Software Composition Paradigms

Sommersemester 2015

Radu Muschevici

Software Engineering Group, Department of Computer Science

TECHNISCHE UNIVERSITÄT DARMSTADT

2015-06-09

Feature-Oriented Software Product Lines

Motivation

A lot of companies make products that are…

- somehow all the same
- but all a little different

Product Diversity

Product Diversity (cont.)

Why are there so many products from a single company that are similar, but not identical?

- Different customer requirements
- Innovation
- Competition
- Customers want new products
- Company cannot afford to develop each product from scratch
  $\Rightarrow$ reuse of ideas, designs, components

History

The old days

- Products are individually handcrafted, tailored to individual needs
- Every product is built from scratch, unique

Industrial Revolution (ca. 1760–1840)

- Mass production from standardised parts
- Efficient production, large quantities, better quality
- Products are all the same

Today: Production lines, mass customisation

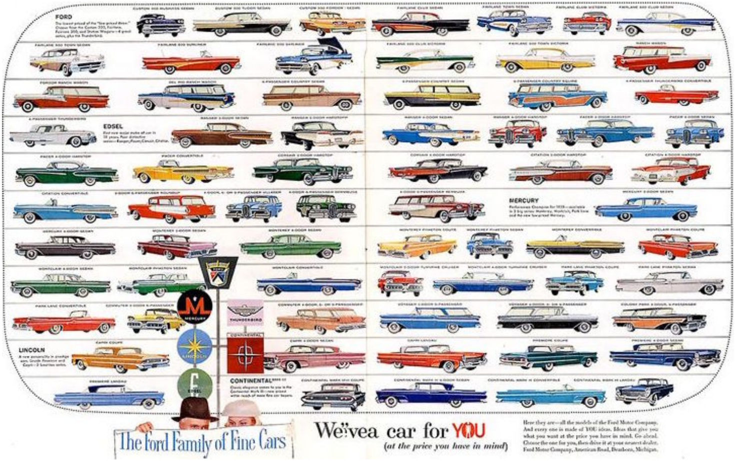
- Mass production from standardised, reusable parts
- Parts can be optional, alternatives
- Different combinations of parts $\Rightarrow$ many different products
- Return to individualism

Product Lines

Customer can…

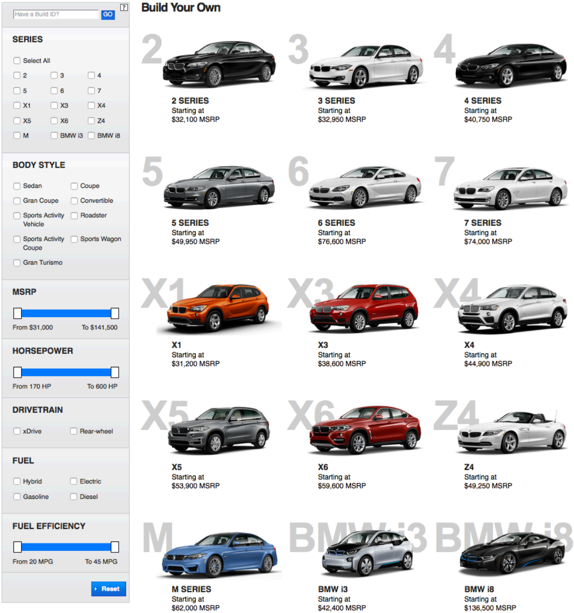
- Choose from a wide range of related products, or
- Choose the parts (features) she wants in a products, and have the corresponding product built.

1958 Car Product Line


FORD
EDSEL
MERCURY
THUNDERBIRD
LINCOLN
CONTINENTAL
The Ford Family of Fine Cars
We've a car for YOU
(at the price you have in mind)
Here they are—all the models of the Ford Motor Company. And every one is made of YOU ideas. Ideas that give you what you want at the price you have in mind. Go ahead. Choose the one for you, then drive it at your nearest dealer. Ford Motor Company, American Road, Dearborn, Michigan.

- Choice between between several models and few extras such as cassette player or roof rack
- One standard variant was responsible for bulk of sales

2015 Car Product Line



Have a Build ID? GO ?

