名古屋大学

NAIST

D-Case入門

ディペンダビリティ・ケースを書いてみよう！

松野裕 山本修一郎
名古屋大学
高井利憲
奈良先端科学技術大学院大学

DEOS

© Nagoya University 2012
© Nara Advanced Institute of Science and Technology 2012

まえがき

- 本入門コースでは、システムのディペンダビリティを達成する手法として，JST CREST DEOSプロジェクトの一環として研究開発を進めているD-Case手法について分かりやすく解説する
  - D-Caseとは、近年欧米で注目されているアシュアランスケースを拡張した手法である
- 本入門コースの受講対象者は、システム開発、運用の実務を担当されている企業の方ならびに，これらの業務に興味のある大学生、大学院生である
- D-Case手法を理解するための前提知識には、ディペンダビリティ、要求工学、リスク分析などがある．本コースではこれらについても，できるだけわかりやすく説明するように努めている
- なお，受講された皆さんのご意見・ご感想は，本入門コースを改善する上で大変貴重であるので、ぜひともご協力をお願いしたい

© Dependability Science Lab. Nagoya Univ. 2012

背景編

1. オープンシステムズ・ディペンダビリティ
    1. 基本概念: システムとディペンダビリティ
    2. オープンシステムズ・ディペンダビリティ
2. 要求工学概論
    1. 要求工学とは
    2. IEEE Std830-1998
    3. ISO/IEC/IEEE 29148
    4. 組込みシステム参照モデル
3. ディペンダビリティ標準規格
4. システムリスク分析
    1. システム障害事例
    2. 独立検証確認
    3. サービス継続性
    4. リスク分析手法

© Dependability Science Lab. Nagoya Univ. 2012

基礎編

5. アシュアランスケース概論
   1. アシュアランスケース
   2. GSN(Goal Structuring Notation)
   3. アシュアランスケースの現況
   4. 関連規格、ツールなど
6. D-Case作成法
   1. D-Caseとは?
   2. 記述基本ルール
   3. 記述ステップ
   4. 記述例
   5. 既存のガイドライン、関連ガイドライン
   6. まとめ: D-Case 10箇条

© Dependability Science Lab. Nagoya Univ. 2012

応用編

7. D-Case演習
    1. GSN構造規則
    2. 構造誤り
    3. GSN構造の省略
    4. 要素の命名規則
    5. GSNの作成
    6. 問題回答の指針
8. 議論分解パターン
    1. 議論分解パターンの基礎
    2. 議論分解パターンの応用
    3. 議論のレビュー方法
9. D-Case Editorの使い方

© Dependability Science Lab. Nagoya Univ. 2012

執筆分担

- 1章、5章、6章、9章
  - 松野裕
- 3章
  - 高井利憲
- 2章、4章、7章、8章
  - 山本修一郎

© Dependability Science Lab. Nagoya Univ. 2012

名古屋大学

D-Case入門
1 オープンシステムズ・ディペンダビリティ

© Dependability Science Lab. Nagoya Univ. 2012

内容

- はじめに
- 基本概念
  - システムとディペンダビリティ
- オープンシステム・ディペンダビリティ
- まとめ
- 参考文献

© Dependability Science Lab. Nagoya Univ. 2012

はじめに

- 本講義は、私達の身の回りにあるシステム、特に情報システムを、どのようにしたら、ディペンダブルに開発、運用できるか議論する
- ディペンダブルなシステムとは、一言で言うなら利用者が安心し、使いたいと思えるシステムである

© Dependability Science Lab. Nagoya Univ. 2012

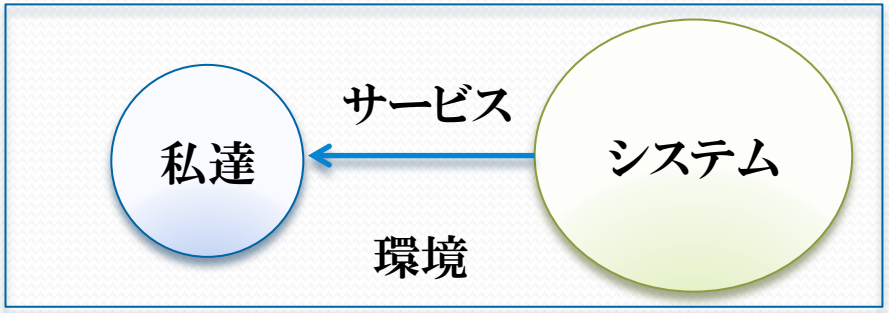
内容

- はじめに
- 基本概念
  - システムとディペンダビリティ
- オープンシステム・ディペンダビリティ
- まとめ
- 参考文献

© Dependability Science Lab. Nagoya Univ. 2012

システムとは

- 本講義では以下の定義とする
  - 私達の身の回りにある、私達に何か（サービス）をしてくれる物や人を抽象化した概念
  - システムは、環境におかれる
  - サービスとは？
    - システムによって、私達に提供される物
    - 私たちは、サービスを介して、システムと関係を持つ

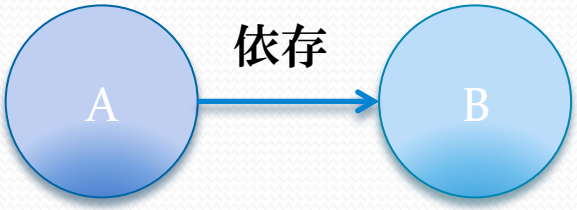


System [1]: an entity that interacts with other entities, i.e., other systems, including other hardware, software, humans, and the physical world with its natural phenomena.
Service [1]: a behavior of the system as it is perceived by its user(s).

© Dependability Science Lab. Nagoya Univ. 2012

ディペンダビリティとは

- 由来は、あるシステムが、別のシステムにDepend (依存)することができる、システムの性質



システムAが、システムBに依存できるための
システムBの性質=Bの(Aにとっての)ディペンダビリティ

- 例
  - A = 運転手、B = 自動車としたとき、自動車はどういった性質を持てばディペンダブルといえるのか？
    - 安全である、かっこいい、運転しやすい、カーナビがついている？

© Dependability Science Lab. Nagoya Univ. 2012

規格におけるディペンダビリティの定義

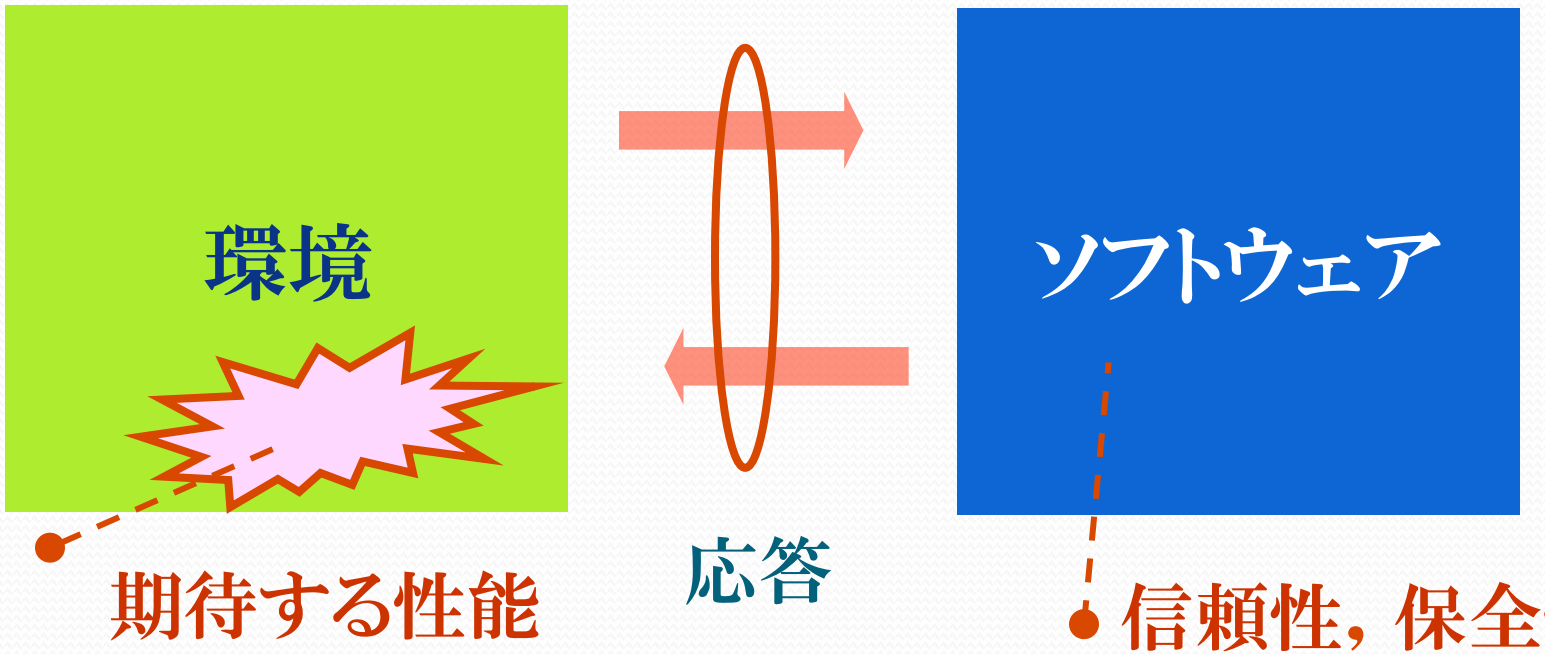
人間行動
コンポーネント
法制度
自然物理現象

ディペンダビリティ

JIS Z 8115

アベイラビリティ性能及びこれに影響を与える要因，すなわち信頼性性能，保全性性能及び保全支援能力を記述するために用いられる包括的な用語

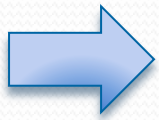
イベント



© Dependability Science Lab. Nagoya Univ. 2012

ディペンダビリティの条件

- システムはどのような性質を持てば、ディペンダブルといえるのか？
  - 利用者、環境によって異なる(要求が異なる)
- システムが、その性質を満たさなくなった場合、どうすればよいか？
  - システムはその性質を満たすよう復旧を行い、サービスを継続しなければならない



システムがディペンダブルであるためには、
利用者、環境において望まれる性質を持つづけ、
サービスを継続しなくてはならない

© Dependability Science Lab. Nagoya Univ. 2012

ディペンダビリティの体系、用語

- IFIP WG 10.4 “Dependable Computing and Fault Tolerance”などにより、ディペンダビリティの体系、用語がまとめられてきた(1980~)
  - Fault Tolerance (耐故障性)がディペンダビリティ研究の元のひとつ
  - 近年では、セキュリティとあわせて、まとめられている
- 以下では、参考文献1によるディペンダビリティの体系、用語を簡単に紹介する
  - 参考文献1: Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transaction on Dependable and Secure Computing, Vol. 1, No. 1, 2004

© Dependability Science Lab. Nagoya Univ. 2012

ディペンダビリティの用語

- ディペンダビリティ属性
  - システムがディペンダブルであるために持つべき属性
- ディペンダビリティへの脅威
  - システムがディペンダブルであることを脅かす物、出来事
- ディペンダビリティへの脅威に対処する手段

© Dependability Science Lab. Nagoya Univ. 2012

ディペンダビリティ属性

- Availability(可用性)
  - 正しいサービスの即応性
- Reliability(信頼性)
  - 正しいサービスの継続性
- Safety(安全性)
  - 利用者と環境へ破壊的影響をもたらさないこと
- Integrity(一貫性)
  - 不適切なシステム変更がないこと
- Maintainability(保守性)
  - 変更と修理を受け入れられること

ディペンダビリティは、Availability、Reliabilityなどを統合した概念として議論されてきた

© Dependability Science Lab. Nagoya Univ. 2012

ディペンダビリティへの脅威

- Fault(欠陥): Errorの原因となるとみなされる・推定されるもの、こと
- Error(誤り): Failureが起こりうるシステムの状態(エラー状態になったからといって、Failureが起こるとは限らない)
- Failure(障害): サービスが正しいサービスから逸脱する出来事
  - サービスはシステムの機能を実現しているとき、正しいサービスと呼ばれる

© Dependability Science Lab. Nagoya Univ. 2012

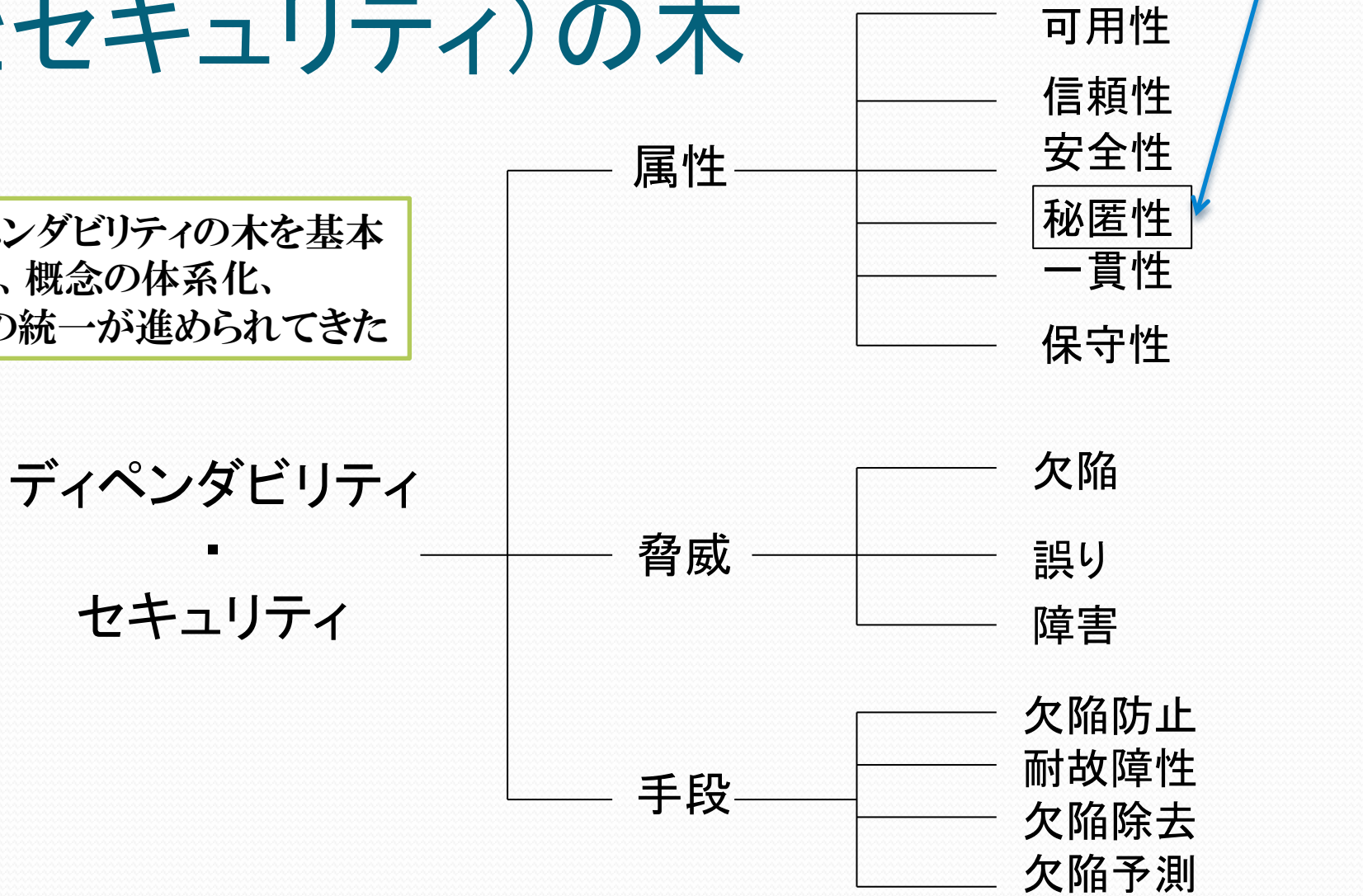
脅威への対処手段

- Fault Prevention(欠陥防止): Faultの導入や発生を防ぐ
- Fault Tolerance(耐故障性): Faultがある中で障害を防ぐ
- Fault Removal(欠陥除去): Faultの数、深刻度をへらす
- Fault Forecasting(欠陥予測): Faultの現在の数、今後の障害、影響などを予測する

© Dependability Science Lab. Nagoya Univ. 2012

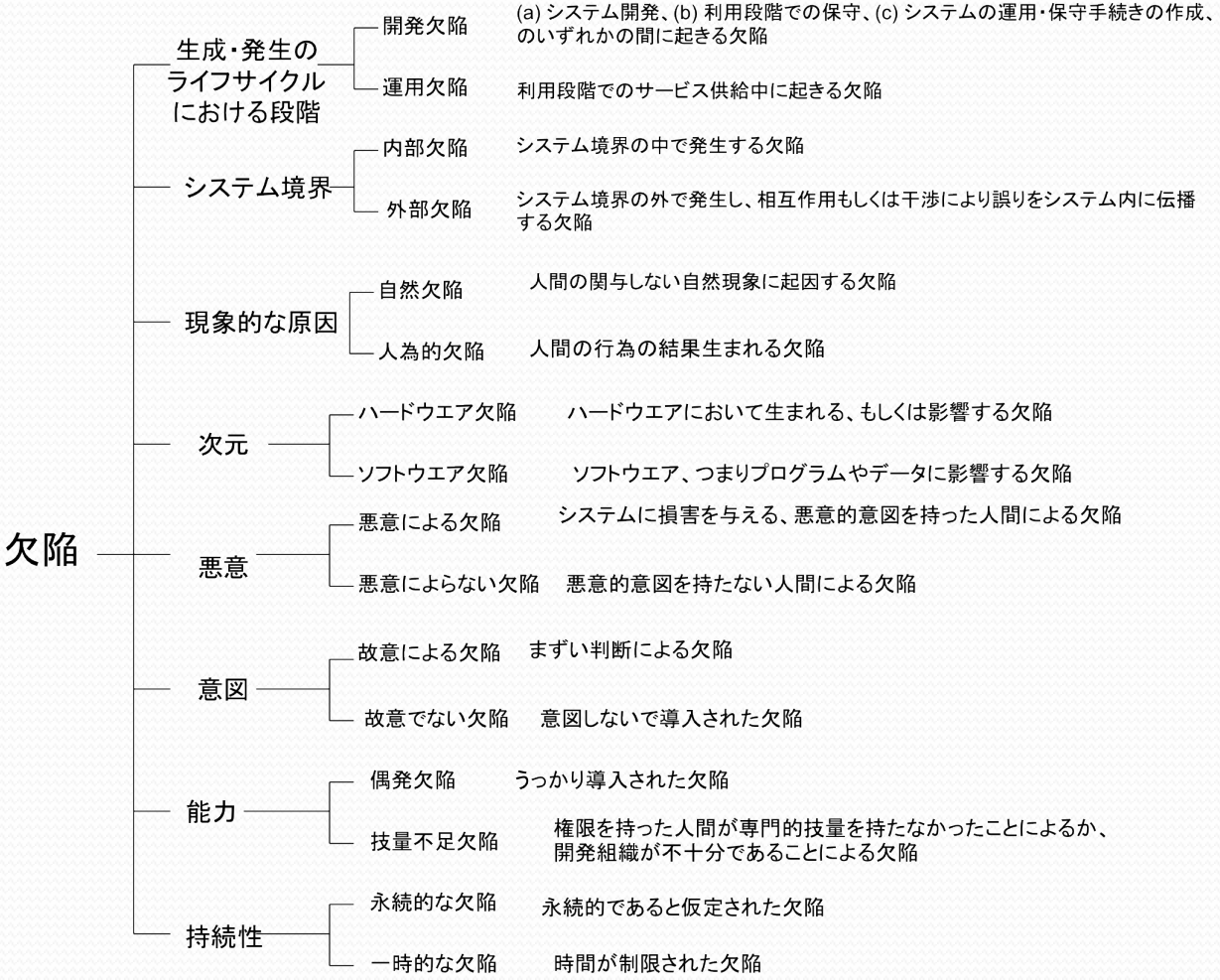
ディペンダビリティ（とセキュリティ）の木

ディペンダビリティの木を基本として、概念の体系化、用語の統一が進められてきた



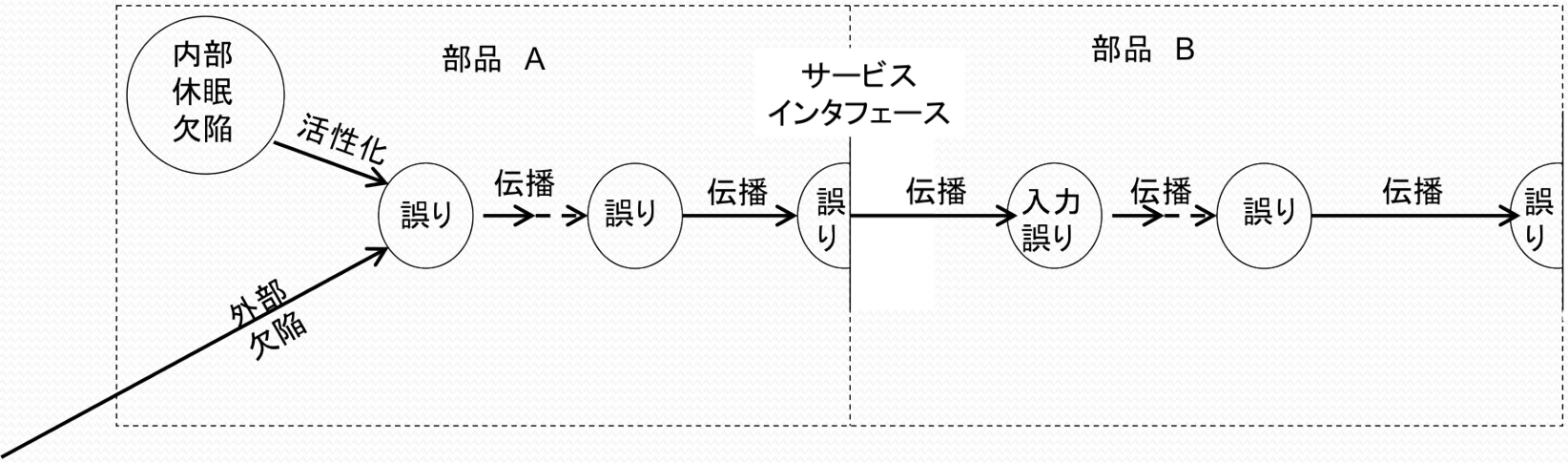
© Dependability Science Lab. Nagoya Univ. 2012

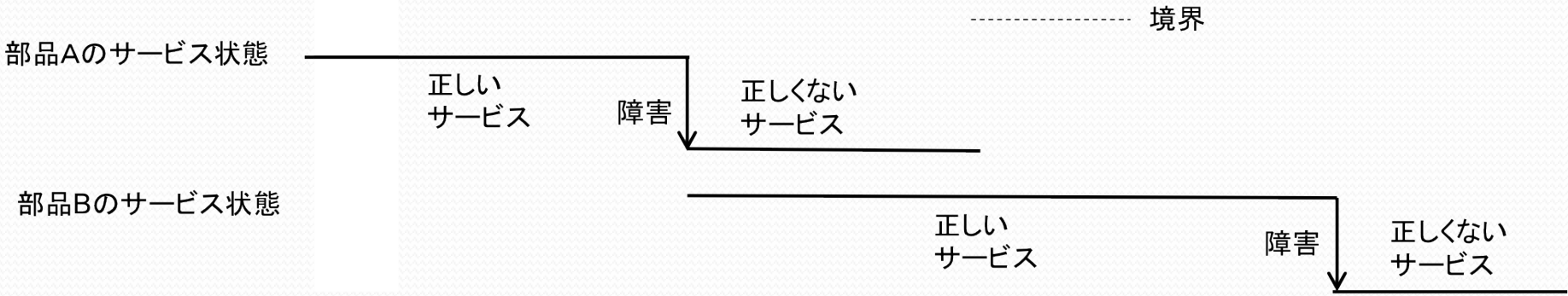
体系化の例：欠陥のクラス分け



© Dependability Science Lab. Nagoya Univ. 2012

体系化の例：誤り伝搬





© Dependability Science Lab. Nagoya Univ. 2012

ディペンダビリティの基礎概念のまとめ

- ディペンダビリティは、Fault Tolerance研究からはじまり、可用性や安全性などを統合した概念として議論されてきた
  - ディペンダビリティへの脅威、およびそれへの対処手段が議論の中心であった
- ディペンダビリティの元々の由来は、あるシステム(利用者)が他のシステム(A)に依存することができるためのAの性質である。信頼性などよりも、より利用者視点にたった概念であるといえる

© Dependability Science Lab. Nagoya Univ. 2012

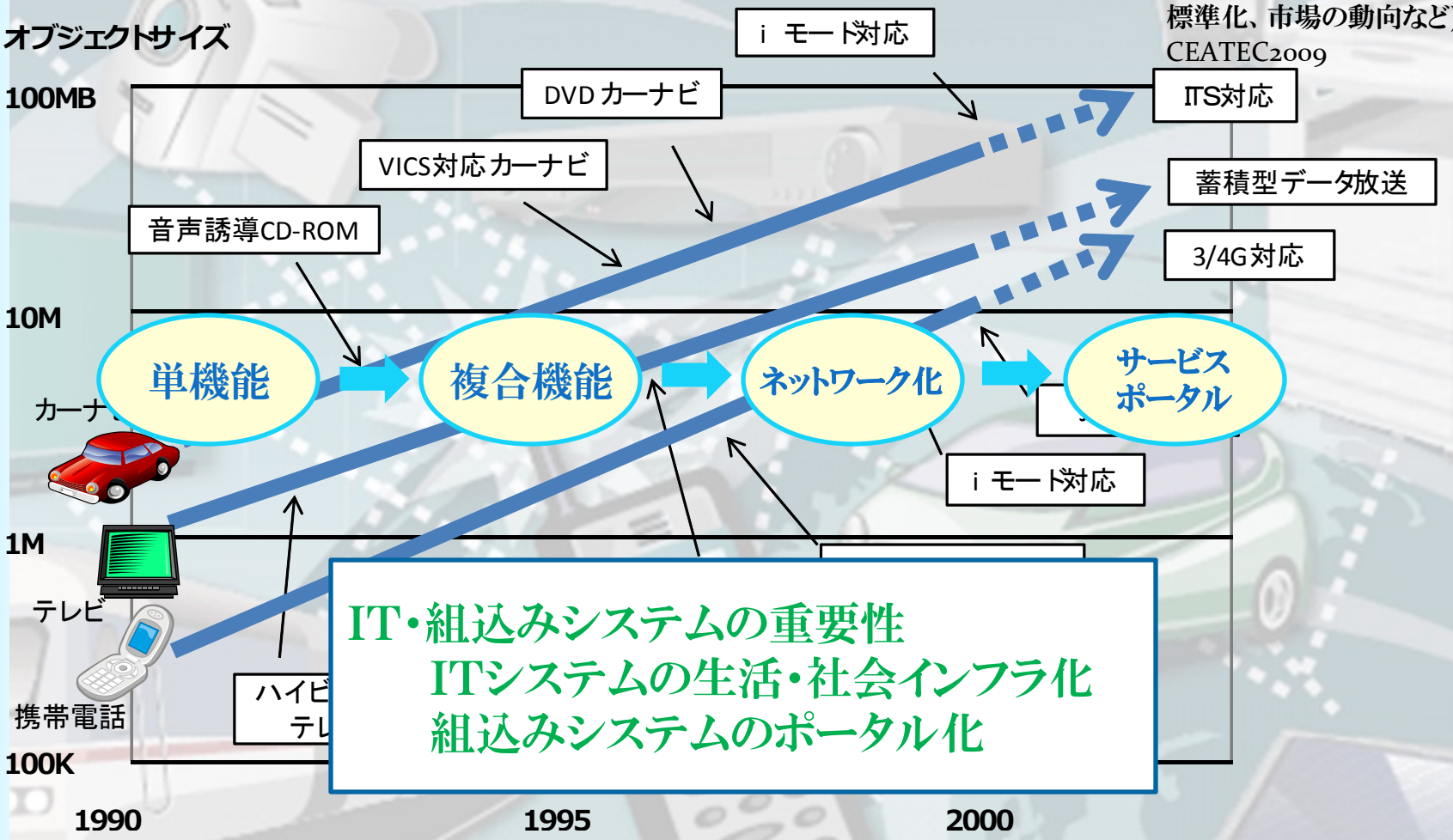
内容

- はじめに
- 基本概念
  - システムとディペンダビリティ
- オープンシステム・ディペンダビリティ
- まとめ
- 参考文献

© Dependability Science Lab. Nagoya Univ. 2012

情報システム規模の拡大とネットワーク化

参考文献2
屋代眞、ディペンダブル
システム組み込みOSに関する
最近の話題(研究開発状況や、
標準化、市場の動向など),
CEATEC2009



出展 :日経エレクトロニクス2009.9-11 (no.778)をベースに追加、修正。
経済産業省 組込みソフト産業の課題と政策展開」平成19年11月14日より

© 2009 Dependable Embedded OS R&D Center

システムの大規模化
- プログラムサイズ
- 多機能化
- ブラックボックス化したコンポーネント
- 複雑化

環境の変化
- 技術進化のスピード
- 接続システム

利害関係者の変化
- 要求の頻繁な変更
- 要求や合意に対する考え方

© Dependability Science Lab. Nagoya Univ. 2012

新たに対応の必要な分野の特徴

- 仕様/実装の不完全さ、システムの完全理解の困難さ
  - 要求、仕様、設計、実装、テスト仕様、テスト実施の不完全さ
  - 構成要素の論理的不透明さ（複雑化、巨大化、ブラックボックス化、既存のコードの利用、他）
- 使用環境の変化に伴う不確実さ、システムの挙動予測の困難さ
  - 要求事項・レベルの変化、想定外の使われ方
  - ネットワークを介しての構成要素の変化、想定外の接続
  - ネットワークを介した外部からの意図的な攻撃

⇒ オープンシステム（開放系）の問題

オープンシステムディペンダビリティへ向けて(1/2)

- 不完全、不確実なシステムがディペンダブルであるためにはどうすればよいか
  - 形式手法、テストなどの従来の手法
  - ...
- しかしながら、不完全、不確実なシステムは、障害から逃れることはできない
  - 深刻な障害が以前よりも起こりやすくなっている

© Dependability Science Lab. Nagoya Univ. 2012

オープンシステム ディペンダビリティへ向けて(2/2)

- システム保証 (System Assurance)
  - システムがどれぐらいディペンダブルか、ディペンダブルでないか、リスク分析などをもとに、利用者などのステークホルダに説明し、納得してもらう
    - 絶対に安全である、ディペンダブルであることは不可能であることを納得してもらう
- 説明責任達成 (Accountability Achievement)
  - システムに障害が発生した場合、原因を迅速に見つけ、2度と起きないよう、システムを改良することをステークホルダに説明する

© Dependability Science Lab. Nagoya Univ. 2012

保証、説明責任の必要性の高まり

- トヨタ Priusリコール問題
  - 原因説明に対する準備ができていなかった
    - 参考文献3 Michael A. Cusumano, Technology Strategy and Management – Reflections on the Toyota Debacle, Communication of the ACM, vol.54, 2011
- ISO 26262
  - “最大のインパクトは、安全性の根拠を、より説明しやすくするよう自動車メーカーに迫る点である。”
    - 参考文献4 日経エレクトロニクス 2011.1.10

© Dependability Science Lab. Nagoya Univ. 2012

オープンシステムズ・ディペンダビリティ

- *“Boundaries, functions, and structure of modern software systems change over time. Hence, it is appropriate to view such a system as an Open System. Open Systems Dependability is a property of a system such that it has the ability to continuously remove problem factors which may cause failures, to take a quick and appropriate action when a failure occurs, to minimize damage, to safely and continuously provide the services expected by users as much as possible, and to maintain accountability for the system operations and processes.”* ***参考文献3***
- cf. **これまでの定義**
  - **原義**: the ability to deliver service that can justifiably be trusted
  - **参考文献1の定義**: the ability to avoid service failures that are more frequent and more severe than is acceptable

© Dependability Science Lab. Nagoya Univ. 2012

まとめ

- ディペンダビリティはFault Toleranceの研究より始まり、これまで、安全性や信頼性を統合した概念として議論されてきた
- 近年のシステムは複合化、ネットワーク化、サービスポータル化などにより、より不完全・不確実なシステム=オープンシステムになってきている
- オープンシステムのディペンダビリティ達成のためには、従来のディペンダビリティに加え、システム保証と説明責任達成が重要になる

© Dependability Science Lab. Nagoya Univ. 2012

参考文献

1. Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transaction on Dependable and Secure Computing, Vol. 1, No. 1, 2004 (邦訳版はhttp://ocvs.cfv.jp/tr-data/PS2009-008.pdf)
2. 屋代眞、ディペンダブルシステム組み込みOSに関する最近の話題(研究開発状況や、標準化、市場の動向など), CEATEC 2009
3. Mario Tokoro ed., Open Systems Dependability-Dependability Engineering for Ever Changing Systems, CRC Press, In Press, 2012

© Dependability Science Lab. Nagoya Univ. 2012

名古屋大学

D-Case 入門
2 要求工学概論

© Dependability Science Lab. Nagoya Univ. 2012

内容

- 要求工学とは
- IEEE Std830-1998
- ISO/IEC/IEEE 29148
- 組込みシステム参照モデル

© Dependability Science Lab. Nagoya Univ. 2012

名古屋大学

2.1 要求工学とは

- 要求工学の動向
- 要求工学プロセス
- 要求と仕様

© Dependability Science Lab. Nagoya Univ. 2012

要求工学の動向

年代	要求工学の特徴	主要な成果
1970	要求分析の時代	外部環境から概念を獲得する標準的なモデル
1980	要求仕様の時代	曖昧性，矛盾，冗長性を解消する標準仕様のテンプレート
1990	要求プロセスの時代	要求プロセスの定義とプロセス改善手法
2000	高水準要求の時代	ゴール記述言語とプロセス
2010	要求整合化の時代	**顧客価値創造**のための要求工学研究と実証評価

参考） Davis, A., Keynote speech, RE2010

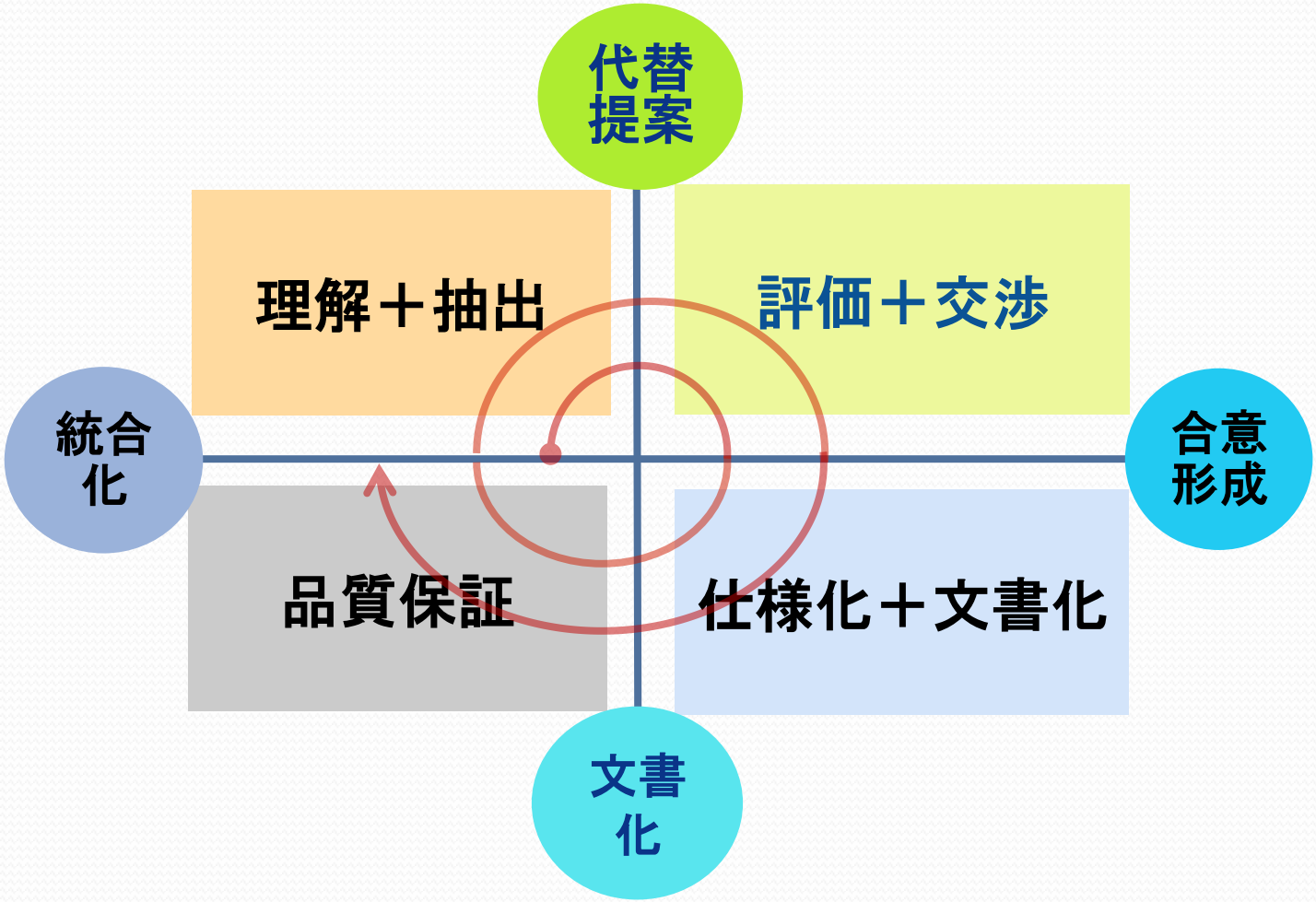
© Dependability Science Lab. Nagoya Univ. 2012

要求工学国際会議のキーワード

開催年	キーワード
2008	持続可能な世界に向けた要求
2009	適用領域とソフトウェアの懸け橋としての要求
2010	多面的世界の中の要求
2011	変化する要求
2012	要求の不確実性の扱い，優先順位，実行時環境

© Dependability Science Lab. Nagoya Univ. 2012

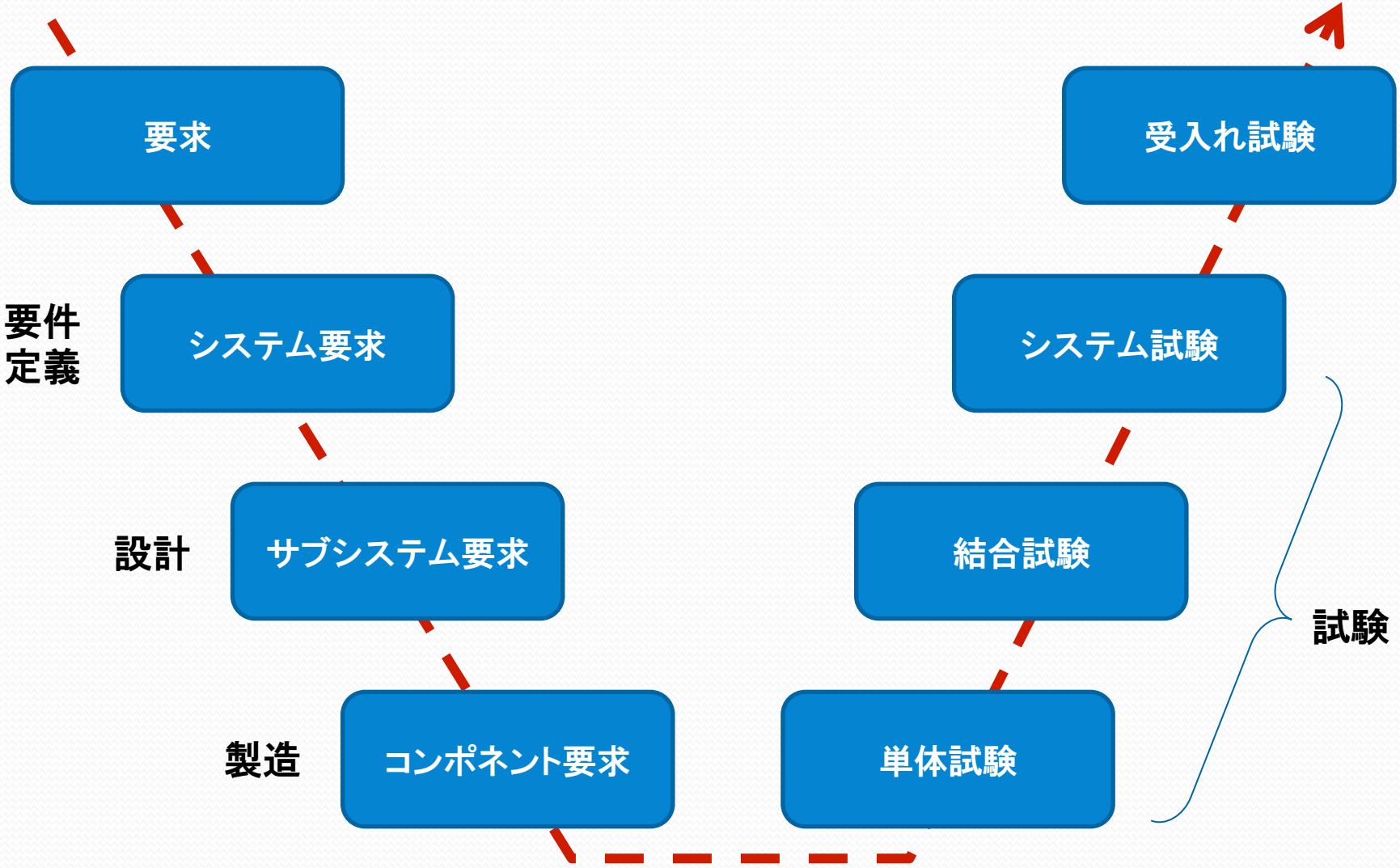
要求工学プロセス



参考）Lamsweerde, A., Requirements Engineering– From System Goals to UML Models to Software Specifications, John Wiley & Sons Inc., 2009

© Dependability Science Lab. Nagoya Univ. 2012

ソフトウェア工学と要求工学



© Dependability Science Lab. Nagoya Univ. 2012

要求と仕様

	説明
要求	システムによって解決すべき問題を定義
仕様	システムが実現すべき問題に対する解決策を定義

参考) John Knight, Fundamentals of Dependable Computing for Software Engineers, CRC Press, 2012

© Dependability Science Lab. Nagoya Univ. 2012

【定義】要件

システムが実現すべき事項をいい、機能要件と非機能要件から成る。

（情報システムの信頼性向上に関するガイドライン(案)，経済産業省，平成18年4月4日）

機能要件	利用者の要求を満足するためにソフトウェアが実現しなければならない機能
非機能要件	機能要件以外のすべての要素(性能、容量、情報セキュリティ、拡張性等)

© Dependability Science Lab. Nagoya Univ. 2012

ディペンダビリティとセキュリティ

属性	可用性(Availability) 信頼性(Reliability) 安全性(Safety) 機密性(Confidentiality) 一貫性(Integrity) 保守性(Maintainability)
脅威	欠陥(Faults) 誤り(Errors) 障害(Failures)
手段	欠陥回避(Fault Prevention) 欠陥耐性(Fault Tolerance) 欠陥除去(Fault Removal) 欠陥予測(Fault Forecasting)

AVIZZIENIS ET AL.: BASIC CONCEPTS AND TAXONOMY OF DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 1, NO. 1, JANUARY-MARCH 2004

Copyright Prof. Dr. Shuichiro Yamamoto 2012

要求マネジメント条件

条件	説明
1	変化する要求に対してサービス継続性を保証できること
2	変化する要求に対して逐次、合意形成できること
3	変化する要求に対して常に説明責任を遂行できること

名古屋大学

2.2 IEEE Std830-1998

© Dependability Science Lab. Nagoya Univ. 2012

IEEE Std830-1998

	目次	重要. 一般的構成
1	はじめに	1-1 目的 1-2 範囲 1-3 定義、頭文字、省略形 1-4 参考文献 1-5 概要
2	概要説明	2-1 製品の背景 2-2 製品機能 2-3ユーザ特性 2-4 制約 2-5 仮定および依存性 2-6 延期要求
3	具体的要求	3-1 外部インタフェース 3-2 機能 3-3 性能要求 3-4 論理データベース要求 3-5 設計の制約 3-6 ソフトウェア・システムの属性 3-7 要求仕様の構成 3-8 追加コメント
	付録	支援情報. 付加的
	索引	重要. 一般的構成

製品の背景	◆ インタフェース ◆ 環境条件
製品機能	◆ソフトウェアが実行する主要機能全体の構成
ユーザ特性	◆製品が対象とするユーザの一般的特性として 教育レベル、 経験、 専門知識などを記述
制約	◆開発者の選択肢を制約する要求項目を記述
仮定および依存性	◆要求項目に影響を及ぼす要因を列挙
延期要求	◆次期開発に延期される可能性のある要求を明記

(参考)要求仕様
http://www.bcm.co.jp/site/2004/2004Dec/04-youkyuu-kougaku-12/04-youkyuu-kougaku-12.htm

© Dependability Science Lab. Nagoya Univ. 2012

IEEE Std830-1998(続)

インタフェース

システム	・システム要求を実現するためのソフトウェア機能およびシステムに適合するインタフェースを記述
ユーザ	・ソフトウェア製品とユーザ間の論理インタフェース ・ユーザインタフェースを最適化する手段の選択肢
ハード	・ソフトウェア製品とハードウェア構成要素とのインタフェースの論理特性を記述
ソフト	・使用するソフトウェア製品および他のアプリケーションとのインタフェースを記述
通信	・LANプロトコルなど、各種の通信プロトコルを記述

環境条件

メモリ	・主記憶装置および補助記憶装置についての制約を記述
オペレーション	・正常系と異常系のオペレーションを記述
サイト適応条件	・設置サイトの状況にソフトウェアを適応させるために必要となるデータや初期化に関する要求を記述

© Dependability Science Lab. Nagoya Univ. 2012

IEEE Std830-1998と話題沸騰ポット

	目次	重要. 一般的構成
1	はじめに	1-1 目的 1-2 範囲 1-3 定義、頭文字、省略形 1-4 参考文献 1-5 概要
2	概要説明	2-1 製品の背景 2-2 製品機能 2-3ユーザ特性 2-4 制約 2-5 仮定および依存性 2-6 延期要求
3	具体的要求	3-1 外部インタフェース 3-2 機能 3-3 性能要求 3-4 論理データベース要求 3-5 設計の制約 3-6 ソフトウェア・システムの属性 3-7 要求仕様の構成 3-8 追加コメント
	付録	支援情報. 付加的
	索引	重要. 一般的構成

1. ハードウェア構成とハードウェア要求仕様
   1. 外観と内部構成
   2. 操作パネル部
2. 操作要求仕様
   1. コンセント
   2. 蓋
   3. 沸騰ボタン
   4. 保温設定ボタン
   5. 解除ボタン
   6. 給湯ボタン
   7. タイマボタン
   8. 水位メータ
3. 温度制御行為
   1. 沸騰行為
   2. 保温行為
   3. 温度制御行為をしない(アイドル)
4. 温度制御方式
5. エラー検知
6. 状態遷移
   1. 話題沸騰ポット状態遷移図
   2. 話題沸騰ポット状態遷移表
7. イベント

参考) 話題沸騰ポット(GOMA-1015型) 要求仕様書 第7版 -
www.sessame.jp/workinggroup/.../POT_Specification_v7.PDF

© Dependability Science Lab. Nagoya Univ. 2012

要求仕様が満たすべき性質

正当性	ソフトウェアが持つべきすべての要求がSRSにふくまれており、それ以外の要求をSRSがふくまないこと
無あいまい性	SRSのすべての要求の意味が一意に識別されること
完全性	SRSが、次をすべて含んでいること ・すべての必要な要求 ・すべての入力データと状況に関する応答の定義 ・用語および図表の説明
一貫性	SRSに含まれる要求間で矛盾がないこと
順位付け	要求が重要性や安定性に関して順位付けられていること
検証容易性	SRSに含まれるすべての要求に対して有限のコストで評価可能な手続きが存在して検証できること
修正容易性	要求の変更に対して、容易かつ完全で一貫性を保ってSRSを修正できるような構造を持つこと
追跡性	要求の根拠が明確で、開発工程全体で参照できること

