Are we ready for computer assisted living?

Tomáš Bureš
bures@d3s.mff.cuni.cz
Application Contexts in General

- Smart phones and on-body systems to communicate in changing and mobile environments that offer users access to information and services while on the move;
- Homes, cars and offices, that offer systems and solutions for improved enjoyment, comfort, wellbeing and safety ...

Adapted from [ARTEMIS AWP 2012]

Example: Road Trains

“Autovlak, kde můžete za volantem číst noviny, už vyjel” [idnes.cz, 8.6.2012]
Priorities (EU FP7 ICT)

FP7 ICT challenges:

1. Pervasive and trusted network and service infrastructures
   • Internet of Things, Internet of Services
2. Cognitive systems and robotics
3. Alternative paths to (hardware) components and systems
4. Technologies for digital content and languages
5. ICT for health, ageing well, inclusion and governance
6. ICT for a lower carbon economy
7. ICT for the enterprise and manufacturing
8. ICT for learning and access to cultural resources
Goal of the Talk

- Do we know how to develop such systems (for “computer assisted living”)?
  - Judging from the perspective of software design & development
  - In particular judging from perspective of component-based software engineering
Running Example: E-mobility

Key Aspects
- Combine a number of concerns
- Distributed interaction
- Autonomy
- Adaptability
- Emergent behavior

[FP7 project ASCENS – Deliverable D7.1 (VW Demonstrator)]
The Focus: Software

Software controls the hardware

Correspondence between physical (HW) and virtual (software) world

Software creates a virtual world which reflects the physical (hardware) world

Both software & hardware represented as components
Challenges of Software

• How to design software with multiple concerns?
  • Minimize development time & costs

• How to design software which adapts to changes in the environment (e.g. physical world)?
  • Flexibility & predictability

Key Aspects

• Combine a number of concerns
• Distributed interaction

• Autonomy
• Adaptability
• Emergent behavior
Challenges of Software

• How to design software with multiple concerns?
  • Minimize development time & costs

• How to design software which adapts to changes in the environment (e.g. physical world)?
  • Flexibility & predictability

Key Aspects

• Combine a number of concerns
• Distributed interaction

• Autonomy
• Adaptability
• Emergent behavior
Software With Multiple Concerns

• Models (blueprints) each addressing particular concerns
  ▪ Cooperating component models – **Component Model Family**
  ▪ Each component model has its own
    • Granularity of component
    • Composition rules
    • Execution semantics
    • Way of interoperability with other component models in the family
  ▪ Benefits: comprehensibility, less errors, easier tool support (e.g. code generation)
Example: Software in a Car (Stability Control)

- **ProCom**
  - Communication via asynchronous exchange of messages
  - Ideal for subsystems connected to network (e.g. CAN bus)

- **ProSys**
  - Explicit data and control flow
  - Synchronous execution
  - Ideal for modeling a single subsystem (e.g. periodic feedback control loop)


Building Component Model Families

• “Meta-component system” – a product line for component model families
  ▪ Produces a component model family based on selection of application domains and requirements


Malohlava M., Plášil F., Bureš T., Hnětynka P.: Interoperable DSL Families for Code Generation, Software: Practice and Experience, John Wiley & Sons, ISSN: 1097-024X, April 2012 (IF: 0.519)


Challenges of Software

How to design software with multiple concerns?
• Minimize development time & costs

How to design software which adapts to changes in the environment (e.g. physical world)?
• Flexibility & predictability

Key Aspects

• Combine a number of concerns
• Distributed interaction

• Autonomy
• Adaptability
• Emergent behavior
Software Which Adapts to Environment

- Component architecture has to constantly change to reflect the situation in the physical world
  - e.g. car ignition switch being turned from off to start position
  - e.g. a new passenger sharing a car

- Definition of a component architecture contains information about how it should adapt to changes in the environment (e.g. physical world)
Limited Adaptation

Pop, T., Plášil, F., Outlý M., Malohlava, M., Bureš, T.: Property Networks Allowing Oracle-based Mode-change Propagation in Hierarchical Components, Proceedings CBSE 2012, June 2012 (Core A)
Limited Adaptation

- Modes form a finite automaton
  - Each state is a mode (defines component architecture)
  - A transition is an event in the environment

- Going beyond – when infinite number states is needed
  - Adaptation described by reconfiguration actions associated with an event
    - Minimal set of reconfiguration actions (orthogonal basis): create/destroy component, create/destroy connection
  - Needed mostly in enterprise systems (user session, security context, etc.)

Extensive Adaptation

- Implicit architecture
  - For extensive adaptations including component mobility
  - Describes architecture by giving interaction templates
  - Architecture forms at runtime based on current state of components (including their location)
  - Emergent behavior

- DEECo (developed at D3S – FP7 project ASCENS)
  - New paradigm – improves existing approaches in terms of
    - Autonomous behavior
    - Management of components belief

Keznikl J., Bureš T., Plášil F., Kit M.: Towards Dependable Emergent Ensembles of Components: The DEECo Component Model, Accepted for publication in Proceedings of WICSA/ECSA 2012, Helsinki, Finland, August 2012 (Core A)
Extensive Adaptation – DEECO (VW case study)

The car that the passenger travels in.

A free parking lot which is closest to the next destination.
Conclusion: Where Do We Stand?

- Mature methods for:
  - modeling relatively static systems by components
    - e.g. car control systems – engine, brakes, ...
  - analysis of such static systems
    - e.g. timing analysis, functional properties, ...
  - combining particular families of component models
    (full development cycle, including code generation)
    - e.g. control loops + subsystem communication

- New methods for:
  - modeling (using components) distributed dynamic systems with emergent behavior
    - e.g. intelligent navigation in e-mobility
Conclusion: What is Ahead of Us?

- We need to elaborate more on
  - Component **self-awareness** and adaptation based on **high-level goals** and strategies
  - Techniques and models with a proper level of abstraction for feasible testing and verification of **correctness** of components with **emergent behavior**
  - **Prediction** and **optimization** techniques for achieving efficient use of **resources** by distributed adaptive components
Are we ready for computer assisted living?

- From software perspective – we are not very far and many things can be done already, we have to
  - combine existing approaches;
  - scale existing approaches and elaborate on new ones to address the large open systems with emergent behavior.