Project Fabre presents

欠陥マスター情報構築ワークショップ

―「悪さの知識」伝承によるバグ予防の為のバグ情報マスター化実践―

Nobuhiro Hosokawa, Yasuharu Nishi,

Aya Ureshino, Makoto Nonaka, Yukiko Hara

Project Fabre presents ワークショップ コンテンツ

1. Project Fabre メンバー

細川 宣啓（日本IBM）

Project Fabreリーダー。
QE実務専門家トップランカー。

CARVIN@jp.ibm.com

野中誠（東洋大学）

メトリクス王子。
Fabreの良心。

nonaka-m@toyo.jp

西 康晴（電気通信大学）

日本（世界！？）ソフトウェア
テスト界のキーマン。

Yasuharu.Nishi@uec.ac.jp

原 佑貴子（日本IBM）

レビュー専門家。仕様書と
ソースにレビュービームを発射。

HARAYUKI@jp.ibm.com

嬉野 綾（ワークスアプリケーションズ「働きがいのある会社 第2位」）

大手企業向けERPパッケージ COMPANY シリーズ QE歴12年。

製品コンセプト:「ノンカスタマイズ」「永続的無償バージョンアップ」

ureshino@worksap.co.jp

2. 欠陥学(Defectology)とは？

欠陥学（Defectology）とは、欠陥の混入確認・存在確認や、各種ソフトウェア開発と検査を助けるための、欠陥情報と標本の提供を目的とする研究です。

