Unanticipated Evolution in Software Product Lines versus Independent Products: A Case Study

By: Mostafa Hamza, Robert J. Walker, Maged Elaasar

Presented By:

Mostafa Hamza

Sept. 26th 2017

Introduction

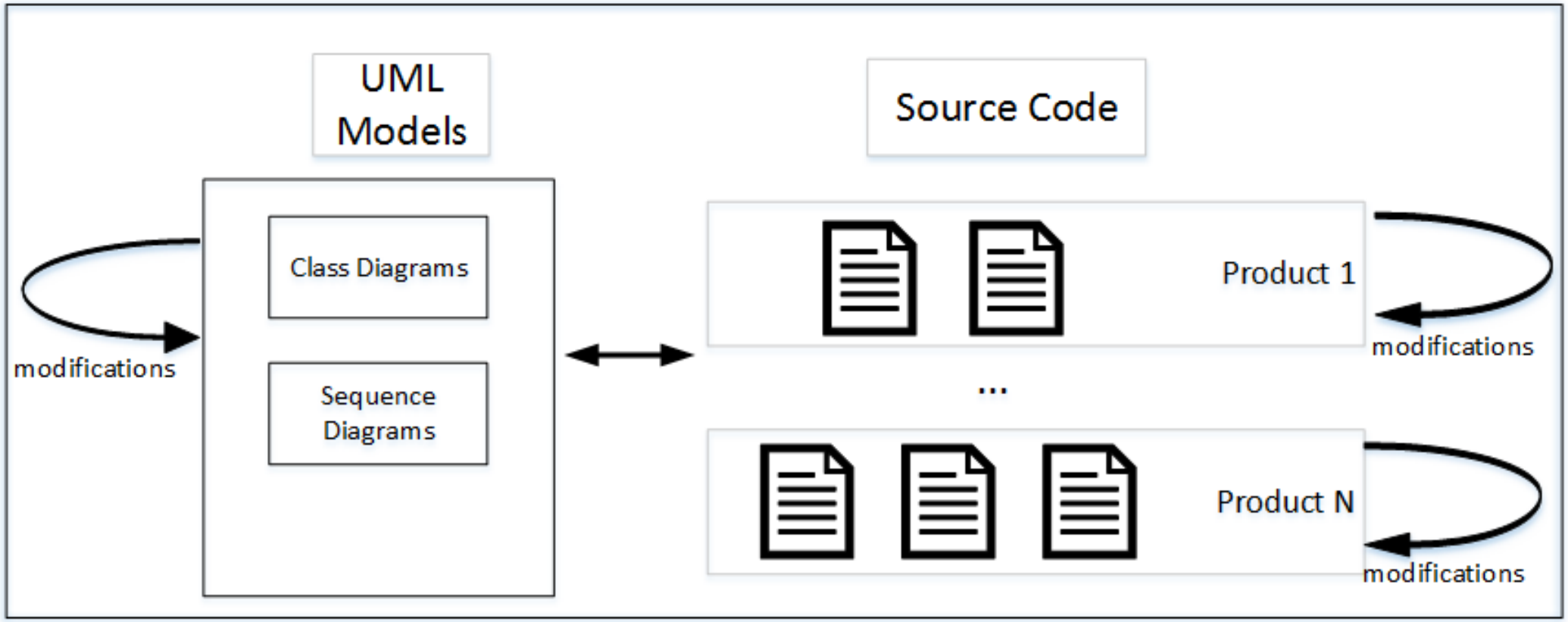
- SPLE
- Software evolution
- Anticipated vs. un-anticipated evolution
- Case Study for un-anticipated SPL evolution

Motivation

UNIVERSITY OF CALGARY

- First hand challenges from managing multiple products as separate ones such as:
  - Code duplication
  - Multiple places to fix a bug across multiple products
  - Refactoring
  - Enhancements
  - Models and code deviations
  - New features

Separate Products



Approach

- A case study to follow the same process we followed for developing the product line but using Delta-oriented programming (DOP)