SERIES

□ Select All

□ 2 □ 3 □ 4

□ 5 □ 6 □ 7

□ X1 □ X3 □ X4

□ X5 □ X6 □ Z4

□ M □ BMW i3 □ BMW i8

BODY STYLE

□ Sedan □ Coupe

□ Gran Coupe □ Convertible

□ Sports Activity Vehicle □ Roadster

□ Sports Activity Coupe □ Sports Wagon

□ Gran Turismo

MSRP

From $31,000 To $141,500

HORSEPOWER

From 170 HP To 600 HP

DRIVETRAIN

□ xDrive □ Rear-wheel

FUEL

□ Hybrid □ Electric

□ Gasoline □ Diesel

FUEL EFFICIENCY

From 20 MPG To 45 MPG

Reset

Build Your Own

2 SERIES
Starting at
$32,100 MSRP

3 SERIES
Starting at
$32,950 MSRP

4 SERIES
Starting at
$40,750 MSRP

5 SERIES
Starting at
$49,950 MSRP

6 SERIES
Starting at
$76,600 MSRP

7 SERIES
Starting at
$74,000 MSRP

X1
Starting at
$31,200 MSRP

X3
Starting at
$38,600 MSRP

X4
Starting at
$44,900 MSRP

X5
Starting at
$53,900 MSRP

X6
Starting at
$59,600 MSRP

Z4
Starting at
$49,250 MSRP

M SERIES
Starting at
$62,000 MSRP

BMW i3
Starting at
$42,400 MSRP

BMW i8
Starting at
$136,500 MSRP

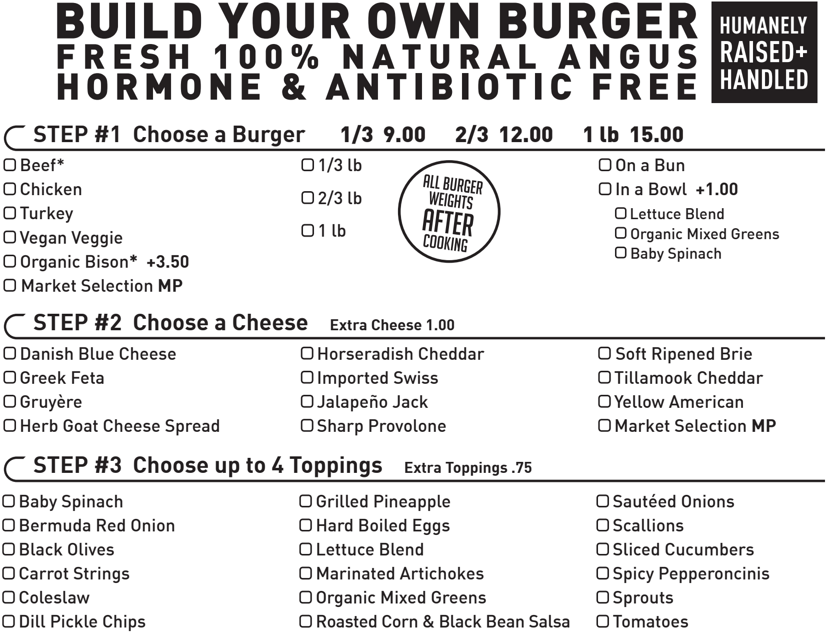
- Huge configuration space: 10^{32} variants of a BMW
- Many different types of components: engine, transmission, seat, mirrors, headlights, brakes, steering wheel, etc.
- Today, car makers hardly ever produce two identically configured cars!

Variability

- To build a product line efficiently, the underlying set of reusable components must be **variable**.
- Optional components: can be part of a product but not another
- Alternatives: some components have the same function but are implemented differently

> Variability is the ability to derive different products from a common set of artifacts.
>
> [Apel et al. 2013]

A Product Line of Burgers



BUILD YOUR OWN BURGER

FRESH 100% NATURAL ANGUS
HORMONE & ANTIBIOTIC FREE

HUMANELY RAISED+ HANDLED

STEP #1 Choose a Burger 1/3 9.00 2/3 12.00 1 lb 15.00

□ Beef*
□ Chicken
□ Turkey
□ Vegan Veggie
□ Organic Bison* +3.50
□ Market Selection MP

□ 1/3 lb
□ 2/3 lb
□ 1 lb

ALL BURGER WEIGHTS AFTER COOKING

□ On a Bun
□ In a Bowl +1.00
□ Lettuce Blend
□ Organic Mixed Greens
□ Baby Spinach

