はじめに

- ソフトウェア開発のプロジェクトで数多くの欠陥が発生し報告されています。
- 現場では、各報告を元に原因を探し出し、修正し、修正確認を行っています。
- これらの情報を整理・分類し、傾向を調査することは、現場で既に実施されていることでしょう。
- 欠陥を分類する方法としてIBM Researchに掲載されている「Orthogonal Defect Classification（ODC：直交欠陥分類）」が知られています。
- ここでは、ワークショップで使用するODCについての簡単な説明を行います。

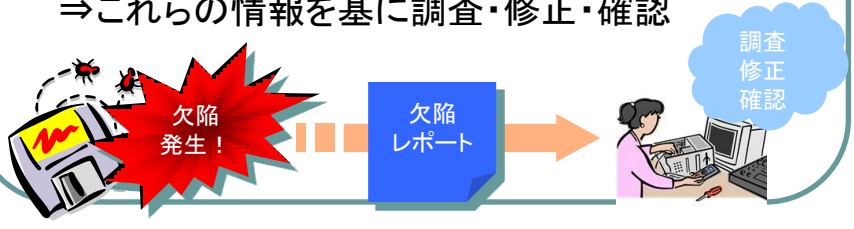
欠陥レポート

- 欠陥レポートの役割
  - テスト実施結果の報告
    - 欠陥（バグ、テスト仕様と異なるなど）を報告

  ⇒欠陥発生状況の連絡

  発生条件、発生手順、発生したテスト項目情報

  ⇒これらの情報を基に調査・修正・確認



欠陥レポート

- 欠陥レポートが報告する対象

⇒現在開発・対応中のプロダクトの欠陥

- 現在のプロダクトが改善対象？

⇒視点を変えてみる

欠陥が発生する＝何かしらの弱点がある

⇒弱点を見つけ対応するための情報と捉える

- 欠陥は弱点が顕になった状況
- 普段見えない弱点が顕になった状況を捉まえる

欠陥レポート

- 経験則：不良品質の発生や原因が偏る

  Dr. Joseph Moses Juran

- 欠陥の発生状況から真の原因を追究・対策することができる

  ※ 1通の欠陥レポートでなく、収集された欠陥レポートの傾向を調査する

欠陥レポートは宝の山

- でも、毎回宝探しはゴメンです

  地図を探して、仲間を探して、道具を探して、
  さあ！出発。。。（は1回きりでいいよ）

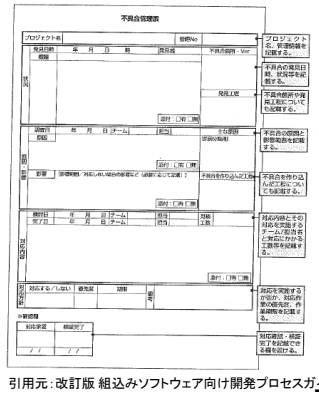



- 誰でも宝の山を手に出来るようにするには
  - 記載情報を明らかにする
  - 情報を確実に得られるようにする
  - 調査・分析の手間を省く

欠陥レポートは宝の山

- 記載情報を明らかにしておく
  - 記載項目、様式を決める
  - 記載方法
    - 要点を明らかにする
    - 正確に記載する
    - 中立性を保つ

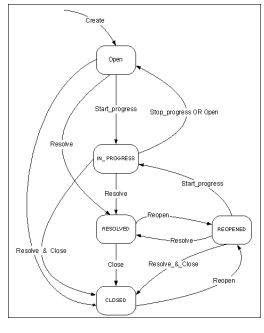
■ (SU601) 不具合管理票の例



引用元：改訂版 組込みソフトウェア向け開発プロセスガイド
P207（IPA/SEC編著）

欠陥レポートは宝の山

- 情報を確実に得られるようにする
  - レポートのステータスを明らかにする
    例：起票直後、調査中、修正中、
    確認中、終了 など
  - レポート処理ルートを決める
    レポートの処理を見失わない
    ワークフローツールを活用する
  - 最後は管理台帳に集約する



JIRAのワークフロー
出典：http://confluence.atlassian.co.jp/pages/viewpage.action?pageId=23724583

