Dependability Case and Metrics for Open Systems Lifecycle

@IFIP WG 10.4

Jin Nakazawa, Keio University, Japan
Yutaka Matsuno, National Institute of Advanced Industrial Science and Technology (AIST)

+DEOS core team

US's Vision for High Speed Rail

www.whitehouse.gov

Which one is the most dependable? (not just safe)

sell

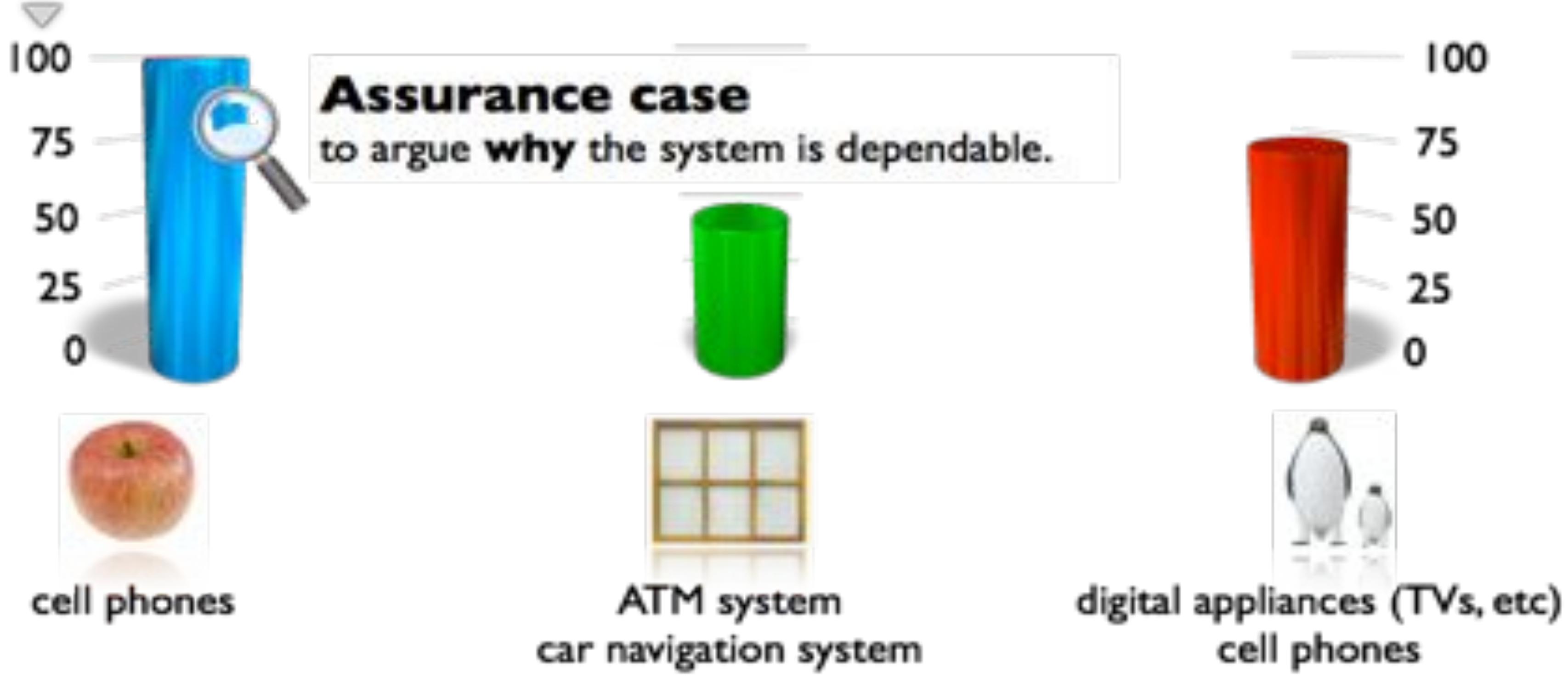
TGV

ICE

Shinkansen

Goal

Metrics to represent **how dependable** a system is.



Assurance case
to argue **why** the system is dependable.

