Variability Management in Software Product Line Engineering

Kyo Chul Kang, POSTECH Fellow

**International Advanced School on
Automotive Software Engineering**

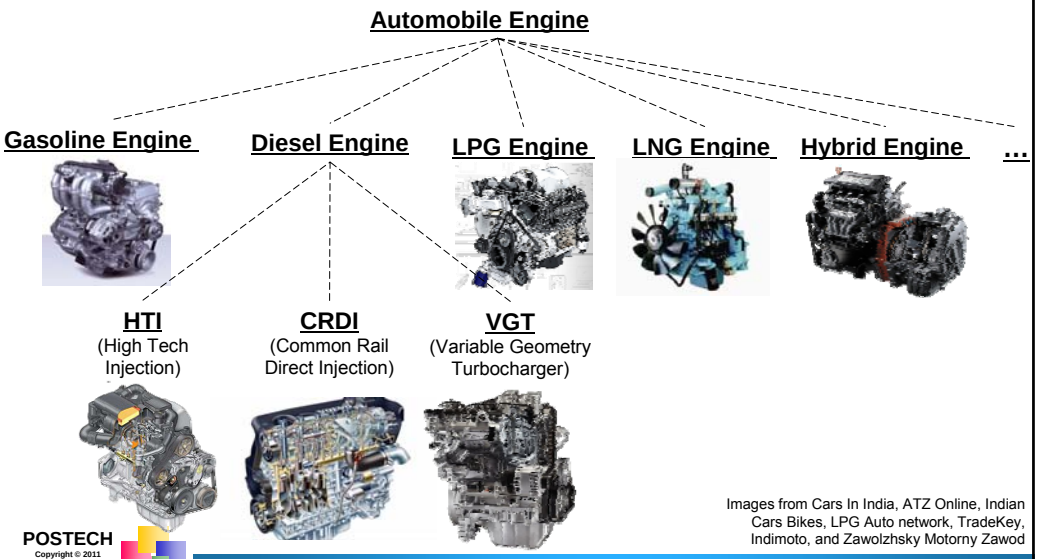
**March 7-8, 2011
Nagoya, Japan
Pohang University of Science and Technology (POSTECH)**

About me

- At the University of Michigan (ISDOS Project)
  - Requirements engineering (PSL/PSA, Meta System)
  - Meta modeling (Meta System, System Encyclopedia Manager)
- At Bellcore and Bell Labs
  - Experienced software reuse issues in industry
- At SEI
  - Developed FODA, a commonality/variability analysis method
- AT POSTECH
  - Extended FODA and created a successor (FORM)
  - Developing a CASE environment (ASADAL)

Prologue

- Variety of automobile engines



Images from Cars In India, ATZ Online, Indian Cars Bikes, LPG Auto network, TradeKey, Indimoto, and Zawolzhsky Motorny Zawod

Prologue

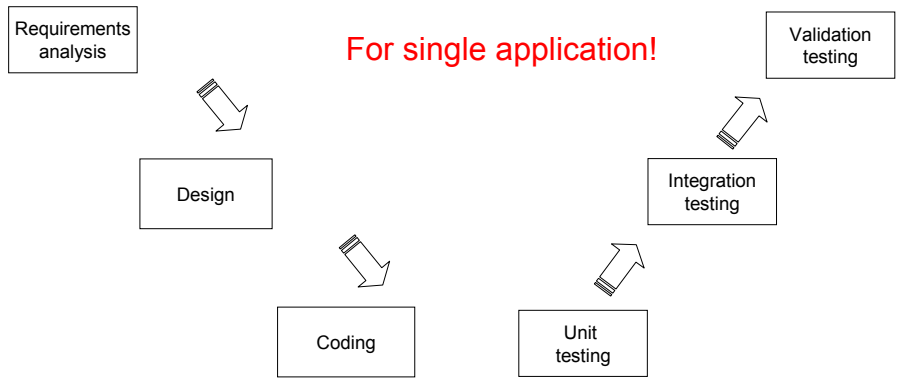
- Soft software lives forever; Hard software has no life
- We are developing software as if it is hardware
- Product line software engineering is about making software soft
- Product line engineering shows what software engineering ought to be

Agenda

- **Traditional Software Engineering**
- Product Line Engineering and Variability Management
- Feature-Oriented Product Line Engineering
- Product Line Adoption Issues

Traditional Software Engineering

- Typical software engineering process



Traditional Software Engineering

- Development of single application system
- Requirements analyst
  - Elicit and specify requirements for a single target system
- Designer
  - Design for the specified requirements

But what happens

- Software maintenance requires
  - Adding new functions
  - Removing existing functions
  - Changing functions
  - Correcting errors and improving performance
- But
  - Source code modification without refactoring
  - Copy-and-modify reuse

As the results

- Proliferation of versions
- Design and code decay
  - Bad structure
  - Spaghetti code
  - Unused code
  - Brittle code
- Software maintenance can cost as much as 80% of the entire lifecycle cost

The problem

- Software is developed like hardware
- There is little effort to build "Softness" into software
- The traditional approach can no longer support development of products with
  - Diverse market needs
  - Time-to-market pressure
  - Fierce feature competition

Softness of Software

- Why important
  - Functionality of products implemented as software (e.g., more than 10M lines of code for TV products)
  - Diverse market needs
  - Time-to-market pressure (e.g., new TV models every 3-6 months)
  - Software as valuable asset of an organization
    - Accumulated systems development knowledge is packaged as reusable asset

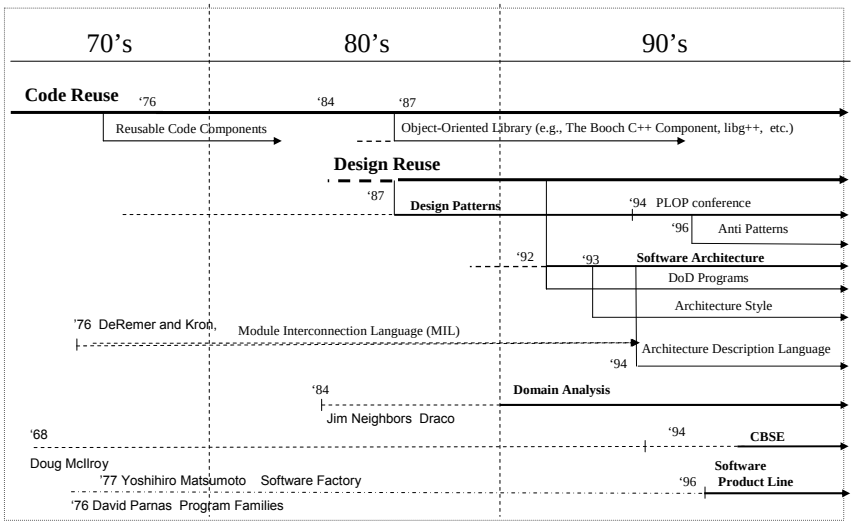
Softness of Software

- Softness
  - Maintainability
  - Adaptability
  - Portability
  - Interoperability
  - Reusability

