Dependability Standards and our Challenge to Establish a New Standard for Open Systems

Toshinori Takai, Hiroki Takamura
AIST, JAPAN
Contents

• Dependability Standards Related Organizations
  – ISO/IEC JTC1/SC7/WG7, IEC TC56, TC65a, OMG,…
  – Current states and activities

• Our challenge and approach to new standards
  – Our concepts
  – Activity of ISO/IEC 15026
    (System and software assurance)

• Relationship among other talks in DEOS Project
Dependability Standards Related Organizations

- IEC60300 Dependability management
  - IEC TC56/WG4
- IEC61508 Functional safety of electrical/electronic/programmable electronic safety-related systems
  - IEC TC65A
- Notation of Assurance case
  - OMG
- ISO/IEC15288 System life cycle processes
  - ISO/IEC JTC1/SC7/WG7
- ISO/IEC12207 Software life cycle processes
- ISO/IEC 15026 System and software assurance
- D-31 General requirements for software controlled measuring instruments
  - OIML TC5/SC2
Dependability Standards and Organizations

- **Dependability management** (IEC TC56)
  - IEC 60300 series
- **System and Software assurance** (JTC1/SC7/WG7)
  - ISO/IEC 15026
- **System/SW lifecycle processes** (JTC1/SC7/WG7)
  - ISO/IEC 15288, 12207
- **Functional safety** (IEC TC65a)
  - IEC 61508, (ISO26262)
    - Safety Integrity Level (SIL)
- **Quality management** (JTC1/SC7/WG6)
  - ISO/IEC 25000 series
- **Security** (JTC1/SC27)
  - ISO/IEC 15408, 27001
Dependability Standards and Organizations

- **IEC TC56**: Dependability
  - Reliability and Maintainability (1965) ⇒ Dependability (1990)
  - Historically it started reliability of electric parts
    Hardware ⇒ + System and Software
  - FTA, FMEA, HAZOP

- **JTC1/SC27** IT Security Techniques
  - ISO/IEC 15408 Common criteria, 27001 Security management

- **JTC1/SC7** Software and Systems Engineering
  - WG6 Quality Management
    - Software quality requirement and evaluation (SQuaRE) 25000s
    - ISO/IEC 9126: product quality, 14598: product evaluation
  - WG7: Lifecycle Processes
    - ISO/IEC 15288, 12207
    - ISO/IEC 15026 system and software assurance
What is Dependability?

- RAS (1960s)
  - Reliability,
  - Availability,
  - Serviceability
- RASIS (1970s)
  - RAS + Integrity + Security
- Dependability definition in IEC TC56
  - collective term used to describe the availability performance and its influencing factors reliability performance, maintainability performance and maintenance support performance

- In Avizienis et paper
  - Focus on Closed system
  - Fault-Error-Failure model
  - Means
    - Fault prevention
    - Fault tolerance
    - Fault remove
    - Fault forecasting
  - Adaptability, Usability, Manageability are mentioned that importance but not be considered enough
Definitions of Dependability

• Definition in Avizienis paper
  – *the ability to deliver service that can justifiable be trusted*
  – *the ability of a system to avoid service failure that are more frequent or more severe than is acceptable*

• Definition in IEC TC56
  – **Now** (IEC 60300-1) *collective term used to describe the availability performance and its influencing factors reliability performance, maintainability performance and maintenance support performance*
  – Software dependability
    ability of the software to perform as and when required when integrated in system operation
    (Guidance on software aspects of dependability 56/1349A/CD)
  – (Network) dependability
    ability to perform as and when required to meet specified communication and operational requirements
    (Methodology for communication network dependability assessment and assurance 56/1351/NP)
Current activities

- JTC1/SC7/WG7(System and Software lifecycle)
  - ISO/IEC 15026 are changing the subject, System integrity levels to System assurance
- IEC TC56(Dependability management)
  - Most generic standard IEC 60300-1,2 need to revise by considering to today’s social needs. Definition of dependability will be reconsidered
  - Risk management are renewal the new standard ISO31000s. TC56 is working within the activity and revised IEC 60300-3-9 to ISO31010 (now final voting are done)
  - Lifecycle costing problem is still considered (IEC 60300-3-3) and further studies about software reusability issues for eco-friendly products (Dependability of Software products containing reused components NP/1332)
- These organizations need to adapt and revise the standards from stand-alone (Closed) system to complex, distributive and networked systems (we call these Open systems)
Our Definition of Open Systems Dependability

• We understand that complex, huge distributive and networked systems are Open systems

• Incompleteness and uncertainty could be factors that may result to failures in the future, and they are inherent to embedded systems (Factors of Open Systems Failure).

