大規模ソフトウェアの保守開発を対象とした故障モード影響解析（FMEA）適用の試み

Investigation on Failure Mode and Effect Analysis on Large-Scale Software Evolutionary Development

山科隆伸，森崎修司

要　約　保守・派生開発型ソフトウェアでは，既存部分の整合性を満たしながら機能追加，改変をする必要がある．既存部分が大きくなるにつれてその制約条件が増え，機能追加や改変が難しくなる．本稿では，ハードウェアの信頼性向上の手法である故障モード影響解析を大規模な保守開発ソフトウェアに適用評価した結果を紹介する．評価において，同一ソフトウェアの新しいバージョンのテスト工程で発見された不具合の中には，古いバージョンを基に抽出した故障モードを用いることで未然に防止できるものがあったことを確認できた．さらに，故障モード影響解析を実際に適用するには煩雑な作業が伴うため，これを支援するツールを開発した．支援ツールは過去の不具合に基づく影響度の自動算出や開発担当者への自動振分けができ，品質向上や適用コスト低減につながる．

Abstract　Evolutionary software development requires adding and changing functionalities without losing consistency of existing part. The larger an existing part grows, the harder maintaining consistency of existing part becomes. In this paper, we introduce an empirical evaluation of applying FMEA (Failure Mode and Effect Analysis) to huge evolutionary software development. The evaluation result shows that our approach works well. We also introduce a support tool for FMEA to evolutionary software development. The support tool provides improved quality and reduced costs by automatic calculation of risks of failure modes and automatic assignment to developers.

1. はじめに

近年多くのソフトウェア開発が保守開発や派生開発といった既存ソフトウェアに機能を追加・改変しながら開発する形態で行われている．このようなソフトウェア開発においては，機能を追加する部分や改変する部分が期待通り動作するだけでなく，既存部分との整合性を保ち，既存部分も期待通り動作することが求められる．一般に，機能追加や改変の際には，それらの制約条件を満たしながら追加・改変部分の設計や実装をする必要があり，既存部分の規模が大きくなれば制約条件もそれに従い著しく増加する．

著者らは，ハードウェアを対象とした，過去の障害を基に類似の故障を予防するための手法である故障モード影響解析手法を用い，ソフトウェアの追加・改変部分の開発に伴って既存機能の制約との不整合により生じた不具合を抽出，抽象化し，類似の不具合が防げるかを試みた[2]．具体的には，バージョン（$n-1$）の試験工程で発見された不具合のうち，既存機能の制約との不整合により発生した不具合を抽象化し，その不具合の要因を故障モードとして列挙した．故障モード毎に発生時のリスクをランク付けし，得られた故障モードをバージョンnの

試験工程で発見された不具合にあてはめてみた．その結果，故障モードと突き合わせることにより未然に防げていた可能性のある不具合を発見し，効果を確認した．

本稿では，文献[2]で報告した大規模保守開発型ソフトウェアを対象とした故障モード影響解析手法を紹介するとともに，適用評価中に得られた知見に基づき開発した故障モード影響解析支援ツールを紹介する．適用評価に際しては，多数のシートを利用するため作業が混乱し，管理が煩雑となった．手作業による影響範囲や過去の不具合の洗い出しの作業に時間を費やした．また，完成した故障モード影響解析結果の適用場面では，各故障モードの対策の実行確認やリスク分析に対する管理者への作業負担増に課題を残した．支援ツールは，これらの問題点を軽減する．作業手順の提示を行い，手順を学習するコストを抑え，多数のシートをデータベース化し煩雑さやミスを軽減する．バグトラッキングシステムと連携することにより，不具合を抽出するとともに，過去の不具合に基づく影響度を自動算出し，作業時間の短縮を図る．また，機能，担当者，工程が記された工程表と連携することにより，担当者に関係のある故障モードを自動的に通知する．支援ツールにより，初期学習の負担低減，故障モード影響解析手法の適用時の適用コスト低減，不具合情報共有の促進が期待できる．

以降，2章でハードウェアの故障モード影響解析について述べ，3章で対象ソフトウェアと適用方法を述べる．4章で適用結果とその考察を述べ，5章でそれらをふまえて開発した故障モード影響解析用ツールを紹介する．6章はまとめと課題を述べる．

2. 故障モード影響解析

2.1 ハードウェアを対象とした故障モード影響解析

故障モード影響解析は，システムの潜在的な故障・不具合の分析方法の一つである．具体的には，システムの識者が，機能やサブシステムなどシステムを構成する要素の一つ一つについて，起こりうる故障の現象や状態を挙げ，なるべく多くの故障にあてはまるよう抽象化して故障モードとし，故障が起きたときの上位要素への影響を解析する．その中からリスクの高い故障モードを抽出し，事前に対策を施すことにより，重大な事故・故障を予防する手法である．故障モード影響解析は，ハードウェアの設計や製造工程での潜在的故障・事故の早期発見や未然防止のために多くの企業で利用されている．故障モードは対象により様々であるが，ハードウェアの製品設計時には，部品や材料の折損，磨耗，短絡などの不具合であり，製造工程であれば，部品や材料の寸法の精度不足，加工時の損傷といったものがある[1]．

故障モード影響解析の具体的手順は以下のとおり．

ⅰ）システム構成要素の抽出
機能やサブシステム等にシステムを分割する．

ⅱ）故障モードの列挙
システムをよく理解しているメンバにより，構成要素毎に故障モード（故障の形態）を列挙する．

ⅲ）影響解析
・故障モードが発生した場合の影響について記述し，その故障モードの影響度・発生頻度・検出度について評価値を決定し，それらの積をリスク優先数（RPN）として求める．
・RPN が大きい順に，原因の特定，対策について検討する．

iv）故障モード影響解析シートの作成

ⅰ），ⅱ），ⅲ）の結果に基づき故障モード影響解析シートを作成する．シートは表形式で構成され，各行が構成要素に対応し，各列には，“機能”，“故障モード”，“故障の影響”，“検知法”，“致命度”が含まれる．