© Dependability Science Lab. Nagoya Univ. 2012

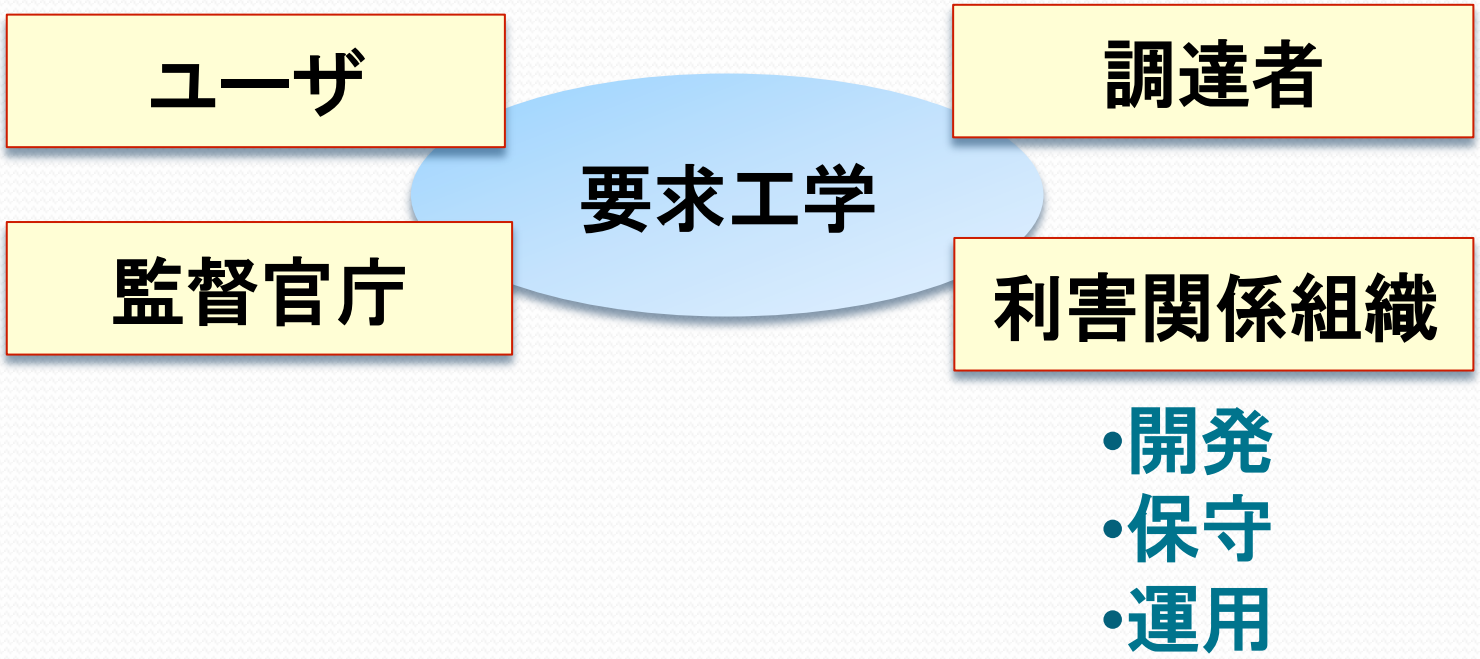
2.3 ISO/IEC/IEEE 29148

- ステークホルダ
- 要求範囲
- ニーズから要求への変換プロセス
- ステークホルダ要求仕様
- システム要求仕様
- ソフトウェア要求仕様

© Dependability Science Lab. Nagoya Univ. 2012

ステークホルダ

- ステークホルダの対象はプロジェクトごとに変化



参考)5.2.2, p.8, ISO/IEC/IEEE 29148:2011

© Dependability Science Lab. Nagoya Univ. 2012

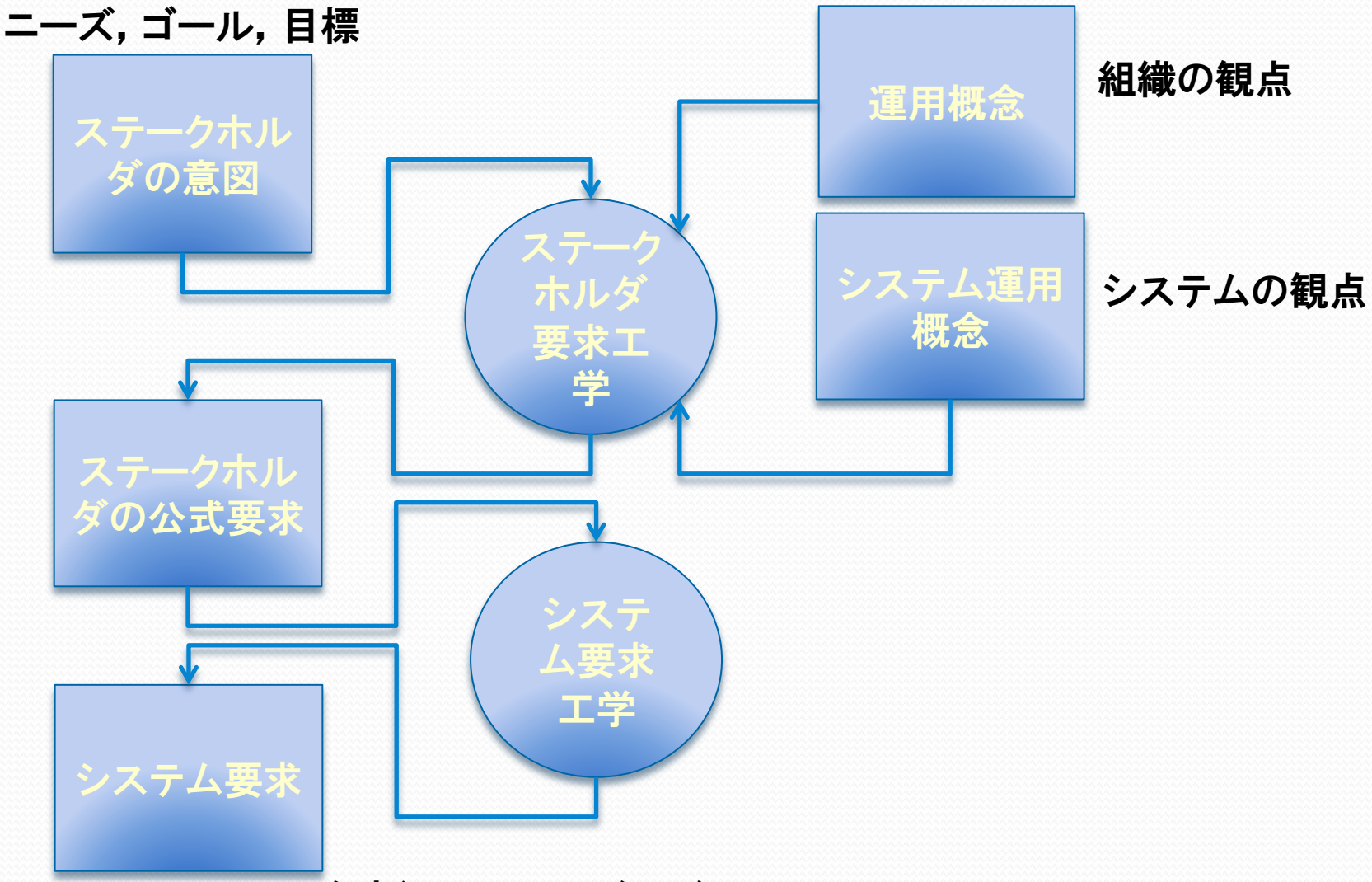
ビジネスコンテクストにおける要求の範囲

環境	内容	要求	対象
外部		ー	ー
組織	市場動向，法制度，法令遵守，社会的責任，技術蓄積，労働能力，競合製品，標準化動向，公共制度，自然環境	ステークホルダ要求	ビジネス運用
ビジネス運用	ビジネス運用プロセス，制約，ポリシー，規則，モード，品質，ビジネス構造	ステークホルダ要求	システム運用
システム運用		システム要求	システム
システム		ソフトウェア要求	ソフトウェア

参考）5.4 Requirement information items

© Dependability Science Lab. Nagoya Univ. 2012

ニーズから要求への変換



参考） 5.2.3, ISO/IEC/IEEE 29148:2011

© Dependability Science Lab. Nagoya Univ. 2012

ステークホルダ要求(StRS)の構成例

章	節
1. はじめに	1.1 ビジネスの目的 1.2 ビジネスの範囲 1.3 ビジネスの概要 1.4 定義 1.5 ステークホルダ
2. 参考文献	
3. ビジネスマネジメント要求	3.1 ビジネス環境 3.2 ゴールと目標 3.3 ビジネスモデル 3.4 情報環境
4. ビジネス運用要求	4.1 ビジネスプロセス 4.2ビジネス運用ポリシーと規則 4.3 ビジネス運用制約 4.4 ビジネス運用モード 4.5 ビジネス 運用品質 4.6 ビジネス構造
5. ユーザ要求	
6. 提案システムの概念	6.1 運用概念 6.2 運用シナリオ
7 プロジェクトの制約	
8. 付録	8.1 Acronyms and abbreviations

© Dependability Science Lab. Nagoya Univ. 2012

システム要求SyRSの構成例

章	節
1. はじめに	1.1 システムの目的 1.2 システムの範囲 1.3 システム概要 1.3.1 システムのコンテクスト 1.3.2 システム機能 1.3.3 ユーザ特性 1.4 定義
2. 参考文献	
3. システム要求	3.1 機能要求 3.2 操作性要求 3.3 性能要求 3.4 システムインタフェース 3.5 システム運用 3.6 システムモードと状態 3.7 物理特性 3.8 環境条件 3.9 システムセキュリティ 3.10 情報管理 3.11 ポリシーと法制度 3.12 システムライフサイクル継続(sustainment) 3.13パッケージ化, ハンドリング, 出荷, 配送
4. 検証	
5. 付録	前提条件と依存関係 Acronyms and abbreviations

© Dependability Science Lab. Nagoya Univ. 2012

ソフトウェア要求仕様SRS構成例

章	節
1. はじめに	1.1 目的 1.2 範囲 1.3 製品概要 1.3.1 製品展望 1.3.2 製品機能 1.3.3 ユーザ特性 1.3.4 限界 1.4 定義
2. 参考文献	
3. 具体要求	3.1 外部インタフェース 3.2 機能 3.3 操作性要求 3.4 性能要求 3.5 論理データベース要求 3.6 設計制約 3.7 ソフトウェアシステム属性 3.8 支援情報
4. 検証	
5. 付録	5.1 前提条件と依存関係 5.2 Acronyms and abbreviations

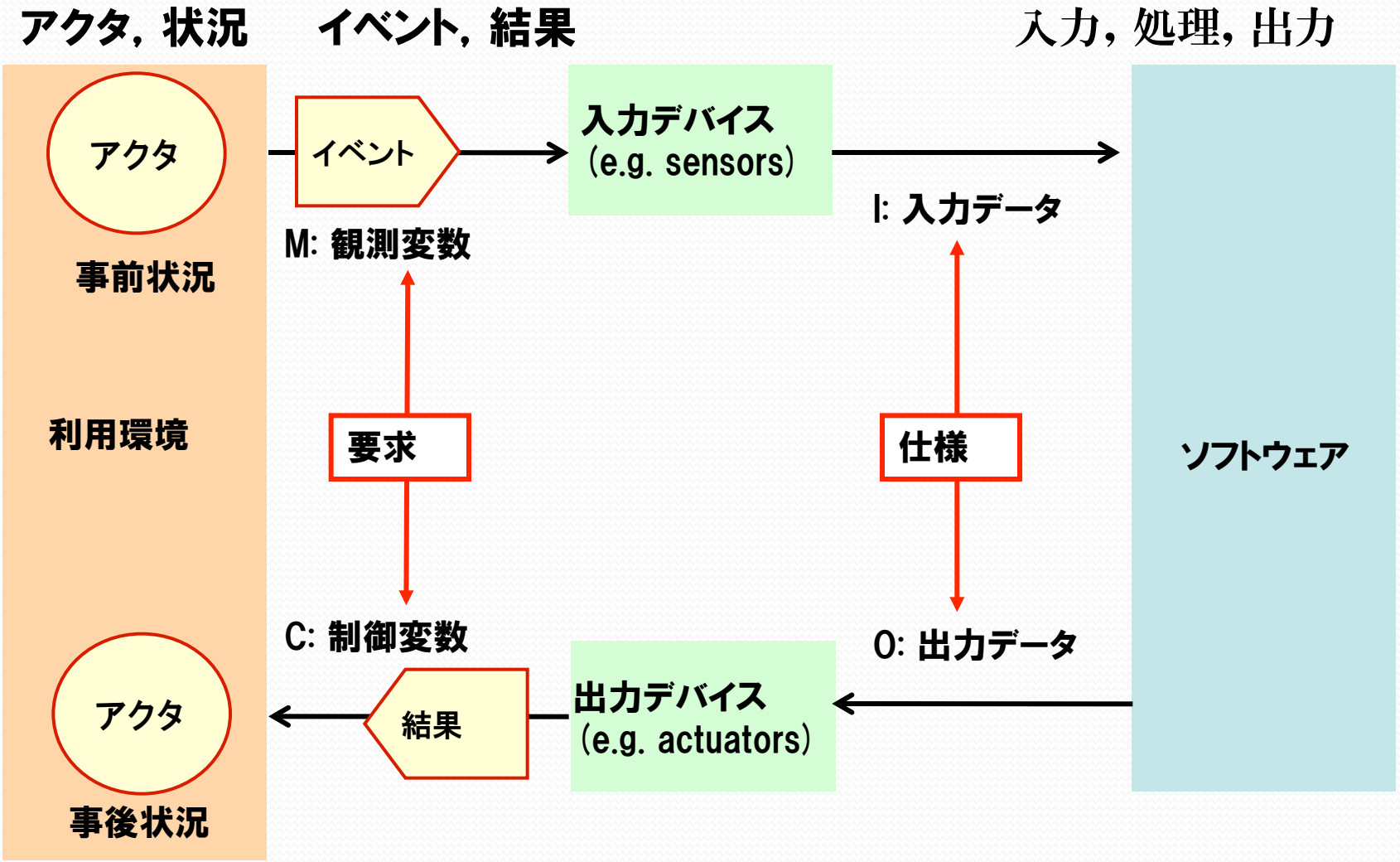
© Dependability Science Lab. Nagoya Univ. 2012

2.4　組込みシステム参照モデル

- Parnas変数モデル
- アクタ関係モデル
- 組込みシステムの論理参照モデル
- 組込みシステムの要求プロセス

© Dependability Science Lab. Nagoya Univ. 2012

Parnas変数モデル

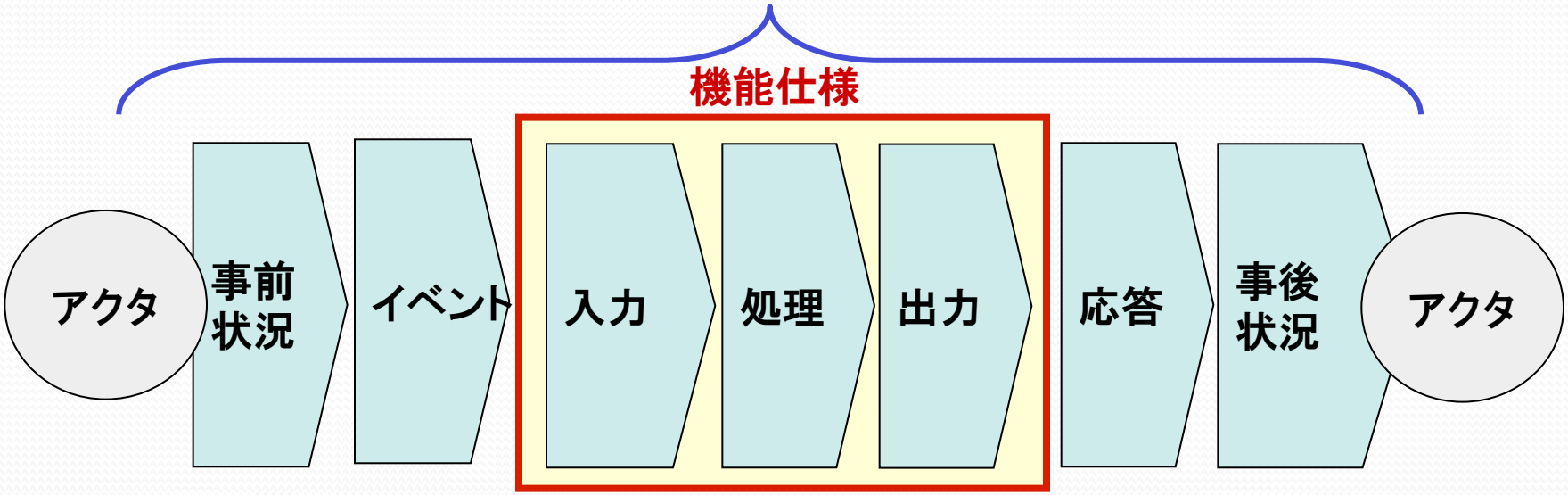


(参考)Parnas, D. & Madey, J., Functional documents for computer systems, 1995

© Dependability Science Lab. Nagoya Univ. 2012

アクタ関係モデル

曖昧性：範囲，内容，関係の多義性・不明性

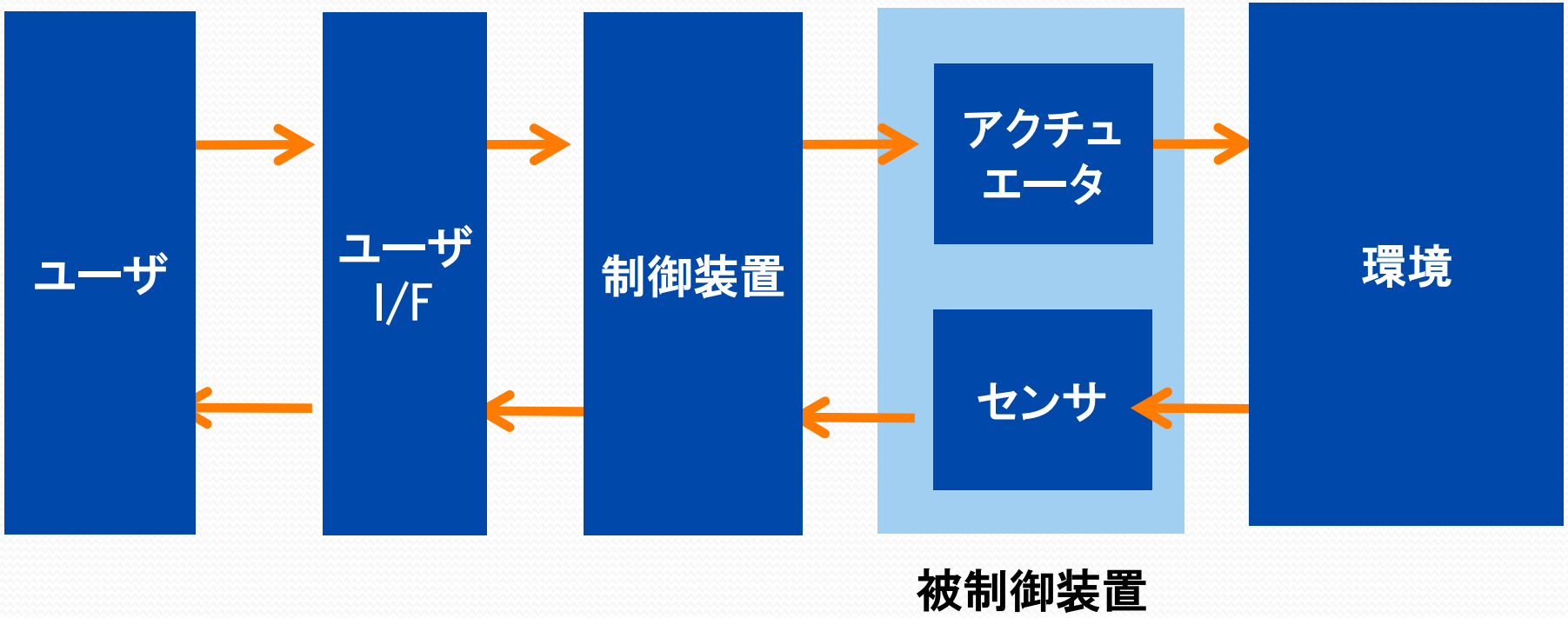


参考)要求の曖昧さ，http://www.bcm.co.jp/site/youkyu/youkyu38.html

© Dependability Science Lab. Nagoya Univ. 2012

組込みシステムの論理参照モデル

- システム全体の振舞い
- 相互作用
- コンポーネントの振舞い



© Dependability Science Lab. Nagoya Univ. 2012

組込みシステムの要求プロセス

プロセス	説明
ドメイン分析	組込みシステムの振舞いモデル
要求抽出	機能要求：シナリオ,ユースケース，正常系，異常系 非機能要求：制約，アスペクト，品質特性
要求確認	要求レビュ，構造化ウォークスルー 一貫性検査，重要安全性の検証 プロトタイピング，シミュレーション
要求追跡	要求が設計,実装で実現されていることを段階的に確認 品質保証で確認
要求検証	形式的証明 テスト，構造化ウォークスルー，コードレビュ

© Dependability Science Lab. Nagoya Univ. 2012

自動車の要求モデルの構成要素例

複数のアクタ
　運転手，乗客，保守者など自動車に係る人
組込みソフトウェア機能の多重性と相互作用
　ユースケース
　振舞い仕様
予期しない相互干渉がないこと
異なる記述レベルの要求仕様の管理
多様な類似仕様
非機能要求：経費,出荷時期，安全性,信頼性,保守性
制約：プラットフォームの再利用

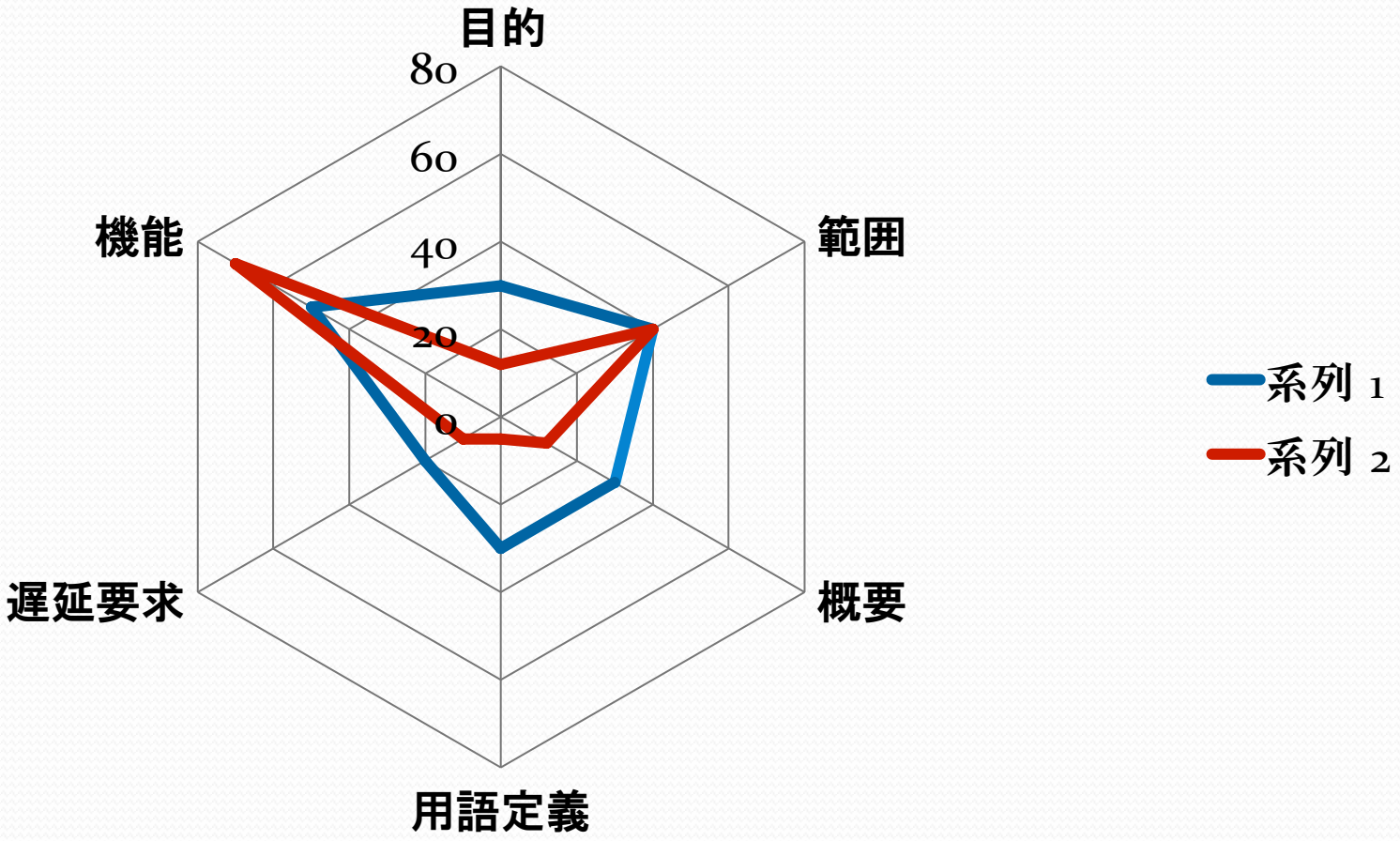
© Dependability Science Lab. Nagoya Univ. 2012

まとめ

- **要求工学とは**
- IEEE Std830-1998
- ISO/IEC/IEEE 29148
- **組込みシステム参照モデル**

© Dependability Science Lab. Nagoya Univ. 2012

クイズ
経費・納期超過プロジェクトはどちらか？



© Dependability Science Lab. Nagoya Univ. 2012

NAIST

ディペンダビリティ標準規格

© Nara Advanced Institute of Science and Technology 2012

目次

- 準備
- ディペンダビリティ
- 関連する属性に依存しない規格
  - ソフトウェア品質一般に関する規格
  - システムアシュアランス
    - アシュアランスケース
  - リスク
  - 認証に関する規格
- 属性別規格
  - 安全性
  - セキュリティ
  - サービス継続
- ディペンダビリティからみた属性別規格

© Nara Advanced Institute of Science and Technology 2012

国際標準規格について

- 国際規格
  - ISO, IEC
- 国内規格
  - JIS, ANSI, EN
    - ただし、実質国際的に利用されているものも多い
- 国際業界規格
  - フォーラム規格、
  - OMG, The Open Group, FlexRay
- 国内業界規格
  - IEEE
    - ただし、実質国際規格として利用されているものも多い
- デファクトスタンダード
  - TCP/IP, Windows, etc.
- BoK (Body of knowledge)
  - SWEBOK (software engineering), PMBOK(project management)

© Nara Advanced Institute of Science and Technology 2012

準備:マネジメントシステム

- 組織が方針及び目標を定め、その目標を達成するためのシステム
  - ITシステムだけでなく、組織体系なども含む
- マネジメントシステム規格の例
  - ISO9000シリーズ
    - 品質マネジメントシステム規格
  - ISO14000シリーズ
    - 環境マネジメントシステム規格

© Nara Advanced Institute of Science and Technology 2012

準備：システムライフサイクル規格

- ISO/IEC15288
  - Systems and software engineering — System life cycle processes
- ISO/IEC 12207
  - Systems and software engineering — Software life cycle processes
- **システムやソフトウェア開発プロセスの手順は限定せず、各プロセスについてその標準的な名前と以下の情報による特徴付けを与える**
  - **必要な入力**
  - **アクティビティ**
  - **アウトカム**

© Nara Advanced Institute of Science and Technology 2012

ディペンダビリティ規格概要

- ディペンダビリティ概念
  - 以下の文献がしばしば引用される
    - [Avizienisら2004] Basic Concepts and Taxonomy of Dependable and Secure Computing
- IEC60300 シリーズ
  - ディペンダビリティマネジメントシステムの規格
  - IEC TC56：歴史的に、電子部品の信頼性を扱ってきたところ
    - 1965: Reliability and Maintainability
    - 1996: Dependability
  - 特徴
    - ディペンダビリティの定義
      - collective term used to describe the availability performance and its influencing factors reliability performance, maintainability performance and maintenance support performance
      - いくつかの属性の集合体といっている

© Nara Advanced Institute of Science and Technology 2012

品質特性

- ISO/IEC9126
  - Information technology software product evaluation: Quality characteristics and guidelines for their use
  - JIS X 0129
  - **機能性、信頼性、使用性、効率性、保守性、移植性の品質6特性とそれらの品質副特性を定義**
  - ISO/IEC25000シリーズとして改訂

© Nara Advanced Institute of Science and Technology 2012

ISO/IEC25000シリーズ

- Software engineering— Software product Quality Requirements and Evaluation (SQuaRE) — Quality models
- 三つの視点：
  - 利用時品質のモデル（Quality in use model）
  - 製品品質のモデル（Product quality model）
  - データ品質のモデル（Data quality model）

© Nara Advanced Institute of Science and Technology 2012

システムアシュアランス

- NATO: Engineering for system assurance in NATO programmes, 2010 での定義
  - *System assurance* is the justified confidence that the system functions as intended and is free of exploitable vulnerabilities, either intentionally or unintentionally designed or inserted as part of the system at any time during the life cycle.
  - 根拠をもつ信用
  - もともと、安全保障に関する情報などに対する各種対策を扱う概念としてInformation Assurance が存在
    - システムやソフトウェアに概念を拡張

© Nara Advanced Institute of Science and Technology 2012

NAIST

システムアシュアランスに関する規格

- ISO/IEC 15026
  - Systems and software assurance
  - Part1 ～Part4
    1. Concepts and vocabulary (2010年)
    2. Assurance cases (2011年)
    3. System integrity levels (2011年)
    4. Assurance in life cycle(現在準備中)
  - もともと、Integrity level の規格として1998年に発行
    - システムアシュアランスの規格として改訂中

© Nara Advanced Institute of Science and Technology 2012

NAIST

システムアシュアランスに関するその他の規格

- ISO/IEC TR 15443 : 2005
  - A framework for IT security assurance
  1. Overview and framework
  2. Assurance methods
  3. Analysis of assurance methods
  - セキュリティを中心としたアシュアランス活動に関わる多くの規格を網羅
  - システムアシュアランスを実現するための指針を目指す

© Nara Advanced Institute of Science and Technology 2012

アシュアランスケース

- ISO/IEC15026-2:2011
  - Systems and software assurance - Assurance cases
  - Claim, Argument, Evidence などの概念を規定
- OMG Argumentation meta-model
  - Argument の種類を規定
    - 例:
      - AssertedInference : Claim から導かれること
      - AssertedChallenge : Claim A は、B が偽であることを導く
    - ただし、Claim, Evidence についての規定も含む
  - Evidence も含めた統合モデルSACMを策定中
- BS 5760-18:2010
  - Reliability of systems, equipment and components. Guide to the demonstration of dependability requirements. The dependability case
  - 英国の国内規格
    - ただし現在、IEC60300などを策定するIEC TC56 にて国際規格化作業中

© Nara Advanced Institute of Science and Technology 2012

アシュアランスケース記述ガイド

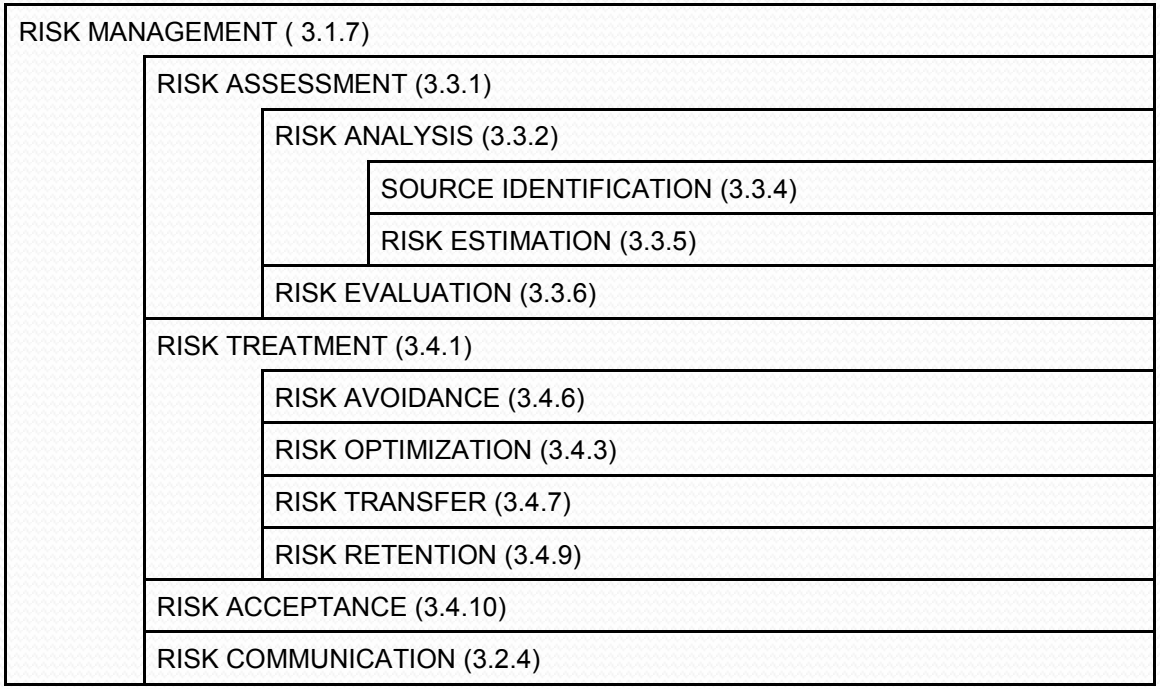
- GSN Standard version 1.0
  - アシュアランスケースの代表的な表現である GSNに関するコミュニティ標準規格
    - GSN によるアシュアランスケース記述の簡単な構成手順を含む
- European Organization for the Safety of Air Navigation, Safety Case Development Manual, 2nd ed., EUROCONTROL, Oct. 2006

© Nara Advanced Institute of Science and Technology 2012

NAIST

リスク

- ISO/IEC Guide 73 : 2009
  - Risk management — Vocabulary — Guidelines for use in standards
  - **語彙の定義**
    - Risk: effect of uncertainty on objectives
    - resilience: adaptive capacity of an organization in a complex and changing environment (2009年版から)



この図は2002年版から

© Nara Advanced Institute of Science and Technology 2012

リスクマネジメント規格

- ISO31000: 2009
  - Risk management — Principles and guidelines
  - 以下などを強調
    - decision making
    - dynamic, iterative and responsive to change
    - accountability
- ISO/IEC16085: 2006
  - Life cycle processes — Risk management
  - ISO/IEC15288,12207などのライフサイクル規格との連携を重視
    - ISO/IEC15026 もこちらをベースにしている
  - リスクマネジメントプランなどの書式例など含む(informative)

© Nara Advanced Institute of Science and Technology 2012

リスクアセスメント技術群

- ISO31010:2009
  - Risk management – Risk assessment techniques
  - **ブレーンストーミングからFTA,HAZOP, FMEAなど**
    リスクアセスメントに関わる技術群の位置づけと説明
- **個別リスクアセスメント技術**
  - IEC 61025, Fault tree analysis (FTA)
  - IEC 60812, Analysis techniques for system reliability – Procedures for failure mode and effect analysis (FMEA)
  - IEC 61882, Hazard and operability studies (HAZOP studies) – Application guide

© Nara Advanced Institute of Science and Technology 2012

NAIST

認証に関する規格

- ISO 19011
  - Guidelines for auditing management systems
  - 内部監査、第二者監査、第三者監査含むマネジメントシステムに対する監査全般を規定する規格
  - 監査精度の設計と実装、監査人の能力と評価など
- ISO 17021
  - Conformity assessment — Requirements for bodies providing audit and certification of management systems
  - 特に、第三者監査（認証）に関する規格

Internal auditing	External auditing	
	Supplier auditing	Third party auditing
Sometimes called first party audit	Sometimes called second party audit	For legal, regulatory and similar purposes For certification (see also the requirements in ISO/IEC 17021:2011)

ISO 19011 より

© Nara Advanced Institute of Science and Technology 2012

ソフトウェアの安全性に関わる規格

ISO/IEC Guide 51
Safety aspects – Guidelines for their inclusion in standards

DO178C
Software Considerations in Airborne Systems and Equipment Certification

IEC 61508
Functional safety of electrical/ electronic/programmable electronic safety-related systems

SIL (Safety Integrity Level) 含む

ISO 26262
Road vehicles — Functional safety

IEC 62278
Railway applications – Specification and demonstration of reliability, availability, maintainability and safety (RAMS)

ASIL (Automotive Safety Integrity Level) 含む

© Nara Advanced Institute of Science and Technology 2012

安全性に関する重要概念

- Safety integrity level (IEC61508等)
  → コストに応じた対応
- Safety case (ISO26262)
  → 定性的な安全性評価
- Safety culture / lifecycle (全て)
  → プロセスによる対応

- 背景
  - ソフトウェアにって実現された安全機能に対しては、従来型の安全性が適応できない
  - 定量的な基準によっては致命的な事故は防ぐことはできない

© Nara Advanced Institute of Science and Technology 2012

セキュリティおよびサービス継続に関わる規格

CC (Common Criteria 含む)

Common Criteria
for Information Technology
Security Evaluation

ISO/IEC15408
Security techniques —
Evaluation criteria for
IT security

ISO/IEC27000シリーズ
Security techniques —
Information security
management systems

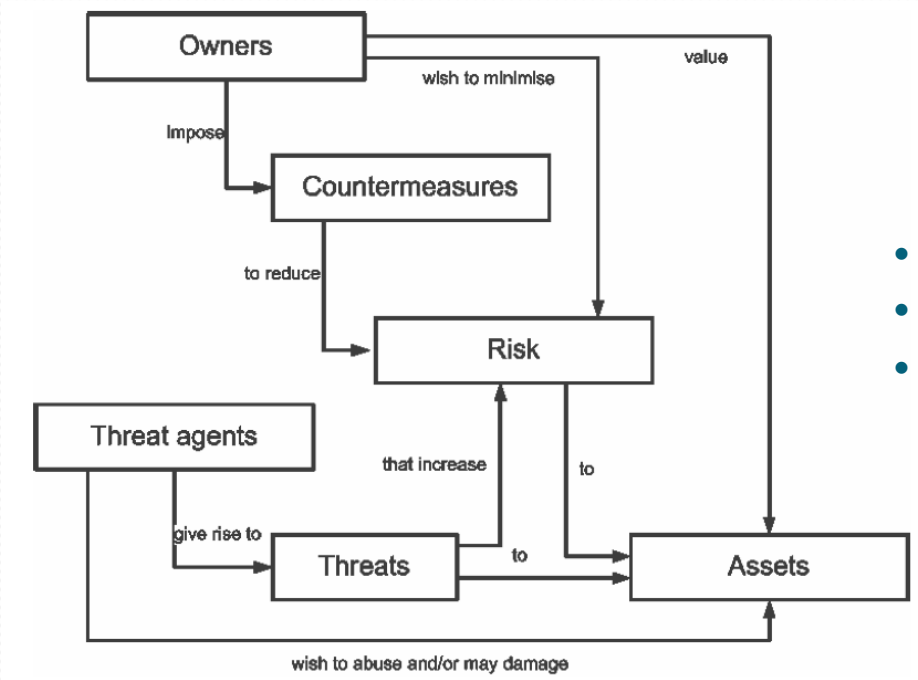
ISO/IEC20000
Information technology —
Service management

SLA (Service Level Agreement) 含む

© Nara Advanced Institute of Science and Technology 2012

セキュリティに関する重要概念

- Common Criteria
  - 具体的なセキュリティ要件を定義するのではなく製品ごとにセキュリティ要件を定義しその達成を示すための枠組みを提供



- assets(資産) : 守るべき対象
- threat(脅威) : asset を脅かす事象
- threat agent (敵対者) : threat を引き起こす存在

ISO/IEC15408-1:1998 Information technology — Security techniques — Evaluation criteria for IT security —

© Nara Advanced Institute of Science and Technology 2012

属性別規格のまとめ

	安全性	セキュリティ	サービス継続
Issues	ソフトウェアの把握困難さ、定量的評価の限界	セキュリティ問題の多様化、一律のセキュリティ要件の限界	障害発生の回避困難、発生時の利害関係者間の紛争
コストに応じた対応に関する合意基準	Safety Integrity Level (SIL)	Evaluation Assurance Level (EAL)	Service Level Agreement(SLA)
納得に基づく合意 / 説明責任の遂行	Safety case	Security Case/ Security Target / Certification Report	Service report
プロセスによる対応	Safety culture Safety lifecycle	Information security management	IT service management
関係国際規格	IEC61508, ISO26262	ISO/IEC15408, ISO/IEC27000	ISO/IEC12207/15288, ISO/IEC20000

© Nara Advanced Institute of Science and Technology 2012

ディペンダビリティからみた属性別規格

	安全性	セキュリティ	ITサービス	ディペンダビリティ
Issues	ソフトウェアの把握困難さ、定量的評価の限界	セキュリティ問題の多様化、一律のセキュリティ要件の限界	障害発生の回避困難、発生時の利害関係者間の紛争	
コストに応じた対応に関する合意基準	Safety Integrity Level	Evaluation Assurance Level (EAL)	Service Level Agreement	Integrity level
納得に基づく合意 / 説明責任の遂行	Safety case	Security Case/ Security Target / Certification Report	Service report	Assurance case
プロセスによる対応	Safety culture Safety lifecycle	Information security management	IT service management	System assurance
関係国際規格	IEC61508, ISO26262	ISO/IEC15408, ISO/IEC27000	ISO/IEC12207/15288, ISO/IEC20000	ISO/IEC15026 ISO/IEC15443 IEC TC56 NWIP

© Nara Advanced Institute of Science and Technology 2012

名古屋大学

D-Case 入門
4 システムリスク分析

© Dependability Science Lab. Nagoya Univ. 2012

内容

- システム障害事例
- 独立検証確認
- サービス継続性
- リスク分析手法

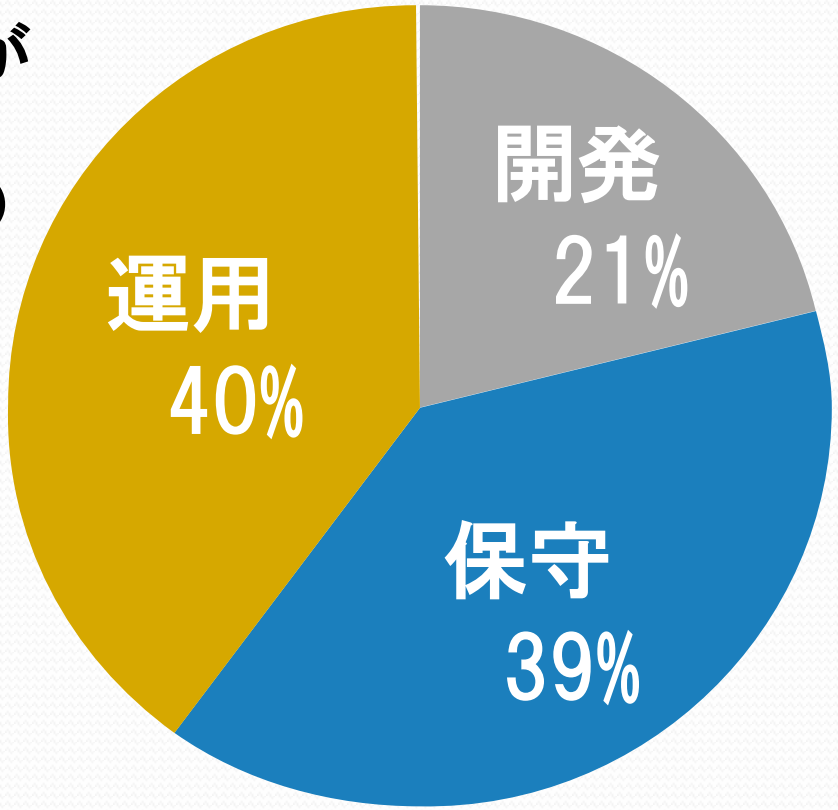
© Dependability Science Lab. Nagoya Univ. 2012

システム障害事例

© Dependability Science Lab. Nagoya Univ. 2012

重要インフラ障害の原因分類

社会的に問題とされマスコミが
取り上げた事例85件の分類
（2006年12月～2007年10月）



参考）経済産業省，重要インフラ情報システム信頼性研究会　報告書，2009, IPA

© Dependability Science Lab. Nagoya Univ. 2012

現代の技術戦略・経営の課題

分類	問題点	問題事象
生産システム	■JITシステムでは，在庫と作業コストの最小化が求められるので生産工程は可視化できるが，利用時に表面化するような設計上の欠陥を検出できない	ー
製品開発プロセス	■過度なコスト削減要求のために，設計・テスト工程で問題が発生した． ■すべての条件下で車が正しく動作することを確認することに対して，もっとよい仕事をする責任があった．	①ペダルがフロアマットに引っかかる ②プラスチックペダルが水分と摩擦で粘着性状態になる ③運転者がブレーキシステムの異常動作だと感じる
経営管理プロセス	■GMを追い越すために急速に経営資源と販路を拡大したことにより，従来と異なるプレーキペダルを採用した． ■高品質生産工程を実現していると過信したため，製品プロセスの細部にまで経営層が注意を払わなくなった． ■問題の原因究明と説明に対する準備ができていなかったため，問題を認めるまで経営層が否定し続けた．	

①運用を前提とする開発プロセスが必要

②ディペンダビリティ確認が必要

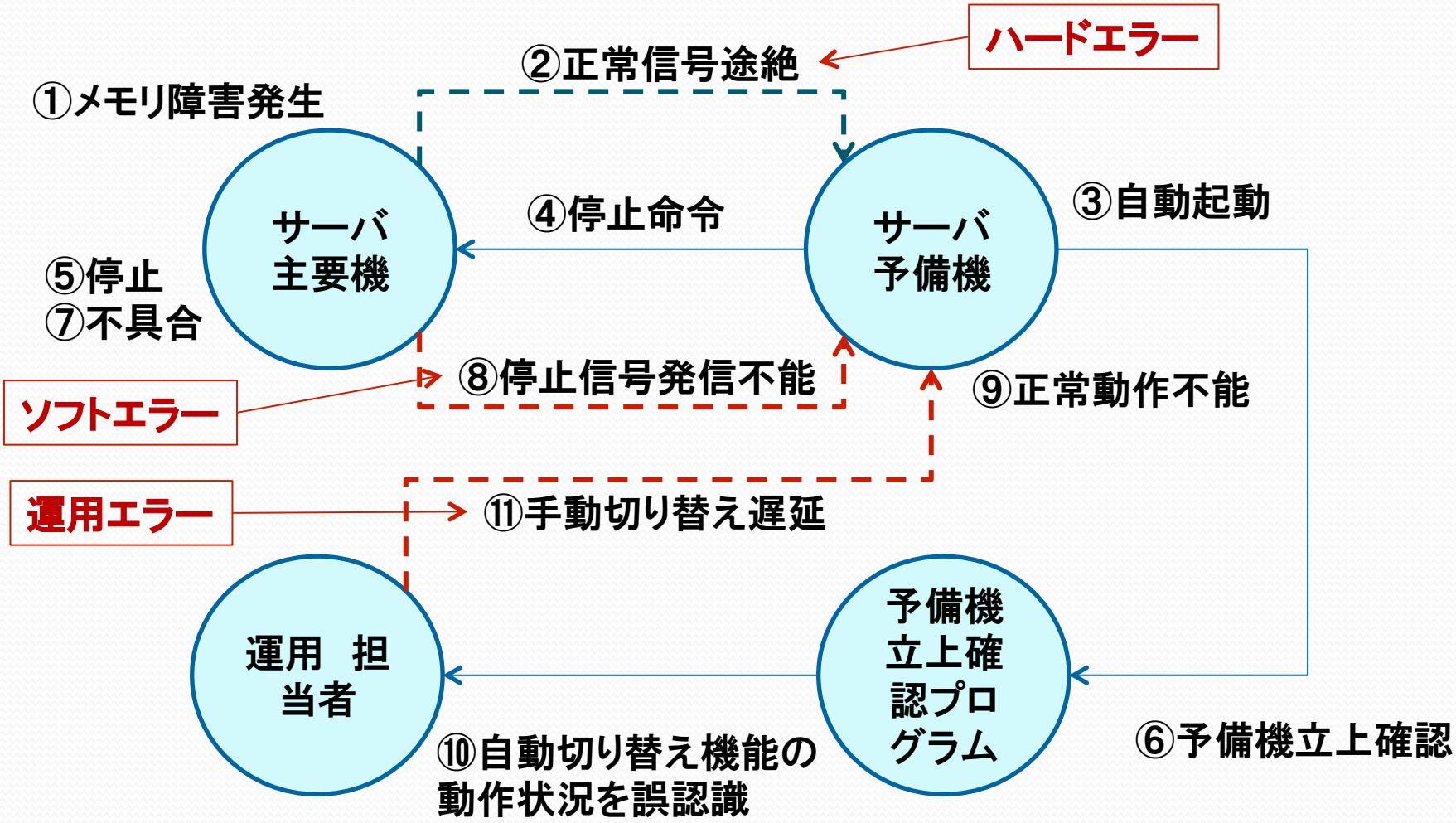
③説明責任遂行プロセスが必要

④ステークホルダ合意形成が必要

参考) Michael A. Cusumano, Technology Strategy and Management – Reflections on the Toyota Debacle, CACM, vol.54, no.1, pp.33-35, 2011

