The Role of Feature Modeling in Software Product Line Engineering

Hassan Gomaa

Dept. of Computer Science
George Mason University
Fairfax, Virginia, USA
hgomaa@gmu.edu

16th CREST Open Workshop
Provenance and Product Lines
November 2011

GEORGE MASON UNIVERSITY

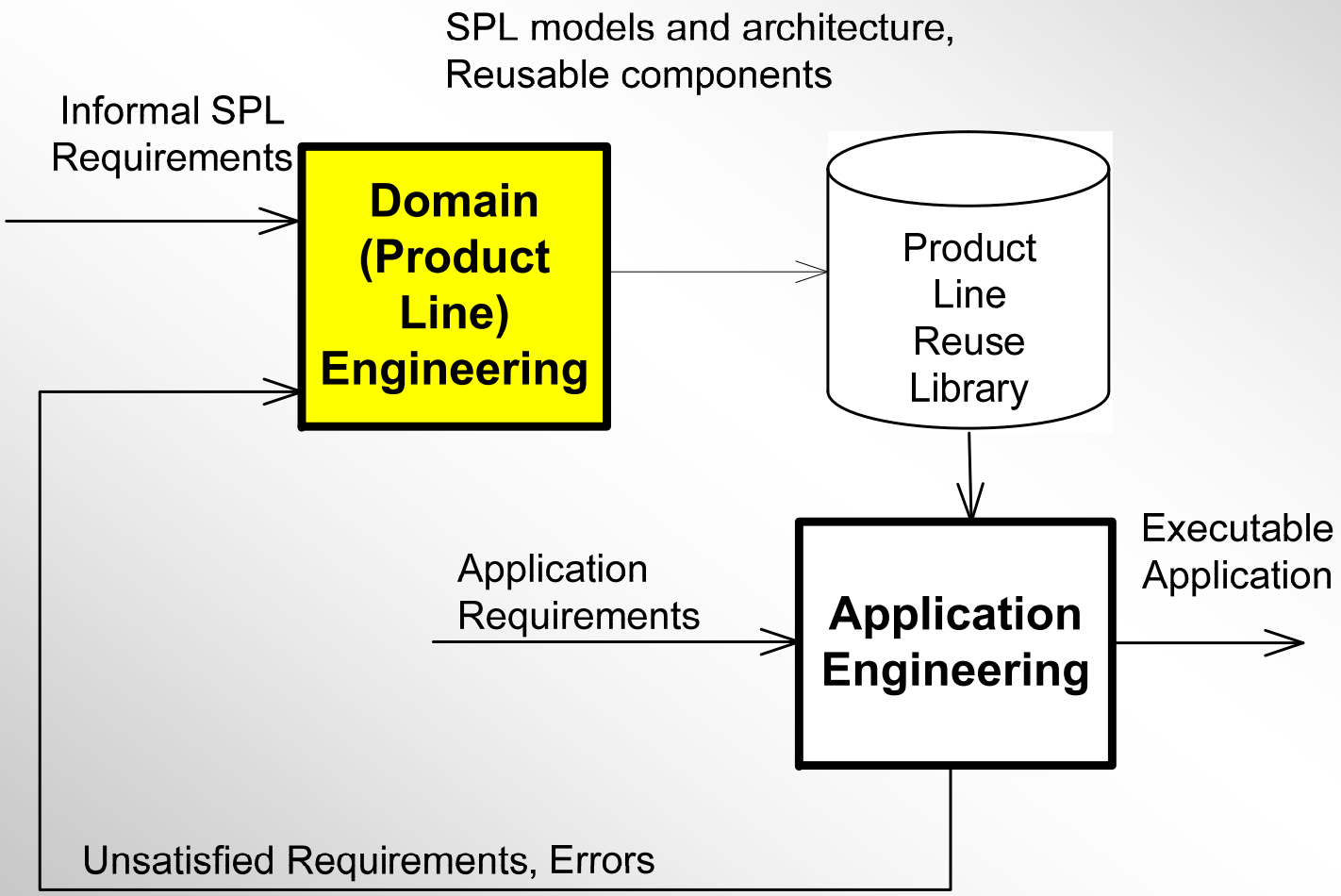
Software Product Line Engineering

- Software Product Line (SPL)
  - Family of products / systems (Parnas, Weiss, SEI)
- Software Product Line Engineering
  - Software engineering for a family of products
- Software Variability
  - Key problem in software product line engineering
  - Need to differentiate among common and variable software requirements and components
- Feature Modeling
  - Widely used in SPLs for specifying commonality/variability in requirements

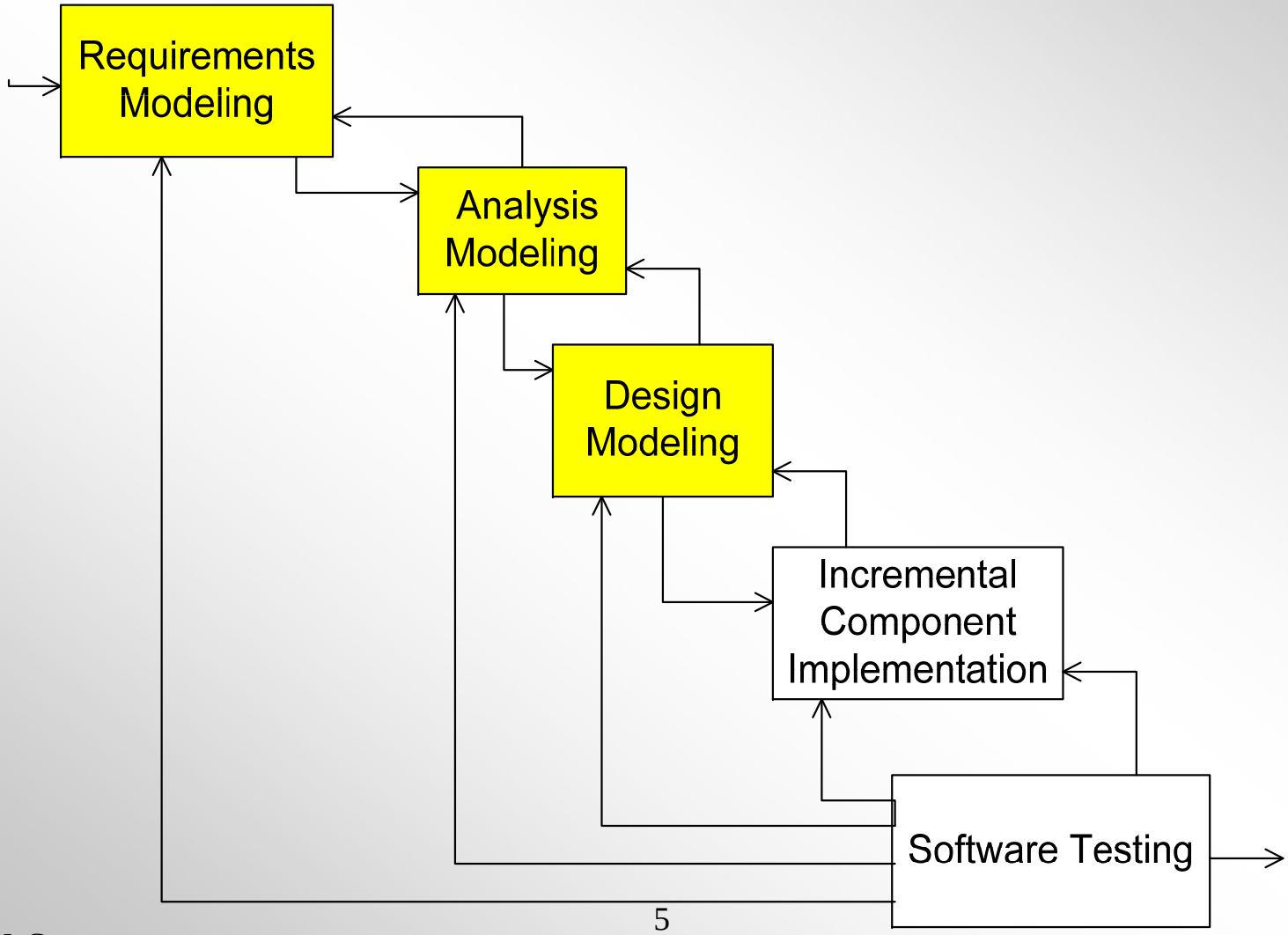
Software Modeling and Design for SPL

- Software Modeling and Design for Single Systems
  - COMET method
  - From Use Case Models to Software Architecture
  - *H. Gomaa, Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*, Cambridge University Press, February 2011
- Software Modeling and Design for SPL (PLUS method)
  - Model commonality and variability among SPL members
  - Integrate Feature Modeling with UML
    - Unifying View of Multiple-View Modeling Approach
  - *H. Gomaa, "Designing Software Product Lines with UML", Addison Wesley, 2005*

