

開発標準プロセスを用いた不完全なソフトウェア要求に対する問題 検出の分類法

あらまし 発表概要（幹事が参考にするだけで公開はされない）：宇宙機のソフトウェアにおいて、ソフトウェアの不具合はミッションの成功に対して大きな妨げとなっている。本研究ではソフトウェア不具合の大きな原因となっている要求漏れに着目した。実際に宇宙機で発生したソフトウェアの不具合を元に、障害の分類と開発標準プロセスの2つの観点から不具合の分類を行っている。この分類を行うことで、どのような不具合が多いのか、どのプロセスにおいて不具合が発生しやすいのかが明確になる。

キーワード

1. はじめに

1 章

2. 背景

2 章

3. 提案する分類の枠組み

本研究で2つの指標を用いる。1つ目は、不具合の原因となっている欠陥についての分類である。2つ目は、開発の工程を示している開発プロセスである。この異なる軸の2つの指標を組み合わせることで、見えていなかった不具合の傾向の発見つながることが期待できる。この章では、それぞれの指標について説明をする。

3.1 欠陥の分類

まず初めに、不具合とは、欠陥とは何かについて説明をする [1]。図 1 に示すように不具合とは、本来提供されるべき、正しいサービスが何らかの影響で正しく提供されない事である。その正しくないサービス状態のことを Failure（障害）という。障害につながりうるシステムの内部状態を Error（誤り）という。誤りが生じるきっかけや、原因となるものが Fault（欠陥）である。今回はその欠陥について着目した。

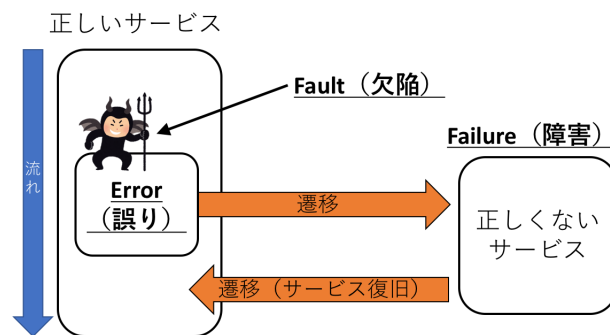


図 1 不具合に関する欠陥、誤り、障害の関係

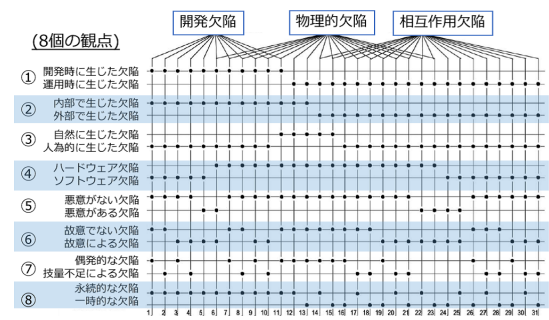


図 2 欠陥の分類結果

先行研究 [1] では、一般的な不具合について整理されている。特に、不具合全体の誤りの原因となる欠陥について着目している。図 2 2 値の値を持つ 8 つの観点を用いて、合計 2 5 6 通りから言葉の定義上存在しないものを除いた、3 1 通りで欠陥を網羅的に分類している。一方で、原因となる欠陥が分類できたとしても、高信頼性システムに致命的な影響を与える”要求の不完全さ”に対処することは困難である。

本研究では、この分類からソフトウェアに関係し、さらに開発時に生じたものに着目した。その結果 3 1 通りの分類が、図 2 に示す 1 ～ 5 番までの 5 通りとなった。さらに、”悪意がある欠陥”は存在しないものとした。よって対象とするのは Intent の有無、Capability の有無の 4 つである。

3.2 開発プロセス

本研究では宇宙航空研究開発機構（JAXA）が提供している、ソフトウェア開発標準を使用した。これは ISO12207 * を元に作られたものである。図 1 に示すようにソフトウェア開発標準は大きく 2 つのプロセスから成り立っている。

1 つ目が主ライフサイクルプロセスであり、もう一

主ライフサイクルプロセス	開発プロセス
	運用プロセス
	保守プロセス
支援ライフサイクルプロセス	文書化プロセス
	構成管理プロセス
	品質保証プロセス
	検証プロセス
	妥当性確認プロセス
	共同レビュープロセス
	アセスメントプロセス
	問題解決プロセス

