect Characteristics". These characteristics consist of 7 characteristics, including definitions and life cycles of defects. We found these characteristics help us reduce misunderstandings of defects, and evaluate our improvement activities on software quality.

1.　はじめに

1.1　背景と問題

ソフトウェア開発の品質を確保するため，多くの企業において品質管理基準や開発ガイドラインを制定しており，品質の確保が図られている．また，それらを基に品質目標の達成状況が日常的にやりとりされている．

一方，品質を損なうソフトウェア欠陥（以降欠陥という）に関する議論は一般にあまり知られていない．例えば，多くの規格が欠陥という言葉を定義しているが，定義内容はそれぞれに異なり[1]、また具体的な個別の欠陥に対する統一された分類や定義も存在しない．更に，欠陥が持つ属性や性質に関する既存研究も広く知られるとは言えない．

これは，ソフトウェア開発において，具体的に何をどのように避けるべきかが共有されないまま，達成すべき品質が議論されている可能性を示す．特に，品質の管理・改善が奏功していない企業においては，立場や状況による欠陥認識の違いから，欠陥の伝達齟齬や曖昧な修正指示により，手戻りが日常的に発生している．また，品質の改善に際して，十

分な理由が説明されないまま，主観のみに基づく効果の薄い対策を採用する例もしばしば見られる．

これらの状況から，以下の 2 点が問題として挙げられる．

(1) ソフトウェア品質を確保するために，避けるべき対象自体が不明確である

(2) ソフトウェア品質を改善するために，どのように欠陥を避けるべきかが不明確である

1.2 研究の狙い

前節の 2 つの問題を解決するためには，特定の技法ではなく，まずは元となる欠陥の属性や性質を理解し，共有することが効果的であると我々は考えた．これは，欠陥に対する理解の統一が，欠陥に関する円滑なやりとりや，混入過程の共通認識に繋がり，ひいては品質基準や予防活動の浸透をより深めると考えたためである．そこで，欠陥が欠陥として認識されるための条件や欠陥の性質を「欠陥特性」として整理することにより，以下の貢献を試みる．

RQ1. 欠陥に関する理解の曖昧さが低減され，欠陥に対する認識が一致するか

RQ2. 合理的・客観的なデータから品質改善に繋がる施策を提案できるか

以降本稿では，欠陥特性の着想と導出を述べたうえで（2 章），各特性の詳細を説明する（3，4 章）．また，上記の研究課題（RQ）を検証し，評価・考察を行う（5 章）．

2 提案する欠陥特性の着想と導出

2.1 欠陥特性の着想

欠陥特性を考えるにあたって，参考にしたものが品質モデル（ISO 9126，25010）である．ソフトウェア開発の現場では欠陥の明確な定義が存在せず，何らかの暗黙的な基準によって欠陥の判定をしているのが現状である．この状況は，品質の明確な定義がない状況で品質管理をしなければならなかった ISO 9126 の策定前の状況に似ていると我々は考えた．

品質モデルは，品質要求と品質評価の基礎を与えるような特性の集合および特性間の関係である[2]．ここで品質モデルと同じアプローチをとれば，現時点で統一された定義のない欠陥に対しても，欠陥を特徴付ける特性に分解することで，評価・予測・予防に繋げられるのではないかという着想を得た．

2.2 基本特性の導出

我々は欠陥が持つ特性（以降欠陥特性という）の整理に際し，まず初めに，欠陥とは何かを明らかにするため，これが満たされれば欠陥と言えるという性質（以降基本特性という）の分類を試みた．

JSTQB（Japan Software Testing Qualifications Board）では，欠陥の定義を「要求された機能が実現できない原因となる，コンポーネント又はシステムに含まれる不備」とし，「故障（期待した機能，サービス，結果からの逸脱）を引き起こす」とも述べている[3]．つまり，欠陥はその性質として，原因としての不備，結果としての逸脱，それらを結ぶ因果関係を伴うものと理解できる．本研究ではこれを設計工程などの中間成果物を含む欠陥にも適用し，基本特性を表 1 に示す特性群として整理することとした．

表 1 導出した基本特性

特性名	特性概要
具現性	成果物(中間成果物を含む)に不備として含まれる性質
有害性	期待された結果からの逸脱を引き起こす性質
因果性	混入背景から混入，表出までの過程が，因果関係を持つ性質

2.3 混入特性の導出

次に我々は、欠陥の混入を回避するために、基本特性の因果性に着目した．欠陥は一度混入された後，複製により数が増大したり，不適切な除去により他の種類へ変異することがある．これはある欠陥の混入から除去までのライフサイクルを考えたとき，欠陥の混入

