DEOS-FY2009-CM-01

Open Systems Dependability Core
– Dependability Metrics –

オープン・システム・ディペンダビリティ・コア
– ディペンダビリティメトリクス –

DEOS Core Team

平成21年9月4日

科学技術振興機構
Japan Science and Technology Agency

目 次

第 1 章 Introduction
はじめに **1**
1.1 Background 1
1.2 Goals 2
1.3 Approach 3

第 2 章 Dependability Metrics
ディペンダビリティメトリクス **6**
2.1 Qualitative Evaluation 6
2.1.1 Fault Source 7
2.1.2 Purpose Property 7
2.1.3 Life Cycle Phase 8
2.1.4 Target Functionality 8
2.2 D-Case 9
2.2.1 Overview 9
2.2.2 Grammar 11
2.2.3 Graph Structure 11

第 3 章 Evaluation and Estimation of Dependability
ディペンダビリティの評価と見積 **13**
3.1 Evaluation of Dependability 13
3.2 Estimation of Dependability 16

第 4 章 Summary
おわりに **19**

付 録 A Case Study
ケーススタディ **21**
A.1 System Overview 21
A.2 Requirements 24
A.3 Qualitative Estimation 24
A.4 Matching 25

概 要

This document is an interim report by the DEOS project on the metrics to evaluate the dependability of a system.

本書は、DEOS プロジェクトのディペンダビリティメトリクスに関する中間報告書である。

DEOS Core Team:

Toshihiro Hanawa	University of Tsukuba
Shinichi Miura	University of Tsukuba
Shuichi Oikawa	University of Tsukuba
Midori Sugaya	JST
Yutaka Ishikawa	University of Tokyo
Hajime Fujita	University of Tokyo
Toshiyuki Maeda	University of Tokyo
Motohiko Matsuda	University of Tokyo
Shinpei Kato	University of Tokyo
Yuki Kinebuchi	Waseda University
Kimio Kuramitsu	Yokohama National University
Hiroshi Yamada	Keio University
Tetsuya Yoshida	Keio University
Jin Nakazawa	Keio University
Yoichi Ishiwata	AIST
Hiroki Takamura	AIST
Yutaka Matsuno	AIST

DEOS コアチーム:

塙 敏博	筑波大学
三浦 信一	筑波大学
追川 修一	筑波大学
菅谷 みどり	科学技術振興機構
石川 裕	東京大学
藤田 肇	東京大学
前田 俊行	東京大学
松田 元彦	東京大学
加藤 真平	東京大学
杵渕 雄樹	早稲田大学
倉光 君郎	横浜国立大学
山田 浩史	慶應義塾大学
吉田 哲也	慶應義塾大学
中澤 仁	慶應義塾大学
石綿 陽一	産業総合技術研究所
高村 博紀	産業総合技術研究所
松野 裕	産業総合技術研究所

第1章　Introduction
はじめに

1.1　Background

背景

Computing devices have prevailed in our social infrastructures, making our lives more dependent on them. When we use trains, automobiles, or aircrafts, our vital state relies on their control and traffic systems. Malfunctioning of such systems could harm our lives. When we entrust banks or other financial service firms with our financial assets, their investment systems hold the outcome of our properties. If such systems falsely operate, our assets would be at risk. Thus, these systems require support to ensure, enhance or monitor certain characteristics. In this paper, we call such support as *dependability support*.

The quantity and the quality of support required by a system depend on its purpose. Automobiles, since they can dominate human lives, require a highly-reliable control system. Furthermore, for such a system may be difficult to update, system defects need to be diminished to the maximum extent in the development phase. Contrarily, television set-top boxes generally do not harm human lives or assets, and hence they may only need to moderately ensure failure detection or fast system restart. In addition, because they can be expected to be connected to Internet, functional

社会のいたるところに計算機が埋め込まれ、人間の生活の様々な局面が計算機に依存するようになっている。我々が電車や自動車、あるいは航空機を利用する際には、我々の生命がそれらの制御・運行システムに依存する。それらのシステムが開発者の意図と異なる挙動をすると、我々の生命に危機が及ぶ。また我々が銀行や証券会社に金銭を預託したり、投資を行ったりする際には、我々の金銭的財産がそれらの会社の運用システムに依存する。それらのシステムが誤動作をすると、我々の財産に危機が及ぶ。従ってこのような組み込みシステムには、特定の性質を保証、向上、あるいは監視するための支援が必要となる。本文書ではこうした支援を、ディペンダビリティ支援と呼ぶ。

あるシステムに求められるディペンダビリティ支援の質や量は、そのシステムの目的により異なる。自動車には、人の生命が依存するため、その制御システムには高い信頼性が求められる。また、システムのアップデートが困難であることから、開発時にシステムのバグを極限まで取り除く必要がある。これに対してテレビ放送用セットトップボックスでは、人間の生命や財産がそれに依存しているわけではないことから、運用時には障害の検知や高速な再起動などを、ある程度保証でき

improvement through system update can be expected.

As above illustrated, quantity and quality of dependability support required differ between products. Therefore it is necessary to discover the dependability support which a system provides, in order to understand the scope of the system.

1.2 Goals

We propose *dependability metrics* (or simply *the metrics* in this paper), which are metrics to quantitatively and qualitatively discuss the dependability support of operating systems installed in embedded devices. The purpose of the metrics is as follows: (1) allow developers, users, or evaluators of an operating system to describe the dependability support it can provide, (2) allow developers of an embedded device to estimate the dependability support it requires, (3) allow developers of an embedded device to compare and select which operating system to use, and (4) allow users of embedded devices to compare the dependability the devices possess based on the metrics. Thus, the targets of the metrics are operating system developers, embedded device developers, and embedded device users.

The metrics, with regard to embedded device developers, enable discussion of dependability support provided not only on device execution time, but also on its development and maintenance time. Therefore, we include testing, debugging, and analyzing tools as well as tools which control device execution, such as operating system kernels and kernel modules, as subjects of evaluation by the metrics. Furthermore, with regard to embedded device

ればよいと考えられる。またインターネットへの接続が期待できることから、保守時のシステムアップデートによる動作改善も期待できる。このように、製品が要求するディペンダビリティ支援は量的および質的に異なることから、あるシステムが充足しうるディペンダビリティ支援を明らかにすることは、そのシステムの適用範囲を知る上で必須である。

目的

組み込み機器に搭載されるオペレーティングシステムのディペンダビリティ支援を質的および量的に議論するための基準(以降、ディペンダビリティメトリクスあるいは単にメトリクスと呼ぶ。)を提案する。同メトリクスの目的は次の通りである。まず、オペレーティングシステムの開発者や利用者、あるいは評価者は、本メトリクスを用いて、そのオペレーティングシステムが提供できるディペンダビリティ支援を表明可能でなければならない。次に組み込み機器の開発者は、開発対象の機器が必要とするディペンダビリティ支援を見積可能でなければならない。開発者はまた、本メトリクスを用いて、採用候補オペレーティングシステムの比較検討を行える必要がある。最後に、組み込み機器の利用者は、類似する複数の組み込み機器が具有するディペンダビリティを、本メトリクスを用いて比較可能でなければならない。以上より本メトリクスは、オペレーティングシステム開発者、組み込み機器開発者、および組み込み機器利用者による利用を対象とする。

本メトリクスは、組み込み機器の開発者を念頭に置き、単に機器の実行時だけでなく、機器のライフサイクルにおける開発時や保守更新時に提供されるディペンダビリティ支援を含めて議論可能とする。従って、単にオペレーティングシステムカーネルやカーネルモ

users, we enable comparison of embedded devices using a coarsely classified grading system. The idea of this grading system is similar to that of the grading system of pencils (which have classifications such as H, HB, or B), or the grading system in educational institutes, which for example range from A to F (with trailing plus or minus signs for each of the alphabetical letters).

1.3 Approach

Dependability metrics, with methods described below, is able to exhaustively evaluate dependability support provided by an operating system. Figure1.1 depicts the concept. First, it categorizes mechanisms of the operating system to those that provide dependability support and those that do not. The dependability of the operating system as a whole is defined as the total of individual evaluations of mechanisms that provide dependability support. On evaluation, we set up various axes including time, space, or degree, and qualitatively evaluate the dependability support provided by a mechanism. On estimating dependability support required by a real system, we use similar axes to qualitatively evaluate the requirements. However, qualitative evaluation cannot prove if a subject operating system actually fulfills a certain attribute, nor can it clarify the performance indices required in real systems.

Given this fact, we complement the qualitative evaluation with benchmarks, fault injections, and other validation techniques. Operating system developers use GSN[1] to create relational graphs between entries included in qualitative evaluation axes and concrete values.

ジュールのような実行時の挙動が重要となる機構だけでなく、製品の開発時や保守更新時に重要となる検証、デバッグ、あるいは解析ツールなども、本メトリクスによる評価対象に含む。また、組み込み機器の利用者を念頭に置き、複数の機器を粗粒度の指標で比較できるようにする。これは、H、HB、あるいはBのように表現される鉛筆の濃さや、5段階や10段階評価の学業成績の発想に類似している。

手法

本メトリクスは、以下に示す手法により、オペレーティングシステムが提供するディペンダビリティ支援を網羅的に評価可能とする。図1.1に概念を示す。まず、オペレーティングシステムが提供する機構を、ディペンダビリティ支援を行うものとそうでないものの2つに分類する。オペレーティングシステム全体のディペンダビリティは、ディペンダビリティ支援を行う各機能を個別に評価した値を合算したものとする。その際、時間、空間、あるいは程度など様々な軸を設けて、評価対象機構が提供するディペンダビリティ支援の内容を定性的に評価する。また実システムにおけるディペンダビリティ要求の見積時には、同様の軸を用いて要求を定性的に評価する。ただし定性評価だけではオペレーティングシステムが、いずれかの属性を実際に満たしているかどうかや、実システムにおいて求められる具体的な性能指標等を扱えない。

そこで本メトリクスでは、ベンチマークやフォルトインジェクション、検証技術等による具体的な値により定性評価結果を補足する。オペレーティングシステム開発者は、GSN[1]を用いて定性評価軸に含まれる項目と具体的な値との関連づけグラフを構

We call this graph as *D-Case* (Dependability Case).

