Composition-based Interaction Design for Adaptable Distributed Software Systems

Kenji Tei
Assistant Professor, National Institute of Informatics

Self-Introduction : Kenji Tei

- Assistant professor at NII
  - tei@nii.ac.jp
  - http://researchmap.jp/teikenji
- Research Interests
  - adaptive software system
  - model-driven development, software architecture
  - networked embedded systems
    - especially wireless sensor networks
    - recently IoT, wireless control systems, and robots

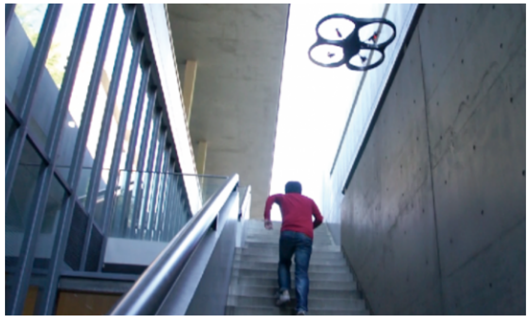
XAC PROJECT

GRACE Center

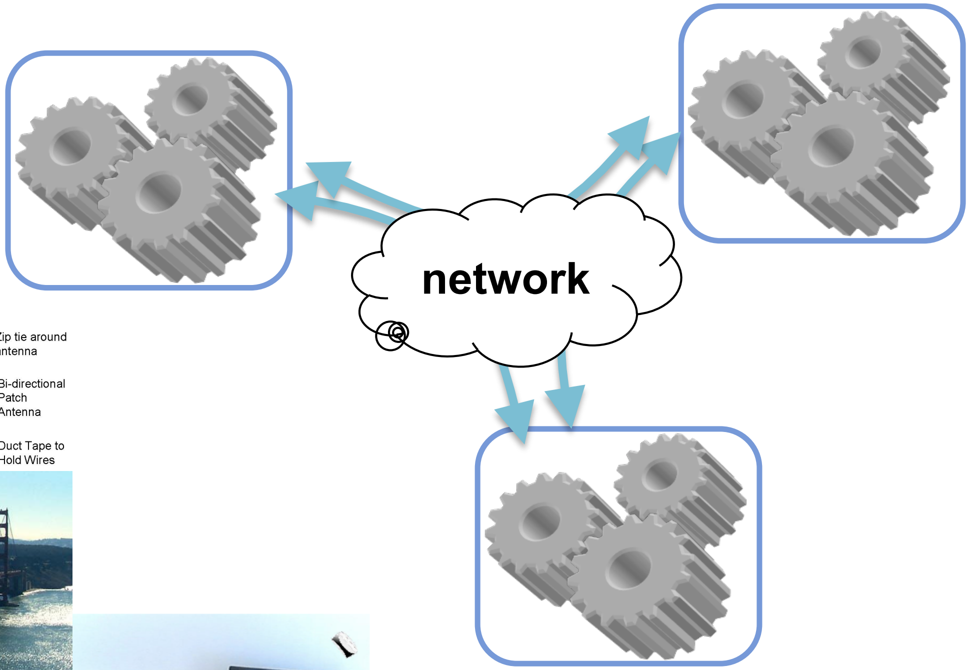
ClouT

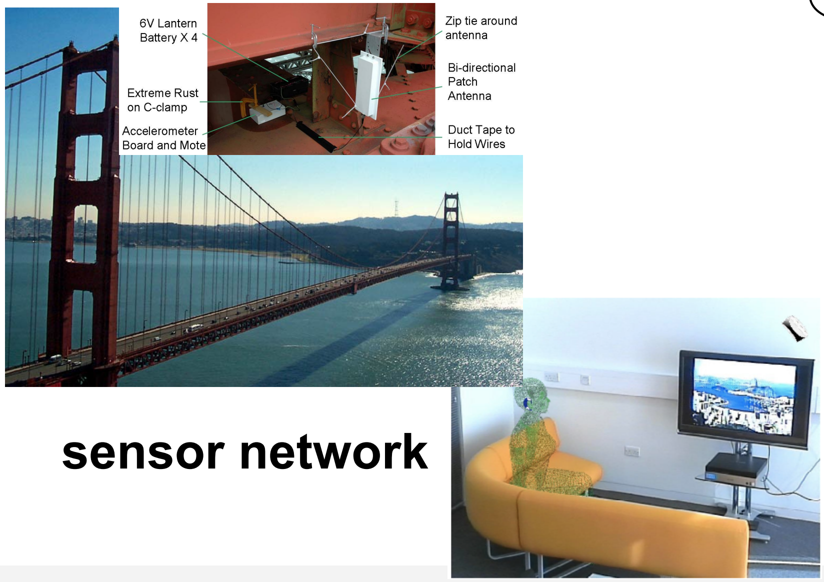
Distributed Software System

Service-oriented (cloud) system

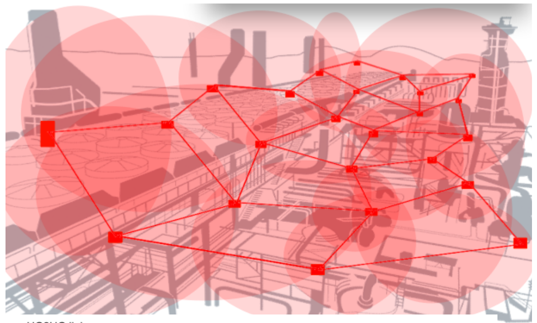
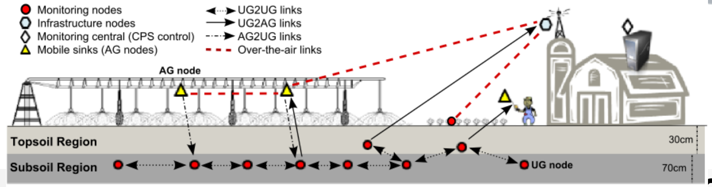