© Dependability Science Lab. Nagoya Univ. 2012

金融系システム障害、故障信号に不具合



参考）日経新聞電子版，2012/2/10 2:00

© Dependability Science Lab. Nagoya Univ. 2012

Fサーバのパンフレット

- 「Fサーバは、これまで以上に安心してレンタルサーバーサービスをご利用いただくために品質保証制度（SLA）を設け、稼働率100%を保証した高いビジネス品質のサービスでお客様のビジネスを支えます」
- 「さらに人為的なデータ損失があった場合にそなえ、日に1回、外部サーバーにデータを保存していますので、お客様によるデータの誤消去があった場合にも、前日の状態に戻すことが可能です」

- 「数年前までは記述の通りだったが、現在は説明の通りと異なる。記述ミスで、修正をせずにいた可能性がある」

© Dependability Science Lab. Nagoya Univ. 2012

独立検証確認

© Dependability Science Lab. Nagoya Univ. 2012

ISO26000　企業の社会的責任

7つの原則	説明責任
	透明性
	倫理的行動
	利害関係者の尊重
	法令順守
	国際行動規範の尊重
	人権尊重

© Dependability Science Lab. Nagoya Univ. 2012

検証確認活動を保証するには？

- 客観的な証跡がない限り，システムの信頼性をいくら当事者が主張しても信用されない

- 当事者による客観的な検証確認結果を示す証跡
- 当事者以外の第三者による客観的な検証確認(IV&V)の証跡

© Dependability Science Lab. Nagoya Univ. 2012

ソフトウェアV&Vとは

活動	説明	判断内容	証跡の内容
検証	開発活動による生産物が開発活動に対する要求に適合していること	ソフトウェアとコンポーネントが，その工程の開始時点で定められた条件を満たすこと	①各活動に対する要求に適合すること ②標準，実践規約，慣例を満たすこと ③各活動を完了すること ④後続工程の開始基準を満たすこと
確認	開発されたソフトウェアが意図された利用法とユーザニーズに適合していること	ソフトウェアとコンポーネントが開発プロセスの完了時までにユーザのニーズとしての要求を満たすこと	①ソフトウェア要求を満たすこと ②正しい問題を解決すること

参考）IEEE Std. 1012-2004 Software Verification and Validation

© Dependability Science Lab. Nagoya Univ. 2012

ソフトウェア完全性水準

完全性水準	無視可能	軽微	重大	破局
4			確実，相当	≧偶発
3		確実，相当	相当，偶発	偶発，まれ
2	確実，相当	相当，偶発	まれ	
1	相当≧	偶発，まれ	まれ	

発生確率

影響度

参考）IEEE Std. 1012-2004 Software Verification and Validation

© Dependability Science Lab. Nagoya Univ. 2012

IEEE Std. 1012-2004の要求V&V

①追跡性分析
②要求評価
③インタフェース分析
④重要性分析
⑤システムテスト計画作成
⑥受入れテスト計画作成
⑦構成管理評価
⑧危険性分析
⑨セキュリティ分析
⑩リスク分析

参考）IEEE Std. 1012-2004 Software Verification and Validation

© Dependability Science Lab. Nagoya Univ. 2012

テストプロセス

システムとその関連生産物に対して、**具体的な証跡**を提供

a) 要求を満足すること

b) 適切な問題を解決すること

- 物理規則を正しくモデル化
- ビジネス規則を実現
- システムの前提条件を利用

c) 意図された利用とユーザニーズを満足すること

参考) IEEE Std. 829-2008 Standard for Software and System Test Documentation

© Dependability Science Lab. Nagoya Univ. 2012

サービス継続性

© Dependability Science Lab. Nagoya Univ. 2012

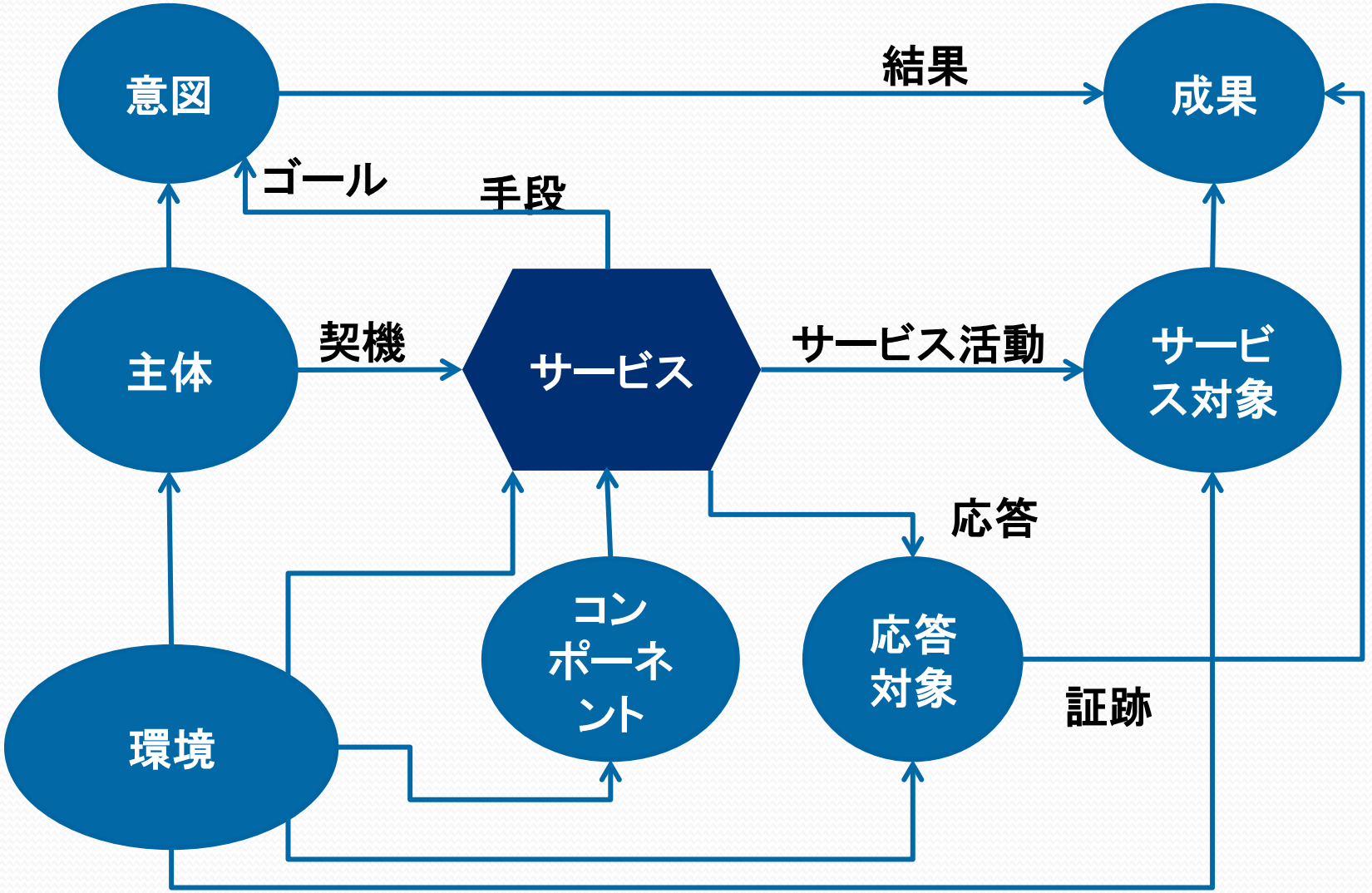
ディペンダビリティとセキュリティ属性

属性	説明
可用性	正しいサービスの提供用意性
信頼性	正しいサービスの持続性
安全性	破滅的な結果をユーザと環境にもたらすことがないこと
一貫性	不正なシステム変更がないこと
機密性	許可のない情報開示がないこと
保守性	修正と修繕を実施する能力

AVIZZIENIS ET AL.: BASIC CONCEPTS AND TAXONOMY OF DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 1, NO. 1, JANUARY-MARCH 2004

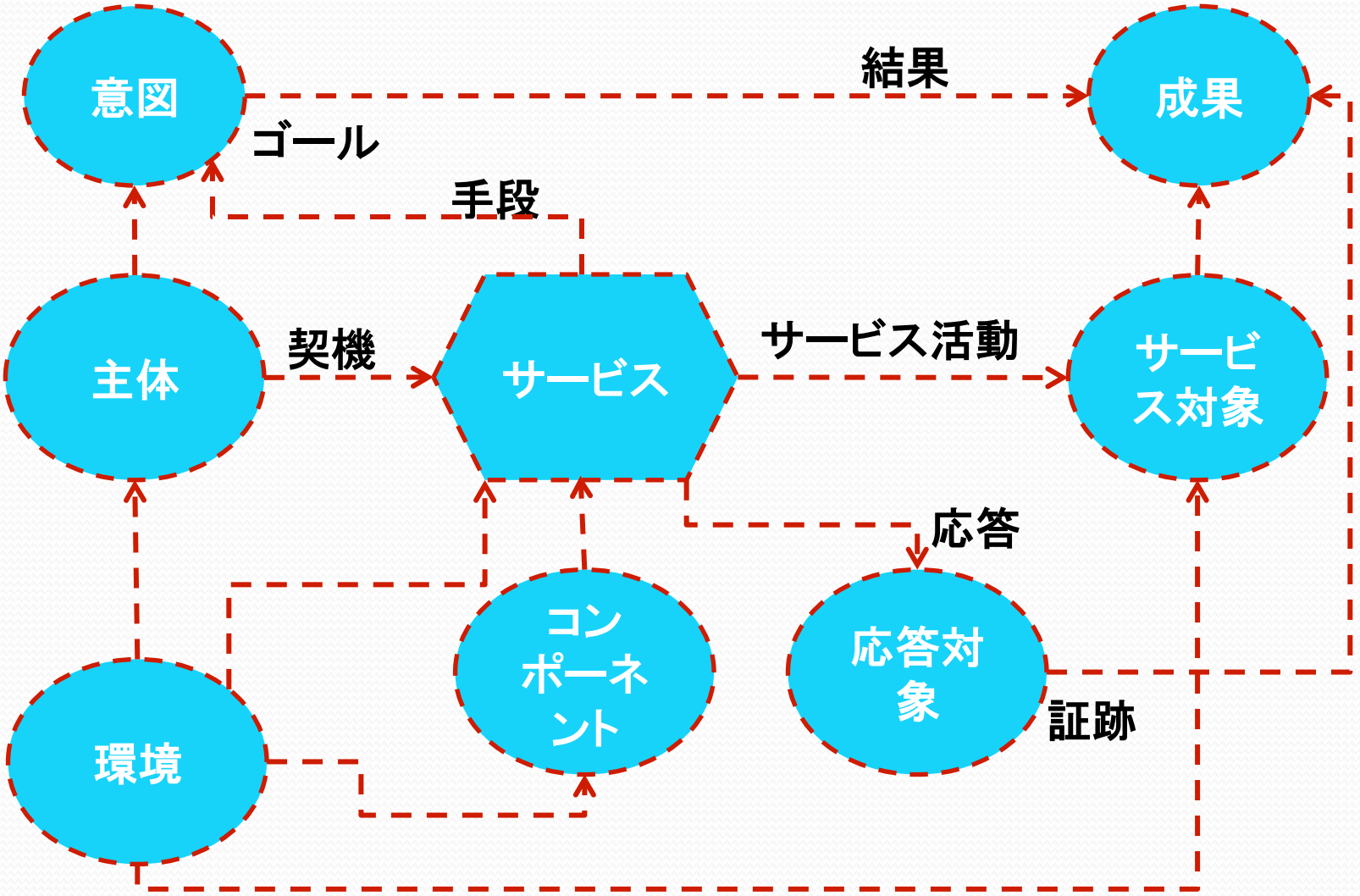
© Dependability Science Lab. Nagoya Univ. 2012

サービスシナリオの構成



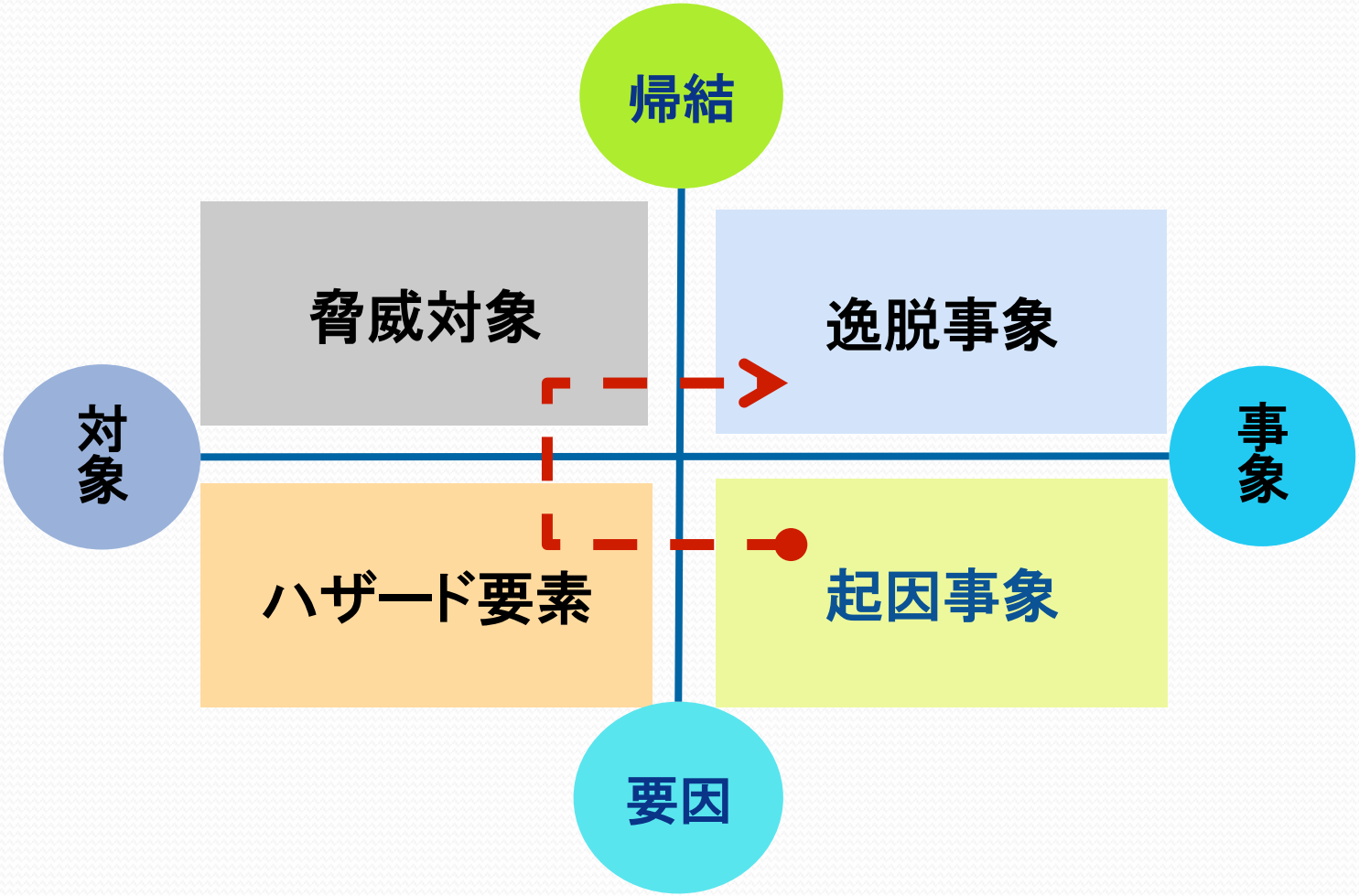
© Dependability Science Lab. Nagoya Univ. 2012

サービスリスク



© Dependability Science Lab. Nagoya Univ. 2012

ハザード条件



© Dependability Science Lab. Nagoya Univ. 2012

システム境界問題：System Boundary Problem

入力境界問題
不正
ない
遅延，早入
過小，過大
入力元例外

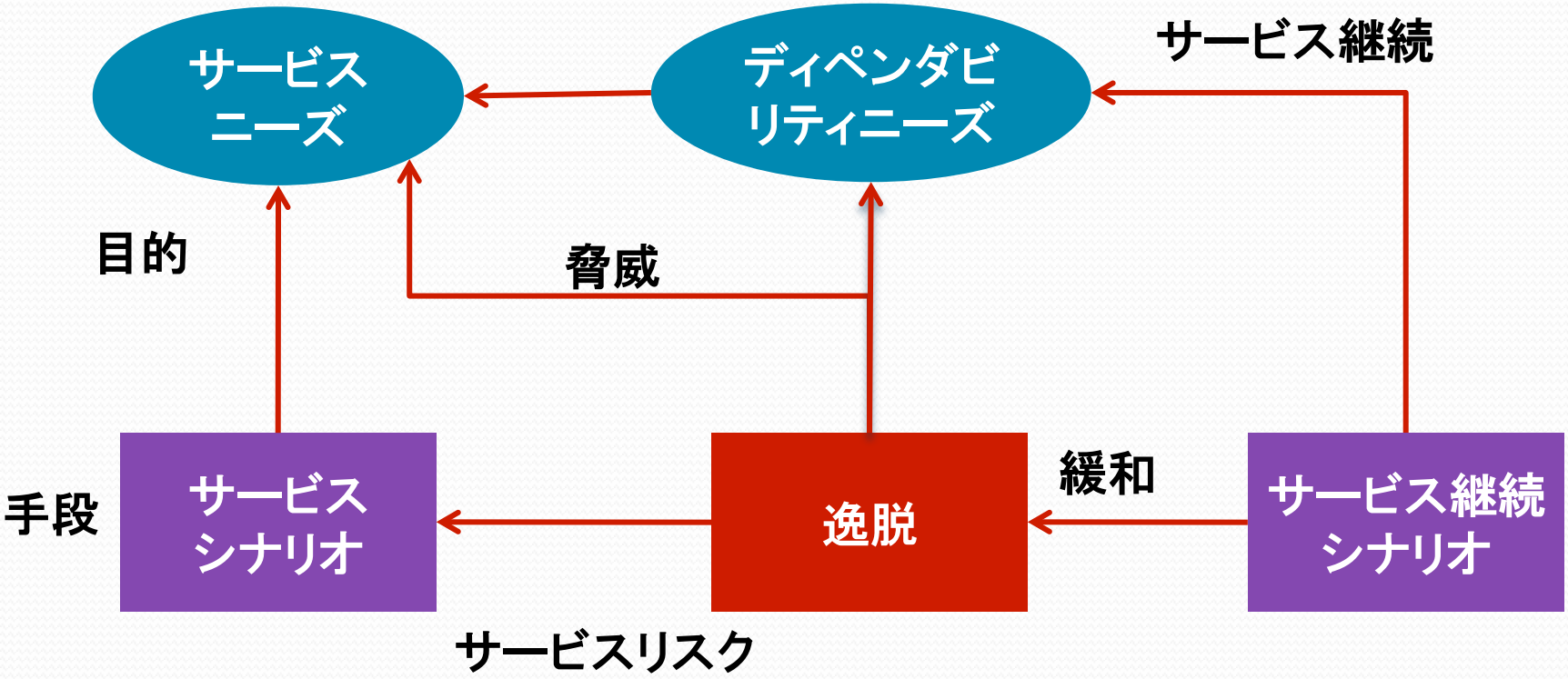
機能境界問題
不正
何もしない
開始しない
終了しない

出力境界問題
不正
ない
遅延，早出
過小，過大
出力先例外

ディペンダビリティケース＝入力，機能，出力の境界問題がないことを系統的に論証
システム構造からD-Case構造を半自動生成可能（証跡は無理）

© Dependability Science Lab. Nagoya Univ. 2012

サービス継続シナリオ



© Dependability Science Lab. Nagoya Univ. 2012

可用性リスクに対するサービス継続シナリオ

[非機能要求] 可用性
[主体] システムの外部環境
[対象] システム
[事前状況] 通常状態
[契機] 想定外のメッセージを受信
[シナリオ]
1) 受信した想定外メッセージから対象となる可用性資源を特定
2) 可用性資源の復旧通知
3) 可用性資源の動作を継続
[入力] 想定外のメッセージ
[出力] 可用性資源の運用続行通知
[応答] オペレータへ運用続行を通知
[事後状況] オペレータが運用続行メッセージを受理
[関係者] オペレータ
[役割分担] 想定外メッセージ受信時にオペレータが対応する.
[品質測定基準] 想定外メッセージ条件で，可用性を保証する資源について，想定外メッセージが発生する条件が定義されている.

参考）山本修一郎，情報連携アーキテクチャ分析技法，KBSE研究会，2011.3.11

© Dependability Science Lab. Nagoya Univ. 2012

リスク分析手法

© Dependability Science Lab. Nagoya Univ. 2012

リスク分析手法

技法		説明
SCBC	サービス合意形成カード Service consensus building card	SCBCによりサービス要求を定義し，ステークホルダ間で合意する
DCB	ディペンダビリティ管理委員会 Dependability Control Board	DCB ではSCBCによる合意形成プロセスを管理する．DCB委員はステークホルダの代表者である
DC Map	ディペンダビリティ制御マップ Dependability Control Map	DC マップでは，ステークホルダの責務とディペンダビリティゴール間の依存関係を記述する
D-Case DB	D-Case データベース	D-Caseは，サービスのディペンダビリティ要求に対するディペンダビリティゴールを達成するために格納される
SRBS	サービスリスク分解構造 Service Risk Brake-down Structure	サービスリスク分解構造SRBSによって，階層的にリスクを分類する
SFT	サービス故障木Service Fault Tree	サービス故障木SFTにより故障の論理的な条件を定義する
SCS	サービス継続シナリオ Service Continuity Scenario	サービス継続シナリオSCSによって，ディペンダビリティ要求に対するリスクを緩和する．D-ScriptsによってSCS を実現する
SRMT	サービスリスク管理表 Service Risk Management Table	サービスリスク管理表SRMTにより，サービスイベントシナリオの確率と影響 に基づいてサービスリスクを定義する
SRSMT	サービス要求状態管理表 Service Requirements State Management Table	サービス要求状態管理表SRSMTでは，要求の開発時だけでなく，要求の運用中も含めて，サービス要求状態を管理する

© Dependability Science Lab. Nagoya Univ. 2012

要求管理技法の関係

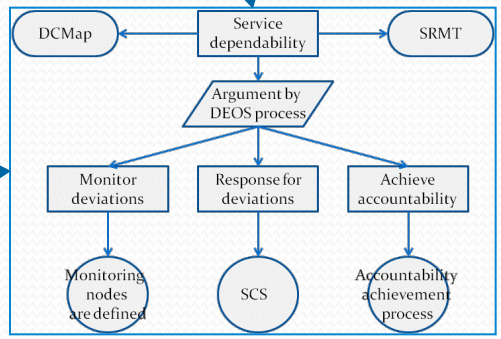
サービス合意形成カード SCBC

Requirements	To continue services against unexpected occurrences by service continuity scenarios		
Event	Deviations of service requirements are occurred	Input	Service continuity parameters
Response	Service continuity scenario (SCS) has been activated and the deviated service has been recovered	Output	Service continuity activity log records
Functional requirements	•Identify deviations through service executions •Determine SCS for each deviations •Apply SCS to the deviated situation •Confirm service continuity by successful achievement of SCS •Report incidents to DCB when the applied SCS unsuccessfully is stopped		
Initiation conditions	DCB is organized Service and dependability requirements are developed Risks and SCSs are developed		
Completion conditions	Valid SCS has been applied to the deviation and successfully completed Otherwise, the result has to be hierarchically escalated to DCB		
Roles of stakeholders	Providers of services or products	Define parameters for service continuity requirements Agree on the results of SCS application	
	Providers of systems	Develop SCSs	
	DCB	Build consensus on service continuity requirements and operations	

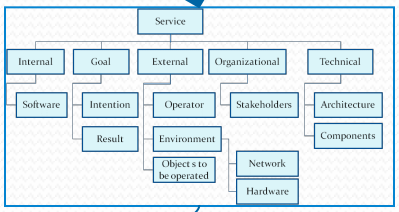
ディペンダビリティ制御マップ DC Map

Stakeholders	Roles	Dependability goals
Users	I	Consensus building
Systems providers	A,R, I	Accountability achievement
Developers	C, R	Customer satisfaction → Dependability
Maintainers	C, R	Valid operation
Hardware providers	C, R	Hardware Dependability
Certifiers	C, I	Valid HW authorization → Valid SW authorization

D-case



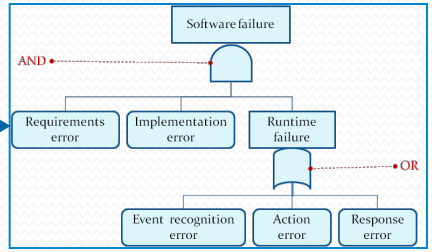
サービスリスク分解構造 SRBS



サービスリスク管理表 SRMT

Initial event	Dependability action			Scenarios	Probability	Impact	Risk
	D-script		Outer loop				
	application	execution					
Deviation detection	Success	Success	--	S_1	P_1	S_1	P_1S_1
		Failure	Success	S_2	P_2	S_2	P_2S_2
			Failure	S_3	P_3	S_3	P_3S_3
--	Failure		Success	S_4	P_4	S_4	P_4S_4
			Failure	S_5	P_5	S_5	P_5S_5
Failure probability			--	--	P_3+P_5	S	PS

サービス故障木 SFT



© Dependability Science Lab. Nagoya Univ. 2012

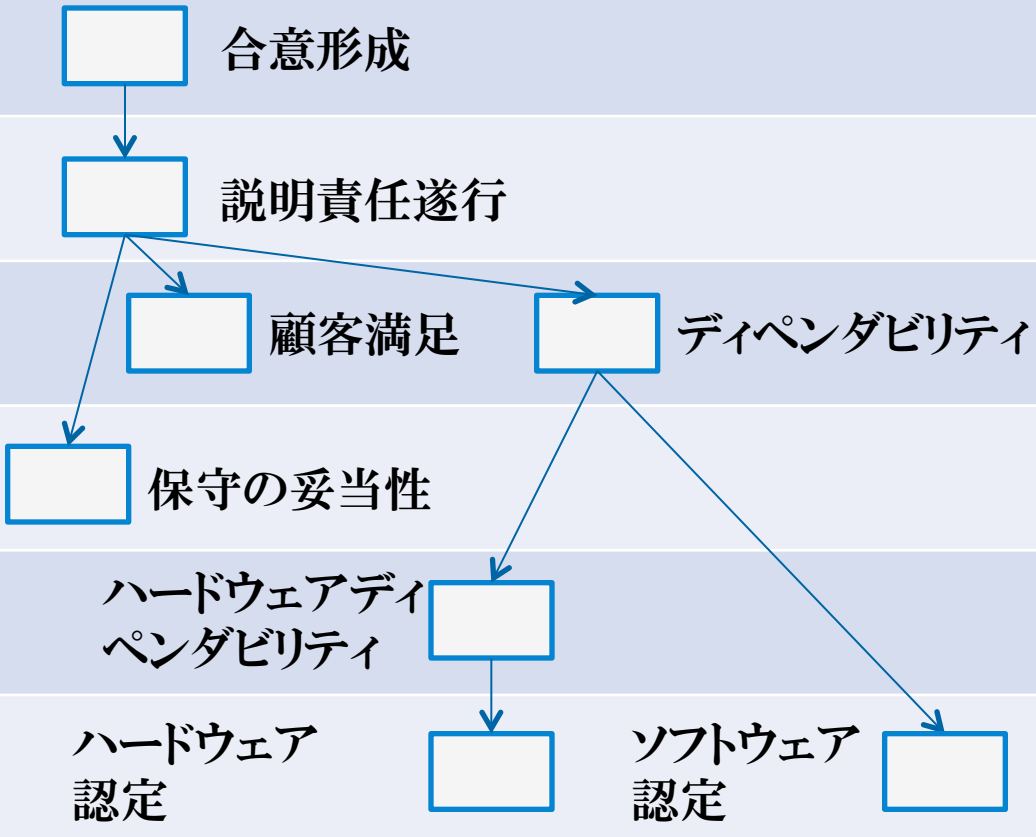
サービス合意形成カードの例

目的	サービス要求リスクを分析し，サービス継続シナリオを作成する		
入力	サービス要求 ディペンダビリティ要求	出力	サービス継続シナリオ
手順	•サービス要求の逸脱を識別する •ディペンダビリティ要求を分析する •逸脱原因への対策を決定する •ディペンダビリティ要求を満足するサービス継続シナリオを作成する		
開始条件	リスク分析規則が定義されている サービス要求，ディペンダビリティ要求が定義されている		
完了条件	リスク分析規則に基づいて，サービス要求とディペンダビリティ要求から，サービス継続シナリオが抽出されている		
役割	サービス，製品の提供者	サービス要求とディペンダビリティ要求を提示する	
	システム提供者	サービス継続シナリオを確認する	

© Dependability Science Lab. Nagoya Univ. 2012

ディペンダビリティ制御マップの例

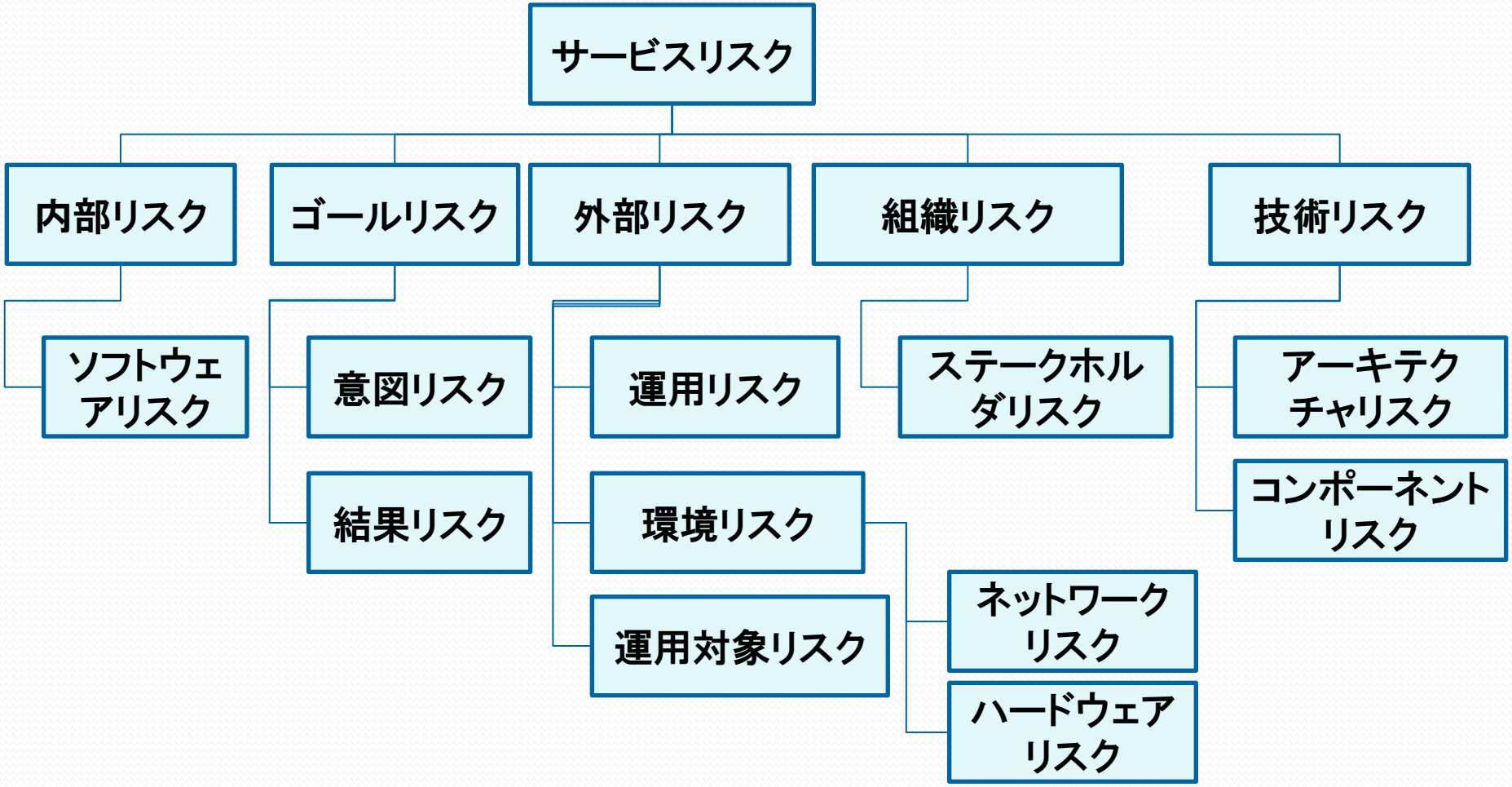
ステークホルダ	役割	ディペンダビリティゴール
ユーザ	I	
システム提供者	A,R, I	
開発者	C, R	
保守者	C, R	
ハードウェア提供者	C, R	
認定者	C, I	



R: Responsible, A: Accountable, C: Consulted, I: Informed

© Dependability Science Lab. Nagoya Univ. 2012

サービスリスク分解構造の例



Ref: PMBOK 4th edition

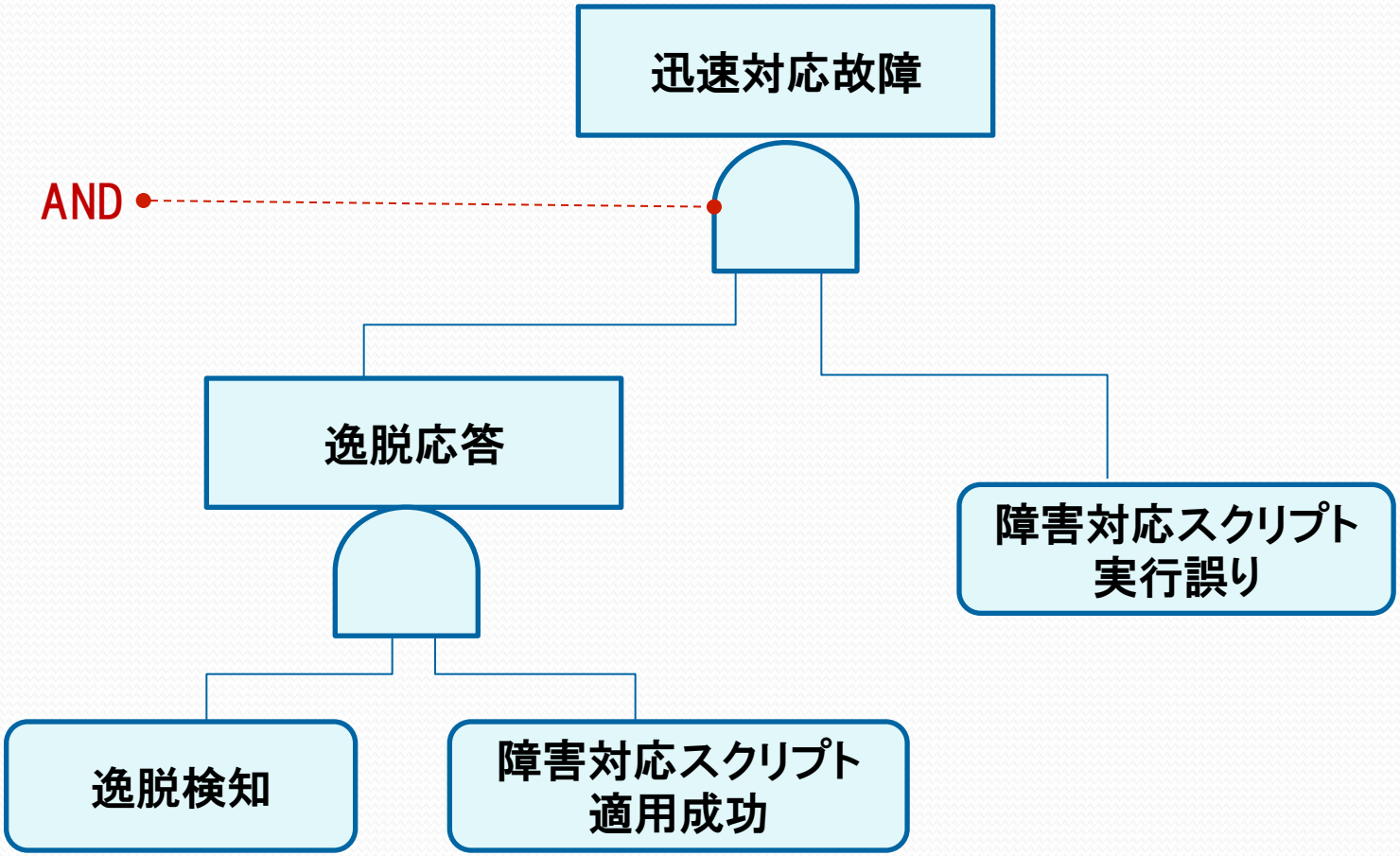
© Dependability Science Lab. Nagoya Univ. 2012

サービスリスク管理表の例

開始イベント	ディペンダビリティ活動			シナリオ識別子	確率	影響	リスク
	障害対応スクリプト		外部ループ				
	適用	実行					
逸脱検知	成功	成功	--	S1	P1	S1	P1S1
		失敗	成功	S2	P2	S2	P2S2
			失敗	S3	P3	S3	P3S3
	失敗		成功	S4	P4	S4	P4S4
			失敗	S5	P5	S5	P5S5
故障確率			--	--	P3+P5	S	PS

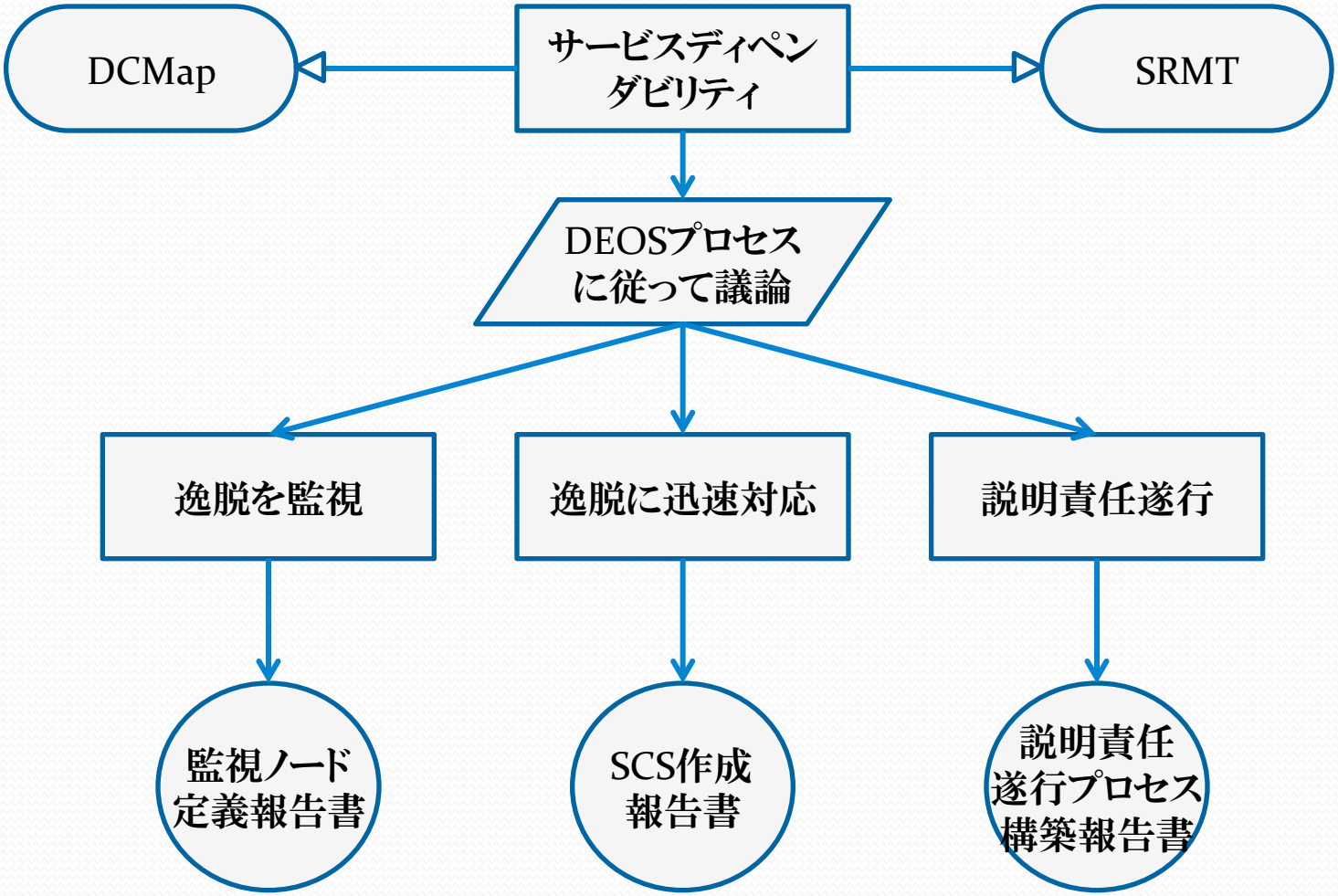
© Dependability Science Lab. Nagoya Univ. 2012

サービスリスク管理表の行に対する故障木の例



© Dependability Science Lab. Nagoya Univ. 2012

ディペンダビリティケースの例



© Dependability Science Lab. Nagoya Univ. 2012

まとめ

- システム障害事例
- 独立検証確認
- サービス継続性
- リスク分析手法

© Dependability Science Lab. Nagoya Univ. 2012

名古屋大学

D-Case入門
5 Assurance Case概要

© Dependability Science Lab. Nagoya Univ. 2012

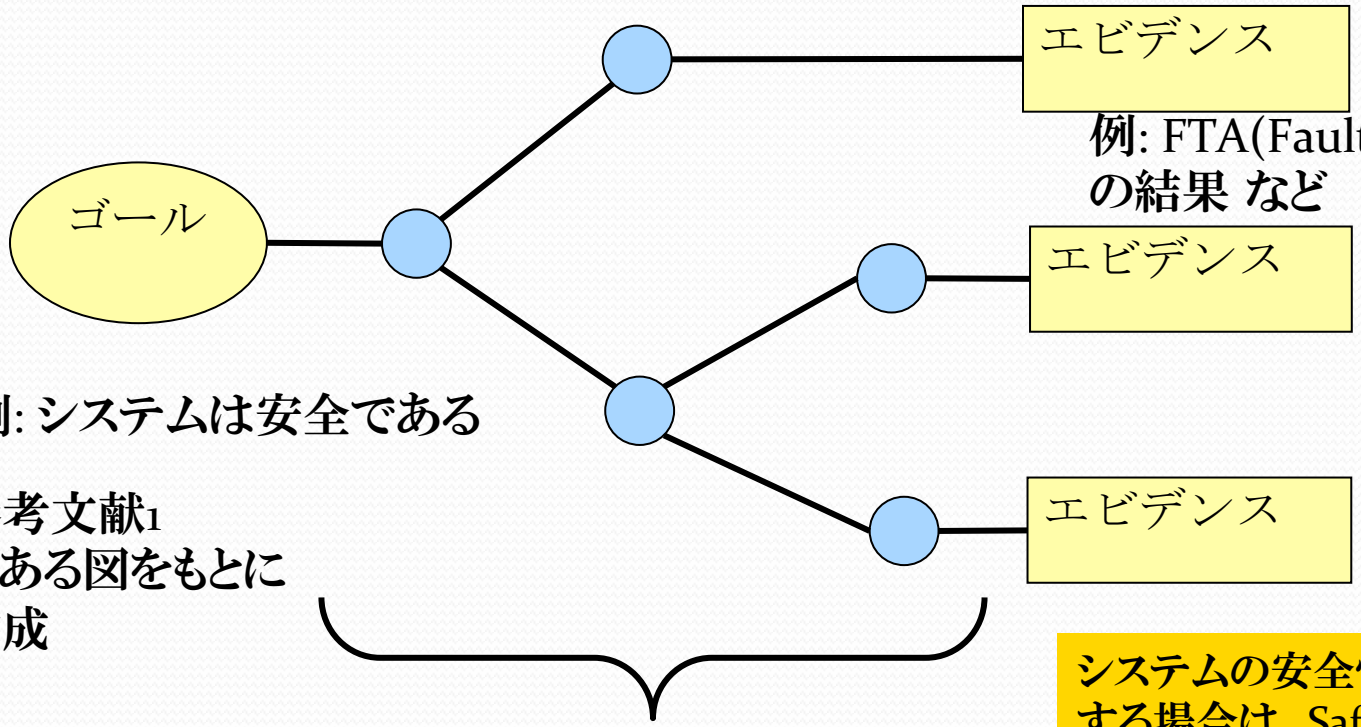
内容

- システムの安全性、ディペンダビリティの保証のための文書であるAssurance Caseについて概説を行う
  - その表記法の一つであるGSN(Goal Structuring Notation)について説明する
  - 現在、どのように使われているのかを説明する
  - 関連規格、ツールなどもあわせて紹介する

© Dependability Science Lab. Nagoya Univ. 2012

Assurance Caseとは

- システムの安全性、ディペンダビリティなどを、エビデンスを元に議論するための文書



システムの安全性を議論する場合は、Safety Case,
ディペンダビリティを議論する場合は、Dependability Caseなどと呼ばれる

© Dependability Science Lab. Nagoya Univ. 2013

Assurance Caseの定義

- 参考文献1 “A *documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment*”
- 参考文献2 ISO/IEC 15026
  - 保証ケースの構造と内容に対する最低限の要求を規定
  - 保証ケースの内容
    - システムや製品の性質に対する主張（claim）
    - 主張に対する系統的な議論（argumentation）
    - この議論を裏付ける証跡（evidence）
    - 明示的な前提（explicit assumption）
  - 議論の過程で，補助的な主張を用いることにより，最上位の主張に対して証跡や前提を階層的に構成

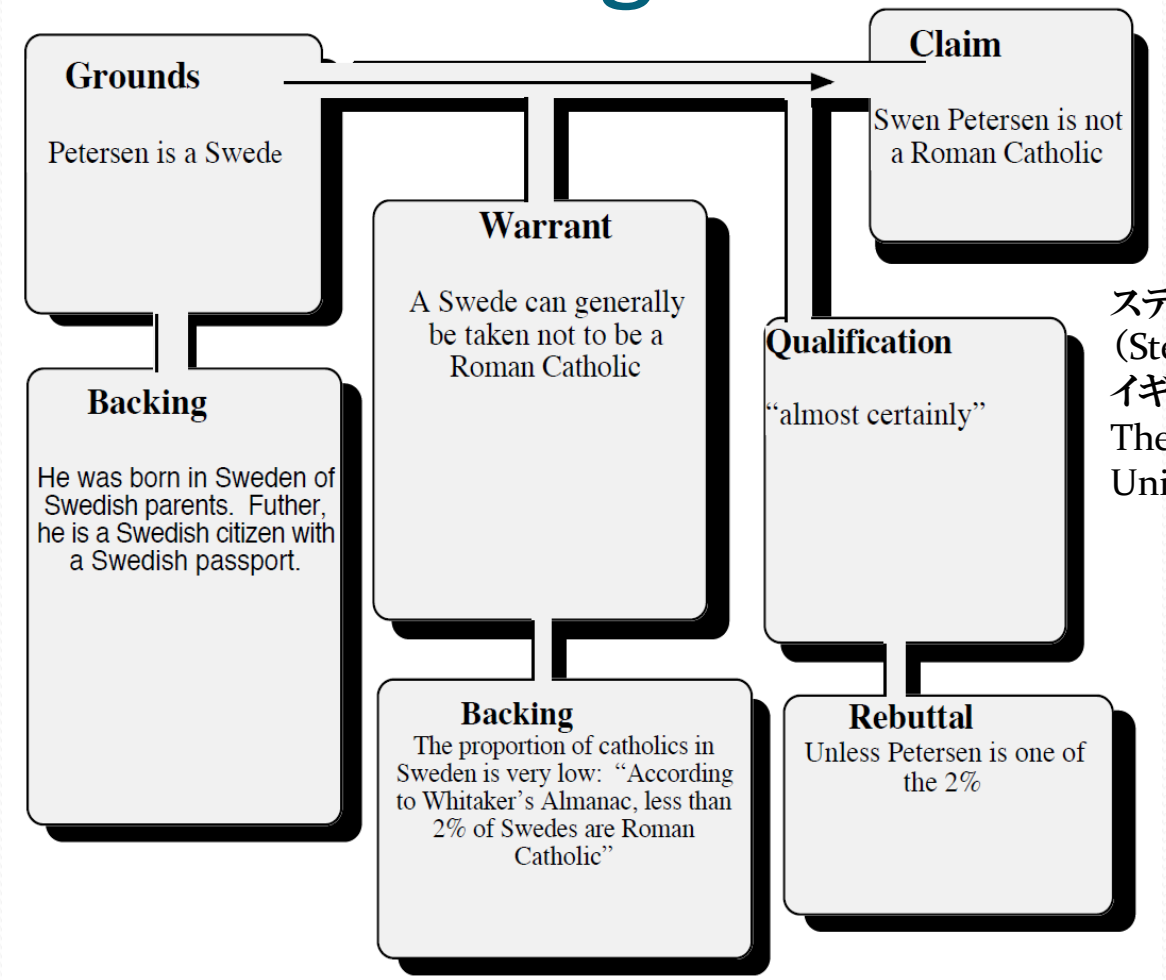
© Dependability Science Lab. Nagoya Univ. 2012

Assurance Caseの背景

- 1988年の北海油田事故(167名死亡)などを契機に、欧米で規格認証の際に提出が義務付けられるまでに普及
  - 手順のみでなく、なぜ安全性が保たれるのか、明示された議論で、エビデンスをもとに保証する
  - 導入により北海油田における事故が減少
- ISO 26262 (TC22/SC3, 自動車の機能安全規格)の要求項目の一つ
- 主な研究者
  - Tim Kelly (York大): GSN (Goal Structuring Notation)
  - Robin Bloomfield (City University London, Aderald社): Safety Criticalシステムコンサルティング、ASCEツール

© Dependability Science Lab. Nagoya Univ. 2012

Assurance Caseの理論的背景: Toulmin's Argument Model



参考文献3
Ed Roberts, Assurance Cases: Motivation, Technologies, Development

スティーヴン・トゥールミン
(Stephen Edelston Toulmin、1922–2009)
イギリス生まれの哲学者。専門は、科学哲学。
The Uses of Argument, (Cambridge University Press, 1958)

© Dependability Science Lab. Nagoya Univ. 2012

Warrant パターン

参考文献3
Ed Roberts, Assurance Cases: Motivation, Technologies, Development

1. *Generalization* - what is true of a well chosen sample is likely to hold for a larger group or population, or that certain things consistent with the sample can be inferred of the group/population.