Evolutionary Process Model for Software Product Lines



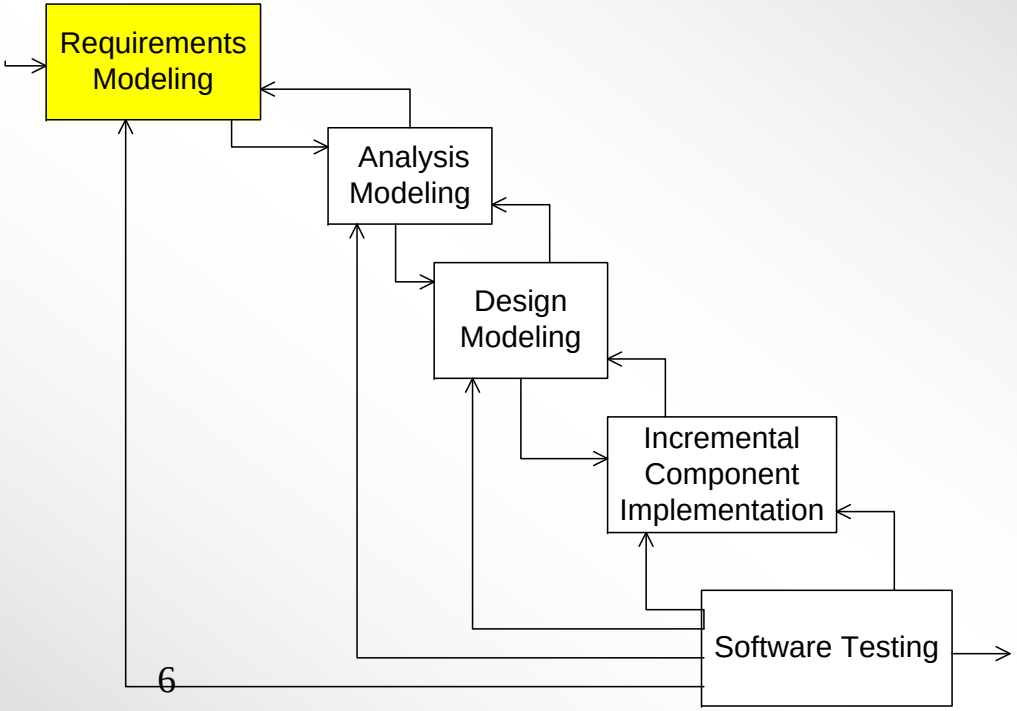
Model-driven Development in Product Line Engineering



Software Product Line Requirements Modeling

What should SPL Modeling Method provide?

- Support variability in use case modeling
  - Kernel, optional, alternative use cases
- Integrate feature modeling with other UML views

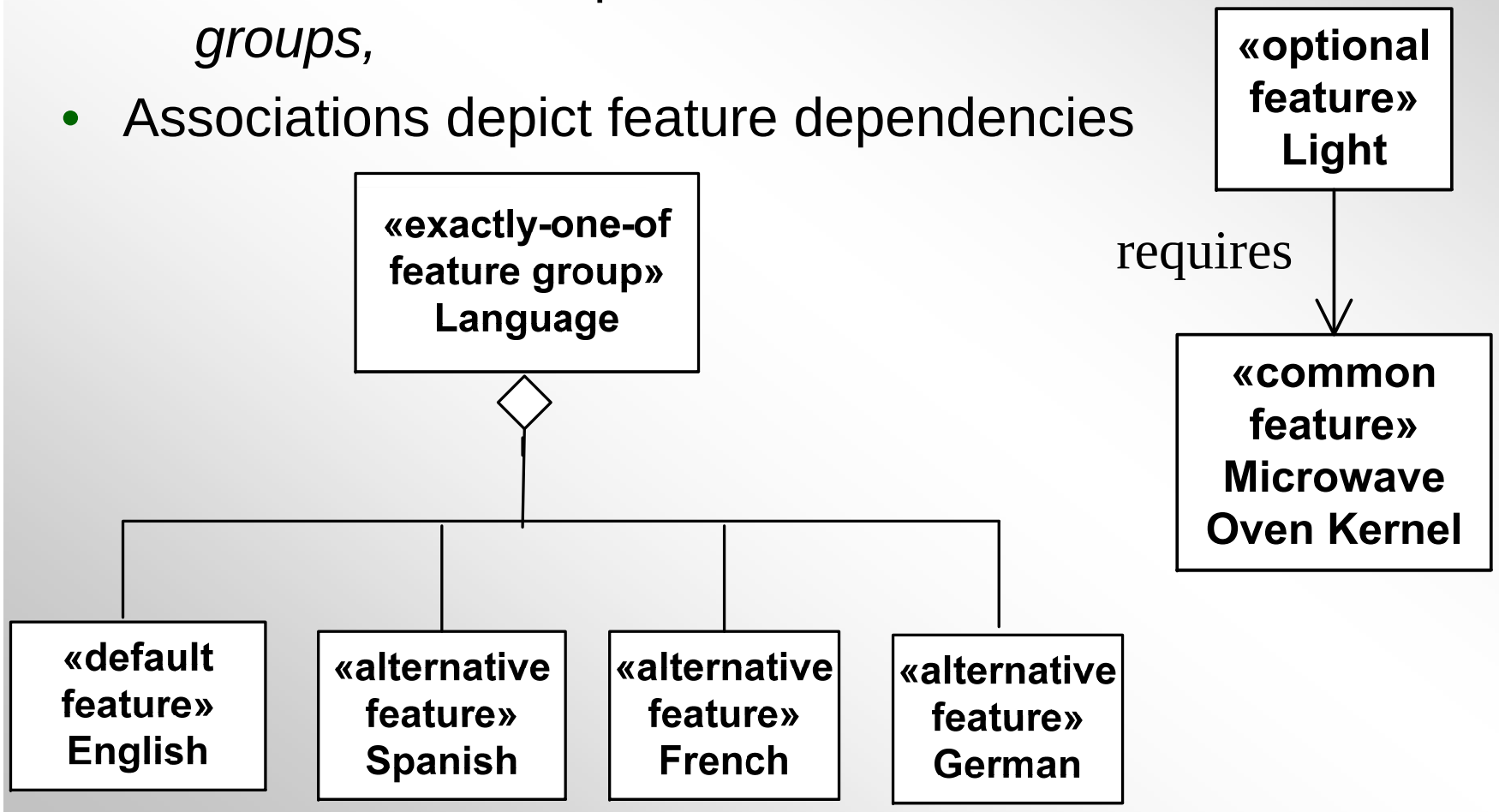


Feature Modeling for SPL

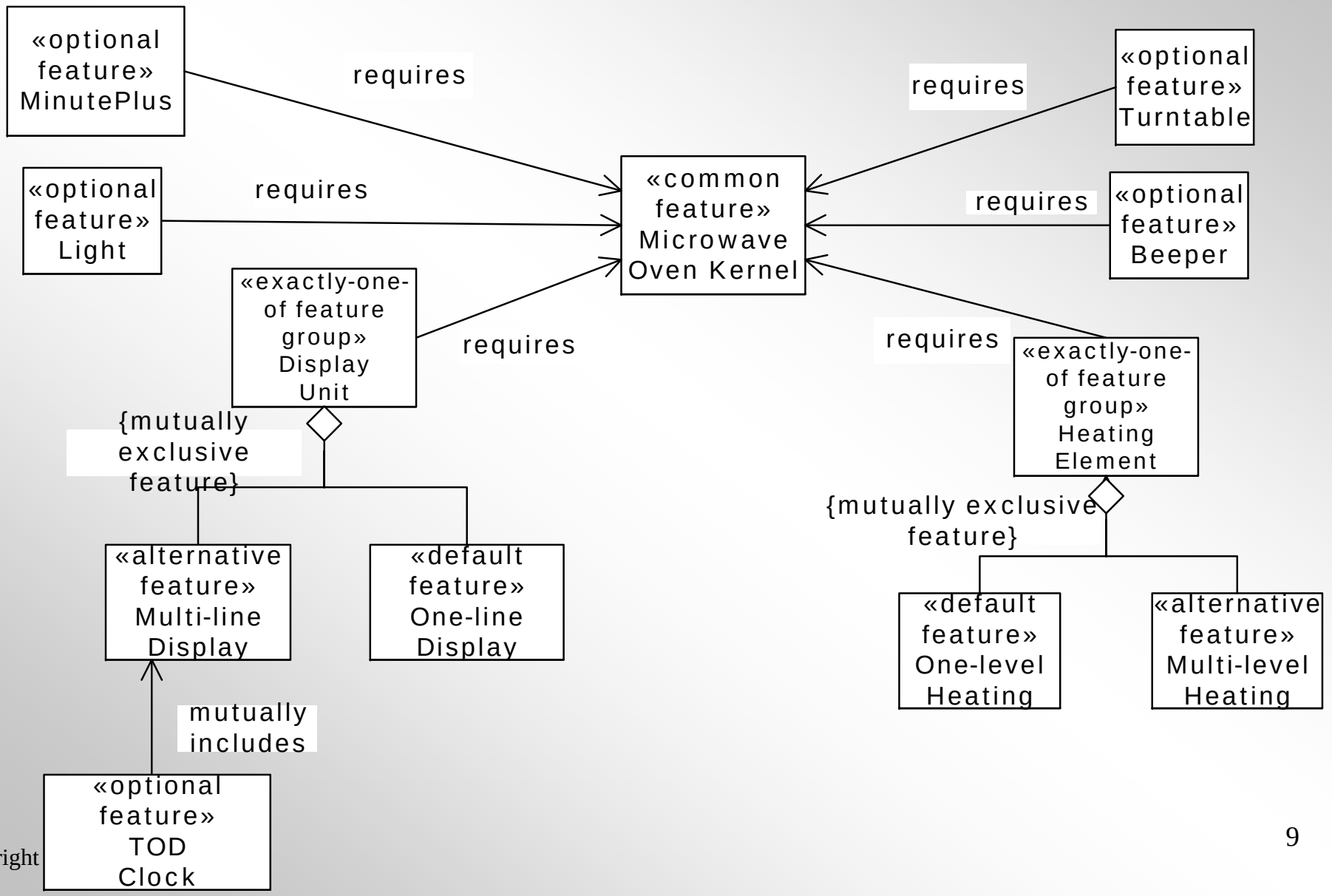
- Feature (Kang, SEI)
  - Function or characteristic that differentiates between members of the software product line
- Feature modeling in PLUS
  - PLUS integrates feature modeling with other UML views
  - Precisely represent dependencies in variability
  - Determine features from use cases and variation points
  - Concentrate on modeling variability

