69

観点を用いたソフトウェアにおける FMEA の効率的・効果的な実施方法とその効果[1]

1. はじめに

東芝では、ソフトウェアの品質向上を達成するため、下記①～④を主な目的として、ソフトウェア開発におけるリスクアセスメント手法の開発・展開に取り組んできた[1]。

① 不具合の再発防止と未然防止の実現
② 開発効率の向上（後戻りの削減）
③ 分散開発管理の改善（思考過程の可視化）
④ 機能安全の国際標準規格（IEC 61508[2]、ISO 26262[3]など）への準拠

ソフトウェアの不具合は、経年劣化をその原因とするハードウェアと異なり、条件の組み合わせで発現することが多い。そのため、リスクアセスメント手法をソフトウェア開発に適用する際には発想が重要で、その難しさから製品開発ではなかなか普及していないという課題があった。

東芝では、リスクアセスメント手法の１つとして FMEA（Failure Mode and Effect Analysis）[4]に着目し、この課題解決のためには不具合（FMEA では故障モードと呼ばれる[2]）の発想を促すキーワード(観点)が重要と考えた。その観点の目録〝観点リスト〞を発想の際に用いることで、ソフトウェアでのリスクアセスメント手法〝ソフトウェア FMEA〞の実施効果が高まることを確認してきた[5]。

そして、実際の製品開発においてより効率的・効果的にソフトウェア FMEA を行うことを目的に、製品分野を考慮した観点リストを用意する方法（手順）を開発した。

本稿では、東芝が開発してきたソフトウェア FMEA について述べるとともに、製品分野を考慮した観点リストを開発するための手順を説明する。そして、実際の製品開発におけるエンジニアの予測に基づいた評価と、実際に計測した不具合の発現密度からその効果を説明し、事業部門へ展開するための工夫や開発プロセスでの運用についても述べる。

---

[1] 事例提供: 株式会社東芝　余宮 尚志 氏

[2] 本稿において、以後は実際に市場などで発現する故障を不具合と呼び、製品開発の過程で導出する故障モードは通常の FMEA と同じ意味で用いる。なお、ソフトウェアでは故障モードと故障（不具合）を明確に区別しなくても良いと考えられる。

2. ソフトウェア FMEA の概要

2.1. ソフトウェア FMEA とは

ソフトウェア FMEA とは、東芝が開発・展開しているソフトウェア開発向けのリスクアセスメント手法である。一般に、FMEA はアーキテクチャ設計フェーズで行われることが多い[2, 3]が、ソフトウェア FMEA はソフトウェアの様々な開発フェーズで活用できる。

図 69-1 に、ソフトウェア FMEA の実施作業の流れを示す。部品定義から、対策までがソフトウェア FMEA シートに記載される。ソフトウェア FMEA では、ソフトウェアを構成する部品（後述）に対して、故障モードを〝なにが（部品の特性）〞、〝どうして（不具合発現のメカニズム）〞、〝どうなる（症状）〞の３つの属性で表現する。そして、故障モードの発想において、観点リストを用いている。

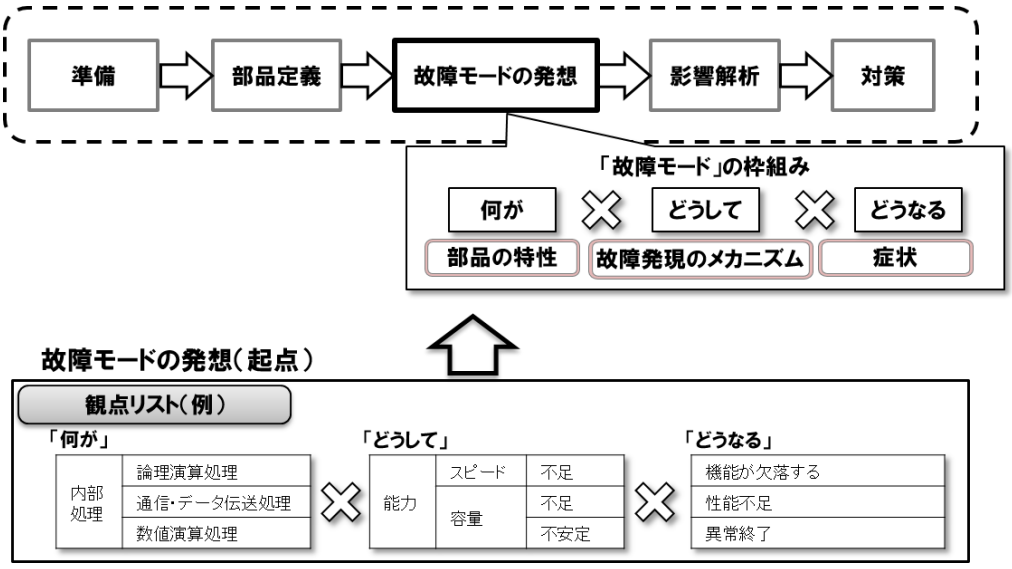

図 69-1　ソフトウェア FMEA の故障モードと観点リスト

各ステップについて詳しく説明する。

2.1.1. 準備

ソフトウェア FMEA を行うための２つの準備について述べる。１つ目がソフトウェア FMEA を実施するための戦略を立案することである。２つ目がソフトウェア FMEA の実施に必要な資料を準備することである。