2.2　故障モード影響解析をソフトウェアに適用する際の問題点

ソフトウェア開発に，ハードウェアの故障モード影響解析をそのまま適用するには問題がある．具体的には，ソフトウェアには劣化に該当する部分が明らかにされていないこと，ハードウェアの故障は物理法則に依存する場合が多いのと比較して，ソフトウェアの故障は自由度が大きいために故障モード列挙の適用コストが大きいことである．また，前節で挙げたⅰ）の適切な選択や定義，及びⅱ）の定義は難しい．たとえば，システムの構成要素として機能名，故障モードとして機能の動作不良を挙げることが考えられるが，ⅰ）で定義する構成要素，ⅱ）で定義する故障モードの抽象度の選択が難しい．抽象度が高すぎると潜在的な不具合を見落としやすくなる．抽象度が低すぎると検知できる不具合の範囲が狭まり，防げる潜在的な不具合の範囲が小さくなる．最も抽象度が低い場合には，機能別の不具合一覧となる．

3.　適用対象ソフトウェアと適用方法

3.1　対象ソフトウェア

適用の対象としたソフトウェアは，10年以上にわたり継続して保守開発が行われている4,500 KLOC程度の規模の図形処理業務アプリケーションである．対象ソフトウェアは，毎年4回以上のバージョンアップが実施されている．対象ソフトウェアの大まかな保守開発工程を図1に示す．1回のバージョンアップで図1に示す工程が一通り実施される．

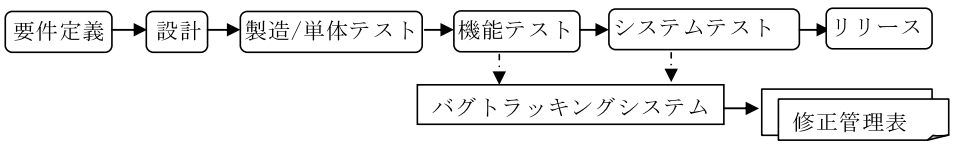


図1　対象ソフトウェアの保守開発工程

機能テスト・システムテストは，テスト専任の担当者で行われ，発見された不具合とその修正内容は，バグトラッキングシステムで管理される．リリース後に，バグトラッキングシステムより修正情報を取り出し，機能テスト・システムテスト毎の修正管理表としてまとめる．次バージョンに対する不具合の再発防止のために，リリース後に開発者が集まり，修正管理表から原因と今後の対策を検討する．

修正管理表は次の項目を含む．

- 不具合現象：不具合を起こしたコマンド・図形オブジェクト名，不具合の現象
- 修正内容：修正した内容，修正担当者，不具合の原因（プログラムの不正な動き等）
- 原因分析：不具合を混入した原因（考慮漏れ等），再発防止のための対策

3.2　適用方法

本稿の評価では，対象ソフトウェアで過去に実施された2回のバージョンアップを対象とし，あるバージョン *n* の修正管理表に含まれた20件の不具合を基に故障モード影響解析を実施し，得られた故障モードがバージョン *n+1* の修正管理表において不具合の未然防止につながるか評価した．

本適用評価で実施した故障モード影響解析の大まかな流れを図2に示す．2章で述べた故障モード影響解析手法そのままでは実施が難しい部分があったので，以下の1)～5)の作業手順で実施した．この分析手順の特徴は，コマンド・図形オブジェクトが関係するイベントを「処理」として分割し（手順1)），その「処理」に対し故障モードを列挙（手順2)，3)）した後に，再度，コマンド・図形オブジェクトと処理のマトリックス表（手順4)）を利用して，故障モードをコマンド・図形オブジェクト毎に列挙（手順5)）するところにある．この手順により，機能をより小さい単位である「処理」に分割し，その「処理」の抽象化を図ることで，より多くのコマンド・図形オブジェクト（新規も含む）で共通して故障モードがあてはまり，故障モードの再利用率の大幅な向上が見込める．なお，手順中のコマンドは外部コマンドラインツールを指し，図形オブジェクトは図形処理ソフトウェアが画面上で扱う図形要素を指すものとする．

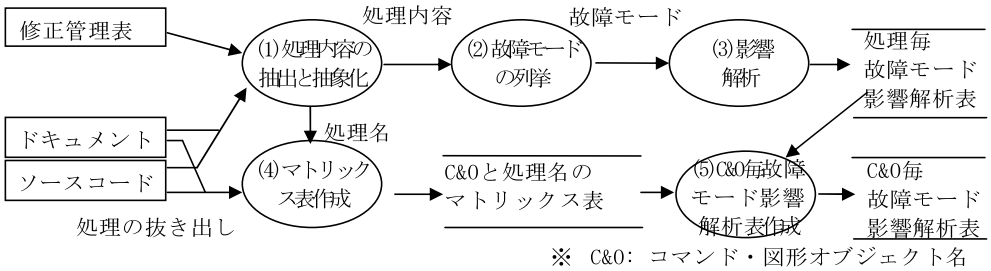


図2　故障モード影響解析の流れ

1)　処理内容の抽出と抽象化（2.1節のⅰ）と対応）

- 対象ソフトウェアが100を超える機能を持つため，構成要素を単一機能とせず処理とした．その理由は，「処理」の方が機能よりも数が少なく，各機能にまたがり共通的に使えると判断したからである．なお，ここでの「処理」は，プログラム中のイベントに準じたものを指すものとする．たとえば，ユーザの入力イベント，入力による影響範囲にある図形の選択，選択された図形同士の関係の判断，関係に基づいた処理，である．
- 修正管理表の修正内容から，不具合が発生した処理を明らかにし，その内容を記述する．修正管理表だけでは不明な場合には，ドキュメントやソースコードも参考にした．
- 処理内容を文章にし，その述部に注目して，記述内容を抽象化した表現（処理名）に変換する．

2)　故障モードの列挙（2.1節のⅱ）と対応）

- 1)で抽出した各処理に対し，不具合の原因となりやすい条件や考慮すべき制約を着目点として列挙した．

・ バージョン n の修正管理表に記述されている不具合について，上で挙げたどの着目点で考慮すべきであったのかを考察し，不具合の要因を故障モードとして記述した．その他，バージョン n 以前の不具合やレビューにおける指摘，入出力データも参考にした．