• Definition: Open Systems Dependability is to remove the said factors before they cause failure, to provide appropriate and quick action when they occur, to manage the failure in order to minimize the damage, and to safety and continuously provide the services expected by users as much as possible
Open Systems Dependability in practice

- Dependability is the degree of **Accountability**
- Accountability is secured by showing *evidences* of having done and doing **Risk Management in best effort**
  - to provide *expected services continuously*,
  - to manage quickly and properly *to minimize damages* when an incident occurs
  - to take countermeasures *never to let the same incidents occur again*
- Provide satisfactory service continuously as and when required, even if system failure happened, protect stakeholders (security, privacy, safety, … = 安心: Anshin)
Our Challenge to Open Systems Dependability

- **Dependability** ≜ **Accountability** + **Risk management**
- **Accountability**
  - Construct a **System profiling** methods
    - Scenario-based, Agent-based, Aspect-oriented analysis
  - **Stakeholder analysis**
    - Roles and Responsibility
      - sharing the responsibility, common-ownership of responsibility
      - Beyond Service level agreement (SLA), SLA is not enough
  - Define the **Consensus building process** among stakeholders (**Assurance case**)
- Define the **Decision-making process** under uncertainty and incompleteness
  - **Risk Management in best effort**
    - Risk communication (**Assurance Case**)
  - Standardization for Accountability (risk management with evidence) with Dependability Level which is determined by system profiling
Our Challenge to Open Systems Dependability

- **Dependability ÷ Accountability + Risk management**
  - **Accountability**
    - Construct a System profiling methods
      - Scenario-based, Agent-based, Aspect-oriented analysis
    - Stakeholder analysis
      - Roles and Responsibility
        - sharing the responsibility, common-ownership of responsibility
      - Beyond Service level agreement (SLA), SLA is not enough
    - Define the Consensus building process among stakeholders (Assurance case)
    - Define the Decision-making process under uncertainty and incompleteness

- **Risk Management in best effort**
  - Risk communication (Assurance Case)
  - unidentified risk

- Standardization for Accountability (risk management with evidence) with Dependability Level which is determined by system profiling
Standardizing Open Systems Dependability

- **Construct Evidence based framework**
  - Define the framework to provide the evidence for Accountability
- **Construct a System profiling method**
  - Who are stakeholders in the system? ⇒ **Stakeholder analysis**
  - Harmonize among dependability attribute
    (including trade-off analysis)
  - Aspects of assets
- **Define the Consensus building process** among stakeholders by **Assurance Case**
- **Define the Decision-making process** under uncertainty and incompleteness
- **Define the improve process** and these assessment
- **Define the Business continuity plan**
  - Contingency plan
- Roughly speaking, we will define the new **specification language**
  which is clear, no ambiguity, readable and compact not only for stakeholders but other persons who can verify the specification documents.
Stakeholders Analysis

• Identify the stakeholders
• Define stakeholders requirements
• Roles and responsibility,
  – sharing the responsibility,
    common ownership of responsibility
• Define the consensus-building process among stakeholders (Assurance case)
  – Risk communication
  – Beyond SLA
• Define the decision-making process
Summary

• Challenge to establish new standard for Open Systems
  – Define Stakeholder Analysis
  – Define Consensus-building process among stakeholders
    Assurance Case
  – Construct Evidence based framework
    Define the framework to provide the evidence for Accountability
  – Define the improve processes and these assessment
  – It is hard to understand entire behavior of an Open system. The structure, interactions, system boundary change dynamically. However, it is necessary to keep trying to understand the system through the lifecycle (System Profiling).
  – Define Risk management for Open systems
    Decision-making process under uncertainty and incompleteness
Assurance Case
ISO/IEC 15026

Toshinori Takai
(summary of the talk by Takamura-san)

Why we focus on assurance cases?

– For Open Systems, all risks cannot be known in advance
  → we must treat unidentified risks

– One solution is agreement between stakeholders
  → assurance case
assurance case

... not take fire because ...