robots





sensor network

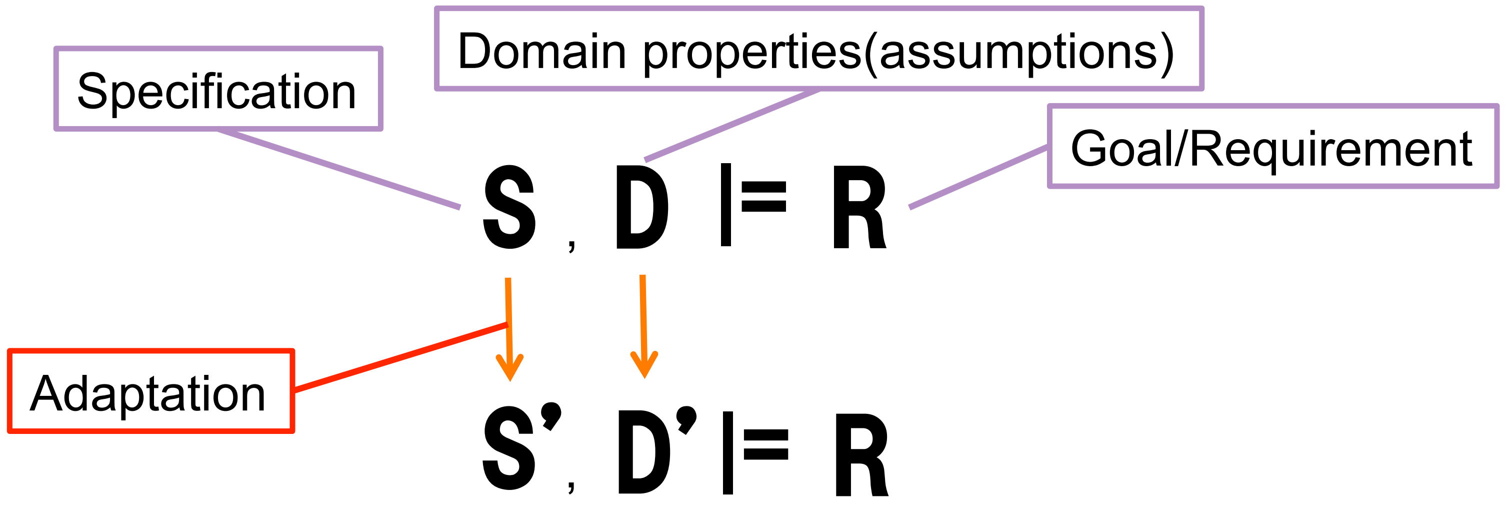
networked control system

Adaptive Systems

self-adaptation

systems that are able to **modify their behavior and/or structure** in response to **their perception of the environment** and **the system itself**, and **their requirements**

Rogério de Lemos, et.al., Software Engineering for Self-Adaptive Systems: A Second Research Roadmap, SEAMS2011.