Vending Machine Simulator (VMS)

VMS Evolution

UNIVERSITY OF CALGARY

- **Product 1 (*Initial version*):**
  - Front end:
    - **Script-Processor** takes input from a simulated user via a simple scripting language, passing it to the **Parser** which parses input **coins** and delivering cans of soda **pop**
  - Product:
    - **VendingMachineFactory** read in, execute, and test a set of scripts for correctness
    - **VendingMachine** controls loading/unloading of coins/pops

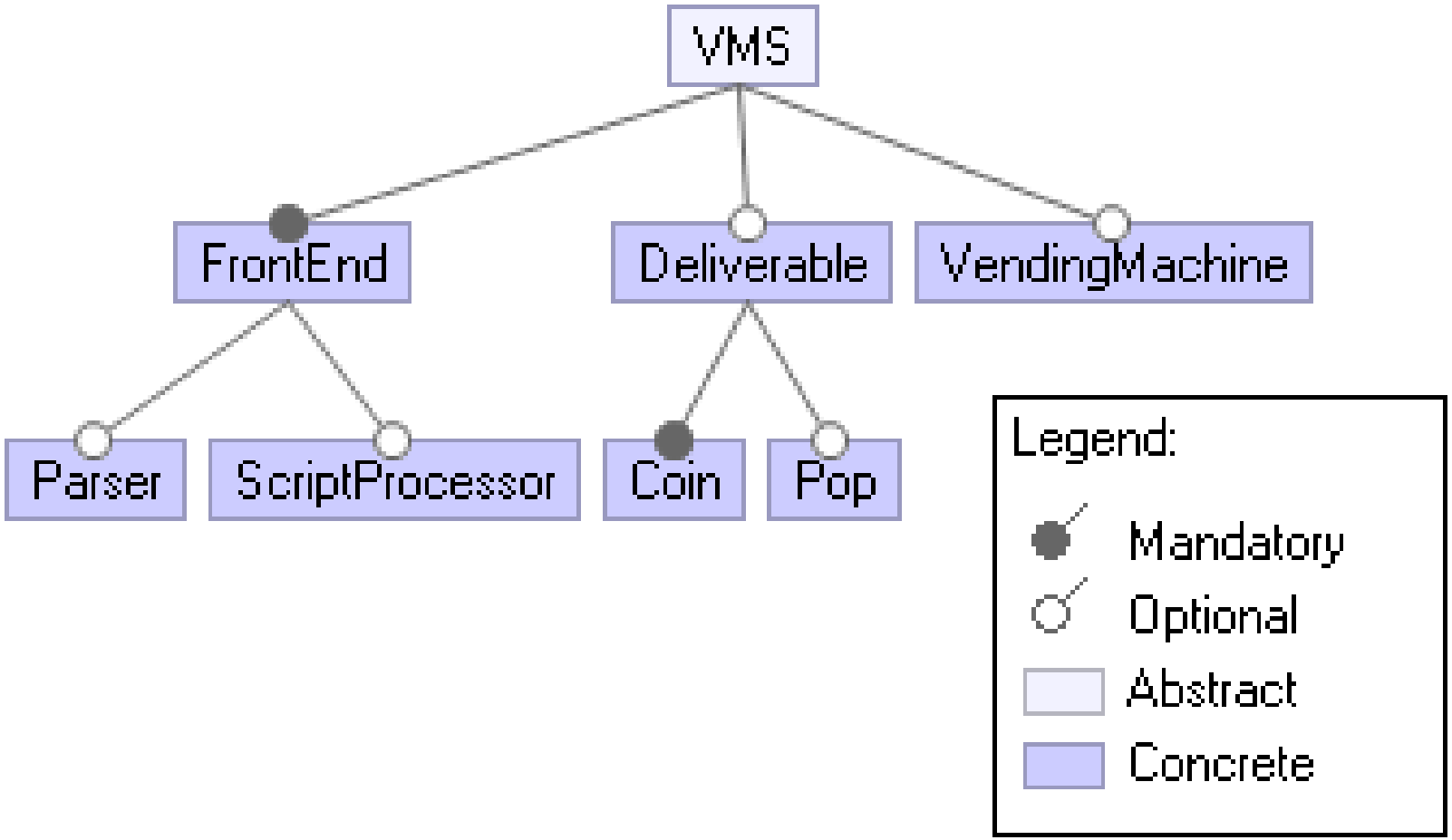
VMS Evolution

- **Product 2 (*New features added*):**
  - A **hardware simulator** to simulate the internal functionalities of a vending machine, such as coin slot, racks, channels, receptacles, delivery chute, display, etc. that replaces the parser;
  - **Pop** was renamed to **PopCan** for improved clarity

