欠陥モデルに基づいたソフトウェア欠陥流出防止アプローチの提案

- 処理に着目し欠陥の特徴を捉える -

A proposal of approach using Defect Based Differ Prevention in software development

- Prevention of defects outflow by characteristics of the defect in software processing -

主査　　：細川　宣啓　（日本アイ・ビー・エム株式会社）
副主査　：永田　敦　　（ソニー株式会社）
研究員　：加藤　賀久　（オムロン株式会社）
　　福田　伊津子　（株式会社東芝　社会インフラシステム社）
　　齋藤　幸裕　（ビジネスキューブ・アンド・パートナーズ株式会社）

研究概要

多くの企業・組織ではソフトウェア欠陥の流出防止に取り組んでいるが，それでもソフトウェア欠陥を市場へ流出してしまっているのが実態である．そこで我々研究チームは，欠陥そのものに着目し，欠陥流出を真に防止できる方法を探った．

本研究では，処理ごとに抽出した欠陥の特徴から対策を導くことで，より上流の開発工程から欠陥の流出を防止する,「欠陥モデルに基づいたソフトウェア欠陥流出防止アプローチ」を提案する．このアプローチにより，欠陥の混入・流出を誘発する因子を特定でき，欠陥流出防止のために対策すべきポイントが明らかになった．処理とソフトウェア欠陥そのものに着目するという従来と違った視点のアプローチにより欠陥流出を効果的に防止できる．

Abstract

Despite the fact that many companies and organizations tackle the outflow of software defects, software defects outflow to the customers. Our research team decides to investigate new method for prevention of the outflow by focusing on the defect itself.

Our research team, with established preventative measure by characteristics of the defect in software processing, proposes “Defect Based Differ Prevention” approach preventing defects at earlier software development process.

This new approach, unlike the ones before, is capable of specifying the factor that lead to the infection and outflow of defects, and clarify the point to take for effective prevention of defects outflow.

1. はじめに

ソフトウェア欠陥（以降，欠陥という）を市場に流出させてしまうと，その欠陥の調査，修正及び検証といった処置に追われる．加えて，同様の欠陥の総点検，顧客への説明などの作業に要員を投入することになり，対応コストが多大に発生する．さらには，他製品の開発遅れにも繋がり，事業の計画に影響を及ぼす．

レビューやテストの強化，プロセス改善といった欠陥の流出防止活動が多くの企業，組織で実施されている．しかし，限られた開発期間の中では内在しているすべての欠陥を除去することは難しく，欠陥は市場に流出しており[1]，流出防止活動が十分な成果を上げて

いると言い難い．

そこで我々研究チームは欠陥そのものに着目し，欠陥流出を真に防止できる方法を探った．

本研究では，処理ごとに抽出した欠陥の特徴から対策を導くことで欠陥の流出を防止する，「欠陥モデルに基づいたソフトウェア欠陥流出防止アプローチ」を提案する．

この提案により次の貢献が期待できる．

- これまでレビューやテストで発見しにくかった欠陥をより上流の工程で検出することができるようになる
- 欠陥の特性を捉え，対策できるとともに，次のプロジェクトへの教訓として欠陥流出防止観点リストを残すことができる

以降，まず 2 章で解決すべき課題を示し，3 章で解決策検討の参考とした関連研究を示す．4 章では「ソフトウェア欠陥流出防止アプローチ」について説明し，5 章においてその有用性を実験により示す．最後に 6 章でまとめと今後の展望を示す．