戦略立案では、以下で説明する４つを考慮する必要がある。

(1)　実施フェーズの決定

ソフトウェア FMEA が実施可能な開発フェーズであるか、実施効果が高い開発フェーズかという視点で、実施フェーズを決定する。１つ目については、分散開発をしている場

合には、自部門で開発している範囲に実施フェーズがとどまる[3]。２つ目については、要求仕様やアーキテクチャの見直しを目的とするならば、少なくともその開発フェーズの後半か次フェーズの前半までに実施すべきである。設計の見直し（設計での見落としがないかの確認）を目的とするならば、設計後までに実施すべきである。検査項目の充実を目的とするならば、さらにそのあとのフェーズで実施することができる可能性がある。

(2)　部品粒度の決定

実施フェーズに、ある程度従属する形で決定できる。粒度は、ソフトウェア開発工程の V 字モデルで、下に行けばいくほど細かくなる。例えば、基本設計フェーズと詳細設計フェーズでは、基本設計フェーズの方が粒度として荒くなる。重要なことは、実施フェーズにおいて、故障モードが同定された時にその原因が分析できる程度に細かいことである。その程度内においては、リソースを考えなければ細かければ細かいほど良いと考えられる。特に、ソフトウェア FMEA をシステム開発工程で認識できる粒度で分析すると、ソフトウェア特有の故障モードにまで深堀できず、十分な分析とならないことがある。ソフトウェアでは、条件の組み合わせが不具合にいたる主要な原因であるが、システム開発工程ではソフトウェアにおける条件の組み合わせまで考慮できているとは限らないからである。一方で、ソフトウェア特有の細かい粒度になればなるほど発想にかかる時間は増してしまう。このように、粒度の決定は「どの粒度が最もソフトウェアの故障モードを洗い出す上で効果的か」を考える必要があるが、「かけられる時間やエンジニアの能力」も考慮する必要がある。

(3)　実施範囲の決定

開発者に選択の任意性がある場合にとりわけ重要である。開発するソフトウェア部品とその関連部品の全てを分析するに越したことはないが、リソースの増大を招く。そのため、次の 4 つに着目することにしている。

- 不具合の多発箇所
- 変更点・変化点
- クリティカルな箇所
- 設計が複雑な箇所（実装結果が複雑になることが想定される箇所）

不具合の多発箇所とは、過去の製品で良く不具合が発現している、品質が安定していない箇所である。もしそのような箇所が分かっている場合には、それを含む部品に着目すると良い。次に変更点とは、これまでの製品に対して仕様が変更されたものや、仕様・実装（結果）が変化した箇所である。変化点は、環境面と影響面がある。環境面とは、開発するソフトウェアの範囲外で起きる変化である。例えばシステムに搭載される OS が変更されたときにその影響を受ける範囲などである。影響面とは、開発するソフトウェアの範囲

[3] 発注側ならば、分散開発先にソフトウェア FMEA を実施してもらうことで、品質に対する取り組みを可視化することが可能である。

内で起きるある変化が、影響を受ける範囲などである。例えばあるソフトウェアが変更された時に、その影響を連鎖的に受ける部品である。クリティカルな箇所とは、特に不具合が許されない箇所、不具合発現時の影響が大きい箇所である。例えば生命に関わる、バックアップがない、後で改修ができない箇所などが考えられる。そして、設計が複雑な箇所（実装結果が複雑になることが想定される個所）に着目するのは、ソフトウェアの不具合は条件の組み合わせで発現することが多く、設計の複雑度は「想定漏れ」を生じやすいからである。不具合の多発箇所や変更点・変化点は過去のソフトウェアがあることを前提とするが、複雑性についてはそれを必ずしも前提としていないため、新規に開発するシステムやソフトウェアの場合にも検討できるのがメリットである。また、ソフトウェア FMEA を検査フェーズで用いる場合には、実装結果からの複雑度を考慮することも良いと考えている。

なお、準備の 1 つ目にあげたソフトウェア FMEA の実施に必要な資料を準備できたかどうかは、この実施範囲の決定時に確認すると良い。

(4) 実施者の決定

実施フェーズについて、技術的な内容を良く知るエンジニアが含まれていることが必要である。ソフトウェア FMEA の実施においては、大人数で実施すべきものではなく、適当な人数は 3 名から 7 名程度としている。これは、人数があまりに少ないと発想の広がりがないこと、逆に人数が多過ぎると工数がかかる割に発想が活発化しないためである。もし大きな部品で関係者が多い場合、人数を絞る方法としてはソフトウェア FMEA の実施範囲を分割することなどがある。ソフトウェア FMEA の実施者の中には、実施フェーズの内容を良く知るエンジニアだけでなく、その他の開発フェーズのエンジニアが含まれている方が良い。例えば、要求分析のフェーズでソフトウェア FMEA を実施する場合、その要求仕様から検査データを作成するエンジニアや、検査担当者などである。

上記の戦略立案については、システム開発者、ハードウェア開発者、関係部署、上長、場合によっては顧客などのステークホルダーと事前に（ソフトウェア FMEA の実施前に）検討し、合意しておくことが重要である。

例えば、ソフトウェア FMEA の実施範囲を決めたとしても、決定した実施範囲とは異なる範囲で不具合が（検査フェーズや市場で）発現することがある。決定した実施範囲とそれ以外の範囲とのインターフェース部分で不具合を生じることもあるし、ソフトウェアとハードウェアとのインターフェースで不具合が生じることもある。品質上、重要な範囲の認識が違うことによるトラブルを避けるためにも、実施範囲を事前に関係者間でレビューし、合意しておくことが重要と言える。これは、実施範囲だけではなく、実施フェーズ、粒度、実施者についても同じことが言える。

