Software Reuse

Software Engineering 2
(3103313-1)

Amirkabir University of Technology
Fall 1399-1400

Software Reuse

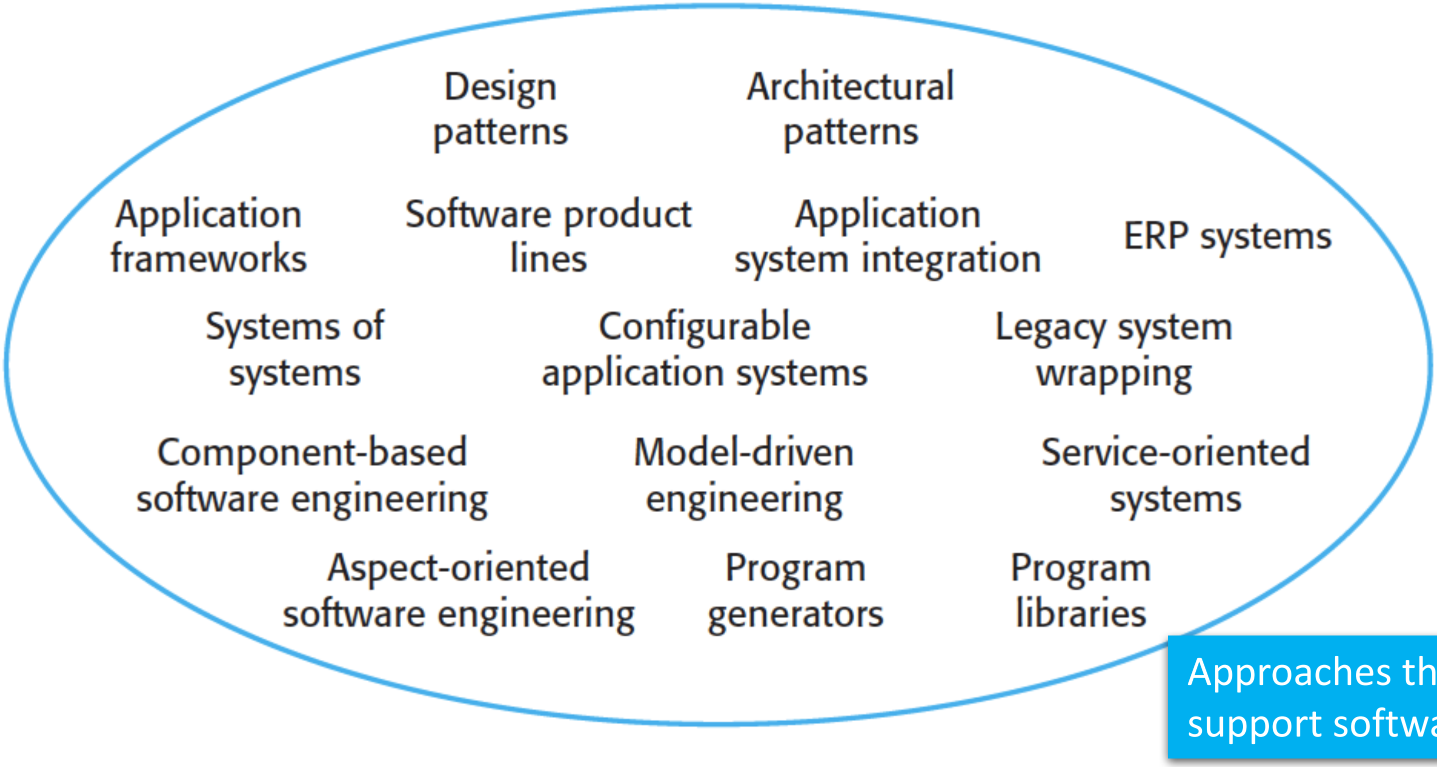
What, Why, How

The Reuse Landscape

Reuse is possible at a range of levels from simple functions to complete application systems

- Reusable Artefact
- Software Asset

- Reusable data
- Reusable architecture/designs
- Reusable programs/systems
  - COTS (Commercial Off-the Shelf)
- Reusable modules/components
- Reusable application framework
- Reusable X



Design patterns

Architectural patterns

Application frameworks

Software product lines

Application system integration

ERP systems

Systems of systems

Configurable application systems

Legacy system wrapping

Component-based software engineering

Model-driven engineering

Service-oriented systems

Aspect-oriented software engineering

Program generators

Program libraries

Approaches that support software reuse

Software Reuse

Benefits

✓ Accelerated development

✓ Effective use of specialists

✓ Increased dependability

✓ Lower development costs

✓ Reduced process risk

✓ …

Disadvantages

✘ Finding, understanding, and adapting reusable components

✘ Increased maintenance costs

✘ Lack of tool support

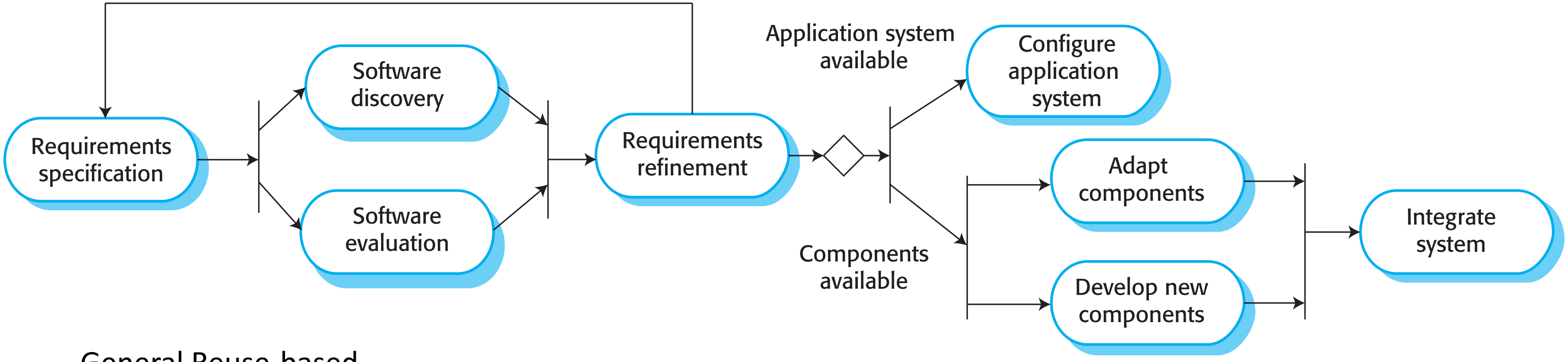
✘ “Not-invented-here” syndrome

✘…

Reuse in Practice

- Systematic vs. Ad hoc

- A disciplined process of software development
  1. (for reuse) Design and development of reusable components
  2. (with reuse) Utilization of reusable components