Related engineering principles and techniques

- Modularity
- Abstraction
- "Layering"
- David Parnas'
  - Information hiding
  - Program families
- Meta programming and application generators
- There are many mechanisms to use; How do we know what to "hide"

Evolution of Reuse Concepts



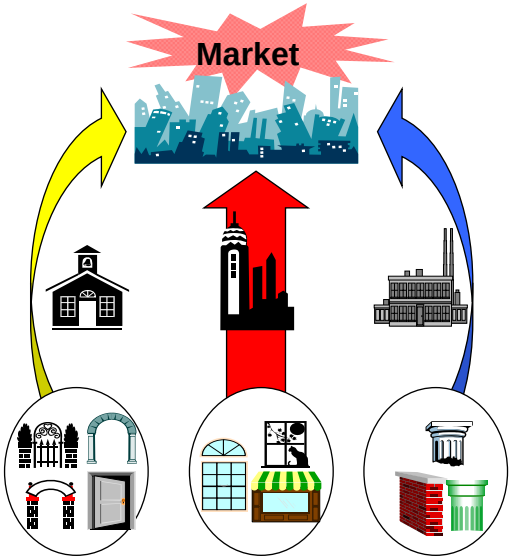
Agenda

- Maintainability and Reusability
- **Product Line Engineering and Variability Management**
- Feature-Oriented Product Line Engineering
- Product Line Adoption Issues

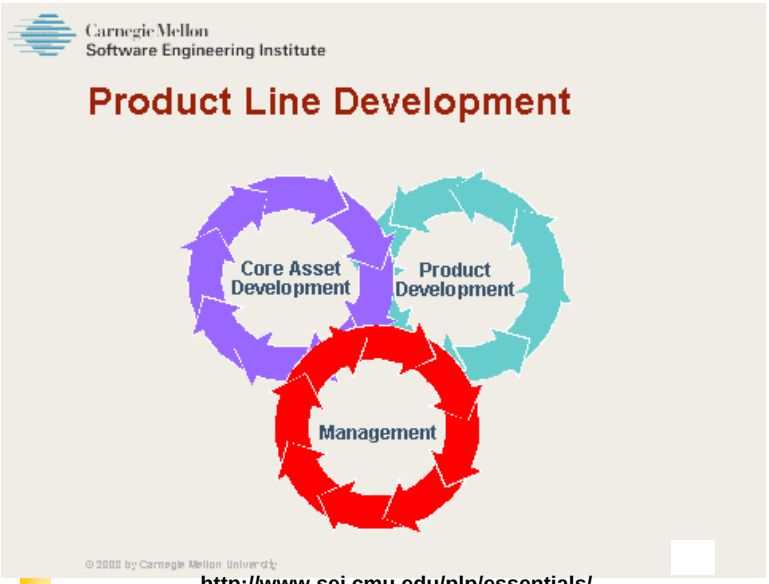
Product Line Engineering

- Systematic Reuse in the Context of a Product Line: "Building softness into software"
  - Product line: "a family of systems sharing a common set of features"

- Software Product Line Engineering (SPLE) is an emerging software engineering paradigm, which guides organizations toward the development of products from core assets rather than the development of products one by one from scratch.



SEI: Software Product Line Practice



http://www.sei.cmu.edu/plp/essentials/

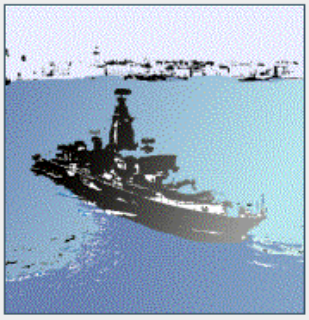
Celsius Tech

Carnegie Mellon
Software Engineering Institute

CelsiusTech: Ship System 2000

A family of 55 ship systems

integration test of 1-1.5 million SLOC requires 1-2 people
rehosting to a new platform/OS takes 3 months
cost and schedule targets are predictably met
performance/distribution behavior known in advance
customer satisfaction is high
hardware-to-software cost ratio changed from 35:65 to 80:20



10

http://www.sei.cmu.edu/plp/essentials/

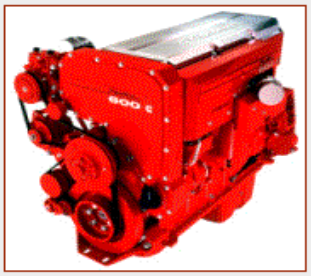
Cummins Inc.

Carnegie Mellon
Software Engineering Institute

Cummins Inc.: Diesel Engine Control Systems

Over 20 product groups with over 1000 separate engine applications

product cycle time was slashed from 250 person-months to a few person-months
Build and integration time was reduced from one year to one week
quality goals are exceeded
customer satisfaction is high
product schedules are met



11

http://www.sei.cmu.edu/plp/essentials/

- Important technical elements of product line engineering
  - Commonality and Variability
  - Architecture (structure, “bone”)
    - Stability based on common properties
  - Variation points and Variants
    - Flexibility based on expected variations
  - Encapsulation of design decisions that may change
    - Information hiding, abstraction, etc.