Target(metrics): Operating systems as the core of application systems

Target(AC): Open systems

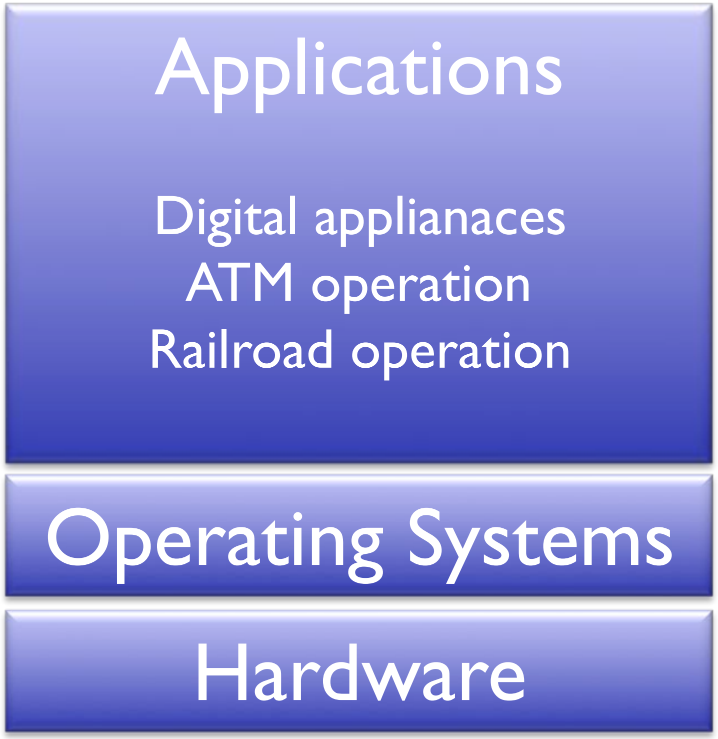
Outline

- Dependability Metrics
  - Initial research outcome aiming at evaluating the amount of dependability of systems.
  - by Jin Nakazawa, Keio Univ.
- Dependability Case (D-Case)
  - A scheme to express dependability of operating systems adopting assurance case.
  - by Yutaka Matsuno, AIST

Dependability Metrics for Open Systems Lifecycle

Jin Nakazawa, Keio University, JAPAN

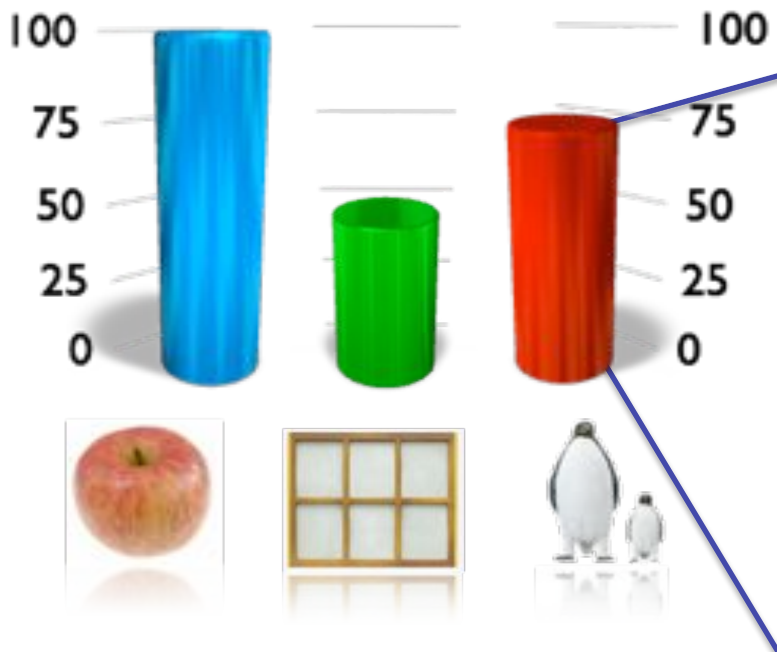
Roles of Operating Systems for Dependability



- Dependable applications must be on a dependable OS.
- Dependable OS provides
  - Development/testing tools
    - Source code verification/validation, Fault injection, Benchmarking
  - Runtime/maintenance technologies
    - Fast reboot, resource reservation, logging, remote updating, etc.