General Reuse-based
Software Development

Legal and Contractual Issues

- Liability in case of failure of a reused component
- Ownership of reused components
- Maintenance costing
- Security of potentially reusable components

Software Reuse
&
OO Design and Programming

Reuse Approaches & Techniques

Application Frameworks

Application System Reuse

Component-Based System Engineering

Software Product Lines

…

Approaches

Application Frameworks

- A framework is a generic structure that is extended to create a more specific subsystem or application.

Application System Reuse

- An application system is adapted to the needs of different customers without changing the source code of the system.

1. Configurable systems
   - Configurable modules, Configuration process
2. Integrated systems
   - API, service interfaces, …
   - Adapter, Wrapper, …

Underlying Principles

- Independent components
- well-defined interfaces
- Reusable infrastructure
- CBSE-tailored Dev. Process
- …

Approaches

Component-Based System Engineering

- Domain Engineering
  - Identify, construct, catalog, and disseminate a set of software components

    => library of reusable components

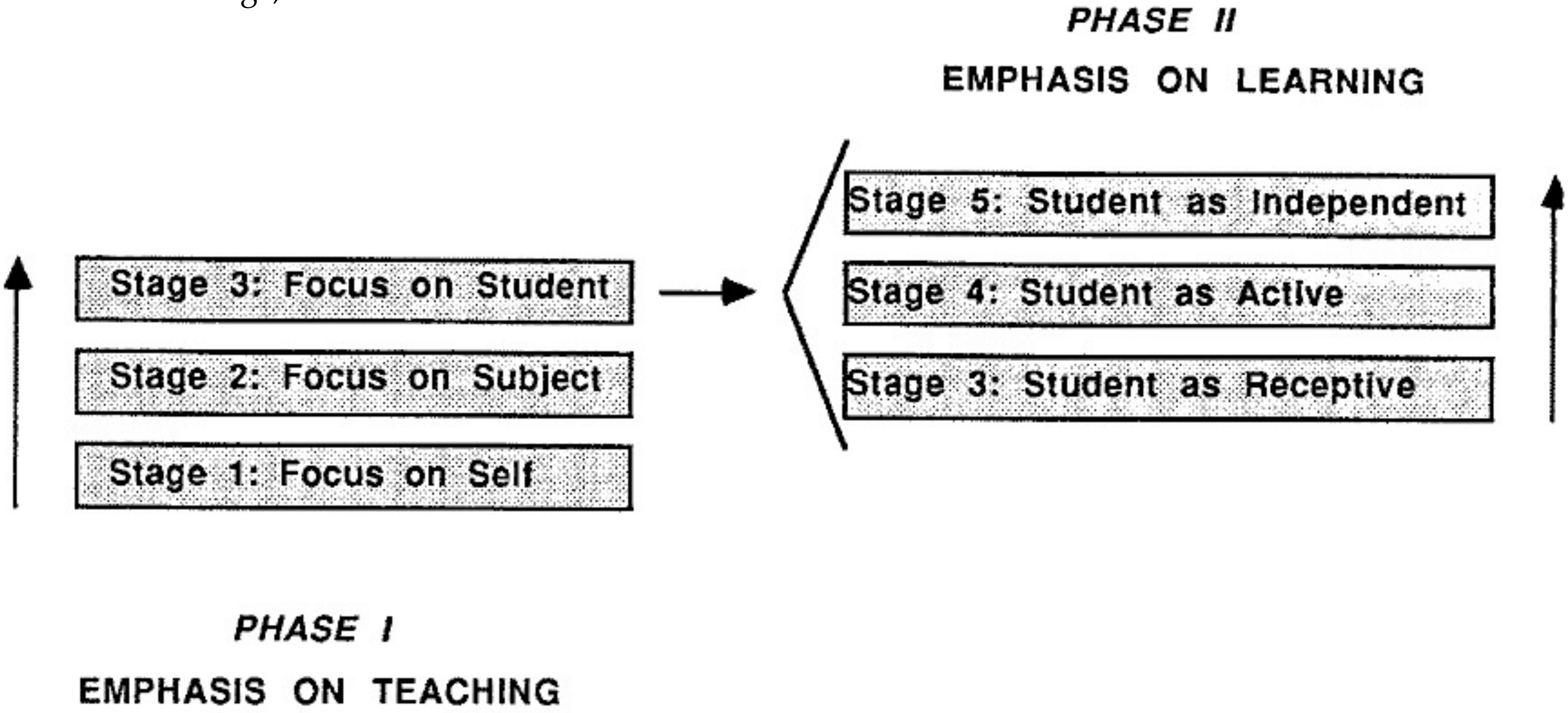
- Component Qualification
  - Does a component “fit” …?

- Component Adaptation & Composition
  - Wrapper
  - Adapter, APIs, ….

