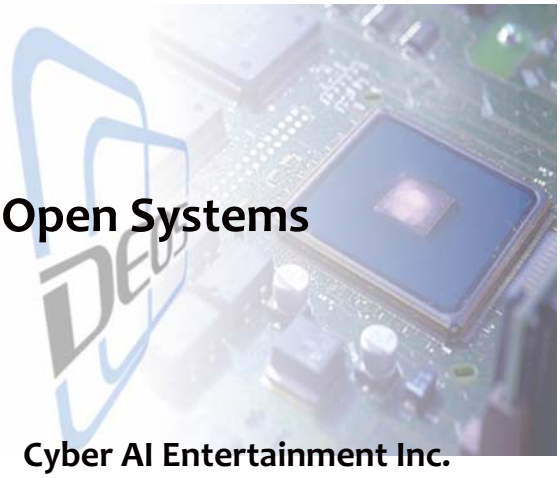
D-fops:
Dependability Framework for Open Systems



Yasuhiko Yokote, Ph.D.

Cyber AI Entertainment Inc.
The University of Tokyo
JST DEOS Project

December 16, 2010

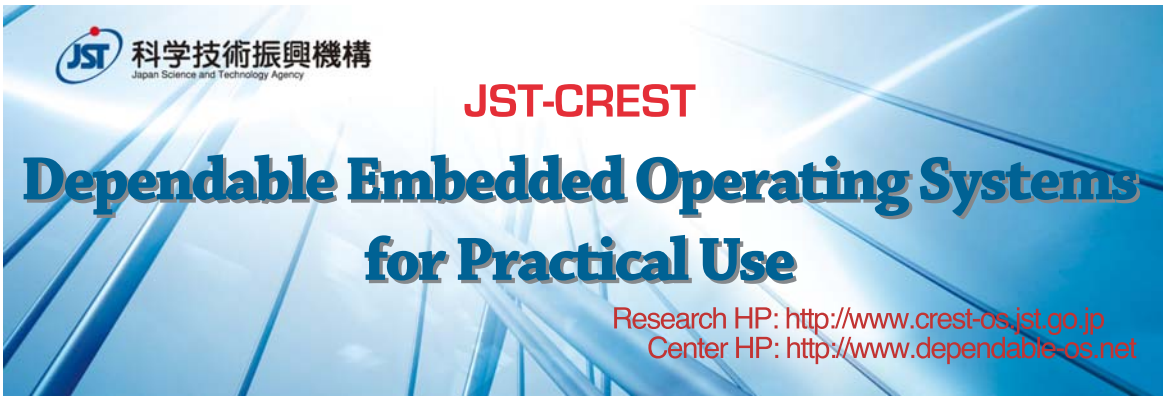
科学技術振興機構
Japan Science and Technology Agency

Agenda

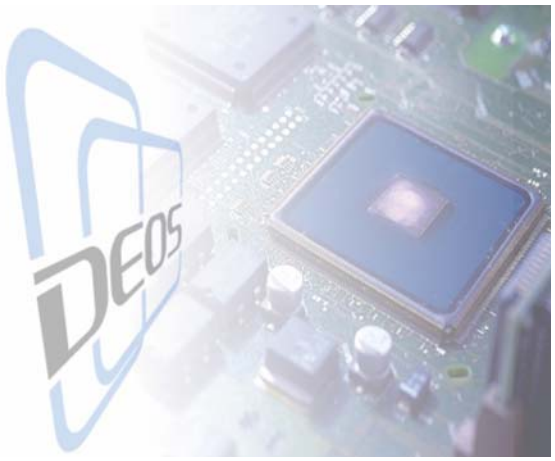
- Introduction
  - What is the position of D-fops in the DEOS process?
  - How stakeholders control system's dependability?
- Dependability Architecture
  - How the DEOS process is maintained?
- System Structure
  - How D-Case works with system services?
- Video demonstration
  - D-Case and D-fops
- Concluding Remarks
  - What is still in open for open systems dependability?