→**Dependability Support**

Need for Dependability Metrics



- Dependable applications must be on a dependable OS.
- Dependable OS provides
  - Development/testing tools
    - Source code verification/validation, Fault injection, Benchmarking
  - Runtime/maintenance technologies
    - Fast reboot, resource reservation, logging, remote updating, etc.

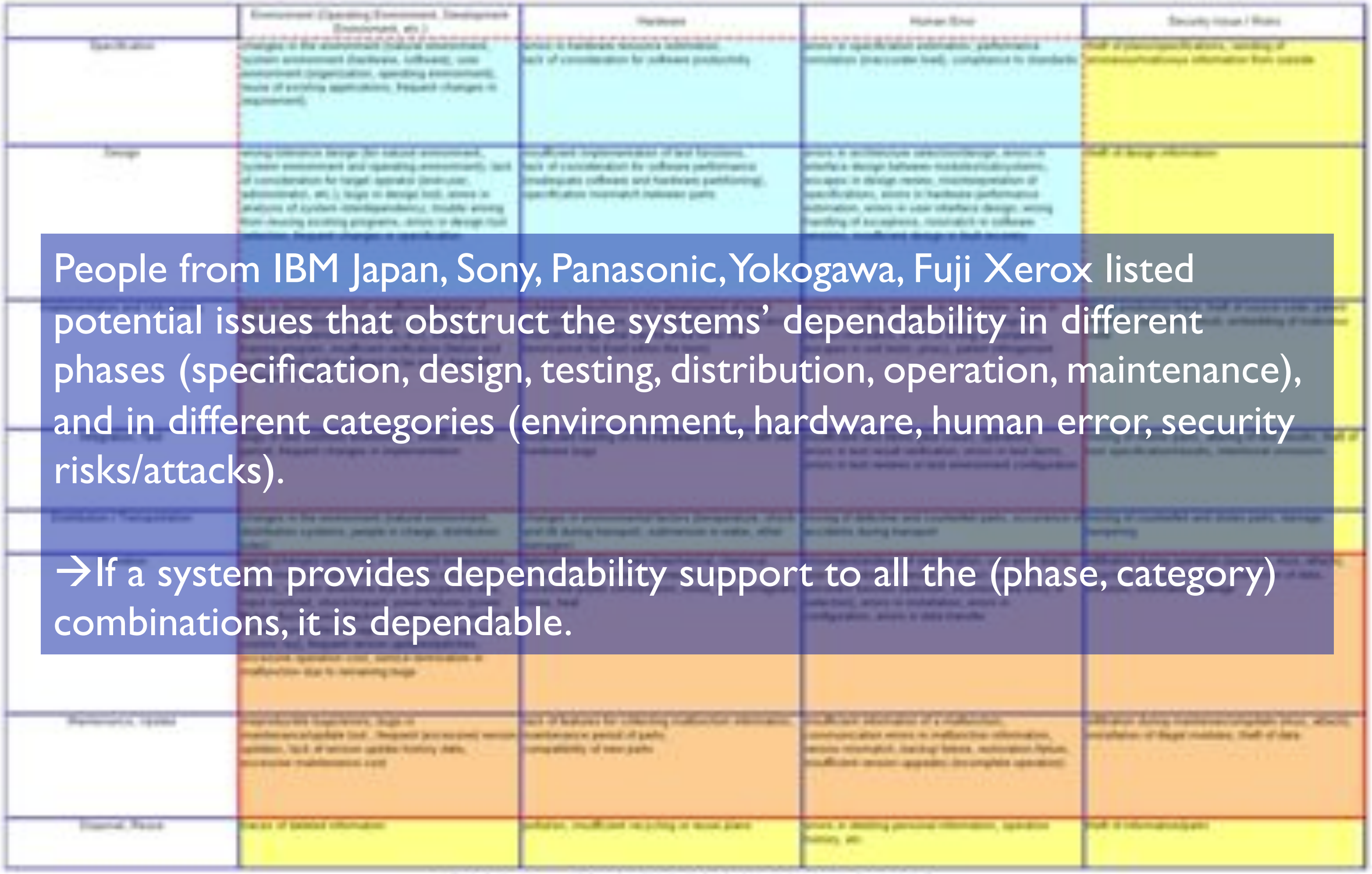
→**Dependability Support**

Dependability Metrics Goals

- **Quantitative scale** to compare dependability of different **system**s.
  - Represents how much the developers can account for in terms of the dependability requirement to their systems.
- **Dependability visualization** to intuitive understandings of dependability.
  - Used as tools for stakeholders to communicate with on dependability.
- Addressing different phases in an **open system's life cycle.**
  - Experimental evaluation of a system describes the system's dependability against currently supposed ***obstructions***.
  - Need to evaluate how the developers cope with dependability in the range of different phases to infer the system's dependability against unsupposed obstructions (open systems support).

Dependability Obstructions

"DEOS Project White Paper"



People from IBM Japan, Sony, Panasonic, Yokogawa, Fuji Xerox listed potential issues that obstruct the systems' dependability in different phases (specification, design, testing, distribution, operation, maintenance), and in different categories (environment, hardware, human error, security risks/attacks).

