Evolvable Middleware Container Architectures for Distributed Embedded Systems

Aleš Plšek
ales.plsek@inria.fr

Under the supervision of:
Lionel Seinturier, Philippe Merle

ADAM Team

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA
FUTURS

RUNES Summer School
10.7.2007, London

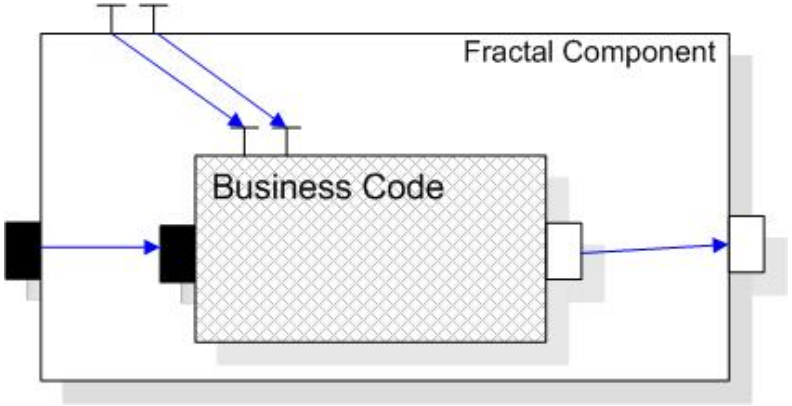
Motivation and Goals

- Motivation
  - Growing complexity of Embedded / Real-time systems
  - How to achieve
    - Evolvable/Adaptive Systems?
  - Employing Component Oriented Programming (COP)

- Goal
  - Middleware Framework which supports
    - Effective development of middleware systems
      - reusability, upgradeability, etc.
    - Tailorable middleware systems fitting different environments
      - facing embedded and real-time constraints
    - Dynamically evolvable systems
    - No additional burdens for developers
      - Avoiding steep-learning curves

Fractal Component Model

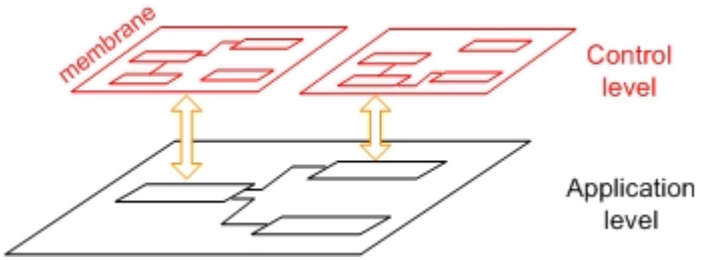
- Component consists of
  - Business Content
    - Functional part
  - Membrane
    - Non-functional part

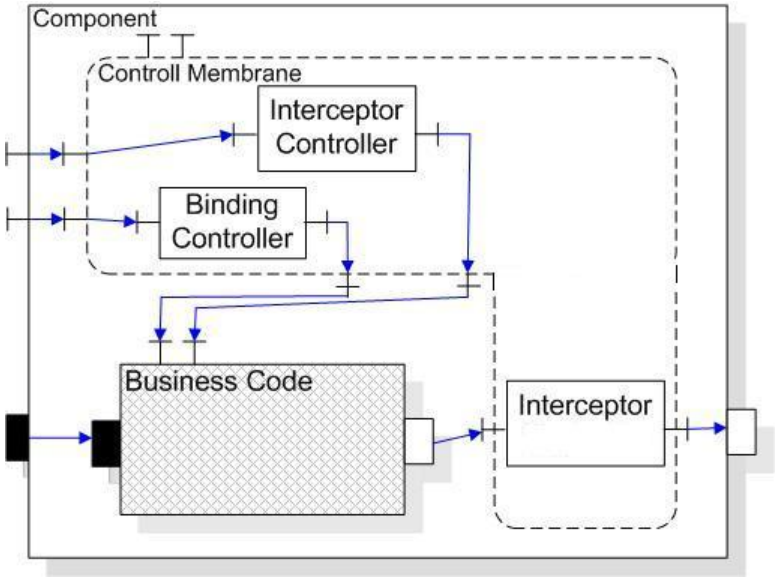


- Interfaces
  - Business interfaces
    - Provided/Required
  - Controller interfaces
    - Lifecycle controller, Binding controller, Content controller , etc.

Component-Oriented Control Membranes

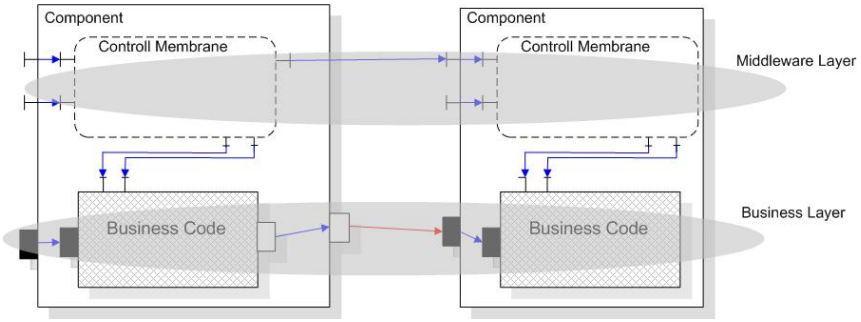
- New Feature
  - Control Membrane
  - Component–Oriented Approach
- Advantages
  - Different membranes fitting specific needs of components
  - Dynamical Adaptability
- Controllers
  - Component Controllers
    - Non-functional aspects implemented as components
  - Membrane Controllers
    - Supports full dynamical control over membrane
  - Interceptors





