Flexible Product Line Engineering With a Virtual Platform

Michał Antkiewicz

Mar 10, 2015

UNIVERSITY OF WATERLOO

Generative Software Development Lab

http://gsd.uwaterloo.ca

necsis

http://necsis.ca

Flexible Product Line Engineering with a Virtual Platform

Michał Antkiewicz, Wenbin Ji,
Thorsten Berger, Krzysztof Czarnecki
University of Waterloo, Canada

Stefan Stănciulescu, Andrzej Wąsowski
IT University of Copenhagen, Denmark

Thomas Schmorleiz, Ralf Lämmel
Universität Koblenz-Landau, Germany

Ina Schaefer
Technische Universität Braunschweig, Germany

ABSTRACT

Cloning is widely used for creating new product variants. While it has low adoption costs, it often leads to maintenance problems. Long term reliance on cloning is discouraged in favor of systematic reuse offered by product line engineering (PLE) with a central platform integrating all reusable assets. Unfortunately, adopting an integrated platform requires a risky and costly migration. However, industrial experience shows that some benefits of an integrated platform can be achieved by properly managing a set of cloned variants.

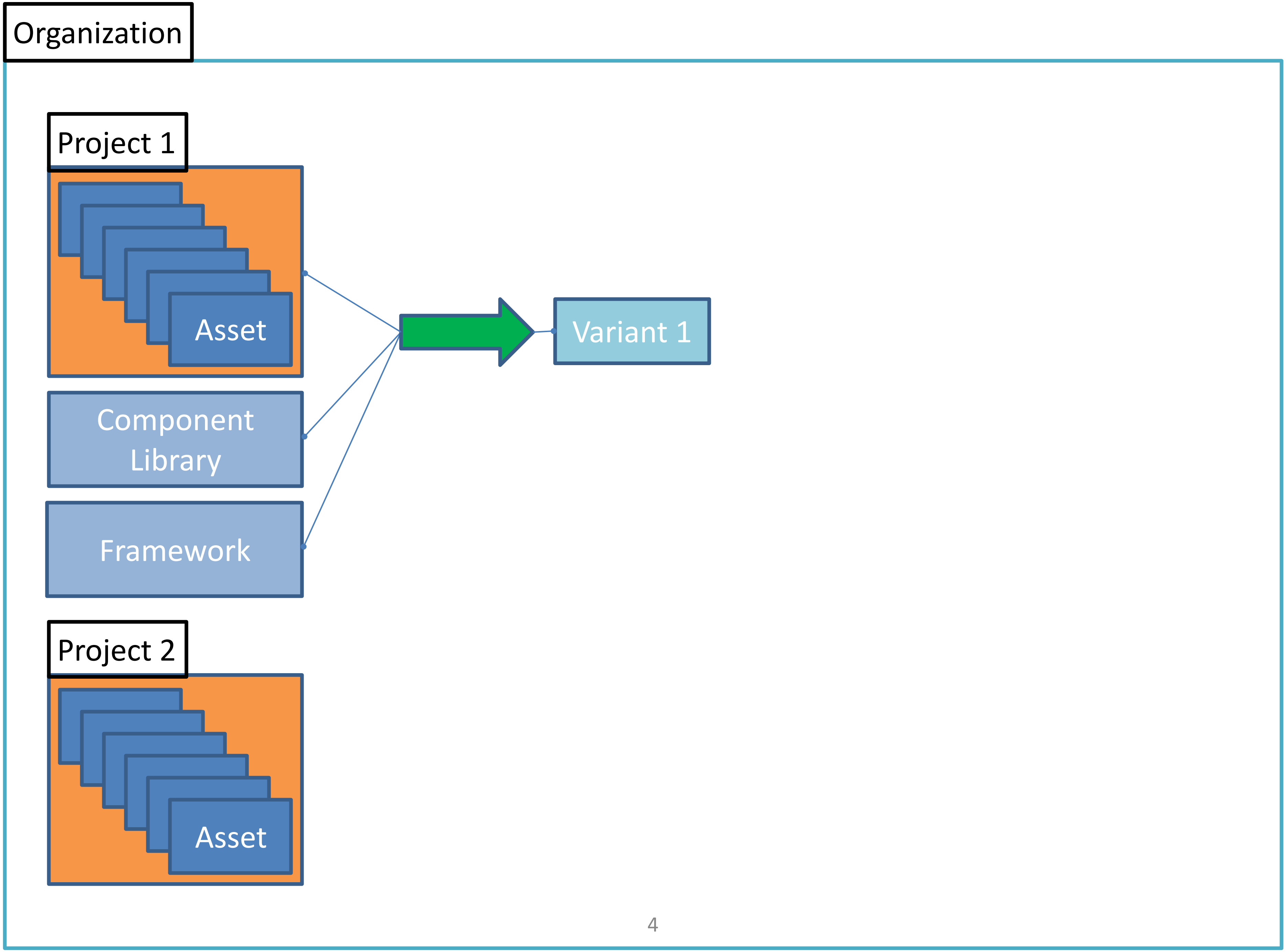
In this paper, we propose an incremental and minimally invasive PLE adoption strategy called *virtual platform*. Virtual platform covers a spectrum of strategies between *ad-hoc clone* ... *fully-integrated platform* divided

