不具合リスク発想のための観点の抽出方法とその効果

The extraction method of viewpoints for imagining the failure risks and its effectiveness

株式会社東芝 IoT テクノロジーセンター プロセス・品質技術開発部
Toshiba Corporation, IoT Technology Center, Process & Quality Technology Department
〇余宮　尚志　　小島 昌一 [1)]
〇Hisashi Yomiya　　Shoichi Kojima [1)]

Abstract Toshiba developed software FMEA, a method for risk assessment which enables recurrence prevention and proactive measure of software failures in product quality. Effectiveness of software FMEA using general viewpoints list for embedded software is already verified though, we expect using viewpoints list for product specific is more efficient and effective. This paper clarifies the procedure of setting up viewpoints list for product specific and shows its effectiveness based on engineer's prediction and density of failure in actual product development.

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を目的に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]等）への準拠

リスクアセスメント手法をソフトウェアに適用する際には、不具合の発想が難しいという課題がある。東芝では、リスクアセスメント手法として FMEA[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）の発想を促すキーワード（観点）が重要と考え、組込みソフトウェア一般向け観点の目録〝観点リスト〟を用意した。そして、それ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た[5]。

一方、実際の製品開発では、組込みソフトウェア一般向けの観点リストでは製品の特性に応じた故障モードの発想につながりにくい言う新たな課題があった。そこで、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。

本論文では、製品分野を考慮した観点リストを開発するための方法を述べる。そして、実製品の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、その効果を述べる。

株式会社東芝 IoT テクノロジーセンター プロセス・品質技術開発部
Toshiba Corporation, IoT Technology Center, Process & Quality Technology Department
神奈川県川崎市幸区小向東芝町 1　Tel: 044-549-2475　e-mail: hisashi.yomiya@toshiba.co.jp
1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, Kanagawa Japan

1)　東芝ソフトウェア・コンサルティング株式会社 Toshiba Software Consulting Corporation

2. ソフトウェア FMEA と観点リスト

2.1. ソフトウェア FMEA

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。

ソフトウェア FMEA では、FMEA における部品を機能と捉える。機能ごとに故障モードを列挙することができれば、その影響や原因、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。よって重要なことは、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ことである。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（既知の）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技術者であれば、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。本手法ではこうした課題を解決し、ソフトウェアに対して体系的に故障モードを列挙することを実現したものである。

本手法では、ソフトウェアを構成する各機能に対して、故障モードを〝なにが（部品の特性）〞〝どうして（故障発現のメカニズム）〞〝どうなる（症状）〞の 3 つの属性で表現する。そして、故障モードを導出するための 4 つのパターンを定義している。1 つは、「なにが」を起点として「どうして」「どうなる」を列挙する方法、もう 1 つは「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。さらに、ユーザの利用シナリオから類推する方法と、過去事例から類推する方法である。

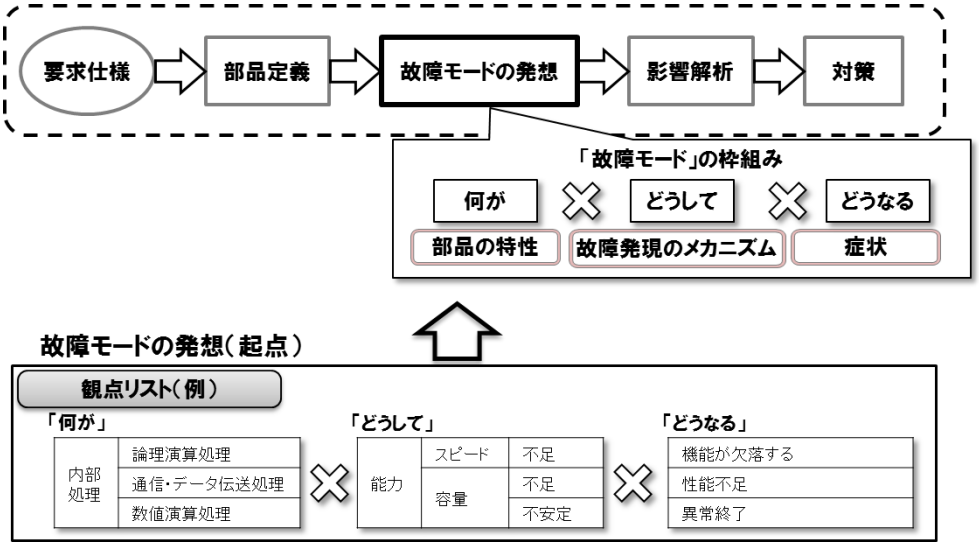

図 1. ソフトウェア FMEA の故障モードと観点リスト

2.2. 観点リスト

ソフトウェアにおける故障モードの導出には発想が重要となる。そのための手段として、東芝では観点リストを用いている。観点リストとは、故障モードを発想しやすくするための技術的な観点（起点）を記述した目録である。