Next, operating system developers derive (quantify) dependability scores from D-Case, and visualize them on a plane based on 2 axes selected from above mentioned evaluation axes, or likewise on a space based on 3 axes. A dependability score is the maximum value of dependability support a mechanism of arbitrary granularity could provide. As the granularity, a single mechanism, multiple mechanisms combined (a configuration), or the operating system itself can be assumed. System developers are able to use these scores and the visualized information to compare different operating systems.

In addition, system developers can describe the dependability requirement estimates using the D-Case, and compare it to those of the operating systems. Consequently, they can obtain the dependability requirements fulfilled by developing on a certain operating system, and configurations (combination of dependability support mechanisms) to use whereat.

築する。このグラフを D-Case(Dependability Case) と呼ぶ。

次にオペレーティングシステム開発者は D-Case からディペンダビリティスコアを導出 (数値化) し、それを上述した評価軸から選択した 2 軸の平面、あるいは同様に 3 軸の空間に可視化する。ディペンダビリティスコアは、任意の粒度のディペンダビリティ支援機構が提供できるディペンダビリティ支援の最大値である。その粒度として、各支援機構、複数の支援機構の組み合わせ (コンフィグレーション)、あるいはオペレーティングシステム全体を考えることができる。システム開発者は、これらのスコアと、それを可視化した情報により、複数の異なる機構やオペレーティングシステム同士の比較が可能となる。

これに加えてシステム開発者は、ディペンダビリティ要求見積結果を D-Case を用いて表現し、それをオペレーティングシステムごとの D-Case と比較することができる。これにより、それらのオペレーティングシステムを用いてシステムを開発した際に満たされるディペンダビリティ要求と、それらを用いる際のコンフィグレーション (ディペンダビリティ支援機構の組み合わせ) を求めることができる。

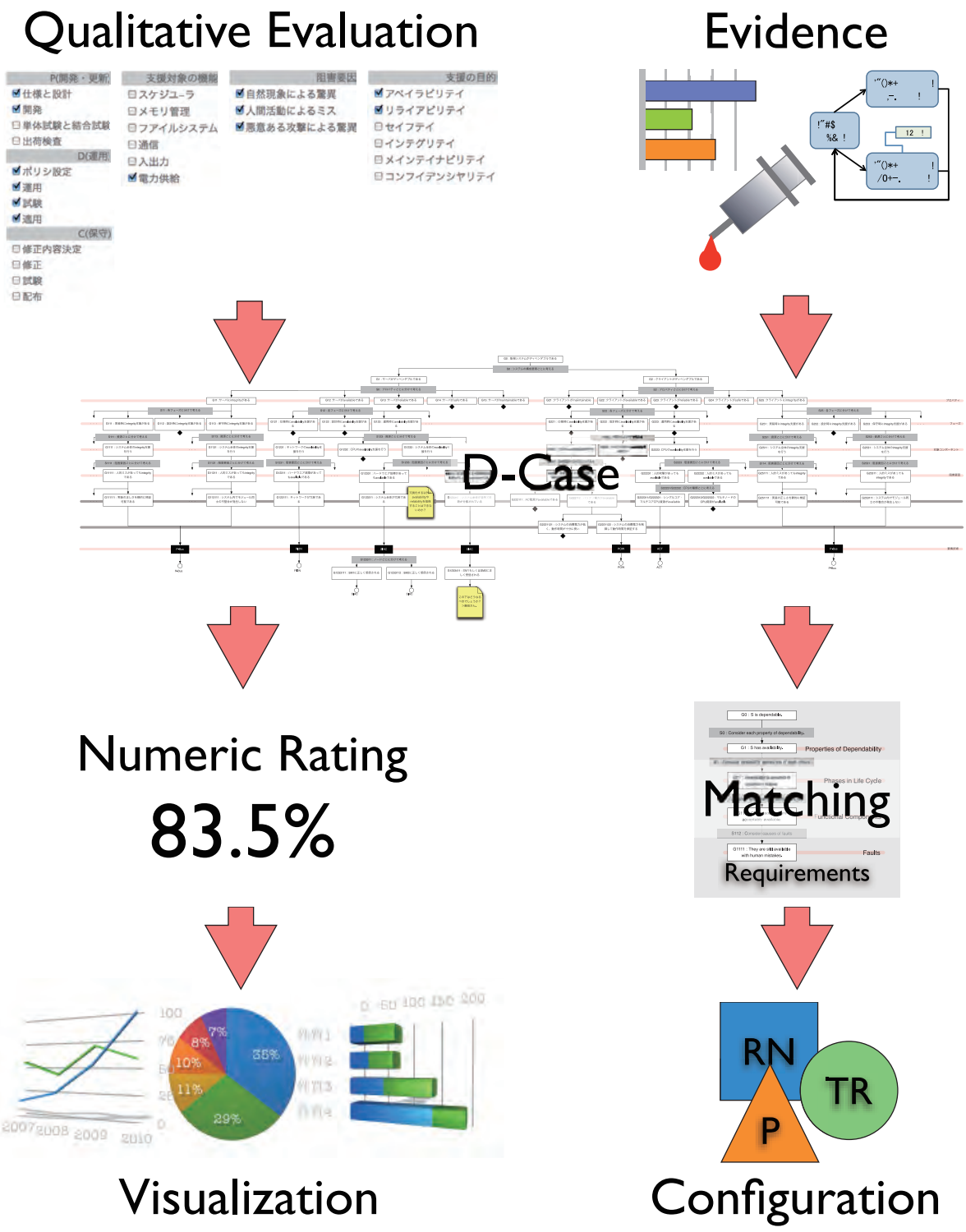

Figure 1.1: Concept

図 1.1: コンセプト

第2章　Dependability Metrics
ディペンダビリティメトリクス

Since dependability is a set of diverse characteristics, first we propose a qualitative evaluation method for evaluation and estimation of dependabilities. Operating system developers will verify dependability support tools or mechanisms in operating systems against qualitative evaluation items, and check which characteristics the subject satisfies. For example, if a dependability support mechanism is expected to enhance availability, availability is checked as one of its qualitative evaluation items. Next, we propose Dependability Case (D-Case), a Goal Structuring Notification (GSN) specialized for dependability evaluation. Using D-Case, operating system developers can add concrete evidences to the qualitative evaluation results using methods such as benchmarks, validations, or fault injections. System developers on the other hand, can request evidences such as a benchmark below a threshold, as a dependability requirement of operating systems.

ディペンダビリティは多様な性質の集合であることから、本文書では、ディペンダビリティに関する評価や見積りを行うための定性評価方式をまず提案する。オペレーティングシステム開発者は、ディペンダビリティ支援ツールや機構それぞれを定性評価項目と照合し、満たすと思われる性質をチェックしていく。例えば、あるディペンダビリティ支援機構が可用性を向上させると考えられるとき、その機構の定性評価項目のひとつとして、可用性をチェックする。次に本文書では、Goal Structuring Notification(GSN) をディペンダビリティ評価に特化させた、Dependability Case(D-Case) を提案する。D-Case を用いて、オペレーティングシステムの開発者は、そのオペレーティングシステムの定性評価結果に対し、ベンチマーク、検証、フォルトインジェクション等の結果により、定性評価結果の具体的根拠を与えられる。システム開発者は、ディペンダビリティ要求として、あるベンチマーク結果がある閾値未満であることなどの証拠を要求できる。

2.1　Qualitative Evaluation

Qualitative evaluation evaluates the effect of dependability support tools against multiple axes shown in table 2.1, including time, space, and target. We will explain each of the axes in the following sections.

定性評価

定性評価は、ディペンダビリティ支援機構やツールの効果を、表 2.1 に示す時間、空間、対象など複数の軸で評価するものである。以下の各項で、それらの軸を詳述する。

Table 2.1: Qualitative Evaluation Items

表 2.1: 定性評価項目

阻害要因	支援の目的	フェーズ	対象機能
自然現象	可用性 (availability)	仕様	CPU
人的ミス	信頼性 (reliability)	設計	メモリ
悪意ある攻撃	安全性 (safety)	実装・単体試験	ファイルシステム
ハードウエア故障	整合性 (integrity)	結合試験	通信
	保全性 (maintainability)	流通	入出力
		運用	電力
		保守・更新	システム全体
		廃棄・再利用	

2.1.1 Fault Source

First, it is important to clarify the problems dealt by dependability support mechanisms. For example, whether an operating system has a mechanism to avoid system failure in a harsh natural environment (e.g. where temperature and humidity change rapidly) or not, will affect the quality of the dependability support provided. We will regard the factors to which dependability support mechanisms take effect as the fault sources of dependabilities. Fault sources can be classified into those that reside within the embedded device, and those without. The former include software bugs inserted by human errors, or malfunctioning hardware, while the latter include natural hazards, or attacks from malicious programs. These categories can be further divided, but we will not do so to keep the metrics simple. Therefore, we will use the entries shown in table 2.1 as fault sources.