Technical Advances

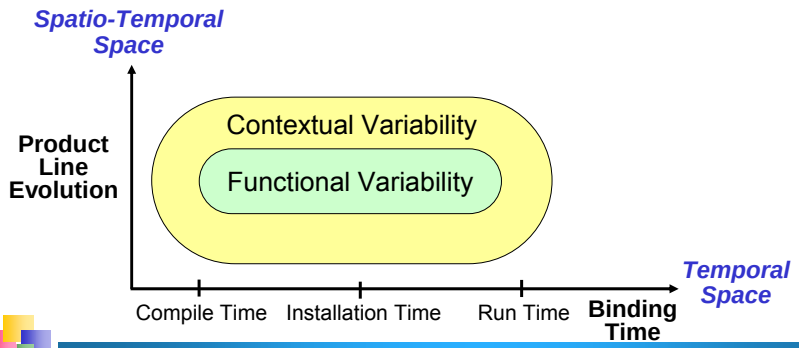
- Paradigm change
  - From single systems to product line/family
  - “Good software engineering” focusing on maintainability and reusability
- Commonality and variability analysis
  - Feature analysis
- Domain-oriented Architectures and Components (from objects and collaborations)
  - Variation points and variants
  - “high option potentials”
- Domain specific languages and generators

Basic Principles

- How do we develop product line software?
  - Variability analysis: Think about what may change!
    - Looking across a family of applications in the product line
      - In different markets
    - Looking ahead for anticipated changes
      - Expected requirements from a market analysis
      - Emerging markets
      - Study of emerging technologies
    - Exploring product usage contexts
      - Both static and dynamic dimensions
  - Building variability into software
    - Architecture: Variation points and variants
    - Component: Hide (encapsulate) design decisions that may change!
      - Abstract and expose unchanging functional properties as interface!

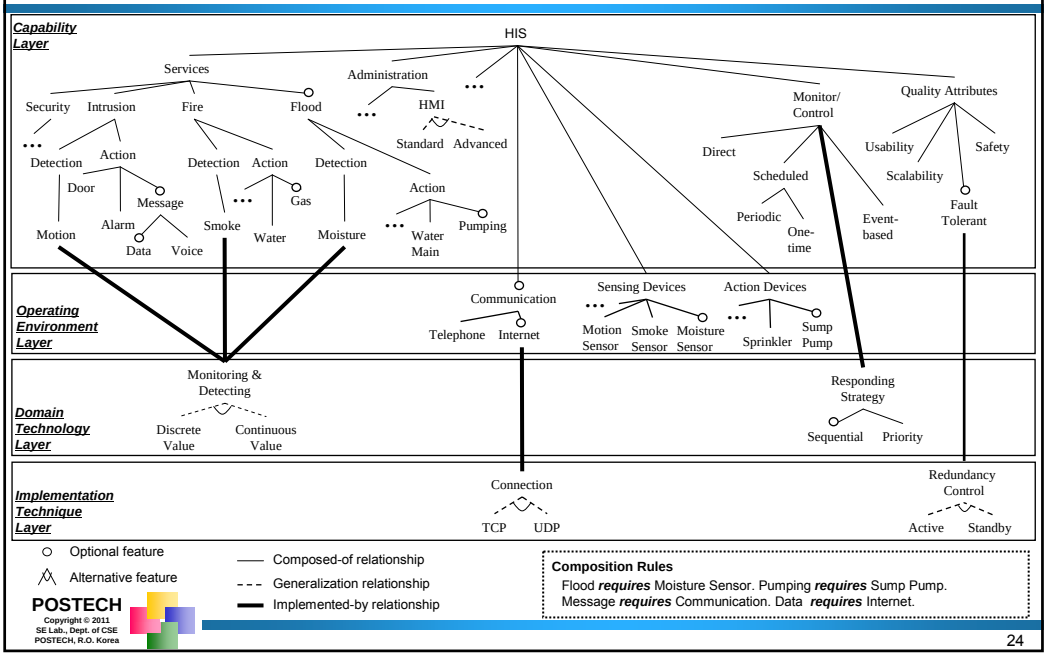
Commonality and Variability

- Functional Variability
- Contextual Variability
- Binding Time Variability
- Evolution of Functional and Contextual Variability



Feature Model Example: HIS Product Line

Functional Variability



Contextual Variability

- Product contexts
  - Different/evolving operating environments (e.g., technologies)
  - Different legal and cultural constraints
  - Different marketing strategies
  - Different/ evolving market needs
  - Dynamically changing usage contexts

Usage Context Example: Elevators

Contextual Variability

Safety & Comfortability

Passenger Elevator

Safety & Capacity

Hospital/Bed Elevator

Capacity

Freight Elevator

Three Perspectives of Feature Binding

Binding Time Variability

What features will be included in the products?

When they will be included?

- Asset or product development time

- Pre-operation time

- Operation time

When they will be available?

- Asset or product development time

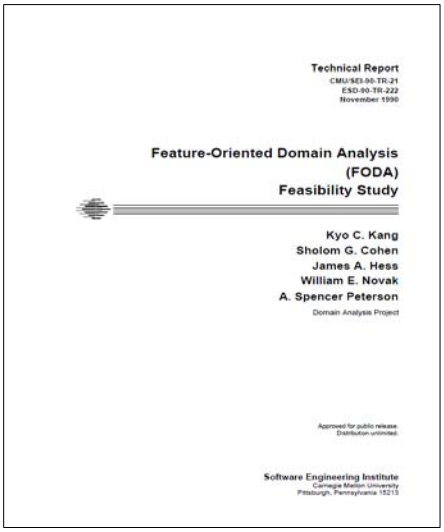
- Pre-operation time

- Operation time

Agenda

- Maintainability and Reusability
- Product Line Engineering and Variability Management
- **Feature-Oriented Product Line Engineering**
- Product Line Adoption Issues

Feature-Oriented Domain Analysis (FODA)