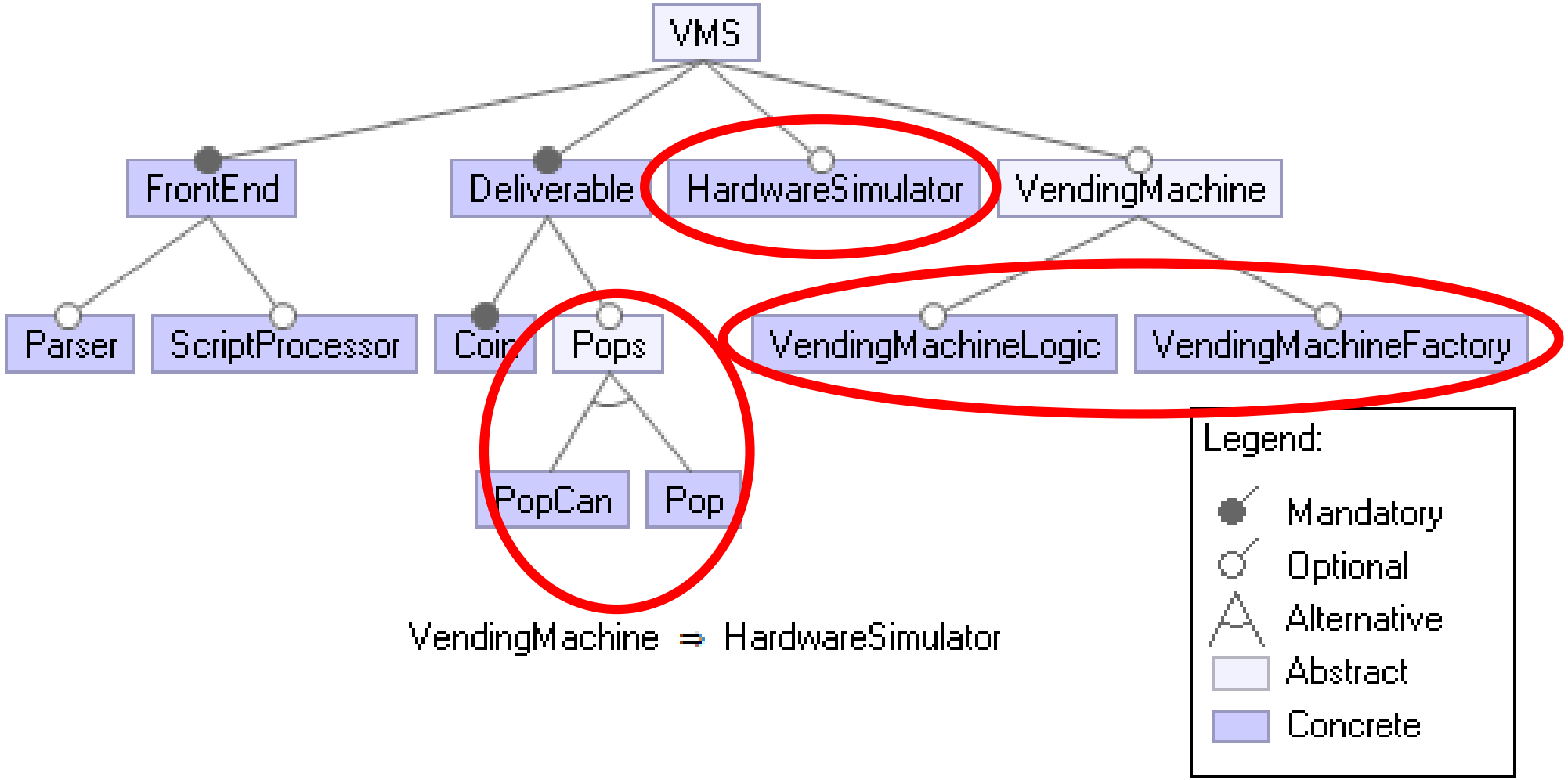
VMS Evolution

- **Product 3 (*More features added*):**
  - The value of coins was to change to instances of the **Cents** class from primitive ints.