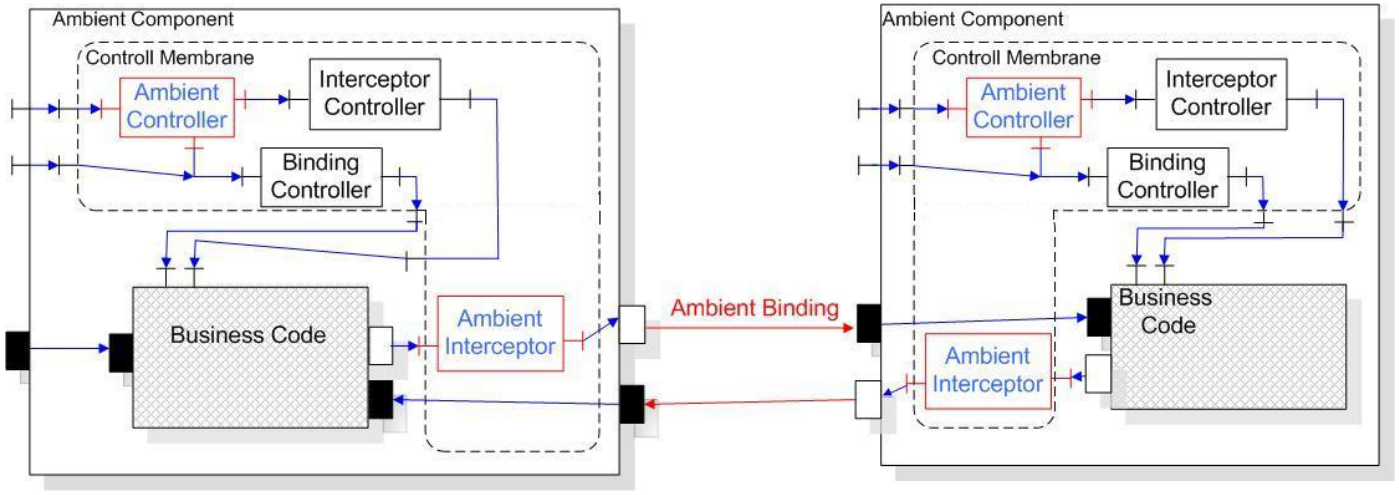
Our Research Proposal

- Motivation
  - Addressing Embedded / Real-time systems
  - Achieving Evolvable System
  - Employing Component Oriented Programming (COP)
    - Component-Oriented Membrane extensions
      - Dynamical adaptability of non-functional properties
- Goal
  - Middleware Framework
    - Effective development of middleware systems
      - reusability, upgradeability, etc.
    - Dynamically evolvable systems
    - Middleware represented by membrane extensions



Case-Study – Ambient Environments

- Ambient Environments
  - Characteristics - connection volatility, ambient resources, autonomy, etc.
- Goal
  - AmOP + COP
    - Ambient Middleware represented by membrane extensions
    - Evaluation of component-oriented control membranes



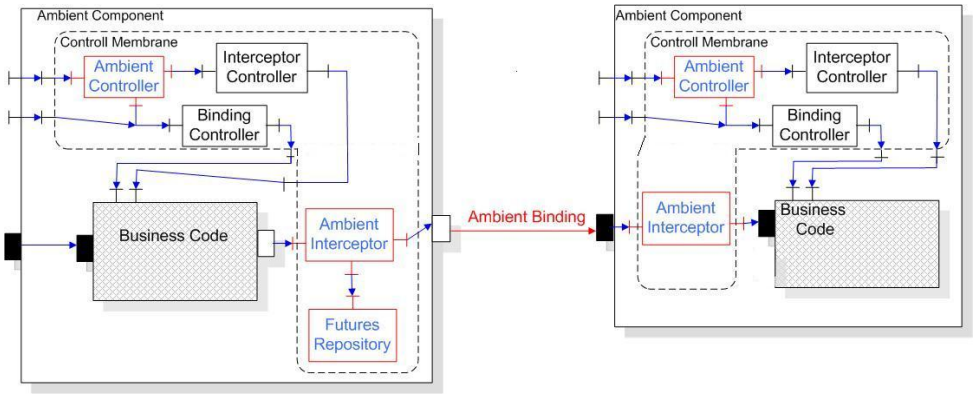
Evaluation

- Evolvability
  - Membrane extensions – Controllers, Interceptors
    - No modification of business code
  - Ambient-Awareness deployed only where needed
  - Ambient Bindings managed by membrane

- Drawbacks & Limitations
  - Callback Binding - Tangled code
  - Black-box view

Future Work

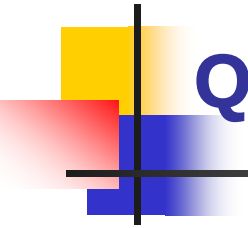
- Control-Membrane extensions
- Embedded Java & Real-Time Java
  - Characteristics, requirements, constraints
  - Support in membrane



- Code Annotations
  - Tagged Futures - Graffiti Spoon [1]
  - Synchronous communication
    - Returned value is temporarily substituted by the Future – placeholder

Conclusion

- Middleware Framework
  - Through the Component-Oriented Control Membranes
  - Achieving Evolvability
- Case Study – Ambient Environments
  - Deploying ambient-awareness in membranes
- Issues
  - Tangled code
- Future Work
  - Membrane extensions
  - Code annotations
  - Embedded & Real-Time Java support



Questions?

References

- [1] Johan Fabry, Carlos Noguera: *Abstracting connection volatility through tagged futures*