に関与する特性（以降混入特性という）として整理できると我々は考えた．そこで，混入特性の分類を，様々な混入事例を用いた因果性の観察と，先行研究による欠陥の検証結果から試みた結果，表 2 に示す特性群を得た．なお，先行研究等の詳細は 3 章以降で述べる．

表 2 導出した混入特性

特性名	特性概要
連鎖性	時間・工程・対象を跨ぐ形で，欠陥が別の欠陥を発生させる性質
好欠乏性	人・組織，作業対象，環境等の不足が大きいほど，欠陥の混入確率が増幅される性質
好複雑性	人・組織，作業対象，環境等が複雑であるほど，欠陥の混入確率が増幅される性質
増殖性	成果物のコピーもしくは再利用により，ある欠陥が再生産される性質

3. 基本特性の詳細

3.1 具現性

欠陥は，要件定義・設計を含む作業対象の成果物に，不備として具現化される性質を持つ．また，欠陥は，作業対象と故障モードの組み合わせで表現できることが先行研究[4]で示されている．作業対象と故障モードの例を図 1 に示す．この組み合わせ表現により「メモリリーク」などの曖昧な欠陥表現が，「モジュール設計の解放制御が／存在なし」もしくは「コードにおけるデータハンドリングにおいてメモリ操作が／場所誤り」などの形で具体化され，欠陥内容を正確に共有することが可能になる．

一方，組み合わせ表現による欠陥の識別において，識別可能な欠陥の粒度は定義された作業対象の粒度に制限され，欠陥の全体集合は全作業対象と全故障モードの組み合わせに制限される．このことから，先行研究等による様々な欠陥の分類も，ある利用目的に応じて粒度や範囲を制限した組み合わせ事例と捉えられ，目的に応じて欠陥の表現が異なることを示す．

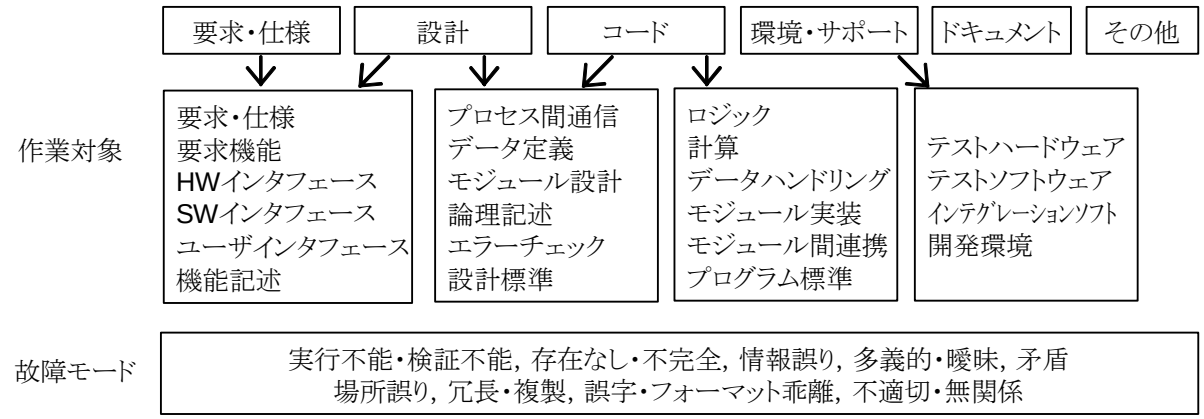

図 1 作業対象と故障モードの例

3.2 有害性

欠陥は，故障や品質の阻害などの形で，期待された結果からの逸脱を引き起こす性質をもつ．成果物に対する期待された結果を，成果物の内容が満たすべき品質と捉えた場合，その逸脱は品質特性の裏返しで表現することが可能である．このことから，具体的な有害性としては表 3 の内容が挙げられる．

品質特性との違いは，その副特性として欠陥の連鎖を助長する要素（連鎖助長性）を含む点である．これは，成果物同士が密接に関連するソフトウェア開発の性質から，中間成果物上の欠陥が，別の欠陥の混入原因となりえることに起因する．

また，品質の阻害について言及する場合，誰にとっての有害性なのかを考慮する必要がある．ソフトウェアが満たすべき品質は，顧客要求に依存することが一般に知られている一方で，テストやレビューの実施者が，顧客要求に基づかない有害性に対し，本来不要な修正を開発者に要求することがしばしばある．そのため，具現性の内容と併せて，誰にと

