Institute of Electronics, Information, and Communication Engineers

一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
DC2012-19 (2012-8)

# 障害対応ワークフローに起こる二次リスクの保証事例の報告

中田　晋平† 養安　元気† 菅谷みどり†† 山本修一郎††† 倉光　君郎††

† 横浜国立大学大学院工学府物理情報専攻
††† 名古屋大学 情報連携統括本部 情報戦略室
†† 横浜国立大学, JST/CREST

**あらまし** 近年，ソフトウェアシステムは我々の社会基盤に広く浸透し，システムの障害は大きな経済的損失だけでなく，我々の社会生活に困難を与える．このような事態を避けるため，多くの，信頼性が重要視されるシステムでは，障害から復旧するための障害対応を備えており，障害発生時に障害対応を実行することで，被害を最小化する努力が行われている．しかし，これらの対応が正しく動作せずに障害が長期化する事例が報告されており，障害対応に含まれる二次リスクへの対応も十分に検討される必要がでてきている．本論文では，実際の障害対応例を対象に二次リスク検討を行い，二次リスク検討およびその保証に関する経験を報告する．

**キーワード** 障害対応，二次リスク，ソフトウェア工学

# A Report on Secondary Risk Assurance for Practical Web-service

Shinpei NAKATA†, Motoki YOAN†, Midori SUGAYA††, Shuichiro YAMAMOTO†††, and Kimio KURAMITSU††

† Graduate School of Yokohama National University
††† Nagoya University, Strategy Office, Information and Communications Headquarters
†† Yokohama National University, JST/CREST

**Abstract** Recently, many parts of our daily life depend on software system. Therefore, the failure of such system results in not only economical loss of service provider but also influence on our daily life. To avoid this disastrous situation, service provider provides a solution, recovery preparation against system failures. However, there are some failure reports of the system failures that have fails of these recovery process fails its execution. This is because there is less consideration of potential secondary risks of recovery processes. In this paper, we propose to discuss and assure that the recovery process correctness.

**Key words** Dependability, Assurance Case

## 1. はじめに

近年，ソフトウェアシステムは我々の社会基盤に広く浸透し，生活のインフラを支える重要な役割を果たしている．これらシステムの障害は大きな経済的損失だけでなく，我々の社会生活に困難を与えてしまう．システム障害を避けるため，多くの，信頼性が重要視されるシステムでは，障害を解決するための障害対応を備えており，障害発生時に障害対応を実行することで，被害を最小化する努力が行われている．

しかし，障害を解決するために用意された，これら障害対応が問題を含み，事態がより深刻になってしまう事例が報告されている [1] [2]．システムに潜むリスク（一次リスク）に備えた障害対応が失敗する可能性は，二次リスクと呼ばれ，上述の事例は二次リスクによる障害の深刻化が問題となった例である．

本研究の目的は，システムの障害対応で起こりうる二次リスクへの対応を議論し，そして議論の結果として二次リスクが解決されていることを示すことにある．本論文では，障害対応のために記述される障害対応ワークフローに着目し，ワークフローから書き下した障害対応スクリプトを対象に，その二次リスクを抽出し，これらが正しく解決されていることを示す試みを行った．

我々は，JST/DEOS プロジェクト [3] の一部として，障害対応のスクリプト化を行う基盤整備を進め，事前の障害対応シナリオから事後の応急対応まで，実施する枠組みを進めている．本論文では，実際の障害対応事例を基に記述されたスクリプトを対象に，その二次リスクを検討，対応に関して議論した事例



This article is a technical report without peer review, and its polished and/or extended version may be published elsewhere.
Copyright ©2012 by IEICE
NII-Electronic Library Service

Institute of Electronics, Information, and Communication Engineers

を報告する．また，二次リスク解決にむけた議論のなかで得られた知見について報告を行う．

本論文の貢献は以下の通りである．

- 障害対応例における二次リスクの抽出
- 実際の障害対応例を基に作成した保証ケースの作成と，そこから得られた知見の報告

本論文の構成は以下の通りである．第二節は，本研究の動機となる二次リスクについて説明する．第三節では，障害対応記述に用いるスクリプト言語である D-Script のスクリプトモデルについて説明を行う．第四節では，我々の提案している二次リスク保証手法について説明し，第五節では実際の障害例とその対応について述べる．第六節では関連研究と比較し，第七節でまとめとした．