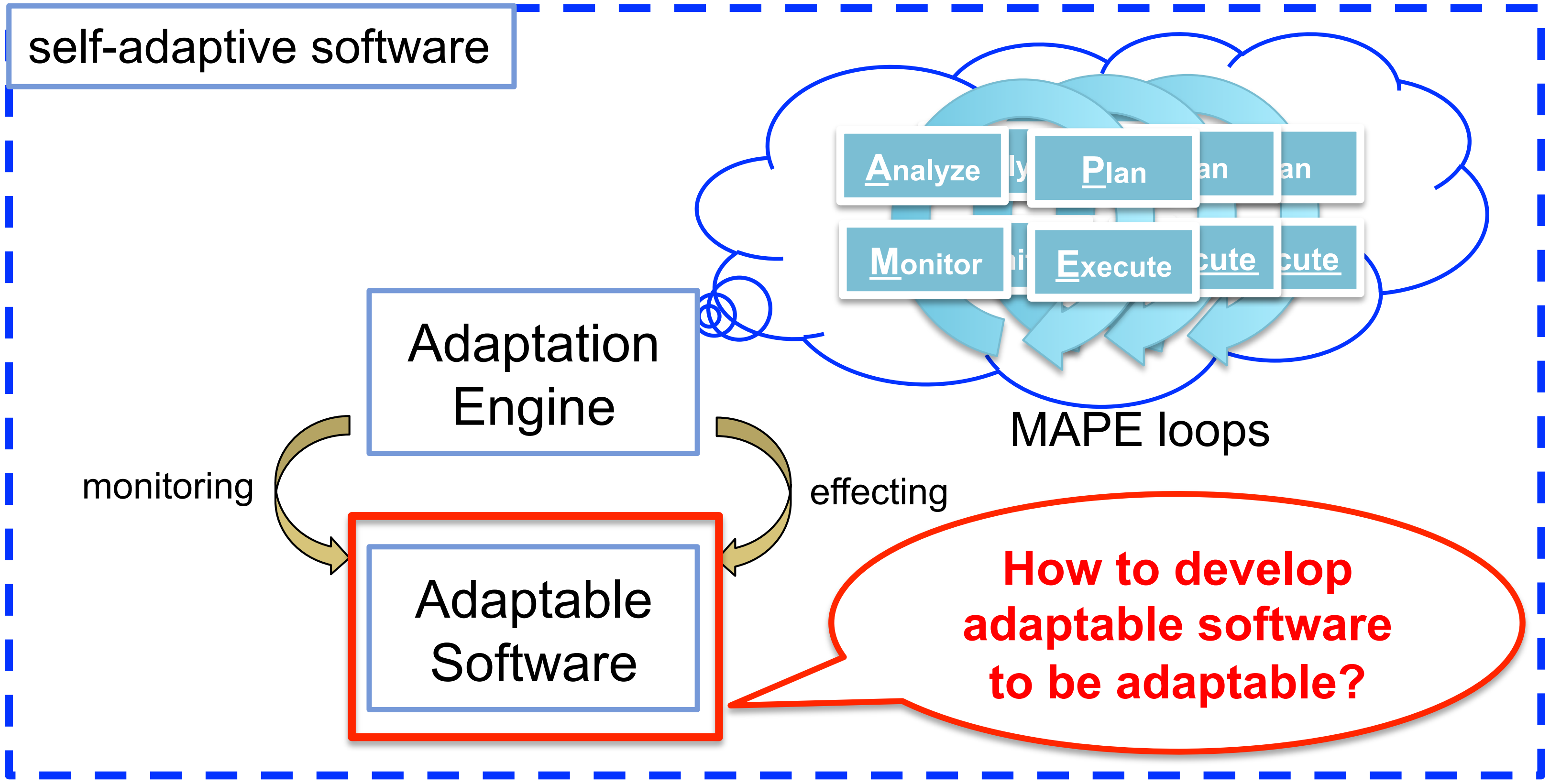
$$S, D \models R$$

$$S', D' \models R$$

Modify **S** in response to changes in **D** to satisfy **R**

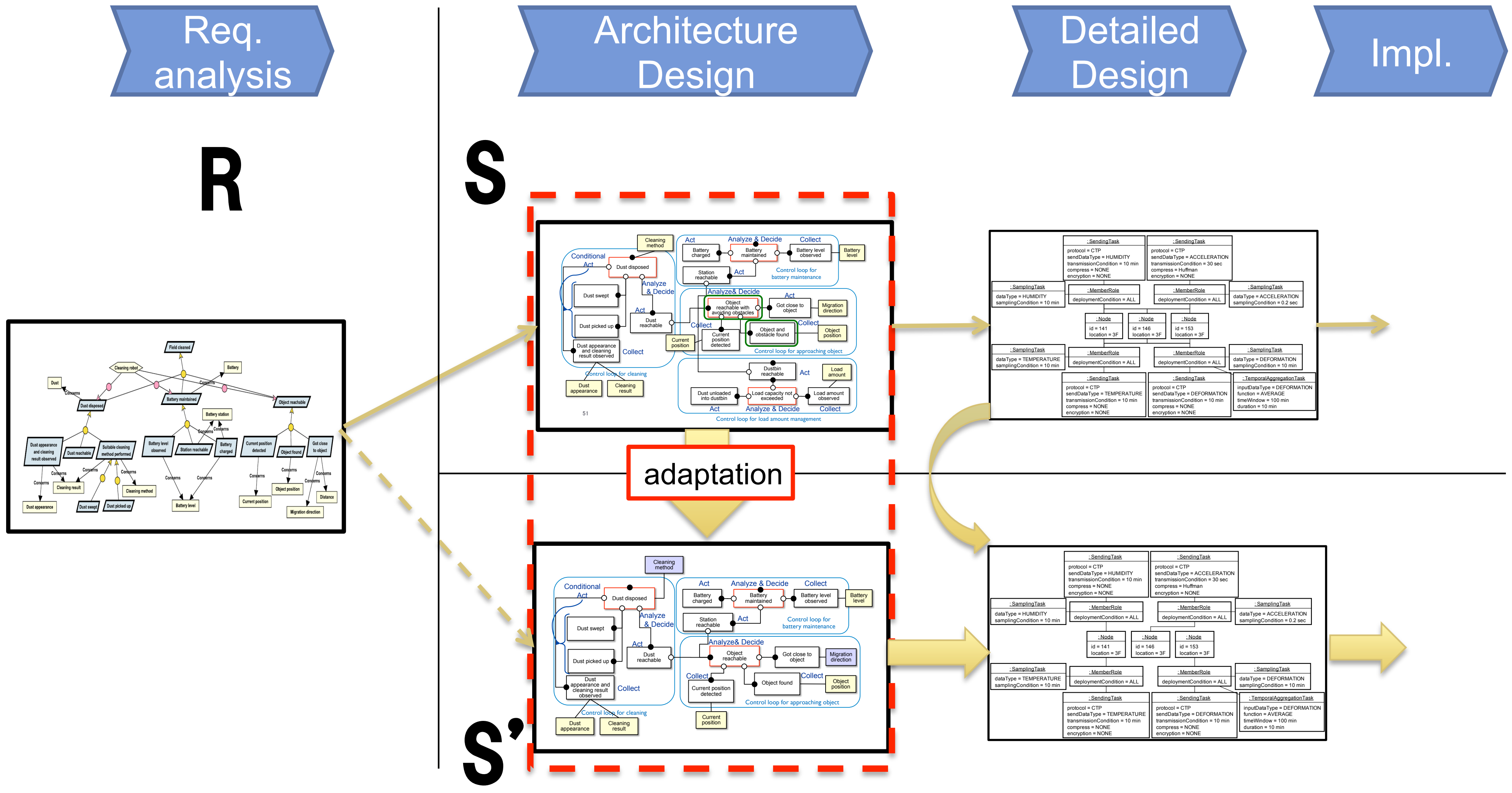
- off-line : Adaptive maintenance
- on-line : Self-adaptation

