人工知能学会
第13回知識流通ネットワーク研究会
SIG-KSN-014-03

議論パターンによるシステム高信頼性保証知識の再利用

山本 修一郎 猿渡卓也

名古屋大学
愛知県名古屋市千種区不老町

Reuse of System Assurance Knowledge with Argument Patterns

Shuichiro YAMAMOTO Takuya SARUWATARI

Nagoya University
Furo-cho, Chikusa-ku, Nagoya Aichi Japan

概要

システムの高信頼性を保障するために，保証ケース（アシュアランスケース）の利用が提唱されている．しかし，保証ケースの作成知識が十分に流通していないため，システム開発現場への保証ケースの導入が進んでいない. 本稿では, 保証ケース作成知識を再利用するための議論パターンの体系的な構成法を提案する. また, 議論パターンの構築事例についても述べる.

Abstract

To assure system dependability in the organization, it is necessary to share assurance information on system development and operation activities appropriately. In this paper, a category of assurance argument patterns is proposed to record and share system assurance argument patterns based on the argument structure of assurance cases. We also explain that the category system works well to gather systematically assurance case patterns.

1 はじめに

システムの安全性を確認するために，安全性ケース(Safety case)，アシュアランスケース（Assurance case，保証ケース）やディペンダビリティケース（Dependability case，D-Case）が注目されている[1][2][3][4][5][6]．このため GSN(Goal Structuring Notation)を用いてこれらを記述する方法が提案されている[1][2][5].

システムのディペンダビリティを保証ケースに基づいて確認するためには，システム開発・運用の構造を反映した保証ケースが必要となる．たとえば，UML を用いたシステム開発文書に対して, 保証ケースを作成する必要がある．しかし，これまでのディペンダビリティケースでは, UML を前提としたディペンダビリティケースの作成手法が明確ではないという問題があった．このため，UML によるシステム開発文書のうち, とくにユースケース図に着目して, 具体的にディペンダビリティケースを作成する方法を提案した[18]. このように, ディペンダビリティを確認する対象の構造に基づいて保証ケースを作成する場合，典型的な議論分解パターンが共通に出現する．また，筆者らが実践している D-Case の研修会では，D-Case による議論分解についての参加者からの質問には共通事項があることが分かってきた．このため, 本稿では, 既存の議論分解パターンに加えて, 筆者らが識別した議論分解パターンを体系的に分類整理する分類法を提案するとともに，その有効性について考察する．

なお本稿では，安全性ケースやアシュアランスケース，ディペンダビリティケースを総称して保証ケースという用語を用いる．

以下では，まず第 2 節で本研究の背景と保証ケースの議論分解パターンの例を説明する. 第 3 節では, 保証ケース作成法の必要性について述べる．第 4 節で議論パターン分類法を提案する．第 5 節で分類法に基づく議論パターンの分類結果を明らかにする．第 6 節で，実験結果に基づいて議論分解パターンの有効性と限界について考察する．最後に第 7 節でまとめと今後の課題について述べる．

2 研究の背景

保証ケースの研究動向については既存の報告を参照されたい[17]. 保証ケースの議論パターンについて

は，従来から研究されている[7][8][9][10][11][12][13][14]．議論パターン分類についてはBloomfieldの分類[15]がある．この内容を表1に示した．しかし，簡単な説明があるだけで具体的な内容については明確にされていない．

表 1　議論分解パターン

項番	パターン	説明
1	アーキテクチャ分解	システム構成に従って分解
2	機能分解	主張を機能構成に従って分解
3	属性分解	特性を複数の属性に分解
4	帰納分解	説明対象の場合分けによる分割
5	完全分解	説明対象のすべての要素による分割
6	単調分解	新システムによる旧システムの改善点による分解
7	修正分解	曖昧性の明確化による分解

しかし，保証ケースの議論パターンが具体化されていなかったため，筆者らはソフトウェア・アーキテクチャを記述するためのパターン体系[16]に基づいて，議論パターンを説明するための記述項目を提案することにより議論パターンを説明している[17][18][19]．また議論パターンの適用実験も実施している[20][21][22]．なお，議論パターンのカタログ[23]でも同様の記述項目が整理されている．しかし，議論パターンを分類するカテゴリについては明らかではない．