break

How Professors Develop as Teachers

PETER KUGEL
Boston College, USA



PHASE II

EMPHASIS ON LEARNING

Stage 5: Student as Independent

Stage 4: Student as Active

Stage 3: Student as Receptive

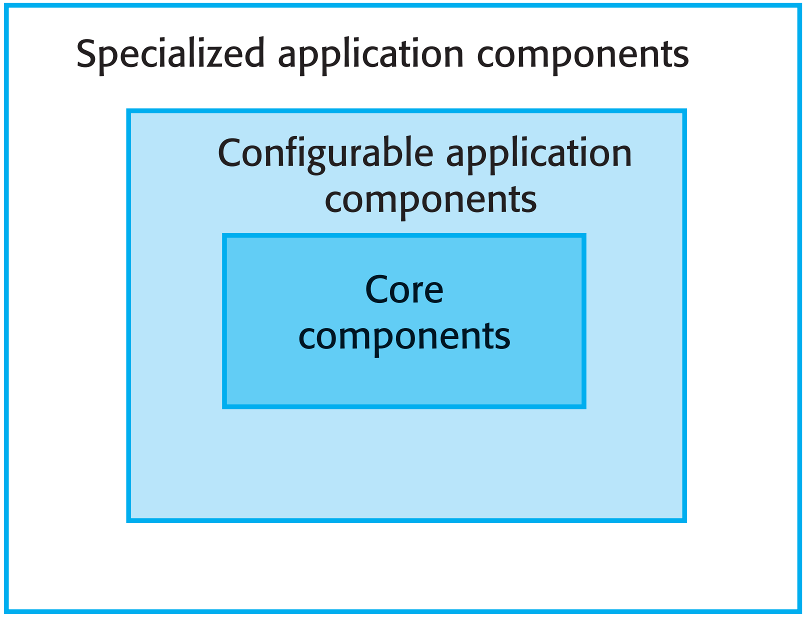
Stage 3: Focus on Student

Stage 2: Focus on Subject

Stage 1: Focus on Self

PHASE I

EMPHASIS ON TEACHING

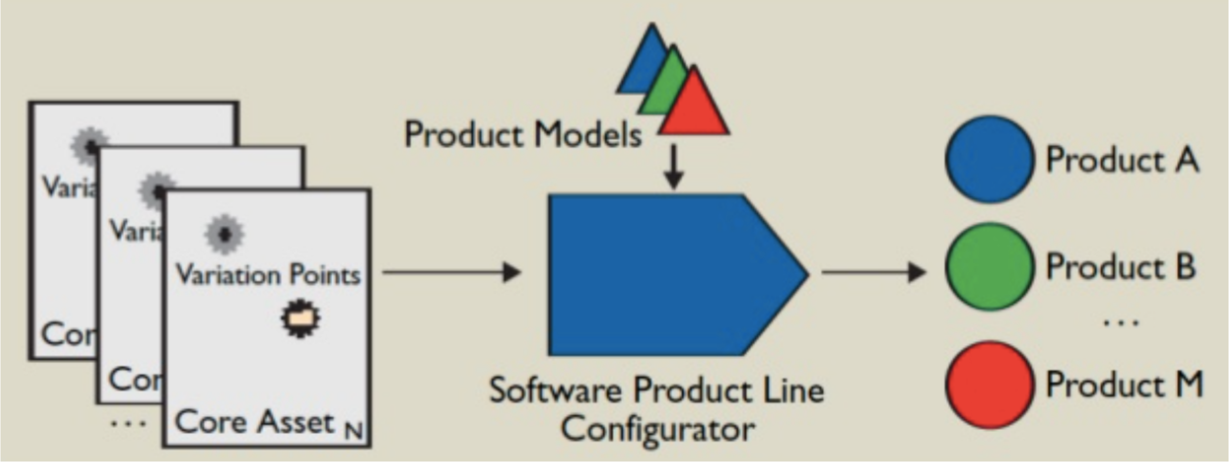

Specialized application components
Configurable application components
Core components

Software Product Lines

A software product line is a **set** of applications with a **common architecture** and shared components, with each application **specialized** to reflect specific customer requirements.