Our Focus



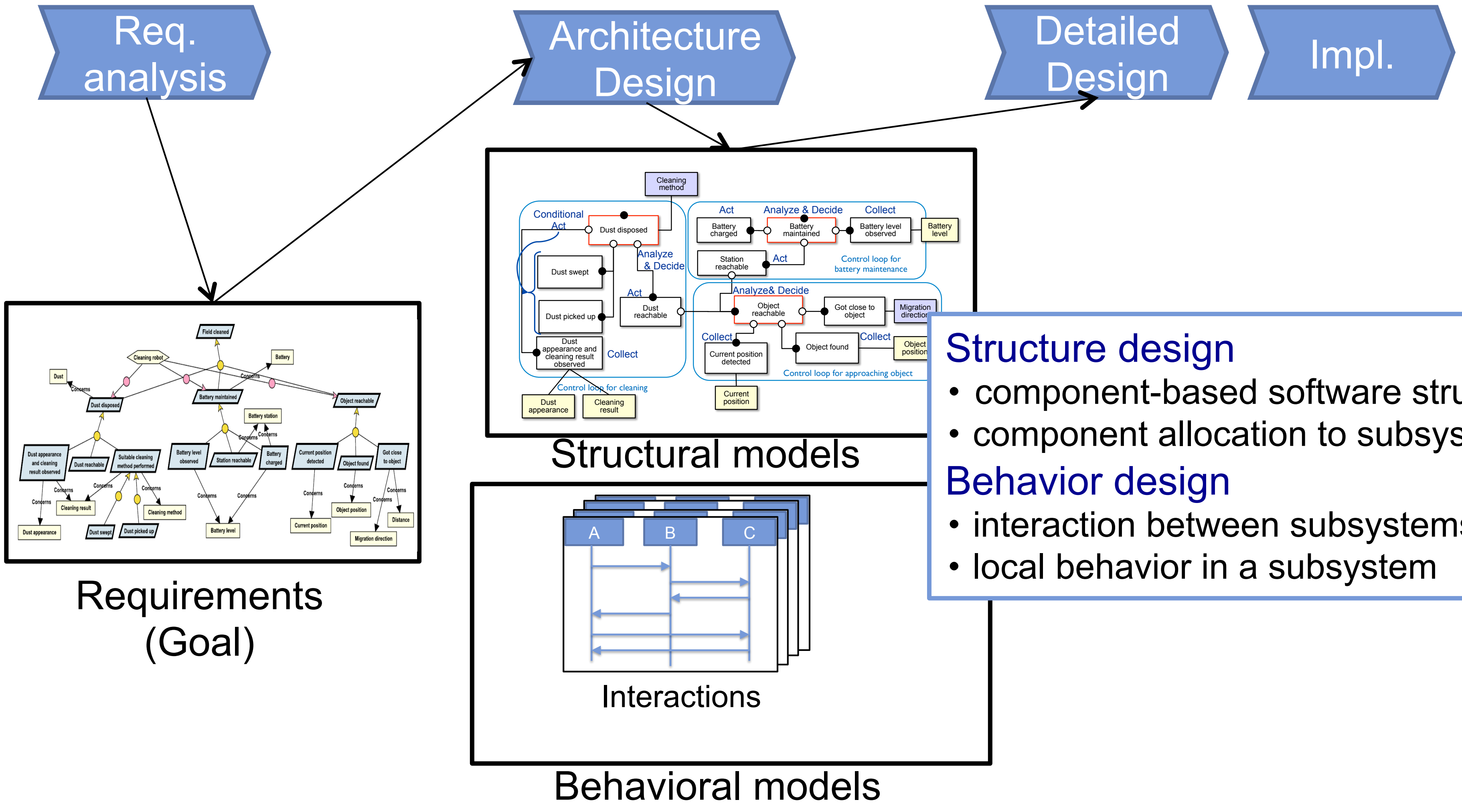
Adaptable software should be developed to support one or more solutions
All solutions should be tightly related to requirements

Adaptable Software Development



Traceability links between R and S should be maintained
Changes in S for adaptation should be localized

Architecture Design for Distributed Software

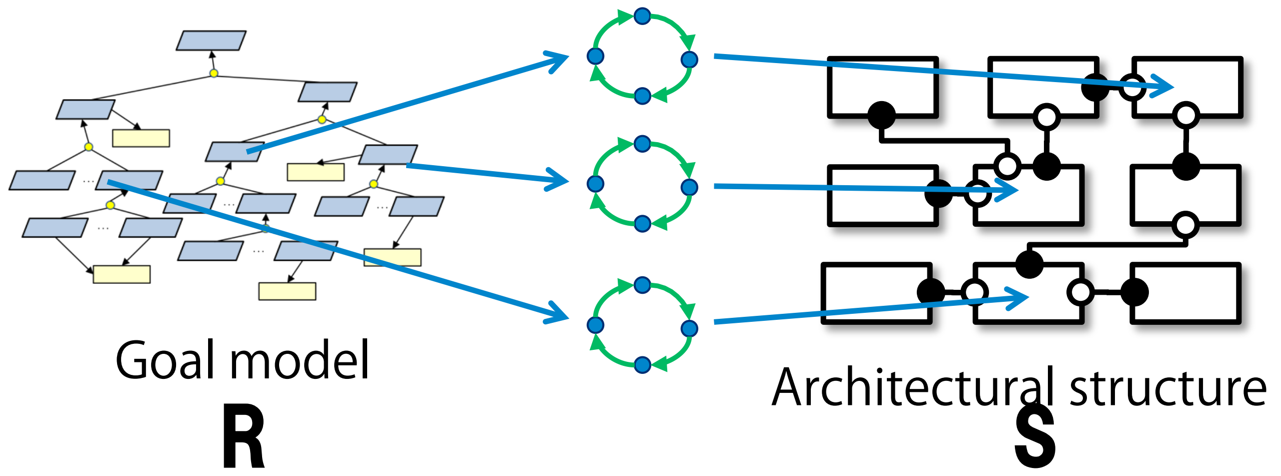


Structure design
- component-based software structure
- component allocation to subsystems

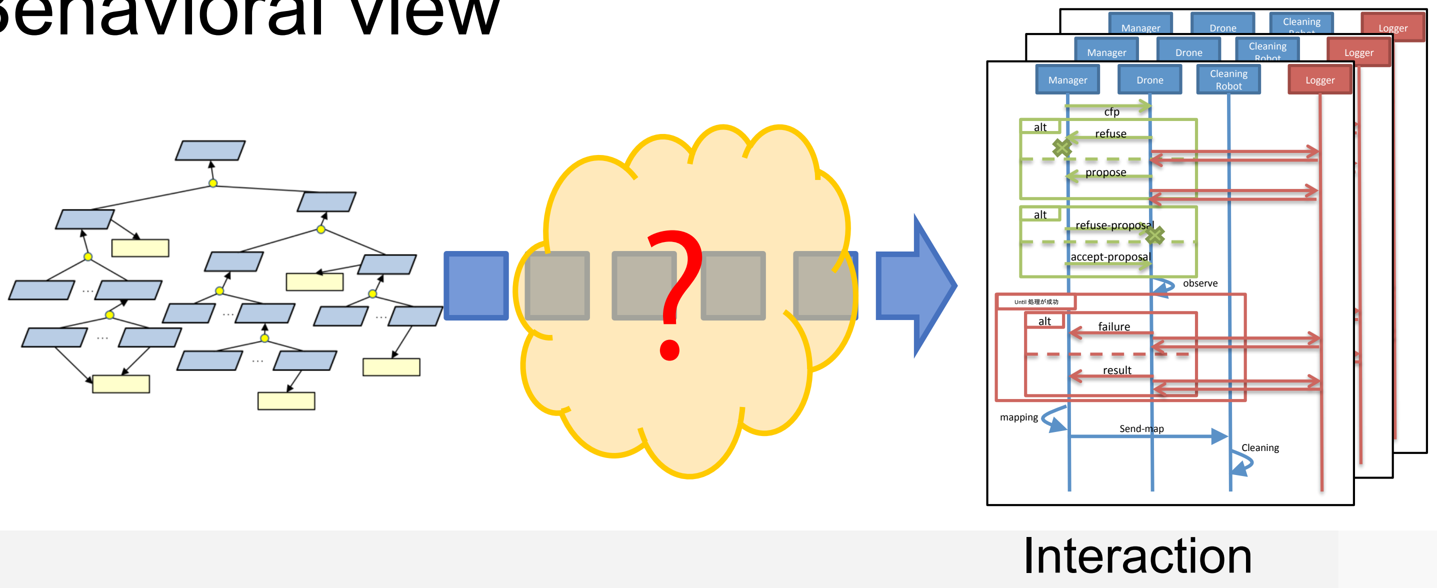
Behavior design
- interaction between subsystems
- local behavior in a subsystem