STEP #2 Choose a Cheese Extra Cheese 1.00

□ Danish Blue Cheese
□ Greek Feta
□ Gruyère
□ Herb Goat Cheese Spread

□ Horseradish Cheddar
□ Imported Swiss
□ Jalapeño Jack
□ Sharp Provolone

□ Soft Ripened Brie
□ Tillamook Cheddar
□ Yellow American
□ Market Selection MP

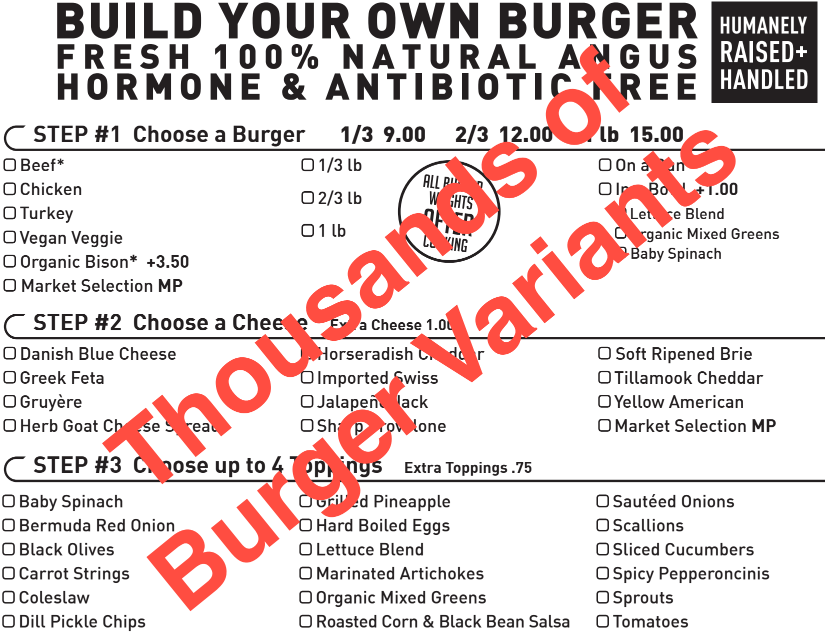
STEP #3 Choose up to 4 Toppings Extra Toppings .75

□ Baby Spinach
□ Bermuda Red Onion
□ Black Olives
□ Carrot Strings
□ Coleslaw
□ Dill Pickle Chips

□ Grilled Pineapple
□ Hard Boiled Eggs
□ Lettuce Blend
□ Marinated Artichokes
□ Organic Mixed Greens
□ Roasted Corn & Black Bean Salsa

□ Sautéed Onions
□ Scallions
□ Sliced Cucumbers
□ Spicy Pepperoncinis
□ Sprouts
□ Tomatoes

A Product Line of Burgers



BUILD YOUR OWN BURGER

FRESH 100% NATURAL ANGUS
HORMONE & ANTIBIOTIC FREE

HUMANELY RAISED+ HANDLED

STEP #1 Choose a Burger 1/3 9.00 2/3 12.00 1 lb 15.00

□ Beef*
□ Chicken
□ Turkey
□ Vegan Veggie
□ Organic Bison* +3.50
□ Market Selection MP

□ 1/3 lb
□ 2/3 lb
□ 1 lb

ALL BURGER WEIGHTS AFTER COOKING

□ On a Bun
□ In a Bowl +1.00
□ Lettuce Blend
□ Organic Mixed Greens
□ Baby Spinach

STEP #2 Choose a Cheese Extra Cheese 1.00

□ Danish Blue Cheese
□ Greek Feta
□ Gruyère
□ Herb Goat Cheese Spread

□ Horseradish Cheddar
□ Imported Swiss
□ Jalapeño Jack
□ Sharp Provolone

□ Soft Ripened Brie
□ Tillamook Cheddar
□ Yellow American
□ Market Selection MP

STEP #3 Choose up to 4 Toppings Extra Toppings .75

□ Baby Spinach
□ Bermuda Red Onion
□ Black Olives
□ Carrot Strings
□ Coleslaw
□ Dill Pickle Chips

□ Grilled Pineapple
□ Hard Boiled Eggs
□ Lettuce Blend
□ Marinated Artichokes
□ Organic Mixed Greens
□ Roasted Corn & Black Bean Salsa

□ Sautéed Onions
□ Scallions
□ Sliced Cucumbers
□ Spicy Pepperoncinis
□ Sprouts
□ Tomatoes

Thousands of Burger Variants

What About Software?