2.1.2 Purpose Property

In general, characteristics such as avail-

阻害要因

まず、ディペンダビリティ支援機構が扱う問題を明らかにすることが重要である。たとえばあるオペレーティングシステムが、温度や湿度など自然環境の急激な変化に適応して挙動を変更し、システムの不具合を避ける機構を有する場合とそうでない場合とでは、提供されるディペンダビリティの質が異なるといえる。従って本メトリクスでは、ディペンダビリティ支援機構が効果を発揮する原因を、ディペンダビリティ阻害要因という視点で捉えることとする。阻害要因は、組み込み機器の内部に存在するものと、外部に存在するものとに分類できる。内部要因には、人的ミスにより混入したバグや、ハードウエアの故障が含まれる。外部要因には、自然現象による脅威や悪意ある攻撃による脅威が含まれる。これらをより詳細に分類することも可能ではあるが、メトリクスの煩雑化を避けるためこれ以上の詳細化を行わない。以上より、本メトリクスでは、表2.1 に示す事項を阻害要因として扱うこととする。

目的性質

一般的に、可用性や信頼性といった概念

ability and reliability are regarded as components of dependability. Dependability support mechanisms which target different structures or functionalities may share a common purpose (e.g. enhancing reliability). By contrast, those that share a same target structure may have different purposes (e.g. availability and safety, respectively). These mechanisms complement each other. It is important to reveal the purpose of dependability support mechanisms to discover and investigate such relationships. We classify the purposes of dependability support mechanisms as shown in table 2.1. These classifications are widely acknowledged in researches concerning dependable systems[2].

は、ディペンダビリティという概念の一部を構成する性質であると考えられている。異なる構造や機能を対象とするディペンダビリティ支援機構同士が、同一の目的を (たとえば信頼性の向上など) を共有することがある。また、同一の構造を対象としながら、可用性や安全性など、目的が異なるものもあると考えられる。それらの機構同士は、相互補完的な関係である。このような関係を発見し、検討する上で、ディペンダビリティ支援機構の目的を明らかにすることは重要である。本メトリクスでは、表 2.1 に示す事項により、ディペンダビリティ支援機構の目的を分類する。これらはディペンダブルシステムに関する論文において広く参照されている概念 [2] である。

2.1.3 Life Cycle Phase

Some dependability support tools take effect even within development and maintenance/update time as opposed to only within system execution time. We call these steps in the system lifecycle as phases. Similar to the development of a system or constituting devices which has multiple processes such as design, implementation, and evaluation, each phase can be broken down into finer granularity. We cannot discuss the dependability support mechanisms which take effect in different phases together. Thus, we list the phases in which dependability support mechanisms take effect in table 2.1.

フェーズ

ディペンダビリティ支援やツールには、単にシステムの実行時だけではなく、開発時や保守・更新時に効果を発揮するものが存在する。システムのライフサイクルにおけるこれらの段階を、本稿ではフェーズと呼ぶ。また、システムやそれを構成する機器の開発自体にも、設計、実装、検査などの作業があるように、各フェーズはより細かな段階に分割して考えることができる。効果を発揮するフェーズが異なるディペンダビリティ支援機構同士を同列に議論することはできない。以上より本メトリクスでは、表 2.1 に示す項目により、ディペンダビリティ支援機構が有効に機能するフェーズを示す。

2.1.4 Target Functionality

The functionalities of an operating system targeted by dependability support mechanisms are diverse. For example, mechanisms which improve real-time properties of a system, resulting in enhanced availability and reliabil-

対象機能

ディペンダビリティ支援機構が支援の対象とする、オペレーティングシステム内の機能は多岐にわたる。実時間性を向上させてシステムの応答性を高め、結果として各時刻におけるシステムの可用性や信頼性を向上させる

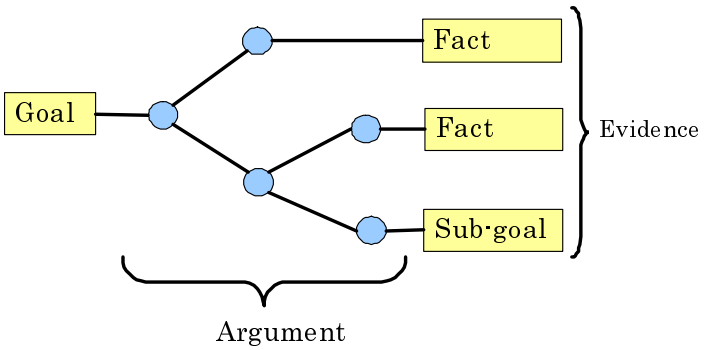

Figure 2.1: The Framework of Assurance Case

図 2.1: Assurance Case の枠組み

ity through quicker system response, can be regarded as targeting the scheduling functionality. Alternatively, a mechanism which enhances network reliability by guaranteeing a bandwidth of network communication by application programs, can be regarded as targeting the network functionality. We cannot discuss dependability support mechanisms which target different functionalities together. Thus, we clarify the functionalities to which the dependability support mechanisms take effect. We discuss around functionalities listed in table 2.1 albeit an operating system has numerous functionalities.

機構は、スケジューリング機能を対象とした機構であると言える。また、アプリケーションプログラムに対して、ネットワーク通信における一定の帯域を保証して、ネットワークの信頼性を向上させる機構は、通信機能を対象とした機構であると言える。異なる機能を対象とした複数のディペンダビリティ支援機構を同列に議論することはできない。そこで本メトリクスでは、ディペンダビリティ支援機構が効果を発揮する対象の機能を明確にする。一つの OS には非常に多数の機能が含まれているが、本メトリクスでは当面、表 2.1 に示す機能を支援の対象として扱うこととする。

2.2 D-Case

Our metrics use a framework called D-Case to make the discussion of qualitative evaluation more objective.

2.2.1 Overview

Qualitative evaluation items, when used in the evaluation of operating system dependability, are checked based on subjective decision

D-Case

本メトリクスでは、D-Case と呼ぶ枠組みを用いて、定性評価による議論をより客観的にする。

概要

定性評価項目は、それがオペレーティングシステムのディペンダビリティ評価に用いられた場合には、あるディペンダビリティ支援

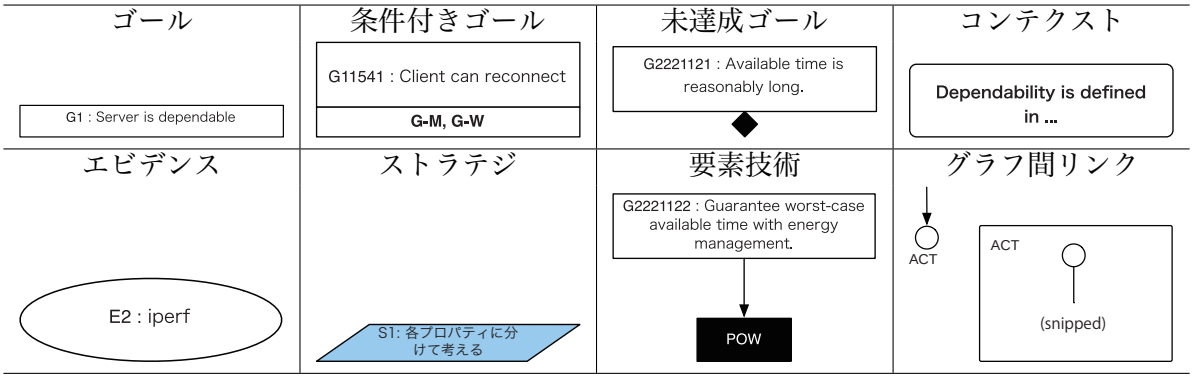

Figure 2.2: GSN Grammar of Dependability Metrics

図 2.2: 本メトリクスにおける GSN 表記法

whether a mechanism or a tool satisfies a requirement. When used in the estimation of dependability requirements by system developers, they are checked in the same manner, deciding whether the entry is actually needed. Our metrics complement this subjective decision with benchmarks, validations, and fault injections, as evidences of evaluation, or concrete guidance for estimation. This enables more objective and concrete evaluation or estimation. Our metrics accomplish this association using Dependability Case (D-Case).

D-Case is an application of a framework called Assurance Case[3]. Figure 2.1 depicts the framework of Assurance Case. It divides facts that need to be proposed to stakeholders into sub-goals, building the framework of the discussion at the same time, and ensures them using evidences obtained from experiments or axioms. Our metrics use the framework of Assurance Case to associate the results of benchmarks, validations, or fault injections to the qualitative evaluation or estimation, enabling them to be discussed more objectively.

機構やツールがそれぞれの項目を満たすかどうかという開発者の主観によって、またシステム開発者がディペンダビリティ要求見積りに用いる場合には、その項目が必要かどうかという主観によって、チェックの有無が決められる。本メトリクスでは、これらの主観に対してベンチマークや検証、フォルトインジェクションによる結果を、主観による評価の証拠あるいは見積りの具体値として対応付ける。これにより、客観的かつ具体的な評価や見積もりを可能とする。本メトリクスは、この対応付けを Dependability Case(D-Case) により可能とする。

D-Case は Assurance Case[3] と呼ばれる枠組みの応用である。図 2.1 に Assurance Case の枠組みを示す。Assurance Case では、ステークホルダへ提示したい事項を議論の枠組みを構築しながらサブゴールに分割していき、実験や公理等から得られるエビデンスによってそれらを保証する。本メトリクスでは Assurance Case の枠組みを用いて、ディペンダビリティ支援機構やツールに対する定性評価・見積り結果にベンチマーク、検証、フォルトインジェクション等の結果を事実として対応させる。これにより、評価や見積りの内

2.2.2 Grammar

Our metrics use *Goal Structuring Notation (GSN)*[1] to describe D-Case. We show the grammar adopted in our metrics in figure 2.2. A goal is what to be assured using D-Case, and is a proposition such as “operating system A is dependable“ or “product X is dependable“. A strategy is used to divide the goal into sub-goals, an element is a functionality to use in an operating system to accomplish a certain goal, and evidence includes experiment results, benchmarks (e.g. DS-Bench), or expert opinions to prove a preceding goal.

