CHARLES UNIVERSITY PRAGUE
faculty of mathematics and physics

Introducing Distribution into a RTSJ-based Component Framework

Michal Malohlava, Aleš Plšek, Frédéric Loiret, Philippe Merle, Lionel Seinturier
michal.malohlava@dsrg.mff.cuni.cz
{ales.plsek|frederic.loiret|philippe.merle|lionel.seinturier}@inria.fr

The Real-Time Specification for Java (RTSJ) is becoming a popular choice in the world of real-time and embedded systems. But, a growing complexness of these systems brings a demand for their distribution. However, there are only a few projects addressing application of RTSJ in distributed environments.
In this paper we introduce our approach based on software connectors to support distribution in a RTSJ-based framework. We propose extensions of our Soleil framework to achieve distribution while still preserving its original benefits: separation of concerns and mitigation of complexities in the system development lifecycle.

Introduction

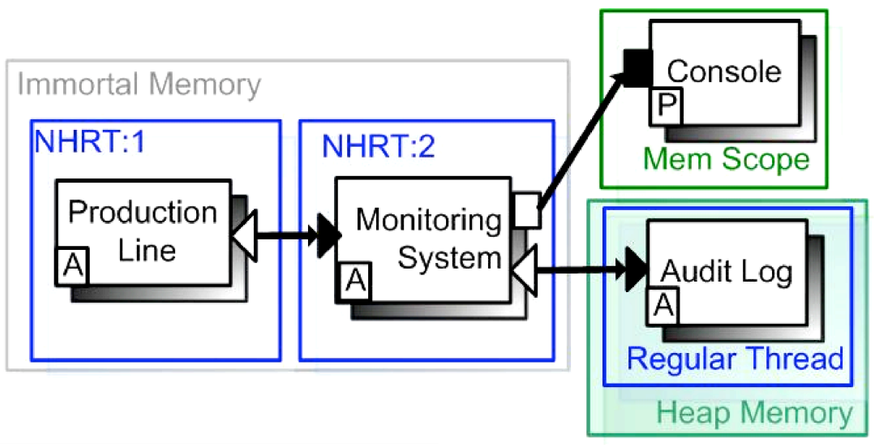
Methods reducing a complexity of developing real-time systems:
- Introducing general-purpose languages (Java)
- Applying software engineering paradigms
  - **CBSE** (Component-based Software Engineering)

Background

Real-time Specification for Java (RTSJ)
- Determinism in Java is achieved by introducing
  - Memory areas (scoped, immortal, heap)
  - Schedulable entities (real-time threads, events)

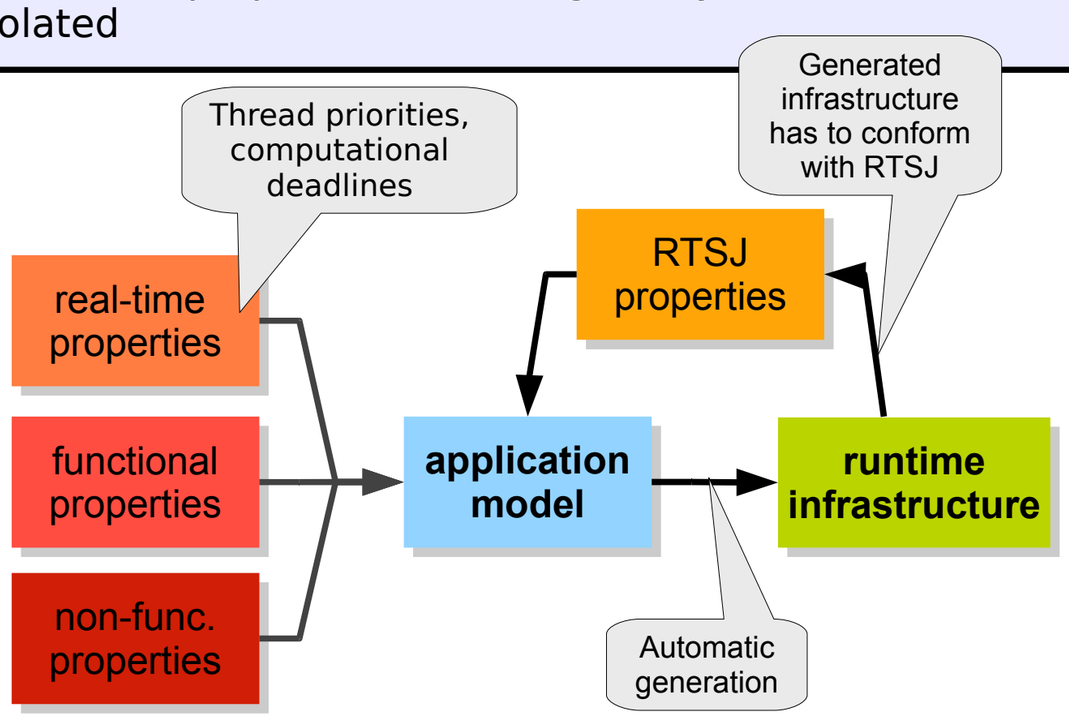
Component based systems
- **Application of CBSE paradigm into RTSJ world**:
  - Component framework ***Soleil*** [4] which brings *RTSJ views* of a component-based application by introducing *domain components*