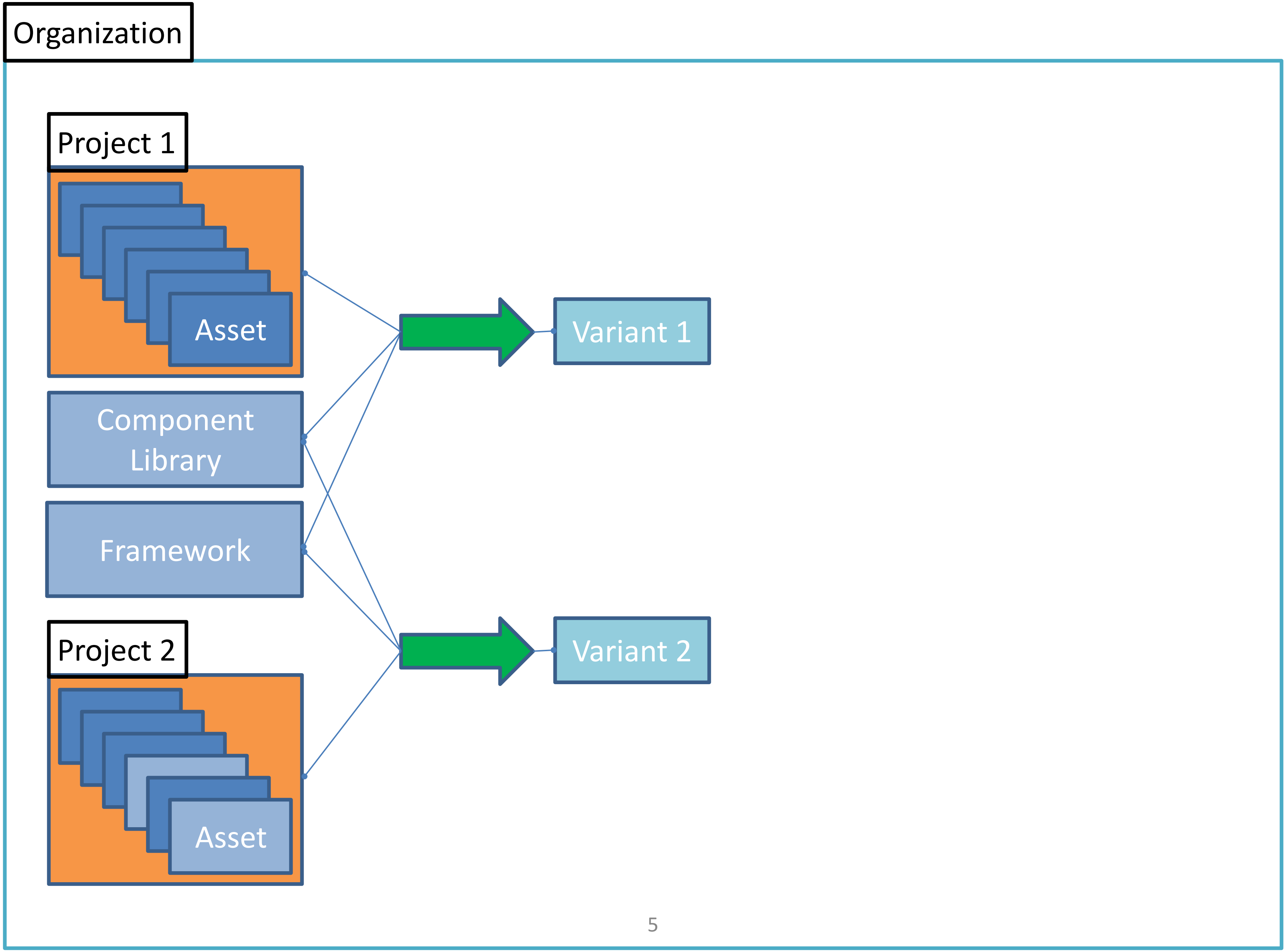
Despite having low adoption costs and allowing indepe dence from other developers, cloning easily leads to incons tencies, redundancies, and lack of control. In the literatu using cloning in the longer term has been considered a ha ful practice [8]. It has been traditionally recommended t organizations adopt a more systematic, strategic reuse offe by *product line engineering* (PLE) [10] based on a cen platform. Such a platform should integrate the reusa assets and it should be used for deriving new variant products. Existing incremental PLE adoption strategie 6] discourage relying on cloning due to maintainability is However, as shown by industrial practice, eliminating clo and adopting the integrated platform is not always desi nor beneficial as it requires high-risk migration proces