3） 影響解析（2.1 節の iii）と対応）

・ 故障モードが発生した場合の影響を検討し，故障モード毎に影響度・発生頻度・検出度の評価値を決定し，それらの積をリスク優先数（RPN）として求めた．

・ RPN が大きい順に抽出し，原因，対策，対策を施すべき開発工程について検討し，結果を記述した．

4） マトリックス表の作成

・ 表の行方向にコマンド・図形オブジェクト名，列方向に 1）で得られた処理名を列挙した．コマンド・図形オブジェクト毎に処理を実行しているかどうかを確認し，処理を実行している場合は 1，実行していない場合は 0 で表を埋めた．

・ 修正管理表に不具合項目として記録されているコマンド・図形オブジェクト名と処理名が存在する場合には，その要素値には不具合項目ひとつに付き 0.1 を加算し，重み付けを行った．

5） コマンド・図形オブジェクト毎の故障モード影響解析表作成

・ 開発や修正をする予定の項目より，コマンド・図形オブジェクトを抽出し，4）で作成したマトリックス表から一致する行を取り出した．新規のコマンド・図形オブジェクトの場合には，新たに行を加えて，マトリックス表を完成させた．

・ 係数の重み付けを調整する．既存機能に修正を加える場合や処理を変更する場合にはそのままの値を，修正を行わない処理の場合には値を半分にした．

・ コマンド・図形オブジェクト毎に，どのような処理名と故障モードが含まれているかを明らかにするために，上記で作成したマトリックスと 3）で作成した故障モード影響解析表の処理名から二つの表を結合し，コマンド・図形オブジェクト名毎に故障モードを列挙した表を作成した．3）で得られた故障モード影響解析表の RPN に対して，マトリックス表の係数を掛けた値を新しい RPN とした．RPN の値の大きな順に並び変えたシートを，要件定義レビューや設計，製造など各工程にて利用する．

4. 実施結果と考察

4.1 実施結果

3.1 節で述べた図形処理業務アプリケーションのあるバージョン n を対象として 3.2 節で示した手順で故障モード影響解析を実施した．

1） 処理内容の抽出と抽象化

表 1 に修正管理表の一部を取り出し，処理内容を抽出する具体例を示す．ただし，対象ソフトウェア固有の名称を“図形オブジェクト”や“種別 ID＝1001”のように置き換えている．表 1 中の各行が修正管理表の一つの修正に対応する．“コマンド・図形オブジェクト分類”以降 3 列は修正管理表からそのまま取り出したものであり，“処理内容”，“処理名”列が手順 1）の実施結果である．表 1 中の ID1 と 3 は，処理内容“図形（種別 ID＝1001）の表示領域を求める（ID 1）”，“図形（種別 ID＝2002）の側面形状を求める（ID 3）”と内容は異

なるが，ともに“オブジェクト”，“形状を作成する”という共通の処理とみなし，“オブジェクト（図形種別 ID＝1000 ～ 2999）の作成範囲を求める”と抽象化した．この抽象化は対象ソフトウェアの内部構造の知識に基づいて行った．また，図形種別 ID＝1000 ～ 2999 は同様の不具合が起こりうる種類の種別 ID を抽象化したものである．

2） 故障モードの列挙

表 2 は，1)で得られた処理名“オブジェクト（図形種別 ID＝1000 ～ 2999）の作成範囲を求める”に対して故障モード解析を行ったものである．“処理名”列は表 1 の“処理名”列と対応する．“着目点”列は“故障モード”列を挙げる際に着目した点を，“具体例”列は表 1 の“不具合現象”列に対応する．“具体例”列が空白のもの（故障モードの先頭に“A1.2.4, A1.2.5”と記述のある行）は修正管理表には記録されていないが，留意すべき故障モードとして追加されたものである．

3） 影響解析

表 3 は影響解析の結果であり，RPN 値上位 3 位を降順に記している．故障モードの先頭の“A1.2.4”等の ID は表 2 の“故障モード”に対応する．影響内容は故障モードにより引き起こされる影響を記述したものである．各行は表 2 の故障モードと対応する．“影響”，“発生”，“検出”は，リスクの深刻度に応じ，1（リスク最小）～ 5（リスク最大）のランク値を決定した．“発生”は，リリース後にその不具合が発生する頻度を推測したものである．“検出”は不具合の発見工程が下流になるほど高い値を設定した．該当する過去の不具合がない場合にはどの工程で検出されるかを推測し，同様に設定した．

表 1　修正管理表からの処理内容の抽出と抽象化（抜粋）

修正管理表から				手順 1)の結果	
ID	コマンド・図形オブジェクト分類	不具合現象	原因	処理内容	処理名
1	図形オブジェクト(図形種別 ID＝1001)	図形(種別ID＝1001)と図形(種別ID＝1002)が重なると不正な表示になる.	図形(種別ID=1002)と接する図形(種別ID=1006)に図形(種別ID=1001)を重ねて入力した場合の考慮がなかった.	図形(種別ID=1001)の表示領域を求める	オブジェクト（図形種別ID=1000～2999）の作成範囲を求める
2	図形オブジェクト(図形種別 ID=2001)	属性値Yを持つ図形(種別ID=2001)が属性値Xを持つ図形（種別ID=1004）でトリミングされない.	属性値Yを持つ図形(種別ID=2001)と属性値Xを持つ図形（種別ID=1004）の図形領域が重ならないためトリミングされていなかった.	入力図形の表示の際に他の図形でトリミングする.	オブジェクト（図形種別ID=1000～2999）の作成範囲を求める
3	図形オブジェクト(図形種別 ID=2002)	図形(種別ID＝2002)の側面形状が出力されない.	図形(種別ID＝2002)に接する形状に穴形状がある場合の考慮がされていない.	図形(種別ID＝2002)の側面形状を求める	オブジェクト（図形種別ID=1000～2999）の作成範囲を求める
4	図形オブジェクトの微調整移動コマンド(コマンド ID=101)	図形(種別ID=1005)を指示したところ，ユーザへの問い合わせで一つの図形に対し，二つの図形が表示された.	図形(種別ID=1005)を画面に表示するときに，線分を二つの図形グループに分割して表示したために，検出処理で 2 図形が検出されてしまった.	図形(種別ID=1005)を画面に表示する.	オブジェクト（図形種別ID=1000～2999）を画面表示する