欠陥学
（Defectology）

1） 欠陥標本の作成とその技術（固定・抽象化・分類・保存）

2） 欠陥標本を利用した検査の実践

3） 欠陥標本を利用したバグ予測・バグ予防の実践

4） 欠陥標本を利用した技術者の育成と「悪さの知識」伝承

3. 欠陥とは何か？：欠陥の特性と仮説

欠陥の定義は諸説存在しますが、ここでは以下のような特性と
そこから導出される仮説を軸に説明します。

特性1）

欠陥は様々な「属性」を持つ

仮説1：属性のない欠陥はない

仮説2：欠陥は属性のコンポジション

仮説3：欠陥は分類可能である

仮説4：欠陥は伝達・移転可能である

特性2）

欠陥は自然発生しない

仮説5：回避・対応は可能である

仮説6：混入阻止は可能である

欠陥

（Defect）

特性3）

欠陥のユーザーが存在する

仮説7：負のユーザー（困る人）がいる

仮説8：正のユーザーがいる

特性4）

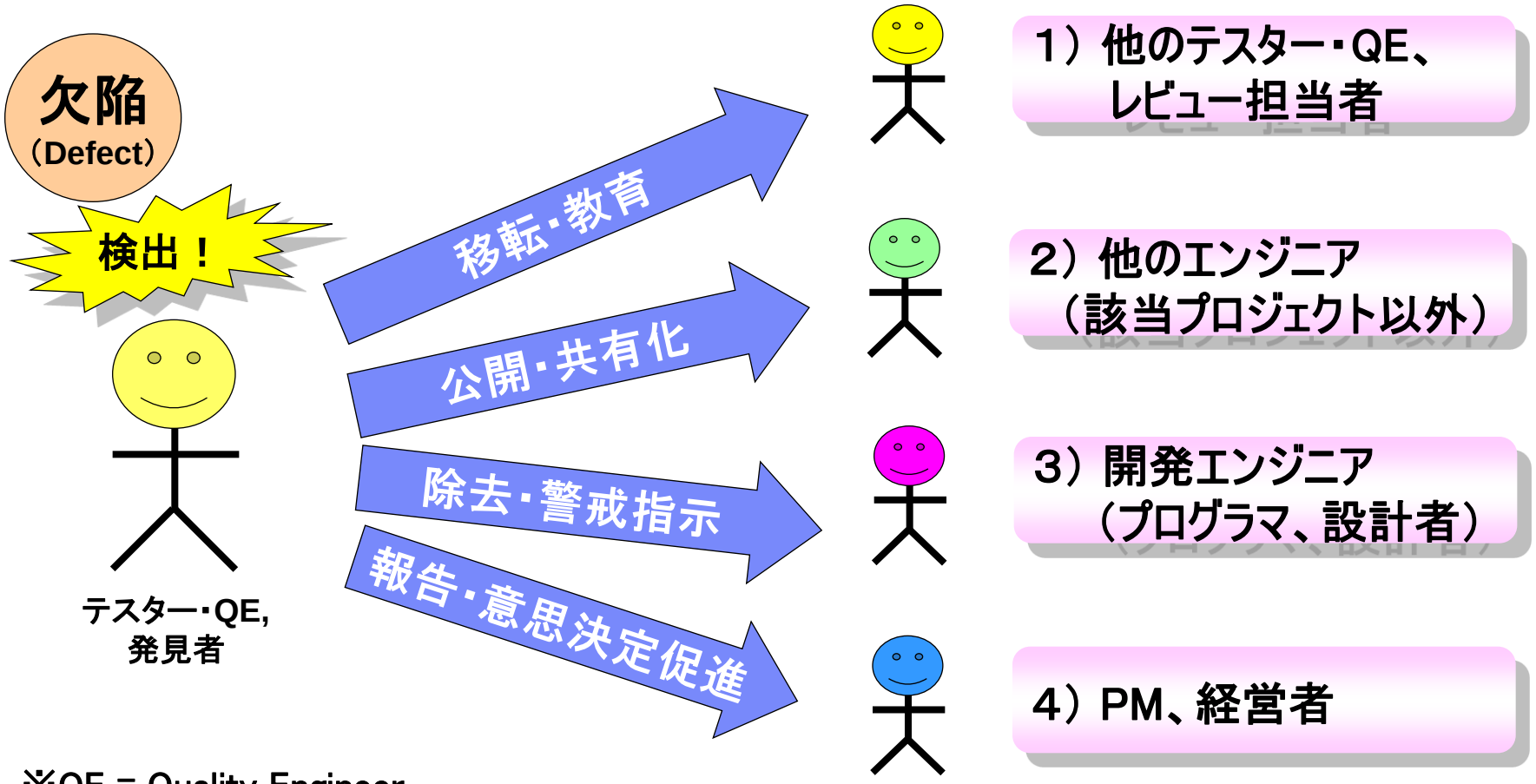
欠陥は動的である

仮説9：欠陥は状態遷移する

仮説10：欠陥にはトレンドや流行がある

4. 欠陥の表現と伝達・伝承 ：欠陥情報の「正のユーザー」＝ダレ得？

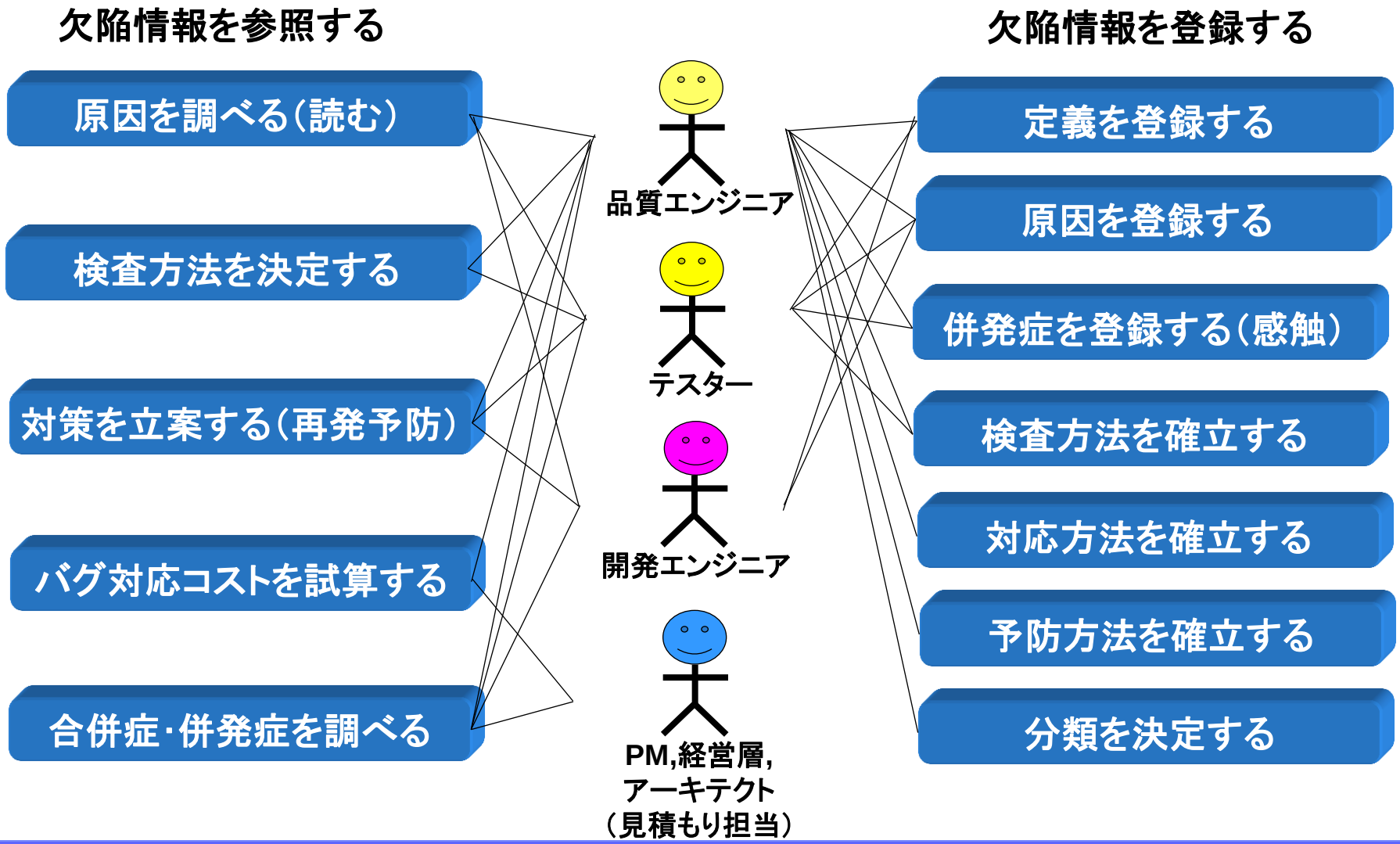
欠陥情報は、誰かに伝達・伝承して初めて意味を持ちます。様々な
「欠陥のユーザー」は、発見者が伝達した欠陥情報から利得を得ます。