3 保証ケース作成法の必要性

システム開発運用工程だけでなく，これらの工程生産物を用いた保証ケースの作成手順の具体化が必要である．この理由は，保証ケースの作成手順が，断片的な取り組みにとどまっており，システム開発プロセスや工程生産物を利用する系統的な手順が明確になっていなかったからである．

また故障解析（FTA）や FMEA 分析，Hazard 分析などのリスク分析技術がある．前述したように逸脱分析を用いて保証ケースを作成する手法も提案されている．しかし，開発運用プロセス全体を通じた保証ケースとリスク分析の具体的な適用関係は必ずしも明確になってはいない．

したがってサービスに対する統一的な分析手法として確立されてはいないという問題があった．

また，保証ケースが前提とする「証跡に基づく論理的な議論による妥当性の論証」という考え方は，日本の文化にはなじまないところがある．この理由は，このような論証の基礎的な訓練が日本では不足しているからである．これは，これまで数年にわたって GSN を技術者に教育してきた筆者らの経験に基づく仮説である．逆にいえば，論証についての知識とスキルがあれば，保証ケースを作成することは難しいことではないともいえる．

このため保証ケースを開発者が作成しようとすると，なにを論証すべきかどのように論証すべきか，なにが証跡なのか分からず初心者が躓くことになる．たとえば, Kelly による 6 段階作成法の最初でゴールを識別するところから悩むことになる．

したがって保証ケースを記述するために GSN の構文を教えただけでは，論証に慣れていない日本の現場における技術者が保証ケースを作成することは困難である. このため, 本連載でも紹介したように，日本では欧米で開発されてきた保証ケースの作成手法よりも具体的な適用対象に特化した手法が必要になる．

3.1 基本的な課題

保証ケースを作成するためには，主張（ゴール），戦略，コンテクスト，証跡とそれらの関係を記述する必要がある保証ケースを作成する際には，まずディペンダビリティについてシステムが満たすべき主張を列挙する必要がある．このために，戦略ノードを用いて主張を下位主張に分解することになる．

この場合, 次のような基本的な疑問が生じることが多い．

① 主張として何をどう書くのか
② 戦略に何を書くのか
③ 戦略で分解する幅をどこまで広げるのか
④ コンテクストに何を書けばいいのか
⑤ 証跡に何を書けばいいのか
⑥ 木構造をどこまで深くするのか
⑦ コンテクストと証跡の関係をどのように分析すればいいのか

これらの疑問に答えるためには，適用対象分野を限定することにより，分野に即した保証ケースの階層構造と構成要素に記述すべき内容を予め規定しておく方法が有効である．

しかし，対象分野が限定できない場合には，より一般的な方法が必要となる．たとえば開発文書や運用保守文書などの既存文書に基づいて，保証ケースを作成する方法が考えられる．このような既存文書に基づく方法の利点としては，指定された文書の構造や内容によって作成すべき保証ケースの構造と構成要素に記述すべき事柄を明確化できることである．

たとえば, 既存文書の章節を証跡として用いる方法が提案されている．またリスク分析で作成される故障木や FMEA, HAZOP などの文書がコンテクストや証跡で用いられている．

3.2 議論構造と制御構造の混同

保証ケース構文の理解不足から，初心者が作成し

た保証ケースでは以下のような誤りが多い．

① 戦略をゴールと取り違えている
② 主張として書くべき内容が命題になっておらず実行文や機能文になっている．
③ 戦略を判断分岐だと誤解する
④ 議論ではなく機能の実行順序で分解している

このような取り違えは，GSN 構文がフローチャートに似ていると考えるからであると思われる．たとえば主張が矩形であることから実行文に対応させ，戦略が平行四辺形であることから判断文に対応させていると思われる．さらに GSN では矢印が上から下に向かうので，フローチャートの実行制御の流れと混同するのであろう．GSN の構造を既知であるフローチャートの構造で理解してしまうことからくる誤解である．

とくに，戦略を用いるときに，場合分け，判断，代替案についての論証と，これらを区別できないことが多い．このため，これらに対する議論を例示して教育しておく必要がある．

3.3 記述範囲の制御

安全性ケース開発マニュアル[24]で指摘されているように，保証ケースで確認する境界範囲を定義することが重要である．このような境界範囲を決めないと，どこまでも際限なく確認することになる．また確認範囲の十分性を保証できない．たとえば，図１で示した保証ケースでは列車運行の安全性を業務上の危険行為，危険な現場，危険な自然現象についての対策を確認している．しかし，列車自体の整備や整備業務についての危険性についての対策までは確認していない．したがって保証ケースの記述では，予め記述して確認する範囲についての合意が必要である．そうしないと，上述したように確認範囲の十分性を保証できなくなるからである．