表2　列挙した故障モード（抜粋）

処理名	着目点	故障モード	具体例
A1：オブジェクト（図形種別ID=1000～2999）の作成範囲を求める	A1.1入力により影響を受けるオブジェクトの選定	A1.1.1特定の入力の組合せ（入力A，及び，B）において影響を及ぼすオブジェクトの選定処理がない	図形(種別ID＝1001)が不正に図形(種別ID＝1002)と重なる.
	A1.2オブジェクト同士の関係に基づいた処理	A1.2.1属性値Xを持つオブジェクトと属性値Yを持つオブジェクトの干渉が考慮されていない	属性値Yを持つ図形(種別ID=2001)が属性値Xを持つ図形（種別ID=1004）でトリミングされない.
		A1.2.2対象とするオブジェクトが穴形状を持つ場合の考慮不足	図形(種別ID＝2002)に接する形状に穴形状がある場合の考慮不足
		A1.2.3対象とするオブジェクトとの高さ方向の干渉が考慮されていない.	図形(種別ID=1004)入力時に，異なるレイヤにある図形(種別ID=1003)との干渉が考慮されていない.
		A1.2.4他のオブジェクトと完全に一致した位置の場合の考慮がされていない.	
		A1.2.5関係のあるオブジェクトが複数ある場合の考慮がされていない.	

表3　影響解析結果の原因と対策（RPN 値上位 3 位）

故障モード	影響内容	影響	発生	検出	RPN	原因	対策	工程
A1.2.4 他のオブジェクトと完全一致した場合の考慮がない.	領域のない図形となり不正処理を引き起こす.	5	4	4	80	領域がなくなることの考慮不足	領域がなくなる場合のオブジェクト同士の組合せは定義済みであるため，単体テストに追加する.	設計製造
A1.2.1 属性値Xを持つオブジェクトと属性値Yを持つオブジェクトの干渉が考慮されていない	形状不正. 後工程に不正形状が連携する.	5	3	4	60	要求仕様書での定義漏れ.	オブジェクト一覧と属性値の表を作成し，要求仕様書に添付でつける．また，レビューでチェックする.	要求
A1.2.5 関係のあるオブジェクトが複数ある場合の考慮がない.	重なり部分が正しい形状にならない.	4	3	3	36	設計書での対象オブジェクト数の考慮不足	設計仕様書や変数名に単数・複数がわかるように記述する．（変数の最後にsをつける）	設計製造

表4　コマンド・図形オブジェクトと処理名のマトリックス表の例（バージョン *n*）

コマンド・図形オブジェクト名	作成範囲を求める	画面に表示する	操作モードを変更する	属性を操作する
図形(種別ID=1001)	1.1	1	0	1
図形(種別ID=2001)	1.1	1.1	0	1
コマンド(ID=101)	1	1.1	0	1

表5　コマンド・図形オブジェクトと処理名のマトリックス表の例（バージョン *n*＋*1*）

コマンド・図形オブジェクト名	区分	作成範囲を求める	画面に表示する	操作モードを変更する	属性を操作する
図形(種別ID=2001)	修	1.1	1.1	0	0.5
コマンド(ID=102)	新	1	0	0	0

4） マトリックス表の作成

3.2節の手順4)に従い，設計ドキュメントよりコマンド・図形オブジェクト名と処理名の関係を抽出し，係数を求めた結果を表4に示す．また，修正管理表に不具合項目として存在したコマンド・図形オブジェクト名と処理名の組み合わせには，不具合項目ひとつに付き0.1を加算した．

5） コマンド・図形オブジェクト毎の故障モード影響解析表作成

表5は作成したマトリックス表の一部である．表5中の“図形（種別ID＝2001）”は，“作成範囲を求める”・“画面に表示する”の処理を修正するため，係数が1.1（表4の3行目2列，3列目の1.1を1.0倍する）である．“操作モードを変更する”については表4の値が0であるため，表5でも0とした．“属性を操作する”は，変更予定がないため，係数が0.5（表4の3行目5列の1.0を0.5倍する）となっている．コマンド（ID＝102）は，新規に作成する機能であり，ドキュメントより“作成範囲を求める”処理に類似した処理のみを開発する予定であるため，係数が1となっている．

以上の手順を実施した上で「開発担当者がコーディングの前に関係する故障モードを与えられていた場合，テスト実行前に不具合現象を事前に発見できたかどうか」について評価する．そのために，バージョン*n*の修正管理表から得られた故障モードを基に，バージョン*n+1*の修正管理表に記述された不具合現象が防げていたかどうかを，当該プロジェクトのプロジェクトマネージャが分析した．具体的には，故障モードが与えられた場合に，担当者が不具合を防げていたかどうかをプロジェクトマネージャが担当者の視点で判断した．その結果，2件の不具合が未然に防げた可能性があることを確認した．2件の不具合を表6に示す．

表6　次バージョンで見つかった不具合と故障モードの対比表

コマンド・図形オブジェクト名	不具合現象	原因	故障モード
コマンド(ID=102)	R1:図形(種別ID=2003)が正常な位置にも関わらず，図形(種別ID=1003)との位置チェックで不正とされる．	図形(種別ID=1003)の高さが正しく取得できずにチェック不正なチェックとされていた．	A1.2.3 対象とするオブジェクトとの高さ方向の干渉が考慮されていない．
コマンド(ID=102)	R2:図形(種別ID=2003)が正常な位置にも関わらず，図形(種別ID=1002)との位置チェックで不正とされる．	図形(種別ID=1006)と図形(種別ID=1002)が完全に一致した場合に領域がなくなるが，そのときの動作が考慮されていない．	A1.2.4 他のオブジェクトと完全に一致した位置の場合の考慮がされていない．