→If a system provides dependability support to all the (phase, category) combinations, it is dependable.

Table 1 Dependability Obstructions

Dependability Obstructions

"DEOS Project White Paper"

	Environment (Operating Environment, Development Environment, etc.)	Hardware	Human Error	Security Issue / Risks
Specification	[illegible] ✓	[illegible] ✓	[illegible] ✓	[illegible] ✓
Design	[illegible] ✓	[illegible] ✓	[illegible] ✓	[illegible] ✓
Implementation and Unit testing	[illegible] ✓	[illegible] ✓	[illegible] ✓	[illegible] ✓
Integration, Test	[illegible] ✓	[illegible] ✓	[illegible] ✓	[illegible] ✓
Distribution / Transportation	[illegible] ✓	[illegible] ✓	[illegible] ✓	[illegible] ✓
Operation	[illegible] ✓	[illegible] ✓	[illegible] ✓	[illegible] ✓
Maintenance, Update	[illegible] ✓	[illegible] ✓	[illegible] ✓	[illegible] ✓
Disposal, Reuse	[illegible] ✓	[illegible] ✓	[illegible] ✓	[illegible] ✓

Table 1 Dependability Obstructions

Approach

(1) Divide

Evaluate the amount of dependability supports included in the target OS. Evaluation is **qualitative**, and conducted for each support.

☑ ☐ ☑
☐ ☑ ☑

current

(2) Conquer

Quantify the amount of the dependability supports based on the amount of dependability requirements.

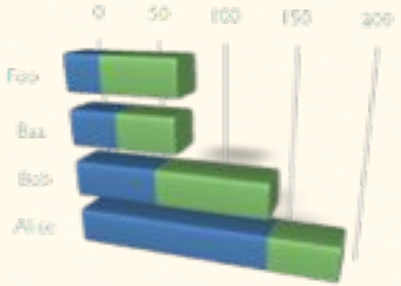
Satisfied
23

Required
123

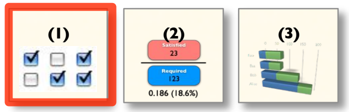
0.186 (18.6%)

(3) Visualize

Visualize the qualitative/quantitative evaluation from a range of different aspects. Used for comparison of different systems.