Feature Modeling with UML

- Use meta-modeling notation
  - Meta-classes depict *features, feature groups,*
- Associations depict feature dependencies

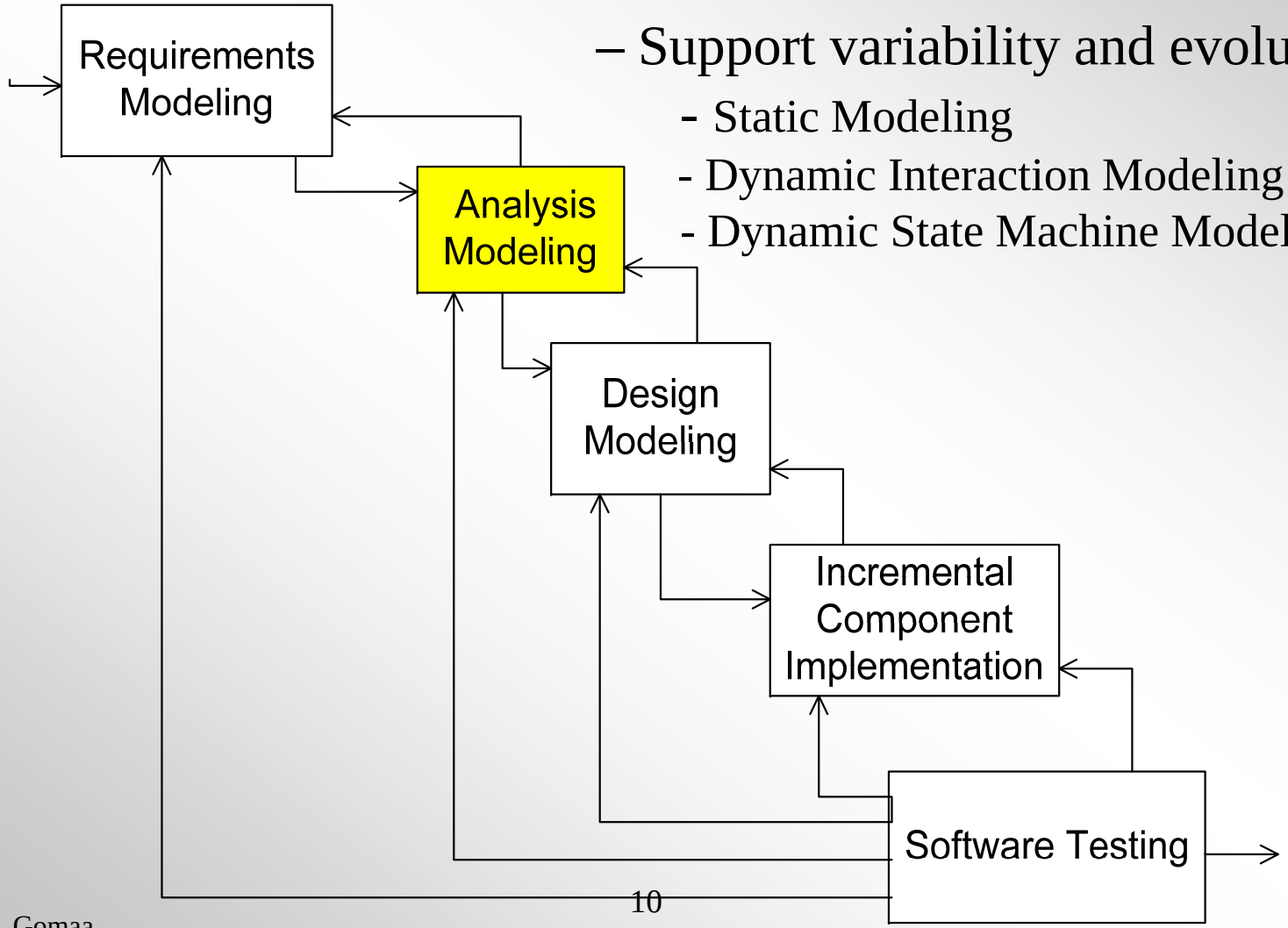


Microwave Oven Feature Model



Software Product Line Analysis Modeling

What should SPL Design Method provide?

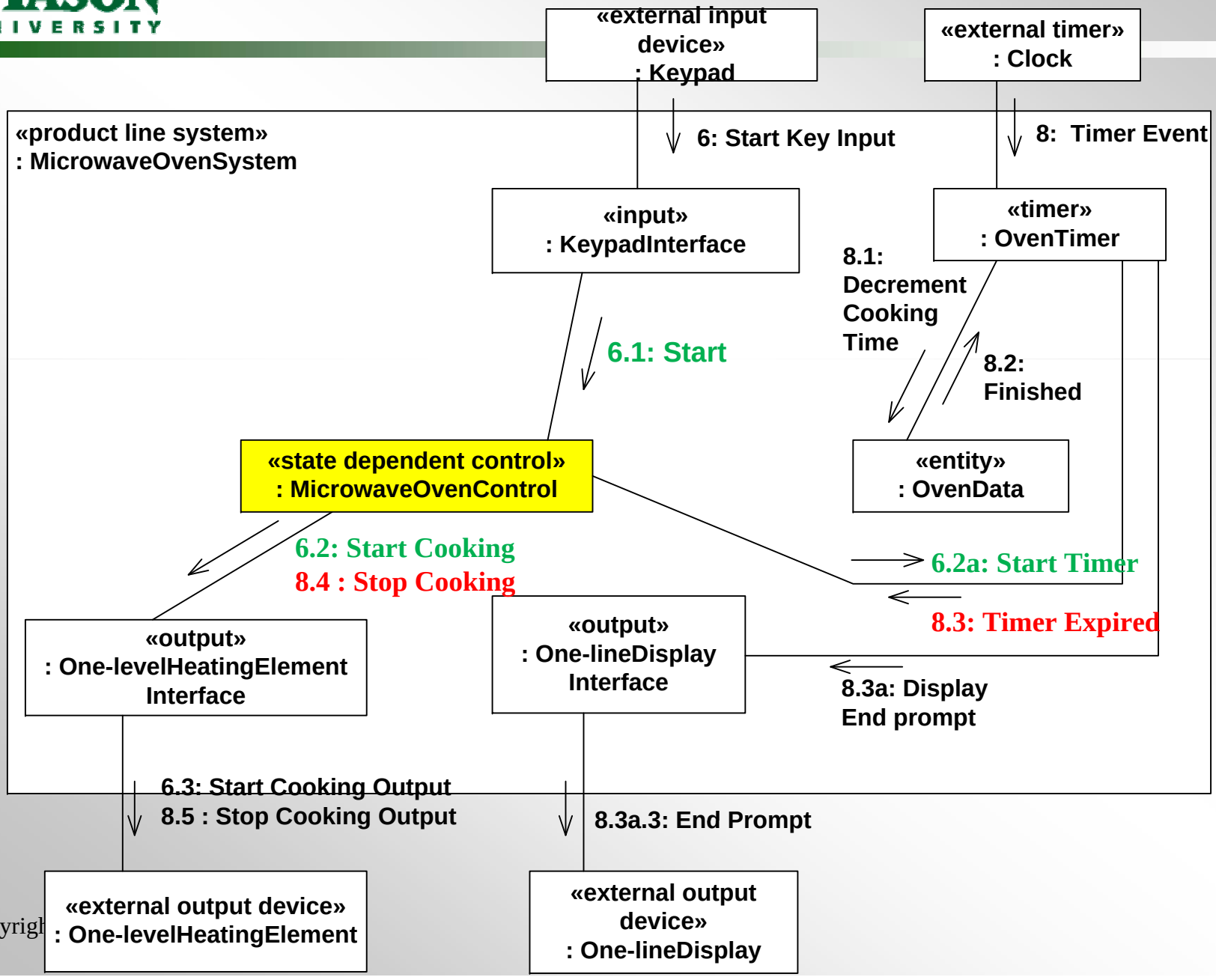


– Support variability and evolution
- Static Modeling
- Dynamic Interaction Modeling
- Dynamic State Machine Modeling