※QE = Quality Engineer

4. 誰得？（誰にとってメリットがあるのか）

この研究の成果は、誰にとって、どんなメリットになるでしょうか？



4. 誰得？（誰にとってメリットがあるのか）

■ プロジェクトマネージャー（開発中）

②兆候情報＋③検出情報＝品質管理上の必要アクション指示

③検出情報＋④除去・予防方法＝品質管理上の予防アクション指示

■ プロジェクトマネージャー・経営層（プロジェクト計画時）

②兆候情報＋④除去・予防方法

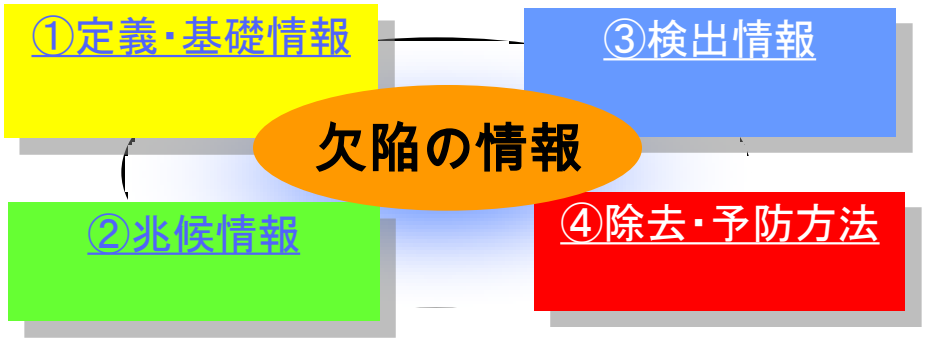
＝当該プロジェクトのリスク判定＋プロジェクト計画

①定義情報＋④除去・予防方法＝品質管理計画

■ プログラマ・設計者

①定義情報＋③検出情報

＝自己レビュー・自浄

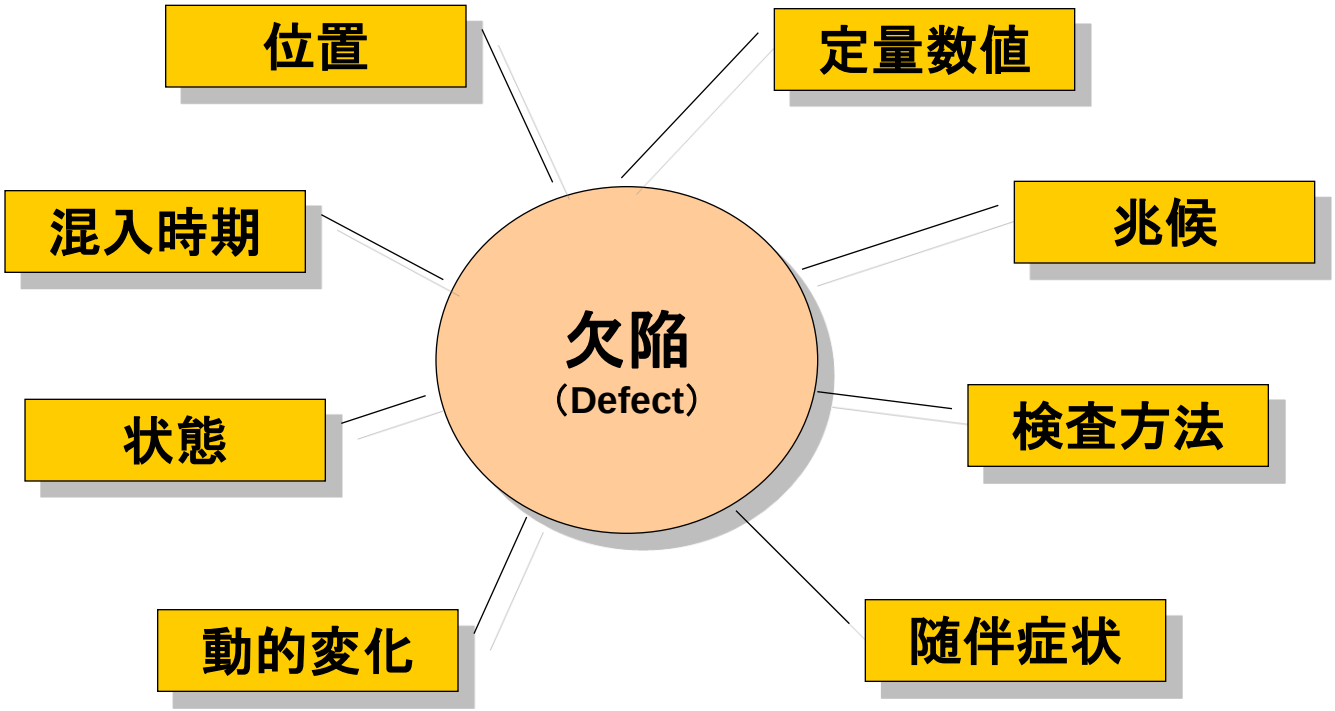


■ QE／テスター／レビュー担当者

①～④の全部！！どうやって？

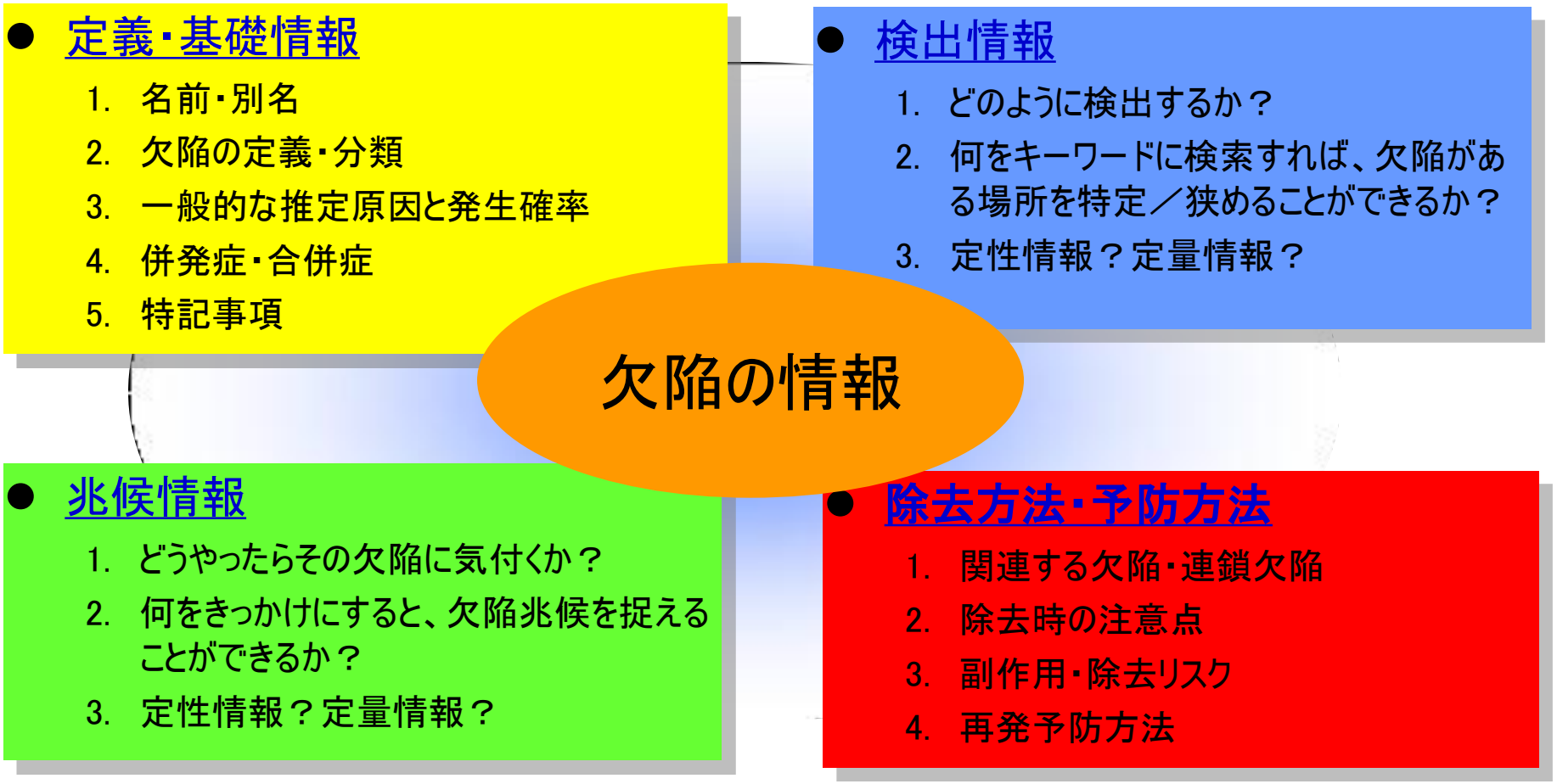
5. 欠陥はどんな属性の塊か？ ：属性集合図(Attribute Set Diagram)