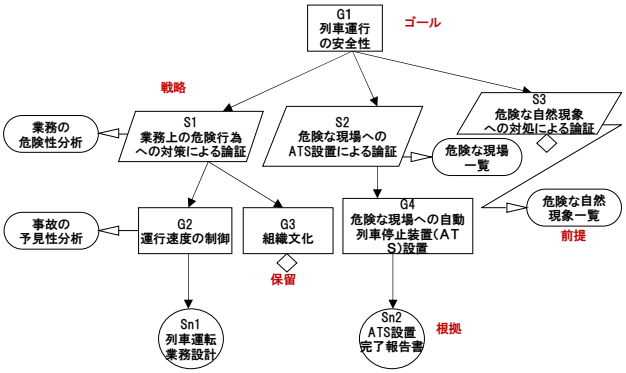

図１　列車運行の安全性についての保証ケース

3.4 議論分解パターン

保証ケースのパターン化の研究として，保証ケースに出現する名票のパラメータ化や分解要素数のパラメータ化が提案されている[7][8]．たとえば，議論パターンを一般的に記述するために，表 1 のような 4 種類の記号が提案されている．

表１　議論分解の一般化のための表記法

項目	記号	説明
要素の具体化	△{system X}	括弧内の名称を具体化
具体化と作成	△ ▽	要素を具体化して下位の GSN を作成
関係展開	●n(n=#機能)	下位の主張を n 個に分解
選択	◆1 of 2	下位の主張のいずれか一方を選択

図 2 では，この記法を用いてアーキテクチャに基づく保証ケースの記述例を示している．

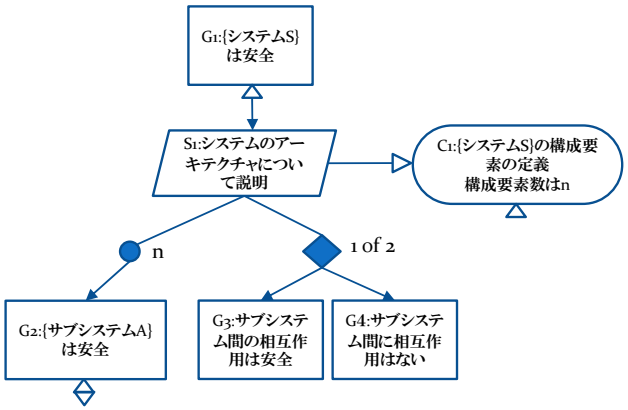

図２　パラメータ化された保証ケースの例

しかし，このような保証ケースの一般化記法だけでは，個別的な事例を一般的に記述することはできても，系統的に保証ケースのパターンを収集することはできない．

4 議論分解パターンの分類

以下では，上述した保証ケースのパターンを体系的に収集するための分類法について説明する．

4.1 分類法

議論分解パターンでは，共通的な議論構造を具体的なパターン事例で整理する．

保証ケースを作成する場合，①保証ケースによってディペンダビリティを確認しようとする対象と，②保証ケースによって議論しようとする説明，③保証ケースで用いられる証拠が必要である．また，④すでに説明が合意された保証ケースを再利用できる．

したがって，保証ケースのパターンには，表 2 で整理したように，保証ケースでディペンダビリティを確認しようとする対象についてのパターン，保証ケースによる説明についてのパターン，証拠のパターン，再利用のありかたについてのパターンがある．

表 2　保証ケースパターンの分類

分類	説明
対象分解	対象物の構成や参照モデルに基づいて主張を分解
説明条件分解	対象についての条件や背理法や帰納法などの推論条件に基づいて主張を分解
証拠分解	主張を証拠によって分解（説明）
再分解	既存保証ケースを用いて主張を分解

対象分解パターンについては，対象の記述法に基づく対象記述分解パターンと，対象の共通構造に基づく参照モデル分解パターンがある．説明条件分解パターンについては，対象についての条件分解パターンと，推論手法に基づく推論分解パターンがある．このことを整理すると，図 3 のようになる．