戦略立案における検討や合意を、ソフトウェア FMEA を実施するエンジニアだけが行う必要はなく、ここでは人数制限を設けずより多くのステークホルダーと協調する方が良いと考えて

いる。これは、効果的なソフトウェア FMEA の実施ができ、製品（システム）全体としての整合性を確保することにつながる。そして、合意エビデンスがあることによって品質を証明する説明責任を果たしやすくなる。さらに、開発期間に対するコミットメントを得ることにもつながりやすい。

2.1.2. 部品定義

ソフトウェア FMEA の分析対象となる部品を洗い出す必要がある。ソフトウェア FMEA では、部品を機能として考えることにしている。具体的には、機能名やモジュール名、タスク名をあげることになる。この粒度や範囲（すなわち分析対象範囲）は、上記の「2.1.1 準備」で決定しておく。

2.1.3. 故障モードの発想

ソフトウェアの機能に対して効果的な故障モードを列挙する。効果的とは、単なる機能の否定形ではなく、原因や対策に結びつく形で故障モードを表現することである。そして、過去に起きた不具合に対する故障モードだけではなく、未然防止につながる（未知の）故障モードを列挙していることである。対象とする製品分野やソフトウェア開発に精通したエンジニアであれば、知識や経験に基づいて効果的な故障モードを列挙できる可能性があるが、一般には困難であり、思考過程が残らないので十分性の判断や再評価を行うことができない。

ソフトウェア FMEA では、故障モードを３つの属性で表現する工夫を行っている。そして、故障モードを導出するための４つのパターンを定義している。

- **技術要素・環境基点**

  「なにが」を起点として「どうして」「どうなる」を列挙する方法である。ここでは、部品ごとに複数の「なにが」を、さらにそれぞれの「なにが」に対して複数の「どうして」「どうなる」を発想する。

- **機能不全基点**

  「どうなる」からさかのぼって「なにが」「どうして」を列挙する方法である。ここでも、部品ごとに複数の機能不全を発想する。例えば、機能欠落、不要な機能の混入、性能不足、異常終了、異なる機能の稼動などのパターンである。その機能不全ごとに複数の「なにが」「どうして」を発想する。

- **シナリオ基点**

  ユーザの利用シナリオから自由に類推する方法である。シナリオ基点では特に方法は定めていない。

- **事例基点**

  過去事例から類推する方法である。事例基点でも特に方法は定めていない。辞書的に過去事例を参照しながら類推することもあるし、自由に思い出された過去事例から類推することもある。

このうち、技術要素・環境基点と機能不全基点での故障モードの発想において、観点リストを

用いる。

2.2. 観点リストとは

観点リストとは、故障モードの発想を促すための技術的な観点(起点)を記述した目録である。図 69-2 は、観点リストの例である。

この組込みソフトウェア一般向け観点リストは、教育(「6 ソフトウェア FMEA の事業部門への展開」)の演習に用いる目的と、製品開発部門に対して、観点リストの例として示すことが目的で用意したものである。

観点リストは、製品分野を考慮したものを用意する方が、効果的かつ効率的な故障モードの発想につながる。製品分野を考慮した観点リストを用意する方法は「3. 製品分野を考慮した観点リスト」で述べるが、その際にも組込みソフトウェア一般向け観点リストを活用している。

ソフトウェア FMEA において、故障モードを列挙することができれば、その影響解析や対策を検討することは通常の FMEA と同様の方法で実施できるので、これらについての説明は本稿では省略する。

何が

技術要素・環境	
環境	制御対象ハードウェア
	OS
	メモリ・レジスタ
	CPU
	MCU
	ハードディスク
	通信ハードウェア
	開発環境
	COTS(商用MW等)
	通信
	DB
内部処理	例外処理
	異常発生時の処理
	データ操作
	書き込み処理
	読み出し処理
	論理演算処理
	通信・データ伝送処理
	割り込み処理
	ポーリング処理
	数値演算処理
	日付計算処理
	タスク制御
	スケジューリング
	文字コード
	分岐処理
	ループ
	初期処理
	終了処理
	再起動処理
	排他制御
	競合制御
	リトライ制御
	入力処理
	出力処理
	タイマ処理
	ノイズ除去処理
	共有データ: リスト・スタック
	共有データ: フラグ
	共有データ: パラメータ
インターフェース	外部I/F
	内部I/F
	ユーザーI/F
データ	
セキュリティ	

×

どうして

失敗メカニズムのパターン		
能力	スピード	不足
	容量	不足
		不安定
	性能	不足
		不安定
	サイズ	不足
	計算精度	不足
整合性	誤内容	
	欠落	
タイミング	速い	
	遅い	
	ずれる	
	重なる	
	ばらつく	
境界値	オーバーフロー	
	アンダーフロー	
条件	不足	
	余分	
	不整合	
処理順序	抜ける	
	逆になる	
	繰り返す(永久ループ)	
起動	しない	
	誤り	
	遅れる	
終了	しない	
	誤り	EOF/EOD
内部状態	変わってしまう	
	変わらなくなってしまう	
	内部状態間の不整合	
	外部状態との不整合	
外部状態	システムリセット	
	電源OFF	
	連続稼働時間(日跨り)	
	取扱説明書との整合性	
	電源サグ、電源事情(周波数・電圧)	
想定外の入力		