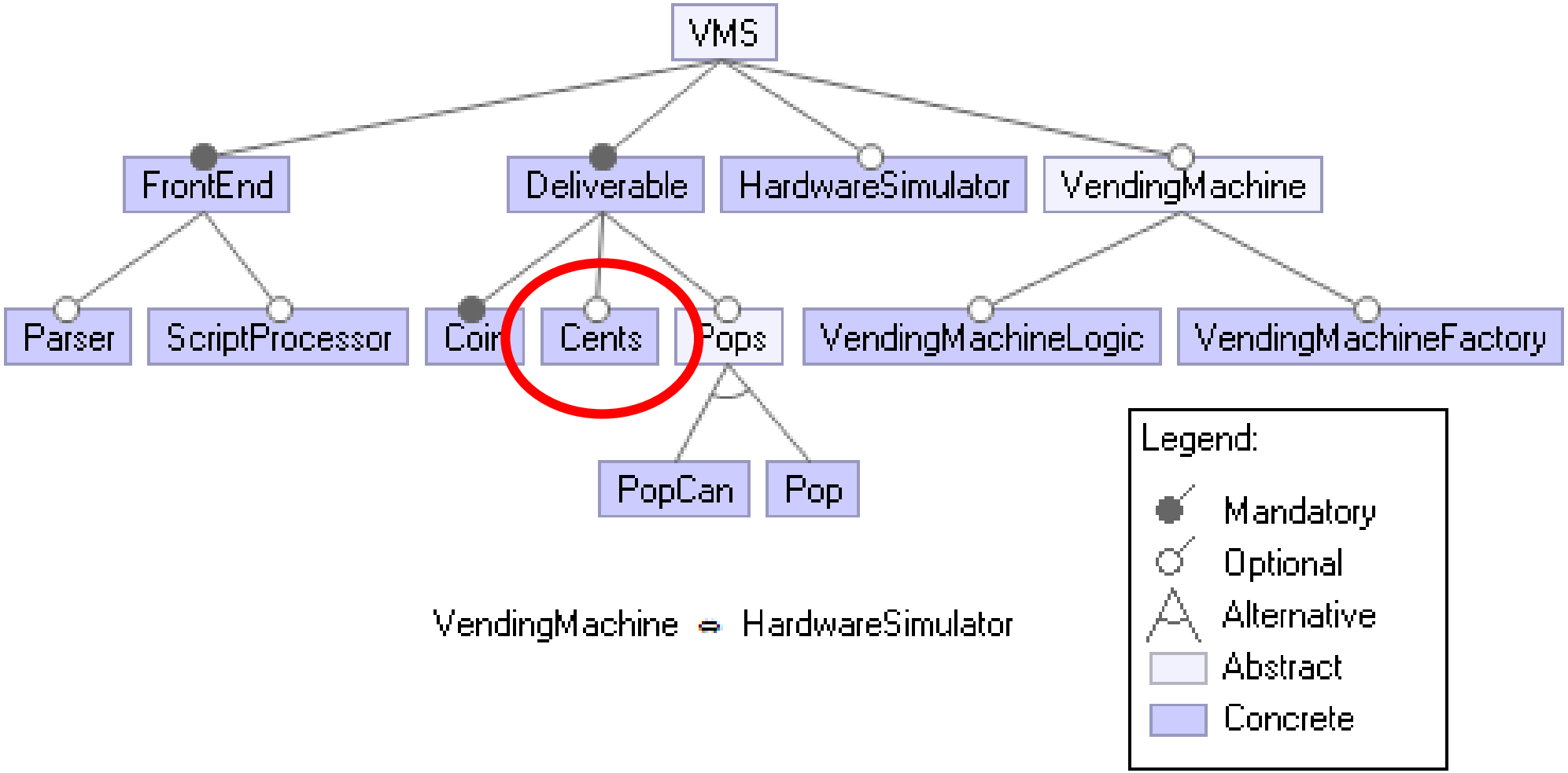
First Product



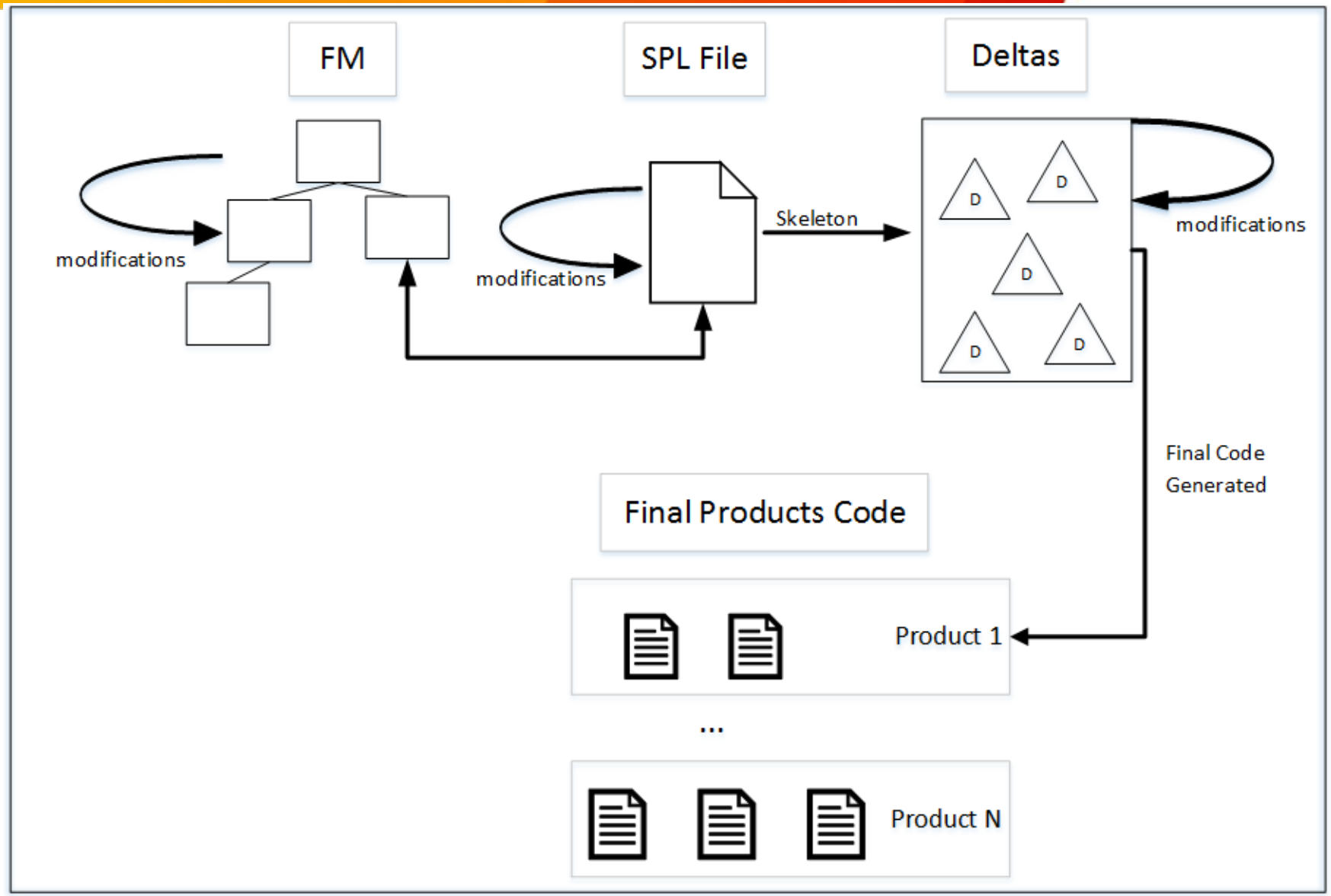
Second Product



Third Product



DOP


FM
SPL File
Deltas
modifications
modifications
Skeleton
D
D
D
D
D
modifications
Final Code
Generated
Final Products Code
Product 1
...
Product N

SPL File Sample

UNIVERSITY OF CALGARY

```
SPL VMS {
    Features = {Frontend, Coin, Pop, Parser, VendingMachine}
    Deltas = {dVendingMachine, dIVendingMachineFactory, dCoin, dPop}
    Constraints { }
    Partitions {
        {dDeliverable, dCoin} when (Coin);
        {dDeliverable, dPop} when (Pop);
    }
Products {
        Product1 = {Frontend, Coin, Pop, Parser, VendingMachine};
    }
}
```

Delta Sample

```
delta dCoin {
  adds {
        package org.lsmr.vending.frontend;
        public class Coin implements Deliverable
         {
            private int value;
            public Coin(int value) {
              if(value <= 0)
                throw new
IllegalArgumentException("The value must
be greater than 0: the argument passed was "
+ value);
             this.value = value;
            }
            public int getValue() {
                    return value;
            }
            public String toString() {
                    return "" + getValue();
            }
   }
  }
}
```

Goals

- Repeat the evolution history for separate products on SPL
- Measure:
  - Lines of code affected
  - Degree of duplication