2.2.3 Graph Structure

We set restrictions on goal division and placement, regarding the composition of discussion (actual examples are illustrated in the next chapter) which uses the above mentioned grammar. To be precise, we compose D-Case from an 8 leveled tree. This allows operating system developers, system developers, and system users to discuss based on the D-Case which have a same structure. This is necessary for the comparison of operating systems by system developers, or the matching between dependability requirements and dependability evaluation of operating systems. The 8 levels within D-Case are as follows:

Level 1 is the final goal which states that the operating system itself is dependable, and is a target of discussion.
Level 2 divides a dependability into sub-goals. Corresponds to *purpose property*.

容をより客観的に議論可能とする。

表記方法

本メトリクスでは、*Goal Structuring Notation (GSN)*[1] を用いて D-Case を記述する。図 2.2 に、本メトリクスで採用する表記方法を示す。ゴールは D-Case を用いて示したいことで、例えばオペレーティングシステム A がディペンダブルである、製品 X がディペンダブルである、といった命題である。ストラテジは、ゴールをサブゴールに分けるときの考え方である。要素技術は、あるゴールを実現するときに用いる OS 中の要素技術を表わす。エビデンスは、その直前のゴールを実証するために用いる、実験結果、ベンチマーク（DS-Bench など）、エキスパートの意見などがくる。

グラフの構造

上述した表記を用いた議論の構成方法 (具体例については次章を参照のこと) について、本メトリクスでは、ゴールの分割法と配置法に関する制約を定める。具体的には、D-Case を 8 レベルの木により構成する。これにより、オペレーティングシステム開発者、システム開発者、およびシステム利用者が同一の構造を持つ D-Case に基づいて議論可能とする。このことは、システム開発者によるオペレーティングシステム間のディペンダビリティ比較や、システムにおけるディペンダビリティ要求とオペレーティングシステムのディペンダビリティ評価のマッチング等に必須である。D-Case における 8 つのレベルとは以下の通りである。

第 1 レベル そのオペレーティングシステムそのものがディペンダブルである、といった最終ゴールで、議論の対象である。
第 2 レベル ディペンダビリティをサブ性質

Level 3 divides level 2 sub-goals according to system or product life cycle. Corresponds to *life cycle phase*.
Level 4 divides level 3 sub-goals according to functionalities of devices which constitute the system. Corresponds to *target functionality*.
Level 5 divides level 4 sub-goals according to fault sources. Corresponds to *fault source*.
Level 6 and 7 divides level 5 sub-goals further. Has finer granularity than the qualitative evaluation entries.
Level 8 complements sub-goals up to level 7 with evidences from benchmarks, validations, or fault injections.

In the above list, level 1 through level 5 is mandatory. That is, evaluators of operating system dependability or estimators of system dependability need to divide goals into level 1 through 5, and place them in the GSN. However, they can annotate an unaccomplished goal node to the items not required, and abort the discussion. Furthermore, level 8 is also mandatory for operating system evaluators.

ごとサブゴールに分解して議論する。定性評価項目の「支援の目的」に該当する。
第3レベル 第2レベルのサブゴールを、システムや製品のライフサイクルにおけるフェーズごとのサブゴールに分割して議論する。定性評価項目の「フェーズ」に該当する。
第4レベル 第3レベルのサブゴールを、システムを構成する機器に含まれる機能ごとのサブゴールに分割して議論する。定性評価項目の「対象機能」に該当する。
第5レベル 第4レベルのサブゴールを、ディペンダビリティ阻害要因ごとのサブゴールに分割して議論する。定性評価項目の「阻害要因」に該当する。
第6,7レベル 必要に応じて、第5レベルのサブゴールをさらに細かいサブゴールに分割して議論する。定性評価項目よりも粒度が細かい。
第8レベル 第7レベルまでのサブゴールに対してベンチマーク、検証結果、フォルトインジェクション結果等のエビデンスを与える。

これらのうち、第1から5レベルは必須である。すなわち、OSのディペンダビリティ評価者、ならびにシステムのディペンダビリティ見積者は、第1から5レベルの項目を上記の方法で分割し、上記の方法でGSNに配置しなければならない。ただし、不要である項目には未達成ゴールノードを付与してそれ以上の議論を中止できる。また、OS評価者は、第1から第5レベルに加えて第8レベルも必須である。

第3章　Evaluation and Estimation of Dependability
ディペンダビリティの評価と見積

In this chapter, we will explain the dependability evaluation of operating systems and the dependability requirement estimation of real systems using dependability metrics.

本章では、ディペンダビリティメトリクスを用いた、オペレーティングシステムのディペンダビリティ評価と実システムにおけるディペンダビリティ要求見積について述べる。

3.1 Evaluation of Dependability

ディペンダビリティ評価

Dependability evaluations of operating systems concern qualitative evaluation of all of the constituting dependability support mechanisms and tools, and describing them as elements in D-Cases. In this section, due to the page space, we will only discuss those parts of DEOS D-Case which Accounting System (ACT) [4] of Sugaya et al. relates. Since DEOS is a work in progress, the elements in this section include hypotheses and assumptions. This evaluation result consist of investigations of [4] and interview to the first author of [4].

First, ACT can be qualitatively evaluated as a dependability support mechanism that supports availability (purpose property) of CPU, memory, and network (target functionalities) in the operation phase (life cycle phase) where human errors and malicious attacks (fault source) occur. Next, on top of this qualitative

オペレーティングシステムのディペンダビリティ評価は、それに含まれる全てのディペンダビリティ支援機構およびツールについて定性評価を行い、それら全てを要素技術として含む D-Case を記述することにより行う。本節では紙面の都合上、DEOS の D-Case において、菅谷らによる Accounting System (以降、ACT) [4] が現れる部分のみを抜粋して議論する。ただし DEOS は開発中であるため、本節に記載している事項には仮定および想定が含まれる。本評価結果は、[4] の調査および [4] の第一著者への取材などにより構成した。

まず、ACT は運用時 (フェーズ) の人的ミスおよび悪意ある攻撃 (阻害要因) の発生に際して、CPU、メモリ、ネットワーク (対象機能) の可用性 (目的性質) を支援するディペンダビリティ支援機構であると定性的に評価できる。次にこの評価に対して、ベンチ

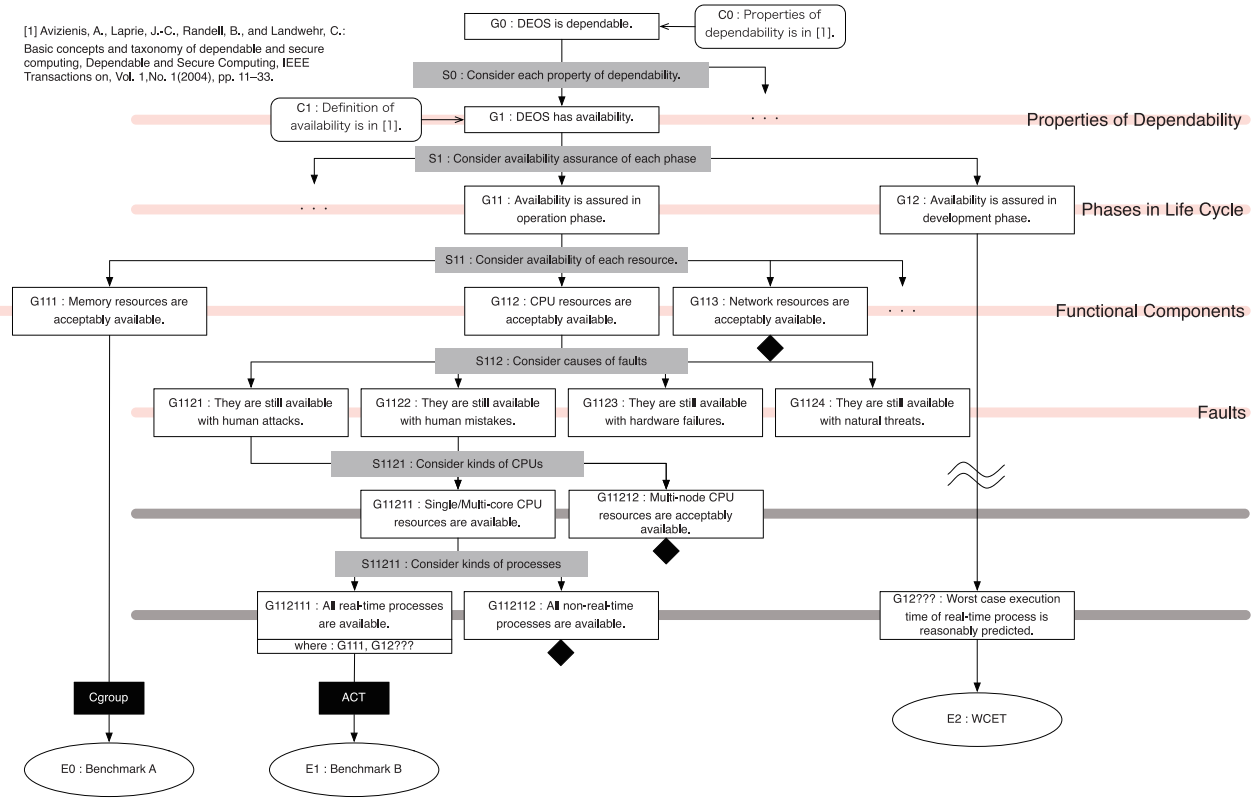

Figure 3.1: Dependability Case Related to Accounting System of Sugaya

図 3.1: 菅谷らの Accounting System が関わる dependability case の一部

evaluation, D-Case can be described to complement it with evidences such as benchmark results or fault injections. Figure 3.1 depicts part of a D-Case where CPUs are concerned. Evidences such as benchmark results are not shown in the diagram, since they are not collected at this point. In figure 3.1, the D-Case attempts to guarantee the availability of real-time processes using ACT, when memory resources are available, and worst case run-times are predictable (Cgroup is not part of DEOS, although it can handle similar mechanisms since it is bases on Linux). ACT is a mechanism to allow all of the processes to be executed as rapidly as possible, by restricting the use of CPU resources. According to information on CPU availability support by ATC we obtained from [4] etc., ATC