2. 解決すべき課題

我々研究チームが解決したい課題は欠陥の流出である．

流出防止の取組みを実施しているにもかかわらず，欠陥を市場に流出してしまう．さらにアンケートの結果，図 1 に示すとおり，類似した欠陥があると感じている人が多い．

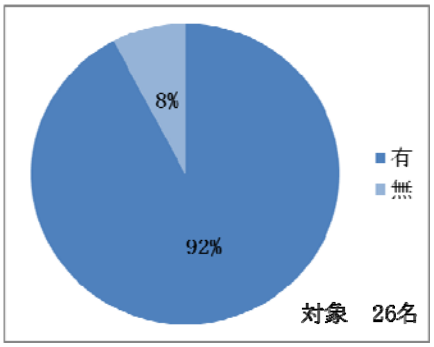

図 1　類似欠陥検出の経験

そこで我々研究チームは，異なるドメインやプロダクトにおいても共通的に実装される初期化処理，同期処理といった処理（以降，基盤処理という）に目を向けた．

図 2 に示すとおり，ソフトウェアのアプリケーションをシステム／装置に特化した「業務個別処理」と「基盤処理」に分け，検討した．

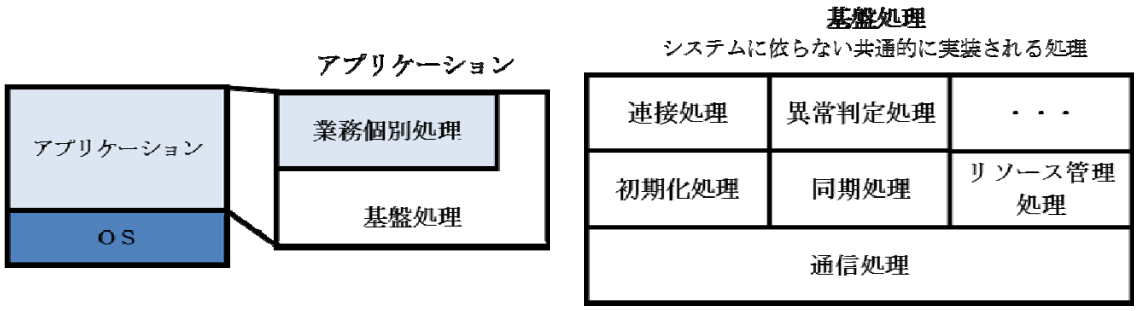

図 2　ソフトウェアの構造

「業務個別処理」と「基盤処理」で区分した欠陥の検出状況では，図 3 に示すとおり，7 つの対象プロジェクトにおいて約 50%の欠陥が「基盤処理」に混入されていた（詳細は付録 1 参照）．

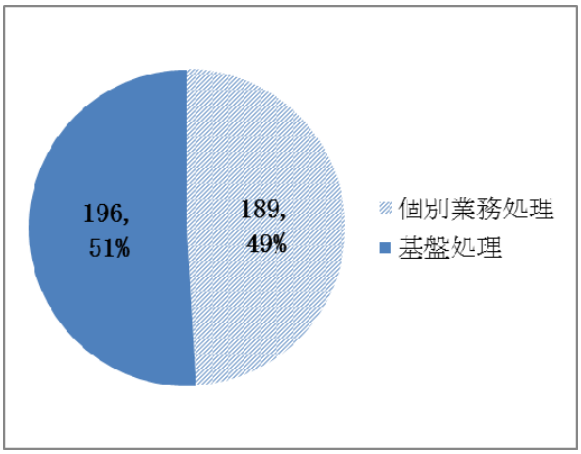

図３　処理で区分した欠陥検出割合

「基盤処理」の欠陥を流出防止することで「個別業務処理」の開発に注力でき，品質向上に繋げることができる．

本研究では「基盤処理」と欠陥そのものに着目し，その特性を捉えることが欠陥の流出防止の鍵になると仮定した．

3.　関連研究

3.1　欠陥モデリング

ソフトウェアテストシンポジウム 2013 東京の「不具合情報の活用」セッションで発表された「過失に着目した欠陥のモデリング」において，欠陥モデリングが定義されている．本研究では，この欠陥モデリングの定義を参考に欠陥について検討を実施した．Project Fabre[2]の欠陥モデリングの定義では，5 項目の要素（誘発因子，過失因子，増幅因子，欠陥，表出現象）に分け欠陥を表現している．この中で我々研究チームは，これらの要素を取り上げ基盤処理における欠陥を分析した（各要素の定義は付録 2 参照）．

- 誘発因子(Induction Trigger)
- 過失因子(Negligence Factor)
- 増幅因子(Amplifier)
- 欠陥(Defect)
- 表出現象(Incident)

3.2　3H

3H とは，昔から言われている安全作業の標語であり，「3H 作業とは人間が作業を行う際に，ミスや失敗を起こしやすい状況を簡潔にまとめた標語」として定義される[3]．次の三種類の作業または状況を指し，事前に 3H の視点で課題を気づき，問題が発生しないよう確認しながら仕事を遂行する未然防止技術である．

- 初めて(はじめて，Hajimete)
- 変更(へんこう，Henkou)
- 久しぶり(ひさしぶり，Hisashibiri)

4.　提案

我々研究チームは，欠陥流出防止アプローチを提案する．このアプローチを DBDP(Defect Based Differ Prevention) アプローチと名付ける．

本提案は，基盤処理に存在する欠陥，欠陥を産み出す過失及び欠陥の過失のトリガとなる誘発因子を欠陥の特徴と捉え，その特徴の共通性から欠陥流出を防止方法である．

アプローチは次の手順を追う．
① 処理の分類
② 欠陥モデルの作成
③ 欠陥の特徴の抽出
④ プロジェクトへの適用
基盤処理ごとに欠陥に着目し，人の過失を誘発するトリガを捉えることでプロジェクトのみならず横断的に組織の欠陥流出を防止する狙いがある．

4.1 処理の分類

企業・組織の不具合情報から欠陥そのものに着目し欠陥表出の対象となったアプリケーションを「業務個別処理」と「基盤処理」に区分する．
「基盤処理」とはシステムに寄らず共通的に実装される処理であるが，組織・ドメインにより細部区分が違う場合がある．ここでは，通常管理している処理名で区分けして構わない．
欠陥が多く混入されていた基盤処理を中心に次の欠陥モデルの作成を実施していくことで，効果的な欠陥流出防止ができる．