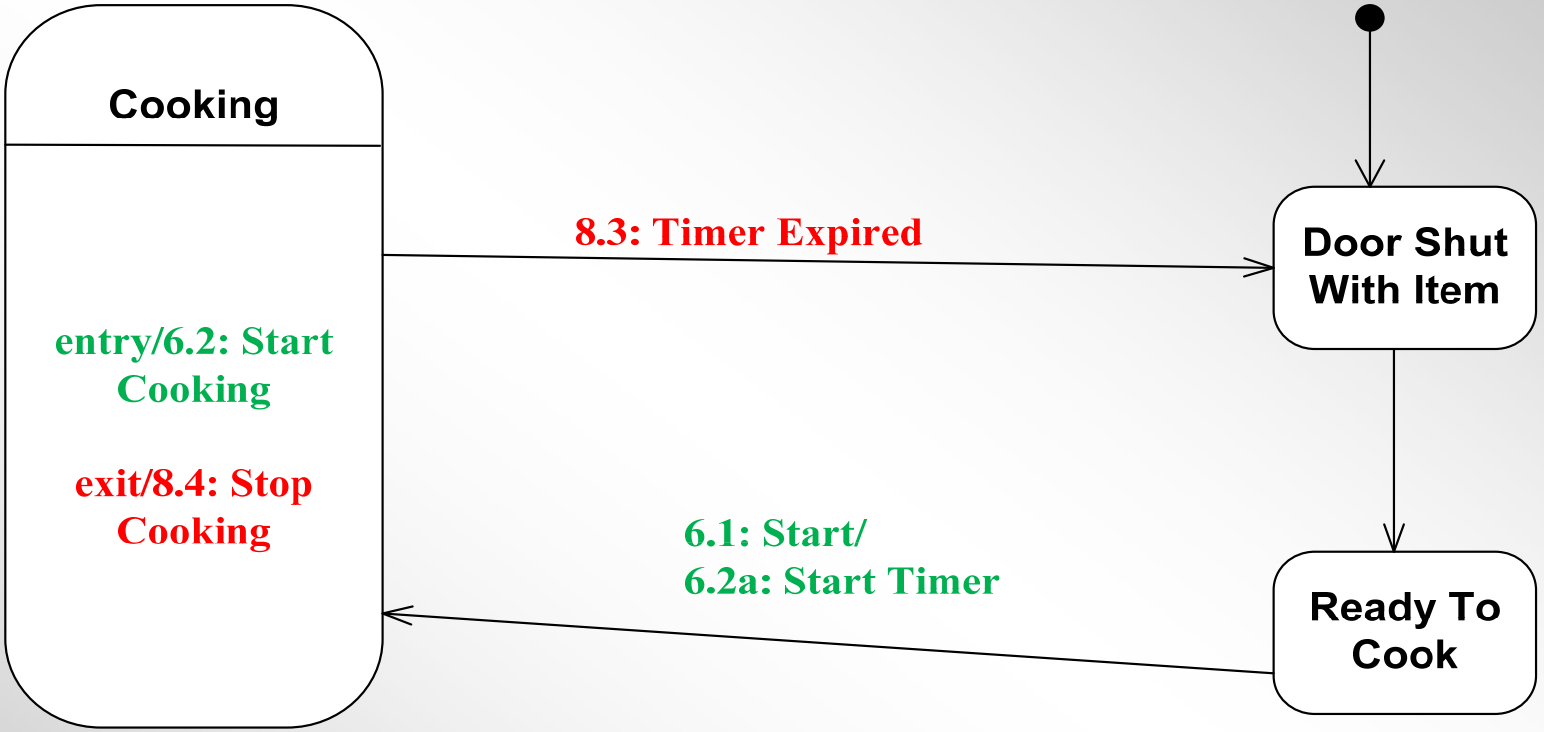
Evolutionary Development for SPL

- Kernel First Approach
  - Develop kernel of SPL
  - Kernel is similar to single system
- Evolutionary Development Approach
  - Consider evolution as an iteration in software development
- Model-based evolution
  - Feature-based Impact Analysis
  - Consider impact of optional and alternative features on kernel
    - Emphasis on dynamic modeling
    - Interaction diagrams, state machines

Kernel Communication Diagram for Cook Food use case



Kernel statechart for Microwave Oven Control

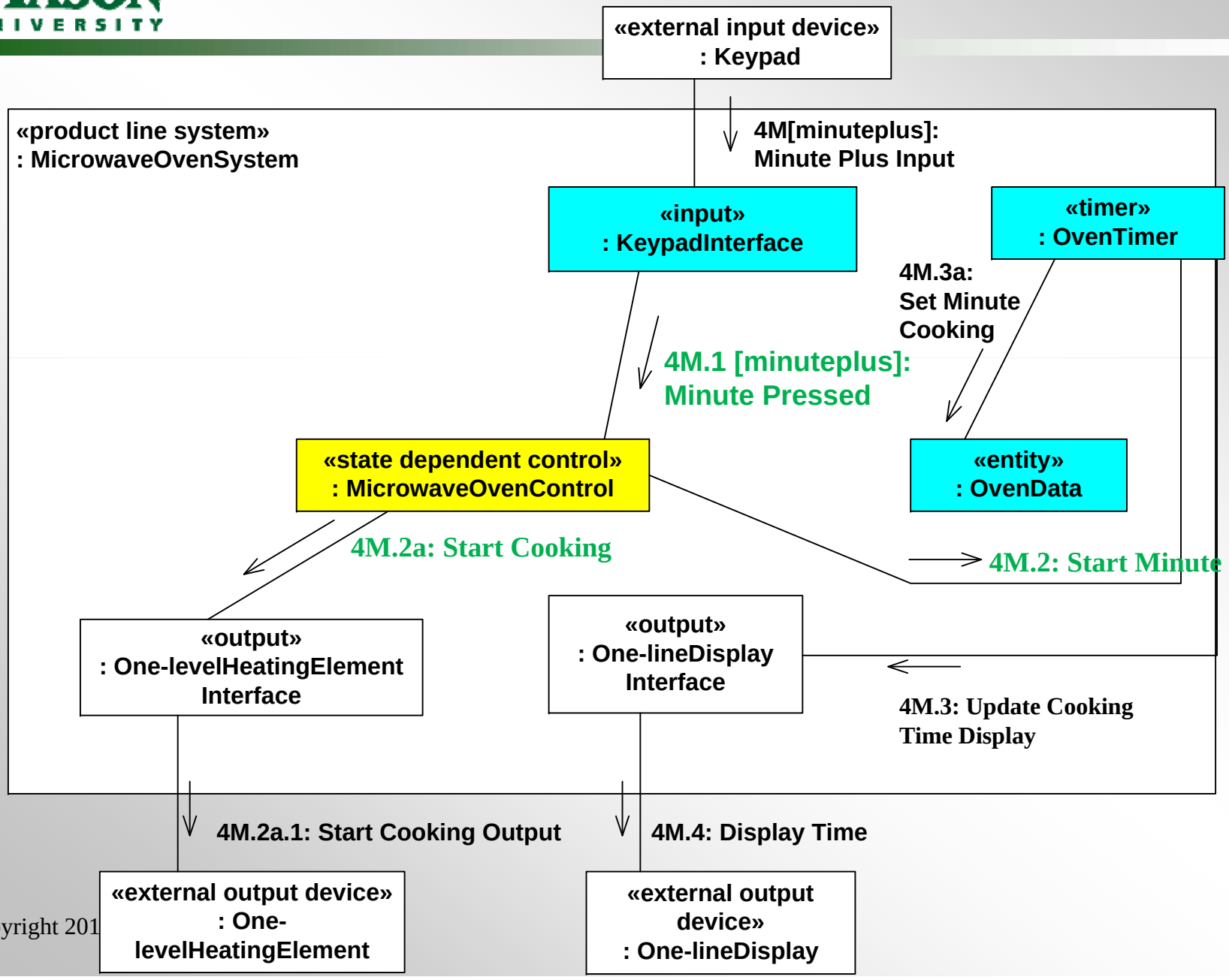


- **Incoming message to object-> input event on statechart**
- **Output event on statechart -> outgoing message from object**

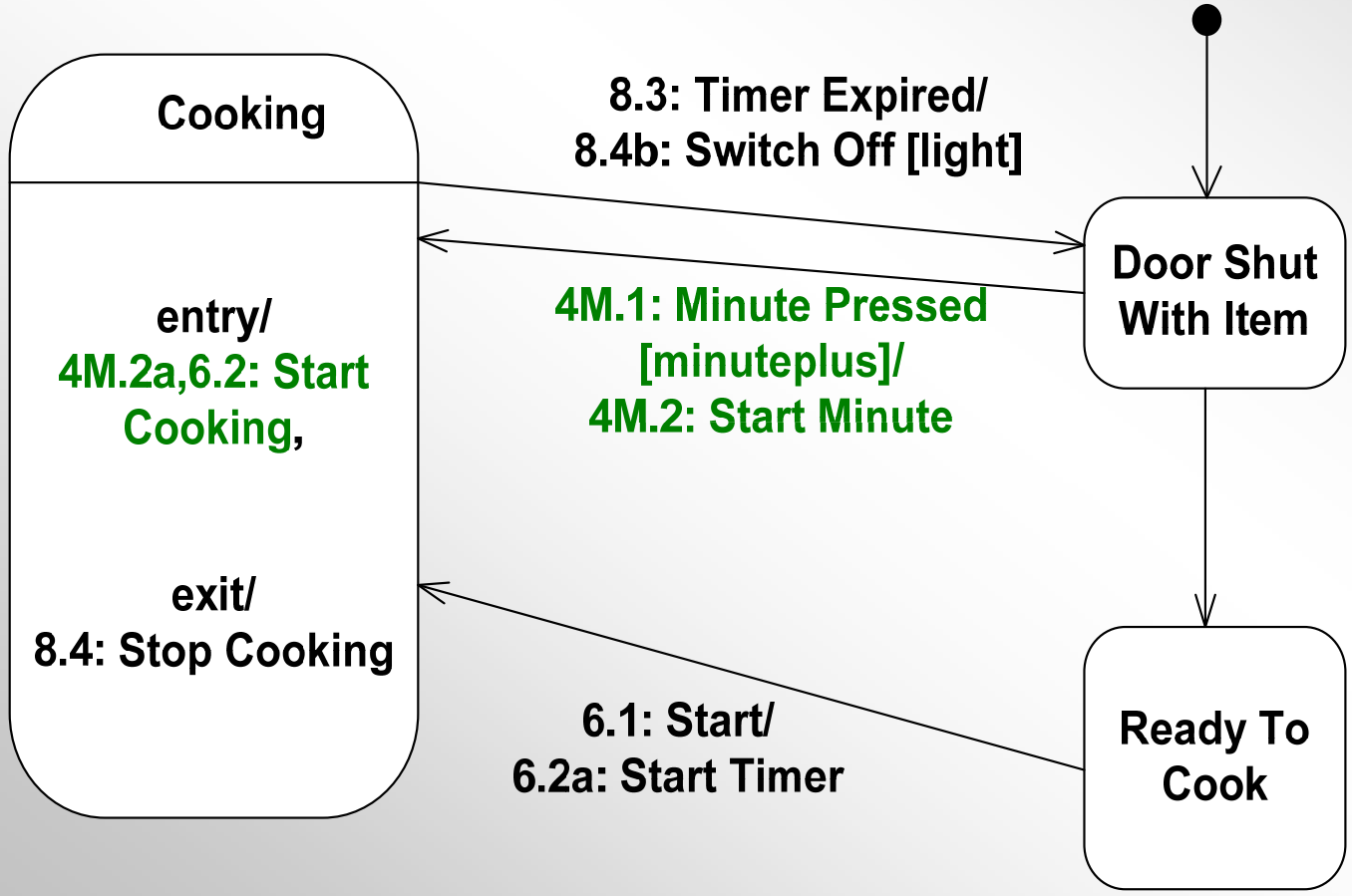
Evolutionary Development for SPL

- Feature Based Impact Analysis
  - Impact of optional/alternative features on interaction diagrams and statecharts
- Feature dependent state transition
  - Use feature condition as guard on state transition
    - Event [Feature Condition]
  - Feature condition is True if feature Selected
- Example
  - Optional feature: Minute Plus
  - Consider impact on objects
  - Consider impact on statechart