(1) covers single/multi core CPUs, but does not cover multi node CPUs.
(2) distinguishes real-time processes from non-real-time processes.
(3) contributes partially to availability of real-time processes.
(4) may miss temporal restrictions even if the CPU resources are guaranteed, if the memory resources are not. Delayed process executions due to unavailable memory (waiting for other processes to free them), may cause temporal restriction to be missed.

From 1 and 2, we can branch from **S1121** and **S11211**. From 4, we can tell that memory availability affects CPU availability. 3 explains the condition that ACT can only guarantee the availability of real-time processes when worst case run-times can be predicted. Thus, the subgoal regarding the availability of real-time pro-

マーク結果やフォルトインジェクション等の結果をエビデンスとして与える D-Case を記述する。図 3.1 に、CPU に関する部分のみを抜粋した D-Case を示す。ただしベンチマーク結果等は現時点でそろっていないため、図には掲載していない。図 3.1 では、メモリ資源が可用であり、かつ最悪実行時間を予測可能であるとき、実時間プロセスの可用性を、ACT を使って保証しようとしている (ちなみに Cgroup は DEOS の要素技術ではない。DEOS は Linux をベースにしているので、Cgroup のような Linux の要素技術も使える)。ACT は、CPU 資源の使用を制御することにより、CPU 資源があるプロセスに独占などされず、すべてのプロセスができるだけすぐに実行されるようにする要素技術である。ACT による CPU の availability 支援に関して、[4] などから以下の情報を得た。

(1) Single/multi core の CPU には対応しているが、multi node CPU はまだ対応していない。
(2) 実時間プロセスと非実時間プロセスに分けて考える。
(3) 実時間プロセスの availability に、部分的に貢献している。
(4) CPU 資源が保証されていても、メモリ資源が保証されていない場合にプロセスの時間制約が守れないといったケースはあるかもしれない。プロセスがメモリを取得できず（他のプロセスのメモリ解放待ち）実行が遅れ、その結果時間制約が守れない場合もある。

1,2 から、**S1121** と **S11211** による場合分けができる。4 から、メモリの可用性が CPU 資源の可用性に影響することがわかる。3 については、ACT が実時間プロセスの可用性を、最悪実行時間の予測を行える場合にのみ保証できるという前提について述べている。

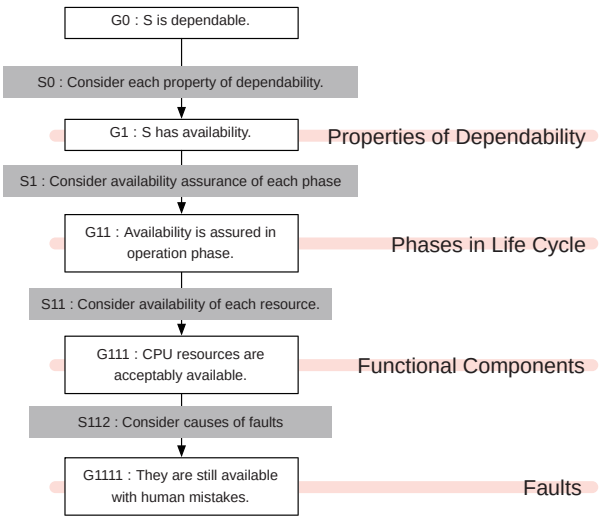

Figure 3.2: Requirement Estimate of System S

図 3.2: システム S の要求見積

cesses (**G112111**) depends on the sub-goal regarding memory availability (**G111**) , and the sub-goal regarding worst case run-time prediction (**G12???**).

以上より、実時間プロセスの可用性に関するサブゴール **G112111** は、メモリの availability に関するサブゴール **G111**、および最悪実行時間予測に関するサブゴール **G12???**に依存する。

3.2 Estimation of Dependability

System developers can use the metrics to describe dependability requirement estimates. Then, they can match them to operating system dependability evaluations, and find an operating system to adopt, and its composition (configuration). In this section, we call this process as matching. We assume that matching occurs in a following manner.

Assume an operating system α which has just the dependability supports shown in figure 3.1, and an operating system β which does not have a dependability support for availability. We will not show the dependability evalu-

ディペンダビリティ見積り

システム開発者は本メトリクスを用いて、システムにおけるディペンダビリティ要求見積りを記述する。そしてそれをオペレーティングシステムのディペンダビリティ評価結果と照合し、システムにおいて採用すべきオペレーティングシステムとその構成 (コンフィグレーション) を知ることができる。本稿ではこれをマッチングと呼ぶ。マッチングは次のような流れで行われることを想定する。

今、図 3.1 に示した内容のディペンダビリティ支援のみを含む架空のオペレーティングシステム α と、可用性に寄与するディペンダビリティ支援を含まない架空のオペレーティ

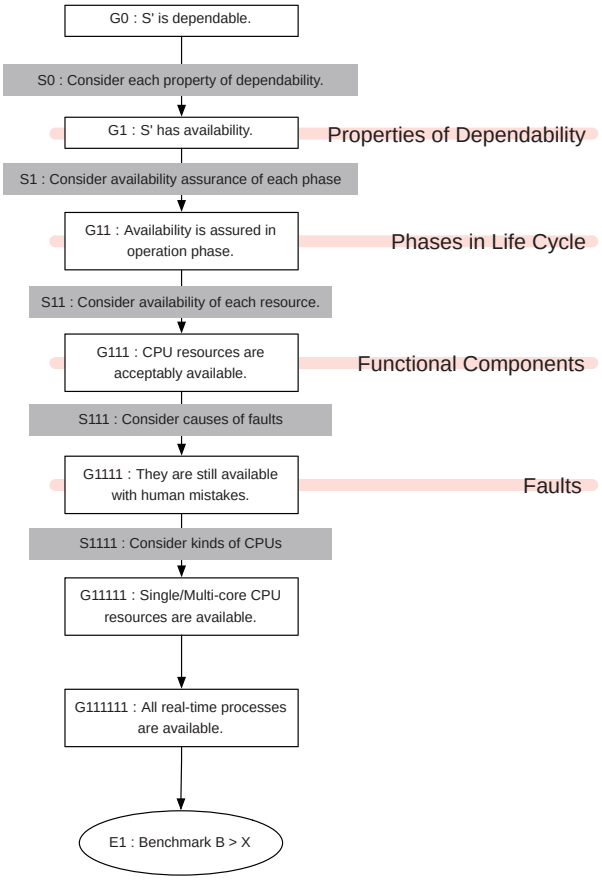

Figure 3.3: Requirement Estimate of System S'

図 3.3: システム S' の要求見積

ation result of β due to the page space. Developers of each operating system are to disclose the results of evaluation using our metrics.

Next, developers of system S and system S' are to estimate dependability requirements as described in figures 3.2 and 3.3 respectively. These estimates are simple compared to D-Cases that describe an operating system evaluation, but they are still D-Cases. The first requirement (requirement S) describes that the system needs to provide availability in an environment where human errors are assumed. The second (requirement S'), in contrast, describes more precisely that every process is required to record values above threshold X in benchmark B, on single/multi core CPUs. System developers can obtain following information from comparing these D-Cases to the evaluation results of both operating systems.

First, α fulfills both of the requirements if ACT is used, and the result of benchmark B is above threshold X. If not, α only fulfills requirement S. By contrast, β does not fulfill both of the requirements, since it does not provide availability. Here, using ACT in α is called a configuration. Actual requirements are more diverse than those in this scenario, and hence mechanisms used are also more diverse. Matching, as shown above, can determine which operating system to adopt, and which dependability support mechanisms or tools to use therein.