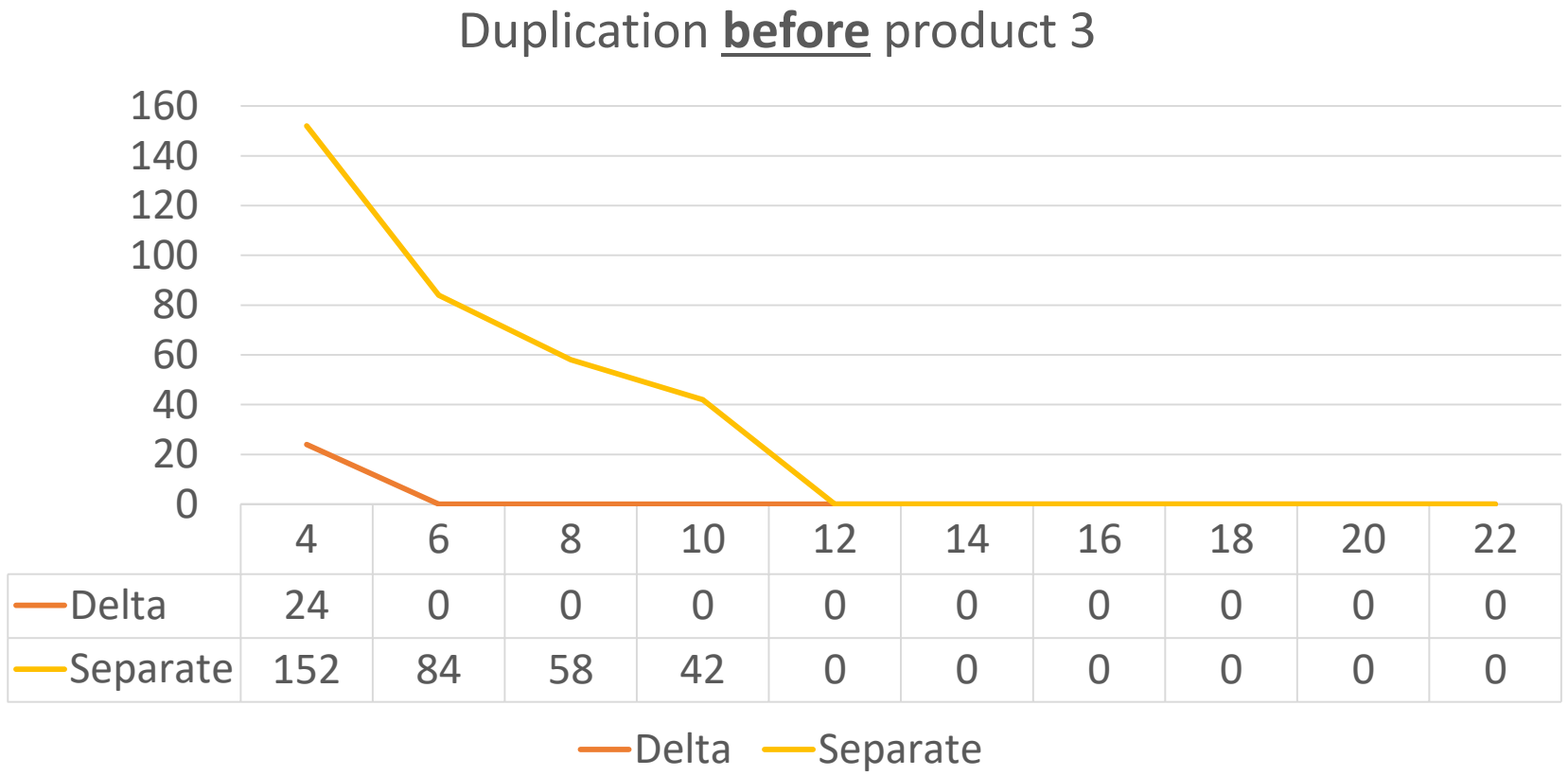
Results – Quantitative

- Impact of change through # of lines changed

	Change Pop to Popcan in Product 2		Evolving to Product 3	
	Separate	**DOP**	**Separate**	**DOP**
Total	42.00	79.00	2614.00	208.00
Mean	3.82	7.18	72.61	29.71
Median	2.00	3.00	39.00	19.00
Standard Deviation	3.30	9.96	90.85	26.95