っての有害性かを関係者間で明らかにすることで，欠陥が残置されるリスクや修正の必要性をより適切に共有することが可能になる．

表 3 有害性の副特性

副特性	副特性概要
機能阻害性	目的との乖離、不正確さ、脆弱性、運用上の連携阻害など、機能性を阻害するもの
連鎖助長性	時間・対象を跨いで、欠陥の連鎖や伝播を助長するもの
不信性	トラブル時の非回復、単一障害による全停止、未成熟など、信頼性を阻害するもの
難使用性	難理解・習得、運用阻害など、使用性を阻害するもの
非効率性	処理の非効率(時間的、資源的)など、効率性を阻害するもの
難保守性	解析困難、不安定、変更困難、試験困難など、保守性を阻害するもの
移植阻害性	環境・規格不適合、設置困難、置換困難など、移植性を阻害するもの

3.3 因果性

欠陥は，その混入から有害性の表出，除去までのライフサイクルにおいて，因果関係を伴う性質を持つ．つまり，欠陥には，混入背景・原因・過失行動などの混入に至った理由が存在する．

3.3.1 因果の多様さ

欠陥の因果関係は多くの因子によって構成され，多様かつ複雑なものとなる．例えば，原因の種類から有害性を一意に特定することは難しい．また，ある混入背景が確実に欠陥を混入させるとも言えない．このことから，欠陥の混入予測や影響予測，未然防止の難しさは，因果の多様さにも起因するものと考えられる．

3.3.2 因果の構成要素

細川らの研究では，欠陥混入メカニズムを表現，伝達するためのモデル（以降欠陥モデルという）を提唱している[5]．欠陥モデルにおいては，欠陥の混入背景や原因を以下の因子の組み合わせで表現できる．

(1) 誘発因子：人間の思考における誤りを誘発する因子
(2) 増幅因子：欠陥混入の確率を増幅させる外乱因子もしくは環境因子
(3) 過失因子：人間の判断の誤りそのもの

この 3 つの因子で構成された混入背景や原因を受けて，欠陥は混入され有害性をもたらす．つまり，因果性は混入の背景や原因，具現性，有害性を結ぶものといえる．また，欠陥モデルにおいて，作業対象と故障モードの組み合わせはランダムに決まるのではなく，3 因子の作用によって必然的に決まるものと考えられる．なお，本研究においても欠陥モデルを基本として欠陥に関する具体事例を検証しており，各種因子により欠陥は混入されるものとした．

4. 混入特性の詳細

4.1 連鎖性

欠陥は，前工程から残存する欠陥や，現工程における欠陥の除去などに起因して，連鎖的に新たな欠陥を混入させる性質を持つ．前工程から残存する欠陥は，中間成果物間における工程を跨いだ入出力関係の流れに乗り，人の判断や誤りを介して新たな別の欠陥を混入させる．また，先行研究[6]からは，欠陥除去の 2～5%程度が別の欠陥を混入させることが示されており，これらから連鎖性が時間，対象，修正を跨ぐ形で存在するといえる．

欠陥の連鎖は，その連鎖が跨がる期間や工程が長いほど，除去コストが指数的に増大する性質を併せ持つ[7]．実際に，テスト工程で検出される欠陥において，要件定義や設計工程で混入された欠陥が多く含まれ，その除去に多大な手戻りコストを要する例も珍しくない．因果性による影響予測の難しさも考慮すると，現工程で混入した欠陥は現工程内で除去することが，一般に言われるとおり望ましいといえる．

4.2　好欠乏性

欠陥は，表 4 の項目が量的質的に不足するほど，混入確率が増幅される性質を持つ．

表 4 欠乏性の内容と具体例

項目	内容	具体例
人的欠乏性	人的，組織的な知見や能力が不足すること	スキル不足，要員変更や離脱，教育不足など
対象欠乏性	要求，設計，コード等に含まれるべき要素や関連性が不足すること	ソースコードのコメント不足，例外漏れ，設計記載ムラ，記載内容不足など
環境欠乏性	プロジェクト制約，開発環境，プロセスなどが不足すること	短納期，短作業期間，レビュー不足，テスト漏れ，標準化不足など

4.3　好複雑性

欠陥は,表 5 の項目が量的質的に複雑であるほど,混入確率が増幅される性質を持つ．

表 5 複雑性の内容と具体例

項目	内容	具体例
人的複雑性	組織構造，関係者(数，役割，増減)が複雑であること	多組織，多関係者，成果物所有の組織上の深さなど
対象複雑性	要求，設計，コード等が含む要素や関連が複雑であること	コード複雑度，変更量，変更頻度，依存関係，境界など
環境複雑性	プロジェクト制約，開発環境，プロセスが複雑であること	重厚な開発標準，過度の管理など