D-fops Team Members

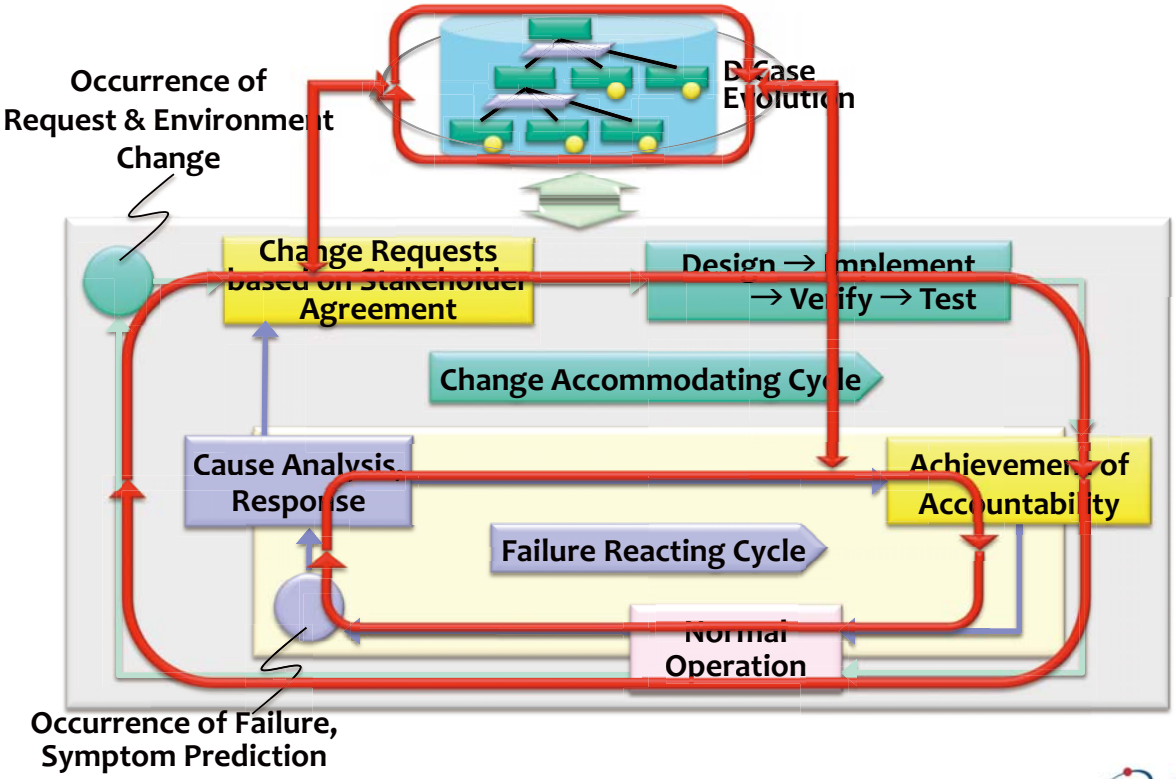
- Asai, Nobuhiro/JST
- Koizumi, Hideaoki/JST
- Matsubara, Shigeru/JST
- Miyahira, Tomohiro/JST
- Noma, Shinichi/JST
- Ono, Kiyoshi/JST
- Sekiba, Jiro/JST
- Takamura, Hiroki/JST
- Takeda, Shingo/JST
- Yashiro, Makoto/JST



Introduction



DEOS Process

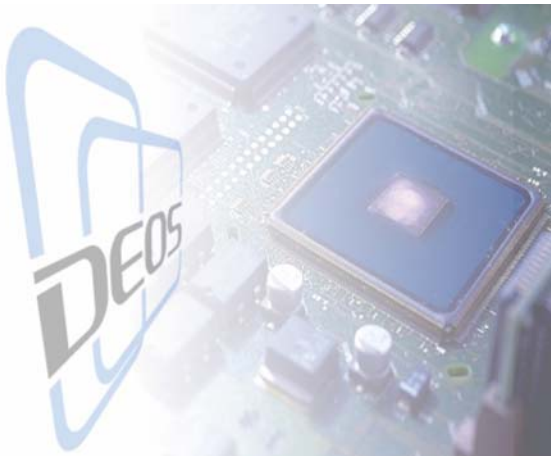


Working with DEOS Process

"And no one puts new wine into old wineskins; or else the new wine bursts the wineskins, the wine is spilled, and the wineskins are ruined. But new wine must be put into new wineskins." (Mark 2:22)

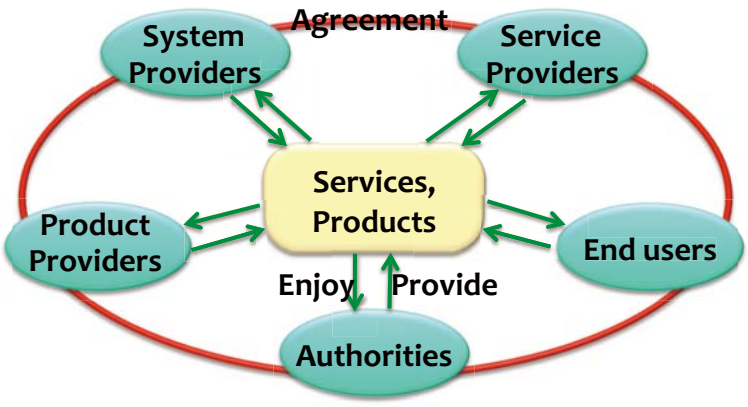
- D-Case from systems view
  - command sets to sustain dependability
- System from D-Case view
  - agent to sustain dependability agreed by stakeholders
- Interaction between "online" and "offline" cycles
  - execute D-Case
  - reflect system's state in D-Case
- Control open systems dependability based on stakeholders agreements