欠陥レポートは宝の山

- 調査・分析の手間を省く
  - 予め分類項目を決めておく
    - レポート作成者が半ば機械的に分類できるとbetter

分類の例

分類名称	特徴
Orthogonal Defect Classification （ODC：直交欠陥分類法）	IBM Resaerchが1990年代に経験を元に考案。 普遍的な分類。本セッションで解説。
ボーリス・バイザーの分類	Boris Beizerが Software Testing Techniquesにて紹介。やや煩雑かつ1980年代の考え方が主と思われるため割愛。
IEEE1044-2009：IEEE Standard Classification for Software Anomalies	IEEEで規定された分類法 IEEE 1044-1, a guide to the classification of software anomalies なるガイドラインあり 規格は有償で入手できる。（有償のため今回は割愛）

Orthogonal Defect Classification（ODC：直交欠陥分類法）とは？

- **迅速にソフトウェアの欠陥を意味的に捉える分類法**
  - **ODCが提示するモデルを基に統計的解析を行う**
    健康診断における血液検査のようなもの
  - **一般的なメトリクスでは露見しない問題が発見でき，多角的な評価が可能になる**

構成

- 次の2つで構成される

構成内容	項目	意味
欠陥報告時（Opener Section）	Defect Removal Activities	どこのプロセスで発見したか？
	Triggers	どのような状況で発生するか？
	Impact	どのような影響を及ぼすか？
欠陥修正時（Closer Section）	Target	どこが悪かったのか？
	Defect Type	何を間違ったのか？
	Qualifer	どのように間違ったのか？
	Age	どの時点で間違ったのか？
	Source	誰の物が間違ったのか？

属性

ー 欠陥報告時(Opener Section）ー

実際に欠陥を発見した場所

	項目	内容
Defect Removal Activities（どこのプロセスで発見・報告したか？）	デザインレビュー	仕様レビューで報告
	コードインスペクション	ソースコードレビューで報告
	ユニットテスト	関数単体テストで報告
	機能テスト	結合テストで報告
	システムテスト	システムテストで報告

属性

ー 欠陥報告時(Opener Section) ー

欠陥が発見された環境や条件

	項目	内容
Triggers（どのような状況で発生するか）	Design Confformannce（仕様適合）	仕様と整合性がとれているか
	Logic／Flow（論理／フロー）	フローそのものが論理的におかしい
	Backward Compatibility（旧バージョン互換性）	旧バージョンで動作していたものが現在のバージョンで動作しなくなった
	Lateral Compatibility（一般的互換性）	一般的な他のシステムとのインターフェースを利用する際の問題
	Concurrency(並行性)	マルチタスクに関わる問題
	Internal Document（内部文書）	記述漏れや記述誤りなどドキュメントの問題
	Langage Dependency（言語依存性）	文法や初期値、メモリ管理などプログラミング言語に起因する問題

属性

ー 欠陥報告時(Opener Section) ー

	項目	内容
Triggers（どのような状況で発生するか）	Side Effect（副作用)	バグ修正による新たなバグ
	Rare Situation（レアケース）	めったに発生しない
	Simple Path（通常使用）	単純な抜け漏れ
	Complex Path（複合条件での使用）	前後条件を考慮されていない
	Coverrage（カバレッジ）	エラー処理や例外処理など、想定されるケースが無い
	Variation（バリエーション）	パラメーターの組み合わせによる問題
	Sequencing（連続操作）	繰返し操作による問題

属性

ー 欠陥報告時(Opener Section) ー

	項目	内容
Triggers（どのような状況で発生するか）	Interaction（相互作用）	各機能が独立して実行すると正常に実行されるが特定の組み合わせで失敗する
	Workload／Stress（負荷／ストレス）	負荷状態での問題
	Recover／Exception（リカバリー／例外）	例外処理のリカバリー対応に問題
	Startup／Restart（起動時／再起動時）	初期値や再起動時の問題
	Hardware Configration（ハードウェア構成）	特定のハードウェアに問題が合った場合
	Software Configuration（ソフトウェアコンフィグレーション）	特定のソフトウェアに問題が合った場合
	Blocked Test (previously Normal Mode)（ブロッキングされたテスト）	他の要因でテスト不能になった場合

属性

ー 欠陥報告時(Opener Section) ー

	項目	内容
Impact（どのような影響を及ぼすか？）	Installability（インストール）	インストールが完了できない
	Serviceability（保守性）	運用時どんな障害が発生しているのかわからない
	Standards（標準）	業界標準や社内基準などに準拠しない
	Integlity／Seculity（完全性／セキュリティ）	システム、プログラム、および不注意または悪意のある破壊、改ざん、または漏洩からデータを保護できない
	Migration（マイグレーション）	アップグレードなどシステム移行できない
	Reliability（信頼性）	安定して期待された役割を果たすことができない
	Performance（パフォーマンス）	速度など性能がでない