4.2 欠陥モデルの作成

各基盤処理に存在している欠陥の混入・流出の原因分析を行い，Project Fabre[2]で定義されている要素を用いて欠陥モデルを作成する（詳細は付録 2 参照）．
各要素間の関係性が理解できるため，モデル図を表記するとよい．
欠陥及び過失因子から基盤処理の細部区分を整理統合することで対象とした基盤処理に関わる欠陥の流出防止に繋げられる．

4.3 欠陥の特徴の抽出

欠陥を基盤処理ごとにモデル化し，過失因子及び誘発因子から欠陥の特徴を捉える．特徴抽出の際，人が過失を起こしやすい観点として誘発因子を次の項目で整理する．

- 対象の基盤処理において欠陥を流出に繋がった設計項目(What)
- 過失を誘発させる要因と管理しておく項目
  - 4H（初めて，変更，ひとりで，急いで（Hurry））
  - 1R（流用）
  - 3C（条件（Condition），複雑（Complexity），コミュニケーション（Communication））

4H は 3H を参考に，また 1R，3C は独自に定義した．
基盤処理ごとに共通に洗い出された因子を欠陥流出防止観点として纏める．このうち、「急いで」については時間的な要因であり確認自体が省かれる可能性がある．このため，プロジェクトマネージャや品質保証担当者等，設計当事者ではないメンバが状態確認する方がよい．

4.4 プロジェクトへの適用方法

基盤処理名称に対応する処理について，欠陥流出防止リストを用い設計前段階に押さえるべきポイントを確認する．レビュー時に漏れがないか，テスト後の検査等にも活用できる．個人の思い込みが入らないよう，グループで実施することで誘発要因，過失要因として挙がりがちであるコミュニケーション不足も解消される．
できるだけ上流の開発工程において誘発因子の発生を防止・抑制することが欠陥流出防止のポイントになる．

5. 実験

4 章の提案の妥当性を検証するため，実験にて次のことを確認した．

RQ1. より上流の開発工程で欠陥を発見できること
RQ2. 設計者以外（品質保証担当者等）でも欠陥流出を防止できること
RQ3. 基盤処理に着目し共通の誘発因子を抽出することで欠陥流出を防止できること
RQ4. レビュー，テスト実施前に欠陥の傾向予測ができ，設計上の弱点がみえること

5.1 実験内容

流出欠陥の特徴から，実行中のソフトウェア開発プロジェクトにおいて，基盤処理レベルで欠陥を検出することができるかの検証を行う．

人が過失を起こしやすい観点を洗い出しプロジェクトに適用し，What，および 4H1R3C が誘発因子となっているかを確認する．

また，基盤処理ごとに抽出した押さえるべき誘発因子を用い欠陥の流出防止ができるかをアンケートで明らかにする．

5.2 実験結果

(1) 欠陥モデルの作成

実験のために収集した欠陥情報を基盤処理ごとに分類し，欠陥そのものに着目し基盤処理を整理した．表 1 に示すとおり初期化処理，同期処理，リソース管理処理に欠陥が多く，かつ複数プロジェクトに存在していた（詳細は付録 1 参照）．

表 1　実験対象とする基盤処理

番号	基盤処理（整理後）	総数	プロジェクト数	対象となる基盤処理
1	初期化処理	19	6	－
2	同期処理	35	4	タイミング同期処理・入力同期処理
3	リソース管理処理	14	4	メモリ確保処理・メモリ解放処理・ファイル読込処理（影響度大）・ファイル出力処理（影響度大）

実験では欠陥が多く存在した表 1 の 3 処理について欠陥モデルを作成し，人の過失を誘発した設計項目（What）と管理点（4H, 1R, 3C）を確認した．

(2) 欠陥の特徴の抽出

作成した欠陥モデルの誘発因子（設計項目（What）及び管理点（4H,1R,3C））から基盤処理ごとに欠陥流出防止観点を抽出した．抽出結果を表 2 に示す．

表 2　共通に抽出した欠陥流出防止観点リスト

基盤処理	欠陥流出防止観点	
	設計項目	管理点
初期化処理	・初期化条件 ・初期化タイミング ・初期値 ・初期化範囲（容量）	・ハードウェア，担当者等変更はないか ・初めて担当する人，器材ではないか ・流用することで条件，タイミング，初期化範囲等見直しの必要はないか ・初期化の条件はすべて洗い出されているか
同期処理	・タイミング ・対象処理 ・処理シーケンス	・ハードウェア，担当者等変更はないか ・初めて担当する人，器材はないか ・流用することでタイミング，同期対象処理等に問題はないか ・条件による同期，複雑な設計になっていないか
リソース管理処理	・メモリマップ（サイズ） ・アクセス（出力）要領 ・制約／条件	・ハードウェア，担当者等変更はないか ・流用することで対象リソースの範囲，条件等問題はないか ・条件による管理，複雑なリソース管理はしていないか