観点リストは、組込みソフトウェア一般向け観点リストをそのまま用いるよりも製品分野を考慮したものになっている方が、効果的かつ効率的な故障モードの導出につながることが期待できる。そこで、製品分野を考慮した観点リストを用意する方法を開発した。

3. 製品分野を考慮した観点リストの開発方法

3.1. 開発方法

製品分野を考慮した観点リストの開発には、当該製品や関連製品において過去にどのような不

具合が、どのような原因によって発生しているのかを把握・分析する必要があると考えた。それは以下の理由によるものである。

- 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

そこで、過去に開発した製品で発生した不具合をなぜなぜ分析等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。

(0) 組込みソフトウェア一般向け観点リストを用意する（開発済み）
(1) 過去に発生した不具合を真因解析し、その結果を 3 つの属性で表現する
(2) 手順 **(1)** に対する応用例・類推を考える
(3) 手順 **(1)** と手順 **(2)** の結果を抽象化する
(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）
(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等をより活かし、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。

- 過去のソフトウェア FMEA の結果
- 設計やソースコード等のレビュー記録
- 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）

3.2. 応用例・類推と抽象度

手順 **(2)** における応用例・類推は、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・類推があ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。3.1 の方法によって開発された観点リストは、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不具合が少なくなり、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不具合は多いが発想が難しくなる。

観点の抽象度を $x(x \geq 0)$、カバーできる不具合の個数を $y(y \geq 0)$ とすると、y は x の関数 $y = f(x)$ となる（**図 2**）。観点の抽象度をあげれば発想できる不具合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 のカバーできる不具合は $(x_0, y_i)(i = 1,2,\cdots, f(x_0))$ 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、$\{x \mid x \leq x_1\}$ 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けばよい。この 2 つは併用できる。

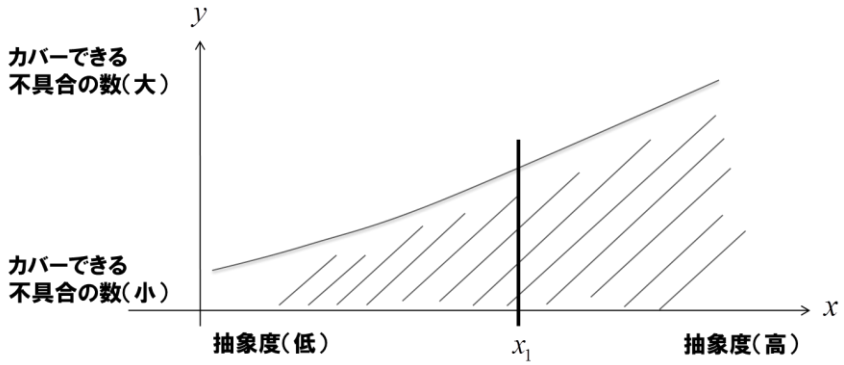

図 2. 観点リストの抽象度と不具合の範囲（イメージ）

2 つ目については、観点をグルーピングして表現することにしている。**図 3** 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（故障発現のメカニズム）〟〝どうなる（症状）〟も同様にグルーピングできる。

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 3. 観点リストのグルーピング例

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。

不具合事例：
あるシステムの入力値には文字コード体系 A 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 B が入力値に混在されていたためだった。

(1) 過去に発生した不具合を真因解析し、その結果を 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（故障発現のメカニズム）	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いれば手順 **(1)** は非常に簡単である。

(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（故障発現のメカニズム）	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（故障発現のメカニズム）	**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」等である。

(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（故障発現のメカニズム）	**異なる**
どうなる（症状）	**意図しない結果が表示される**

(1) と **(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 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の実施者が、もとの不具合「あるシステムの入力値には文字コード体系 A を想定していたが、異なる文字コード体系 B 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」等を表記することができる。

4. 製品分野を考慮した観点リストを用いたソフトウェア FMEA の評価方法

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て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.** 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られる。

定量的な評価を行うためには、同じ部品（ソフトウェア）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の 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、アンケートを行った。製品開発の終了直後（最終試験後）に、以下の評価項目を回答してもらった。

- 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
  どの程度市場に不具合が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った。

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した。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ととした。この評価には、2 つの方法が考えられた。