Impact Analysis of Minute Plus feature - Impact on Communication diagram for Cook Food use case



Feature dependent transition: Minute Pressed [minuteplus]

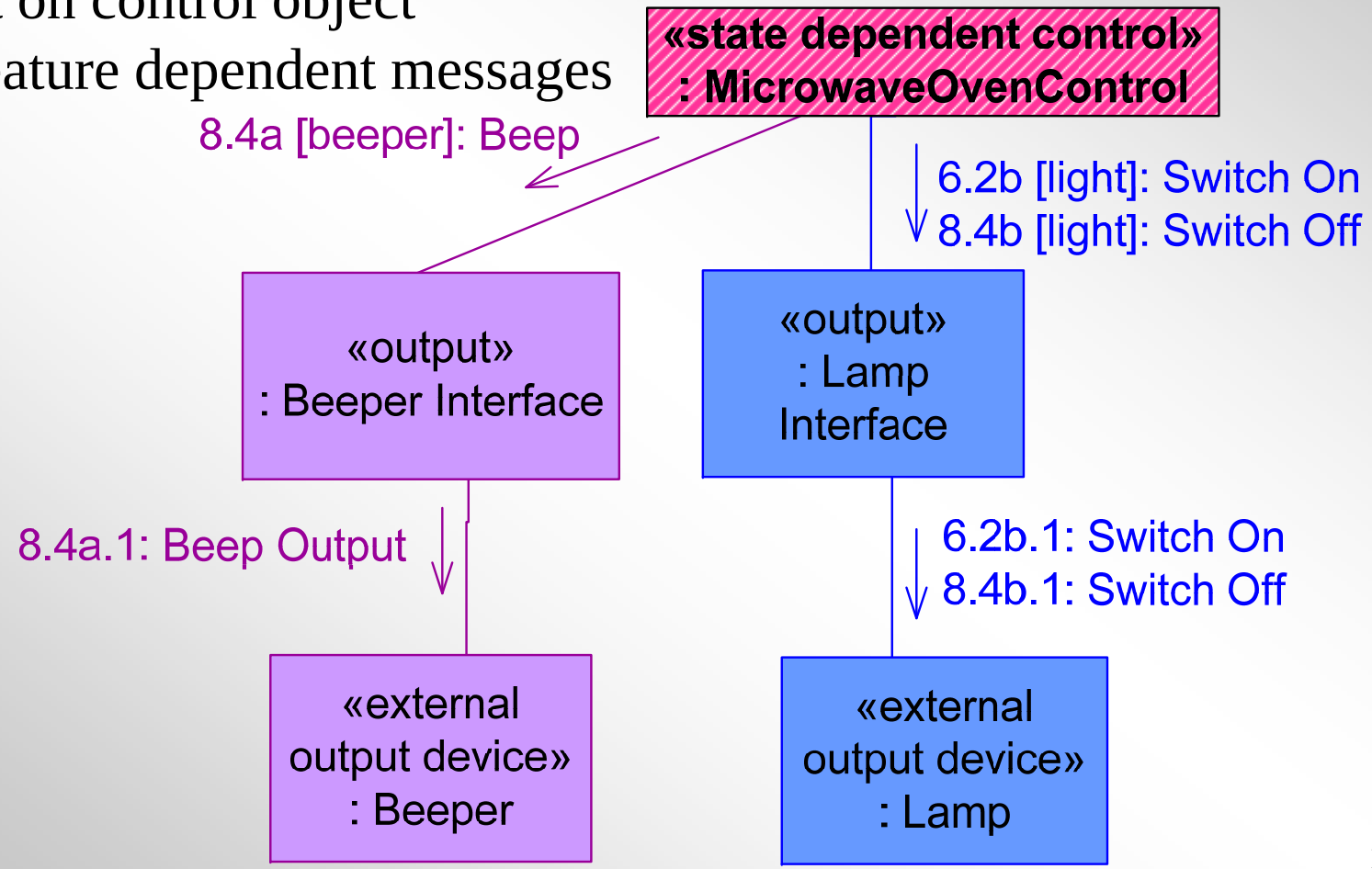


Feature Based Impact Analysis

- Feature dependent action
- Action is only executed if Feature Condition is True
  - Action [Feature Condition]
    - Switch On [light], Switch Off [light]
  - Feature condition is
    - True if feature Selected
- Example
  - Optional features: Light, Beeper
  - Consider impact on objects
  - Consider impact on statechart

Feature-Based Impact Analysis

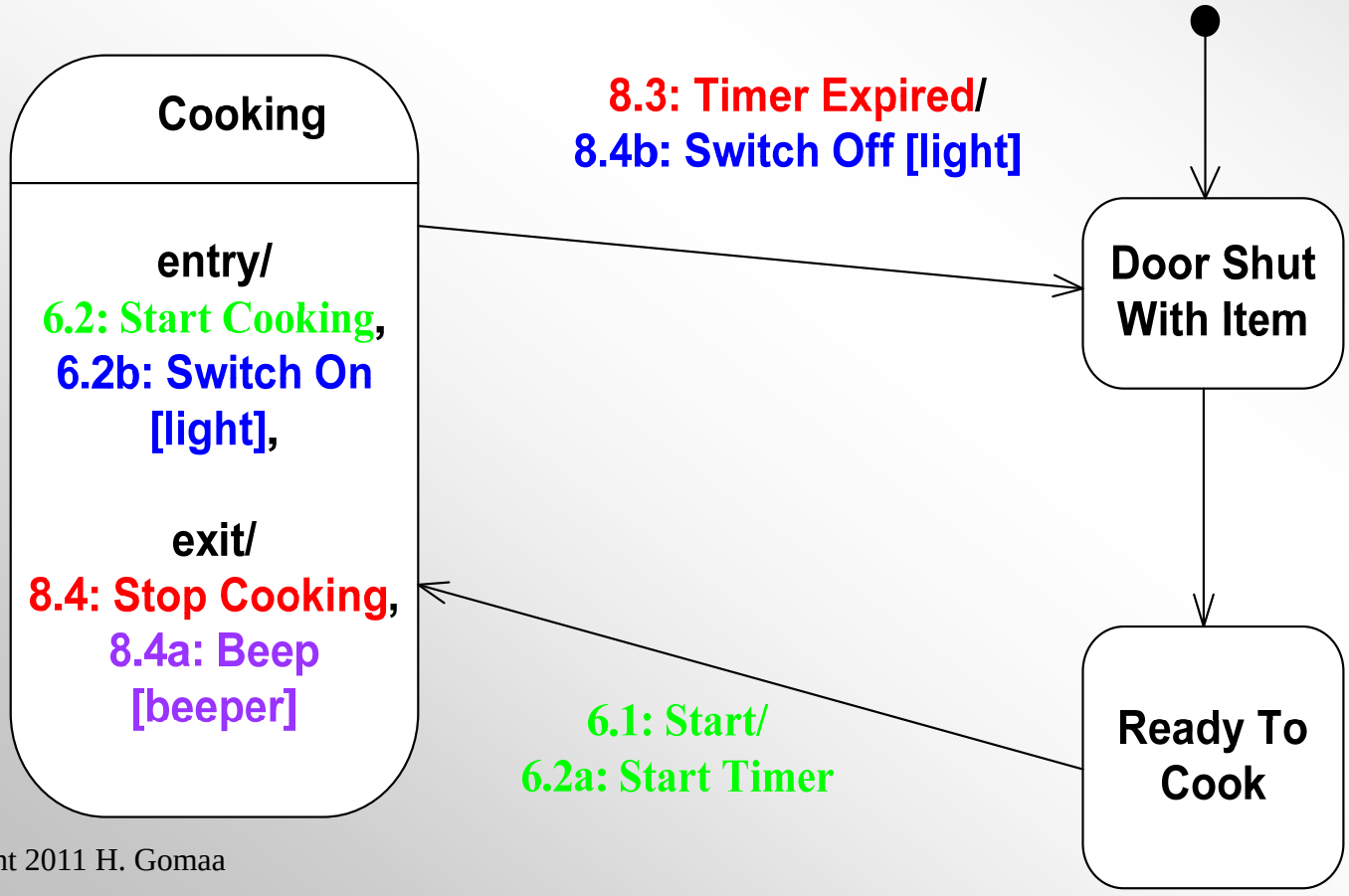
- Impact Analysis of Beeper and Light optional features
  - 2 optional objects added
  - Impact on control object
    - Feature dependent messages