What＋4H1R3C の観点で共通項抽出したことにより，見落としがちな欠陥の特徴が誘発因子に表れ整理できた．また，特に技術的な表現ではなくとも設計者以外（品質保証担当者等）でも説明できる管理点として示すことができた．

(3)　プロジェクトへの適用

DBDP アプローチを適用し欠陥検出した結果を表 3 に示す．ここではコンポーネント統合テスト実施後に DBDP アプローチを適用し，基盤処理ごとの欠陥流出防止観点で仕様書を元に確認し欠陥を検出した．表 3①はその結果である．また，表 3①実施後，次工程であるシステムテストで検出した欠陥を表 3 の②に示した．

表 3　プロジェクトへの適用結果

	欠陥検出工程[4]	プロジェクト	内訳（件）		
			初期化処理	同期処理	リソース管理処理
①	コンポーネント統合テスト後	プロジェクト E	1	38	8
		プロジェクト X	2	5	1
②	システムテスト	プロジェクト E	0	1	0
		プロジェクト X	0	1	0

特にプロジェクト E については，付録 1 表 7 に示すとおりコンポーネント統合テストで同期処理，リソース管理処理に欠陥が多く検出していた．しかし，システムテスト前に DBDP アプローチを適用した確認により，さらに多くの欠陥を検出することができた．

表 4 では DBDP アプローチの適用により検出できた欠陥の混入工程を示す．

表 4　DBDP アプローチで検出した欠陥の混入工程別件数

指摘工程[5]	プロジェクト	見直し対象処理（件）		
		同期処理	リソース管理処理	初期化処理
ソフトウェア要求定義	プロジェクトE	18	0	0
	プロジェクトX	1	0	1
ソフトウェアアーキテクチャ設計	プロジェクトE	14	8	1
	プロジェクトX	4	1	1
ソフトウェア詳細設計	プロジェクトE	6	0	0
	プロジェクトX	0	0	0
件数計	プロジェクトE	38	8	1
	プロジェクトX	5	1	2

(4)　アンケートの結果

基盤処理に着目し，欠陥モデルから抽出した誘発因子を用い欠陥流出を防止する「DBDP アプローチ」について，アンケートを実施した．有効に活用できるが 75%を超え，設計者以外の活用についても「処理による」の回答を含めると 80%を超え活用の可能性が判断できる（詳細は付録 5 参照）．

5.3　実験結果に対する考察及び評価

我々研究チームの実験結果に対し，考察を次に示す．

RQ1. より上流工程で欠陥を発見できること

欠陥モデルから得られた欠陥流出防止観点に基づき，上流の開発工程の成果物をレビューした結果，表 4 のとおり，不備を指摘することができた．実験ではコンポーネント統合テスト後に DBDP アプローチを適用したが、より上流の開発工程で適用することで，より上流の開発工程での欠陥検出が可能であることを示せた．

RQ2. 設計者以外（品質保証担当者等）でも欠陥流出を防止できること

アンケートにおける「処理による」の回答には，技術的な観点では指摘が難しいとする意見が多かったが，表 2 に示す欠陥流出防止観点を示すことで問題の洗い出しに目を向けることができ，設計者以外でも指摘することが可能といえる．

また，付録 4 表 15 に示すとおり，設計者以外から上流工程で作成した設計書に対して指摘ができている．設計者以外でも欠陥流出の防止に貢献できることが示された．

RQ3. 基盤処理に着目し共通の誘発因子を抽出することで欠陥流出を防止できること

基盤処理に着目した DBDP アプローチに対し欠陥流出防止に有効活用できるという回答が 75%を超えている．実際にプロジェクト D においては，処理に絞り込み欠陥を確認したことによりシステムテスト前に検出することができた欠陥が多い．

RQ4. レビュー，テスト実施前に欠陥の傾向予測ができ，設計上の弱点がみえること

表 4 からは上流工程の設計に対する不備の分布が読み取れる．例えば，プロジェクト D においては，同期処理の不備の半分近くがソフトウェア要件仕様書に存在していた．このことから，実現しようとしているソフトウェア要件そのものの不備に起因する欠陥が検出されることが予想できる．

DBDP アプローチを用いることでレビュー，テスト実施前に設計上の弱点が見え，欠陥の傾向予測ができる．

5.4 妥当性への脅威

実験は我々研究チームのメンバが実施しており，経験や環境に対する影響を受けている可能性がある．

また，実験の対象は我々研究チームが所属する組織における一つの代表的な製品群に対し限定したものであり，これは外的妥当性への脅威である．ただし，さらに欠陥モデルを増やし，特定ドメインの影響を受けないよう汎化していき抽象度を上げることで適用できる．