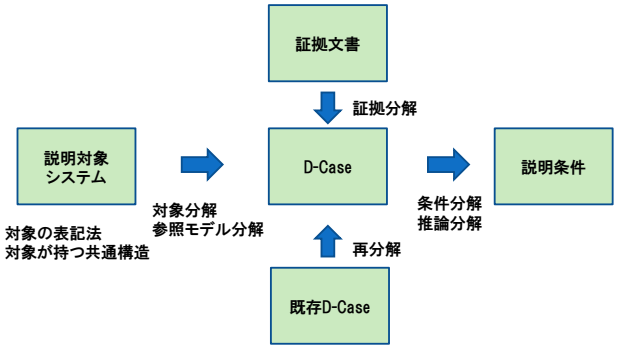

図 3 保証ケース分解パターンの構成

5 具体例

以下では上述した分類法に従って，収集した保証ケースパターンの具体例を説明する．

5.1 対象分解パターン

対象分解パターンとして，以下の 15 パターンを収集した．

[アーキテクチャ分解] システム構成に従って主張を分解

[機能分解] 機能構成に従って主張を分解

[属性分解]複数の属性に従って主張を分解

[完全分解] 説明対象のすべての要素によって主張を分割

[プロセス分解] プロセスの入力，処理，出力に対して主張を分解

[プロセス関係分解] プロセスの先行後続関係に従って，主張を分解

[階層分解] 対象の階層構成に従って，主張を分解

[DFD 階層分解] データフロー図(DFD)の階層構成に基づいて，主張を分解

[ビュウ分解] UML のビュウ構成に基づいて，主張を分解

[ユースケース分解] ユースケースの定義に従って，主張を分解

[要求記述分解] 要求の記述項目に従って，主張を分解

[状態遷移分解] 状態遷移の定義に従って，主張を分解

[運用要求記述分解] 運用要求定義票に従って，主張を分解

[シーケンス分解] シーケンス図に従って，主張を分解

[ビジネスプロセス分解] ビジネスプロセスモデル記法（BPMN）に従って，主張を分解

5.2 参照モデル分解パターン

対象分解パターンとして，以下の 10 パターンを収集した．

[リスク対応分解] システムリスク参照モデルに基づいて，主張を分解

[DEOS プロセス分解] DEOS プロセスに基づき，主張を分解

[組込み参照モデル分解] 組込みシステム参照モデルに基づいて，主張を分解

[CC 分解] セキュリティのコモンクライテリア(CC)に基づき，主張を分解

[要求仕様記述分解] 要求文書の章構成に対して，主張を分解

[システム境界分解] システム境界に基づいて，主張を分解

[ITIL プロセス分解] ITIL プロセスに基づいて，主張を分解

[非機能要求指標分解] 非機能要求品質指標に基づき主張を分解

[テスト項目分解] テスト項目参照モデルに基づき，主張を分解

[問題フレーム分解] 問題フレームのパターンに基づき，主張を分解

5.3 条件分解パターン

条件分解パターンとして，以下の 7 パターンを収集した．

[ECA 分解] イベント，条件，活動に対して主張を分解

[条件判断分解] 条件判断に対して，主張を分解

[代替案選択分解] 代替案の選択肢に対して，主張を分解

[矛盾解決分解] 矛盾とその解決策に対して，主張を分解

[均衡分解] 互いに依存・対立する属性が均衡するように，主張を分解

[改善分解] 新システムによる旧システムの改善点による分解

[精緻分解] 曖昧性の明確化による分解

5.4 推論分解パターン

推論分解パターンとして，以下の 4 パターンを収集した．

[帰納推論分解] 数学的帰納法によって，主張を分割

[消去分解] 消去法によって，主張を分解

[否定推論分解] 消去法によって，主張を分解

[三段論法分解] 三段論法によって，主張を分解

5.5 証拠分解パターン

証拠分解パターンとして，以下の 11 パターンを収集した．

[レビュ分解] レビュ結果を証拠として，主張を確認

[評価分解] チェックリストや投票の結果を証拠として，主張を確認

[欠陥モード分析分解] 対象物の欠陥モード分析に基づき主張を確認

[テスト分解] テスト結果を証拠として，主張を確認

[証明分解] 形式的証明を証拠として，主張を確認

[モデル検査分解] モデル検査結果を証拠として，主張を確認

[シミュレーション分解] モデル検査結果を証拠として，主張を確認

[合意分解] 合意文書を証拠として，主張を確認