属性

ー 欠陥報告時(Opener Section) ー

	項目	内容
Impact（どのような影響を及ぼすか？）	Documentation（ドキュメンテーション）	ユーザーマニュアル・仕様書等に記載が無いなど、説明が無い
	Requirements(必要条件)	客の明示する期待を満たしていない
	Maintenance（メンテナンス）	ソースが整理されておらず修正が容易に行なえない
	Usability（ユーザビリティ）	使い勝手が悪い
	Accessbility（アクセシビリティ）	障がいのある人に情報を提供できるようになっていない
	Capability（ケイパビリティ）	保存ボタンがあっても保存できない等、要求を処理する能力がない

属性

ー欠陥修正時(Closer Section) ー

	項目	内容
Target（どこが悪かったのか？）	Requirements（要件）	顧客の要件に問題
	Design（仕様／設計）	仕様書や設計書に問題
	Code（コーディング）	ソースコードに問題
	Build／Package（ビルド）	ビルド時やパッケージを行う際に問題
	Information Development（メッセージ）	ユーザーへの情報提供に問題
	National Language Support（言語）	マルチ言語対応に問題

属性
－欠陥修正時(Closer Section) －

何を修正したか？

	項目	内容
Defect Type（何を間違ったのか？）	Assign／Init（割当／初期値）	変数の値／初期値が問題
	Checking（チェック）	チェックのやり方が問題
	ALG／Method（アルゴリズム／方法）	アルゴリズムや実装方法が問題
	Func／Class／Object（関数／クラス／オブジェクト）	関数やクラス、オブジェクトが問題
	Timing／Serial（タイミング／シリアライズ）	タイミングや順序が問題
	Interface／O-Omessages（インターフェース／オブジェクト間メッセージ）	呼び出し時の方法が問題
	Relationship（関係性）	クラス間の継承関係が問題

属性
－欠陥修正時(Closer Section) －

	項目	内容
Qualifer（どのように間違ったのか？）	Missing（抜け漏れ）	抜けや漏れがあった
	Incorrect（不正確（誤り））	間違った使い方をした
	Extranecus（無関係なものが存在）	例えば消し忘れなど紛れ込んだ異物が原因

属性
－欠陥修正時(Closer Section) －

	項目	内容
Age（どの時点で間違ったのか？）	Base（元から）	変更などしていない既存の箇所
	New（新規）	新規に作成した箇所
	Rewritten（書き直した際）	再設計又は古い機能の書き直しを行った箇所
	ReFixed（修正時）	欠陥の修正を行った箇所

属性
－欠陥修正時(Closer Section) －

	項目	内容
Source（発生源はどこか？）	Developed in-House（自社開発）	自社にて開発を行った箇所
	Reused From Library（ライブラリからの再利用）	ライブラリ関数を再利用した箇所
	Outsourced（アウトソーシング）	アウトソースにて開発を行った箇所
	Ported（移植）	他のプラットフォーム向けより移植を行った箇所

参考文献

番号	種別	内容
1	Web	Center for Software Engineering ホームページ http://www.research.ibm.com/softeng/ODC/ODC.HTM
2	Web	JIRA 日本代理店 ホームページ http://www.atlassian.com/ja_JP/software/jira/
3	Web	IEEE1044-2009 ダウンロードページ http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5953439
4	Web	メタボリックス社 IEEE解説ページ http://www.metabolics.co.jp/SoftwareProcess/ieee-scsm.html
5	論文	松村知子，森崎修司，玉田春昭，大杉直樹，門田暁人，松本健一 「プロセス改善を目的とする ODC を用いた欠陥修正工数分析」 http://www.empirical.jp/paperdb/papers/archive/149/149
6	論文	中沢俊彦，「欠陥を設計開発プロセスの課題として認識するための欠陥分類方法に関する研究」，2007 http://www.rdcmodel.com/Defect%20classification%20to%20identify%20inadequacies%20in%20design%20processes.pdf
7	書籍	ボーリス・バイザー，「ソフトウェアテスト技法」，日経BP社，1994
8	書籍	独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター編著， 「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
9	書籍	高橋寿一，湯本剛，「現場の仕事がバリバリ進む ソフトウェアテスト手法」，技術評論社，2006
10	Web	@IT情報マネジメント 用語集「パレート図」http://www.atmarkit.co.jp/aig/04biz/paretochart.html