Evolvable Middleware Container Architectures for Distributed Embedded Systems

Aleš Plšek

Under the supervision of:
Lionel Seinturier, Philippe Merle

CALA Junior Days
20.2.2007, Brussels

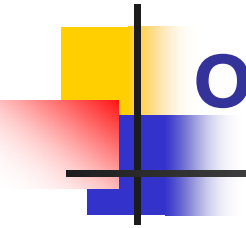
INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA
FUTURS

Goal of the Presentation

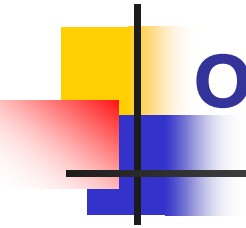
- Ph.D. topic: Evolvable Middleware Container Architectures for Distributed Embedded Systems

- Domain
  - Middleware Architectures
  - Component Oriented Programming
  - Embedded Systems



Outline

- Context
- Motivations & Goal
- State of the Art
- Research Proposal
- Conclusion



Outline

- Context
- Motivations & Goal
- State of the Art
- Research Proposal
- Conclusion

Context

- Distributed Systems
  - Full-fledged computer networks
  - Mobile networks
  - Embedded and Real-Time Systems
- Trends
  - Growing complexity
  - Convergence of mobile devices and PCs
- Demands on middleware layer
  - Software engineering challenges
  - More effective development – time to market delivery
  - Evolvability

Outline

- Context
- Motivations & Goal
- State of the Art
- Research Proposal
- Conclusion

Motivations & Goals

- Motivation
  - Employing Component Based Software Engineering (CBSE)
  - Achieving Evolvable/Adaptive System
  - Addressing Embedded and Real-time Systems

- Goal
  - Framework which supports
    - Effective development of middleware systems
      - reusability, upgradeability, etc.
    - Tailorable systems fitting different environments
      - facing embedded and real-time constraints
    - Dynamically evolvable systems

Outline

- Context
- Motivations & Goal
- State of the Art
- Research Proposal
- Conclusion

Programming languages in Middleware

- Low-level languages
  - Tedious and error-prone
  - Assembly language, C or C++
- Distributed programming languages
  - Steep-learning curve
  - AmbientTalk
- Java
  - unpredictability
  - Real-Time Java

Real-time Java

- Classical Java
  - Unpredictable: Garbage Collection, Thread scheduling
  - unsuitable for Real-time environments

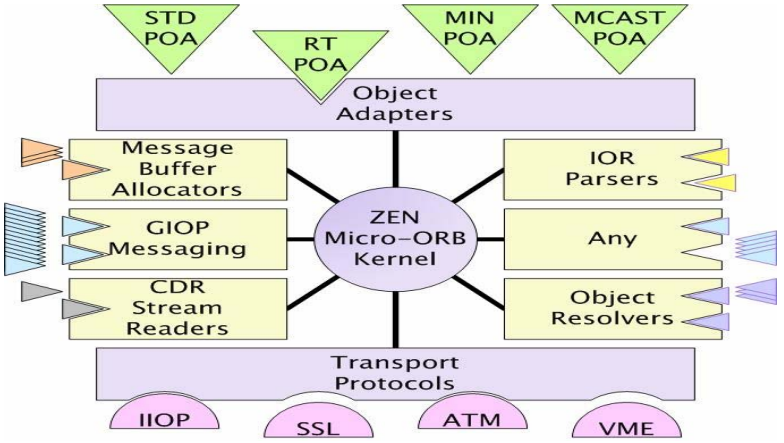
- Real-Time Java
  - Reducing unpredictability
  - Introduces
    - Scoped Memory
    - Real-time Threads
    - New scheduling mechanisms

Real-Time Java Disadvantages

- Scoped Memory
  - Predictable memory access
  - Outside the scope of the garbage collection
  - Scopes: immortal, thread scope, ...
- Disadvantages
  - Memory management put back upon the shoulders of the developer
- Remedy
  - Introducing Component Oriented Programming
  - Each component has its own scoped memory
    - The scope is allocated while the component is active
  - Hierarchical scopes
  - Flexibility