今後は，実験で得られた結果をもとにモデル化にあたっての条件（モデルの抽象度，ドメインの拡大，管理すべき項目等）をさらに検討し影響の範囲を確認したい．

6. おわりに

6.1 まとめ

従来の欠陥流出防止策は，表出現象の発生を防止することに注力し欠陥の本質を捉えることなく表出現象に対する処置を実施していた．

本研究では，欠陥そのものに着目し，基盤処理ごとにその過失を引き起こす誘発因子が抑えられる「DBDP（Defect Based Differ Prevention）アプローチ（ソフトウェア欠陥流出防止アプローチ)」を確立し，一定条件下においてその有効性を確認した．

次の 3 点がアプローチのポイントとなる．

- 基盤処理，すなわち，システムに依らない共通的な処理に分類することで欠陥流出を防止することができる
- 欠陥モデルを作成することにより，欠陥そのものを捉え過失を誘発する要因を識別し，より上流の開発工程での欠陥流出防止に繋げることができる
- 設計すべき項目（What）と 4H（初めて，変更，ひとりで，急いで（Hurry)），1R（流用),3C(条件(Condition),複雑(Complexity),コミュニケーション(Communication)) を欠陥誘発の要因とすることで，設計者以外でも欠陥流出を防止することができる

6.2 今後の展望

今後，今回対象とした基盤処理以外の欠陥モデルを作成し誘発因子の抽出を行うとともに欠陥モデルの汎化を進めることで，プロジェクトへの適用実験をさらに進めていく．プロジェクトへの適用を広げることで，組織の欠陥流出防止に貢献できる欠陥流出防止リストを残していく．また，欠陥流出防止観点をさらに整理し，欠陥モデルを汎化することで，ドメイン間でも共有可能となるかを検証し，さらなる横展開を図るとともに，あらゆる処理に適用できるアプローチとしていきたい．

参考文献

[1] 太田忠雄ほか,“高信頼化ソフトウェアのための開発手法ガイドブック”,IPA(2010-9)

[2] 細川宣啓，西康晴，嬉野綾，野中誠，原佑貴子,“過失に着目した欠陥のモデリング －バグ分析はなぜうまくいかないのか？”－ ソフトウェアテストシンポジウム 2013 東京 セッション C4 不具合情報の活用

[3] SDC 検証審査協会［編］,“3H　初めて、変更、久しぶりの理論と実践”，日刊工業新聞社（2011-12)

[4] ISTQB(翻訳:JSTQB)，ISTQB テスト技術者資格制度 Foundation Level シラバス日本語版 Version2011J02

[5] 【改訂版】組込みソフトウェア向け開発プロセスガイド，IPA(2007-11)，翔泳社

付録

付録１：処理で区分した欠陥検出割合

各工程で検出した欠陥について「個別業務処理」及び「基盤処理」に分類し，その検出割合を確認した．結果を表５に示す．７つの対象プロジェクトにおいて，いずれも約半数の欠陥が基盤処理に混入されていたことが分かった．

表５　処理で区分した欠陥検出割合

	計	個別業務処理				基盤処理				件数割合		工程[3]
		影響度				影響度				個別業務処理	基盤処理	
		計	大	中	小	計	大	中	小			
プロジェクトA										47%	**53%**	出荷後
プロジェクトB										43%	**57%**	システムテスト
プロジェクトC										45%	**55%**	システムテスト
プロジェクトD										38%	**62%**	システムテスト
プロジェクトE										54%	46%	コンポーネント統合テスト
プロジェクトF										48%	**52%**	コンポーネント統合テスト
プロジェクトG										52%	48%	コンポーネント統合テスト

また，表６に示すとおり，影響度の重みで比較しても基盤処理の欠陥流出を防止することは効果があるといえる．

表６　欠陥の影響度で見た処理ごとの割合

	影響度			
	度合の定量化(*)		割合	
	個別業務処理	基盤処理	個別業務処理	基盤処理
プロジェクトA			48%	**52%**
プロジェクトB			40%	**60%**
プロジェクトC			47%	**53%**
プロジェクトD			37%	**63%**
プロジェクトE			49%	51%
プロジェクトF			53%	47%
プロジェクトG			51%	49%
	(*)大：10，中：5，小：2で計算			

調査対象とした７つのプロジェクトにおける基盤処理の欠陥混入状況を表７に示す.