... can cook foods ...

(usual) specification

developer

convince

agree

agree

convinced

acquirer

agree

agree

understand

design

(usual) specification

Claim

argument

Evidence

safe

one
Definition and Role of Assurance Case (AC)

- **Our definition:**
  
  A *documented body* of *evidence* which provides a convincing and correct *argument* that a system adequately satisfies the specified *claim* for a given application in a given environment.

- **Role:**
  
  - to give framework for reasoning and showing properties about systems
    - framework for evidences
    - information for decision making
  
  - risk communication tool
    - to convince the stakeholders of the claim which the AC argues for
An Example of Assurance Case

The microwave oven does not take fire under the condition that it is used only for foods.

The hardware is the same one of the previous model of this product. The software will almost be rewritten but the specification is well-written in formal way and it is developed using formal methods.

Certification which states there have been no significant accident of the previous model for ten years.

Some reasoning may seem not reasonable but AC is for agreement or decision making, both may include subjectivity.

Process model and plan for software development using formal methods.

Formal software specification.
Background: Safety Case

Most countries make efforts to continuously review and update the safety case (safety analysis report, procedures and other relevant technical documentation)


A software safety case, which justifies the suitability of the software development process, tools and methods, is required for software of all integrity levels

UK Defence Standard: Requirements for safety related software in defence equipment, 00-55(PART 1)/Issue 2, 1997

Dependability Cases

- **Dep Case**: G. Despotou, T. Kelly. Extending the Safety Case Concept to Address Dependability. ISSC 2004

- **D-Case**: developed in DEOS project
  - tomorrow talk

- “Assurance case” is a general word for *-lity cases
  - safety case
  - dependability case
  - security case
  ...
Communities for AC
(Assurance Cases)

• International Working Group on Assurance Cases
  – Regular Members
    City University, London; UIUC; SEI; Adelard; University of York;
    CERT; Swedish Nuclear Power Inspectorate; CEC JRC, Ispra;
    CMU; DSTL (Defense Science and Technology Laboratory, UK);
    Mitre; NATO; CDA...

• ISO/IEC JTC1/SC7/WG7: ISO/IEC 15026
  – 1998 version is only for system integrity levels
    • integrity level of IEC61508 is an instance
  – currently, major revision process is running
    • 15026 Part1: Concepts and vocabulary
      – Technical Report (will be published soon)
    • 15026 Part2: Assurance case
      – Y. Kinoshita is a co-editor
    • 15026 Part3: System integrity level
      – T. Takai is a co-editor
    • 15026 Part4: Assurance in the life cycle
      – Y. Kinoshita is a co-editor

• OMG
  – notation
Characteristics of Standards proposed by JTC1/SC7/WG7

- Focus on sharing the concepts and providing the best practice
  - Unlike ISO9000, ISO14000, IEC61508 (those are the standards for conformance assessment)
  - main aim is NOT the conformance assessment
- c.f. ISO/IEC 12207 and 15288 are the standard for sharing the concepts about each process in software/system lifecycle

**Integrity Level (IL)**

- **Def. (integrity level):** A denotation of a range of values of a property of an item necessary to maintain system risks within tolerable limits

---

<table>
<thead>
<tr>
<th>Frequency of Occurrence</th>
<th>Indicative Frequency (per year)</th>
<th>Severity of Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequent</td>
<td>$&gt;1$</td>
<td>Catastrophic</td>
</tr>
<tr>
<td>Probable</td>
<td>$1 - 10^1$</td>
<td>High</td>
</tr>
<tr>
<td>Occasional</td>
<td>$10^{-2}$</td>
<td>High</td>
</tr>
<tr>
<td>Remote</td>
<td>$10^{-3} - 10^4$</td>
<td>Intermediate</td>
</tr>
<tr>
<td>Improbable</td>
<td>$10^{-4} - 10^6$</td>
<td>Low</td>
</tr>
<tr>
<td>Incredible</td>
<td>$&lt;10^{-6}$</td>
<td>Trivial</td>
</tr>
</tbody>
</table>

**Table 2 - Example Risk Matrix**

<table>
<thead>
<tr>
<th>Risk Class</th>
<th>System Integrity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>A</td>
</tr>
<tr>
<td>Intermediate</td>
<td>B</td>
</tr>
<tr>
<td>Low</td>
<td>C</td>
</tr>
<tr>
<td>Trivial</td>
<td>D</td>
</tr>
</tbody>
</table>
Is Safety Integrity Level (SIL) Enough?