欠陥は様々な属性を持ちます。換言すれば、欠陥情報には欠陥名称以外の属性の情報がなく、欠陥＝「属性のComposition」と捉えることができます。



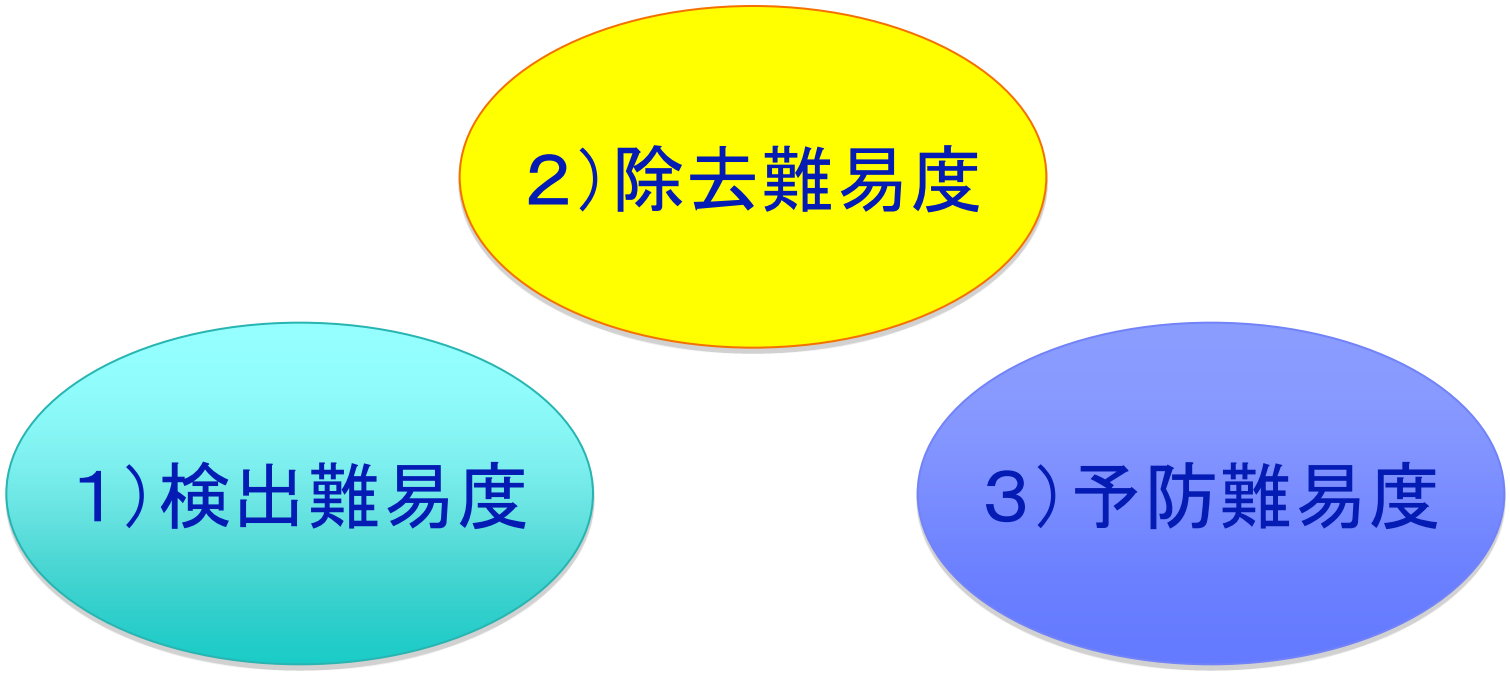
6. 欠陥の情報分類: #1「病理学」風の整理

医学の視点で、定義と用途を重視した永続的・長期的な情報整理を行います。



6. 欠陥の情報分類: #2「品質管理」専門家の整理

品質管理の専門家の立場からは、以下の3つの観点での欠陥分類が必要です。



検出難易度---開発者またはプロジェクト内部メンバーによる自己レビューで検出しにくい欠陥

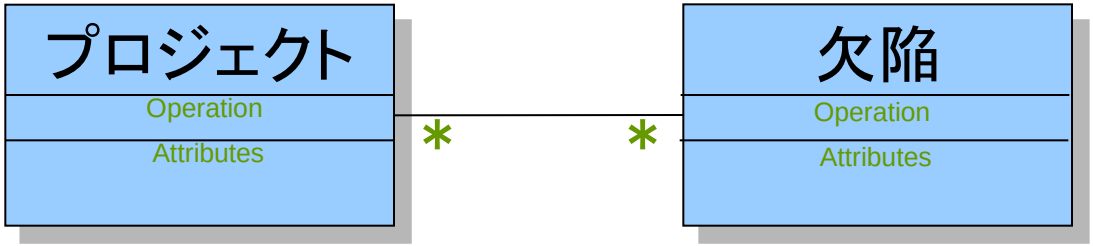
除去難易度---欠陥の存在は確認できたが、本番環境から除去しにくく、除去工数のかかる欠陥

予防難易度---他の困難度に関係なく頻発し、根本原因対策が難しいため、再発予防の難しい欠陥

7. 従来のバグ票のままではダメな理由

1）プロジェクトで発生する個々のバグ票は、プロジェクト固有条件を含むため欠陥の「インスタンス」である
2）現場プロジェクトでいつでも観察・参照可能な欠陥情報として、欠陥マスターの存在が必要不可欠