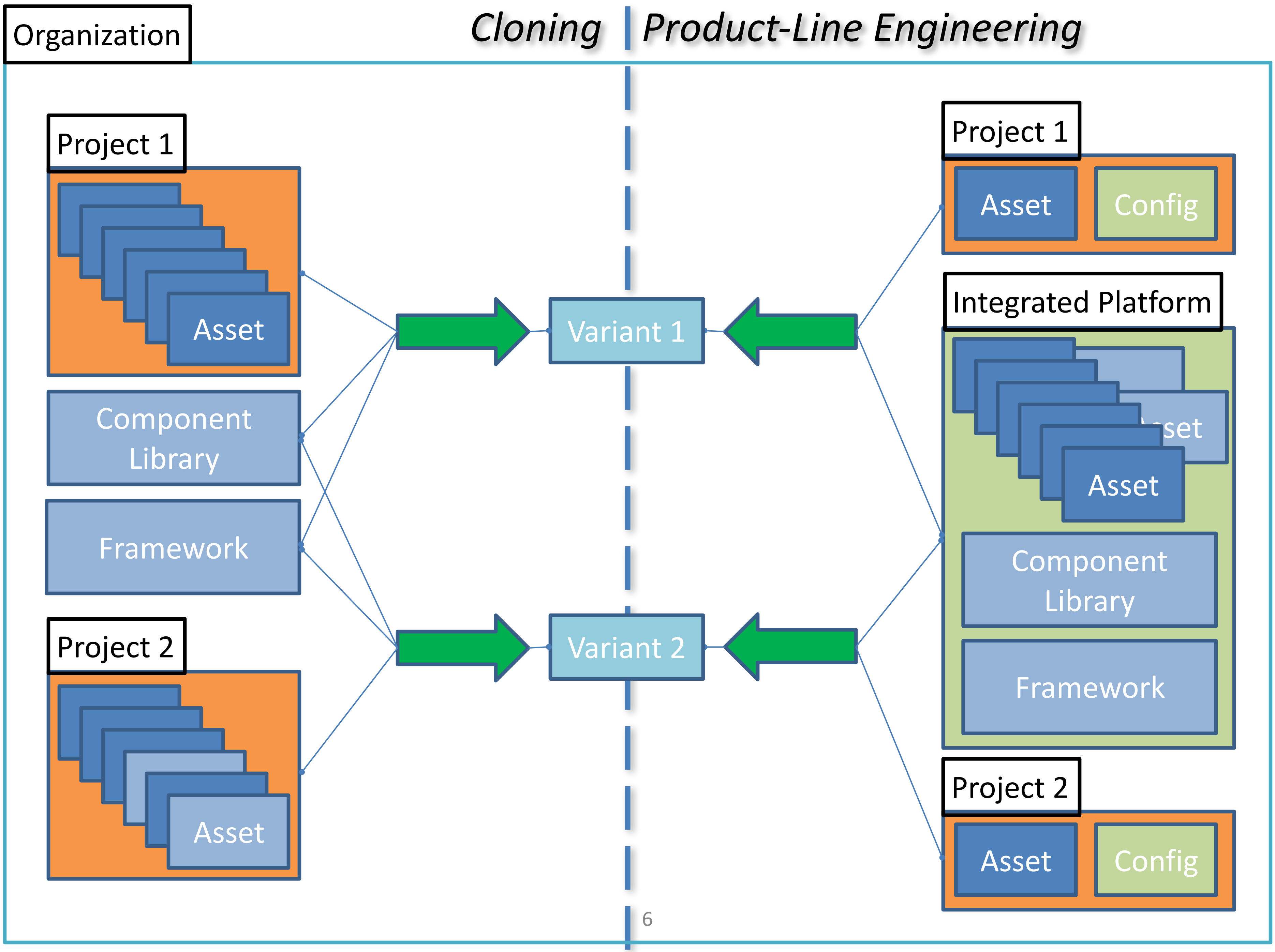
In this paper, we present an *incremental* and *mini* ... strategy for adoption of product-line engin ... platform allows organiz