## 2. 研究動機

### 2.1 障害対応の二次リスク

情報システムには，その機能やサービスに由来する障害を発生させるリスクが存在する．まずシステムの機能やサービスに直接由来するリスクを一次リスクと呼ぶ．二次リスクは，一次リスクへの対策中に発生するリスクである．同様に，$n$ 次リスクにはそのリスクへの対策中に発生する $n+1$ 次リスクが存在し得る．東証 [2] や Gmail [1] の例にも見られるように，二次リスクへの対応が考慮されていないと障害対応が正しく動作しなかった場合に，障害が深刻化してしまう恐れがある．　本研究では，実際に問題になっている事例を考慮し，特に二次リスクとその対応方法を議論する．

### 2.2 障害対応ワークフロー

システムの一次リスクに備えるため，信頼性が必要となるシステムでは障害対応が検討される．障害対応はなんらかの手順を持ち，我々はこれを障害対応ワークフローと呼ぶ．障害対応ワークフローには，システムの再起動であったり，担当運用者の呼び出しであったり，障害対応で実行される手順が含まれる．これら，障害対応ワークフローの一部，または全部は，スクリプトとして記述されていることが多い．スクリプト化は，行うべき手順が量的に多い場合の自動化，複雑な手順実行に伴うヒューマンエラーの軽減，障害検出のタイミングでの自動的な実行，といったメリットがあり，多くの統合管理システムでも障害対応のスクリプト化は実践されている．[4] [5]　しかし，これらのスクリプトが正しく動くかどうかの議論は，内部的には検討されていても，システムを利用するステークホルダへは一次リスクへの対応手段がある，という形でしか通知されない．そこで我々は，障害対応ワークフローをもとに，二次リスクへの対応が考慮されているということを示す手法が必要だと考える．

### 2.3 保証ケース

障害対応で起こりうる二次リスクへの対応が考慮されていることを示す手法として，我々は保証ケースに着目している．保証ケース (Assurance Case) [6] は，製品やシステムが望ましい性質を持ち，危険な状況に陥らないことを示すための論理的な議論とその証跡を示すためのドキュメントである．システムの保証ケースを記述する場合は，システムが安全性や信頼性を持つ事を示すために用いられる [7] [8].

## 3. D-Script スクリプトモデル

D-Script 言語は，D-Script フレームワークにおける分散実行管理用の上位スクリプト言語である．D-Script フレームワークの設計，実行モデルなどは論文 [9] に詳しい．本論文では，障害対応スクリプトを D-Script 言語 (以下，D-Script) を用いて記述する．本節では，二次リスク検討に必要な D-Script のスクリプトモデルについて説明を行う．　D-Script は 2 種類のスクリプトレベルを持ち，High-level な記述である D-Task, D-Control について，その役割を述べる．Low-level な記述は，具体的なシステムへの操作となるため，通常のシェルスクリプトなどを用いても差し支えない．

### 3.1 D-Task

D-Task は，アトミックに処理を行うべきスクリプトの単位である．ここでアトミックとは，D-Task が完了すると，必ず成功もしくは失敗のどちらかの状態を報告することである．失敗した場合は，状態が変更されないことを保証する．

### 3.2 D-Controll

D-Control は，複数の D-Task を実行するワークフローを記述する記法である．理解しやすさから，ビジュアル表記を用いている．D-Script がサポートする制御フローは，次の 2 種類である．