Technical Report
CMU/SEI-90-TR-21
ESD-90-TR-222
November 1990

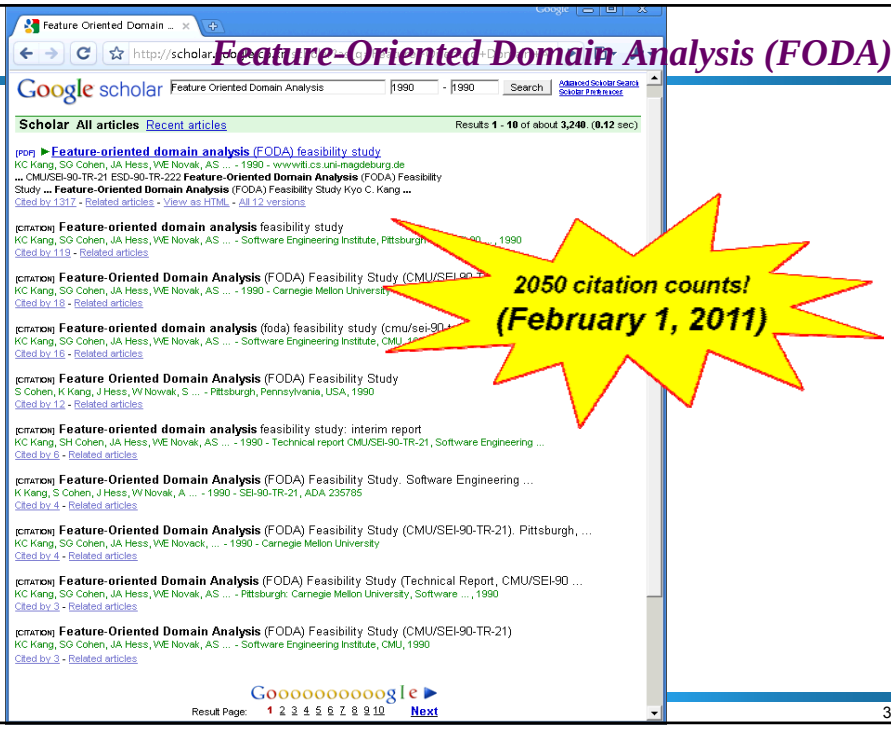
Feature-Oriented Domain Analysis
(FODA)
Feasibility Study

Kyo C. Kang
Sholom G. Cohen
James A. Hess
William E. Novak
A. Spencer Peterson
Domain Analysis Project

Approved for public release.
Distribution unlimited.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

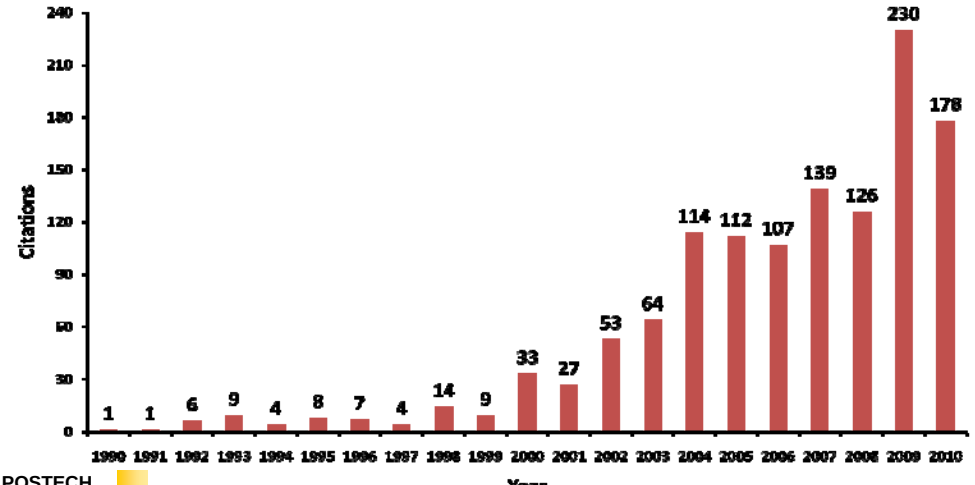
Feature-Oriented Domain Analysis (FODA)



Feature-Oriented Domain Analysis (FODA)

Number of Citations

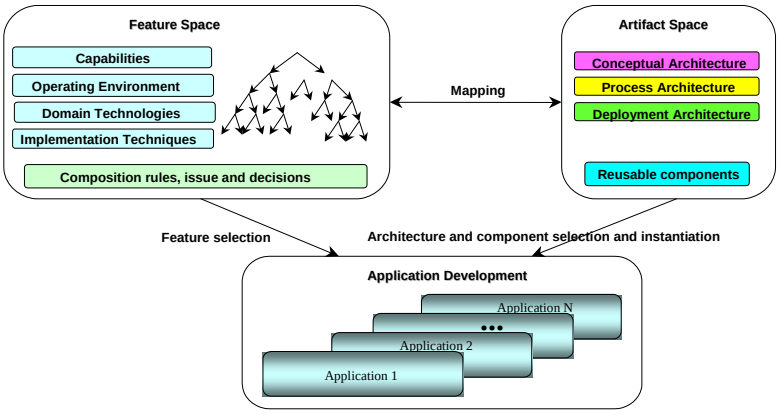
1990~2010
Total: 2050
Unknown/Overlapping: 804



POSTECH
Copyright © 2011
SE Lab., Dept. of CSE
POSTECH, R.O. Korea
31

Feature-Oriented Domain Analysis (FODA)

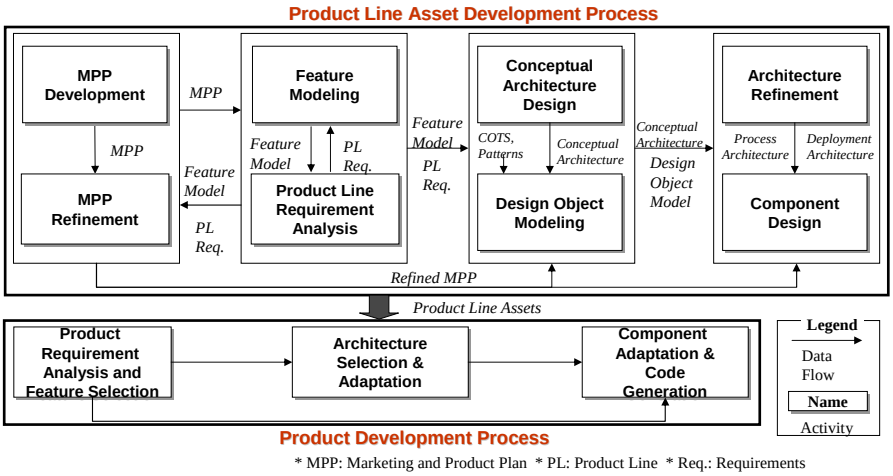
Method Concept



The core of FORM lies in the analysis of domain features and the use of these features to develop reusable and adaptable domain artifacts.