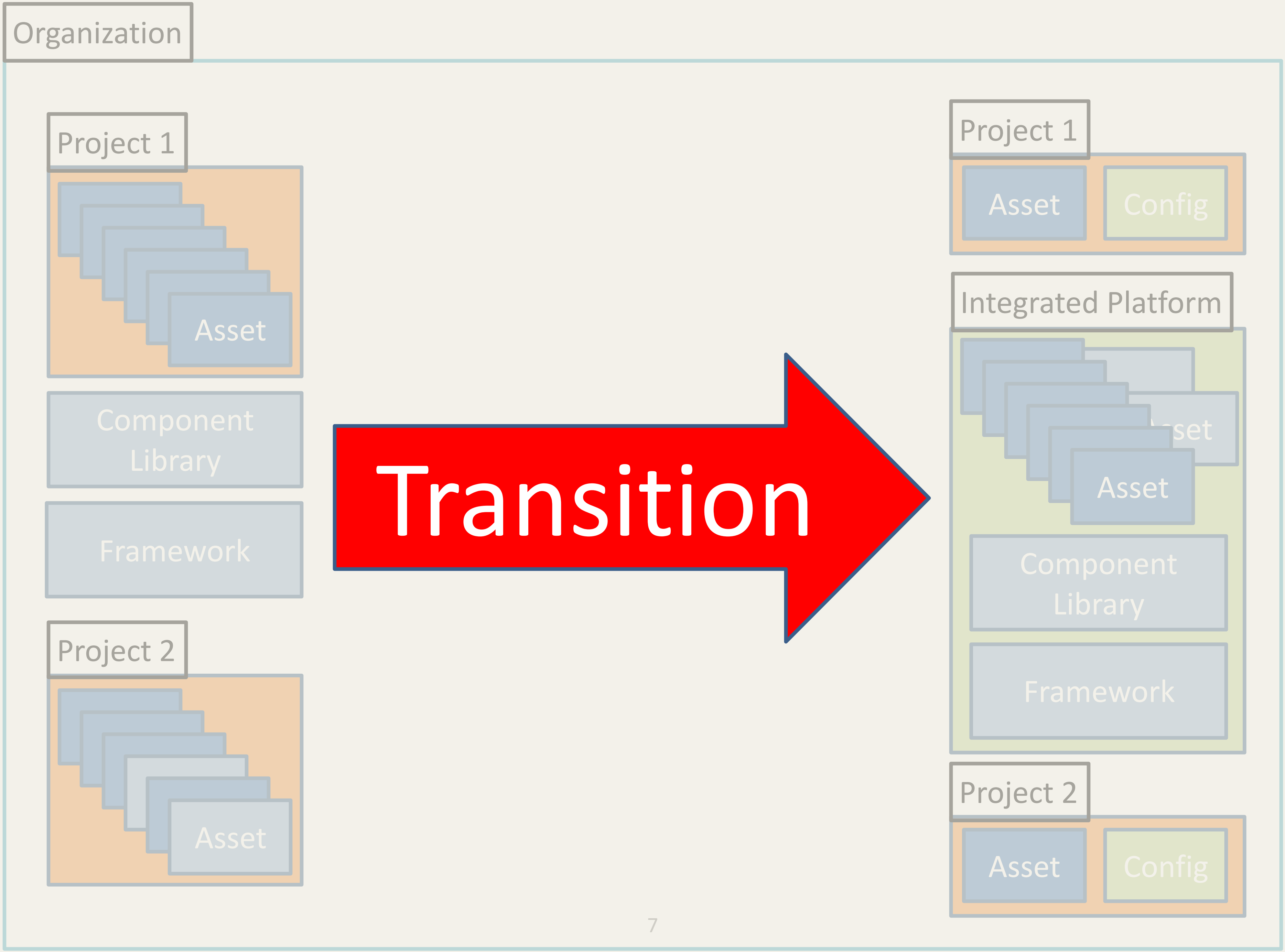
“Virtual Platform” is …

an *incremental* and *minimally invasive* **strategy** for adoption of product line engineering

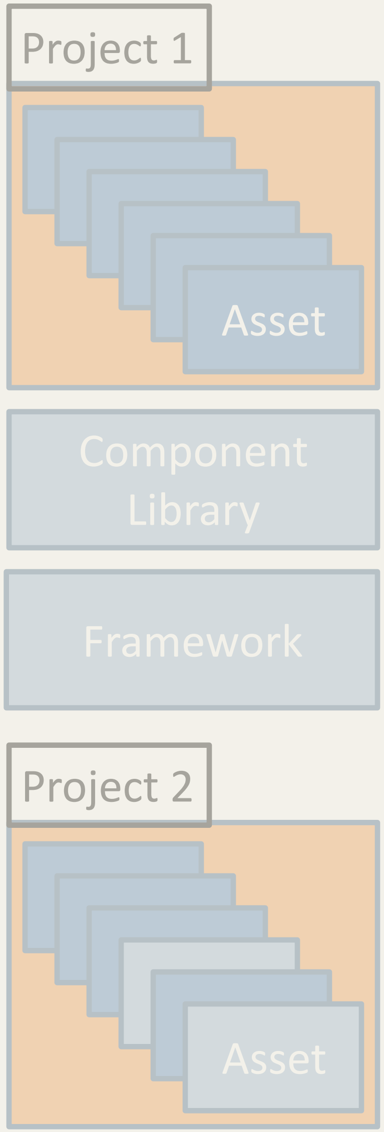

Organization
Project 1
Asset
Component Library
Framework
Variant 1
Project 2
Asset


Organization
Project 1
Asset
Component Library
Framework
Project 2
Asset
Variant 1
Variant 2


Organization
Cloning
Product-Line Engineering
Project 1
Asset
Component Library
Framework
Project 2
Asset
Variant 1
Variant 2
Project 1
Asset
Config
Integrated Platform
Asset
Component Library
Framework
Project 2
Asset
Config


Organization
Project 1
Asset
Component Library
Framework
Project 2
Asset
Transition
Project 1
Asset
Config
Integrated Platform
Asset
Component Library
Framework
Project 2
Asset
Config

Organization



+independence

+flexibility

+innovation

+speed

+low cost of initial reuse

?