Traceablity between R and S to Localize Changes

- Structural view



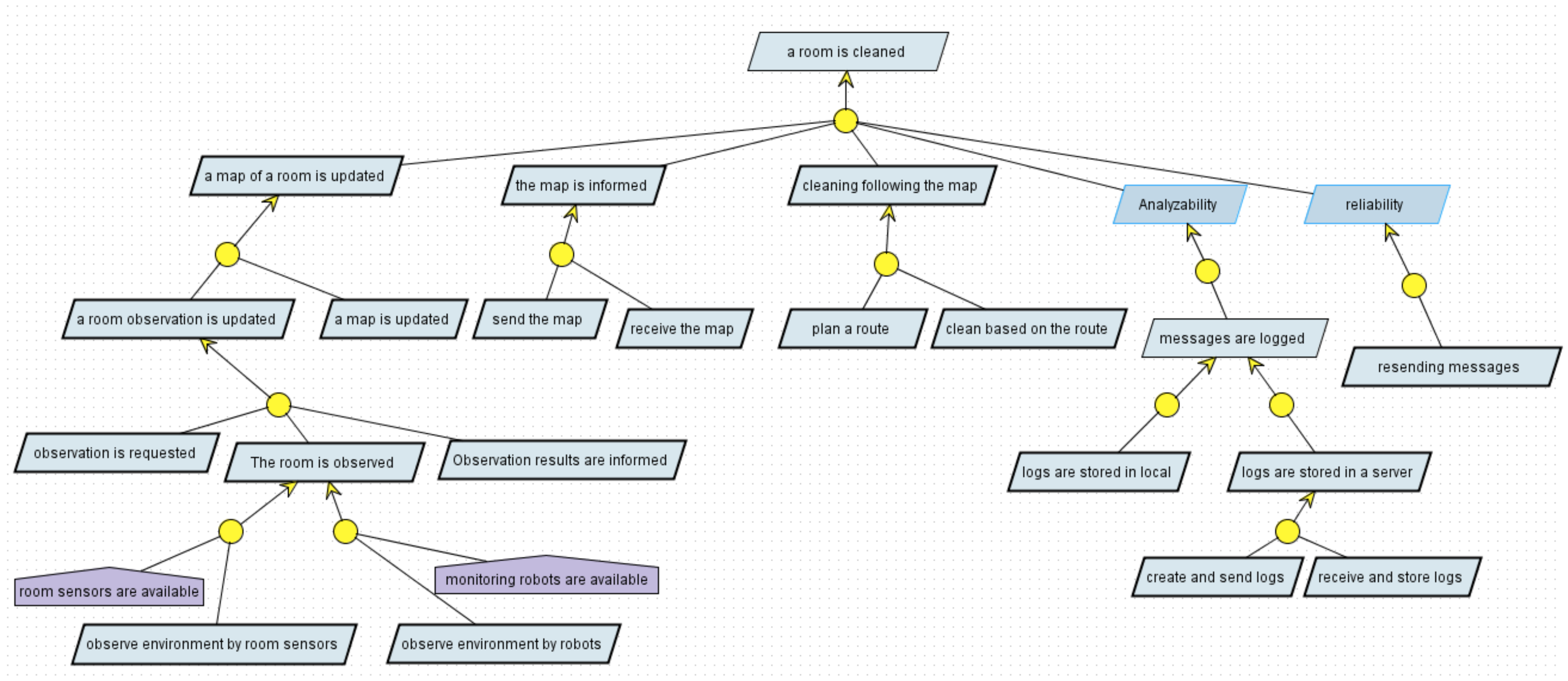
- Behavioral view



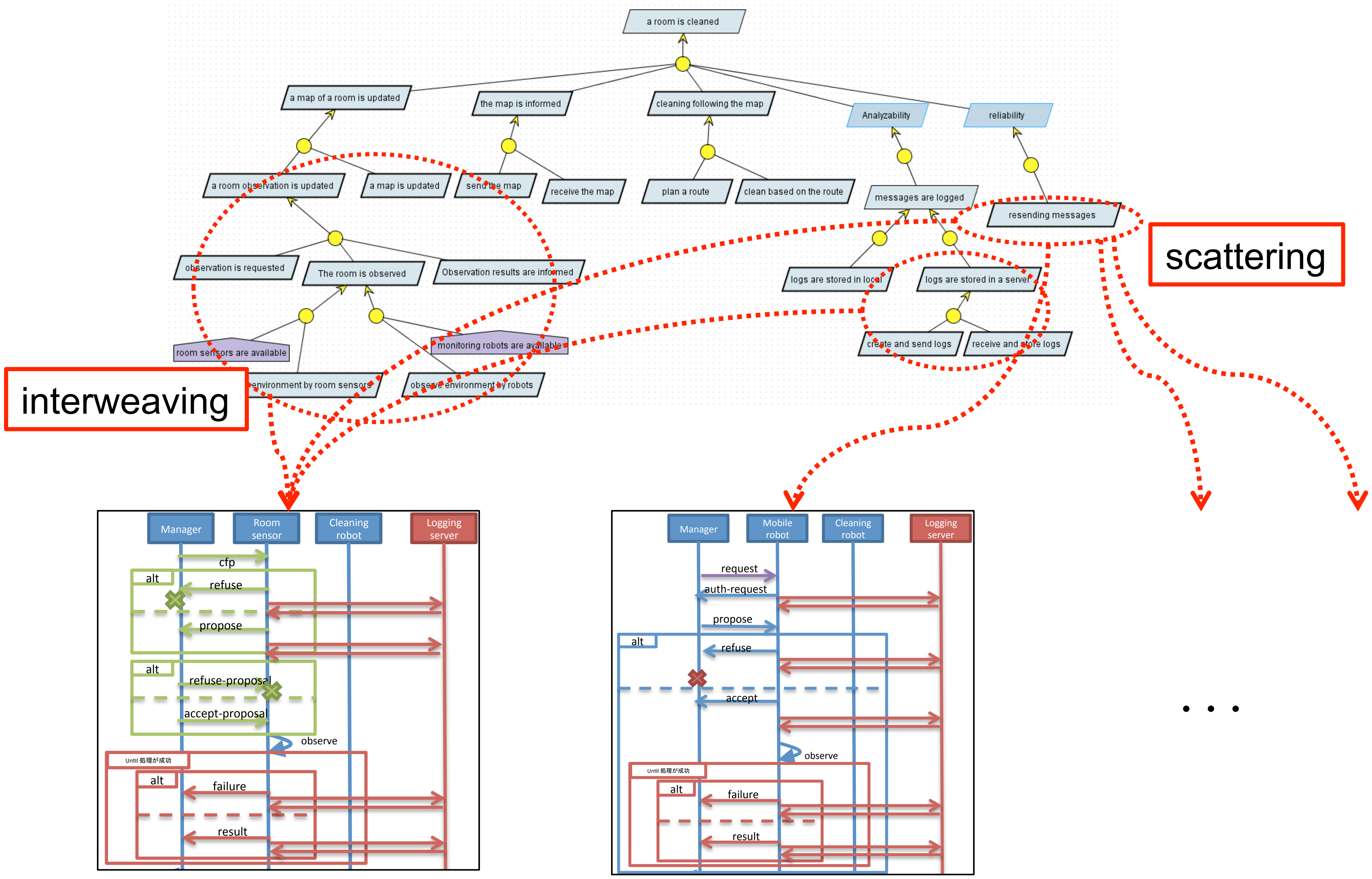
Importance of Interaction Design

- Interaction is usually designed to satisfy one or more requirements
  - especially, for networked embedded