Dependability Architecture



Stakeholders

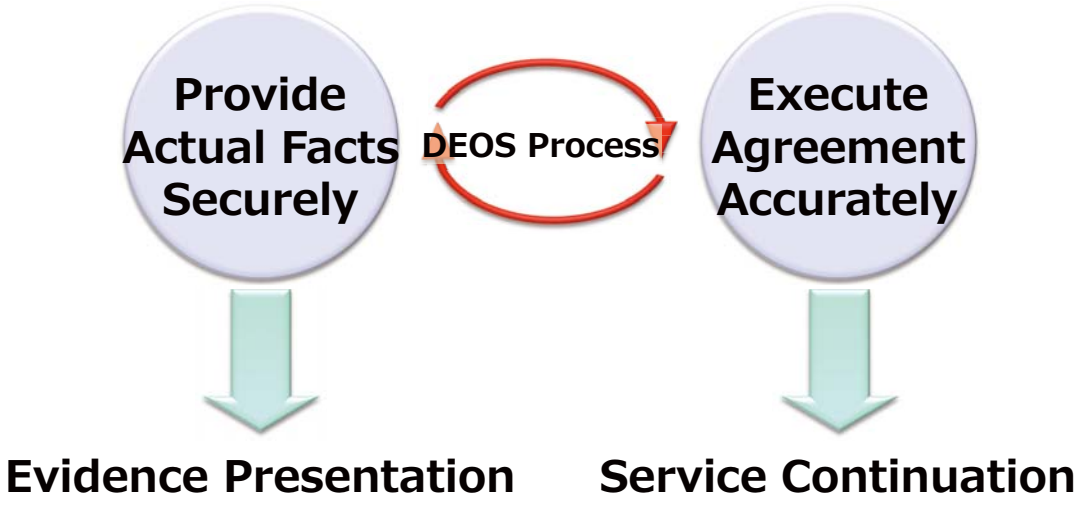
- Provider/Consumer relationship
  - Provider responsibility
  - Consumer satisfaction
- Agreement among stakeholders
- Systems working for stakeholders



Architecture Requirements

- "Added" value (or benefit) to stakeholders
  - What is value to stakeholders by sustaining system's dependability?
- Change of Stakeholder's requirements
  - How is this change resolved quickly?
- Change of environment
  - How is this change adopted quickly?

Architecture to Run DEOS Process Securely



Basic Mechanisms

- Isolation
  - Make any entity independent of others.
- Reconfiguration
  - Make a new policy effective.
- Monitoring
  - Make system state visible.
- Recording
  - Make system history persistent.
- D-Case update
  - Make D-Case of two cycles in the DEOS process consistent.

Isolation Requirements

What is Isolated?	requirement
address space	independent address range, which CPU is referencing.
name space	file name structure, process id, and other id's of an OS.
real memory	access validation of a range of memory from others.
cache memory	no affect of a range of cache from others.
CPU scheduling	guarantee of the maximum CPU utilization.
CPU's assignment	assignment of one or more processor cores to each VM.
I/O bandwidth	guarantee of the maximum I/O bandwidth.
bus bandwidth	guarantee of the maximum bus bandwidth.
interrupt	routing an interrupt to an appropriate VM.
time-of-day	controlling a time without affect of other VM's.
privilege	violation between different privilege levels.

Reconfiguration

- Difficulty in coordinating component software in change.
  - software download and software update
- Difficulty in understanding a result of coordination.
  - dependency description of components

- Interface to accept written descriptions of behavior.
  - D-Case, assimilation of various descriptions
- Coordinate written descriptions of behavior.
  - policy control and management