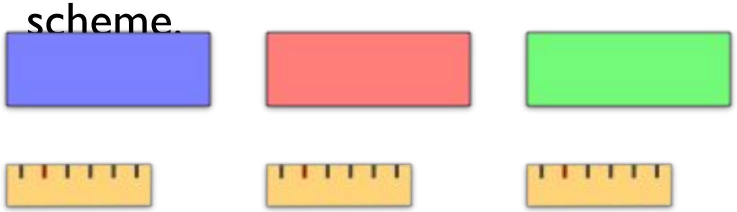


Qualitative Evaluation
(1) Target



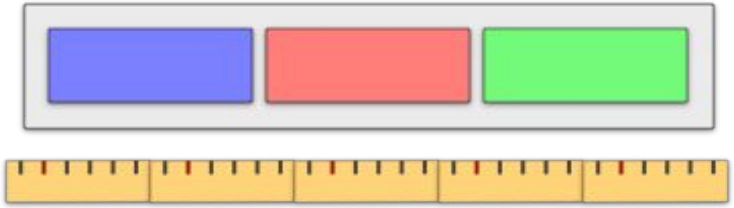
Elemental Technologies and Tools

- An OS includes a range of different technologies and tools to support dependability.
- DEOS includes 20+ supports.
- Each support is evaluated with our scheme.

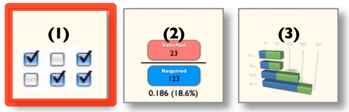


Entire Operating System

- The results of elemental technologies and tools are merged to represent the dependability of the entire OS.
- They are complementary; some are valuable at development time, and some others are at run time.



Qualitative Evaluation (2) Labeling



- We use the following “qualitative measures” to evaluate dependability support in an OS.
  - Labeling dependability support with these words.

Phase

Specification
Design
Implementation
Test
☑ Operation
Maintenance
Disposal

Component

☑ CPU
RAM
File system
Communication
Input/output
Power supply

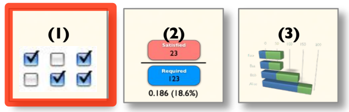
Cause

Environment
Hardware
☑ Attack
☑ Mistake

Property

☑ Availability
Reliability
Safety
Integrity
Maintainability

Qualitative Evaluation (3) Example



- Advanced Real-time in DEOS
  - http://sourceforge.net/projects/art-linux/

Phase

☑ Specification
☑ Design
☑ Implementation
☑ Test
☑ Operation
☑ Maintenance
☑ Disposal

Component

☑ CPU
RAM
File system
Communication
Input/output
Power supply

Cause

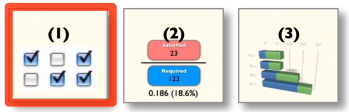
Environment
Hardware
Attack
☑ Mistake

Property

Availability
☑ Reliability
Safety
Integrity
Maintainability

* The developer's self-assessment

Qualitative Evaluation (4) Example



- Source code model checker in DEOS
  - http://www.computer.org/portal/web/csdl/doi/10.1109/STFSSD.2009.35

Phase

☑ Specification
☑ Design
☑ Implementation
Test
Operation
Maintenance
Disposal

Component

☑ CPU
☑ RAM
☑ File system
☑ Communication
☑ Input/output
☑ Power supply

Cause

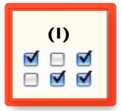
Environment
Hardware
☑ Attack
☑ Mistake

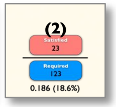
Property

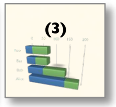
Availability
☑ Reliability
☑ Safety
Integrity
Maintainability

* The developer's self-assessment

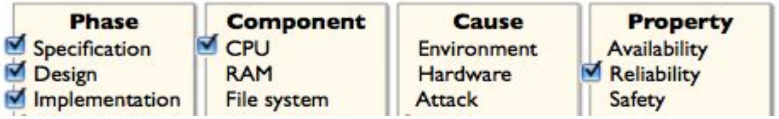
Evidence of ☑



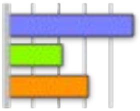


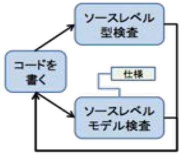


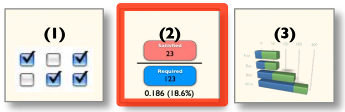
- A tick and its evidences should be linked to clarify the check actually satisfies the property.
  - Result of benchmarking, fault injection, etc.
- We use assurance cases for this purpose.
  - To be presented next.



Phase	Component	Cause	Property
☑ Specification	☑ CPU	Environment	Availability
☑ Design	RAM	Hardware	☑ Reliability
☑ Implementation	File system	Attack	Safety







Quantification (conquer)

System Developers: Represent the dependability required in a system.