- D-Task の逐次実行
- D-Task の並列実行

D-Control はプログラム言語の基本ブロックに相当し，内部に制御の分岐や合流を含まない D-Task の列として表現される．

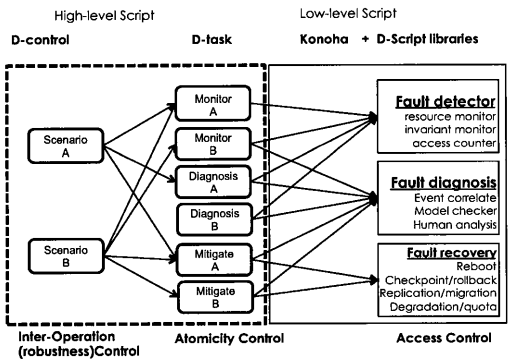


図 1　D-Script スクリプトモデル

### 3.3 ワークフロー表記

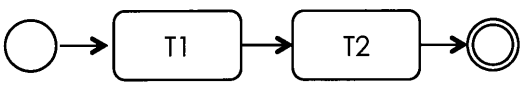


図 2　D-Script のワークフロー表記

D-Script では，D-Task 間の実行順序の表記に図のような表記を用いる．本表記は BPMN [10] のような一般的に利用され



NII-Electronic Library Service

Institute of Electronics, Information, and Communication Engineers

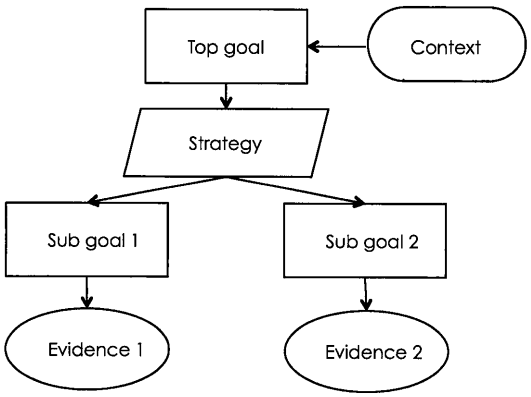


図 3　GSN

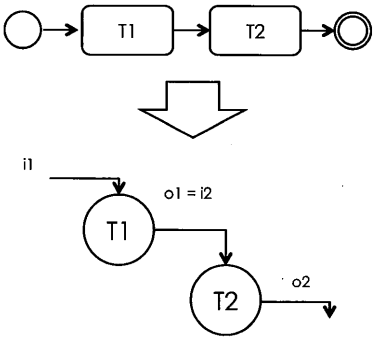


図 4　GSN

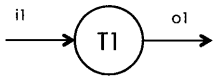


図 5　D-Script からの DFD 記述

ているプロセスモデルとの混乱を起こさないよう，記法自体は同様のものを用いている．図は，２つの D-Task を逐次実行する D-Script のワークフロー図を示す．図の左右にある円形は左からそれぞれ開始イベント，終了イベントを表している．間にある２つの矩形は D-Task に相当する．

D-Script を用いる利点は，後に詳述するとおり，D-Task 毎に保証ケースを細分化して議論できる点である．上のワークフローを例にあげれば，このワークフローの二次リスクへの解決がなされていることを保証するためには，T1 の二次リスク，T2 の二次リスク，それぞれのリスクが解決されていることを保証することで示すことができる．

## 4. 障害対応ワークフローへの保証ケース保証手法

我々は，D-Script で記述された障害対応に潜む二次リスクへの対応を示すため，スクリプト化された障害対応ワークフローを基に，そのリスクを保証ケースへ展開した．以下では，その手法について説明を行う．

### 4.1 保証ケースによる障害対応保証手法

一般的な保証ケースの表記法の一つとして GSN [11] が知られている． GSN はゴール指向の木構造を構築して視覚的にある主張の議論と証跡を示す方法である．

#### 4.1.1 GSN

我々の提案手法は保証ケースの GSN 表記を用いる．そのため，本節では GSN について説明する．GSN(Goal Structuring Notation) とは，ゴール指向を用いた保証ケースの表記法である．図 1 に，GSN で記述された保証ケースを示す．

GSN はゴール指向の記述法で，数種類のノードとエッジから構成される．記述はトップゴールと呼ばれるノードから開始する．トップゴールは，記述された主張がどのような制約の元で実現されるかを記述するコンテキストノードが付随することがある．次に，トップゴールをより細かいゴール (サブゴール) に分割する．この際の分割理由をストラテジーノードで記述する． ストラテジーに従い分割されたノードがサブゴールノードである．最後に，葉ノードとなるサブゴールに対して，その根拠となる具体的な証跡 (エビデンス) を添える．証跡は，ゴールが目的としていることを支持するための論拠となる具体的なものであり，一般には，システムの仕様を記述したドキュメント，テスト結果，実績などが挙げられる．

#### 4.1.2 保証ケースへの二次リスク展開

GSN は視覚表記を持つため，保証ケースでコミュニケーションを取るステークホルダ間でのインタフェースとして適している．保証ケース上でリスク分析を行うには，「あるリスク R が解決されている」と主張するゴールを記述する．このゴールを満たすための証跡は，そのリスク R が発現しても，問題がないことを示すものとなる． 我々は追加する証跡として，設計時のテスト結果や，障害対応スクリプトの実行時情報を用いる事を提案している [12]．本論文では，あるリスク R の証跡として適切なものが予め定義されているものとして議論を進める．