[モニタ分解] 監視結果を証拠として，主張を確認

[文書分解] 開発や運用に関する文書を証拠として，主張を確認

[法制度分解] 規格や法制度についての文書を証拠として，主張を確認

5.6 再分解

再分解パターンとして，以下の 2 パターンを収集した．

[水平分解] 互いに独立に具体化されている複数の分解を用いて，共通する主張の下で分解されている複数の下位の主張を分解

[垂直分解] 具体化された分解を用いて，一つの主張を分解

6 考察

以下では，上述した議論パターンの有効性について考察する．

6.1 分類の妥当性

提案した議論パターン分類を用いることにより，49 個の議論パターン知識を，対象，説明条件，証拠，再利用の 4 側面から系統的に登録できることが明らかになった．これによって，保証ケース作成知識の一貫性のある管理とそれに基づく効率的な共有が実現できる．

提案した分類法では，対象，説明条件，証拠，再利用という 4 側面で議論パターンを分類している．この 4 側面だけで分類の観点が充分であるかどうかについては，より多くの例に適用して評価する必要がある．

6.2 議論パターン・コミュニティ

前節で示したように，本提案によって，保証ケース作成上の議論パターン分解知識を活用できるので，保証ケースの作成を効率化できるだけでなく，系統的に議論パターンを抽出整理できることから，組織内外で保証ケース作成知識を共有できることは明らかである．

しかし，議論パターンによる高信頼性保証知識を共有するためには，分類手法だけでなく，議論パターンを活用する高信頼性保証技術者のコミュニティ活動が必要である．

6.3 議論パターン・リポジトリ

本提案による議論パターンの分類法に基づいて，議論パターンを管理するリポジトリを構築できる．たとえば，議論パターンリポジトリのメタモデルは図 4 に示すように構成できる．

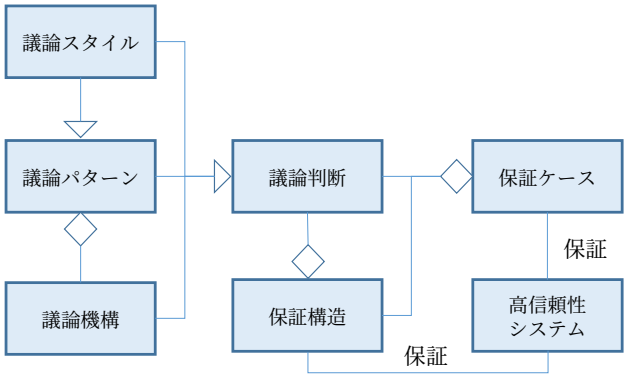

図 4　議論パターンのメタモデル

本稿で提案した議論パターンの分類構造は議論判断と保証構造を扱っている．すなわち，4 つの観点が保証構造の種類を示していると考えられる．しかし，どのような判断の下で議論パターンを選択するかについては明示的に整理できていない．また，議論スタイルならびに議論機構については参照モデルに基づく議論パターンが対応している．ここで，議論スタイルは複数の議論パターを包括する上位レベルの議論パターンである．また，議論機構は議論パターンを構成する主要な議論判断のことである．今後，議論パターンのリポジトリを構築する上で，保証構造と議論判断の関係ならびに，議論スタイルや議論機構と，参照モデル議論パターンとの関係についても検討する必要がある．

6.4 限界

本稿で提案した議論パターンの分類の有効性について評価する必要がある．たとえば，どのような能力の要員がどれくらいの工数で議論パターンを理解活用し，自ら議論パターンを作成・登録できるかなどの生産性や品質に関する有効性について，定量的に評価する必要がある．

7 まとめと今後の課題

本稿では，議論パターンを共有するための分類手法を提案した．また，提案手法によって具体的な議論パターン 49 個を分類できることを明らかにした．この結果から，この分類手法に基づいて，他の議論パターン事例についても収集・分類できると考えている．また，本稿の分類手法を定量的に評価するまでには至っていない．今後，実際のソフトウェア開発・運用工程を対象とする議論パターン分類手法の評価実験を進める必要がある．

謝辞

本研究は 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」研究領域（DEOS プロジェクト）の支援を受けたものである[26][27] [28].