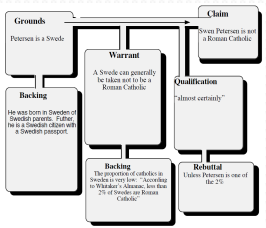
2. *Analogy* - Extrapolating from one situation or event based on the nature and outcome of a similar situation or event (e.g. 'case-based' and precedent-based reasoning used in legal discourse)

3. *Sign/Clue* - certain types of evidence are symptomatic of some wider principle or outcome. E.g. , smoke is often considered a sign for fire.

4. *Causal* - given occurrence or event is the result of, or is effected by, factor X. (Danger: Mixing up correlation with causation)

5. *Authority* - Person X or text X constitute an authoritative source

6. *Principle* - Locating a principle that is widely regarded as valid and showing that a situation exists in which this principle applies.



© Dependability Science Lab. Nagoya Univ. 2012

Assurance Case, Safety Case, Dependability Case, …

- Assurance Caseは日本語だと保証ケース
- Assurance Caseは、安全性を議論する場合はSafety Case, ディペンダビリティを議論する場合はDependability Caseと呼ばれる

© Dependability Science Lab. Nagoya Univ. 2012

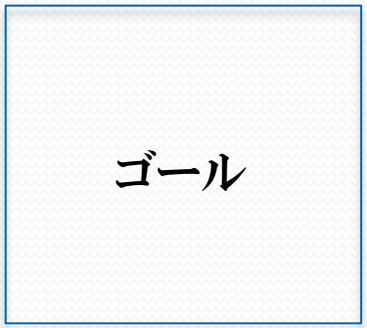
Assurance Caseの表記法

- 主要なグラフィカルな表記法
  - GSN (Goal Structuring Notation)
    - イギリスUniversity of Yorkで開発
  - CAE (Claim, Argument, Evidence)
    - イギリスAdelard社、City University Londonで開発

本入門ではGSNを説明する

© Dependability Science Lab. Nagoya Univ. 2012

GSNの主要なノードの種類



保証したいこと、命題
例：
システムは安全である

ゴールをサブゴールに分けるときの考え方：
個別の障害ごとに議論する

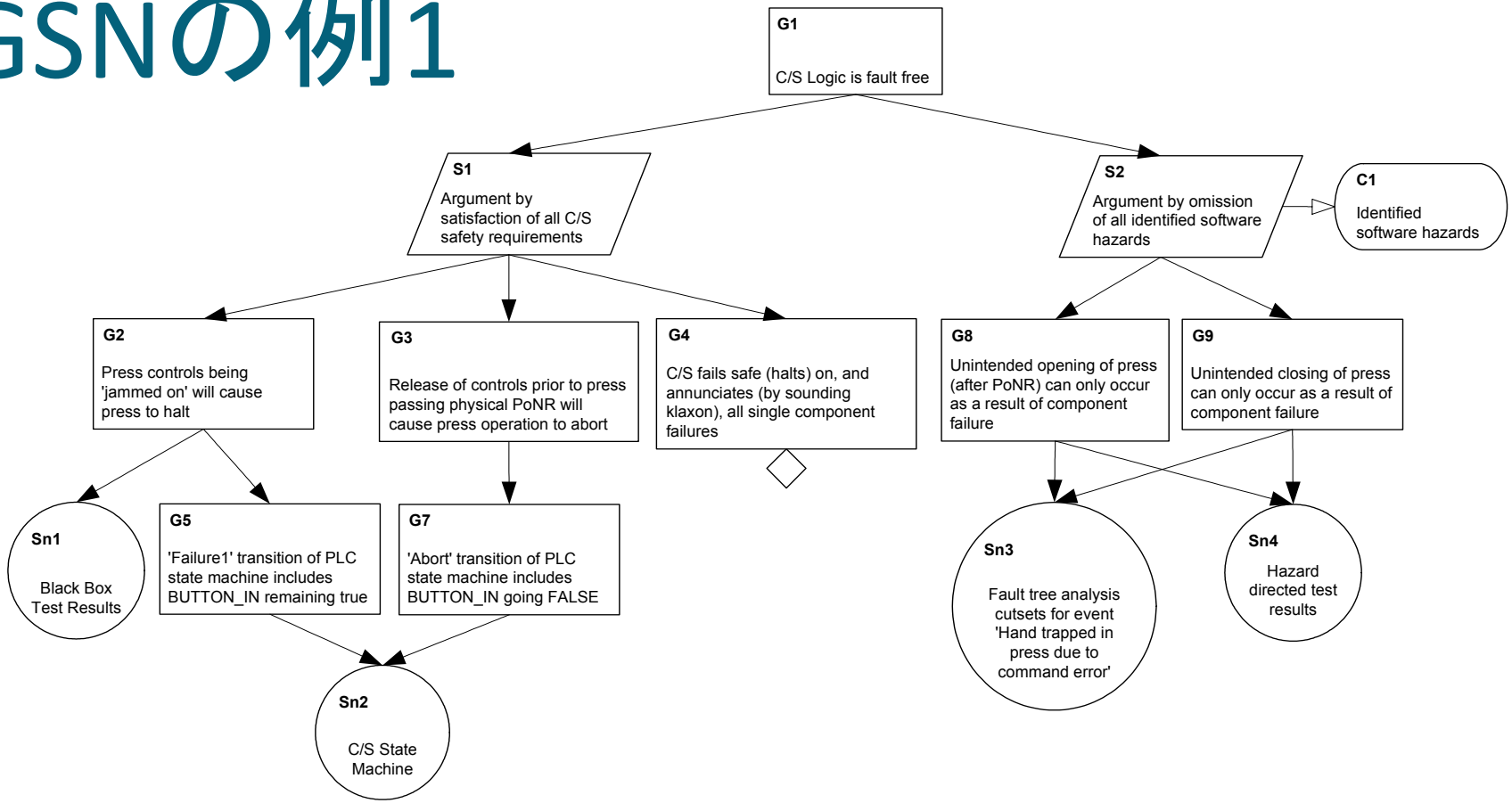
ゴールが成り立つことを最終的に保証するもの
例：テスト結果、運用事例など

システムの状態、環境など、ゴールを議論するときの前提など
例：リスク分析の結果得られたハザードのリスト

ゴールをサポートするエビデンス、議論がまだないときに使う

© Dependability Science Lab. Nagoya Univ. 2012

GSNの例1

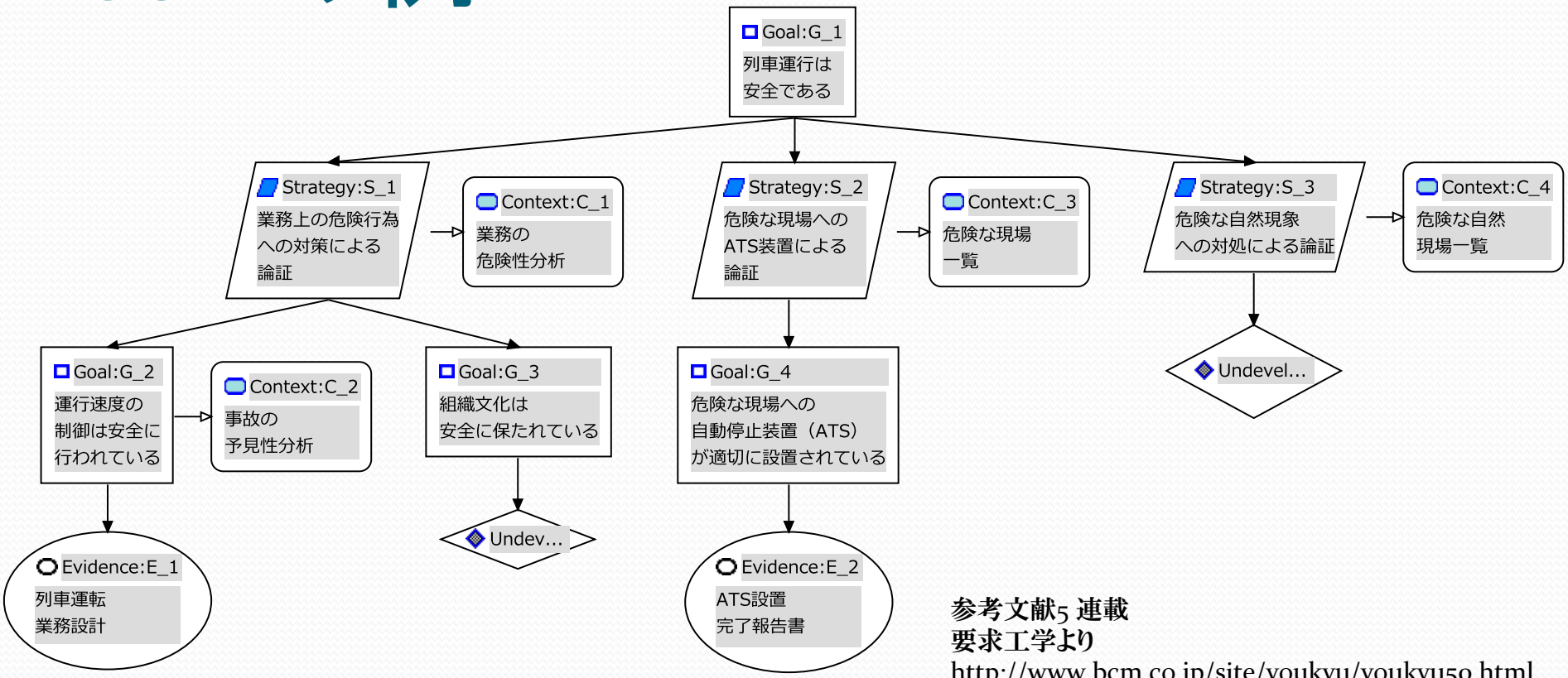


参考文献4

T.Kelly, R. Weaver, Goal Structuring Notation: A Safety Argument Notation , In Proc. Workshop on Assurance Cases, DSN 2004

© Dependability Science Lab. Nagoya Univ. 2012

GSNの例2



参考文献5 連載
要求工学より
http://www.bcm.co.jp/site/youkyu/youkyu50.html

© Dependability Science Lab. Nagoya Univ. 2012

Assurance Caseの現況(1/2)

- イギリスにおいては、高安全、軍事システムなどの調達に必須
  - システム供給者、防衛省、第3者コンサルティング会社間になどにおけるリスクコミュニケーションに使われる
  - MoD Defence Standard 00-56
- US. Food and Drug Administration(FDA)
  - Infusion Pump (参考文献7)
    - “… and for making an argument as to why the evidence, your data and analyses, supports the claim”
    - “A safety case is the best way to both document and review a submittal based on a risk management approach because the argument shows the proportionality of the mitigation”

参考文献6
2007年にNational Academy of Scienceから、”Software for Dependable Systems: Sufficient Evidence?”という調査報告書が発行され、Dependability Caseの必要性が言及されたことが一つの契機になっている

© Dependability Science Lab. Nagoya Univ. 2012

Assurance Caseの現況(2/2)

- 2006年9月2日、アフガニスタンで作戦飛行中のMR.2 XV230が、空中給油を受けた直後に火災が発生して墜落する事故が発生した。2007年12月4日、イギリス国防省は調査報告を発表し、墜落した機体は、給油後タンクから燃料漏れが生じており、高温空気パイプの熱によって発火、拡大して墜落に至った、と分析した
  - Safety Caseがいい加減であったことがわかった。きちんと書かれていれば、欠陥を発見できたであろう報告された

参考文献8 The Nimrod Review
http://www.official-documents.gov.uk/document/hc0809/hc10/1025/1025.pdf

© Dependability Science Lab. Nagoya Univ. 2012

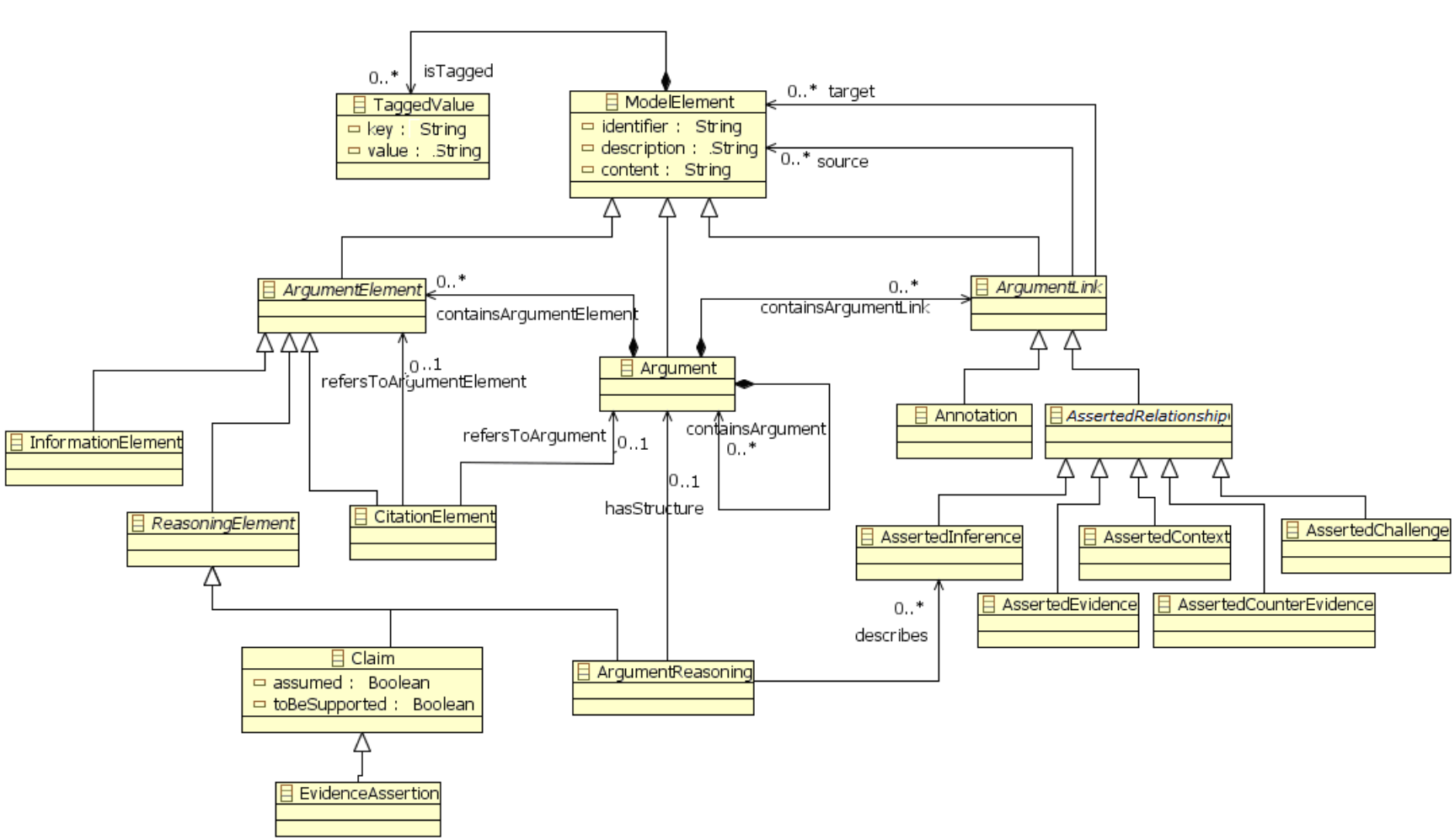
Assurance Caseに関する国際標準規格

- ISO/IEC
  - 15026 System and Software Assurance
    - Assurance Caseの基本構造、用語の定義
- OMG
  - ARM (ARgument Metamodel), SAEM(Software Assurance Evidence Metamodel)
    - GSNとCAEを統合したメタモデル、ソフトウエア保証のためのエビデンスのメタモデルの標準化
  - ARMとSAEMは統合されてSACM(Structured Assurance Case Metamodel)となった(2012年6月)

© Dependability Science Lab. Nagoya Univ. 2012

名古屋大学

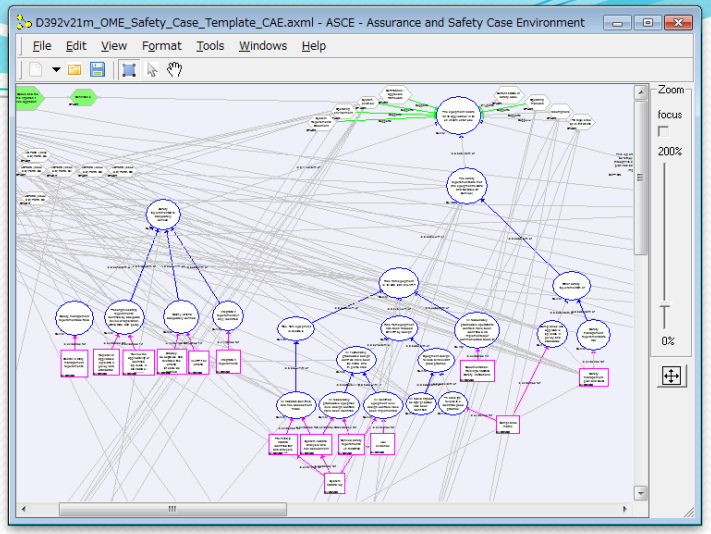
ARM



© Dependability Science Lab. Nagoya Univ. 2012

ツール

- ASCE Tool
  - CAE, GSNなどを編集できるツール
  - イギリスAdelard社が開発・販売している
  - 軍事関係だけでなく、医療機器の安全性記述などにも近年用いられている
- Certware(http://nasa.github.com/CertWare/)
  - NASAが開発したオープンソースのツール
  - 既存のARMなどの規格に多く準拠
  - 多くの実験的な機能が入っている

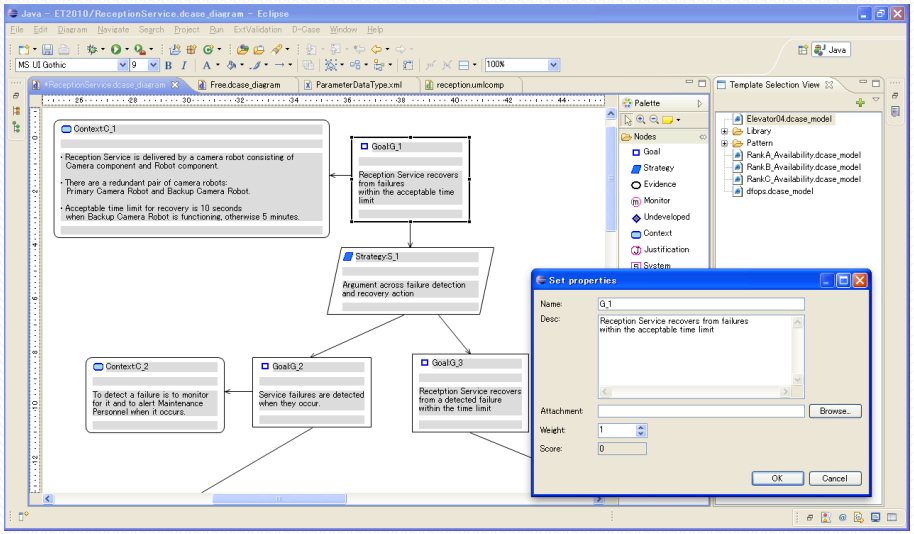


ASCE Tool

© Dependability Science Lab. Nagoya Univ. 2012

D-Case Editor

- DEOSで開発中のEclipseプラグイン
  - フリーウエアとして公開中



http://www.il.is.s.u-tokyo.ac.jp/deos/dcase/

© Dependability Science Lab. Nagoya Univ. 2012

まとめ

- システム保証のための手法であるAssurance Caseの概説を行った
  - その表記法の一つであるGSNを説明した
  - 現在どのように使われているか説明した
  - 関連規格、ツールなどもあわせて紹介した

© Dependability Science Lab. Nagoya Univ. 2012

参考文献

1. Peter Bishop and Robin Bloomfield, A Methodology for Safety Case Development, In Proc. Safety-critical Systems Symposium (SSS 98)
2. ISO/IEC 15026-2:2011, Systems and Software engineering—Systems and Software assurance—Part2: Assurance case
3. Ed Roberts, Assurance Cases: Motivation, Technologies, Development, Presented in RTES forum, Open Group Conference, Jan 2012, San Francisco
4. T. Kelly, R. Weaver, Goal Structuring Notation: A Safety Argument Notation, In Proc. Workshop on Assurance Cases, DSN 2004
5. 山本修一郎、連載要求工学、http://www.bcm.co.jp/site/youkyu/
6. Daniel Jackson, Martyn Thomas, and Lynette I. Millett, Software for Dependable Systems – Sufficient Evidence?, The National Academies Press, Washington D.C., 2007
7. Richard Chapman, Assurance Cases for External Infusion Pumps http://www.fda.gov/downloads/MedicalDevices/NewsEvents/WorkshopsConferences/UCM217456.pdf, 2010
8. The Nimrod Review, http://www.official-documents.gov.uk/document/hc0809/hc10/1025/1025.pdf

© Dependability Science Lab. Nagoya Univ. 2012

名古屋大学

D-Case入門
6 D-Case作成法

© Dependability Science Lab., Nagoya University, 2012

はじめに

- DEOSプロジェクトではこれまでに、いくつかのD-Case/Assurance Case記述実験を行なってきた
  - デモシステム
  - エレベータ (過去の経験から)
  - センサネットワークシステム
  - マインドストームロボットコンテスト
  - バージョンコントロールシステム
  - 受付ロボット
  - スーパーコンピュータの運用マニュアル
  - TOGAF
  - 自動車のエンスト問題
  - などなどその他予定中

© Dependability Science Lab., Nagoya University, 2012

D-Case/Assurance Caseのガイドブックの必要性

- Assurance Caseはまだ普及が始まった段階にある
  - “There are problems in every aspect of assurance cases: Building them, Reviewing them, Maintaining them, and Reusing them” [20] T.Scott Ankrum. Assurance Case Frameworks Part of High Confidence Software, MITRE Software Engineering Center, March 11, 2004
- 本講義では開発中のD-Caseの記述ステップを説明する
  - まとめて、D-Caseポケットブックを出版する予定

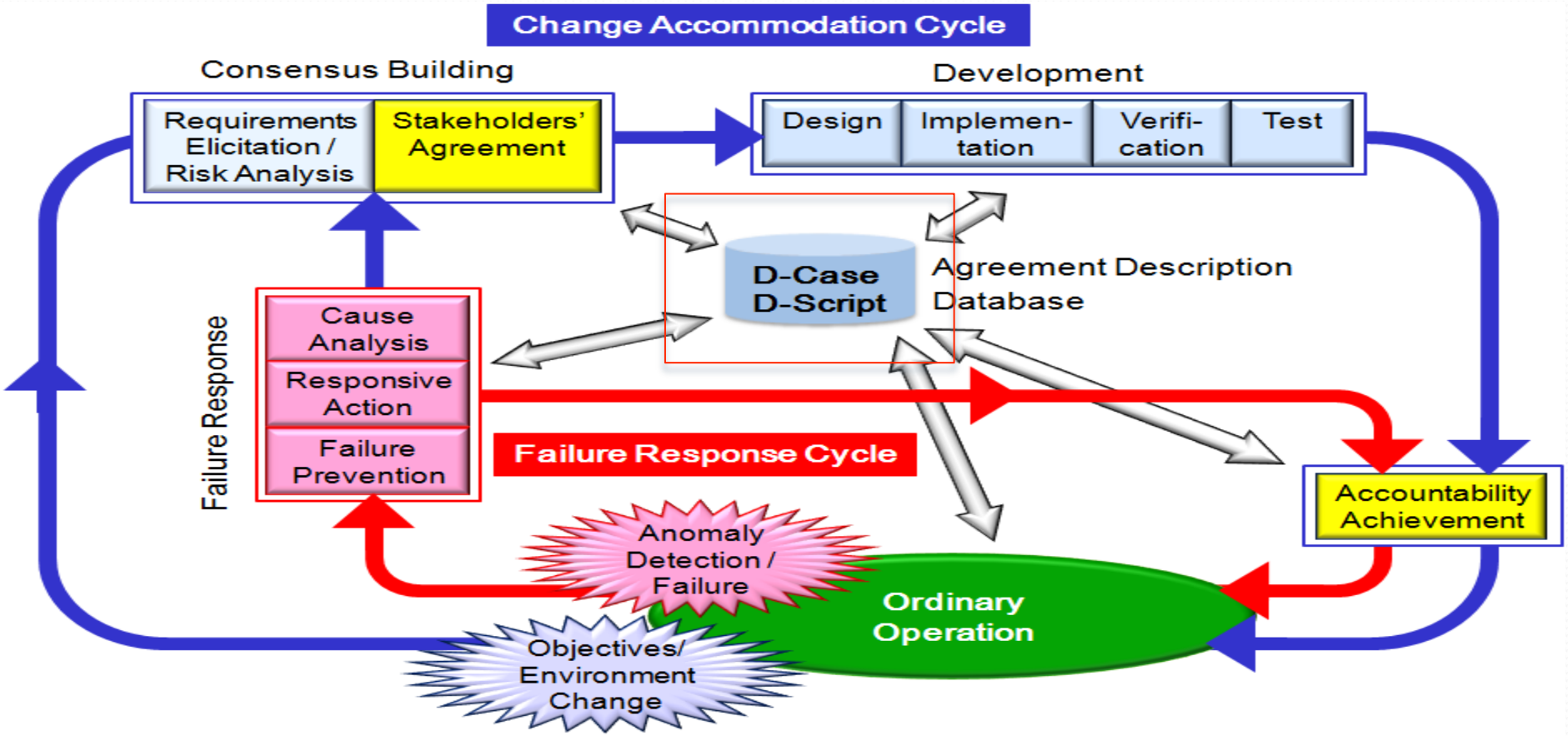
© Dependability Science Lab., Nagoya University, 2012

本講義の内容

- はじめに
- D-Caseとは？
- 記述基本ルール
- 記述ステップ
- 記述例
- 既存のガイドライン
- まとめ: D-Case 10箇条
- ツールと参考文献

© Dependability Science Lab., Nagoya University, 2012

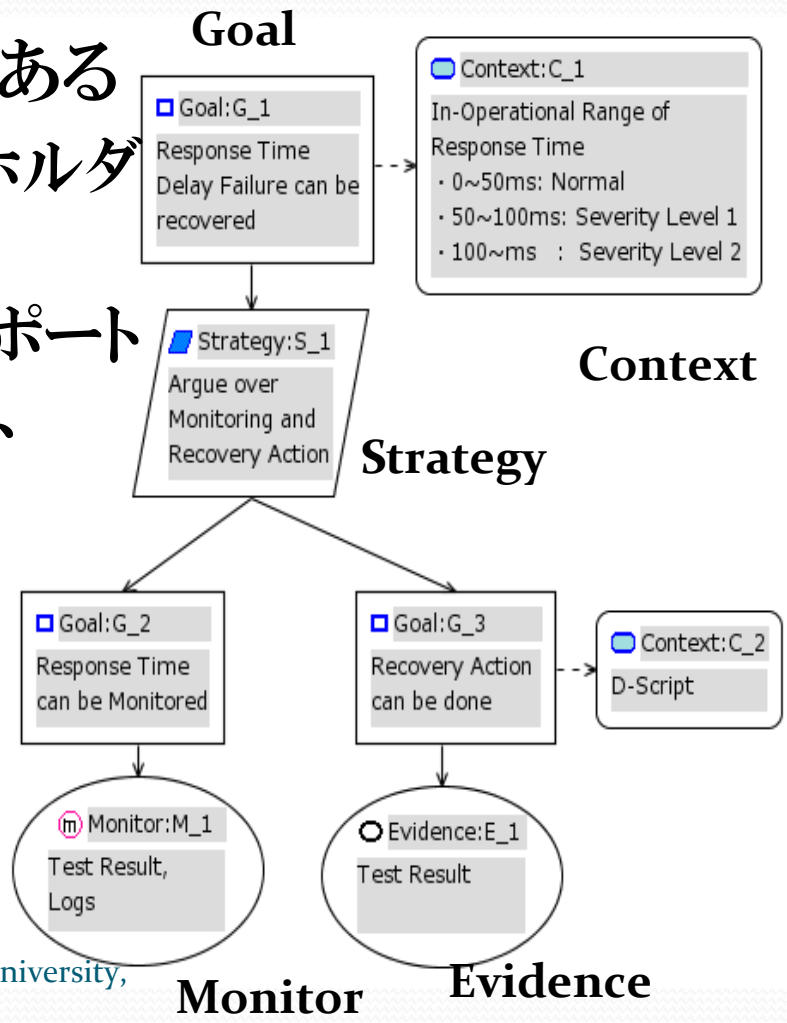
DEOSプロセスとD-Case



© Dependability Science Lab., Nagoya University, 2012

D-Caseとは

- D-CaseはAssurance Caseの拡張である
  - 議論とエビデンスに基づくステークホルダ合意のための構造化された記法
  - ステークホルダと合意のマネージサポート
  - 運用時のゴール達成の監視サポート（モニタリングノード）
  - 合意記述の整合性検査サポート
- Assurance Case一般に関する情報は5 アシュアランスケース概論を参照のこと



© Dependability Science Lab., Nagoya University, 2012

D-Caseの基本ルール概要

- D-CaseはGSN (Goal Structuring Notation)を元にしている

  [5] Tim Kelly and Rob Weaver, The Goal Structuring Notation – A Safety Argument Notation, DSN Workshop on Assurance Cases, July 2004
- D-Caseはおおまかには木構造をしている。トップゴールはシステムのディペンダビリティに関する命題である
- トップゴールはサブゴールに分解される

© Dependability Science Lab., Nagoya University, 2012

D-Case Nodes

- **Goal**

System is Dependable

- **Context**

List of all identified hazards

- **Strategy**

Argue over identified hazards

- Evidence

FTA Results

- Undeveloped

- Monitor

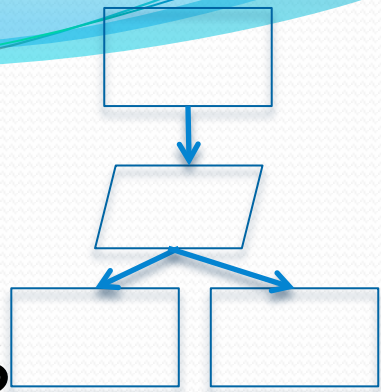
Runtime Log Results

- External

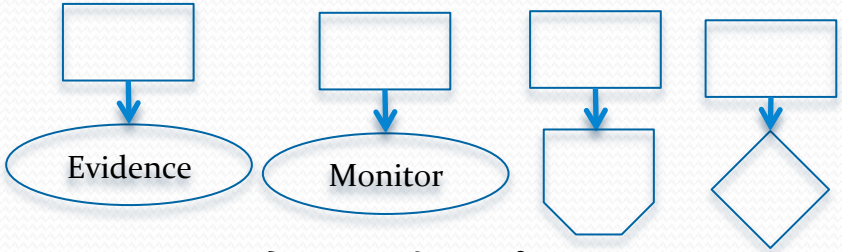
© Dependability Science Lab., Nagoya University, 2012

D-Caseの拡張

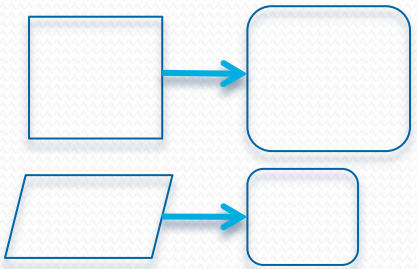
構文に関する基本規則



- ゴールはストラテジを通してサブゴールに分解される
- D-Caseのリーフは、エビデンス、モニタ、エクスターナル、アンデベロップドのいずれかである



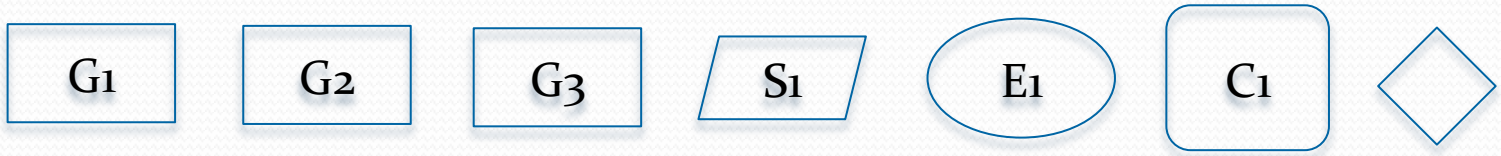
- コンテキストはゴールもしくはストラテジに付与される
- 矢印は2種類ある
  - Solved By: ゴールからストラテジ, ストラテジからゴール, ゴールからエビデンス, モニタ, エクスターナル, とアンデベロップド
  - In Context of: ゴールからコンテキスト、ストラテジからコンテキスト



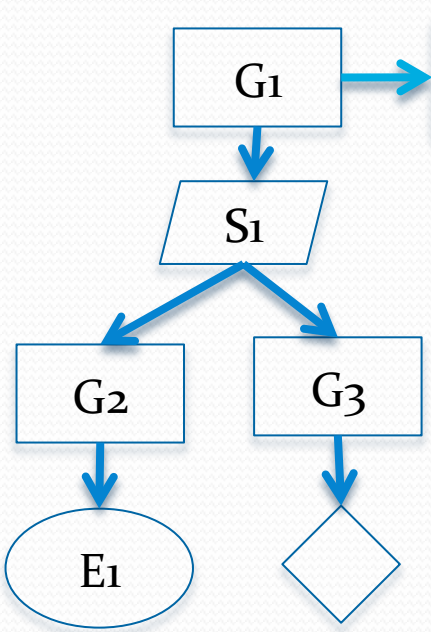
© Dependability Science Lab., Nagoya University, 2012

D-Case構文クイズ

- 下のノードを使って、正しい構文のD-Caseを、→と→を使って作ってください



解答例



本格的な演習は
7 D-Case演習に
あります

© Dependability Science Lab., Nagoya University, 2012

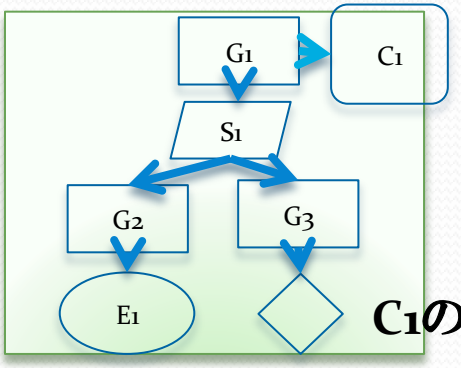
D-Case記述基本規則 (1/2)

- ゴールはS + Vの形をした、命題である
  - System X is dependable, System Y is acceptably safe,...
  - ゴールは命令であったり挨拶などではない
    - Not: “execute a program” , “Hello, World”...
- ストラテジはゴールをサブゴールに分割するときの議論の仕方を説明する
  - Argue over each identified hazard（識別されたハザードごとに議論する）, Argue over system lifecycle phases（ライフサイクルフェーズごとに議論する）, ...
- エビデンスは物 (ドキュメント), であり、文章ではない。エビデンスはゴールを最終的にサポートするものである
  - FTA解析結果, 専門家の意見, 運用訓練結果, ...

© Dependability Science Lab., Nagoya University, 2012

D-Case記述基本規則 (2/2)

- モニタは運用中のシステムから得られるデータ、ドキュメントでである
  - 運用ログデータ
- エクスターナルは他のシステムのD-Caseへのリンクである（現在開発中）
- コンテキストはシステム運用環境、定義などをゴールもしくはストラテジを議論する上での前提が書かれたドキュメントである
  - 運用環境、識別されたリスクのリスト、システム構造、 ディペンダビリティ要求, 語彙の定義
  - コンテキストはそれが付与されたゴール、ストラテジ、その下のサブツリー全体にかかる



C1のスコープ

© Dependability Science Lab., Nagoya University, 2012

D-Case記述クイズ

- 以下をゴール、ストラテジ、コンテキスト、エビデンス、モニタに分類してください
  - System is acceptably safe **Goal**
  - Argue over each fault **Strategy**
  - System Structure Model **Context**
  - Fault Tree Analysis Result **Evidence**
  - Good Morning! **これは間違いです**
  - Network Access Counter Logs **Monitor**

本格的な演習は7 D-Case演習にあります

© Dependability Science Lab., Nagoya University, 2012

D-Case記述ステップ

- D-Caseは様々な種類のドキュメントを元に作成される。例えば、UML図、データフロー図、リスク分析、検証、テスト結果、障害対応スクリプト(D-Script)などなど
- 本講義では、そうしたドキュメントからD-Caseを作成するステップを提案する

© Dependability Science Lab., Nagoya University, 2012

ステップ概要 (1/2)

1. システムライフサイクルを整理し、それぞれのフェーズの入力、出力ドキュメントをまとめる
2. 入出力ドキュメントを分類する。例えば
    1. 規格
    2. リスク分析結果
    3. ディペンダビリティ要求
    4. システムライフサイクル文書
    5. システム構造モデル
    6. 運用情報
    7. 環境情報
    8. テスト、検証結果
    9. プログラムコード
3. トップゴールを置く: “システムはディペンダブルである”

© Dependability Science Lab., Nagoya University, 2012

ステップ概要 (2/2)

4. ディペンダビリティ要求、環境情報、語彙定義をコンテキストとしてトップゴールにおく
5. 大まかにD-Caseの構造を考える
6. 必要なドキュメントをコンテキストとして置く
7. ドキュメントからD-Caseのサブツリーを作る
8. サブツリーができていない部分を作る
9. 上記を必要なだけ繰り返す

© Dependability Science Lab., Nagoya University, 2012

ステップ1: システムライフサイクルを整理し、それぞれのフェーズの入力、出力ドキュメントをまとめる

- D-Caseはシステムライフサイクルで作られるドキュメントを元に作られるので、ライフサイクルと、フェーズの入出力ドキュメント整理する必要がある。例えば

システムライフサイクルの例

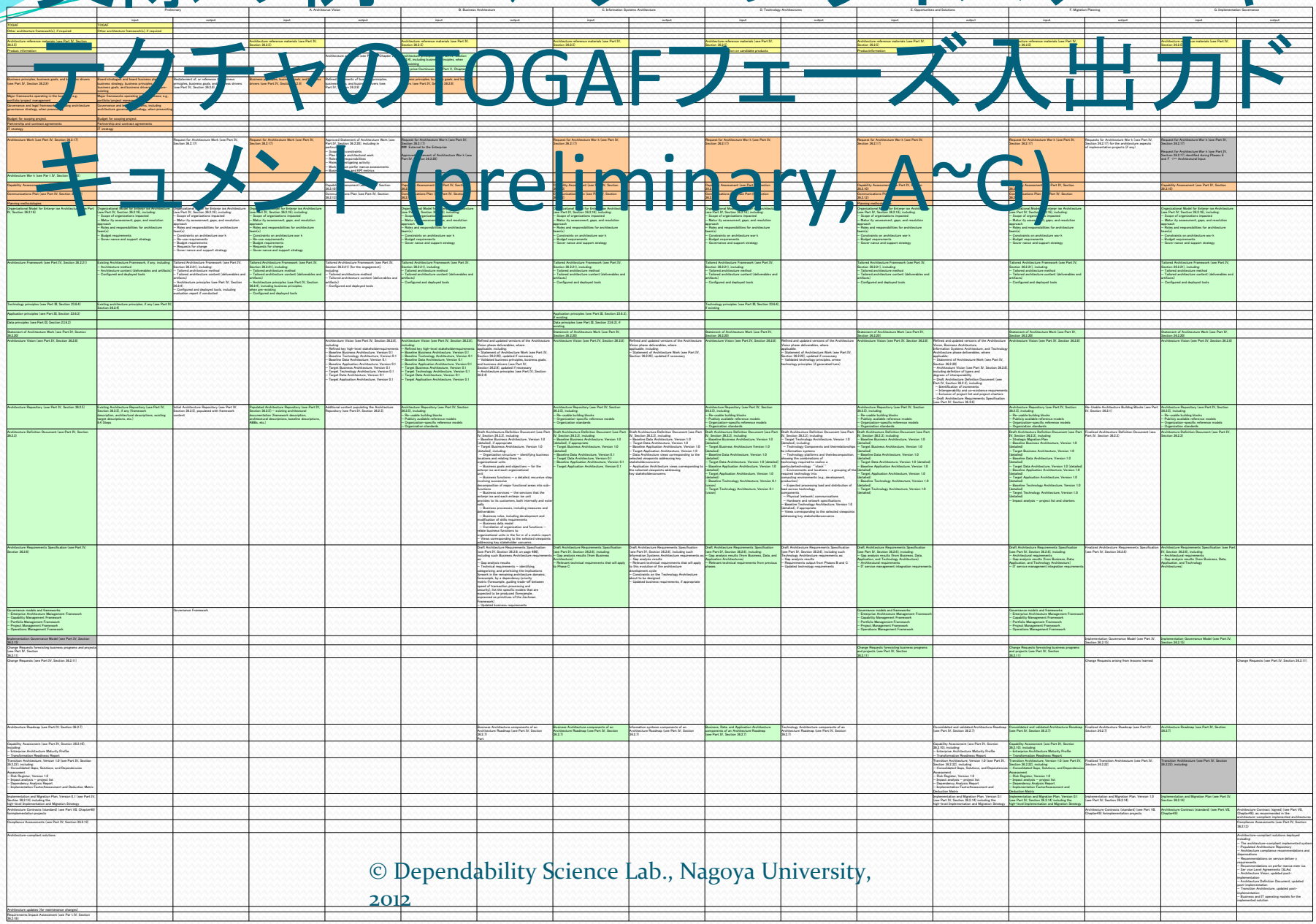
	Input	Output
Requirement Def.	User Interview Doc	Req. Def.
Architecture Design	Req. Def.	Arch. Spec Doc, Operational Doc
Implementation	Arch. Spec Doc	Program Code
Test	Program Code	Test Result
Operation and Change Management	Arch. Spec Doc, Operational Doc	Operational Logs

各フェーズの入出力ドキュメント

© Dependability Science Lab., Nagoya University, 2012

名古屋大学

実際の例:エンタープライズアーキテクチャのTOGAFフェーズ入出力ドキュメント(preliminary, A~G)



© Dependability Science Lab., Nagoya University, 2012

ステップ2.ドキュメントを整理する

- 入出力ドキュメントがどのようにディペンダビリティに関係するか、整理する
  1. 規格: ISO 12207, ISO 26262, …
  2. リスク分析結果: Hazard Analysis, FMEA results,…
  3. ディペンダビリティ要求: 99.999% Availability,…
  4. システムライフサイクルドキュメント: Lifecycle definition, …
  5. システム構造モデル: Archimate diagram,…
  6. 運用情報: Operational manual,…
  7. 環境情報: Place, Weather,…
  8. テスト、検証結果: Model checking, …
  9. Programコード: D-Script, …

© Dependability Science Lab., Nagoya University, 2012

ステップ 3.トップゴールを置く: “システムはディペンダブルである”

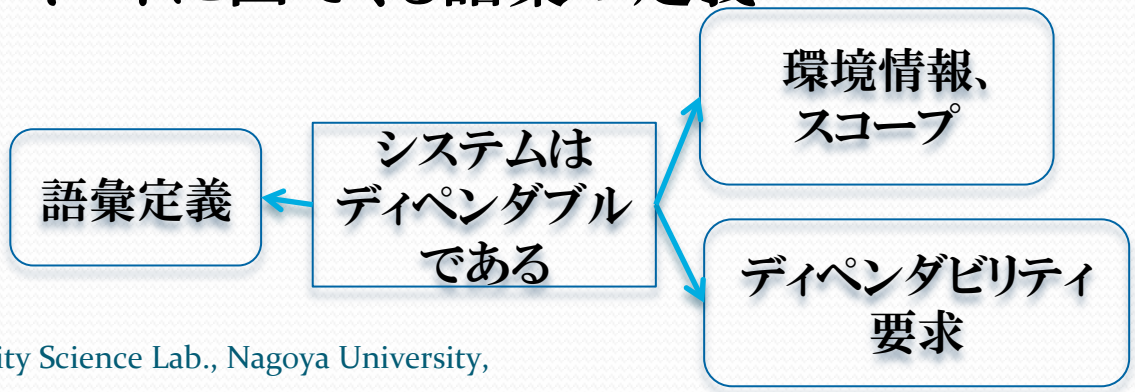
- “システムはディペンダブルである” は仮置きのトップゴールである
- 必要に応じて、変更、詳細にする。例えば
  - System X is acceptably safe
  - All identified failures are acceptably mitigated in System Y
- 変更、詳細化が現時点で難しいならば、とりあえず“システムはディペンダブルである”と置き、後で変更、詳細化する