4.2 考察

評価結果は，新規コマンドであっても前バージョンの故障モードを再利用できたことから，ソフトウェア故障モード影響解析に対する我々のアプローチの有用性を示唆していると言える．再利用できた理由としては，対象ソフトウェアの構成要素をコマンド・図形オブジェクトではなく処理内容としたこと，処理内容の述部に着目しそれを抽象化した上で処理名としたことが挙げられる．その結果次の利点が得られたと考える．

- 多種におよぶコマンド・図形オブジェクトを処理名として抽象化することにより共通化が可能になった．故障モードの抽出や影響解析を処理名に対して実施でき効率化できた．

- 新規機能を追加する時も追加する処理を抽象化された処理名に分割することで，既存の故障モードや影響解析を再利用できる．
- コマンド間にまたがる類似の処理に対して，横断的に故障モードによる分析が可能になる．

また，対象ソフトウェアでは主要機能の規模が大きく 100 を超える機能から構成されるものであったため，機能毎の故障モードの列挙が困難であり，列挙できたとしてもそのチェックが現実的には難しいことが推測された．分割の観点を機能から処理に変更したことにより，これらを現実的な数に抑えることができた．

対象ソフトウェア開発プロジェクトにおいても修正管理表を用いた開発ミーティングを実施し，不具合の再発防止や改善に一定の役割を果たしていた．本稿で実施した故障モード影響解析の実施により，機能をまたがって発生するような不具合項目を発見することや複数の機能に共通する不具合を未然に防げることが期待される．なぜなら，機能と処理名のマトリックス表を使うことで，処理の観点からもれなく統一的に複数の機能を精査することができるからである．故障モード影響解析表の作成には対象ソフトウェアの知識，業務知識が必要になるが，一旦作成してしまえば比較的経験の浅い開発者でも利用することができる．

本件のプロジェクトは，バージョンアップを繰り返して行う保守開発のため，最初は時間や労力をかけてでも故障モード影響解析表を作成し，以降のバージョンにわたって利用可能とすることで，作業効率や品質の確保が見込めることから，1 バージョンあたりの費用対効果が大きくなると考える．今回は，機能単位ではなく処理単位に故障モードを列挙する方が，少ない故障モードになると考えられた．しかし，処理の抽象化が難しく，処理の数が多くなる場合には，故障モードが多数となり，故障モード影響解析として有効に利用するのが難しくなることが予想される．この場合には，RPN による優先順位つきチェックや，機能群毎のリスクを勘案した故障モード影響解析の部分的な適用等が考えられる．

5. 故障モード影響解析支援ツール

5.1 故障モード影響解析実施時の問題点と支援ツール開発の動機

本稿で報告している作業の多くは表形式のシートの作成である．シート作成は，市販の表計算ソフトを利用して行った．その際に明らかになった課題を示す．

1） 表形式シートの多さによる作業負担

故障モード影響解析表やマトリックス表など 6 種類のシート作成が必要である．また，故障モードの分析と次期バージョン用の分析のシートフォーマットが若干異なるため，複数のシートフォーマットが必要となり作業が混乱しやすい．それに加え，プロジェクトのバージョン毎にシートを分ける必要があるため，管理が煩雑になる．また，複数のシート間で重複する項目や内容も多いが別々に作成する必要があり，作成効率やメンテナンス効率に影響を与える．

2） 不具合数の集計やシート間の組み合わせに時間がかかる

修正管理表から不具合を拾い出し，コマンド・図形オブジェクトと処理別に数をカウントすることやマトリックス表と処理毎故障モード影響解析表からコマンド・図形オブジェクト毎故障モード影響解析表を作成することは，手作業で行うと多大な時間を消費する上，ミスを犯す可能性も高い．

3） 手作業による影響範囲の確認に手間がかかる

故障モードの影響範囲や発生頻度，検出度を求めるために，修正管理表，ソースコードリポジトリ，バグトラッキングシステムを調査したが，これらは連携されておらず，個々に調査する必要があり非常に手間であった．

4） 抽出した故障モードの対応状況の確認コストが大きい

プロジェクト計画時に作成したコマンド・図形オブジェクト名毎故障モード影響解析表を使って，その故障モードへの対応を行っているか，正しく実行されたか，有効な対策であったのかなどの対応状況を，各開発者に確認するための手間やコストは少なくない．これらの状況を一元的に管理できれば，確認コストが小さくなり，確認漏れも減少することが期待される．また，計画立案時にあらかじめどの時期にどれくらいのリスクが存在するのかを知っておくことができれば，担当者に対して適切な支援ができる．

5.2 ツールの概要

図3にツールの概要図を示す．本ツールでは，メニューに示された項目に従い作業することにより，煩雑な手順を手引きし，不慣れな作業者でも故障モード影響解析表を作成することを可能とする．故障モード影響解析表とマトリックス表の各項目のデータベース化を行い，重複入力の削減や項目候補値の表示など入力を補助する機能を実現し，前節1)の作業負担を軽減する．重み付けのマトリックス表やコマンド・図形オブジェクト毎の故障モード影響解析表を自動作成する機能を実現し，前節2)を解決する．故障モードの影響度の算出時には，ソースコードリポジトリとの連携により，修正ソースファイル数，修正行数を求め自動算出を行う．検出度についても，修正管理表やバグトラッキング情報より修正時期を参照することにより算出する．各システムとの連携により作業負担の軽減を図り，前節3)を解決する．各コマンド・図形オブジェクトの担当者および各工程のスケジュールとコマンド毎の故障モード影響解析表のコマンド名・工程を連携させることにより，各担当者別，日程別の故障モード影響解析表を作成し，管理者の確認コストを低減し，前節4)の解決を図る．