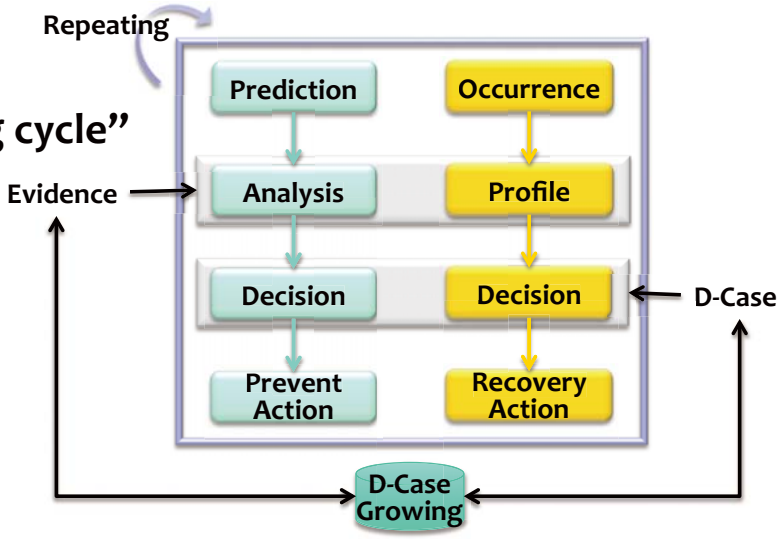
Monitoring and Recording

- Difficulty in identifying source to monitor.
  - complex component dependency
- Difficulty in identifying an error causing failure.
  - insufficient error reports

- Monitor source identified at run-time.
  - D-Case as common description of its policy
- Record system state identified at run-time securely and accurately.
  - D-Case as single representation of system state

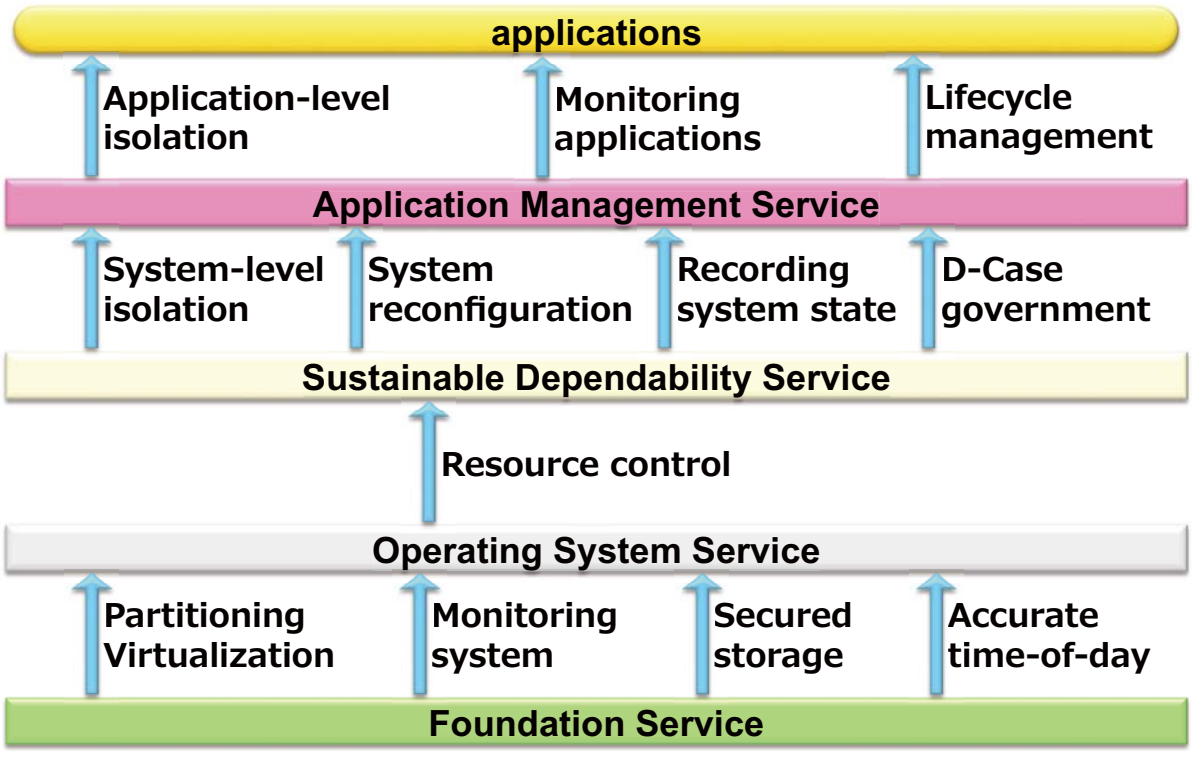
D-Case Update

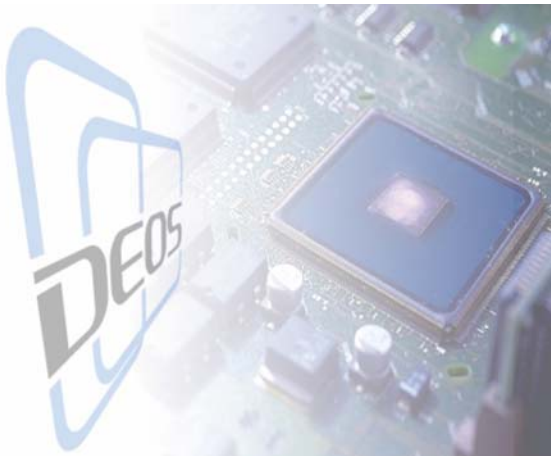
- D-Case is central description to interact "online" with "offline."
  - two loops in "failure reacting cycle"



- Update "online" D-Case consistent with "offline" D-Case.

