論　文

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。

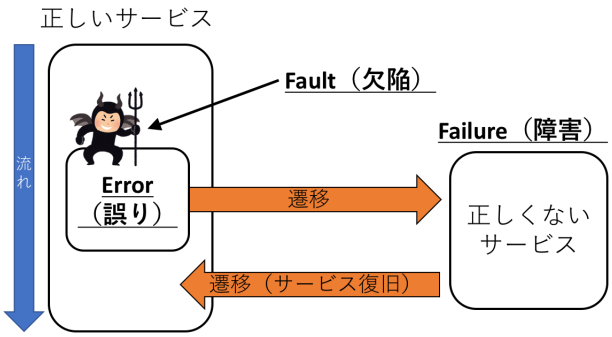


図 1　不具合に関する欠陥、誤り、障害の関係

†

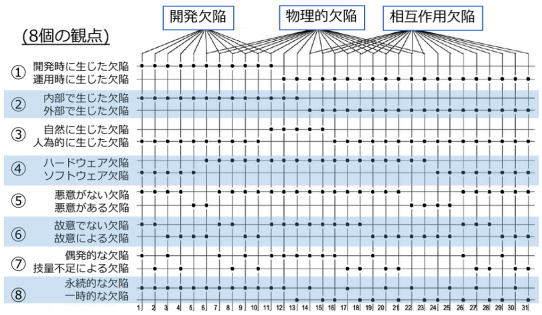


図 2　欠陥の分類結果（要修正）

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２２値の値を持つ８つの観点を用いて、合計２５６通りから言葉の定義上存在しないものを除いた、３１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３１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。表 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、もう一つが、支援ライフサイクルプロセスである。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わけられている。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、表２に示す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。

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

4.　データの適用及び結果

この章では本研究で使用したデータについて説明する。また、２つの指標をどのように使ったのかについて述べる。

4.1　使用したデータ

今回 JAXA から４８件提供の実際に発生した不具合情報のレポートを提供してもらった。このデータは機密事項な為、詳細には記述しない。また、この４８件のデータ１つにつき１つの障害を示しているが、欠陥は１つとは限らない。

4.2　適 用 方 法

本実験では以下の手順で適用した。

- 不具合１つ１つに対して、欠陥が何であるかを明確にした
- 明確になった欠陥に対して、欠陥分類のどれに該当するのかをマッピングした
- マッピングされた欠陥に対して、それがどのプロセスで見つけるべきであったかを確認した
- 確認したデータを集計した

4.3　適 用 結 果

4.3.1　欠陥の抽出および欠陥分類のマッピング

今回の４８件の不具合から１１０件の欠陥を抽出することができた。この欠陥を分類した結果を表 3 に示す。分類区分とは図 2 で示した分類 1 ～4 に対応している。Dellberate fault であり Accidental faultsa である☒は３１件、Dellberate fault であり Incompetence faults であるは件２２、Non-Dellberate fault であり Accidental faults であるは件１３、Non-Dellberate fault であり Incompetence faults であるは件４４であった。各不具合に対する結果の分類の細かなデータは付録にて掲載する。

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
	インターフェース要求を抽出
ソフトウェア要求分析	ソフトウェア要求仕様書の作成
	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
	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
	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
	環境や HW の影響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
	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
	ソフトウェア試験を立案
	前提条件などを更新
ソフトウェア製作	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
	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	ベースラインを確定
	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
	試験実施

表 2　詳細な開発プロセス（内容要変更、3.1 などの数字も入れたい）

分類区分	欠陥の種類	件数
☒	Dellberate fault であり Accidental faults	31
☒	Dellberate fault であり Incompetence faults	22
☒	Non-Dellberate fault であり Accidental faults	13
☒	Non-Dellberate fault であり Incompetence faults	44
合計		110

表 3　各欠陥に対する不具合の数

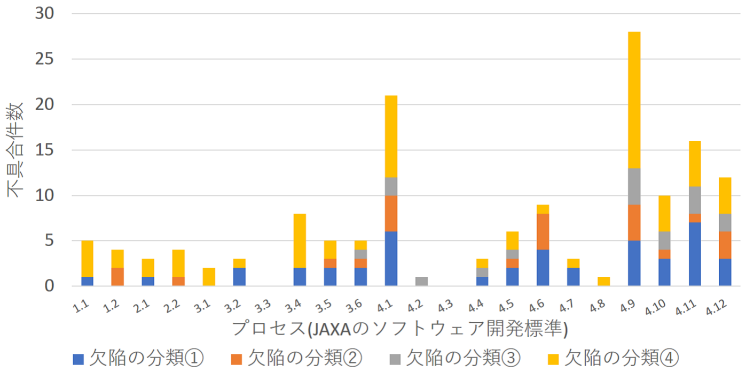


図 3　積み上げグラフ

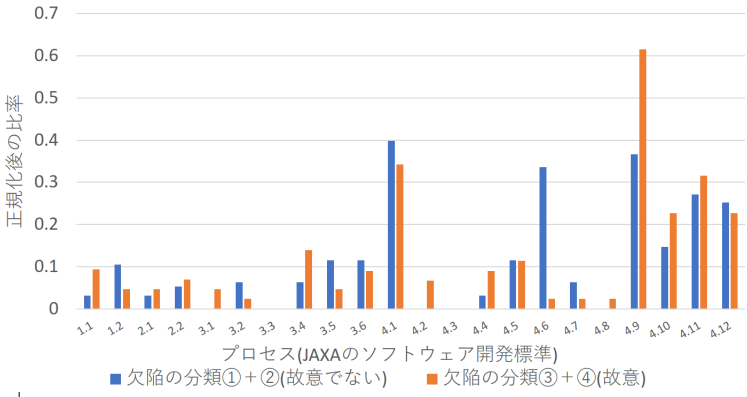


図 4　☒+☒と☒+☒の比較

4.3.2　集　　計

上記の結果よりどのプロセスで発見すべきだったのかをマッピングした。マッピングした結果は付録に掲載する（図 A·1）1 つの欠陥に対して複数の個所が該当することもある。