プロジェクトが対象とするシステム／装置により，欠陥が混入されている基盤処理も異なるが，複数プロジェクトで検出される欠陥も少なくない。

表７　基盤処理ごとの欠陥混入状況

No.	基盤処理	欠陥混入		プロジェクト(PJ)						
				A	B	C	D	E	F	G
		総計	PJ数	計	計	計	計	計	計	計
1	初期化処理	19	6							
2	終了処理	3	1							
3	キャンセル処理	1	1							
4	異常判定・通知処理	5	2							
5	タイムアウト監視処理	4	1							
6	連接処理	6	3							
7	リモート接続処理	1	1							
8	画面表示処理	35	3							
9	画面終了処理	3	2							
10	表示制御処理	9	1							
11	描画処理	5	3							
12	メッセージ表示処理	6	3							
13	メッセージ送受信処理	9	3							
14	パラメータ設定処理	1	1							
15	データ入力処理	4	2							
16	データ設定処理	1	1							
17	データ書込処理	2	1							
18	データ削除処理	2	1							
19	ファイル読込処理	3	3							
20	ファイル出力処理	7	4							
21	データベース登録処理	4	2							
22	データベース検索処理	1	1							
23	データアクセス管理処理	1	1							
24	メモリ確保処理	10	2							
25	メモリ解放処理	2	1							
26	入力同期処理	3	1							
27	タイミング同期処理	32	4							
28	排他処理	2	2							
29	日付管理処理	1	1							
30	値取得処理	1	1							
31	入力値範囲チェック処理	5	3							
32	単位変換処理	1	1							
33	単位表示処理	2	2							
34	型変換処理	1	1							
35	ソート処理	1	1							
36	通信処理	3	2							
計		196	―							

付録２：Project Fabre の欠陥モデリング

Project Fabre の欠陥モデリングでは次の要素を定義している．

- 誘発因子(Induction Trigger)
  成果物の中に含まれる，人間の思考の誤りを誘発する“トリガ”となる要素のこと．誘発因子が存在すれば，開発者能力・経験・技術力と関係なく過失が引き起こされやすくなる．
- 過失因子(Negligence Factor)
  人間の思考や判断の誤りそのもののこと．欠陥は過失因子の集合（=連続）として生み出される．
- 増幅因子(Amplifier)
  過失の連鎖を助長し，欠陥の混入確率を増幅させる要素．多くは定量的に測定可能である．外乱・環境特性ともいう．
- 欠陥(Defect)
  成果物に含まれた，人間の思考の過ちが具現・表出化したもの．不具合・障害等の「表出現象」を発生させる．
- 表出現象(Incident)
  欠陥によって引き起こされる実害を伴う不具合・障害．多くは定量的に測定／加算可能．

図４の欠陥モデルを使用し欠陥を表記し活用することで，欠陥を生み出した過失及び過失を誘発したトリガの各要素が整理できる．

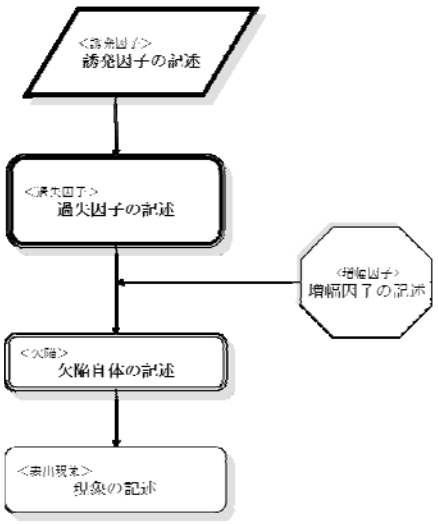

図４　Project Fabre　欠陥モデリング表記方法

欠陥モデル作成の手順と作成の際のポイントを示す．

① 不具合事象を表出事象ノードとして配置する．
② 不具合事象を引き起こした欠陥を欠陥ノードとして配置し，表出事象ノードと接続する．
③ 過失因子ノードを配置する．欠陥流出の要因をより明確にするため，１つの欠陥モデルにつき過失因子は１つのみとする．複数の過失因子が考えられる場合は複数の欠陥モデルに分ける．
④ 過失を招く要因となった誘発因子ノードを配置する．レビューやテストは増幅因子であり，過失因子ではない点に注意する．

付録 3：実験で作成した欠陥モデリング

付録 1 表 7 の集計結果から複数プロジェクトで欠陥が検出された基盤処理を中心に欠陥モデルの欠陥及び過失因子から実験対象として次の基盤処理を抽出した。

- 初期化処理
- 同期処理（タイミング同期処理，入力同期処理）
- リソース管理処理（メモリ確保処理，メモリ解放処理，ファイル読込処理(*)，ファイル出力処理(*)）(*)資源消費に関わる欠陥を対象とする

初期化処理における欠陥モデルを表 8 に，誘発因子の整理を表 9 に示す．また，同期処理における欠陥モデルを表 10，誘発因子の整理を表 11 に、リソース管理処理における欠陥モデルを表 12，誘発因子の整理を表 13 に示す．