- **プロジェクトと欠陥の関係を整理すると次のような関係。**
  1. プロジェクトには複数の欠陥が発生する
  2. 一つの欠陥は複数のプロジェクトで検出される



- **上記多対多の関係は、1対1関係の「関連エンティティ」をおくと整理できる**
  1. プロジェクトと欠陥の間に「プロジェクト欠陥」エンティティを設定。
  2. 各1対多の関係でリレーションを整理

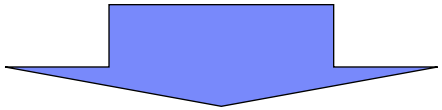
- プロジェクト欠陥（①）をどれほど多量に集めても、観察・参照・利用可能できない
- 欲しいのは②の「欠陥マスター」情報。固定化・抽象化・整理されなくては参照できない

7. 従来のバグ票のままではダメな理由　:欠陥マスターDBの必要性

従来、開発中・改変時を問わず「バグ報告書」として欠陥情報が記録されてきました。今後は、欠陥をインスタンス情報から転化させる必要があります。

従来の欠陥記録 = バグ報告書

ソフトウェアのライフサイクルにおける特定の品質特性の一静的断面をあらわす情報の塊



将来の欠陥記録 = 欠陥のマスター情報記録

不連続に変化するソフトウェア品質において、期待と現状のGAPが大きくなった時に、欠陥を固定化・抽象化・整理して保存し、関わる人に改善・受容を促すための情報群

参考） 医学との対比

医学には．．．

✓医学では、稀な症例や現場の事例を学会等を通じて共有する仕組みがある

✓現場の実例から抽象化して「病気」だけを定義・管理・維持する学会・権威機関がある。そういう医学分野がある

✓現場医師は最新動向や検査・対処方法を継続的に学ぶ習慣が付いている

IT業界には．．．

- ✓ **稀なバグや被害の大きな欠陥を業界全体で共有する仕組みがない**
- ✓ **そもそもバグを研究する「バグ屋」という分野の人材も・技術分野も存在しない**
- ✓ **現場は「除去」さえできればよいため、欠陥の情報の獲得を行わない**

PubMed

PubMed comprises more than 20 million citations for biomedical literature from MEDLINE, life science journals, and online books. Citations may include links to full-text content from PubMed Central and publisher web sites.

医学分野で世界最大の文献データベース。1966年からNLM（米国国立医学図書館）でデータ収集が始まり，現在毎月約3万件の文献が新たに追加される。現在では，米国を中心に約70ヵ国から，900万件を超える文献が収録される。

補足資料：ソフトウェア欠陥についての学術研究

欠陥の「分類」に関する学術研究はこれまでに行われている

V.R. Basili and R.W. Selby, "Comparing the Effectiveness of Software Testing Strategies," *IEEE Trans. Software Eng.*, vol. 13, no. 12, pp. 1278-1296, Dec. 1987.

R. Chillarege, I.S. Bhandari, J.K. Chaar, M.J. Halliday, D.S. Moebus, B.K. Ray, and M. Wong, "Orthogonal Defect Classification—A Concept for In-Process Measurements," *IEEE Trans. Software Eng.*, vol. 18, no. 11, pp. 943-956, Nov. 1992.

W.S. umphrey, *A Discipline for Software Engineering*. Addison-Wesley Longman, 1995.

B. Beizer, *Software Testing Techniques*. Van Nostrand Reinhold, 1990.

IEEE, *IEEE Standard Classification for Software Anomalies*, IEEE Std. 1044-1993, 1994.

Mäntylä, M. V. and Lassenius, C., "What Types of Defects Are Really Discovered in Code Reviews?," *IEEE Trans. Softw. Eng.*, vol. 35, no. 3, pp. 430-448, 2009.

欠陥「そのもの」や「マスターDB」については、
学術研究において十分に示されてきていない。

補足資料：欠陥の「分類」 ～最近の研究から

機能性欠陥
(functional defects)

大規模欠陥	機能性の欠如、大幅な追加・修正が必要
サポート	構成管理システムの問題やライブラリの欠陥
タイミング	マルチスレッド環境で生じる問題
チェック	妥当性確認ミス、不正な値の検出ミス
リソース	データ、変数、リソース初期化、解放などの誤り
ロジック	比較演算子、制御フロー、計算などの誤り
インタフェース	ライブラリ、デバイス、DB、OS等とのｲﾝﾀﾌｪｰｽ不整合

発展性欠陥
(evolvability defects)

構造	コード構造
視覚的表現	コードの読みやすさ
ドキュメンテーション	コードの説明

Mäntyläらの欠陥分類：過去に示された分類を包括した定義

コードインスペクション指摘の約70%は、発展性欠陥である (Mäntylä, 2009)

Mäntylä, M. V. and Lassenius, C., "What Types of Defects Are Really Discovered in Code Reviews?," *IEEE Trans. Softw. Eng.*, vol. 35, no. 3, pp. 430-448, 2009.

