ソフトウェア開発における FMEA適用に関する考察

JaSST'09 Tokyo
2009年1月29日(木)
電気通信大学大学院
河野 哲也

プレゼンテーションの流れ

- デモンストレーション
- FMEAの考え方
- ソフトウェア開発におけるFMEA適用
- まとめ

FMEAとは

- 故障モード・影響解析(Failure Mode and Effects Analysis の略)
  - 過去の故障を分析して、再発防止/未然防止を行う
  - 下位アイテムから上位へ影響解析を進めるのでボトムアップである
  - 大きく故障モードと影響の解析の2つ視点に分けることができる
- 故障モード(主にシートの左側)
  - 故障モードとは、故障の一般的現象を表す言葉
    - 例えば、脱落、ずれ、さび、破損 など
  - 故障モードをテコにし解析対象アイテムに発生する故障を予測する
- 影響解析(主にシートの右側)
  - 予測した故障がどのような影響があるのかを解析する
  - 予測した故障が上位アイテムにどのような影響があるかを解析する
- 本発表では、ワークシートの左側に焦点をあてる
  - ただし、FMだけではなくEAも含めて議論する

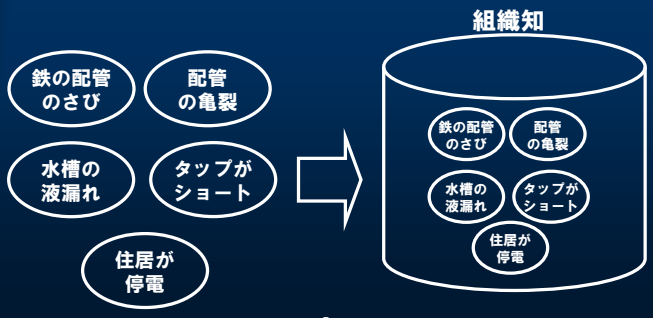
過去の故障の分析

- 過去の故障を分析しFMEAで使える情報を得る
  - 主に故障モードを抽出する

鉄の配管のさび	配管の亀裂	水槽の液漏れ	タップがショート	住居が停電
6月12日 一般住居の水道の配管がさびが原因であった	12月12日 一般住居の水道の配管に亀裂が入っていた	8月1日 貯水用水槽からの液漏れを確認した	7月2日 雨漏りによりテーブルタップがショートした	9月20日 そういえば自宅が停電した

故障モードの収集

- 分析結果である故障モードを収集し組織的に使えるようにする

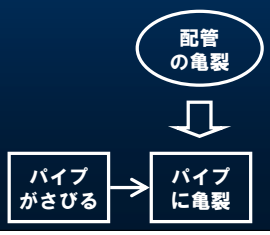


工場の開発における配水用パイプの設計

- 工場の開発における配水用パイプの設計
  - パイプの材質を鉄に指定した場合
    - 鉄に指定したことにより、配水用パイプに要求されている機能が達成できるかを考えると同時に鉄に指定したことによって、何らかの問題が起きないかを検討する
- 故障モードをテコに発生しうる故障を予測する
  - 「鉄の配管のさび」から「パイプのさび」が予測できる

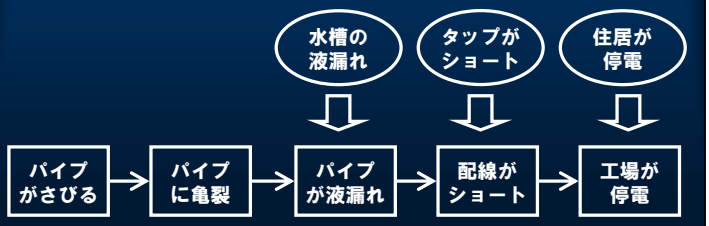
設計におけるFMEA

- 予測した故障の影響の解析を行う
  - パイプがさびることによる影響を検討する
- 故障の影響によって上位アイテムもしくは同位アイテムにさらに故障が発生しないか検討する
  - パイプがさびたことによって、パイプに亀裂が入る



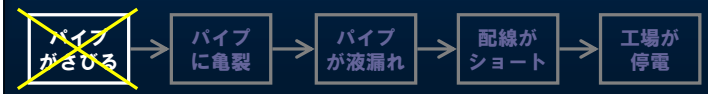
設計におけるFMEA

- さらに、影響の解析を進める
  - パイプの亀裂によって、液漏れが発生する
  - 液漏れによって、配線がショートする
  - 配線のショートによって工場が停電する



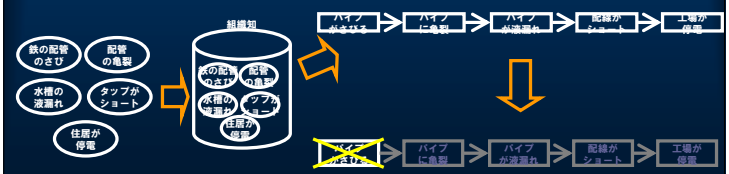
設計におけるFMEA

- 設計案の修正
  - パイプの材質を鉄に指定するという設計案を修正する
  - 工場が停電することを未然に防止できた
    - 過去に起こった工場の停電を分析したわけではないので再発防止ではない
  - 上位システムへの影響が致命的ではなければ設計案は修正しなくて良い
    - 故障が影響しないような仕組みを設ける場合もある