Software production is challenging and expensive!

Standard Software

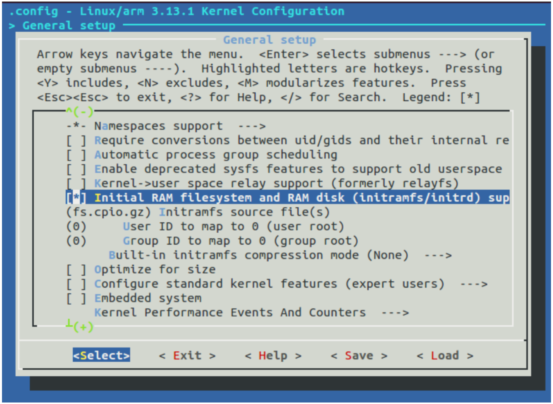
- Off-the-shelf products that run on standardised platform: e.g. Microsoft Office, IBM DB2, SAP R/3 – **one-size-fits-all** solution
- Satisfies the needs of most customers, but contains far more functionality that any individual user needs. Complex, slow, buggy, high resource demands.

Software Product Lines (SPL)

- Build software from reusable parts
- Mass customization: **individual** solutions based on individual customer requirements
- Reduced complexity and resource usage, better performance
- Better fit for systems with limited resources: embedded, mobile

Example SPL: Linux Kernel

- ca. 10,000,000 lines of code
- Reusable parts: > 10,000 features (most are optional)
- 2^{10000} different configurations



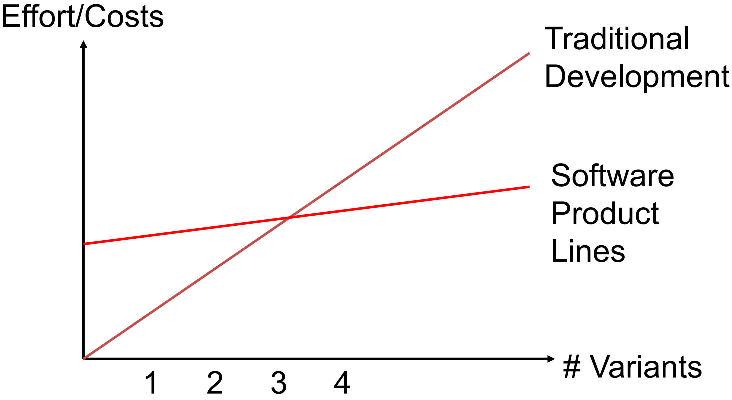
```
.config - Linux/arm 3.13.1 Kernel Configuration
> General setup ──────────────────────────────────────────────────
                         General setup
   Arrow keys navigate the menu.  <Enter> selects submenus ---> (or
   empty submenus ----).  Highlighted letters are hotkeys.  Pressing
   <Y> includes, <N> excludes, <M> modularizes features.  Press
   <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
       ^(-)
       -*- Namespaces support  --->
       [ ] Require conversions between uid/gids and their internal re
       [ ] Automatic process group scheduling
       [ ] Enable deprecated sysfs features to support old userspace
       [ ] Kernel->user space relay support (formerly relayfs)
       [*] Initial RAM filesystem and RAM disk (initramfs/initrd) sup
       (fs.cpio.gz) Initramfs source file(s)
       (0)     User ID to map to 0 (user root)
       (0)     Group ID to map to 0 (group root)
             Built-in initramfs compression mode (None)  --->
       [ ] Optimize for size
       [ ] Configure standard kernel features (expert users)  --->
       [ ] Embedded system
           Kernel Performance Events And Counters  --->
       ┴(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

Software Product Lines in Industry

- HP: Printer firmware
- Nokia: Mobile phone OS, Web browser
- Phillips: High-End TVs, Medical Systems, …
- General Motors: Powertrains
- Boeing: operational flight programs
- Bosch: engine-control software for gasoline systems
- Many more: Turbines, train control, ship control, frequency converter, Internet payment gateway, helicopter avionics software

[Apel et al. 2013]

Software Product Lines: Development Effort



Software Product Lines: Basic Idea

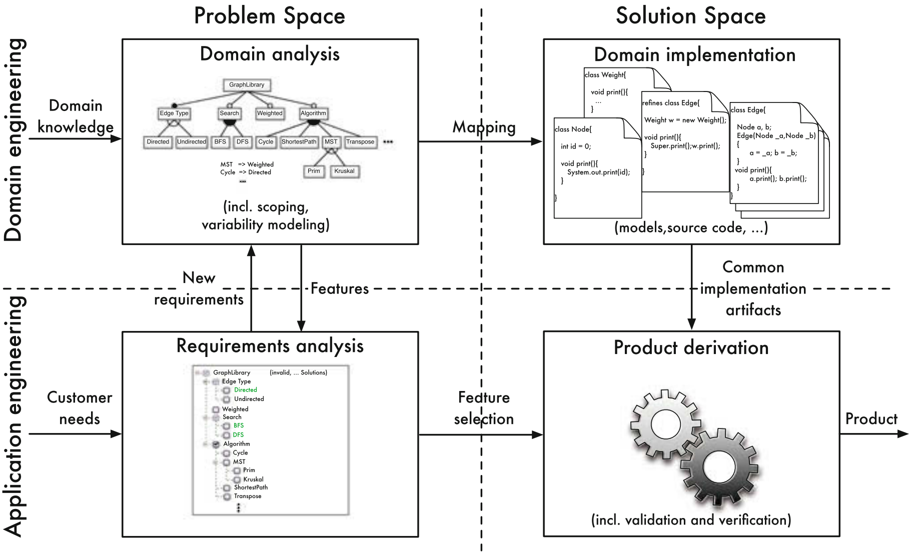
Main goals:

- Explicit handling of variability
- Systematic reuse

Two engineering processes:

1. **Domain engineering:** Develop set of reusable software artifacts.
2. **Application engineering:** Compose artifacts in various ways to obtain individually tailored software products.

Software Product Lines Engineering Process[1]