プロセスにより分類した結果を☒～☒に対してそれぞれ集計し、グラフ化を行うことでそれぞれの特徴が見えてくる。図 3～図 5 はその結果である図 3 は集計したものを合わせ、どのプロセスでの不具合が多いのかを示したグラフとなっている。図 4 は☒～☒に対して正規化を行い、Dellberate fault と Non-Dellberate fault で比較したものである。図 5 も同様に☒～☒に対して正規化を行い Accidental faultsa と Dellberate fault で比較したものである。

5.　考　　察

本章では、JAXA のデータから導かれた結果につい

て考察を行う。

5.1 積み上げグラフについて

5.2 ☒+☒について（タイトル要変更）

5.3 ☒+☒について（タイトル要変更）

5.4 各フェーズの関係性について

ゼミのラストの結果の部分、タイで分析した結果の考察などを載せる

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 on Dependable and Secure Computing, vol.1, no.1, pp.11–33, Jan. 2004.

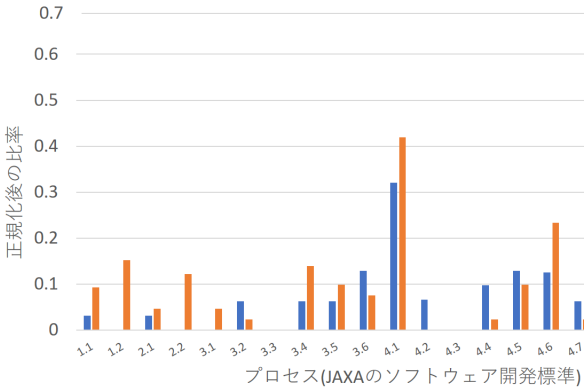


図 5 ☒+☒と☒+☒の比較

付 録

表 3 の詳細なデータである。

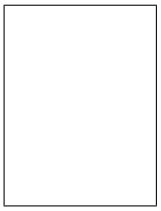
図 A·1 はプロセスによって欠陥を分類した結果である。

1.

（平成 xx 年 xx 月 xx 日受付）

不具合事例	欠陥の分類				不具合事例	欠陥の分類				不具合事例	欠陥の分類			
	☒	☒	☒	☒		☒	☒	☒	☒		☒	☒	☒	☒
1		1		1	17		1		1	33	1	1		2
2	1	1			18				1	34	1			2
3	1				19			1	1	35	1	1		1
4				1	20	1			1	36				1
5	1		1	1	21	1	1	1		37		1		2
6	1			2	22		1			38		2	1	
7	1			1	23		1			39			1	2
8	1			1	24		1			40	1			
9	1			1	25	1				41		1		
10	1			1	26	1	1		1	42		1		2
11	1			1	27	2	1	2	1	43		1	1	
12				1	28	1		1	3	44		1	1	
13				1	29	2	1		3	45				1
14	1				30	2	2	1		46	1			
15	1			2	31		1			47			1	1
16	3		1	2	32	1			1	48				1

表 A·1 詳細な欠陥に対する不具合の数



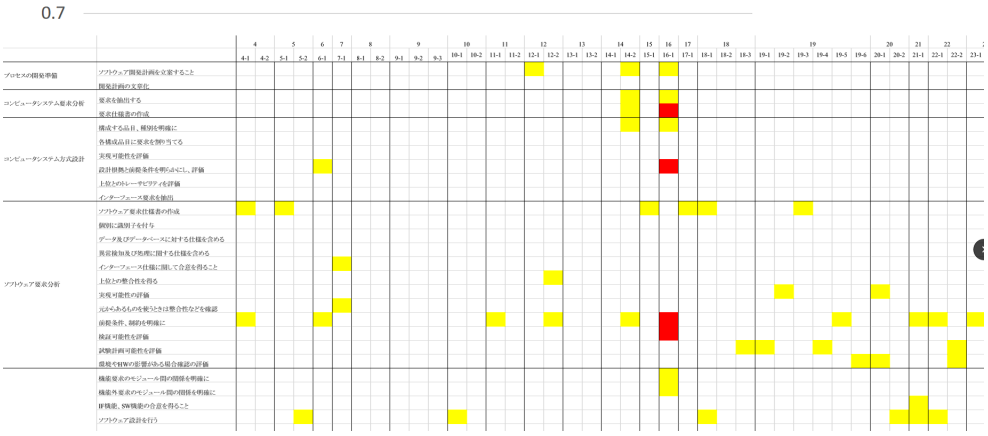


図 A·1　プロセスによる分類

Abstract

Key words