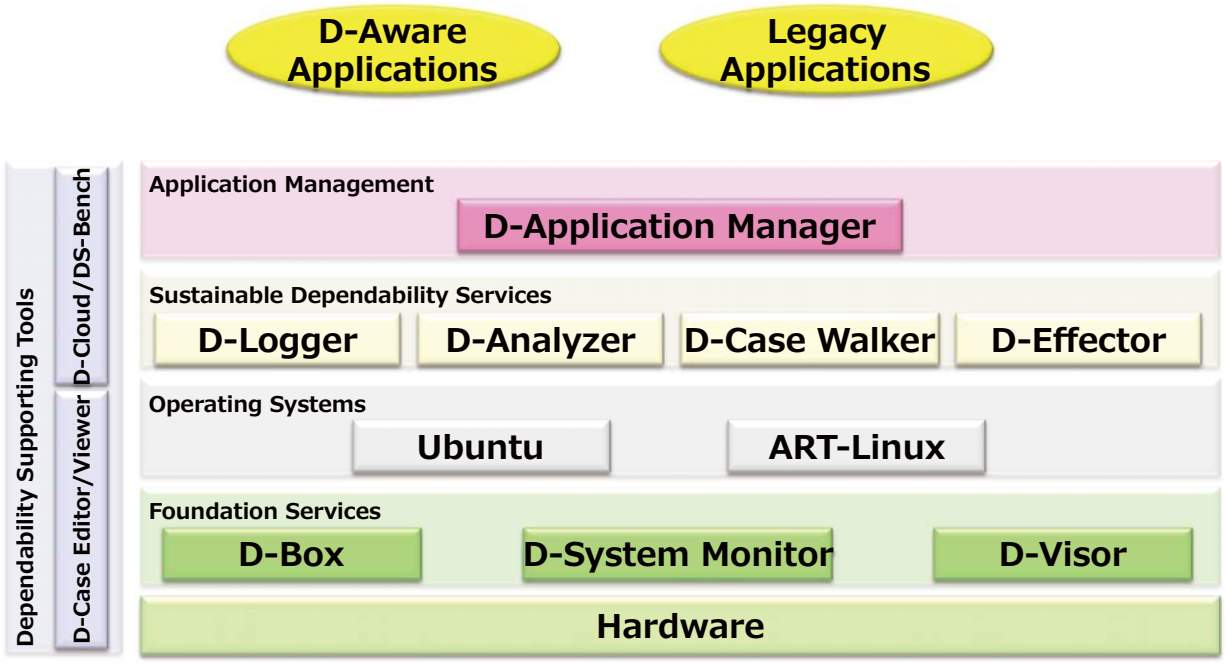
Layered Responsibility



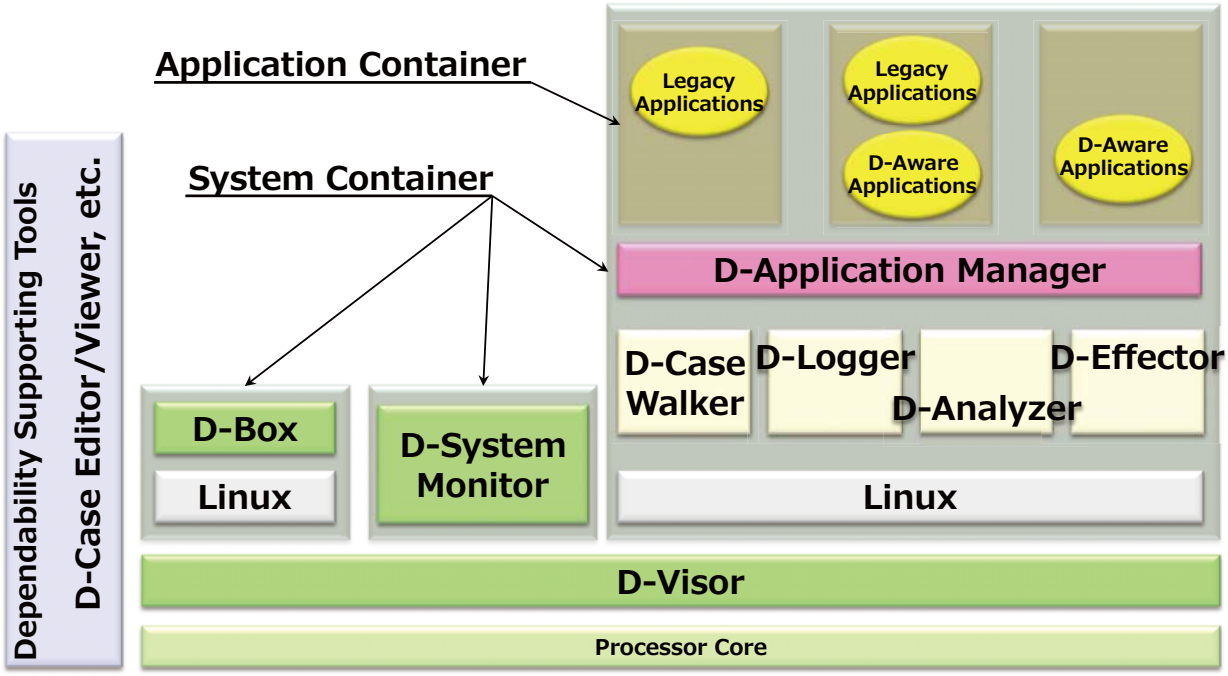


System Structure

Service Level Abstraction



Software Structure in Run-time View

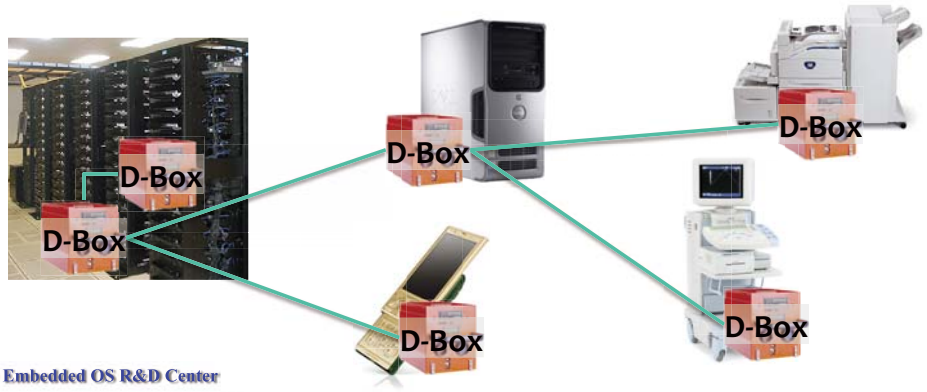


D-Box – Securing Any Records

- D-Box is securely isolated storage, where
  - it is a place to store evidences,
  - one D-Box is placed at a device,
  - a chain of D-Box can be structured.
- Access to D-Box is secured, where
  - an end-point is authenticated, and
  - transmission between an end-point and D-Box is encrypted.
- Examples of D-Box content are:
  - system configuration,
  - event/log records,
  - time-stamp, and
  - key files.
- D-Box is similar to a flight recorder.

Network of D-Boxes

- Network of D-Boxes is representing a dependability chain, i.e. Open Systems Dependability is sustained in a system consisting of linked D-Boxes.
- Each D-Box contains evidences indicating its associated device is secured by Open Systems Dependability.
- Exchanging evidences between D-Boxes enables us to know a "trend" of dependability.
  - sample use case: remote maintenance



D-Case Walker

- D-Case stored in D-Box is processed in steps of:
  - Interpretation of D-Case,
  - Extraction of decision points, and
  - Confirmation of evidences.
- A script associated with a D-Case node is executed to:
  - Enforce a policy with D-Effector, and
  - Synchronize stakeholders agreement and system behavior securely.

D-Effector

- D-Case with agreement among stakeholders is accurately executed.
- The following functions are performed:
  - Dynamic reconfiguration
    - with self-description of software
    - upon software update or application download
  - Undo
    - to revert the state before configuration change
  - Snapshot of system
  - Fail-safe default