Feature Based State Machine

Feature conditions: [light], [beeper]

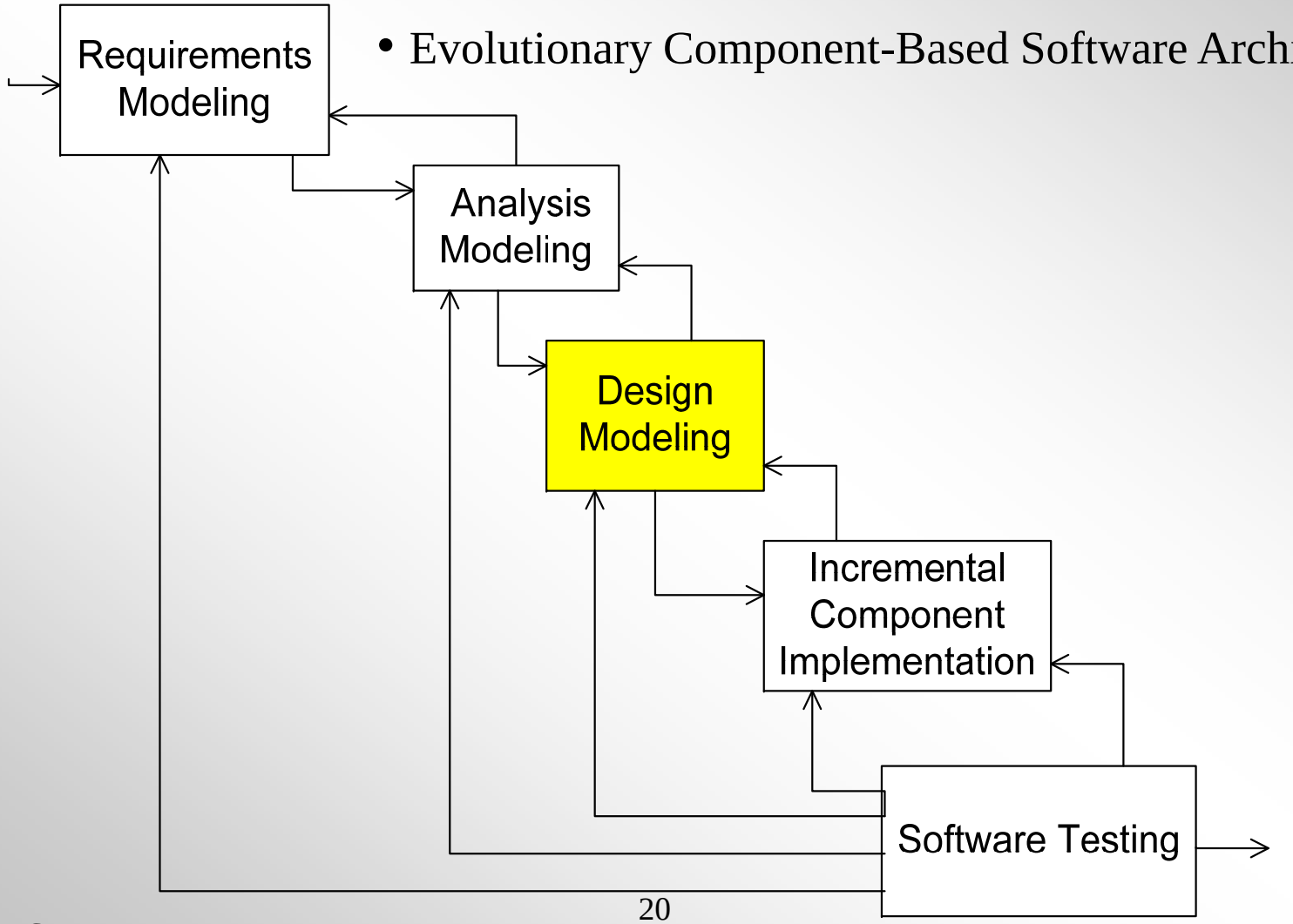
Feature dependent actions: Switch On [light], Switch Off [light], Beep [beeper]



Software Product Line Design Modeling

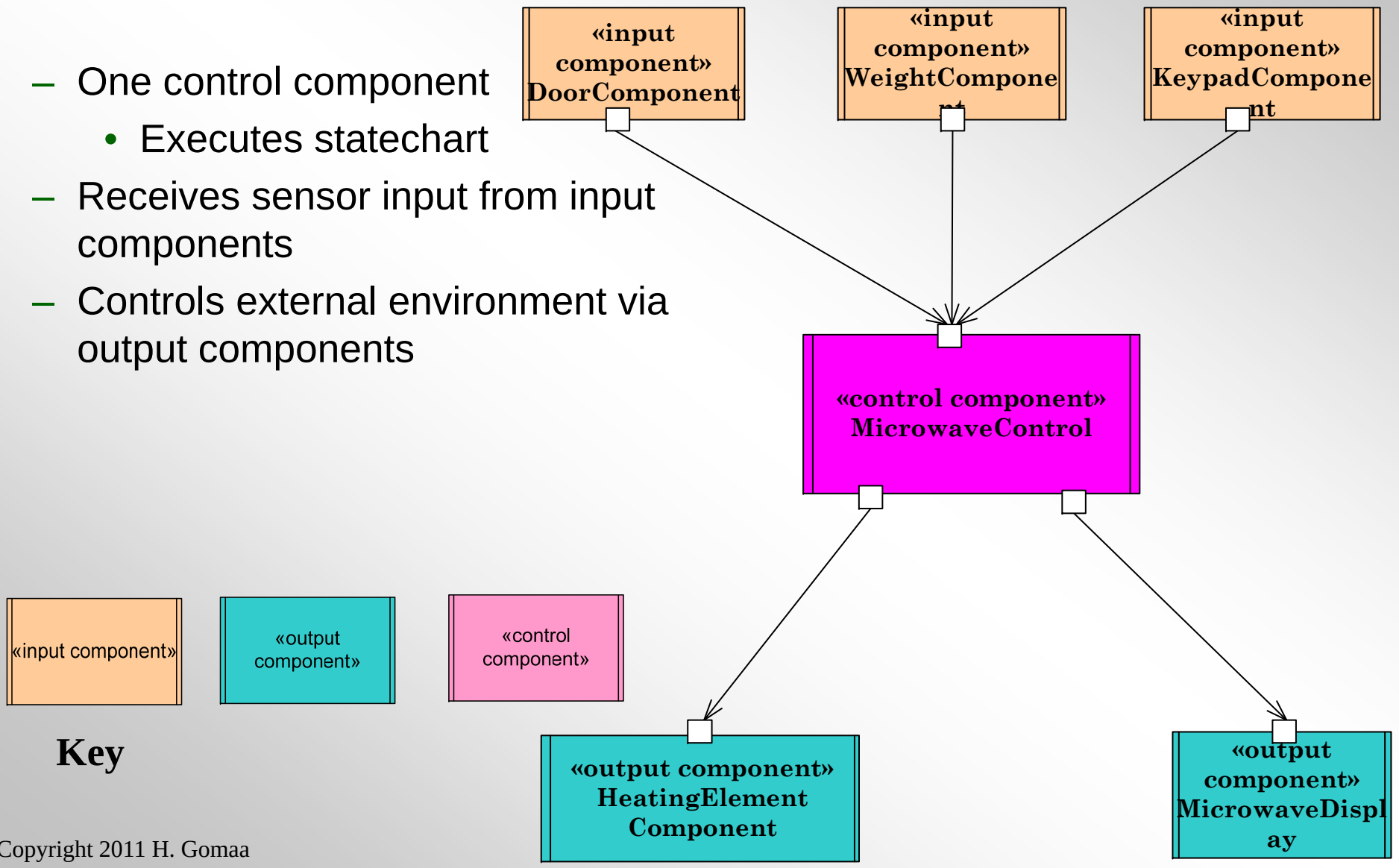
What should SPL Design Method provide?

- Software Architectural Patterns
- Evolutionary Component-Based Software Architectures