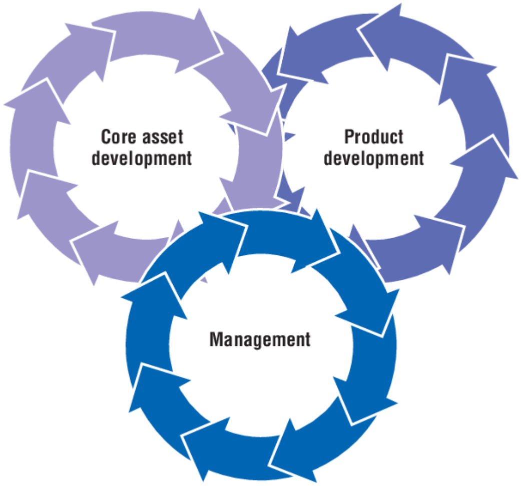
Families of Software Products

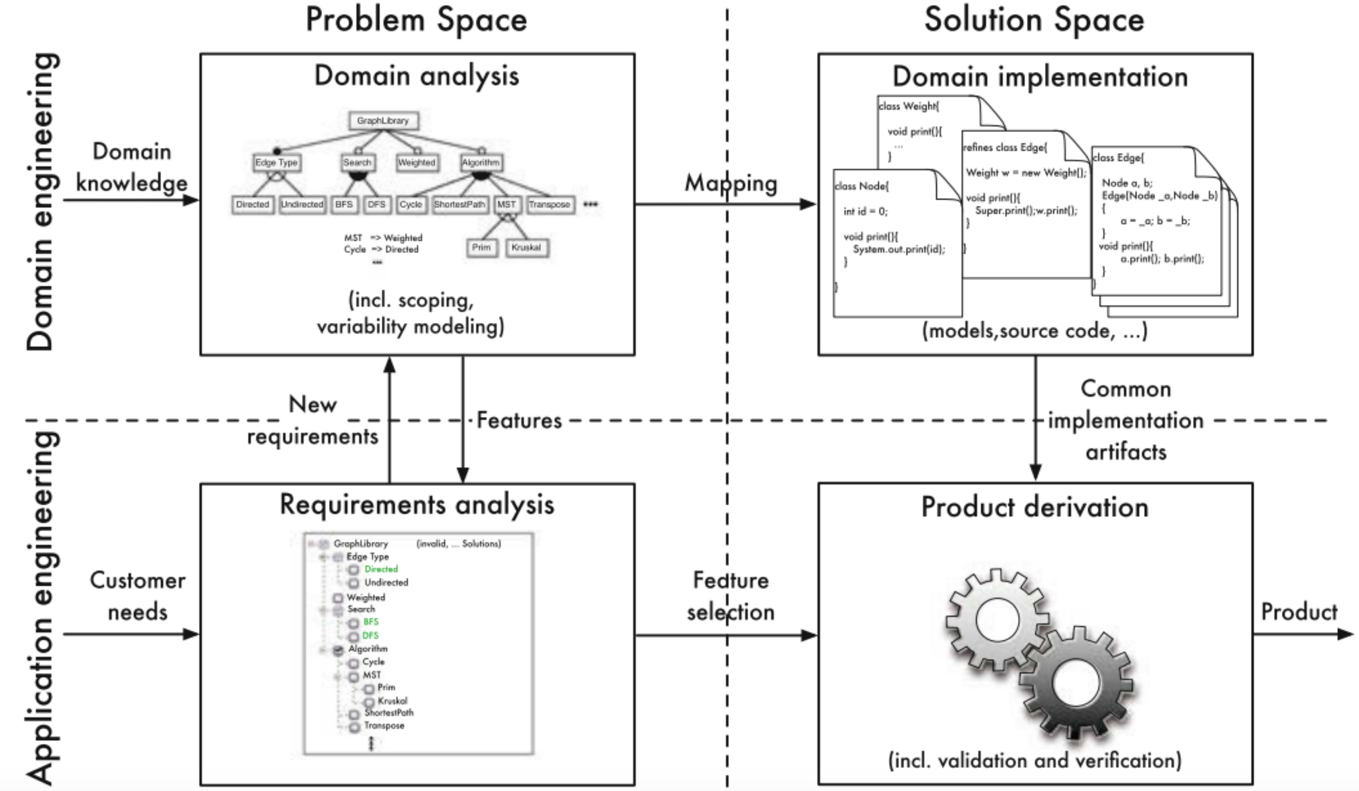
Software Product Line



- Core Asset for the basis of an SPL.
  - The **architecture**, reusable software components, domain models, requirements statements, documentation and specifications, performance models, schedules, budgets, test plans, test cases, work plans, and process descriptions
- Production Plan, SPL Configurator, …
  - How products are produced from the core assets.

Product Line Engineering





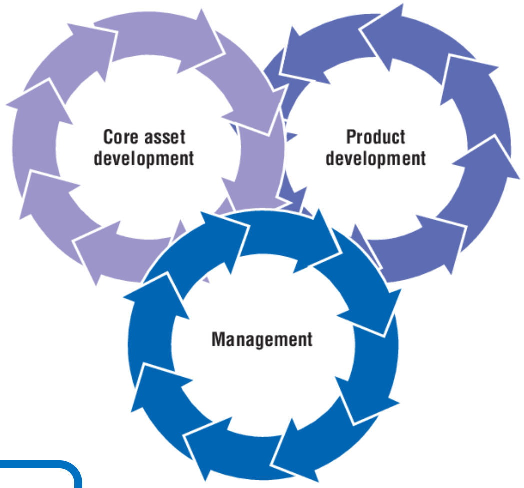
```
class Weight{

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

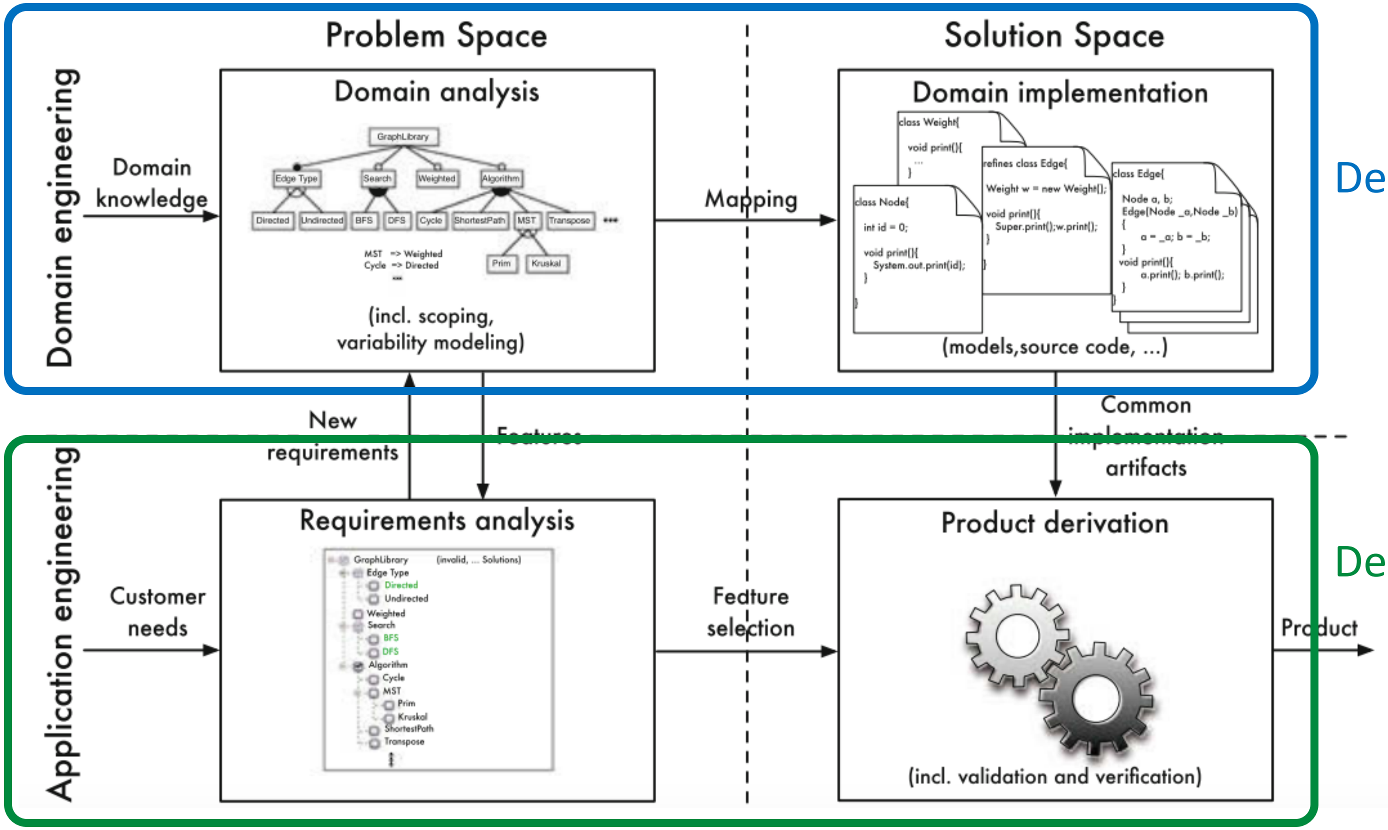
```
class Node{