ツールの機能は，故障モード抽出と現プロジェクトの分析の二つから構成される．故障モード抽出では，ユーザは過去の修正管理表より処理名を抽出し，処理名毎に故障モードを記載し，処理名毎の故障モード影響解析表を作成する（表1，2に対応）．ユーザが作成したコマンド・図形オブジェクトと処理のマトリックス表を基に重み付きマトリックスをツールが自動作成する．現プロジェクトの分析では，今回保守開発の対象となるコマンド・図形オブジェクトとその処理名についてのマトリックス表を作成することで，コマンド・図形オブジェクト毎の故障モード影響解析表を自動作成することができる．また，工程表を記述することで，どの時期・どの担当者にどのようなリスクが存在し，そのリスクの大きさはどれくらいあるのかを予想する時系列リスクグラフを自動作成することができる．故障モード抽出機能のユーザは対象ソフトウェアに深い知識をもつ技術者である．現プロジェクト分析機能のユーザはプロジェクトマネージャ，リーダ，担当者であり，必ずしも対象ソフトウェアに深い知識を持つ必要はない．

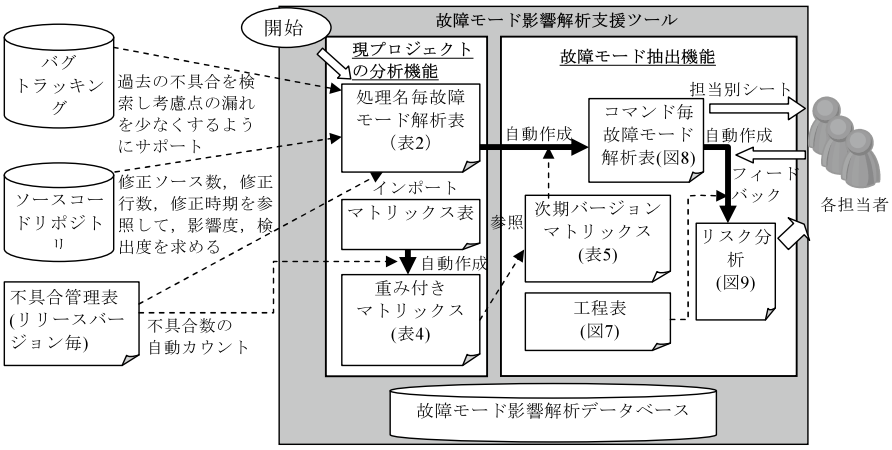


図3　故障モード影響解析支援ツールの構成

5.3　故障モード抽出機能

修正管理表より故障モードを抽出するための手順（2.1節ⅰ)～ⅲ))をサポートする機能を実現している．図4の左側のメニューに示すように，ユーザの入力が必要な入力用のメニューと自動作成が可能な出力用のメニューに分離し作業内容の視認性を高めている．

5.3.1　入力項目

処理名の抽出，処理毎の故障モード影響解析表の作成，マトリックス表の作成の機能がある．処理名の抽出では，既存修正管理表のインポート機能により，簡単に表計算ソフトからシートを取り込むことができる．記述した処理名は，データベースに保存され，修正管理表の番号とリンクされる．処理毎の故障モード影響解析表作成では，図4に示すように表形式のシート一覧とそのセルでの入力候補値（既存の故障モードからリスト表示）が右横に表示されるため効率よく値を設定することができる．また，故障モード抽出時には，過去の不具合を検索する機能を利用することで，故障モードの漏れを防ぐことができる．故障モード毎に定義する影響度と検出度は，その不具合が発見された時期，修正された時期，修正したソースコード行数に応じて，自動算出される．図5は，ある不具合修正について求めた影響度，検出度を示している．

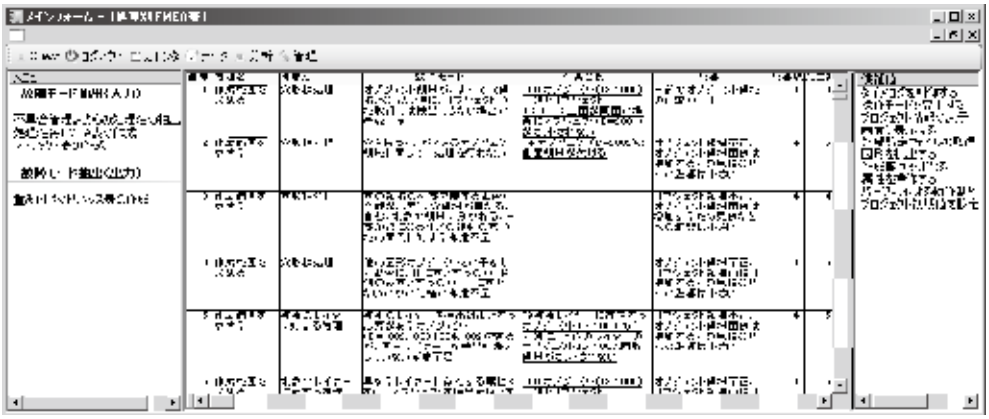
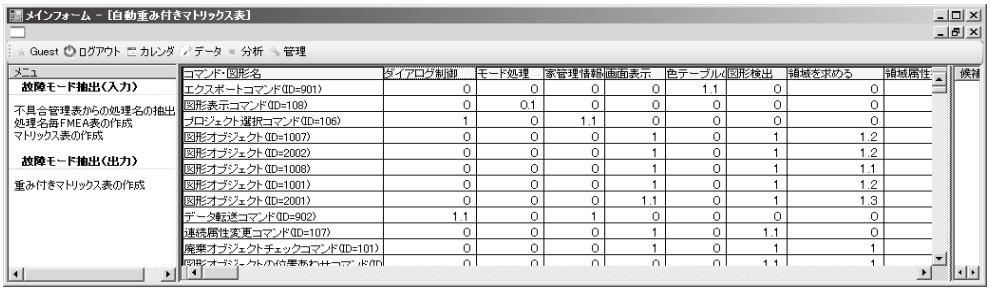