system, reliability, performance, security, etc..., will be affected by interaction design
- One interaction is related to many concerns
  - centralized/decentralized coordination, logging, compression, encryption, retransmission, etc…

Room Cleaning System Scenario

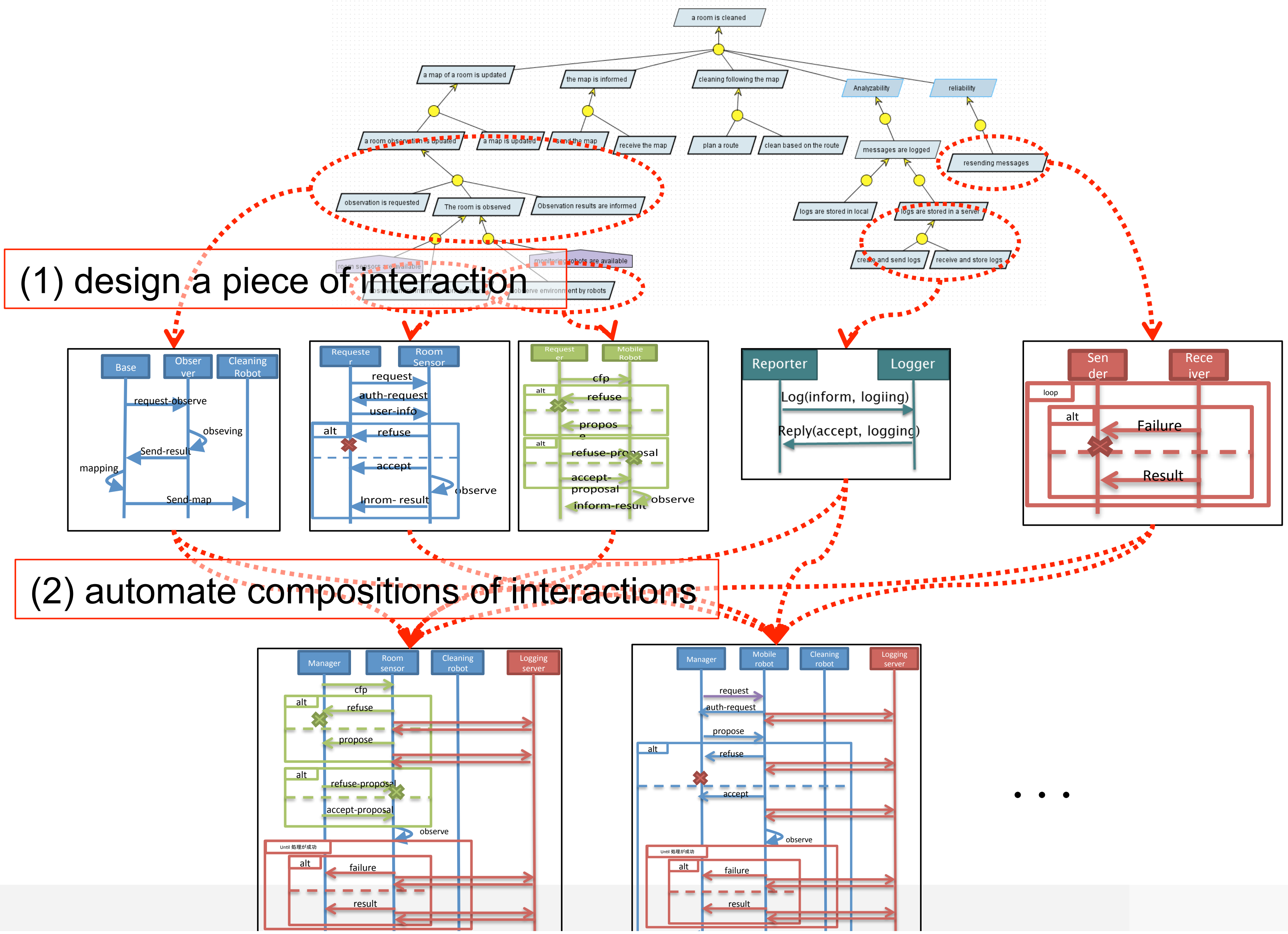


Less Traceability between R and S(Interaction)



