Explicit Connectors for Coordination of Active Objects

Author: Manuel Günter
Platform: Smalltalk (VisualWorks2.0)
Context: Diplomarbeit (~Masters Thesis) 1997/98

Presentation: 1 Title, Contents, Goal
2 Separation of Concerns
3 Explicit Connectors
4 Coordination Abstraction, Rules & Operators
5 Toy Example

Goal: The FLO/c model allows object-oriented declaration of high-level multi-object coordination.
1. Separation of Concerns

- Components vs connectors in software architecture design (Allen & Garlan).
- Computation vs coordination in parallel programming (Carriero & Gelernter).
- Domain specific code vs synchronization code in concurrent programs (Bloom).
2. Explicit Dynamic Connectors

- Run-time objects that implement interaction between active objects.
  - Therefore the ideal location for coordination mechanisms.
  - Run-time dynamics.
- Connectors connect components. A Connector has roles. The connected components play roles in their connector.
- Independency of components (and vice versa).
- Monitor messages, react with messages and executions.
  - The connector can have states and methods, and use them to react.
  - The reaction is defined by a set of user defined rules, that trigger on requests (messages).
- Can connect groups of objects.
- Can collaborate with other connectors, adding additional global properties (e.g. fairness).
3. Coordination Abstraction using Rules

Connectors react upon message sends of the objects they control. The reaction is coded into rules.

**Rule** = request message, operator, consequence messages.

**Coordination abstraction** = Synchronized multi-object joint actions.

Operators to compose joint actions:

- **implies** push style computation ordering.
- **impliesBefore** pull style computation ordering.
- **permittedIf** balking style conditional synchronization.
- **waitUntil** blocking style conditional synchronization.

Operator to propagate requests (asynchronous & unprotected): **impliesLater**.

---

Universität Bern 26.2.98 Coordination Abstraction using Rules
An Example Connector Specification

MetaConnector new;

inheritsForm: AbstractConnector;
name: 'JumpNCatchConnector';
roles: 'dancer1 dancer2';
rules: ' 

dancer1 jumps. implies dancer2 catches. endRule 
dancer2 catches. waitUntil dancer2 armsAreFree. endRule 
dancer1 jumps. impliesLater dancer1 jumps. endRule ' 

Joint actions can model:
- Multi-object constraints (Synchronizers of Frolund&Agha).
- Mutual exclusion on shared resources.
- Transactions.