補足資料：欠陥（失敗）の伝達は容易ではない

例：「失敗学」における「失敗」知識の伝達アプローチ

・逆流
・誤差蓄積
・脆弱構造
・フィードバック系暴走
・…

「失敗」の原因を探り、抽象化し、類型化し、連想キーワードで表現

→ 少なくとも、HW非専門家の目には、面白そうに思える

中尾政之
『続・失敗百選』
森北出版（2010）

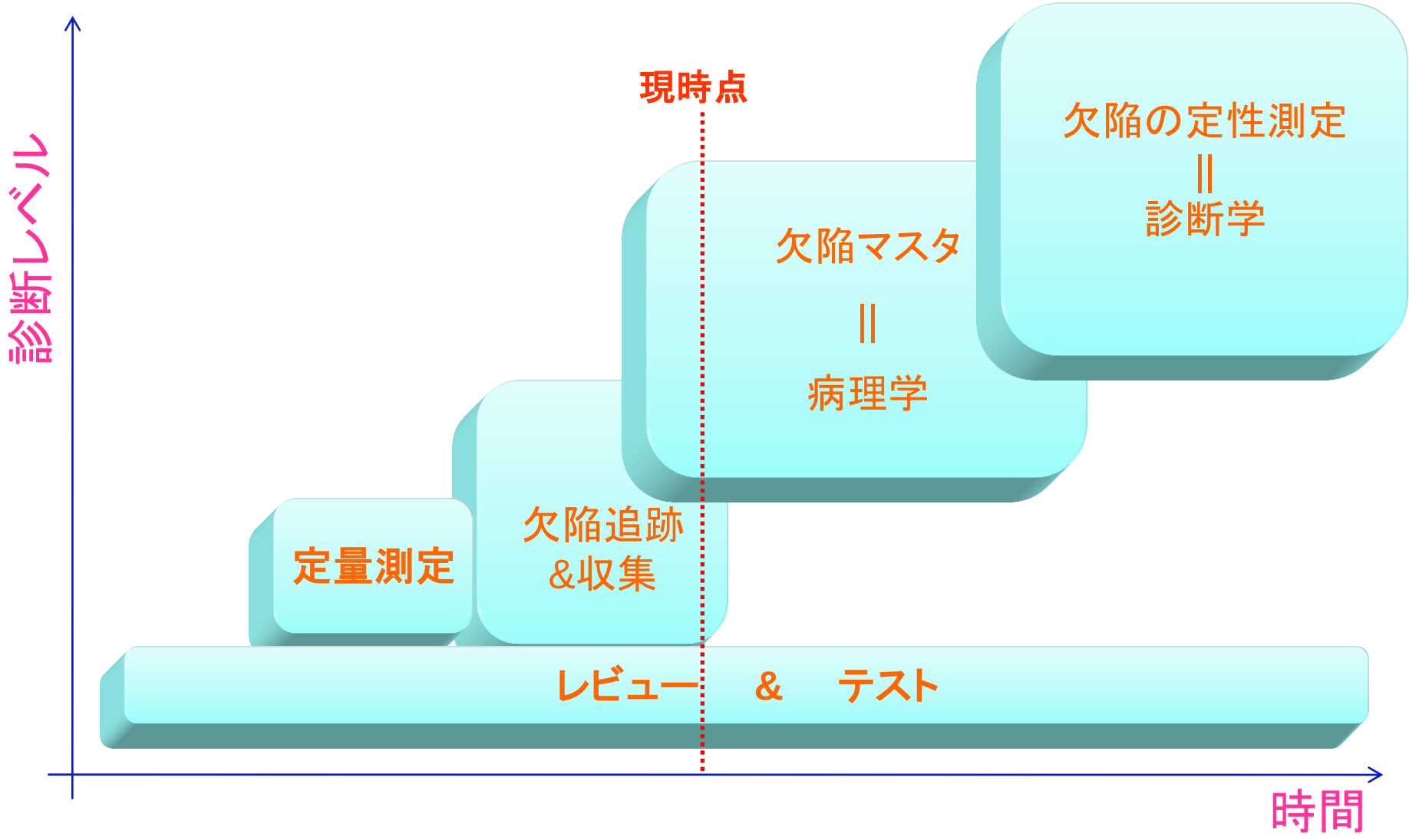
例：IPA/SEC「重要インフラ情報システム信頼性研究会」報告書（2009）に見る、「失敗」伝達の難しさ

http://sec.ipa.go.jp/reports/20090409.html

重要インフラ情報システムの障害事例の収集（約100事例）
→ 原因の分析（ソフトウェア欠陥 ／ 運用ミス ／ … ）
→ 問題の構造を分析（当該システム障害事例の当事者ではないが、専門家が時間をかけて議論）
→ 再発防止のポイントを分類・整理
　ー例：**「複数の経験者によるレビューを徹底し、仕様、プログラムなどを充分レビューする」**