  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();
  }
}
```

Product Line Engineering





```
class Weight{

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

```
class Node{

  int id = 0;

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

}
```

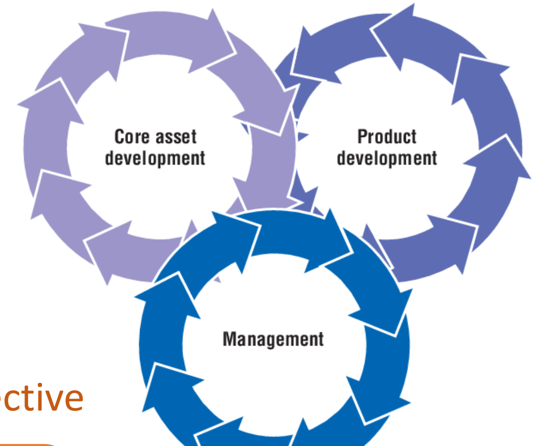
```
refines class Edge{

 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();
  }
}
```

Dev. for reuse

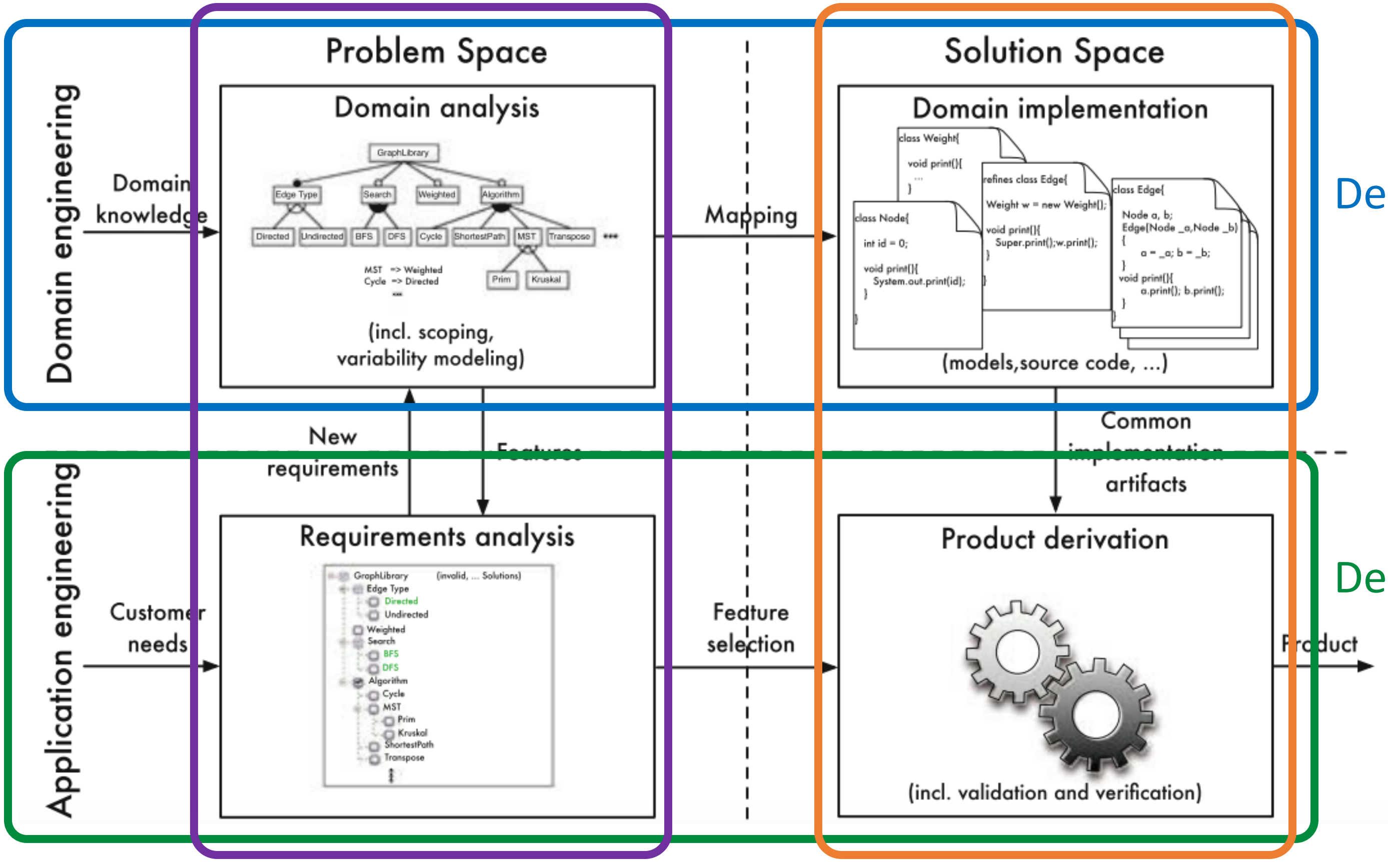
Dev. with reuse

Product Line Engineering



Problem Domain perspective

Implementation perspective



```
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();
  }
}
```

```
GraphLibrary        (invalid, ... Solutions)
  Edge Type
    Directed
    Undirected
  Weighted
  Search
    BFS
    DFS
  Algorithm
    Cycle
    MST
      Prim
      Kruskal
    ShortestPath
    Transpose
```