ングシステムβがあるとする。紙面の都合により、βのディペンダビリティ評価結果は示さない。各オペレーティングシステムの開発者は、それらを本メトリクスにより評価した内容を公開しているものとする。

ここで、システム S の開発者とシステム S' の開発者がそれぞれ図 3.2 と図 3.3 に示すディペンダビリティ要求見積りを行ったとする。これらは、オペレーティングシステムの評価を記述した D-Case と比較して非常に単純であるが、やはり D-Case である。前者の要求 (要求 S とする) は、人的ミスが想定される環境下でシステムが可用性を提供する、というように曖昧に与えられている。これに対して後者の要求 (要求 S' とする) は、前者に加えてシングルコア・マルチコア CPU 上で全ての実時間プロセスが、ベンチマーク B で閾値 X 以上の値を記録する必要がある、というように具体的に与えられている。システム開発者は、これらの D-Case を両 OS の評価結果を記述した D-Case と比較して、次の情報を得ることができる。

まずαは、ベンチマーク B の結果が閾値 X 以上であるならば、要素技術 ACT を使うことで双方の要求を満たす。そうでないならば、αは要求 S のみを満たす。これに対してβは可用性を提供しないので、双方の要求を満たさない。ここで、αで要素技術 ACT を使う、という情報をコンフィグレーションと呼ぶ。実際の要求は本シナリオより多岐にわたるため、使われる要素技術もより多い。従ってマッチングでは、採用すべきオペレーティングシステムとそこで利用すべきディペンダビリティ支援機構やツールを決定できる。

第4章　Summary
おわりに

This paper proposed *dependability metrics* which enables discussion of dependability support required by a system or provided by an operating system. The metrics consist of 4 types of qualitative evaluations, and *D-Case* which complements them with evidences including benchmark results. Our metrics extend GSN in order to clarify relationships between dependability support mechanisms/tools and goals. Furthermore, we put restrictions in separation and placement of the goals, allowing more exhaustive discussion on dependability. We plan to extend the metrics as follows:

1. The metrics need to enable on the fly dependability evaluation during system run-time. As a result, a feedback loop where the system users monitor the policies regarding dependability, and change the system behavior can be realized.

2. The metrics currently cannot analyze risks of fault sources. In order to enable comparison between different dependability support mechanisms, we plan to put a mechanism to measure the anticipated results of fault sources.

3. The metrics does not determine a method to quantify D-Case, which avoids quantitative

本文書では、システムが必要とする、あるいはシステムを構成する機器に搭載されるOSが提供するディペンダビリティ支援の量や質を議論するための基準(以降、ディペンダビリティ支援メトリクスあるいは単にメトリクスと呼ぶ。)を提案した。同メトリクスは、4種の定性評価項目と、それらをベンチマーク結果等の事実によりバックアップするD-Caseにより構成される。本メトリクスでは、ディペンダビリティ支援機構やツールなどの要素技術とゴールとの関係を明確化できるよう、既存のGSNを拡張した。またゴールの分割方法と配置方法に制約を設け、ディペンダビリティに関する議論をより網羅的に行えるようにした。また、本メトリクスに関して以下の拡張を検討している。

1. 本メトリクスをシステム動作中のリアルタイムなディペンダビリティ評価に利用できるよう、拡張する必要がある。システム利用者が指定したディペンダビリティに関するポリシをモニタリングして、その結果に基づいてシステムの挙動を変更するフィードバックループの実現を検討している。

2. 本メトリクスでは、ディペンダビリティ阻害要因が発現した際のリスク分析を行えない。ディペンダビリティ支援機構同士の重要度を比較できるよう、実システムにおける阻害要因の影響度合いを考慮できる仕組みを検討している。

evaluation. We will investigate on methods to calculate overall dependability support provided by an operating system, with effects of dependability mechanisms and secondary effects such as overheads in consideration.

Finally, we are developing a dependability benchmark that aims to quantitatively evaluate above mentioned effects and secondary effects. Refer to “DEOS Benchmarks“ on this subject. Furthermore, refer to “DEOS Configurations“ for discussion on quantity and quality of dependability support (dependability configurations) required by embedded systems with different compositions or purposes.

3. 本メトリクスでは D-Case を定量化する方式を決定していない。そのため、ディペンダビリティ支援の量的な評価を行えない。今後、各ディペンダビリティ支援機構がもたらす効果とそのオーバーヘッド等の副次効果から、オペレーティングシステム全体のディペンダビリティ支援の量を算出する方式を検討する。

最後に、上記の効果や副次効果を定量的に測定することを目的とした、ディペンダビリティベンチマークの開発を行っている。これについては、別稿「DEOS Benchmarks」を参照されたい。また、構成や目的の異なる組み込みシステムがそれぞれ必要とするディペンダビリティ支援の量や質 (ディペンダビリティコンフィグレーション) に関する議論については、別稿「DEOS Configurations」を参照されたい。

付 録A　Case Study
ケーススタディ

We will illustrate the dependability estimation and the matching of real systems using a demo system, which will be introduced in the DEOS mid-term meeting scheduled on 4th of September 2009, as an example (the demo system described is being designed/developed, and may differ at the actual meeting). Figure A.1 shows the overview of the demo system in design/development.

A.1　System Overview

Figure A.1 shows a system to emulate industrial monitoring systems. Client machines on the left (CM1, 2, and 3) monitor using cameras. The recorded image is sent to the server machines (SM1, 2) and is stored. At the monitoring client on the right (SH), the images are manually monitored. Following DEOS technologies in development are planned to be used in this demo system:

ACT [4] guarantees the processing ability of multiple processes through CPU consumption restriction. Actually, it restricts CPU work load of face recognition software which runs in the camera clients, and guarantees tasks such as image transfers. It enhances reliability/availability through resource optimization.

SIAC [5] realizes unification of duplex servers and fail over on a server failure. It provides

2009 年 9 月 4 日に行われる、DEOS プロジェクト中間成果報告会で公開されるデモシステムを例として、実システムのディペンダビリティ見積もりとマッチングのイメージを示す (以下で示すデモシステムは設計・開発中であり、実際に発表されるシステムと異なる可能性がある)。設計・開発中のデモシステムの概要を図 A.1 に示す。

システムの概要

図 A.1 は、産業用監視システムをエミュレーションするシステムである。左にある、3 台のクライアントマシン (CM1, 2, 3 とする) は、カメラで監視を行い続ける。記録した画像は中央のサーバー群 (SM1,2 とする) に送り、保存する。また右にある監視用クライアント (SH とする) では、画像を人的に監視する。このデモシステムでは、以下の DEOS で開発中の要素技術が使われる予定である。

ACT [4] CPU 使用量制御による複数プロセスの処理性能を保証する。具体的には、カメラクライアントで動作する顔認識ソフトの CPU 負荷を制限し、画像送信等の実行を保証する。資源最適化による信頼性・可用性の向上が狙いである。

SIAC [5]　二重サーバの一元化と片側故障時のフェールオーバーを行う。負荷分散（CM からのデータを、SM1 と SM2 のうち負荷の

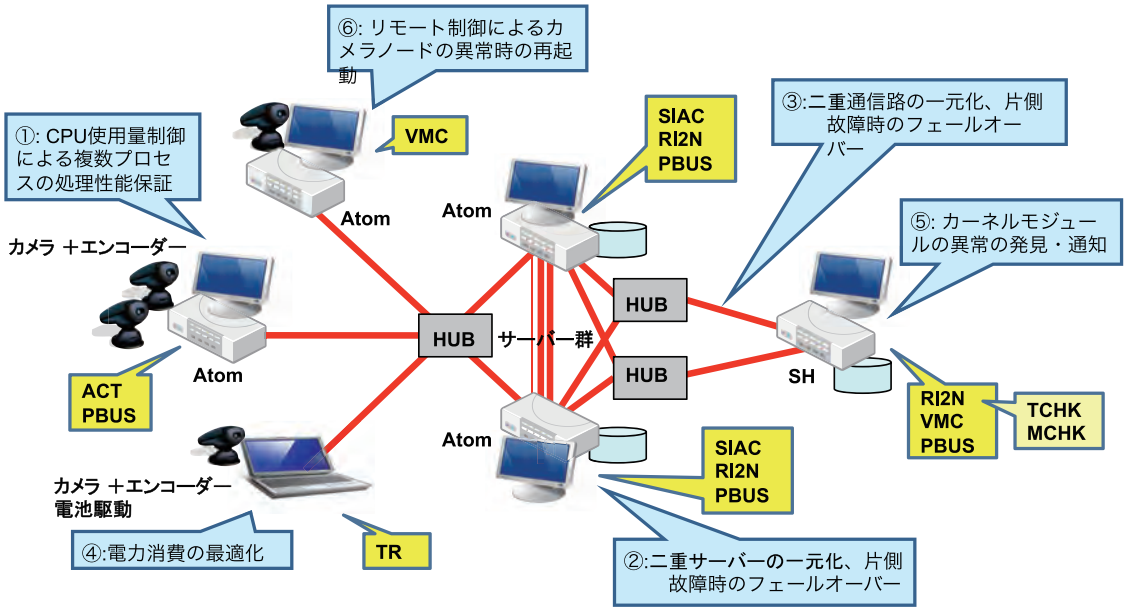

Figure A.1: An Emulation System of Industrial Monitoring Systems

図 A.1: 産業用監視システム・エミュレーションシステム

load balancing (sending data from CM to either SM1 or SM2 with lower work load) and robustness (switching the destination when one of the SM which receives data from CM fails). It provides dependability support through redundancy.

RI2N [6] realizes unification of duplex communication channels and fail over on a channel failure. Actually, it multiplexes the communication channel between SM and SH, and while regularly using both the channels, enables communication on a single channel in case of a failure. It provides dependability support through redundancy.