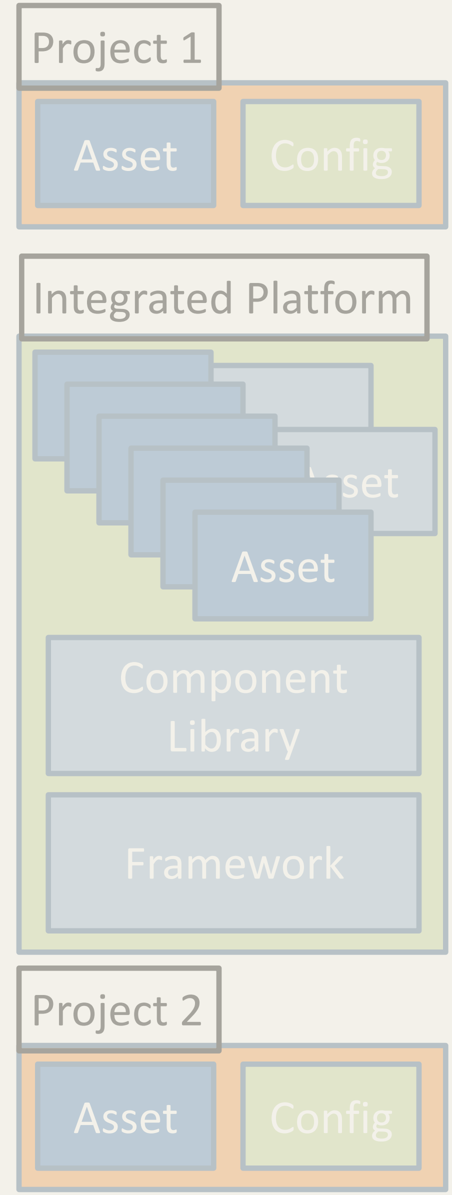
+scale

+propagation

+low redundancy

+new variants

+configuration over implementation



“Make the distributed assets reusable instead of integrating them into a platform”

Key Idea 1

Clone Management Framework

Rubin et al., SPLC’12, ICSE NIER’13, SPLC’13 (Best Paper Award)

“Offer incremental benefits for incremental efforts”

Key Idea 2

Virtual Platform = 6 Governance Levels

For each level

- Description
- Advantages
- Disadvantages
- Tactics
- (Example)
- Recommendation

Governance Levels

L6: PLE with a Fully Integrated Platform

L5: PLE with an Integrated Platform and Clone & Own

L4: Clone & Own with a Feature Model

L3: Clone & Own with Configuration

L2: Clone & Own with Features

L1: Clone & Own with Provenance

L0: Ad-Hoc Clone & Own

“Each level is ‘good’ given the specific needs”