Feature-Oriented Product Line Engineering

- FORM (Feature-Oriented Reuse Method)



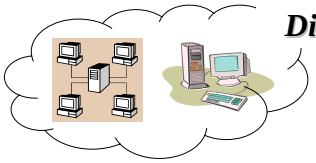
* MPP: Marketing and Product Plan * PL: Product Line * Req.: Requirements

Kyo C. Kang, Jaejoon Lee, and Patrick Donohoe, Feature-Oriented Product Line Engineering, IEEE Software, Vol. 9, No. 4, Jul./Aug. 2002, pp. 58-65.

Marketing and Product Plan (MPP)

Different cultural traits and legal constraints

Different computing knowledge backgrounds



Different computing environments

Marketing Plan

A *marketing plan* includes a market analysis with an assessment of the market, and a marketing strategy with a plan for realizing the business opportunities with products that meets the needs.

The market analysis includes:

- need assessment
- customer profiles
  - end-user skill levels
  - cultural and legal constraints
- business opportunities
  - price range
  - time to market

The marketing strategy may initially include:

- an outline of product delivery methods : how the products will be delivered to customers

Elements of MPP

Product Line Initiation

Marketing Plan (Business Concerns)

Market analysis

- Market segment
  - Needs assessment
  - User profile
  - Cultural and legal constraints
- Business opportunities
  - Time to market,
  - Price range, etc.

Marketing strategy

- Product delivery methods
- Other business considerations

Product Plan (Engineering Concerns)

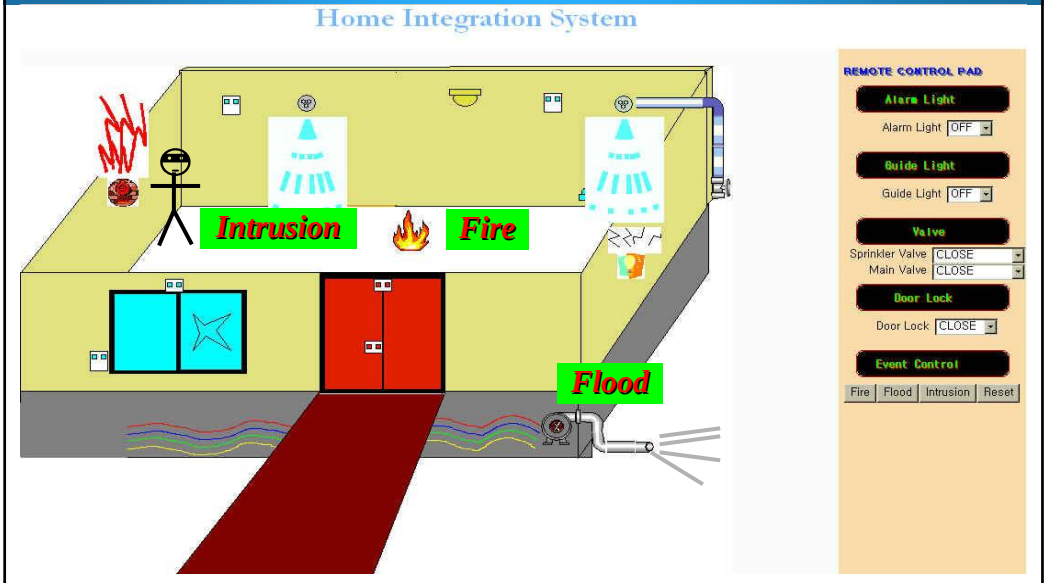
Product features

- Product functional features
  - Feature lists
  - Feature description
- Non-functional features
  - Usability, scalability, etc.

Product features delivery methods

- Feature coverage
- Feature binding time
- Feature binding techniques

FORM Application: Home Integration System (HIS) Product Line



Product Plan

Once the marketing plan has been defined, it is important to spend some effort on identifying the characteristics of products in a product line in terms of features and developing a plan for incorporating features.

The *product plan* includes

- Product features:
- Product feature-delivery methods:
- Feature release plan for future:

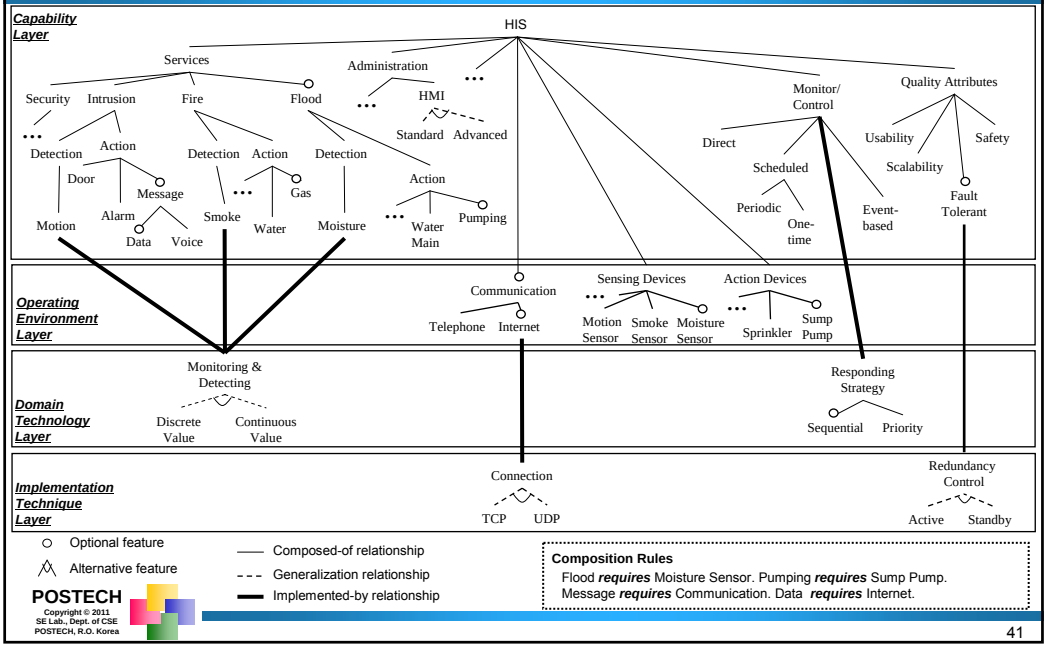
MPP for HIS PL

Product Line Initiation

Marketing and Product Plan for HIS PL		
Market Segments	**Office building: High-End (HE) product**	**Household: Low-End (LE) product**
User/Maintainer Profile	Dedicated engineers with computer science backgrounds.	No computer knowledge is assumed.
Legal Constraints	Emergency control services must conform to codes of each country.	Emergency control services must conform to codes of each country.
Feature Delivery Method	Feature selection from a predefined set of features (Feature Selection Method)	Prepackaged Method
Product Features	Fire, Intrusion, Flood, Security, and other customer specific features	Fire, Intrusion, Flood
Quality Attributes	Safety, Reliability, Scalability	Safety, Reliability, Scalability, Usability
Product Feature Binding Time	Product Delivery Time	Product Build Time