TR [7] guarantees the run-time of only the primary processes, for energy conservation. Actually, running in the camera client powered by batteries, it controls power of devices such as LCDs according to conduction of image checking. It provides dependability support through

低い方へ送る）および、耐故障機能（CM からデータを受け取る SM のうち一方が故障した場合、他方へ受信を切り替える）を提供する。冗長化によるディペンダビリティ支援が狙いである。

RI2N [6] 二重通信路の一元化と片側故障時のフェールオーバーを行う。具体的には、SM と SH の間の通信経路を多重化し、通常時は両方の経路を使い、一方に障害が生じたとき他方で通信を継続できるようにする。冗長化によるディペンダビリティ支援が狙いである。

TR [7] 電力節約のために、主なプロセスだけ実行時間を保証して、消費電力を落とす。具体的には、電池駆動のカメラクライアントにおいて、画像の確認を行う場合と行わない場合とで、LCD 等のデバイスの電源を詳細に制御して動作時間を長くする。資源最適化によるディペンダビリティ支援が狙いである。

resource optimization.
P-Bus [8] realizes safe attachment/detachment of kernel modules. ACT, SIAC, RI2N, and TR are all provided as P-Bus components (P-Component). It provides dependability support through abstraction of the kernel structure. However, we will not discuss it here, since it targets operating system mechanism developers.
VMC conducts remote control of the system from failure detection of the components. Actually, it runs in CM1 or SH, and reboots the guest OS from the host OS, when the guest OS stops due to a failure. Moreover, it removes a P-Component of unverified version from the kernel when it is attached. It provides dependability support through virtualization.
TCHK,MCHK [9][10] verifies P-Component (RI2N) through type/model checking. It provides dependability support through verification. However, we will not discuss it here, since it targets operating system mechanism developers.

In the following sections, we will describe the process to derive a constitution of above mentioned DEOS technologies (configuration) for the development of industrial monitoring systems using dependability metrics. System developers who develop industrial monitoring system using embedded OSs, compares the evaluation of DEOS or other OSs using the metrics, with the estimation of required dependabilities. As a result, they can decide on which OS to adopt. Since DEOS is currently under development, we will simulate the use of the metrics instead of conducting a real evaluation in which DEOS would be compared to other OSs.

P-Bus [8] カーネルモジュールの安全な抜き差しを実現する。ACT, SIAC, RI2N, TR はすべて P-Bus コンポーネント (P-Component) として提供される。カーネル構造の抽象化によるディペンダビリティ支援が狙いである。ただし、P-Bus は OS 要素技術開発者に対するものなので、ここでは扱わない。
VMC 構成要素の異常検出によるシステムのリモート制御を行う。具体的には、CM1 もしくは SH において、ゲスト OS が障害により停止したとき、制御 OS からゲスト OS を再起動する。また、不適切なバージョンの P-Component が挿入された際にそれをカーネルから排除する。仮想化技術を活用したディペンダビリティ支援が狙いである。
TCHK,MCHK [9][10] 型チェック・モデルチェックによる P-Component（RI2N）の健全性の検証を実現する。検証技術によるディペンダビリティ支援が狙いである。ただし、P-Bus は OS 要素技術開発者に対するものなので、ここでは扱わない。

以下では、産業用監視システムの開発のために、ディペンダビリティメトリクスを用いて、上記の DEOS の要素技術の組み合わせ（コンフィグレーション）を導出するプロセスを示す。今回のデモシステムは、DEOS のデモであるので初めからこれらの DEOS 要素技術を使うことが決まっている。組み込み OS を使って産業用監視システムを作る開発者は、本メトリクスによる DEOS や他の OS の評価と、システムに必要なディペンダビリティの見積りとを比較する。結果として、DEOS や他の OS のいずれを採用するかを決める。ただし、現段階では DEOS は開発中であるため、他の OS との比較を行う実際の評価ではなく、本メトリクスの使用例を示すためのシミュレーションを行う。

A.2 Requirements

The demo system is assumed to record high resolution images and to store the large scale data semi-permanently in a server. Current surveillance cameras generally have VGA (640x480) or QVGA (320x240) resolutions, but we will assume requirements to record higher resolution images intended for example use in areas with high crime rate, where image recognition of criminal faces or number plates may be needed. Moreover, we aim for the system to scale to the number of cameras. A developer of a monitoring system may set the requirements as follows:

1. In CM2, processing time for high resolution images need to be ensured, and future increase in numbers of cameras need to be managed.

2. SM needs to recover from failures in order not to lose important images. It needs to provide high availability by managing both overall/inner server communication failures.

3. Surveillance cameras need to operate for long period of time, but the power supply may fall short. Hence, high efficiency of power usage is required.

4. CM needs to provide safe software update in the future, and remote system reboot on failures.

要件

デモシステムは、高精度の画像を記録し、それらの大規模なデータを半永久的にサーバーに保存することを想定したものである。現在の監視用のカメラは VGA（640 × 480）あるいは QVGA（320 × 240）程度の解像度であることが多いが、たとえば犯罪多発地域を監視するために、犯人の顔や使用した車のナンバーなどがはっきりわかる高精度な画像を記録できるような監視カメラを使うための要件を考える。また、多くの場面を記録できるようにカメラの数を増やしても大丈夫なようにしたい。そのために以下の要件を、監視システム開発者が考えたとする。

1. CM2 において、高精度画像処理時間を確保し、将来監視カメラの追加を保証したい。

2. SM は、重要な画像を残しそこなわないように、故障しても、すぐに復旧できるようなものにしたい。サーバ全体のレベルと、サーバ内部レベルの回線の切断などの障害両方に対処できるようにして、高い可用性を持たせたい。

3. 監視カメラを長期間運用したいが、監視場所によっては十分な電力が得られないかもしれない。そのために、電源効率の性能をあげたい。

4. CM は、将来の安全なソフトウエアアップデート、並びに障害発生時の再起動を遠隔から行えるようにしたい。

A.3 Qualitative Estimation

We will now conduct a qualitatively estimation to meet above mentioned requirements. We will describe each of the axes of the metrics corresponding to the requirements.

1. Processing time for high resolution im-

定性的見積

これらの要件をみたすための定性評価を行う。上の要件に対して、本メトリクスの各軸の該当項目を示す。

1. 高精度画像処理時間を確保したい。

阻害要因： DOS アタック (悪意ある攻撃) や

ages need to be ensured
Fault Source: DOS attack (malicious software), software bug (human error)
Purpose Property: availability
Life Cycle Phase: operation
Target Functionality: CPU (image encoding and transfer are real-time processes, while image display is a non-real-time process)

2. High resolution images always need to be stored.
Fault Source: hardware failure
Purpose Property: availability
Life Cycle Phase: operation
Target Functionality: overall system (server)

3. Energy efficiency of surveillance cameras need to be enhanced.
Fault Source: hardware failure
Purpose Property: availability
Life Cycle Phase: operation
Target Functionality: power

4. Surveillance cameras need to be controlled remotely.
Fault Source: stops due to remaining software bugs (human error), version mismatch of P-Component (human error)
Purpose Property: reliability
Life Cycle Phase: operation, maintenance/update
Target Functionality: overall system (client)

A.4 Matching

The developer of the surveillance system describes above mentioned requirements using D-Case, and compare them to the D-Case of DEOS. They will obtain D-Cases shown in fig-

プログラムのバグ (人的ミス) などが考えられる。
支援の目的： 可用性
フェーズ： 運用
対象機能： CPU （画像のエンコーディング、送信は実時間プロセス, 画像の表示は非実時間プロセス)

2. 高精度画像データを常に保存できるようにしたい。
阻害要因： ハードウエア故障
支援の目的： 可用性
フェーズ： 運用
対象機能： (サーバ) システム全体

3. 監視カメラの電源効率をよくしたい。
阻害要因： ハードウエア故障
支援の目的： 可用性
フェーズ： 運用
対象機能： 電力

4. 監視カメラを遠隔から監視したい。
阻害要因：残バグによる停止 (人的ミス)、P-Component のバージョンミスマッチ (人的ミス)
支援の目的： 信頼性
フェーズ： 運用、保守・更新
対象機能： (クライアント) システム全体

マッチング

監視システム開発者は、上述した要件を D-Case を用いて記述し、DEOS 全体の D-Case と比較する。これにより、監視システム開発者は、図 A.2 および図 A.3 に示す D-Case を

ures A.2 and A.3 as products. These are obtained by matching the D-Case of the requirements with that of DEOS, and by extracting elemental technologies to be used. Furthermore, the two D-Cases combined is the D-Case of the overall surveillance system. The system, since it is designed after the elemental technologies to be used are determined, all of the dependability requirements will naturally be fulfilled. Following information can be obtained from these D-Cases:

1. ACT contributes to the CPU availability during operation. **2.** SIAC contributes to the availability of overall system (server) during operation. **3.** RI2N contributes to the availability of communication during operation. **4.** TR contributes to the availability of power during operation. **5.** VMC contributes to the reliability of the overall system (client) during operation.

The composition of elemental technologies as illustrated above, is the configuration of DEOS which fulfills the dependability requirements of the system.