### 4.2 記述手法

4.1.2 の手順に従い，二次リスクの保証ケース記述を行う．具体的には以下の三つの手順から成る．

（1） D-Script で記述される D-Task を一つのプロセスととらえて，データフローダイアグラム (DFD) を記述する．

（2） DFD への入出力及び処理内容のリスクを書き出す．

（3） リスクに対する証跡を取得し，保証ケースに記述する．

以下の節では，それぞれについて説明を行う．

#### 4.2.1 D-Script からの DFD 記述

D-Script は，ワークフロー中の手順として D-Task へ細分化され，記述されている．我々は D-Script で記述されたワークフローに存在する D-Task を DFD でのプロセスとして，その入出力に対するリスクを列挙した．図 4.2.1 に D-Script から DFD への対応付けを示す．DFD で考慮しなければならないのは，$T1, T2$ への入出力データである．図では，$i1, i2, o1, o2$ で示した．次節では，$i1, i2, o1, o2$ に関してリスクを考えていく．

#### 4.2.2 DFD でのリスク分析

DFD が記述されたら，次はプロセスへの入力データ，処理内容，出力データを書き出し，それぞれのリスクを考えていく．図 4.2.2 にその例を示す．プロセス $T1$ の入力データ $i1$ のリスク，$T1$ 内容のリスク，出力データ $o1$ のリスクをそれぞれ表に示す．この作業を DFD 中のすべてのプロセスについて行



NII-Electronic Library Service

Institute of Electronics, Information, and Communication Engineers

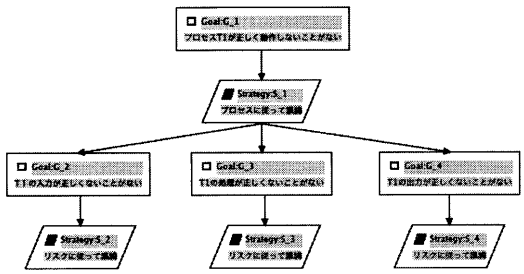


図 6　DFD でのリスク分析

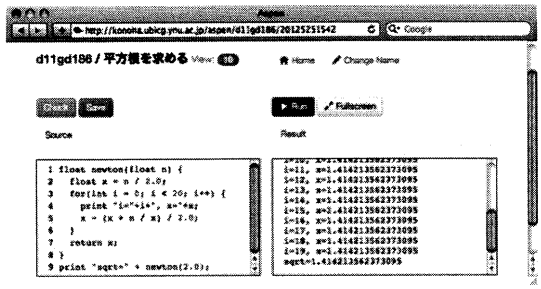


図 7　ASPEN で提供される開発環境画面

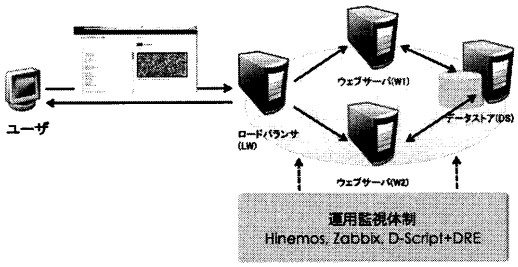


図 8　ASPEN システム構成

う．これを DFD でのリスク分析と呼ぶ．

**4.2.3**　保証ケースへのリスク展開

DFD でのリスク分析で取り出されたリスク群を保証ケースにリスクゴールとして展開する．この際のトップゴールは，「プロセス T1 が正しく動作しないことがない」となる．ゴール記述の方法には，肯定系，否定形の 2 種類があるが，本論文では否定形で統一した．(肯定系，否定形の 2 種類の記述の差異は本論文の議論対象ではないため，割愛する．) トップゴールは，ストラテジー「プロセスに従って議論する」で分割される．この際，できるサブゴールは，先のリスク分析手順に従うため，入力，処理，出力の 3 つとなる．これらそれぞれは「リスクに従って議論する」というストラテジーで分割され，それぞれにリスクゴールが記述されていく．図 4.2.3 にここまでの保証ケースを示す．ここで説明した作業を保証ケースへのリスク展開と呼ぶ．