• Probabilistic characterization is not FIT for Software failure (including unidentified risk)
  – For instance, SIL 4 (the highest degree); Formal methods (FM) is Highly recommended in Software production
  – Define (Traceable) Evidence showing what and how to use FM in SW lifecycle

• Make Consensus-building based on the Evidence among Stakeholders

⇒ Assurance Case
Observation: AC and Open system

1. For **identified risks**, claims in AC are given in terms of IL
   - argument is important
   - the main role of AC is to show that a system has a certain IL
   - decision can be made **automatically**

2. For **unidentified risks**, claims in AC cannot be given in terms of IL
   - justification of the top level claim is important
   - the main role of AC is to make clear that what claim has to be considered
   - decision has to be made by **human being**

AC was introduced for open systems (?) (maybe without intention)
Problems on “Tree-Based” Assurance Case

- **Problem 1**: “dividing” dependability is not obvious
  - reductionism does not work effectively
  - top-down analysis does not matched to dependability
- **Problem 2**: Each system has to have its own dependability attributes (availability, safety, etc)
  - Even if attributes are common, each system has its own dependence relation between those attributes
- **Problem 3**: Need to provide a framework to modify the top level claim
  - when new threat is identified
  - the purpose of a system can also be changed
A new component of an AC (JTC1/SC7/WG7 version)

• **Claim** about an attribute or property of the system (usual)
• **Support** which can be either *Evidences*, (e.g., based on established scientific principles and prior research), *assumptions*, or *sub-claims*, derived from a lower-level sub-argument (usual)
• **Argument** showing how the evidence supports the claim, which can be deterministic, probabilistic or qualitative (usual)
• **Justification** which justifies the choice of claim (proposed by Japan national body of JTC1/SC7/WG7)
Definition of justification

- A **justification** consists of data and explanation from which one can infer how the claims are chosen and why e.g.
  - results of risk assessments
  - results of requirement analysis and
  - explanation how these results lead to the choice of the top level claim

- proposed by Japan national body

Justification explains how the claim contributes to a property of “super” system, e.g. a business or an organization which uses the system
An Example of Justification

The microwave oven does not take fire under the condition that it is used only for foods.

- Almost all significant accident of microwave ovens is to take fire
- The use of the product can be restricted by user-manual

If an accident not related to fire (i.e. catch a finger by the door), the developer does not need to bear any responsibility.
Possible structure of Justification

To record the reason of the choice of the top level claim in AC

→ When a new threat is identified the top level claim can be changed
Duality between risks and evidences

Known risks
identified by some stakeholders

Evidences
which convince any stakeholders

Claim = a representation of dependability which can be understood and convinced by related stakeholders
Tool?
Future Plan: to propose Dependability Process View

- ISO/IEC12207/15288 do not define “right” or “wrong” lifecycle process, but define concepts and vocabulary
  - to define criteria for evaluating lifecycles, deciding a point of view is necessary
- Process View: in an system/software lifecycle, in order to achieve some specific goal, to select relating processes from 12207 or 15288 and to give necessary activities for their processes, e.g. usability process view
Observation: What’s new on AC?

Usual specification can be seen as a document produced by agreement between **acquirer** and **developer**.

**AC** is a special case of usual specification?

Specification is “shown” by implementation.

Claim is shown by evidences.

Implementation of constraint (e.g. assertion or implementation of “functional safety”)

Assumption or evidence of environment
It is hard to show that some program code has some property relating to **dependability**

- **formal specification languages** will play an important role for traceability of dependability
Summary

• **Our observation**: for open systems, unidentified risks have to be considered
• **Our focus**: one solution is *agreement* between stakeholders
  – assurance case (AC) can be a tool for those agreements
• **Activities for AC**:  
  – ISO/IEC JTC1/SC7/WG7 is working for standardizing AC and  
  – DEOS project is now developing an implementation of AC, called D-Case (appeared in the following talks)
• **Our proposal**:  
  – Unlike cases for specific attribute, (e.g. safety case, security case) justification plays an important role  
  – Formal specification languages for dependability is important for traceability of dependability
Relationship among other talks in DEOS Project