システムは
ディペンダブルである

© Dependability Science Lab., Nagoya University, 2012

ステップ4. ディペンダビリティ要求、環境情報、語彙定義をコンテキストとしてトップゴールにおく

- トップゴールに付けられたコンテキストは、システムのディペンダビリティを議論する上での前提である
  - ディペンダビリティ要求
  - 環境情報: いつ、どこで、どのような条件下でシステムは運用されるのか. *特に, システムのスコープ(範囲)を明確にすることが重要である*
  - トップゴール、コンテキストに出てくる語彙の定義
  - その他必要な情報



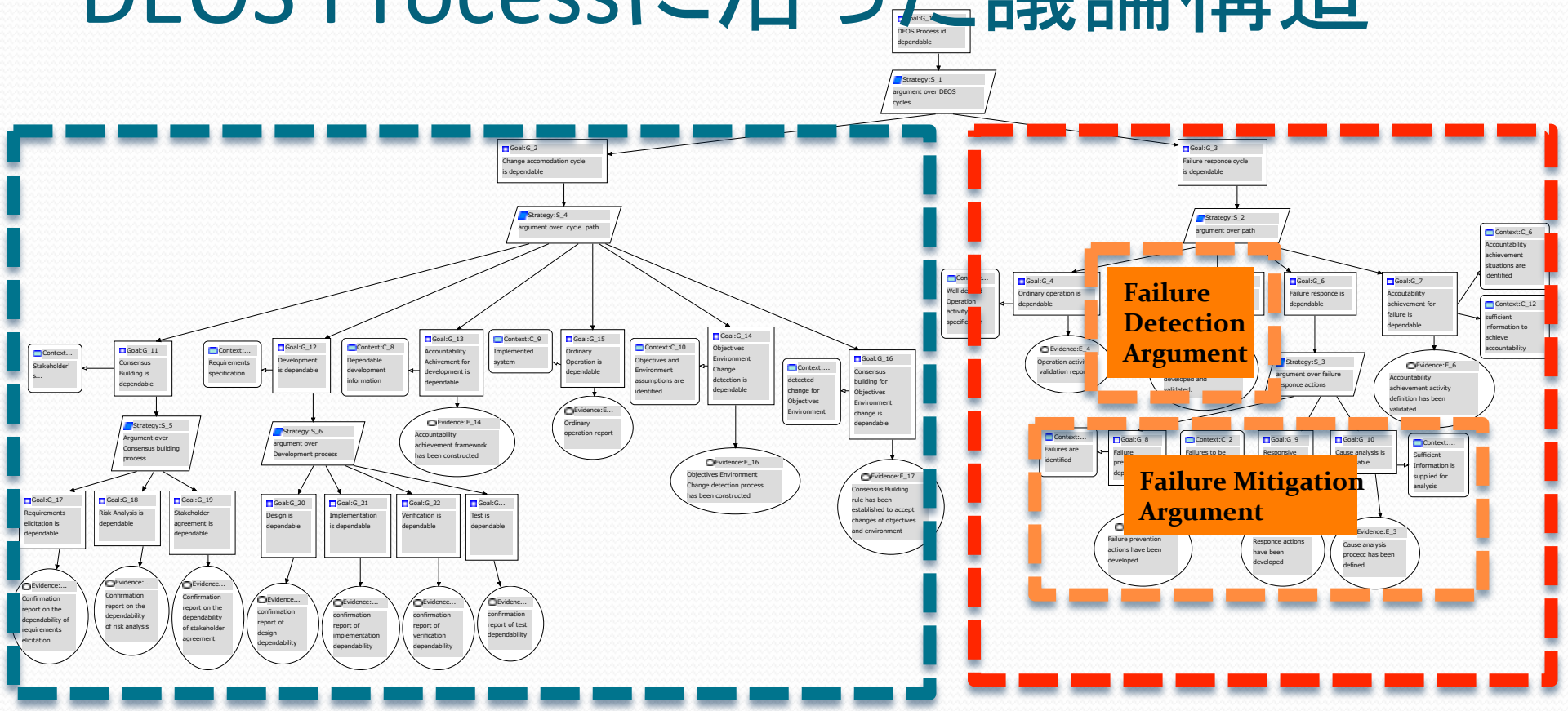
© Dependability Science Lab., Nagoya University, 2012

ステップ5.大まかにD-Caseの構造を考える

- 詳細に入る前に、おおまかな構造を考えたほうがよい
- いくつかの典型的な議論構造がある
  - ライフサイクルに沿った構造
  - システムの機能に沿った構造
  - システムの構造に沿った構造
  - ワークフローに沿った構造
  - 障害、リスク低減に沿った構造
  - *DEOS Processに沿った構造*
  - およびそれらの組み合わせ

© Dependability Science Lab., Nagoya University, 2012

DEOS Processに沿った議論構造



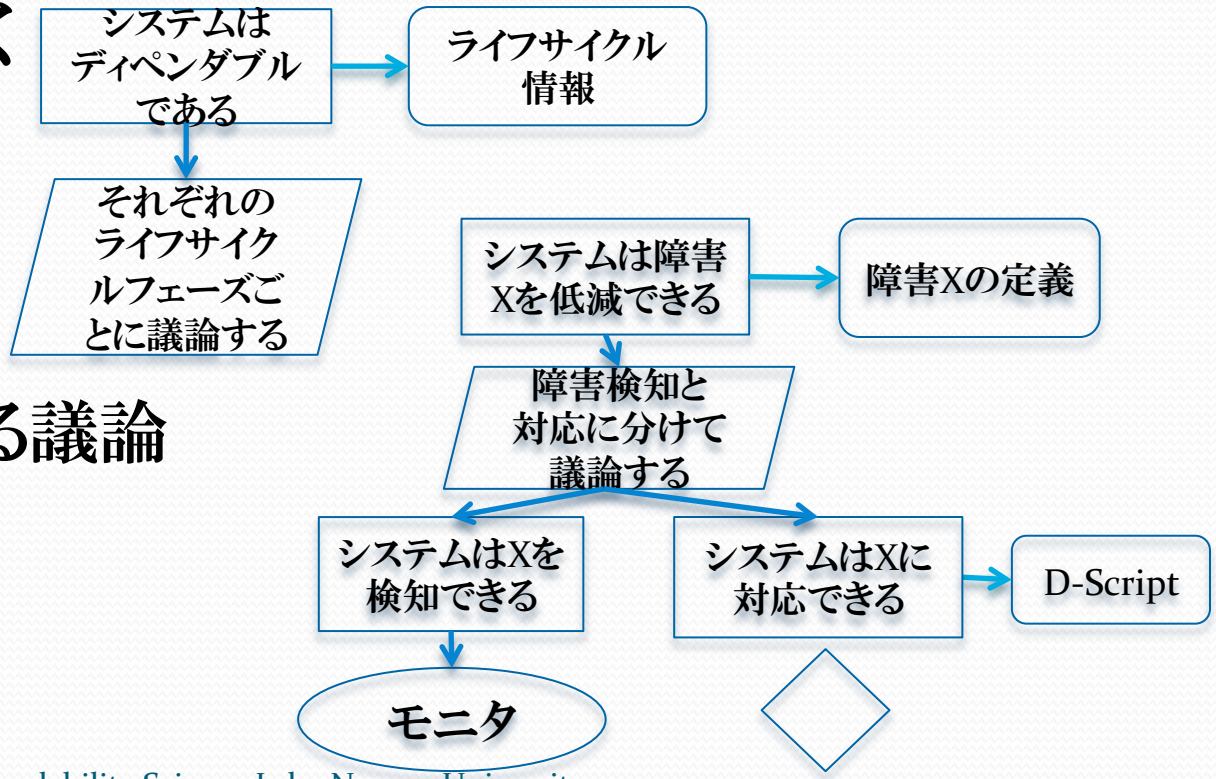
Change Accommodation Cycle Argument

Failure Response Cycle Argument

© Dependability Science Lab., Nagoya University, 2012

ステップ6.必要なドキュメントをコンテキストとして置く

- もしストラテジが“それぞれのライフサイクルフェーズごとに議論する”だったら、ライフサイクルに関するドキュメントをコンテキストに置く



- 障害低減に関する議論には障害の定義、D-Scriptを置く

© Dependability Science Lab., Nagoya University, 2012

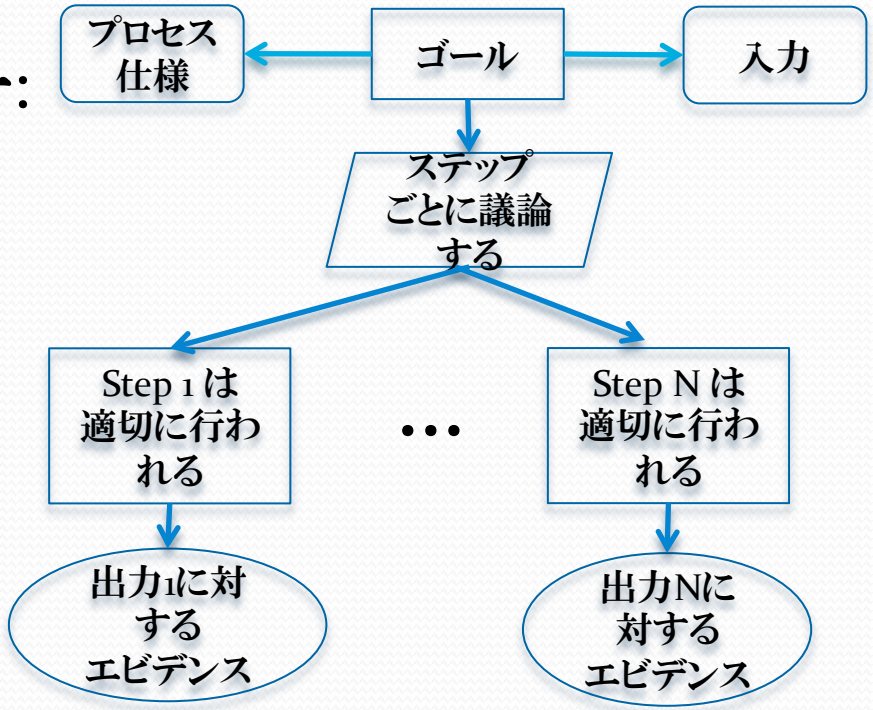
ステップ 7.ドキュメントからD-Caseのサブツリーを作る

- ドキュメントの詳細を議論する必要がある場合、そのドキュメントをもとにサブツリーを作る
  - 多くのドキュメントは（半）自動的にD-Caseに変換できる
- 例
  - プロセス
  - ディペンダビリティ属性
  - システム境界問題
  - D-Script

© Dependability Science Lab., Nagoya University, 2012

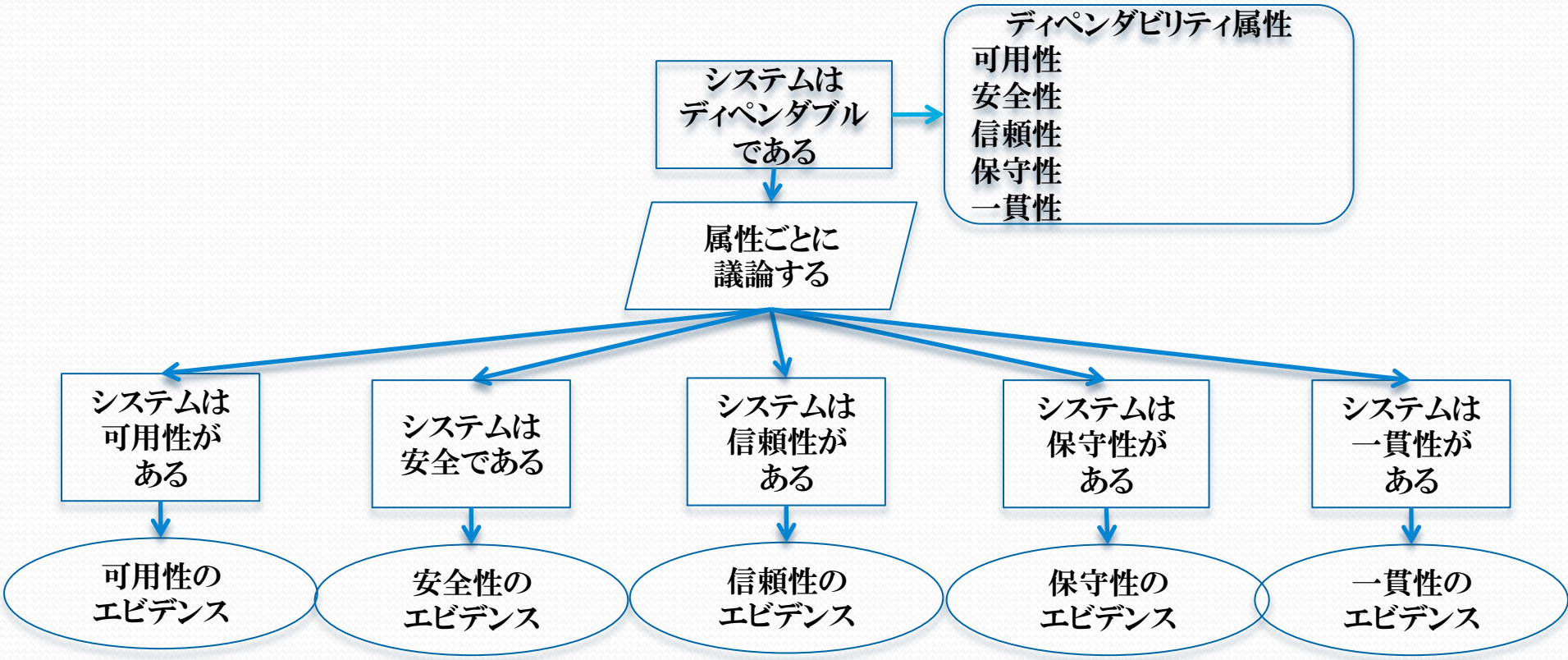
プロセスからのD-Case生成

- プロセスはゴール(目的), 1 ~ N ステップ, 入力, および出力をもつ
- 対応するD-Caseサブツリー:



© Dependability Science Lab., Nagoya University, 2012

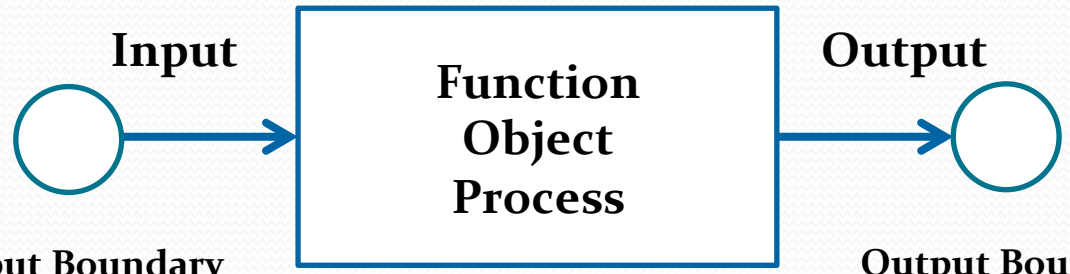
ディペンダビリティ属性からのD-Case生成



© Dependability Science Lab., Nagoya University, 2012

名古屋大学

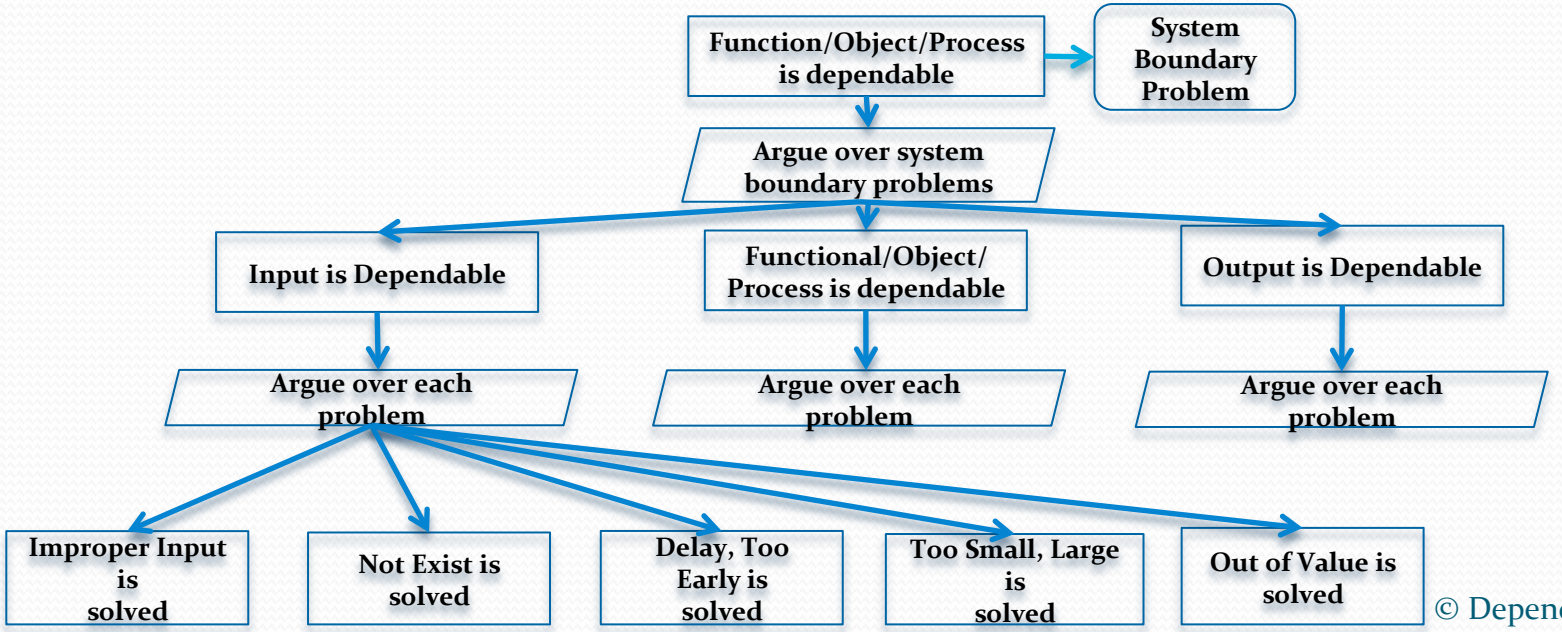
システム境界問題からのD-Case生成



Input Boundary Problems
Improper Input
Not Exist
Delay, Too Early
Too Small, Large
Out of Value

Functional Boundary Problems
Improper Function
Not execute
Not start
Not finish

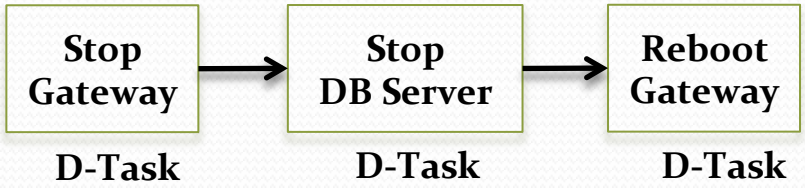
Output Boundary Problems
Improper Output
Not Exist
Delay, Too Early
Too Small, Large
Out of Value



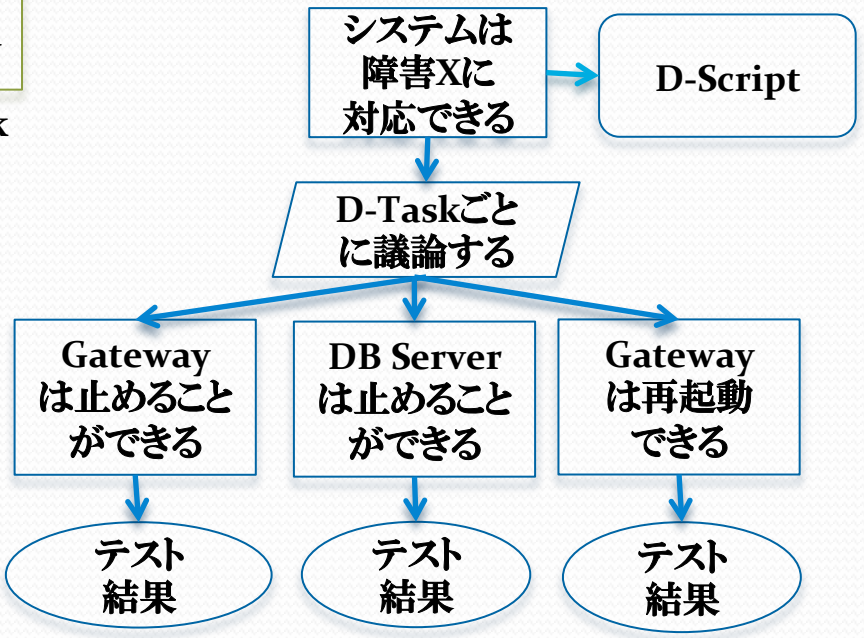
© Dependability Science Lab., Nagoya University, 2012

D-ScriptからのD-Case生成

- D-ScriptはD-Taskを単位として構成される
- 例: ウエブサーバシステムでの障害Xに対応する障害対応D-Script



- 対応するD-Case:



© Dependability Science Lab., Nagoya University, 2012

ステップ 8.サブツリーができていない部分を作る

- ドキュメントから生成できない部分は、自分でサブツリーを作る
  - できるだけ既存の典型的な議論構造を用いる
    [2] T. Kelly. “Arguing Safety, a Systematic Approach to Managing Safety Cases”. PhD Thesis, Department of Computer Science, University of York, 1998
    [13]R. Bloomfield and P. Bishop, Safety and assurance cases: Past, present and possible future – an Adelard perspective, in Proc. 18th Safety-Critical Sys. Symp., Feb. 2010.
  - 独自の議論展開は、他の人にはわかりずらいことが多い

Table 1. Formal Claim Decompositions (to be published)

Main types – keywords	Comment
architecture	splitting a component into several sub-components
functional	splitting a component into several sub-functions
set of attributes	splitting a property into several attributes
infinite set	inductive partitioning from a base case (e.g., over time)
complete	capturing the full set of values for risks, requirements, etc.
monotonic	the new system only improves on the old system
concretion	making informal statements less vague

[13]より

© Dependability Science Lab., Nagoya University, 2012

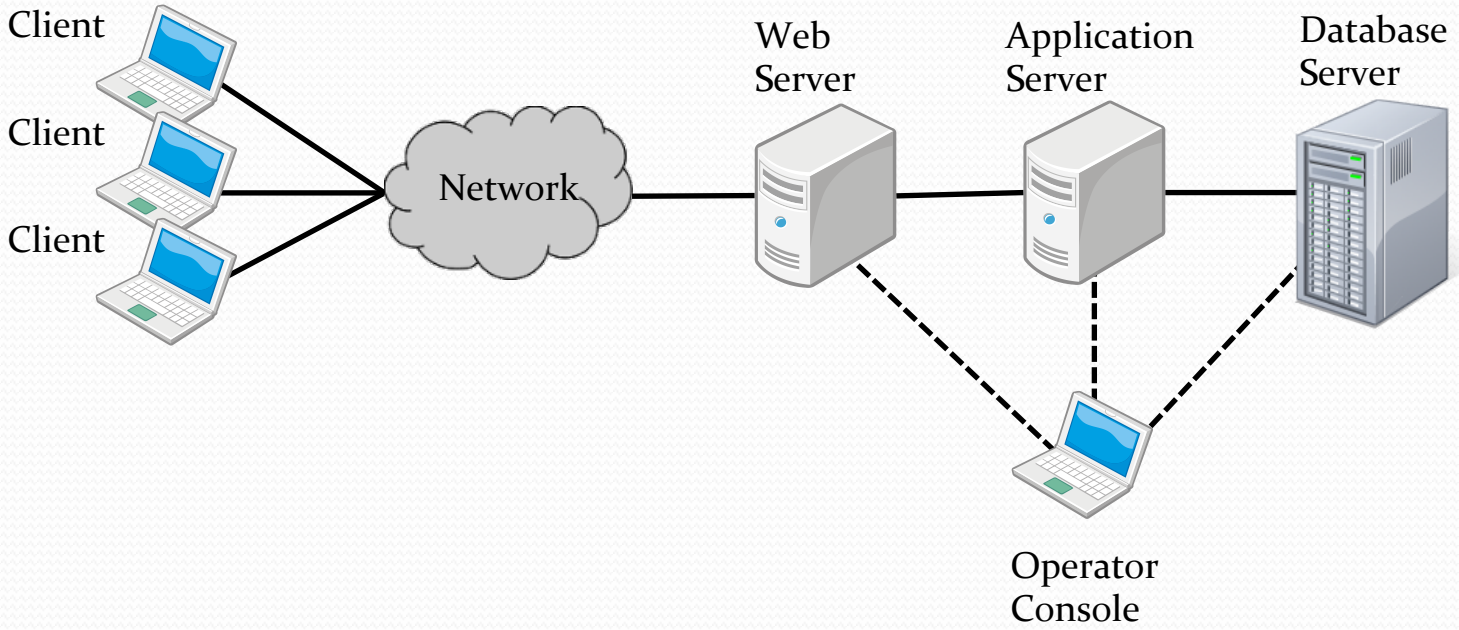
ステップ9.
上記を必要なだけ繰り返す

- 完全なD-Caseは存在しません。常によりよいD-Caseを目指しましょう

© Dependability Science Lab., Nagoya University, 2012

記述例: ウエブサーバシステム

- ウエブサーバシステムのD-Caseを書いてみよう



© Dependability Science Lab., Nagoya University, 2012

ステップ1. システムライフサイクルを整理し、それぞれのフェーズの入力、出力ドキュメントをまとめる

- ウエブサーバは既存のサーバを組み合わせて作られたとする

- 入出力ドキュメントは下のようであったとする

	Input	Output
Requirement Def.	User Interview Doc	Req. Def., SLA Doc
Architecture Design	Req. Def., SLA Doc	Arch. Spec Doc, Workflow Definition, Risk Analysis Doc
Integration	Arch. Spec Doc, Workflow Definition, Risk Analysis Doc, Server Spec	Integrated code and system
Test	Integrated code and system	Test Result
Operation and Change Management	Workflow Definition	Operational Logs, System Logs

© Dependability Science Lab., Nagoya University, 2012

ステップ2. ドキュメントを分類する

- このシステムには以下のドキュメントがあったとする
  1. 規格:
  2. リスク分析結果: Risk Analysis Doc
  3. ディペンダビリティ要求: Req. Def., SLA Doc
  4. システムライフサイクル文書: Lifecycle definition
  5. システム構造モデル: Arch, Server, D-RE Spec
  6. 運用情報: Workflow Definition
  7. 環境情報:
  8. テスト、検証結果: Test Result
  9. Programコード: Integrated Code, D-Script

© Dependability Science Lab., Nagoya University, 2012

ステップ 3.トップゴールを置く: “システムはディペンダブルである”

- **このシステムでは、サービスの継続性に重点を置いた:** “The System Can continue CD Online Sales Service as much as possible”

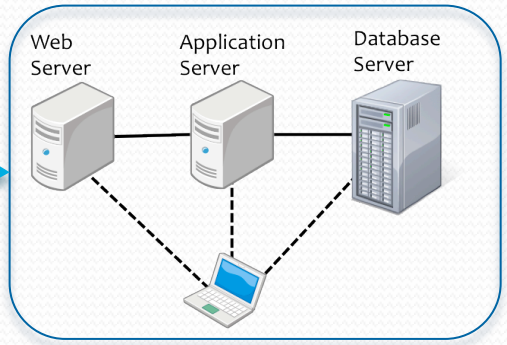
The System can continue CD online Sales Service as much as possible

© Dependability Science Lab., Nagoya University, 2012

ステップ4. ディペンダビリティ要求、環境情報、語彙定義をコンテキストとしてトップゴールにおく

SLA Doc

The System can continue CD online Sales Service as much as possible



Scope

サービスの継続性
に関するゴールなので、
SLA(Service Level Agreement)
文書をコンテキストに置く

© Dependability Science Lab., Nagoya University, 2012

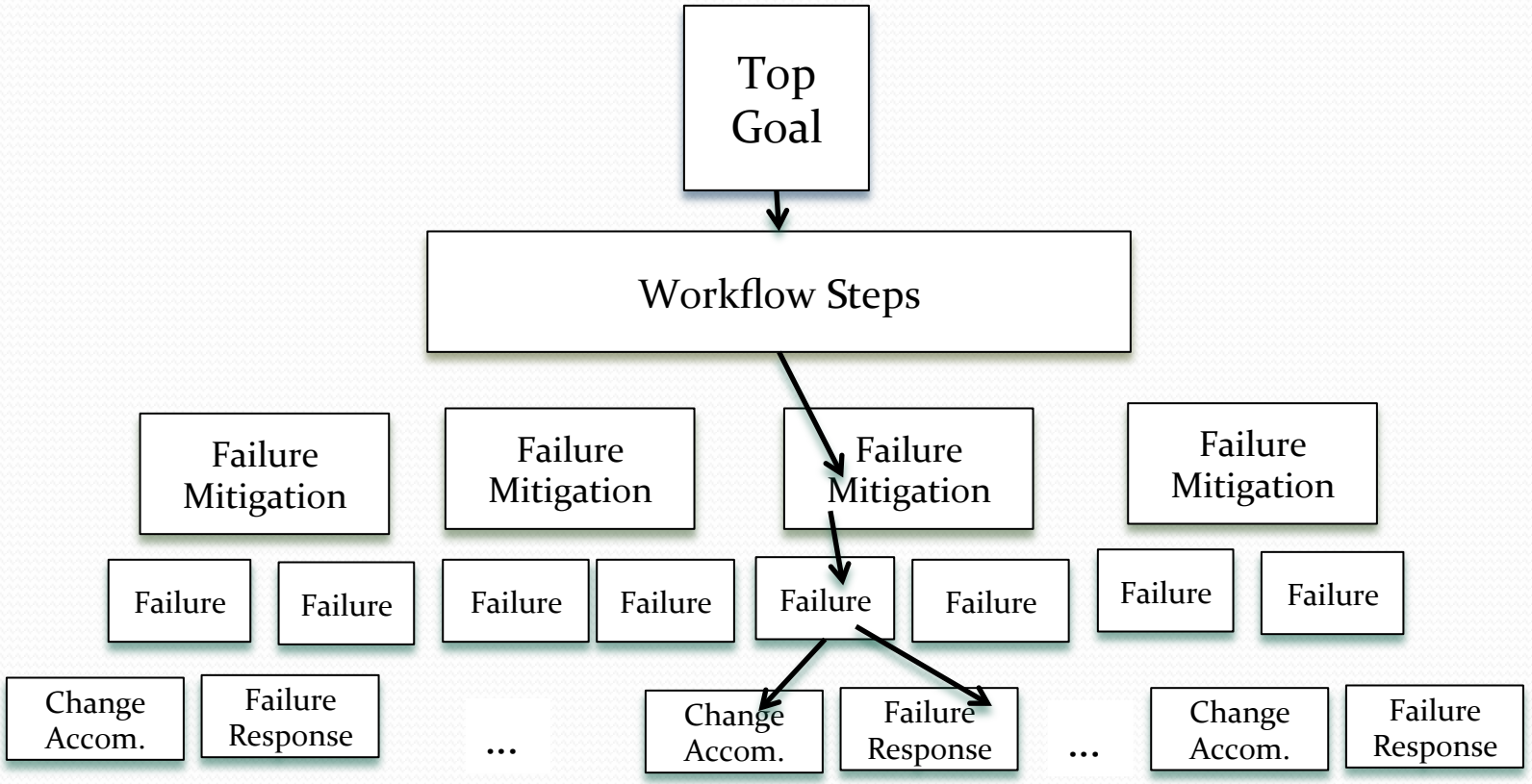
ステップ5.大まかにD-Caseの構造を考える

- ステークホルダの議論が以下であったとする
  - サーバの技術的詳細は分からなかった
    - サーバ技術は十分成熟している
  - 最近のサーバ障害は運用ミスによって起こっている
    - 2012年6月20日に深刻なサーバ障害があった
  - 運用ワークフローを注意深く議論する必要がある
  - その上で障害対応、変化対応を議論した方がいい

運用ワークフローに沿って、
障害・変化対応を議論しよう

© Dependability Science Lab., Nagoya University, 2012

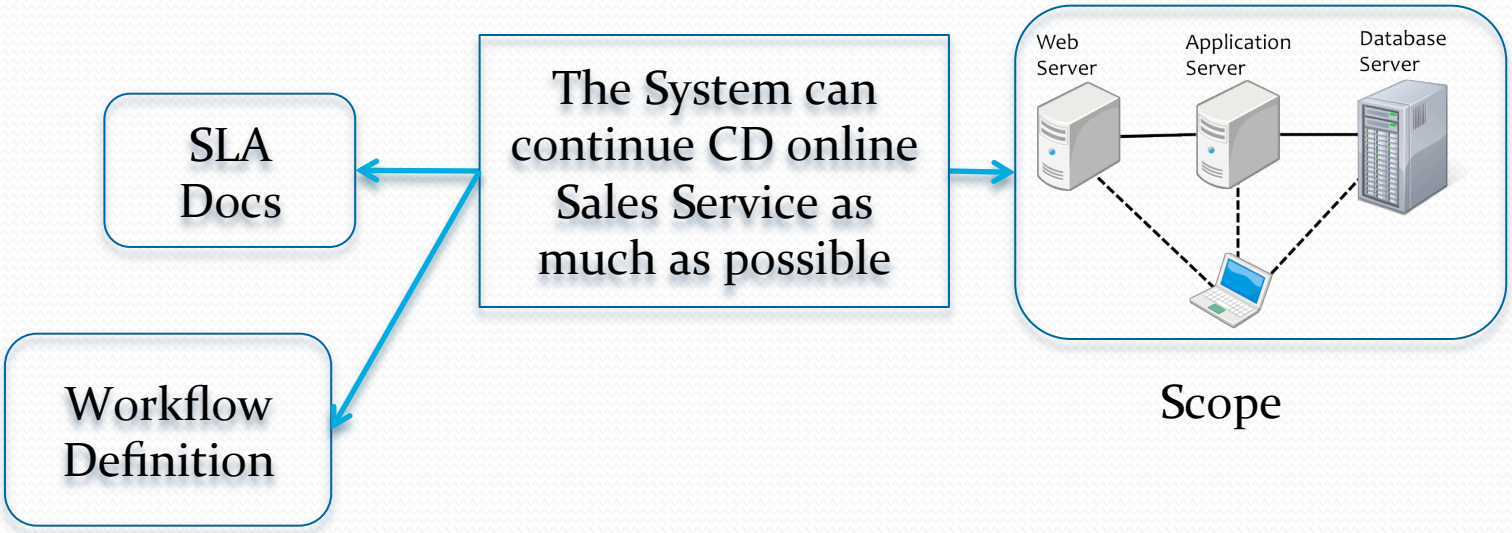
大まかな議論構造



© Dependability Science Lab., Nagoya University, 2012

ステップ6.必要なドキュメントをコンテキストとして置く

- 運用ワークフローに沿った議論をするために、運用ワークフローの定義ドキュメントを置く

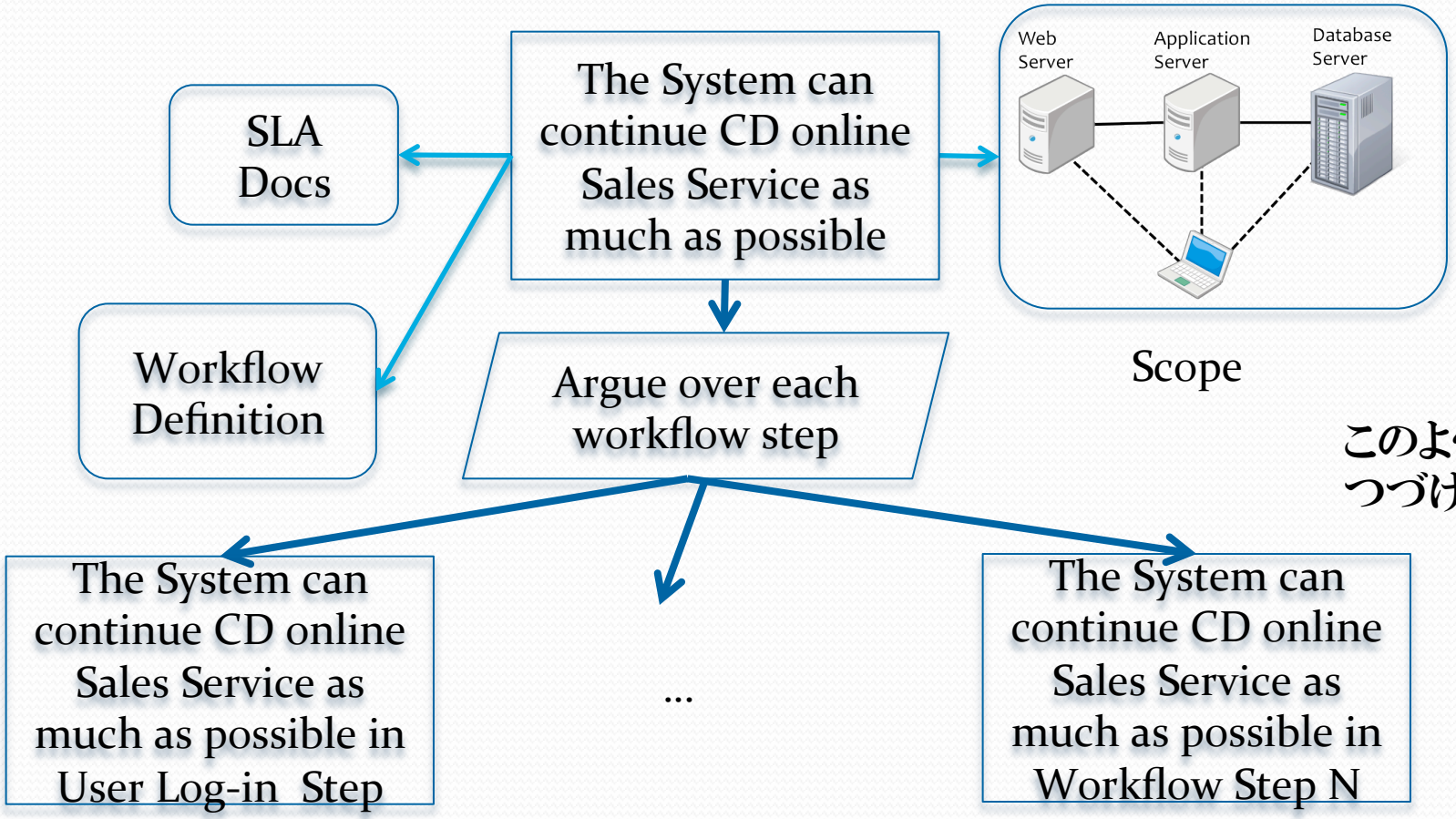


© Dependability Science Lab., Nagoya University, 2012

ステップ 7.ドキュメントからD-Caseのサブツリーを作る

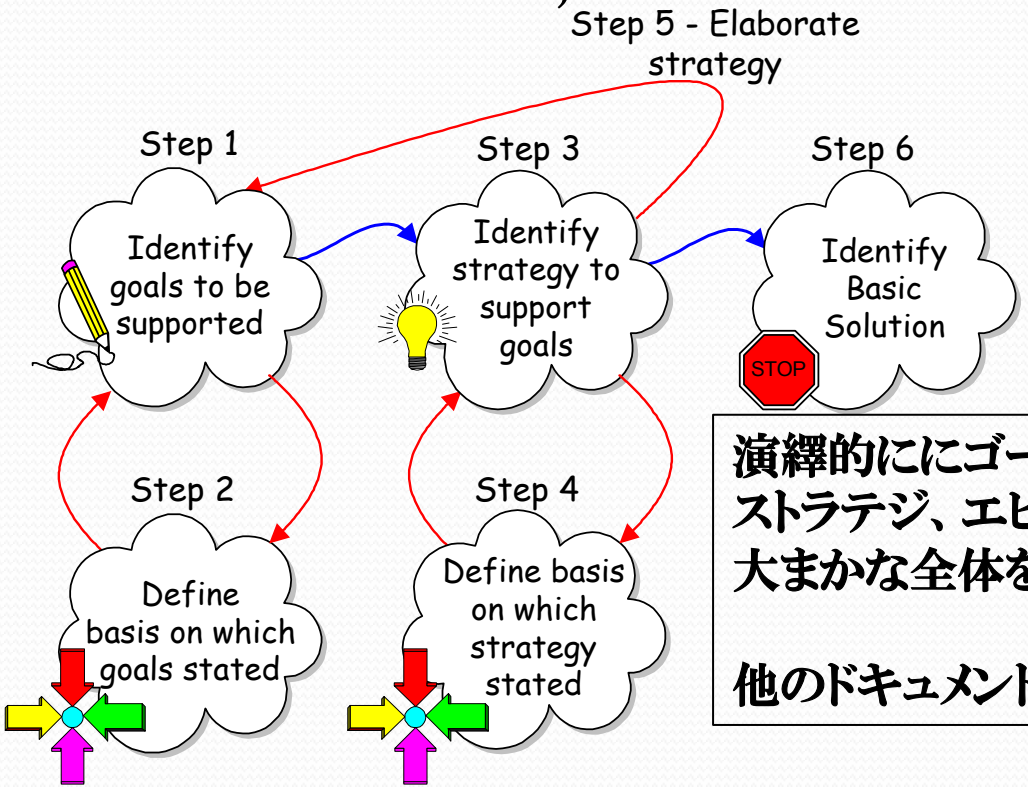
© Dependability Science Lab., Nagoya University, 2012

- 運用ワークフロー定義に従ってサブゴールに分割する



既存のガイドライン

- [17] GSN Community Standard Version 1, Origin Consulting (York) Limited, on behalf of the Contributors, Nov. 2011



演繹的ににゴール、
ストラテジ、エビデンスをステップごとに設定していく
大まかな全体を把握することは明示的に書かれていない

他のドキュメントとの関連が明示的に書かれていない

© Dependability Science Lab., Nagoya University, 2012

既存のガイドライン: ASCE Manual by Adelard [21]

- Safety Case構築に関するガイドブック
- Safety Caseへの要求と、その要素をリストアップ
- 1998年に出版され、2006年に少し改訂された。現時点でも有用なガイドブックと考えるが、最新の情報へのアップデートが望まれる

Adelard Safety Case Development Manual

Preliminary
Architectural
Implementation
Operation and Installation

Elements of Safety Cases

© Dependability Science Lab., Nagoya University, 2012

© Dependability Science Lab., Nagoya University, 2012

他のガイドライン、関連ガイドライン

- [10] European Organisation for the Safety of Air Navigation, Safety Case Development Manual, 2nd ed., EUROCONTROL, Oct. 2006
- [24] Software Assurance: A Curriculum Guide to the Common Body of Knowledge to Produce, Acquire and Sustain Secure Software, Software Assurance Workforce Education and Training Working Group, Oct 2007, Homeland Security
- [25] National Defense Industrial Association (NDIA) System Assurance Committee, Engineering For System Assurance, 2008, version 1.0

GSNの紹介、いくつかのパターンが紹介されているが、記述法に関する説明は少ない

Assurance Caseの重要性が指摘されているが、記述法に関する説明は少ない

まとめ: D-Case10箇条

1. ゴールは命題の形をしていなくてはいけない
2. コンテキストは必要な場所のみにできるだけおく
3. トップゴールに、システムのスコープをコンテキストとしておく
4. ライフサイクル、各フェーズの入出力ドキュメントを整理する
5. D-Caseの構造を大まかに把握する
6. 典型的な議論構造を用いる
7. 他のシステムとの関係を考えながら記述する
8. 常に、予期できないこと、開放性を考えながら記述する
9. 完全なD-Caseは存在しない。常によりよいD-Caseを目指す
10. ゴールは肯定形か否定形か、どちらかに決めよう(この1条は仮置き)

© Dependability Science Lab., Nagoya University, 2012

D-Caseツール

- D-Case Editor
  - http://www.dependable-os.net/tech/D-CaseEditor
  - 1年で約200ダウンロード
  - ソースコードも提供可能
- システム監視機能との連携機能
  - http://web.sfc.keio.ac.jp/~jin/dm/
- D-Case/Agda
  - http://agda.cvs.gr.jp/agda/D-Case-Agda/
- DS-Bench/TEST_ENV Tool
  - http://www.dependable-os.net/tech/DSBenchDCloud/index.html
  - ベンチマークツールとの連携機能
  - DSN 2012で発表

© Dependability Science Lab., Nagoya University, 2012

参考文献 1/7

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, 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

© Dependability Science Lab., Nagoya University, 2012

参考文献 2/7

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. G. Despotou, T. Kelly. Using Scenarios to Identify and Trade-off Dependability Objectives in Design. Proceedings of the 23rd International System Safety Conference (ISSC), 2005

© Dependability Science Lab., Nagoya University, 2012

参考文献 3/7

8. T. Kelly. Using software architecture techniques to support the modular certification of safety-critical systems. In Proceedings of the 11th Australian Workshop on Safety-Related Programmable Systems, August 2005.
9. G. Despotou, T. Kelly, EXTENDING SAFETY DEVIATION ANALYSIS TECHNIQUES TO ELICIT FLEXIBLE DEPENDABILITY REQUIREMENTS, In proceedings of the 1st IET (Former IEE) International Conference on System Safety Engineering (ICSS), London, UK, June 2006.
10. European Organisation for the Safety of Air Navigation, Safety Case Development Manual, 2nd ed., EUROCONTROL, Oct. 2006

© Dependability Science Lab., Nagoya University, 2012

参考文献 4/7

11. G. Despotou, T. Kelly, Design and Development of Dependability Case Architecture during System Development, . In proceedings of the 25th International System Safety Conference (ISSC), Baltimore, MD USA. Proceedings by the System Safety Society, 2007
12. Jackson, D. et al, Software for dependable systems– sufficient evidence?, NATIONAL RESEARCH COUNCIL, 2008
13. R. Bloomfield and P. Bishop, Safety and assurance cases: Past, present and possible future – an Adelard perspective, in Proc. 18th Safety-Critical Sys. Symp., Feb. 2010.
14. Ibrahim Habli, Ireri Ibarra, Roger Rivett, Tim Kelly, Model-Based Assurance for Justifying Automotive Functional Safety, 10AE-0181, 2010

© Dependability Science Lab., Nagoya University, 2012

参考文献 5/7

15. Oleg Sokolsky, Insup Lee, Mats Heimdahl, Challenges in the Regulatory Approval of Medical Cyber-Physical Systems, In Proceedings of the International Conference on Embedded Software (EMSOFT 2011), Taipei, Taiwan, October 2011.
16. DEOS Project White Paper, FY2011-WP-03J, www.dependable-os.net/ja/topics/file/White_Paper_V3.0J.pdf, 2011
17. GSN Community Standard Version 1, Origin Consulting(York) Limited, on behalf of the Contributors, November 2011
18. Ewen Denney and Ganesh Pai, Ibrahim Habli, Perspectives on Software Safety Case Development for Unmanned Aircraft, DSN2012