**4.2.4**　証 跡 提 示

保証ケースにリスク列挙を通じて追加されたリスク群に記述されている D-Script が対応できることを示すためには，証跡が必要になる．先の例では，ファイルシステムが壊れても対応できることを示す証跡が必要である．ひとつの証跡は，冗長化されていることを示す仕様書を掲げることである．どのような証跡が必要になるか，という情報は，リスクキーワードによって異なるため，外部情報源はリスクキーワードと証跡の対応関係を保持しておく必要がある．

D-Script は，証跡として必要となる情報の一部を出力できる機能を持つ．「ファイルシステムが壊れていた」際に対応できるかどうかは D-Script からはわからないが，「(ファイルシステムが壊れていない場合に) 正しくコピーできる」ことへの証跡は，実行前のテスト結果と，実行時のログを証跡として示すことができる．この例のように，証跡の一部は D-Script からの言語支援を行えるため，外部知識はこの組み合わせを正しく保存し，D-Script から取得できる証跡は自動的に保証ケースへ提示することで，作業の一部を機械的に行うことができる．これは，D-Script を用いる大きなメリットとなる．

最後に，証跡候補があっても，例えば経済的な理由から，証跡を示すことが困難な場合は，そのゴールを満たせないというエビデンスを示すことで保証ケースが記述できる．この部分はリスクアセスメント等の作業を通じて，証跡を吟味することが必要となる．

## 5. 事 例 研 究

本節では，我々の運用する実システムで検討された障害対応例をもとに，障害対応の二次リスクについて検討し，二次リスク検討，およびその保証に向けた議論を行う．

### 5.1　対象システム

我々は，プログラミング教育の支援を目的とした，オンラインプログラミング演習システム ASPEN を開発し，大学の情報教育授業で運用している [13]．ASPEN は，プログラミング教育の支援を行う，以下のサービスをウェブサービスとして提供する．

- ウェブブラウザ上でプログラムの編集，コンパイル，実行を可能とするエディタ
- ユーザが記述したプログラムのソースコードのサーバへの保存及びバージョン管理

図 5.1 に ASPEN で提供する開発環境のスクリーンショット示した．画面左側には，テキストエディタ，画面右側にはプログラムの実行結果を表示する画面が表示される．

図 5.1 は，ASPEN のシステム構成である．サーバ側には負荷分散を行うロードバランサ，アプリケーションやコンテンツを提供するウェブサーバ，データ管理を行うデータストアがある．また，Hinemos，Zabbix という統合運用監視ツールを導入し，システムの異常を常時監視する．

### 5.2　障害回復例

ASPEN では，実際にアクセス過多によるサービスダウン障害が発生した．この際，考えられた障害対応案を基に，それぞれ D-Script が記述された．以下にその案を示す．

（1）　内部データベースを git から DBMS へ切り替える

（2）　NFS の設定を非同期へ書き換える

（3）　高性能マシン上に RAM ディスクを作り，ロードバランサで切り替える



NII-Electronic Library Service

Institute of Electronics, Information, and Communication Engineers

| プロセス | 入力リスク | 処理リスク | 出力リスク |
|---|---|---|---|
| P1 | 高性能計算機の応答がない | データが正しく保存されない<br>コピーが正しく動作しない | 高性能計算機の応答がない |
| P2 | 高性能計算機の応答がない<br>ロードバランサの応答がない | 切り替えに失敗しない | ロードバランサの応答がない |

表 1　DFD でのリスク分析結果

ASPEN では，ユーザのソースコード管理に分散コード管理ソフトウェアである git(注1) を用いており，git の運用はそれぞれローカルのファイルシステム上で行われていた．このため，git の負荷は 2 台のウェブサーバ上にかかっていた．このため，ウェブサーバにかかる負荷が大きくなりすぎているのでは，と障害原因予測が行われ，負荷を DBMS サーバへ集中させるためにソースコード管理を DBMS へ任せる，という対策が考えられた．次の案は，ウェブサーバ間でのデータ共有に用いている NFS の同期設定を変更することで，負荷を減らすというものだった．最後の案は，ソフトウェア構成は変更せずに，ハードウェアを代替させるという案だった．最終的には 3 つ目の案が採用されたが，その際に行われた二次リスク分析，およびその保証ケースについて説明する．