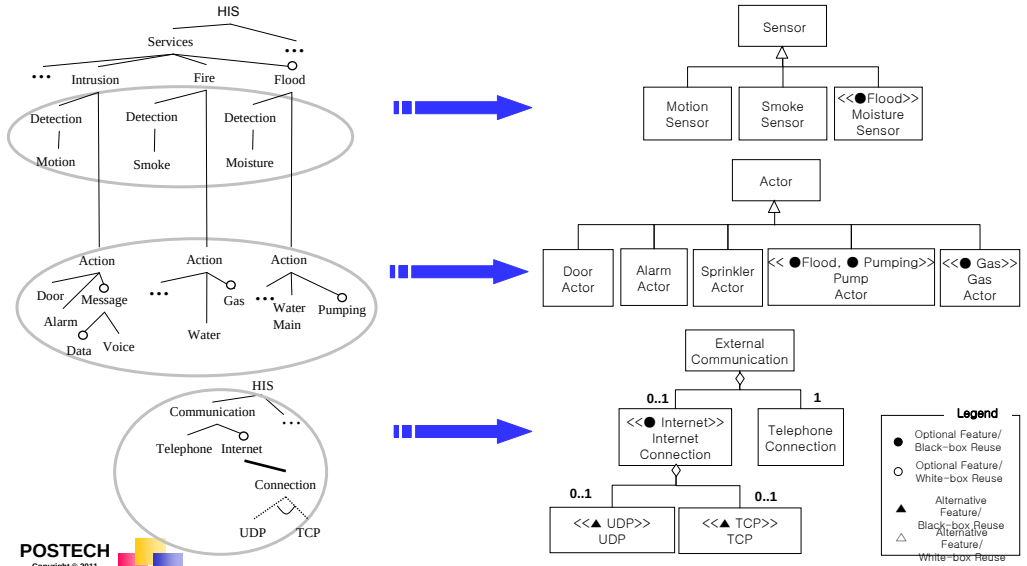
Feature Model Example: HIS Product Line

Product Line Analysis



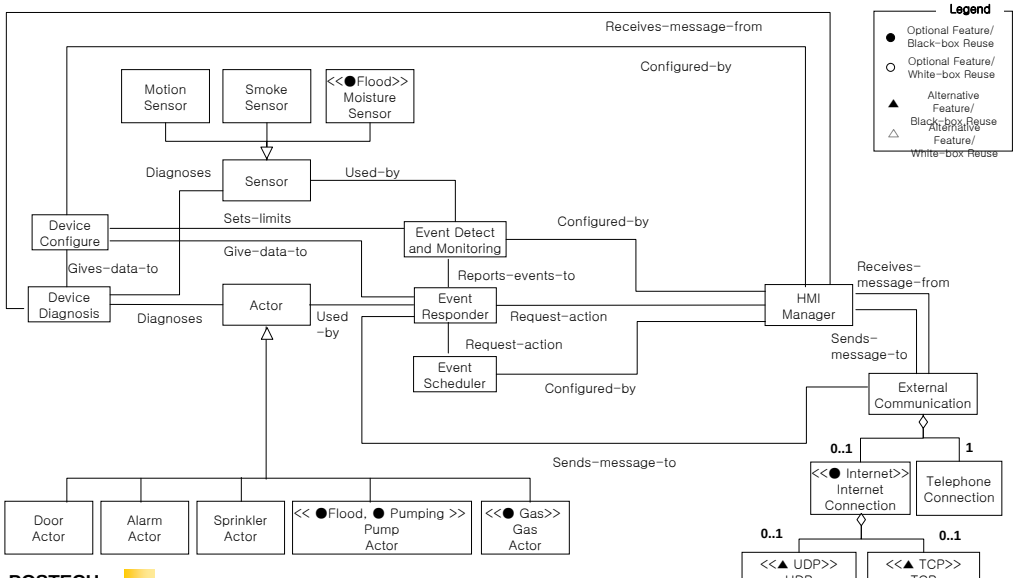
Object Identification Example from the Feature Model