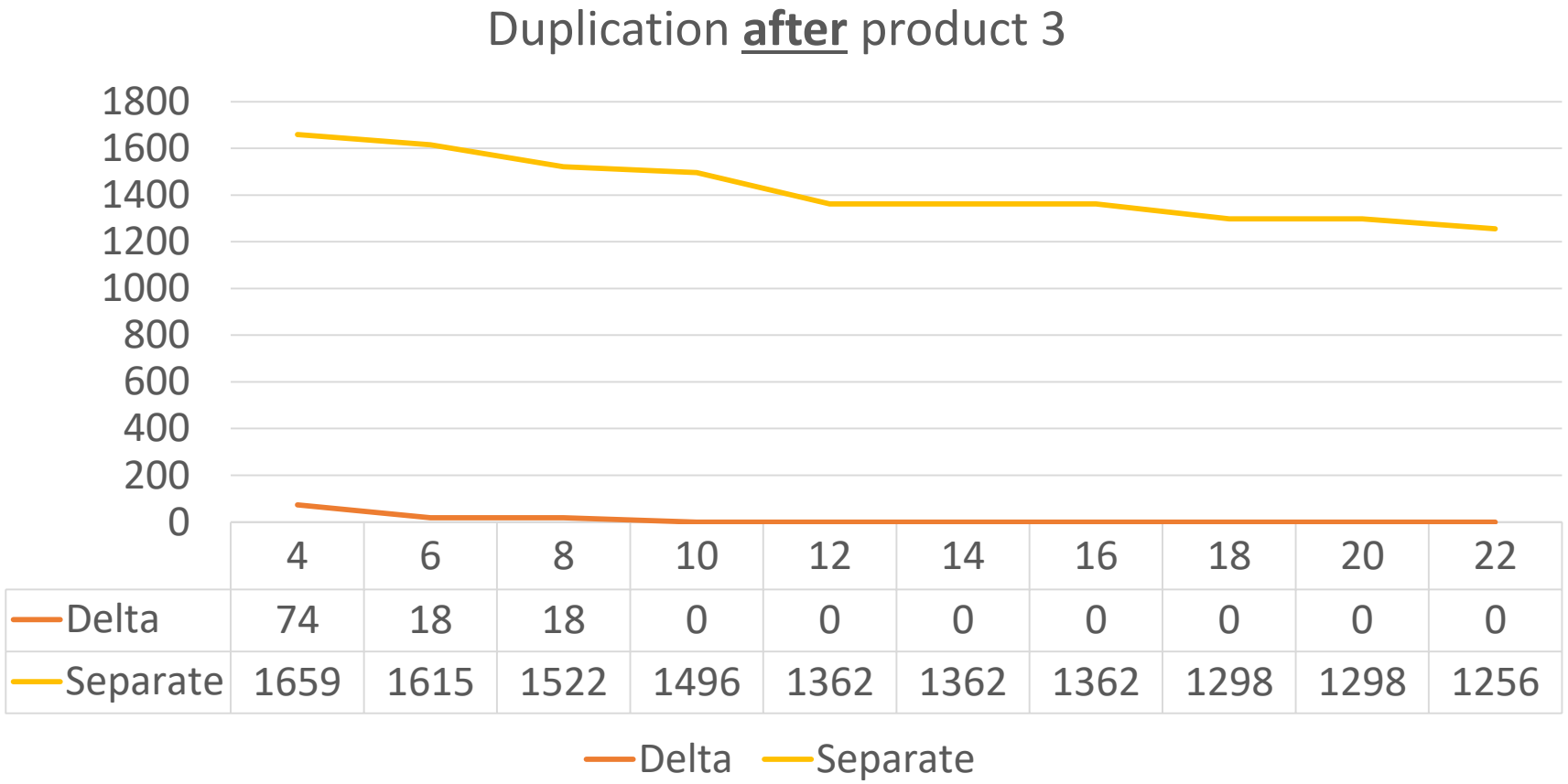
Results – Quantitative

- Degree of code duplication



Results – Quantitative

- Degree of code duplication



	4	6	8	10	12	14	16	18	20	22
Delta	74	18	18	0	0	0	0	0	0	0
Separate	1659	1615	1522	1496	1362	1362	1362	1298	1298	1256

Qualitative Observations & Discussion

UNIVERSITY OF CALGARY

Criteria	DOP	Separate
Complexity	**High** due to the addition of delta's layer	**Low** as products diverge more easily and evolve in an uncoordinated manner
Duplication	**Less**	**Higher**
Complications of Bug fixing	Results in **lower** duplication	Results in **higher** duplication
Tool Support	**Not mature**	**Mature**

Threats to Validity

- Our conclusions may not generalize
- Focused only coarse-grained changes in the system’s actual evolution

Future Work

- Extend the study to the complete set of products (8 products)
- Study the effect of refactoring the FM and the codebase
- Repeat the methodology on other systems
- Study the evolution on the individual commit level

Related Work

- SPL implementations: such as pre-processors, object-oriented, component-oriented, feature-oriented, aspect-oriented, or delta-oriented programming [7], [13], [4], [29], [30], [32]
- One-time transformations from independent products to a product line architecture [e.g., 2, 12, 16, 28]
- Difficulties from SPL evolution [e.g., 15, 34, 36]
- Code clones [9, 17, 18, 20, 26, 27].

Summary

- Problem
  - Unclear if the benefits outweigh the costs during unanticipated SPLE evolution
- Solution
  - A case study that measure the evolution of separate products approach vs. DOP
- Results
  - No winner, each approach has its own merits and faults