参考文献

[1] Kelly, T. P, A Six-Step Method for the Development of Goal Structures, York Software Engineering, 1997
[2] T. Kelly. “Arguing Safety, a Systematic Approach to Managing Safety Cases”. PhD Thesis, Department of Computer Science, University of York, 1998
[3] J. A. McDermid. Software safety: where's the evidence? In SCS '01: Proceedings of the Sixth Australian workshop on Safety critical systems and software, pages 1-6, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.
[4] I. Bate, T. Kelly, Architectural considerations in the certification of modular systems, Reliability Engineering and System Safety 81, pp.303–324,2003
[5] Tim Kelly and Rob Weaver, The Goal Structuring Notation – A Safety Argument Notation, Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases, July 2004
[6] Despotou G., Kelly T., Extending the Concept of Safety Cases to Address Dependability. In proceedings of the 22nd International System Safety Conference (ISSC), Providence, RI USA, proceedings by System Safety Society 2004.
[7] T. Kelly and J. McDermid, Safety Case Construction and Reuse using Patterns, 1998
[8] Robert Alexander, Tim Kelly, Zeshan Kurd, John McDermid, Safety Cases for Advanced Control Software: Safety Case Patterns, 2007
[9] A. Wardzinski, Safety Argument Strategies for Autonomous Vehicles, 2008
[10] Ewen Denny and Ganesh Pai, A Lightweight Methodology for Safety Case Assembly, 2012
[11] A. Hauge and K. Stolen, A Pattern-Based Method for Safe Control Systems Exemplified within Nuclear Power Production, 2012
[12] A. Ruiz, I. Habli, H. Espinoza, Towards a Case-Based Reasoning Approach for Safety Assurance Reuse, 2012
[13] P. Graydon, T. Kelly, Assessing Software Interference Management When Modifying Safety-Related Software, 2012
[14] E. Denny and G. Pai, Formal verification of a safety argumentation and application to a complex UAV system AdvoCATE: An Assurance Case Automation Toolset, 2012
[15] Robin Bloomfield and Peter Bishop, Safety and Assurance Cases: Past, Present and Possible Future – an Adelard Perspective, 2010
[16] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sornmerlad, Michael Stal, Pattern-Oriented Software Architecture : A System Of Patterns, 1996, John & Wiley & Sons, Ltd.
[17] 松野裕，高井利憲，山本修一郎，D-Case 入門，～ディペンダビリティ・ケースを書いてみよう！～，ダイテックホールディング, 2012，ISBN 978-4-86293-079-8
[18] 松野裕, 山本修一郎, 実践 D-Case,～ディペンダビリティ・ケースを活用しよう！～, ダイテックホールディング, 2013，ISBN 978-4-86293-091-0
[19] 山本修一郎, 松野裕，ディペンダビリティケース作成法に関する一考察，KBSE研究会，IEICE-112, vol. IEICE-SS-164, No. IEICE-KBSE-165, pp.61-66, 2012
[20] 松野裕, 山本修一郎,ユースケース分析に基づくディペンダビリティケース作成法の提案, KBSE研究会，信学技報, IEICE-112(419), pp.19-24, 2013
[21] S. Yamamoto, Y. Matsuno, An Evaluation of Argument Patterns to Reduce Pitfalls of Applying Assurance Case, pp.12-17, Assure 2013
[22] 山本修一郎, 議論分解パターンに基づくディペンダビリティケースの作成実験, KBSE研究会，信学技報, IEICE-112, vol. IEICE-SS-164, No. IEICE-KBSE-165, pp.61-66, 2013
[23] R. Hawkins and T. Kelly, A Software Safety Argument Pattern Catalogue, YCS-2013-482, 2013
[24] European Organisation for the Safety of Air Navigation, *Safety Case Development Manual*, 2nd Edition, EUROCONTROL, 2006
[25]Donald G., Firesmith, et.al., The Method Framework for Engineering System Architecture, CRC Press. 2009.
[26] DEOSプロジェクト, http://www.crest-os.jst.go.jp
[27]DEOS プロジェクト，2011 科学技術振興機構 White Paper DEOS-FY2011-WP-03J, *www.dependable-os.net/ja/topics/file/**White_Paper**_V3.0J.pdf*
[28] Mario Tokoro eds., Open Systems Dependability, Dependability Engineering for Ever-Changing Systems, CRC Press, 2012