Robotics DSIG Agenda ver.1.2.0

# OMG TC Meeting - Burlingame, CA, USA -- December 5-9, 2005

| | | TF/SIG | | | | |
|---|---|---|---|---|---|---|
| | | Host | Joint (Invited) | Agenda Item | Purpose | Room |
| **Monday (Dec. 5)** | | | | | | |
| | | | | | | |
| 12:00 | 13:00 | | | **LUNCH** | | |
| 13:00 | 18:00 | | | Architecture Board Plenary | | Harbor B, Lobby Lv1 |
| 15:00 | 17:00 | Robotics | SDO | Steering Committee of Robotics DSIG | | Sandpebble A, Lobby Lv1 |
| **Tuesday (Dec.6) SDO Plenary and Robotics Plenary** | | | | | | |
| | | | | | | |
| 12:00 | 13:00 | | | **LUNCH** | | |
| 13:00 | 13:10 | Robotics, SDO | | Welcome and Review Agenda | Robotics/SDO Joint Meeting Kick-off | Gr Peninsula C, Lobby Lv1 |
| 13:10 | 14:10 | Robotics | (SDO) | <Special Talk><br>Introduction to the Agent-SIG activities [tentative]<br>- James J. Odell (OMG Agent-PSIG) | Informative | |
| 14:10 | 15:00 | Robotics | (SDO) | Response from Real-Time Innovations<br>- Hung Pham (RTI) | RFI response | |
| | | | | **Break (20min)** | | |
| 15:20 | 16:10 | Robotics | (SDO) | Response from Java.net [tentative]<br>- Bruce Boyes (Systronix) | RFI response | |
| 16:10 | 17:00 | Robotics | (SDO) | The Robot Software Communications Architecture (RSCA): Embedded Middleware for Networked Service Robots<br>- Seongsoo Hong (Seoul National Univ.) | RFI response | |
| 17:00 | 17:50 | Robotics | (SDO) | Capabilities: Human Interface of the Robotic Systems RFI<br>- Soo-Young Chi (ETRI) | RFI response | |
| **Wednesday (Dec.7) Robotics Plenary** | | | | | | |
| 9:00 | 10:00 | Robotics | (SDO) | <Special Talk><br>High Assurance Security and Safety for Robotics<br>- Joseph M. Jacob (Objective Interface Systems) | Informative | Sandpebble DE, Lobby Lv1 |
| | | | | **Break (20min)** | | |
| 10:20 | 11:10 | Robotics | (SDO) | Hardware Abstraction to the Robotic Systems RFI<br>- Seung-Ik Lee (ETRI) | RFI response | |
| 11:10 | 12:00 | Robotics | (SDO) | **Response from NEC**<br>**- Yoshihiro Fujita (NEC)** | RFI response | |
| 12:00 | 14:00 | | | **LUNCH and OMG Plenary** | | |
| 14:00 | 14:40 | Robotics | (SDO) | <Special Talk><br>Open source robotic Control - Teambotica1.0<br>- Regis Vincent (SRI International) | Informative | Sandpebble DE, Lobby Lv1 |
| | | | | **Break (10min)** | | |
| 14:50 | 15:30 | Robotics | (SDO) | Network Robot Platform for Information Shareing<br>- Ken-ichiro Shimokura (NTT) | RFI response | |
| 15:30 | 16:10 | Robotics | (SDO) | Human Robot Interaction in Network Robots<br>- Norihiko Hagita (ATR) | RFI response | |
| 16:10 | 16:50 | Robotics | (SDO) | Network Robots Standardization Activity in Japan<br>- Miwako Doi (Toshiba) | RFI response | |
| | | | | **Break (10min)** | | |
| 17:00 | 17:20 | Robotics | (SDO) | Chartering Robotics Domain Task Force and voting | voting | |
| 17:20 | 17:40 | Robotics, SDO | | Next Meeting Agenda Discussion, etc | Robotics/SDO Closing session | |
| 17:40 | | | | Adjourn | | |
| 18:00 | 20:00 | | | **OMG Reception** | | |
| **Thursday** | | | | | | |
| | | | | | | |
| 12:00 | 13:00 | | | **LUNCH** | | |
| 13:00 | 14:00 | MARS | SDO, Robotics | RFI response midterm report and discussion | | Bayside A, Lobby Lv1 |
| 14:00 | 14:30 | MARS | SDO, Robotics | Proposal of Robotics DTF and discussion | | Bayside A, Lobby Lv1 |
| 13:00 | 18:00 | | | Architecture Board Plenary | | |
| **Friday** | | | | | | |
| 8:30 | 12:00 | | | AB, DTC, PTC | | |
| 12:00 | 13:00 | | | **LUNCH** | | |
| | | | | | | |

## Other Meetings of Interest

| | | | | | | |
|---|---|---|---|---|---|---|
| **Monday** | | | | | | |
| 8:00 | 8:45 | OMG | | New Attendee Orientation | | |
| 9:00 | 12:00 | OMG | | Introduction to the MetaObject Facility (MOF) | | |
| 13:00 | 17:00 | OMG | | The Abstract Syntax Tree Meta Model (ASTM) | | |
| 18:00 | 19:00 | OMG | | New Attendee Reception (by invitation only) | | |
| **Tuesday** | | | | | | |
| 9:00 | 12:00 | OMG | | Introduction to UML 2.0 | | |
| 13:00 | 17:00 | OMG | | MDA--Where it Came From and Where It's Going | | |
| **Wednesday** | | | | | | |
| 9:00 | 12:00 | OMG | | Applying Model Driven Architecture to Services Oriented Architectures using Web Services | | |
| **Thursday** | | | | | | |

# Robotics-DSIG Atlanta Meeting Minutes
# (robotics/2005-12-02)

## Overview and votes

As the Robotics-DSIG is still waiting for responses to the RFI that was issued during the Boston Meeting (June 2005) and due to November 2005, no particular issue has been addressed. The Atlanta meeting consisted mainly in presentations. A call has been made for electing the co-chairs of the Robotics-DSIG.

## OMG Documents Generated

robotics/2005-09-01 Final Agenda (Tetsuo Kotoku)
robotics/2005-09-02 Robotics DSIG approved Boston Meeting Minutes (Tetsuo Kotoku)
robotics/2005-09-03 Opening Presentation (Tetsuo Kotoku)
robotics/2005-09-04 Robotics DSIG Roadmap (Tetsuo Kotoku)
robotics/2005-09-05 Presentation : "Implementing and Teaching Emerging Robotics Standards at the University Level" (Bruce Boyes)
robotics/2005-09-06 Presentation : "Common Robot Interface Framework for Device Abstraction" ( Seung-Ik Lee)
robotics/2005-09-07 Presentation : "Standards in Action: Prototype Robots at Aichi International Exposition 2005" (Masayoshi Yokomachi)
robotics/2005-09-08 Presentation : "Machine vision and actuators for robotics and automation" (Kok-Meng Lee)
robotics/2005-09-09 Presentation : "Slashing development time with component-based programming" (Hung Pham)
robotics/2005-09-10 Presentation : "Korean intelligent robot standardization status" (Yun Koo Chung )
robotics/2005-09-11 Presentation : "Introduction to Toshiba Home Robots and Our Approach to RT Standardization" (Fumio Ozaki)
robotics/2005-09-12: Atlanta Robotics DSIG DTC Report Presentation (Tetsuo Kotoku)
robotics/2005-09-13 Meeting Minutes – Draft (Olivier Lemaire and Fumio Ozaki)

## Agenda

09:00- 09:20 Welcome and Review Agenda Meeting Kick-off
09:20-10:50 "Implementing and Teaching Emerging Robotics Standards at the University Level" - Bruce Boyes (Systronix)
10:50-10:20 "Common Robot Interface Framework for Device Abstraction" - Seung-Ik Lee (ETRI)
10:40-11:10 "Standards in Action: Prototype Robots at Aichi International Exposition 2005" - Masayoshi Yokomachi (NEDO)
11:10-12:00 "Machine vision and actuators for robotics and automation"- Kok-Meng Lee (Georgia Institute of Technology)
14:00-14:50 "Slashing development time with component-based programming" - Hung Pham (RTI)
15:10-15:40 "Korean intelligent robot standardization status" - Yun Koo Chung (ETRI)
15:40-16:10 "Introduction to Toshiba Home Robots and Our Approach to RT Standardization" - Fumio Ozaki (Toshiba)
16:10-16:40 Robotic Systems RFI (mars/2005-06-12) promotion
16:40-17:00 Next Meeting Agenda Discussion, etc Robotics Closing
17:00 Adjourn

## Minutes

*14 September, Wednesday*
Tetsuo Kotoku, presiding co-chair
Robotics DSIG Plenary
**Meeting Week – Kick-off**

Meeting was called to order at 09:00
Tetsuo Kotoku provided a brief overview of the Boston Minutes.
***Action:***
With some minor correction, the minutes were approved.
- Robotics DSIG Boston Meeting Minutes (robotics/2005-09-02)

Tetsuo Kotoku reviewed the roadmap and today's agenda.
- Opening Presentation (robotics/2005-09-03)

**Presentation: "Implementing and Teaching Emerging Robotics Standards at the University Level"**
Bruce Boyes introduced the educational activity using Embedded Hard Real-time Java for LEGO at Univ. of Utah., introducing his home developed control board which provides a very high level of functionalities while keeping its price as low as 100$ and could be a breakthrough for the development of large scale distributed robotic systems.

**Presentation: "Common Robot Interface Framework for Device Abstraction"**
Seung-Ik Lee talked about CRIF(Common Robot IF) and his attempt to define standardized interface to achieve the device abstraction necessary to achieve reusability and connectivity in the multiple robotic systems that are developed at ETRI in Korea.

**Presentation: "Standards in Action: Prototype Robots at Aichi International Exposition 2005"**
Masayoshi Yokomachi introduced the robotic systems that have been developed and displayed at the Aichi International Exhibition with the sponsorship of NEDO, and their applications.

**Special Talk: "Machine vision and actuators for robotics and automation"**
Kok-Meng Lee presented his present research activities at Georgia Tech. This included the necessity, possibility and achievement to develop a vision system that would be cheaper and more efficient than traditional Machine Vision Systems. This was followed by an argumentation on the advantage of the ACC(Artifitial Color Contrast) space versus the RGB space for image processing in environment incurring a large amount of noise and uncertainty in images like a chicken meat processing line.

**Presentation: "Slashing development time with component-based programming"**
Hung Pham's main point is that component-based development requires some technology to support the following aspects:

- component creation and management
- application modeling and development
- testing and integration support
- a run-time execution framework

... but that complex control systems have additional complexity ("periodic" and "event-driven" semantics, parallelism, distributed deployment over networks of varying topology, high testing and integration costs, etc.).

The speaker described a specific application that was developed for the Office of Naval Research to perform real-time control of crate loading equipment on ships while at sea.

**Presentation: "Korean intelligent robot standardization status"**
Korea went through the following stages in terms of robotics:

- Rapid expansion of industrial robots for factory automation in 1987-96

- The IMF crisis halted factory automation in 1997-2002
- A new focus, starting in 2003, is intelligent robots that can improve quality of life.

The world market for robots should increase from $150 billion in 2010 to $500B in 2020, but just in Korea it should grow from $10B (6%) to $100B (20%). The current market size is $1.4B, with $200M in robot exports and $110M in robot imports.

After this preamble, the speaker reviewed some of the architectural and standardization efforts taking place in Korea in this area. There is a Korea Intelligent Robot Standard Forum (KIRSF), but this poses the question on how this can be harmonized with standards from OMG, ISO, IEEE, or even ITU.

**Presentation: "Introduction to Toshiba Home Robots and Our Approach to RT Standardization"**
Fumio Ozaki showed a video demonstration of the ApriAlpha and AppriAttenda home assistant robots. ApriAlpha uses voice recognition (as well as the direction from which the voice comes) to interact with up to 6 users and perform tasks such as turning on and off Bluetooth-equipped appliances, reading the news, voicing alerts, etc. ApriAttenda recognizes a person and accompanies him/her in the middle of a potentially complex scene and potential obstacles. See:
http://www.toshiba.co.jp/about/press/2005_05/pr2001.htm

Toshiba's work is based on the ORCA (Open Robot Controller Architecture) framework, which uses the following technologies:
- Jython/Groovy
- Java
- JNI
- C++
- HORB, an Object Request Broker from AIST

The speaker said that robotic technology standardization faces several obstacles, one of which is that most embedded software developers do not seem to care, and even resist, advanced development technologies like Java.

**Robotics-DSIG / Robotic Systems RFI promotion**
- RoboNexus will be held on October 06-09
The Robotics SIG will hold BoF meeting at RoboNexus on Thursday from 6PM to 7:30PM
Other robotics related events will take place in Santa-Clara from the day before:
Arm Developers Conference www.arm.com/developersconference/
WiFi Conference www.wifiamericas.com/2005/

- Announcements
The Robotics-DSIG homepage has been updated and all presentation made during the meetings are available for download. Its URL is http://robotics.omg.org/.

- Development of Mediator (Liaisons)
Bruce Boyes offered to be the liaison with Java.net (which also deal with Sensor Networks)
Yun Koo Chung offered to be the liaison with KRISF

Jon Siegel announced that OMG have official liaisons and they will have a privilege of OMG resource access.

- Advertisement of the RFI
Bruce Boyes offered to put a link to the RFI on his blog
Rely in JARA to dispatch the RFI to the Japanese companies
Should contact IEEE for collaboration

RTI will contact American companies (especially during RoboNexus)
Olivier Lemaire proposes to write a formal introduction letter that should be emailed to organizations involved in the robotics field.

**Next Meeting in Burlingame**
Monday : Steering Committee
Tuesday : Robotics DSIG plenary (during which the co-chairs will be elected)

ADJOURNED @ 17:00PM

## Participants (Sign-in)

- Seung-Ik Lee (ETRI)
- Makoto Mizukawa (Shibaura Institute of Technology)
- Charles Rush (Objective Interface)
- Roy Bell (Raytheon)
- Claude Baudoin (Schlumberger)
- Bruce Boyes (Systronix)
- Fumio Ozaki (Toshiba)
- Seiichi Shin (University of Tokyo)
- Takashi Suehiro (AIST)
- Masayoshi Yokomachi (NEDO)
- Yun Koo Chung (ETRI)
- Jan Popkin (Telelogic)
- Olivier Lemaire (JARA)
- Tetsuo Kotoku (AIST)
- Hung Pham (RTI)
- Shihobu Koizumi (Hitachi)
- Kok-Meng Lee (Georgia Tech.)
- Victor Giddings (Objective Interface)
- Gerardo Pardo (RTI)
- Carlo Cloet (RTI)

Prepared and submitted by **Olivier Lemaire** (JARA) and **Fumio Ozaki** (Toshiba)

# Robotics/SDO DSIG
## Plenary Meeting

## December 6, 2005
## Burlingame, CA, USA
Hyatt Regency San Francisco Airport
Gr Peninsula C, Lobby Lv1





# Approval of Atlanta Minutes

- Ask for a volunteer (minutes taker)
  - Olivier Lemaire (AIST)
  - Seung-Ik Lee (ETRI)
- Atlanta Minutes review

[Robotics] We were waiting Robotic Systems RFI. We had one special talk (Prof. Kok-Meng Lee) and 6 presentations.

[SDO] We completed the Robot Technology Components RFP and voted to issue the RFP. It was approved to issue in PTC sponsored by MARS.

# Roadmap Review

- Robotics WG in **SDO-DSIG** :
  discussions about the SDO model for robotic applications.
  <focus on interoperability> **RFP**
- **Robotics-DSIG** :
  discussions about a wide variety of standardizations on robotics domain.
  <focus on its priority> **RFI => White Paper**

*visible*

***Two activities in parallel***





# Review Agenda

**Tuesday, Dec. 6, 2005** Gr Peninsula C, Lobby Lv1

**13:00-13:10 Welcome and Review Agenda**
**13:10-14:10 Special Talk:" Introduction to the Agent-SIG activities "**
**- James J. Odell (OMG Agent-PSIG)**
**14:10-15:00 " Response form RTI"**
**- Hung Pham (RTI)**
<break>
**15:20-16:10 " Response from Java.net"**
**- Bruce Boyes (Systronix)**
**16:10-17:00 " Response from SNU "**
**- Seongsoo Hong (SNU)**
**17:00-17:50 " Response form ETRI "**
**- Soo-Young Chi (ETRI)**

**Joint Meeting with MARS/RTESS**
**Thursday, Dec 8, 2005**
**13:00-14:30 (Bayside A)**

# Review Agenda

**Wednesday, Dec. 7, 2005** Sandpebble DE, Lobby Lv1

09:00-10:00 Special Talk: " High Assurance Security and Safety for Robotics "
- Joseph M. Jacob (OIS)

<break>

10:20-11:10 " Response form ETRI "
- Seung-Ik Lee (ETRI)

11:10-12:00 " Response from NEC"
- Yoshihiro Fujita (NEC)

<lunch>

14:00-14:40 Special Talk:"Open source robotic Control – teambotica1.0 "
- Regis Vincent (SRI International)

14:50-15:30 " Response form NTT"
- Ken-ichiro Shimokura (NTT)

15:30-16:10 "Response from ATR"
- Norihiko Hagita (ATR)

16:10-16:50 " Response from Toshiba"
- Miwako Doi (Toshiba)

<break>

17:00-17:20 Chartering Robotics Domain Task Force

17:20-17:40 Next Meeting Agenda Discussion, etc

17:40 Adjourn

Joint Meeting with MARS/RTESS
Thursday, Dec 8, 2005
13:00-14:30 (Bayside A)





# Organization

- Steering Committee
  the day before plenary (open meeting)
  next meeting agenda adjustment
  SIG/TF co-chairs and candidates, WG co-chairs, etc.
- "Contact" between related organizations
  call for volunteers
  one page PowerPoint presentation

**We needs volunteers**

# **contact** between the related organizations

Call for volunteers

One page PowerPoint presentation

post it robotics@omg.org 3weeks before the meeting

Two-minute presentation at the meeting

- **JAUS**: Hui-Ming Huang (NIST)
- **ORiN**: Makoto Mizukawa (Shibaura Institute of Technology)
- RTmiddleware: Tetsuo Kotoku (AIST)
- KIRSF: Yun Koo Chung (ETRI)
- NRF: Miwako Doi

FYI:

OMG official liaisons will have a privilege of OMG resource access.
Contact to the Liaisons Sub Committee, please.





# Next Meeting Agenda

February 13-17, 2006 (Tampa, FL, USA)

**Monday :**

Steering Committee

**Tuesday-Wednesday :**

**Robotics-DSIG Plenary Meeting**

- RFP response presentation (SDO-DSIG joint meeting)
- RFI response presentation
  - Olivier Lemaire (AIST)
  - Ricardo Sanz (Control Systems WG/RTESS)
  - Fumio Ozaki (Toshiba)
- RFI response summary and Chartering WGs
- Contact reports

# Roadmap for Robotics Activities

robotics/05-12-04 & sdo/05-12-04

| Item | Status | Atlanta Sep-2005 | Burlingame Dec-2005 | Tampa Feb-2006 | St. Louis Apr-2006 | Boston Jun-2006 | Anaheim Sep-2006 | DC Dec-2006 | POC / Comment |
|---|---|---|---|---|---|---|---|---|---|
| Charter on Robotics WG in SDO | done | | | | | | | | Kotoku(AIST), Mizukawa(Shibaura-IT) |
| Robot Technology Components RFP (SDO model for robotics domain) | In Process | RFP | | Initial Submittion | | | Revised Submittion | | Suehiro(AIST), Sameshima(Hitachi), Kotoku(AIST) |
| SDO model for xxx Domain | no plan | | | discussion | draft RFP | RFP | | Initial Submittion | TBD |
| | | | | | | | | | |
| Charter on Robotics SIG | done | | | | | | | | Kotoku(AIST), Mizukawa(Shibaura-IT) |
| Robotics Information Day [Technology Showcase] | done | | | | | | | | Yokomachi(NEDO), Kotoku(AIST) |
| Robotic Systems RFI [Robotics: Initial Survey] | In Process | | Response Presentation | Response Presentation | review whitepaper | review whitepaper | Whitepaper | | Lemaire, Chung, Lee, Mizukawa, Kotoku |
| Charter on Robotics TF | Planned | | discussion voting | | | | | | Kotoku(AIST), |
| | | | | | | | | | , |
| (Robot Middleware for Controller) | Future | | Grouping | Chartering WG | draft RFP | RFP | | Initial Submittion | to be discussed |
| (Robot Middleware for Specific Applications) | Future | | | Grouping | Chartering WG | draft RFP | RFP | | to be discussed |
| (Robot Middleware Common Services) | Future | | | Grouping | Chartering WG | draft RFP | RFP | | to be discussed |
| (Robot Middleware for Common Data Structures) | Future | | | | Grouping | Chartering WG | draft RFP | RFP | to be discussed |
| etc… | Future | | | | | | | | to be discussed |

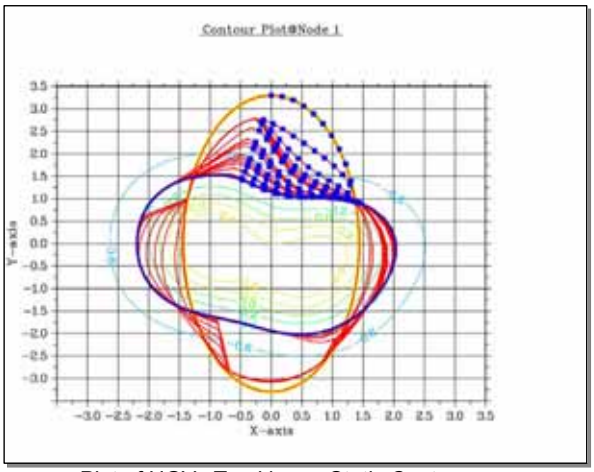
# Robots with agents!

**James Odell**
**Distributed Intelligent Systems**
**Intelligent Automation Inc**
**15400 Calhoun Drive, Suite 400**
**Rockville, MD 20855**
**www.i-a-i.com**

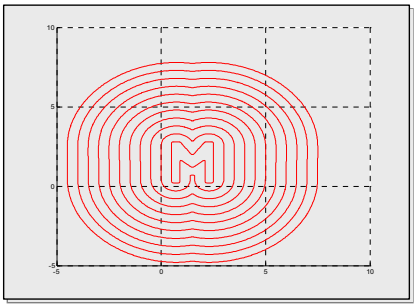


## Evolvable Curve Based Dynamic Tracking (ARO)

**Each robot running multiple agents under distributed autonomous infrastructure**




Plot of UGVs Tracking a Static Contour



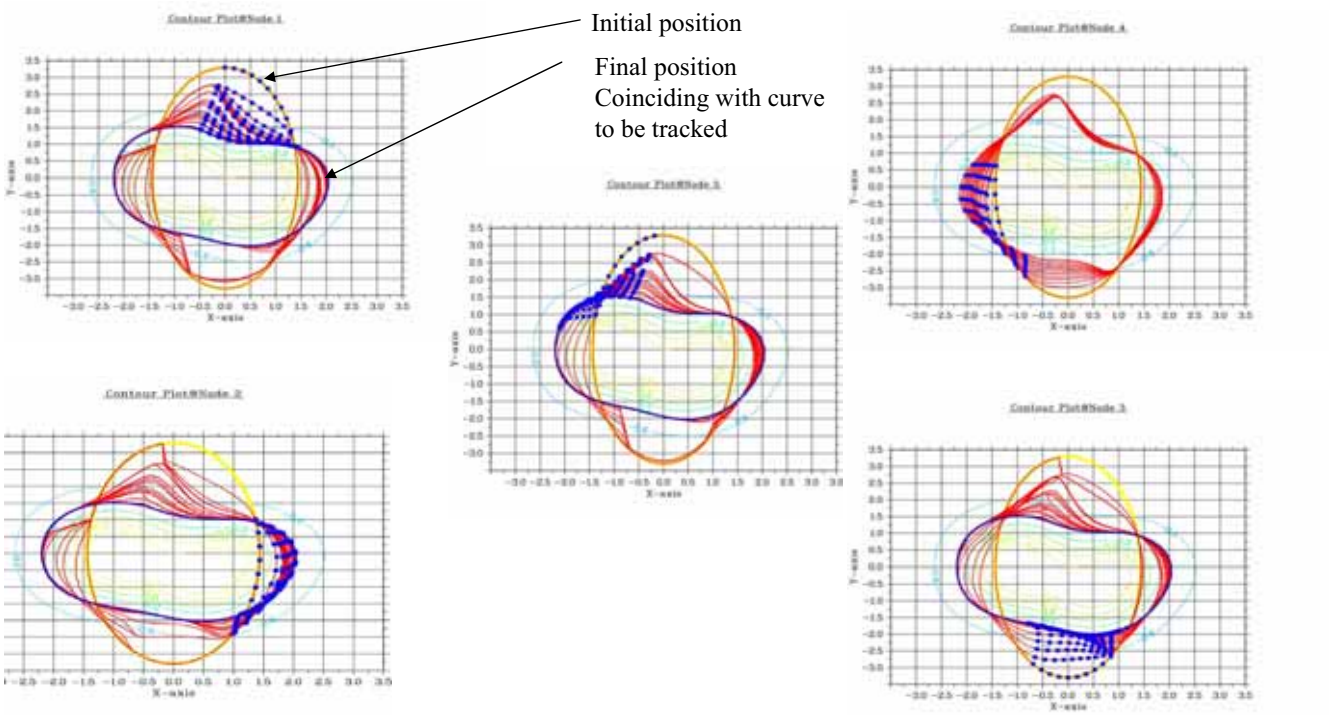
Test Goal: Evolution of front

**Amigobot**

- Dynamic front evolution
- Evolvable curve based swarming to track the front
- Cooperative ad-hoc network for robust communication
- Emergent sensing behavior
- Cybele based robot simulation and implementation

# Tracking in action!!

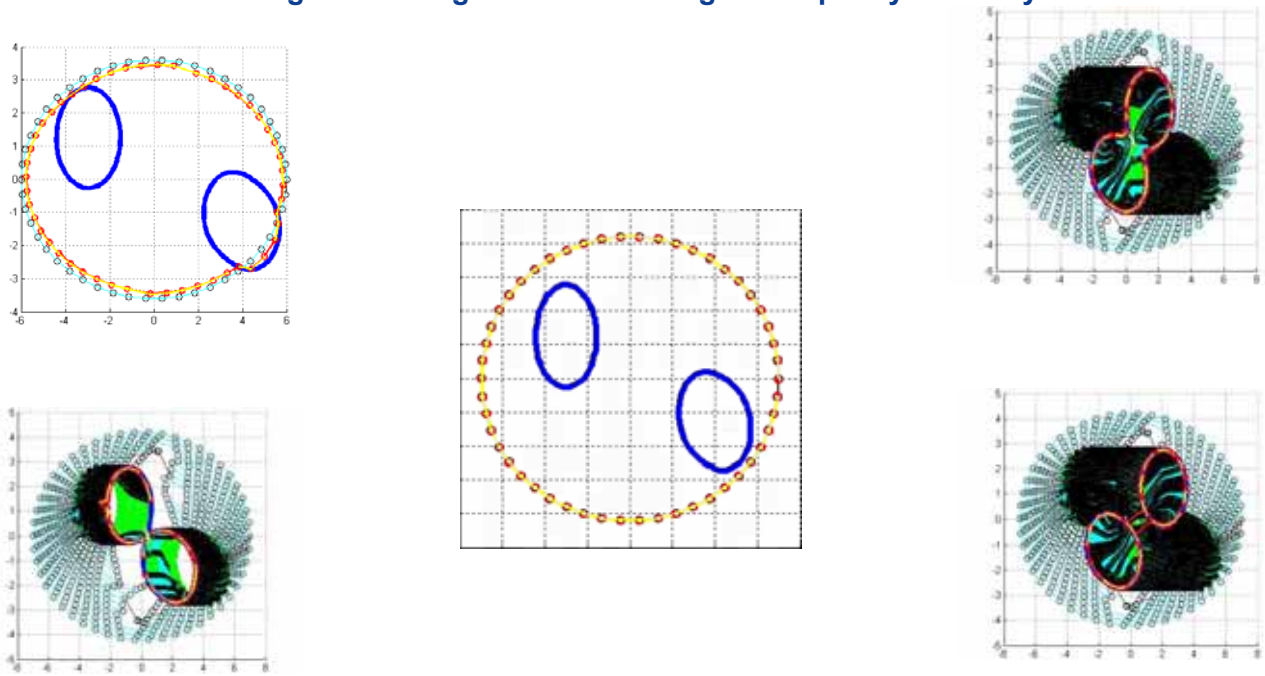
## Five groups of autonomous communicating robots tracking a static boundary






# Tracking in action as front moves!!

## Robots tracking the moving boundaries merge and split dynamically



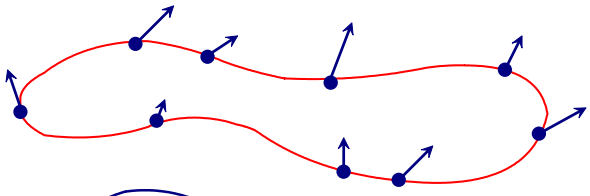
## Tracking Framework

- Describing an evolving boundary:
  - Euler versus Lagrange
  - Explicit convection and rescaling
- Sensor models
  - Types
  - Sensing requirements
- Tracking:
  - State and covariance
  - Observations
  - Propagation
  - Update
  - Control
- Implementation-System design, Integration and Testing

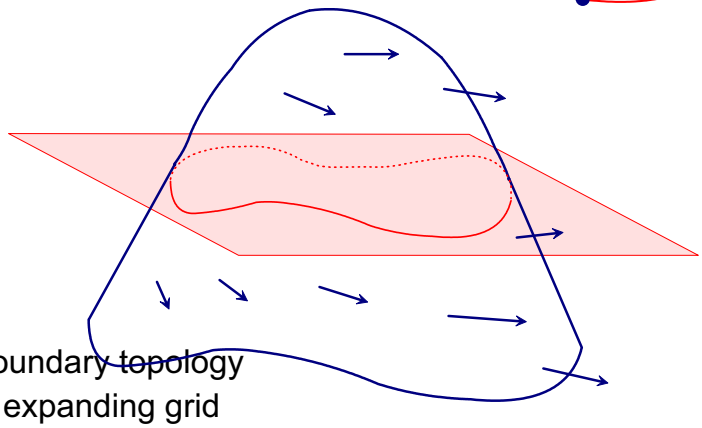


## Evolving Boundaries: how does it all work?

- Lagrange approach



- Euler approach



- Tradeoffs:
  - Changes of boundary topology
  - A shifting and expanding grid
- A mixed model: explicit convection and rescaling

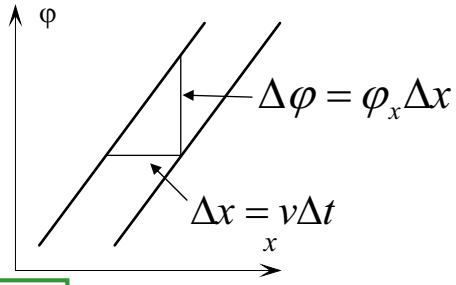
## Explicit Convection and Rescaling: Some math..

- Example: a diffusing cloud
- Level set formulation:

$$\varphi_t(x,y,t) = k(x,y,t)\Delta\varphi(x,y,t) + \mathbf{v}(x,y,t)\cdot\nabla\varphi(x,y,t)$$

Diffusion+Deformation — Convection

- Requires time step $\Delta t \propto \frac{\Delta x}{v}$
- Requires growing domain




- Explicit convection and rescaling:

$$u_t(\xi,\eta,t) = c(\xi,\eta,t)\Delta u(\xi,\eta,t)$$ Deformation

$$\varphi(x,y,t) = u\left(x(\xi,\eta,t), y(\xi,\eta,t), t\right)$$ Convection and Rescaling

$$|c(\xi,\eta,t)| << 1$$



## Approach: Control and Communication

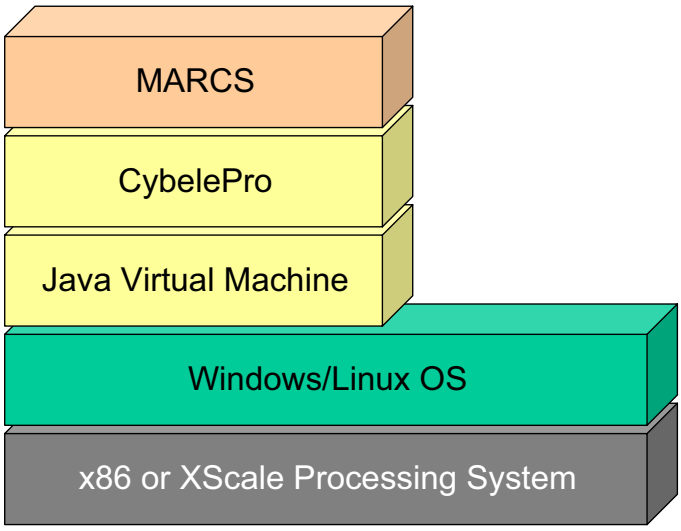
Network is crucial for distributed implementation of a group of robots!

## Communication and Control Challenges

- How can we integrate the problem of detection and communication ?
- How much does the communication among robots helps in tracking the boundary?
- Should the robots keep the flock together risking segmentation of the network? How to handle the merging and splitting and what are the “emotional” pains for networking/connectivity?
- Whats the optimal number of robots? From ad-hoc network results, capacity of an adhoc network scales like 1/n (Gupta and Kumar, 2000) where n is the number of nodes in an ad hoc network. Basic question is how do we co-ordinate the communication as number of robots increase.
- Use price based approach for ad-hoc network co-ordination (CDC-03)

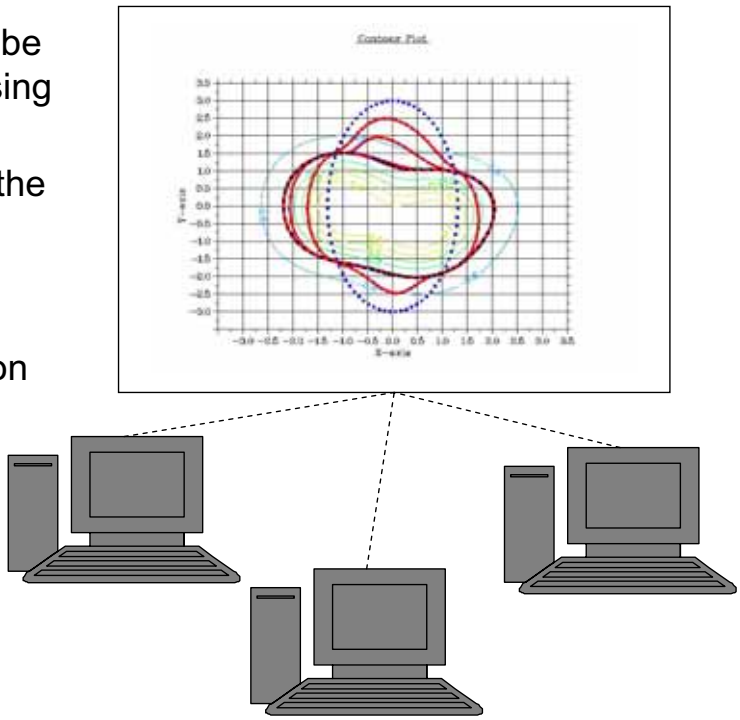


## System Architecture



## MARCS In Action

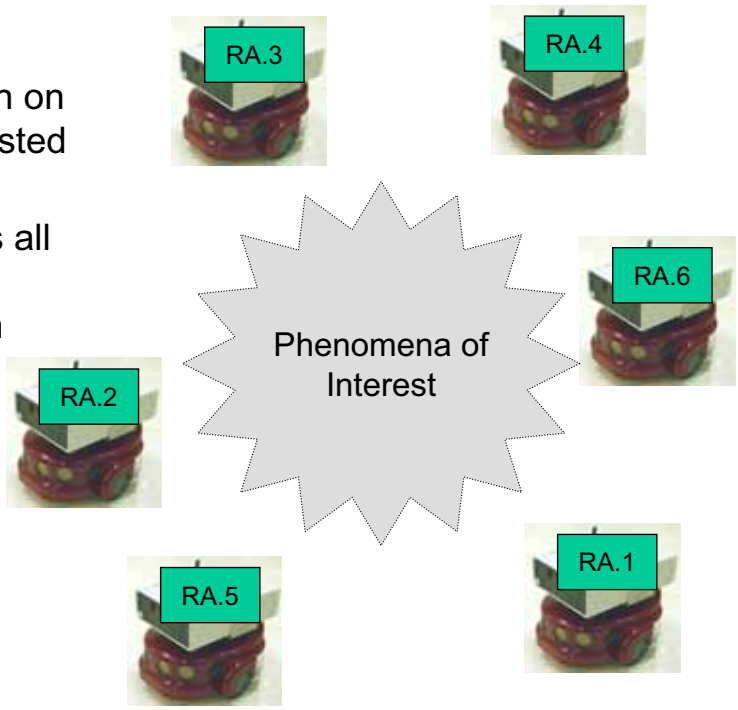
- Control algorithms would be designed and tweaked using pure simulation
- RobotAgents all run with the local or network clock
  - Individual agents can be started/stopped at will
- Since MARCS is built upon CybelePro it is inherently distributed





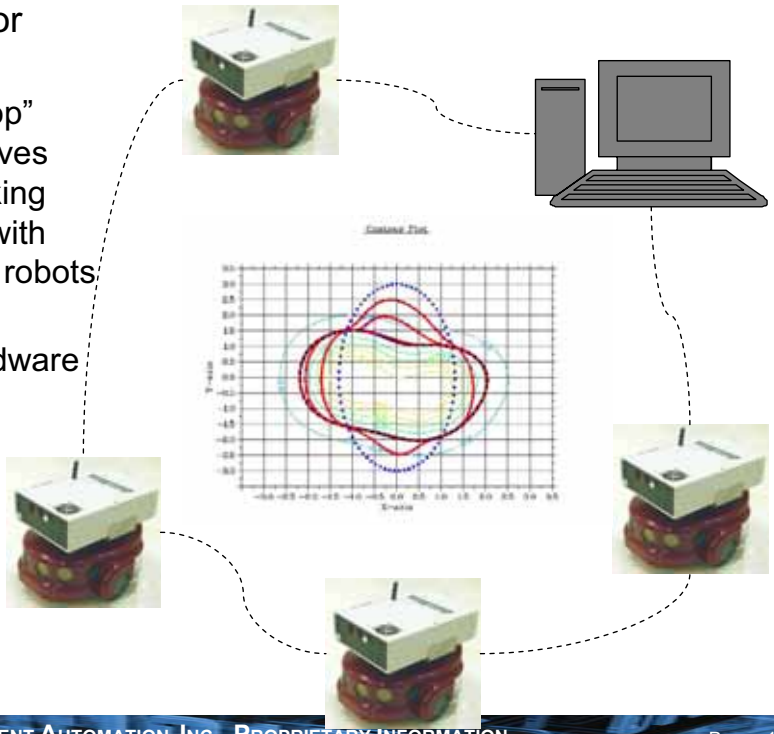
## MARCS In Action

- The code that ran the simulation can then be run on *real* RobotAgents, manifested by the platforms
- RobotAgent encapsulates all sensing and control into a stand-alone agent system



# MARCS In Action

- Future development for MARCS:
  - “Hardware-in-the-loop” simulation; this involves real robot agents taking part in a simulation with dozens of simulated robots
  - Expand the list of supported robot hardware




# On Robotics Middleware

**Dec 05 OMG Technical Meeting**

## Usage of Robotics Technology

- RTI provides software tools and services to developers of complex, distributed control systems.

| Humanoid Robot | Unmanned Vehicle | Transport/Hazmat | Medical |
|---|---|---|---|
|  |  |  |  |
| Anthropomorphic robotics, autonomous and telepresence control system testbed. | Unmanned ROV with robotic arms for undersea exploration. | Robotic crane for at-sea cargo transfer from/to ships. | Surgical simulator. |

# Robotic Technology Vertical Layer

| | Technology layer | User visible | Examples |
|---|---|---|---|
| T7 | **Application layer** – Integrated subsystems with user interface, mode switching; where applicable, automatic detection and self-organization capability | X | UML FSM automata realization for mode switching; GUI to interact with system; plug-and-play capability. |
| T6 | **Domain layer** – A subsystem composed of engineering domain-specific software | | Sensor processing; motion control; machine vision; world modeling; navigation; behavior generation |
| T5 | **Middleware layer** – application framework<br>• Executes software in nested components during runtime<br>• Provides unobstrusive access to each component during execution for tuning, monitoring, debugging<br>• For distributed system, shuttles data between hosts & synchronizing execution | | Constellation™ (COTS robotic design tool) runtime; custom developed infrastructure that allows interface with Matlab™; scripting tools. |
| T4 | **Data layer** – Platform-independent data and interface representations and real-time distribution of the data | | OMG IDL; CORBA; DDS; HLA; MPI; TCP/IP; shared memory communication |
| T3 | **OS layer** – Operating system level service or equivalent, providing such vital services as thread of execution & their real-time scheduling. | | Real-time OS, scheduler, resource (memory, I/O, etc.) manager |
| T2 | **Hardware abstraction layer** – Software/hardware interface | | Device driver |
| T1 | **Physical layer** – Physical device | X | Processor, A/D, D/A, encoder, data bus, button, switch, display, speaker |

# Robotic Technology Vertical Layer

| | Technology layer | User visible | Examples |
|---|---|---|---|
| T7 | **Application layer** – Integrated subsystems with user interface, mode switching; where applicable, automatic detection and self-organization capability | X | UML FSM automata realization for mode switching; GUI to interact with system; plug-and-play capability. |
| T6 | **Domain layer** – A subsystem composed of engineering domain-specific software | | Sensor processing; motion control; machine vision; world modeling; navigation; behavior generation |
| T5 | **Middleware layer** – application framework<br>• Executes software in nested components during runtime<br>• Provides unobstrusive access to each component during execution for tuning, monitoring, debugging<br>• For distributed system, shuttles data between hosts & synchronizing execution | | Constellation™ (COTS robotic design tool) runtime; custom developed infrastructure that allows interface with Matlab™; scripting tools. |
| T4 | **Data layer** – Platform-independent data and interface representations and real-time distribution of the data | | OMG IDL; CORBA; DDS; HLA; MPI; TCP/IP; shared memory communication |
| T3 | **OS layer** – Operating system level service or equivalent, providing such vital services as thread of execution & their real-time scheduling. | | Real-time OS, scheduler, resource (memory, I/O, etc.) manager |
| T2 | **Hardware abstraction layer** – Software/hardware interface | | Device driver |
| T1 | **Physical layer** – Physical device | X | Processor, A/D, D/A, encoder, data bus, button, switch, display, speaker |

# Standardization Opportunities

- Middleware layer offers biggest opportunity for standardization.
  - Middleware services are **mundane** and transparent to the end-user (*i.e.,* will not be an obvious product differentiator between 2 robotics systems)
    - Will not pose direct threat to customers of the standard, *i.e.*, robotic component developer or system integrator.
    - Solving the most common re-occurring problems.
  - Middleware does not (overly) constrain higher-layer software.
  - Details of runtime execution, diagnostic support, and data distribution are left to implementers of the standard.

# Motivations for a Standard

- To establish common language
  - Architects with object modeling background have object-oriented or component-oriented mental models.
    - *E.g.*, UML.
  - Engineers with controls background have data-oriented or signal-oriented mental models.
    - *E.g.*, Simulink, LabView.
  - A “robotics standard” can bridge gap and provide common language for mutual understanding and collaboration.
- Reduce time-to-market
  - Eliminate need for each vendor to develop and test own application framework.

# Motivations for a Standard (cont'd)

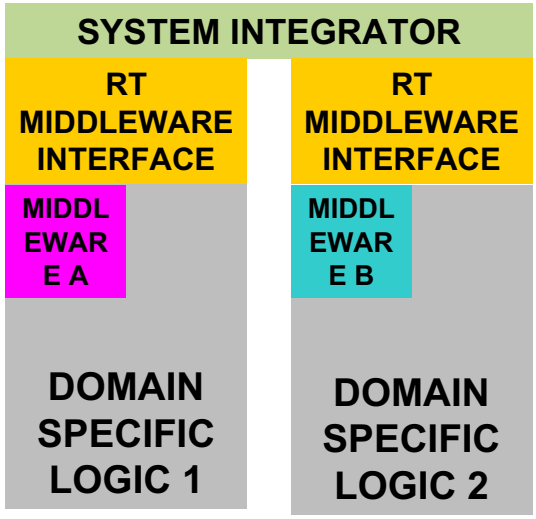
- Faster technology adoption, for less cost
  - Standard interface to components allow easier swapping of existing components
    - Lessens risks.
    - Fosters competition, innovation.
- More, better, cheaper tools
  - Designing, debugging, & maintaining real-time component-based software is difficult.
  - Software tools can help reduce pain, but need standard interfaces to the infrastructure in order to achieve economies of scale.

# Requirements for Robotics Middleware

- R1: Offer platform, vendor independent standard interface
  - compile-time compatible
- R2: Extendable components
- R3: Real-time executable with minimum OS support
- R4: Real-time interactable

# R1: Standard Interface

- Common operations that are done similarly (but slightly differently) by every implementer
  - Real-time execution
  - Sorting
  - Behavior change
  - Querying
  - Debugging
- But still allow performance/architectural differentiation




# R2: Extendable

- In OO sense; “programming by difference”
  - Derive from it
  - “Compose-able”
    - more instance
    - change execution order
- Apply to both data driven and event driven execution

## R3: Real-time Executable with Minimum OS Support

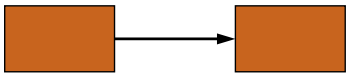
- Periodic and event-driven
- Can hook up any low-level rate/event generator to components
  - main( )
  - interrupt
  - thread

## R4: Real-time Interact-able

- Change behavior during runtime
  - Support intelligent behavior, and tuning a prototype
  - Atomic change of modes (i.e., execution paths) and parameters
- Log, playback, & analyze
  - Debugging and maintenance
- w/ right set of tools, set a break-point

# Standardization Points

- A middleware standard should have following (3) technical characteristics:
  - Component definitions and connection concepts from UML and CCM are valuable and should be retained.
  - Standard should support data-centric designs (ala Simulink and LabView) and data-centric communications (ala DDS).
    - In many controls application, the data being communicated between components can be more important to understanding of system than components themselves



  - Standard must be agnostic to underlying inter-component communications mechanism, *e.g.*, CORBA, DDS, shared memory, *etc.*

# Standardization Points for Interoperability

- Complete middleware standard must ensure that different implementations (of middleware and development tools) can achieve interoperability
  - **Platform independent model: d**escribes the component model and the programming interfaces necessary to navigate, inspect, and manipulate it.
  - **Graphical notation:** should leverage existing standards for component notation, but extend them as necessary for robotics software, *e.g.*,
    - Value assignable component attributes.
    - Periodic, data flow-based process model.
    - FSM style asynchronous event-based process model .
  - **On-the-wire:** the data layer of the robotics technology application stack.

# Who Is RTI?

- RTI has long supplied tools for software development
  - Complex controls.
  - Distributed application.
  - Real-time, multi-threaded applications.
- RTI has history in robotics
  - Spin-off of the Aerospace Robotics Laboratory in Stanford University.
  - Original founders were developing tools to help them develop their robotics application.
  - RTI's services organization currently provide consulting help to wide range of clients, ranging from military to industrial to surgical.


# How Can RTI Help?

- RTI has proven track record at OMG
  - Co-authors of original DDS RFP.
  - Co-authors of the DDS specification adopted in 2003.
  - Chair of the DDS FTF, completed in Mar 2004.
  - Chair of DDS Revision Task Force.
  - Providers of the first COTS implementation of the specification.
- RTI is committed to help achieve similar success with Robotics Technology middleware
  - Important part of RTI business.
  - RTI's proprietary robotics middleware, *Constellation*, can provide a good starting point for discussion.

SYSTRONIX
Embedded Java Spoken Here

robotics2005-12-07

# Self-Configuring Smart Java Robots through Plug and Play I/O Boards

**Bruce Boyes**

Technical Director, Systronix, Inc at www.jcx.systronix.com
Robotics community leader at www.java.net

java.net The Source for Java Technology Collaboration

http://community.java.net/robotics/
2005 December OMG Robotics DSIG

## Practical Robot Swarm Programming

XML makes robot I/O boards “plug-and-play”

XML tagging memory makes programming 'n' robots much simpler, especially as n→∞. You can maintain one code base which is shared by multiple (heterogeneous) robots. The same idea applies to maintaining complex factory I/O points [IEEE1451].

# What is java.net?

150,000 member online community

Java.net is a collection of online communities (>20), projects (hundreds), discussions, blogs, and other resources: see http://java.net/faq.csp.

Sponsored by Sun, with support from O'Reilly and Collabnet. For the java.net vision please see http://java.net/vision.csp



# What are Systronix and JCX

JCX = Java Control “Xystem”

Systronix is a long-time developer of embedded Java systems used in industry.

JCX is our modular, open architecture for university-level robotics, and industry.

We believe I/O tagging is essential for practical robotics, especially collaborative swarms.

# Program a robot swarm
## (in your lifetime and without losing your mind)

Topics in this presentation

Description of the robot swarm problem

Self-descriptive hardware

JCX hardware architecture & tagging

JCX tagging software

Robot Demonstration (tentative)



# How many robot code bases do you want to develop and maintain?

How about “one”, even with heterogeneous robots?

> “Right now we have two robots on Mars. What if we had 2,000? ...they would have to be autonomous... work on their own and cooperate and communicate. But nobody has any idea how to program 2,000 robots right now”.

James McLurkin, 2003 Lemelson-MIT Student Prize Winner
http://www.pbs.org/wgbh/nova/sciencenow/3204/03-talk.html

# Robot swarms

## Assumptions about Systronix heterogeneous JCX robots

- Common core architecture
  - Common CPU
  - Common pool of sensors and effectors
- Sensors
  - Light, touch, rotation, color vision, sonar, compass, human body, etc
- Effectors & mechanical construction
  - Wheels, tracks, air/water-borne, crevasse-crossing
- Communications capability
  - Local, long-range, gateways – JXTA, satellite,...



# I/O Tagging Requirements

- Knowledge of all possible I/O devices
  - Write I/O support one time, make it generic
  - Codebase can be incrementally expanded and distributed to all robots
- Some self-calibration at runtime
  - Example: each robot needs to be able to adjust its motor drive setting to speed, this cannot be hardcoded in the code base. However, it could be soft-coded in the tagging memory.
- Requires hardware support
  - Can't just add it on later in firmware

# What if I/O could self-configure?

Give each I/O board local memory for each I/O point

- Each board could store basic information
  - Manufacturer, board type, revision, etc
  - Runtime class(es) used to support this board
  - (Protect this data from accidental erasure)
- Each I/O point has specific information
  - Type of sensor (light, touch) or effector (wheel, tread)
  - Performance and calibration (thermocouple, etc) data
  - Space for a useful name such as “steering servo”
- Why not use XML to store this data?



# JCX hardware architecture & tagging

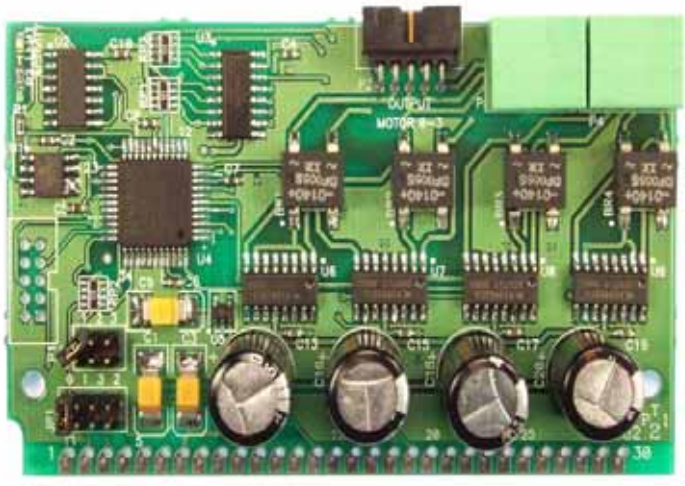
EEPROM tags are built into the I/O address space

- All I/O devices are SPI or $I^2C$ slaves
  - Each board or device uses 1 or more slave selects
  - 24-32 slave select addresses are available
  - Each slave select is typically 1-8 motors, sensors, etc
- Each SPI slave has “shadow” tagging memory
  - Tag memory exists in the lsb of each slave address
  - Enumeration thus must find tagging at each slave
  - Enumeration “tells all” about that device
  - CRC detects slave address conflict
- Same idea applies to $I^2C$

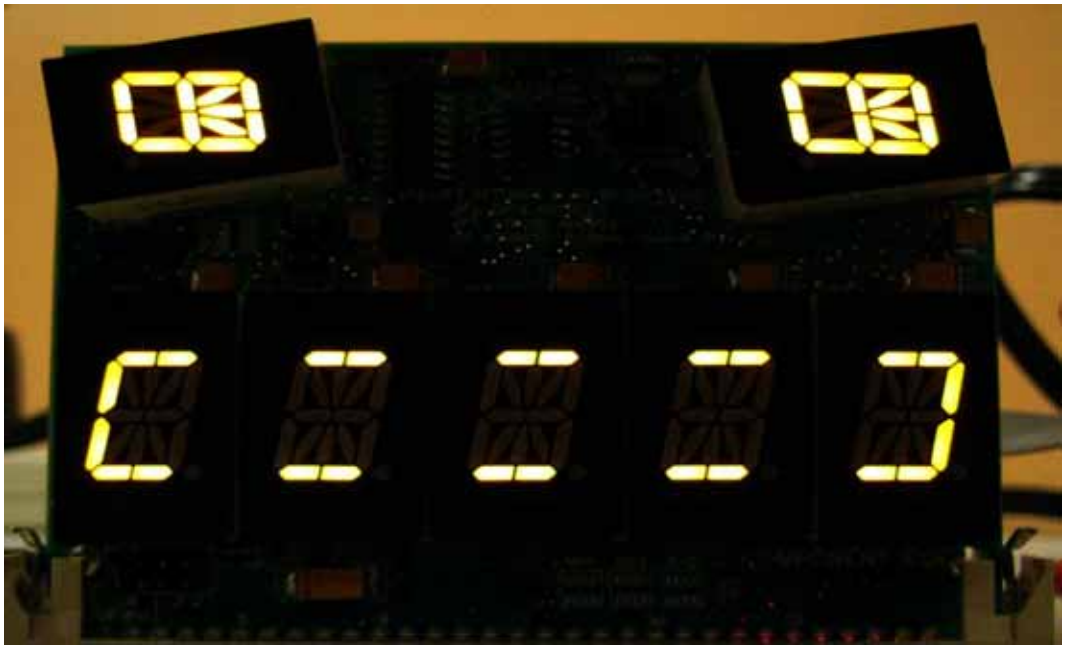
# JCX Sensor Board Photo



# JCX Motor Board Photo



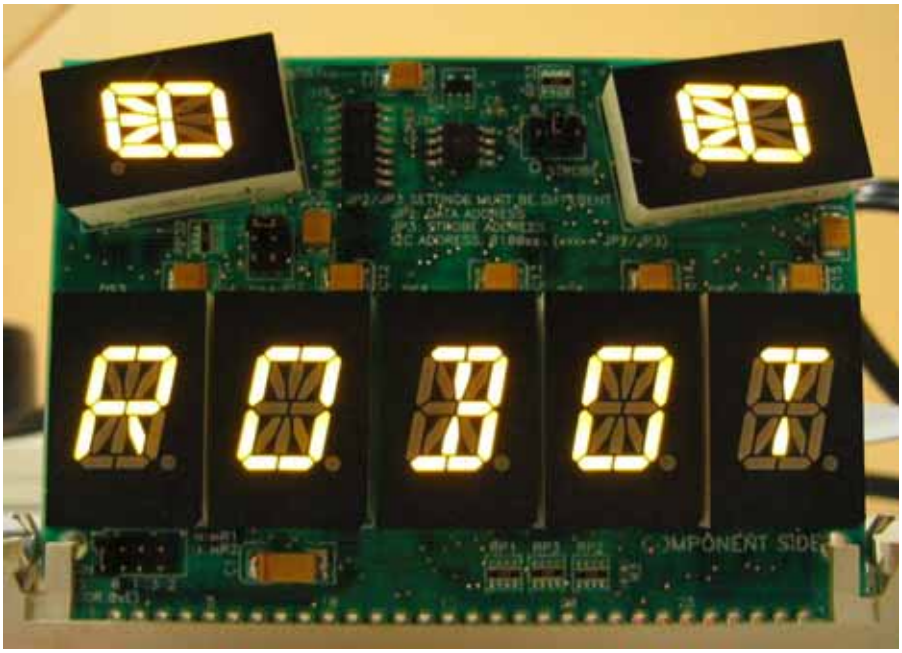
# JCX Robot Face Photo





# JCX Robot Face Photo

this photo shows all the display segments



# How to use this approach

- Support an I/O device:
  - Write the I/O driver
  - How is the I/O device attached?
    - Where is tag physically located?
    - I2C, SPI (motor), serial (CmuCam), ethernet, 1-Wire?
    - Connects to a plug-in board? Thermistor, motor, etc
  - Define tagging parameters for each device
  - Write support for enumeration for this I/O device
- Program the tag – sacred area 1st, then user
- Add use to application
  - Enumeration at startup,runtime binding of class
  - Read and use tag data



# JCX tagging software

- "Sacred" and "User" portions of the tag memory segregate vital and optional information
- Runtime class name is included for late binding
  - Class.forname is used on the "class" parameter
- Human readable XML text format
  - Easy to program, debug and parse
  - Lots of embeddable XML parsers exist

# JCX Tag Rules

- Required systronix tag parameters:
  - `tagsize=""` - size of eeprom
  - `pcbrev=""` - number of the pcb revision
  - `sn=""` - serial number (32 bit hex-encoded numbers only)
  - `dev=""` - device type (ie JCX.Sensor)
  - `crc=""` - crc check number (user will not see this value - stripped by verifier class)- 16-bit ASCII encoded hex
  - `numadd=""` - number of addresses between JCX address jumper settings
- parameters may appear in any order
- Optional sacred area tags
  - no restrictions (other than space restrictions)
  - Datasheet info (local data), like 1451 TEDS
  - URL to online reference (remote data)



# Sample XML Tag - “sacred” area

```
<?xml version="1.0"?>
<systronix tagsize="2048"
    pcbrev="4"
    sn="4294967296"
    crc="2AE3"
    numadd="2"
    dev="Quad Motor Board">
</systronix>
```

# Sample XML Tag – user area

```
<user id="Main Motor Board"/>
<jcx class="com.systronix.jcx.Motor" port="0">
    <tag>Left Tread</tag>
    <type>Lego Standard</type>
</jcx>
<jcx class="com.systronix.jcx.Motor" port="2">
    <tag>Kicking Mechanism</tag>
    <type>Lego Standard</type>
</jcx>
```



# More IEEE1451 support

Smart Transducer Interface Module (STIM) per 1451.2

- Format conversions
  - Some 1451 fields are binary; tag is ASCII XML
  - UUID, data units, etc
- Required STIM functions
  - Functional and channel addresses: used for control, status and data transfer
  - Triggers, interrupts, etc
- Options such as TEDS calibration data
- Wrappers for tagged devices
  - Abstract JCX object which can make any tagged JCX device appear to be fully 1451-compliant

# XML Benefits

- Open standards = good!
  - Even a mediocre standard is better than yet another proprietary format...
- Several compact parsers are available (kXML)
  - XML is also used by JXTA and other apps
- Parameters can appear in any order
- New data types can be easily added
- Human readable text is easy to debug
- Easy incorporation into Java and C applications



# Code now available online

- Code online at java.net in robotics community
- Early version, but is usable:
  - Tag memory simulator for use on PC, without actual robot hardware
  - Sacred and user sample routines
  - Enumeration
  - Documentation (in early release form)
  - XML parser
  - Concrete class for 25LC160 EEPROM

# Remaining work

- Adoption by lots of developers
- Improve documentation and examples
- Make tagging more like IEEE-1451
- Support for denser, flash tagging memory
  - 512 Kbyte ST M25PE40 devices ($1.30 budgetary)
    - Driver already written, just needs to be integrated with tagging
- Support for $I^2C$ memory devices
  - This is natural when the I/O device is $I^2C$



# Future directions

IEEE1451 contemplates these capabilities

- Tagging at the actual I/O device
  - Memory at the sensor and actuator. This would require all devices to have some smart, digital interface.
  - Sensors such as thermistors would be replaced with a smart digital module
  - It is possible to convey data over an analog interface for legacy devices [National Instruments]
- Small data sheet in the tagging device
  - Useful in offering services to other agents or robots

# ROBOT DEMO
## Coming Soon...

Multiple, non-identical robots sharing one code base in a swarm of Lego-based devices are under development now.

Proof of concept has been previously demonstrated.