Domain components describe utilization of memory areas as well as thread types used by active components



Challenges

Introducing distribution into RTSJ-based component framework requires:
- *Reflecting various properties* at different stages of a system's development lifecycle;
- *General binding representation*
  - To allow reflecting properties and encapsulating a selected middleware and communication style (method call, messaging) in the automatically generated binding's infrastructure
- Real-time properties of a designed system cannot be violated



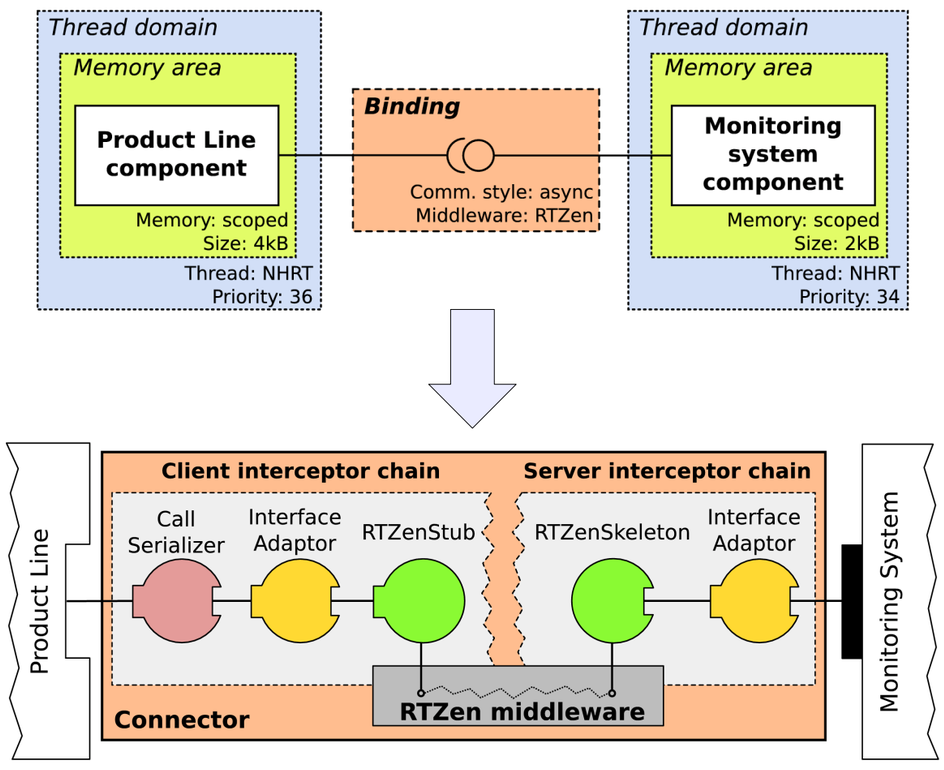
Approach

Design time of application
- Binding is described by properties

Architecture of a connector ([2], [3])
- *Chains of interceptors* represent connection end-points
- Interceptors reflect specified properties (1:1)

Binding infrastructure generation process
- Selecting connector architecture (interceptors in chains)
- Interceptor code generation

Runtime infrastructure
- Has to reflect specified properties & comply to RTSJ



Design of application specifies binding properties

Generation process produces binding infrastructure

Runtime infrastructure of binding encapsulating *RTZen* middleware [5]

Conclusion & Future Work

- Applying connector paradigm in RTSJ component-based framework *Soleil* and mitigating the complexity of designing and implementing distributed RTSJ-based component systems.

- Taxonomy of RTSJ-based connectors
  - Mapping between properties and connector arch.
- Advanced examples, benchmarks

Bibliography

[1] T. Bures. *Generating Connectors for Homogeneous and Heterogeneous Deployment*. PhD thesis, Department of Sofware Engineering, Mathematical and Physical Faculty, Charles University, Prague, 2006.

[2] G. Bollela, J. Gosling, B. Brosgol, P. Dibble, S. Furr, M. Turnbull. *The Real-Time Specification for Java*. Addison-Wesley, 2000.

[3] N. R. Mehta, N. Medvidovic, and S. Phadke. *Towards a Taxonomy of Software Connectors*. In Proceedings of ICSE'00, New York, USA, 2000. ACM.

[4] A. Plšek, F. Loiret, P. Merle, and L. Seinturier. *A Component Framework for Java-based Real-time Embedded Systems*. To Appear in Proceedings of 9th International Middleware Conference, Leuven, Belgium, December 2008.

[5] K. Raman, Y. Zhang, M. Panahi, J. A. Colmenares, R. Klefstad, and T. Harmon. *RTZen: Highly Predictable Real-Time Java Middleware for Distributed and Embedded Systems*. In Middleware 2005.