4.3.1　複雑性の因子と欠陥予測

好複雑性の根拠としては，欠陥予測の正しさが挙げられる．先行研究において，複雑性や欠乏性を伴う個々の因子を基にした欠陥予測モデルが示されており，その正しさが確認されている．一例として Nagappan らの研究結果[8]では，組織構造，コード変更量，コードの複雑さ，依存関係などを予測上の因子として用いることで，因子毎に 8 割程度の確率で欠陥群を予測でき，欠陥全体の 8 割程度を網羅できることが示されている．

4.3.2　欠陥の偏在

欠陥が偏在することは一般に知られているが，偏在は好複雑性，好欠乏性に起因して発生する．これは，好複雑性の因子を利用した欠陥予測モデルの検証結果と，別研究による偏在検証の結果[9]から説明が可能である．

欠陥の偏在は，好複雑性，好欠乏性の各因子が重なる形で欠陥の混入確率が増幅された結果，特定の場所に集中したと捉える事ができる．換言すると，偏在の度合いは，組織・人・対象・環境に属する各因子が持つそれぞれの複雑さの度合い，ならびに欠乏の度合いが組み合わされることにより、決まることになる．

一方，組み合わされる各因子はそれぞれ異なる影響範囲を持つ．作業対象の複雑さに関する因子の影響は作業対象に閉じ，組織もしくは環境の複雑さに関する因子の影響はプロジェクト全体に及ぶ可能性がある．そのため，欠陥の遍在に着目して検出・予防活動を行う場合，偏在を導く各因子の影響範囲と影響の大きさに留意することで，それらの活動がより効果的にはたらく可能性が高いと考えられる．

4.4　増殖性

欠陥は，成果物のコピーもしくは再利用などによって再生産される性質を持つ．

5.欠陥特性の有効性検証

1.2 節で述べた研究課題の達成を評価するため，基本特性と混入特性に対する検証を行った．RQ1 に対する基本特性の検証では，特性の利用により欠陥が持つ曖昧さが低減され，

組織内での欠陥に対する認識がより統一されるかを確認する．また，RQ2 に対する混入特性の検証では，組織が蓄積する検出欠陥の情報から，特性の利用により合理的・客観的な品質改善の提案に繋がるかを確認する．

5.1　基本特性の検証

5.1.1　検証方法

組織内における欠陥への認識の変化を確認するため，欠陥に対する従来の表現（表 6，表現 1）と，特性を考慮した表現（表 6，表現 2）の 2 表現を用いてアンケート調査を実施した．表現 1 は，我々の組織におけるレビューでの指摘に見られる表現を用い，表現 2 は具現性と有害性を考慮した表現を用いる．

表 6 検証に用いた欠陥の表現

	表現1：従来表現	表現2：基本特性による表現
事例1	コード内に固定値で記述されている	[具現性](対象)コード内のデータ定義 ／ (故障モード)誤り, フォーマット乖離 [有害性] 現在機能は阻害しない(機能阻害性なし). 但し, 開発者に対し, デバッグ・機能拡張時の保守を困難にする(難保守性)可能性
事例2	例外処理が抜けている(但しcatchしないと思われる場所)	[具現性] (対象)コード内のエラーチェック ／ (故障モード)存在なし, 不完全 [有害性] 現在機能は阻害しない(機能阻害性なし). 但し, 将来的に拡張(連鎖助長性)した場合, ユーザに対し, エラー時の非回復を起こす(不信性)可能性
事例3	「○○した場合, 入力を拒否する」と仕様が曖昧に記載	[具現性] (対象)設計における機能記述 ／ (故障モード)曖昧, 検証不能 [有害性] 開発者が実装時に(連鎖助長性), 仕様誤認から目的乖離(機能阻害性)を起こす可能性. 期待動作が不明確なため, テスト担当者の試験を困難にする(難保守性)可能性

アンケートは，欠陥を表す 3 つの事例に対して，それぞれが欠陥といえるか，修正を行うかの観点から回答させるものとした．アンケートの回答者は，表現 1 に対して回答をしたのち，具現性と有害性の説明を受け，表現 2 に対して回答を行う．なお，回答者は組織が異なる 30 人のソフトウェア技術者を抽出し，本検証を実施した．

5.1.2　検証結果

アンケート結果の人数を表 7 に示す．表現 1 は平均で 40%の人が「欠陥ではない」と回答し，欠陥認識にばらつきが出たが，表現 2 では「欠陥ではない」と回答したのはわずか 6%程であった．また，回答過程においても，幾つかの違いが確認された．

表 7 表現による欠陥認識の変化

	表現1:従来表現			表現2:基本特性による表現		
回答	事例1	事例2	事例3	事例1	事例2	事例3
欠陥であり, 修正する	10	18	26	28	27	30
欠陥ではない	20	12	4	2	3	0
合計	30	30	30	30	30	30