## Summary

- Robot swarms can - and should - share code
  - 2000 robots with one (or few) code base(s)
- Robot hardware needs to be self-descriptive
- A $1 chip can easily “tag” all I/O board addresses
- XML data can be stored in tagging memory
- Each I/O point thus becomes plug-and-play
- It really works!
- Industry is moving in this direction (IEEE1451)

# Future Directions

- Configuration information in the sensor/effector
  - IEEE1451 is one approach
- Open standards for robotics
  - Standards are a wonderful thing...everyone should have their own!
- Verification of robot code base correctness
- Dynamic reconfiguration
  - Robots configure to meet the “task du jour”
  - Adapt to loss of team members



# References

Items referenced:


- http://community.java.net/robotics/
- http://www.jcx.systronix.com
- http://www.ajile.com
- http://kxml.sourceforge.net/
- JavaOne 2005 TS-1464, available online
- http://www.ieee.org – 1451 specification
- *Creating Socially Interactive Robots*, Cynthia Breazal

# More on Related Topics

Standards and swarm research

- http://www.jdroid.com
- http://www.omg.org/robotics-corner/2.htm
- http://news.zdnet.com/2100-9584_22-993385.html
- http://www.inel.gov/adaptiverobotics/default.shtml



# Q&A

Or email: bboyes@systronix.com
please include 'robot' in the subject

SYSTRONIX
Embedded Java Spoken Here

# Self-Configuring Smart Java Robots through Plug and Play I/O Boards

**Bruce Boyes**

Technical Director, Systronix, Inc at
www.jcx.systronix.com
Robotics community leader at www.java.net

robotics2005-12-08

# RSCA: Robot Software Communications Architecture

Dec. 6, 2005

Seongsoo Hong

Response from
Real-Time Operating Systems Lab,
Seoul National University
Http://redwood.snu.ac.kr

## Contents

# Background



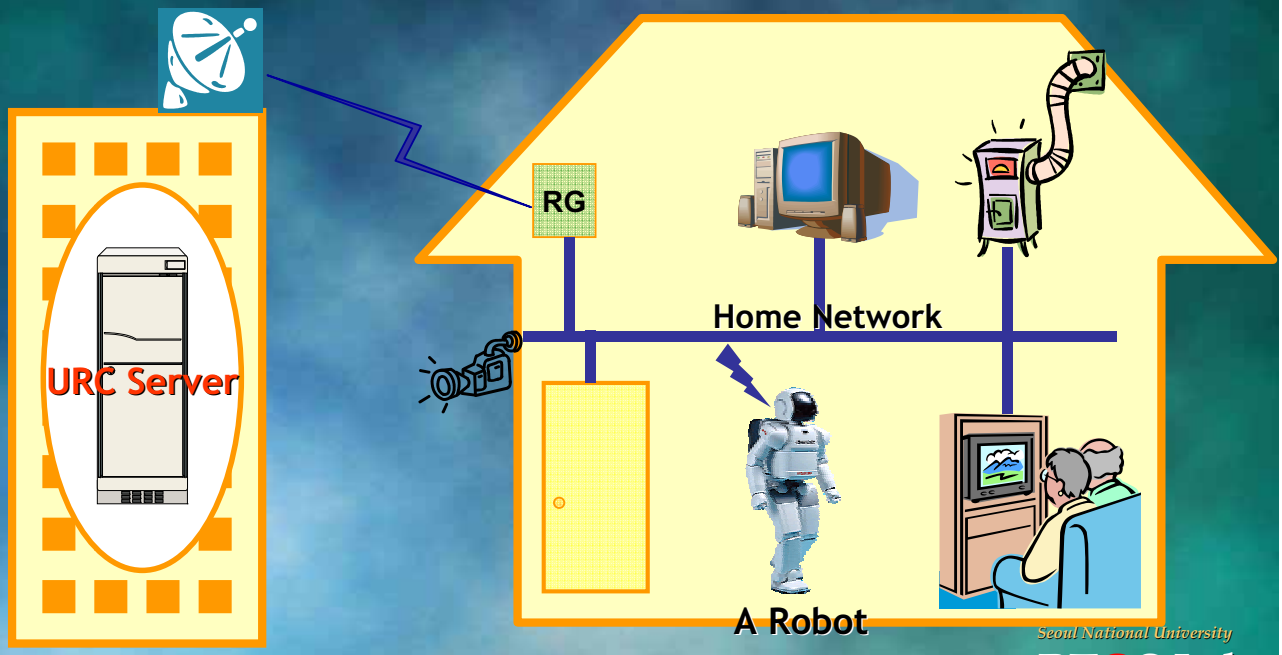


## National Robotics Project in Korea

- Project Name
  - URC (Ubiquitous Robotic Companion) project
- Goals
  - Putting networked home service robots into practical use
  - Development of inexpensive and affordable robots
- Approaches
  - A robot as a thin client by utilizing high capacity remote servers

# Networked Home Service Robots (1)

## : Robot is a part of an overall distributed system

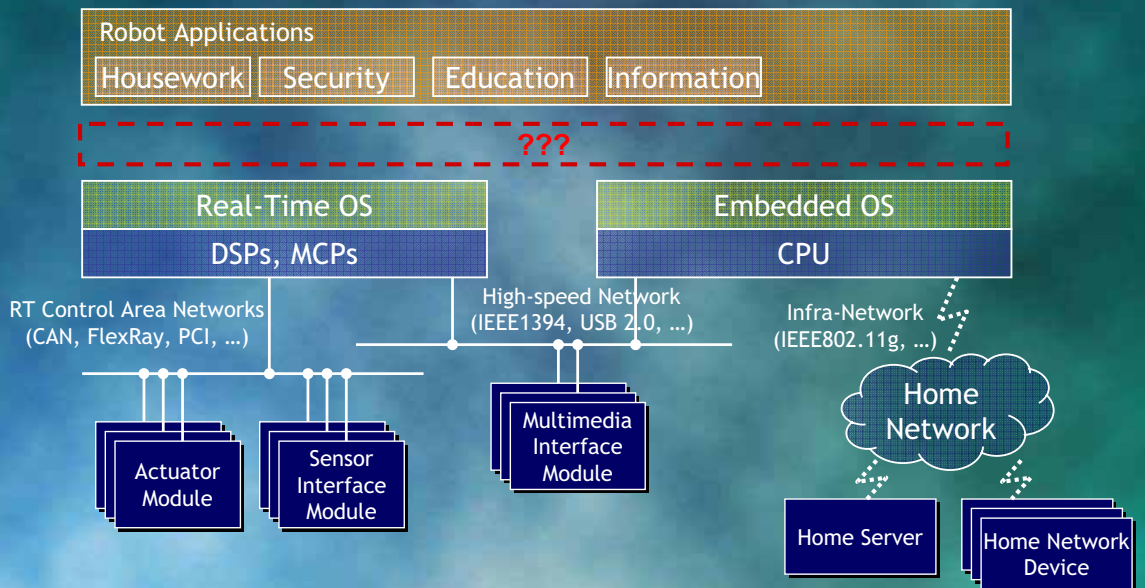





Seoul National University
RTOS Lab

# Networked Home Service Robots (2)

## : Robot makes a heterogeneous distributed system by itself






Seoul National University
RTOS Lab

# Technical Challenges and Solutions

- Challenges of robot SW stem from 'complexity'
  - Complex distributed system
  - Diversified services

**Solutions**

- Sophisticated software platform supporting
  - Distributed nature of hardware and software
  - Component-based development of software
  - Dynamic deployment and reconfiguration
  - Real-time and QoS capabilities
- Standardization of the platform to serve the robotics community at large





# What is RSCA? (1)

- Standard operating environment for URC robot software
  - Consist of
    - Standard real-time operating system
    - Standard distribution middleware
    - Standard deployment middleware
  - Serve as such a software platform

# What is RSCA? (2)

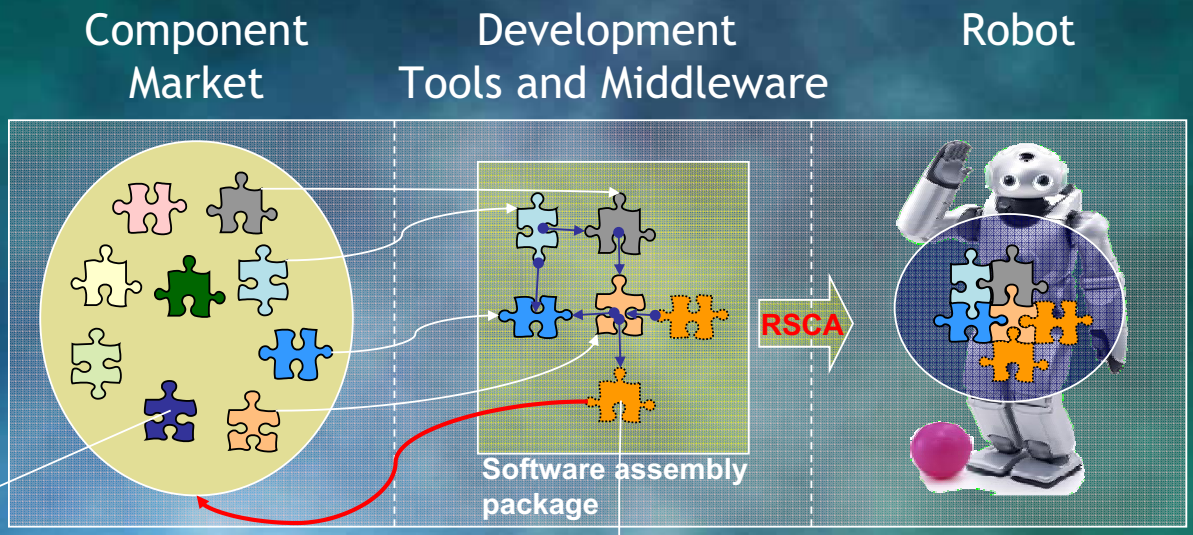
- RSCA is developed based on existing distributed object technologies
  - Joint Tactical Radio Systems (JTRS) SCA
    - De facto standard middleware adopted by Software Defined Radio (SDR) forum
    - OMG also adopted it and is making the Software Based Communications standard (SWRadio PIM and PSM)
  - OMG CORBA
    - Adopted as standard distribution middleware in SCA





# Visions of RSCA Standardization

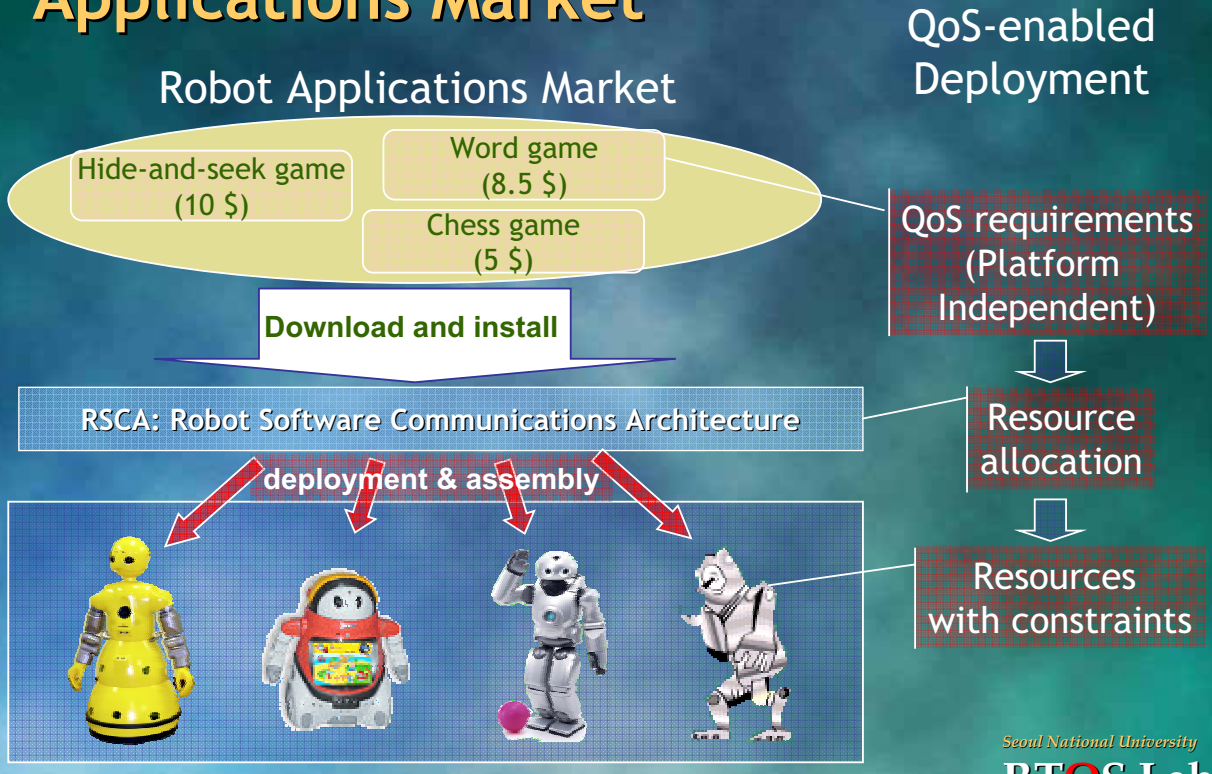
# Scenario 1: Creating New Robot SW Components Market






Seoul National University
RTOS Lab

# Scenario 2: Creating New Robot Applications Market






Seoul National University
RTOS Lab

# Scenario 3: Support Run-time Reconfiguration

- Dead Reckoning

"Going to the door" application

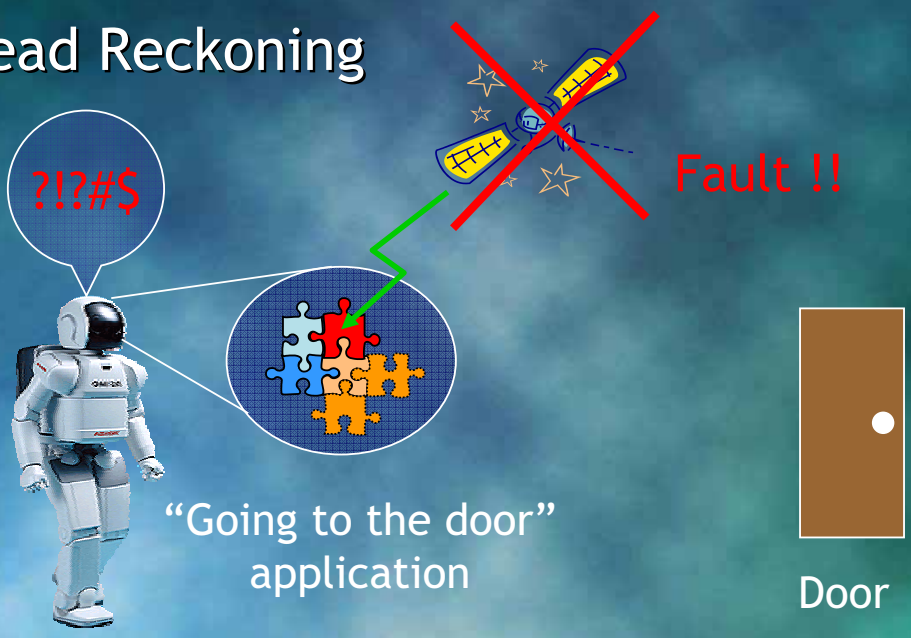
Door





# Scenario 3: Support Run-time Reconfiguration

- Dead Reckoning

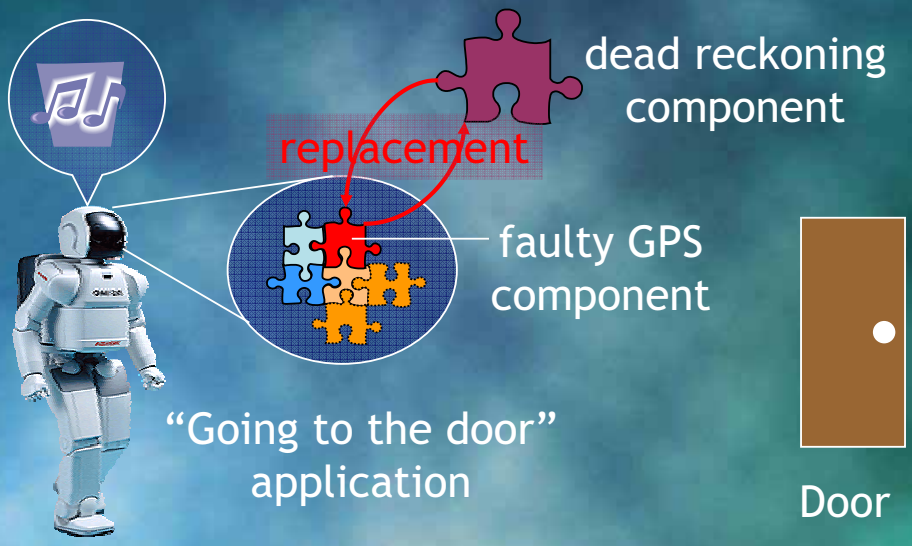



"Going to the door" application

Door

# Scenario 3: Support Run-time Reconfiguration

- Dead Reckoning



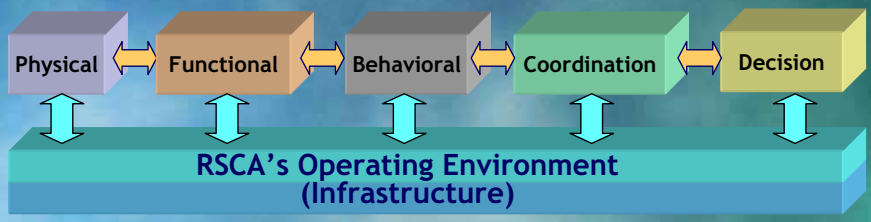





# Overall Structure of RSCA Operating Environment

# Operating Environment (1)

- Software bus
  - Through which application components are deployed and connected, and communicate
  - Without concerns for the particular implementation or structure of the target system
- Deployment and assembly engine
  - Automated deployment and assembly of component-based distributed applications








# Operating Environment (2)

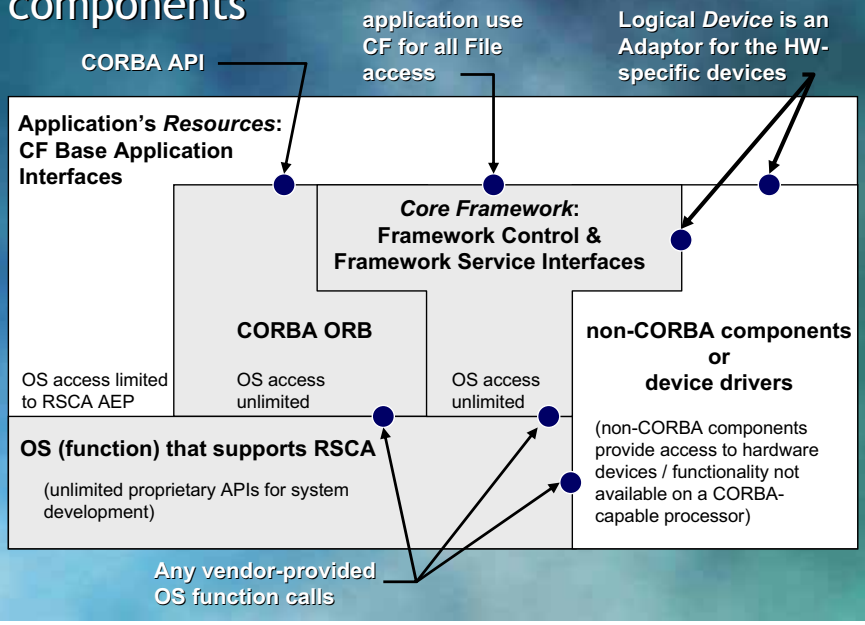
- Standardized set of APIs
  - Operating System
    - Compliant with at least PSE52 (IEEE POSIX.13 Real-time Controller System Profile)
  - Distribution Middleware and Services
    - Minimum CORBA and Real-Time CORBA v1.1
    - CORBA Services (Naming, Log, and Event)

Any COTS SW can be used

  - Deployment Middleware (a.k.a. core framework)
    - CF control and CF service interfaces

# Operating Environment (3)

- Standardized to maximize the portability and reusability of an application and application components








# Operating System

- Based on POSIX.13 real-time controller system profile
  - PSE: Generic Profiles for (System) Environment
  - PSE5: Realtime Environments

RSCA's RTOS requirements

| | PSE51 | PSE52 | PSE53 | PSE54 |
|---|---|---|---|---|
| | Minimal Realtime | Realtime Controller | Dedicated Realtime | Multi-Purpose Realtime |
| Target System | Control one or more specific IO devices without user interaction | | | Mix of real-time and non real-time tasks, interaction with users |
| Target HW | Single processor without MMU | | Single/multi-processor, with/without MMU | High-speed storage, display, network devices, MMU |
| Multi-tasking | Single-process (Multi-thread) | | Multi-process | |
| File System | None | Minimal interfaces | No directories | Full file system support |

# Distribution Middleware (1)

- Distribution middleware provides
  - A standard way of communication, while hiding heterogeneities that exist in
    - Heterogeneous HW platform
    - Diverse OSes
    - Diverse network topologies
    - Diverse communication protocols
    - Diverse implementation languages
  - A standard way of using various services
    - Naming, logging, and event services
- Defined as compliant with minimum CORBA and RT-CORBA v.1.1





# Distribution Middleware (2): Real-Time CORBA v.1.1

- Priority mechanisms
  - Priority mapping: between native and CORBA priority (both are static priority)
  - Two priority models: server declared vs. client propagated
- Thread pools (with lanes)
- QoS for inter-ORB communication
  - Provide server/client-side protocol properties
  - Explicit binding
  - Priority Banded Connections
- Prevent priority inversion: RT Mutex

RSCA's Distribution Middleware Requirements

# Deployment Middleware

- Enables dynamic deployments and reconfigurations including
  - Dynamic installation/uninstallation
  - Dynamic instantiation/tear-down
  - Dynamic start/stop
  - Dynamic configuration
- Specified in terms of
  - A set of interfaces called core framework interfaces
  - XML descriptors called domain profiles





# Deployment Middleware (Core Framework)

# RSCA Core Framework

- Serves as deployment middleware
  - Dynamically deploys component-based robot applications
    - Component model
    - Dynamic deployment





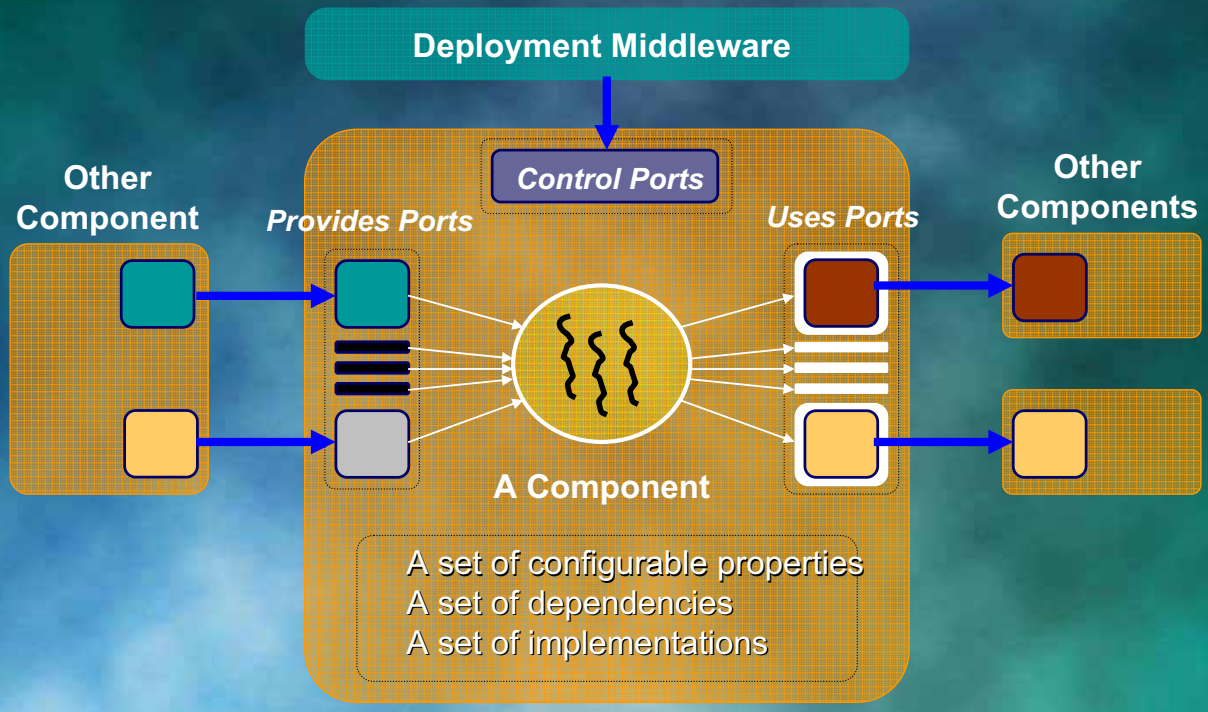
# Component Model (1)

- A RSCA component is
  - A unit of reuse
  - A unit of deployment
  - A unit of replacement
- Definition of component in general terms

*"a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."*

*by Clemens Szypersky, ECOOP'96*

# Component Model (2)






Seoul National University
RTOS Lab

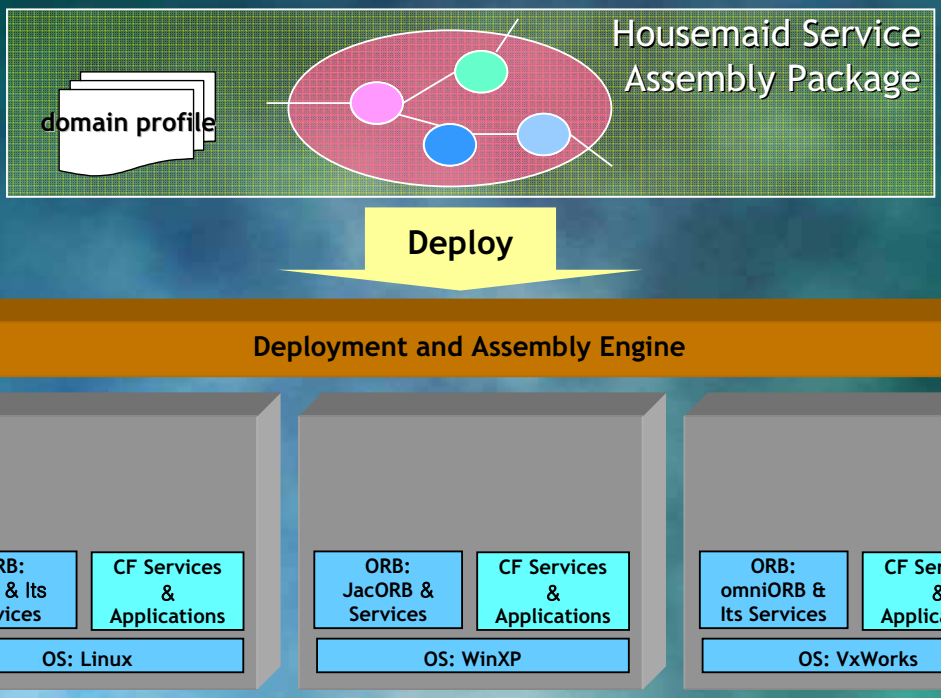
# Component Model (3)

- What is a port ?
  - Used to provide (use) services to (from) other components
  - A port implements a set of port-specific interfaces for other components
- RSCA's component at least
  - Should provide control ports implementing a set of interfaces for management purpose
  - Should be described in a set of XML descriptors
    - Called domain profile



Seoul National University
RTOS Lab

# Dynamic Deployment of RSCA CF (1)



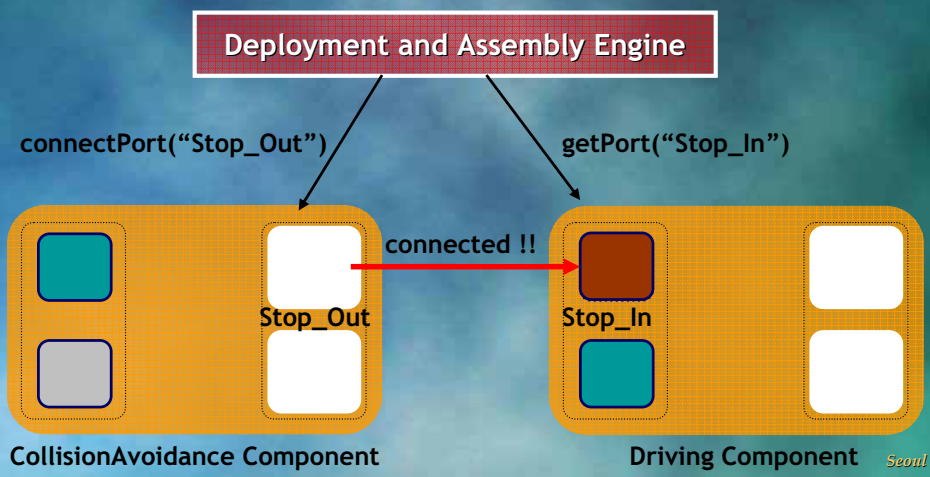

Based on the information described in the domain profile





# Dynamic Deployment of RSCA CF (2)

- Dynamic deployment is based on the port (CORBA object) and IOR



# What does RSCA CF Specify ?

- Core framework interfaces
- Domain profile

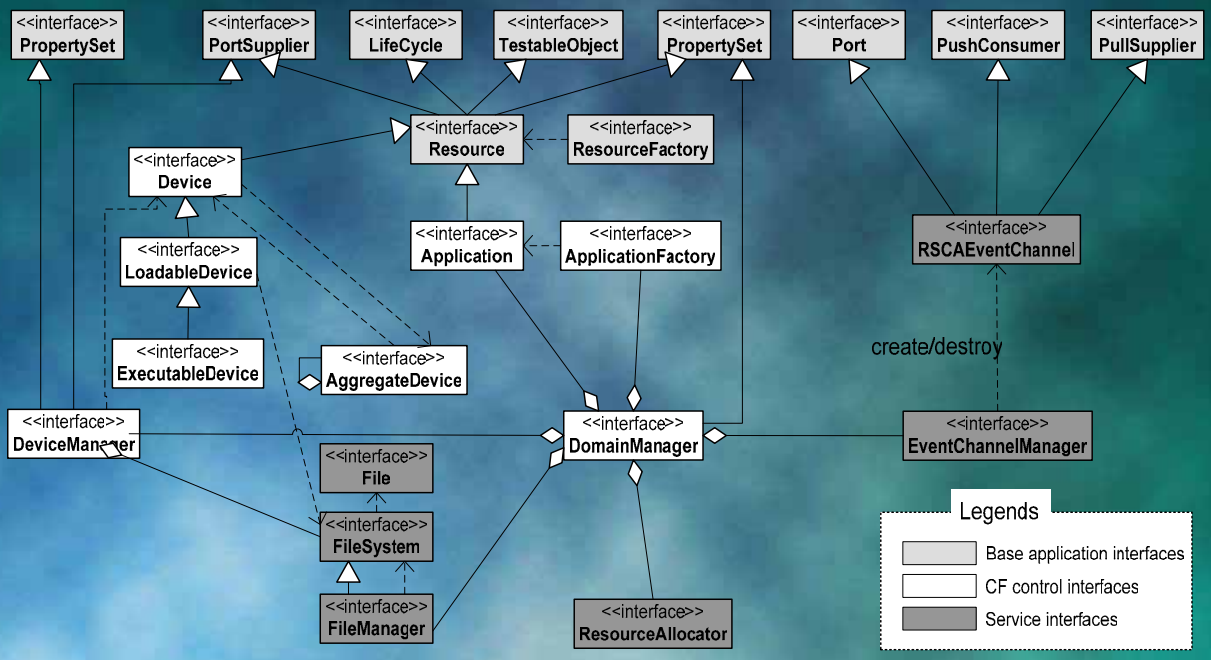




# RSCA CF Interfaces (1)

- Consist of
  - Base Component Interfaces — Application
    - For exchanging information between software application components
  - CF Control Interfaces — Deployment Middleware
    - For the start-up, control, and tear-down of software application components, and the allocation and control of hardware assets
  - CF Service Interfaces
    - For distributed file access service, event logging service, and QoS service for software application components

# RSCA CF Interfaces (2)

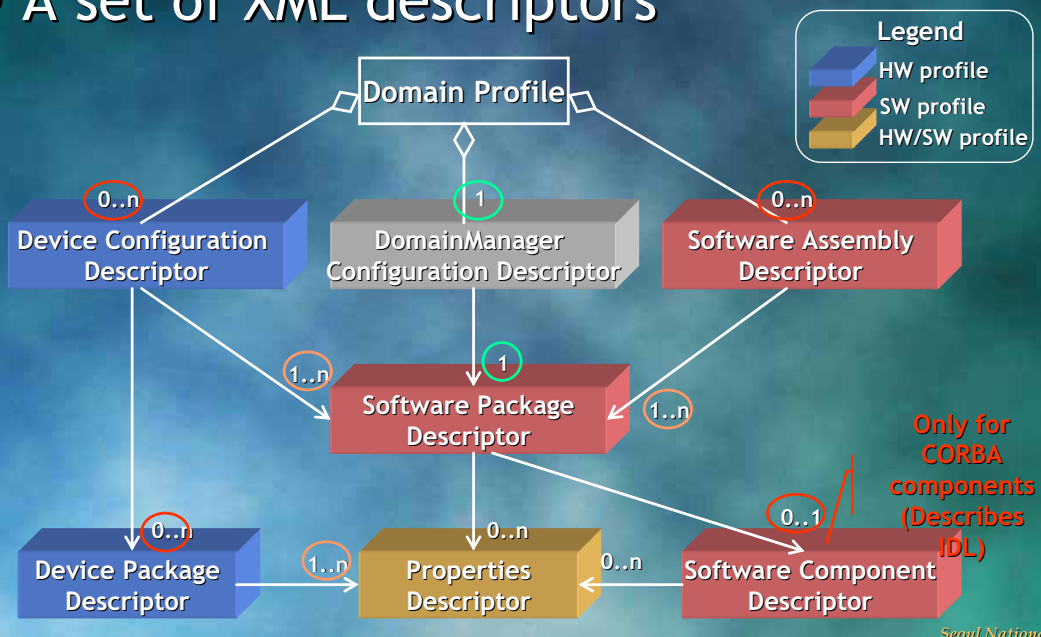





Seoul National University
RTOS Lab

# Domain Profile (1)

- A set of XML descriptors

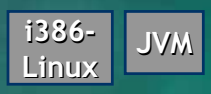


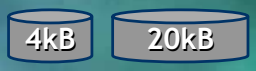



Seoul National University
RTOS Lab

# Domain Profile (2)

- Describes (1) the individual components of a SW application, (2) their interconnection, (3) the properties
  - Which components exist
  - How components are connected
  - Which pre-installed components are required
  - What kind of operating environment is required (operating system, CPU type)
  - How much resources should be allocated (MIPS, memory, storage)








# QoS and Event Support

# QoS and Event Support

- We have significantly extended SCA for QoS and event support
  - Home service robots are heavily involved in real-time signal processing such as vision and voice processing
  - There are several critical problems in using CORBA event services

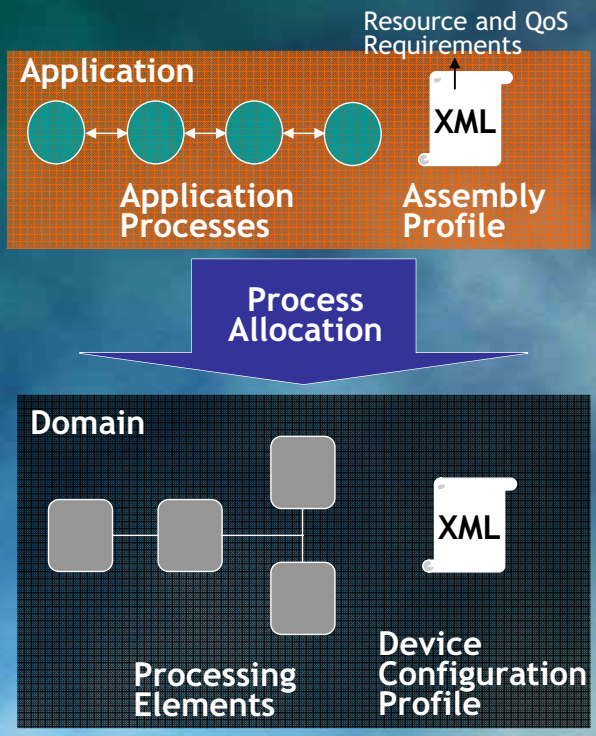




# QoS Support (1)

- Extended SCA for QoS specification and enforcement
  - Extend domain profiles to allow for describing resource and QoS requirements — Application
  - Provide new services for admission control and resource allocation
  - Modify application creation process — RSCA Deployment Middleware
  - Use RT-CORBA v1.1 to get static priority scheduling and prioritized communications — RSCA Distribution Middleware

# QoS Support (2) : RSCA CF Extension




- *ResourceAllocator* service
  - Newly added CF service
  - Integrated admission controller and resource allocator
- Modified application creation sequence
  - (1) [ResourceAllocator] tests feasibility of the admission
  - (2) [ResourceAllocator] generates a resource allocation mapping
  - (3) [ApplicationFactory] realizes the resource allocation





# Event Support (1)

- Problems in using CORBA event service
  - Reusability of component is seriously damaged
    - Hard-coded event channel names
    - Naming conflicts might occur
  - It is very difficult for non-CORBA expert programmers to use
    - A significant amount of manual coding
  - It is required to manage lifecycles of event channels manually
- We have extended SCA to overcome those problems

# Event Support (2)

- Extend SCA for event support
  - Extend SAD to allow for describing connections using event channels — Application
  - Introduce interfaces to use event channels
  - Provide new service dynamically creating/destroying event channels
  - Modify application creation process — RSCA Deployment Middleware





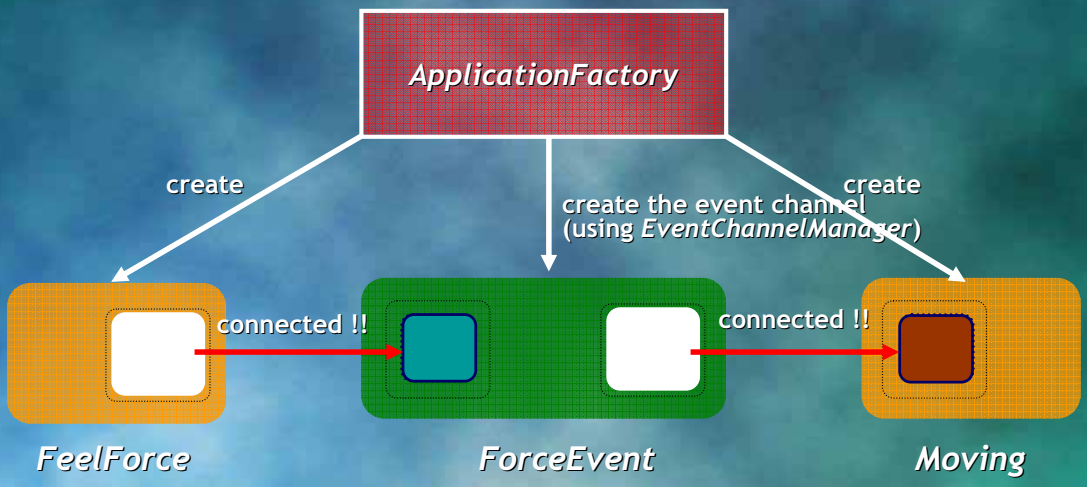
# Event Support (3):
## How to Use RSCA Event Channel

- E.g. *FeelForce* wish to send data to *Moving* via an event channel called *ForceEvent* using *push* model


1. *FeelForce* implement *connectPort()* interface accepting IOR of the event channel (*FeelForce* publish data to the event channel by invoking *push()* operation on the IOR)
2. *Moving* implement *push()* operation to subscribe the data
3. Describe the connections in SAD

# Event Support (4):
## Modified Application Initiation Process

- *ApplicationFactory* does following








# Implementation

# RSCA Core Framework Implementation

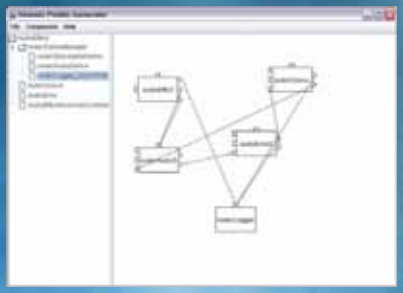
- We have fully developed the RSCA
  - Full-featured C++ implementation of the RSCA specification
  - Linux v.2.4.20
  - TAO RT-ORB v.1.3.1
- We are incorporating RT-CORBA features into omniORB v.4.0.5 for its use in RSCA
- Visit http://rsca.snu.ac.kr for more information





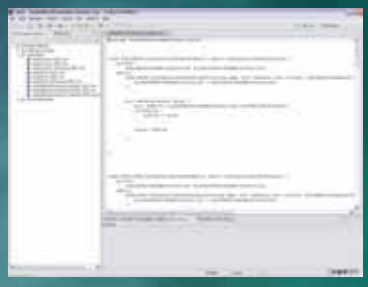
# Tool Support

- We are developing tools for supporting the proposed RSCA framework to show its utility for robot software



Model-based design tool

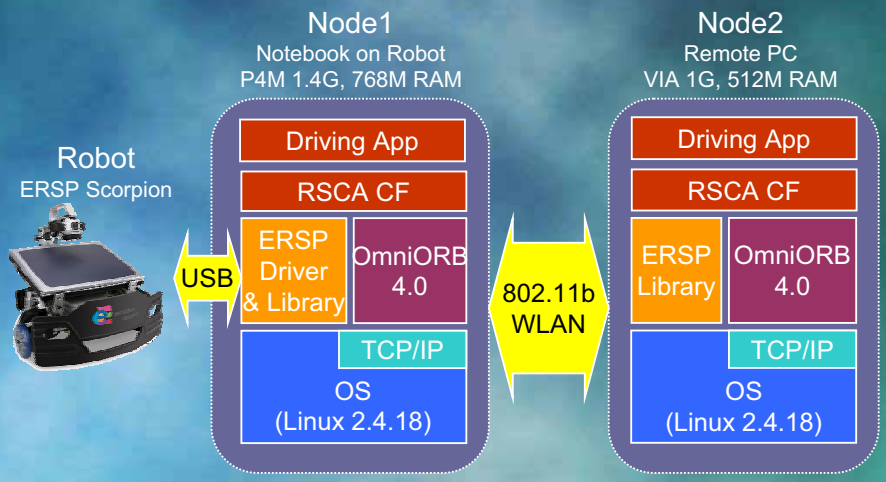
generated domain profile (XML)



CORBA stub code (C++)

# Performance Evaluation of RSCA with ERSP SDK (1)

- Implemented driving prototype system
  - Target platform: Scorpion robot from evolution robotics with ERSP™ SDK

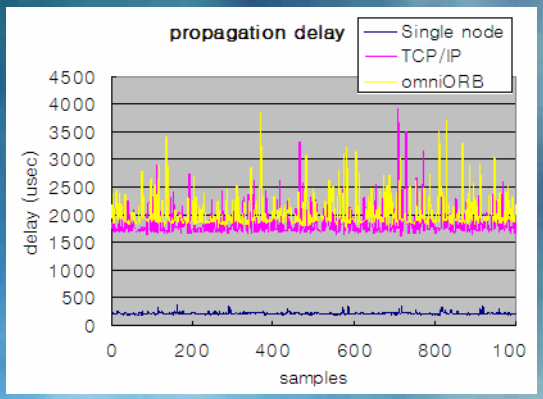


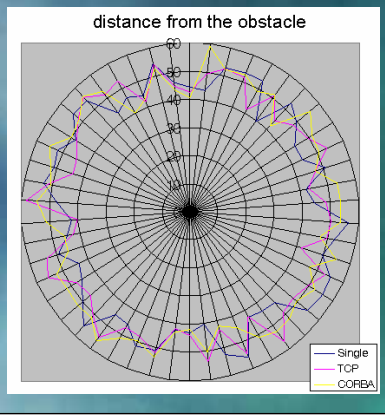



Seoul National University
RTOS Lab

# Performance Evaluation of RSCA with ERSP SDK (2)

- Scenario 1
  - Robot stops moving if an obstacle is detected within 60 cm from the robot




| Single Node | TCP/IP | CORBA |
|---|---|---|
| 222.2 us | 1872.6 us | 2045.9 us |

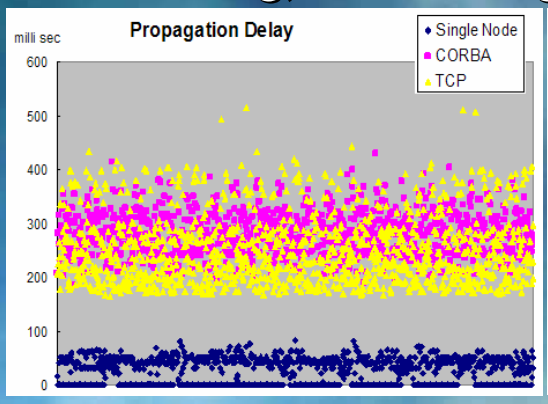
| Single Node | TCP/IP | CORBA |
|---|---|---|
| 48.35 cm | 48.67 cm | 49.97 cm |



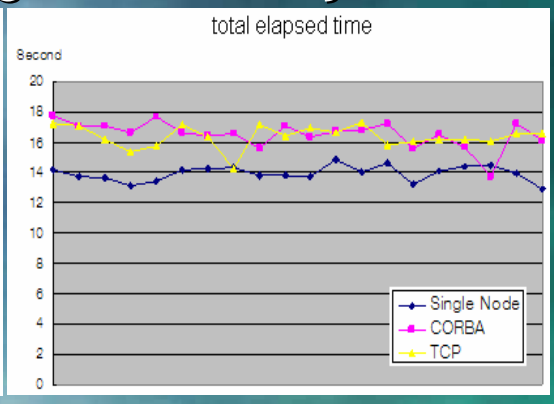
Seoul National University
RTOS Lab

# Performance Evaluation of RSCA with ERSP SDK (3)

- Scenario 2
  - Robot detects the targeted object while rotating, and then goes to the object




| Single Node | TCP/IP | CORBA |
|---|---|---|
| 25 ms | 251 ms | 280 ms |




| Single Node | TCP/IP | CORBA |
|---|---|---|
| 13.9 s | 16.3 s | 16.5 s |



# Conclusions

## Summary

- Systematic robot software development requires to standardize the operating environment
- Automatic deployment is one of the most important feature that should be provided by the operating environment
  - Component model, packaging model, deployment model
  - Interfaces to access/control components comprising the *domain*





## Summary

- RSCA provides standard operating environment consists of
  - Operating system: POSIX PSE52
  - Distribution middleware: minimum CORBA and RT-CORBA
  - Deployment middleware: RSCA core framework

# Summary

- Deployment middleware called RSCA core framework provides
  - A component model
  - Dynamic deployment mechanism
  - Real-time and QoS capabilities
  - Resource management mechanism





# Conclusions

- RSCA address most but not all of topics being sought by the RFI
  - Operating environment
    - Standard real-time operating system
    - Standard distribution middleware
    - Standard deployment middleware
  - Application component interfaces
    - Standard robot component types
    - Standard robot service components
  - Standard functional layering method

not addressed by this response

# Conclusions

- Seeking opportunities for the collaborations with other organizations in OMG
  - SDO DSIG: SDO extensions for Robot Technology Components (RTCs)
    - Interoperable data structures and interfaces
  - SBC DTF: MDA extensions of SCA





# Thank You !!

For more information, please visit our website at

http://rsca.snu.ac.kr

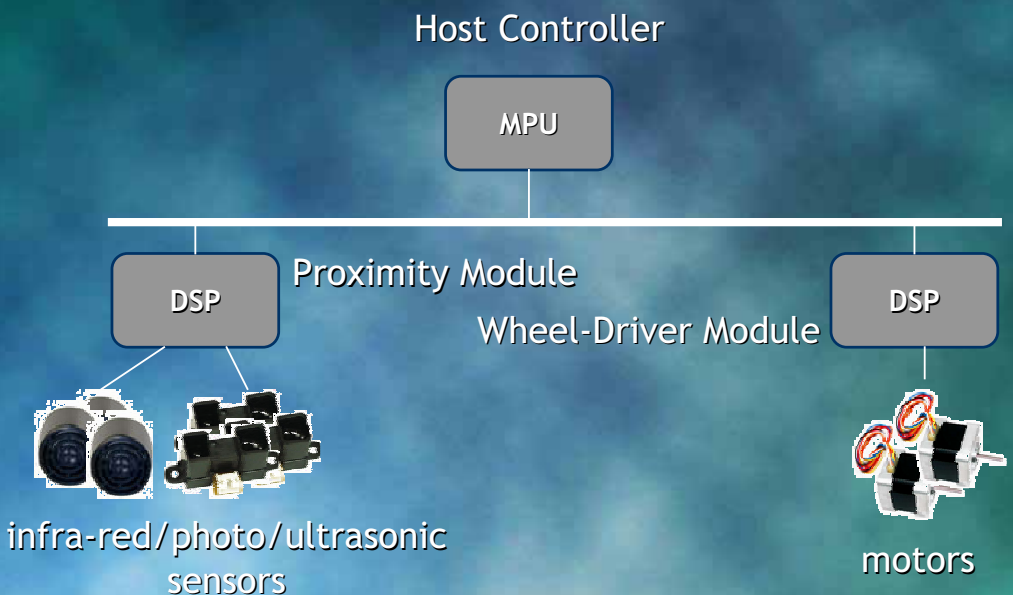
Response from
Real-Time Operating Systems Lab,
Seoul National University
Http://redwood.snu.ac.kr

# Appendix: Working Scenarios

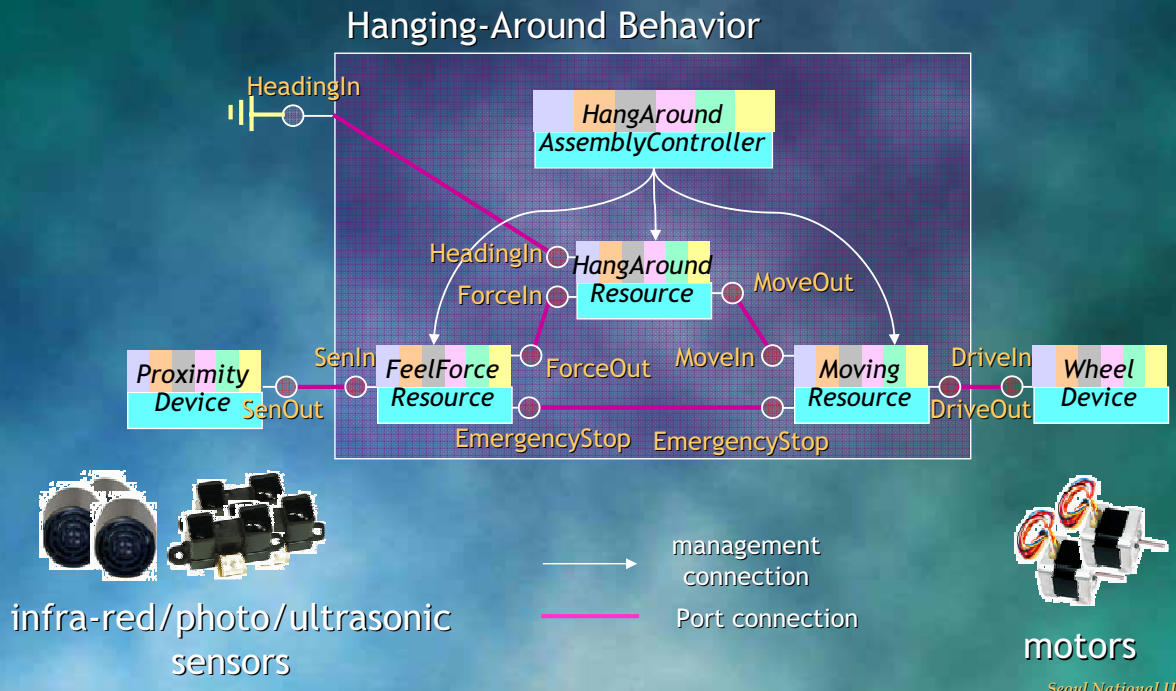




## Hardware Configuration



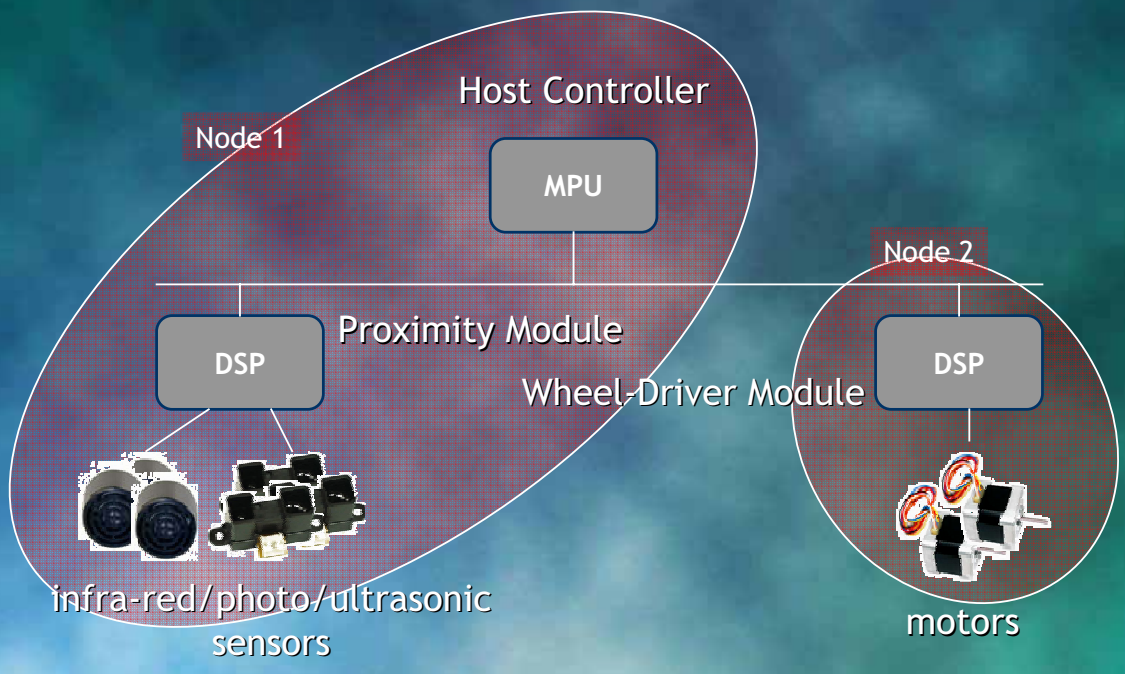
# Example Application : Hanging-Around Behavior






Seoul National University
RTOS Lab

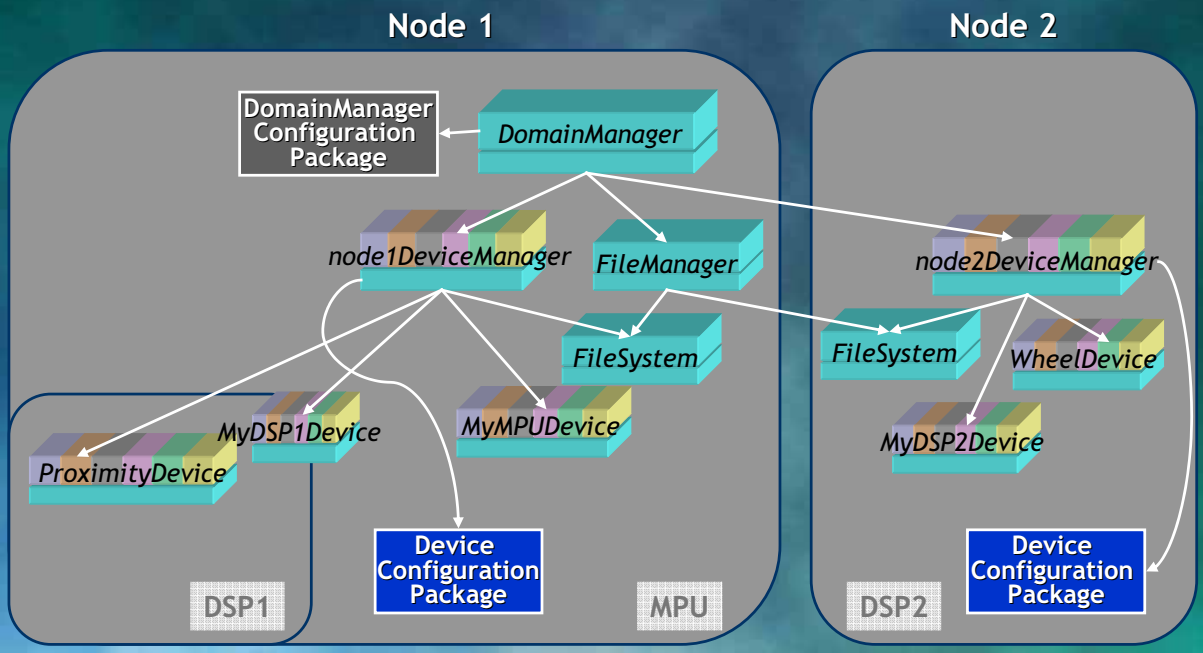
# Boot-Up: Logical Node Configuration






Seoul National University
RTOS Lab

# Boot-Up Process

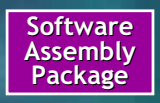






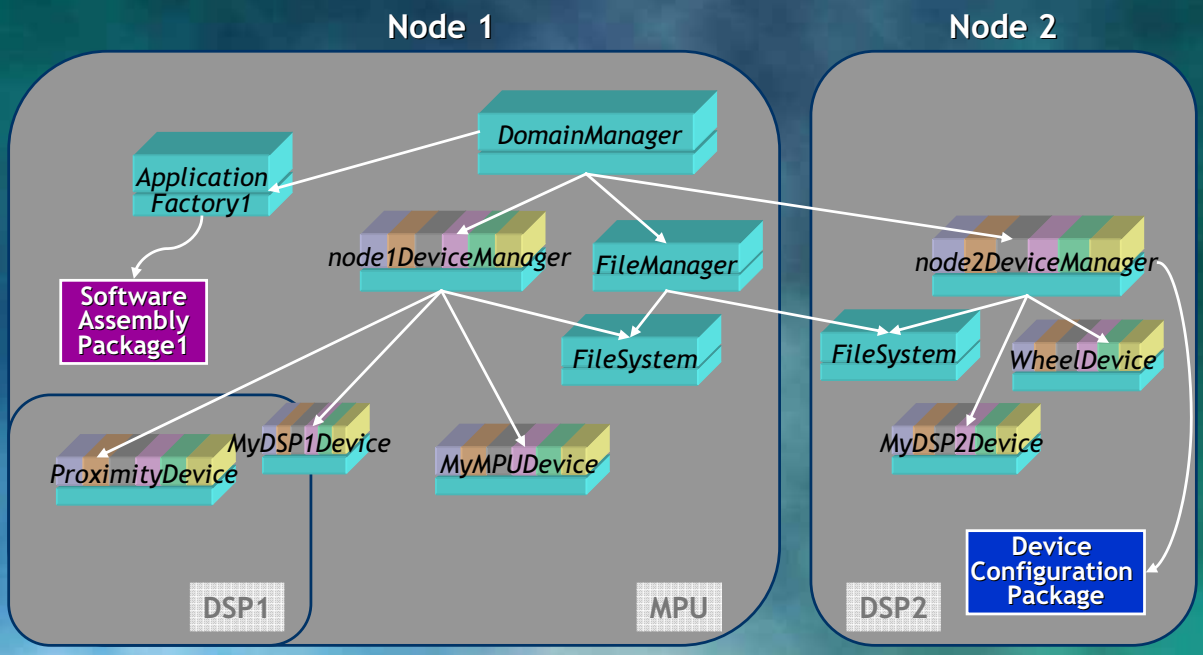

# Application Installation Process (1)

- Download a software assembly package




- `DomainManager` creates an `ApplicationFactory` object using software assembly descriptor contained in the downloaded package

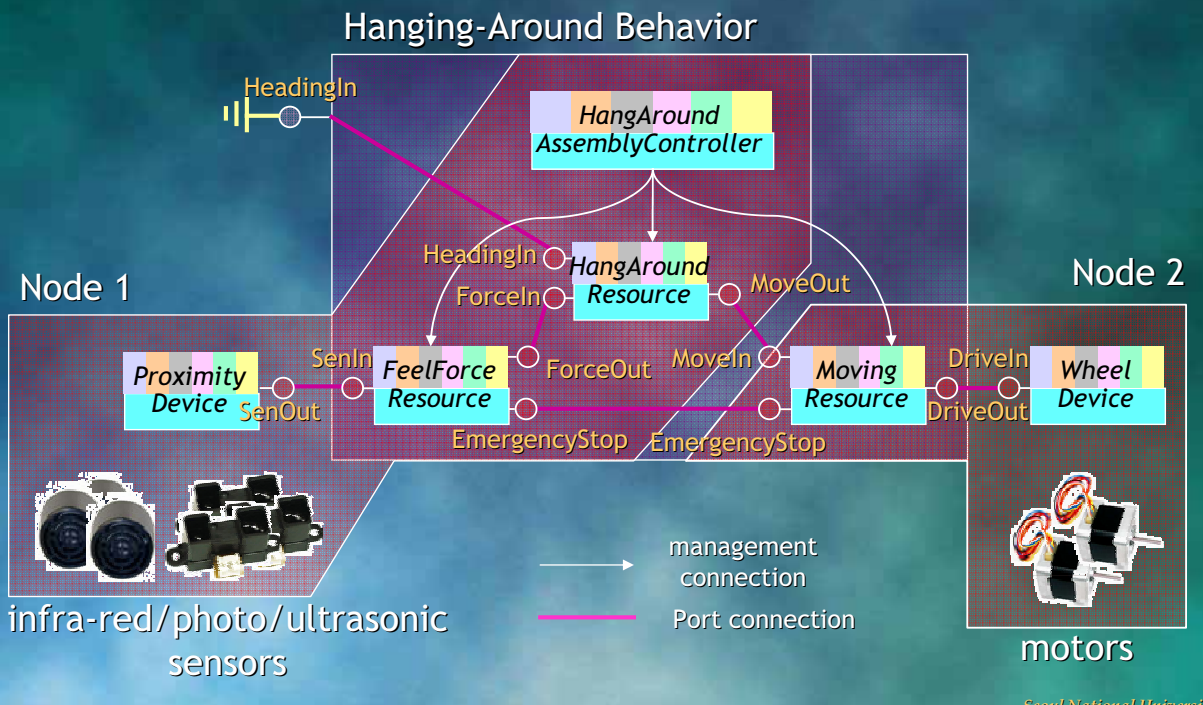
# Application Installation Process (2)