### 5.3　D-Script からの DFD 記述

記述された D-Script のワークフロー表記を図 5.3 に示す．

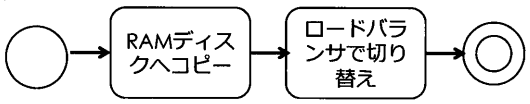


図 9　障害回復案 3 の D-Script

図のとおり，障害対応は 2 つの D-Task から構成される．ここから記述された DFD を図 5.3 に示す．この DFD に基づき，次節ではリスク分析を行う．

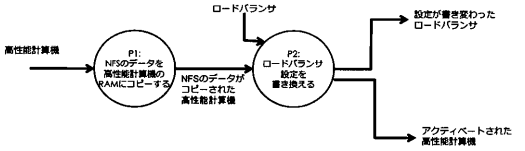


図 10　障害回復案 3 の DFD

### 5.4　リスク分析

表 5.4 は，DFD の各プロセスにおけるリスク分析結果である．表のとおり，DFD で入出力を明確にし，それぞれのリスクを考えていくだけで，表が作成できる．

### 5.5　証跡の提示

証跡はそれぞれのリスクゴールに沿って，リスクゴールを満たすようなものを提示する．リスクゴールを満たすための証跡を示すことができない，もしくは，技術的には可能だがなんらかの理由があり提示が難しい場合は，「証跡なし」として，保証ケース記述を行った．また，D-Script は出力するログから，一部の証跡を示す事ができる．今回の事例では，障害対応案 3 について，データを RAM ディスクへコピーするという D-Task に対して，コピーが正しく行われていることを，事前のテスト結果および，テスト実行中のログを証跡として用いた．

### 5.6　記述された保証ケースと知見

図 5.6 に，本事例で作成された保証ケースを示す．図の通り，目的とする障害対応をトップゴールとし，次にプロセス（タスク）を分割単位として，それぞれの D-task が行う作業の目的をサブゴールとして記述する．次に，DFD への入出力，処理で分割を行い，さらにそれぞれのリスクに沿って分割を行う．最後に，今回の事例で取得した論証をつけて，保証ケースを完成させる．記述した保証ケースは，障害対応の正しさを示しており，二次リスクまでの考慮とその根拠が存在していることをステークホルダへ示すことができる．

今回の経験から得られた知見を以下にまとめる．

- 今回の事例では二次リスクの検討とその対応への保証を保証ケース上で行えた．D-Script と DFD を用いた，保証ケース上での二次リスク議論および保証を行えた．
- 障害対応は，ハードウェアの再起動や入れ替えを伴う可能性があるため，今回の事例では物理的な計算機をデータと捉えることで，リスク分析を行った．しかし一方で，ロードバランサ設定を書き換えるための定数のチェックなど，より細かく記述していく事で詳細なリスク分析が行える可能性があることがわかった．
- 本手法では，隣接するプロセス間での入出力はお互い同じデータを参照しているため，一部保証ケースが冗長になった．保証ケースはステークホルダに提示するため，簡潔であることが望ましいが，今回の事例で作成した保証ケースは，重複部分を統合することで，より簡素化できると考えられる．
- リスクに対して，いくつかの証跡候補が考えられた．これらのうち，ひとつを証跡として残すが，それぞれの証跡への検討結果も保証ケース上に示す事で，より説得力のある保証ケースが記述できると考えられる．

## 6.　関 連 研 究

本節では保証ケースの記述に関する関連研究について述べ，本論文との比較を行う．Hawkins らは保証ケースの一種である Safety case に対して，議論をより綿密に行うことを目的としてゴールと証跡の間にさらに Safety case を記述する手法を提案した [14]．本研究とは，ある目的に対して複数の保証ケースを記述するという点では共通しているが，我々は，議論を深めるためではなく，具体的に証跡が正しく動作することを示すことを目的とした保証ケースを追加している．