欠陥を定義しない状態（表現 1）での回答では，回答者は 1 分程度の比較的長い時間を要し，機能阻害性の欠陥のみを欠陥と識別する傾向が見られた．この状態では，欠陥認識の曖昧さに起因した品質のばらつきが懸念される．

基本特性の説明を受けた後の状態では，表現 2 に対して 5 秒程度で回答を返しており，連鎖助長性を含む内容についても,即座に欠陥であり修正するべきものと識別した.また，表現 2 に対する感想として「誰の有害性か明らかで納得感がある」,「レビューのお手本のような表現だ」などの回答を得た．

基本特性に基づく表現は，認識された事柄と望まれた事柄を，具現性と有害性により明確に表す．これらの状況から，基本特性が欠陥の曖昧さを低減させ，組織内における欠陥認識の改善に寄与するものといえる．

5.2　混入特性の検証

5.2.1　検証方法

既存の欠陥情報から，品質改善の提案に繋がるかを確認するため，欠陥情報と混入特性の関連を評価した．具体的には，ある欠陥集合における混入特性の出現比率を算出することで，欠陥がどのような混入背景に起因して最も混入されたかを評価する．なお，欠陥情報は，各組織が保有するバグ票を用い，バグ票に記載された複数の混入原因を，検証のために細分化した混入特性に関連付けたうえで算出を行った．

本検証では，まず組織間で混入特性の比較を行う．組織の成熟度や対象とするソフトウェアの性質などに応じて，混入特性の出方は組織毎に異なることが予想される．層別により異なる結果が得られれば，最も頻度が高い混入過程に対してそれぞれの組織に応じた予防措置や混入予測が可能になると考えられる．

また，各組織で取り組んでいた改善施策と混入特性の比較を行い，従来の改善施策に対する妥当性の評価を試みる．なお，本検証は以下の 3 組織に対して行い，合計約 100 欠陥を抽出のうえ実施した．

・組織 A　：　組み込み系の開発を多く含む組織．開発プロセスの見直しに取組中
・組織 B　：　パッケージのアドオン開発を中心とする組織．単体テストの
　　カバレッジ強化，品質保証組織の上流工程に対する参画に取組中
・組織 C　：　業務アプリケーションの開発を中心とする組織．

5.2.2　検証結果

混入特性は，予想通り組織毎に大きく異なる結果となった．評価結果を表 8 に示す．

表 8 組織別 混入特性の内訳

組織	連鎖性	好複雑性			好欠乏性			増殖性	合計
		人･組織	対象	環境	人･組織	対象	環境		
組織A	9%	6%	9%	5%	21%	12%	**34%**	5%	100%
組織B	**24%**	4%	**36%**	0%	**28%**	0%	8%	0%	100%
組織C	0%	**29%**	3%	0%	**29%**	21%	19%	0%	100%

組織 A における混入特性は環境欠乏性が多く表れ，組織 B では対象複雑性が主要素となった．また，組織 C では人的複雑性・人的欠乏性が大きく表れる結果となった．混入特性を細分化した詳細情報を以下に示す．

・組織 A　：（環境欠乏性）テスト不足，短納期，（人的欠乏性）知識不足
・組織 B　：（対象複雑性）設計での連結部分の複雑性
・組織 C　：（人的複雑性）多組織，（環境欠乏性）短納期，（人的欠乏性）情報連携の不足

検証した欠陥集合は，各組織とも機能阻害性の欠陥として検出されたものだが，改善に向けて各組織に必要な予防措置や，改善に向けて必要となる分析内容は大きく異なるものになる．これは，品質特性の裏返しである「機能性の喪失」からは導かれ得ず，混入特性への分解から初めて明らかとなった，混入メカニズムそのものを示すと考えられる．

検証結果から各組織における改善施策を導く場合，以下の施策が一例として考えられる一方，各組織で従来実施していた改善施策とはやや異なる結果となった．

・組織 A　：テスト強化，システムに関する組織的な知識の維持　など
・組織 B　：連結部分に対する複雑性の解消，連結部分のレビュー強化　など
・組織 C　：コミュニケーション方法・管理の強化　など

この結果は，混入特性を利用した検証結果と開発現場が認識する課題の比較が，見落とされた課題の発見に繋がる可能性を示す．組織 B の例では，以前から欠陥数や密度による管理に基づいて改善活動を行っていた反面，重大欠陥の減少に繋がらない問題があった．一方，本検証の結果は，既に実施している改善施策とは異なる導出方法により，具体的かつ異なる改善施策を導いている．このことから，混入特性を利用し，欠陥の性質や混入メカニズムに注目することで，より適切な改善活動に繋がる可能性が高いといえる．