RTZen

- Real-Time CORBA
  - Supports distributed, real-time and embedded applications
- Pluggable Design
  - Footprint reduction

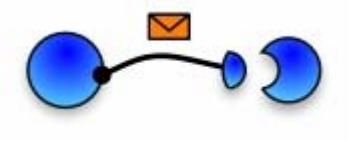


DREAM

- Framework for Message-Oriented Middleware systems
- Component Oriented Approach
  - Fractal Component Model
- Design and development of Component-Based and Reflective Middleware
  - Easily extensible and adaptable middleware systems
- Services
  - Deployment Service
  - Type-Checking Tool

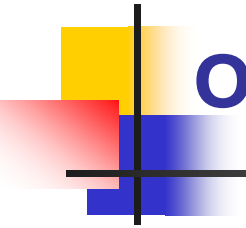
AmbientTalk

- Designed for Mobile Ad-hoc networks
  - To deal with the mobile hardware characteristics
- Propose Distributed Programming Language
  - Directly incorporates features to cope with the environment specifics
- Concepts
  - AmbientReference
  - Mailboxes
  - Distributed Garbage Collection



Current Trends Summary

- Component Oriented Programming
  - Development of distributed applications
  - Monolithic systems – COP not employed in the middleware layer development
- Java
  - Real-time Java
- Pluggable design
  - Memory Footprint Reduction
  - Facing embedded system constraints
- Reflective Middleware
  - Evolvability
  - Adaptability



Outline

- Context
- Motivations & Goal
- State of the Art
- Research Proposal
- Conclusion

Requirements

- Time-to-market delivery
  - Utilize effective middleware system development
- Evolvability
  - Adaptable to changing mission requirements
- Tailorability
  - Towards Embedded systems
- Easy to use
  - Avoid steep-learning curves

Our Proposal

- Goal
  - Middleware Framework which supports
    - Effective development of middleware systems
      - reusability, upgrades, etc.
    - Tailorable systems fitting different environments
      - facing embedded and real-time constraints
    - Dynamically evolvable systems
- Additionally
  - Hide the complexities of embedded, real-time and distributed application from the developer

Employed Concepts

- Component Based Software Engineering
  - Middleware build as an assembly of interacting components
  - Time-to-market delivery
- Reflective Middleware
  - Dynamical Evolvability
- Pluggable Design
  - Plugging and unplugging services on demand
  - Achieving Tailorability

Addressing Embedded and Real-time Systems

- Embedded and Real-Time constraints
  - Constrained Resources (memory, processor,...)
  - Predictability
- Constrained Resources Solution
  - Tailorable Middleware System
  - Dynamical Adaptation
- Predictability – Solution
  - Introducing Real-time Java
    - Scoped Memory
    - Real-Time Threads

Research Methodology

- State of the Art (limitations, solutions)
- Design
  - Concepts to employ
- Implementation
  - Fractal Component Model modification
  - Design and implementation of component oriented containers providing different services:
    - Scoped memory components
    - Different communication models: messages, RPC, AmbientReferences
    - Component Oriented Interceptors (hiding potentional complexities form the user)
- Validation
  - Developing and evaluating a middleware system
  - Measuring system requirements, container footprint

Contributions

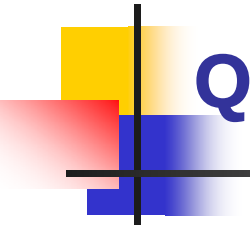
- Next step in middleware system development
  - Middleware framework
  - Employing Component Based Software Engineering techniques
  - Dynamically evolvable middleware
    - Reflective middleware
  - Facilitates to develop middleware fitting specific needs
    - Embedded and real-time environments

Outline

- Context
- Motivations & Goal
- State of the Art
- Research Proposal
- Conclusion

Conclusion

- Problem Statement
  - Growing complexity of distributed systems
  - Middleware layer forms performance bottleneck for new distributed applications
- State of the Art
  - Middleware systems – Concepts, Solutions and Limitations
- Research Proposal
  - Framework for development of evolvable middleware systems
    - Addressing embedded and real-time systems
- Contributions
  - General framework
  - Component Based Software Engineering techniques in middleware development



Questions?