© Dependability Science Lab., Nagoya University, 2012

参考文献 6/7

19. K.Tanaka, Y.Matsuno, Y.Nakabo, S.Shirasaka, S.Nakasuka. Quality Assurance of Microsatellite with Assurance Case, Proc. Spring Conference of Reliability Engineering Association of Japan, June 2012 (In Japanese)
20. Assurance Case Frameworks Part of High Confidence Software MSR T. Scott Ankrum MITRE — Software Engineering Center, March 11, 2004
21. Adelard, www.adelard.com/asce/index.html
22. AAMI, Safety Assurance Cases for Medical Devices, http://www.aami.org/meetings/courses/safety.html
23. Dependability computing, assurance case course, http://www.dependablecomputing.com/courses.html

© Dependability Science Lab., Nagoya University, 2012

参考文献 7/7

24. Software Assurance: A Curriculum Guide to the Common Body of Knowledge to Produce, Acquire and Sustain Secure Software, Software Assurance Workforce Education and Training Working Group, Oct 2007, Homeland Security
25. National Defense Industrial Association (NDIA) System Assurance Committee, Engineering For System Assurance, 2008, version 1.0

© Dependability Science Lab., Nagoya University, 2012

名古屋大学

D-Case入門
7 D-Case演習

© Dependability Science Lab. Nagoya Univ. 2012

内容

- GSN構造規則
- 構造誤り
- GSN構造の省略
- 要素の命名規則
- GSNの作成
- 問題回答の指針

© Dependability Science Lab. Nagoya Univ. 2012

GSN構造規則

- 構成要素
- 要素間関係

© Dependability Science Lab. Nagoya Univ. 2012

GSNの基本となる7個の記号を示しなさい

名称	図式要素	説明
ゴール		システムが達成すべき性質を示す．下位ゴールや戦略に分解される
戦略		ゴールの達成を導くために必要となる論証を示す．下位ゴールや下位戦略に分解される．
コンテクスト		ゴールや戦略が必要となる理由としての外部情報を示す
未展開要素		まだ具体化できていないゴールや戦略を示す
証跡		ゴールや戦略が達成できることを示す証跡
コンテクスト関係		コンテクストとゴール，戦略との関係
ゴール関係		ゴールと戦略，ゴールと証跡の関係

© Dependability Science Lab. Nagoya Univ. 2012

問題2
GSNの基本的な関係を示しなさい

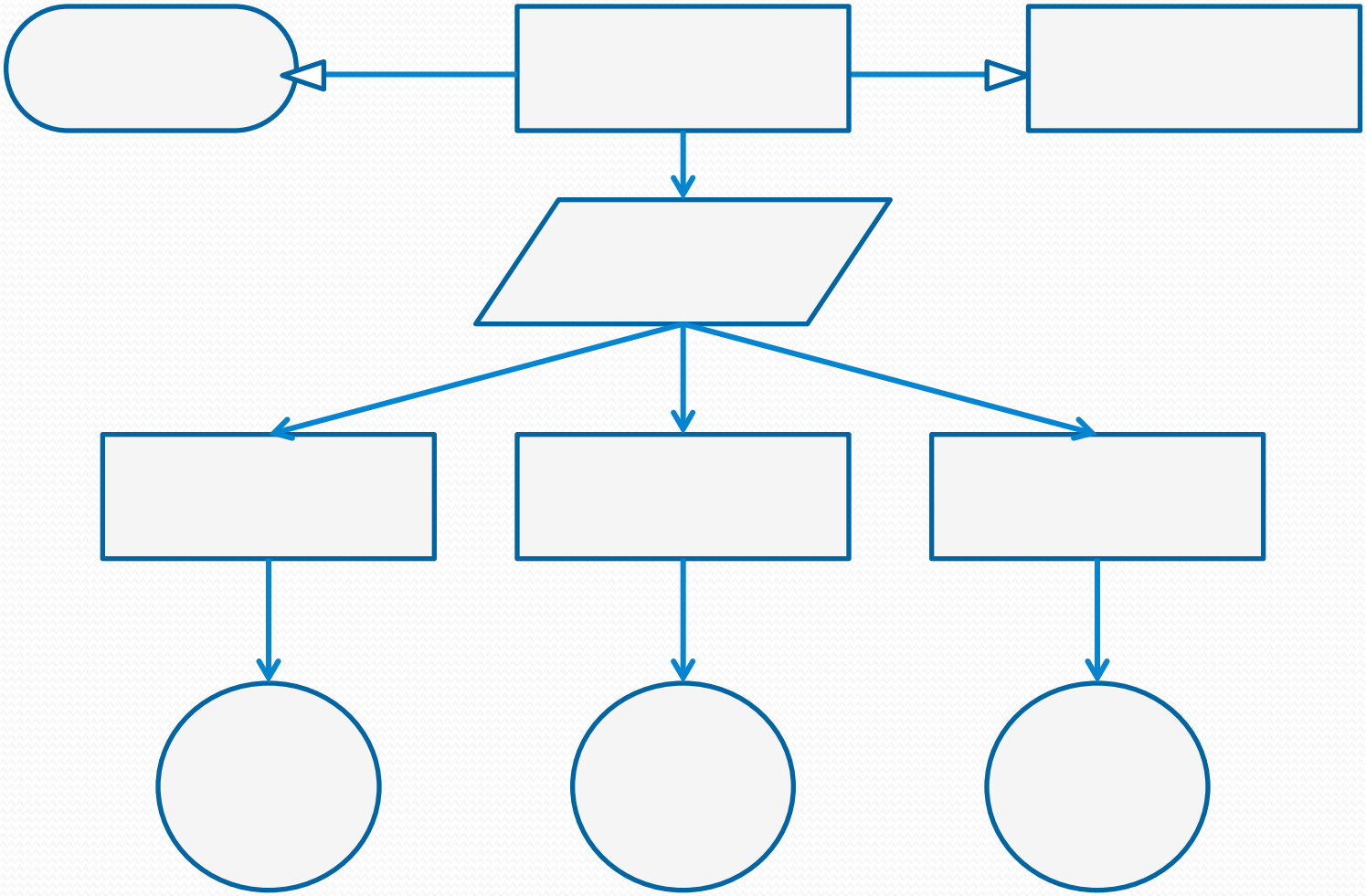
関係	図式表現	説明
ゴール・コンテクスト関係		ゴールのコンテクストを示す関係
戦略コンテクスト関係		戦略のコンテクストを示す関係
ゴール戦略関係		ゴールを戦略によって分解することを示す関係
戦略ゴール分解関係		戦略によって下位ゴールに分解されることを示す関係
ゴール証跡関係		ゴールが証跡によって達成できることを示す関係
ゴール保留関係		ゴールが保留されていることを示す
戦略保留関係		戦略が保留されていることを示す

© Dependability Science Lab. Nagoya Univ. 2012

GSN構造の誤り

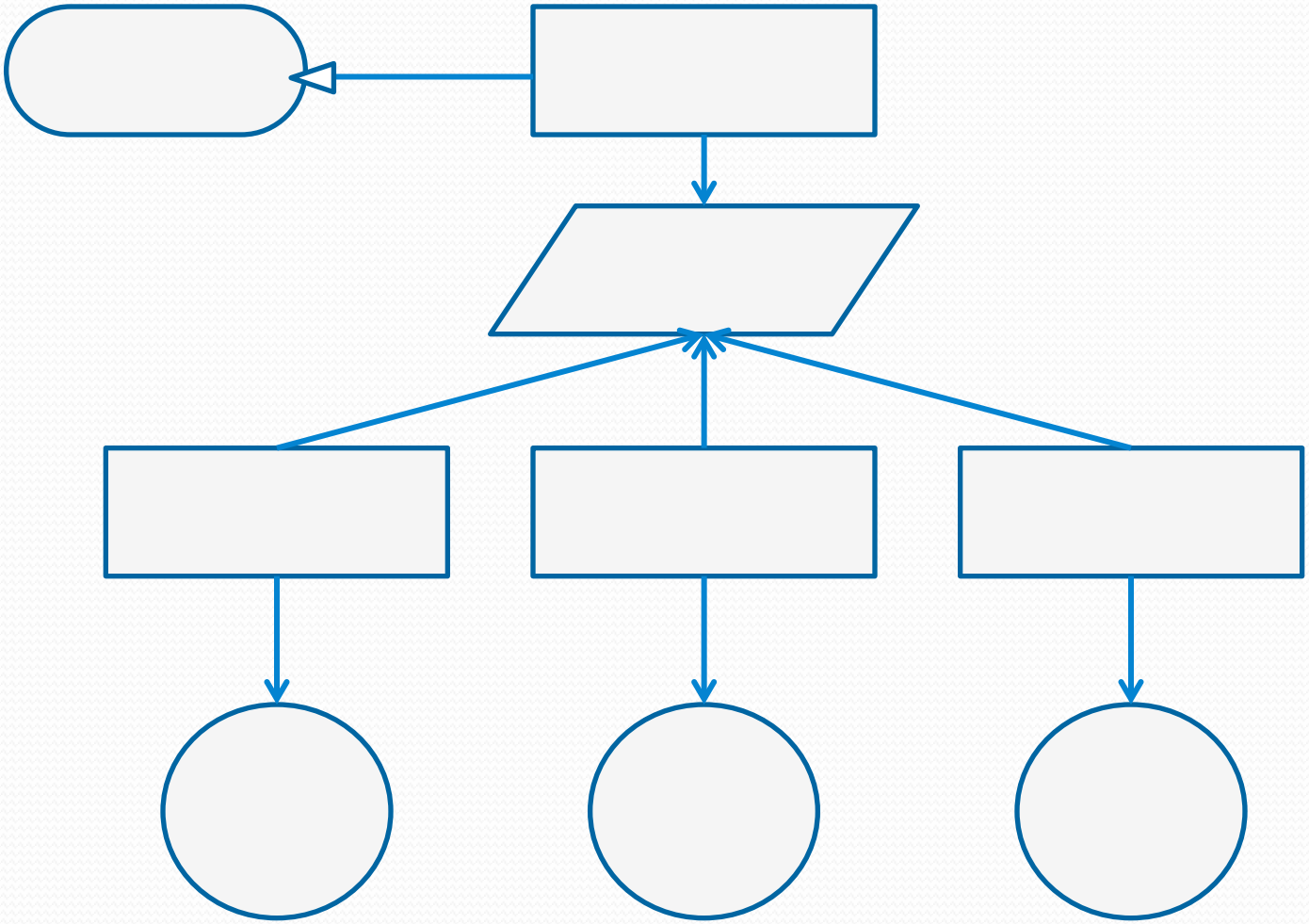
© Dependability Science Lab. Nagoya Univ. 2012

問題1
次のGSN構造の誤りを指摘しなさい



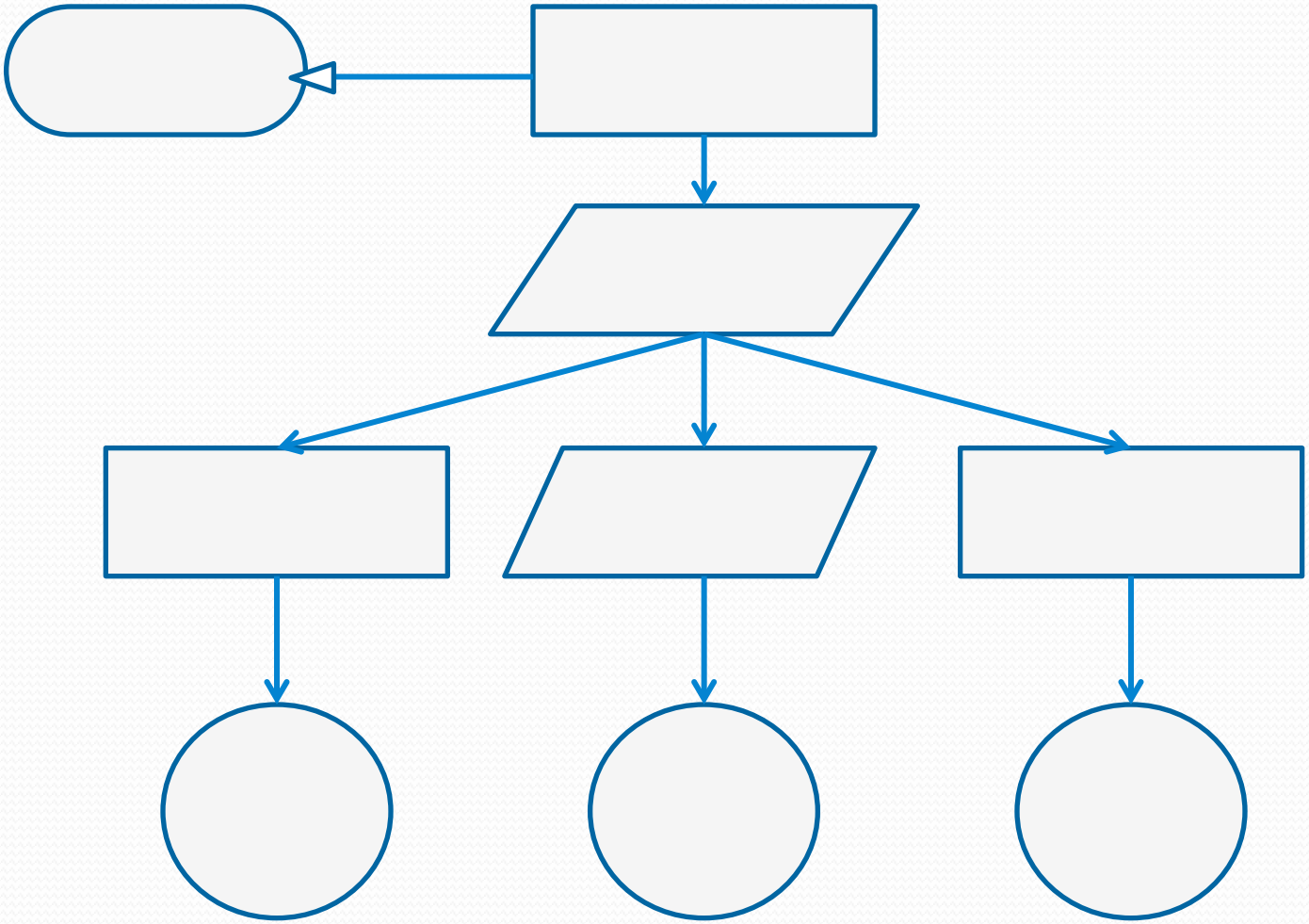
© Dependability Science Lab. Nagoya Univ. 2012

次のGSN構造の誤りを指摘しなさい



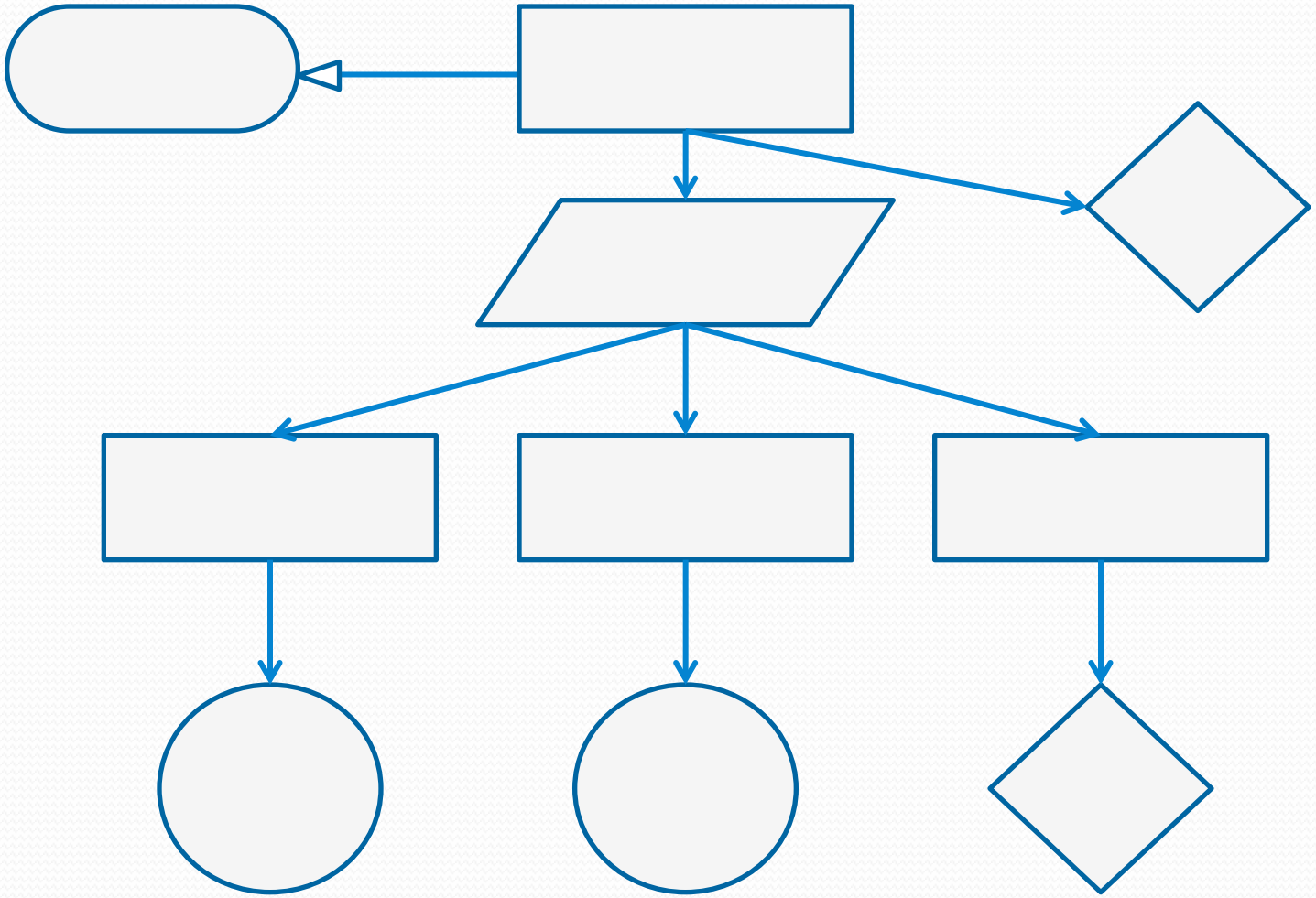
© Dependability Science Lab. Nagoya Univ. 2012

問題3
次のGSN構造の誤りを指摘しなさい



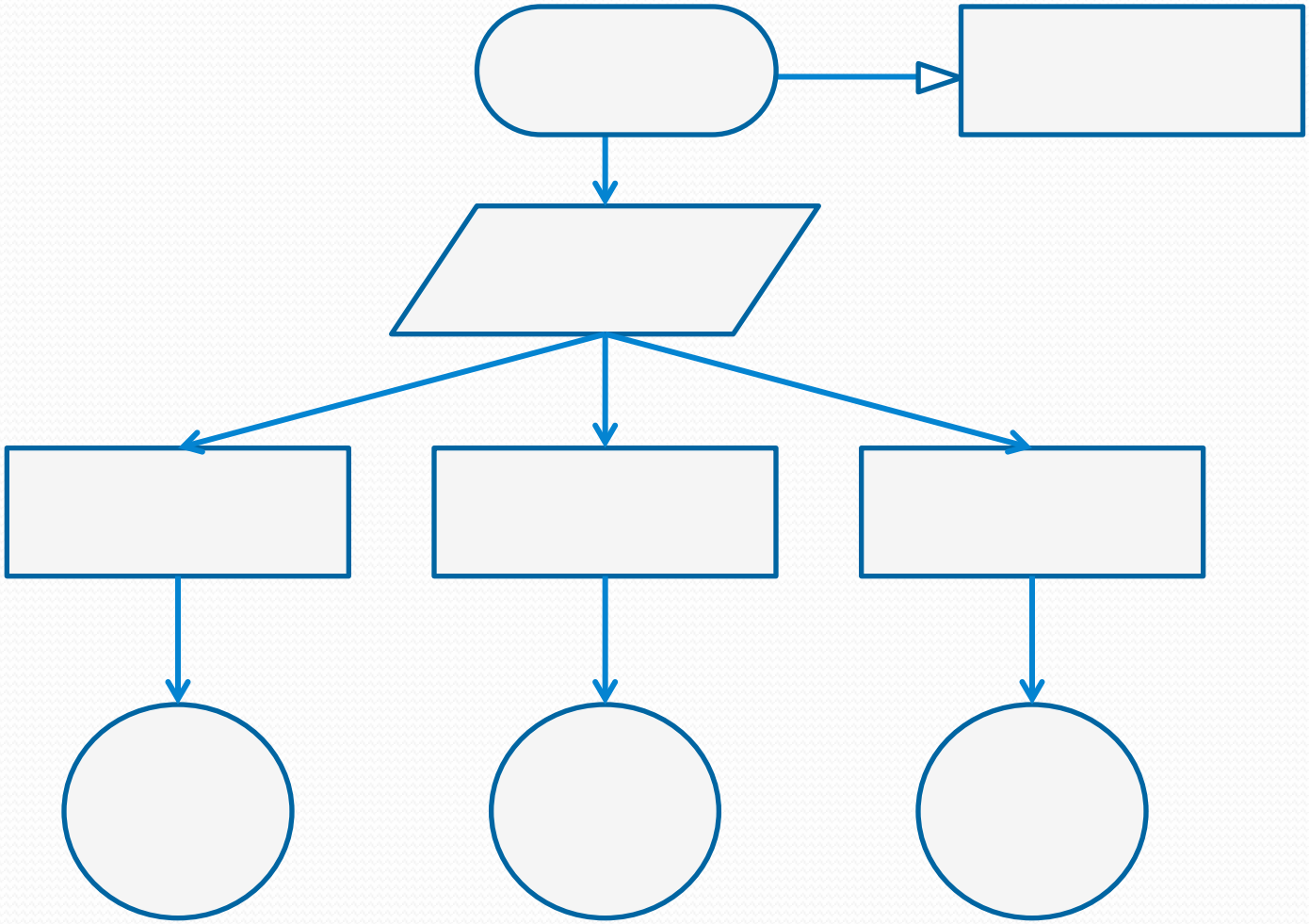
© Dependability Science Lab. Nagoya Univ. 2012

問題4
次のGSN構造の誤りを指摘しなさい



© Dependability Science Lab. Nagoya Univ. 2012

問題5
次のGSN構造の誤りを指摘しなさい



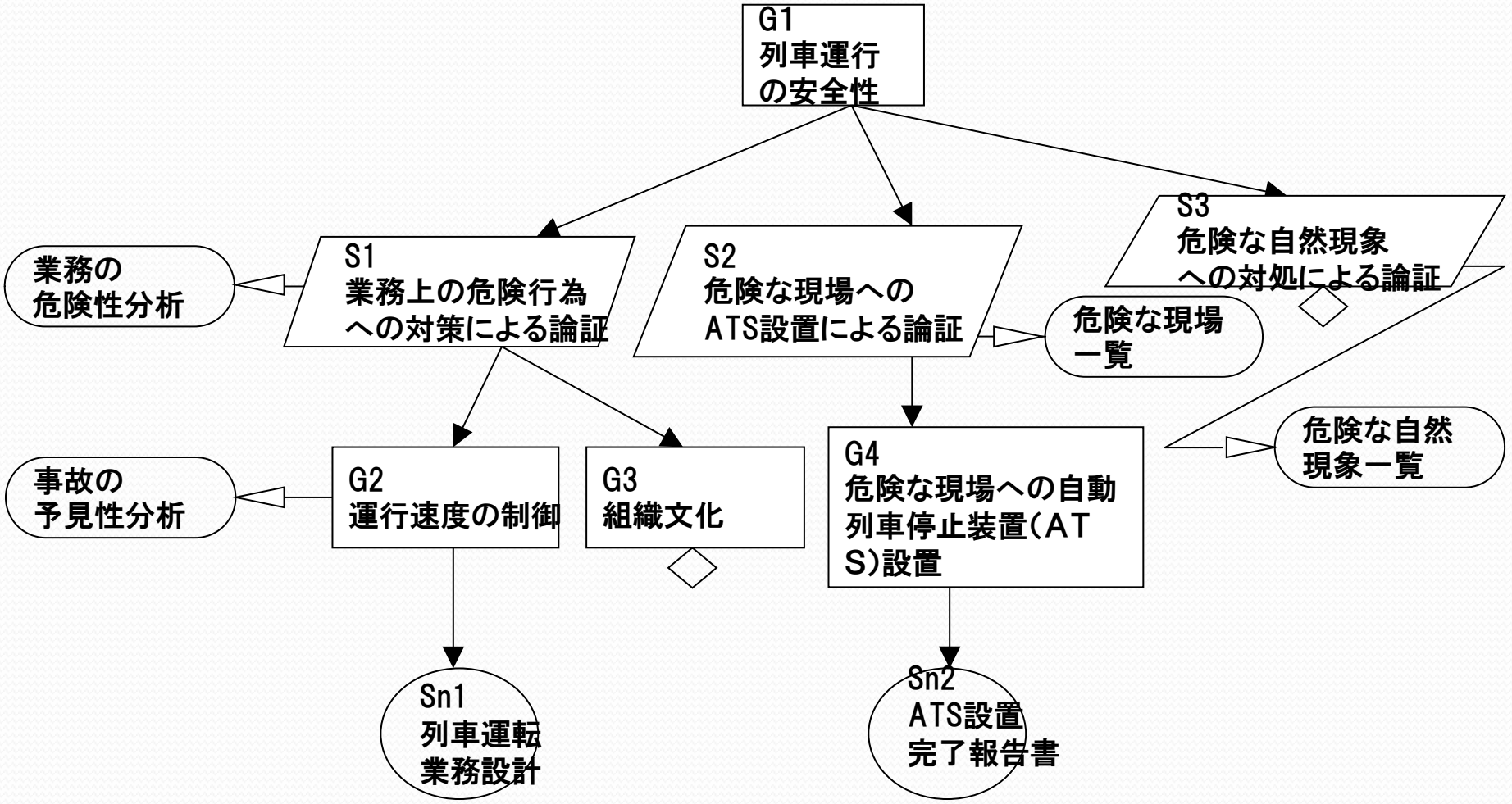
© Dependability Science Lab. Nagoya Univ. 2012

GSN構造の省略

© Dependability Science Lab. Nagoya Univ. 2012

問題1

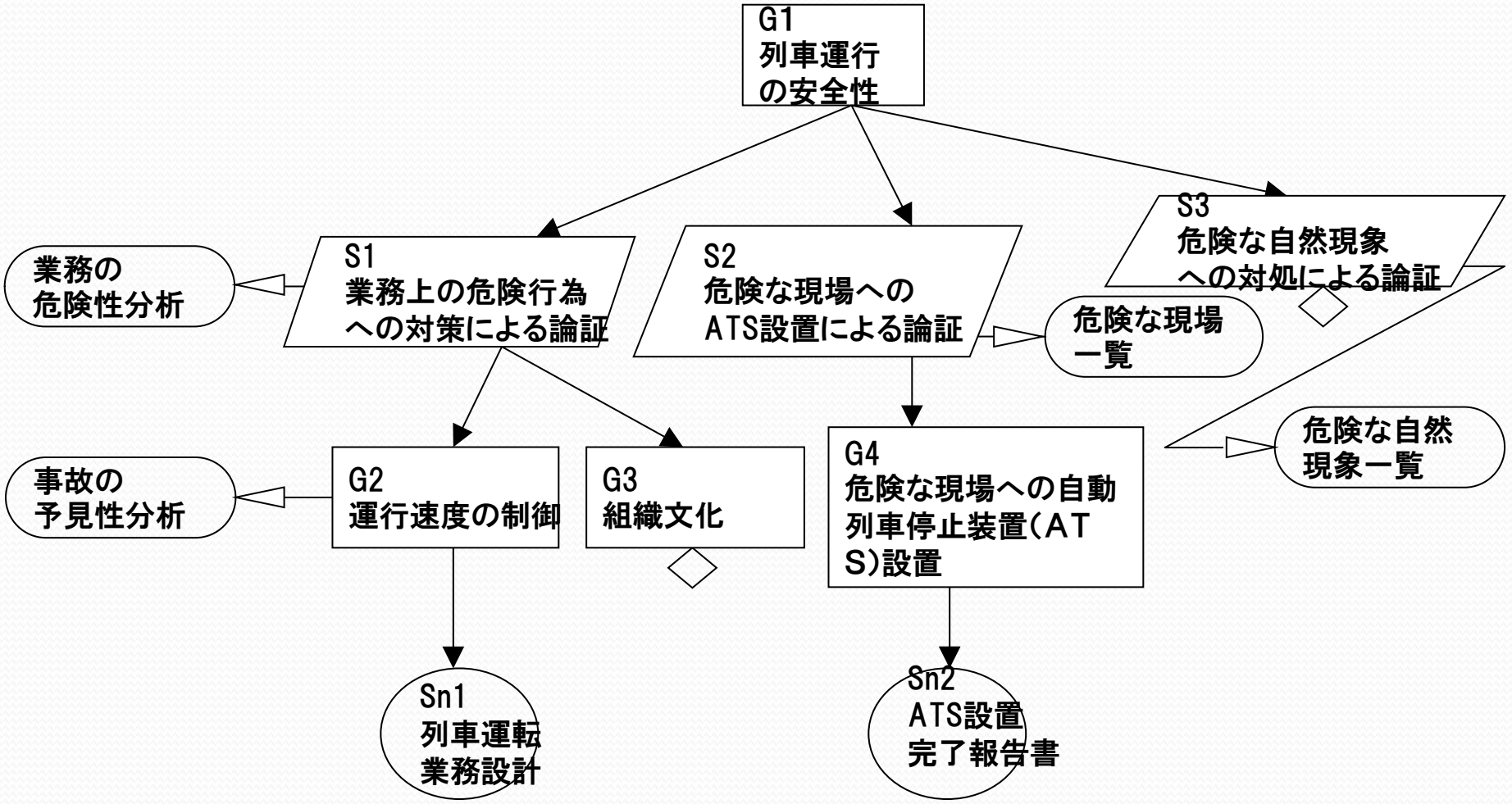
次のGSNで，論証が省略されている部分はどこか？
指摘して，省略された論証を復元しなさい．



© Dependability Science Lab. Nagoya Univ. 2012

問題2

次のGSNで省略されているスコープは何か？
省略されているスコープの例を指摘しなさい.



© Dependability Science Lab. Nagoya Univ. 2012

要素の命名規則

© Dependability Science Lab. Nagoya Univ. 2012

次の中で，正しい記述を指摘しなさい

1. コンテクストを手順だと考えて順番に繋いでよい
2. サブゴールを手順だと考えて順番に繋いでよい
3. 戦略ノードを省いて，コンテクストやサブゴールを分岐・合流してもいい
4. 戦略を分岐条件だと考えて記述するとよい
5. コンテクストによって手順の実行状況を逐次的に表現してはいけない

© Dependability Science Lab. Nagoya Univ. 2012

次の中で，正しい記述を指摘しなさい

1. ゴールには実行文を書いてはいけない
2. ゴールには疑問文を書いてもよい
3. ゴールでは根拠となる理由を述べなくてはならない
4. ゴールでは主張を書くのであるから，信念や主義を述べるべきである

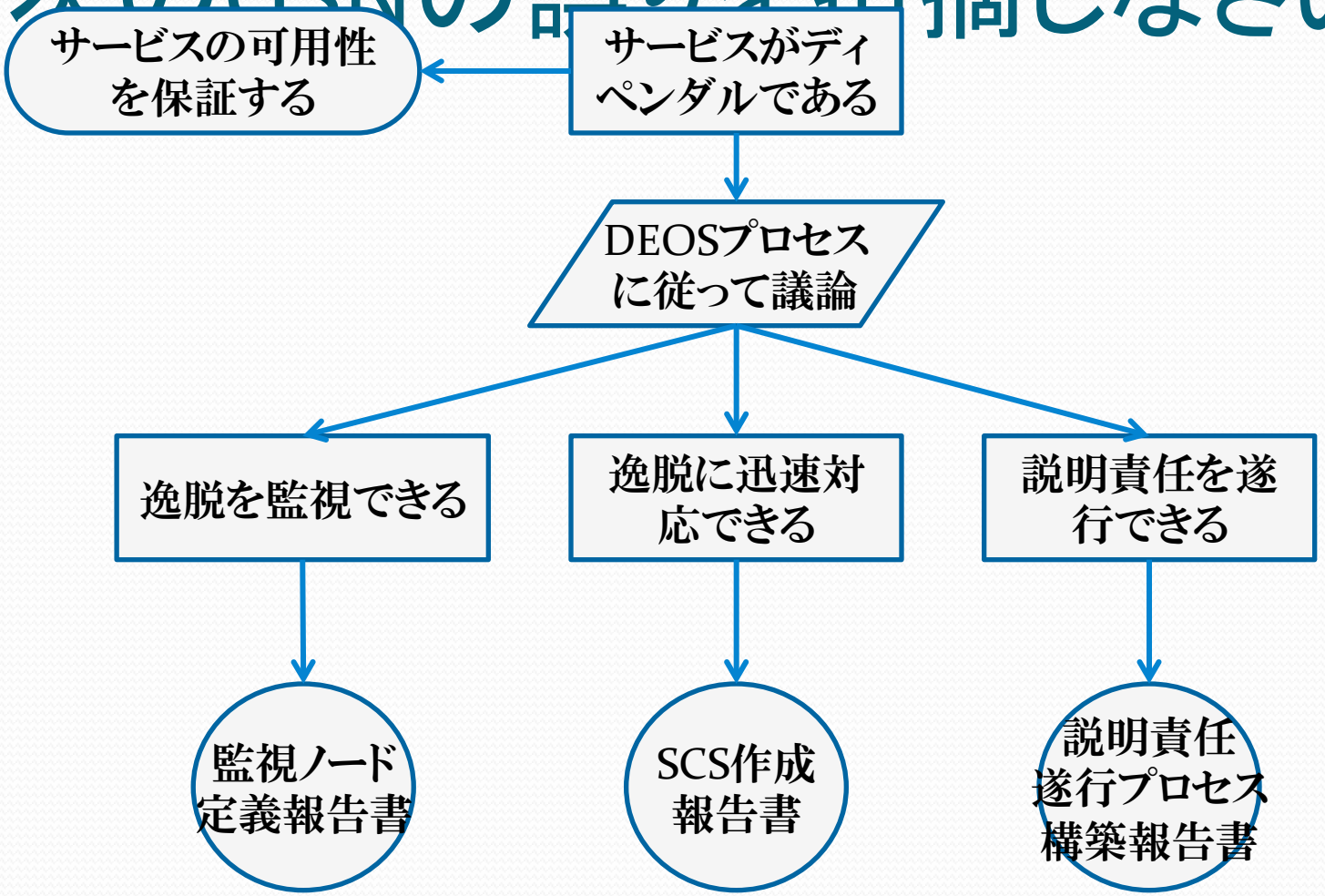
© Dependability Science Lab. Nagoya Univ. 2012

次の中で，正しい記述を指摘しなさい

1. 戦略では条件文を書いてよい
2. 戦略には証跡を接続してもよい
3. 戦略では主張を分解する根拠となる観点を記述しなくてはならない
4. 戦略では主張を分解するのであるから，命題として真偽が判定できるように記述すべきである

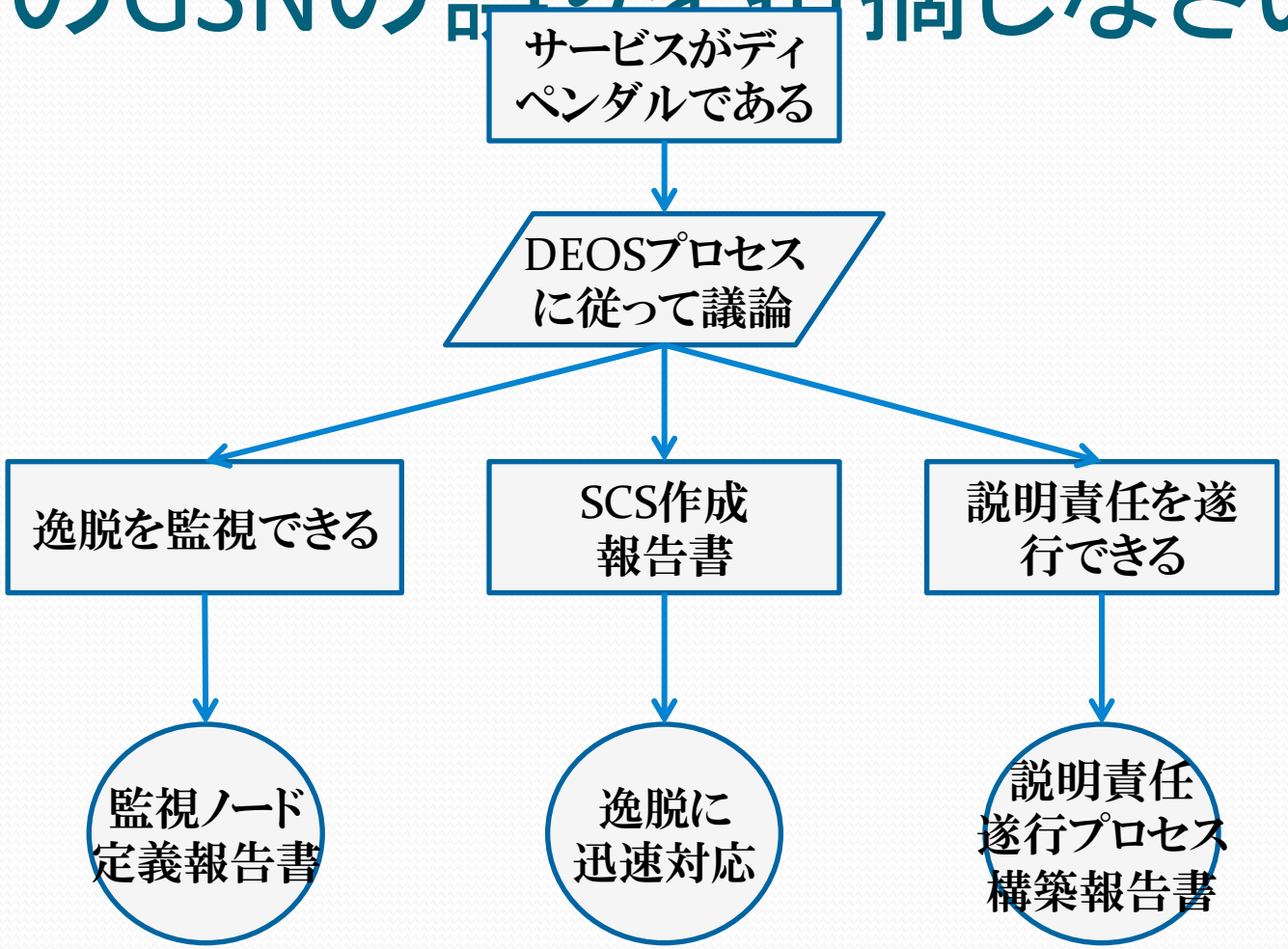
© Dependability Science Lab. Nagoya Univ. 2012

問題4
次のGSNの誤りを指摘しなさい



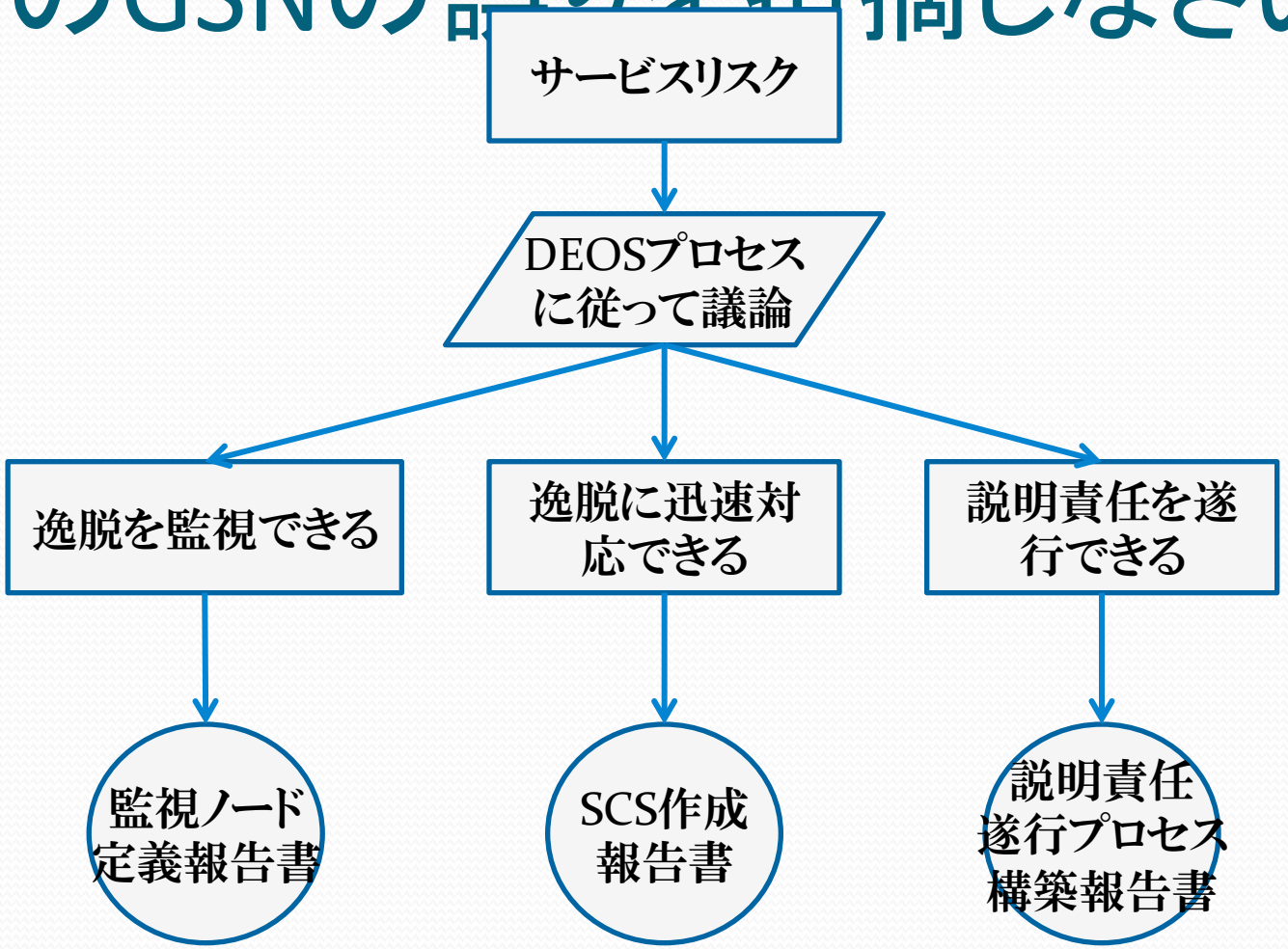
© Dependability Science Lab. Nagoya Univ. 2012

問題5
次のGSNの誤りを指摘しなさい



© Dependability Science Lab. Nagoya Univ. 2012

問題6
次のGSNの誤りを指摘しなさい



© Dependability Science Lab. Nagoya Univ. 2012

D-Case作成

- 電気ポット
- 組込みシステム参照モデル

© Dependability Science Lab. Nagoya Univ. 2012

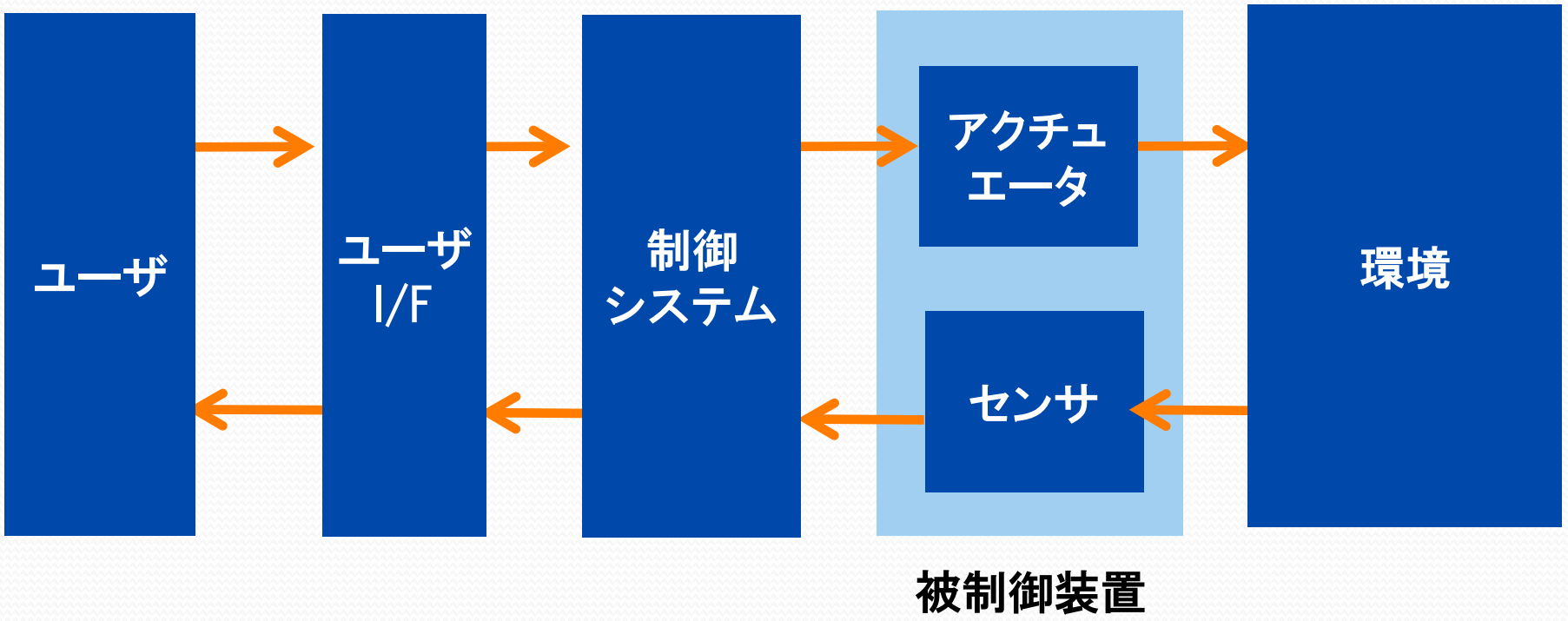
問題1

電気ポットの制御ソフトウェアのディペンダビリティに対するD-Caseを作成しなさい

© Dependability Science Lab. Nagoya Univ. 2012

問題2

組込みシステムの論理参照モデルの制御システムに対するディペンダビリティをD-Caseで確認しなさい

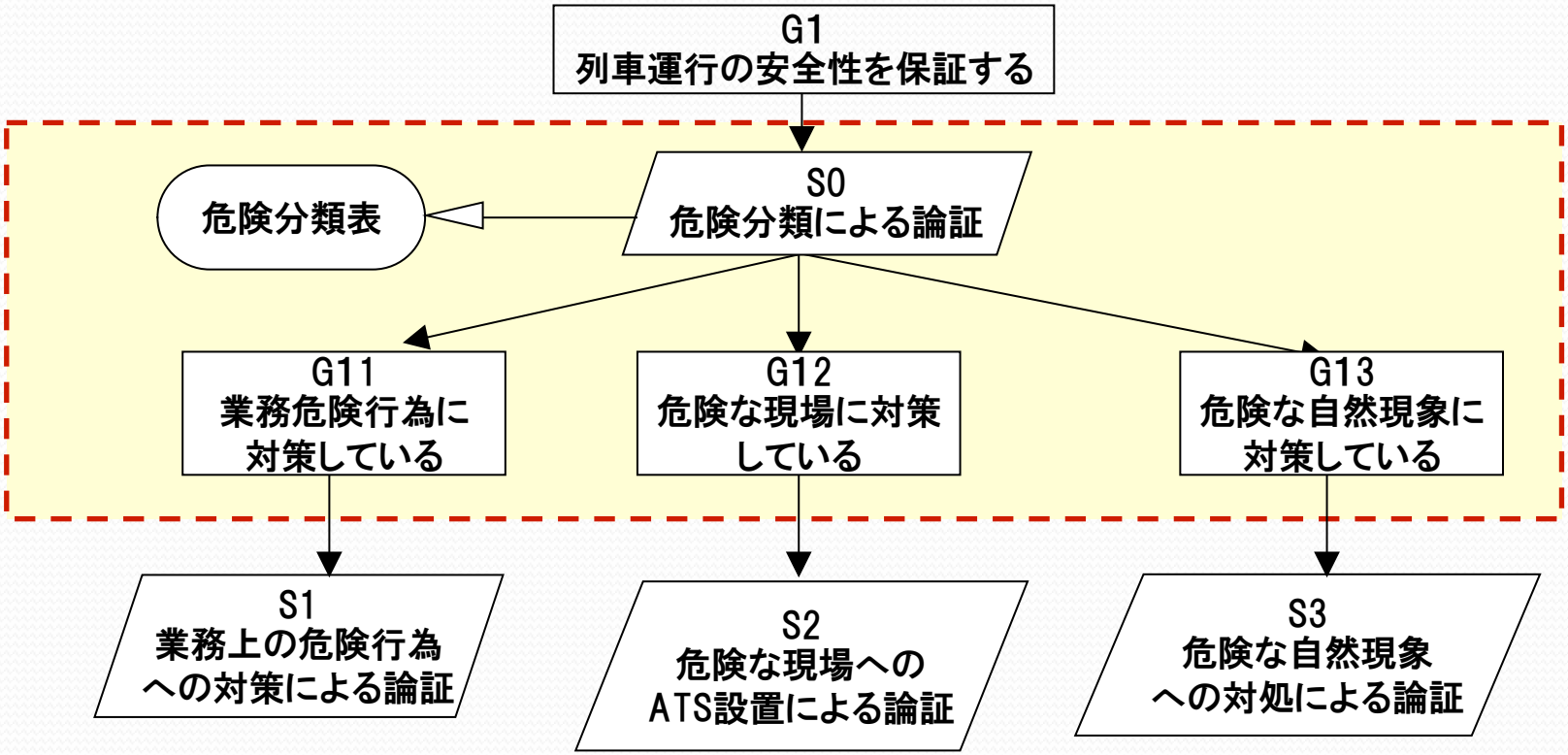


© Dependability Science Lab. Nagoya Univ. 2012

問題回答の指針

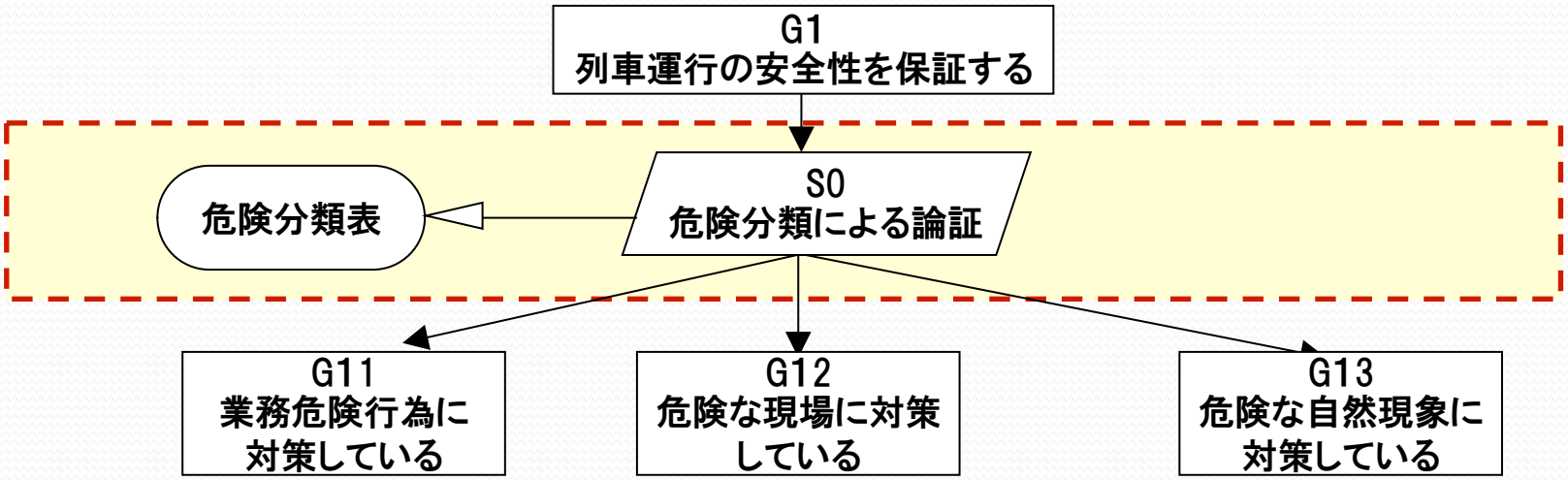
© Dependability Science Lab. Nagoya Univ. 2012