Key Idea 3

"Cloning Considered Harmful" Considered Harmful

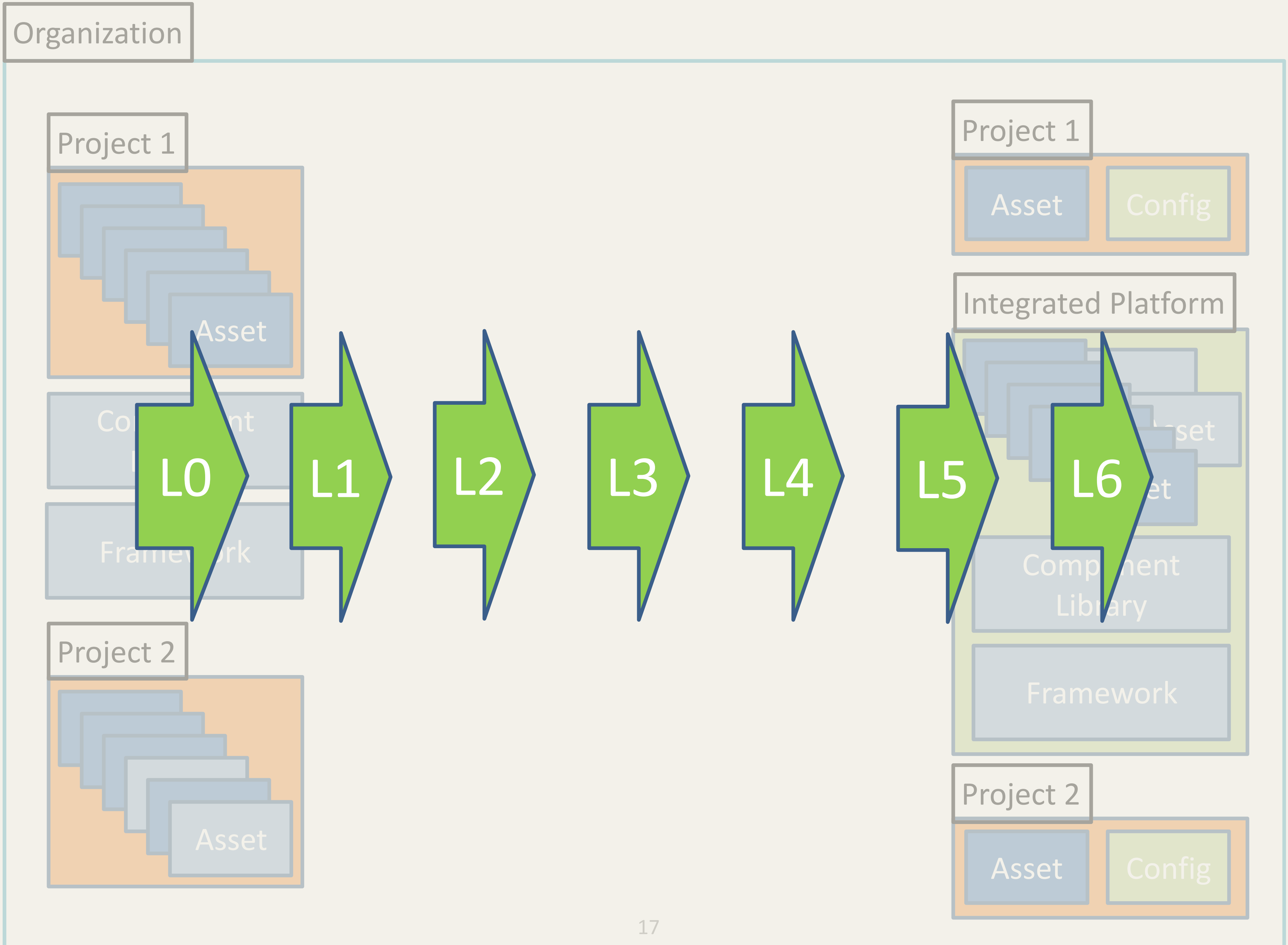
Kapser and Godfrey, WCRE '06

“Integrated Platform not Always Desirable”

