

---

# Integrating Product Line Engineering and Agile Methods: Flexible Design Up-Front vs. Incremental Design

---

**Ralf Carbon**

**Mikael Lindvall\***

**Dirk Muthig**

**Patricia Costa\***

Special thanks to J. Prof. Dr. Rausch, Daniela Cruzes, and Thomas  
ASPL - 1st International Workshop on Agile Product Line Engineering  
ASPL  
10th International Software Product Line Conference (SPLC 2006)  
August 21st, 2006

---



© 2006 Fraunhofer Center Maryland and  
Fraunhofer IESE

"Integrating Product Line Engineering and Agile Methods"

---

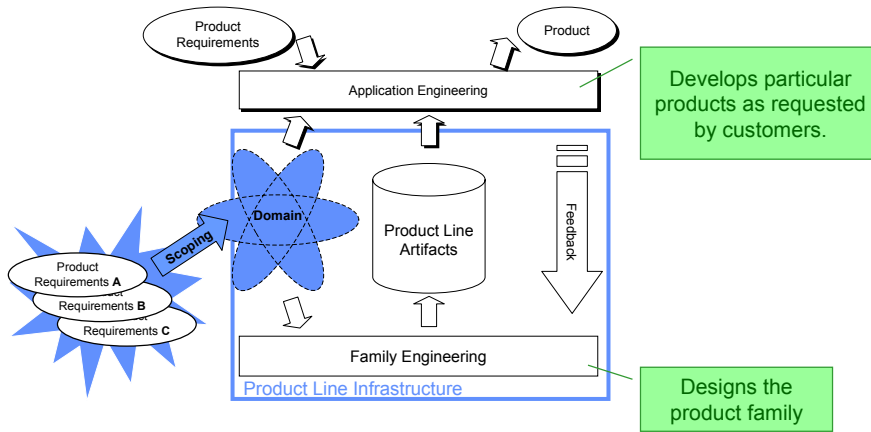
## Product Line Engineering and Agile Methods

- Both approaches have shown
    - to reduce time to market, and
    - to guarantee quality
  - However, these approaches differ
    - Example: Design strategies
      - PLE: flexible, up-front design to set up a reference architecture for a family of products
      - AM: simple incremental approach that only designs for the product at hand
  - PLE and AM can be combined to further speed time to market and increase quality!
- 



© 2006 Fraunhofer Center Maryland and  
Fraunhofer IESE

## Product Line Life Cycle



## PuLSE-I

- PuLSE™ (Product Line System and Software Engineering) is a **full life cycle product line process** that has been successfully used for many years at Fraunhofer IESE.
- PuLSE-I is an instantiation of PuLSE™: the **application engineering process** of PuLSE™

## PuLSE-I Process

- Plan a product line instance:
  - Identify match between product specific requirements and the scope of the product line. Estimate effort for constructing new product line assets (Family Engineering) and implementing product specific items (Application Engineering).
- Instantiate and validate product line model:
  - Make decisions related to assets reused in the product at hand
- Instantiate and validate reference architecture:
  - Define product-specific architecture based on reference architecture
- Construct product:
  - Perform implementation of product at hand and product line architecture; integration and testing
- Deliver the system

## PuLSE-I and Agile Methods

- Product Line Engineering:
  - up-front investment for reusable and flexible artifacts to shorten development time of new products;
  - success depends on anticipation of the right change scenarios
- Agile Methods:
  - Focus is on the requirements at hand: design emerges from the code
  - commonalities are not leveraged but products are always more or less built from scratch.
- Combination of PLE and AM: empower organizations to reach agility

## Combining PuLSE-I and Agile Methods

- Coordination between application engineering (AE) and family engineering (FE) crucial to leverage full agility of an organization:
  - Example 1 - Planning game: direct communication between customers and both the AE and FE teams can bring the following benefits:
    - Reuse rate can be increased
    - Redundant implementations of product specifics in several AE projects can be reduced
  - Example 2 – Incremental design: focus on requirement at hand; does not aim at building flexibility
    - FE concentrates on supporting in their current iterations by increasing the reusability of the components already chosen for reuse
      - In this case, FE builds flexibility for the right change scenarios!

## Does built-in flexibility pay off? A Class-room Experiment

- 67 students participated in the experiment
- The first two weeks of the lecture “Foundations of Software Engineering” at the University of Kaiserslautern
- The students worked in pairs
- There were 38 groups
  - 19 simple incremental design
  - 19 were assigned flexible up-front design

## Assigned Design Strategy

- Simple Approach
  - You should design simple and incrementally, i.e., take only the the actual change request into account in every iteration.
- Flexible Approach
  - You should invest up-front in design
  - i.e. take the first change request into account and take future changes into account in the first design phase.