論証を追加したゴール分解(1)



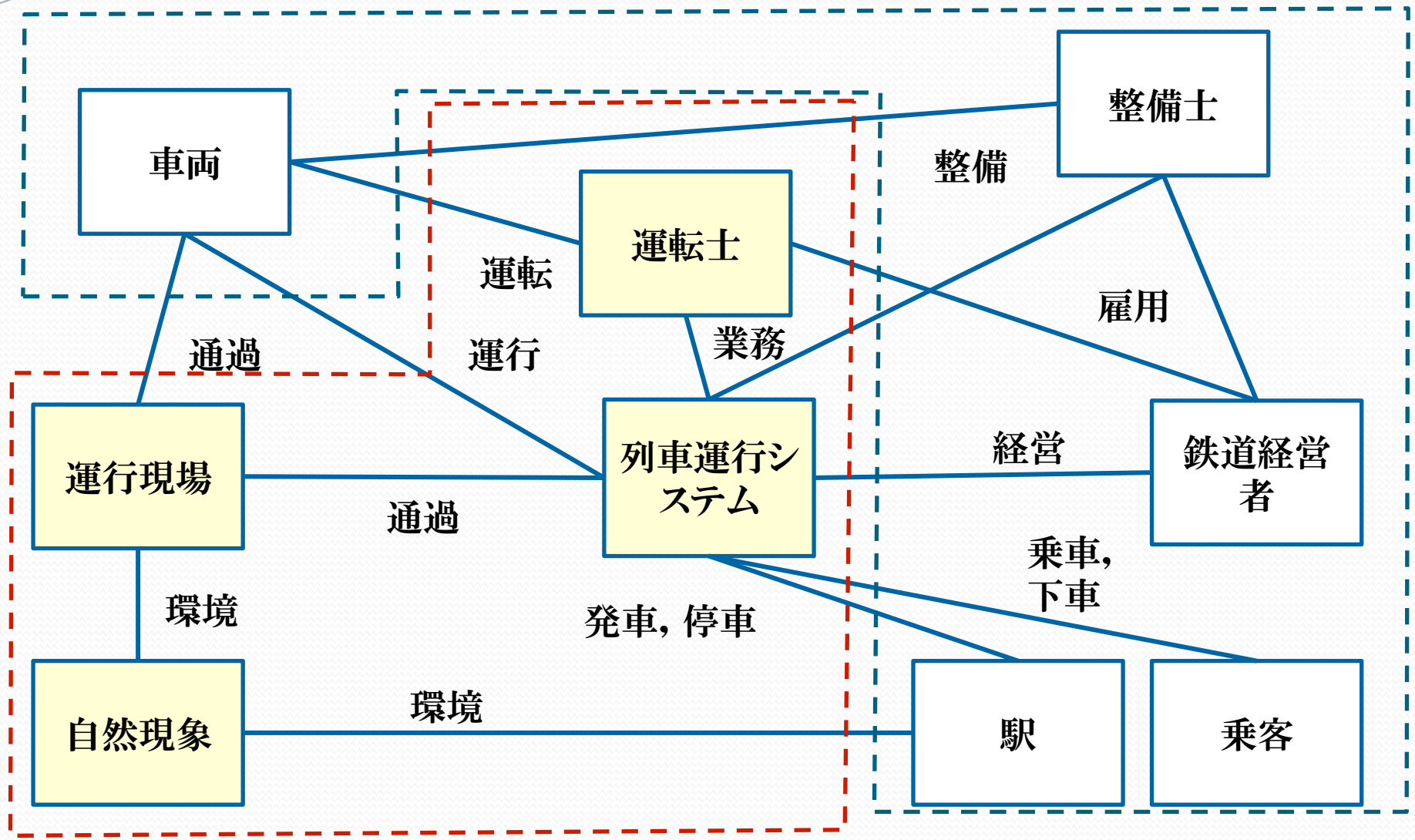
© Dependability Science Lab. Nagoya Univ. 2012

論証を追加したゴール分解(2)



© Dependability Science Lab. Nagoya Univ. 2012

GSNのスコープ分析



対象とするGSNによって保証しようとする範囲

GSNによって保証されていない範囲の例

© Dependability Science Lab. Nagoya Univ. 2012

まとめ

- GSN構造規則
- 構造誤り
- GSN構造の省略
- 要素の命名規則
- GSNの作成
- 問題回答の指針

© Dependability Science Lab. Nagoya Univ. 2012

名古屋大学

D-Case入門
8 議論分解パターン

© Dependability Science Lab. Nagoya Univ. 2012

内容

- **議論分解パターンの基礎**
- 議論分解パターの応用
- 議論のレビュ手法

© Dependability Science Lab. Nagoya Univ. 2012

分解パターンの基礎

- Bloomfieldの分類

© Dependability Science Lab. Nagoya Univ. 2012

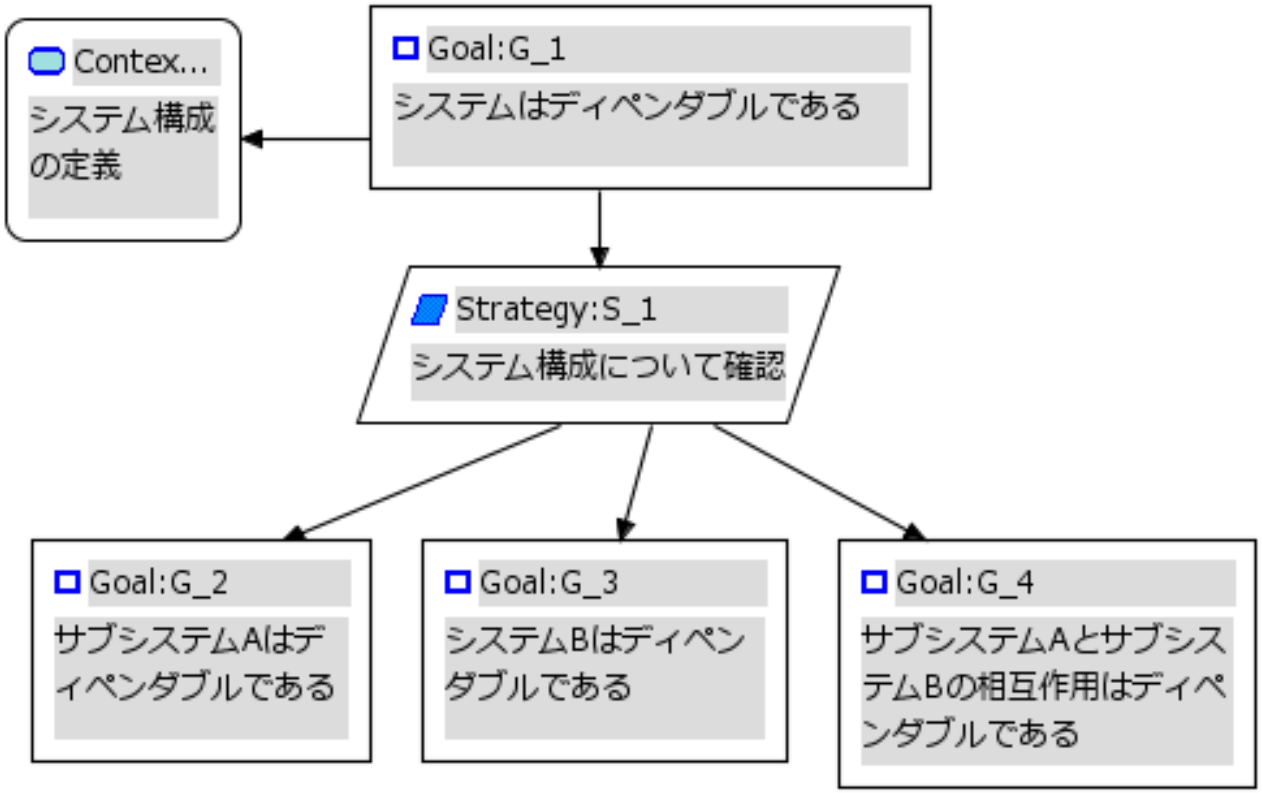
議論分解のタイプ

分解	説明
システム分解	システムを構成するサブシステムごとに，主張が成立することを確認
機能分解	システムの機能構成に基づいて，下位機能システムごとに，上位機能の主張が成立することを確認
属性分解	対象が持つ属性ごとに，対象の主張が成立することを確認
帰納分解	第1段階で主張が整理することと，第N段階で主張が成立するとき第N+1段階でも主張が成立することを確認
完全分解	主張がそれらのすべての要素で成立することを確認
単調分解	既存システムの問題点を将来システムで解決することに限定された主張について確認
具体分解	対象が持つ曖昧さが解消されているという主張を確認

参考) Robin Bloomfield and Peter Bishop, Safety and Assurance Cases: Past, Present and Possible Future – an Adelard Perspective , http://www.scsc.org.uk/pubs/2010%20Bloomfield.pdf

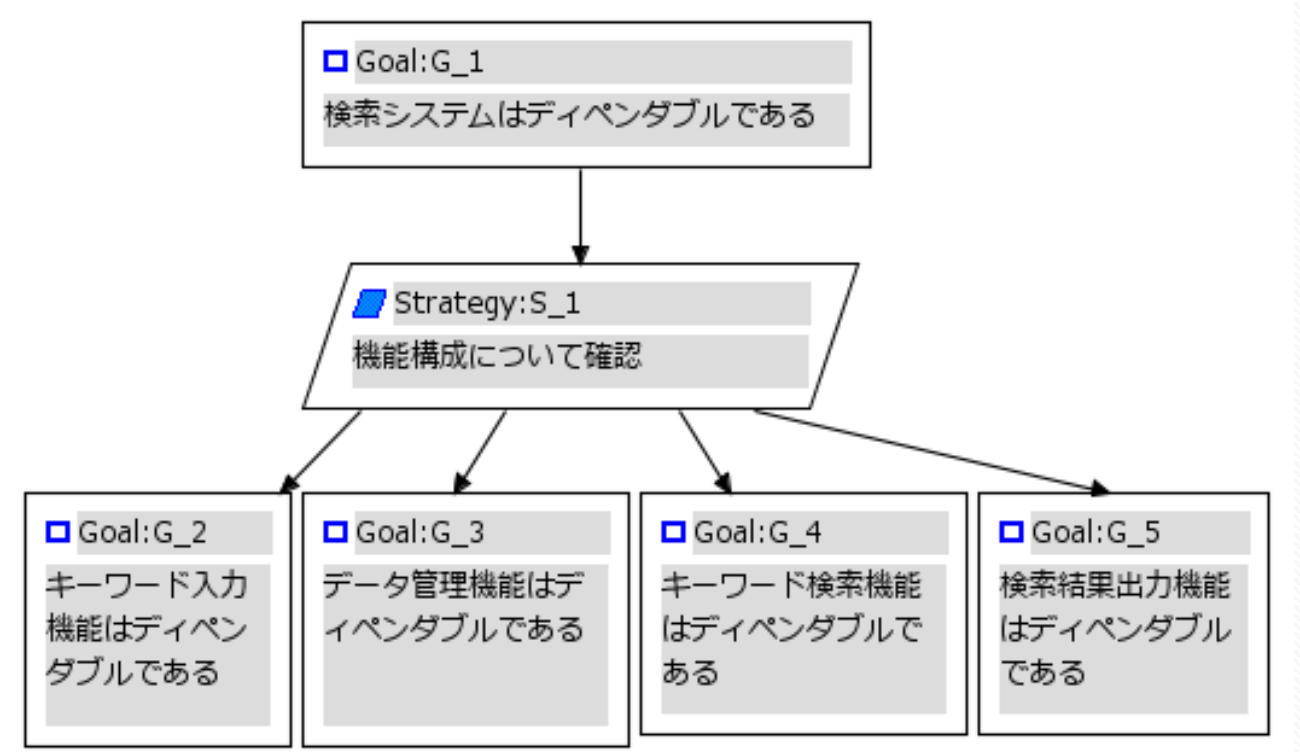
© Dependability Science Lab. Nagoya Univ. 2012

システム公理



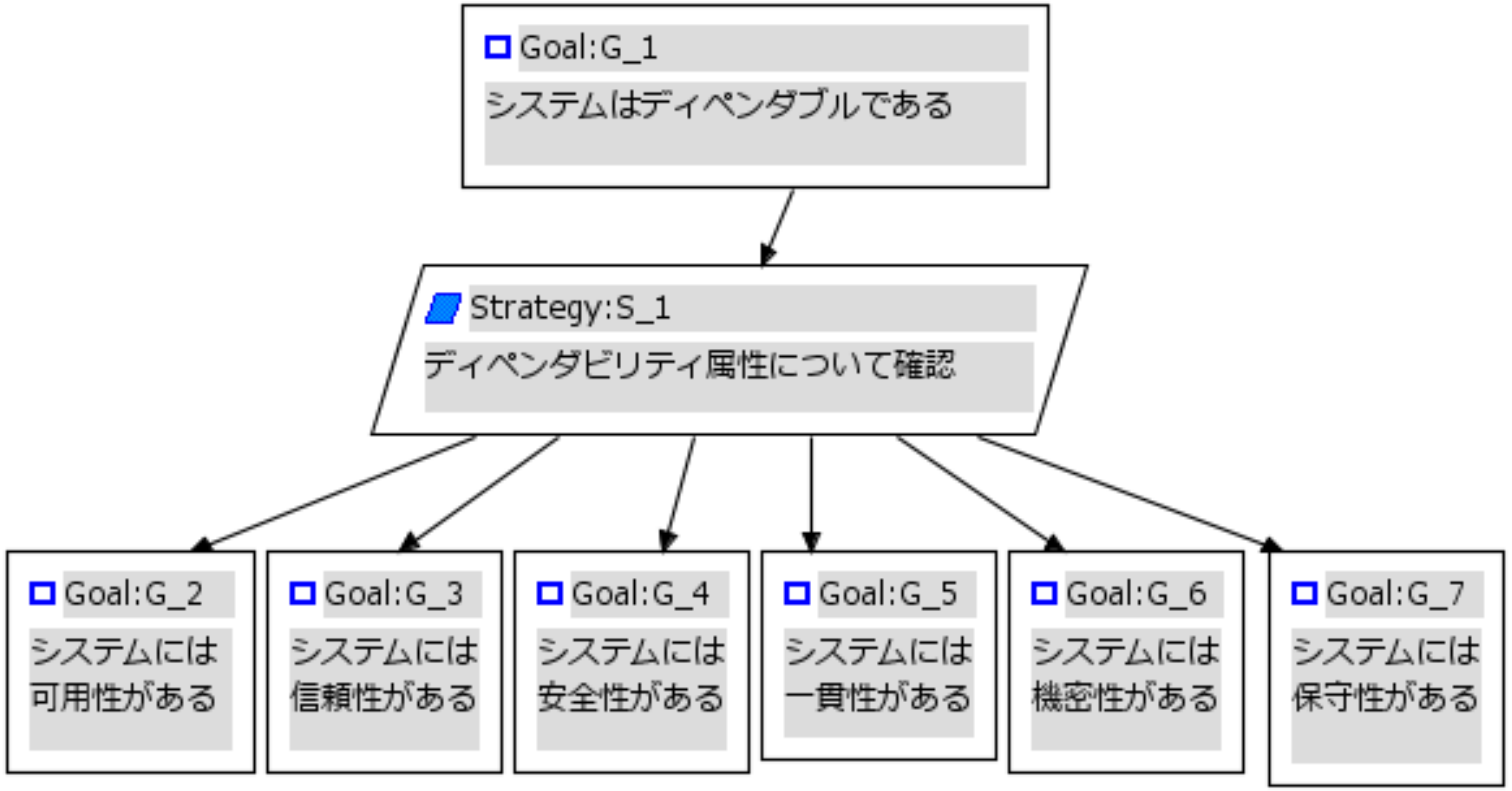
© Dependability Science Lab. Nagoya Univ. 2012

機能分解



© Dependability Science Lab. Nagoya Univ. 2012

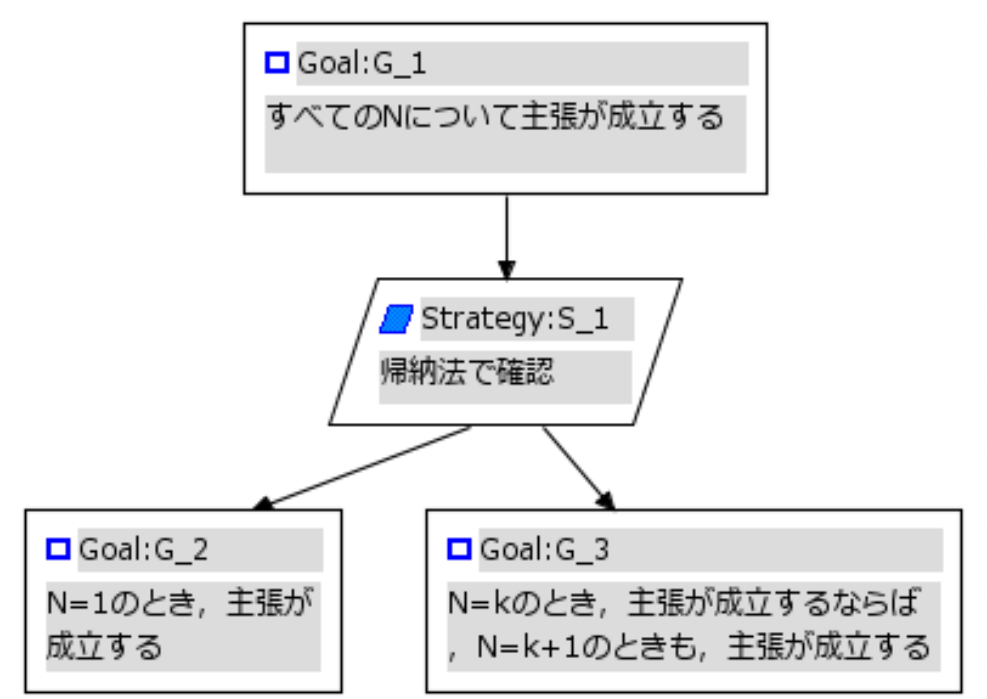
属性分解



© Dependability Science Lab. Nagoya Univ. 2012

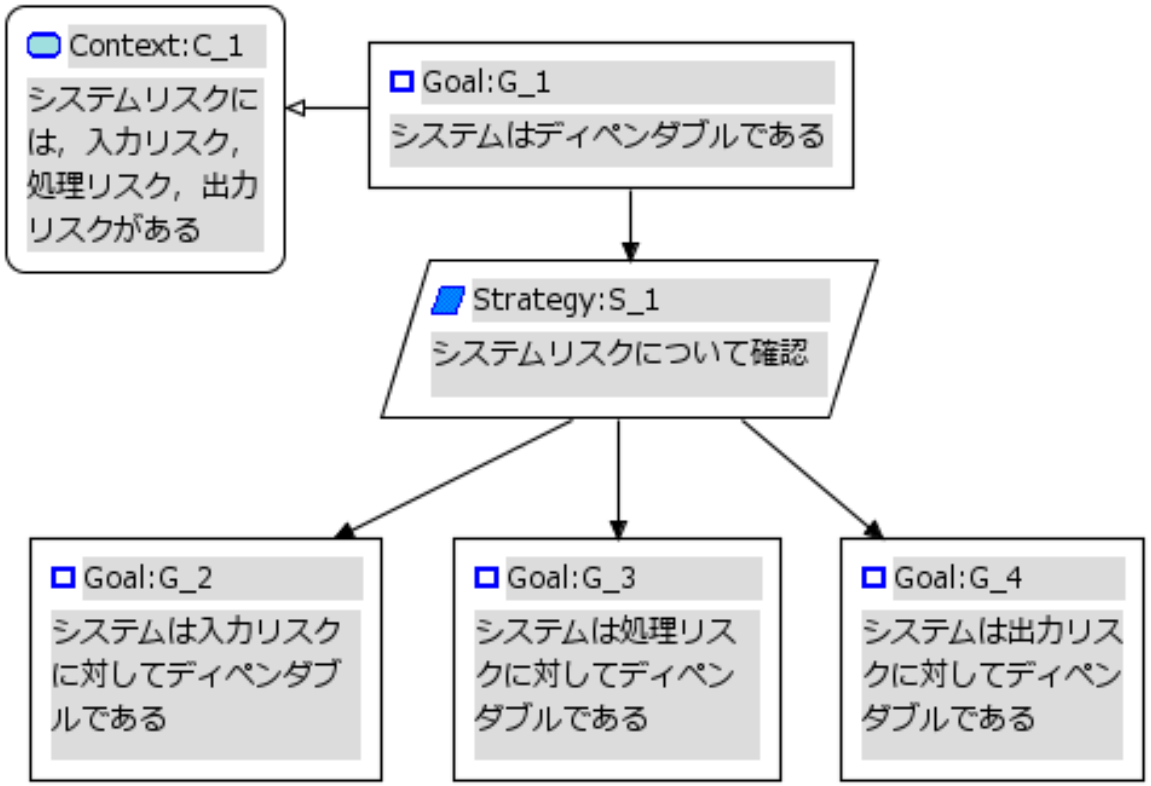
帰納的分解

第1段階で主張が整理することと，第N段階で主張が成立するとき第N+1段階でも主張が成立することを確認



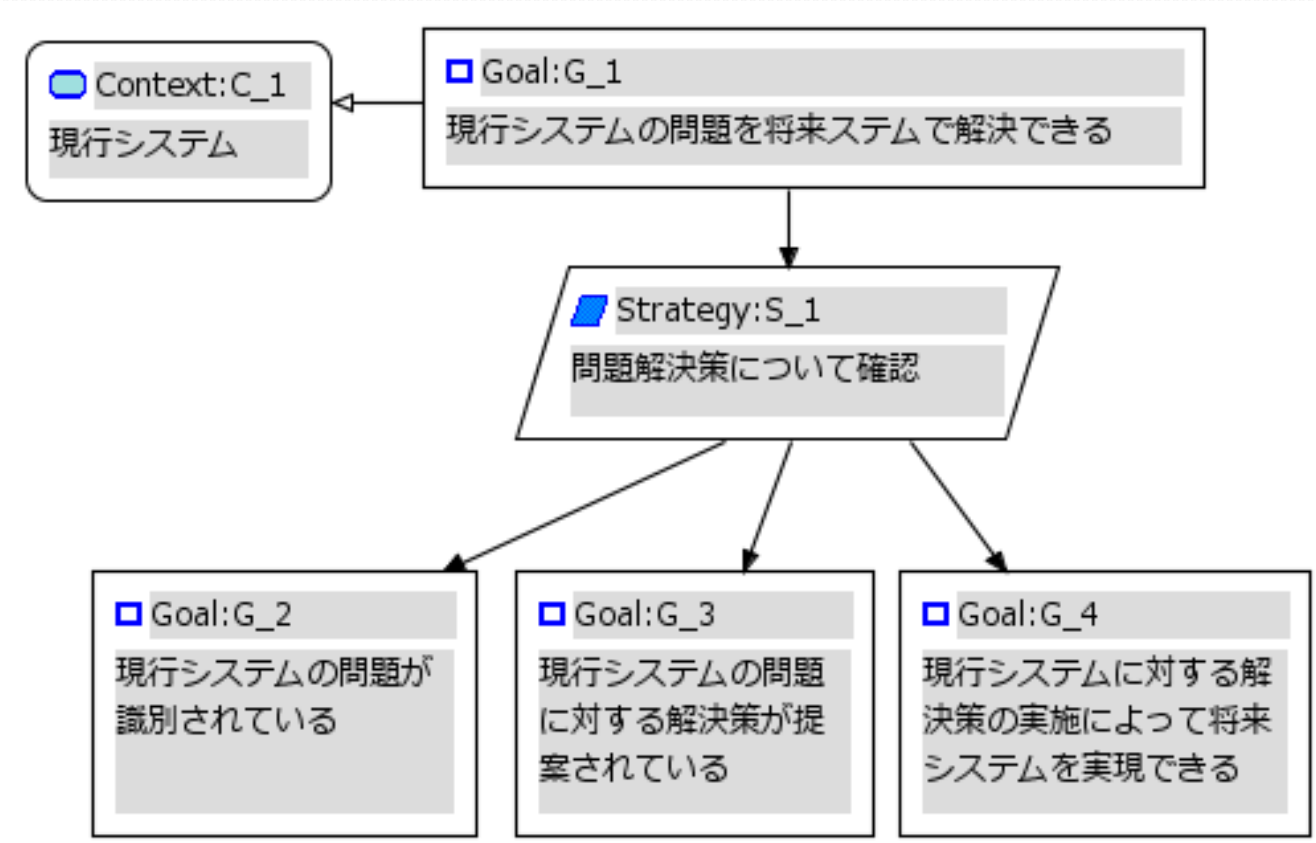
© Dependability Science Lab. Nagoya Univ. 2012

完全分解



© Dependability Science Lab. Nagoya Univ. 2012

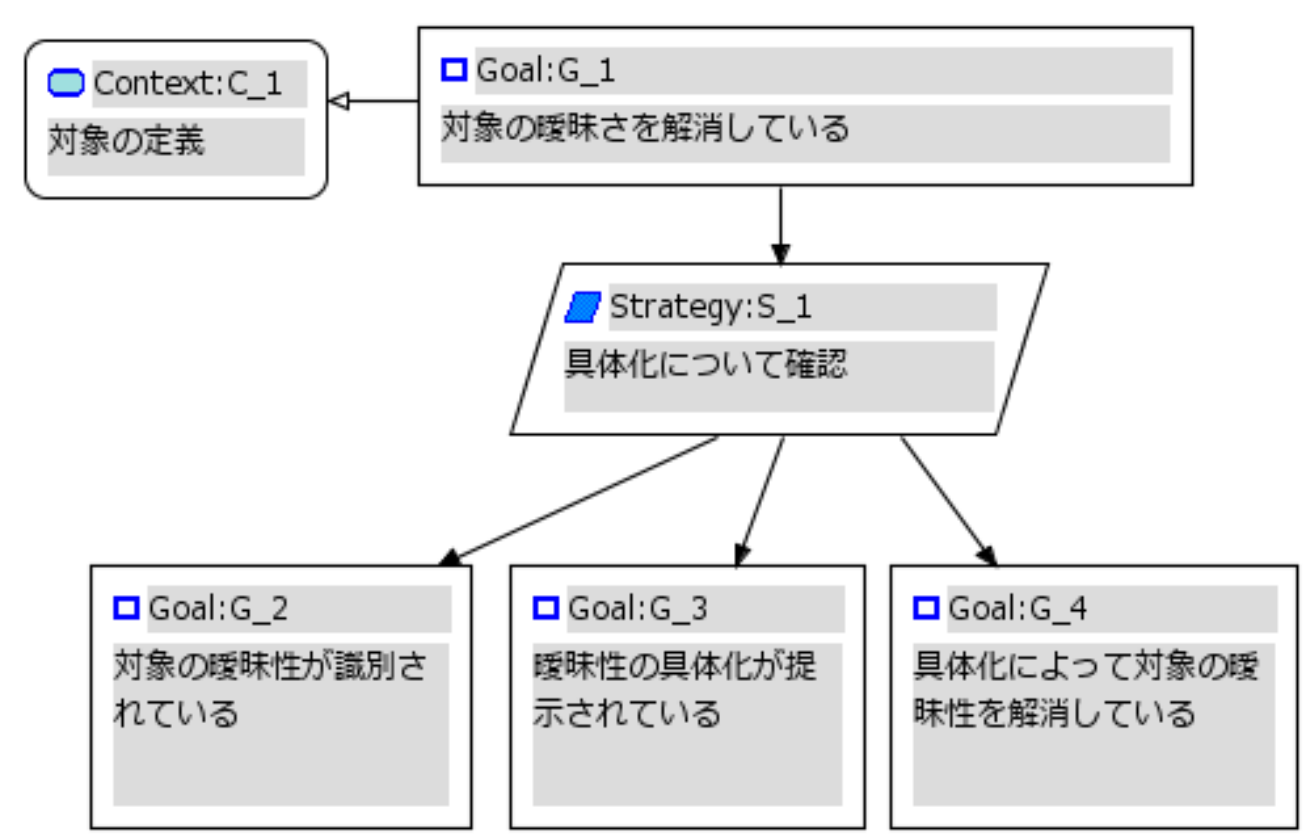
単調分解



参考）問題解決では，現行システムから将来システムへの一方向性があるので単調分解という

© Dependability Science Lab. Nagoya Univ. 2012

具体分解



© Dependability Science Lab. Nagoya Univ. 2012

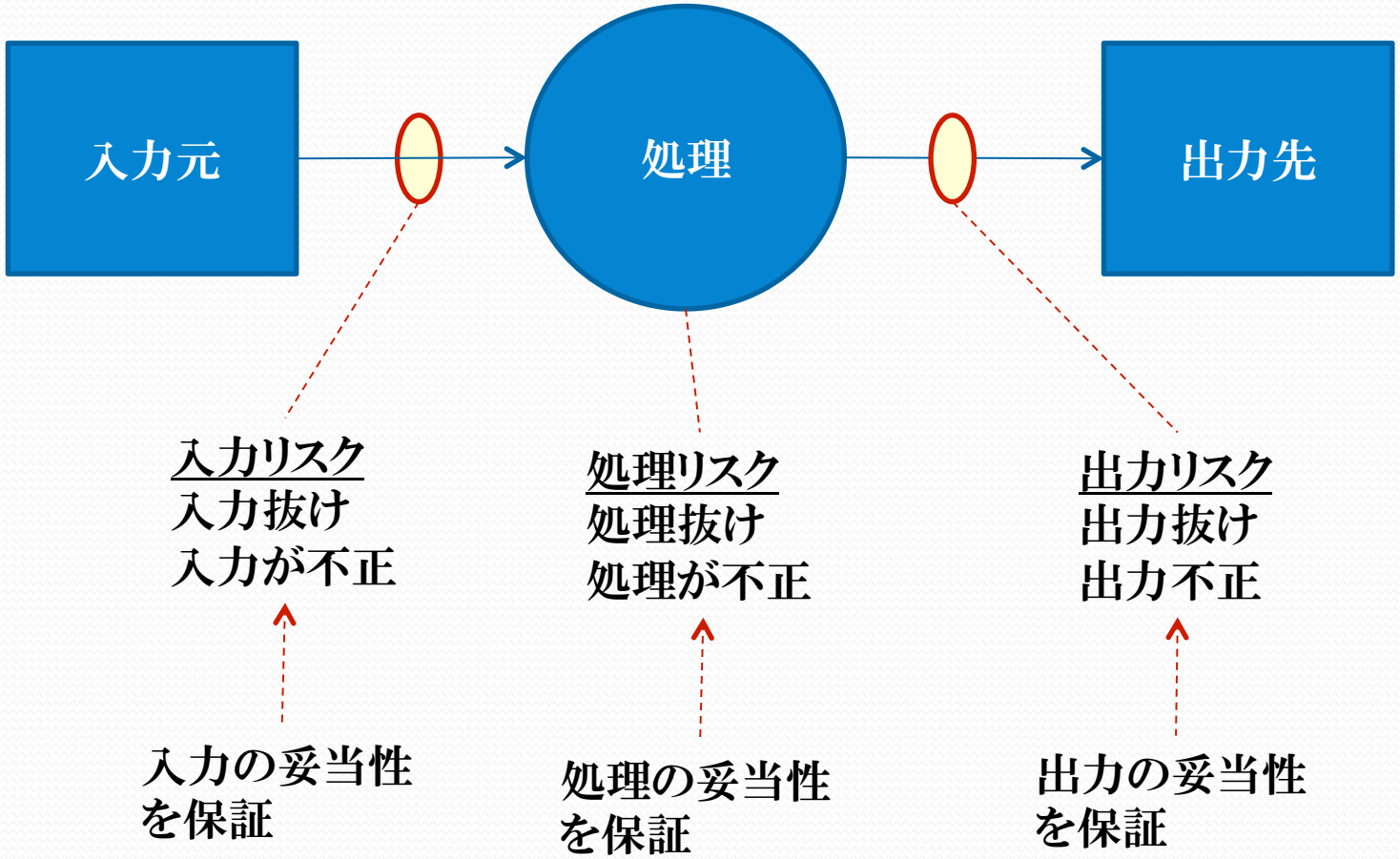
分解パターンの応用

- プロセス分解
- 階層分解
- DFD階層分解
- ECA分解
- 条件判断分解
- 代替案選択分解

© Dependability Science Lab. Nagoya Univ. 2012

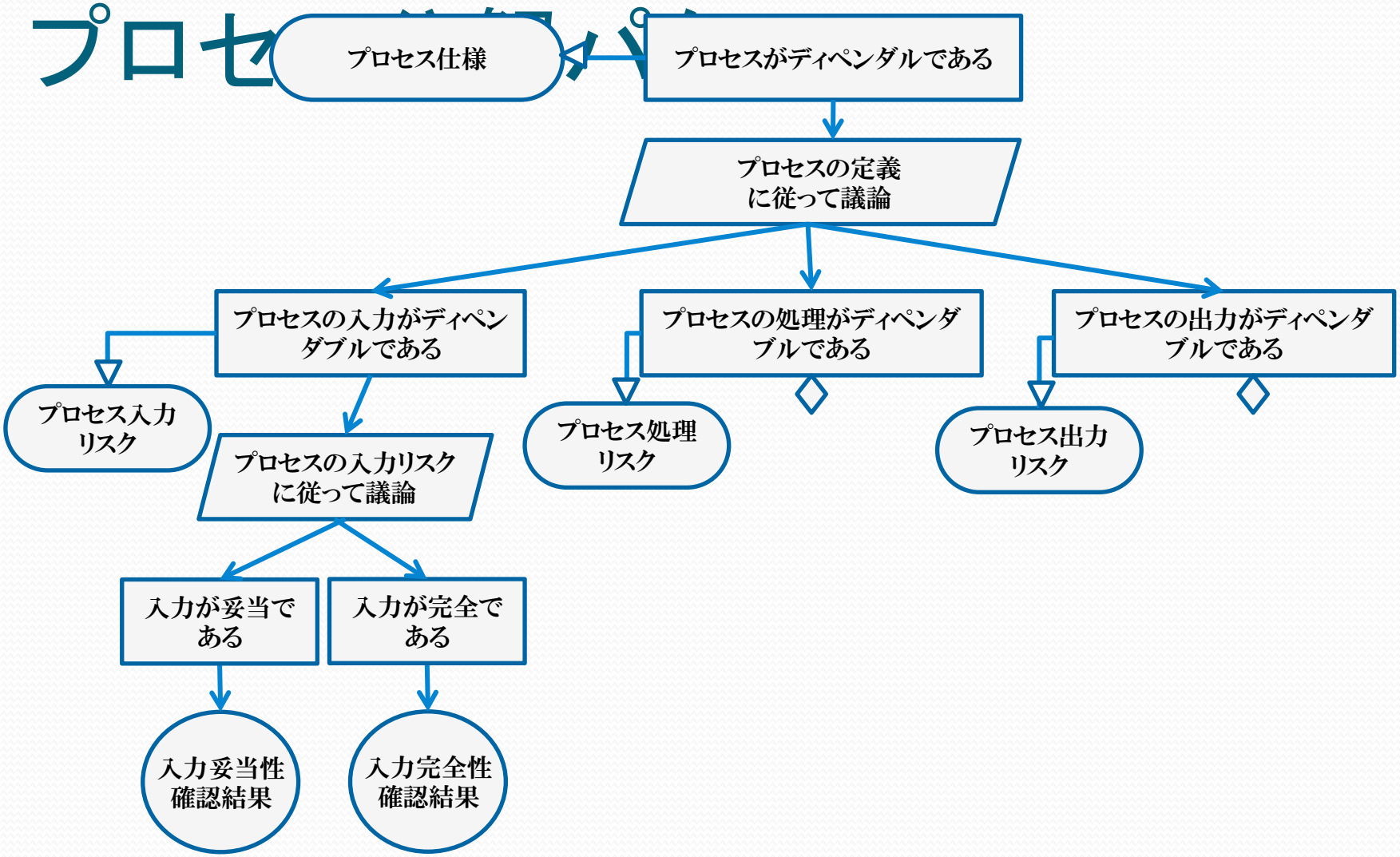
プロセス分解

- プロセスのリスク分析に基づいて分解する



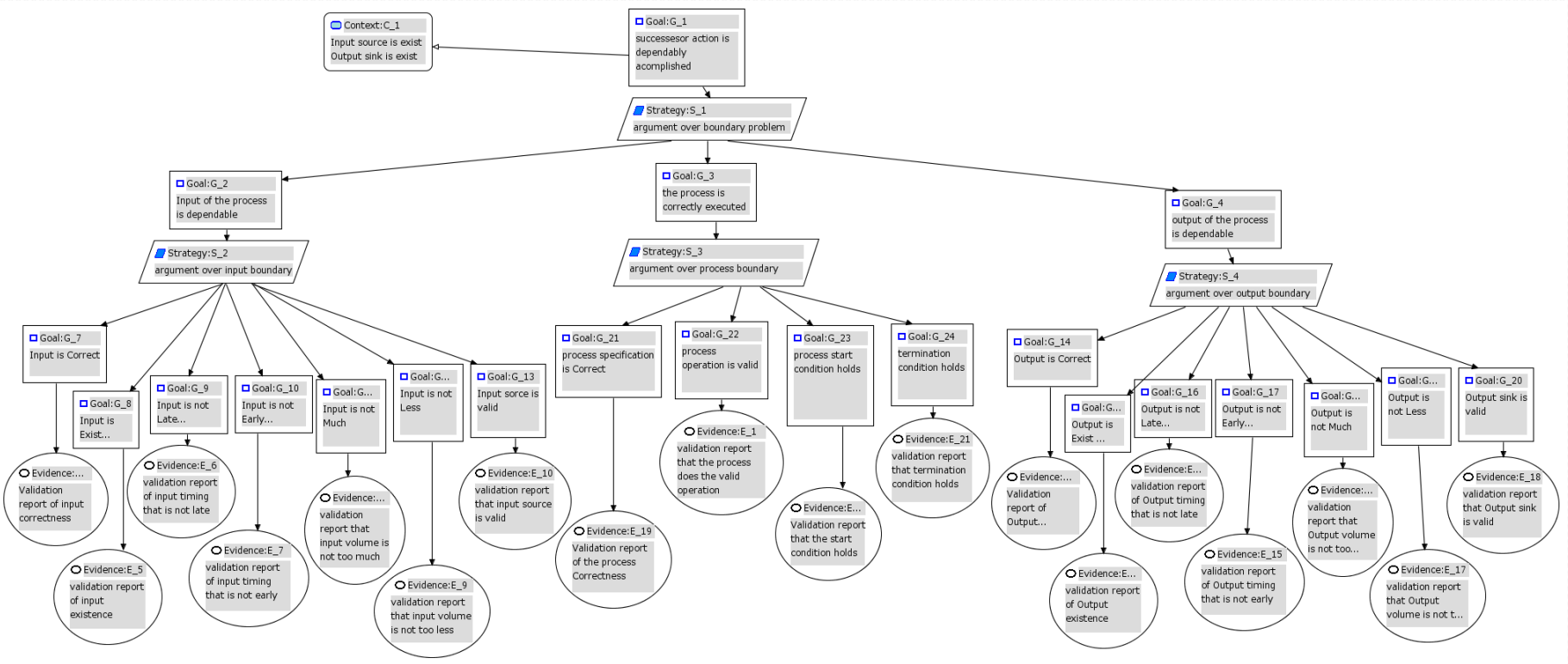
© Dependability Science Lab. Nagoya Univ. 2012

プロセス



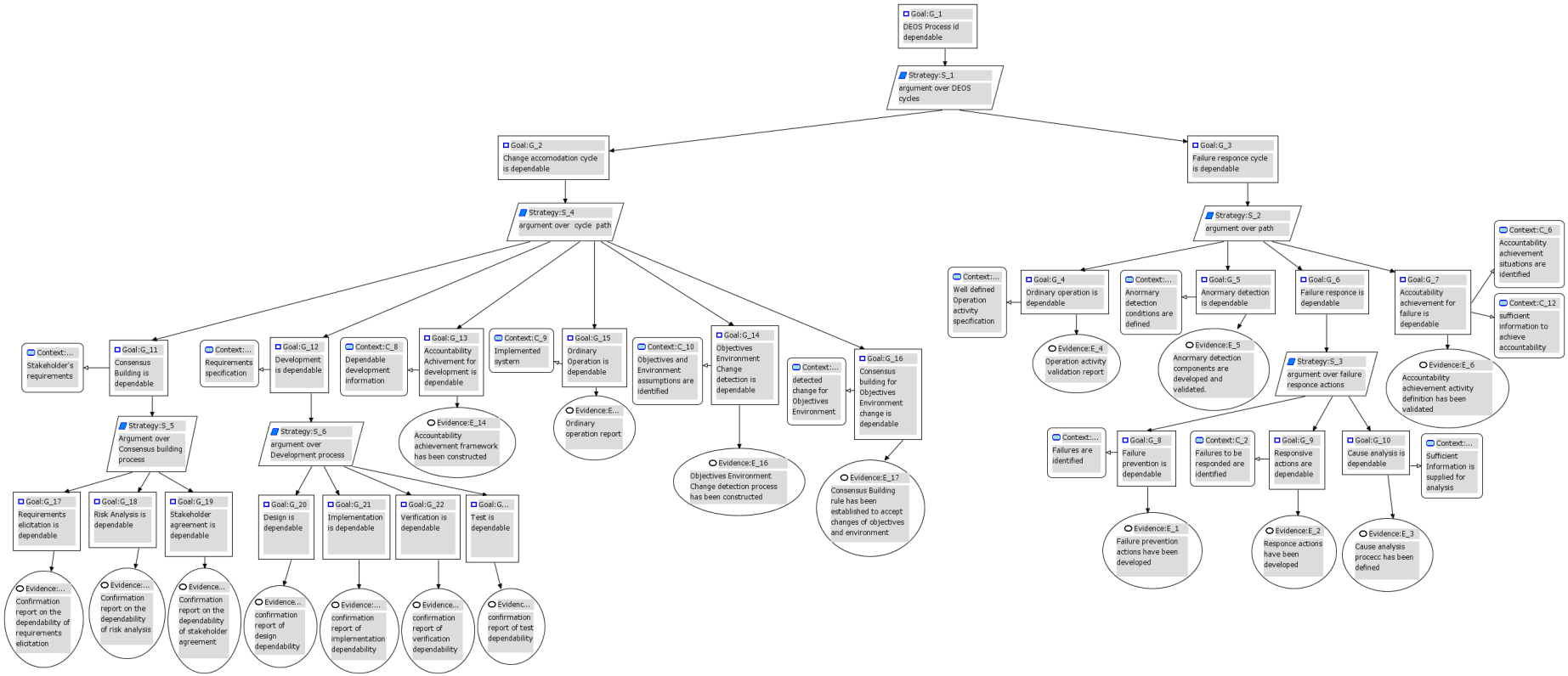
© Dependability Science Lab. Nagoya Univ. 2012

プロセス分解パターン(プロセス全体)



