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
Distributed Software System

Service-oriented (cloud) system

network

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


\[ S, D \models R \]

Modify \( S \) in response to changes in \( D \) to satisfy \( R \)

off-line: Adaptive maintenance

on-line: Self-adaptation

Carlo Ghezzi, The fading boundary between development time and run time, SEAMS 2011
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

**Requirements (Goal)**

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

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

**Behavioral models**

**Structural models**

**Detailed Design**

**Impl.**

**Architecture Design**

**Req. analysis**
Traceability between R and S to Localize Changes

• Structural view

• Behavioral view

Interaction
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

- A room is cleaned
  - A map of a room is updated
  - The map is informed
  - Cleaning following the map
    - Analyzability
      - Reliability
    - Messages are logged
    - Logs are stored locally
    - Logs are stored in a server
    - Create and send logs
    - Receive and store logs
  - Plan a route
    - Clean based on the route
  - Send the map
    - Receive the map
  - A map is updated
  - A room observation is updated
    - Observation is requested
      - The room is observed
    - Observation results are informed
      - Monitoring robots are available
        - Observe environment by robots
      - Room sensors are available
        - Observe environment by room sensors
Less Traceability between R and S (Interaction)

Interweaving

scattering

hard to localize changes for adaptation
Our Approach: Composition-based Interaction Design

(1) design a piece of interaction

(2) automate compositions of interactions
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


- Detailed behavior design and implementation should also be changed according to the changed interaction
  - local behaviors and implementation are also designed to be composable
• 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 achieved by composing interaction pieces
  – easy to change solutions of a certain requirement
Overview of works in NII

- Exploration of adaptation space
  Fuyuki Ishikawa
  Adaptation space analysis

- Designing Self-adaptive System using Control Loops
  Shinichi Honiden

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

- Putback-based Bidirectional Programming
  Zhenjiang Hu
  Bidirectional Graph Transformation Infrastructure and its Applications
  Soichiro Hidaka

Traceability maintenance to localize changes

Change propagation