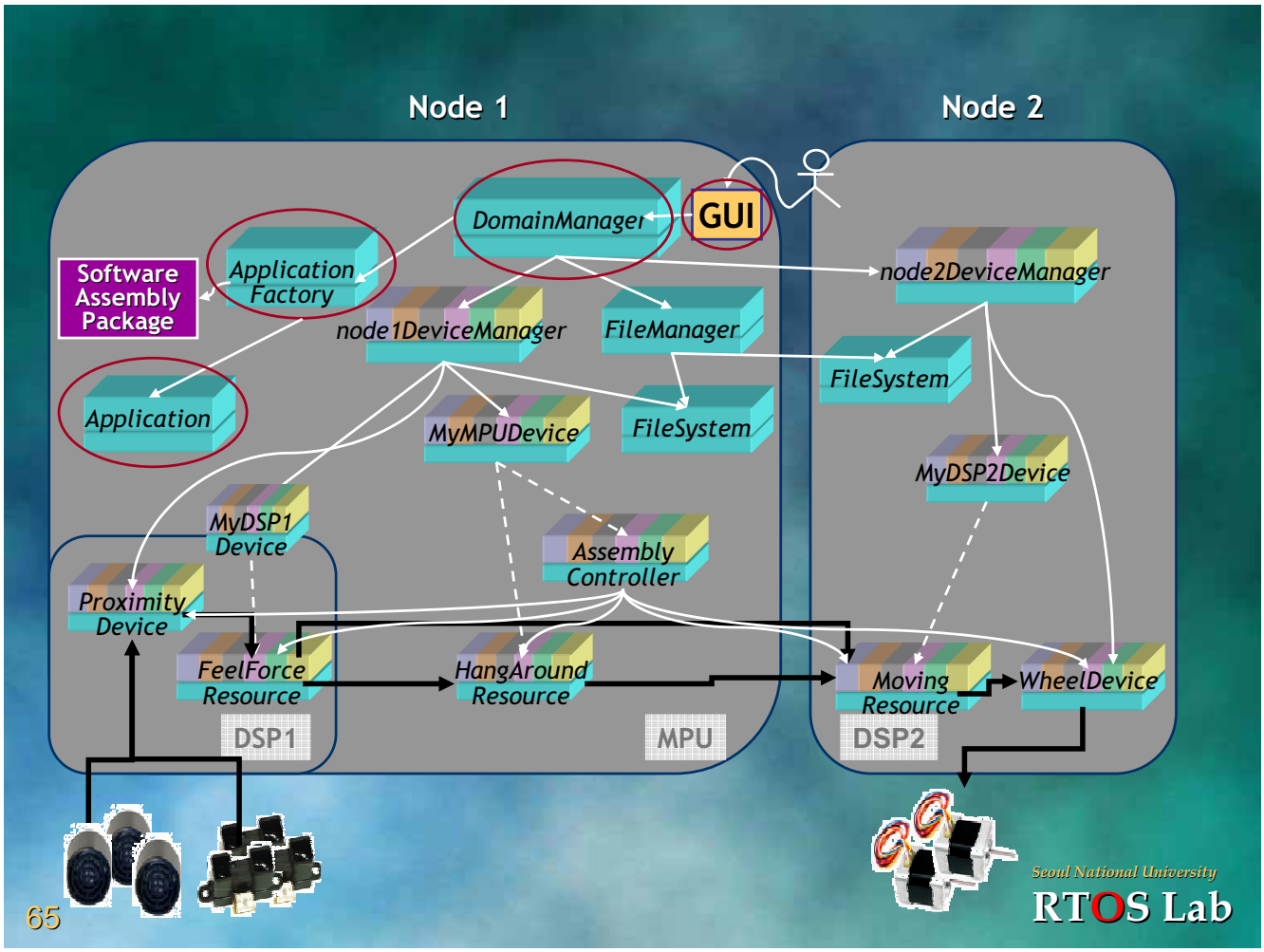

Seoul National University
RTOS Lab

# Application Instantiation Process (1)






Seoul National University
RTOS Lab






Seoul National University
RTOS Lab

# Example Software Assembly Descriptor for Application (1)

- Describes component implementation files to be used

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE softwareassembly SYSTEM "dtd/softwareassembly.2.1.dtd">
-<softwareassembly id="DCE:9601C10A-249F" name="HangingAround">
    -<componentfiles>
        -<componentfile id="HangAroundFactoryFile" type="SPD">
            <localfile name="HangAroundFactory_SPD.xml"/>
         </componentfile>
        -<componentfile id="FeelForceResourceFile" type="SPD">
            <localfile name="FeelForceResource_SPD.xml"/>
         </componentfile>
     </componentfiles>
     . . . . . .
```

Software Package Descriptor

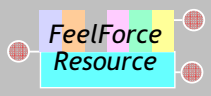


Seoul National University
RTOS Lab

# Example Software Assembly Descriptor for Application (2)

- Describes component instantiation id and usage name and specifies property values

```
-<partitioning>
    -<componentplacement>
        <componentfileref refid="FeelForceResourceFile"/>
        -<componentinstantiation id="DCE:916B1F9F-25BA">
            <usagename>FeelForceResource</usagename>
            -<componentproperties>
                <simpleref refid="Frequency" value="10"/>
                <simpleref refid="Sensitivity" value="0.5"/>
            </componentproperties>
        </componentinstantiation>
    </componentplacement>
    +<componentplacement>
    +<componentplacement>
</partitioning>
. . . . . .
```



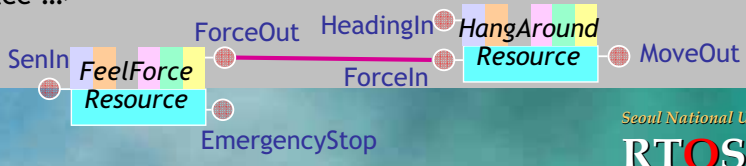





# Example Software Assembly Descriptor for Application (3)

- Describes how components are connected

```
-<connections>
    -<connectinterface id="FeelForceToHangAround">
        -<usesport>
            -<usesidentifier>ForceOut</usesidentifier>
            -<findby> <namingservice name="FeelForceResource"/> </findby>
        </usesport>
        -<providesport>
            -<providesidentifier>ForceIn</providesidentifier>
            -<findby> <namingservice name="HangAroundResource"/> </findby>
        </providesport>
    </connectinterface>
    +<connectinterface id="FeelForceToMovingResource">
    +<connectinterface id="HangAroundToMovingResource">
    +<connectinterface ...>
</connections>
</softwareassembly>
```



# Human Interface of the Robotic System RFI

Soo-Young Chi

Team Leader, Ph.D.
Human Robot Interaction Research Team
Intelligent Robot Research Division
Electronics and Telecommunications Research Institute

## Table Of Contents

# SPECIFICATION OVERVIEW

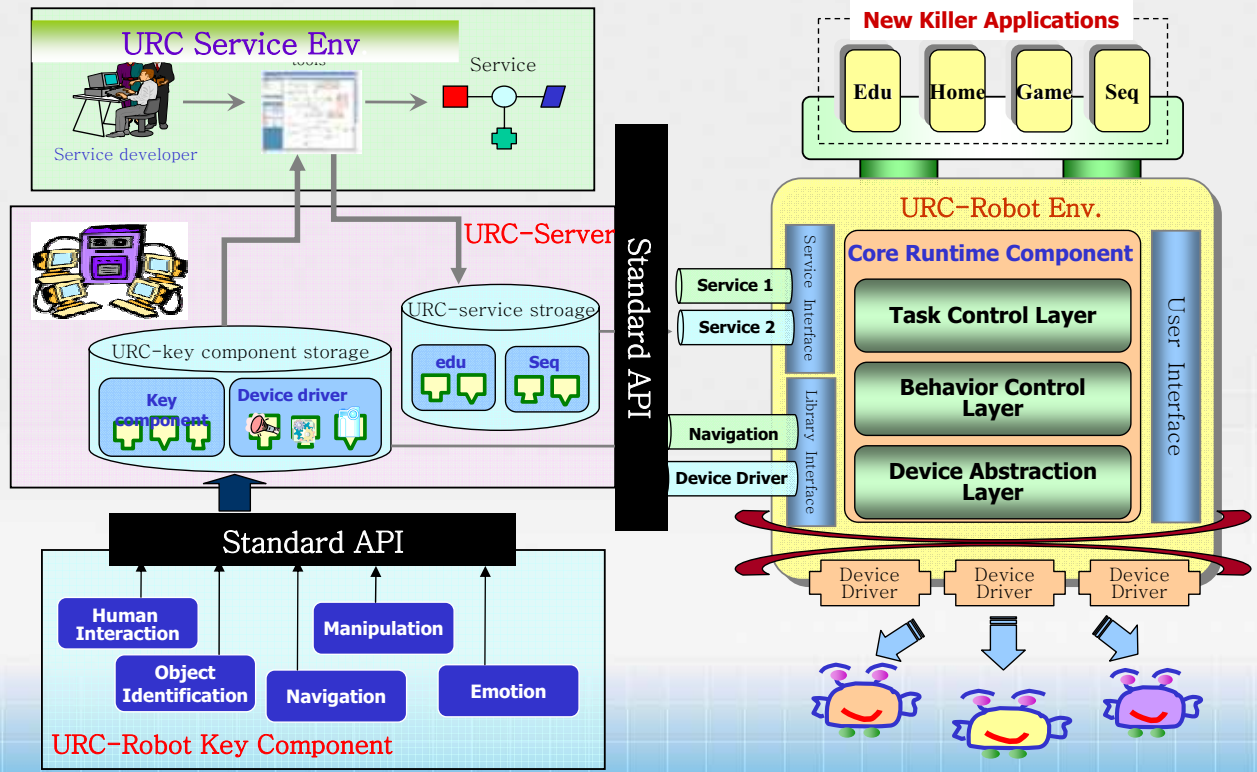
## PURPOSE

This document specifies the **User Recognition Component Standard API for HRI (Human Robot Interaction)** Specification that provides one suited for any form of user recognition for HRI technology used by user recognition system and defines the application interface to cover the basic functions of Enrollment, Verification and Identification.



# SPECIFICATION OVERVIEW

Co-works with CMU, USC, KAIST, Samsung Inc, LG Inc ,..



# SPECIFICATION OVERVIEW

## SCOPE

User Recognition Component Standard API for HRI is the standard for application program interface. It includes the standard interface of basic functions - Enrollment, Verification and Identification - and the interfaces of user-friendly programs employing User Recognition programs for HRI.



# User Recognition Component API Model for HRI

There are three principal high-level **abstraction** functions in the API:

**1) Enroll**

Samples are **capture**d from a device, **process**ed into a usable form from which a **template** is constructed, and returned to the application.

**2) Verify**

One or more samples are **capture**d, **process**ed into a usable form, and then **match**ed against an input **template**. The results of the comparison are returned.

**3) Identify**

One or more samples are **capture**d, **process**ed into a usable form, and **match**ed against a **set of templates**. A list is returned showing how close the samples compare against the top candidates in the set.

# User Recognition Component API Model for HRI

There are three **primitive** functions in the API which, when used in sequence on client and server, can accomplish the same result as the high-level abstractions:

**Capture**

Capture is always executed on the client machine; attempting to execute Capture on a machine without a device will return “function not supported”. One or more samples are acquired (either for Enrollment, Verification or Identification). The Capture function is allowed to perform as much



# User Recognition Component API Model for HRI

**Process**

The “processing algorithms” must be available on the server, but **may** also be available on the client.

The Process function is intended to provide the processing of samples necessary for the purpose of verification or identification (not enrollment). It always takes an “intermediate” as input, and may complete the processing of the data into “final” form suitable for its intended purpose.

On the client, if it completes the processing, it returns a “processed” otherwise it returns an “intermediate”

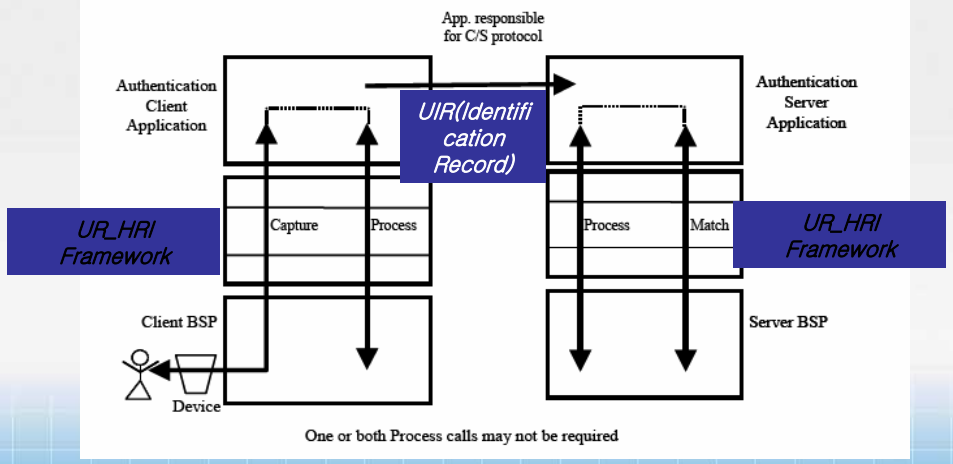
## Match

Performs the actual comparison between the “processed” UIR and one template (**VerifyMatch**), or between the “processed” UIR and a set of templates (**IdentifyMatch**). The support for IdentifyMatch is optional, but the supported Match functions are always available on the server, and **may** be available on the client.



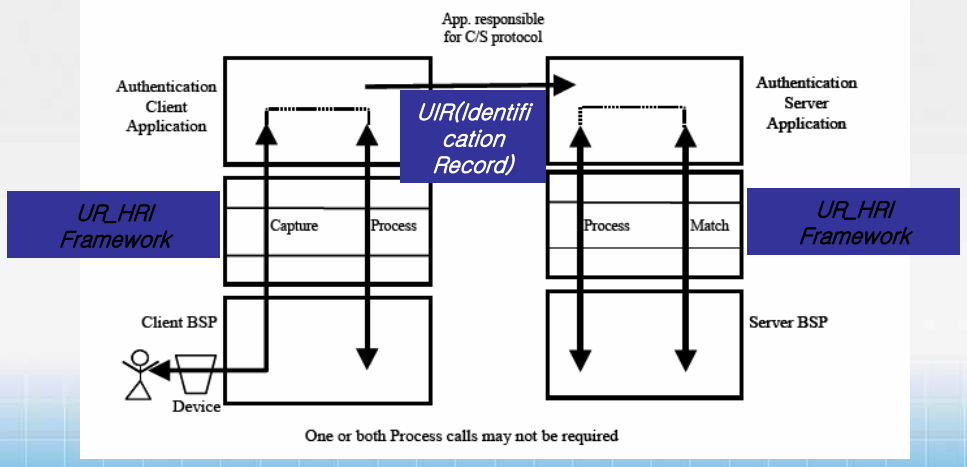


However, as Figure 1 shows, the processing of the UIR data from the capture of raw samples to the matching against a template may be accomplished in many stages, with much CPU-intensive processing.



# User Recognition Component API Model for HRI

The API has been defined to allow the HRI developer the maximum freedom in the placement of the processing involved, and allows the processing to be shared between the client machine (which has the device attached), and a server machine. It also allows for self-contained devices, in which all the HRI processing can be done internally.







# User Recognition Component API Model for HRI


There are several good reasons why processing and matching may take place on a server:

1. The algorithms will execute in a more secure environment
2. The Client PC may not have sufficient power to run the algorithms well.
3. The user database (and the resources that it is protecting) may be on a server.
4. Identification over large populations can only reasonably be done on a server.

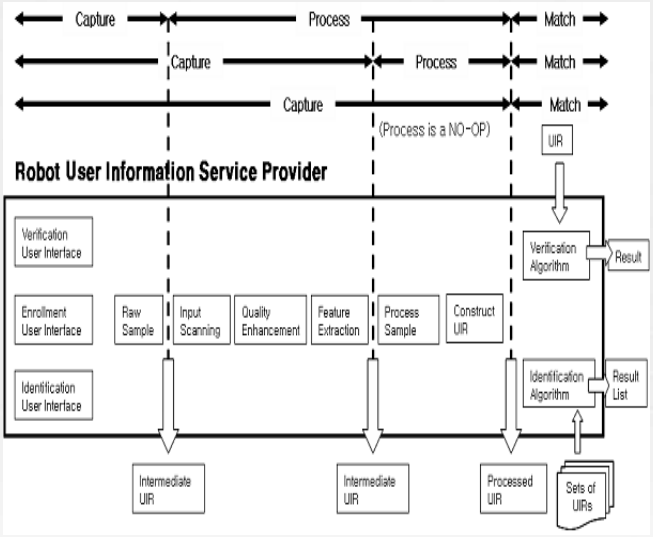
# User Recognition Technology of HRI

•The basic model is the same for all types of User Recognition technology to HRI. First, the initial registration “template” of the user has to be constructed. This is done by collecting a number of samples through whatever sensor device is being used. Salient features are extracted from the samples, and the results combined into the template. The construction of this initial template is called Enrollment. The algorithms used to construct a template are usually proprietary. This initial template is then stored by the application, and essentially takes the place of a password.



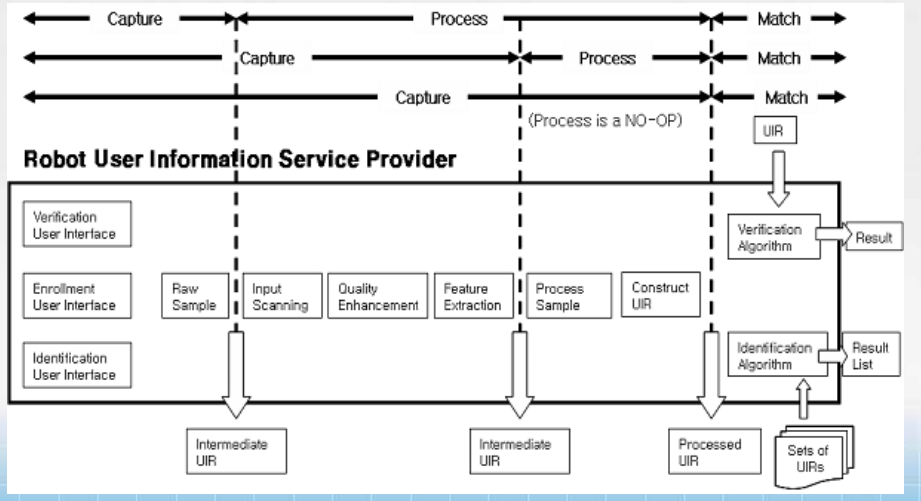
# User Recognition Technology of HRI

Figure 2 shows some possible implementation strategies. The various steps in the verification and Identification operations are shown in the box labeled **RUI Service Provider.** The stages identified above the box refer to the primitive functions of the top-level interface: **Capture**, **Process** and **Match**, and it shows that a RUI **Service Provider** has degrees of freedom in the placement of function in these primitives.



# User Recognition Technology of HRI

There is a degree of freedom not shown in the figure; the manufacturer is free to put most, if not all the RUI **Service Provider** function in the sensing device itself. In fact, if the device contains a UIR database, all functions may be performed in the device.






# User Recognition Technology of HRI

## UIRs and Templates

This standard uses the term **template** to refer to the biometric enrollment data for a user. The template must be matched (within a specified tolerance) by **sample**(s) taken from the user, in order for the user to be **authenticated**.

The term **User identification record** (UIR) refers to any UIR data that is returned to the application; including raw data, intermediate data, processed sample(s) ready for verification or identification, as well as enrollment data. Typically, the only data stored persistently by the application is the UIR generated for enrollment (i.e., the template). The structure of a UIR is shown in Figure 3 below.





UIRs and Templates

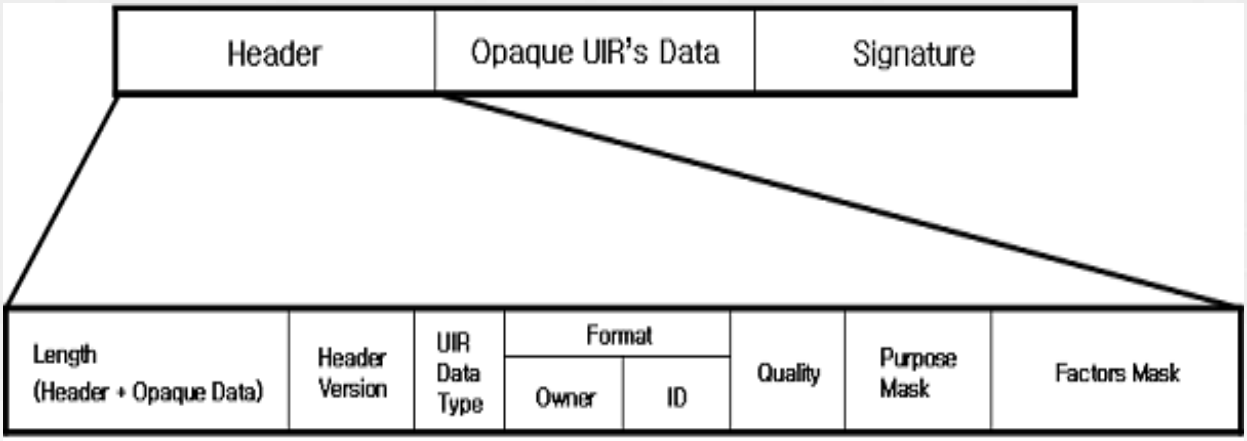



Figure 3. Structure of User Identification Record (UIR)

# User Recognition Technology of H

The format of the Opaque UIR's Data is indicated by the Format field of the Header. This may be a standard or proprietary format. The Opaque data may be encrypted.

The URC API UIR definition is compliant with the "Common Biometric Exchange File Format (CBEFF)", of which it is one of the CBEFF Patron Formats. CBEFF is described in the National Institute of Standards Publication, NISTIR 6529, January 3, 2001, developed by the CBEFF Technical Development Team



# HRI API Definition

**HriAPI Data Structures**

The opaque UIR data is of variable length, and may be followed by a signature. The signature itself may not be a fixed length, depending on which signature standard is employed. The signature is calculated on the combined Header and UIR Data.

```
typedef struct ruic_uir
{
    RUIC_UIR_HEADER     Header;      // Header of UIR
    RUIC_UIR_DATA_PTR   pUirData;    // Opaque UIR data
    RUIC_UIR_DATA_PTR   pSignature;  // Signature
} RUIC_UIR;
```

# HRI API Definition

**HriAPI Data Structures**

**Factor type**
A mask that describes the set of authentication factors supported by an authentication service.

```
typedef uint32 RUIC_UIR_FACTORS;
#define RUIC_FACTOR_MULTIPLE                (0x00000001)
#define RUIC_FACTOR_FACIAL_FEATURES         (0x00000002)
#define RUIC_FACTOR_SPEECH                  (0x00000004)
#define RUIC_FACTOR_HEIGHT                  (0x00000008)
#define RUIC_FACTOR_CLOTHES                 (0x00000010)
#define RUIC_FACTOR_LIP_MOVEMENT            (0x00000020)
#define RUIC_FACTOR_GAIT                    (0x00000040)
#define RUIC_FACTOR_PASSWORD                (0x80000000)
```



# HRI API Definition

**HriAPI Data Structures**

**RUIC_UIR_DATA_FORMAT**

```
typedef struct ruic_uir_data_format {
    uint16 FormatOwner;
    uint16 FormatID;
} RUIC_UIR_DATA_FORMAT,
*RUIC_UIR_DATA_FORMAT_PTR;
```

# HRI API Definition

## HriAPI Data Structures

### RUIC_UIR_DATA_TYPE

```
typedef uint8 RUIC_UIR_DATA_TYPE;
#define RUIC_UIR_DATA_TYPE_RAW                  (0x01)
#define RUIC_UIR_DATA_TYPE_INTERMEDIATE         (0x02)
#define RUIC_UIR_DATA_TYPE_PROCESSED            (0x04)
#define RUIC_UIR_DATA_TYPE_ENCRYPTED            (0x10)
#define RUIC_UIR_DATA_TYPE_SIGNED               (0x20)
```



# HRI API Definition

## HriAPI Data Structures

### RUIC_UIR_HEADER

```
typedef struct ruic_uir_header {
    uint32 Length; /* Length of Header + Opaque Data */
    RUIC_UIR_VERSION HeaderVersion;
    RUIC_UIR_DATA_TYPE Type;
    RUIC_UIR_DATA_FORMAT Format;
    RUIC_UIR_FACTORS FactorsMask;
} RUIC_UIR_HEADER, *RUIC_UIR_HEADER_PTR;
```

# HRI API Definition

## HriAPI Data Structures

### Capture

RUIC_RETURN RUIC_Capture(int nCaptureNum, RUIC_UIR* pCapturedUIR, int  nTImeout);

### Enrollment

RUIC_RETURN RUIC_StartEnrollment(RUIC_UIR* pCapturedUIR, RUIC_UIR_HANDLE* pNewTemplate);



# HRI API Definition

## HriAPI Data Structures

### Verification

RUIC_RETURN RUIC_Verification(RUIC_UIR_HANDLE Template, RUIC_UIR Sample, double *pScore);

### Matching

RUIC_RETURN RUIC_Matching(RUIC_UIR_HANDLE Template, RUIC_UIR Feature, double *pScore);

# HRI API Definition

## HriAPI Data Structures

Error-Handling