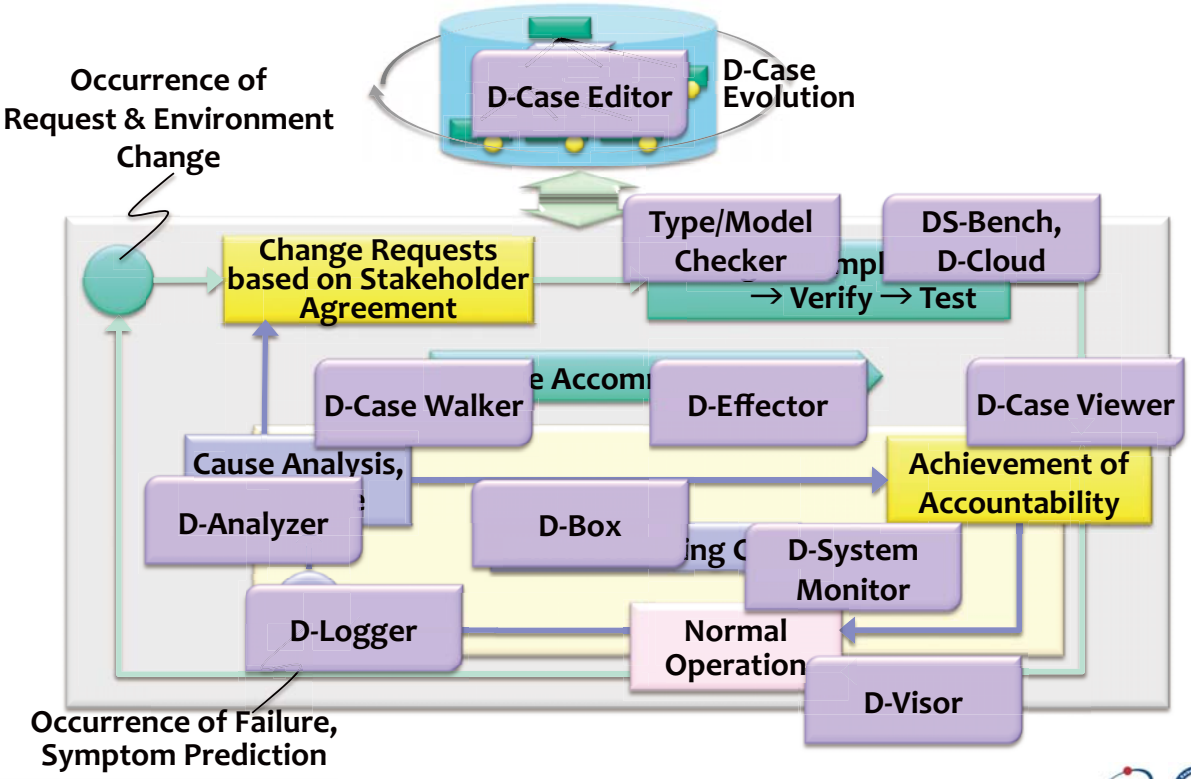
D-System Monitor

- D-System Monitor guarantees OS is working as expected.
  - OS offers several critical mechanisms that do not trust if OS is hijacked.
- The runtime behavior and internal data structures of the kernel are monitored from its outside.
  - Inspect I/O operations, internal data structures, the access to privileged registers, the execution of privileged instructions, precisely.
  - Difficult to compromise D-system Monitor due to Isolation of virtual machines.
  - In open systems, the mechanisms would evolve as the demand of applications changes.
- Details are presented at Workshop 2 of this Symposium.