×

どうなる

機能不全のパターン
違う機能になってしまう
機能が欠落する
不要な機能が動く
性能不足
異常終了
DBを壊す
データを壊す

図 69-2　観点リストの例（組込みソフトウェア一般向け観点リスト）

3. 製品分野を考慮した観点リスト

3.1. 製品分野を考慮した観点リストの目的

組込みソフトウェア一般向け観点リストを用いても、ソフトウェア FMEA を実施することはできる[4]。その場合でも、ソフトウェアの品質が改善することを確かめている[5]。

東芝では、さらにソフトウェア FMEA による実施効果を高めるため、以下の２つを目的に製品分野を考慮した観点リストを用意する手順を開発した。

- 製品の特性によって起こりやすい不具合の再発・未然防止につながる
- 開発組織やエンジニアの特性によって起こりやすい不具合の再発・未然防止につながる

3.2. 製品分野を考慮した観点リストの開発手順

「3.1」の２つの目的を達成するためには、当該製品や関連製品において過去にどのような不具合や制限事項が、どのような原因によって発現しているのかを把握・分析する必要があると考えた。

そこで、過去に開発した製品で発現した不具合や制限事項をなぜなぜ分析[6]などを用いて真因解析し、観点リストで定義している３つの属性に従って表現することにした（下記の**手順(1)**に該当）。しかし、これを観点リストに反映するだけでは〝不具合の再発防止〞はできるが〝不具合の未然防止〞にはつながらない。そこで、以下の**手順(2)**以降を行うこととした。

手順(0) 組込みソフトウェア一般向け観点リストを用意する（「2.2 観点リストとは」）
手順(1) 過去に発現した不具合や制限事項を真因解析し、結果を３つの属性で表現する
手順(2) **手順(1)** に対する応用例・類推を考える
手順(3) **手順(1)** と**手順(2)** の結果を抽象化する
手順(4) 製品分野を考慮した観点リストに反映すべき結果を選択する
（抽象度の調整やグルーピングも行う）
手順(5) **手順(0)** における組込みソフトウェア一般向け観点リストを更新する

さらに過去の製品開発での経験や実績などをより活かし、リスクアセスメントとしての効果を高めるために、以下の内容も**手順(1)**に含めた。これは、故障モードの抜け漏れ防止には過去事例による知識とともに､エキスパートのノウハウや品質部門の知見を盛り込むことで後フェーズでの発見を防ぐ狙いがある。これらも観点リストにある属性で表現して、**手順(2)**以降を行った。

[4] 後述する HAZOP などで用いられているガイドワードでも、ソフトウェア FMEA は実施できる。ただし、実施効果は限定的と考えられる。

- 過去のソフトウェア FMEA の結果
- 設計やソースコードなどのレビュー記録
- エキスパートの経験や知見
- 関連製品の観点リスト（入手可能な場合）

エキスパートの経験や知見を引き出すための方法として、事例ベース（エキスパートが事例の解説を行いながら観点を抽出）、レビューベース（エキスパートがレビューで何を見ているか、なぜこのレビューコメントが出てきたかの解説を行いながら観点を抽出）、設計ベース（設計の手順をばらしていきエキスパートの考える注意点を抽出）、ブレーンストーミングベース（エキスパートが直接ノウハウを思いつくだけ追加）などをとっている。

また、関連製品の観点リストとは、同じ業界（同種のドメイン）の不具合や制限事項からの情報、社内における関連製品の観点リストなどを活用している。

3.2.1. 真因解析

過去の不具合の真因解析は、なぜなぜ分析などを用いて技術的な観点での掘り下げを行う。この時、「情報伝達ミス」「コミュニケーション不足」「コピー・アンド・ペーストミス」「ケアレスミス」などを含めてしまうと適切な故障モードにつながらず、影響や対策も技術的なものにならないので、真因解析の結果としては扱わない。

不具合管理がなされている場合でも、この真因解析が不十分となっていることがあり、実際の製品開発では最も時間と労力を費やすことが多い。

3.2.2. 応用例・類推と抽象度

手順(2) における応用例・類推では、可能な限り全てあげることになるが個数に決まりがあるわけではない。実際の事例では、不具合の種類にもよるが２つから４つの応用例・類推が挙げられることが多かった。

そして**手順(3)** においては、適度な抽象度が必要である。観点は、具体的過ぎると既知の不具合の再発防止にはつながるが観点リストのカバーできる故障モードが少なくなり、不具合の未然防止への効果が限られてしまう。一方、観点リストを抽象化し過ぎると、観点リストのカバーできる故障モードは多いが発想が難しくなる。

観点の抽象度を $x(x \geq 0)$、カバーできる故障モードの個数を $y(y \geq 0)$ とすると、y は x の関数 $y = f(x)$ となる（図 69-3）。観点の抽象度をあげれば発想できる故障モードの範囲が広がることから、$f(x)$ は単調増加である。また、ある抽象度 x_0 のカバーできる故障モードはは $(x_0, y_i)(i = 1,2,\cdots, f(x_0))$ と解釈することができる。真因解析のもととなった不具合を (α, β) とし、決定した観点の抽象度を $x_1(\alpha < x_1)$ とすると、(α, β) は範囲 $\{(x, y) \mid y \leq f(x_1),$ かつ $x = x_1\}$ に含まれるが、ソフトウェア FMEA を実施するエンジニアは（少なくとも）該当する過去の不具合 (α, β) を故障モードとして発想できる必要がある。その条件下において、x_1 はできるだけ大きな値を取ることが望ましい。この抽象度 x_1 は製品開発を行う組織やエンジニアに

