Taming Dynamically Adaptive Systems using Models and Aspects

Brice Morin, Olivier Barais, Grégory Nain and Jean-Marc Jézéquel

DiVA: European Project on Dynamic Variability in Complex Adaptive Systems
http://www.ict-diva.eu/

S-Cube: European Network of Excellence on Software Services and Systems
http://www.s-cube-network.eu/
Outline

• Context, Problems and Related Work

• Solutions to meet the challenges

• Conclusion

• Future works
Context

• Home-automation to help disabled people to stay at home
  • Aging society
  • Hospital have limited resources, rooms, etc
    • → Very short stays
  • Long stays very expensive for people and society
  • Houses, flats, etc should be equipped
Many Different Needs 1/2

Mrs. Dupont

- Living at home
- Motion troubles
- Memory loss
- Speaks French (only)
- Home equipped with:
  - LonWorks (lights)
  - Velux (shutters)
Many Different Needs 2/2

Mr. John Doe

- English student
- Living at home
- He had an accident
- He likes technology
- Wheelchair equipped with remote access for:
  - Lights and shutters (KNX)
  - Multimedia (UPnP)
Their needs

Both

Medical/technical staff should be able to
  • Check their health state
  • Check home configuration (shutters, lights, heaters...)

Mrs. Dupont
Some daily tasks should be automated (motion troubles) or reminded (memory loss).

Mr. Doe
Would like to control everything remotely, with a unified protocol.
Different variability dimensions

- **Protocols**
  - Low-level protocols: KNX, X2D, X10, etc
  - High-level protocols: UPnP, DPWS, etc

- **Devices**
  - Lights, heaters, shutters, etc

- **Languages**
  - Mainly French
  - But also main European languages

- **Adaptation to Handicap**
  - Motion, memory, perception, etc
Challenges

- Explosion of the number of possible configurations
  - $10^{14}$ possible configurations! $\Rightarrow 10^{28}$ transitions!

- Dynamic Adaptation
  - Evolution of the handicap
  - Houses should be configured remotely
    - No wires to connect/disconnect in the walls
  - No service interruption
    - Rebooting the system cannot be a solution (lives depend on the system)

- Reliability
  - **Safe migration path**
    from a valid configuration to another valid configuration
  - Performance issue (time) not critical
Related works

Reliability, Validation

K. Czarnecki et al.  
GPCE’06

J. Whittle et al.  
MODELS’07

F. Fleurey et al.  
Models@Run.Time’08

B. Cheng et al.  
ICSE’06, AOSD’09

Oreizy et al.  
ICSE’98

OSGi, Fractal,  
OpenCOM, etc

Variability Management

S. Appel et al.  
ICSE’06

M. Mezini et al.  
FSE’04

Hallsteinsen et al.  
Computer’08

B. Morin et al.  
MODELS’08

Garlan et al.  
Computer’04

P. David et al.  
SC’06

Dynamic Adaptation
How to validate DAS?

- Specify everything!
  - all the configurations: $>10^{14}$
  - all the transitions: $\sim10^{28}$
- Model checking, code generation

Problems

- Explosion: Time consuming, error-prone
- Evolution of the system (not predicted)
  - Stop all -> Evolve the specifications -> model check
    -> re-generate -> re-deploy
How to manage dynamic variability?

- Do not focus on configurations!
  - Write reconfiguration scripts, encapsulating « features »
- Depending on the context and/or user needs
  - Choose the most adapted scripts
  - Executes all the selected scripts to dynamically adapt the system

Problems

- Scripts written by hand (calls to reconfiguration API)
- Interactions, dependencies between scripts?
- Does the configuration (after executing scripts) make sense?
  - Hopefully yes…
Outline

• Context, Problems and Related Work

• Solutions to meet the challenges

• Conclusion

• Future works
Adopting a DSPL approach

• Focus on variability, not on configurations

• Build (derive) configurations when needed

• Validate configurations before actual adaptation

• Automate the reconfiguration process
Had a flu last week
→ stayed in bed
→ a nurse every day

Now in a better shape
Extensive design-time validation

- Still possible to validate everything, for small systems
  - Produce all the possible configurations by aspect weaving
  - Validate all the configurations

Discussion

- Time/resource consuming
- The number of configurations explodes
- … but they are automatically generated, by aspect composition

- Not scalable
Validation of aspect models

• Aspect-Oriented Modeling
  • Validate the DSPL at design-time
  • Strong theoretical background (graph theory)
  • Modular reasoning
  • \(\rightarrow\) interactions and dependencies detection
    • Using Critical Pair Analysis
  • \(\rightarrow\) weaving order
Two interacting aspects

Device Proxy

Light

Filter

require I18N

Light1

Shutter1
Two dependent aspects

- Device Proxy
- Light Filter
- Shutter1
- Light1

require I18N

Simplified I18N
Limitations of CPA

• Critical Pair Analysis has limitations
  • Aspect1, Aspect2 $\rightarrow$ OK
  • Aspect1, Aspect3 $\rightarrow$ OK
  • Aspect1, (Aspect2, Aspect3) $\rightarrow$ ?

• Need to validate woven configurations
  • At runtime, when they are produced
Checking configurations at runtime

• Focus on one configuration
  • Not the whole dynamically adaptive system

• Efficient roll-back
  • The running system is not yet adapted
  • Just discard invalid models
  • Report to user
Invariant checking

General Invariants

```aspect
aspect class Component {
  inv mandatoryClientPortBound is
  do
    self.type.ports.select{p |
      not p.isOptional and
      p.role == PortRole.CLIENT
    }.forAll{p |
      self.binding.exists{b |
        b.client == p
      }
    }
  end
}

aspect class TransmissionBinding {
  inv wellFormedBinding is
  do
    //link a client port to a server
    //port of the same type
  end
}
```

Specific Invariants

```aspect
aspect class System {
  inv hasEnglishI18N is
  do
    self.allComponents.contains{c |
      c.type.services.contains{s |
        s.name == "org.entimid.I18N"
      } and c.name == "EN"
    }
  end
}
```
Defining checking strategies

Simplified Metamodel
Outline

• Context, Problems and Related Work

• Solutions to meet the challenges

• Conclusion

• Future works
Conclusion

Explosion of the number of possible configurations

- DSPL to manage variability
- AOM to automatically derive configuration

Dynamic Adaptation

- Reflection model causally connected to the running system
- Changes not directly reflected

Reliability

- At design-time
  - Still possible to validate all the possible configurations
  - AOM provides more scalable mechanisms
- At runtime
  - focus on one configuration
  - Efficient roll-back
Perspectives and on-going works

Dual-view AOM

• Structural + behavioral view
• More advanced validation (deadlocks, livelocks, invariants)
• Simulation (performance, impact on QoS)

Towards higher-level adaptations

• We still manipulate components and bindings
• → drive the adaptations using domain concepts: device, scenarios
• Use MDE to map domain concepts to architecture
Thank you

Questions?