表 1 主ライフサイクルプロセスと支援ライフサイクルプロセス

プロセスの開発準備	ソフトウェア開発計画を立案すること
コンピュータシステム	開発計画の作成
ム	要求を抽出する
コンピュータシステム	要求仕様書の作成
ム	構成する部品、機能の明確化
方式設計	各機能及び部品に要求を割り当てる
ム	実装可能性の評価
ム	設計制約と前提条件を明らかにし、評価
ム	上位とのトレーサビリティの評価
ム	インターフェース要求の抽出
ソフトウェア要求分析	ソフトウェア要求仕様書の作成
新	面別・面内子を行な
新	データ及びデータベースに対する仕様を含める
新	実装制約及び制約に依存する仕様を含める
新	インターフェース仕様に関して合意を得ること
新	上位との整合性を保つ
新	実装可能性の評価
新	北からあるものを言うときは整合性などを確認
新	前提条件、制約を明確に
新	検証可能性の評価
新	機能や目的の矛盾がある場合確認の評価
ソフトウェア設計	機能要求のモジュール間の関係を明確に
ソフトウェア設計	機能外要求のモジュール間の関係を明確に
ソフトウェア設計	評価結果、SW 構成の合意を得ること
ソフトウェア設計	ソフトウェア設計を行う
ソフトウェア設計	詳細化し合意を得る
ソフトウェア設計	上位との整合性を保つ
ソフトウェア設計	実装可能性の評価
ソフトウェア設計	前提条件、制約を明確に
ソフトウェア設計	ソフトウェア試験を立案
ソフトウェア設計	前提条件などを更新
ソフトウェア設計	コーディングのルールを明確に
ソフトウェア設計	エラー処理の策定と実装計画を考慮
ソフトウェア設計	コード作成
ソフトウェア設計	規約を守っているか
ソフトウェア設計	非公式仕様書を作成
ソフトウェア設計	非公式仕様書を実装
ソフトウェア設計	分岐を十分に把握しているか
ソフトウェア設計	非公式仕様書を実装
ソフトウェア設計	トレーサビリティを確認
ソフトウェア設計	前提条件、制約を更新
ソフトウェア設計	ベースラインを確定
ソフトウェア設計	問題解決プロセスを実行
ソフトウェア設計	試験計画
ソフトウェア設計	試験実施

図 3 詳細な主ライフサイクルプロセス（中身変更の可能性あり）

つが、支援ライフサイクルプロセスである。図??に示すように主ライフサイクルプロセスは、ソフトウェア開発に直接かかわってくるソフトウェアライフサイクルのプロセスであり、3 つに分けられ、さらにそれらが細かく分けられている。支援ライフサイクルは、主ライフサイクルプロセスを支える、他のプロセスから呼びだされるプロセスであり、8 つに分けられおり、こちらも同様に細かく分けられている。本研究では、”要求が不完全であることに起因するソフトウェア問題を減らす”という目的があったため、主ライフサイクルプロセスの中の開発プロセスに着目した。

4. データの適用

4 章

5. 考 察

5 章

6. ま と め

6 章

謝辞

文 献

[1] A. Avizienis, J.C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” IEEE Transactions

付 録

1.

(平成 xx 年 xx 月 xx 日受付)

フェーズ	詳細な内容
プロセスの 開発準備	ソフトウェア開発計画を立案すること 開発計画の文章化
コンピュータシステム 要求分析	要求を抽出する 要求仕様書の作成
コンピュータシステム 方式設計	構成する品目、種別を明確に
	各構成品目に要求を割り当てる
	実現可能性を評価
	設計根拠と前提条件を明らかにし、評価
	上位とのトレーサビリティを評価
ソフトウェア 要求分析	インターフェース要求を抽出
	ソフトウェア要求仕様書の作成
	個別に識別子を付与
	データ及びデータベースに対する仕様を含める
	異常検知及び処理に関する仕様を含める
	インターフェース仕様に関して合意を得ること
	上位との整合性を得る
	実現可能性の評価
	元からあるものを使うときは整合性などを確認
	前提条件、制約を明確に
	検証可能性を評価
	試験計画可能性を評価
ソフトウェア 設計	環境や HW の影響がある場合確認の評価
	機能要求のモジュール間の関係を明確に
	機能外要求のモジュール間の関係を明確に
	IF 機能、SW 機能の合意を得ること
	ソフトウェア設計を行う
	詳細化し合意を得る
	上位との整合性を確認
	実現可能性を評価
	前提条件、制約を明確に
ソフトウェア 製作	ソフトウェア試験を立案
	前提条件などを更新
	コーディングのルールを明確に
	エラー処理の実装指針を考慮
	コード作成
	規約を守っているか
	単体試験仕様を作成
	単体試験を実施
	分岐を十分に網羅しているか
ソフトウェア 統合	静的解析を実施
	トレーサビリティを確認
	前提条件、規約を更新
ソフトウェア 統合試験	ベースラインを確定
ソフトウェア 統合試験	問題解決プロセスを実施
ソフトウェア 統合試験	試験準備
ソフトウェア 統合試験	試験実施

表 2 詳細な主ライフサイクルプロセス（内容要変更）

Abstract

Key words