```
#define RUIC_BASE_ERROR
#define RUIC_ERR_UNABLE_TO_CAPTURE
#define RUIC_ERR_TOO_MANY_HANDLES
#define RUIC_ERR_INVALID_UIR_HANDLE
#define RUIC_ERR_UIR_SIGNATURE_FAILURE
 :
 :
```



# Q/A

robotics2005-12-10

# High Assurance Security and Safety for Robotics

*High Assurance Components*
*for*
*High Assurance Systems*

OBJECTIVE INTERFACE



# High Assurance Security and Safety for Robotics

## Presentation to Robotics DSIG
## 7 December 2005

Joseph M. Jacob, Senior Vice President
Objective Interface Systems, Inc.

## Acknowledgements

Acknowledgement of significant contributions from:

- Michael Dransfield, National Security Agency
- Jahn Luke, Air Force Research Laboratory
- Mark Vanfleet, National Security Agency
- Dr. Ben Calloni, P.E., Lockheed Martin
- Michael McEvilley, MITRE



## Why Security and Safety for Robotics?

- **The nature of the next generation of robotics involves independent robots acting within an environment, not under human control**
  - These robots will interact with other independent robots acting within an environment, not under human control
  - These robots will modify their behavior and take actions on their own without any human control, based on
    - Stimuli from the environment
    - Communications from other robots
- **Examples include**
  - Unmanned aerial vehicles
  - Battlefield robots
  - Commercial Robots
  - Home Robots

## Requirements for Security and Safety in Robotics



To be viable, a High-Assurance security and safety architecture for Robotics must

- ensure that developers do not need to rewrite their applications
- ensure that developers can continue to use legacy operating systems and legacy middleware
- provide an easy migration path
- provide the highest level of security and safety that can be achieved

OBJECTIVE INTERFACE

## The Vision

High Assurance Security and Safety for Robotics

**Fuse the best from Safety and Security technologies**

- **Safety**
  - RTCA DO-178B Level A
    - Highest level for flight safety-critical. Failure of the system is catastrophic, resulting in loss of control of aircraft and fatal injuries to large numbers of individuals
  - ARINC-653
- **Security**
  - Common Criteria – Evaluation Assurance Level 6+/7, the highest level of security assurance
  - High Robustness
  - DCID 6/3 Separation

**to enable provision of high assurance security and safety for mission critical robotic systems**

## Industry Standards

- **RTCA DO-178B, *Software Considerations in Airborne Systems and Equipment Certification***
  - Adopted by FAA
  - Encompasses the entire project
- **ARINC-653, *Avionics Application Software Standard Interface***
  - Time and space partitioning to prevent cascading failure of applications
- **ISO-15408, *Common Criteria for Information Technology Security Evaluation***
  - International standard for assurance in IT
- **DCID 6/3, *Protecting Sensitive Compartmented Information Within Information Systems***
  - Procedures for storing, processing and communication of classified intelligence
  - U.S. Federal directive

High Assurance Security and Safety for Robotics

## What is High Assurance?

- **To the FAA:**
  - One failure per $10^9$ (1 Billion) hours of operation
    - How long *is* a Billion hours? Do the math!
      - $1{,}000{,}000{,}000\,\text{hours} \times \frac{1\,\text{day}}{24\,\text{hours}} \times \frac{1\,\text{year}}{365.25\,\text{days}}$
      - 114,077 *YEARS!*
- **For National Security Systems processing our most valuable data under severe threat:**
  - Failure is *Unthinkable*
- *How do we implement systems that we can trust to be this reliable?*

- **Safety-Critical or High-Assurance systems today require software that must meet stringent criteria**
  - Reliability
  - Safety
  - Security
- **Traditionally these systems have been custom designed**
  - Expansion of this type of system => stove-pipe designs have become impractical
  - Looking to COTS



- **Most commercial computer security architectures**
  - The result of systems software where security was an afterthought
    - Operating systems
    - Communications architectures
  - **Reactive** response to problems
    - Viruses, Worms, and Trojan Horses
    - Hackers and Attackers
    - Problems are only addressed ***after*** the damage has been done
- **Inappropriate approach for mission critical systems**
  - Does not safeguard information or the warfighter
  - **Proactive** measures are required to ***prevent*** damage

## Foundational Threats

- **Software can only be as secure as its foundation**
- **If the foundation can be successfully attacked, then any system security function that runs on that foundation can easily be rendered ineffective**
- **Foundational threats include:**
  - Bypass
  - Compromise
  - Tamper
  - Cascade
  - Covert Channel
  - Virus
  - Subversion

High Assurance Security and Safety for Robotics

## Where We've Been: Monolithic Security Kernels

- **All security policy enforcement was performed by the security kernel**
  - For performance reasons
  - No other way to insure enforcement was nonbypassable
- **As security policy became more complex:**
  - Code grew in security kernel
  - Certification efforts become unmanageable
  - Evaluatability of kernel decreased
  - Maintainability of kernel code decreased
  - Policy decisions were based upon incomplete/unauthenticated information

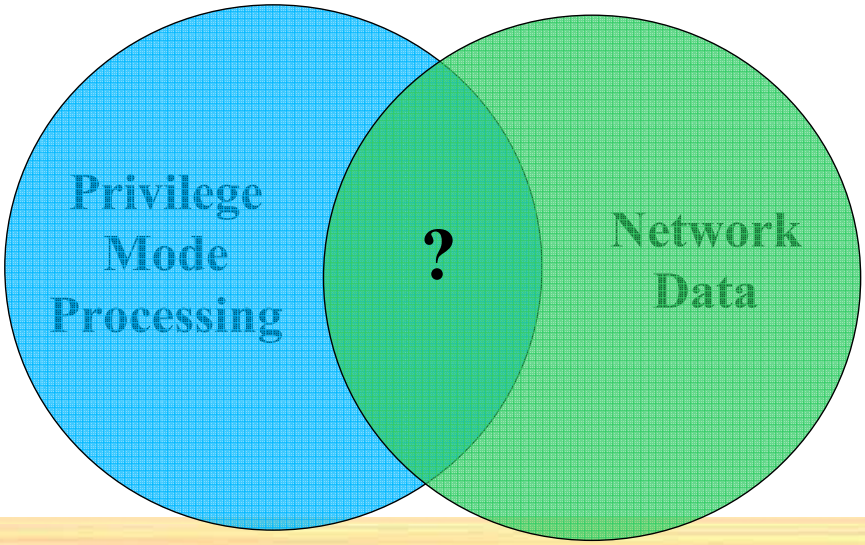
## A New Approach

- **Multiple Independent Levels of Security/Safety:** *MILS*
- **Each layer/application can be evaluated separately without impact to the evaluation of the other layers/applications**
- **High assurance applications**
  - Can be developed
  - Can be evaluated
  - Can be maintained
- **High assurance applications can become a full partner in enforcing complex Security Policies**
- *Goal: MLS/MSLS capabilities become more practical, achievable and affordable.*

High Assurance Security and Safety for Robotics

## MILS Overview

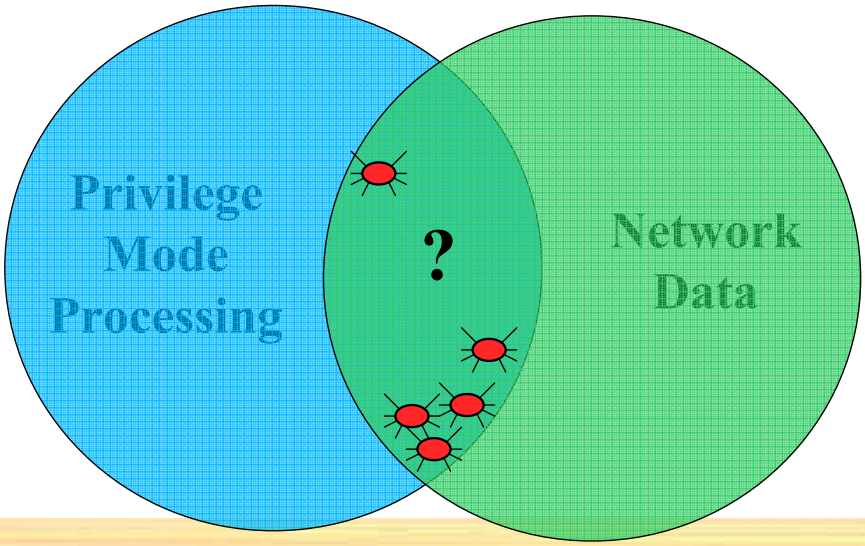
- MILS: Multiple Independent Levels of Security
- Security Kernel is the only privileged code
- Security Kernel enforces only four very simple security policies
- All other security policy enforcement is divided among middleware and the applications
- Enables application layer to enforce its own security policies in a manner that is “N.E.A.T.”
  - More about what that means later




What happens when network data is processed in privilege mode?

MILS Security Policy

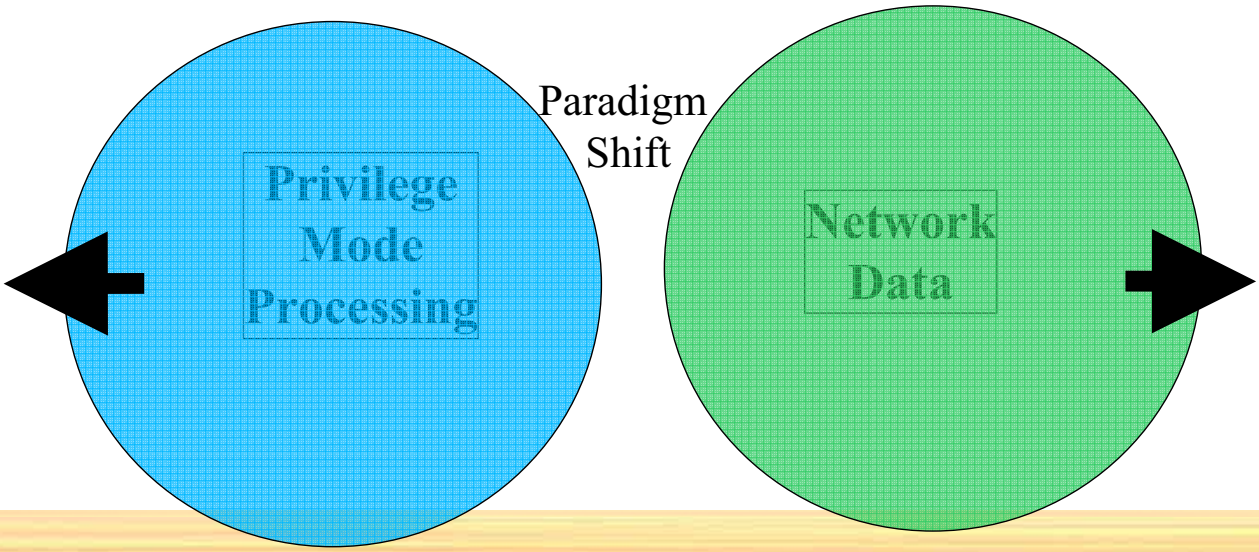
High Assurance Security and Safety for Robotics




**Wild Creatures of the Net: Worms, Virus, . . .**

## MILS Security Policy



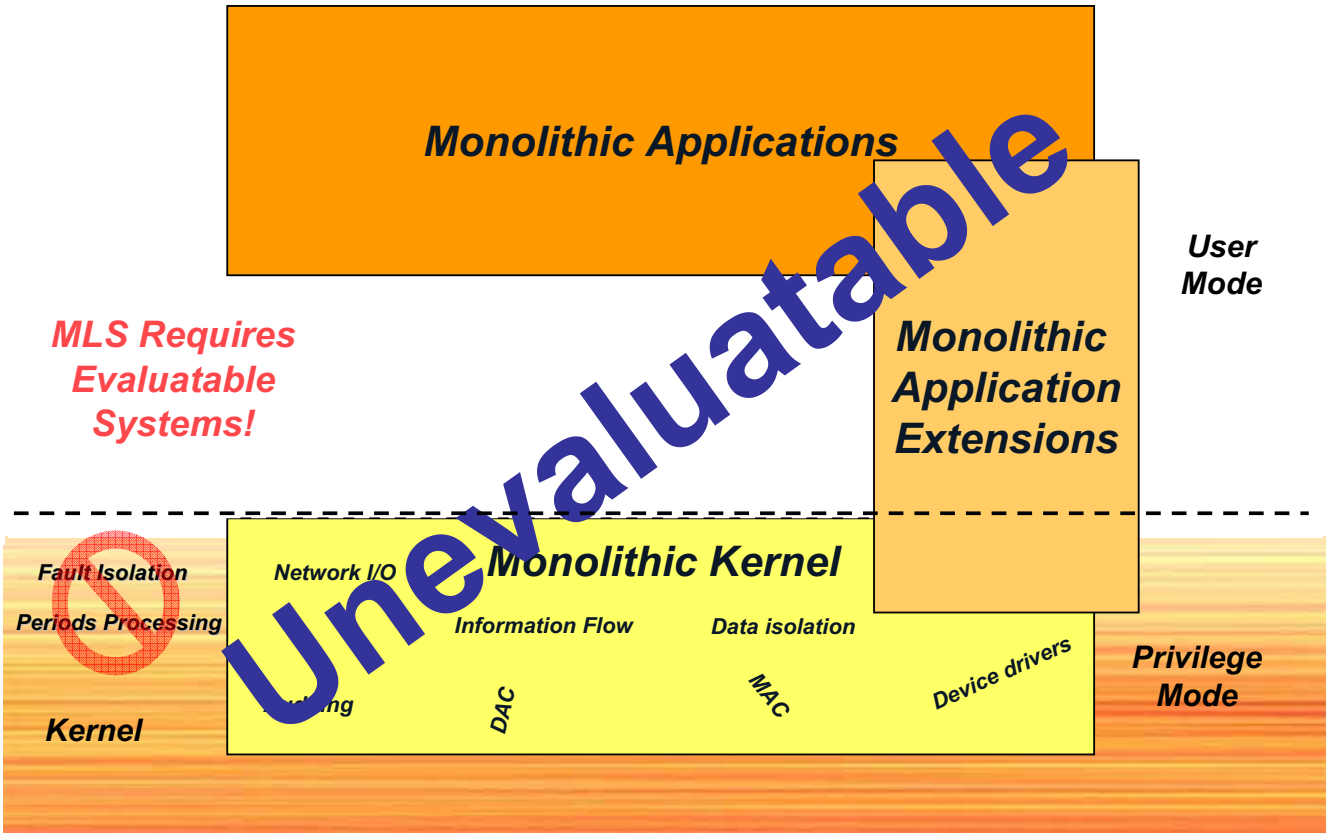



**Under MILS Network Data and Privilege Mode Processing are Separated**



## Where We've Been: Starting Point for Architectural Evolution





- **60,000 LOC running in privileged mode is not evaluatable.**

  **Re-arrange the code** so there is less security critical code. This enables practical application of formal methods so we can do a better job of proving that the code can be trusted. Large monolithic architectures have a very slim chance of being properly evaluated.

- We want to be able to **mathematically** verify the code does what is intended **and nothing else**.

- Outcome is **secure COTS products**

The Whole Point of MILS

High Assurance Security and Safety for Robotics

## Really very simple:

- Dramatically reduce the amount of *safety/security critical code*

So that we can

- Dramatically increase the scrutiny of *safety/security critical code*

## The MILS Architecture

**Three distinct layers (John Rushby, PhD)**

- Separation Kernel
  - Separate process spaces (partitions)
  - Secure control of information flow between partitions
  - Really small: 4K lines of code
- Middleware
  - Device Drivers, File Systems, Network Stacks
  - CORBA, DDS, Web Services, etc.
  - Partitioning Communications System
- Applications
  - Firewalls, Downgraders, Guards, CDS, Multi-Nation Web Server, etc.
  - Application-specific security functions/reference monitors

High Assurance Security and Safety for Robotics

## The MILS Architecture (cont'd)

**Separation Kernel**

- Microprocessor Based
  - Time and Space Multi-Threaded Partitioning
  - Data Isolation
  - Inter-partition Communication
  - Periods Processing
  - Minimum Interrupt Servicing
  - Semaphores
    - Synchronization Primitive's
  - Timers

*And nothing else!*

**MILS Middleware**

- **Traditional RTOS Services**
  - Device Drivers
  - File Systems
  - Token and Trusted Path
- **Traditional Middleware**
  - CORBA (Distributed Objects)
  - Data Distribution (Pub-Sub)
  - Web Services
- **Partitioning Communication System** (PCS)
  - Global Enclave Partition Comm
    - TCP, UDP, Rapid-IO, Firewire, etc.
    - Partition Based Attestation

## Separation Kernel



- **Where should Separation Kernel (RTOS) reside?**
  - To be tamper-proof
    - Must be in a separate address space from ***any*** application code
  - To be non-bypassable
    - Must be part of every input or output service request issued by an application

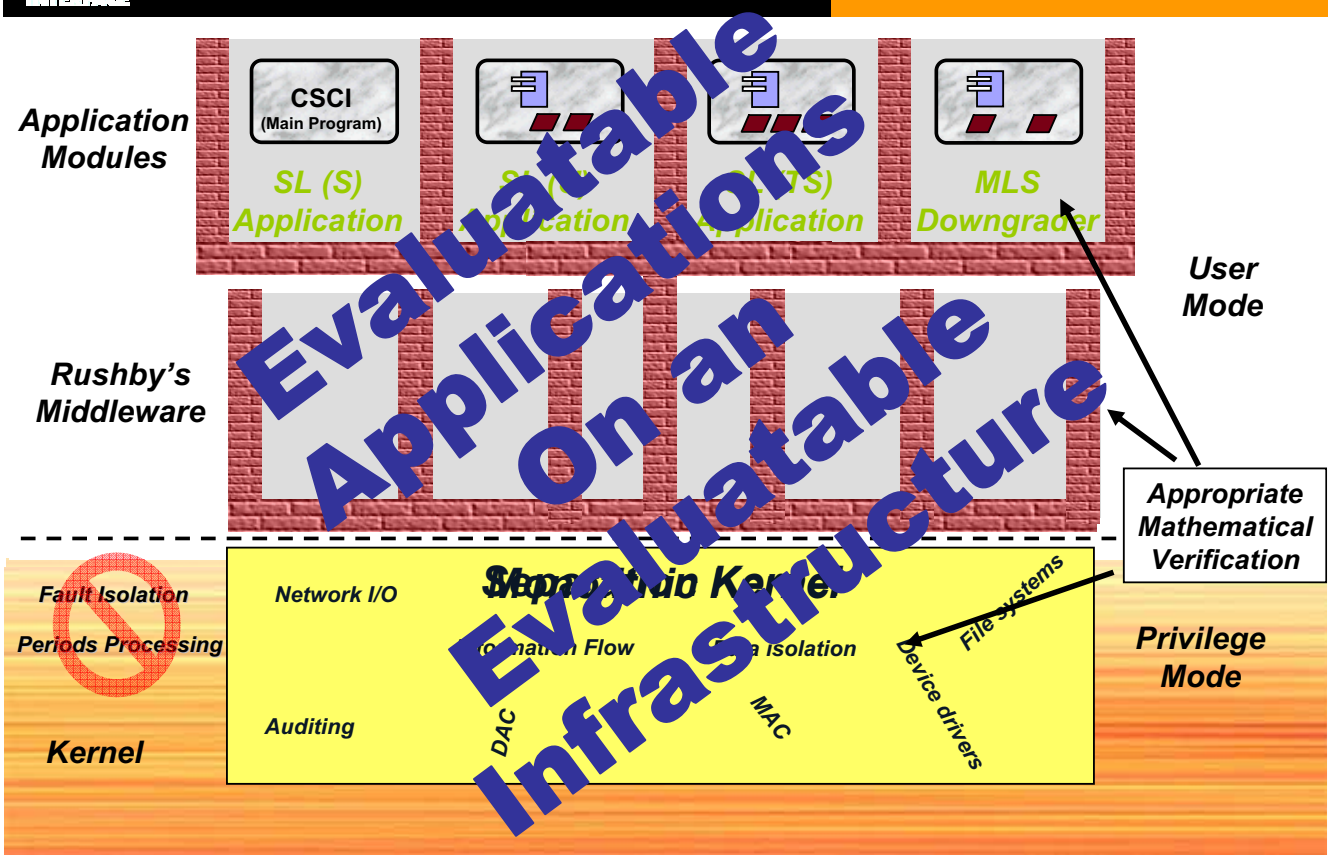
- **Why keep security functions out of the kernel?**
  - Security functions are often application-specific
  - Any code co-resident with security functions could interfere with those security functions
  - Entire kernel must be analyzed for weaknesses and malicious code

- ***The Separation Kernel must be the only code that runs in privileged mode***



## MILS Architecture Evolution





## MILS Architecture Objectives

What does MILS do?

Enables **Application Layer Entities** to

**Enforce, Manage, and Control**

their own

**Application Level Security Policies**

such that enforcement of the Application Level Security Policies is

***Non-bypassable*** **Reference**
***Evaluatable*** **Monitor**
***Always-Invoked*** **Concept**
***Tamper-proof***

**The MILS architecture allows the Security Kernel to *SHARE* the responsibility of Security with the Application.**

High Assurance Security and Safety for Robotics

## MILS Architecture Objectives

How does MILS achieve its goals?

It Enforces an

**Information Flow,**
**Data Isolation,**
**Periods Processing, and**
**Damage Limitation**

**Security Policy** between multiple address spaces:

First, in a **Microprocessor Centric Manner**, i.e., MILS RTOS,
Second, in a **Network Centric Manner**, i.e., MILS Middleware,

in such a manner that the ***layered*** Security Policies are also

***Non-bypassable*** **Layered**
***Evaluatable*** **Reference**
***Always-Invoked*** **Monitor**
***Tamper-proof*** **Concept**

## What Does NEAT Really Mean?

### Separation Kernel & Trusted Middleware must be:

**N E A T**

- ***N***on-bypassable
  - Security functions cannot be circumvented
- ***E***valuatable
  - Security functions are small enough and simple enough for mathematical verification
- ***A***lways Invoked
  - Security functions are invoked each and every time
- ***T***amperproof
  - Subversive code cannot alter the security data or functions

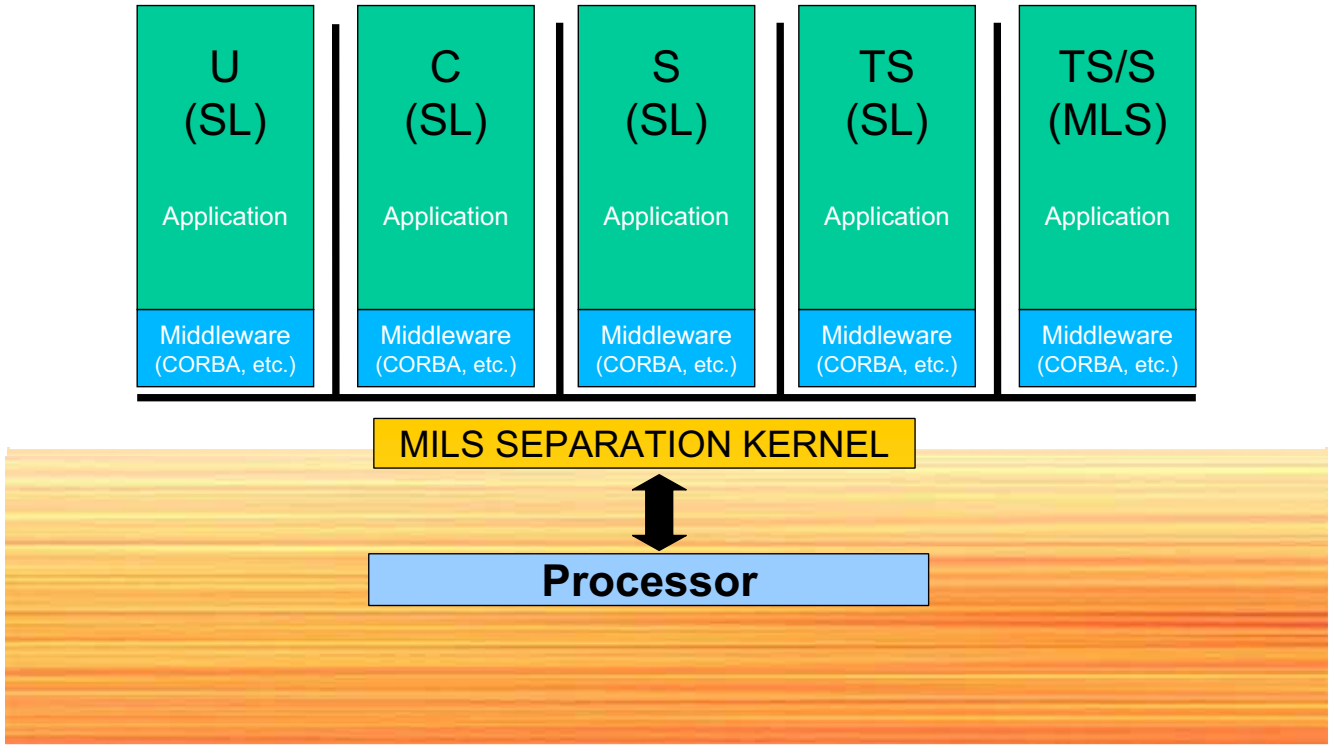
High Assurance Security and Safety for Robotics

## MILS Security Policies

- **Information Flow**
  - Information originates only from authorized sources
  - Information is delivered only to intended recipients
  - Source of Information is authenticated to recipient
- **Data Isolation**
  - Information in a partition is accessible only by that partition
  - Private data remains private
- **Periods Processing**
  - The microprocessor itself will not leak information from one partition to another as it switches from partition to partition
- **Damage Limitation**
  - A failure in one partition will not cascade to another partition
  - Failures will be detected, contained, & recovered from locally

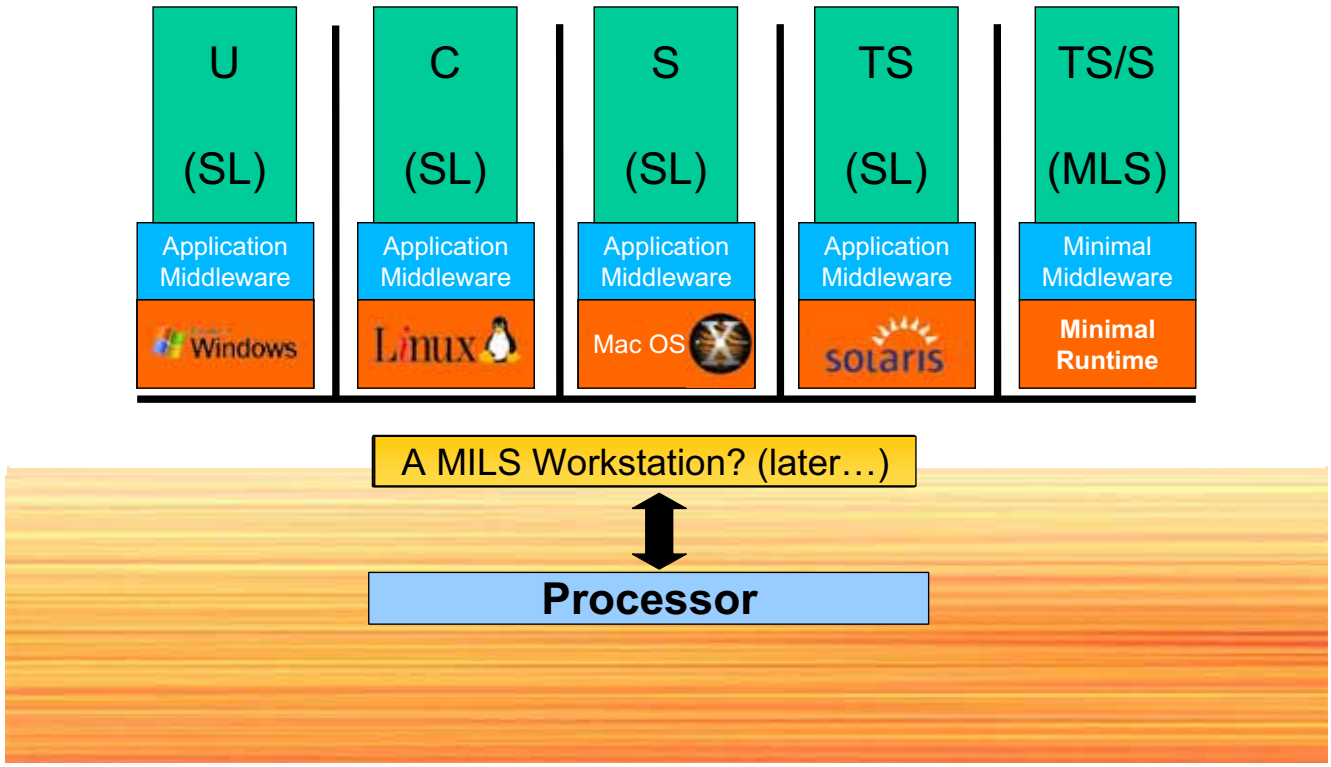
## The MILS Architecture

*High Assurance Security and Safety for Robotics*




## Guest OS Architecture

*High Assurance Security and Safety for Robotics*



## Distributed Security Requirements

- **Extend single node enforcement to multiple nodes**
- **Do not add new threats to data Confidentiality or Integrity**
- **Enable distributed Reference Monitors to be** NEAT
- **Optimal inter-node communication**
  - Minimizing added latency (first byte)
  - Minimizing bandwidth reduction (per byte)
- **Fault tolerance**
  - Infrastructure must have no single point of failure
  - Infrastructure must support fault tolerant applications

High Assurance Security and Safety for Robotics

## PCS: The Partitioning Communications System

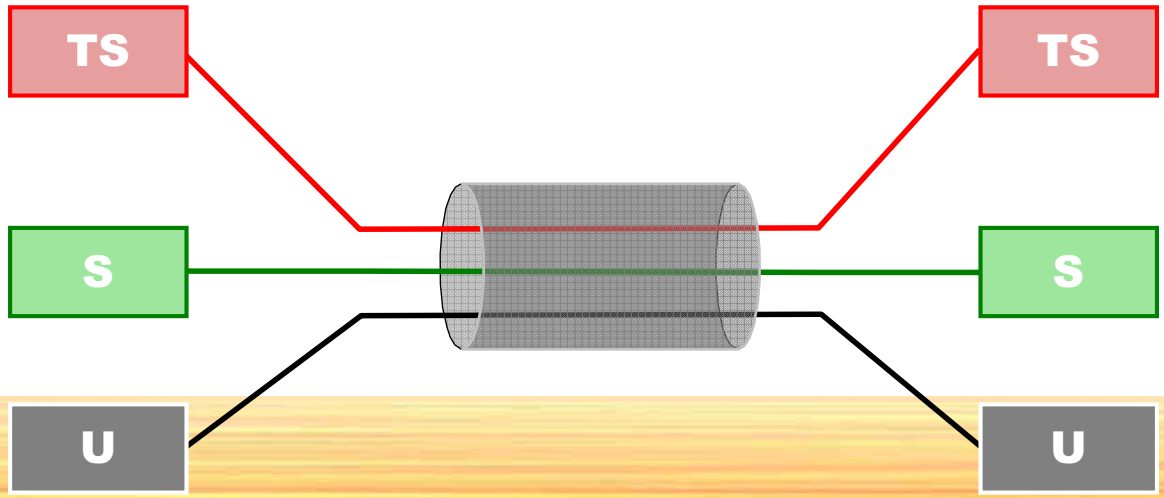
- **Partitioning Communications System (PCS) is communications middleware for MILS**
- **Always interposed in inter-node communications**
- **Interposed in some intra-node communications also**
- **Parallels Separation Kernel's policies**

## PCS Specific Requirements

- **Strong Identity**
  - Nodes within enclave
- **Separation of Levels/Communities of Interest**
  - Need cryptographic separation
- **Secure Configuration of all Nodes in Enclave**
  - Federated information
  - Distributed (compared) vs. Centralized (signed)
- **Secure Loading: signed partition images**
- **Secure Clock Synchronization**
- **Suppression of Covert Channels**
  - Bandwidth provisioning & partitioning
  - Network resources: bandwidth, hardware resources, buffers

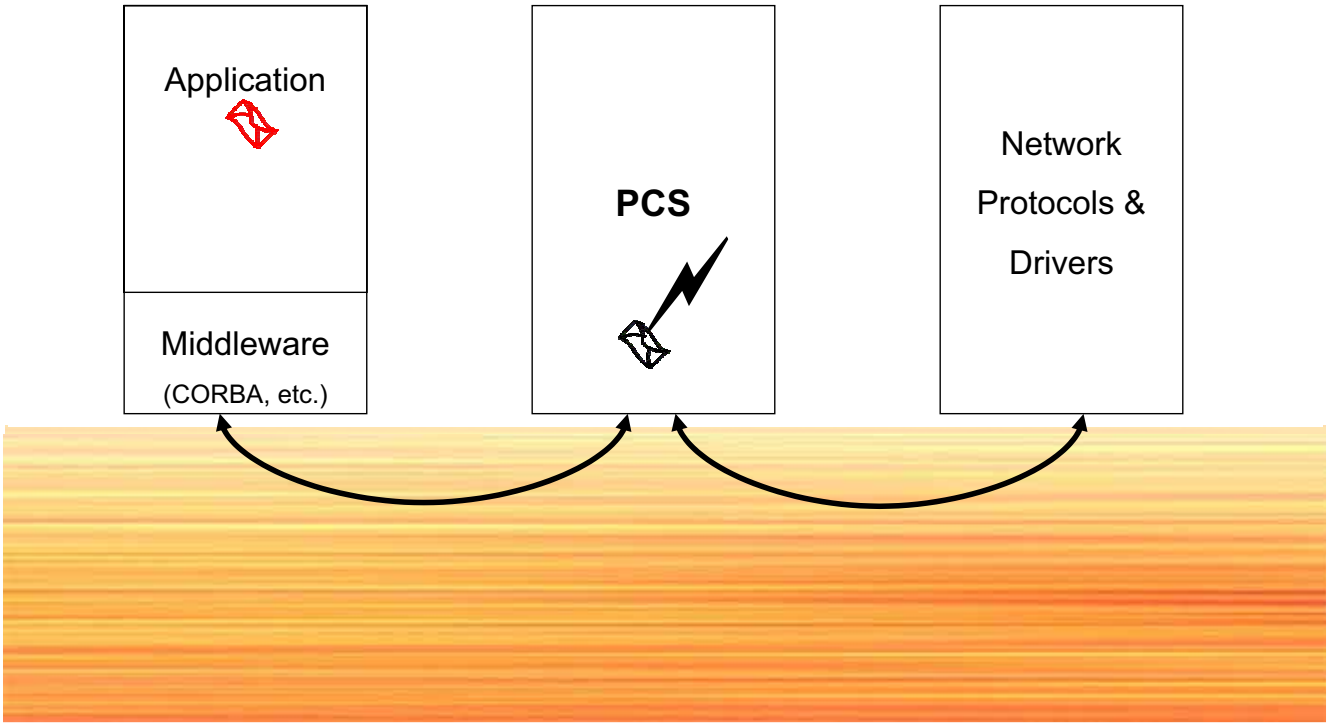
High Assurance Security and Safety for Robotics

## Partitioning the Channel



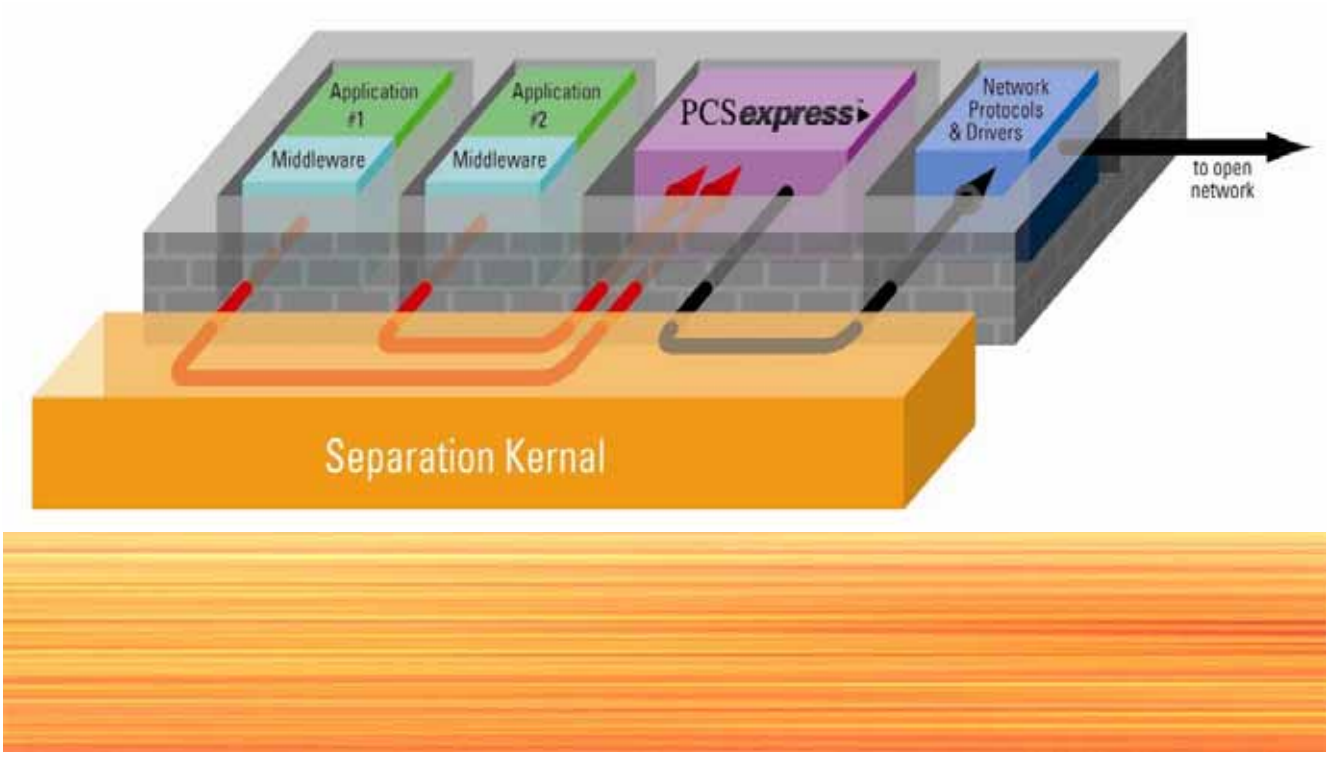
## System Architecture with PCS

High Assurance Security and Safety for Robotics




## System Architecture with PCS (cont.)

High Assurance Security and Safety for Robotics



## The PCS APIs

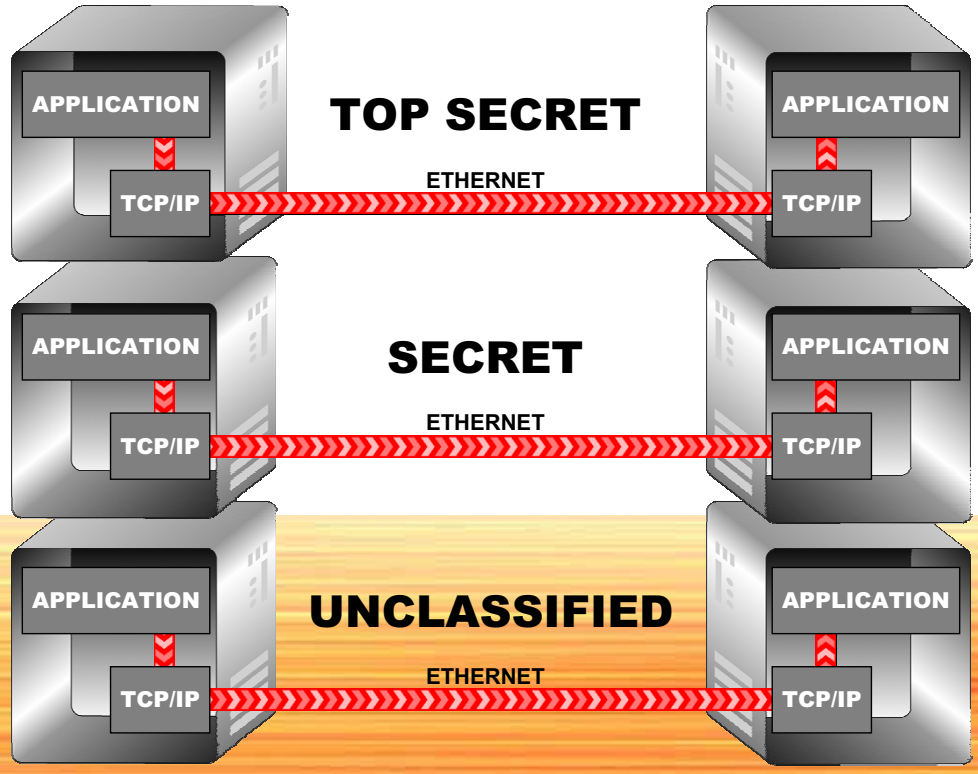


- There are no APIs for PCS!
- PCS is transparent to …
  - Application code
  - Network protocol stacks and drivers
- MILS network interface libraries provide semantics identical to existing user space APIs (MILS libsocket.a, et al)

OBJECTIVE INTERFACE

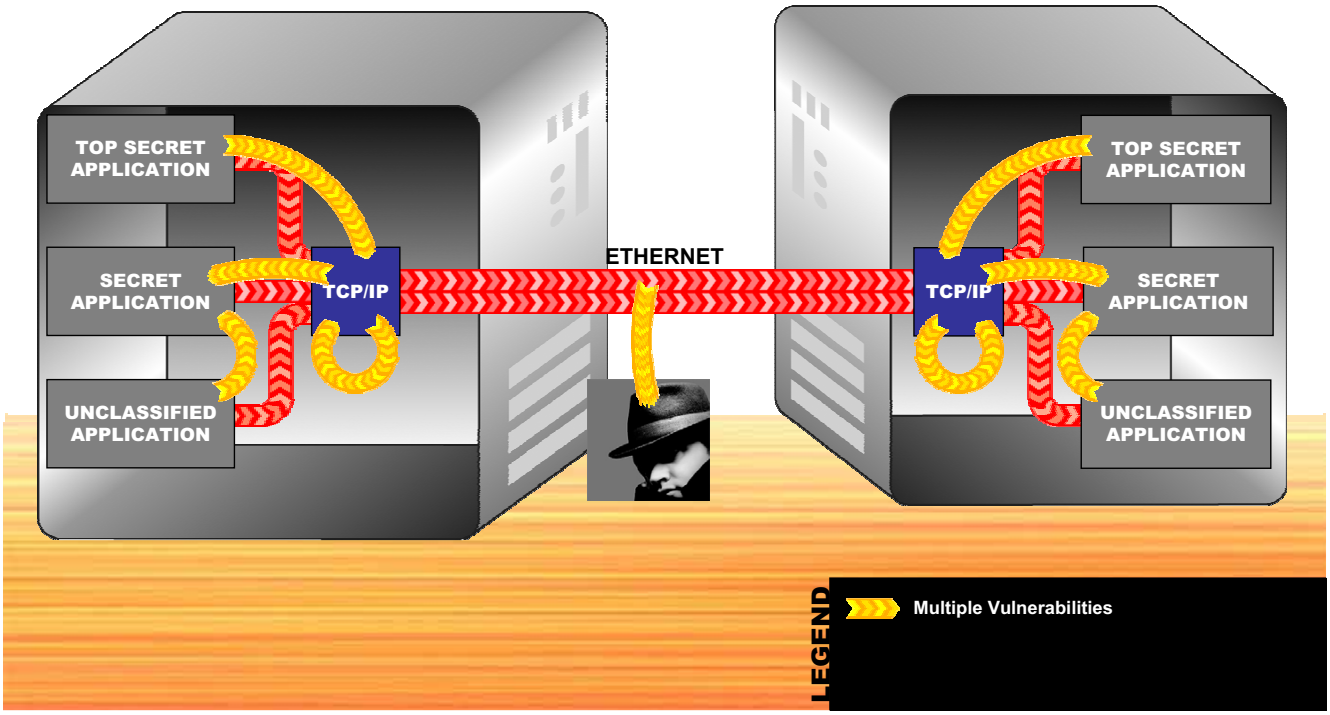
## Air Gap Works But…. Costly, Inflexible, & Awkward

High Assurance Security and Safety for Robotics



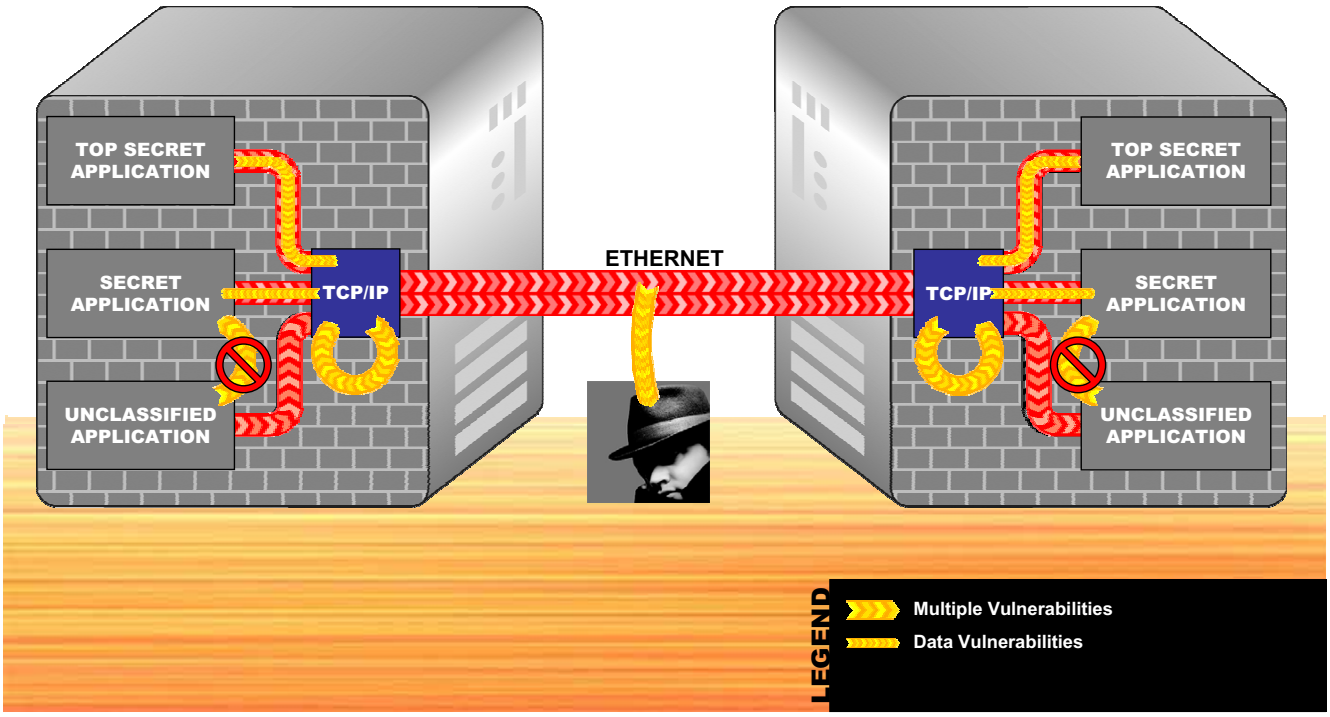
## Combining Levels On Medium Assurance Platforms Is Unsafe

High Assurance Security and Safety for Robotics



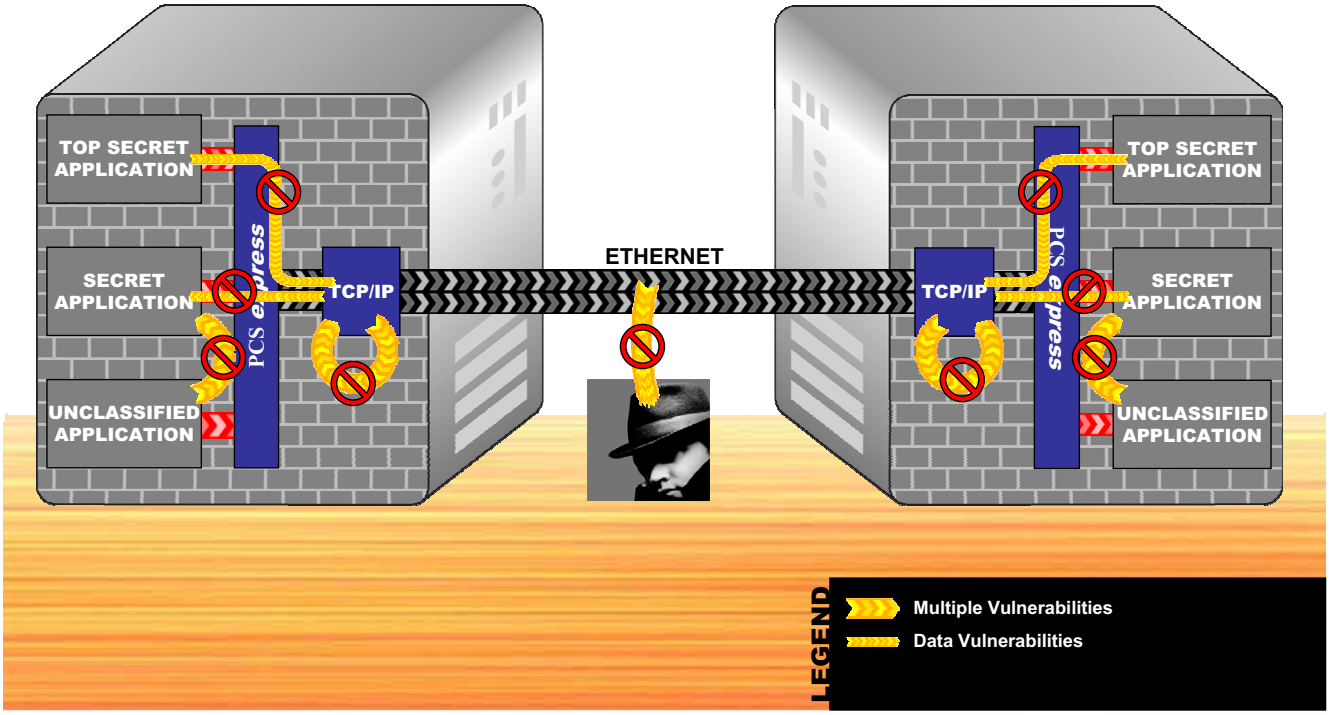

## MILS Separation Kernels Counter Most Internal Threats

High Assurance Security and Safety for Robotics



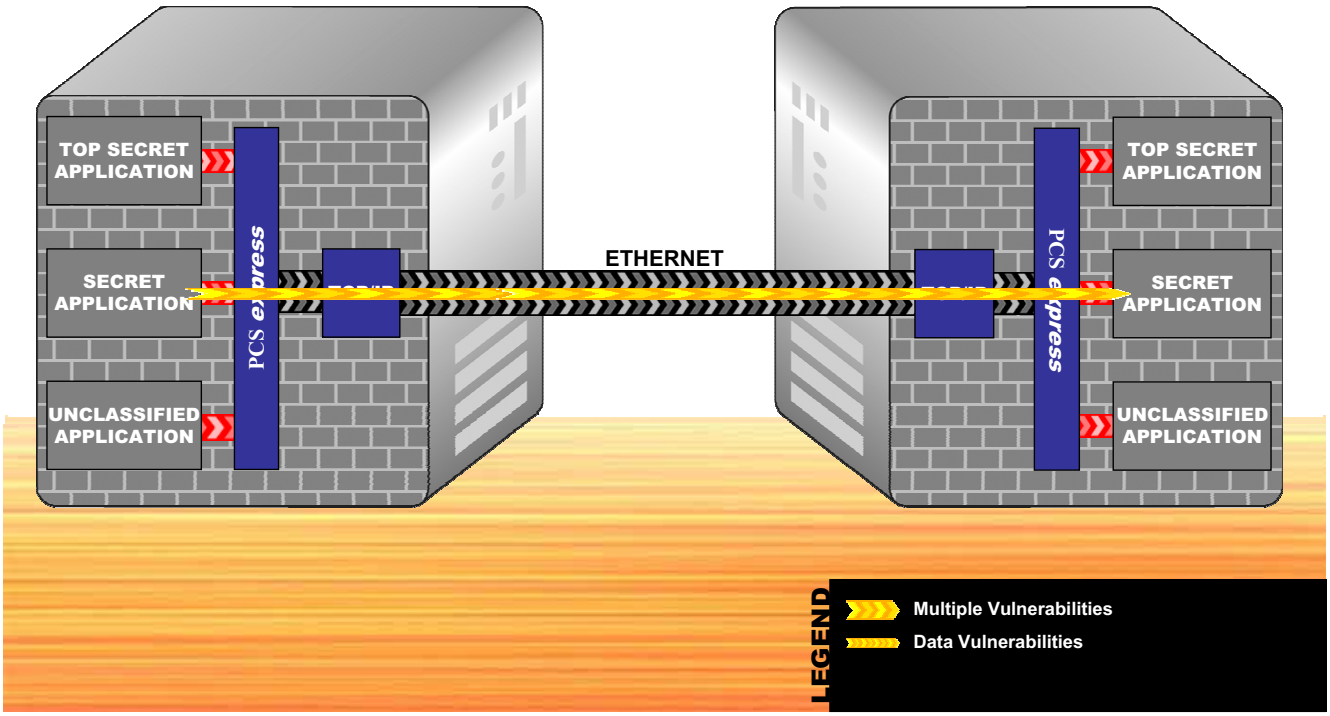
## PCS Completes MILS Separation Kernel

*High Assurance Security and Safety for Robotics*




## Guards Still Needed ... even for Authorized Flows

*High Assurance Security and Safety for Robotics*



## What PCS Is and Is Not

- **PCS is**
  - Like a super VPN configured between partitions in distributed nodes
  - Adds techniques for covert storage and time channel suppression
- **PCS is not**
  - Applications middleware like CORBA, DDS, or Web Services
  - A Guard or Application Firewall
    - Doesn't examine message content
    - Can't enforce security policies delegated to the application layer
  - A total, end-to-end security solution
    - Foundation for application level security
    - *Not a replacement* for application level security

High Assurance Security and Safety for Robotics

## PCS is Trusted Plumbing

- **PCS assumes the network can't be trusted**
  - Leverage COTS stacks, NICs, media, switches, and routers
- **PCS provides trusted data flow among distributed applications and guards**
  - Code that was typically duplicated from partition to partition
- **Access guards and data guards can be tightly focused on the data owner's specific requirements**
- **Trusted data flow enables higher assurance**
  - Smaller code body
  - Simpler logic
  - Formal methods more practical

## Real-time MILS CORBA



- **Real-time CORBA can take advantage of PCS capabilities**
  - ORB*express* RT + PCS = Real-time MILS CORBA
  - Additional application-level security policies are enforceable because of Separation Kernel and PCS foundation

- **Real-time MILS CORBA represents a single enabling application infrastructure**

OBJECTIVE INTERFACE

## RT CORBA & MILS Synergy

High Assurance Security and Safety for Robotics

- **Synthesis yields an unexpected benefit**
  - Flexibility of Real-time CORBA allows realization of MILS protection
  - **MILS is all about *location awareness***
    - Well designed MILS system separates functions into separate partitions
    - Takes advantage of the MILS partitioning protection
  - **Real-time CORBA is all about *location transparency***
    - The application code built with Real-time CORBA is not aware of locality of system logic
    - CORBA flexibility allows optimizations by rearranging where objects live
    - System layout optimizations late in the development cycle

- **Combination of MILS and Real-time CORBA allows system designer**
  - To rearrange system functions to take advantage of protection without introducing new threats to data confidentiality and integrity

## Summary

- **Objective Interface is working closely with:**
  - U.S. Air Force Research Laboratory
  - National Security Agency
  - University of Idaho
  - Lockheed Martin
  - Boeing
  - Raytheon
  - Rockwell Collins

- **To architect high performing MILS middleware that will provide transparent performance for MILS systems**

High Assurance Security and Safety for Robotics

## Objective Interface Systems Overview

- **16 years of supporting mission-critical customers**
- **World leaders in Real-time CORBA products**
- **No venture capital. Zero debt. History of profits. All profits reinvested in growth of company.**
- **Committed to Industry Standards via key contributions to**
  - Object Management Group
  - Open Group
  - IEEE
- **All products written by current employees**
- **Standards-based, Distributed Communications Software for:**
  - Real-time and embedded systems
  - High-performance enterprise systems

**Committed to the success of our customer's projects**

## Standards Commitment

- **Key contributors to the OMG standards**
  - Real-Time CORBA v1.0
  - Fault Tolerant CORBA
  - High Assurance CORBA
  - Data Distribution Specification
  - Dynamic Scheduling (Real-Time CORBA v2.0)
  - Portable Interceptors
  - Multicast: Unreliable and Ordered Reliable
  - High Performance Enablers
  - Lightweight Services
  - … and many more
- **Key contributors to Open Group standards**
  - MILS Partitioning Kernel Protection Profile
  - MILS Partitioning Communications System Protection Profile

High Assurance Security and Safety for Robotics

## Product Line

- **ORB*express*™**
  - High-performance, real-time ORB built from the ground-up for real-time and embedded systems
  - C++, Java, Ada 95
- **PCS*express*™**
  - High-assurance, high-performance secure MLS communications for MILS systems
  - Full information separation and isolation without the performance penalties
  - Allows use of COTS networking technologies for secure MLS communication
- **DDS*express*™**
  - Incredibly fast, small implementation of OMG Data Distribution Service (DDS)
  - Fully interoperable with ORB*express*
  - C, C++, Java and Ada95

## *Additional Information*

- **Obtain a MILS White Paper**
  http://www.ois.com/MILS/
- **Partitioning Communications System Protection Profile**
  http://www.ois.com/download.asp
- **Separation Kernel Protection Profile**
  http://niap.nist.gov/pp/draft_pps/index.html

OBJECTIVE INTERFACE

*High Assurance Security and Safety for Robotics*

**End of Slides**

# Appendix

## Foundational Threats



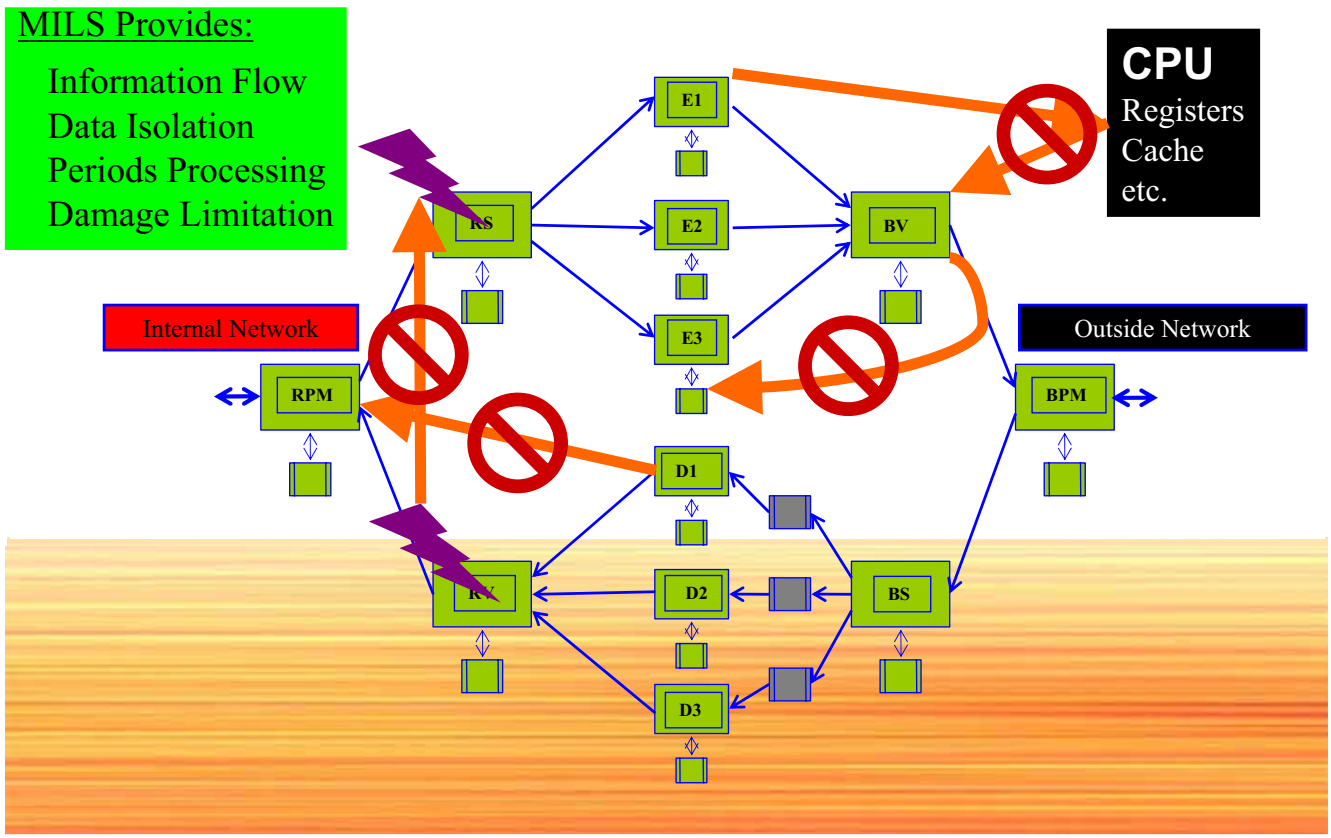
- **Software can only be as secure as its foundation**
- **If the foundation can be successfully attacked, then any system security function that runs on that foundation can easily be rendered ineffective**
- **Foundational threats include:**
  - Bypass
  - Compromise
  - Tamper
  - Cascade
  - Covert Channel
  - Virus
  - Subversion

## MILS Security Policy Example

High Assurance Security and Safety for Robotics

MILS Provides:

- Information Flow
- Data Isolation
- Periods Processing
- Damage Limitation




## Foundational Threats Sample Application

High Assurance Security and Safety for Robotics

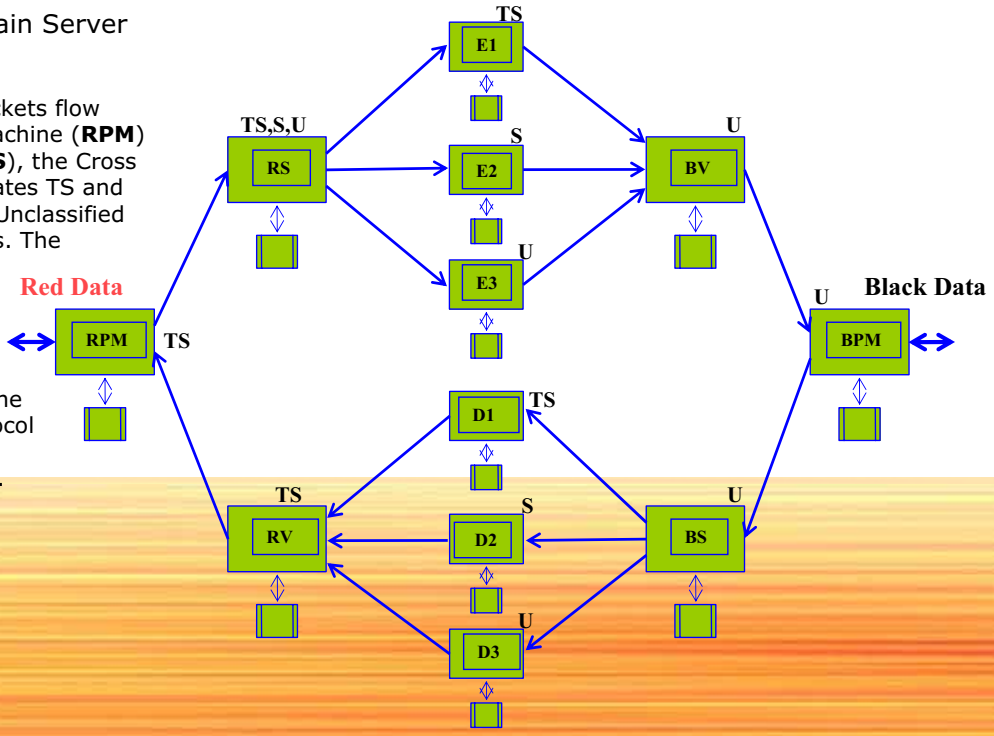
Multilevel Cross Domain Server

Outgoing data:

Top Secret clear text packets flow from the Red Protocol Machine (**RPM**) to the Red Separator (**RS**), the Cross Domain Server, who creates TS and downgraded Secret and Unclassified versions of those packets. The packets are then routed to the appropriate Encryptor, according to level (**E1-E3**). The Black Verifier (**BV**) ensures that this was done properly. The Black Protocol Machine (**BPM**) then transmits the cyphertext.

Incoming data:

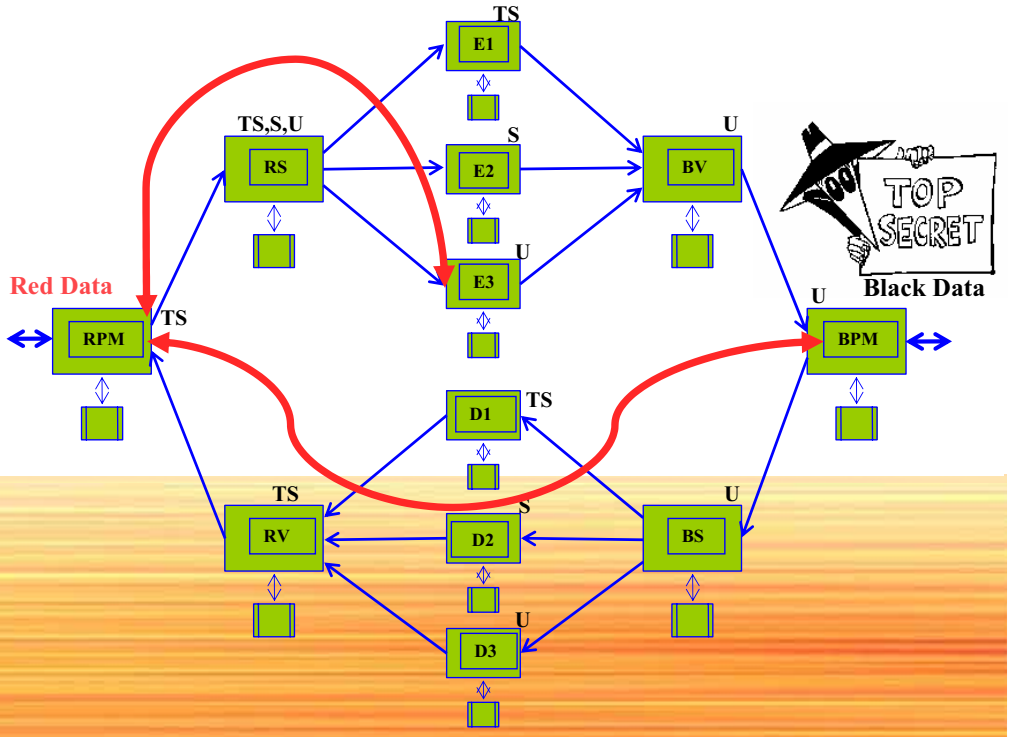
Similar to the above, but in the opposite direction.



## Foundational Threats: Bypass

High Assurance Security and Safety for Robotics

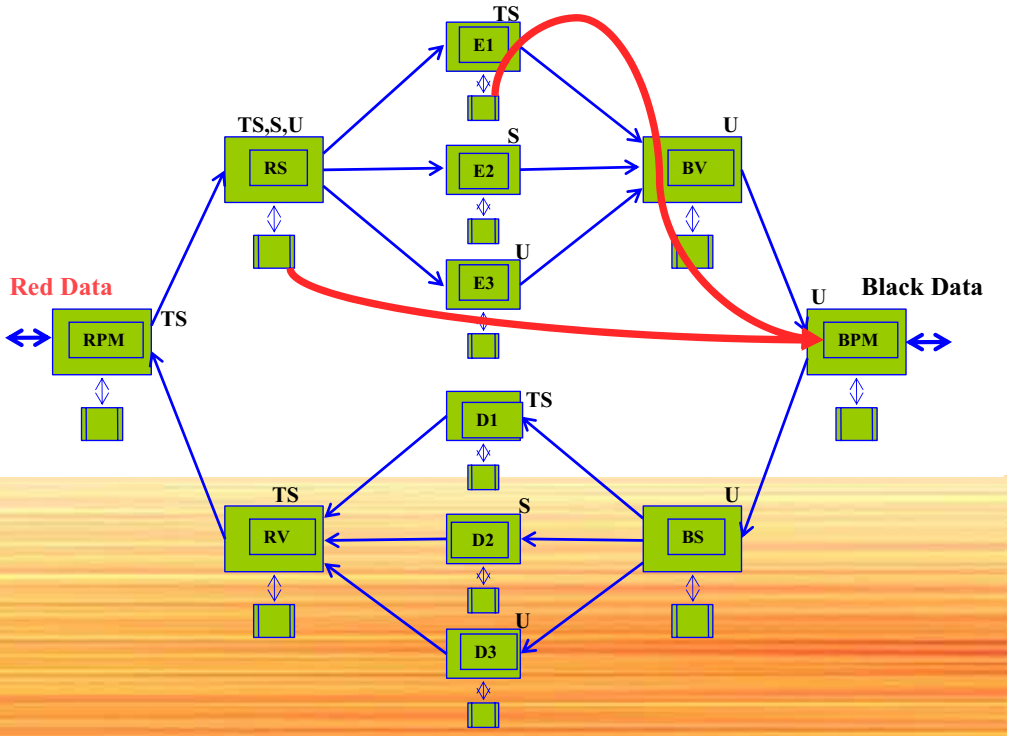
- ✓ **Bypass**
- ✓ Compromise
- ✓ Tamper
- ✓ Cascade
- ✓ Covert Channel
- ✓ Virus
- ✓ Subversion




## Foundational Threats: Compromise

High Assurance Security and Safety for Robotics

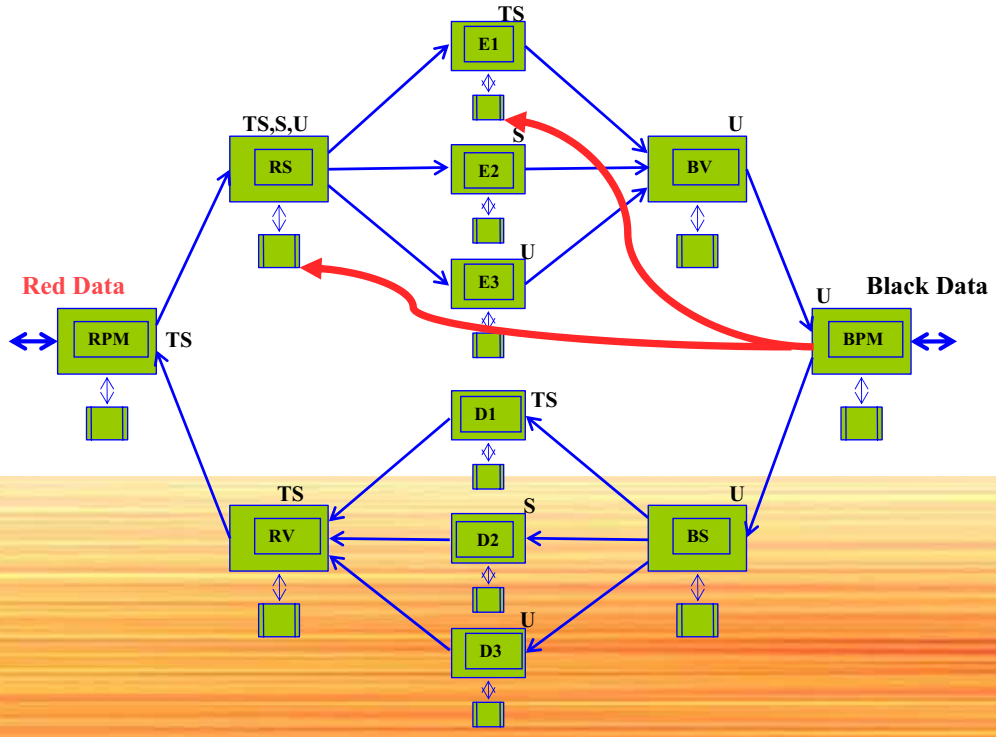
- ✓ Bypass
- ✓ **Compromise**
- ✓ Tamper
- ✓ Cascade
- ✓ Covert Channel
- ✓ Virus
- ✓ Subversion



## Foundational Threats: Tamper

High Assurance Security and Safety for Robotics

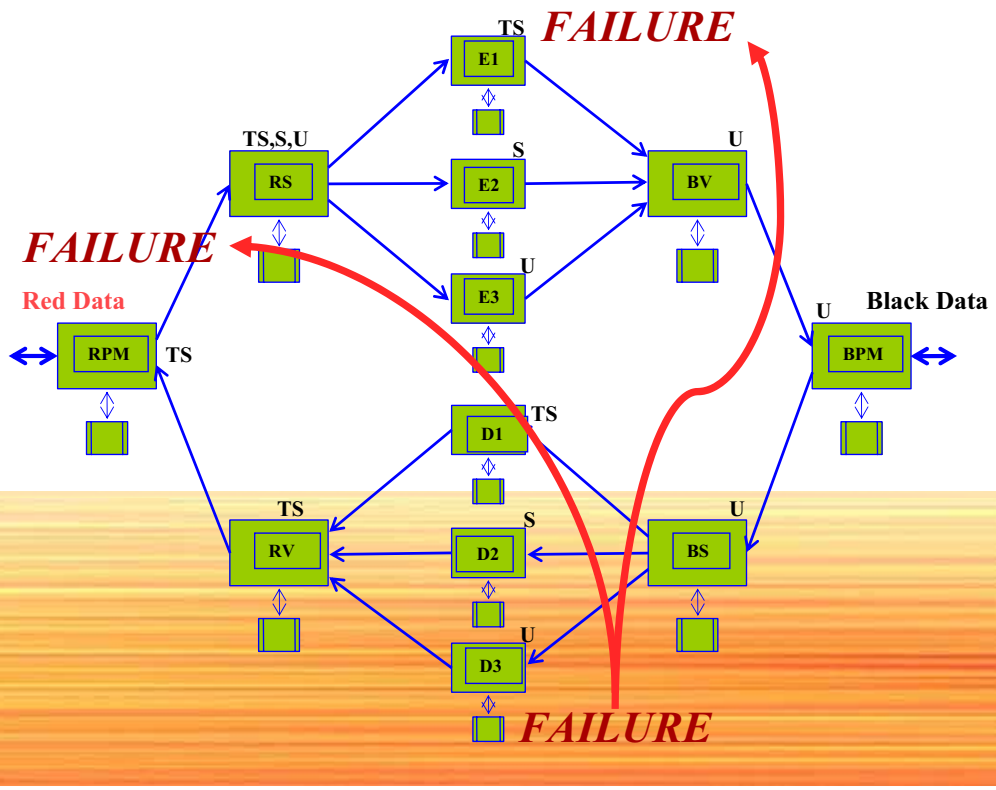
- ✓ Bypass
- ✓ Compromise
- ✓ **Tamper**
- ✓ Cascade
- ✓ Covert Channel
- ✓ Virus
- ✓ Subversion




## Foundational Threats: Cascade

High Assurance Security and Safety for Robotics

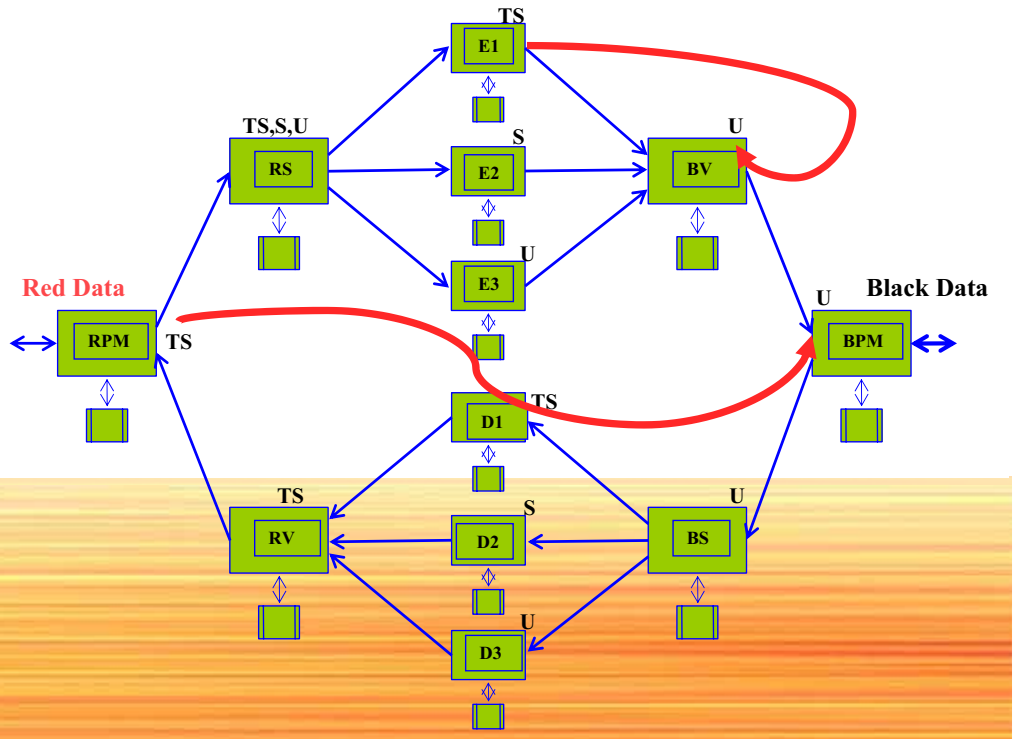
- ✓ Bypass
- ✓ Compromise
- ✓ Tamper
- ✓ **Cascade**
- ✓ Covert Channel
- ✓ Virus
- ✓ Subversion



## Foundational Threats: Covert Channel

High Assurance Security and Safety for Robotics

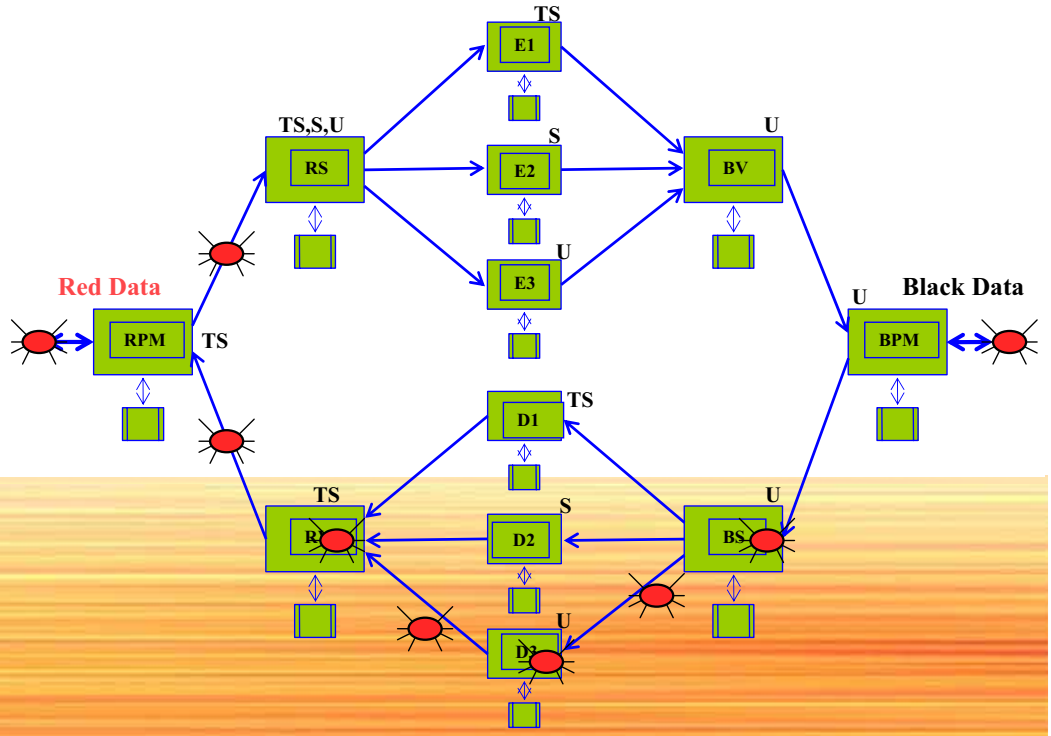
- ✓ Bypass
- ✓ Compromise
- ✓ Tamper
- ✓ Cascade
- ✓ **Covert Channel**
- ✓ Virus
- ✓ Subversion




## Foundational Threats: Virus

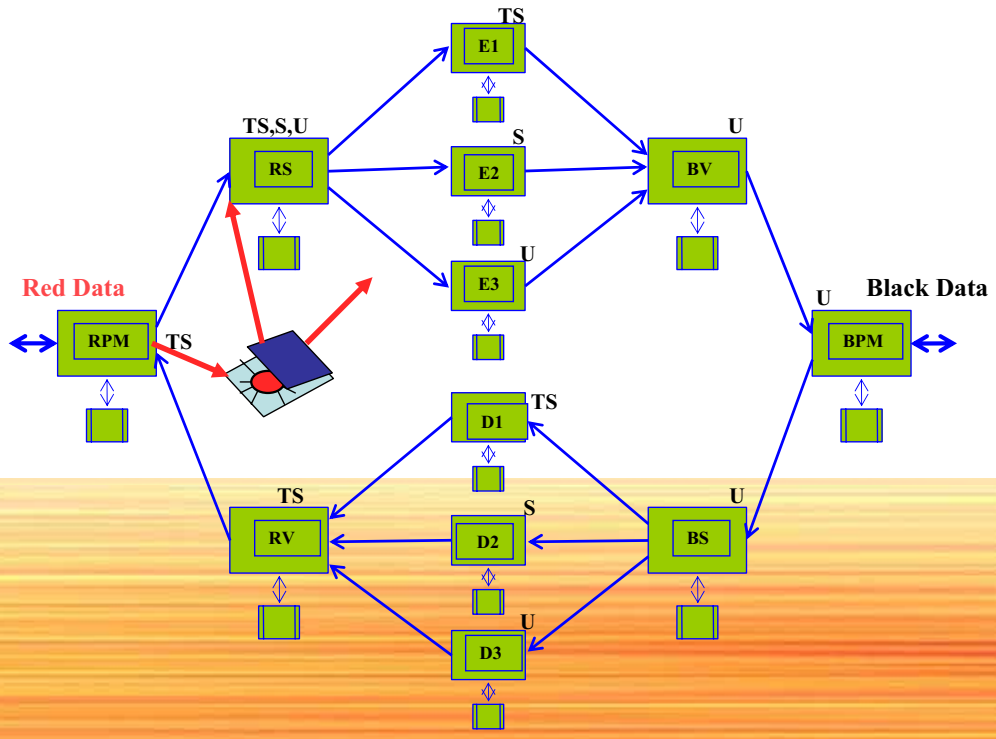
High Assurance Security and Safety for Robotics

- ✓ Bypass
- ✓ Compromise
- ✓ Tamper
- ✓ Cascade
- ✓ Covert Channel
- ✓ **Virus**
- ✓ Subversion



# Foundational Threats: Subversion

- ✓ Bypass
- ✓ Compromise
- ✓ Tamper
- ✓ Cascade
- ✓ Covert Channel
- ✓ Virus
- ✓ **Subversion**



robotics2005-12-11

# Standardization on Interfaces to Robot Devices

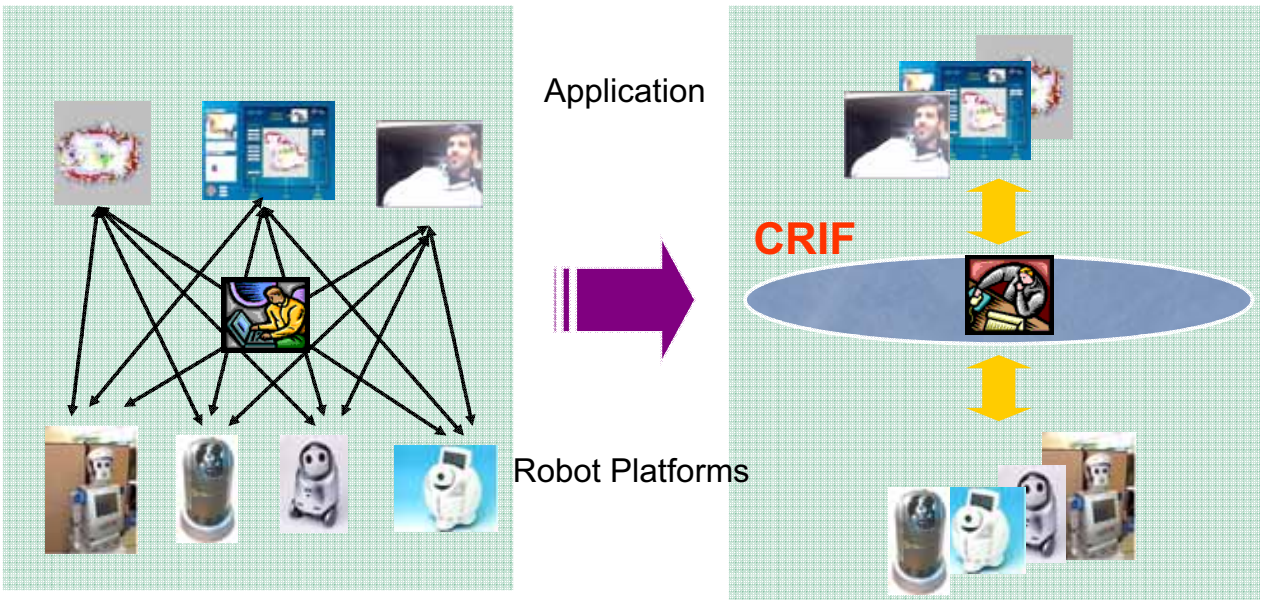
A Response to OMG Robotic Systems RFI
December, 2005

Electronics and Telecommunications Research Institute
Intelligent Robot Division

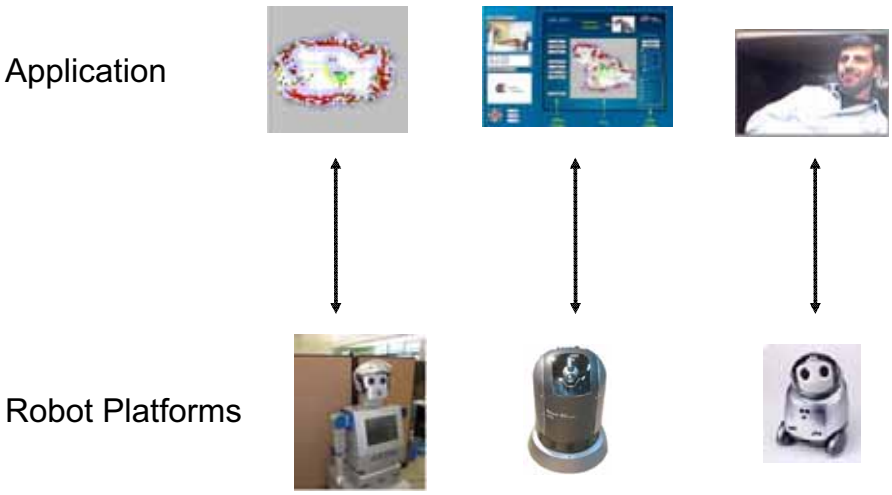
Lee, Seung-Ik

## Needs for Standardized Interface

- Provides common access methods to robot hardware devices → CRIF, what we call
- CRIF: Common Robot Interface Framework



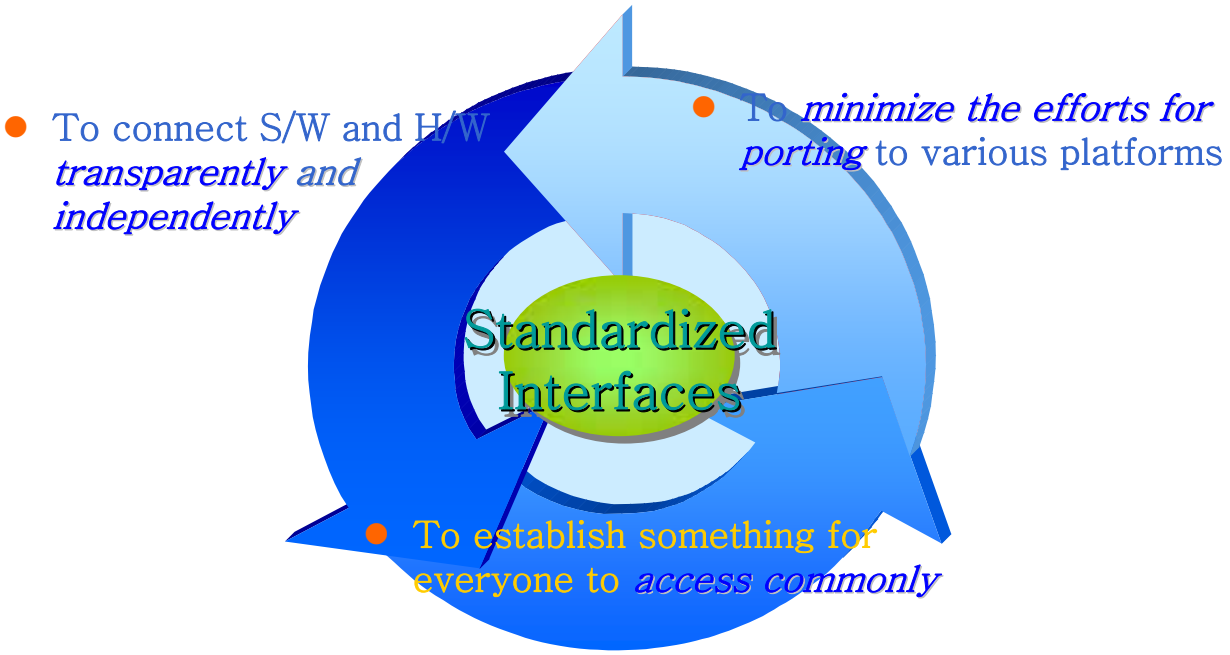
# Current Status




Each robot developer writes an application for his/her own robot in their own way (Everyone makes their own CRIF)



# What can be achieved?



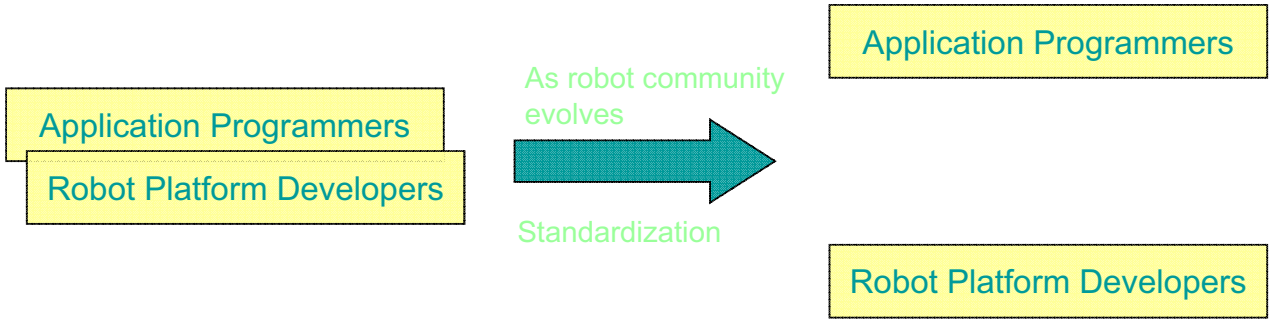
# Two Users of This Standardization

## Application Programmers

They want to write a good service S/W for robots without considering target platform

## Robot Platform Developers

They want as many applications on their platform as possible






# By Introducing CRIF

## Application Programmers

⇑ their applications run on various robot platforms

⇓ their control is restricted by the APIs of the framework

→ make the framework more general

## Robot Platform Developers

⇑ many applications are available for their robots

⇓ they don't like to take a burden of implementation

→ make the framework more simple

# Strategy in Designing CRIF

*Simplicity comes First*

We do NOT cover all kinds of robot platforms

➡ Focusing on Wheel-type robot, probably typical for service robots

We focus on general functionalities rather than sophisticated ones

➡ It is sufficient for most applications of service robots

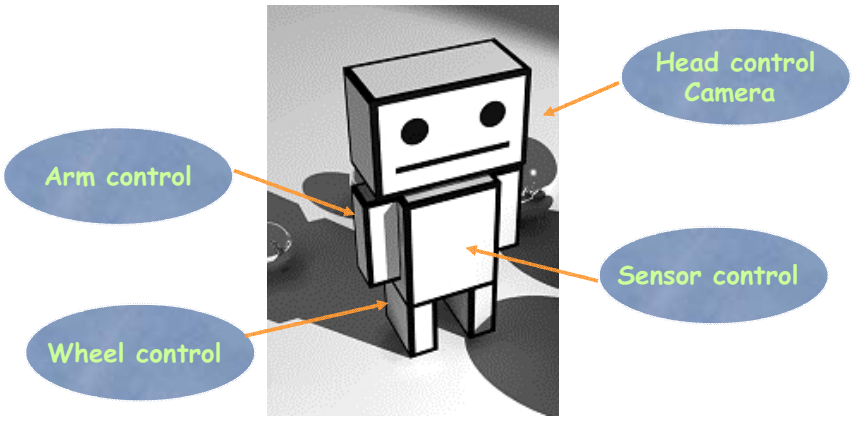
We provide developer's kit for robot platform developers

➡ Minimum efforts for robot platform developers



# Design Flow

- Building Virtual Robot
- Defining the functions for controlling Virtual Robot and its components
  →Constructing API set



# Assumptions

- Wheel type Robot
  → Robot Abstraction Model

- Client/Server structure
  to solve the problem of resource monopoly
  - Client : Applications
  - Server: Robot H/W platform



# We should define..

- Types of hardware devices
  - Wheel, sensors, cameras, head, **...**

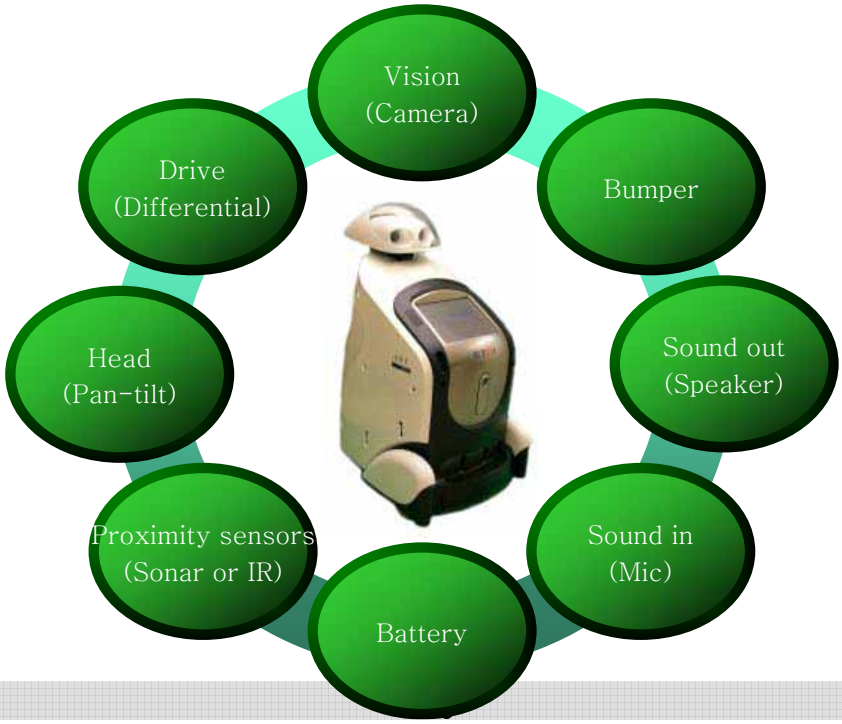
- Interfaces
  - Defines abstracted common interfaces (BTW applications and robot platform), that is, APIs

- Data types
  - Required to pass parameters and return values from the APIs

- Coordination system
  - Global coordination system, local coordination system

# Hardware devices

- Includes most popular robot devices and morphology
- In the beginning stage, we may start from hardware devices for mobile robot



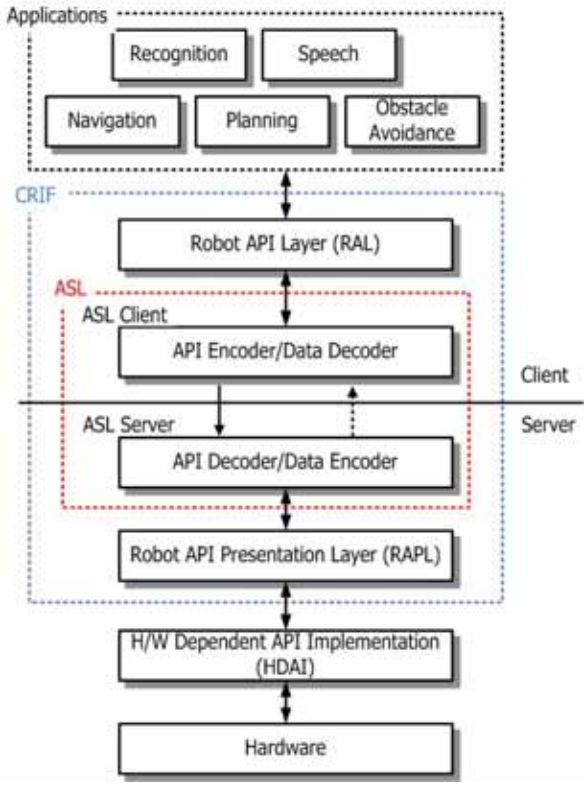



# API Set

- API set
  - About 50 APIs including 15 APIs for drive and 13 for vision device
- API examples
  - Drive (differential wheel type)
    - Move, Rotate, SetAccel, StopWheel, GetRobotPosition, IsWheelRunning , ...
  - Head
    - GetRobotPosition, MoveHead, GetHeadInfo, ...
  - Vision
    - GetRawImage, GetCompressedImage, GetCameraZoom, ...
  - Proximity sensors
    - GetProximitySensorInfo, GetProximitySensorData

# Structure of CRIF-Framework

- In charge of communication between robot applications and robot platform
- Client-Server model
- Remote procedure call
  - Socket, COM, CORBA, etc.
  - Not specified by CRIF-Framework itself

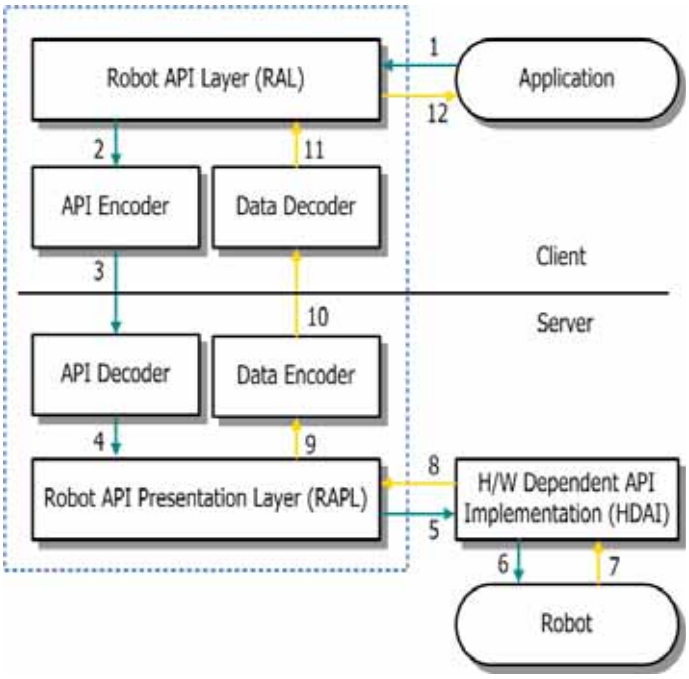




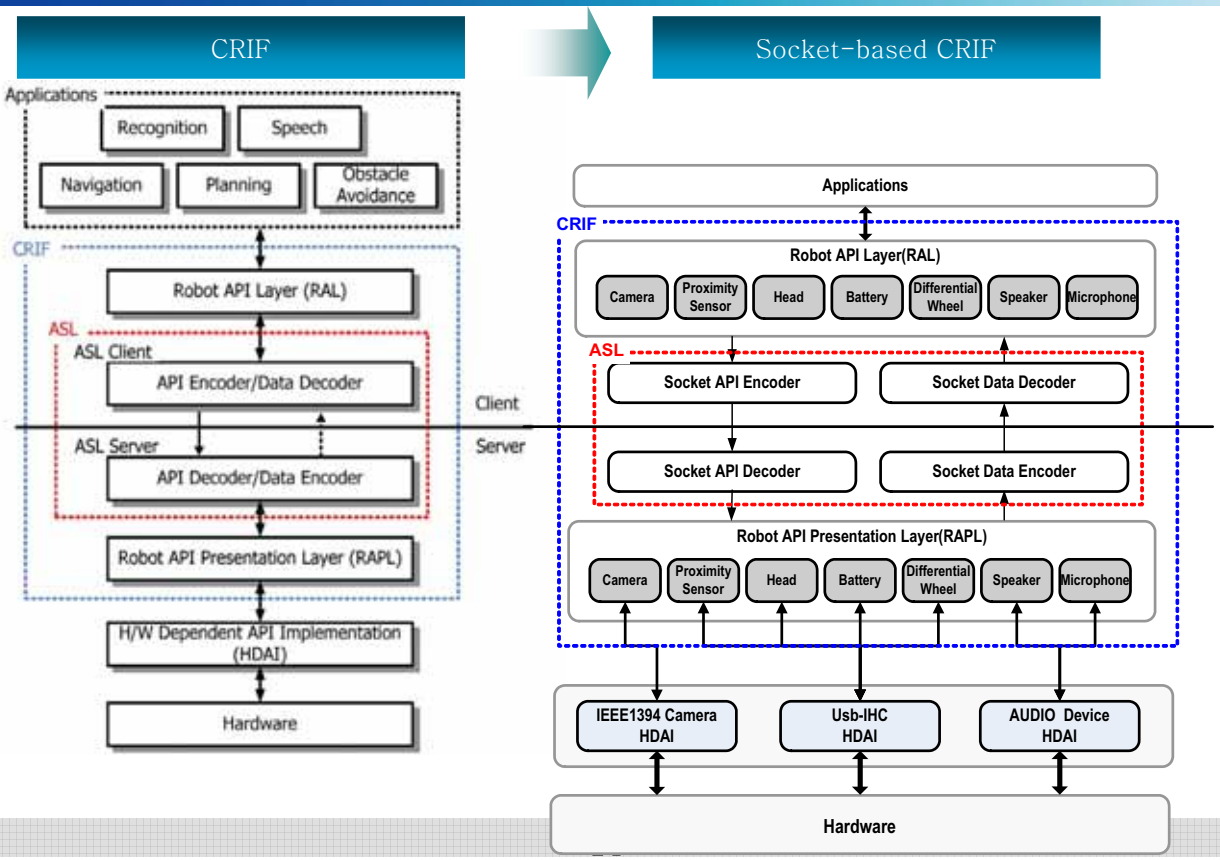



# CRIF-Framework Process Flow

1. **An application requests an API call defined in RAL**
2. **Information about the requested API such as its name and parameters is passed to API Encoder**
3. **The passed API information is transmitted to API Decoder according to the specification of the predefined remote procedure call**
4. **According to the information transmitted to API Decoder, the same named API with the one called in 1 is called in RAPL**
5. **The requested API in RAPL in turn calls an API of the same name in HDAI. The called API of HDAI contains the real implementation unders tandable by the real hardware devices**
6. **The requested HDAI API is sent to the robot devices after being transformed into a sequence of commands that they can understand**
7. **Execution results of the called API go back to its caller in succession. When an API, for example, requesting to read values of a infrared sensor is called, the sensed values of the infrared sensor is returned back until it reaches the original caller application**



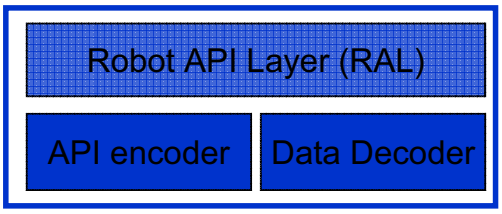
# Reference Implementation using Socket



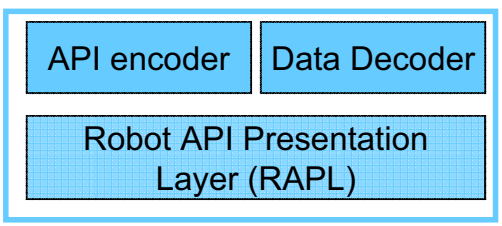


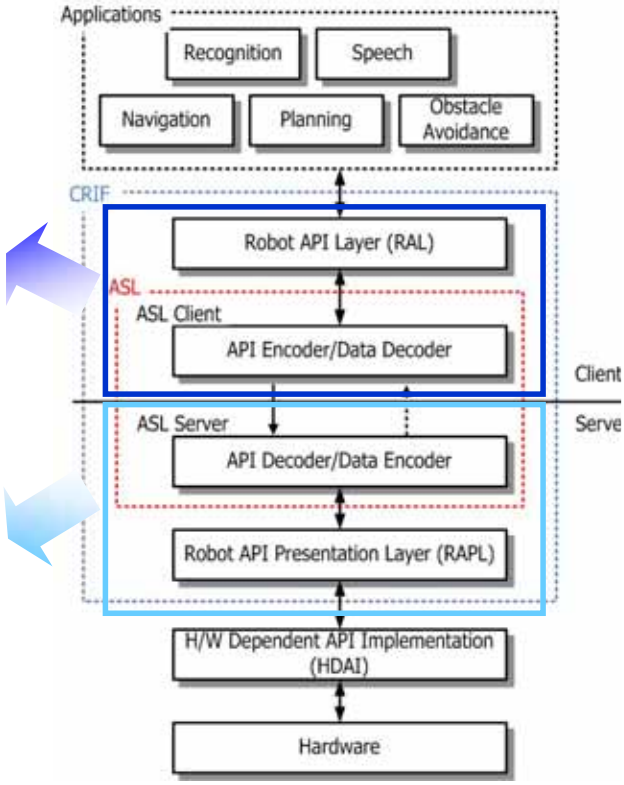

# CRIF-Framework Library

- For application developers

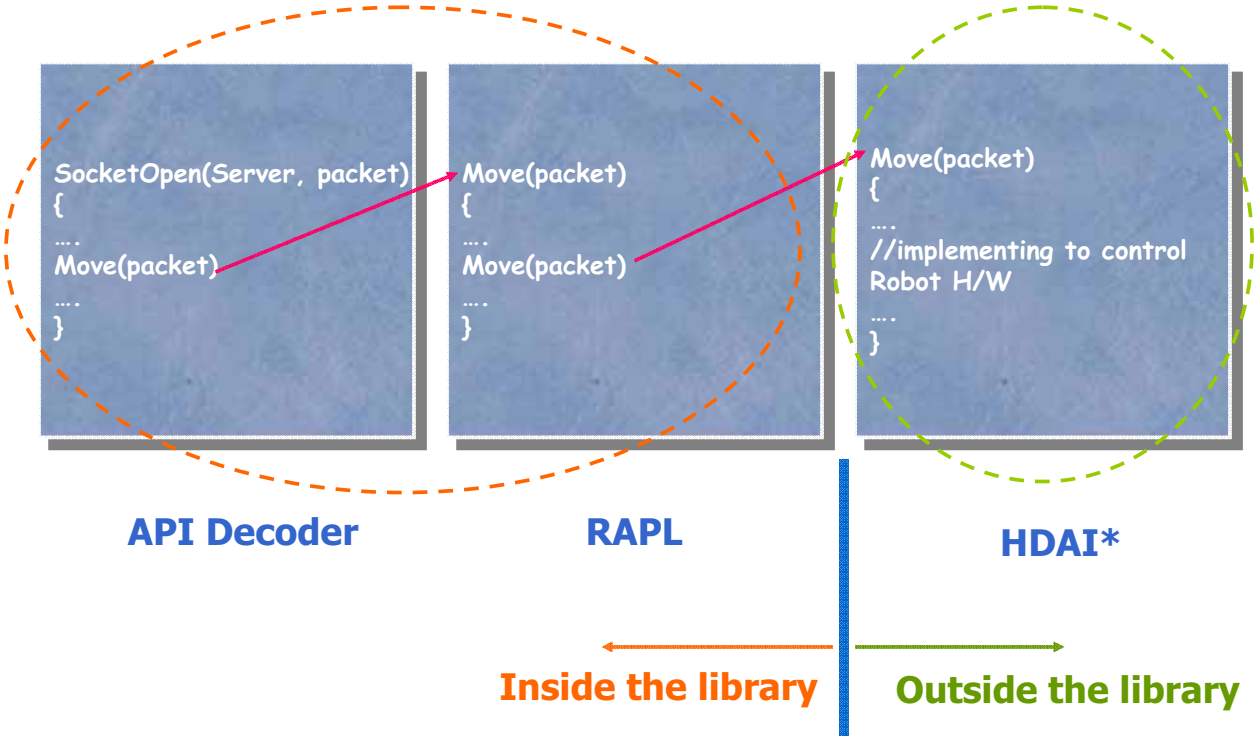



- For robot platform developers



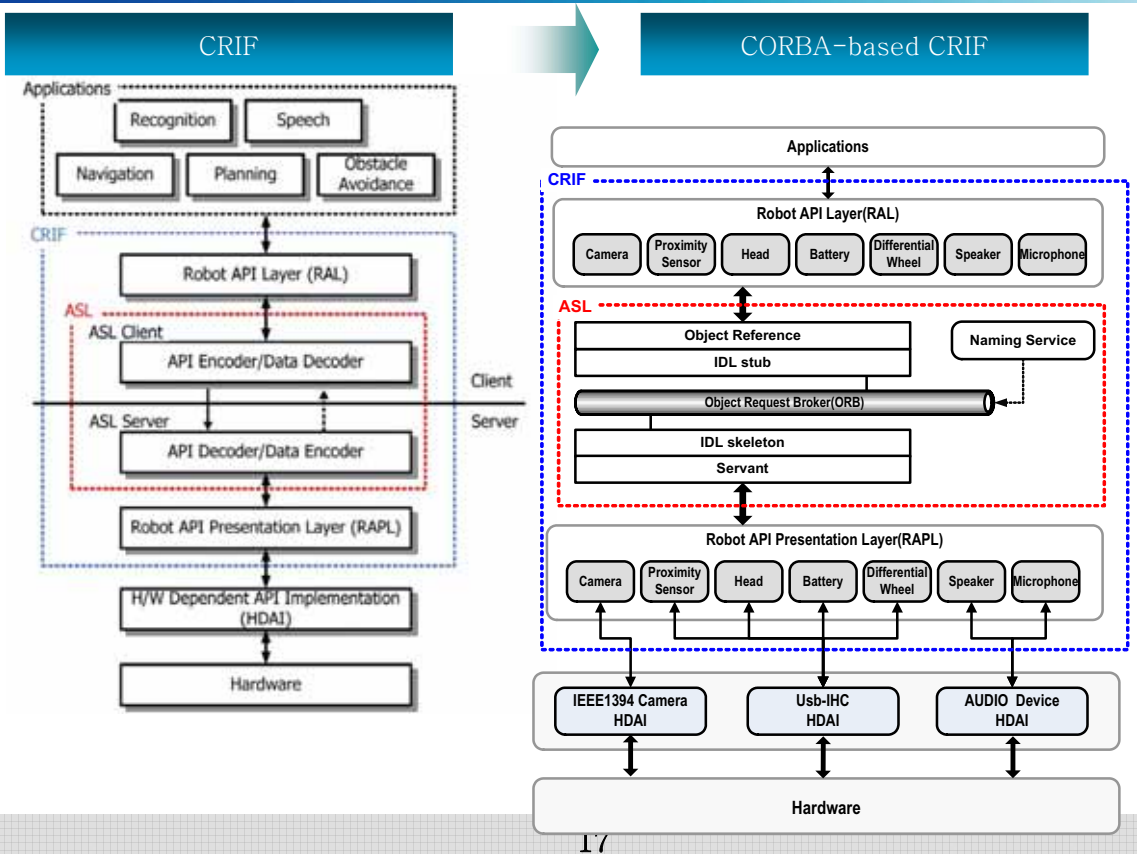



# A Snapshot of the library

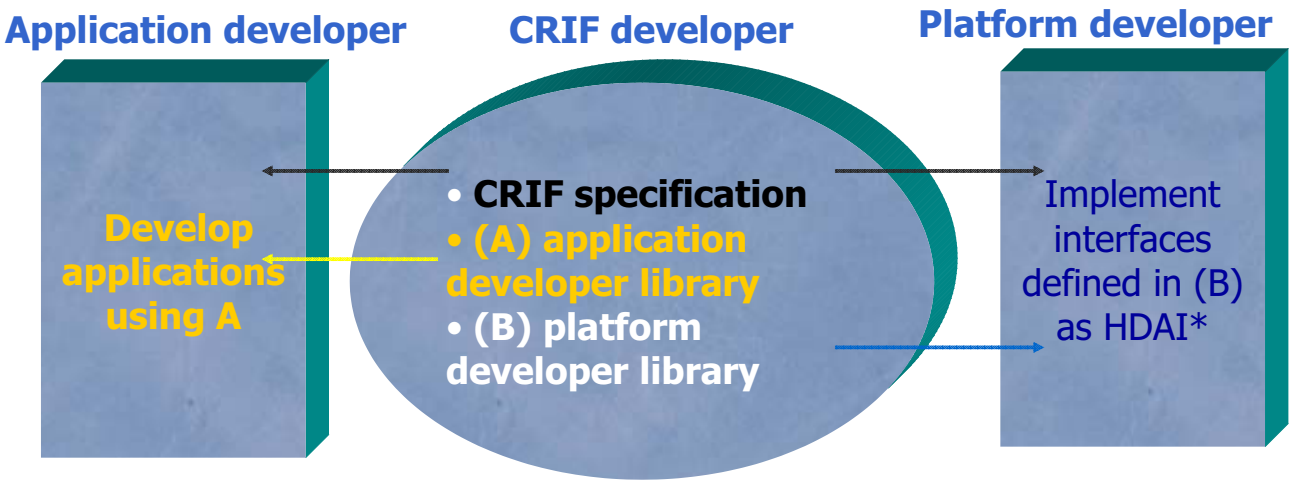





# Reference Implementation using CORBA



# Developers' role




*HDAI(H/W Dependent API Implementation)



# Current status

- Version: 1.2
- Supported platforms

| Robot | Company | OS | Status |
|---|---|---|---|
| Wever-R1 | ETRI | Windows / Linux | Implemented |
| ETRO | ETRI | Windows | Implemented |
| | LG Electronics | Linux | Implemented |
| Scorpion | Evolution Robotics | Linux | Implemented |
| iRobi | Yujin Robotics | Windows | On Going |
| Robo PC | ETRI | Windows | On Going |
| Robo Air Cleaner | ETRI | Windows | On Going |

# Concluding Remarks

- CRIF = CRIF-Interface + CRIF-Framework
- Packaged support for device abstraction and network access

- Pros and cons
  - Portability, applicability, independence of developmental process, support for network environment
  - Almost impossible to define every kinds of APIs or functions
  - Platform-specific codes are inevitable

- In the future, we are going to
  - Extend CRIF-Interface to include more hardware devices



# Research and Development of Personal Robot in NEC

December 7, 2005

Yoshihiro Fujita
Media and Information Research Laboratories
NEC Corporation

http://www.incx.nec.co.jp/robot/





## Personal Robot as User Friendly Computer

**Goal:**

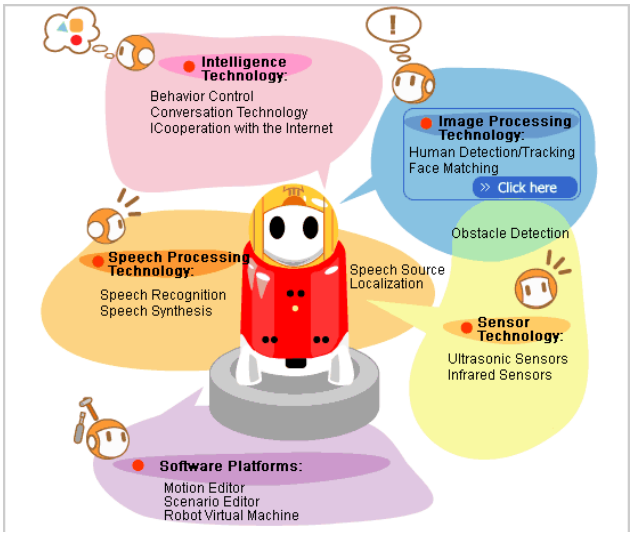
To develop easy, delightful and enjoyable Interface that the elderly, children and novice can use without any stress, anxiety and effort.

**Assumption:**

Friendly shape and motion of robot can lower users mental barrier.

# Research Platform: Personal Robot PaPeRo

- Research Prototype Robot (not for sale)
- PaPeRo = Partner-type Personal Robot
- Focusing not on mechatoronics development, but on interaction between human and robot
  - Integration of sensing, recognition and behavior control software on laptop PC based hardware.
  - Improvement of software and quest for applications based on experiment in real environment.






# Research Activities

- Exploitation and Improvement of Sensing
  - Sound : Sound Direction Detection, Speech Recognition
  - Vision : Face Recognition, Obstacle Detection
  - Others : Ultrasonic, Infrared and touch sensors
  - Interaction
- Development and Improvement of Software Platform
  - Script-base behavior and action Programming
  - Development Environment
- Experiment in Real Environment to find applications
  - Monitor test at 100 homes
  - Experiment at nursery, nursing home, kindergarten and exhibition (including 6 month test at EXPO Aich Japan)

# Why we develop domestic software platform

- Developing behavior of robot in real application is a huge task. (We once developed a huge program but it last only three weeks. After three weeks user get bored as they enjoy all contents.)
- We want to develop application more efficiently, so that researcher can take a rest.
- There was no appropriate software platform for our robot. So we developed by ourselves.
- We look forward to using sophisticated robot platform under OMG specification in future!

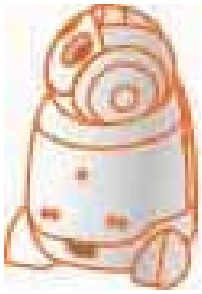




# Software Platform development

- Development and Improvement based of experiment
  → Practical and efficient development is most important

| Requirement | Solution |
|---|---|
| Intuitive software development by designers | Visual behavior and motion editor |
| Customizability | Extension of editor function by Plug-in mechanism |
| | Verification of Customization → Coordinate whole system to the target robot |



# Intuitive Software development

## Two types of application programming language and tools

| Type of Language | Duration of action | Drive Criteria | Example | Development tools |
|---|---|---|---|---|
| Action script | Action for short period | Time driven | Gesture, dance, short distance motion | Action editor |
| Behavior script | Behavior for long period | Event Driven | Dialogue, interaction, longdistance motion | Behavior editor |

※ Behavior script activates action script.



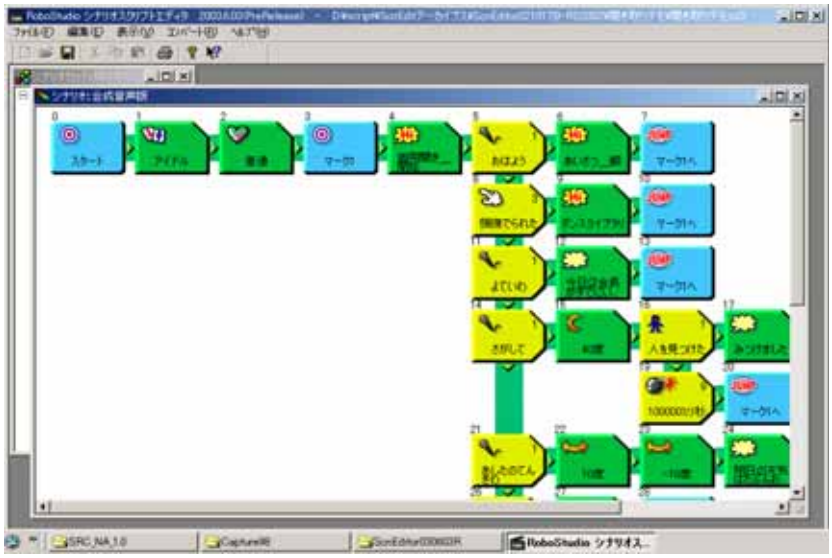


# Action editor

- Each line correspond to robot parts (functions)
- Right direction correspond to time scale
- Parts can be added by developing a plug-in module
- Actions are used by behavior scripts

# Behavior editor

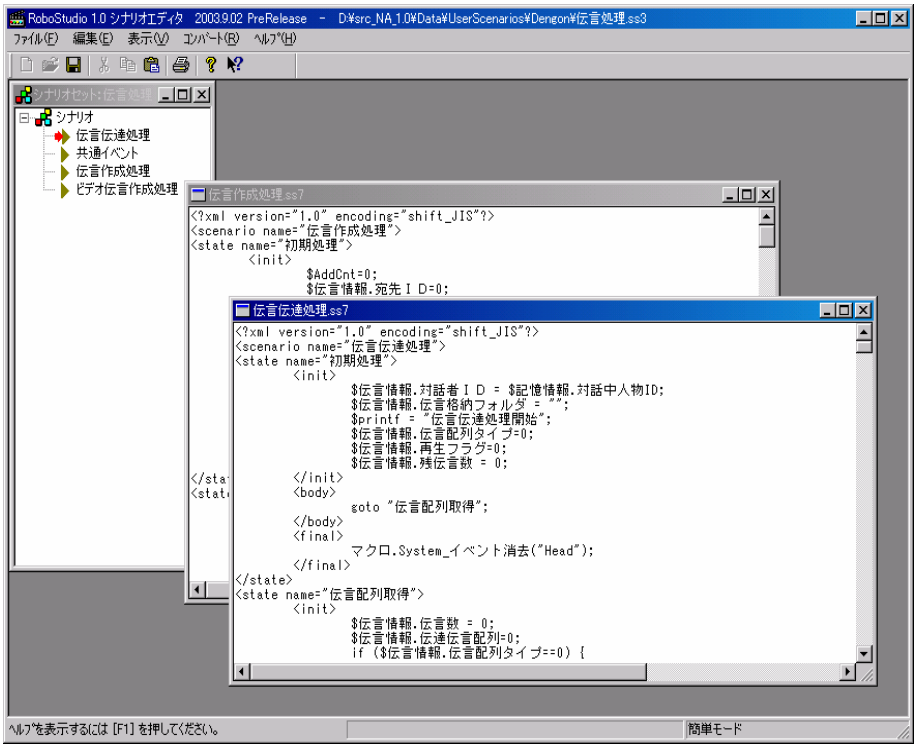
- Each node correspond to action
- Flow can be programmed by connecting nodes
- Program is compiled to script language
- Higher level programmers can write script directry



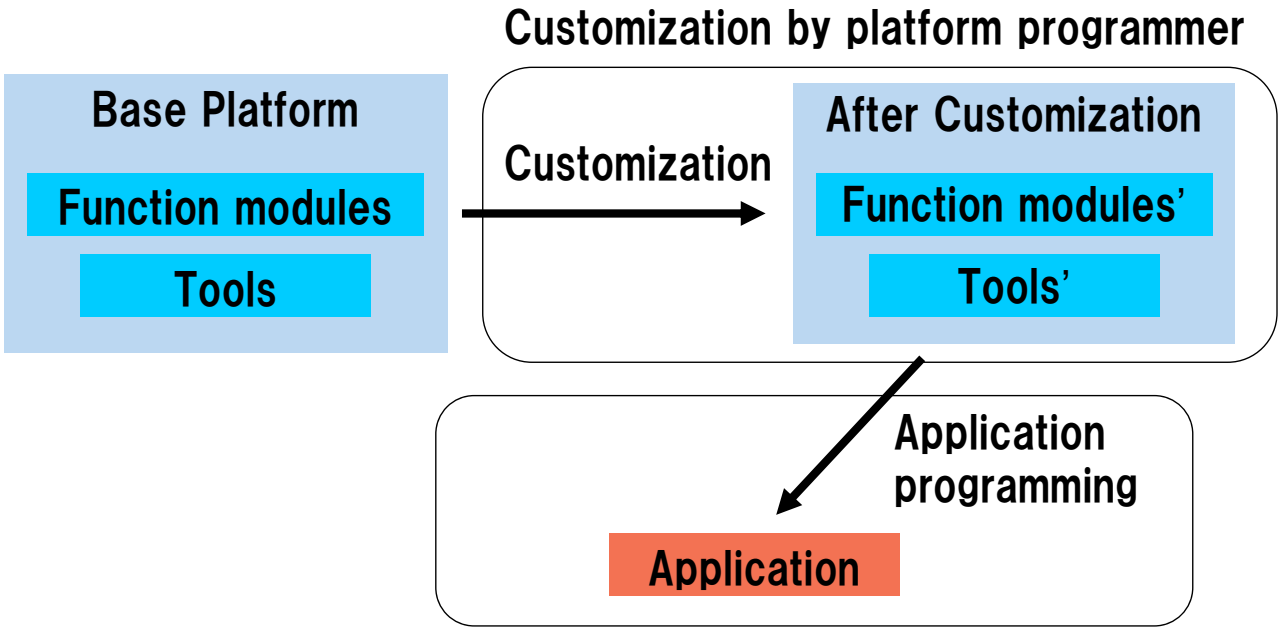




# Editor support for behavior script



# Customization




Application development by application programmer
(In PaPeRo's case: designers, art university students)





# Customization process

**Develop function modules**

(1) Develop worker (function module)

(2) Define interface (Step 1)

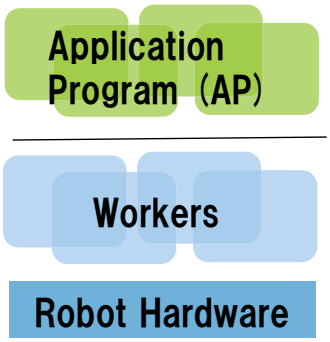
**Customize tools**

(3) Develop plug-ins (Step 2)

(4) Verify interface definition (Step 3)

# Step 1: Define interface

- Functions consist of “workers”
  (worker is a hardware dependent function module)
- Worker’s interface is defined by Worker Definition File (WDF)




API defined by WDF

Example of workers in PaPeRo
- Mechanical worker (feet, head)
- LED worker
- Speech recognition worker
- Face recognition worker





# Example of Worker Definition File (WDF)

- Define interface by XML description