得ることができる。これらは、要件の D-Case を DEOS 全体の D-Case に重ね合わせ、使用すべき要素技術を抽出した物である。また、これらを合体した物が、監視システム全体のディペンダビリティ要求である。本システムは、使用する要素技術を決定してから策定したため、そのディペンダビリティ要求は必然的に全て満たされている。これらの D-Case より以下の情報を得ることが出来る。

1. ACT が、運用時における CPU の可用性に貢献する。**2.** SIAC が、運用時における、(サーバ) システム全体の可用性に貢献する。**3.** RI2N が、運用時における、通信の可用性に貢献する。**4.** TR が、運用時における、電力の可用性に貢献する。**5.** VMC が、運用時における、(クライアント) システム全体の信頼性に貢献する。

以上の要素技術の組み合わせが、本システムのディペンダビリティ要求を満たす DEOS のコンフィグレーションである。

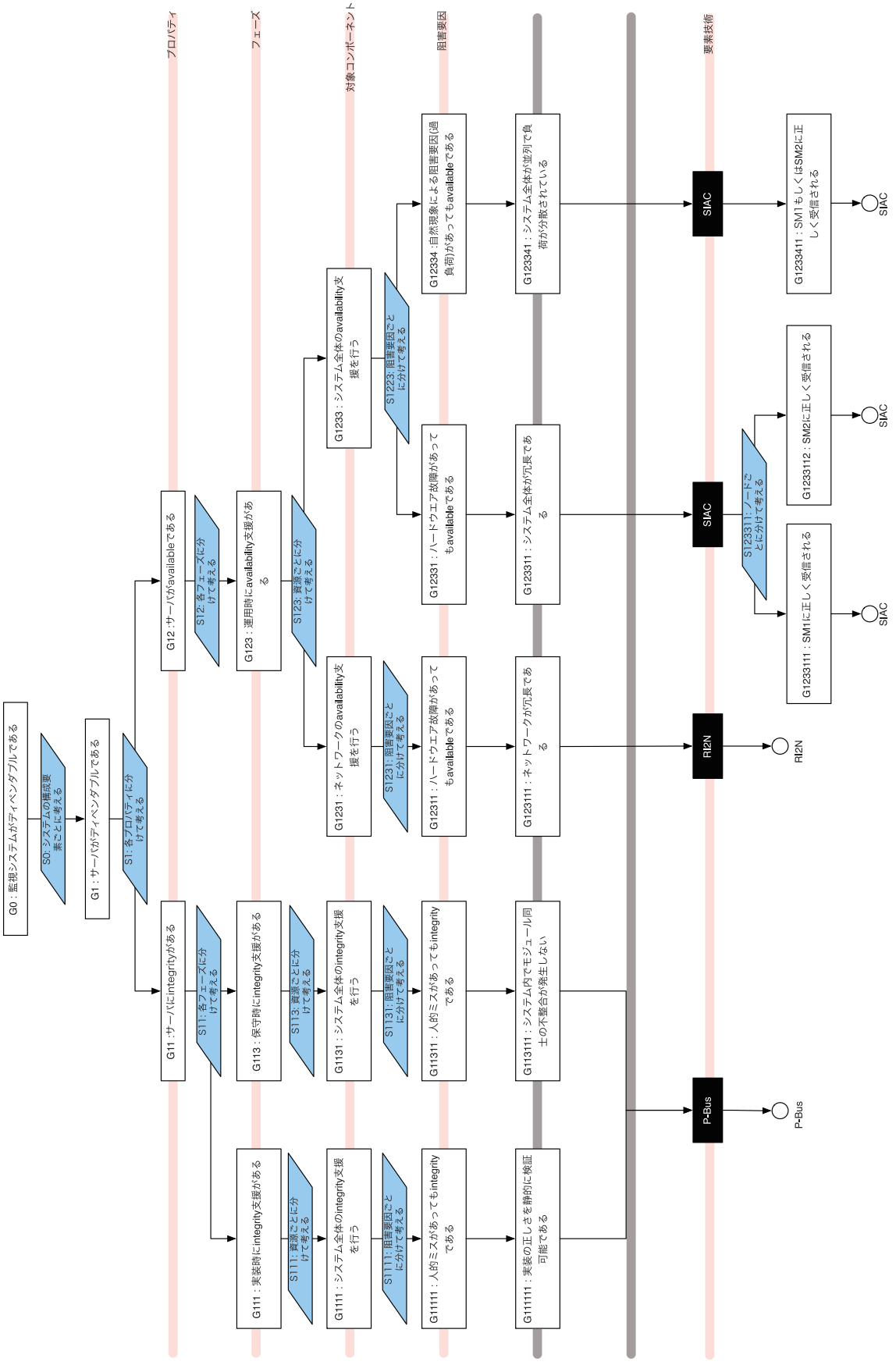

Figure A.2: The Requirement Estimate of The Surveillance System (Server) and The Result of Matching

図 A.2: 監視システム (サーバ) のディペンダビリティ要求見積とマッチング結果

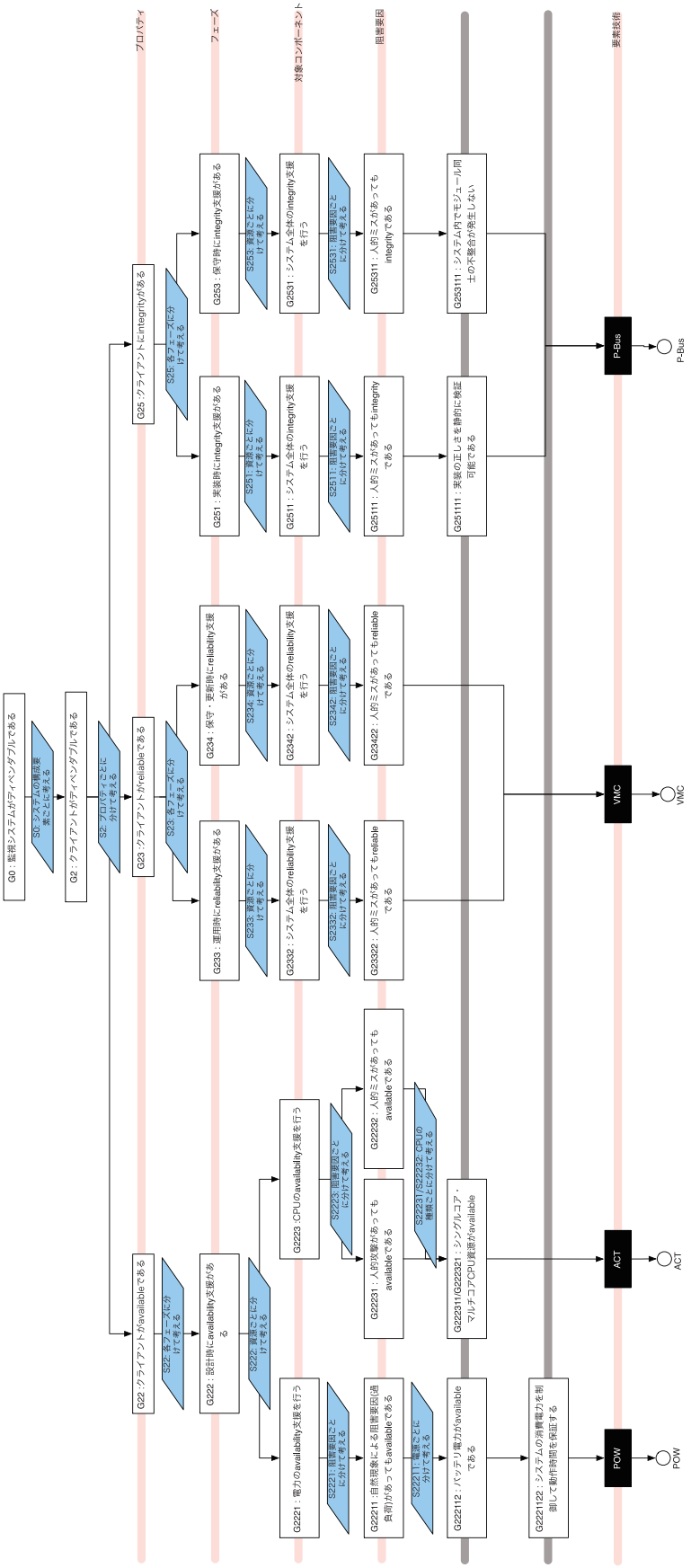

Figure A.3: The Requirement Estimate of The Surveillance System (Client) and The Result of Matching

図 A.3: 監視システム (クライアント) のディペンダビリティ要求見積とマッチング結果

関連図書

[1] Tim Kelly and Rob Weaver. The goal structuring notation - a safety argument notation. In *Proc. of the Dependable Systems and Networks 2004, Workshop on Assurance Cases*, 2004.

[2] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11–33, Jan.-March 2004.

[3] Keishi Okamoto and Makoto Takeyama. An approach to assurance cases. In *proceedings of Dependable System Workshop (DSW'09)*, 2009.

[4] Midori Sugaya, Shuichi Oikawa, and Tatsuo Nakajima. Accounting system: A fine-grained cpu resource protection mechanism for embedded system. In *Proc. of ISORC*, pages 72–84, 2006.

[5] Hajime Fujita, Hiroya Matsuba, and Yutaka Ishikawa. Tcp connection scheduler in single ip address cluster. In *CCGRID*, pages 366–375, 2008.

[6] Shin'ichi Miura, Toshihiro Hanawa, Taiga Yonemoto, Taisuke Boku, and Mitsuhisa Sato. RI2N/DRV: Multi-link Ethernet for high-bandwidth and fault-tolerant network on PC clusters. In *The 9th Workshop on Communication Architecture for Clusters in IPDPS*, May 2009.

[7] Masto Maori, Junichi Yura, Jin Nakazawa, and Hideyuki Tokuda. P-survive: Process level energy reservation for networked sensing systems. In *5th International Conference on Networked Sensing Systems (INSS2008)*, pages 242–245, June 2008.

[8] Yutaka Ishikawa. P-bus specification, 2008.
http://www.il.is.s.u-tokyo.ac.jp/deos/pbus/spec/.

[9] Motohiko Matsuda, Toshiyuki Maeda, and Akinori Yonezawa. Towards design and implementation of model checker for system software. *Future Dependable Distributed Systems, Software Technologies for*, 0:117–121, 2009.

[10] Takahiro Kosakai, Toshiyuki Maeda, and Akinori Yonezawa. Compiling c programs into a strongly typed assembly language. In *ASIAN*, pages 17–32, 2007.