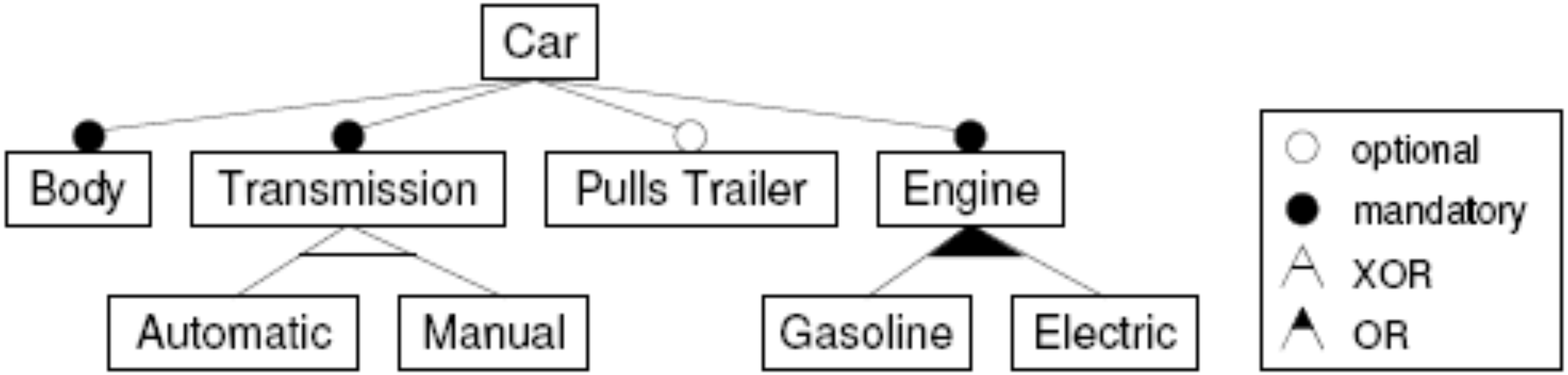
Dev. for reuse

Dev. with reuse

Domain Analysis

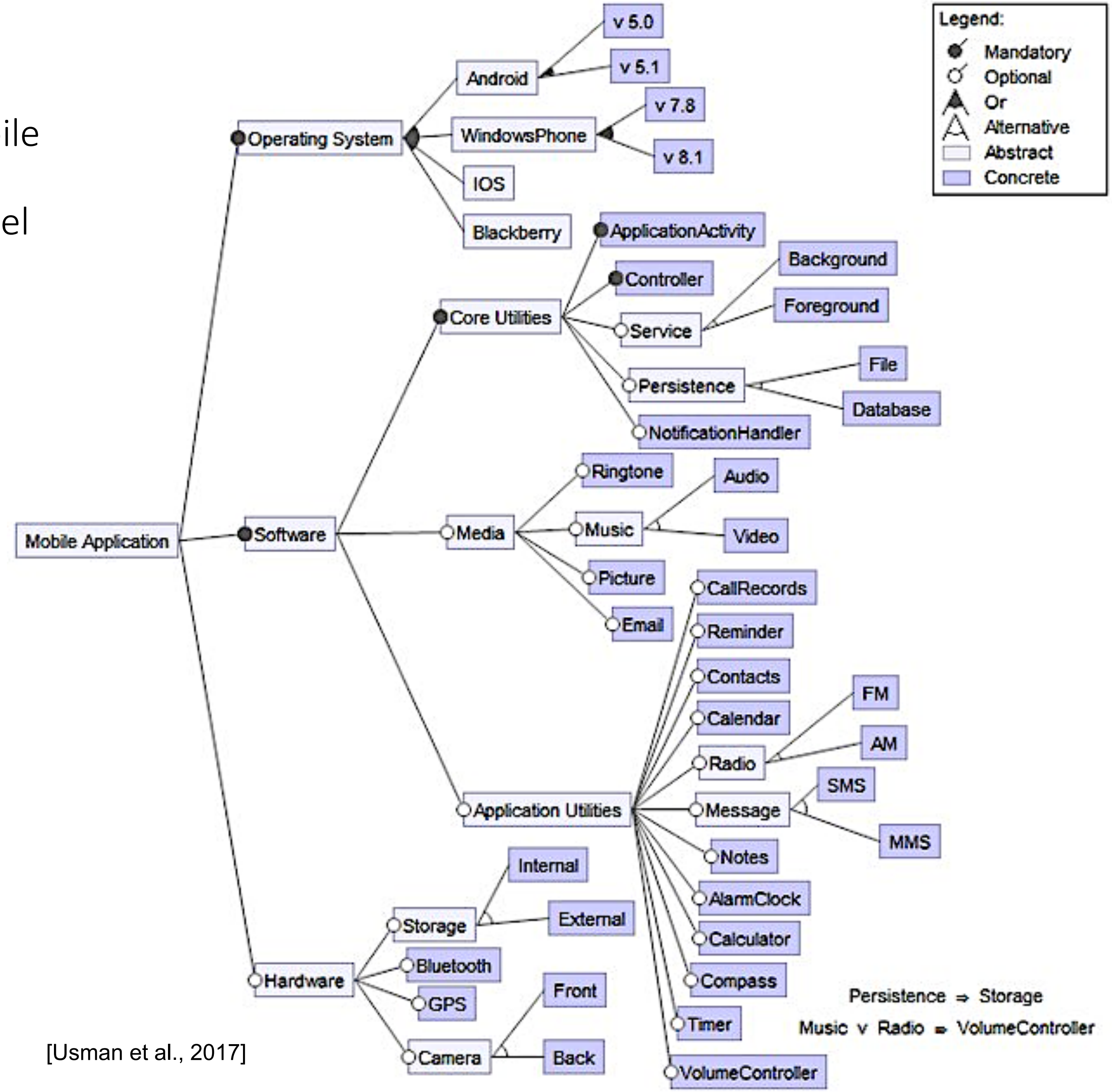
- Domain Scoping
  - Deciding on product line's extent or range

- Domain Modelling
  - Captures & documents the commonalities & variabilities
  - E.g., Feature Model
  - Feature-Oriented SPL