図4　処理名毎の故障モード影響解析表（表3の編集イメージ）

図 5　リポジトリ，バグトラッキング情報



コマンド・図形名	ダイアログ制御	モード処理	家管理情報	画面表示	色テーブル	図形検出	領域を求める	領域属性
エクスポートコマンド(ID=901)	0	0	0	0	1.1	0	0	
図形表示コマンド(ID=108)	0	0.1	0	0	0	0	0	
プロジェクト選択コマンド(ID=106)	1	0	1.1	0	0	0	0	
図形オブジェクト(ID=1007)	0	0	0	1	0	1	1.2	
図形オブジェクト(ID=2002)	0	0	0	1	0	1	1.2	
図形オブジェクト(ID=1008)	0	0	0	1	0	1	1.1	
図形オブジェクト(ID=1001)	0	0	0	1	0	1	1.2	
図形オブジェクト(ID=2001)	0	0	0	1.1	0	1	1.3	
データ転送コマンド(ID=902)	1.1	0	1	0	0	0	0	
連続属性変更コマンド(ID=107)	0	0	0	1	0	1.1	0	
廃棄オブジェクトチェックコマンド(ID=101)	0	0	0	1	0	1	1	

図 6　重み付きマトリックス表の出力例（表 5 シートのイメージ）

5. 3. 2　出力項目

図 6 に示すように，重み付きマトリックス表の出力機能を実現している．ここでは，修正管理表から処理名とそのコマンド・図形オブジェクト名を抽出し，その発生数を自動カウントし，重み付けをしてマトリックス表に出力する．この機能により，不具合数のカウントをユーザが行う必要はなくなる．また，何バージョンにもわたり，不具合数をカウントし管理することが可能である．

5. 4　抽出した故障モードに基づく現プロジェクトの分析機能

処理毎の故障モード影響解析表を基に，現在のプロジェクトで開発するコマンド・図形オブジェクト毎に，どのようなリスクが存在するかを示すコマンド・図形オブジェクト毎の故障モード影響解析表を作成するための手順（2. 1 節 iv））を支援する機能を実現している．本機能では，コマンド・図形オブジェクト毎の故障モード影響解析表に加え，時系列のリスクグラフを作成する機能を持つ．

5. 4. 1　入力項目

図 7 は，工程管理表の作成画面である．コマンド・図形オブジェクト毎の各工程（要求，設計，…）における担当者名，納期を入力する．担当者名の候補値はリストとして表示し入力補助を行う．

5. 4. 2　出力項目

コマンド・図形オブジェクト毎の故障モード影響解析表および時系列リスク表の出力が可能である．図 8 は，自動作成したコマンド・図形オブジェクト毎の故障モード影響解析表を画面

表示しているイメージである．この項目の機能には，明らかにコマンド・図形オブジェクトに関係のないリスクを除外する機能やログインしているユーザ名に応じたリスクの一覧を表示する機能がある．リスク値別や日時別に並び変えることで，詳細なリスクの分析も可能である．

図 9 に，時系列リスクグラフの出力例を示す．リスクグラフでは，横軸に日付，縦軸にリスク割合（各日付におけるリスク値を加算したものを全体のリスク値で除算した値）を示す．図

図 7　工程管理表の入力画面

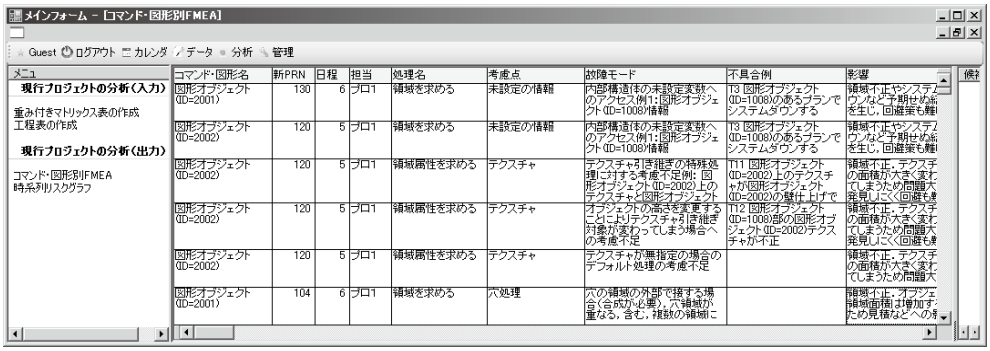

メインフォーム - [コマンド・図形別FMEA]
Guest ログアウト カレンダ データ 分析 管理
メニュー
現行プロジェクトの分析（入力）
重み付きマトリックス表の作成
工程表の作成
現行プロジェクトの分析（出力）
コマンド・図形別FMEA
時系列リスクグラフ
コマンド・図形名 新PRN 日程 担当 処理名 考慮点 故障モード 不具合例 影響
図形オブジェクト(ID=2001) 130 6 プロ1 領域を求める 未設定の情報 内部構造体の未設定変数へのアクセス例1：図形オブジェクト(ID=1008)情報 T3 図形オブジェクト(ID=1008)のあるプランでシステムダウンする 領域不正やシステムダウンなど予期せぬ振を生じ，回避策も難
図形オブジェクト(ID=2002) 120 5 プロ1 領域を求める 未設定の情報 内部構造体の未設定変数へのアクセス例1：図形オブジェクト(ID=1008)情報 T3 図形オブジェクト(ID=1008)のあるプランでシステムダウンする 領域不正やシステムダウンなど予期せぬ振を生じ，回避策も難
図形オブジェクト(ID=2002) 120 5 プロ1 領域属性を求める テクスチャ テクスチャ引き継ぎの特殊処理に対する考慮不足例：図形オブジェクト(ID=2002)上のテクスチャと図形オブジェクト T11 図形オブジェクト(ID=2002)上のテクスチャが図形オブジェクト(ID=2002)の継仕上げで 領域不正，テクスチの面積が大きく変わってしまうため問題大発見しにくく回避も難
図形オブジェクト(ID=2002) 120 5 プロ1 領域属性を求める テクスチャ オブジェクトの滑さを変更することにより，テクスチャ引き継ぎ対象が変わってしまう場合への考慮不足 T12 図形オブジェクト(ID=1008)型の図形オブジェクト(ID=2002)テクスチャが不正 領域不正，テクスチの面積が大きく変わってしまうため問題大発見しにくく回避も難
図形オブジェクト(ID=2002) 120 5 プロ1 領域属性を求める テクスチャ テクスチャが無指定の場合のデフォルト処理の考慮不足 領域不正，テクスチの面積が大きく変わってしまうため問題大
図形オブジェクト(ID=2001) 104 6 プロ1 領域を求める 穴処理 穴の領域の外部で接する場合（合成が必要），穴で領域が重なる，含む，複数の領域に 領域不正，オブジェ領域面積，計増加すため見積などへの