```
class Weight{

  void print(){
    ...
  }
```

```
class Node{

  int id = 0;

  void print(){
    System.out.print(id);
  }

}
```

```
refines class Edge{

 Weight w = new Weight();

 void print(){
   Super.print();w.print();
 }

}
```

```
class Edge{

  Node a, b;
  Edge(Node _a,Node _b)
  {
      a = _a; b = _b;
  }
 void print(){
      a.print(); b.print();
  }
}
```

[1] [Apel et al. 2013]

Domain and Application

Domain

- The products of an SPL are tailored to an application domain
- Examples: operating systems, database systems, middleware, automotive software, compilers, healthcare applications

Application

- An application is a specific product, built for the needs of a particular stakeholder (user, customer).
- Examples: elements of the domain above

Problem and Solution Space

Two different perspectives:

Problem space

- Customer/user/stakeholder's perspective
- Map requirements to features; define products as feature selections

Solution space

- Developer's perspective
- Develop reusable code artifacts; build software products from those artifacts

Software Product Lines

A software product line (SPL) is a **set** of software-intensive **systems** that **share** a common, managed set of **features** satisfying the specific needs of a particular market segment or mission and that are **developed** from a **common set of core assets** in a prescribed way.

[Clements and Northrop 2001]

Feature-Oriented SPLs

Idea: Use **features** to distinguish the products of a product line.

Examples:

- "My text editor provides a spell-checking feature."
- "Database system A provides multi-user support, Database B does not."
- "Email client A supports IMAP and POP3, client B supports only POP3."
- "The game we are developing will run on Android and iOS."
- "Both financial software products support international transactions."

Features

What is a feature?

- Domain abstraction; usually a **name**
- Features are derived from customer requirements.
- Express variability: used to distinguish the products (variants) of an SPL from each other
- Means to communicate between stakeholders
- Means to specify variants: **feature selection** as input for variant generation

Automated Product Derivation

Traceability of user requirements across software life cycle

Features are a central concept in all phases of product-line development: all development artifacts – requirements, design elements, code, tests, etc. – can be traced to features.

Automated product derivation

1. User selects a set of desired features & pushes a button
2. SPL build system…
   - Automatically selects corresponding reusable code artifacts,
   - Generates product by composing code artifacts,
   - Runs test suite,
   - Compiles product to executable format.

Correctness?

- Variability $\Rightarrow$ Complexity
- Example: 33 optional, independent features: $2^{33} = 9$ billion variants (one for each person on Earth)
- How to verify that all these products behave correctly?

Note:

- Not all features are freely composable with other features.
- Usually, features have constraints.

Feature Modelling

Features **relate to** other features. Examples:

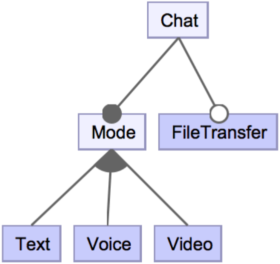
- Secure email download requires SSL protocol