よい議論を経て生み出された成果物のはずなのに、再利用可能な「悪さの知識」に昇華されているとは言いがたい

8. 欠陥エンジニアリングの未来



9. 作る側の面白さ－未来予想図－

メリットを提供できる「欠陥（バグ）視点」だから面白い。
面白さの一部を紹介します。

1） 欠陥モデリング言語（DML）

2） バグDB参照「Quality Assist」

3） バグ登録サポート（自動分類）

4） 「欠陥１件」の数え方から、現場の実践に繋がる統計解析手法まで

5） 抗体／免疫作成

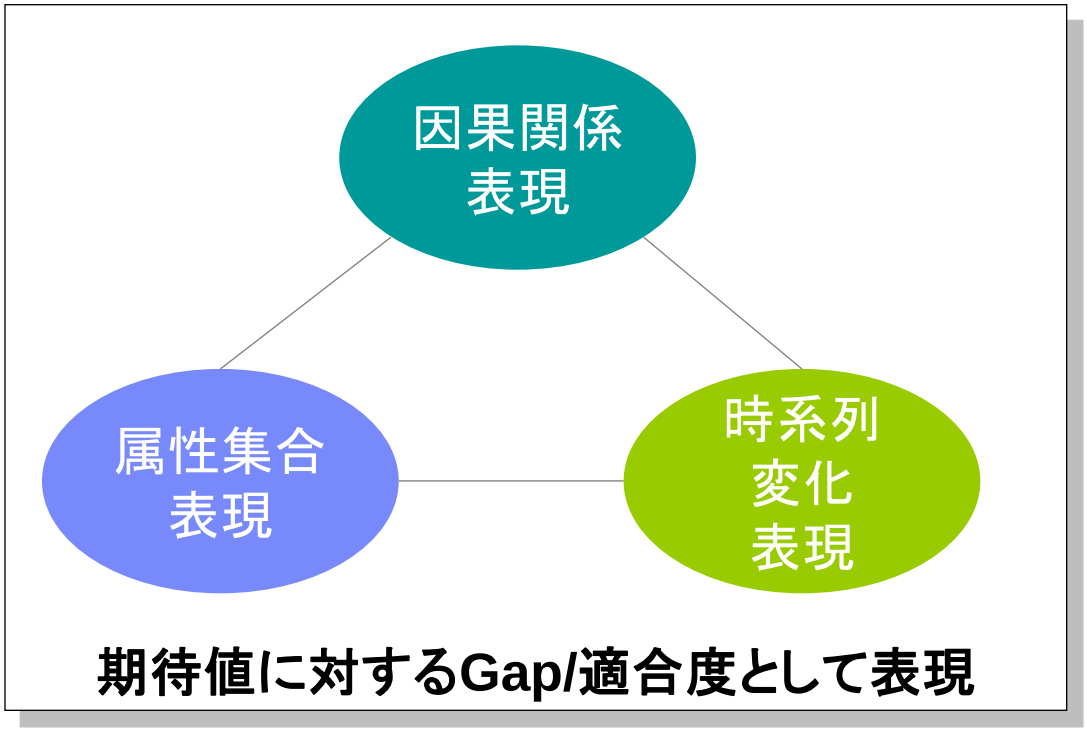
6） バグ屋の集結→コラボレーション

7） 日本人だからこそ可能、日本発信

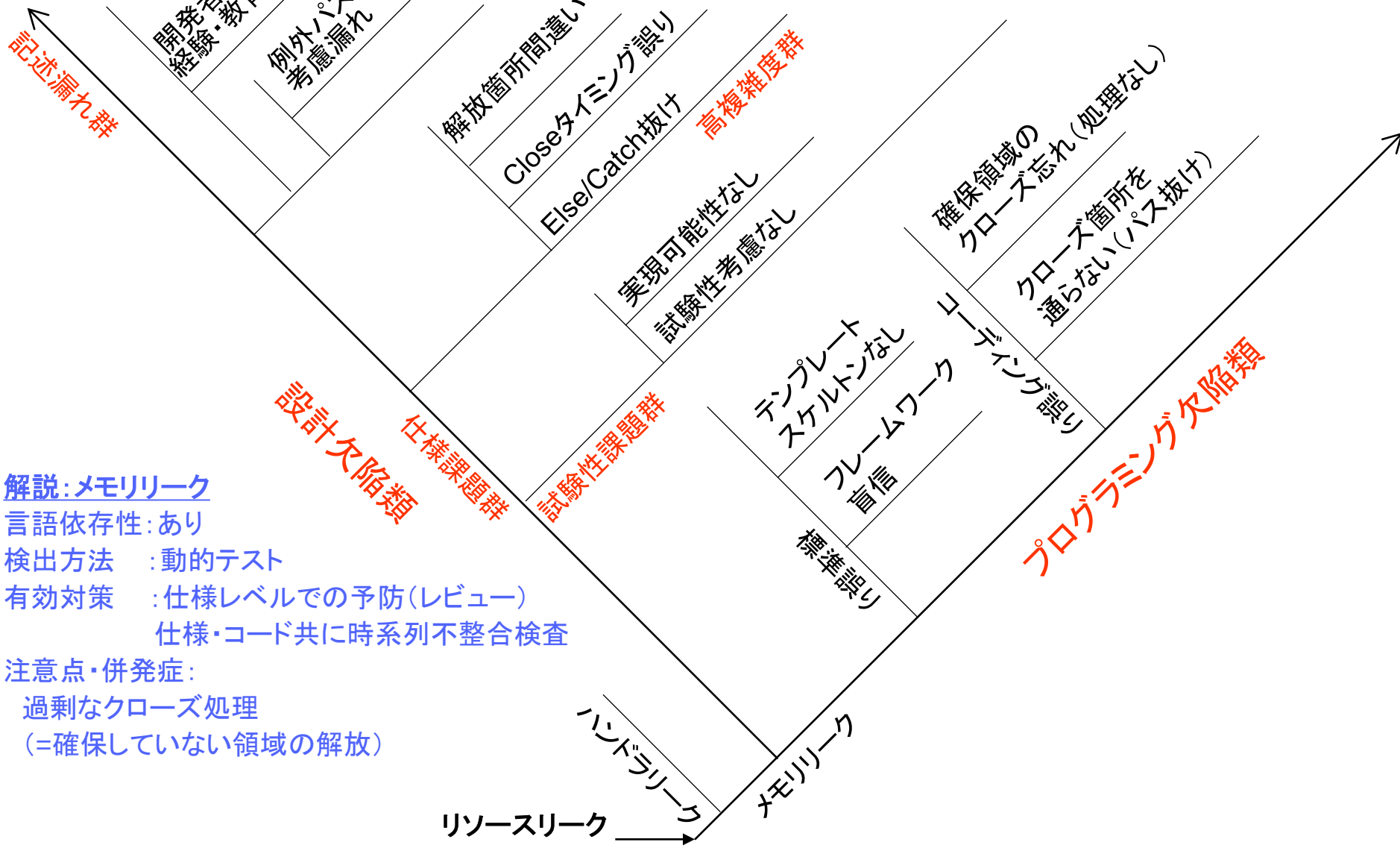
8） Project Fabreでの議論を通じた、バグにまつわる果てしない構想…

9. 作る側の面白さ:欠陥モデリング言語(DML) 現在検討中

- 欠陥の伝達・移転・展開のための共通記述言語を作成中
- 欠陥の保存・分類・検索のための付加属性表現方法



欠陥の表現例：リソースリーク Specimen :: メモリリーク



解説：メモリリーク

言語依存性：あり
検出方法　：動的テスト
有効対策　：仕様レベルでの予防（レビュー）
　　仕様・コード共に時系列不整合検査
注意点・併発症：
　過剰なクローズ処理
　（=確保していない領域の解放）

10. 具体的に何をするのか？

例題を使って、検出欠陥の「固定・抽象化」「分類」「整理」を体験します。
登録した抽象データの利用を実際に考えます。

バグの定義と概要
原因
　　混入原因
コスト・期間影響
　　一般的なコスト影響
　　時期　混入時期
　　　　検出時期（推奨）
　　難易度（検出・除去・予防）
　　合併症／併発症

　　兆候情報
　　　定量的兆候　　測定方法
　　　定性的兆候
　　　検査依頼目的／観察所見／
　　　触診所見他
推奨検査方法
　　使用自動検査ツール
　　目視検査方法
推奨対応方法

Project Fabreメンバーによるカルテの紹介 → 例題からカルテ作成・利用

【ワークショップ】例題のバグ票とコンテキスト、ドメイン情報

■1) 現象
勤怠情報入力の画面で、時分入力を使用していると、
端数が合わないことがある。
例えば、残業時間を19.51（19時間51分）と入力すると、Enter押下で19.50（19時間50分）になる。十進法では19.85時間となるはずが、19.84時間と表示される。