5.3 評価・考察

基本特性は，欠陥の定義や性質に関する共有を促し，組織内での品質管理基準に対する認識の統一を導くものと評価できる．従来の品質管理では，開発に関するガイドラインと成果物の適合を評価し，不適合の場合に修正を行っていた．しかしながら，具体的な欠陥の特性を提示することなく，明確なガイドラインや品質管理基準を示すことは困難であるため，ガイドラインの不明確さによる手戻りが懸念される．このことから，基本特性の利用は，ガイドラインの不適切さを改善し，手戻りの抑制に繋がるものと期待できる．

混入特性は，各組織が保有する欠陥情報から，品質改善に向けた議論の深化を導くものと評価できる．検証では，対象となる成果物から混入特性を分離し，特性そのものの偏りを明らかにした．その結果，従来考えられていた混入の原因の他に，見落とされた原因を洗い出している.これは,混入特性が品質改善に関して一定の示唆を与えるものといえる．また，混入特性を，欠陥の基本的な原則として共有することも有用と考えられる．混入メカニズムに対する共通の理解は，基本特性と同様に，品質管理基準への認識の統一に寄与する可能性が高い.その結果,ガイドラインや予防策が,より奏功することも期待できる．

これら欠陥特性の利用は，品質に関する組織内のやりとりにおいて，欠陥を共通言語として扱うことを意味する．従来の品質目標の活用に加え，その達成に向けて避けるべき対象・避け方・その根拠を組織内で合意することは，効果的かつ自律的な品質改善活動を促進するものと考えられる.

6. おわりに

6.1 まとめ

本研究では，欠陥の定義が持つ曖昧さを低減させ，混入メカニズムへの理解を助ける欠陥特性を提案した．欠陥特性は，欠陥が共通して持つ特性と，混入に関する特性を定めており，欠陥の性質に対する理解を促進する．また，特性に対する有効性の検証結果から，品質管理基準に対する組織内での認識の統一と,混入への対策に有効であることを示した.

6.2 今後の展望

本研究に関する今後の課題としては，特性間における関連性の整理，各特性の指標検討と評価，各特性の更なる具体化などが挙げられる.

本研究は，欠陥に関する特性分解から，欠陥のコントロールに繋がる特性の整理を試みたものでもある．欠陥の性質に対する理解を通じて，各組織に適した新たな欠陥分類や改善方策が確立され,将来的な欠陥エンジニアリングとしての議論が高まることを期待する.

参考文献

[1] 独立行政法人 情報処理推進機構(IPA)，″組込みソフトウェア開発における品質向上の勧め[バグ管理手法編]″,2013
[2] JIS X 0129-1 2003.ソフトウェア製品の品質-第1部:品質モデル(ISO/IEC 9126-1:2001)
[3] JSTQB 技術委員会,″ソフトウェアテスト標準用語集日本語版″,2014
[4] L.Margarido et al,″Classification of Defect Types in Requirements Specification″, Information Systems and Technologies (CISTI), 2011 6th Iberian Conference,2011
[5] 細川宣啓ら,″過失に着目した欠陥のモデリング″,JaSST2013,2013
[6] A.Oram,″Making Software: What Really Works, and Why We Believe It″, OReilly,2010
[7] B.Boehm,″Software Engineering Economics″,IEEE TSE,vol.10,no.1,pp.4-21,1984
[8] N.Negappan et al, ″The influence of organizational structure on software quality: An empirical case study″,30th International Conference on Software Engeneering,2008
[9] J.Ostrand et al,″Predicting the Location and Number of Faults in a Large Industrial Software System″,IEEE TSE,31(4):340-55,2005

付録

付録 1. 標準規格などによる用語の定義

出典	障害 (fault)	フォールト (fault)	故障 (failure)	欠陥 (defect)
JIS X 0133-1:1999 ソフトウェア製品の評価	計算機プログラム内の不正確なステップ、プロセスまたはデータの定義	(定義無し)	要求された機能を 遂行する製品の能 力が尽きる状態、または事前に仕様化された制限内での機能を遂行する能力が無い状態	(定義無し)
JIS X 0014:1999 情報処理用語-信頼性、保守性及び可用性	要求された機能を機能単位が遂行できなくなる偶発的条件	(定義無し)	要求された機能を遂行する、機能単位の能力が無くなること	(定義無し)
JIS Q 9000:2006 品質マネジメントシステム-基本及び用語	(定義無し)	(定義無し)	(定義無し)	意図された用途または規定された用途に関連する要求事項を満たしていないこと
JIS C 0508-2:201 電気・電子・プログラマブル電子化安全関係の機能安全	(定義無し)	機能ユニットに要求される機能遂行能力の低下または損失を引き起こすであろう異常状態	ある機能ユニットの要求される機能 遂行能力の終結	(定義無し)
JIS Z 8115:2000 ディペンダビリティ（信頼性）用語	(定義無し)	ある要求された機能を遂行不可能な アイテムの状態、また、その状態にあるアイテムの部分 ※アイテムが要求機能達成能力を失うこと	(定義無し)	