欠陥モデルは表出現象に対し、欠陥そのものに着目し欠陥、過失因子及び誘発因子を展開していくことが誘発因子の整理に繋がる。

表 8　初期化処理における欠陥モデル

番号	表出現象	欠陥	過失因子	誘発因子		
1	データ自動更新時に色が変わった	自動更新時の初期化のタイミングが違っていた	初期化が他の機能に連動していることに気がつかなかった	関連する機能の条件が整理できていなかった		
2	通信の確立が遅れた					
3	２回目の表示データの値が違った					
4	表示を拡大すると違うモードで表示された					
5	初期値が設定されていない					
6	画面を切り替えてもデータが残っている					
7	起動時のモードが違う					
8	前回のデータが残る					
9	位置情報が更新されない					
10	データ項目にゴミが入っている					
11	削除したデータが表示された					
12	初期値が設計書で記述されている値と違う					
13	変更したデータが反映されない					
14	２回目に送ったデータが反映されない					
15	操作するとデータがクリアされてしまう					
16	起動時の初期値が違がっている					
17	通信異常が発生した					

表 9　初期化処理における誘発因子の整理

番号	設計項目(What)	着目点							
		4H				R	3C		
	対象	初めて	変更	ひとりで	急いで（Hurry）	流用	条件（Condition）	複雑（Complexity）	コミュニケーション
1	・タイミング						○		
2	・容量		○			○	○		
3	・初期状態の条件 ・設定内容								
4	・初期設定値		○		○				
5	・初期設定値	○							
6	・初期設定値	○							
7	・初期状態の条件 ・設定内容				○		○		
8	・初期状態の条件						○		○
9	・初期状態の条件				○				
10	・初期状態の条件						○		○
11	・タイミング	○					○		○
12	・設定内容	○				○			
13	・タイミング								○
14	・初期状態の条件						○		
15	・タイミング								○
16	・初期設定値	○							○
17	・初期化条件		○						○
計		5	3	0	3	2	7	0	7

共通性を見出し観点として纏めるが，１事例でも要因があれば欠陥流出の可能性があるとしてリストに盛り込んだ.「コミュニケーション」については，要因として多いが、DBDP アプローチを適用することでコミュニケーション問題は解決するとしリストからは外している.

表 10　同期処理における欠陥モデル

番号	表出現象	欠陥	過失因子	誘発因子		
1	通信異常が発生した	入力信号が同期せず、信号生成に失敗した	入力信号が非同期になると思わなかった	同期についての仕様記述がなく暗黙だった	プログラムを流用しており、同期するものと思っていた	ケーブル長が変わった
2	プログラムが停止した					
3	表示のちらつき、色ずれが発生した					
4	計算結果が違う					
5	通信異常が発生した					
6	画面を切り替えてもデータが残っている切り替えると通信異常になる					
7	データ登録中止で処理が停止する					
8	通信ができなくなった					
9	画面に何も表示されない					
10	電圧異常でシステムが動作しない					
11	電圧異常でシステムが動作しない					

表 11　同期処理における誘発因子の整理

番号	設計項目(What)	着目点							
		4H				R	3C		
	対象	初めて	変更	ひとりで	急いで（Hurry）	流用	条件（Condition）	複雑（Complexity）	コミュニケーション
1	・入力タイミング ・タイミング遅延規定		○			○			
2	・同期対象処理 ・タイミング	○			○				○
3	・入力タイミング ・タイミング遅延規定		○			○			
4	・同期対象処理 ・タイミング	○							○
5	・同期対象処理 ・タイミング		○			○			
6	・同期対象処理 ・タイミング								
7	・同期対象処理 ・条件別同期処理シーケンス ・タイミング						○		
8	・タイミング		○			○		○	
9	・タイミング （表示バッファアクセス）		○			○		○	○
10	・タイミング （電圧異常検出タイミング）		○				○		
11	・タイミング （正常の通知のタイミング）		○				○		
計		2	7	0	1	5	3	2	3

表 12　リソース管理処理における欠陥モデル

番号	表出現象	欠陥	過失因子	誘発因子		
1	装置が停止した	出力の上限を設定していなかった	資源枯渇するとは思わなかった	出力方法、出力サイズの調整がない	運用要件についての情報がない	機能要求でないので重視していない
2	正しい動作でない					
3	装置が停止した					
4	システムが停止した					
5	処理が停止した					
6	機器の動作がおかしい					
7	動作が重くなる					
8	処理が停止した					
9	予定外の場所でリセットされる					

表 13　リソース管理処理における誘発因子の整理

番号	設計項目(What)	着目点							
		4H				R	3C		
	対象	初めて	変更	ひとりで	急いで（Hurry）	流用	条件（Condition）	複雑（Complexity）	コミュニケーション
1	・出力方法 ・出力サイズ								○
2	・メモリマップ		○			○			
3	・出力方法 ・出力サイズ								○
4	・メモリマップ						○		○
5	・メモリ確保サイズ		○			○			
6	・メモリ解放の責務分担 ・フレームワーク使用ガイド								○
7	・メモリ解放の条件						○		○
8	・発生条件						○	○	
9	・割込みベクタマップ		○			○	○		
計		0	3	0	0	3	4	1	5