よって異なる。

再発防止の抜け漏れを防止する対策は２つある。１ つは、低い抽象度で（やや具体的に）観点を記載しておく方法である。２つ目は、$\{x \mid x \le x_1\}$に含まれる観点（例えば x_{10}, x_{11}, x_{12}）を観点リストに全て記載しておくことである。**手順(1)** から得られた観点はこの中に含めておく。この２つは併用できる。

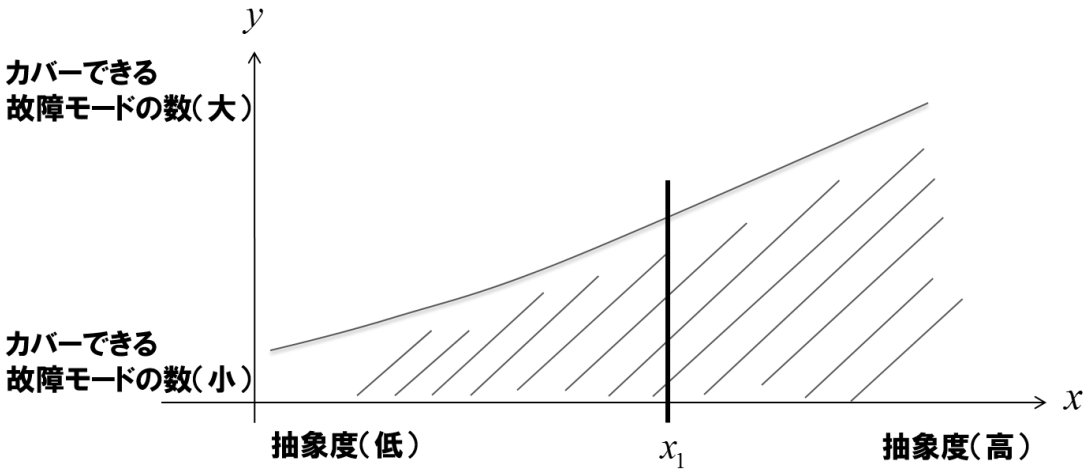

図 69-3　観点リストの抽象度と故障モードの範囲（イメージ）

２つ目については､観点をグルーピングして表現するようにしている｡観点のグルーピングは、発想能力に違いのあるエンジニアにも対応することができる。図 69-4 では〝なにが（部品の特性）〞に対するグルーピングの例を示しているが、〝どうして（不具合発現のメカニズム）〞、〝どうなる（症状）〞も同様にグルーピングする。

なにが（部品の特性）	
データ（抽象度 x_1）	音声データ（x_{10}）
	画像データ（x_{11}）
	文字コード（x_{12}）
通信・データ伝送処理	
：	

図 69-4　観点リストのグルーピング例

3.2.3.　観点リストの更新

観点リストに記載される観点は、厳密に決められた抽象度や数に決まりはない。これらは、組織やエンジニアの知識や経験に応じて最適なものを作り上げる必要がある。数については、あまり少な過ぎると効果が得られにくくなるし多過ぎると分析に時間がかかる。そのため、観点は追加するだけではなく適度に削除することで、効果を維持しながら作業効率の向上を図ることができる。分量的には、A4 で１枚程度におさまる範囲になるよう工夫している。

3.3. 製品分野を考慮した観点リストの開発例

以下の不具合事例に対して、「3.2」で示した開発手順に従った例を示す。考え方は不具合内容によらないため、分かり易さを重視した例となっている。

不具合事例：
あるシステムの入力値には文字コード体系 M を想定していたが、出力値が文字化けしてしまった。原因は、文字コード体系 N が入力値に混在されていたためだった。

手順(1) 過去に発現した不具合や制限事項を真因解析し、結果を３つの属性で表現する

なにが（部品の特性）	**文字コード体系が**
どうして（不具合発現のメカニズム）	**異なる**
どうなる（症状）	**文字化けする**

上記は、不具合の真因解析結果を直接表現したものである。不具合の真因解析が十分にされてさえいれば、**手順(1)** は非常に簡単である。

手順(2) **手順(1)** に対する応用例・類推を考える

なにが（部品の特性）	数値の表現方法が
どうして（不具合発現のメカニズム）	異なる
どうなる（症状）	意図しない計算結果になる

なにが（部品の特性）	国ごとの言語表現が
どうして（不具合発現のメカニズム）	異なる
どうなる（症状）	意図しない言語が表示される

応用例・類推は、考え得るものは可能な限り全てあげる。上記は一般的な理解しやすい表現にしてあるが、より具体的な表現が望ましい。例えば「意図しない…」と言う表現ではなく、「オーバーフローになる」、「アラビア語が表示される」などである。

手順(3) **手順(1)** と**手順(2)** の結果を抽象化する

なにが（部品の特性）	**表現が**
どうして（不具合発現のメカニズム）	**異なる**
どうなる（症状）	**意図しない結果が表示される**

手順(1) と**手順(2)** の結果を包含するよう抽象化する。抽象度については「3.2.2」にある通り、具体的過ぎず、かつ抽象化し過ぎないように配慮する。抽象化のし過ぎを防ぐために、例えばソフトウェア FMEA によるリスクアセスメントを実施するエンジニアが、もとの不具合に対応する故障モード「あるシステムの入力値には文字コード体系 M を想定していたが、異なる文字