Feature Model - Document the features of a product line & their relationships

Generic Mobile Application Feature Model

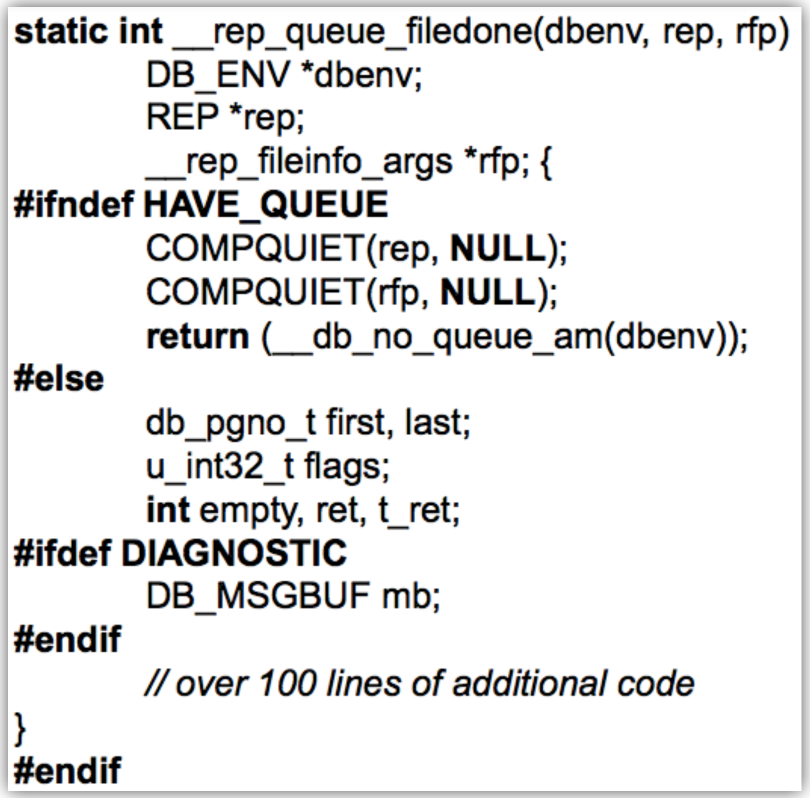


[Usman et al., 2017]

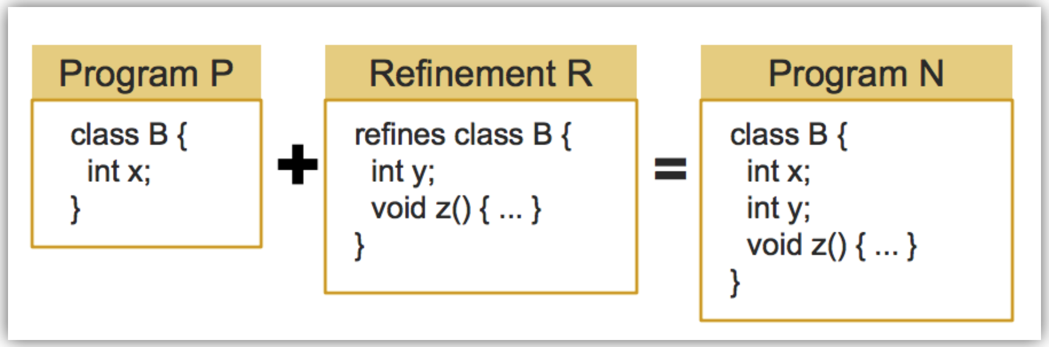
Domain Implementation

- Binding times
  - Compile-time, load-time, and run-time
- Representation
  - Annotation vs composition

- Variability Implementation
  - Parameters, Design patterns, Build systems, Preprocessors, FOP, AOP, DOP, …



```
static int __rep_queue_filedone(dbenv, rep, rfp)
        DB_ENV *dbenv;
        REP *rep;
        __rep_fileinfo_args *rfp; {
#ifndef HAVE_QUEUE
        COMPQUIET(rep, NULL);
        COMPQUIET(rfp, NULL);
        return (__db_no_queue_am(dbenv));
#else
        db_pgno_t first, last;
        u_int32_t flags;
        int empty, ret, t_ret;
#ifdef DIAGNOSTIC
        DB_MSGBUF mb;
#endif
        // over 100 lines of additional code
}
#endif
```