おさらい：FMEAの考え方

- FMEAのキーコンセプト
  - 過去の故障を分析して、再発防止 / 未然防止を行う
  - 下位アイテムから上位へ影響解析を進めるのでボトムアップである
  - 故障モードと影響解析の2つ視点に分けることができる



プレゼンテーションの流れ

- デモンストレーション
- FMEAの考え方
- ソフトウェア開発におけるFMEA適用
- まとめ

ソフトウェア開発におけるFMEA

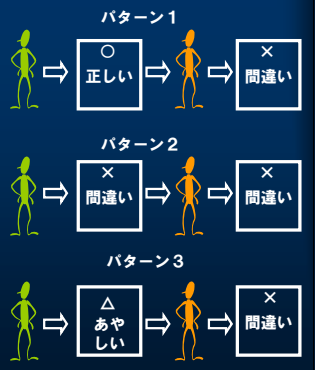
- ソフトウェア開発におけるFMEA適用の現状
  - 上手くやれてます、という話はそう多く聞かない
    - 単純に考え方が難しい
    - 時間がない、コストがかかる
    - 故障モードが存在しない、もしくは分からない
  - ワークシートに従った適用している現場が圧倒的に多い
    - 発生する可能性のある故障の網羅性よりもリスク評価にやたらこだわる
- 一度、ソフトウェア開発におけるFMEAの適用を見直す必要がある
  - まず、ソフトウェア開発の特徴を整理する
  - その特徴に基づいてFMEA適用について議論する

ソフトウェア開発の特徴

- 開発の全ては知的変換作業である
  - 各工程は、ある成果物が入力され違う側面の成果物が出力されるという変換作業である
  - その変換作業は開発者の知的作業によって実現される
    - 分かりやすくいうと、大規模な伝言ゲーム
- 開発の上工程の全てに設計行為が介在している
  - 要求分析、仕様策定、設計、実装の工程の全ては広義の設計行為である
    - 実装においても、満たすべき要求や達成すべき課題に対して特定の解を導き出すという設計行為を行っている

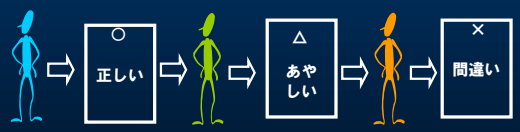
ソフトウェア開発特有の知的変換作業

- ハードウェアにおけるFMEAとは違う視点が必要
  - 例えば、伝言ゲームでは次の3つのパターンでの間違いがある
    - パターン1：正しい⇒間違い
    - パターン2：間違い⇒間違い
    - パターン3：あやしい⇒間違い
  - 自分の伝言の正しさの有無だけではなく伝え手が誤解しないかどうかも気遣う必要がある
    - つまり、×ではなく△があるかどうかも気にしないといけない



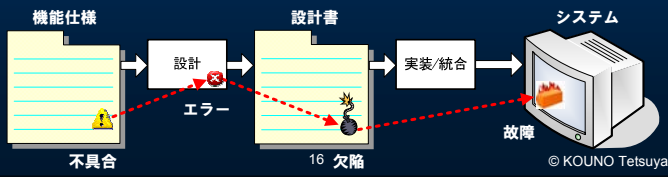
失敗パターンの拡張

- 伝言ゲームの失敗パターン
  - ○から△、△から×に変遷する
  - ×はすぐ見つけれるが、△は見つけるのが難しい



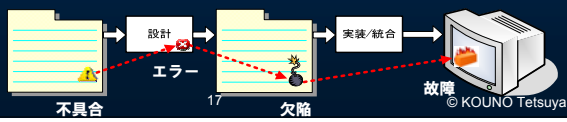
不具合から故障発生までの流れ

- 不具合から故障発生までの流れを整理する
  - 一言にバグと言っても、人それぞれ捉え方が違う
    - 故障：システムなどが要求された機能を実行できない状態
    - 欠陥：要求した機能を実行できなくする開発成果物の不備や誤記(仕様書やプログラムの誤記など)
    - エラー：間違った結果をうみだす人間の行為
    - 不具合：エラーを発生させてしまう開発成果物の望ましくない形態(欠陥が作り込まれるリスク、つまり欠陥リスクともいえる)