Phase
☑ Specification
☑ Design
☑ Implementation
Test
Operation
Maintenance
Disposal

Component
☑ CPU
☑ RAM
☑ File system
☑ Communication
☑ Input/output
☑ Power supply

Cause
Environment
Hardware
☑ Attack
☑ Mistake

Property
Availability
☑ Reliability
☑ Safety
Integrity
Maintainability

N_r : amount of requirements

$$N_r = 3 \times 6 \times 2 \times 2 \text{ checks}$$
$$= 72$$

OS Developers: Represent the dependability that an OS can satisfy.

Phase
☑ Specification
☑ Design
☑ Implementation
Test
Operation
Maintenance
Disposal

Component
☑ CPU
☑ RAM
File system
Communication
Input/output
Power supply

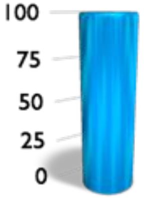
Cause
Environment
Hardware
☑ Attack
Mistake

Property
Availability
☑ Reliability
☑ Safety
Integrity
Maintainability

N_s : amount of supports

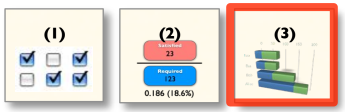
$$N_s = 3 \times 2 \times 1 \times 2 \text{ checks}$$
$$= 12$$

Dependability Score: Coverage of dependability support of an OS.



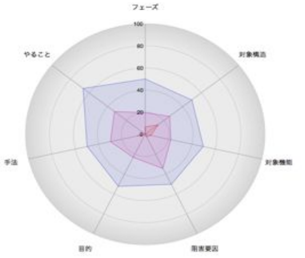
$$DS = N_s / N_r$$

Used for matching
between OS and apps

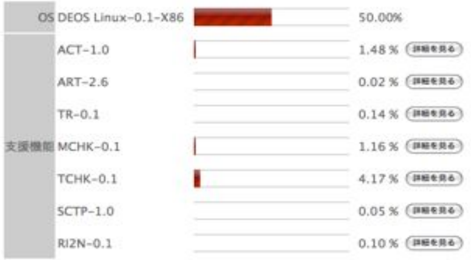


Visualization

- Visualize ticks and scores for intuitive understandings of
  - What properties are covered by an OS,
  - How each dependability support contributes to,
  - How the dependability support in an OS is balanced,
  - What evidences the ticks,
  - Etc..

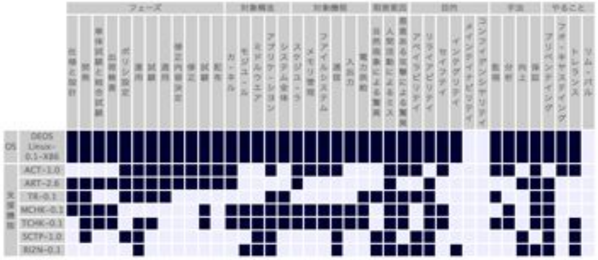


balance



OS	DEOS Linux-0.1-X86	50.00%	
支援機能	ACT-1.0	1.48 %	詳細を見る
	ART-2.6	0.02 %	詳細を見る
	TR-0.1	0.14 %	詳細を見る
	MCHK-0.1	1.16 %	詳細を見る
	TCHK-0.1	4.17 %	詳細を見る
	SCTP-1.0	0.05 %	詳細を見る
	RI2N-0.1	0.10 %	詳細を見る

contribution



coverage

evidence

Limitation

- Ticks are still abstract.
  - E.g., a security mechanism is tolerant of DoS attacks only.
  - Such a detailed argument is done with assurance cases.
- Overhead of dependability support mechanism in an OS cannot be described with checks.
  - Represented in assurance cases with benchmark results (evidences).

Summary

- Qualitative evaluation categories are proposed.
    - Its target is operating systems (not generic open systems yet).
- Initial ideas for quantification and visualization are addressed.
    - Used for comparison of different operating systems, and matching the OS's against applications' dependability requirements.
- Future work
    - Extend the metrics to cope with open systems more systematic way.
→ will be done based on D-Case description.
    - Further research on quantification and visualization