Design Object Modeling



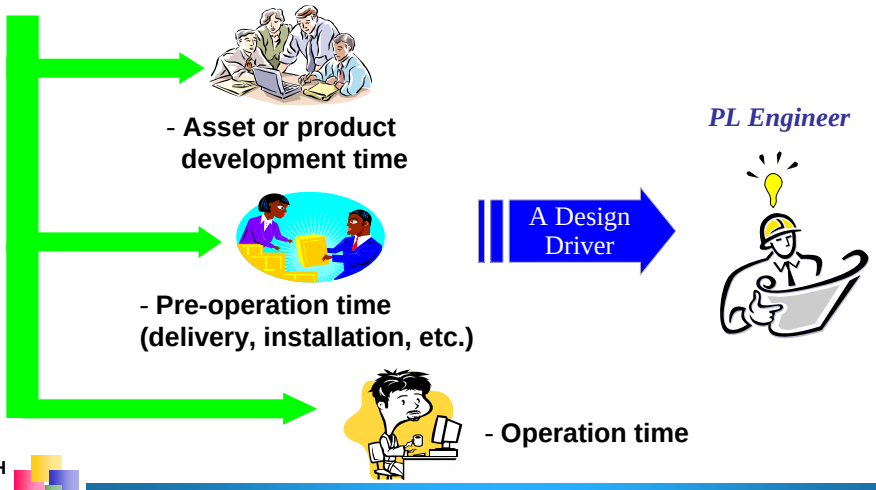
Design Object Model Example

Design Object Modeling

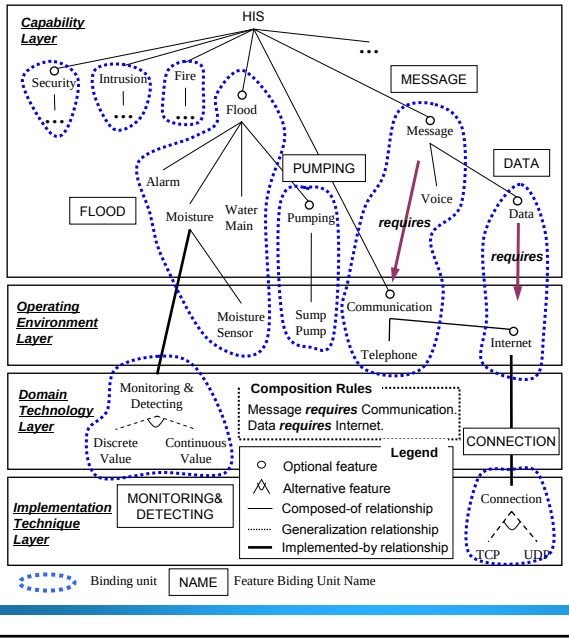


Feature Binding Analysis

- Feature binding: When and how features are included to products and delivered to customers.



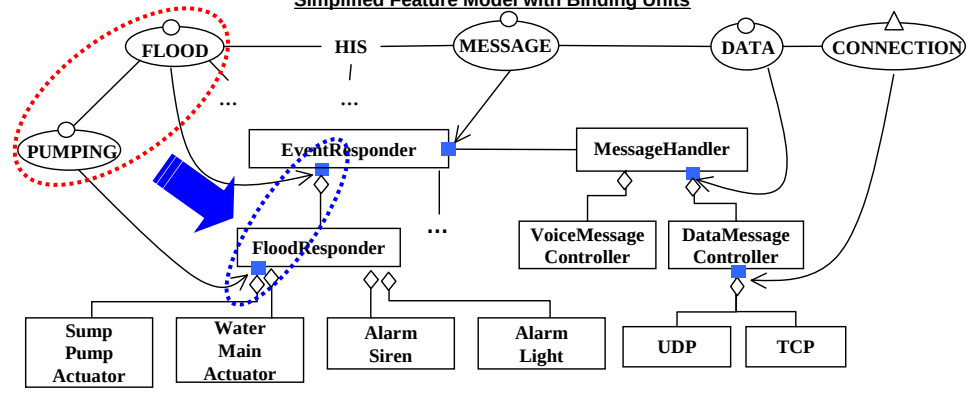
Feature Binding Units



Variation Point Identification

Binding Dependency

Simplified Feature Model with Binding Units

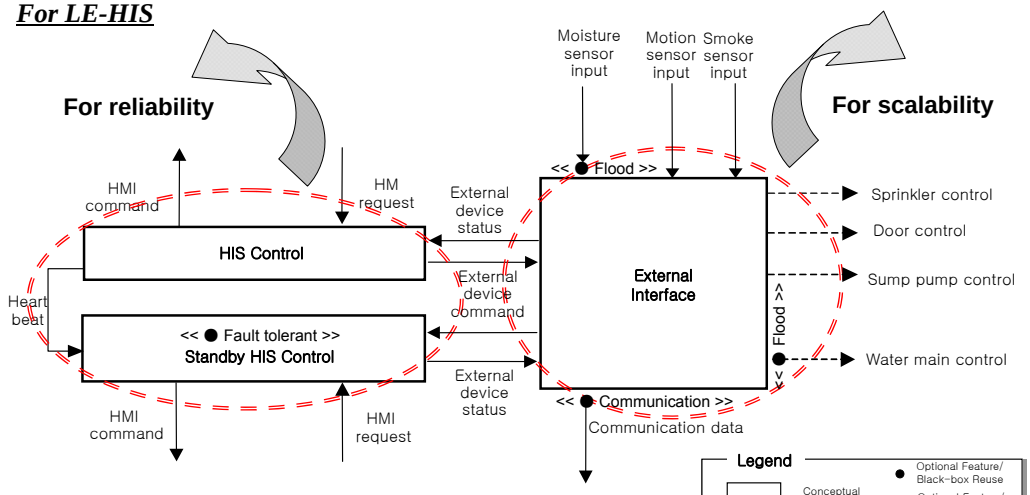


Refinement of "Event Responder" Object

Conceptual Architecture

Architecture Design

For LE-HIS



What is Product Development?

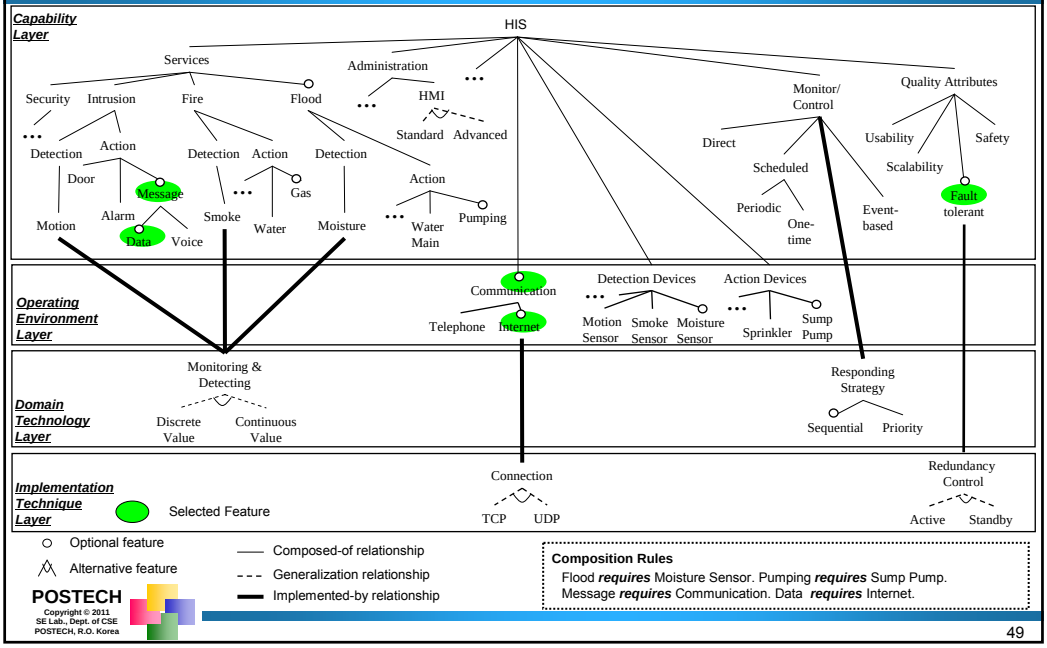
Product Development

- Product development is a process of developing a specific product making use of the product line asset developed during the product line asset development process.
- Product development proceeds by:
  - analyzing user's requirements,
  - selecting appropriate and valid product line features from the feature model,
  - identifying the corresponding architecture models,
  - completing the product development by reusing software components, and
  - adapting components as needed.