コード体系 N が入力されたことで、出力値が文字化けがしてしまう」を発想できるかを確認する。

そして、**手順(4)** と**手順(5)** によって、組込みソフトウェア一般向け観点リストから製品分野を考慮した観点リストを作成する。グルーピングとしては、「表現」の階層下に「文字コード」や「数値」、「言語」などを表記することができる。

4. ソフトウェア FMEA の効果計測

東芝では、ソフトウェア FMEA を展開する中で実施効果を確認している。ここではその計測手段を述べ、その結果の一例を「5. ソフトウェア FMEA の評価事例」で紹介する。

4.1. 評価の考え方と評価方法

ソフトウェア FMEA の実施は、実施しない場合に比べると開発工数の増大につながる場合がある。そのため、ソフトウェア FMEA の実施が、ソフトウェアに対する不具合の再発防止や未然防止（「1.はじめに」の目的①）に効果があることを定量的に示すことは重要と考えている。

定量的な評価を行うためには、同じソフトウェア（部品）に対して、ソフトウェア FMEA を実施した場合と実施しなかった場合とで、発想した故障モードの個数や重要度にどの程度違いがあるかを計測することが理想である。さらに同じ経験や知識のあるエンジニアがそれを行うべきである。ところが、そうしたエンジニアが複数人いるとは限らず、かつ実際に開発を行っている製品ではこのような試行を行うことは開発コストの観点から難しい。

そこで、用意した代替え手段が次に述べる「4.2」と「4.3」である。

4.2. エンジニアの予測に基づいた評価方法

ソフトウェア FMEA を用いて製品開発を行ったエンジニアに対して、アンケートを行う。アンケートは以下の評価項目で、回答は製品開発の終了直後（最終試験後）に行う。

- **故障モードの発想密度（ソースコードのステップ数あたりの個数）**
  どの程度、故障モードを発想し事前に対策する機会が得られたか
- **新たに認識された故障モードの割合**
  どの程度、新たに（発想した故障モードを）事前に対策する機会が得られたか
- **市場流出が予想される故障モードの割合**
  どの程度、発想した故障モードが市場に不具合として流出することを防止できたか
- **実施工数**
  どの程度、ソフトウェア FMEA の実施に工数が必要だったか

この他、ソフトウェア FMEA の実施を通して感じたことを自由に回答してもらう。

4.3. 市場における評価方法

製品開発の終了後に、どの程度、ソフトウェア FMEA が対象とする不具合が発現したかで評価する。この評価は、開発規模（ソースコードのステップ数）に対する不具合の発現密度で評価する。この評価には、２つの方法がある。

A) 同じ製品での比較（異なる部品との比較）
同一製品の開発対象のうち、ソフトウェア FMEA を実施した部品と実施していない部品の不具合発現密度の比較

B) 同じ開発対象での比較（過去製品との比較）
ソフトウェア FMEA を実施した開発対象と、ソフトウェア FMEA を実施していない過去の同じ開発対象との不具合発現密度の比較

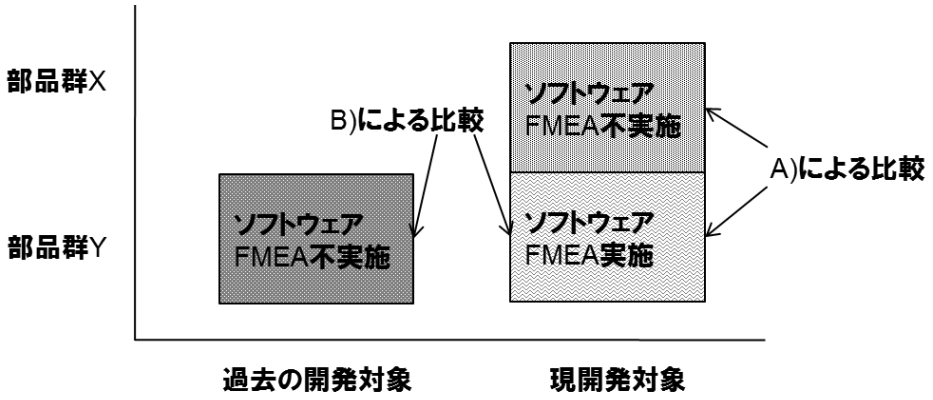

図 69-5　市場における評価方法

なお、開発規模が分からない部品（例えば他社開発製品）が含まれている場合には、検査項目数に対する不具合の発現密度を計って評価する。

5. ソフトウェア FMEA の評価事例

5.1. エンジニアの予測に基づいた評価結果

東芝におけるある実製品の開発で、「3.2」の手順に従って製品分野を考慮した観点リストを開発し、ソフトウェア FMEA によるリスクアセスメントを実施した。

そして、ここで取り上げる製品 P では、19 あるソフトウェア部品のうち 4 部品でソフトウェア FMEA によるリスクアセスメントを行った。この 4 部品を選定した理由は、過去機種からの変更点や変化点が多くリスクアセスメントの必要性が非常に高いと判断したためである（「2.1」）。 4 部品の総ステップ数は約 5300 である。エンジニアによる「4.2」で示した項目に対するアンケート結果を、表 69-1 にまとめた。