付録 4：欠陥モデル利用による適用確認

付録 3 で抽出した各基盤処理の欠陥流出防止観点をプロジェクトで適用した．結果を表 14 に示す．また，表 15 に適用時の指摘者を示す．

表 14　DBDP アプローチによるプロジェクトへの適用結果

番号	プロジェクト	基盤処理	適用確認結果	具体的内容
1	プロジェクトD	同期処理		
2	プロジェクトD	同期処理		
3	プロジェクトD	同期処理 （入力同期）		
4	プロジェクトD	同期処理		
5	プロジェクトD	リソース管理処理		
6	プロジェクトD	リソース管理処理		
7	プロジェクトD	リソース管理		
8	プロジェクトD	初期化処理		
9	プロジェクトX	同期処理		
10	プロジェクトX	リソース管理処理		
11	プロジェクトX	初期化処理 （マイコン端子）		
12	プロジェクトY	リソース管理処理 （ファイル出力）		

表 15　DBDP アプローチ適用時の基盤処理ごとの指摘者

番号	プロジェクト	基盤処理	指摘者
1	プロジェクトD	同期処理	顧客
2	プロジェクトD	同期処理	ソフトウェア設計者
3	プロジェクトD	同期処理（入力同期）	ソフトウェア設計者
4	プロジェクトD	同期処理	ソフトウェア設計者
5	プロジェクトD	リソース管理処理	テスト技術者
6	プロジェクトD	リソース管理処理	ソフトウェア設計者
7	プロジェクトD	リソース管理	テスト技術者
8	プロジェクトD	初期化処理	テスト技術者
9	プロジェクトX	同期処理	ソフトウェア設計者
10	プロジェクトX	リソース管理処理	ソフトウェア設計者
11	プロジェクトX	初期化処理（マイコン端子）	ハードウェア設計者
12	プロジェクトY	リソース管理処理（ファイル出力）	品質保証担当者

付録 5：DBDP アプローチに関するアンケート

基盤処理に着目し欠陥流出防止の観点を洗い出すことについて，アンケートを実施した．我々研究チームの所属する組織内外の,役割の異なる 27 名から客観的な意見を得ることができた．図 5 は回答者の役割の分布を示したものである．アンケート結果を表 16 に示す．

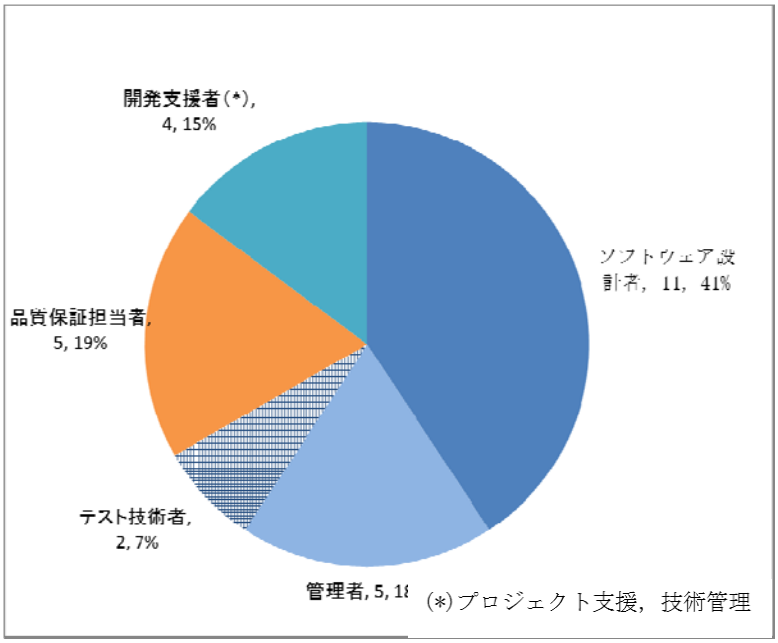

図 5　アンケート対象者分布

表 16　DBDP アプローチに関するアンケート結果

	はい	処理による	いいえ	分からない	計
欠陥流出防止に有効活用できるか？	21	4	1	1	27
	78%	15%	4%	4%	－
設計者以外（品質保証担当者等）でも活用できるか？	9	14	3	1	27
	33%	52%	11%	4%	－

「処理による」,「いいえ」と回答しているメンバから，次のとおり，技術的な観点で内容が分からないと指摘は難しいという意見が出ている．ただし，示唆を促す観点では有効であるとの意見をもらった．

- 一般的に危なそうというのは理解できるが，フレームワーク構造を知らない場合，指摘は難しい
- こういうときはどうした？ということは問える
- 特に気をつけた方が良い点として意識することはできるが、自分が指摘できるかどうかはわからない