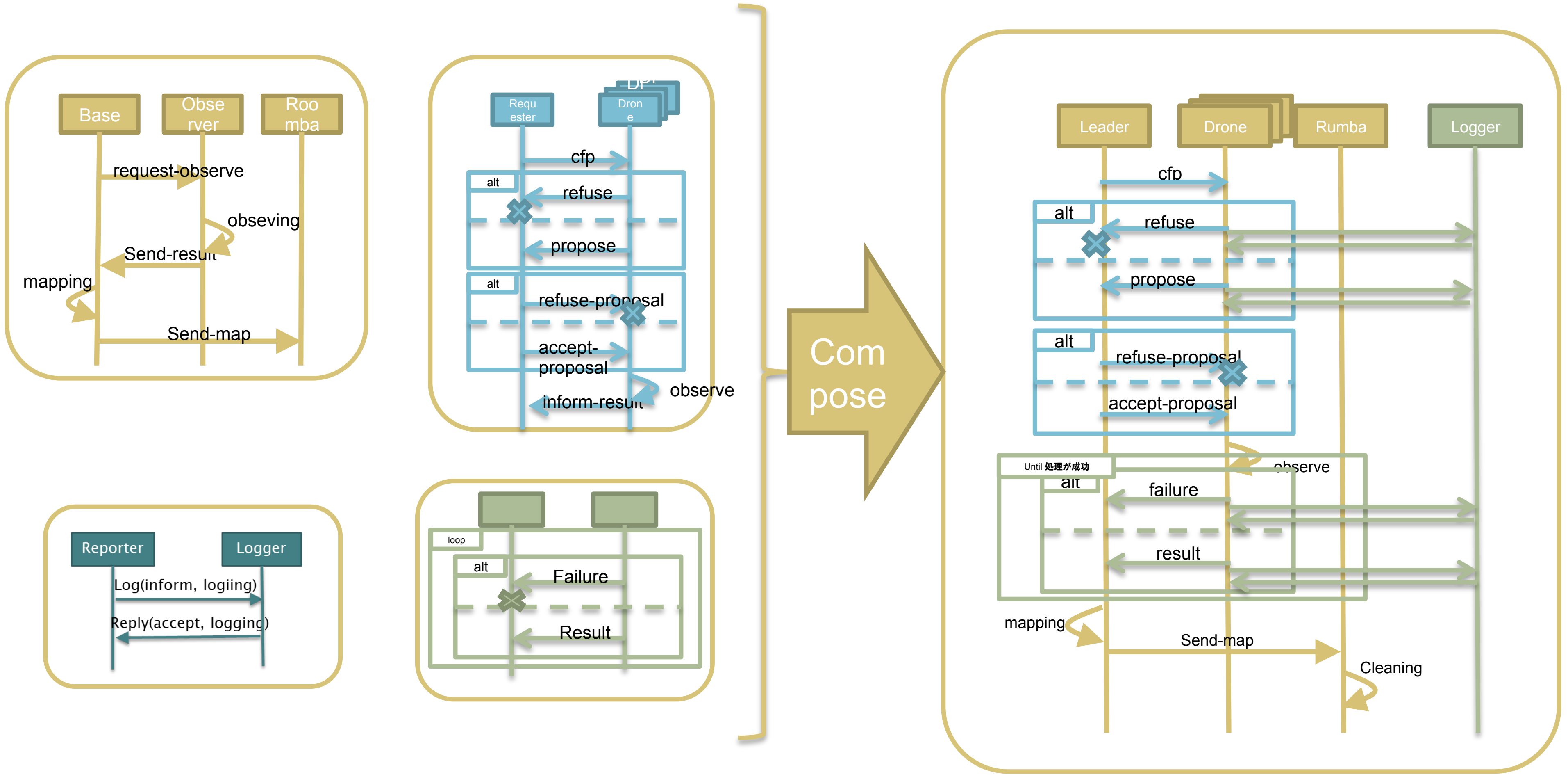
hard to localize changes for adaptation

Our Approach : Composition-based Interaction Design



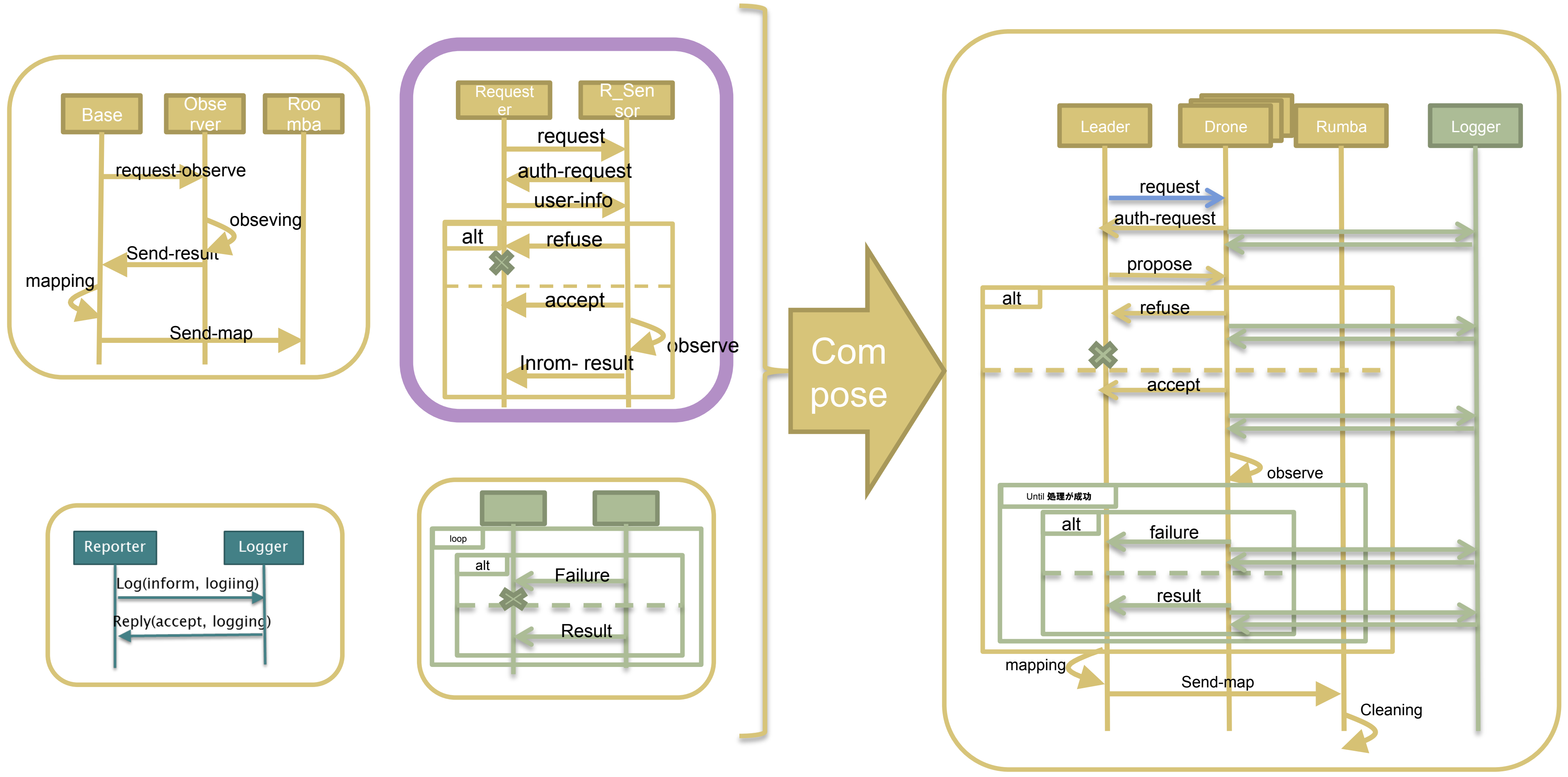
Interaction Composition

- Complete interaction can be achieved by composing sub-interactions corresponding to a requirement



Interaction Composition

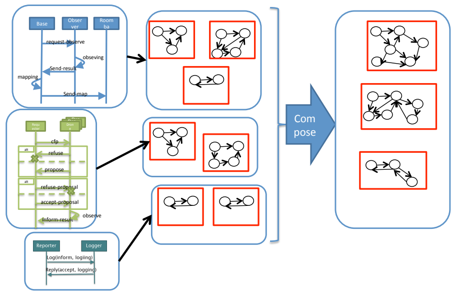
- Complete interaction can be achieved by composing sub-interactions corresponding to a requirement



Ongoing Work : Composition-based Interaction Design

- Identify specifications of interaction pieces from goal model
  - goal elaboration process to clarify requirements for interaction pieces,
- Compose interactions
  - bind roles and data, and merge message sequences
  - explore and find a composition satisfying all their constraints

    R. Takahashi, F.Ishikawa, K.Tei, and Y.Fukazawa: Intention-based Automated Composition Approach for Coordination Protocol, ICWS2013.
- Detailed behavior design and implementation should also be changed according to the changed interaction
  - local behaviors and implementation are also designed to be composable



Summary

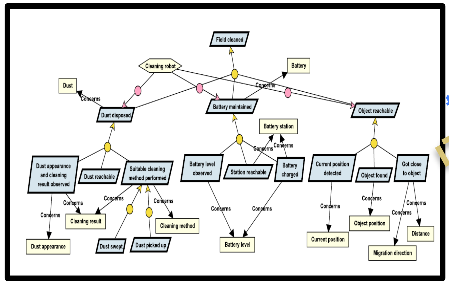