なお、アンケートは 7 人のエンジニアの結果をまとめたものだが、回答の内訳はエンジニアごとにばらつきが見られた。そのばらつきは、エンジニアが担当していた部品の違いやエンジニアの知識や経験の差と考えられる。

表 69-1　エンジニアによるアンケート結果

項目		結果
故障モードの認識密度		13.0 個/K step
新たに認識された故障モードの割合		44%
新たに認識された故障モード(44%)の内訳	設計段階	5%
	実装段階	10%
	テスト段階	74%
	市場流出	11%
実施工数		68 人時(7 人)

1 人時当たりの新たに認識された故障モードを計算すると、0.45 個と言う結果である。この数値は、組込みソフトウェア一般向け観点リスト（「2.2」の図 69-2）によるソフトウェア FMEA での実績値、0.39 個よりも 15%高い数値となっている。

また、アンケートの自由回答では以下のような内容があがった。これらは、ソフトウェア FMEA を実施しない製品開発に対し、より高い効果が実感できたことを推察できる内容と言える。

- 従来まで想定していなかった故障モードを発想できたことを体感できた
- （再度仕様を検討し）仕様を厳密にできた
- 開発途中の（修正による）部品の差し替え回数が減少した
- 仕様上の不備をマニュアルやカタログの関連個所に早めにフィードバックできた
- 他の自社開発部品に展開していれば品質向上が見込めると感じた
- 開発初期の段階で課題が抽出できたことが早期対策につながった
- 既製品にも展開したい

5.2. 市場における評価結果

製品開発における最終試験後から、3 ヶ月後のデータを集計した結果を表 69-2 にまとめた。この評価は、「4.3」 A)および B)の方法による評価で、製品 Q は過去に開発した製品 P と同じシリーズ製品である。**表 2** では、2 行目に部品数を示し、3 行目は部品の総ステップ数に対する不具合の発現密度を PPM（Parts Per Million）の値で示した。この PPM は、ソースコードのステップ数 100 万行あたりの不具合個数の割合を表す数値である。

表 69-2　市場における評価結果

	ソフトウェア FMEA 実施部品 (製品 P)	不実施部品 (製品 P)	不実施部品 （製品 Q)
部品数	4	15	2
不具合の発現密度 （PPM)	0	123	48

このように、製品分野を考慮した観点リストを用いてソフトウェア FMEA を実施した部品では、不具合が発現していない。他方、その不実施部品では 48PPM－123PPM の範囲で不具合が発現していた。すなわち、同観点リストを用いたソフトウェア FMEA の実施によって、不具合の再発防止と未然防止に効果があったことが推察できる。

6. ソフトウェア FMEA の事業部門への展開

東芝では、ソフトウェア FMEA を製品開発に適用する前に、当該の事業部門に対して半日から 1 日の教育を行っている。教育では、ソフトウェア FMEA の概要を説明するとともに、簡単な演習を行って分析方法の習得を行っている。この教育によって、多くの受講者が観点リストを用いたソフトウェア FMEA の実施効果を実感している。

その後、実際に製品分野を考慮した観点リストを開発するのは、はじめての事業部門やエンジニアだけでは難しいことが多い。そのため、製品分野を考慮した観点リストの開発からソフトウェア FMEA の試行まで、一週間程度のコンサルティングを行ってはじめて取り組む事業部門やエンジニアを支援する枠組みを用意している。コンサルティングでは準備段階として、ソフトウェア FMEA 実施範囲の決定や実施メンバの選定などに対する助言から、なぜなぜ分析による過去の不具合事例の分析についても支援している。故障モードの発想は製品開発担当のエンジニアが行うが、慣れていないうちは発想が広がらないことが多々ある。そのような場合には、発想のヒントを与えて発想を手助けしている。またソフトウェア FMEA 実施後の評価、振り返りや開発プロセスでの運用（「7. ソフトウェア開発プロセスでの運用」）など、継続して、より効果的・効率的にソフトウェア FMEA が実施できるよう、アフターフォローを行っている。

さらに、ソフトウェア FMEA だけでソフトウェアの品質が担保されないよう、開発プロセスやガイドライン、チェックリストなどの品質向上を行いながら、不具合の削減に努めている。

7. ソフトウェア開発プロセスでの運用

ソフトウェア FMEA の効果と効率をより高めるには、ソフトウェア開発プロセスの中に観点リストのメンテナンスを織り込んでいくことが重要である。ソフトウェア FMEA によるリスクアセスメントを実施したにも関わらず、その対象製品（部品）において市場で発現してしまった不具合は、観点リストを用いても故障モードとして発想から抜け落ちてしまったものと言える。

将来、不具合の再発を防ぎ、そして未然防止の効果を高めるためには「3. 製品分野を考慮した観点リスト」で示した手順を繰り返すことによる観点リストの更新が必要となる。