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

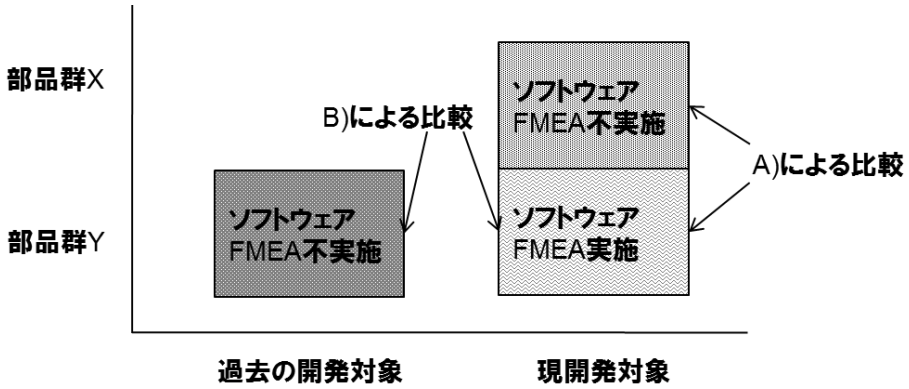

図 4. 市場における評価方法

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ることが考えられる。

5. 製品分野を考慮した観点リストを用いたソフトウェア FMEA の評価結果

5.1. エンジニアの予測に基づいた評価結果

東芝における実製品の開発で、3.1 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。

評価対象となった製品 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が高いと判断したためである。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 1 にまとめた。アンケートの時点では、市場で実際に不具合が発生したわけではなく不具合となる可能性にとどまるため、ここでは故障モードのことを不具合リスクと表記する。

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。

項目		結果
不具合リスクの発現密度		13.0 個/K step
新たに認識された不具合リスクの割合		44%
新たに認識された不具合リスク(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)

表 1. エンジニアによるアンケート結果

1 人時当たりの新たに認識された不具合リスクを計算すると、0.45 個と言う結果が得られた。この数値は、東芝で従来から行っていた組込みソフトウェア一般向け観点リスト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、自由回答では以下のような内容があがっていた。これらは、組込みソフトウェア一般向け観点リストのとき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（不具合リスク）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 2 にまとめた。この評価は、4.3 A)及び 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。

	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

表 2. 市場における評価結果

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、不具合が発現しなかった。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た。

6. 品質改善プロセスへの適用

本手法の効果をさらに高めるには、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組み込んでいくことが重要と考えられる。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、その対象部品において市場で発生した不具合は、観点リストを用いても

発想から抜け落ちていたものである。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.**で示した方法を繰り返すことによる観点リストの更新が必要である。

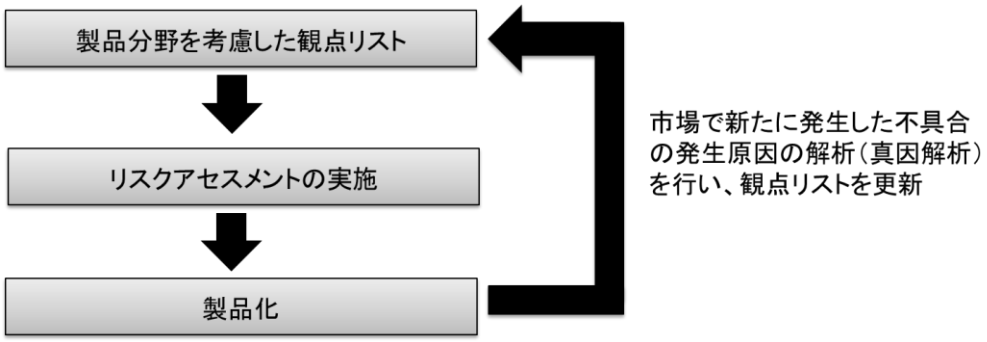

図 5. 観点リストの更新

7. まとめと今後の展開

東芝では、観点リストを用いたソフトウェア FMEA によるリスクアセスメントによって、ソフトウェアの品質を高めている。そして、より効果的かつ効率的に故障モードの発想ができることを期待し、製品分野を考慮した観点リストを用意する方法を開発した。本論文では、実際の製品開発における同観点リストを用いたソフトウェア 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含めて報告した。

エンジニアの予測では、従来の組込みソフトウェア一般向け観点リストに比べて、製品分野を考慮した観点リストでは新たに認識された不具合リスクの数が 1 人時当たり 15%高い数値となった。また、市場に流出してしまう可能性のある不具合リスクが、開発途中で 11%抽出できている。

そして、最終試験後 3 ヶ月による実績では、市場においてソフトウェア FMEA によるリスクアセスメントを実施した部品に不具合が発現していない。ソフトウェア FMEA の不実施部品では 48－123PPM の範囲で不具合が発現していることからすると、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結果が得られた。

ソフトウェア開発では、FMEA に限らず、FTA[6]や HAZOP[7]等の様々なリスクアセスメント手法が利用できる。それらの選択は適用する開発工程や目的によって最善のものが選択されるべきだが、ソフトウェア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、いずれに対しても発想が必要である。本研究では、ソフトウェア FMEA による観点リストの活用について報告したが、今後は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、ソフトウェア品質の向上をより確実にする予定である。そして、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明確にし、開発プロセスへのより良い適用も目指していく。

8. 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] IEC 61025 Ed2.0 : Fault tree analysis, 2006
[7] IEC 61882 : Hazard and operability studies – Application guide, 2001