- **Elemental Technology**
  - P-Bus and DEOS verification tools: Dr. Ishikawa
  - Composition Kernel for multi core dependable embedded systems: Dr. Nakajima

- **Architecture**
  - Evidence based Computing and its Demo: Dr. Yokote and Dr. Kuramitsu

- **Process and Management**
  - Dependability Metrics and D-case: Dr. Nakazawa and Dr. Matsuno
  - Dependability Standard for Open Systems: Takai and Takamura
  - Open Systems Dependability Concept: Dr. Tokoro
資料など
課題

• システムのステークホルダーは誰か明示し、規格化
  – 誰が何をするのか（役割）
  – 責任はどこにあるのか（責任分担・責任共有）
  – 利害関係者間の合意形成
  – 情報の共有と情報の流れ
  – リスクコミュニケーション

• どのフェーズでもエビデンスを提示する機構の実現
  – D-fopsでの実装

• オープンシステムには、システム境界を変化による影響分析を実施

• リスク分析の徹底（回避・転嫁・軽減・受容）

• 緊急時（不確定要因）への対応：コンテンツジェンシープラン
課題

・ 生物としてシステムをとらえる方法論の確立
  - 創発し階層を超える活性をどう記述するのか？
  - 部分の総和≠全体
  - 自己治癒・内科的処置・外科的処置に対応する技術
    - Self-adaptability, Self-monitoring
    - Policy based reconfiguration
    - Property based isolation

・ システム評価は稼動中と廃棄後の両面で
  - 資産の観点の導入
    - 後継システムに何を残したのか？
    - 資産には再利用可能なコードのみならずプロジェクト管理法など
オープンシステムに対して
-不完全・不確定には-

• 発展・改善プロセスをどうするのか？
  – 誰がどこで実施するのか？PDCAサイクルで対応するのか？
  – 運用・保守フェーズが重要か？
  – 発展の方法論をどのように定義するのか？
  – 改善されているとは？
• 柔軟に対応できるようにシステムを設計する
  – 泛用的な設計
  – ポリシーによる再構成
• Evidenceを残す仕組みを用意
  – 何かあったときに説明するため
    ⇒ 説明責任
• 対策に解はないかもしれないし、あっても唯一ではない可能性も考慮
  ⇒ マネージメントで対応する
Our challenge to Open Systems
Dependability

• オープンシステムへの対応
  – 不確定・不完全への対応
  – 説明責任の徹底
  – リスクマネージメント・リスクコミュニケーション
    • リスクの裏返し不確定・不完全への対応
    • unidentified riskへの対応
  – 生物としてのシステム: 恒常性の維持
    • 全体≠部分の総和 創発への対応
  – 緊急時対応は行動で
    • コンティジェンシー計画
  – システム評価: 運用中、廃棄後、資産の観点、再利用、LCC
  – システムプロファイリング
  – 安全文化、改善: PDCAなど醸成するもの
  – ディペンダビリティ属性間の調和の決定法
  – manageability, adaptability, usability への対応
  – stakeholder間の合意形成 assurance case
Our challenge to Open Systems
Dependability

• Dependability ≅ Accountability + Risk management
• How to deal with uncertainty and incompleteness
• Risk management and Risk communication
  – How to treat unidentified risk
• System as a Life (Human)
  – Keep a service continuously ≅ maintain homeostasis
  – Sum of parts in system ≠ Whole system
  – Emergence
• Business continuity plan
  – Contingency plan
  – manageability, adaptability, usability
• System profiling
• Improvement processes
  – PDCA cycle
  – harmonize among dependability attribute
    (including trade-off analysis)
• Define the Consensus building process among stakeholders
  (Assurance case)
安全と安心

- 危険を排除
  - 安全性を高めた
- 不安を軽減
  - 安心を提供した
- 不満に対応
  - 満足を提供した

- 安全とは、
  - 危険がないこと（絶対安全）から、許容できないリスクがないこと（機能安全）へ
  - リスク=ハザード×遭遇確率
  - 安全文化の醸成
    - マネージメントでも安全を保障する仕組み

- 開発者は安全性を高めるで信頼性やらを技術・プロセスにより保証する

- 利用者には技術の中身よりも、漠然とした「不安」が軽減されている方が重要かも知れない
  - 安心を得るためには、不安（anxiety）を軽減する仕組み

- 安全文化では、被害を受ける可能性がある立場について充分に考慮されていない
  - 利便性などとの相反は？
Realize 安心(Anshin)・安全(Anzen) -connecting requirement and technology-