```
<?xml version="1.0" encoding=“UTF-8" ?>
<!DOCTYPE worker SYSTEM "./worker_def.dtd">

<worker name="U01W01" dll="UsrWorker.a" class="CUsr01Worker01"
        errsound="v_err_dekinai_03.wav">
  <event name=“Kitchen Timer Time Notification"/>
  <command name=“Kitchen Timer Start” mode=“sync” type="int">
    <arg type="int" name=“Duration(Second)"/>
  </command>
  <command name=“Kitchen Timer Stop" mode="sync" type="int"/>
</worker>
```

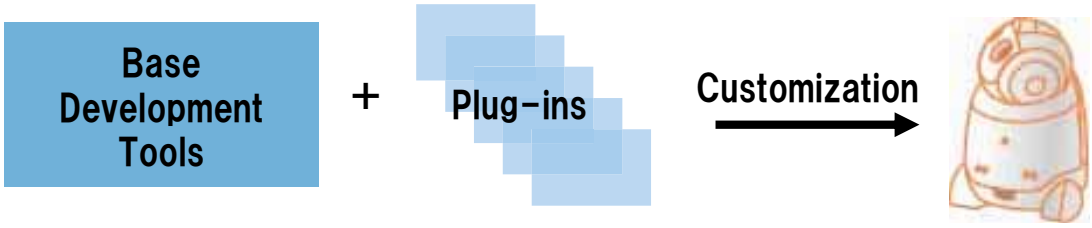
Worker definition

Event devinition

Command definition (Synchronous)

# Step 2: Develop plug-ins for development tools

- Plug-ins provides program interface for robot dependent parts and functions
- Action editor is made by Java and its plug-ins is also made by Java libraries

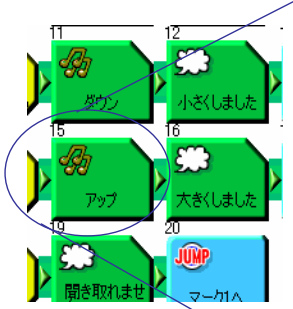







# Example of plug-in for behavior Editor

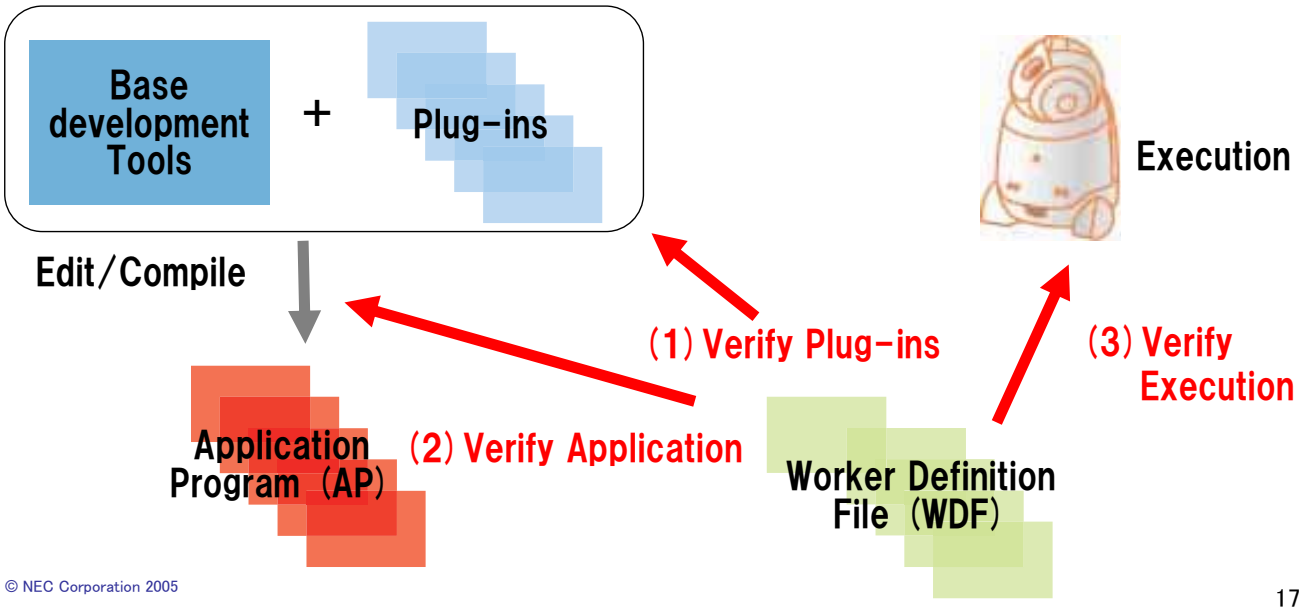
- Parts for application programmer consist of basic parts (such as flow control) and plug-ins (robot dependent modules)
- Part for Behavior editor is defined by XML description




```
<?xml version="1.0" encoding="shift_JIS" ?>
<partdef name="volume">
  <type> action </type>
  <argtype> selectable </argtype>
  <description> up-down volume </description>
 <selection>
    <item value="10" > up </item>
    <item value="-10" > down </item>
  </selection>
  <macros>
     <command name="deneral" id="23510001"
          arg="{Speech.default volume($x);}" />
      <trans exit="true"/>
  </macros>
</partdef>
```

# Step 3: Verify interface definition

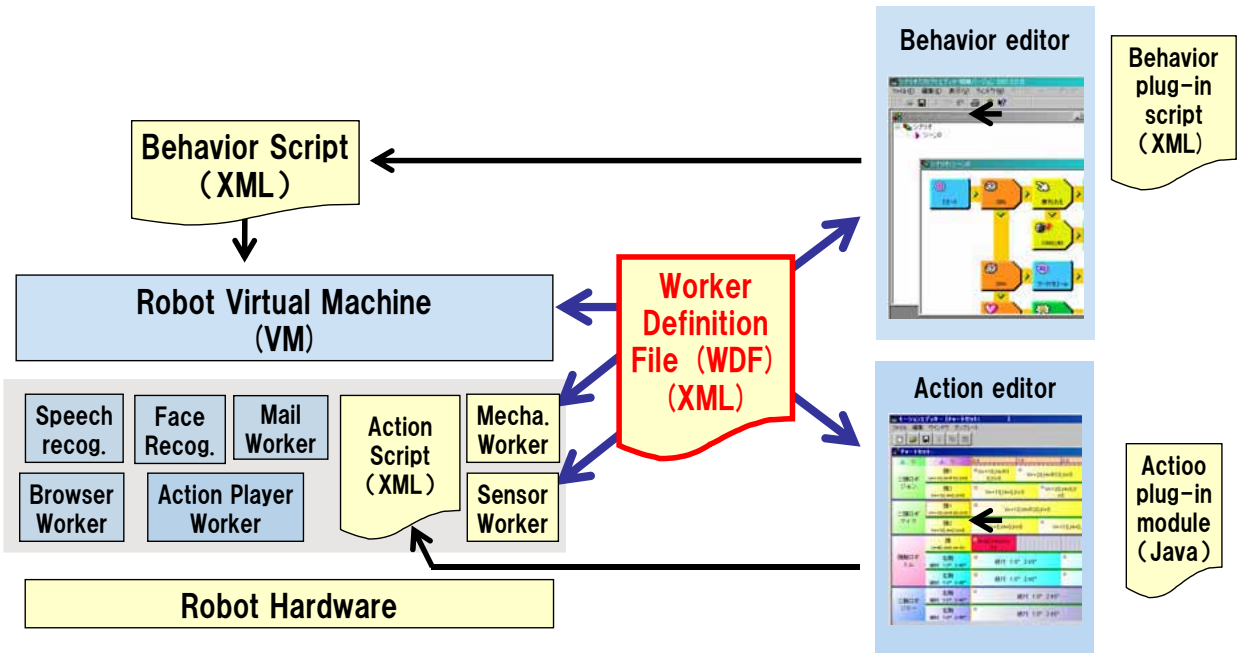
- Veruify plug-ins, Application Program and Execution based on WDF





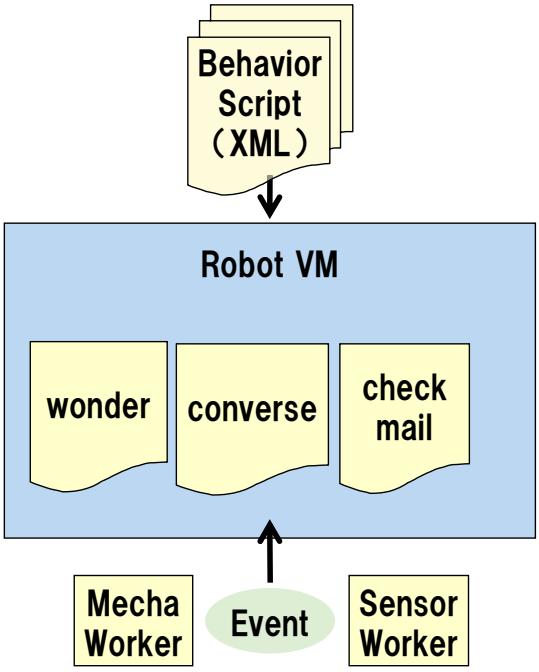



# Robot VM: Script Execution Mechanism



# Robot VM: multiple script execution

- Multiple Scripts can be executed at a time

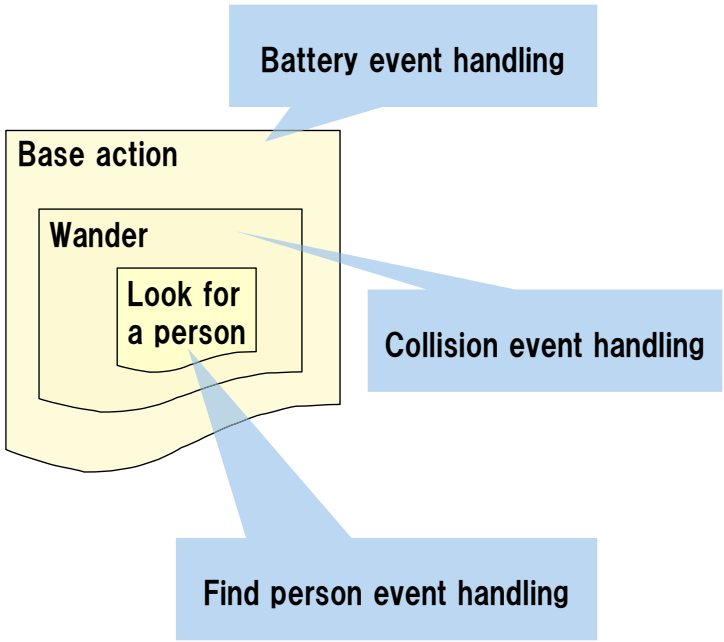






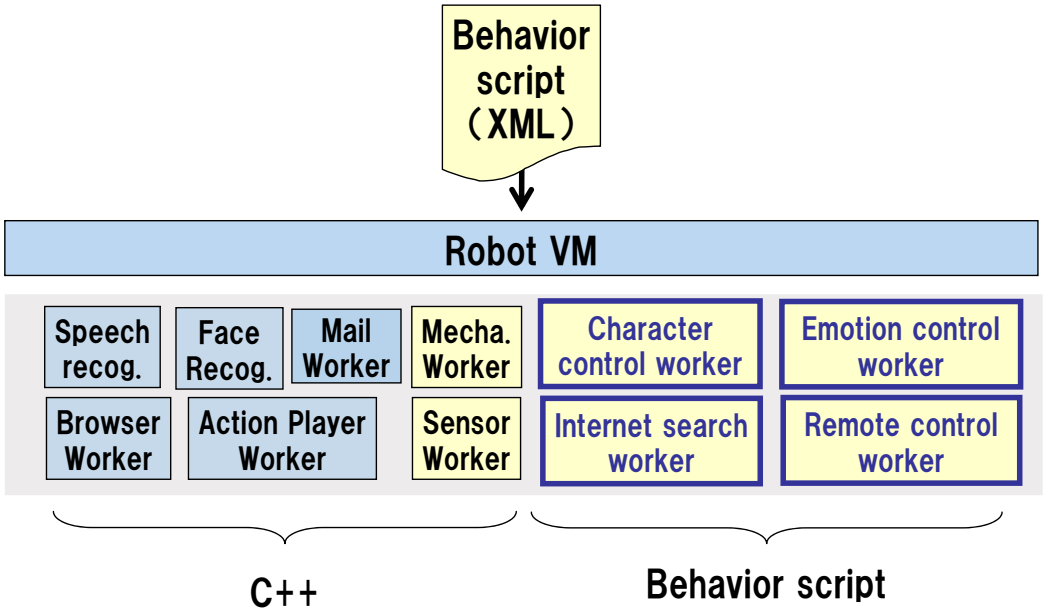

# Robot VM: Hieratical event handling

- Event handler in higher level script is used if reaction for a event is not defined in current script



# Robot VM: Script worker

- Behavior script can be defined as worker








# Robot VM: XML database

- To separate contents from behavior, speech data, parameters, behavior history etc. is stored as data as well as variables in behavior script

# The Player/Stage/Gazebo project: Open Source tools for robotics research

Brian P. Gerkey — SRI International AI Center

Richard T. Vaughan — Simon Fraser Autonomy Lab

Andrew Howard — Machine Vision Group

http://playerstage.sourceforge.net

Player

## Robotics research is hard

- Robots embody nearly all the problems of AI
  - perception, control, reasoning under uncertainty, planning, scheduling, coordination
- But we also get many of the problems of systems
  - real-time constraints, limited computation & memory, imperfect limited-bandwidth communication, distributed processing, physical dynamics

# Tedious aspects of robotics

- Wide variety of hardware, each robot a little different from the next
- Code must be ported from robot to robot
- Simple things, like visualizing the sensor state of the robot, require a lot of work
- Interface libraries (if you're lucky enough to have them) often restrict the choice of programming language and/or style
- Well-understood algorithms get re-implemented over and over and over and over…






# Can we make it easier?

- Learn from OS design:
  - Identify and abstract common components
    - keyboards, disk drives, displays
  - Define a common development environment
    - POSIX
  - Create standard tools
    - top, bash, ls, rm, X
  - Support any programming language
    - C, C++, Java, Python, Tcl, LISP…
  - Implement standard algorithms and data structures
    - qsort(), TCP, STL linked lists
  - Provide access at all levels
    - malloc()/free() vs. brk(), printf()/scanf() vs. read()/write()



Player

# What do we need?

- Good robotics infrastructure, just like we have good OS infrastructure
- Such infrastructure should:
  - be agnostic about programming language, compute platform, control and coordination architecture
  - be portable across different robot hardware
  - implement standard algorithms
  - include development tools
  - support code re-use
  - be *flexible*
  - plausibly simulate a wide variety of robot systems
  - be extensible
- It should also be Open Source, aka Free (free as in speech and free as in beer)



Player

# Improving research practice

- Little shared equipment, no shared data*, no shared environments, few shared tasks, little shared code
  - Huge duplication of engineering effort
  - Systems are not directly comparable
    - Trial by video
- Interaction between researchers is papers and meetings
- Recently challenges (e.g., RoboCup, DARPA Grand Challenge) have stimulated and guided research and boosted education, but:
  - Competitive overhead is HUGE, as is the tendency to overfit
- Goal: accelerate development by improving interaction between researchers via good infrastructure
  - Sharing the engineering burden
  - A means for comparing systems

*Except for radish: http://radish.sourceforge.net



Player

# The Player/Stage/Gazebo project

- A collaborative development effort aimed at producing high-quality Open Source tools for robotics researchers
- The project primarily maintains three pieces of software, all released under the GNU GPL:
  - Player: networked robot device interface
  - Stage: scalable 2-D multiple robot simulator
  - Gazebo: high-fidelity 3-D multiple robot simulator




Player

# Overview of the P/S/G Project

- 2000: Player/Stage project originated at USC
  - Brian Gerkey, Richard Vaughan, Andrew Howard
- 2001: Project moves to SourceForge.net (neutral territory)
- 2003: Gazebo (3D simulator) released
  - Nate Koenig, Andrew Howard
- 2002-2003: Support from DARPA/IPTO
- 2005 (ongoing): Support from SRI AI Center
- Current active developers @
  - SRI, USC, Stanford, Simon Fraser Univ., JPL, BBN, UMass, UPenn, WUSTL, Univ. Sherbrooke, Univ. Sydney, Simon Fraser Univ., DRDC, Univ. Auckland…
- Large user community (>40,000 downloads) around the world in academic, industrial, and government labs, as well as classrooms
- Free software (speech and beer)



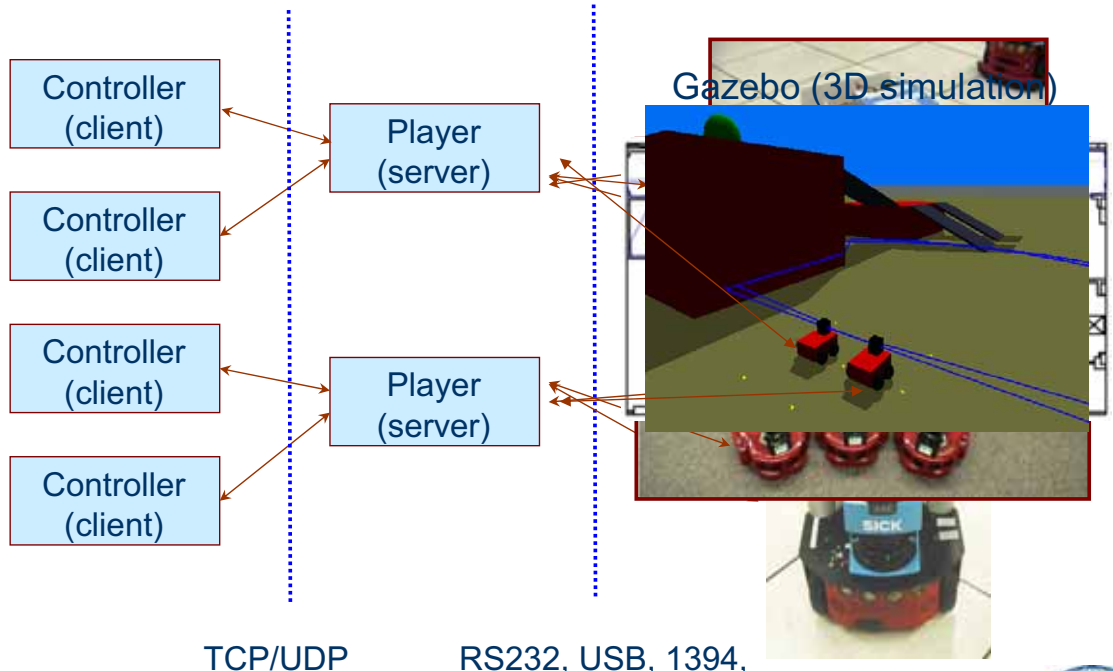
# Design decisions

- How do you interface with a physical robot?
  - direct link vs. network / IPC
- How do you interface with a simulated robot?
  - simulation-specific vs. same as physical
- How do you interface with different robots?
  - robot-specific vs. robot-independent
- How is a “robot” represented?
  - one entity vs. a collection of devices
- How is the system structured?
  - microkernel (e.g., Mach) vs. monolithic kernel (e.g., Linux)
- How is new functionality added?
  - static link vs. dynamically-loaded plugin

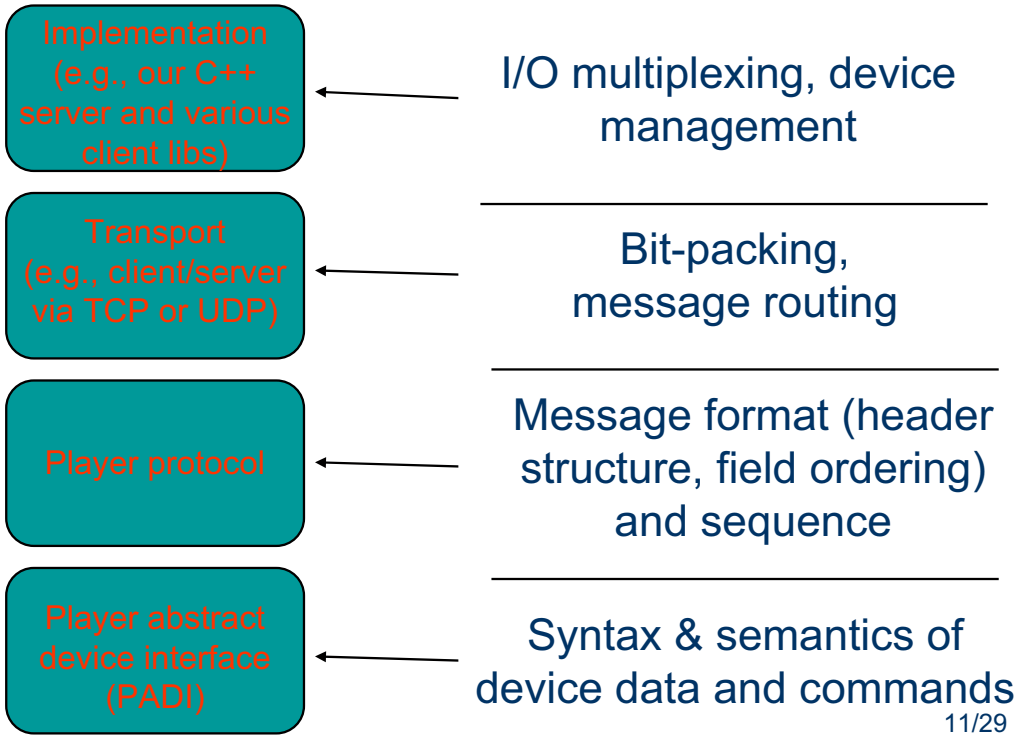




# Architectural overview



# Player abstractions








# PADI

- Device model inspired by Unix (file-like semantics); a device is:
  - source of data (read)
  - sink for commands (write)
  - source/sink for configuration (ioctl)
- Device model uses familiar OS abstraction:
  - Device = interface + driver (+ index)
  - Interface: generic API
  - Driver: hardware/software specific
  - Multiple drivers can support the same interface
  - Two drivers that support the same interface appear (almost) identical to the client

# PADI examples

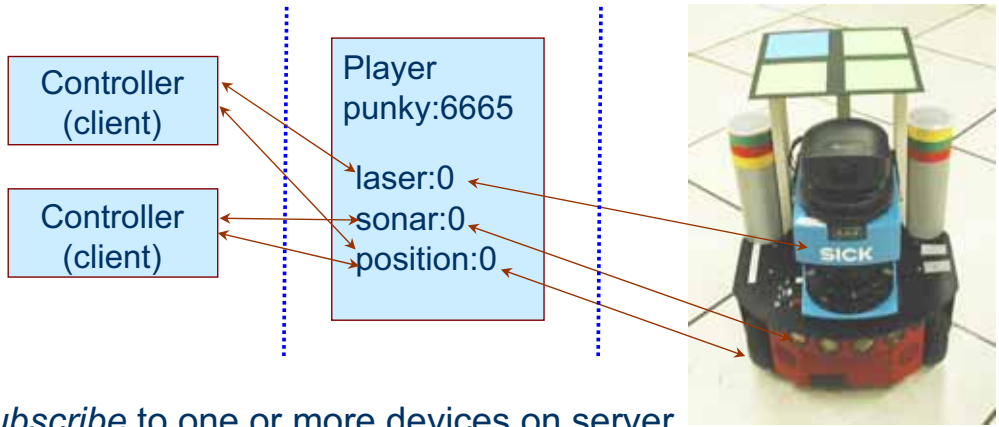
- The *position2d* interface*:*
  - data: robot position, velocity
  - command: desired position and/or velocity
  - configurations include: enable/disable motors
  - Supporting drivers:
    - p2os_position, rwi_position, gz_position, ...
- The *laser* interface:
  - data: ranges, intensities
  - command: none
  - configurations include: get/set angular/range resolution
  - Supporting drivers:
    - sicklms200, hokuyo, rwi_laser, gz_laser, ...





# Player's client/server model

- Player is a *networked device server;* control programs are *clients*
- Clients can use any control architecture and can be written in any programming language that can control a socket (that's pretty much *any* language)
- *Client libraries* help in writing clients




- Clients *subscribe* to one or more devices on server
- Multiple clients may subscribe to the same device

# Client libraries

- A *client library* facilitates the development of control programs
- Could be simple (e.g., just implement the Player transport) or sophisticated (e.g., post-process data somehow)
- Should provide a language-appropriate API
- Currently available for: C, C++, Python, LISP, Tcl, Ruby, Java, and Octave

```
#include <playerclient.h>
#include <stdlib.h> /* for exit() */

int main(int argc, char *argv[])
{
  PlayerClient robot("localhost");
  SonarProxy sp(&robot,0,'r');
  PositionProxy pp(&robot,0,'w');

  int newturnrate,newspeed;

  for(int i=0;i<1000;i++)
  {
    if(robot.Read())
      exit(1);

    // print out sonars for fun
    sp.Print();

    // do simple collision avoidance
    if((sp.ranges[0] + sp.ranges[1]) <
       (sp.ranges[6] + sp.ranges[7]))
      newturnrate = -20; // turn CW 20 degrees per second
    else
      newturnrate = 20; // turn CCW 20 degrees per second

    if(sp.ranges[3] < 500)
      newspeed = 0;  // obstacle in front, so stop
    else
      newspeed = 100; // no obstacle in front, move 100 mm/s