## Design

- All groups received **exactly** the same information:
  - Documentation and Code of TSAFE 2b
  - Five change requests (handed out one at a time)
  - A document describing desirable future features
- The **only** variation was the assigned process
- All groups were requested to hand in the same artifacts (including design documents, source code, and output from test runs) for every change request.
- In addition, they reported the effort spent

## Reasoning about ROI

- Base0 represents the initial system
- In order to implement a change request CR<sub>n</sub>, new code needs to be added, existing code needs to be deleted or changed.
- This change is represented by  $\Delta CR_n$
- For each CR, flexible groups will strive to build-in flexibility to prepare for what might be required by any set of future CR<sub>n</sub>.
- Thus additional code needs to be added, existing code needs to be deleted or changed. This additional change is represented by  $\Delta CR_x$

## The work spent on developing CR1 and CR2

- CR1. The changes simple groups perform:
  - $BaseS1 = Base0 + \Delta CR1$
- CR1. The changes flexible groups perform:
  - $BaseF1 = Base0 + \Delta CR1 + \Delta CRX1$
- For the next change request(s), CR<sub>n+1</sub>, flexible groups will gain from the flexibility built into the base:
  - flexibility(BaseF<sub>n</sub>),
- Simple groups will pay a penalty for built in inflexibility:
  - + inflexibility(BaseS<sub>n</sub>)
- CR2. Simple:  $BaseS2 = BaseS1 + \Delta CR2 + inflexibility(BaseS1)$
- CR2. Flexible:  $BaseF2 = BaseF1 + \Delta CR2 - flexibility(BaseF1) + \Delta CRX2$

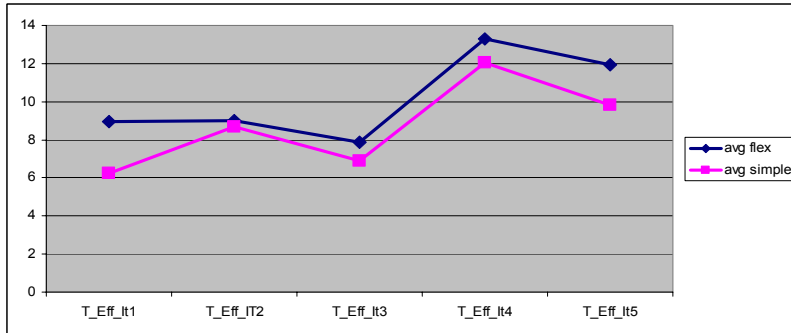
## ROI for a series of change requests

- Work for flexible groups:
    - $WF = \Delta CR_{n+1} - \text{flexibility}(\text{BaseFn}) + \Delta CR_{n+1}$
  - Work for simple groups:
    - $WS = \Delta CR_{n+1} + \text{inflexibility}(\text{BaseSn})$
  - $WF > WS$  (negative ROI)
    - More work devoted to adding flexibility than benefits from built-in flexibility
  - $WF \approx WS$  (zero ROI)
    - Same work devoted to adding flexibility as the benefits received from built-in flexibility
  - $WF < WS$  (positive ROI)
    - Less work devoted to adding flexibility than benefits received from built-in flexibility
- 

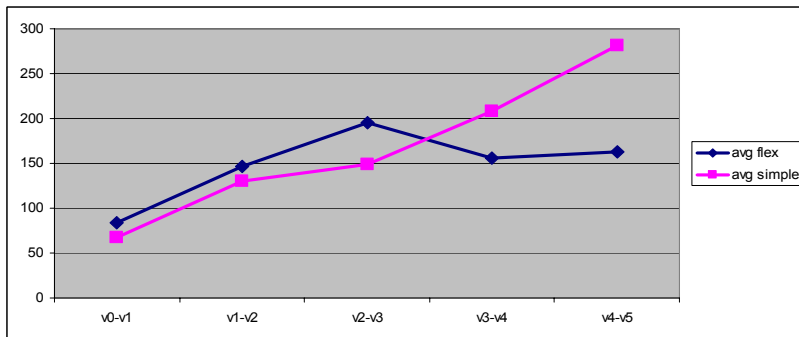
## Some Preliminary Results from Experiment

- Ongoing work!
- Turned out to be difficult to analyze data
- E.g. Many different ways to implement the same change request

### Flexible groups seem to spend more time than Simple groups



### Flexible groups seem to change less than Simple groups



Note: Only data from 6 simple, 11 flexible groups used at this point!

## Conclusion

- Product line engineering and Agile Methods can be beneficially combined to help an organization to deliver products faster with higher quality
- We need to better understand how to combine PLE and AM in the different aspects of the software development process
- Experimentation can help develop such understanding:
  - BDUF vs. simple design: flexible groups seem to spend more time and less changes than simple groups
- The results of the experiment seem to indicate that design is a good area to start with balancing agility and product line engineering:
  - AM in PLE reduces time spent on design (increase speed)
  - Use PLE to keep changes to a minimum (increase quality)
- Next: try to apply the ideas in a real product line context - Ralf's PhD topic!

## Conclusion

- Product line engineering and Agile Methods can be beneficially combined
  - PLE for family engineering (FE) and
  - AM for application engineering (AE)
- But the division is not so black and white
  - For example agile practices, such as the Planning Game, can be used for FE!
- We need to better understand when each approach is more appropriate and beneficial
- Experimentation can help develop such understanding
- E.g. Flexible groups seem to spend more time and less changes than Simple groups
- The results of the experiment