- Realize Dependability (安心・安全) by connecting elemental technology and management processes.
- It is necessary to provide the consensus-building process among the stakeholders ⇒ Assurance Case
安全・安心を繋ぐ-要求と技術を結ぶ-

安心・安全を実現するために、さらに技術でそれらを担保するために、ステークホルダー間に何らかの形での合意形成を、そのための一手段としてAssurance Caseなどによるものが最低限用意されている必要がある。

ライフサイクル全体に対して安心・安全の実現
ステークホルダーの合意形成
マネージメント改善サイクル

安心

要求工学
（非機能要求の見える化なども）

説明責任のため
運用フェーズ以降もEvidenceの提示を可能とする仕組み

開発者
安全
技術で支えるできる限り客観的に

利用者

Assurance Case
XX-case

技術A、B、C
プロセスX、Y、Z
規格について

- オープンシステムを対象に全ライフサイクルでディペンダビリティを担保するための規格
  - システムプロファイルリングによるシステムディペンダビリティレベルの設定
- 続続的に満足いくサービスを、使いたいときに提供し、それが出来ない状態になったときにでも、保護してくれる安心な仕組みの提供
  - リスクマネージメントも含めて、各種（運用・保守等）マネージメントを
  - PDCAにかわる改善サイクルの提案
- Evidenceを基礎に設計から運用フェーズ以降も論拠が提出できる仕組みを規定
  - 全容把握は困難なので、どこかで切り分け；システム境界などカバーする範囲を設定しながら
- ステークホルダ間の合意形成（Assurance Caseの導入）
  - 契約を超えた納得（SLAで充分ではない）
  - 責任の明確化は分担だけではなく、共有と被覆・融合（和解のプロセス？）も含めて検討
System Profiling

- システム境界を変化させる、ステークホルダーの役割と情報の流れ（シナリオベース）、アスペクト指向ベース等、さまざまな視点・観点から、システムの挙動・障害発生時の影響等を解析して対応を考える
- ステークホルダーそれぞれの立場に応じてどのようなサービスを提供し（され）、責任として何をすべきなのかを明確にする
- 役割・責任共有など合意形成をスムーズに行うためのシステムの分析と運用方法など総合的にとらえるためのプロファイリング
- システムが故障したときの影響についても評価、その評価に応じた対策をシステム構築時に課すことも
- 全容把握は困難（不可能？）でも、つねに全体像の把握を試みること
System as a Life (Human)

- システムを生物（人間）と見る観点から、システムへの対応を探る
- 生命を維持するための仕組み:
  - 恒常性の維持のために必要な事柄: 自己回復力（免疫力）
  - 内科的処置（medical therapy）: Policy based reconfiguration
  - 外科的処置（surgical manipulation）: Property based isolation
- 環境変化へのシステムの対応: adaptanility
  - システム発展（進化）・超回復（overcompensation）の観点から
- 「部分の総和≠全体」への対応:
  - 自己組織化（self-organization）、創発（emergence）をどう表現し、どう対応するか
  - autonomic computing の考え方 MAPE ループ
  - 階層を超えた挙動などの取り扱い
• What can be said at all can be said clearly; and whereof one cannot speak thereof one must be silent. (Ludwing Wittgenstein, Tractatus Logico-Philosophicus)
• Management Processes
  – what we need to manage?
    • resource, process (development, operation and maintenance,…),
      information, knowledge, assurance, asset, configuration, performance, project, …
  – Processes for PDCA (using available software processes, etc)
  – Processes for Risk Management
  – Standardization for Accountability (risk management with evidence) with Dependability Level
• 合意せざるを得ないもの
  – 合意形成プロセスは明示的に、第三者が見てもわかるようにする。
• SW工学、要求工学でなされてはいるが、不十分な点を浮き彫りにしてよりよい解法を提案する。ラフにいえば新しい仕様記述言語の設計
  – より可読性が高く、コンパクトにステークホルダーの全てが納得のいく形で記述、意思決定し、合意形成をするための方法としてassurance case
• In general, we can view a specification as the statement of an agreement between a producer of a service and a consumer of the service or between an implementer and a user