Modeling and Validating Interaction Aspects in UML

Jon Whittle, QSS/NASA Ames
João Araújo, Universidade Nova de Lisboa
Dae-Kyoo Kim, Colorado State University
Aspects at Use-Case Level

• Why?
  – Separate cross-cutting requirements
  – Avoid tangled requirements documentation
  – Facilitate requirements change
  – Support clear thinking process

• Clear thinking process
  – Represent aspects/non-aspects separately
  – Weave aspects/non-aspects
  – Execute aspects/non-aspects for validation
Represent Aspects/Non-Aspects

- Focus on interactions (UML Sequence Diagrams)
  - Non-aspects are just sequence diagrams
- Aspects are *Interaction Pattern Specifications* (Kim et al.)
- Weaving is specialized composition
- Execution is translation to state machines (Whittle & Schumann algorithm)
Interaction Pattern Specifications

• Specialization of the UML metamodel
  – each element is a role (a UML metaclass with additional properties)
• Instantiate by assigning UML model elements to each role
• Conformance: the assigned model elements satisfy the properties of that role
Example IPS

Car parking system: driver gets a ticket from entry machine after pressing a button. Driver parks. On leaving, driver inserts ticket into exit machine and pays.

roles in red
Weaving

Instantiate
Weave
Composition operator
Car parking example

- **Barrier**
  - open
  - close

- **Driver**
  - takeTicket
  - drive
  - ejectTicket

- **Lot Exit Machine**
  - insertTicket(t)
  - checkTicket(t)

- **Data Record**
  - recordTransaction(t)
  - sensorValidatedExit
Instantiation

|Machine| binds to Lot Exit Machine
|Action| binds to insertTicket
|a| binds to t
|CannotRespond| binds to timeout
Composed Interaction

**Barrier**

**Driver**

**Lot Exit Machine**

**Data Record**

**Supervisor**

- `insertTicket(t)`
- `checkTicket(t)`
- `ejectTicket`
- `takeTicket`
- `sensorValidatedExit`
- `recordTransaction(t)`
- `open`
- `close`
- `drive`
- `timeout`
- `alertSupervisor(t)`
- `displayErrorMessage`

*red = former roles*
Composition Operators

- **OR**: alternative interactions with choice point to decide
- **AND**: interactions occur concurrently
- **IN**: insert one interaction inside the other
- Others: further work…
Validation of Composed Interactions

- Use Whittle & Schumann algorithm to transform composed interactions into a set of (executable) state machines

  - Whittle & Schumann:
    - State machine generated for each participant involved in the interaction
    - For each participant:
      - Incoming message becomes a trigger in the state machine
      - Outgoing message becomes an action in the state machine
      - Interactions can be joined using “state labels”
Example: state machine generated

- insertTicket(t)
- timeout / alertSupervisor(t);
  displayErrorMessage
- / recordTransaction(t);
  ejectTicket; open
- sensorValidatedExit/ close
- takeTicket
- entry/
  checkTicket(t)
Summary

• Representation/Instantiation/Weaving of aspectual/non-aspectual interactions (UML sequence diagrams)
• Translation to set of state machines for validation purposes
• Future work: how to feedback results of validation to augment or correct the interaction models