■2) 対象製品、画面
給与計算システム、勤怠入力画面（Delphiクライアントモジュール）

■3) 発生条件
以下の全てを満たす場合に発生する可能性があります。
(入力する数字によっては正しく変換されます。)

①メインメニュー⇒月例給与⇒勤怠処理
⇒勤怠データ入力⇒勤怠情報入力⇒(ツールバー)オプション(O)
⇒「時間項目の時分入力」にチェックが入っている。
②メインメニュー⇒月例給与⇒給与マスタメンテナンス
⇒基本設定⇒項目名称データ⇒(明細 F5 押下)項目名称データー覧の時間区分が「時分入力→時間」となっている。
③上記の項目に結び付く端数処理パターンが 小数点以下第3位を切捨てる設定になっている。
(端数パターン設定はメインメニュー⇒月例給与
⇒給与マスタメンテナンス⇒その他マスタ⇒端数処理パターン設定より設定できます。)

■4) 起動経路
[クライアント] メインメニュー⇒月例給与⇒
勤怠処理⇒勤怠データ入力⇒勤怠情報入力

■5) 混入経緯
派生開発時（入力補助機能開発時）

■6) 混入～発見までの期間
長期

■7) 発見経緯
お客様業務（保守フェーズ）

■8) 回避策
有

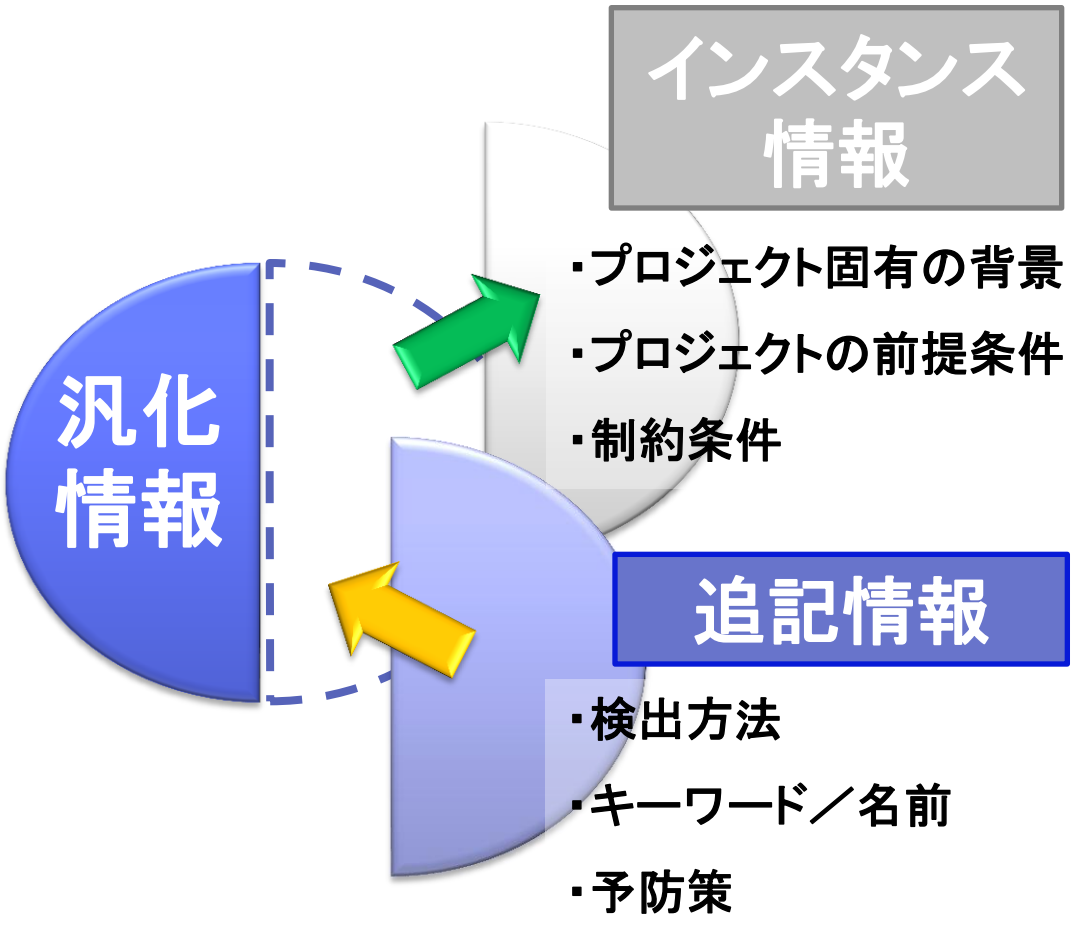
■9) エラーメッセージ
なし

■10) 不具合分類
画面の動作不備

■11) 影響範囲
他サブシステムへの影響：なし

10. 具体的に何をするのか？　:欠陥マスターDBの作り方

インスタンス情報から、欠陥マスターDBへ転化させるには？



「固有」部分を切り出し

汎化された情報

所見を追記して欠陥マスターDBへ

10. 具体的に何をするのか？欠陥マスターDBを作る際のポイント

ポイントは、3C！

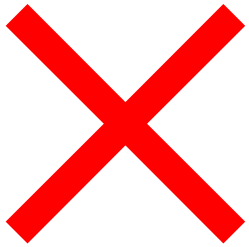
Context（システム目標他、ビジネス課題等）の理解齟齬

Constraints（制約事項）の考慮漏れ・認識齟齬

Condition（前提条件）の説明不足・理解度の差が存在

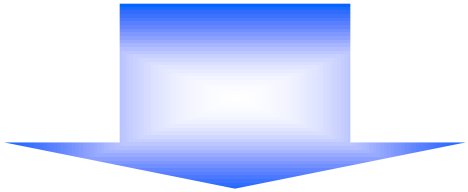
※ただし、上記3Cを取り除いた固有部分も必ず保持して、「具象化」の必要に応じて戻せる状態を作る。

11. 本ワークショップのゴール



バグ票やBTSデータを分析しているけども、
活用し切れていない。

そもそも、バグ票はクローズしたら終わりです。



身近なバグの財産化・資産化を可能に。

会社単位でも、プロジェクト単位でも。