知的変換作業における考慮点

- 次工程の開発者を気遣う
  - 次工程の開発者のスキルやクセの把握し設計を改善する
  - 他の成果物と組み合わせた場合に次工程の開発者に混乱をうまないかを確認する
- 不具合ビューによるセルフレビュー
  - 欠陥があるかどうかの確認だけでなく次工程で欠陥を作らせるような内容がないか確認する必要がある
    - 日本語が正しいか？誤解をうまないか？図が複雑ではないか？
- 不具合のパターンを不具合モードとして整理する
  - 不具合モードによって、自分の成果物に不具合があるかどうかを確認する
  - 同様に、欠陥モード、故障モードも整理する必要がある



ソフトウェア開発の特徴

- 開発の全ては知的変換作業である
  - 各工程は、ある成果物が入力され違う側面の成果物が出力されるという変換作業である
  - その変換作業は開発者の知的作業によって実現される
    - 分かりやすくいうと、大規模な伝言ゲーム
- 開発の上工程の全てに設計行為が介在している
  - 要求分析、仕様策定、設計、実装の工程の全ては広義の設計行為である
    - 実装においても、満たすべき要求や達成すべき課題に対して特定の解を導き出すという設計行為を行っている

全ての工程は設計行為である

- 要求分析、仕様策定、設計、実装の工程の全てにFMEAという考え方が適用できる
  - 各工程でワークシートを作成すると大変なので思考的にFMEAを実施する必要がある
    - バグの少ない開発者は恐らくやっているだろう
  - 例えば、仕様策定工程において設計対象で指定したソフトウェア構造が対象以外の機能で使われる可能性があるのか、対象以外からどのような影響があるのかを考える
    - もちろん、求められた要求はクリアする
    - さらに、設計で指定した構造やアルゴリズムによって顧客価値を損失させるような問題が発生可能性を検討する

体系的にFMEAを実施するには

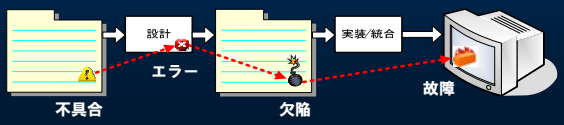
- 設計対象が対象外からどのような影響を受けるのかを事前に知っておく必要がある
  - 出荷後の故障の発生の予測
    - 対象システムや製品がどのような外部環境におかれるのか
      - 他のシステムや製品と組み合わされるのか？どんな環境なのか？
      - どのような使われ方をするのか？誰が使うのか？
    - 設計対象がどのような影響を受けるのかを考慮し故障モードより故障を予測する
  - 開発中の故障の発生の予測
    - 設計対象がソフトウェア構造のどの部分に位置するのか
      - 設計対象がどのような機能やコンポーネントと関係があるのか？
      - もし自分が欠陥を作り込んだとしたらその影響が把握できるのか？
      - 全体を見るのが困難な場合は機能分割などで周りくらいは見えるようにする
    - 設計対象の欠陥モードより欠陥を予測しその影響を解析する

おさらい：ソフトウェア開発におけるFMEA

- 知的変換作業
  - 不具合ビューによるセルフレビュー
    - 次工程を気遣う
  - 不具合のパターンを不具合モードとして整理し、セルフレビューに使う
    - 不具合モードによって、自分の成果物に不具合がなるかどうかを確認する
- 設計行為
  - 出荷後の外部環境の特性の徹底的な洗い出し
    - どこで？誰が？どのように？何の目的で？
  - ソフトウェア構造の整理とその情報の共有
    - 設計者が担当している設計対象がどこに位置するのかが把握できるようにする
- 問題知識の整理
  - 影響の解析を行う際、不具合や欠陥、故障が想起できるようにそれぞれのモードを整理する
  - 開発成果物ビューのアンチパターンを検討し、整理する

テストで何ができるのか

- 故障ドリブンで上流の改善に取り組む
  - テストは故障を出すプロの集まり
    - 故障検出をきっかけに欠陥や不具合を明らかにする
    - 欠陥や不具合を整理して、開発のクセをパターン化する
      - 開発でのFMEAを支援する活動
- 欠陥モードや不具合モードで故障を効果的に検出する
  - 仕様書などの上流成果物の不具合、欠陥よりピンポイントテストを行う
    - ピンポイントテストの観点をもとに上流の改善につなげる



プレゼンテーションの流れ

- デモンストレーション
- FMEAの考え方
- ソフトウェア開発におけるFMEA適用
- まとめ

まとめ

- ソフトウェア開発においてFMEAを効果的に適用できているところはそう多くない
  - ハードウェアでもFMEAをきちんと適用するのは難しい
  - ソフトウェア開発ではFMEAを見直す必要があるのではないだろうか
    - ソフトウェアは目に見えないから、経年劣化がないから、故障モードが存在しない
      - FMEAのうまみを探そう
- ソフトウェア開発でFMEAを適用するための素地を提示した
  - ソフトウェア開発の特徴を整理し、それに基づいてFMEA適用について議論した
- 欠陥の作り込み防止や故障の未然防止にはFMEAという考え方は有効である
  - ただし、課題は山積している
  - FMEA適用を支援するために現場の視点で地道に解決していくしかないだろう