- Interaction is high level design decision about behavior of distributed software system
  - one or more solutions exist for one requirement
- Adaptable software should be designed to support one or more interactions for each requirement
  - however, traceability link between req. and interactions is unclear
- To clarify traceability between requirements and interactions, we adopt composition-based approach
  - one requirement corresponds to one interaction piece
  - complete interaction can be achieve by composing interaction pieces
  - easy to change solutions of a certain requirement

Overview of works in NII

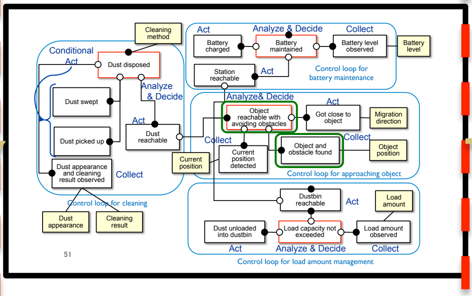
Req. analysis → Architecture Design → Detailed Design → Impl.

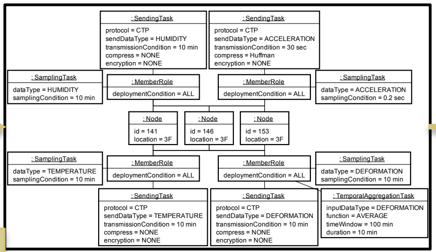
•Exploration of adaptation space
Fuyuki Ishikawa

Adaptation space analysis



traceability link





•Designing Self-adaptive System using Control Loops
Shinichi Honiden

adaptation

•Putback-based Bidirectional Programming
Zhenjiang Hu

•Bidirectional Graph Transformation Infrastr
and its Applications
Soichiro Hidaka

Traceability maintenance to localize changes

Change propagation

•Composition-based interaction design for adaptable distributed software systems
Kenji Tei