    // command the motors
    pp.SetSpeed(newspeed,newturnrate);
  }
}
```





# Some supported hardware/software

- **Robots**: ActivMedia, RWI, K-Team, Segway, Evolution
- **Lasers**: SICK, Hokuyo
- **Vision**: ACTS, CMVision, CMUCam
- **Pan-tilt(-zoom)**: Sony, Canon, Amtec, Direct Perceptions
- **IMUs**: Microstrain, ISense
- **WiFi**: linuxwifi, iwspy, aodv
- **GPS**: any NMEA-compliant (e.g., handheld) unit
- **Speech**: Festival

Player

# Abstract drivers

- Well-understood algorithms can be encapsulated in Player and offered as standard services for all to use
- Player's driver API provides a common implementation framework
- Player becomes a **community code repository** for useful algorithms
- Opportunity for *real* code reuse in robotics



Player

# VFH on 3 very different robots

QuickTime™ and a
Cinepak decompressor
are needed to see this picture.

QuickTime™ and a
Cinepak decompressor
are needed to see this picture.

Sped up ~3x

Real time (1.5mps !)

Sped up 5x

QuickTime™ and a
Cinepak decompressor
are needed to see this picture.



Player

# The simulators

- Stage
  - 2-D
  - sensor-based
  - Player interface
  - kinematic; good for slow, statically stable, indoor robots
  - algorithms: O(1)-O(n)
  - large teams (100s)
- Gazebo
  - 3-D
  - sensor-based
  - Player interface
  - dynamic; good for fast, dynamically stable, outdoor robots
  - algorithms: O(n)-O(n^3)
  - small teams (10s)

QuickTime™ and a YUV420 codec decompressor are needed to see this picture.

QuickTime™ and a Microsoft Video 1 decompressor are needed to see this picture.



Player

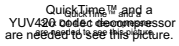
# Gazebo example: Segway RMP

QuickTime™ and a
Cinepak decompressor
are needed to see this picture.



# Tools

- **playerv**: sensor visualization
- **playernav**: localization / navigation control panel (OCU)
- **playerjoy**: joystick / keyboard teleoperation
- **playermap**: scan-matching mapmaker
- **dgps_server**: differential GPS correction server








# Portability

- Player is developed primarily on x86/Linux, but builds and runs on nearly* any POSIX platform, whether conventional or embedded, e.g.,
  - ARM/Linux (iPAQ, nanoEngine, XScale/Stayton)
  - PPC/Linux (ipEngine)
  - PPC/Darwin (OS X)
  - Sparc/Solaris
  - x86/Cygwin
- Stage and Gazebo additionally require some other Open Source packages (e.g., GTK+, ODE), and are known to build and run on:
  - x86/Linux
  - PPC/Darwin (OS X)

*Work is ongoing for native Windows builds



# Community interest

- RETF (was) using the Player protocol as a starting point for a standard in robot device control
- Player interface to the Webots simulator (by Cyberbotics, Inc.) under development
- Stage used as a testbed for NASA/JPL's MDS

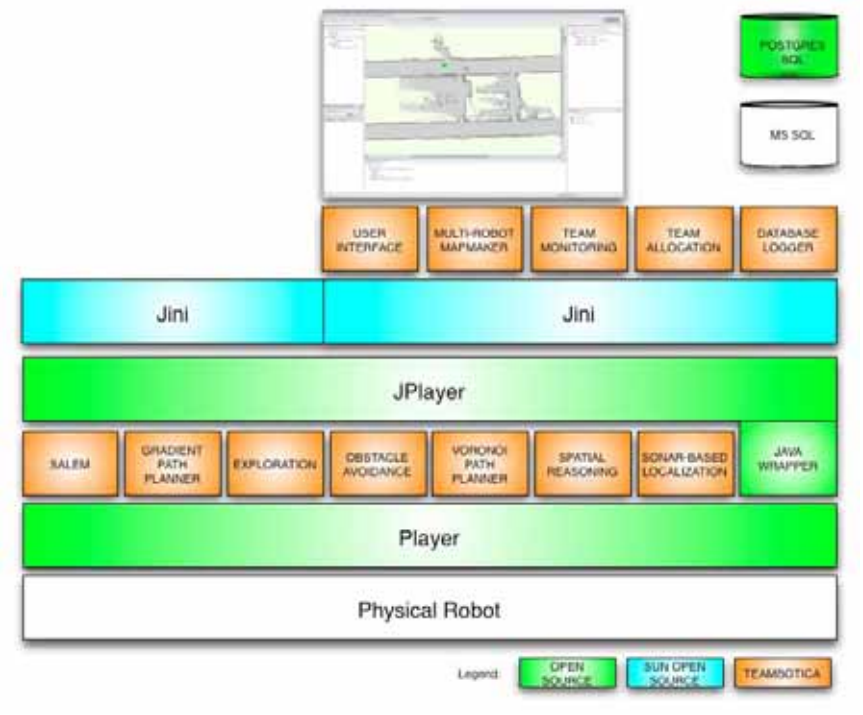



Player

# Player 2.0 (coming soon)

- Message-passing model (instead of state reflection)
- Standardized units (MKS)
- Normalized interfaces
- Library division (transport independence!):
  - libplayercore: messaging basics
  - libplayerdrivers: common drivers
  - libplayerxdr: XDR data marshaling
  - libplayertcp: TCP client/server transport
- JPlayer: Java/JINI wrappers for Player libraries

# Example: SRI Teambotica






Player

# The beauty of Open Source

- More people will try your code if it's Open (even if they have no intention of hacking on it)
- Many (most?) people who hack on your code will send you patches
  - And you (eventually) learn to deal with the clueless users…
- More people will contribute if they feel some ownership over the result: hence the neutral territory of SourceForge.net

# The beauty of Open Source

- If you build it, they will come: Player has become a common environment for robotic systems development and integration
- Player currently contains about 90 drivers, including some for hardware that I've never seen, much less used
- The developer / user community (sometimes) supports itself
  - They answer each others' questions
  - A user in Pennsylvania tests a new driver written by a developer in New Zealand
- It's even possible to get government funding for Open Source!
- And you can protect your IP by distributing binary-only Player plugins (like Linux kernel modules)






# Summary

- Together, Player, Stage, and Gazebo form the infrastructure necessary for any mobile robotics research program
- Player imposes as few constraints as possible on the researcher (use any architecture, any language)
- Player allows code reuse through its driver and device APIs, and its client/server protocol (all fully documented)
- Stage and Gazebo allow for simulation of a wide variety of robot platforms, both real and imagined
- Following the Open Source model, all code is freely available online, and everyone is encouraged to download, use, and modify it (please send us your patches!)



# Resources

- Player/Stage project website:
  - http://playerstage.sourceforge.net
  - (or just Google for “player stage”)



# Network Robot Platform for Information Sharing

Ken-ichiro Shimokura
*NTT Cyber Solutions Laboratories, Japan*

NTT

## Table of Contents

# Network Robots Research Project

**Period** 2004 - 2008

**Funding from** Ministry of Internal Affairs and Communications, Japan

**Project Partners**

- ATR
- NTT
- Toshiba
- Mitsubishi Heavy Industries, Ltd
- Matsushita Electric Industrial Company

# Network Robots

NTT

Ubiquitous Network

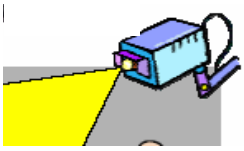
**"Visible" type**

Apri-alpha Robovie

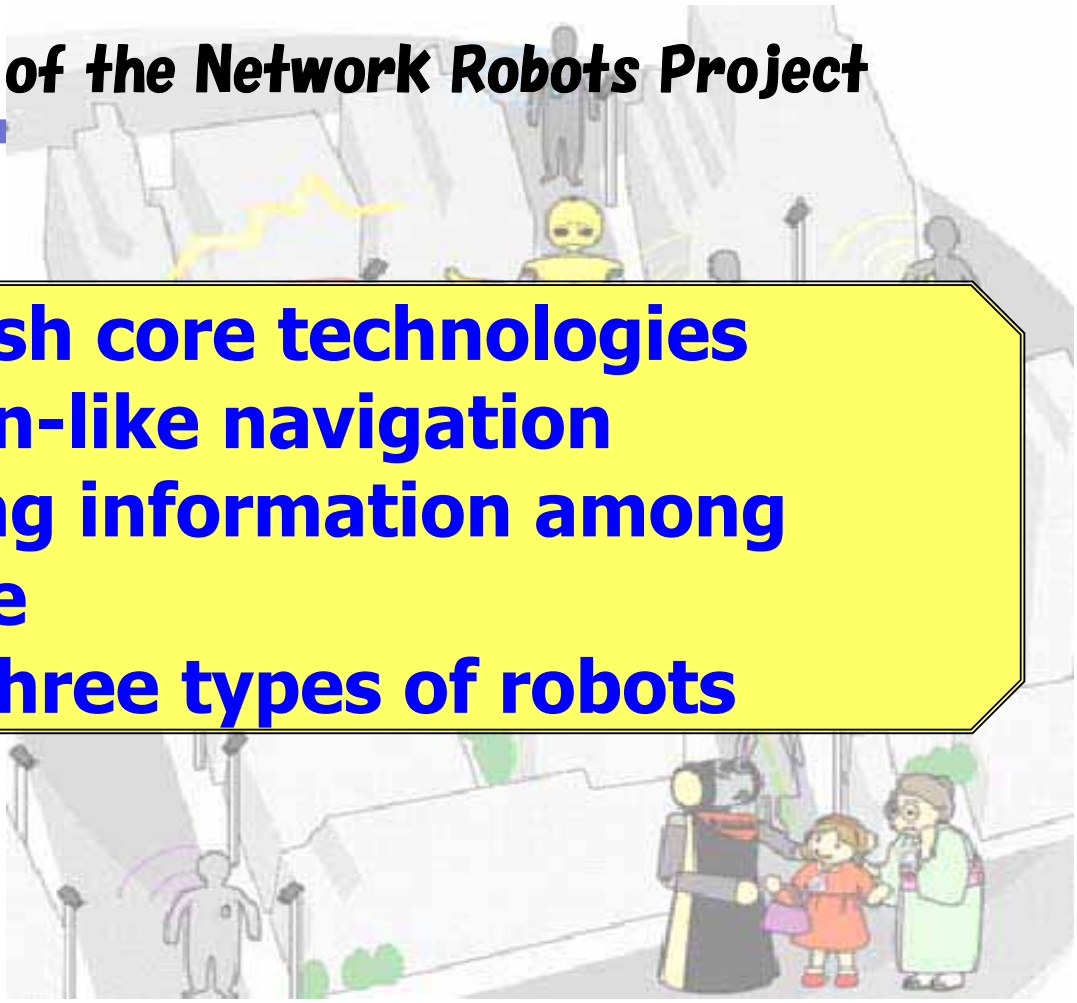
## Network Robots

**"Virtual" type**

**"Unconscious" type**



# Goals of the Network Robots Project



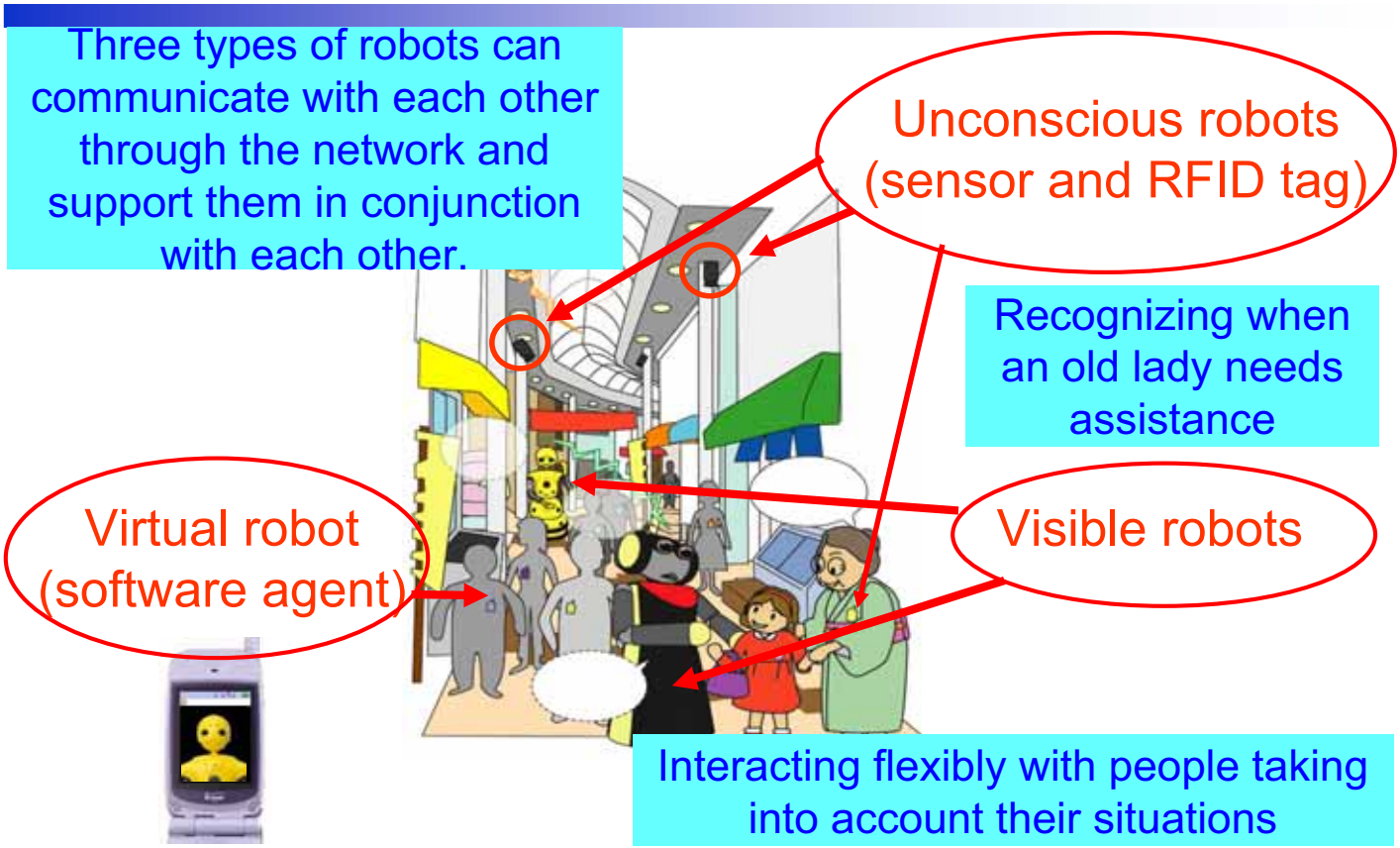
**Establish core technologies**
- **human-like navigation**
- **sharing information among people**

**using three types of robots**

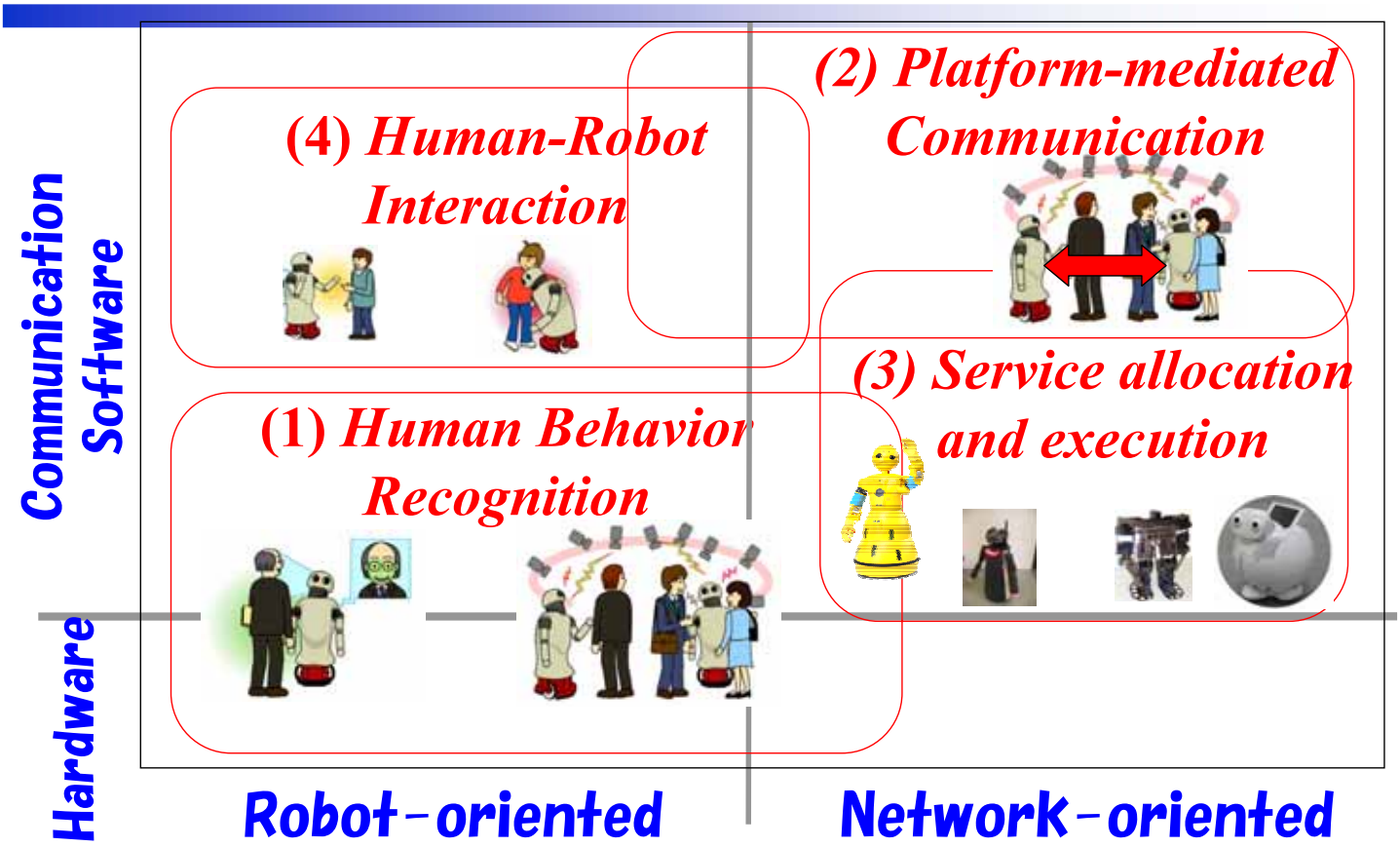
# Future Network-Robot Environment

NTT

Three types of robots can communicate with each other through the network and support them in conjunction with each other.



# Research Issues in the Network Robots Project

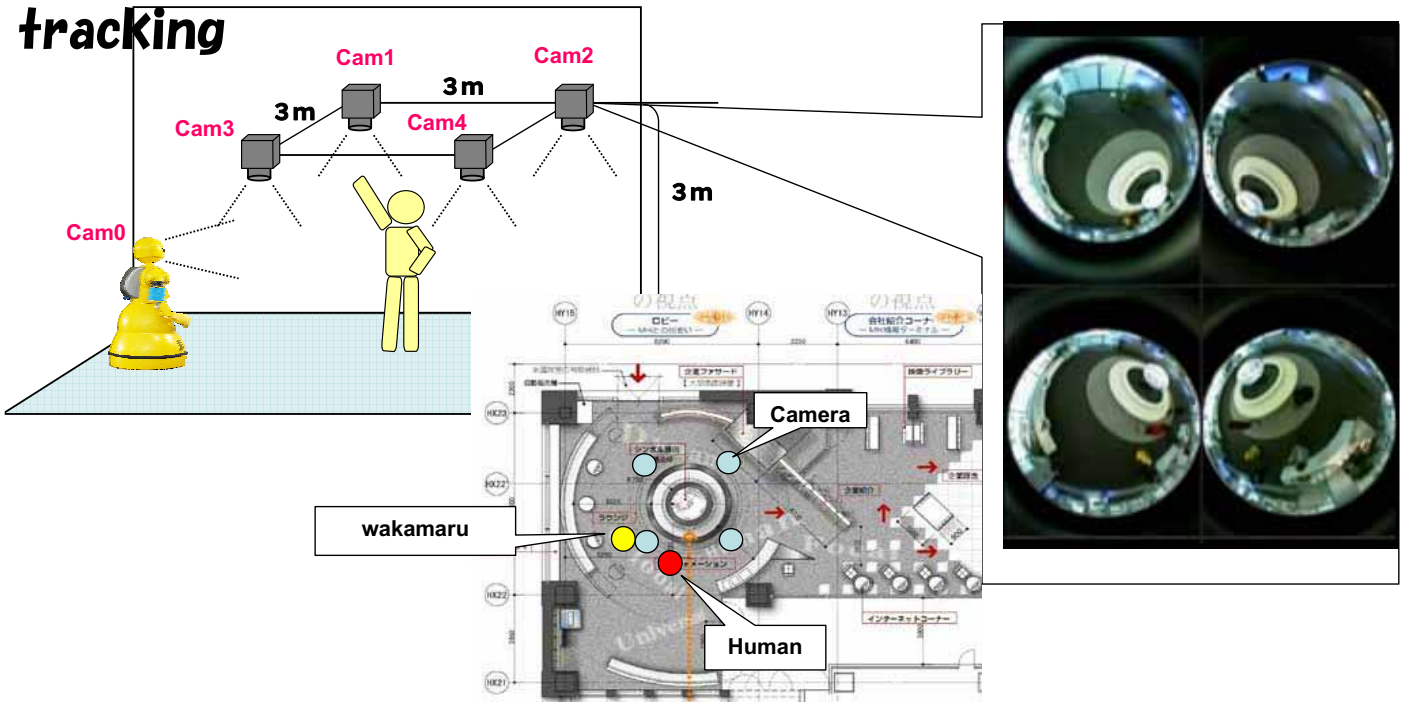



1. Schematic of Network-robot Project
2. Human Behavior Recognition
3. Platform-mediated Communication
4. Service allocation and execution
5. Human-robot Interaction
6. Future Work

# Logical sensing example:MHI

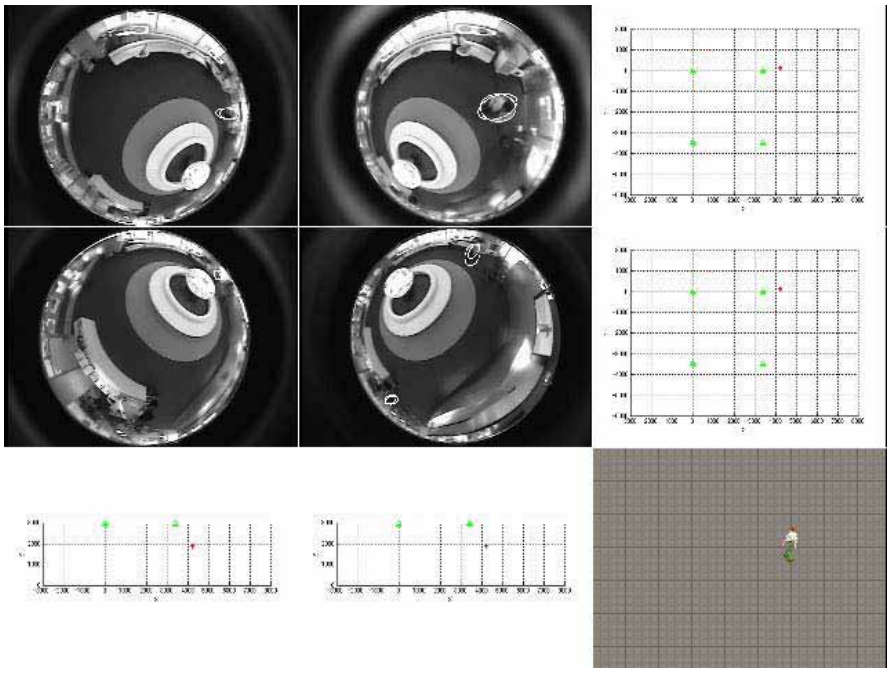


## Human position and behavior estimation by using unconscious robots with multiple viewpoint image tracking

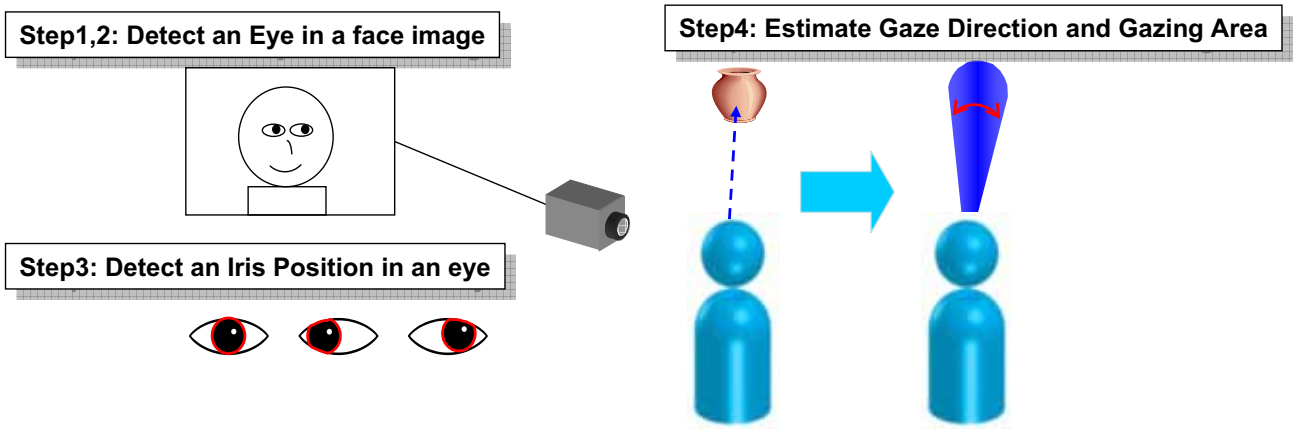



Logical sensing example:MHI

52



# Logical sensing example: Panasonic




**Estimating Gazing Area from a Face Image**

Gazing area is estimated as follows:

1) Detecting faces from images captured by cameras embedded in the environment and by camera-equipped visible robots.
2) Selecting a full face image shot from the front of the person.
3) Localizing an eye and iris in the full face image.
4) Estimating Gaze direction from the position and the shape of iris.

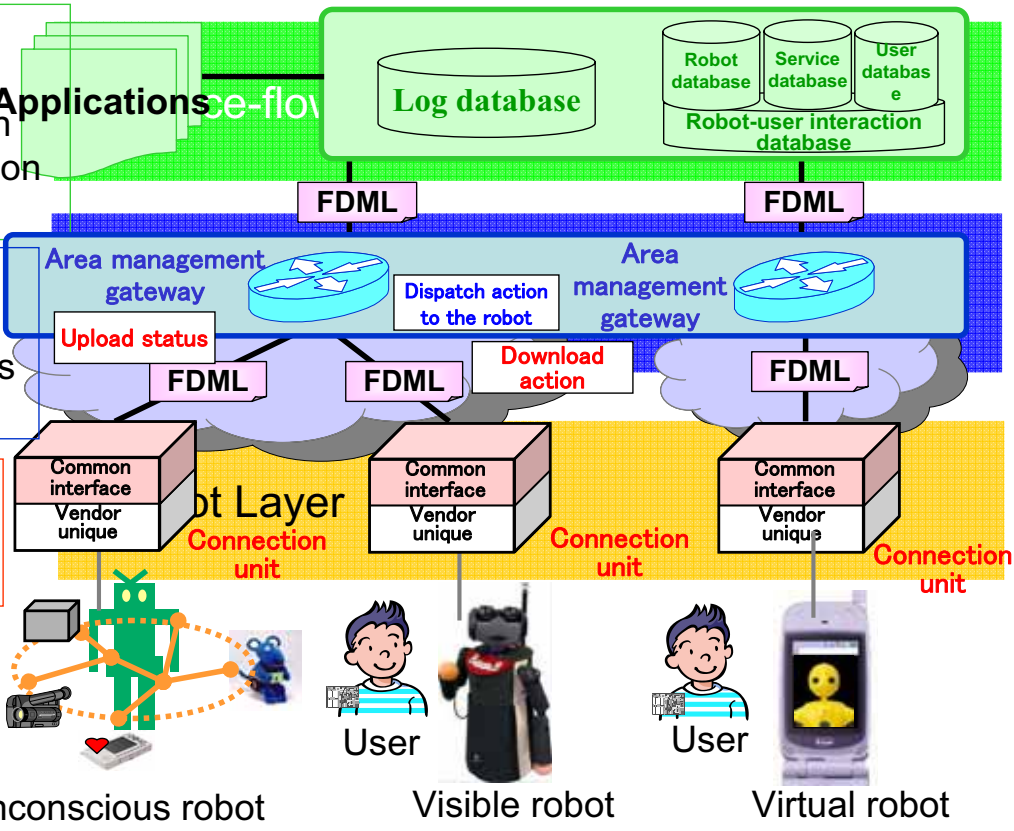
1. Schematic of Network-robot Project
2. Human Behavior Recognition
3. **Platform-mediated Communication**
4. Service allocation and execution
5. Human-robot Interaction
6. Future Work

# Schematic of Network Robot Platform

**Robot-user interaction database**
- Reflection of information about robots and users on available services

**Area management gateway**
- Management of services and robots in an area

**Connection unit**
- Interface between real world and platform




# Implementation of Connection Unit

Robot-related information is converted into an expression of status with a unified format

*Real world*

Product-dependent part

Product-independent part

*Network robot platform*

Visible robot

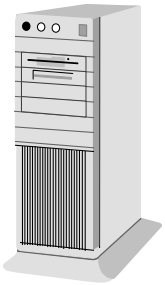
Adapter

Translation to FDML

Command translation

FDML

Command



Commands with a unified format are converted to equipment-specific commands

# Information Abstraction with FDML

- Robot-related information can be abstracted to 4 W's (who, what, when, where) for the purpose of easily handling the robot and user information in the network robot platform
- FDML (Field Data Markup Language)

  An XML format for sharing information between user and robot

FDML's main features

1. Robot-related information is described in a unified format, which makes the information type-independent.
2. Time-stamp and robot-related information are connected in the tag.
3. Simple tag definitions and a small-size memory enable high-speed parsing.

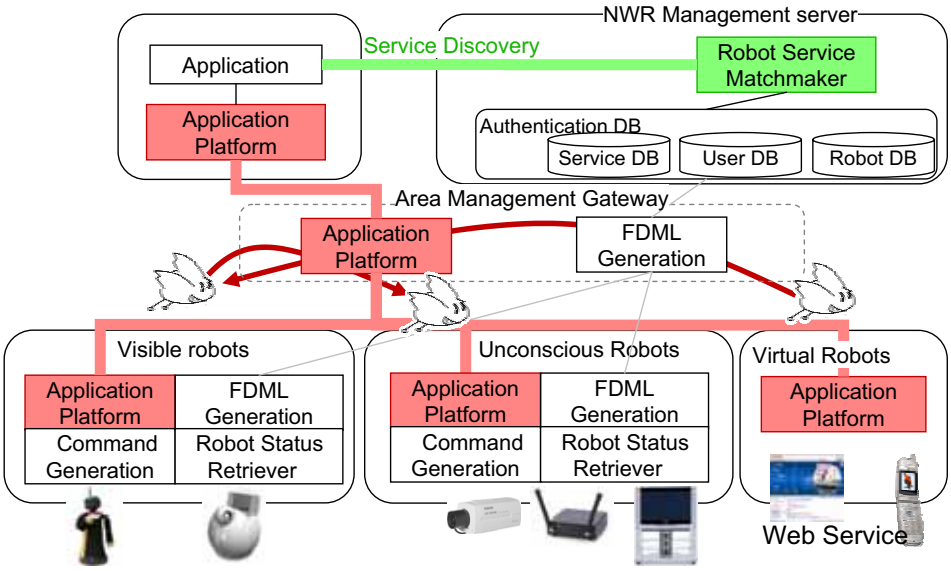
```
<?xml version="1.0">
<FDML version="1.0.1">
 <Info>
  .....
<Definition>
   .....
<Data>
          4W
</FDML>
```

*FDML tag structure*

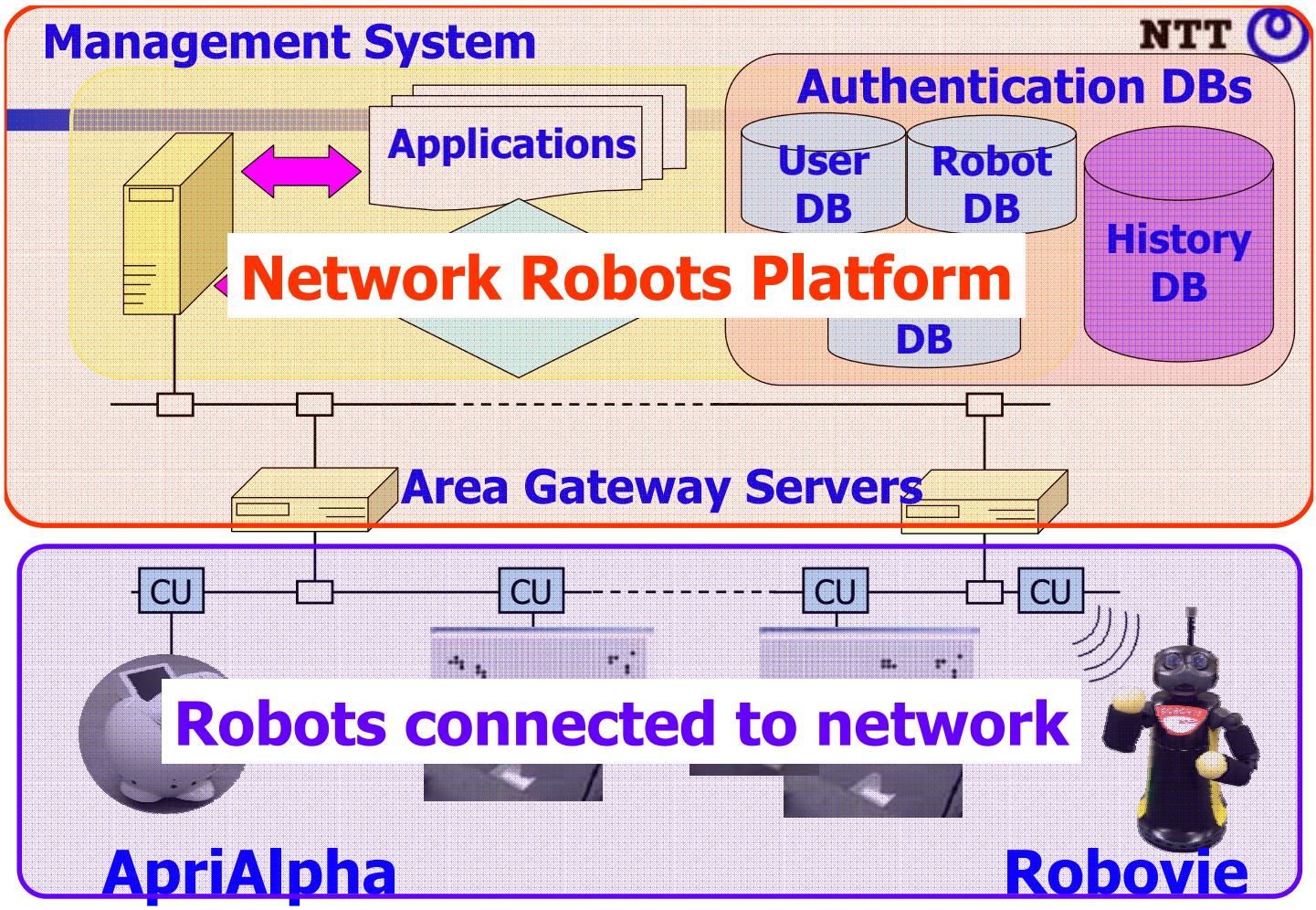
1. Schematic of Network-robot Project
2. Human Behavior Recognition
3. Platform-mediated Communication
4. **Service allocation and execution**
5. Human-robot Interaction
6. Future Work

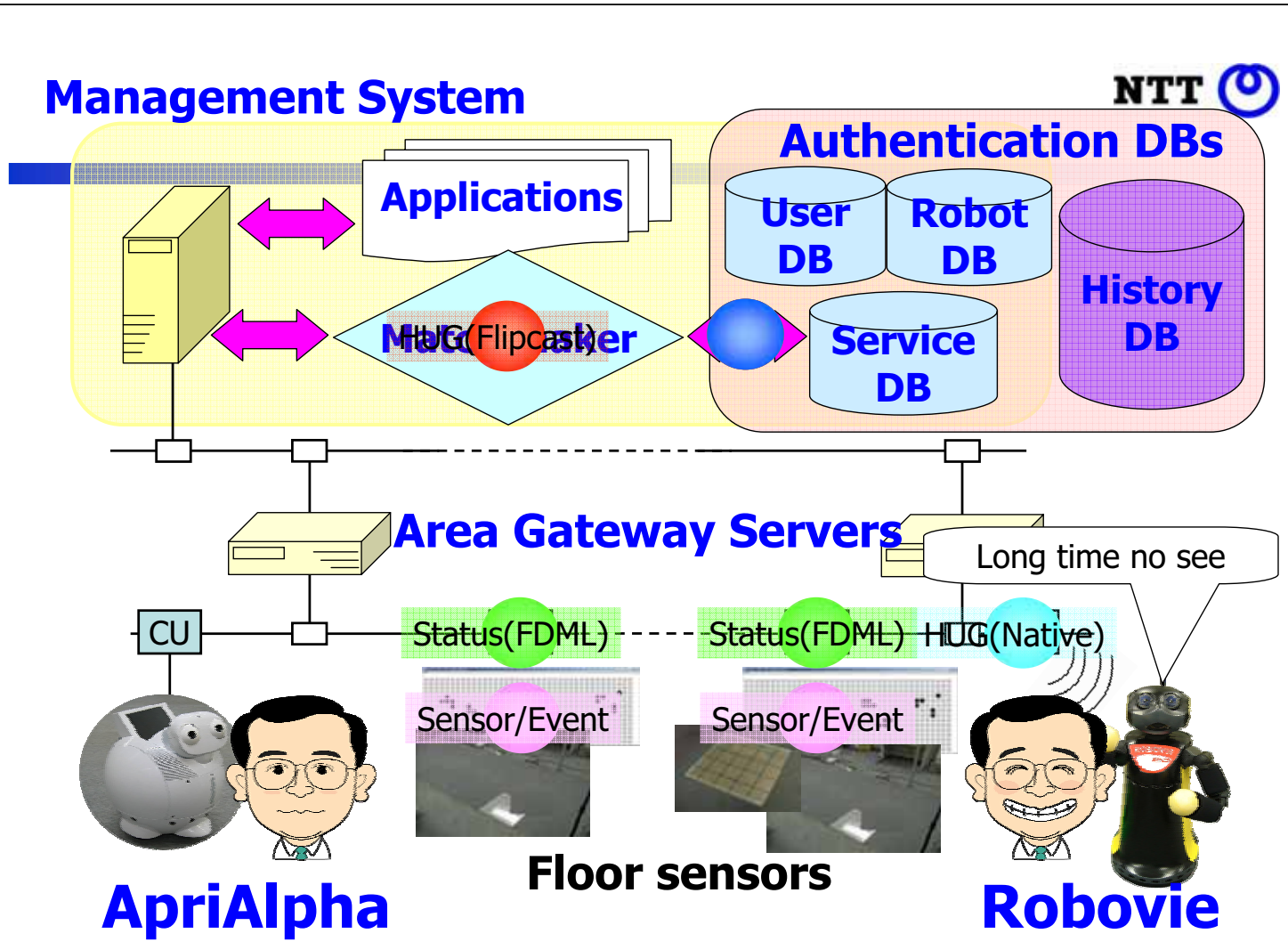
# Service allocation and script-based execution: Toshiba

- **Script**: A script language and its runtime environment
- **Mobility**: It jumps from one device to another at runtime
- **Interface Abstraction**: API level loose coupling interface to devices
- **Small**: It runs even on a cellular phone's Java environment




# Connection Experiment@Ubicomp


Management System
NTT
Authentication DBs
Applications
User DB
Robot DB
History DB
Network Robots Platform
DB
Area Gateway Servers
CU
CU
CU
CU
Robots connected to network
AprilAlpha
Robovie


Management System
NTT
Authentication DBs
Applications
User DB
Robot DB
History DB
Service DB
Area Gateway Servers
Long time no see
CU
Status(FDML)
Status(FDML)
HUG(Native)
Sensor/Event
Sensor/Event
Floor sensors
AprilAlpha
Robovie

NTT

1. Schematic of Network-robot Project
2. Human Behavior Recognition
3. Platform-mediated Communication
4. Service allocation and execution
5. Human-robot Interaction
6. Future Work

NTT

"Human-robot Interaction in Network Robots" by Dr. Norihiro Hagita

1. Schematic of Network-robot Project
2. Human Behavior Recognition
3. Platform-mediated Communication
4. Inter-Robot Communication
5. Human-robot Interaction
6. Future Work

## Future Work

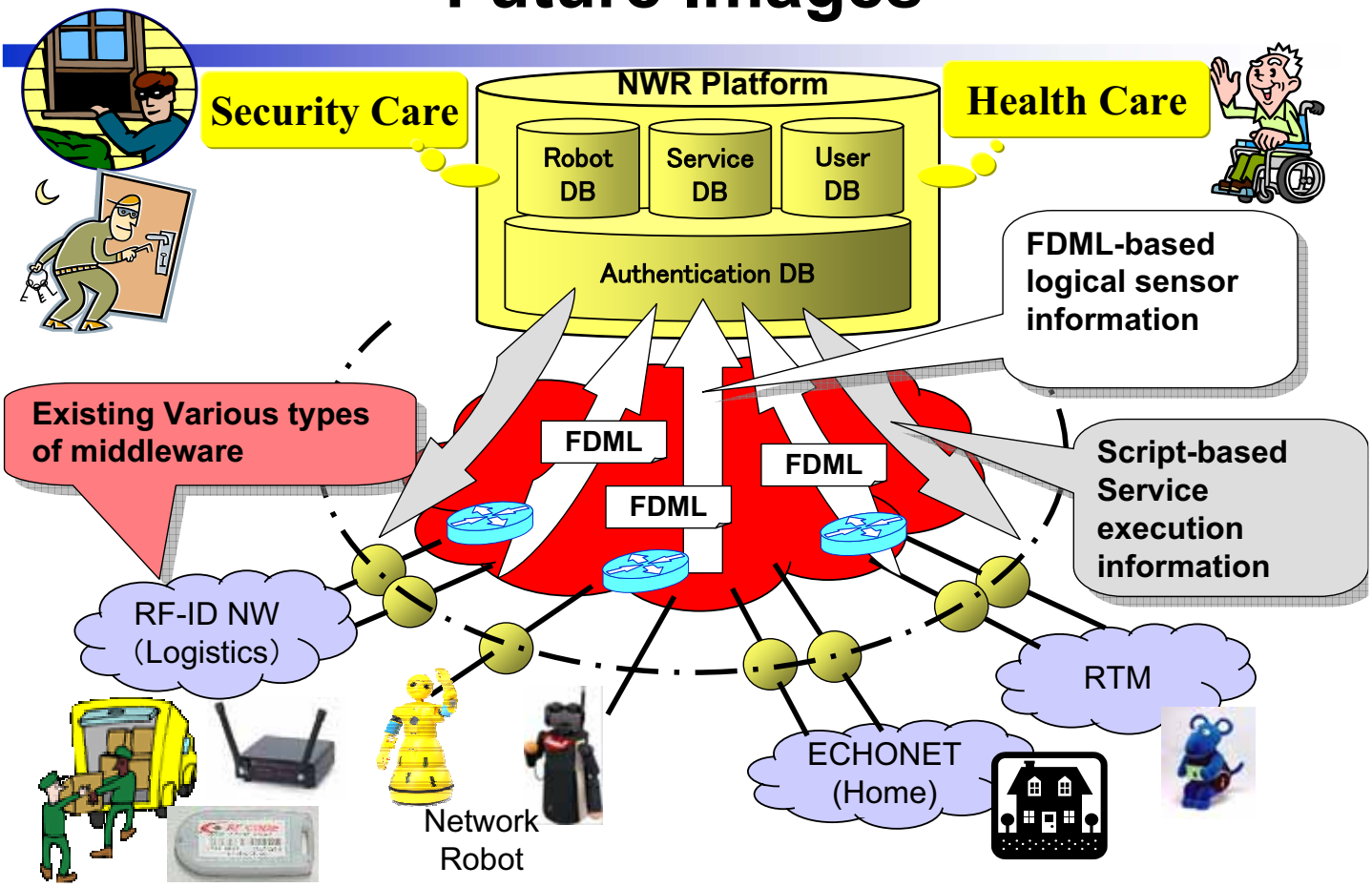
**Increase**

- Variation of Logical Sensors.
- Number of robots Connected to Platform.
  - Robots through standardized middle ware (from the standardization viewpoint)
- Response time in the Platform

**Establish**

- Visible-virtual and Unconscious-virtual robots collaboration

# Future Images



# Human Robot Interaction in Network Robots

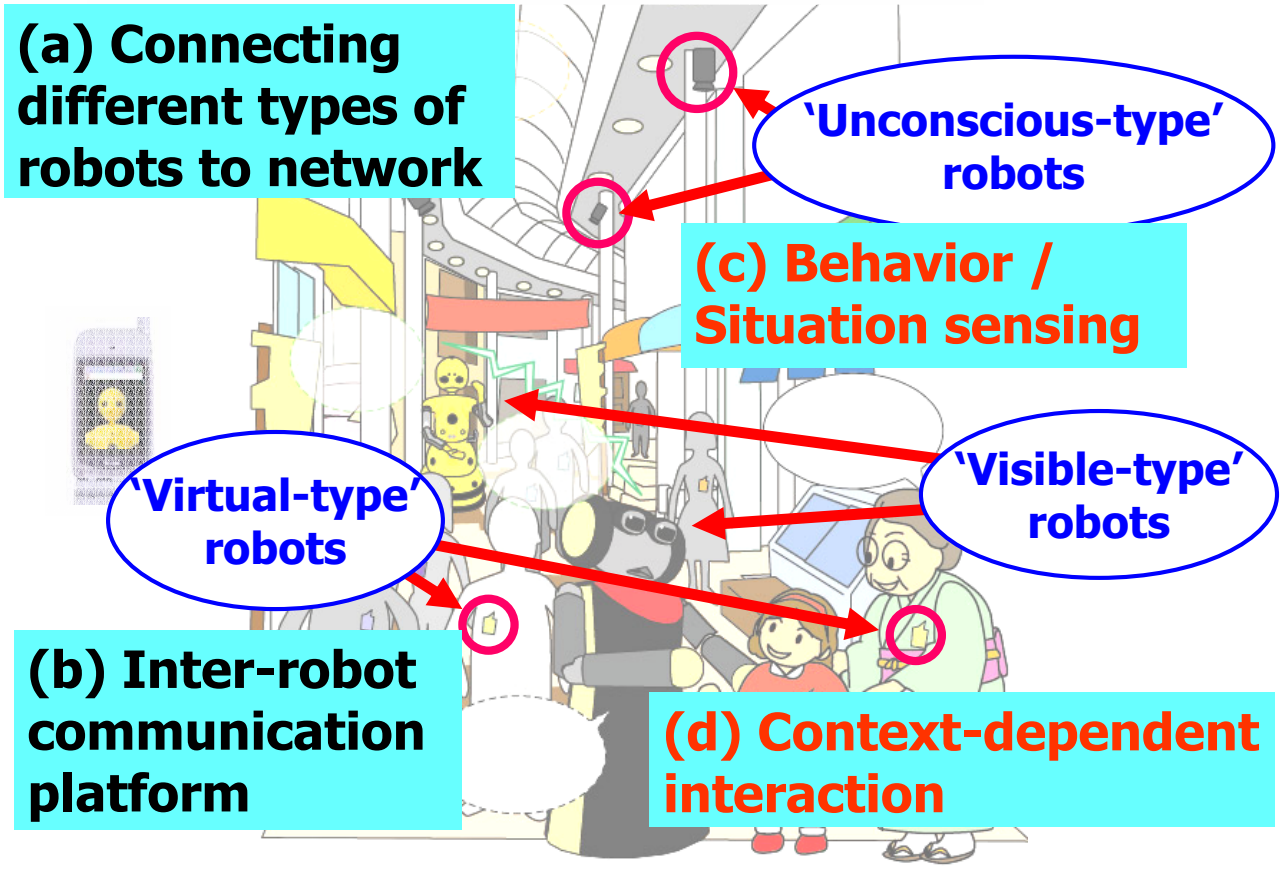
December 7, 2005

ATR Intelligent Robotics & Communication Labs

Norihiro Hagita



# Research Issues



# Robots

## Physical existence media of communication

## As another partner



## Social Interaction with or via Robots

**Elementary School** **Science Museum**

# Research Issues

1. Allow human robot communication
   individual communication skills (perception, human-like behaviors, intelligence, etc.)
2. Allow communication with other robots, ubiquitous sensors, and PCs
   *More than two 'robots' are better than one.*
   obtaining missing/additional information from the Internet & environment.
3. Analyze whether humans could accept them
   Field experiments in busy streets, schools, etc.



# Far from standardization?

1. Behavior languages for human robot interaction
   → basic behavior languages for different-type robots
2. Social intelligence based on network robots
   → software modules for social intelligence
3. Ubiquitous experience media based on communication robots and ubiquitous sensors
   → interaction primitives and corpus from multiple sensor data

# 1. Behavior languages for human robot interaction



# Robovie II: Hardware


Height 114cm

Weight 39kg without battery

**CPU :** Pentinum 2, Linux, wireless LAN

**Moving:**

- 3-joint head, two 4-joint arms
- 3-wheeled pedestal

**Seeing :**

- Omni-directional camera
- Stereo cameras

**Touching:**

-Tactile sensors
-Ultrasonic distance sensors

**Hearing & talking**

- Microphone, speaker

# Robovie II: Software

**Hearing & Talking:**

- Speech recognition for 300 words (Japanese) and 50 ones (English)
- Speech synthesis for 300 sentences

**Seeing:**

- Human and object recognition
- Eye contact with a specific person

**Behavior:**

- 100 behaviors
- 700 Situated modules

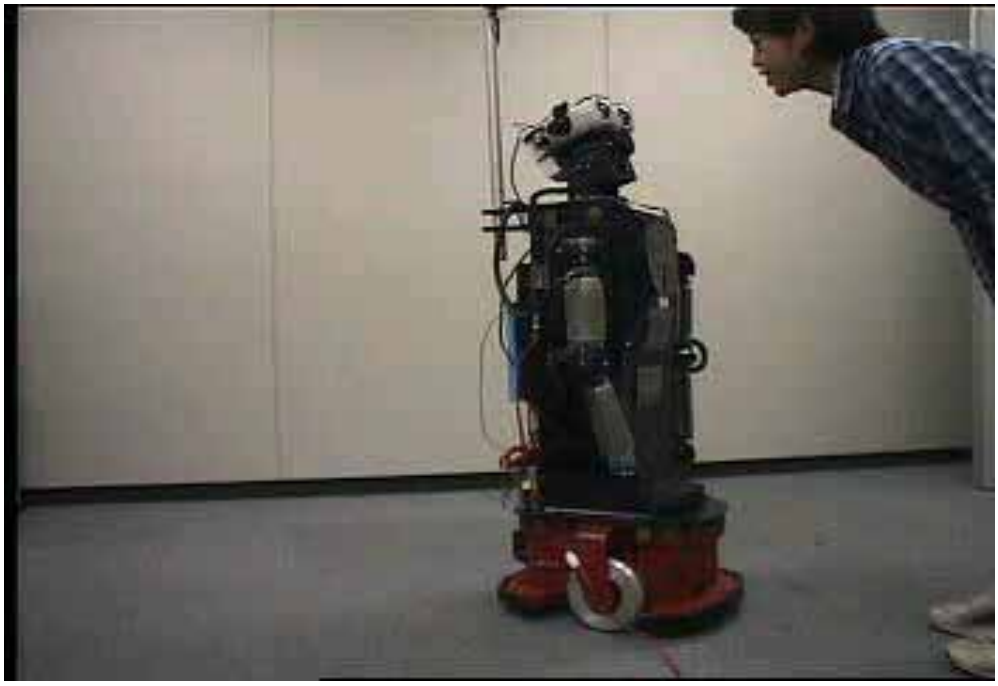


# Human communication

**A chain of short-time interactions:**

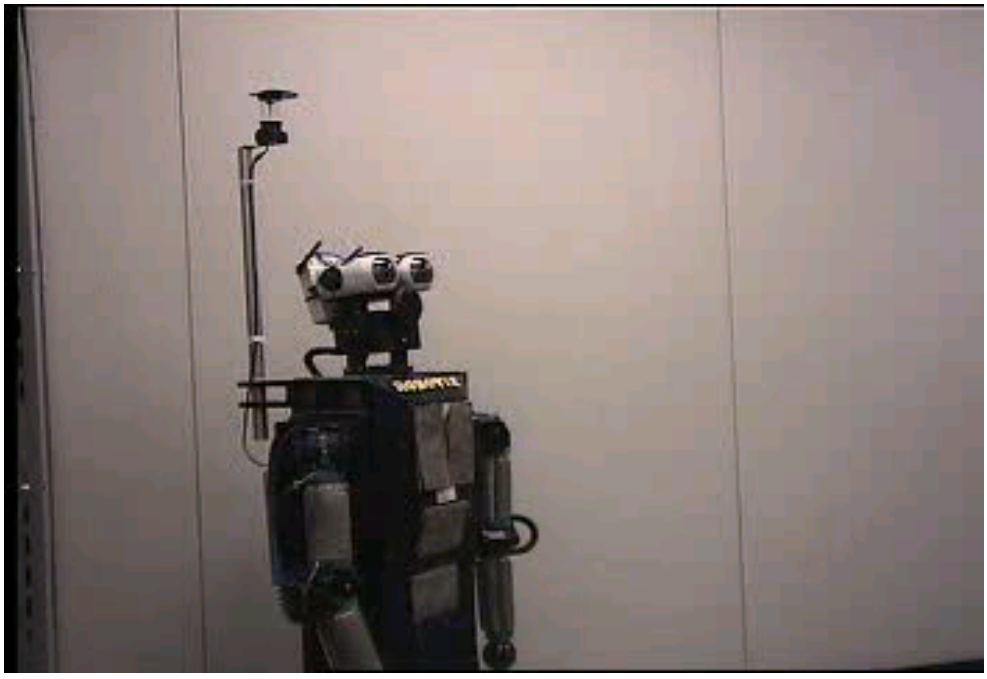
- Introduction 起
- Development 承
- Turn 転
- Conclusion 結

# ‘Hello’

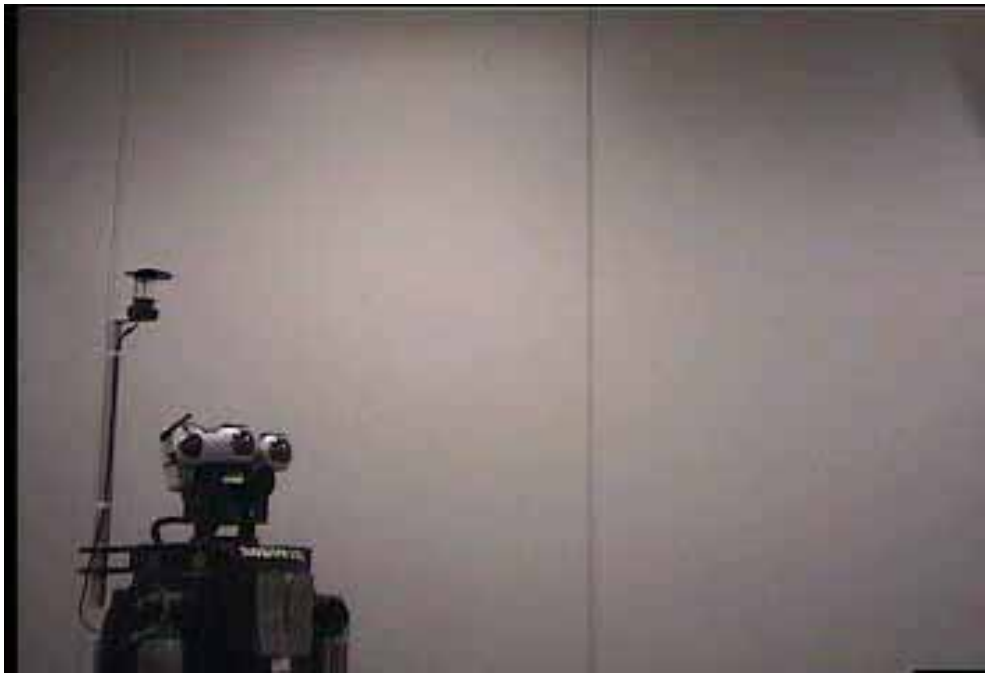




# ‘Shake your hand’



# Eye-contact





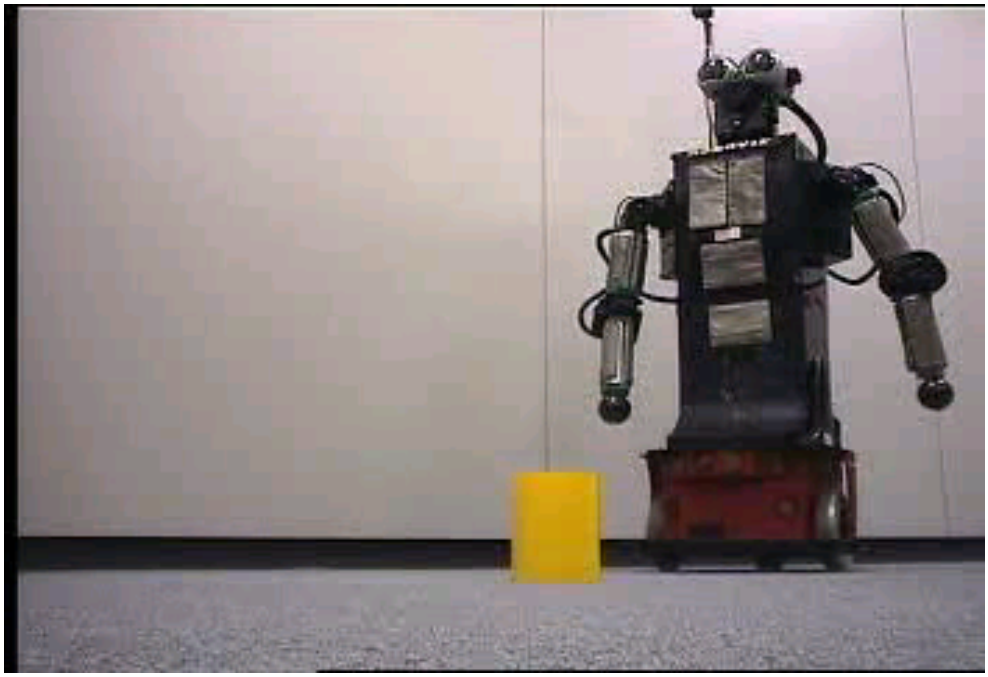
# 'Paper-rock-scissors'

# ‘Please hug me’



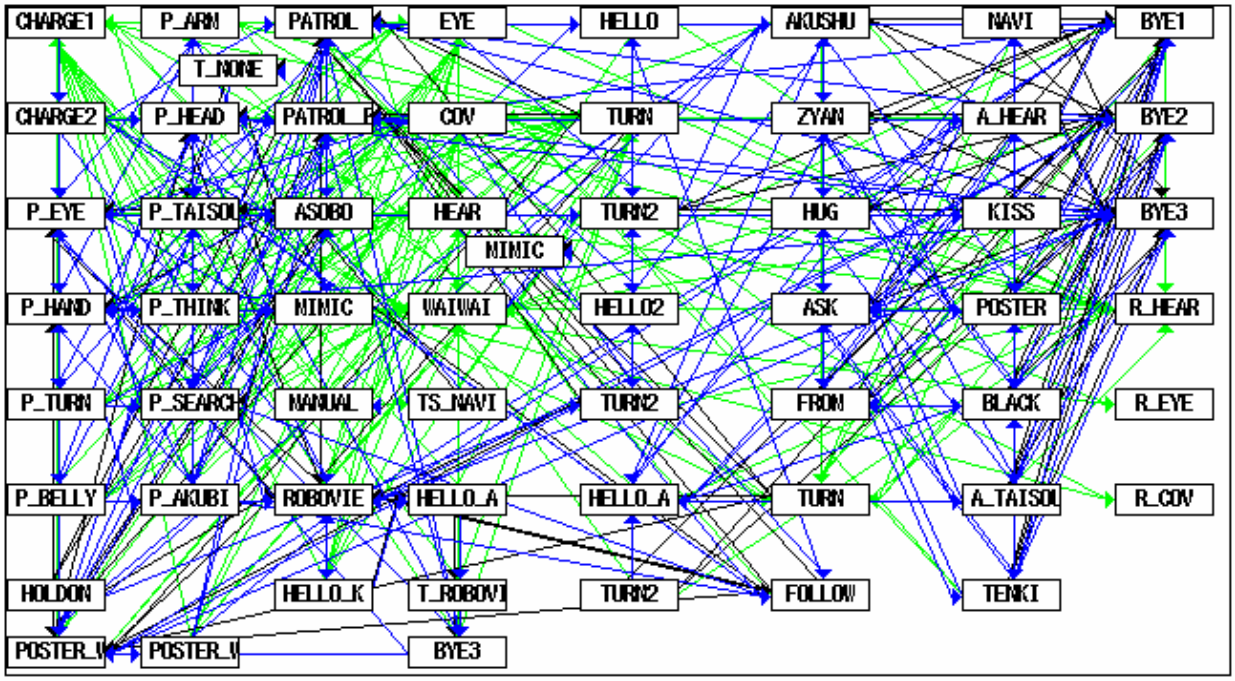
# Bye, bye

# Keep an appropriate distance

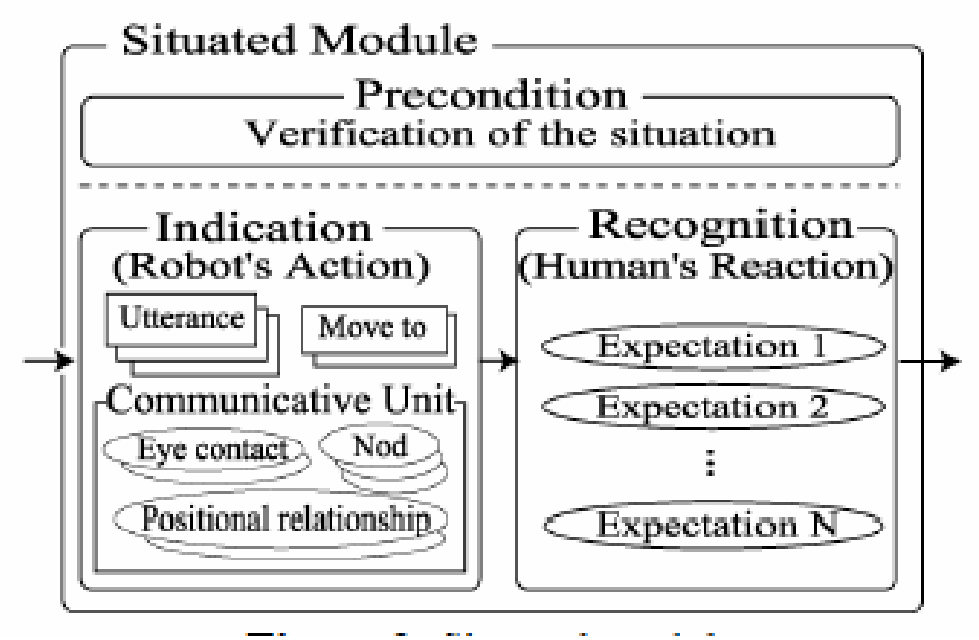




# Situated Modules

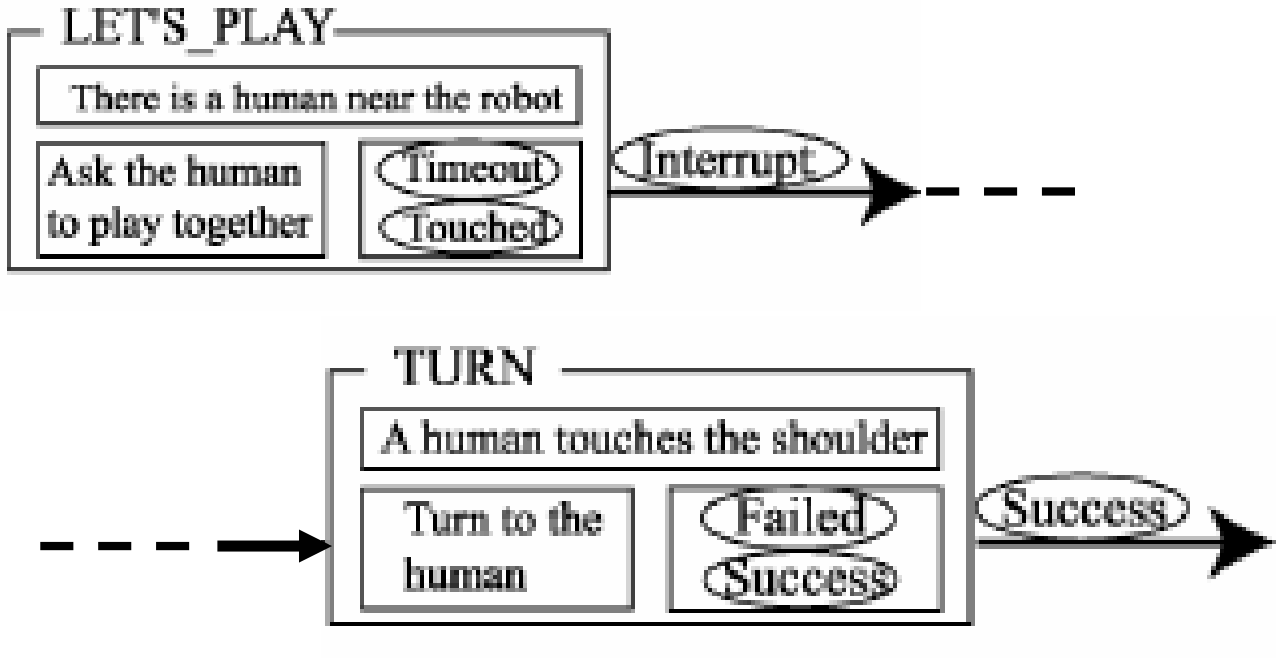


# Situated modules

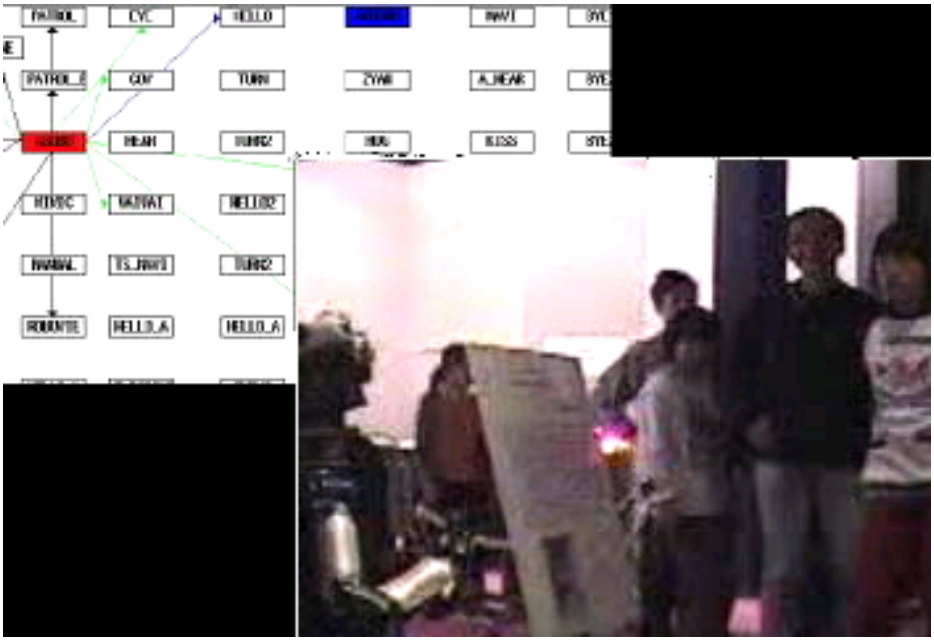





# An example of situated modules



# An example of situated modules





# 2. Social intelligence based on network robots

# Network Robot in School

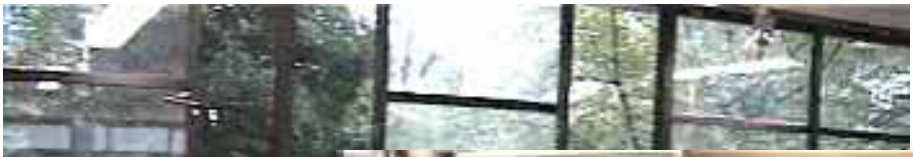
wireless tag

Antenna



# First Day

1st grade



6th grade

# First week passed

1st grade

6th grade



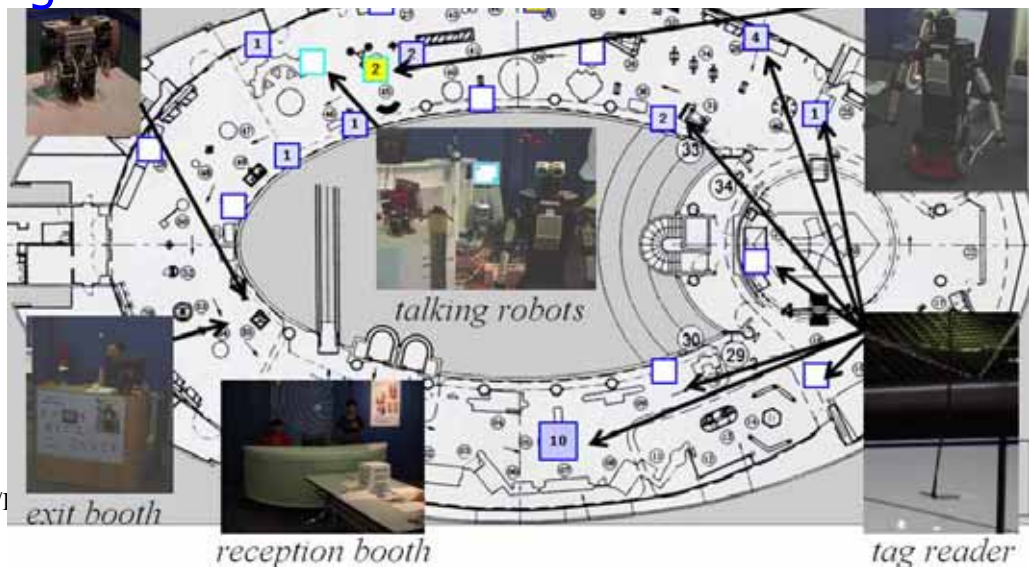
# Two weeks passed

6th grade

1st grade

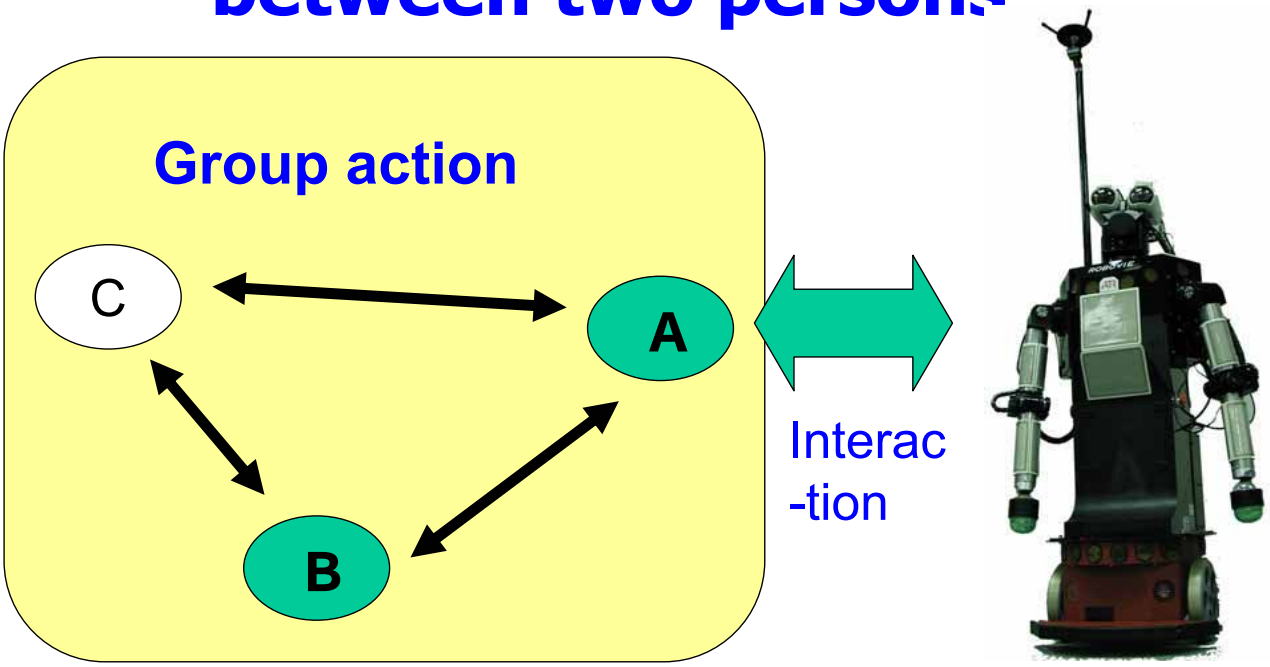
# System Configuration

- Two Robovie II's/M's
- 16 RFID tag Readers and 400 RFID tags to visitors
- Two stationary cameras in ceiling
  Four video cameras for monitoring
- Registration in entrance

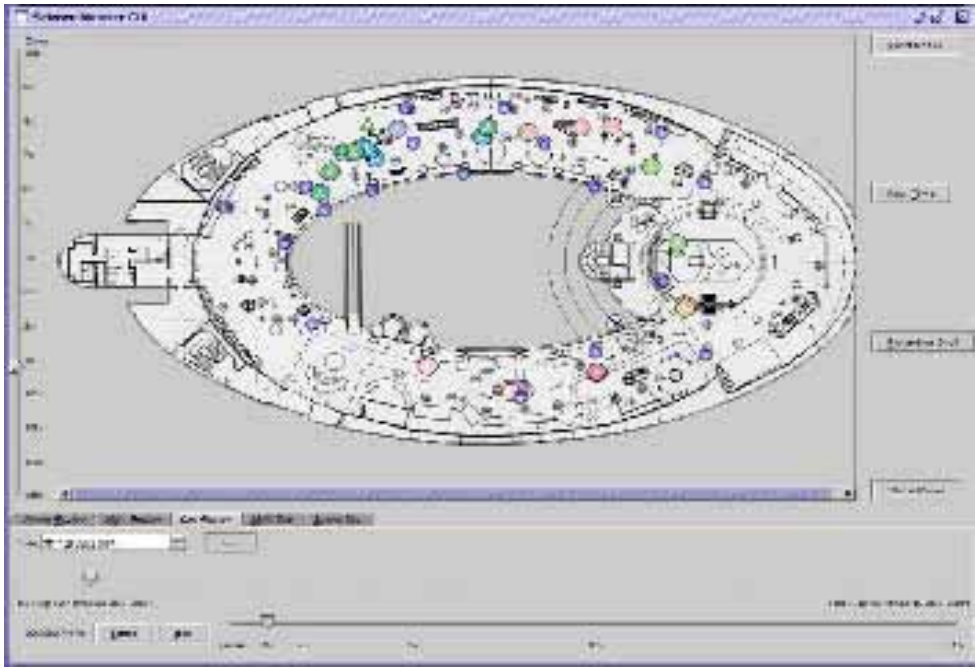





# Estimating friendship relationship between two persons

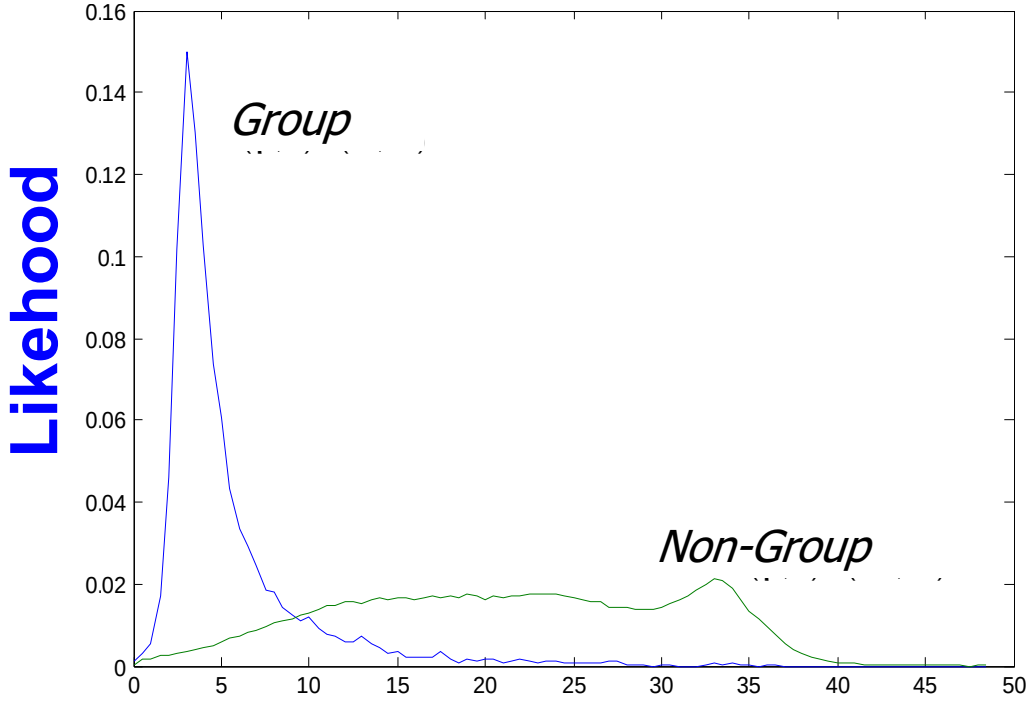


# Group's Behaviors

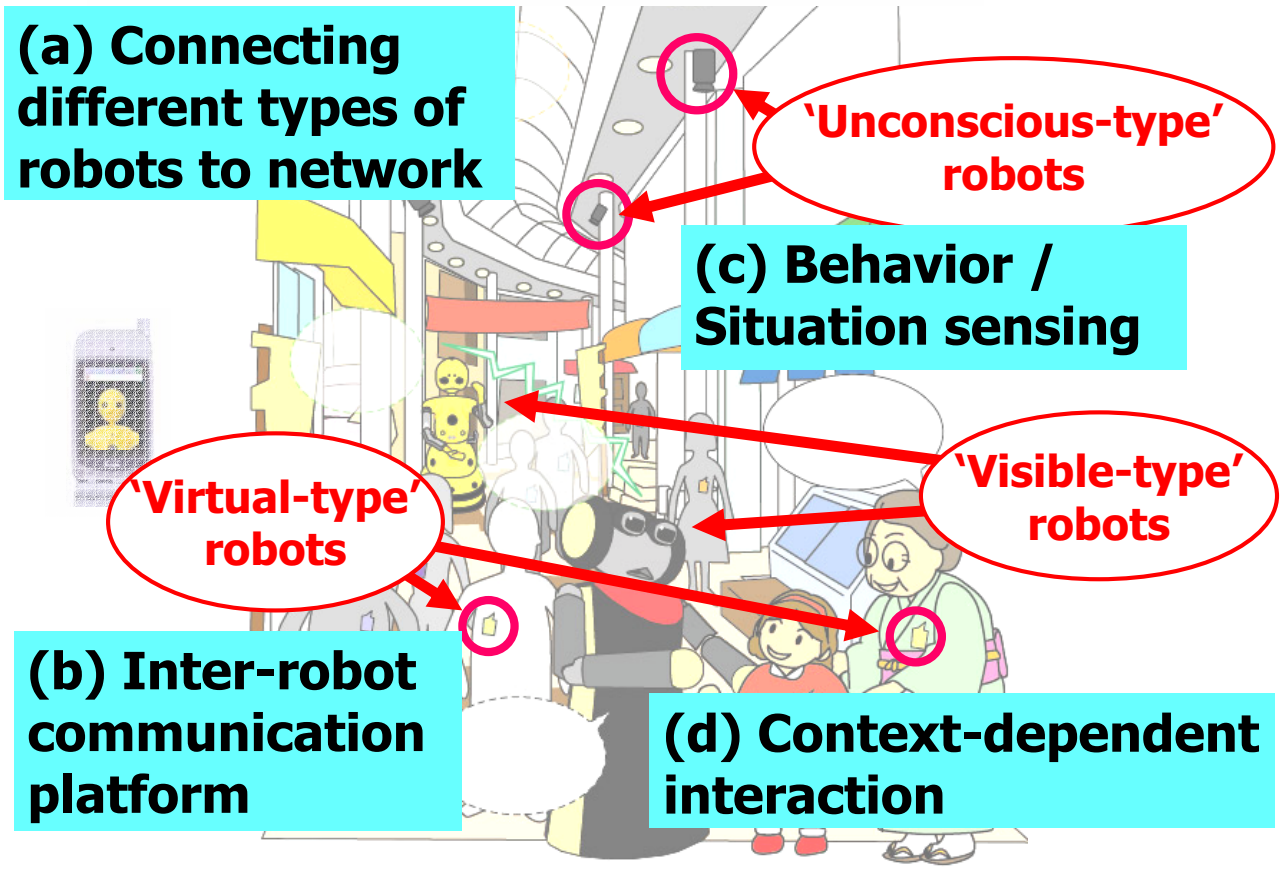




# Distribution of distances between two persons (about 12,000 persons)



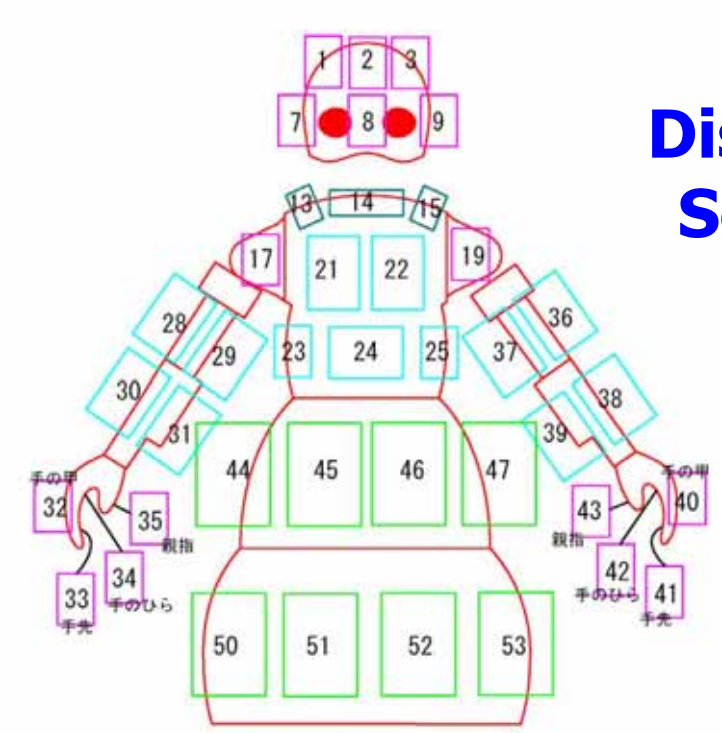
# Research Issues





**Robovie II** **Robovie IV**

# Physical Contact Communication




## Distributed Tactile Sensor Elements

- 53 elements film-type piezo-electric sensors
- 64 levels/ element



# Human Robot Interaction ATR

# 3. Ubiquitous experience media based on communication robots and ubiquitous sensors



## Ubiquitous Experience Media Research Project

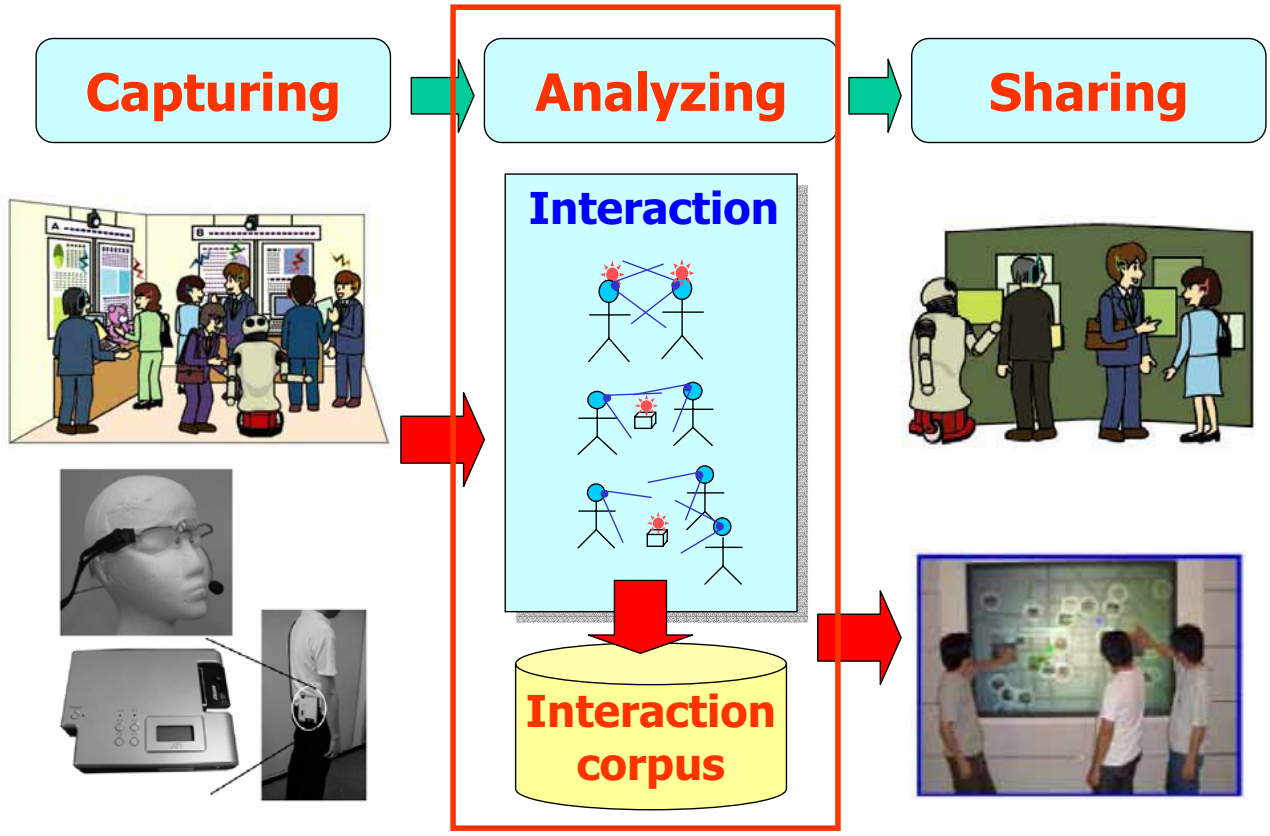
**Purpose**

**Build an interaction environment that allow anybody to observe and share human experiences anytime, anywhere, and in any way**

**Ubiquitous sensors**

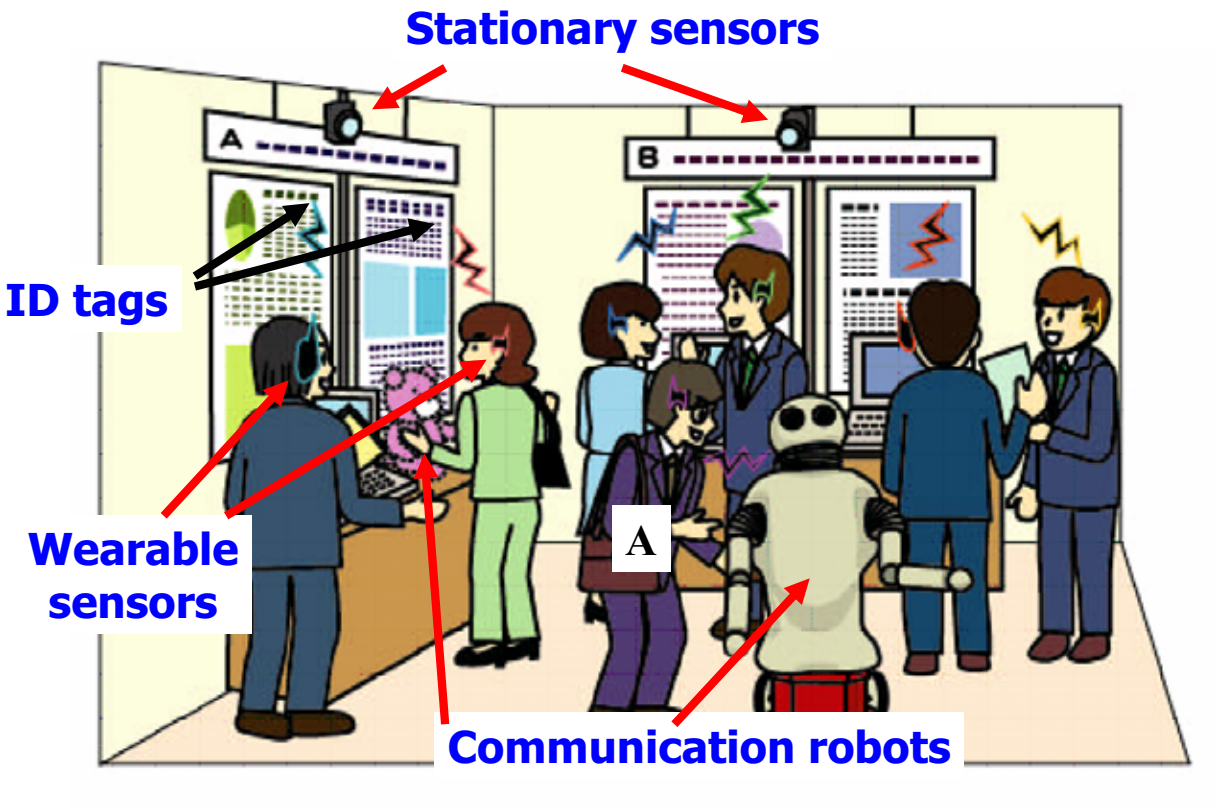
**Communication robots**

# Ubiquitous Experience Media



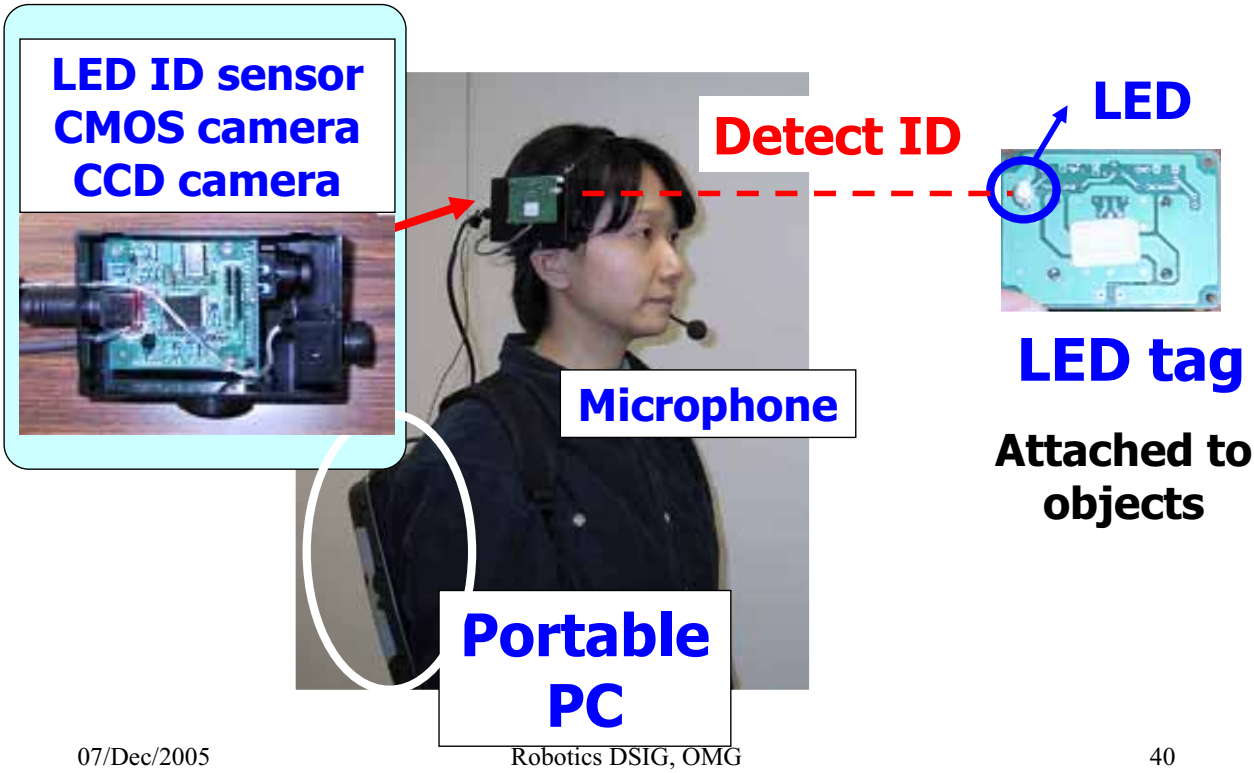

# Vistor's Experiences in Exhibition Room

# Capturing Experiences in Room E

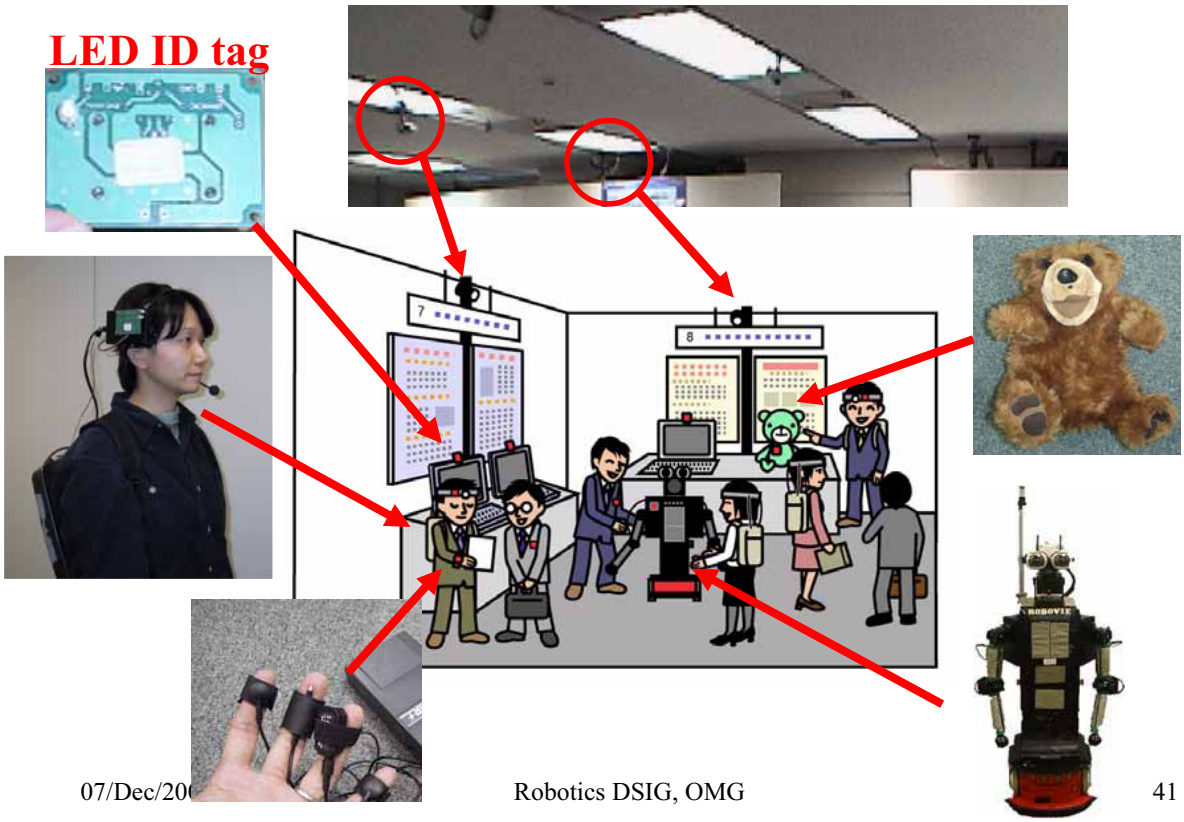





# 2002 Wearable Sensors

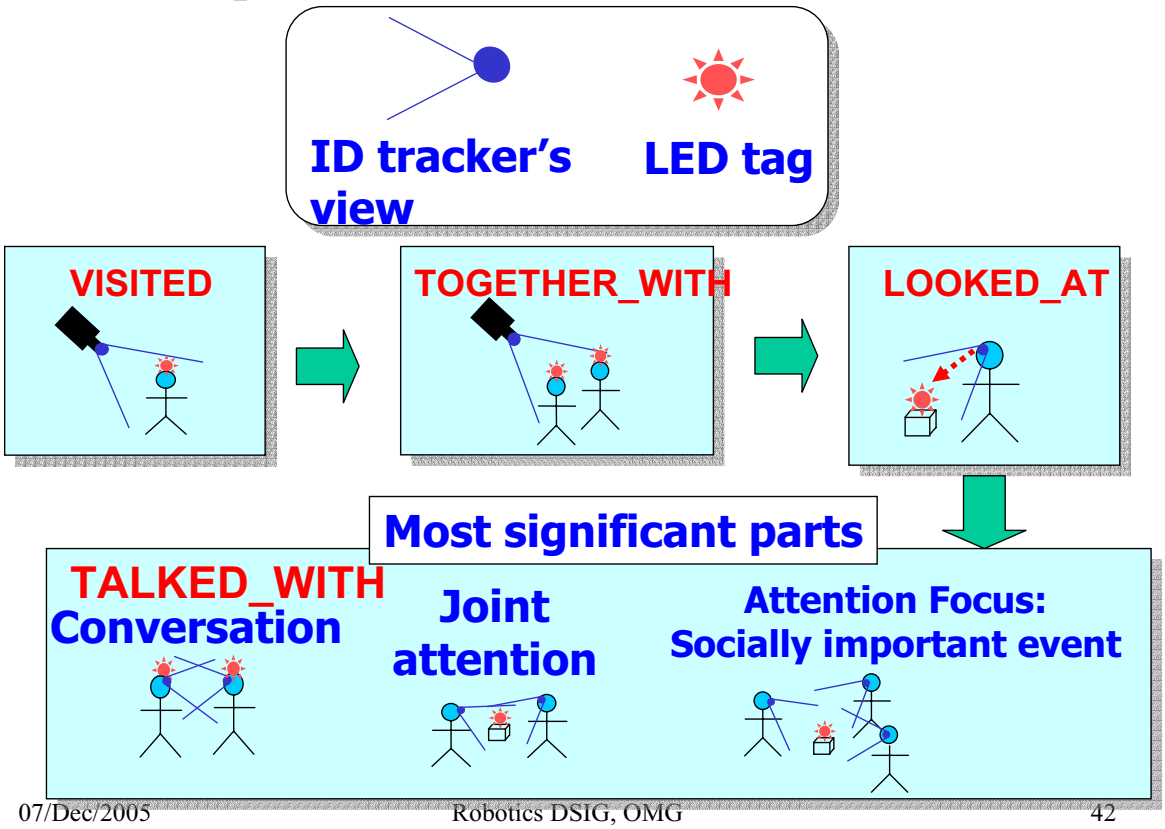


# Our Intelligent Environment in 2002

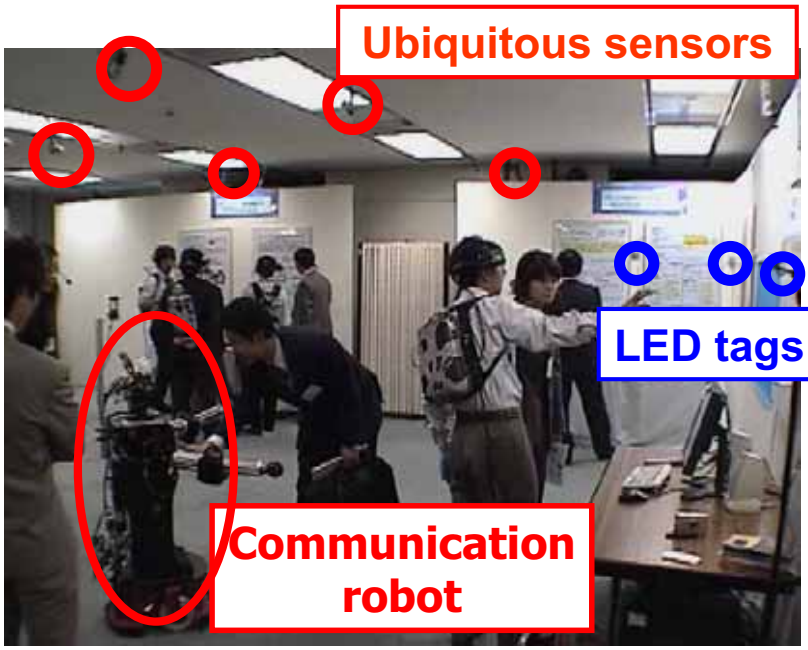





# Priority of Interaction Primitives



# Intelligent Environment at ATR



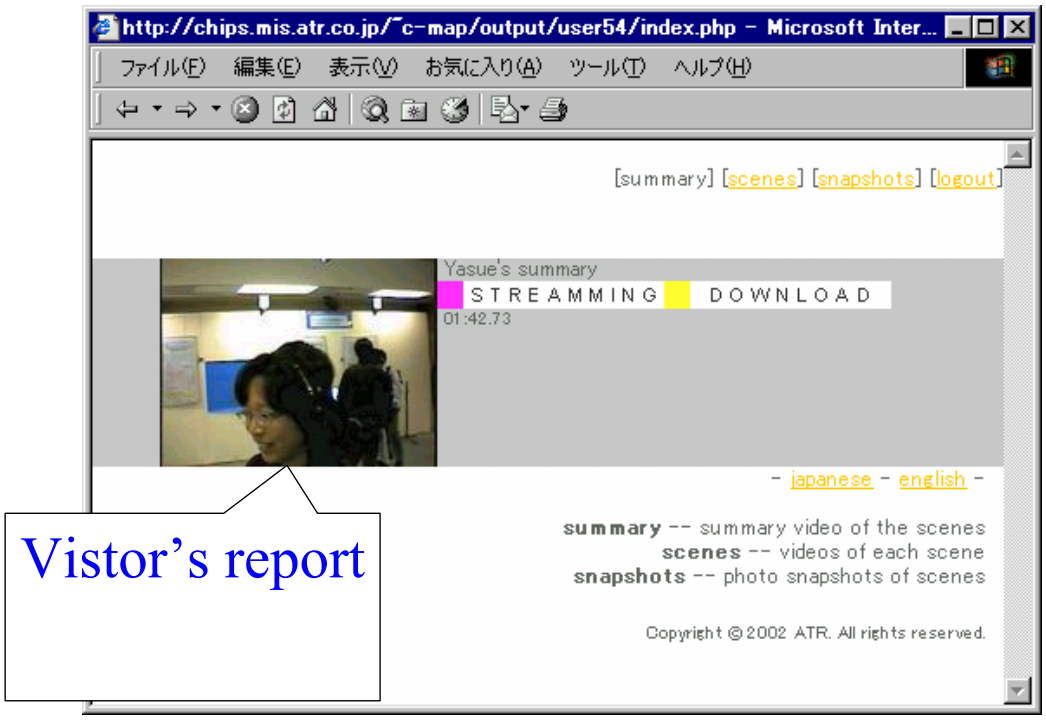

**Experiments**

**16 presenters**
**63 visitors**
**340GB**
**380,000 ID data**

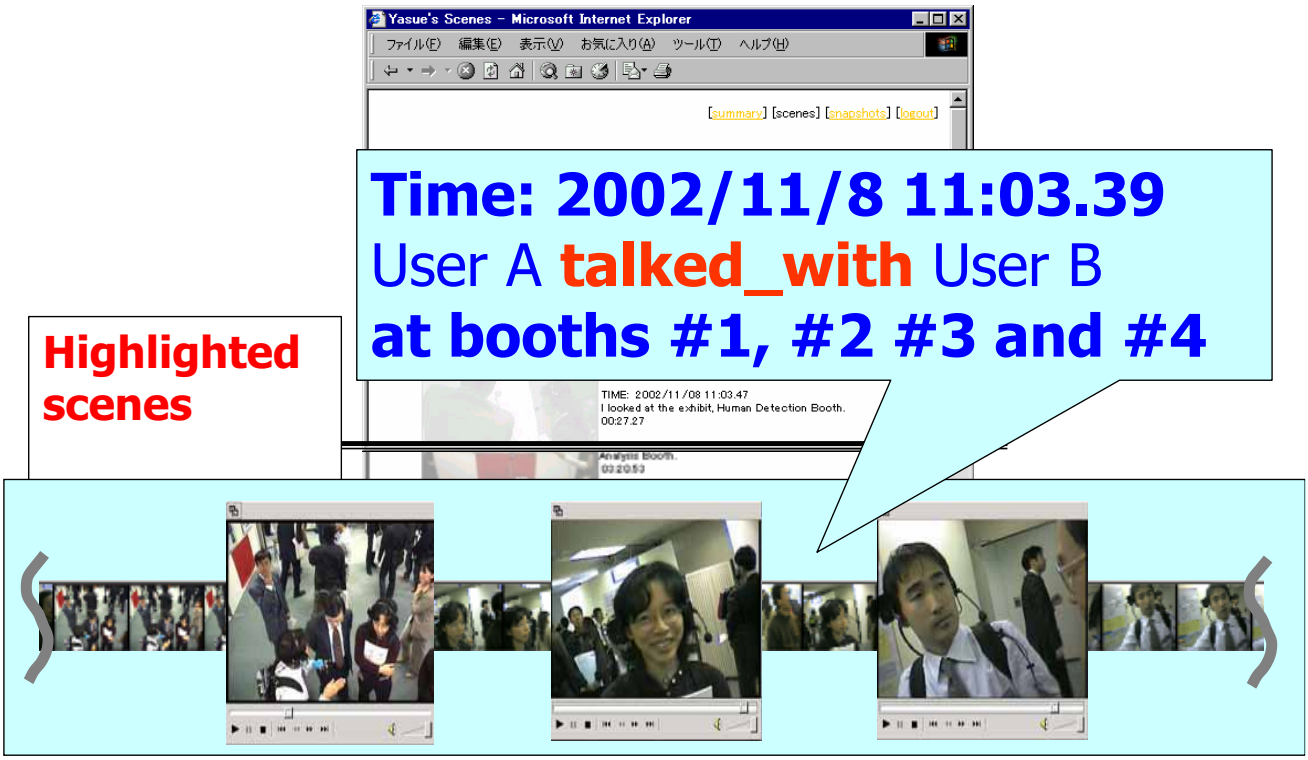
**Nov. , 2002**
**ATR Open House**



# HTML-based Video Summary

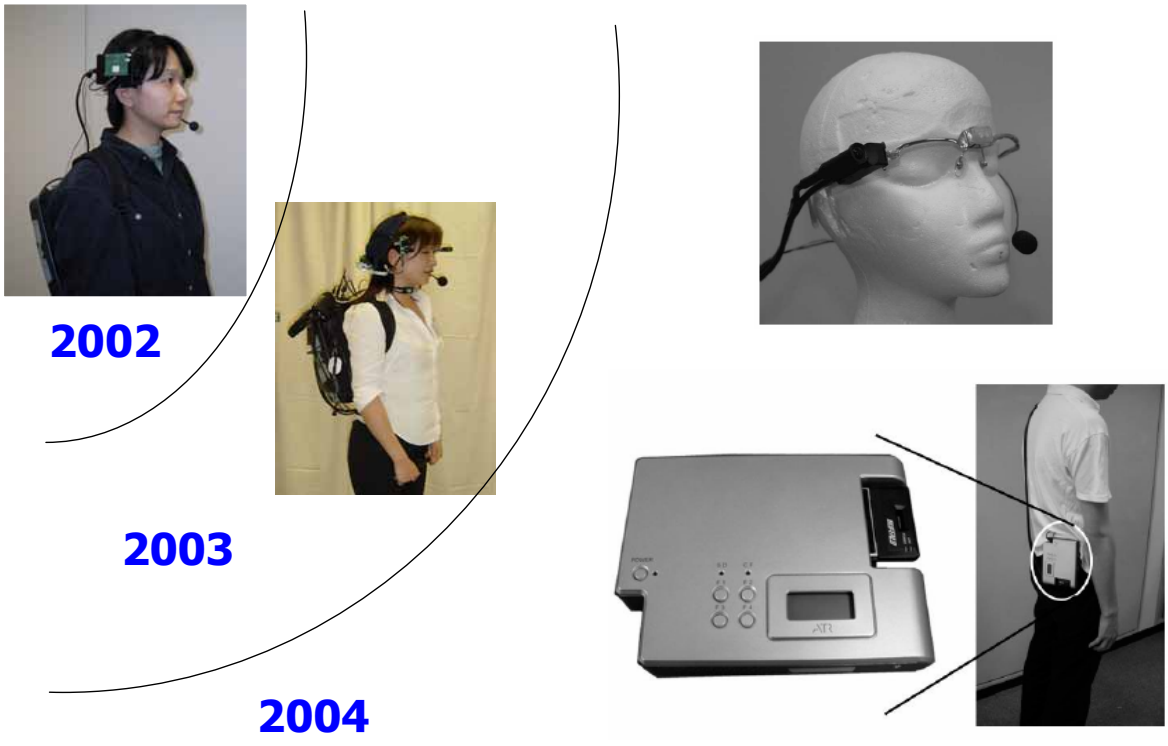


# HTML-based Video Summary

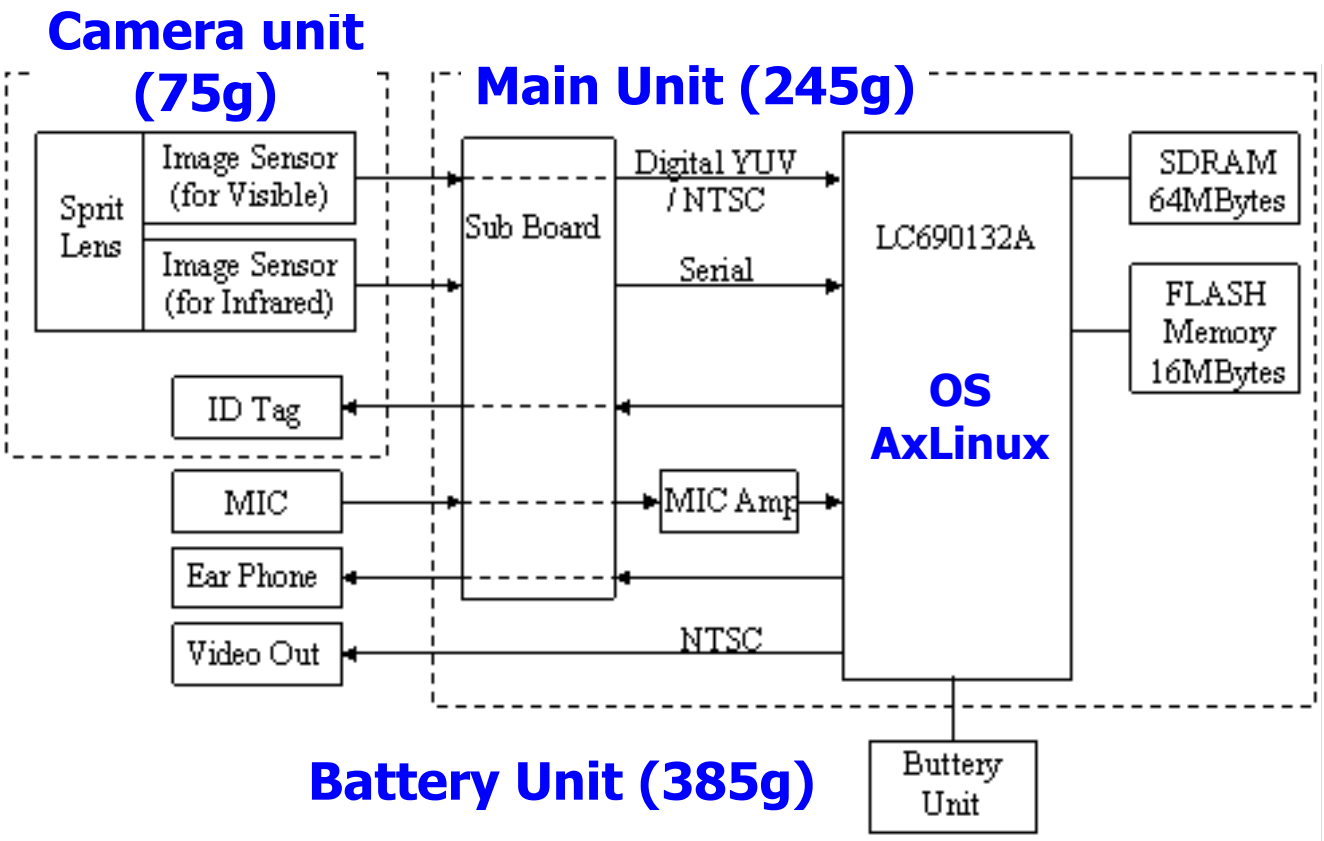





# Wearable Computer Unit

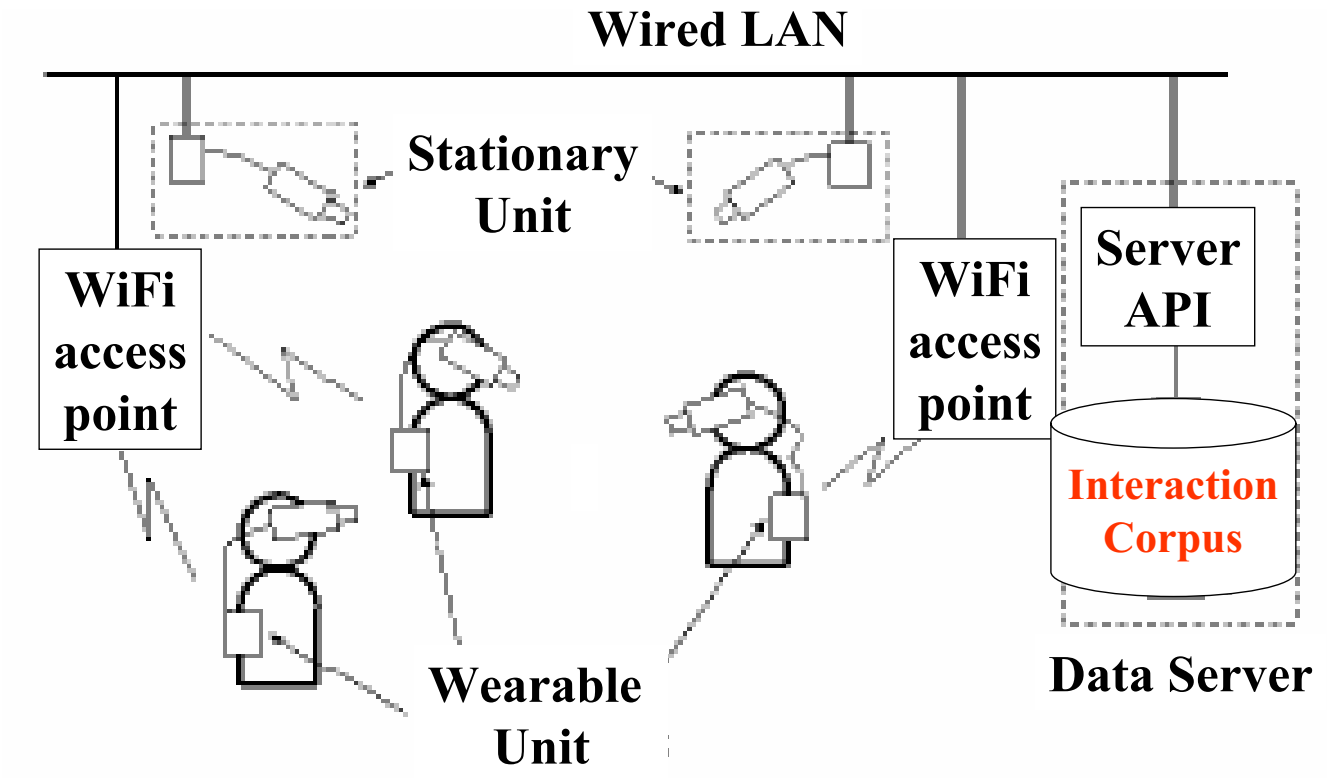


# Wearable unit in 2004

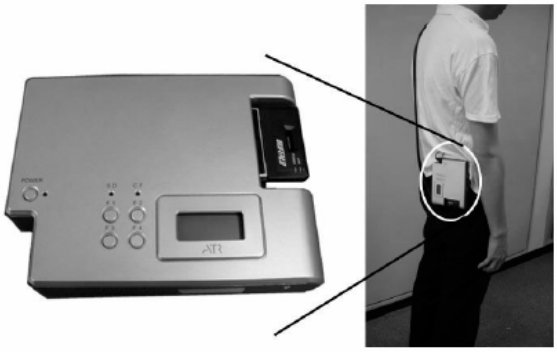