Feature Selection for LE-HIS

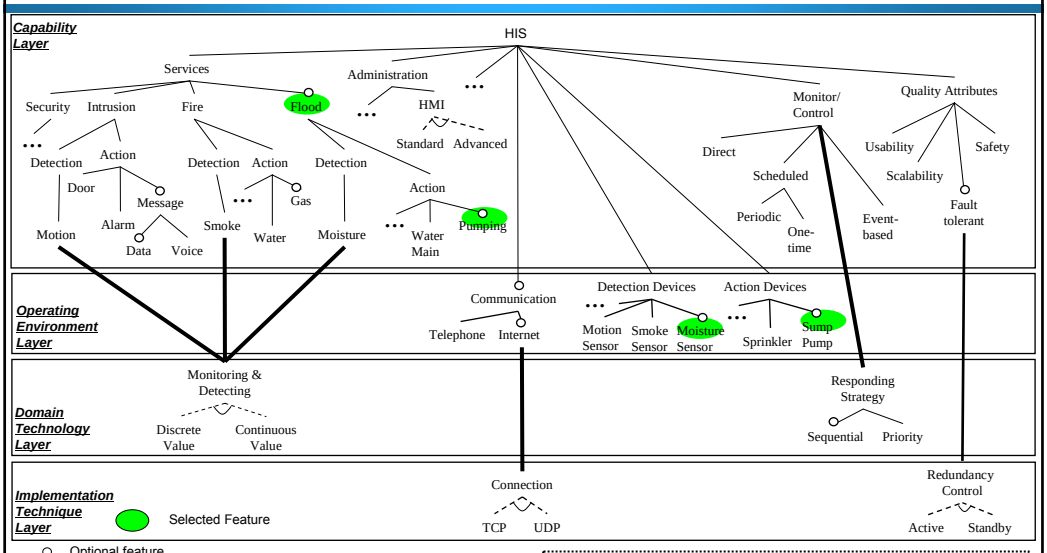
with **the fault tolerant and** ***without*** **the flood feature**

Product Development



Feature Selection for LE-HIS
without **the fault tolerant and** ***with*** **the flood feature**

Product Development



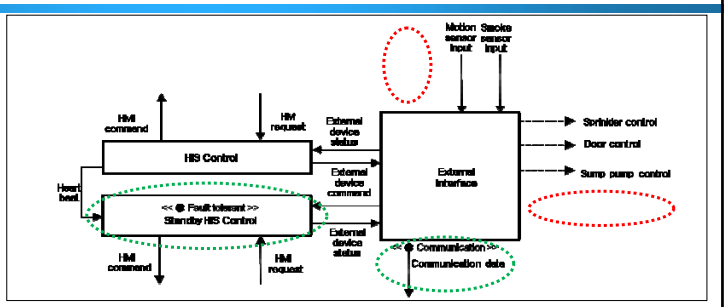
Composition Rules

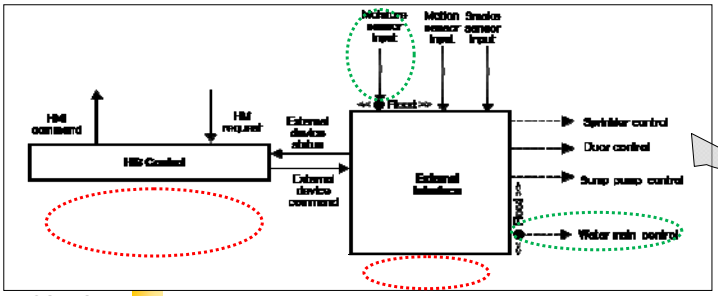
Flood ***requires*** Moisture Sensor. Pumping ***requires*** Sump Pump.
Message ***requires*** Communication. Data ***requires*** Internet.

Conceptual Architecture Selection

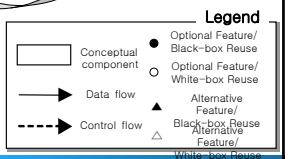
Product Development

With the **Fault Tolerant** and **Communication** features, without the **Flood** feature





With the **Flood** feature, without the **Fault Tolerant** and **Communication** features



Agenda

- Maintainability and Reusability
- Product Line Engineering and Variability Management
- Feature-Oriented Product Line Engineering
- **Product Line Adoption Issues**

Process *Adoption Issues*

- How to change to PL-based organization
  - How to evolve: staged process model for reuse adoption
  - Key process areas
    - Best practices
  - Metrics
    - Key indicators: cost of production, time to market, project completion time, etc.
    - Relationship between reuse, quality, and productivity
    - Relationship between reuse and ROI for sustainability of a reuse program
- Process models
  - Proactive vs. reactive vs. extractive models
    - Best practices
  - PL process vs. agile methods

Management — *Adoption Issues*

- ROI analysis
  - Estimating ROI from a reuse program
  - Estimating benefits from strategic market position
- Asset management (How to make PL-based development happen in an organization)
  - Who should develop assets (with variation points)
  - Who should maintain assets (variation management)
  - Who will be responsible for quality assurance
  - Who should enforce the use of assets
  - Models (best practices)
    - Centralized vs. distributed

Variability Management — *Adoption Issues*

- Discovery and Modeling
  - Functions
  - Non-functional attributes
  - Usage and operating contexts
- Configuration
  - Decision model
  - Rationales
  - Goals and Issues
- Management
  - Centralized
  - Distributed

Epilogue

- For good software engineering, maintainability and reusability must be built into software while designing!
- For softness , design must base on commonalities and variations of a product line
  - Think about what may change!
    - Look across similar applications
    - Look ahead for anticipated changes
    - Look at product contexts as well as functionality

Epilogue

Thank you!