- Spell checking requires at least one natural language dictionary
- The game variant for Android will not run on iOS

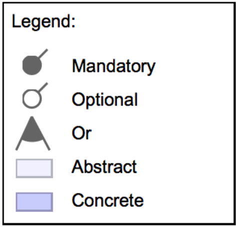
Feature Model

- Models the **variability** of an SPL.
- Describes relationships between features ⇒ which feature combinations are valid.
- A valid feature combination (selection) is a **product** or **variant**.
- A popular, graphical notation for feature models are **feature diagrams**.

Feature Diagram Example

- Feature model of an SPL of chat programs
- Features: text, voice, video communication; file transfer





- What are some valid feature selections of this feature model?

Feature Diagrams

- Tree structure
- Nodes represent features (optional feature denoted by circle)
- Edges represent standard constraints, such as choice (one-out-of-many, some-out-of-many, all-out-of-many)
- More general constraints (“cross-tree” constraints) need to be specified with additional propositional logic formulas,
  E.g. $\text{Video} \Rightarrow (\text{Voice} \wedge \text{Text})$

Feature Diagram Semantics: Propositional Formula

- Features diagrams map to propositional formulas.
- Features represented as Boolean variables (true if selected)
- Constraints represented using logical connectors

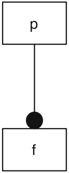
- **Example:** propositional formula for the chat feature diagram:

  $\text{Chat} \wedge (\text{Mode} \Leftrightarrow \text{Chat}) \wedge (\text{FileTransfer} \Rightarrow \text{Chat}) \wedge ((\text{Text} \vee \text{Voice} \vee \text{Video}) \Leftrightarrow \text{Mode})$

- Enables automated reasoning (SAT solvers, Binary Decision Diagrams):
  - Check validity of given feature selection (does it satisfy the propositional formula?)
  - Find all valid feature selections ($=$ all solutions of the formula)
  - Compare feature models, etc.

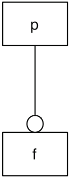
Feature Diagram Semantics (cont.)

Mandatory



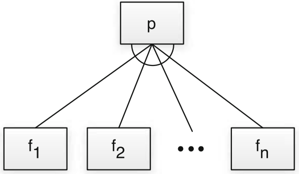
$$f \Leftrightarrow p$$

Optional



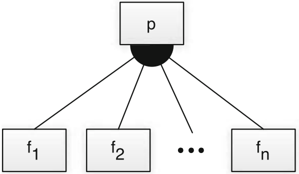
$$f \Rightarrow p$$

Choice: one-out-of-many



$$((f_1 \vee \ldots \vee f_n) \Leftrightarrow p) \wedge \bigwedge_{i<j} \neg(f_i \wedge f_j)$$

Choice: some-out-of-many



$$(f_1 \vee \ldots \vee f_n) \Leftrightarrow p$$

Variations, Extension of Feature Models

- Different notation exist to express feature models – feature diagrams is just one among many
- In addition to Boolean feature variables, some variants allow non-Boolean **feature attributes**: add more detail to feature model

Outlook

- How to implement features and generate products of the SPL?
- Languages/language constructs that help to develop SPLs

This Week's Reading Assignment

- **Chapter 2** of:
  Apel, S., Batory, D., Kästner, C., and Saake, G. **Feature-Oriented Software Product Lines: Concepts and Implementation**. Springer, 2013.

- Download: see Course Announcements forum at
  `https://moodle.informatik.tu-darmstadt.de/course/view.php?id=438`

References I

Apel, Sven, Don Batory, Christian Kästner, and Gunter Saake (2013). **Feature-Oriented Software Product Lines: Concepts and Implementation**. Springer.

Clements, Paul and Linda Northrop (2001). **Software product lines: practices and patterns**. Vol. 0201703327. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.