## 7.　結　　論

近年，我々の生活を支える情報システムは信頼性向上のため，様々な障害対応策が用意されている．しかし，これら障害対

(注1)：git: http://www.git-scm.com



NII-Electronic Library Service

Institute of Electronics, Information, and Communication Engineers

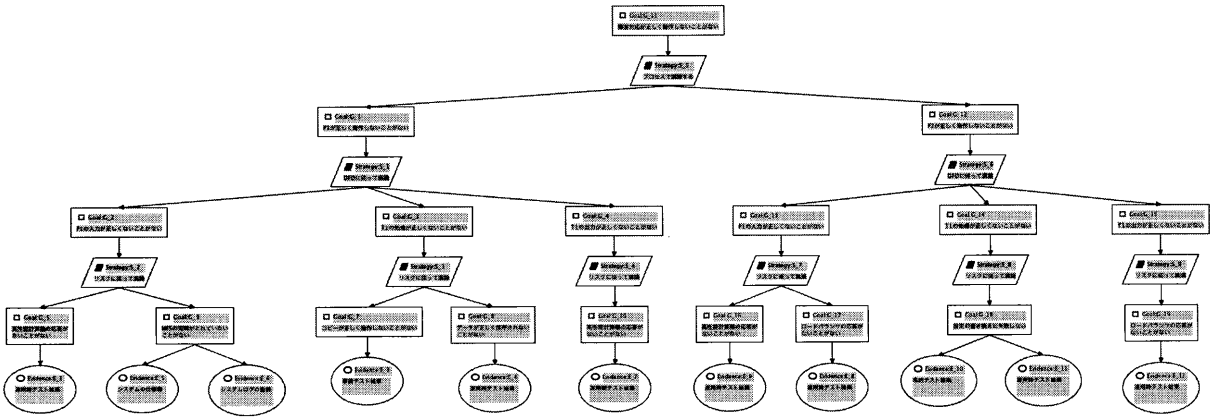

図 11　障害対応案 3 に対して記述された保証ケース

応には，二次リスクが潜んでおり，障害対応が失敗してしまったことで障害が長期化してしまう事例が報告されている．本論文では，障害対応に潜む二次リスクを検討，議論し，解決が行われていることを保証ケースに示す手法を，実際の事例に適用した経験を報告した．本事例では，記述された障害対応ワークフローを対象に，二次リスクの抽出，解決方法の検討などを行い，実際に適用された障害対応案を保証ケース上に示すことができた．

今後は得られた知見を基に，二次リスクに対する保証ケース記述のための機械的支援を検討し，他の事例を含めてその有用性を評価していく．また，本論文では，システムの二次リスクを対象としたが，何次リスクまで議論することが妥当であるかという議論も今後必要であると考えている．

**謝辞**　本研究は,JST/CREST「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」領域の研究助成を受けて行われた．

## 文　献

[1] Google, Google Apps-Gmail Incident Report, February 24, 2009, http://www.google.com/appsstatus/ir/1nsexcr2jnrj1d6.pdf 2009.
[2] ITPro, http://itpro.nikkeibp.co.jp/article/NEWS/20120202/380044/?ST=NC 2009.
[3] M. Tokoro, “Communications and open systems,” IUCS, 2009.
[4] Hinemos, “NTT Data”. http://www.hinemos.info/.
[5] Zabbix SIA, “Zabbix”. http://www.zabbix.com/.
[6] H. Lipson, “Assurance case overview,” Technical report, Carnegie Mellon University, 2007.
[7] T.P. Kelly, “Arguing safety – a systematic approach to managing safety cases,” PhD thesis, York University, 1998.
[8] G. Despotou and T. Kelly, “Extending the safety case concept to address dependability,” 22nd Int’l System Safety Conference, 2004.
[9] 菅谷みどり，岡本悠希，中田晋平，井出真広，若森拓馬，倉光君郎，“分散障害管理のためのアクターベースのスクリプトフレームワーク,” 情報処理学会　第 121 回オペレーティングシステム研究会，2012.
[10] S.A. White, “Using bpmn to model a bpel process,”●●.
[11] T. Kelly and R. Weaver, “The goal structuring notation - a safety argument notation,” Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases, 2004.
[12] 中田晋平，平岡佑太郎，菅谷みどり，倉光君郎，“Dependability case を用いたソフトウェアの運用時検査法,” IPSJ/SIGSE ソフトウェアエンジニアリングシンポジウム 2010，2010.
[13] 若森拓馬，菅谷みどり，倉光君郎，“Aspen: Psp 評価機構を備えた web ベースプログラミング学習システム,” 情報処理学会情報教育シンポジウム 2012，2012.
[14] R. Hawkins, T. Kelly, J. Knight, and P. Graydon, “A New Approach to creating Clear Safety Arguments,” Advances in Systems Safety, eds. by C. Dale and T. Anderson, chapter 1, pp.3–23, Springer London, London, 2011.



NII-Electronic Library Service