図 8　コマンド・図形オブジェクト毎の故障モード影響解析表（表 6 のイメージ）

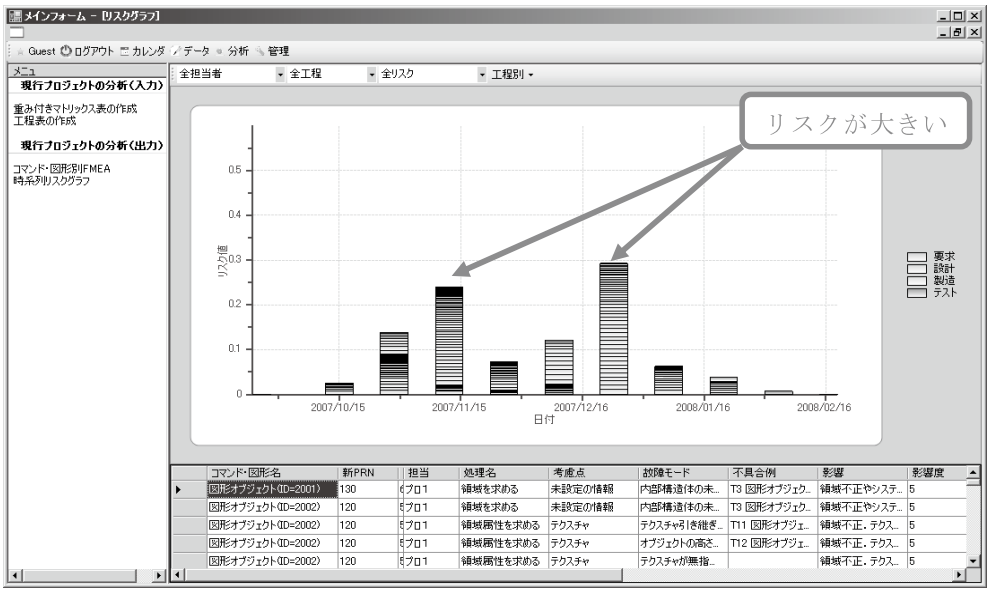


図 9　時系列リスクグラフの例

9 では，リスク値が 2 回大きな値になる日付が存在することがわかる．このようにグラフを確認することにより，いつどのようなリスクが存在するかを即座に確認することが可能である．担当者や工程別，リスクの大中小毎に表示することにより，リスクの分散や担当者の再割り当てをシミュレーションすることに役立つことが期待できる．

5. 5　本支援ツール使用による効果

作成したツールにより，自動化による効率化のメリットだけでなく次の効果が期待できる．

故障モードがデータベース化されているため，故障モードが追加された場合には，その処理名を使用しているコマンド・図形オブジェクトを担当している開発者へ追加通知を行い，注意を促すことでリアルタイムに類似ミスを減らすことが期待できる．また，複数のプロジェクトを対象にするなどにより規模が大きくなった場合でもデータベースで一元管理でき，故障モードの共有化や過去のバージョンの故障モードの参照などを簡単に行うことができる．このように故障モードの一元管理・共有化は，様々な開発者の知見を蓄積・共有化する場となり，再利用性を高め，品質向上を図る手段となることが期待できる．

6. お　わ　り　に

本稿では，ソフトウェアへの適用事例の報告があまりなかった故障モード影響解析を商用の保守開発ソフトウェアに適用した結果を述べた．故障モードの抽出では，システムテストの修正管理表からソフトウェアの処理内容に注目し，処理内容を抽象化し処理名とし，処理名に対する着目点とその故障モードを列挙することで，従来の“エラーメッセージとともにプログラムが停止する”のような抽象的で不具合の発見に直接的に貢献しにくい故障モードではなく，より具体的な故障モードを抽出できた．また，同一ソフトウェアの異なるバージョンの修正管理表と照らし合わせることにより，限定的ながら，抽出した故障モードが新機能に対しても再利用可能であることを確認した．また，得られた知見に基づき故障モード影響解析支援ツールを開発した．

今後は，従来のチェックリスト手法と本稿の故障モード影響解析手法との作業効率や品質向上の比較を行い，故障モード影響解析の優位性を検証する予定である．

なお，本稿の執筆にあたって，奈良先端科学技術大学院大学 松本健一先生，飯田元先生，日本ユニシス・エクセリューションズ株式会社　芹澤清隆氏，森山嘉通氏に有益なコメント，ご支援を頂いた．この場を借りて感謝申し上げる．

参考文献　［1］ 小野寺勝重，“実践 FMEA の手法”，日科技連，1998 年
［2］ 山科 隆伸，森崎 修司，飯田 元，松本 健一“保守開発型ソフトウェアを対象としたソフトウェア FMEA の実証的評価”，ソフトウェア品質シンポジウム 2008 発表報文集，pp.157-164，2008 年

執筆者紹介　山　科　隆　伸（Takanobu Yamashina）

1990年大阪大学工学部卒業．同年，日本ユニシス(株)入社．住宅専用CADの開発，適用サポートに従事．2007年奈良先端科学技術大学院大学 博士前期課程入学，コードクローンとソフトウェアレビューの研究に従事．

森　崎　修　司（Shuji Morisaki）

2001年奈良先端科学技術大学院大学 情報科学研究科 博士後期課程修了．2001年より情報通信企業にてオンラインストレージサービスの立ち上げ/マネージメント，RFIDソフトウェアの国際標準策定活動に従事．2007年より奈良先端科学技術大学院大学　助教．ソフトウェア計測とソフトウェアレビュー・インスペクションの研究に従事．