Program P

```
class B {
  int x;
}
```

➕

Refinement R

```
refines class B {
  int y;
  void z() { ... }
}
```

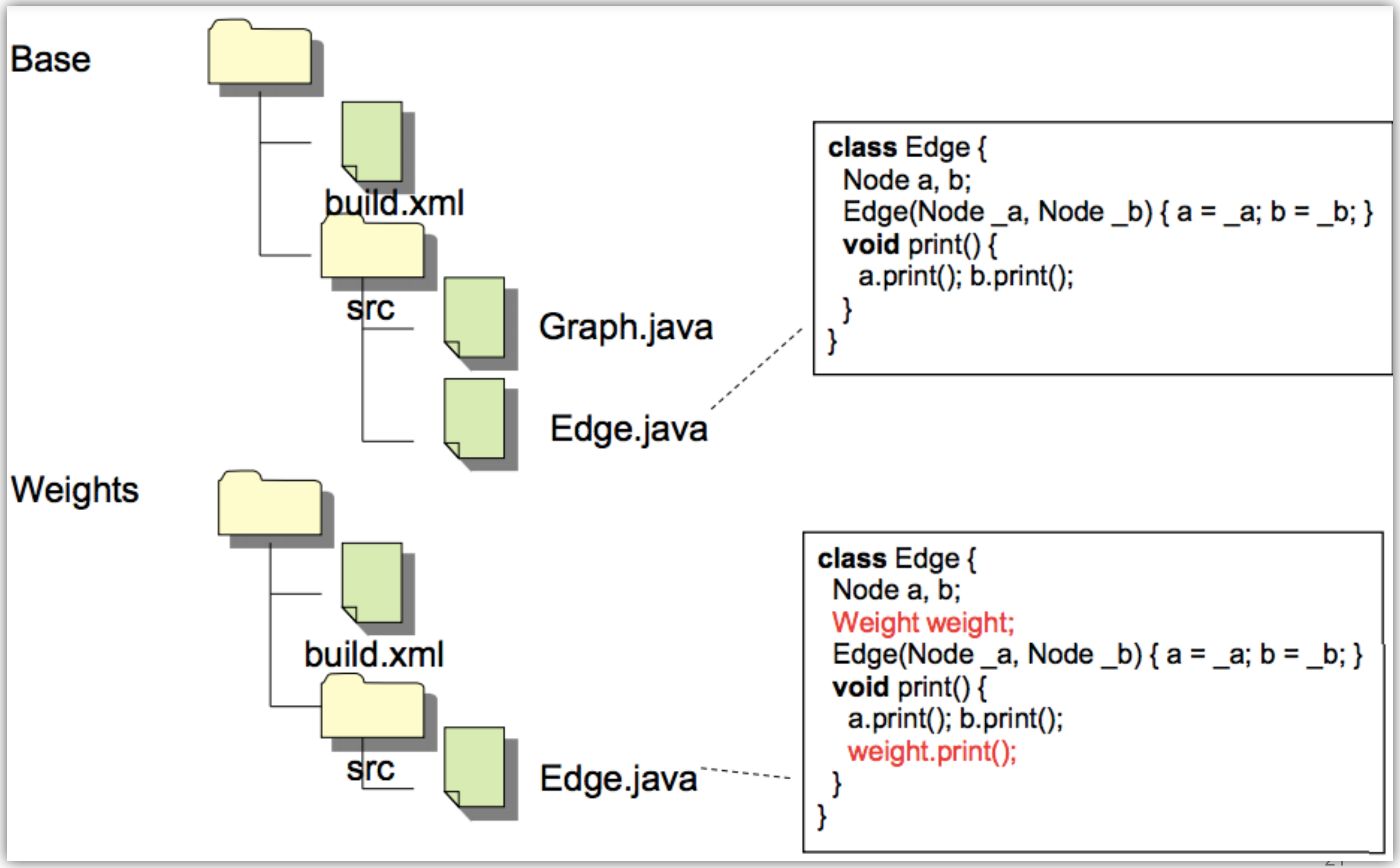
=

Program N

```
class B {
  int x;
  int y;
  void z() { ... }
}
```

DeltaJava

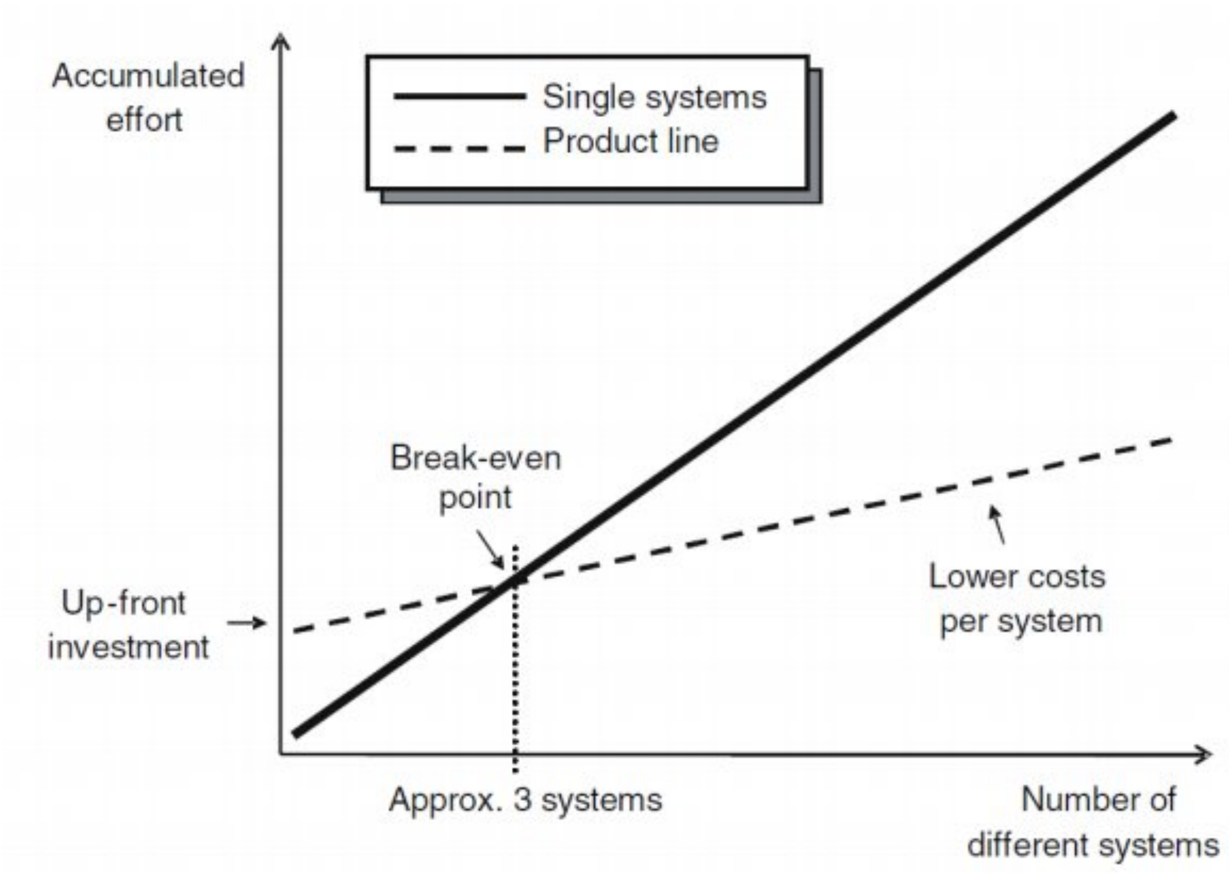
Build Systems



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

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

Economics of SPL



break

lowcomote

Low-code development platforms (LCDP) are software development platforms on the Cloud, provided through a Platform-as a-Service model, which allow users to build completely operational applications by interacting through dynamic graphical user interfaces, visual diagrams and declarative languages. They address the need of non-programmers to develop personalised software, and focus on their domain expertise instead of implementation requirements.

https://www.lowcomote.eu/

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UNIVERSITY of York

UAM Universidad Autónoma de Madrid

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

JKU JOHANNES KEPLER UNIVERSITÄT LINZ

BT

intecs Solutions the Brainware company

UGROUND

CLMS

IncQueryLabs

SPARX SYSTEMS

METADEV

THE Open GROUP

aws

References

- M. Usman, M. Iqbal and M. Khan (2017). A Product-line Model-driven Engineering Approach for Generating Feature-based Mobile Applications. Journal of Systems and Software.

- L. Northrop and P Clements (2012). A Framework for Software Product Line Practice, Version 5.0, Software Engineering Institute.

- T. Thüm (2020). A BDD for Linux? the knowledge compilation challenge for variability. In Proceedings of the 24th ACM Conference on Systems and Software Product Line.