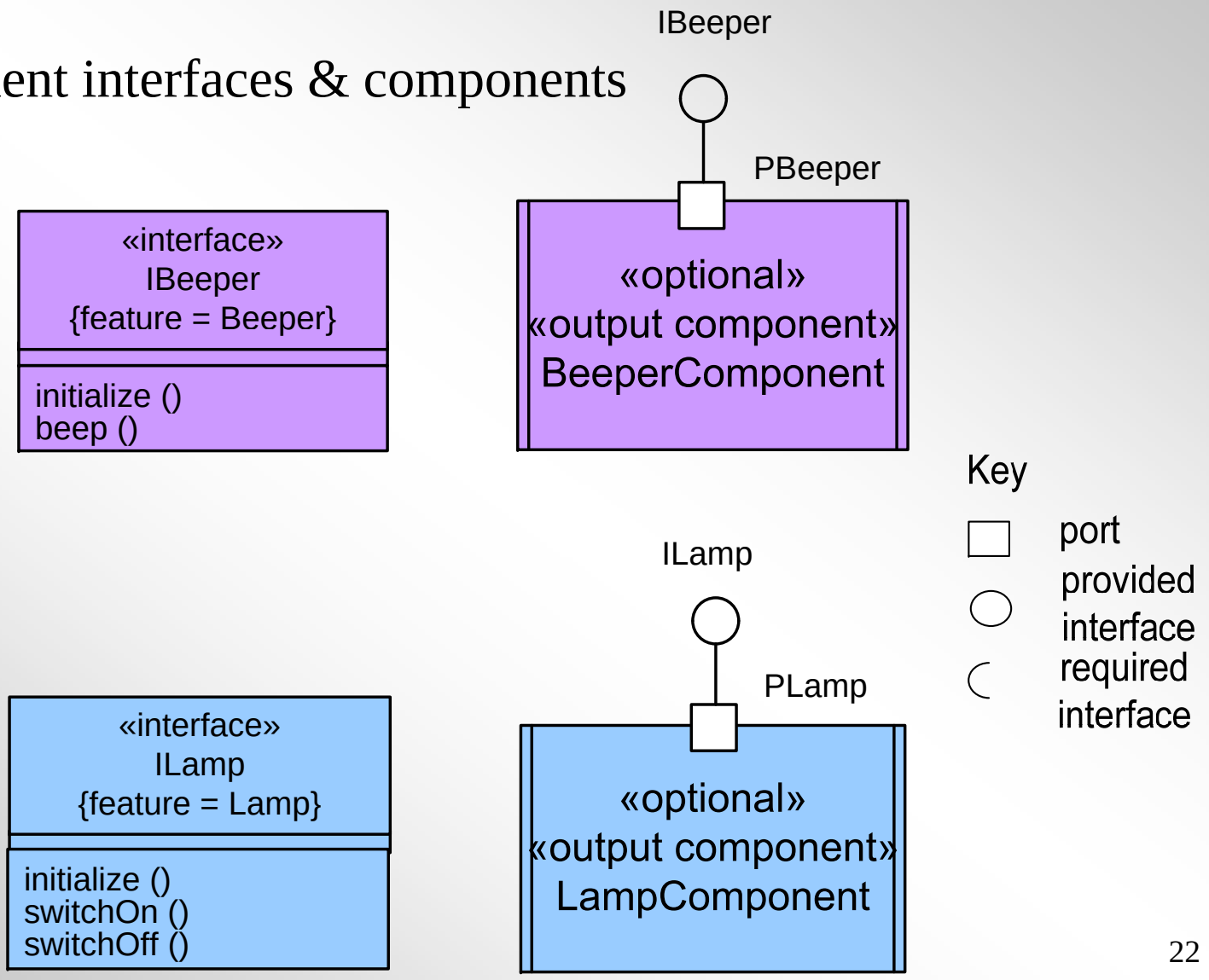
Develop Kernel Software Architecture
(Example based on Centralized Control Pattern)

- One control component
  - Executes statechart
- Receives sensor input from input components
- Controls external environment via output components



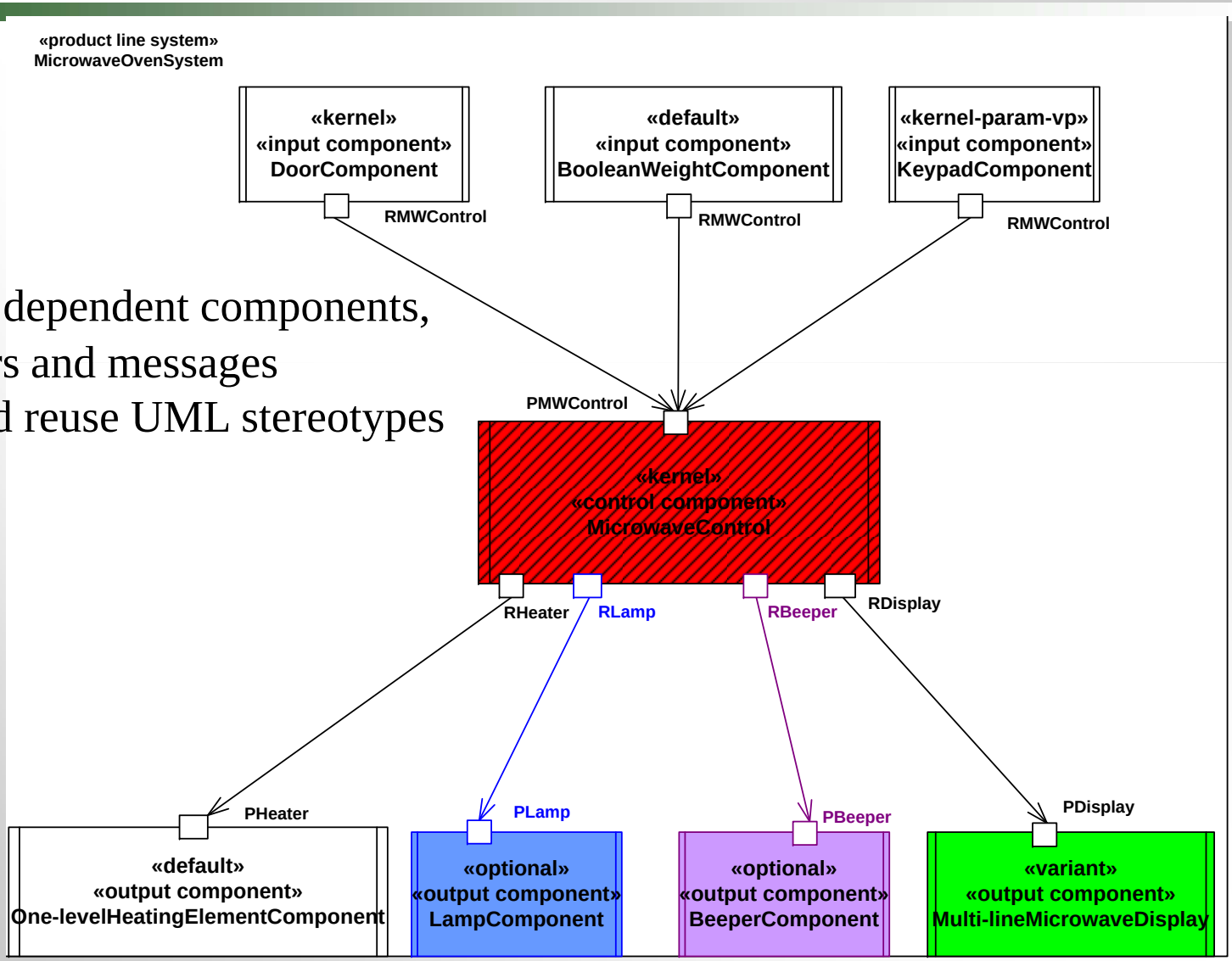
Design of Optional Components

- Feature dependent interfaces & components

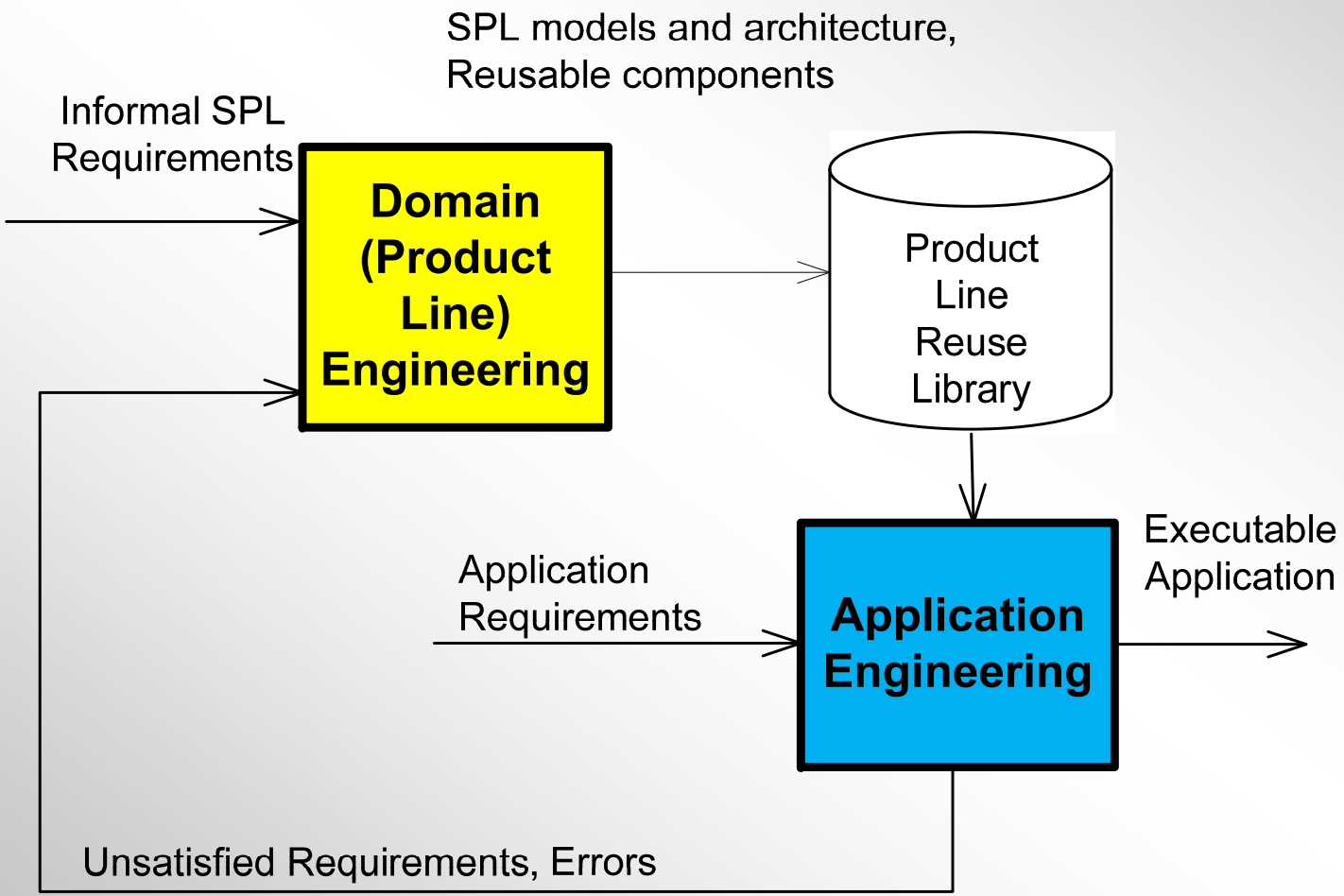


Feature-based Microwave Oven Architecture

- Feature dependent components, connectors and messages
- Role and reuse UML stereotypes