© Dependability Science Lab. Nagoya Univ. 2012

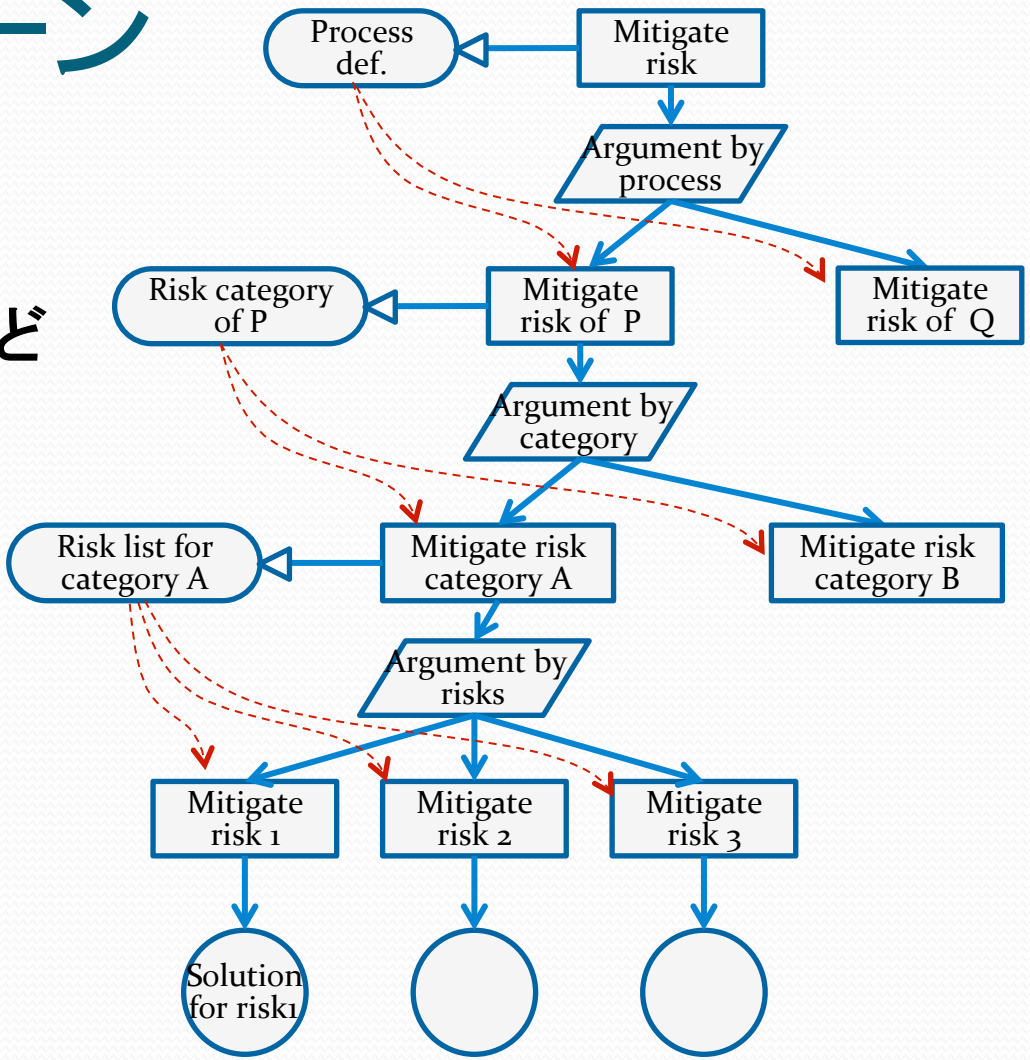
プロセス分解パターン（DEOSプロセス）



© Dependability Science Lab. Nagoya Univ. 2012

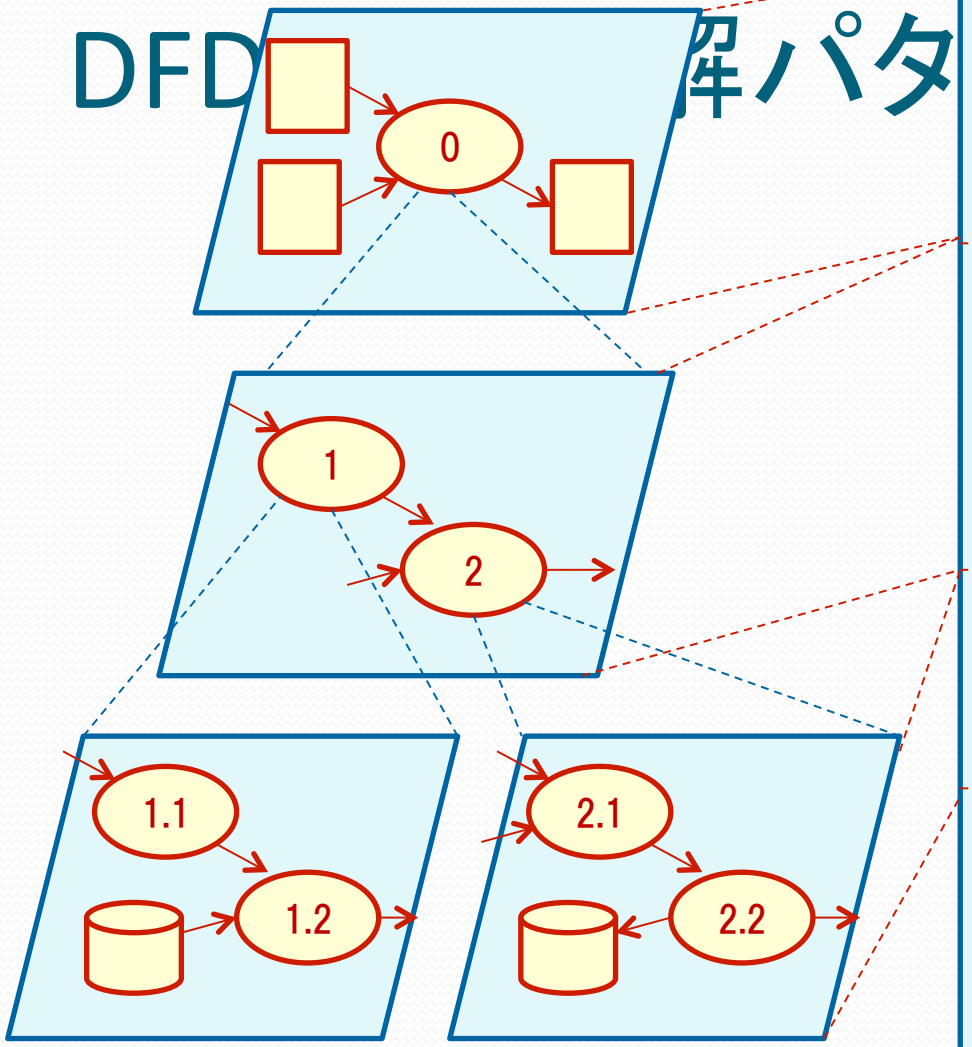
階層分解パターン

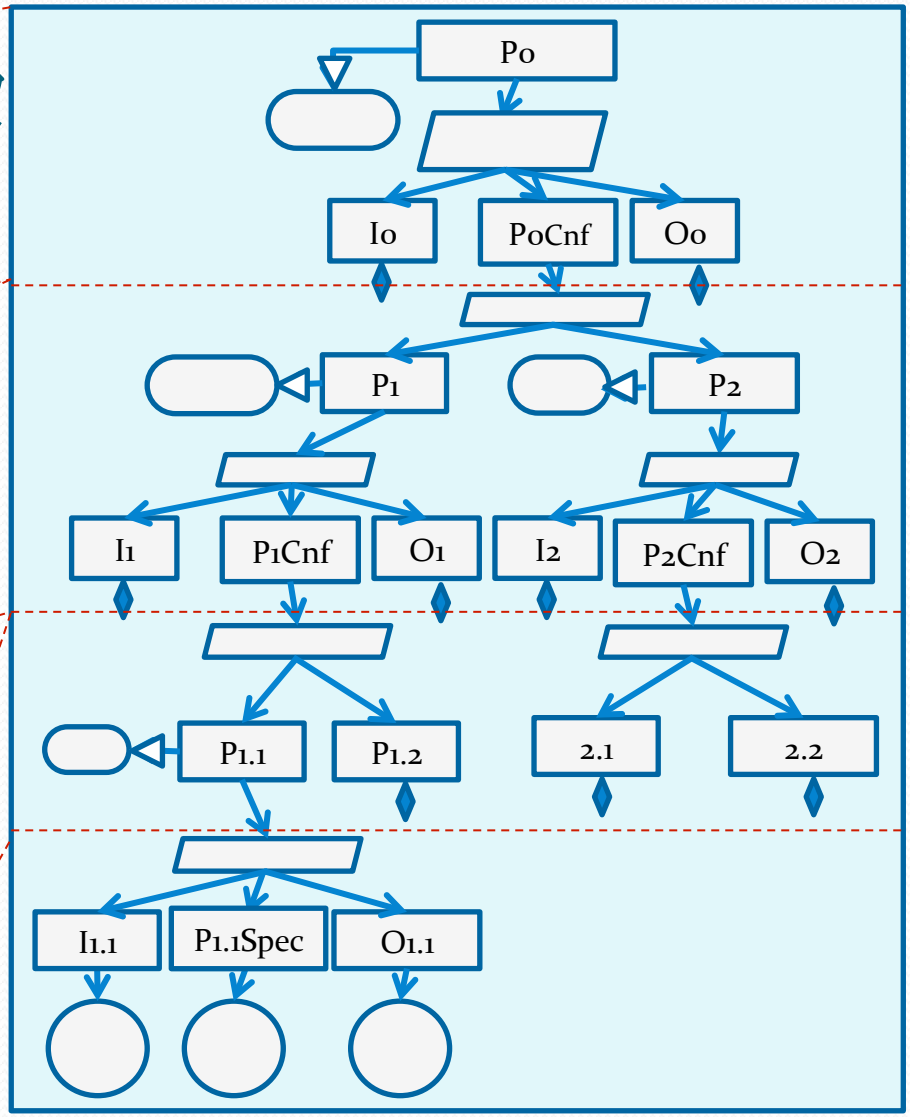
- 4 階層パターン
  - プロセス階層
    - 要求，設計，製造など
  - リスク分類層
    - FR, NFR, UI, etc.
  - リスク分解層
  - リスク対策層
  - リスク対策証跡



© Dependability Science Lab. Nagoya Univ. 2012

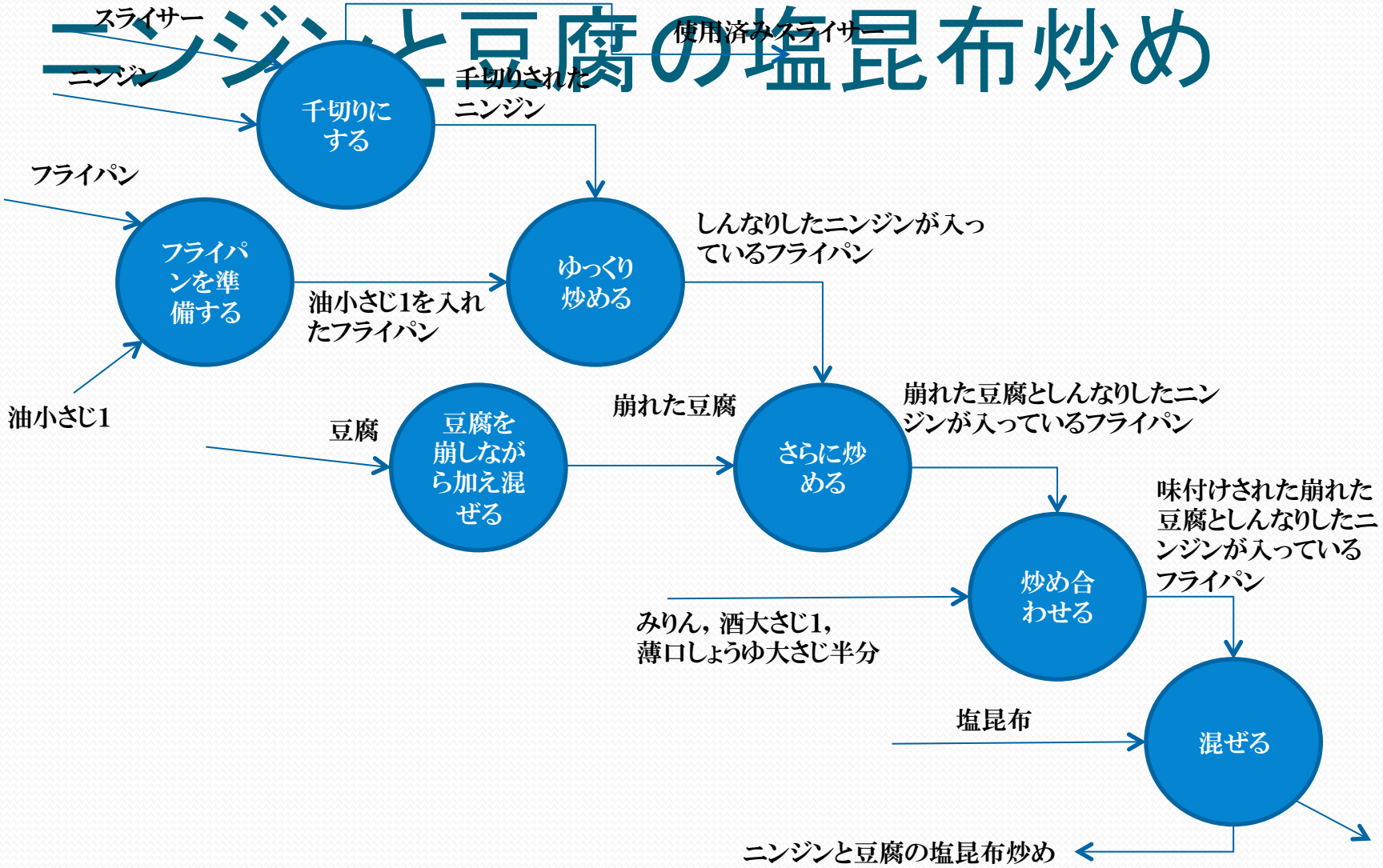
DFD分解パタ





© Dependability Science Lab. Nagoya Univ. 2012

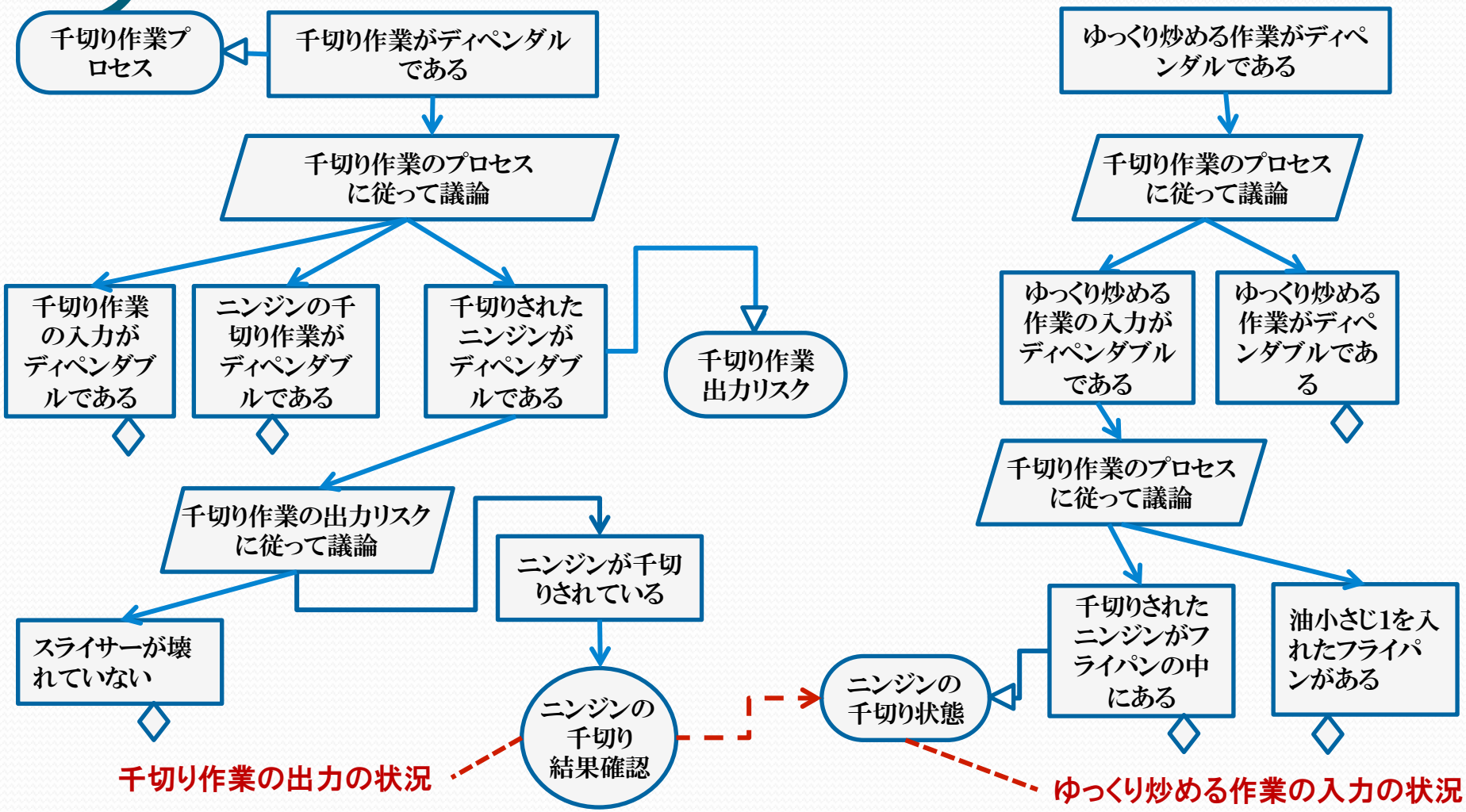
ニンジンと豆腐の塩昆布炒め



© Dependability Science Lab. Nagoya Univ. 2012

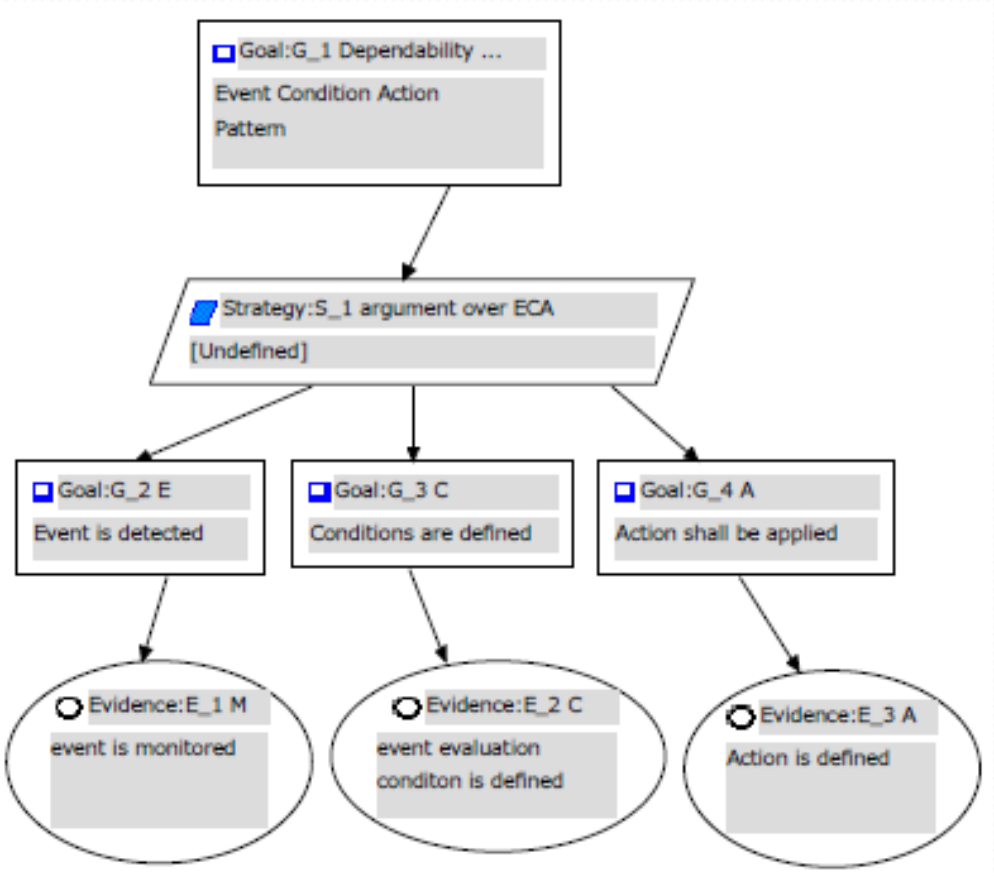
プロセス関係に基づく分解パター

先行プロセスの出力証跡が，後続プロセスの入力についての主張のコンテクストになる



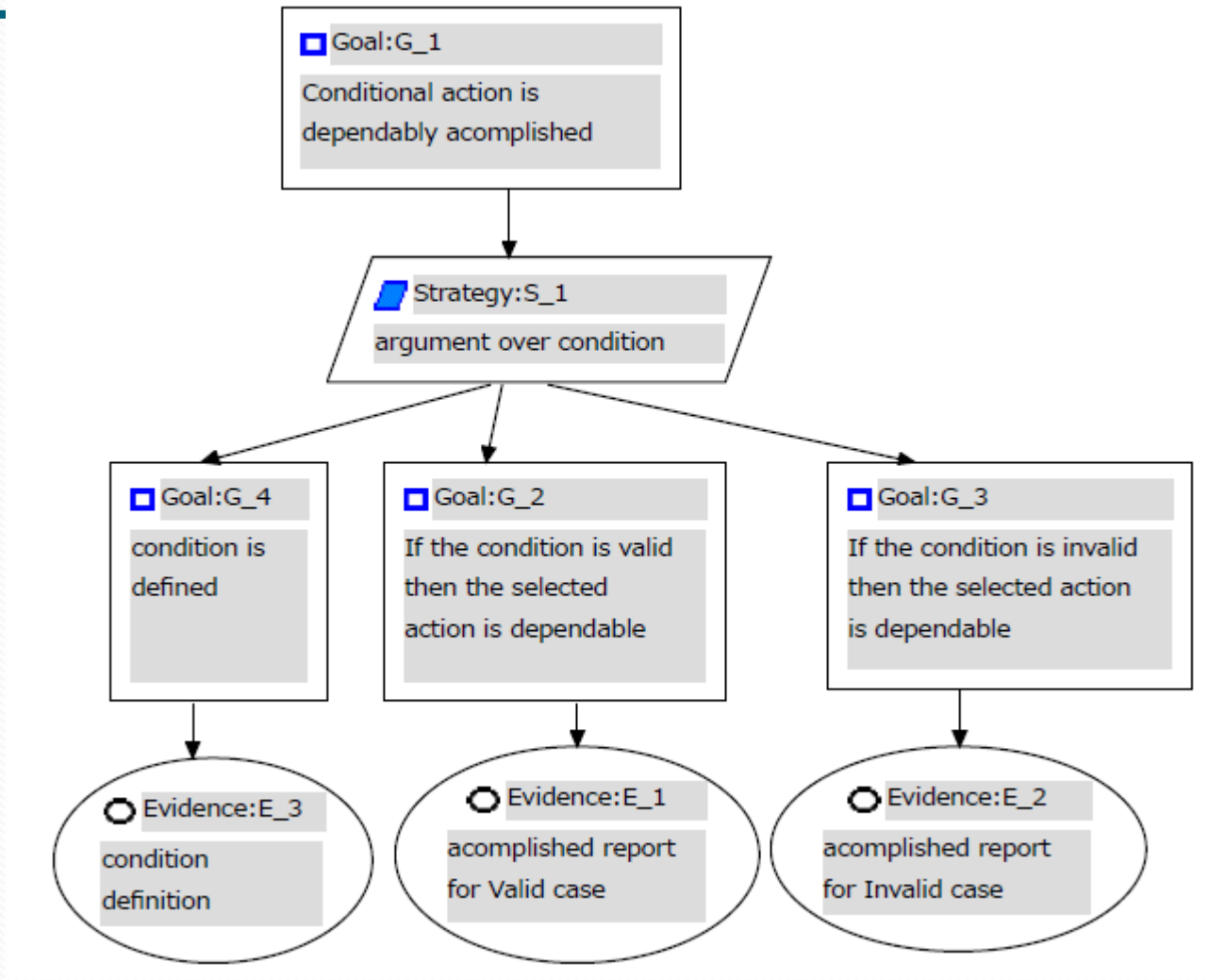
© Dependability Science Lab. Nagoya Univ. 2012

ECA D-Caseパターン



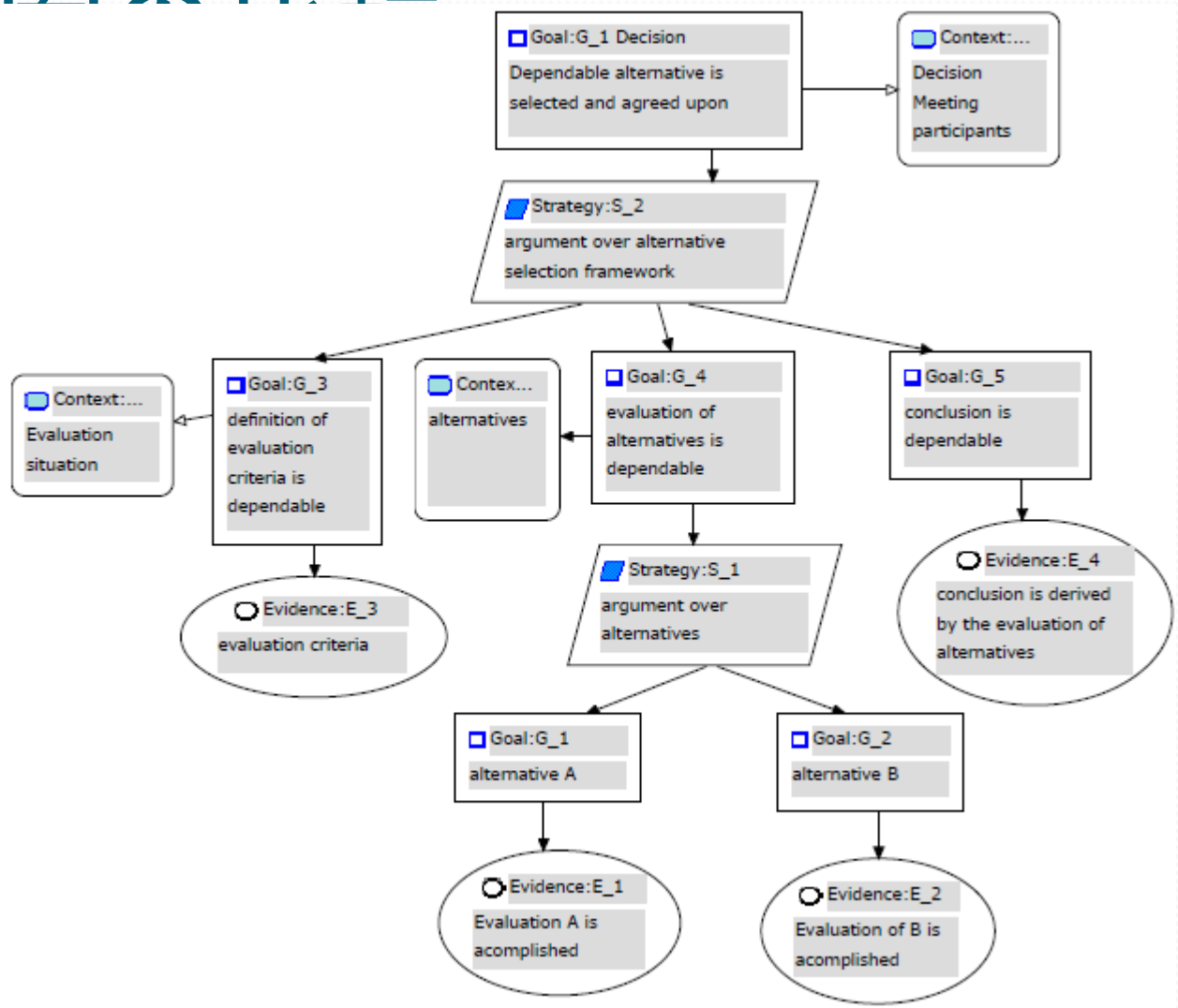
© Dependability Science Lab. Nagoya Univ. 2012

条件選択のD-Case



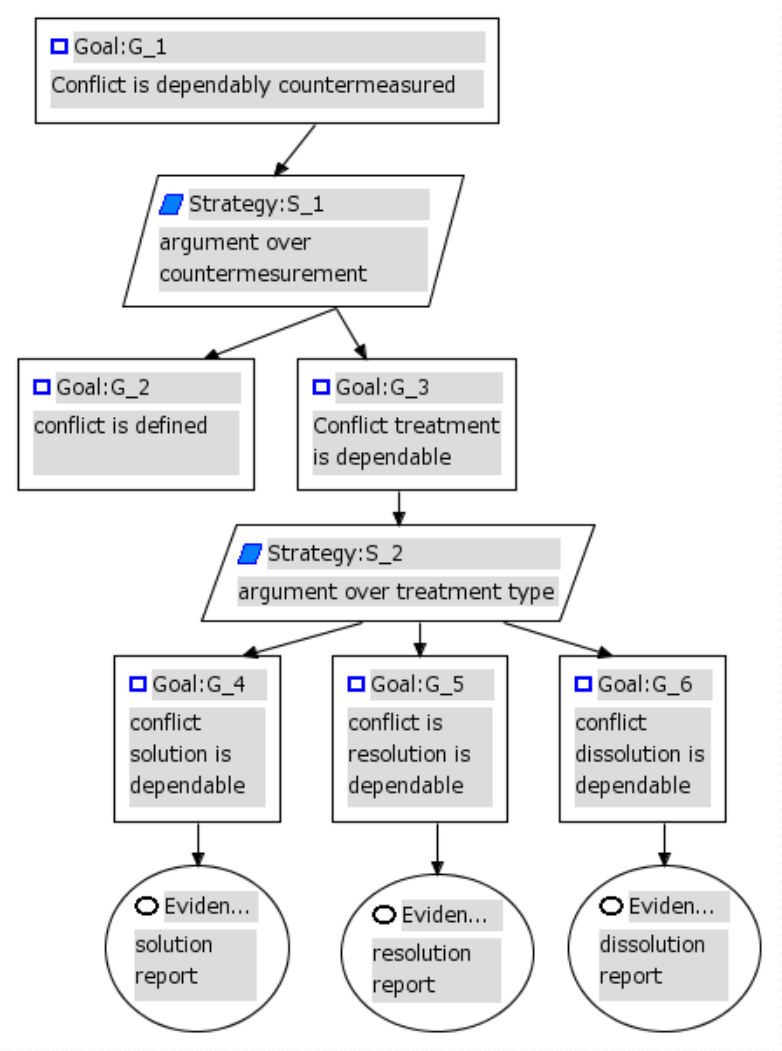
© Dependability Science Lab. Nagoya Univ. 2012

代替案の比較



© Dependability Science Lab. Nagoya Univ. 2012

矛盾解決



© Dependability Science Lab. Nagoya Univ. 2012

レビュ手法

© Dependability Science Lab. Nagoya Univ. 2012

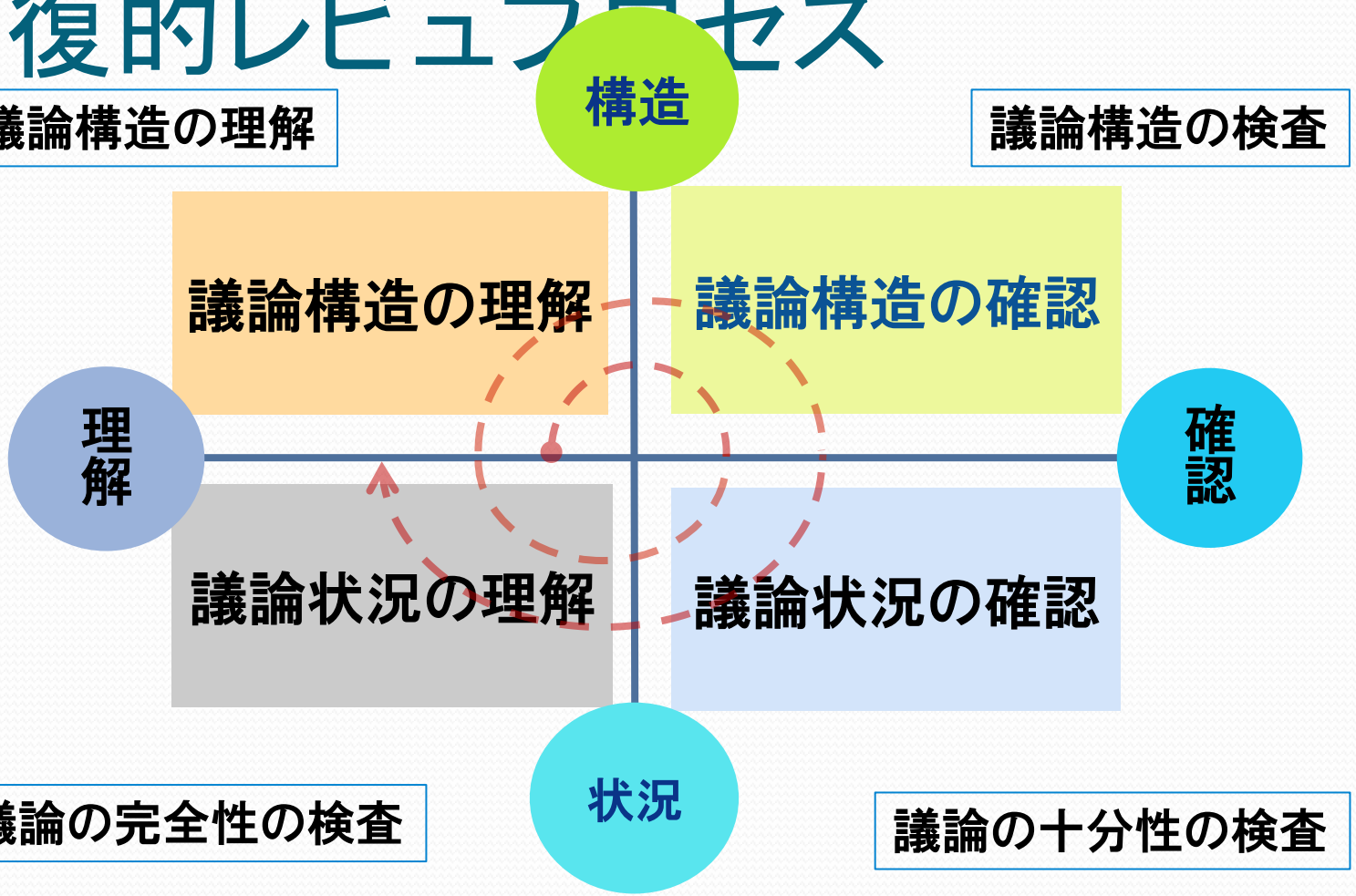
段階的レビュー手法

段階	目的	環境
1	議論構造の理解	保証ケースの基本要素を識別する 議論の関係を識別 構造記法を用いて，議論構造を再表現
2	議論構造の検査	議論の構造的な欠陥を検出 議論の循環性，不完全性，証跡の役割の曖昧性 要素の結合性
3	議論の十分性の検査	議論の構成要素に抜けがないことを検査 コンテクスト情報を追加
4	議論の批評と反論	演繹証明的議論と帰納推論的議論を識別 演繹証明的議論では真偽の妥当性を確認 帰納推論的議論では結論に対する前提の十分性を確認

参考) Kelly, T. P. “Reviewing Assurance Arguments—A Step-by-Step Approach.” http:// www-users.cs.york.ac.uk/~tpk/dsnworkshop07.pdf (2007)

© Dependability Science Lab. Nagoya Univ. 2012

反復的レビュープロセス



© Dependability Science Lab. Nagoya Univ. 2012

特性	内容	指摘例
網羅性	議論と証跡が主張をどの範囲まで保証できるか	すべてのハザードについての結論に対して，一部のハザードの証跡しか提示していない
独立性	証跡間に独立性があるか	2つの証跡が同じ故障モデルに基づいていると，相互に依存しているため，結論を強く確信できなくなる
限定性	議論や証跡が適切に限定されているか	議論の範囲が狭すぎると，保証範囲が不十分になる
直接性	議論や証跡が結論を直接保証しているか	特定の製品についての主張に対してプロセスによる証跡は間接的である
関連性	結論に対して議論や証跡の要素が適切な関連をもっているか	要求をコードが実現していないと，要求に対するコードの追跡性がない
頑健性	証跡と主張の変化に対して議論が対応できるか	議論が目的に正確に一致していると，議論に変化の余裕がないので議論の頑健性が低い

© Dependability Science Lab. Nagoya Univ. 2012

議論の妥当性

	問題状況	対策
反駁	ディペンダビリティケースの主張と反対の結論を導く証跡がある	主張と相対する具体化可能な主張との対立関係を記述する
限定	推論を破たんさせる付加的な議論や証跡がある	同じ前提条件の下で結論が成立しないような状況がないか調べる

© Dependability Science Lab. Nagoya Univ. 2012

まとめ

- 議論分解パターンの基礎
- 議論分解パターの応用
- 議論のレビュ手法

© Dependability Science Lab. Nagoya Univ. 2012

名古屋大学

D-Case入門
9 D-Case Editor

© Dependability Science Lab. Nagoya University, 2012

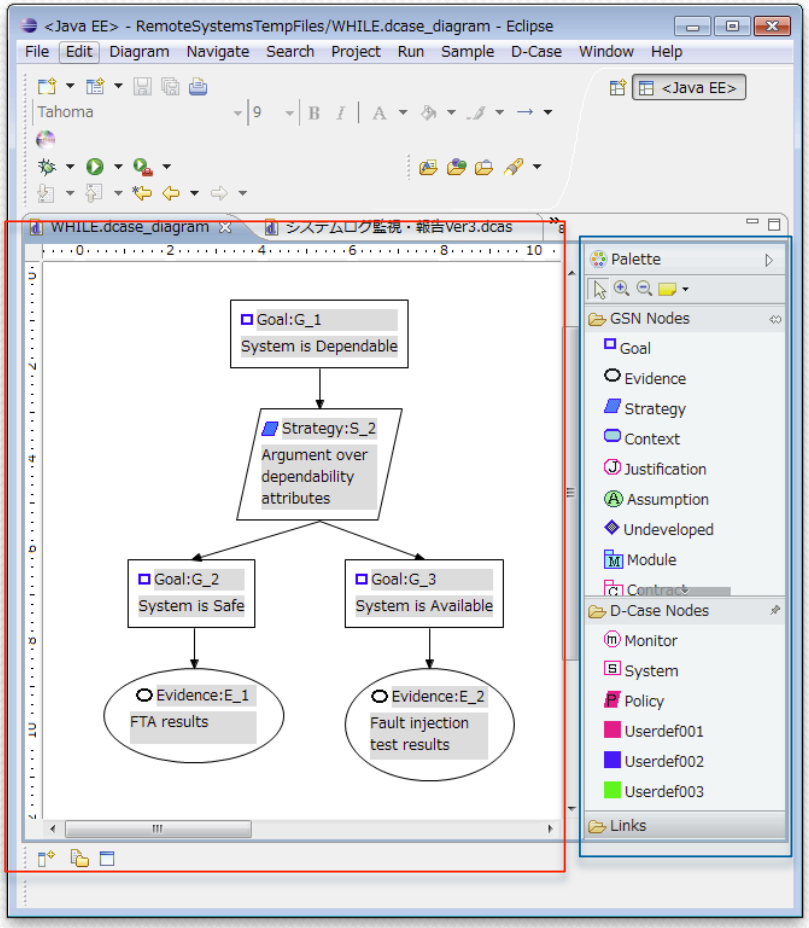
D-Case Editorについて

- JST CREST DEOS Projectで開発中の、D-Case/GSNを記述・編集するためのグラフィカルエディター
- EclipseのPlug-inとして開発されている
- 以下の機能を備えている
  - D-Case/GSN(Goal Structuring Notation)をサポート
  - D-Case/GSNパターンライブラリ、基礎的な変数型チェック機能
  - DEOSツールとの連携機能
    - D-Case/Agda
      - http://agda.cvs.gr.jp/agda/D-Case-Agda/
    - D-RE (DEOSのランタイムシステム)
    - システムモニタリング
      - http://web.sfc.keio.ac.jp/~jin/dm
    - DS-Bench/D-Cloud
      - http://www.dependable-os.net/tech/DSBenchDCloud/

© Dependability Science Lab. Nagoya Univ. 2012

D-Case Editorスナップショット

キャンバス

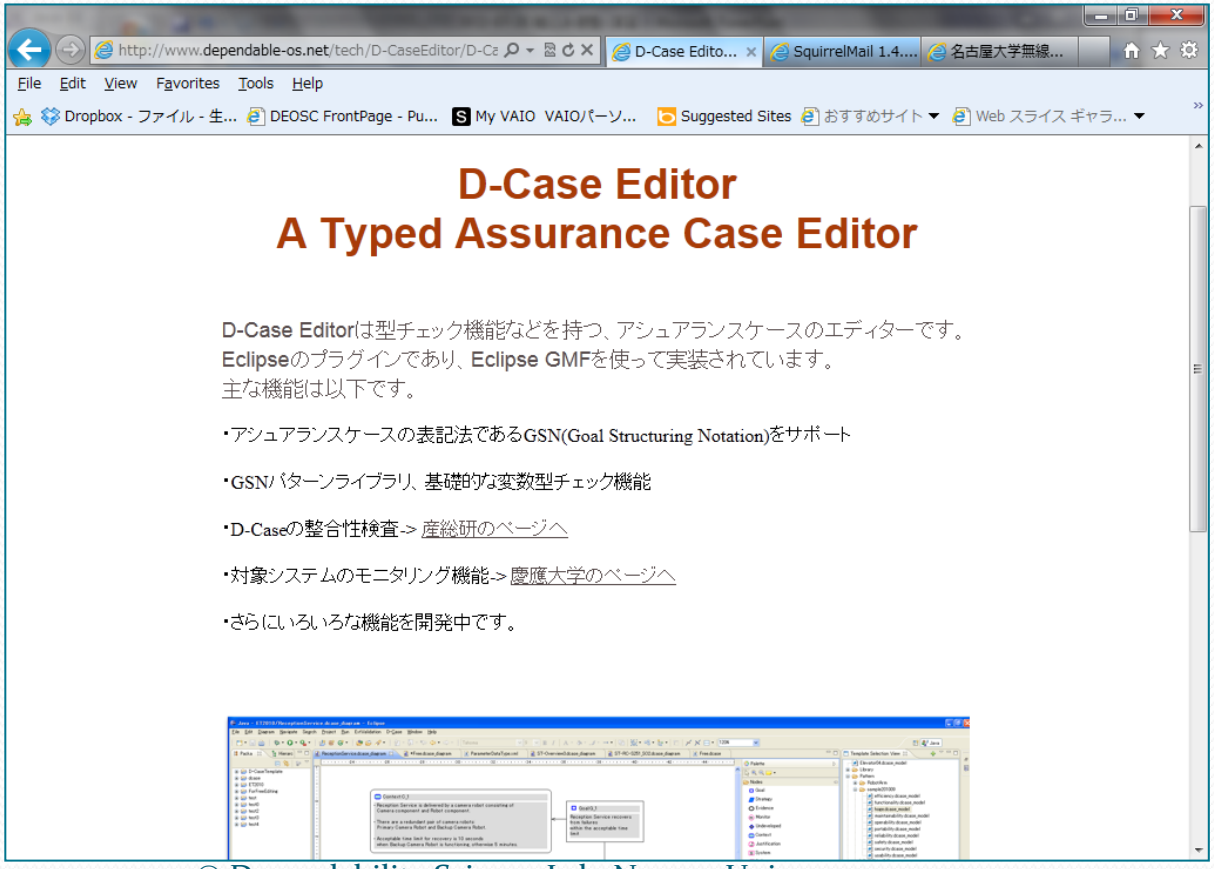


ノード
パレット

© Dependability Science Lab. Nagoya Univ. 2012

D-Case Editorの入手

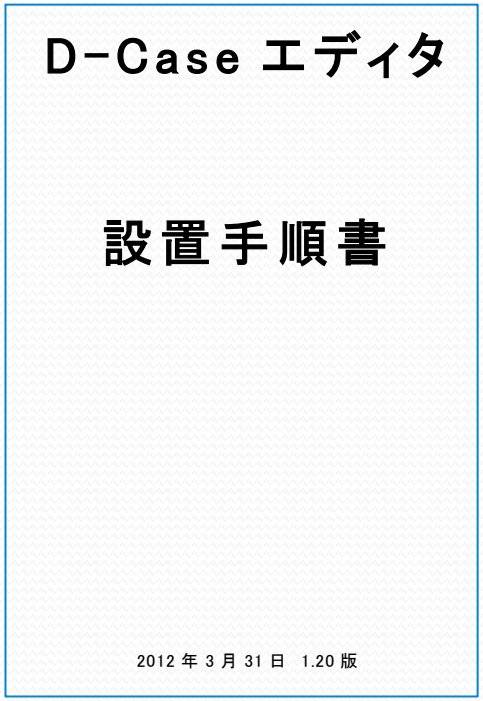
http://www.dependable-os.net/tech/D-CaseEditor/
よりダウンロードできる



© Dependability Science Lab. Nagoya Univ. 2012

D-Case Editorのインストール

- http://www.dependable-os.net/tech/D-CaseEditor/Download.html にある、インストール手順書にしたがってインストールする

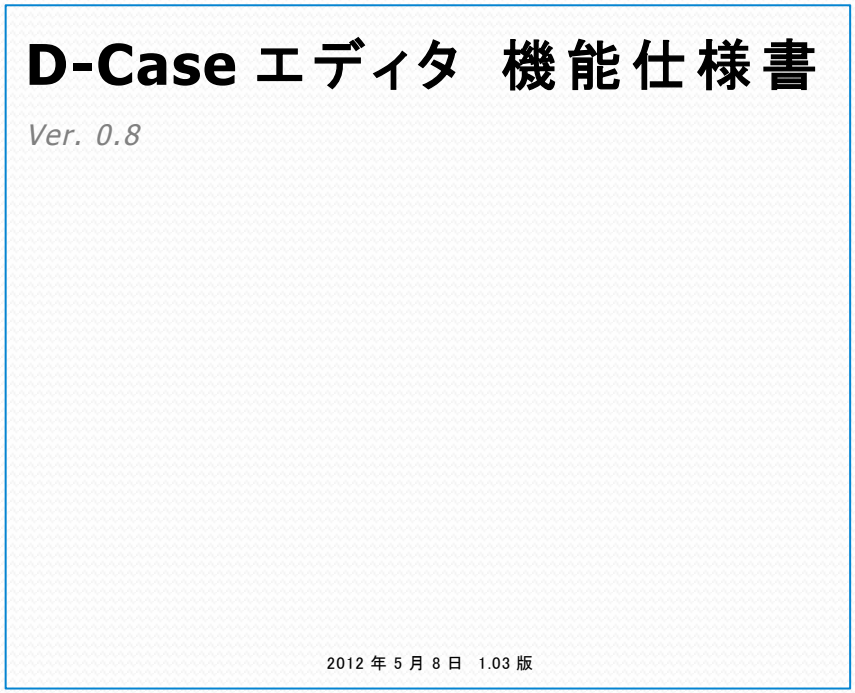


みなさん、事前に
インストールできましたか？
できてない方は、周りのひとに
教えてもらいましょう

© Dependability Science Lab. Nagoya Univ. 2012

D-Case Editorの利用

- http://www.dependable-os.net/tech/D-CaseEditor/Download.html にある、説明書(機能仕様書)を参照



© Dependability Science Lab. Nagoya Univ. 2012

D-Case Editor演習

本講義では、以下を
実際にD-Case Editorを
動かしながら説明する

- 最低限のEclipse機能、概念
  - Workspace, Save機能
- 基本描画機能
  - ノード追加、記述、リンク、パターン追加機能
- 知っていると便利な機能
  - 整頓、プリント、表変換機能
- 応用編・他のツールとの連携機能
  - D-Case/Agda
  - DS-Bench/D-Cloud

© Dependability Science Lab. Nagoya Univ. 2012

まとめ

- D-Case Editorのインストール方法、使い方を説明した
- Eclipse plug-inとして開発されているため、他のツールとの連携が取りやすい一方、インストール、使い方などに制約があるかもしれない
- 今後より使いやすく、かつ他のツールと連携をとりやすくしていく予定である
  - マルチプラットホーム化
  - ウエブベースの簡易版ツールも開発中

© Dependability Science Lab. Nagoya Univ. 2012