ソフトウェアテスト 標準用語集 V2.0 (日本語版)	defect 参照	(定義無し)	コンポーネントやシステムが、期待した機能、サービス、結果を提供できないこと	要求された機能をコンポーネントまたはシステムに果たせなくする、コンポーネントまたはシステム中の不備。例えば、不正な命令またはデータ定義 。実行 中に欠陥に遭遇した場合、コンポーネントまたはシステムの故障を引き起こす。
IEEE Std 1044-2009 (バグ管理手法部会訳)	ソフトウェアにおけるエラー(error)の現れ	(定義無し)	・要求された機能を遂行するための製品の能力が終了すること、または前もって指定された制限内で要求された機能を果たせないこと・システムまたはシステムのコンポーネントが指定された制限内で要求された機能を果たさない事象	プロジェクトの構成要素において、要求や仕様に合致せず修復か取り替えが必要であり、不完全であること、または不足していること
IEEE Std 982.1-2005 (バグ管理手法部会訳)	コンピュータプログラムにおける正しくないステップまたは処理、データ定義 (IEEE 610.12-1990 fault(2)の引用)	(定義無し)	システムまたはコ ンポーネントが指定された性能要求内で要求された機能を果たせないこと (IEEE 610.12-1990 failure の引用)	原因である fault、または結果である failure のいずれも言及できる総称(Hand book of Software Reliability Engineering R.Lyu 1996 の引用)

出典：組込みソフトウェア開発における品質向上の勧め[バグ管理手法編] 独立行政法人情報処理推進機構(IPA)

付録 2. 具現性における作業対象の詳細分類例

作業対象	詳細	該当成果物			
		要求・仕様	設計	コード	環境
要求・仕様		○			
要求機能		○			
HWインタフェース		○	○		
SWインタフェース		○	○		
ユーザインタフェース	画面, 帳票	○	○		
機能記述	場合分け, 領域, メッセージ	○	○		
プロセス間通信	プロトコル, 型, 初期値		○	○	
データ定義	型, 次元, 初期値, 範囲, アクセス		○	○	
モジュール設計	サブルーチン, コード, 制御フロー, シーケンス, 例外処理		○	○	
論理記述	状態, 視覚表現(括弧・空行など)		○	○	
エラーチェック	機能, 変数, ユーザ入力		○	○	
設計標準	標準化		○	○	
ロジック	比較, アルゴリズム, ネスト, パフォーマンス, 制御フロー			○	
計算	式, 演算			○	
データハンドリング	データ・メモリ, 管理・初期化・操作			○	
モジュール実装	処理, 例外処理			○	
モジュール間I/F	関数呼出, パラメータ, 初期・戻り, 割込			○	
プログラム標準	コメント, デバッグ情報, 名前			○	
テストハードウェア	テスト設計, 初期値, 確認, 構成				○
テストソフトウェア					○
インテグレーションソフト					○
開発環境	言語(型・要素・視認性), OS, 回復, 性能				○

以下の出典での欠陥分類から，作業対象の詳細としてキーワードを分類
出典：「ソフトウェアテスト技法」B.Beizer 著

付録 3.欠陥群予測の精度と因子例

複雑・欠乏の因子	正確性	再現性
組織構造	86%	84%
コード変更量	79%	80%
コードの複雑さ	79%	66%

複雑・欠乏の因子	正確性	再現性
依存関係	74%	70%
テストカバレッジ	83%	54%
リリース前の欠陥発生	74%	63%

・正確性とは，モデルによる予測数のうち実際に欠陥群が得られた比率を指す
・再現性とは，実際の欠陥群のうちモデルで予測された比率を指す
出典：″The influence of organizational structure on software quality: An empirical case study″ N.Negappan et al.