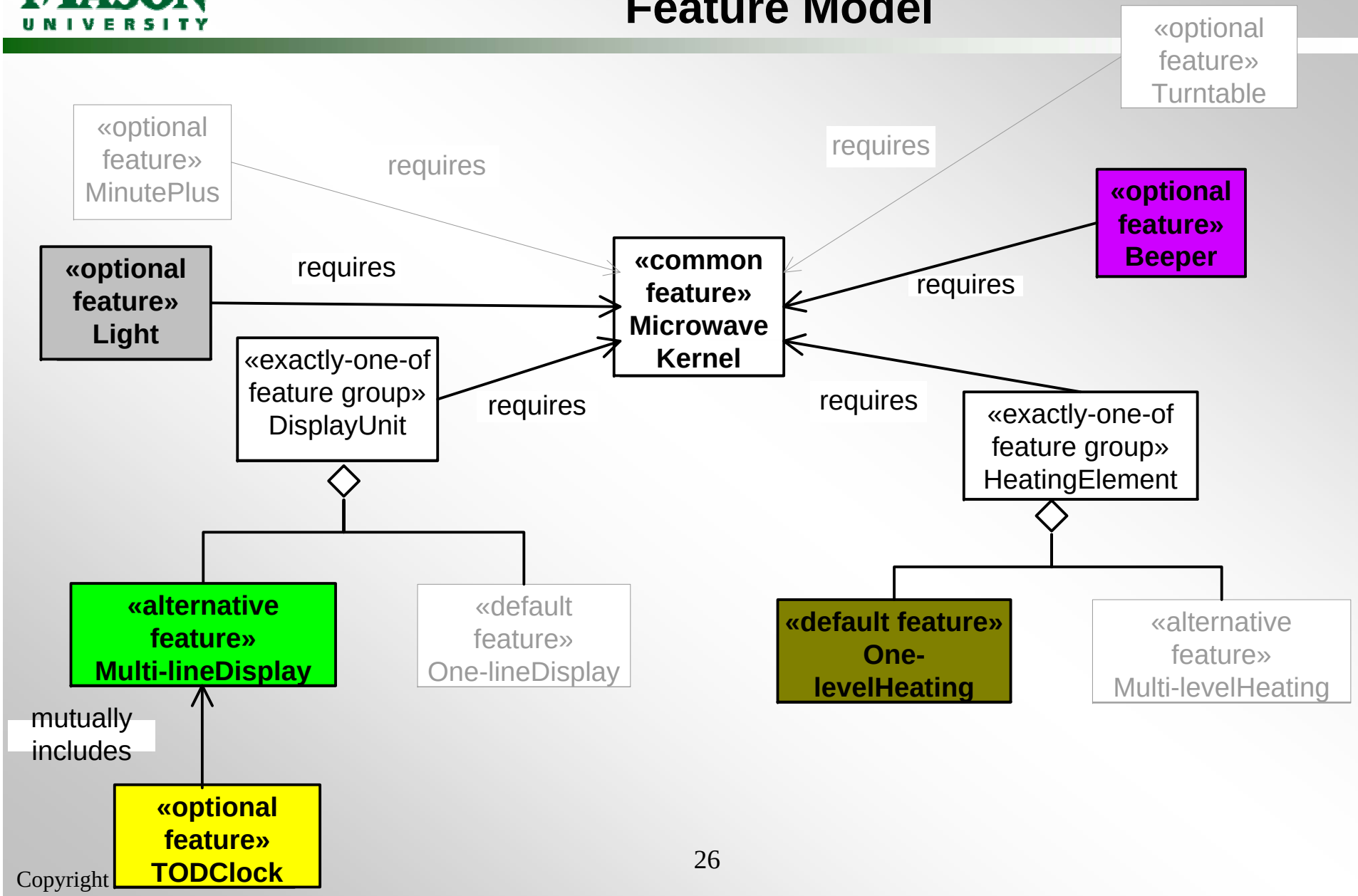
Application Engineering



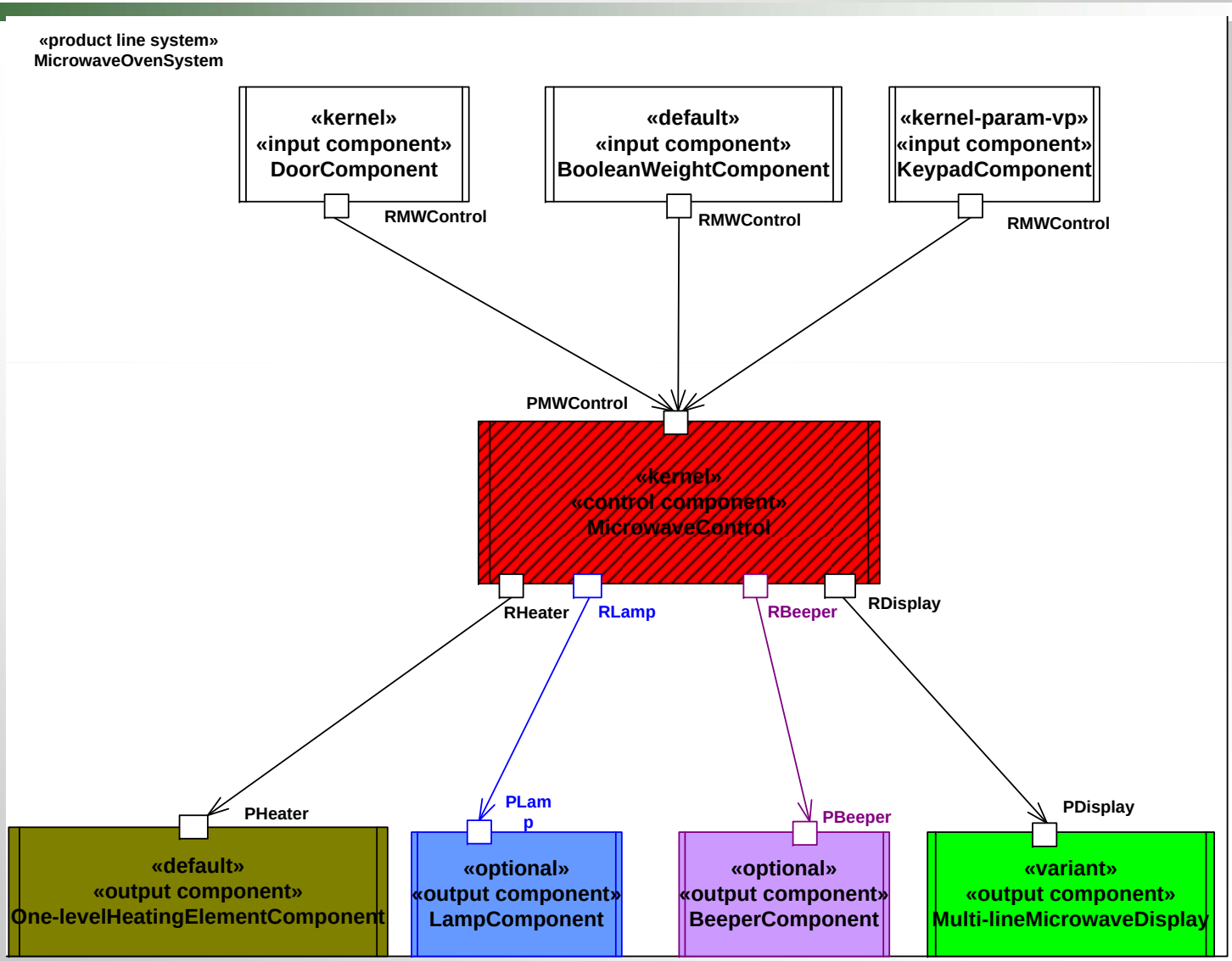
Software Application Engineering

- Software Application
  - Member of software product line
- Software Application Engineering
  - Derive application architecture from SPL architecture
- Select application features subject to
  - Feature dependencies and relationships
- Derive software application architecture
  - Kernel components always selected
  - Optional and variant components correspond to features selected

Microwave Oven Application Feature Model



Microwave Oven Application architecture



Conclusions

- Software Product Line Engineering
  - Software engineering for a family of products
- Software Variability
  - Key problem in software product line engineering
- Feature modeling
  - Unifying view in multiple view SPL
  - Feature relationships with other views explicitly depicted
- Evolutionary development
  - Kernel first approach for new SPL
  - Reverse SPL engineering from existing systems
  - Use feature modeling to model evolution of requirements

Related Research

- Multiple-view modeling and meta-modeling for SPL (with L. O'Hara, M. Shin, M. Abu-Matar)
- Software process modeling and SPL (with L. Kerschberg)
- Tool support for SPL development and product derivation (with V. Sugumaran, M. Shin, M. Abu-Matar)
- Model-based SPL Testing (with E. Olimpiew)
- Aspect-oriented modeling and SPL (with M. Shin)
- Executable Architectural Patterns for RT SPLs (with J. S. Fant)
- Feature Modeling & Variability Modeling in Service-oriented SPL (with M. Abu-Matar)