Motivation: a message delivery system
Adaptation scenario: Change of security level
**Motivation**

- It is necessary to build software that can change at runtime
- When replacing the implementation of the Encoding and Decoding component by another one:
  - All messages encoded by Encoding must be decoded by Decoding before realizing the adaptation process

**Building adaptive softwares: problems**

**Development process**

How to build adaptive software architectures?
→ development process

**Adaptation at runtime**

When can dependent components be replaced?
→ dynamic dependency
Outline

- Development process
  - Variability modeling
- Dynamic dependencies
- Adaptation process
- Conclusion

The first question

How to build adaptive software architecture?
Variability modeling

Common variability language (CVL) to specify the variability
- Base model: architecture of a software product line
- Variability model:
  - VSpec tree: represents the variability and the commonality
  - Variation points: links VSpecs to corresponding elements in the base model
- Resolution model: represents a configuration of VSpec tree

How to specify the CVL model?

(A) Domain experts
(B) Commonality
(C) Variability
(D) Variation points

Base model

VSpec tree

Existing application documents
Existing application architecture
CVL specification strategies

- Variability-driven process (top-down approach)
  - A-B-C-D

- Architecture-driven process (bottom-up approach)
  - A-C-B-D

- VSpec tree - base model independent process (hybrid approach)
  - A-B/C-D

Development process
The second question

When can dependent components be replaced?
Dynamic dependencies

- Transaction
  - Local transaction
  - Global transaction

Dynamic dependencies

- A new proposition: “dependsOn” constraint
Dynamic dependencies management

- The role of components

Reconfigurator

- Control the adaptation process
- Receive transactions information from first and last components of each global transaction
### Conclusion

- **The approach represents how to specify the CVL models and build adaptive software architectures**
  - Being applicable to other variability model, e.g., feature model
- **Manage dynamic dependencies using the new constraint, “dependsOn”**
- **A prototype based on iPOJO/OSGi component model. Our approach can be applied with various component implementation models, e.g., Fractal component model**
- **Future work:**
  - Considering functional errors during execution
  - Distributed adaptation