# Base server/client system



# 2004 Wearable Computer Unit





# Towards standardization ATR

1. Behavior languages for human robot interaction
   → basic behavior languages for different-type robots

2. Social intelligence based on network robots
   → software modules for social intelligence

3. Ubiquitous experience media based on communication robots and ubiquitous sensors
   → interaction primitives and corpus from multiple sensor data

TOSHIBA

# Network Robots Standardization Activities in JAPAN

**Miwako DOI**
**Corporate Research & Development Center**
**TOSHIBA Corp.**



TOSHIBA

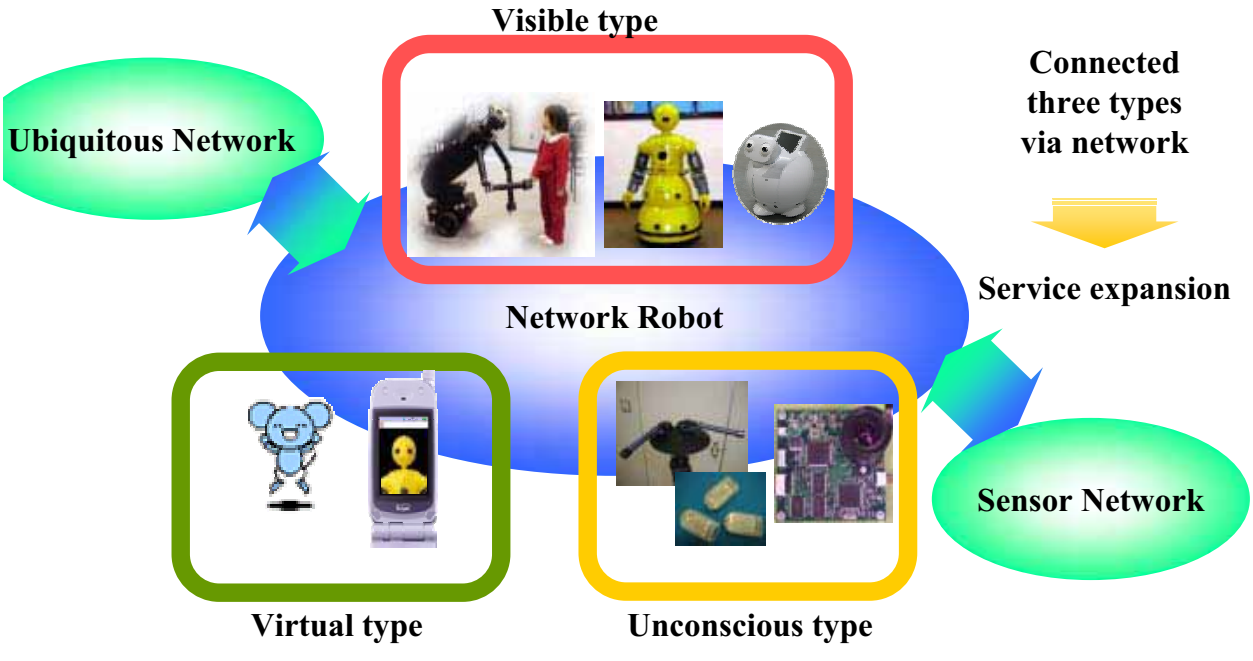
## Agenda

- Network robot standardization by Network Robot Forum
- Service flows of network robots
- Features and difficulties of network robots services
- Case studies of network robots cooperation

- **Network robot standardization by Network Robot Forum**
- Service flows of network robots
- Features and difficulties of network robots services
- Case studies of network robots cooperation

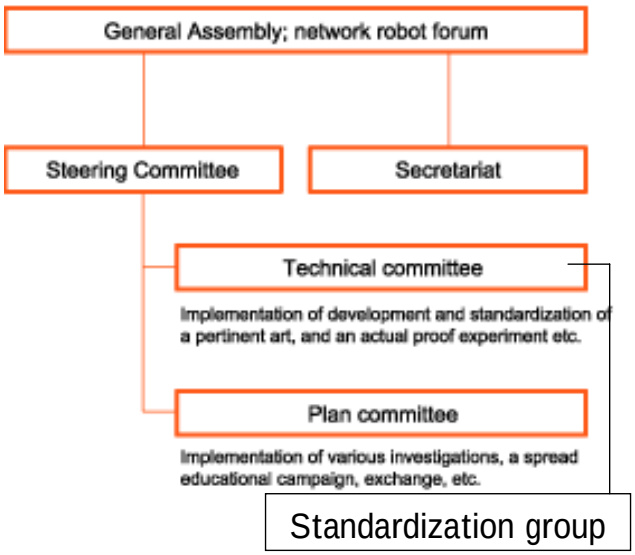


## What's the network robot?



TOSHIBA

## Organization of Network Robot Forum




**Purposes**

1. Promotion of R&D/standardization on network robots
2. Support for various surveys and verification experiments on network robots
3. Promotion of information/personnel exchanges with relevant organizations including ones in Europe, the U.S. and Asia

**Major activities**

- R&D/standardization on network robots
- Support for and liaison with relevant organizations concerning network robots
- Awareness campaigns for network robots

**Officers**

Chair: Prof. Hideyuki Tokuda, Keio University
Vice Chair: Mr. Yutaka Ozawa, Mitsubishi Heavy Industries Ltd.
Vice Chair: Mr. Hiroshi Miyabe, Nippon Telegraph and Telephone Corp.

**Secretariat**

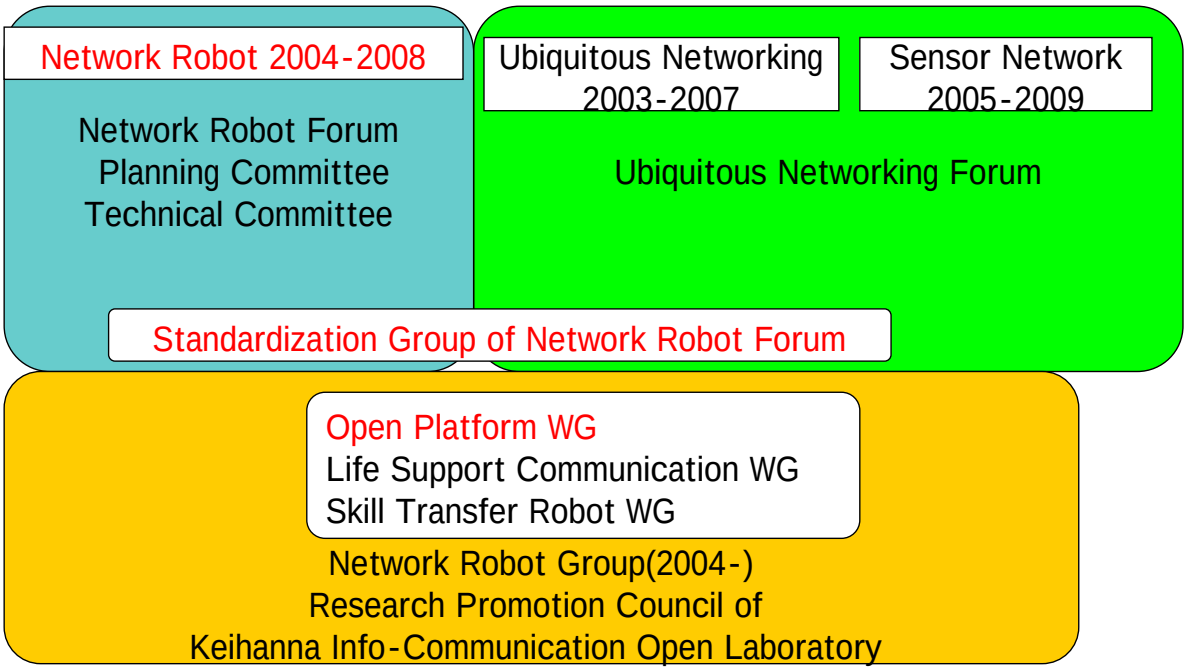
Support Center for Advanced Telecommunications Technology Research, Foundation (SCAT)

**Foundation**

September 30, 2003

TOSHIBA

## Relations with National Projects and Forums



- Network robot standardization by Network Robot Forum
- **Service flows of network robots**
- Features and difficulties of network robots services
- Case studies of network robots cooperation

TOSHIBA

## Service flows of network robot

- **Interpretation**: to appropriately interpret user's requests according to the context based on the cooperation among network robots, ubiquitous networks, and sensor networks.
- **Discovery and retrieval**: to find the appropriate robots which meet the interpreted requests based on the constraints of location, time intervals and processing performances.
- **Organization**: to combine or connect finding network robots services into appropriate services according to user's requests and environment conditions.
- **Execution**: to move visible robots into the service providing point physically or to move the functions of network robots via networks.

## Features of network robots services

- Action to the physical space
- Two way communication between the physical space and the cyber space
- Sharing and synchronization among intelligent application layer, network layer and physical actuation layer for the safety
- Consistency with locality and ubiquity

TOSHIBA

## Difficulties of network robot services

- Lack of common concept concerning the services
- Diversity of the device performances
- Inconsistency between locality and ubiquity

## Policies of Network Robot Standardization

- Bridging standardization among Ubiquitous Network, Sensor Network, and Network Robot. The minimum information distribution for enhancement of application service coverage
  - Bridging standardization
  - Start up with discovery and retrieval
  - Bottom-up declaration
- Independent on device performance
- Including service coverage; location, duration, and so on



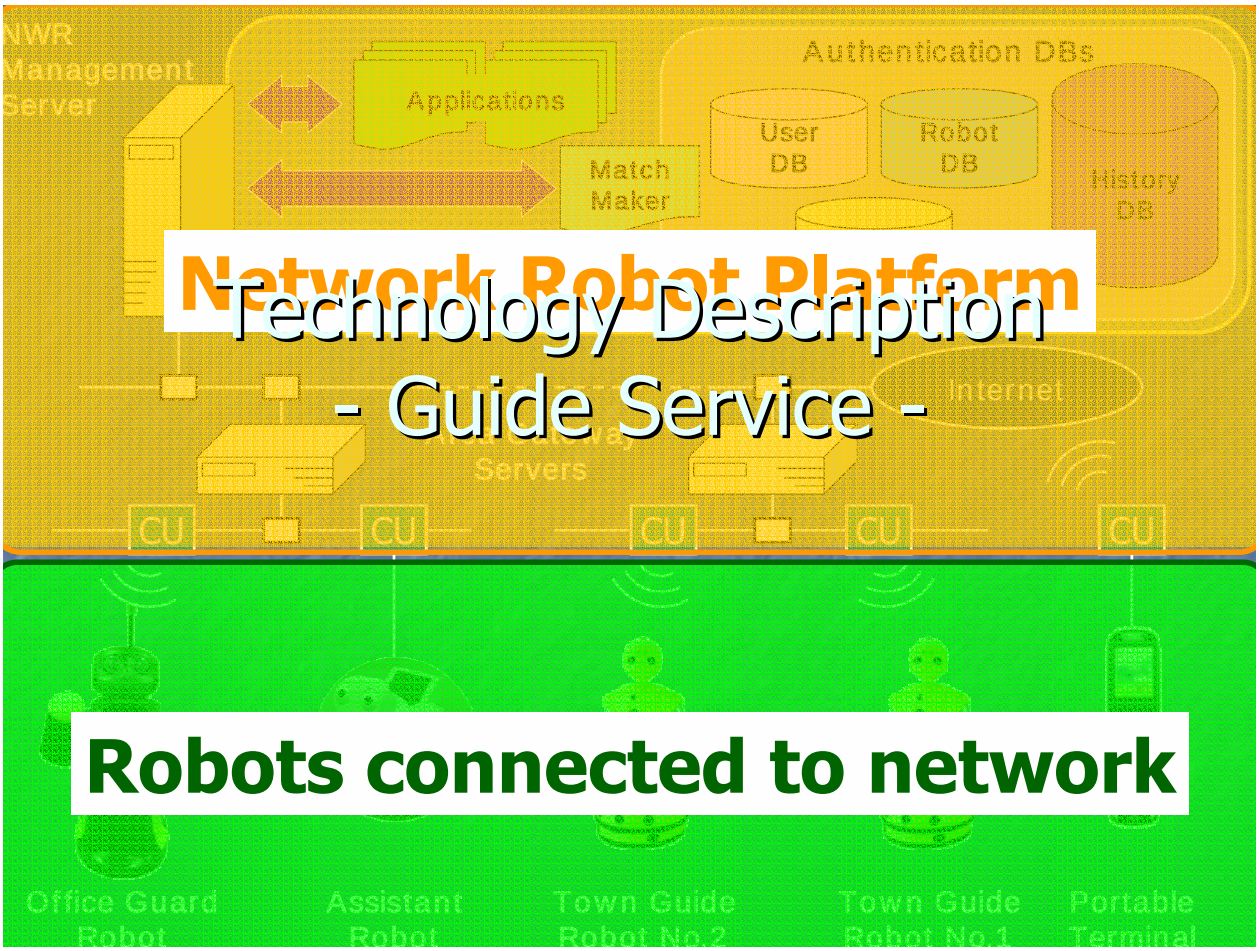
TOSHIBA

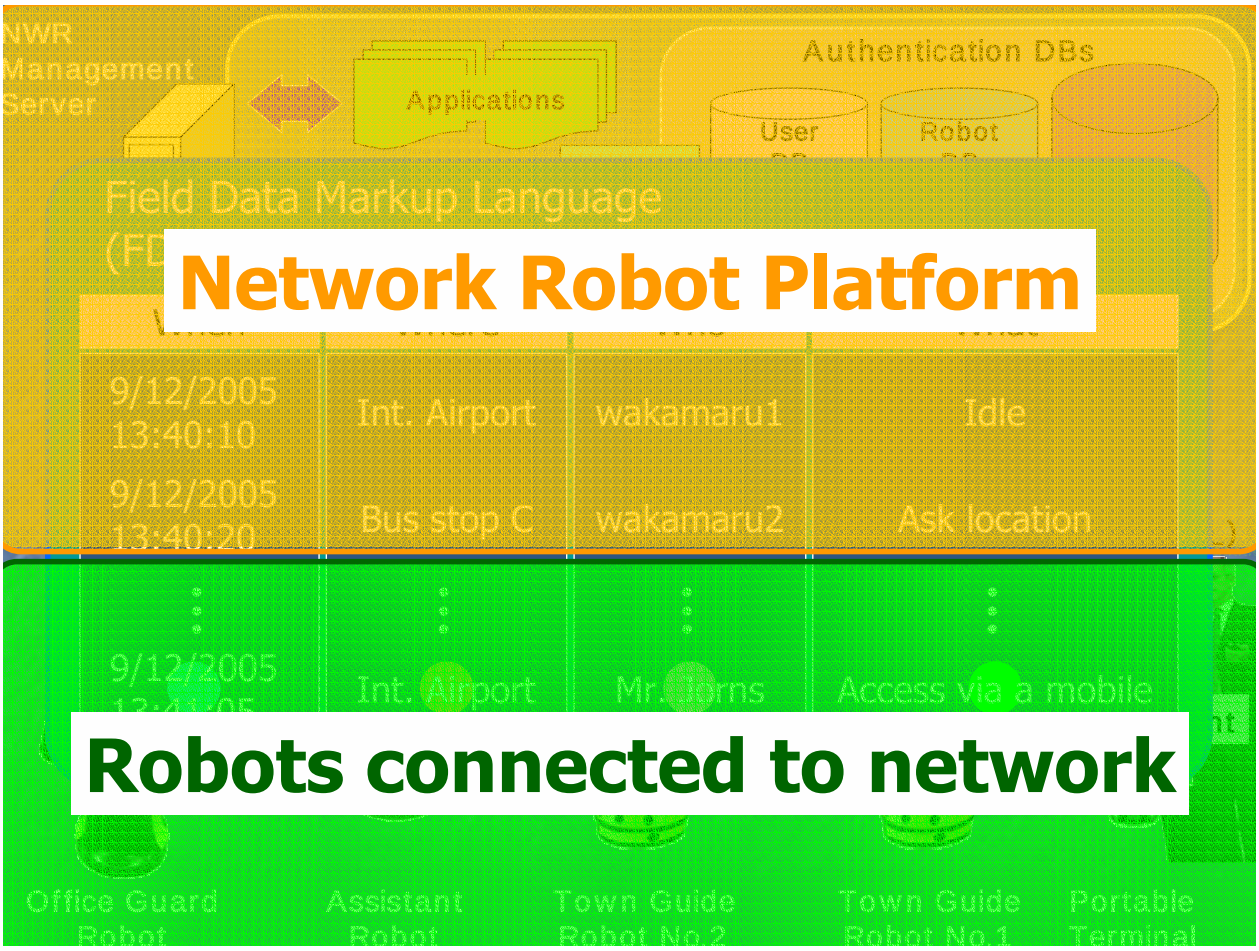
- Network robot standardization by Network Robot Forum
- Service flows of network robots
- Features and difficulties of network robots services
- **Case studies of network robots cooperation**

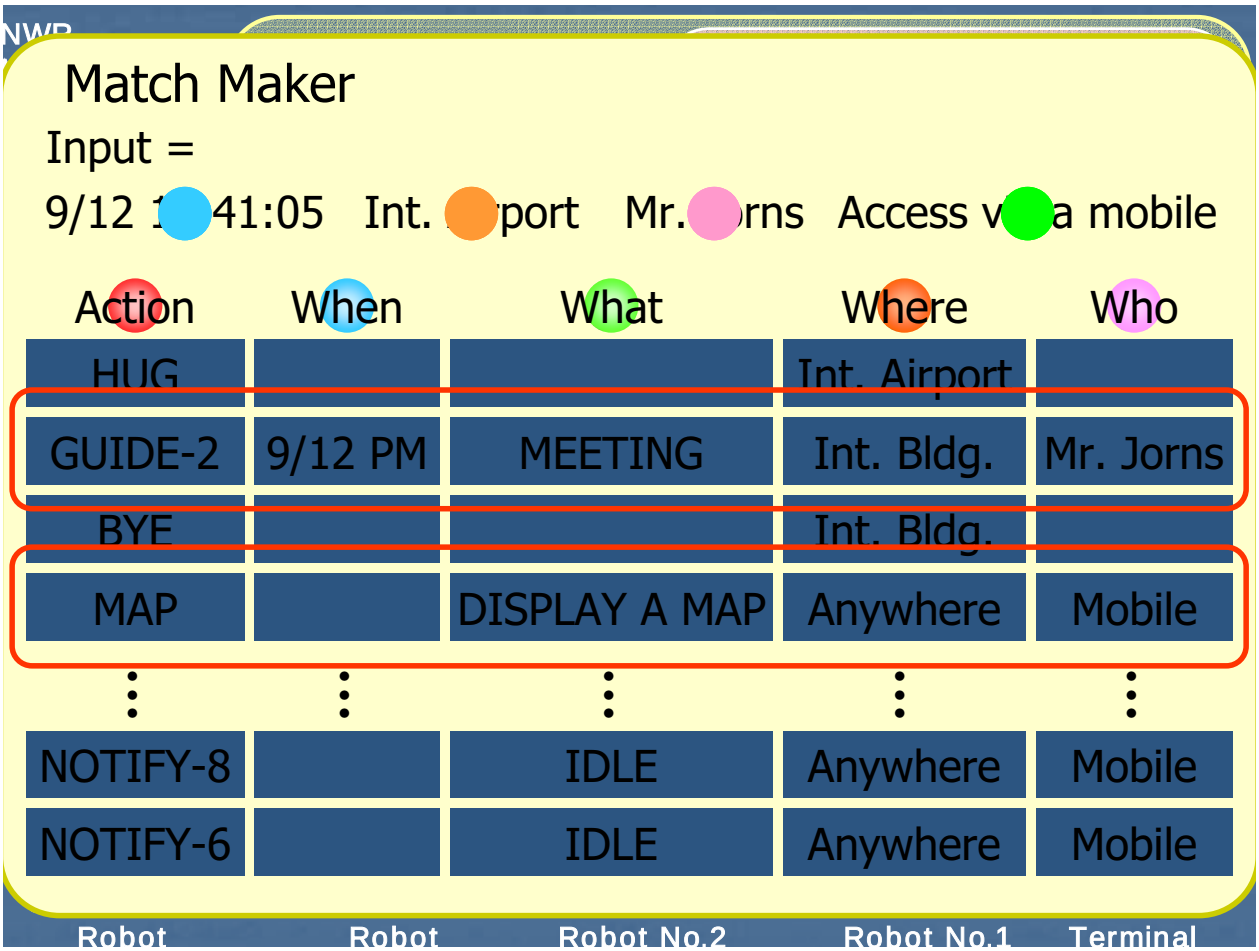
# Service Model
# ~Guide Service~

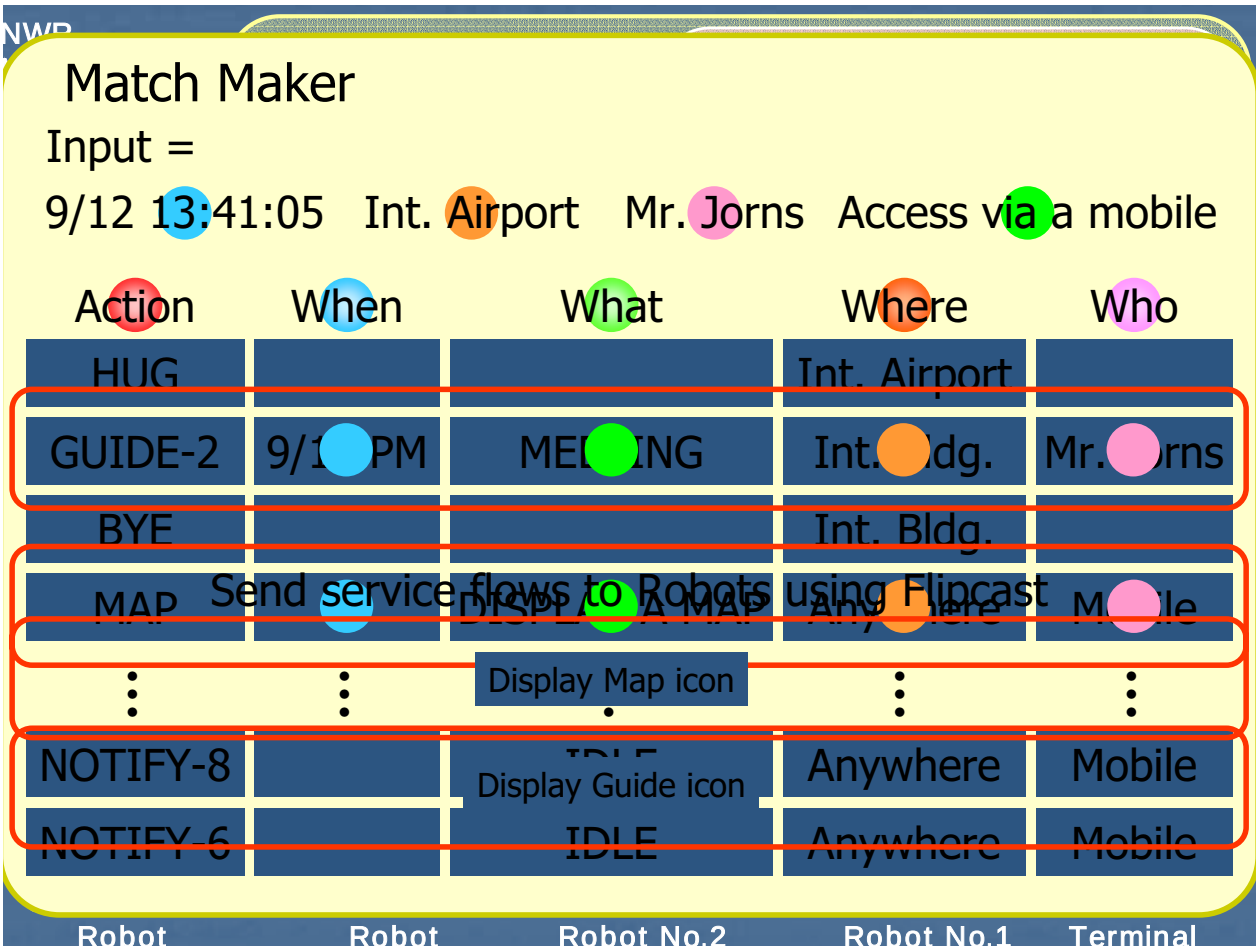
Service Model
~Guide Service~

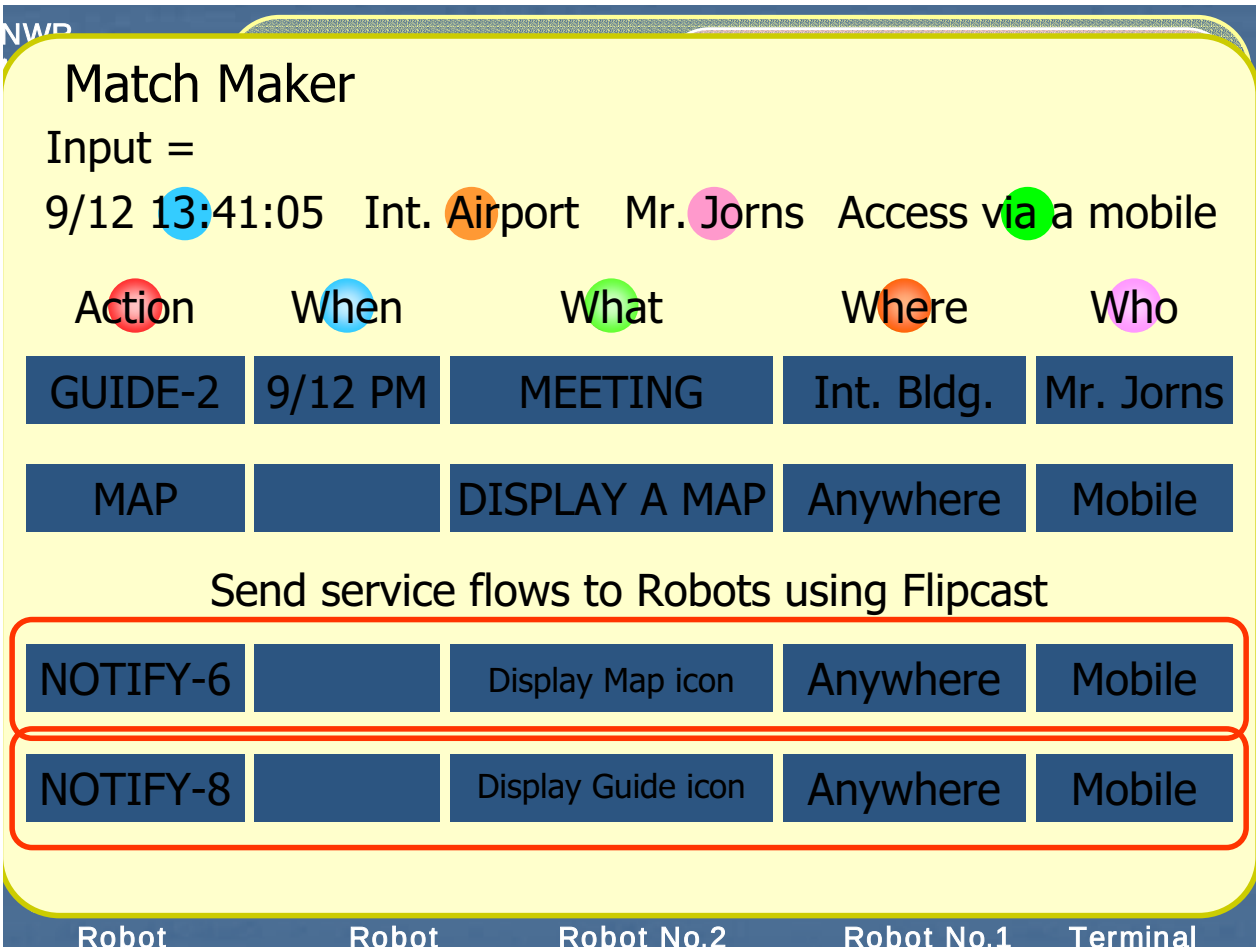
# Technology Description
# - Guide Service -

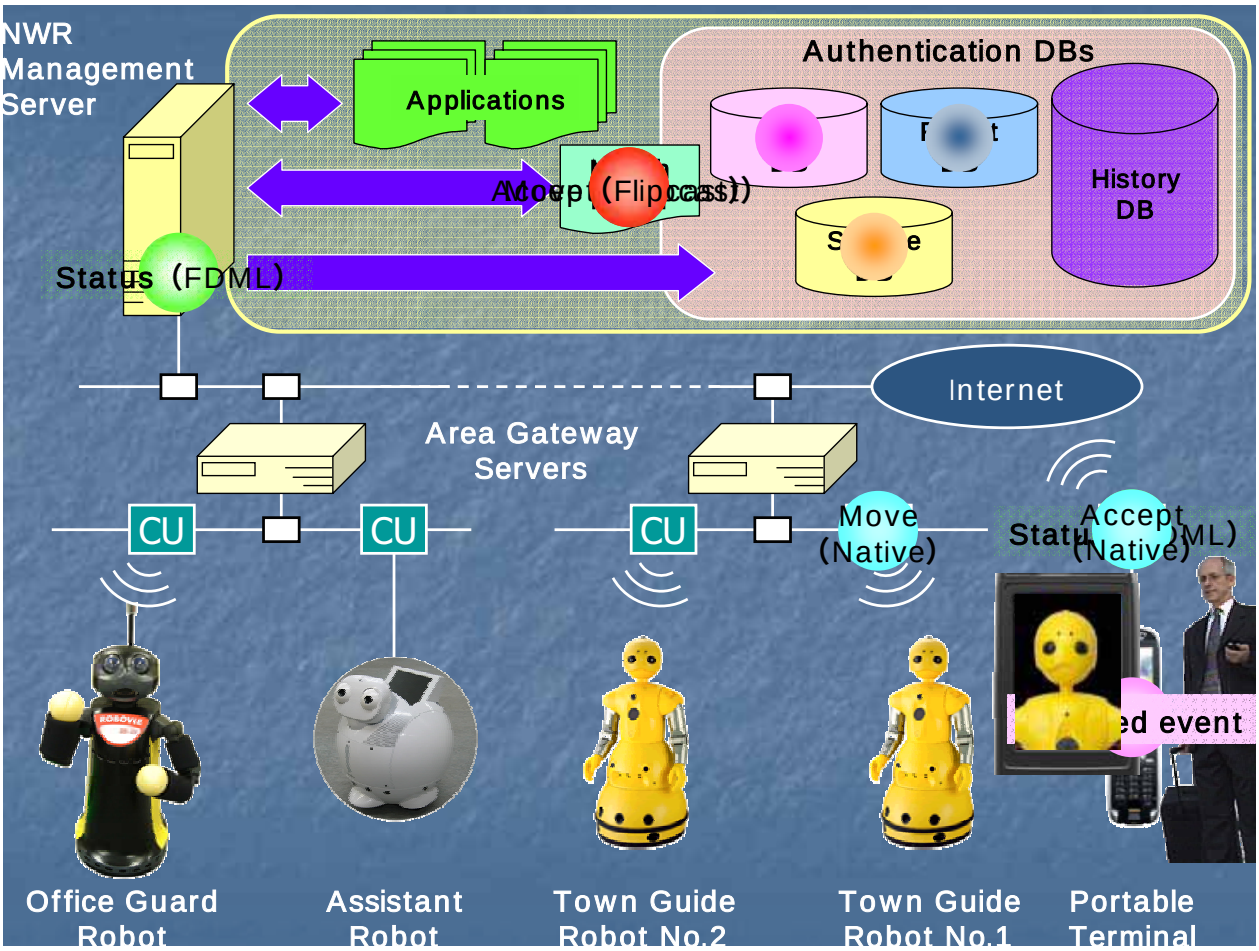


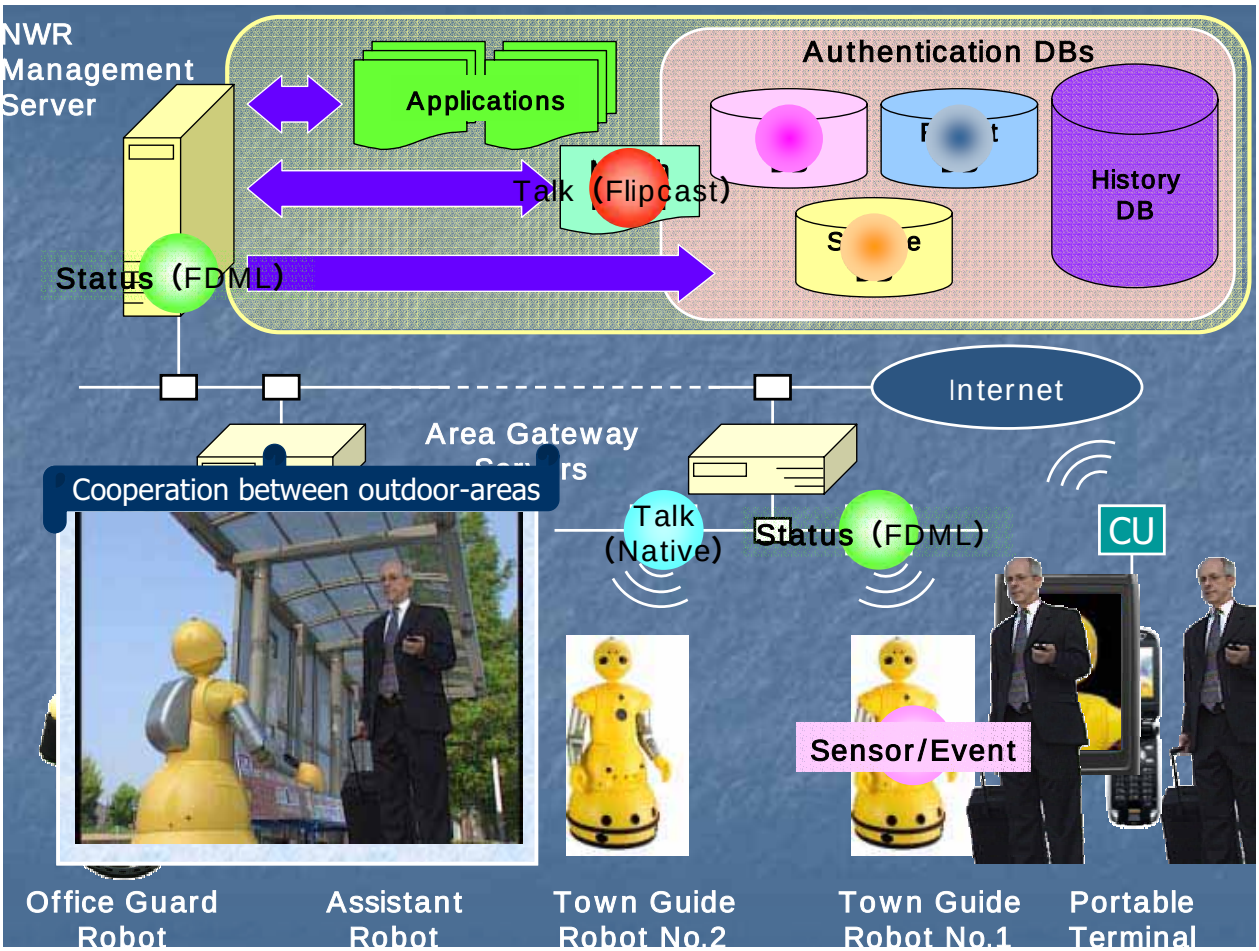

NWR Management Server
Applications
Authentication DBs
User
Robot
Field Data Markup Language
Network Robot Platform
9/12/2005 13:40:10
Int. Airport
wakamaru1
Idle
9/12/2005 13:40:20
Bus stop C
wakamaru2
Ask location
Robots connected to network
Office Guard Robot
Assistant Robot
Town Guide Robot No.2
Town Guide Robot No.1
Portable Terminal

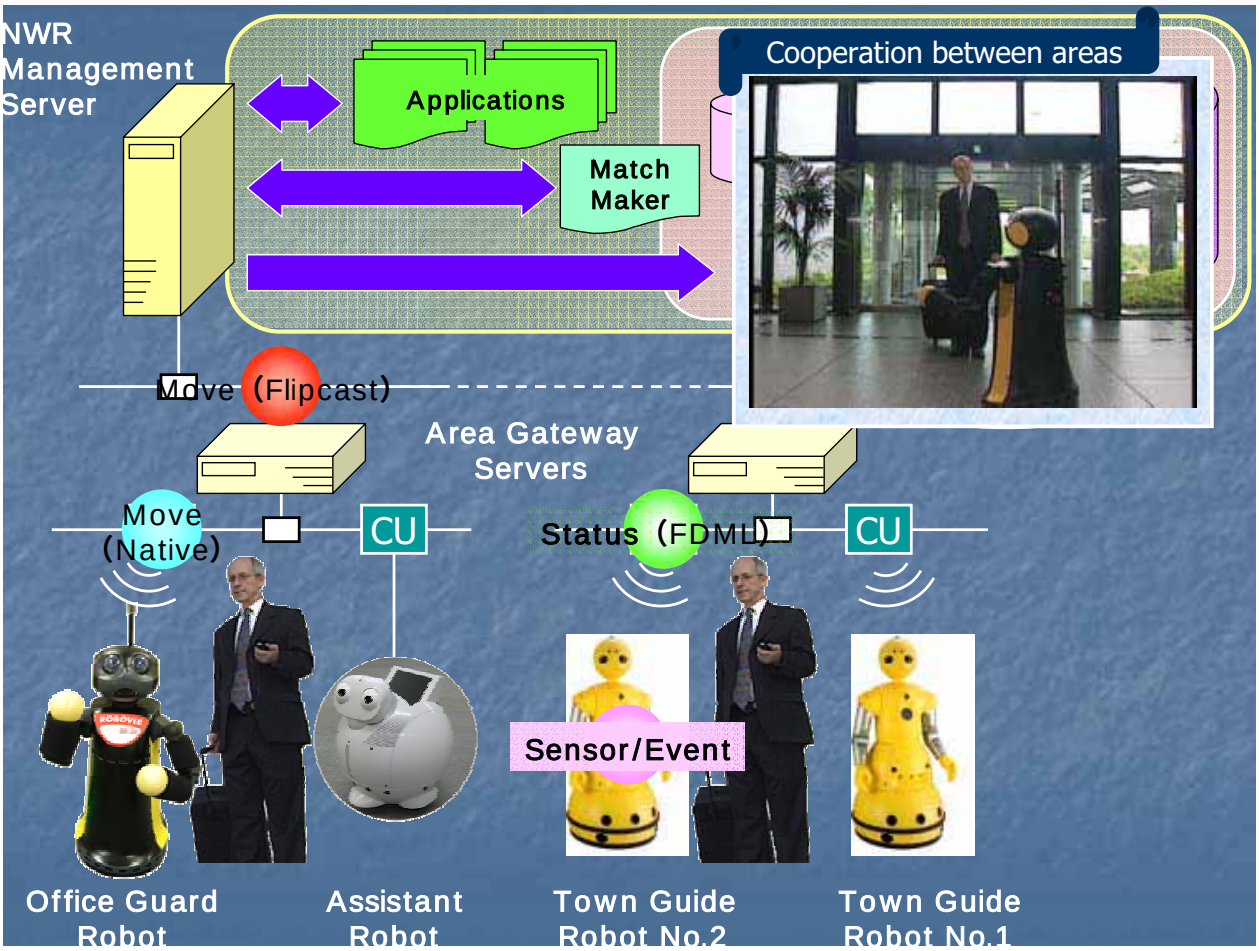

Match Maker
Input =
9/12 1 41:05 Int. port Mr. rns Access v a mobile
Action
When
What
Where
Who
HUG
Int. Airport
GUIDE-2
9/12 PM
MEETING
Int. Bldg.
Mr. Jorns
BYE
Int. Bldg.
MAP
DISPLAY A MAP
Anywhere
Mobile
NOTIFY-8
IDLE
Anywhere
Mobile
NOTIFY-6
IDLE
Anywhere
Mobile
Robot
Robot
Robot No.2
Robot No.1
Terminal

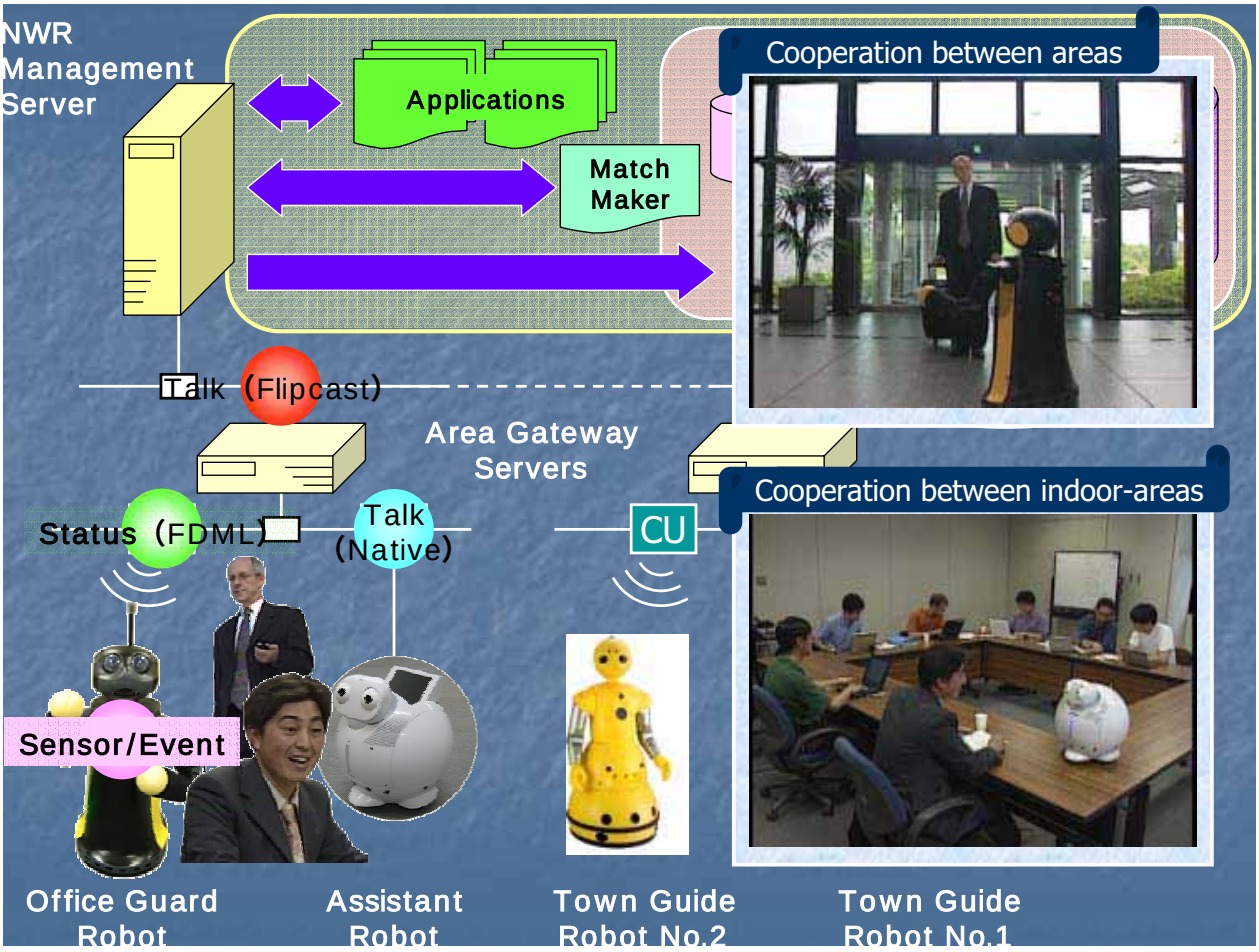

Match Maker
Input =
9/12 13:41:05 Int. Airport Mr. Jorns Access via a mobile
Action When What Where Who
HUG Int. Airport
GUIDE-2 9/12 PM MEETING Int. Bldg. Mr. Jorns
BYE Int. Bldg.
MAP DISPLAY A MAP Anywhere Mobile
Send service flows to Robots using Flipcast
Display Map icon
Display Guide icon
NOTIFY-8 Anywhere Mobile
NOTIFY-6 IDLE Anywhere Mobile
Robot Robot Robot No.2 Robot No.1 Terminal


Match Maker
Input =
9/12 13:41:05 Int. Airport Mr. Jorns Access via a mobile
Action When What Where Who
GUIDE-2 9/12 PM MEETING Int. Bldg. Mr. Jorns
MAP DISPLAY A MAP Anywhere Mobile
Send service flows to Robots using Flipcast
NOTIFY-6 Display Map icon Anywhere Mobile
NOTIFY-8 Display Guide icon Anywhere Mobile
Robot Robot Robot No.2 Robot No.1 Terminal


NWR Management Server
Applications
Authentication DBs
History DB
Status (FDML)
Internet
Area Gateway Servers
CU
CU
CU
Move (Native)
Accept (Native)
Office Guard Robot
Assistant Robot
Town Guide Robot No.2
Town Guide Robot No.1
Portable Terminal


NWR Management Server
Applications
Authentication DBs
History DB
Talk (Flipcast)
Status (FDML)
Internet
Area Gateway
Cooperation between outdoor-areas
Talk (Native)
Status (FDML)
CU
Sensor/Event
Office Guard Robot
Assistant Robot
Town Guide Robot No.2
Town Guide Robot No.1
Portable Terminal


NWR Management Server
Applications
Match Maker
Cooperation between areas
Move (Flipcast)
Area Gateway Servers
Move (Native)
CU
Status (FDML)
CU
Sensor/Event
Office Guard Robot
Assistant Robot
Town Guide Robot No.2
Town Guide Robot No.1


NWR Management Server
Applications
Match Maker
Cooperation between areas
Talk (Flipcast)
Area Gateway Servers
Status (FDML)
Talk (Native)
CU
Cooperation between indoor-areas
Sensor/Event
Office Guard Robot
Assistant Robot
Town Guide Robot No.2
Town Guide Robot No.1

TOSHIBA

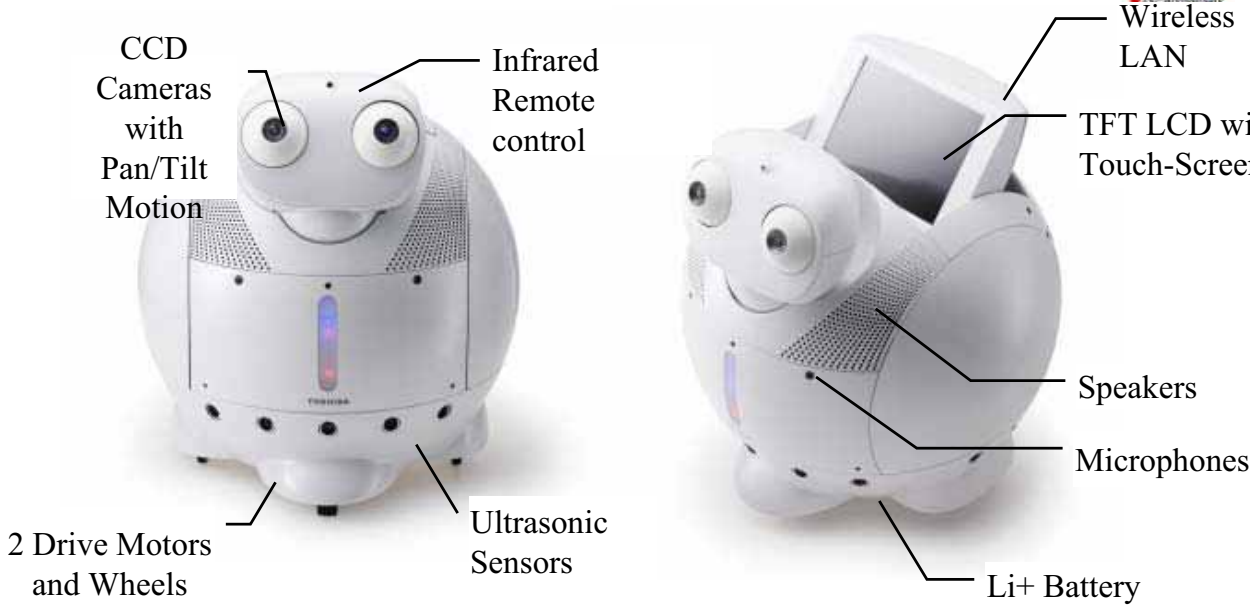
## Ubiquitous Home in NICT

- NICT: National Institute of Information and Communication Technology
- Project members: NICT, TOSHIBA, NEC, SANYO, OKI, NIAST, et.


TOSHIBA

## Robotic Information Home Appliance, "ApriAlpha"

**– ApriAlpha ：Advanced Personal Robotic Interface Type α –**




Size: 350x350x380 mm Weight 9.5 kg

TOSHIBA

## Ubiquitous Home


TOSHIBA

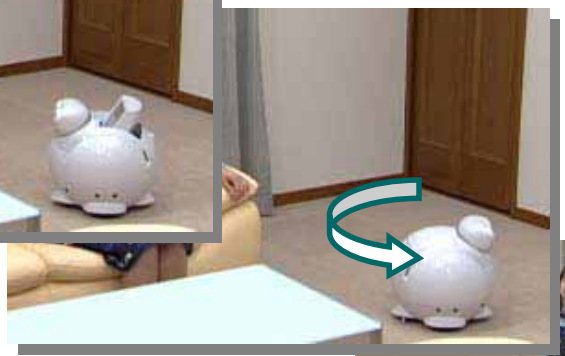
## Demonstration

Go to Welcome!

○Welcoming behavior

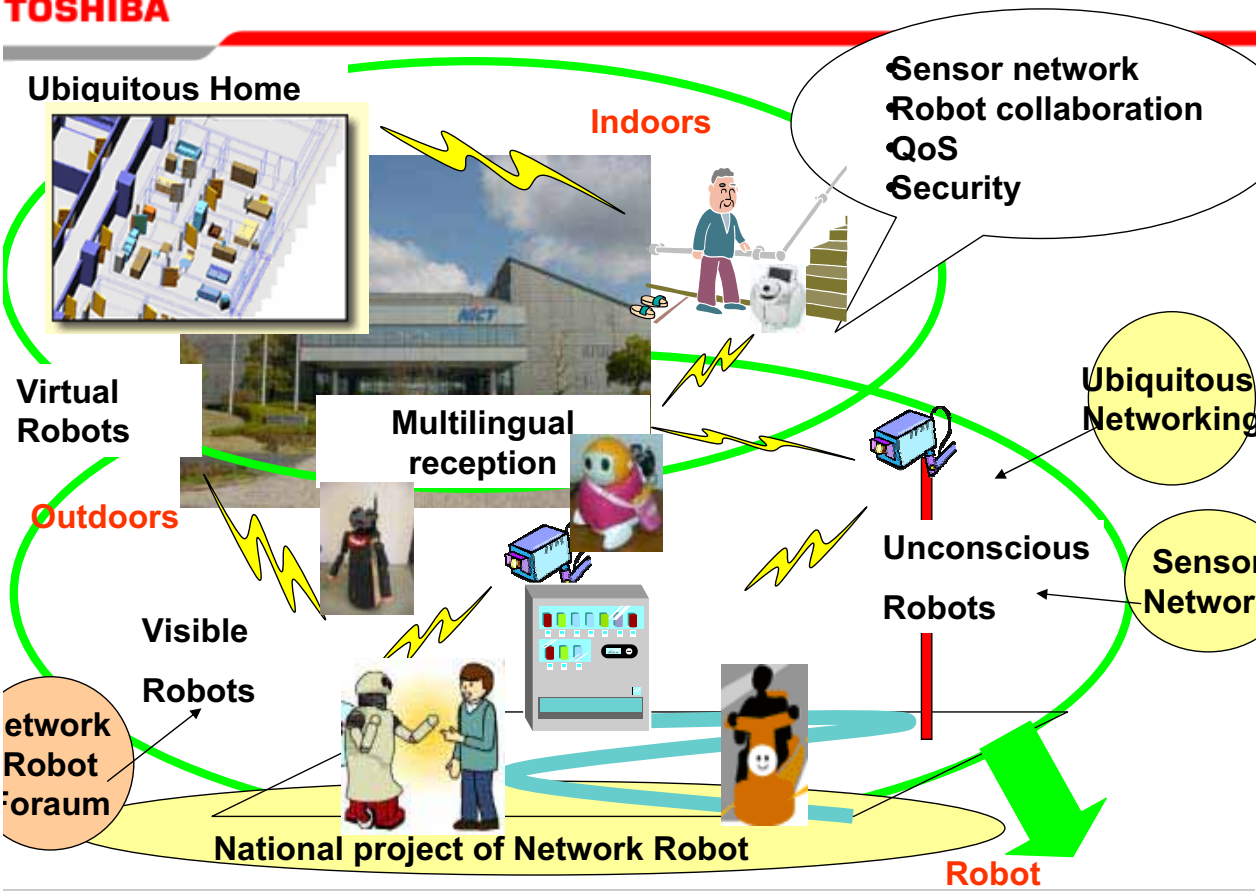
Go to the door

& "Welcome!,Welcome!"



Welcome!
Welcome!

Ubiquitous and Universal Town








## Summary

- Services offered by Network Robots
- Network robot standardization by Network Robot Forum
- Features and difficulties of network robots services
- Case studies of network robots cooperation

TOSHIBA

# Thank you for your attention.

This research was supported by Ministry of Internal Affairs and Communications

robotics2005-12-17

# Proposed Charter for Robotics DTF

**TC Meeting Date:** 9 December 2005
**Presenter:** Tetsuo Kotoku
**Group email:** robotics@omg.org
**Group Home Page (URL):**
http://robotics.omg.org/

## RFI midterm summary:

- **In the Burlingame meeting, we have 9 RFI presentations (include 4 RFI response)**
- **We will have more RFI responses until next Tampa meeting (extending deadline)**

## Roadmap:

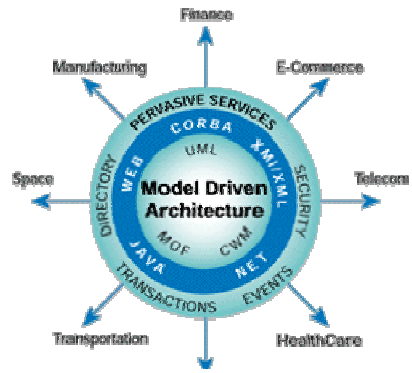
**(Potential RFP WGs)**

- **Robotics Component Middleware (RTI, SNU, etc.)**
- **Robotics Component Profile (Systronix, ETRI, NEC, SRI etc.)**
- **Robotics Service Profile (ETRI, ATR, Toshiba, etc.)**
- **Robotics Data Structure (NTT, NEC, ETRI, etc.)**

# Proposed Charter for Robotics DTF

**TC Meeting Date:** 9 December 2005
**Presenter:** Tetsuo Kotoku
**Group email:** robotics@omg.org
**Group Home Page (URL):**
http://robotics.omg.org/

## Jump up to TF or continue as SIG?




### Pros:

- **Publicity**
  **(authorized domain in OMG)**
- **Faster Process**
  **(recommend RFPs to DTC directly)**

### Cons:

- **Collaboration with TFs**
  **(relationship with sponsoring TF)**

**-> active interaction**
**(WG should keep discussion with related TFs)**
**(joint WG with other TFs)**

- **Quorum**

# Proposed Charter for Robotics DTF

**TC Meeting Date:** 9 December 2005
**Presenter:** Tetsuo Kotoku
**Group email:** robotics@omg.org
**Group Home Page (URL):** http://robotics.omg.org/

- **Mission:**
  The purpose of the Robotics Domain Task Force is to foster the integration of robotics systems from modular components through the adoption of OMG standards. To realize this purpose, we will:
  - Adopt and extend OMG technologies that apply to the specific domain of robotics systems where no current baseline specifications exist, such as MDA for Robotics. The object technology is not solely limited to software but is extended to real objects. This effort promotes the use of OMG technologies in various markets.
  - Promote mutual understanding between the robotics community and the OMG community.
  - Endeavor to collaborate with other organizations for standardization, such as the one for home information appliances, and make an open effort to increase interoperability in the field of robotics.
  - Coordinate with the appropriate OMG subgroups and the Architecture Board, for technology areas that overlap with other OMG Task Forces, to determine where the work will be accomplished.

- **Chair(s):**
  - **Hung Pham (RTI)**
  - **Tetsuo Kotoku (AIST)**
  - **Yun Koo Chung (ETRI)**

# Robotics

Date: Friday, 9th December, 2005
Reporting: Tetsuo Kotoku
Group URL: http://robotics.omg.org/
Group email: robotics@omg.org

## ➢Highlights from this Meeting:

### Robotics/SDO Joint Plenary (Tue. & Wed.):

- **9 RFI response presentations**
  (RTI, Systronix, SNU, ETRI * 2, NEC, NTT, ATR, Toshiba)
- **3 Special Talks**
  - James J. Odell (OMG Agent-PSIG) [robotics/05-12-05]
  - Joseph M. Jacob (OIS) [robotics/05-12-10]
  - Brian P. Gerkey (SRI International) [robotics/05/12-13]

### Joint Meeting with MARS-PTF (Thu.):

- RFI Summary report
- Robotics DTF Charter Proposal

# Robotics

Date: Friday, 9th December, 2005
Reporting: Tetsuo Kotoku
Group URL: http://robotics.omg.org/
Group email: robotics@omg.org

## ➢Deliverables from this Meeting:

- **Robotic Systems RFI [mars2005-06-12] deadline extension**
  (until Jan. 23rd 2006, 3 weeks before Tampa Mtg.)
- **Proposed Charter for Robotics DTF**

## ➢Future deliverables (In-Process):

- **RFI responses**

## ➢Next Meeting (Tampa, FL, USA):

- **RFP initial submission presentation**
  **(joint with MARS-PTF, SDO-DSIG)**
- **RFI response presentation**
- **RFI response summary and Chartering WGs**
- **Contact reports**

# Robotics-DSIG Meeting Minutes – DRAFT
# Burlingame, CA, USA
# (robotics/2005-12-19)

## Overview and votes

Following the issuance of the "Robotic Systems RFI" at the Boston meeting (June 2005) this meeting has mainly consisted of reviewing the responses to the RFI.
Although there have already been a large number of responses, it has been decided to extend the deadline for responding to the RFI to 3 weeks before the Tamp meeting. This decision has been made in order to allow some organizations who have expressed the will to respond to the RFI but had no time or learned only too recently about our activity.
Especially to be mentioned is the motion made by Tetsuo Kotoku for our activity to move from the status of a Domain Special Interest Group (DSIG) to the one of a Domain Task Force (DTF). The motion was approved by all voters. Along with the new status, the 3 new co-chairs of the Robotics DTF have been elected. From the next OMG meeting, the three co-chairs will be : Hung Pham (RTI), Yun Koo Chung (ETRI) and Tetsuo Kotoku (AIST).

## OMG Documents Generated

robotics/2005-12-01 Final Agenda (Tetsuo Kotoku)
robotics/2005-12-02 Atlanta Meeting Minutes [approved] (Olivier Lemaire)
robotics/2005-12-03 Opening presentation (Tetsuo Kotoku)
robotics/2005-12-04 Robotics-DSIG Roadmap (Tetsuo Kotoku)
robotics/2005-12-05 Special Talk: "Robots with Agents!" (James Odell)
robotics/2005-12-06 "On Robotics Middleware" (Hung Pham)
robotics/2005-12-07 "Self-Configuring Smart Java Robots through Plag and Play I/O Boards" (Bruce Boyes)
robotics/2005-12-08 "RSCA: Robot Software Communications Architecture" (Seongsoo Hong)
robotics/2005-12-09 "Human Interface of the Robotic System RFI" (Soo-Young Chi)
robotics/2005-12-10 "High Assurance Security and Safety for Robotics" (Joseph M. Jacob)
robotics/2005-12-11 "Standardization on Interfaces to Robot Devices " (Seung-Ik Lee)
robotics/2005-12-12 "Research and Development of Personal Robot in NEC" (Yoshihiro Fujita)
robotics/2005-12-13 "The Player/Stage/Gazebo project: Open Source tools for robotics research" (Brian P. Gerkey)
robotics/2005-12-14 "Network Robot Platform for Information Sharing" (Ken-ichiro Shimokura)
robotics/2005-12-15 "Human Robot Interaction in Network Robots" (Norihiko Hagita)
robotics/2005-12-16 "Network Robots Standardization Activity in Japan" (Miwako Doi)
robotics/2005-12-17 Proposed Charter for Robotics Domain Task Force
robotics/2005-12-18 : Atlanta Robotics DSIG DTC Report Presentation (Tetsuo Kotoku)
robotics/2005-12-19 : Meeting Minutes (Olivier Lemaire, Seung-Ik Lee, Claude Baudoin)

## Agenda

<u>05 December, Monday</u>
15:00-17:00 – Steering Committee of Robotics DSIG

<u>06 December, Tuesday</u>
13:00-13:10 – Welcome and Review Agenda
13:10-14:10 – "Introduction to the Agent-SIG activities" - James J. Odell (OMG Agent PSIG)
14:10-15:00 – "Response to RFI from Real-Time Innovation" – Hung Pham (RTI)
15:20-16:10 – "Response to RFI from Java.net" - - Bruce Boyes (Systronix)
16:10-17:00 – "Response to RFI from SNU - The Robot Software Communications Architecture (RSCA) : Embedded Middleware for Networked Service Robots" - Seongsoo Hong (Seoul National Univ.)

17:00-17:50 – “Capabilities: Human Interface of the Robotic Systems RFI” - Soo-Young Chi (ETRI)
07 December, Wednesday
9:00-10:00 – “High Assurance Security and Safety for Robotics” - Joseph M. Jacob (Objective Interface Systems)
10:20-11:10 – “Hardware Abstraction to the Robotic Systems RFI” - Seung-Ik Lee (ETRI)
11:10-12:00 – “Response to RFI from NEC” - Yoshihiro Fujita (NEC)
14:00-14:40 – "The Player/Stage/Gazebo project: Open Source tools for robotics research" - Brian P. Gerkey (SRI International)
14:50-15:30 – “Network Robot Platform for Information Sharing” - Ken-ichiro Shimokura (NTT)