付録 4．検出技法と欠陥特性の関係

欠陥特性		HDR法における欠陥兆候	研究会で得たプラクティス	検出技法
具現性		・(曖昧)文字ばかりで図表がない ・(曖昧・[条件]漏れ)「場合・時」の多用 ・(曖昧)「TODO」「TBD」「ASAP」の利用 ・(曖昧)「など」の利用 ・(フォーマット乖離)様式不統一 ・(フォーマット乖離)記述粒度の差	・(曖昧)全角「？」を検索 ・(曖昧)「はず」「かも」を検索	
有害性				・形式手法によるレビュー ・アルゴリズム分析 ・制御フロー分析 ・インタフェース分析 ・PQ分析(不信性)
因果性			・過去検査でエラー検出履歴あり	・エラー推測テスト
連鎖性			・過去検査でエラー検出履歴あり	
好 複雑性	人的	・章番号の抜け・重複 ・文体が混在 ・異なるフォントが存在 ・ファイルサイズが極端に大きい		
	対象	・ファイルサイズが極端に大きい	・設計者やPGが容易ではないと感じた ・ファイル・コードサイズが極端に大きい ・一覧系などで4つ以上出たら掘る ・広く洗ってみて出たところを掘る ・インデントのへこんだ所を探す ・黒っぽい(文字の多い)所を探す ・「場合」の記述を探す	・複雑度分析 ・境界値テスト ・fault-prone分析
	環境			
好 欠乏性	人的	・様式不統一 ・記述粒度の差	・経験の浅い設計者が作成	
	対象	・空白が多い頁がある	・ファイル・コードサイズが極端に小さい ・記述構成の均一さを見る ・関数名・目次だけを読む	
	環境	・ファイル作成と最終更新が極端に短い ・ファイル更新時間が深夜や早朝	・過去にレビュー不足、相当修正発生 ・遅い時期に設計変更が発生 ・更新時刻や期間を見る	
増殖性		・文体が混在 ・異なるフォントが存在 ・ファイル作成と最終更新が極端に短い	・コピペを発見するために記述をソート	

以下の出典ならびに研究会内の議論で得た，検出兆候，プラクティス，検出技法を関連する欠陥特性に分類

出典 1：「ソフトウェア品質知識体系ガイド v2」SQuBOK 策定部会著

出典 2：「欠陥知識を有効活用したレビュー方法の提案」細川ら(SQiP2013)

付録 5. 予防技法と欠陥特性の関係

欠陥特性		予防技法
具現性		・機能要求分析 ・比機能要求分析 ・USDM ・ODC
有害性	機能 阻害性	・機能分割 ・セキュリティリスク低減技法、設計 ・セキュアコーディング ・セキュリティパターン
	連鎖助長性	
	不信性	・可用性マネジメント ・フレームワーク(保守性・信頼性) ・コーディング規約(信頼性・保守性)
	難使用性	・人間中心設計
	非効率性	
	難保守性	・部品化(保守性) ・設計原則(保守性・再利用性) ・アーキテクチャパターン(保守性) ・フレームワーク(保守性・信頼性) ・デザインパターン(保守性・再利用性) ・コーディング規約(信頼性・保守性) ・TDD(試験性) ・リファクタリング(保守性)
	移植阻害性	
因果性		・ODC ・問題マネジメント

欠陥特性		予防技法
連鎖性		・依存関係マトリクス ・CI ・プロトタイピング ・スパイラルモデル ・ODC
好複雑性 好欠乏性	人的	・スキル標準
	対象	・部品化 ・アーキテクチャパターン ・アーキテクチャ評価技法 ・コーディング規約 ・デザインパターン ・設計原則 ・依存関係マトリクス ・リファクタリング ・複雑度のメトリクス ・内部メトリクス
	環境	・CI ・IDE
増殖性		・コードクローン分析

以下の出典による予防技法を，関連する欠陥特性に分類
出典：SQuBOK 策定部会著「ソフトウェア品質知識体系ガイド v2」

付録 6. 基本特性の検証における回答者の属性

組織	所属部門	業務経験年数
組織A	開発部門	10
組織A	開発部門	34
組織A	開発部門	22
組織A	開発部門	3
組織A	開発部門	12
組織A	開発部門	5
組織A	開発部門	8
組織A	開発部門	15
組織A	開発部門	7
組織A	開発部門	9
組織A	開発部門	13
組織B	QA部門	20
組織B	QA部門	17
組織B	QA部門	15
組織B	QA部門	10
組織B	QA部門	5
組織B	QA部門	4
組織B	QA部門	2
組織B	QA部門	4
組織B	QA部門	11
組織B	QA部門	14
組織C	開発部門	20
組織C	開発部門	15
組織C	開発部門	3
組織C	開発部門	1
組織C	開発部門	11
組織C	開発部門	2
組織C	開発部門	3
組織C	開発部門	7
組織C	開発部門	9

組織	所属部門	業務経験年数
組織C	開発部門	20
組織C	開発部門	15
組織C	開発部門	3
組織C	開発部門	1
組織C	開発部門	11
組織C	開発部門	2
組織C	開発部門	3
組織C	開発部門	7
組織C	開発部門	9