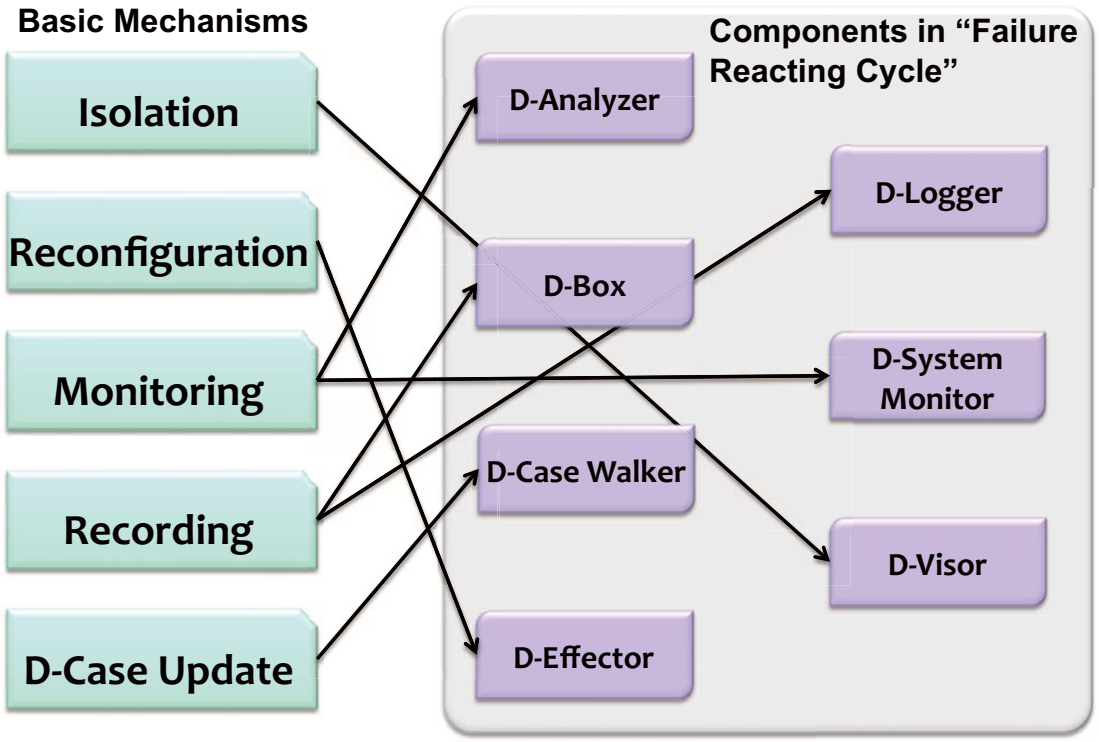
D-Visor

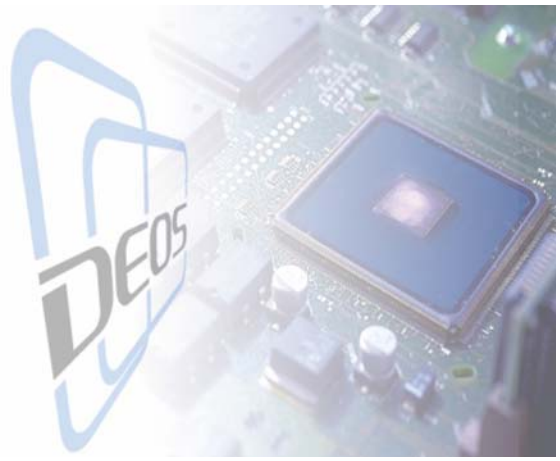
- D-Visor is an infrastructure to accommodate OS and D-System Monitor.
  - Provision of isolation – spatial and temporal.
  - Practical solution for embedded systems, in terms of:
    - Less overheads, and
    - No support of hardware assist for virtualization necessary.
- Developing two implementations focusing on multi-core processors for embedded systems.
  - SUPMONE: lightweight D-Visor for embedded systems.
  - ART-Linux: D-Visor for hard real-time systems.
- Verification by model checking.
  - Provision of a specification description of a virtualization layer.
  - DEOS model checker verifies the description and checks the correct uses of lock variables, pointers, and processor hardware.
- Details are presented at Workshop 2 of this Symposium.

DEOS Process and D-fops



Component Map





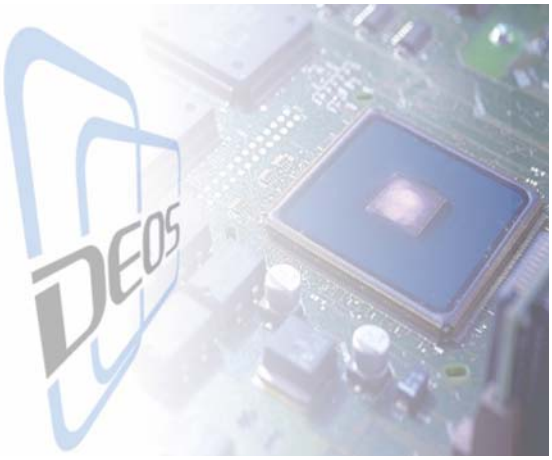
Video Demonstration

Robot Controlled by D-Case

- Reception robot
  - Due to disconnection of Camera, its backup takes over its responsibility to welcome a visitor.
  - This action (fail-over) is written in D-Case, and D-fops accesses it and starts the “fail-over” action as part of the “failure reacting cycle” of the DEOS process.

Application Containers

- Application Container is a partition at the application level.
- Two application containers: Face recognition and high load application
- Load of one application container (high load app) increases, then it limit its resource use to protect another application container (Face recognition).
- Such a policy is written in D-Case.



Concluding Remarks

Open Questions

- How cost reduction caused by sustaining dependability is estimated?
- What value is delivered to stakeholders through system's dependability?
- What is "better" for stakeholders?
- How human beings are involved to sustain dependability?
- What are missing to be a solution to sustain dependability for stakeholders?

Summary

- Dependability architecture
  - Work with DEOS process
  - Dependability Framework for Open Systems
- Its reference implementation: D-fops
  - Integration of elemental technologies from the project
- Schedule
  - Practical experiments with real applications
  - Release as Linux packages
  - Consortium, plan to start on Autumn 2011

More Information

http://www.dependable-os.net/index-e.html

Thank you!