15:30-16:10 – “Human Robot Interaction in Network Robots” - Norihiko Hagita (ATR)
16:10-16:50 – “Network Robots Standardization Activity in Japan” - Miwako Doi (Toshiba)
17:00-17:20 – Chartering Robotics Domain Task Force and voting
17:20-17:40 – Next Meeting Agenda Discussion, etc Robotics/SDO Closing session
17:40 Adjourn

## Minutes

**06 December, Tuesday**

**Opening Presentation (Tetsuo Kotoku)**
Tetsuo Kotoku provided a brief overview of the Atlanta Minutes.
***Action:***
The minutes were approved.
http://www.omg.org/docs/robotics/05-12-02.pdf

Tetsuo Kotoku reviewed the SDO-DSIG/Robotics DSIG roadmap and the agenda.
http://www.omg.org/docs/robotics/05-12-03.pdf
http://www.omg.org/docs/robotics/05-12-04.pdf

**Special Talk: "Robots with Agents!"**
Jim Odell defined an agent as "an autonomous entity that can adapt and interact with its environment" and commented on the concepts of autonomy, interaction, and adaptation, as they apply to software agents. His point was that robots are very much like agents, and that a number of the concepts developed in the OMG's Agents SIG would be relevant to the work of the Robotics SIG. For example, agents can coordinate their actions using different processes: negotiation, competition, persuasion, and deliberation.

**Presentation – RFI Response: "On Robotics Middleware" - Hung Pham (RTI)**
RTI provides software tools and services to developers of complex, distributed control systems (including robotic systems).
Hung Pham presented a "robotic technology vertical layer" model, with 7 levels:
7 = application layer: integrated subsystem with user interface, mode switching, automatic detection and self-organization capability
6 = domain layer: a subsystem composed of engineering domain-specific software
5 = middleware layer: application framework
4 = data layer: platform-independent data and interface representations
3 = operating system layer
2 = hardware abstraction layer (device drivers, etc.)
1 = physical layer
He proposes that the biggest opportunities for standardization are in levels 5 and 4. Middleware services are "mundane and transparent to the end user" and are not a product differentiator; therefore vendors should be amenable to their standardization.

Proposed requirements for robotics middleware:
R1: offer a platform- and vendor-independent standard interface to which a system integrator can write
R2: extendable components, in the OO sense of "programming by difference"
R3: real-time executable with minimum operating system support
R4: real-time "interact-able": behavior can be observed and changed at runtime, there need to be integrated log/playback/debug capabilities.
Three technical characteristics of a standard should be met:
Retain component definitions and connection concepts from UML and CCM
Support data-centric designs and communications
Be agnostic with regard to underlying communication mechanisms (CORBA, DDS, etc.)

**Presentation - RFI Response: "Self-Configuring Smart Java Robits through Plug-and-Play I/O Boards" - Bruce Boyes (Systronix)**

Bruce Boyes explained the importance of self-descriptive hardware (especially for swarm robots) and introduced the concept of XML-tagging. By assuming that most hardware (sensors & actuators) is I/O based, the idea is to develop a generic I/O driver common to all devices and differentiate the functionalities provided by the hardware not in the base code but by using standardized XML tags (similarly to the IEEE1451 standard) describing these functionalities at the I/O pin level. The recent drop of the price of memory chips makes this solution viable and an example of application of the concept using the Systronix JCX board was presented.

http://www.omg.org/docs/robotics/05-12-07.pdf

**Presentation - RFI Response: "RSCA: Robot Software Communication Architecture" by Seongsoo Hong of Seoul National University**

The National Robotics Project in Korea is predicated on the assumption, made at the level of the Minister of Information and Communication, that home service robots will become ubiquitous in homes in the near future. Their approach is to develop affordable robots with limited capabilities and distribute the task processing to high performance remote servers.
The main requirements for the software platform supporting the development of such robotic application are:
- Distributed nature of hardware and software
- Component-based software development
- Dynamic deployment and reconfiguration
- Real-time and QoS capabilities

After a detailed presentation of RSCA, the speaker did address the mapping of RSCA to the robotics technology and where RSCA should be extended to fulfill all the requirements for the development of a robotics application

http://www.omg.org/docs/robotics/05-12-08.pdf

**Presentation - RFI Response: "Human Interface of the Robotic System RFI" by Soo-Young Chi**

This presentation by ETRI specifies the "user recognition component of the human-robot interface."
The presenter proposes an interface comprising three main high level primitive functions in a user recognition component: - Enroll, Verify and Identify. After describing different possible distributed implementations strategies (actual recognition process on client or server side), the main data structures used during the recognition process were presented.

http://www.omg.org/docs/robotics/05-12-09.pdf

**06 December, Tuesday**

**Informative Talk : "High Assurance Security and Safety for Robotics" - Joseph M. Jacob (Objective Interface Systems)**
The speaker first listed the requirements related to security and safety in robotics applications as well as already existing standards. He then introduced the concept of MILS (Multiple Independent Levels of Security/Safety) in which the code running in privileged mode is reduced to a minimum kernel (dealing only with security policies) which then becomes mathematically verifiable, non-bypassable, always invoked and tamper-proof
The presentation then covered a tool called Partition Communication System (PCS), developed for military/intelligence needs, which applies the MILS principles

http://www.omg.org/docs/robotics/05-12-10.pdf

**Presentation - RFI Response: "Hardware Abstraction to the Robotic Systems RFI" - Seung-Ik Lee (ETRI)**
In order to facilitate interoperability between robotic systems, ETRI has developed an abstract "Common Robot Interface Framework" (CRIF) which clearly defines types of hardware devices (wheels, sensors, cameras, head, ...), interfaces (CRIF includes a Robot API Layer with about 50 APIs), data types and coordinate systems (local and global). The framework makes a clear distinction between the Application Programmers APIs (which are hardware independent) and the Robot Platform Developers APIs (which are application independent). The communication mechanism between the 2 sets of APIs is interchangeable and transparent to the developers (ETRI provides a socket based implementation).
Dr. Lee said that one challenge is going to be that there are many types of devices in the robotics area, and more will appear, so it will be impossible to avoid custom interfaces. Olivier Lemaire (JARA) suggested that a useful first step would be to create a UML Profile for Robotics.

http://www.omg.org/docs/robotics/05-12-11.pdf

**Presentation - RFI Response: "Response to RFI from NEC" - Yoshihiro Fujita (NEC)**
The presenter relates on the difficulty of developing a robotic application (not a robot itself) of which users will not get bored after just a few weeks. To overcome this difficulty, more developers from different backgrounds should be involved (even the end-user itself) in the development, which implies the availability of an easy to use application development platform as well as customizable graphical tools. After describing the two fundamental concepts of robotic application programming (Actions and Behaviors), the presenter introduced the framework developed by NEC to support the development of Robotic Applications and based on XML interface definition and scripting.

http://www.omg.org/docs/robotics/05-12-12.pdf

**Informative Talk : "The Player/Stage/Gazebo project: Open Source tools for robotics research" - Brian P. Gerkey (SRI International)**
The SRI team has developed, and placed in open-source, a robotic control library called Player, which contains about 90 drivers for many different devices. They also have written a 2-D and a 3-D simulator, which respond to the same commands as the real robots. This makes this library appealing for development, in particular in universities, since it is possible to move back and forth seamlessly from simulation to actual execution. One of the machines that can be controlled is the robot version of the Segway transporter.

http://www.omg.org/docs/robotics/05-12-13.pdf
http://playerstage.sourceforge.net

**Presentation - RFI Response: "Network Robot Platform for Information Sharing" - Ken-ichiro Shimokura (NTT)**
The speaker introduced the 4 year-long Network-robot Project funded by the Ministry of Internal Affairs and Communication of Japan and which mission is to develop the core technologies necessary to a successful integration of the robotic technology into everyday's life, focusing on human like behaviors and interaction with humans. The 4 main themes of the project are :
- Human Behavior Recognition
- Platform-mediated communication
- Service allocation and execution
- Human-robot Interaction

After describing the 4 themes, especially "Platform mediated Communication", for which the Field Data Markup Language (FDML) was used as a mean to abstract information provided by a robot (a pointer to more detailed information was unfortunately not provided), a video showing the early results of the project was played.

http://www.omg.org/docs/robotics/05-12-14.pdf

**Presentation - RFI Response: "Human Robot Interaction in Network Robots" - Norihiko Hagita (ATR)**
In the context of social robots and after presenting several cases of Human-Robot interaction and results of investigations made by ATR regarding that matter, the speaker expressed the need of the standardization of :
- basic behavior languages for different-type robots
- Software modules for social intelligence
- Interactive primitives and corpus from various sensor data

http://www.omg.org/docs/robotics/05-12-15.pdf

**Presentation - RFI Response: "Network Robots Standardization Activity in Japan" - Miwako Doi (Toshiba)**
After describing the Robotic Technology standardization effort in Japan and especially within the Network Robot Project, the speaker presented her views on the way to fulfill users' needs in the context of distributed service oriented robotic architecture, as well as the requirements and the difficulties met to develop such a system. A call was then made for leveraging the work that has already been done in the domains of ubiquitous computing and network sensors by trying to bridge the existing standards. Finally a case study cooperation between human and networked robots was described.

http://www.omg.org/docs/robotics/05-12-16.pdf

**Chartering Robotics Domain Task Force and voting**
Tetsuo Kotoku, from AIST, presented the roadmap for the SIG. There is an RFI in progress, for which 4 responses have been received, and at least two more are expected. To allow these additional responses, a motion was adopted to extend the reply deadline to 1/23/2006, three weeks before the Tampa meeting. Following the RFI, there seem to be four potential groups for RFP discussions on the roadmap:
1 - robot middleware for controllers
2 - robot middleware for specific applications
3 - robot middleware for common services
4 - robot middleware for common data structures

A quick survey showed that for each potential working group, at least 6 to 7 participants would be interested in pursuing the activity, which was encouraging.

This was followed by a discussion concerning the Robotics DSIG becoming a Task Force. Tetsuo Kotoku described the pros and cons of such a move :
Pros : More visibility and publicity; Faster process of adoption

Cons : Collaboration with other Task Force no more necessary -> risk of isolation : need to actively maintain relations to other task forces (joint meetings ?); when voting a quorum is necessary

A motion was then made by Tetsuo Kotoku for our activity to move from the status of a Domain Special Interest Group (DSIG) to the one of a Domain Task Force (DTF). The motion was approved by all voters. The charter of the Task Force has been adopted (same as the DSIG), the word “adapt” in the first bullet being replace by “adopt”

Following, a call for volunteers to become the new co-chairs of the task force has been made. Three volunteers stood up (Hung Pham (RTI)-Tetsuo Kotoku(AIST)-YunKoo Chung (ETRI)) and have been approved unanimously.

http://www.omg.org/docs/robotics/05-12-17.pdf

Finally, a call for volunteers for being the official OMG contact with other organizations has been made. Miwako Doi (Toshiba) offered to become our contact with NRF (Network Robot Forum , http://www.scat.or.jp/nrf/English/).

**Next Meeting in Tampa**
Monday : Steering Committee
Tuesday-Wednesday : Robotics DSIG plenary

- RFP initial submission presentation (joint with MARS-PTF, SDO-DSIG)
- RFI response presentation
- RFI response summary and Chartering WGs
- Contact reports

**ADJOURNED @ 17:40PM**

## Participants (Sign-in)

05 December, Monday (15 participants)

- Makoto Mizukawa (Shibaura Institute of Technology)
- Yun Koo Chung (ETRI)
- Seongsoo Hong (SNU)
- Saehwa Kim (SNU)
- Hung Pham (RTI)
- Claude Baudoin (Schlumberger)
- Seung-Ik Lee (ETRI)
- Olivier Lemaire (JARA)
- Carlo Cloet (RTI)
- Gerardo Pardo (RTI)
- Masayoshi Yokomachi (NEDO)
- Takashi Suehiro (AIST)
- Noriaki Ando (AIST)
- Tetsuo Kotoku (AIST)
- Jaesoo Lee (SNU)

06 December, Tuesday (28 participants)

- Miwako Doi (Toshiba)
- Takashi Suehiro (AIST)

- Duane Clarkson (John Deer)
- Saku Egawa (Hitachi)
- Makoto Mizukawa (Shibaura Institute of Technology)
- Noriaki Ando (AIST)
- Olivier Lemaire (JARA)
- Yoshihoro Fujita (NEC)
- Bruce Boyes (Systronix)
- Adam Howell (Lockheed Martin)
- John Hogg (Zeligsoft)
- Roger Burkhart (John Deer)
- Masayoshi Yokomachi (NEDO)
- Rick Warren (RTI)
- Yun Koo Chung (ETRI)
- Seung-Ik Lee (ETRI)
- Soo-Young Chi (ETRI)
- Takashi Tsuboushi (University of Tsukuba)
- Hung Pham (RTI)
- Henri Choi (RTI)
- Seongsoo Hong (SNU)
- Saehwa Kim (SNU)
- Jaesoo Lee (SNU)
- Ken Shimokura (NTT)
- Roy Bell (Raytheon)
- Virginie Watine (Thales)
- Gerardo Pardo (RTI)
- Tetsuo Kotoku (AIST)

07 December, Wednesday (28 participants)

- Takashi Suehiro (AIST)
- Makoto Mizukawa (Shibaura Institute of Technology)
- Masayoshi Yokomachi (NEDO)
- Seung-Ik Lee (ETRI)
- Noriaki Ando (AIST)
- Bruce Boyes (Systronix)
- Claude Baudoin (Schlumberger)
- Jaesoo Lee (SNU)
- Joseph Jacob (Objective Interface)
- Yoshihoro Fujita (NEC)
- Stan Schneider (RTI)
- Tetsuo Kotoku (AIST)
- Yun Koo Chung (ETRI)
- Soo-Young Chi (ETRI)
- Miwako Doi (Toshiba)
- Saku Egawa (Hitachi)
- Ken Shimokura (NTT)
- John Hogg (Zeligsoft)
- Adam Howell (Lockheed Martin)
- Rick Warren (RTI)
- Olivier Lemaire (JARA)

- Saehwa Kim (SNU)
- Takashi Tsuboushi (University of Tsukuba)
- Roy Bell (Raytheon)
- Juergen Boldt (OMG)
- Regis Vincent (SRI International)
- Brian P. Gerkey (SRI International)
- Hung Pham (RTI)

Prepared and submitted by Olivier Lemaire with the assistance of Seung-Ik Lee, Claude Baudoin, Masayoshi Yokomachi and Makoto Mizukawa.