Component Model with Support of Mobile Architectures

Marek Rychlý

Department of Information Systems
Faculty of Information Technology
Brno University of Technology

10^{th} International Conference on
Information System Implementation and Modeling,
April 23–25, 2007

Outline

Outline

Information Systems

- Information systems (ISs) as distributed systems are collections of software components, which communicate and coordinate their actions via a middleware.
- Architecture of IS has to follow the organisational structure of a company, integration of well-established ISs, technology constraints, security requirements, etc.
- Architecture of IS is evolving according to the changing requirements.
- The middleware can provide dynamic connections, e.g.
  - according to functionality (available services),
  - according to free resources,
  - according to policies of individual components, etc.

Software Architecture

static architectures – architecture does not evolve during the execution of a system, it is described at design-time,

dynamic architectures – architecture can evolve during the execution of a system, but it is described at design-time,
(e.g. components can be created, deleted, reconfigured, or moved at run-time)

mobile architectures – components can logically move during the execution of a system, according to functional requirements.

- How to control dynamic and mobile architectures and reflect the architecture evolution at design-time?

Component-Based Development

Definition (Component models)

specific meta-models of component-based software architectures including rules for components, connectors, their interconnections, rules for changes according to the dynamic architecture, etc.

component – self contained entity, a system's part without externally observable state, accessible via its interfaces,

connector – connection of compatible interfaces of cooperating components,

configuration – actual organisation of components interconnected via connectors.

$\Rightarrow$ oriented to composability and reusability.

Outline

Motivation

Basic idea

To develop a component-model and its formal basis supporting features of a mobile architecture, integration of the model into a software development process.

- **dynamic reconfiguration** – changes according to the dynamic architecture,
- **component mobility** – the ability to move and copy components into new locations,
- **combination of control and functional interfaces** – functional requirements imply changes of architecture,
- **appropriate formalism** – native support of composition, reconfiguration and mobility.

Structure of the component

Component:

abstraction – specification and behaviour given by its formal description (a black-box view),

implementation – specific implementation of the component's behaviour (a grey-box view).

Component implementation:

primitive – realised directly, beyond the scope of architecture description,

composite – decomposable on a system of subcomponents at the lower level of architecture description.

Component's interfaces

Interfaces by accessibility/location:

public – interfaces of component abstraction, accessible to its neighbouring components,

private – interfaces of composite component, accessible only to its subcomponents.

Interfaces by function:

functional – represent business oriented services of the component (functional requirements),

control – provide services for control of the component (control of its life-cycle and connections).

A Calculus of Mobile Processes (π-Calculus)

- In 1992 by R. Milner, J. Parrow and D. Walker as modification of CCS.
- Algebraic approach to a system of concurrent and mobile processes.
- Two concepts:

  agents – communicating processes,
  names – communication channels, data, etc.
- Key features:
  - passing of names – passing of parts of architecture,
  - replication – ability to fork processes (lazy replication).

Description of a Component's Behaviour

component – parametric π-calculus process with names for functional interfaces p_i and control interface r

$$C(r, p_1, \ldots, p_m) = \\ !C_f(r, p_1, \ldots, p_m) \mid !r(x).\overline{x}\langle p_1 \rangle \ldots \overline{x}\langle p_n \rangle$$

primitive component – the process C_f describes externally observable behaviour of the component according to communication on its interfaces.

composite component – the process C_f describes functional part of the component as parallel composition of processes of subcomponents and connectors.

connector – process forwarding communication between names p_i for provided interfaces and q_i for required interfaces

$$B(p_1, \ldots, p_u, q_1, \ldots, q_v) = \\ \sum_{i=1}^{u} \sum_{j=1}^{v} q_j(x).\overline{p_i}\langle x \rangle.B(p_1, \ldots, p_u, q_1, \ldots, q_v)$$

Summary

- Distributed ISs create needs for component-based design with dynamic architecture.
- It is difficult to control run-time reconfiguration and reflect it at design-time.
- The presented work outlines the component model for mobile architectures with semantics in π-calculus.

Future work

- Completing exact description of the formal semantics.
- Integration of the model into a software development process.
- Case-study and its evaluation.

For Further Reading

Clemens Szyperski.

Component Software: Beyond Object-Oriented Programming.

Addison Wesley Professional, second edition, November 2002.

Davide Sangiorgi and David Walker.

The π-Calculus: A Theory of Mobile Processes.

Cambridge University Press, October 2003.

Marek Rychlý.

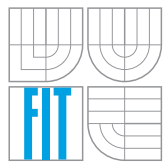
Towards verification of systems of asynchronous concurrent processes.

In *Proceedings of 9th International Conference Information Systems Implementation and Modelling (ISIM'06)*, pages 123–130. MARQ, April 2006.

Marek Rychlý and Jaroslav Zendulka.

Distributed information system as a system of asynchronous concurrent processes.

In *MEMICS 2006 Second Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 206–213. Faculty of Information Technology BUT, October 2006.



Thank you for your attention!