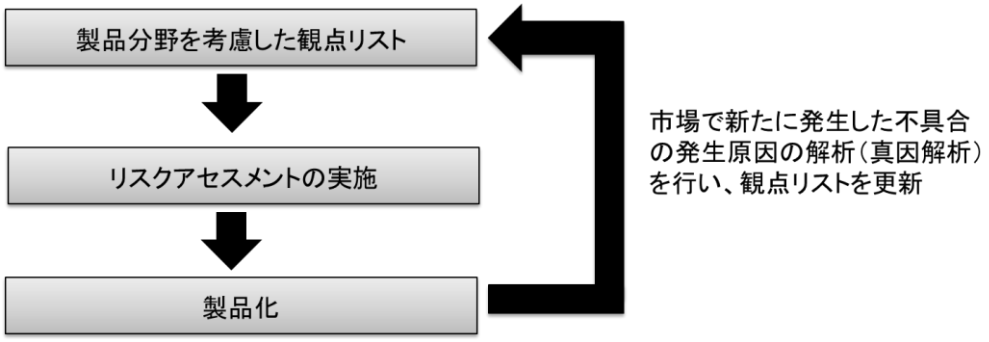

図 69-6　観点リストの更新

図 69-6 は、製品分野を考慮した観点リストをはじめに用意した場合を表している。初期の開発コストを抑えるために、組込みソフトウェア一般向け観点リストを用いてソフトウェア FMEA を実施し、その後に発現した市場での不具合をもとに、製品分野を考慮した観点リストとして改良していく場合もある。「3. 製品分野を考慮した観点リスト」で述べたように、この手順であってもソフトウェア FMEA はソフトウェアの品質向上に有益である[5]。

8. 関連技術

FMEA 以外に、リスクアセスメント手法として古くから用いられるものに、FTA（Fault Tree Analysis）[7]、HAZOP（Hazard and Operability Study）[8]がある。

FTA は、機能不全などの発現が好ましくない事象について、発現経路、発現原因および発現確率を故障（不具合）の木を用いて分析する。FTA は、望ましくない事象に対する要因を探るのには適しているものの、事象を網羅的に扱うのが難しいという特徴がある。ソフトウェア開発でも用いられる。ボトムアップ型の FMEA に対して、トップダウンの分析方法であり、ソフトウェア FMEA とは目的に応じて使い分けるべきである。

HAZOP は設計意図からのずれをもれなく洗い出すための分析手法である。特徴としてはガイドワードを用意していることである。それを、対象とする製品の設計・運転パラメータおよび操作とを組み合わせることにより、設計意図からのずれを見つける。ガイドワードは、ソフトウェア FMEA の観点リストにおける観点と考え方が似ている。違いとしては、ガイドワードは主にパラメータに着目しておりいくつかの種類（多くは 11 種類）を定めているが、ソフトウェア FMEA における観点は製品分野ごとに（11 種類より多くを）定める点である。

また、最近は STAMP/STPA[9]が注目されている。STAMP/STPA でも 4 つのガイドワードを定義しているが、それらはハザード要因に対する網羅性が高い反面、発想が難しいと言われている。本稿で述べた観点リストの開発手順は、ガイドワードをより効果的に拡張できる方法として利用できる可能性があると考えている。

9. 今後の展開

本稿では、東芝が製品開発で用いているソフトウェア FMEA の手法と、そこで特に重要となる製品向け観点リストの開発手順を説明した。また、実際の製品開発でソフトウェア FMEA を実施した結果として、エンジニアの予測に基づいた評価結果と市場における評価結果を説明し、事業部門へ展開するための工夫や開発プロセスでの運用についても述べた。

本稿で紹介した、製品分野を考慮した観点リストによるソフトウェア FMEA は、ソフトウェア開発における故障モードの発想を効率的・効果的に行うことにつながっている。

ソフトウェア開発では、FMEA に限らず FTA や HAZOP、近年では STAMP/STPA など様々なリスクアセスメント手法が利用されている。それらの選択は適用する開発フェーズや目的によって最善のものが選択されるべきだが、特にソフトウェアでは条件の組み合わせが多いことによる抜け漏れが生じやすく、どの手法に対しても発想が必要となる。その発想において、本稿で説明した（製品分野を考慮した）観点リストの考え方が利用できる。特に、FMEA においては本稿におけるソフトウェア FMEA の手法が有益と言える。

今後は、リスクアセスメントの効果と効率性から、観点の抽象度に対する基準をより明確にする。そして、他のリスクアセスメント手法に対しても適用を進めることで、ソフトウェア品質、さらにはシステム品質、製品品質の向上をより確実にする予定である。また、セーフティとセキュリティ分野への応用も進めていく予定である。

参考文献

[1] 夏目珠規子ほか：ソフトウェア開発における FMEA の適用可能性検討、第 41 回信頼性・保全性シンポジウム発表報文集、p.359-364、2011

[2] IEC 61508 Ed.2.0 : Functional Safety of Electrical/Electronic/ Programmable Electronic Safety - related Systems, 2000

[3] ISO 26262 : Road Vehicles – Functional Safety, 2011

[4] IEC 60812 Ed.2.0 : Analysis techniques for system reliability – Procedure for failure mode and effects analysis, 2006

[5] 余宮尚志ほか：組込みソフトウェアに対するソフトウェア FMEA の試行実験とその考察、情報科学技術フォーラム講演論文集、12th 巻 第 1 分冊、p.283-284、2013

[6] 大野耐一：トヨタ生産方式 脱規模の経営をめざして、ダイヤモンド社、1978

[7] IEC 61025 Ed2.0 : Fault tree analysis, 2006

[8] IEC 61882 : Hazard and operability studies – Application guide, 2001

[9] はじめての STAMP/STPA～システム思考に基づく新しい安全性解析手法～、独立行政法人情報処理推進機構（IPA）、2016

[10] 余宮尚志ほか：不具合リスク発想のための観点の抽出方法とその効果、ソフトウェア品質シンポジウム 2016（経験論文）、2016