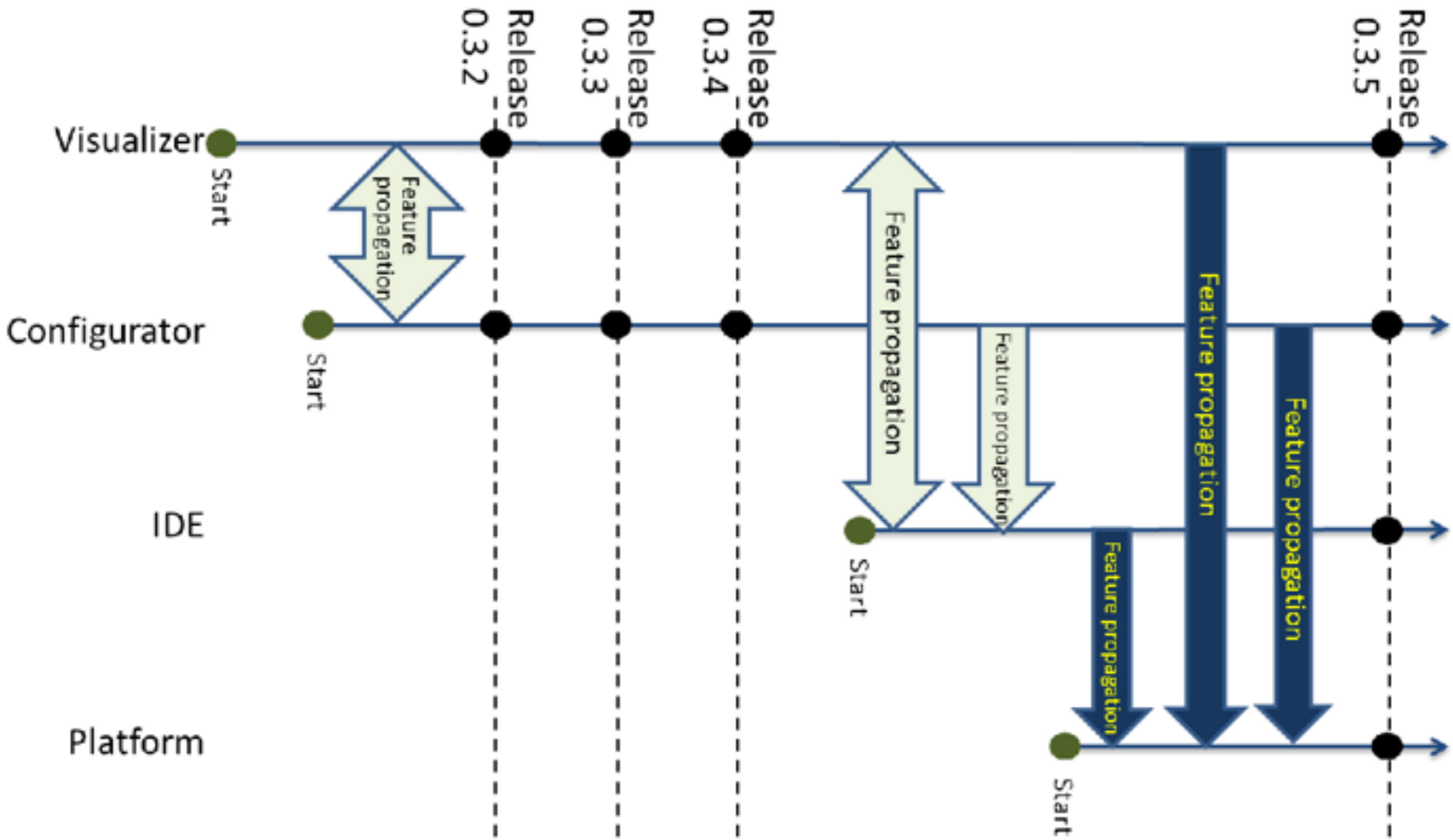
Dubinsky et al., CSMR, 2013

(Best Paper Award)

Stallinger et al., PLEASE, 2011


Organization
Project 1
Asset
Project 2
Asset
L0
L1
L2
L3
L4
L5
L6
Project 1
Asset
Config
Integrated Platform
Framework
Project 2
Asset
Config

Case Study: Clafer Web Tools



Maintaining Feature Traceability with Embedded Annotations

Wenbin Ji, Thorsten Berger, Michał Antkiewicz, Krzysztof Czarnecki
Generative Software Development Lab
University of Waterloo, Waterloo, Canada
Email: {w6ji,tberger,mantkiew,kczarnec}@gsd.uwaterloo.ca

Abstract—*Features* are commonly used by developers and users to describe the functional and non-functional aspects of software. While widely used in general, the notion of features is predominant in software product lines, where features are used for distinguishing among variants of software. As such, features are often the main units of software reuse, abstracting over implementation details. To effectively evolve and reuse features, their location in software assets—such as requirements, code, tests, build systems—has to be known. However, locating features is often difficult given their crosscutting, non-modular nature. Once a feature is implemented, the knowledge about its location quickly deteriorates when the software evolves or development teams change—requiring expensive recovering of these locations.

We present a lightweight annotation approach to record feature traceability information throughout the development. It is based on a principle that *features belong to software assets*: traceability information should be declared, embedded, and evolved together with the software assets implementing the features. We apply our annotation approach in a case study, simulating the development of a set of cloned/forked projects. We investigate cost and benefit of these annotations for propagating features, maintaining their consistency, and migrating cloned features into an integrated product line platform. Our results show that the cost of adding and maintaining annotations is small compared to the actual development cost, while they enhance ... migration of clones into a platform ...

features are implemented in a specific project or variant. This representation allows comparing projects on level of abstraction higher then code, supporting developers and stakeholders in keeping a common understanding of the projects [11].

Performing tasks on features, such as, fixing bugs, modifying, refactoring, disabling, reusing, and cloning, requires knowing the *location* of the features in the code. In fact, feature location is considered one of the most common activities of developers [12], [13], [14], [15].

Organizations that rely on features are faced with the difficult question: *"How to effectively maintain traceability between the features and the corresponding software assets?"* Two strategies are possible: either organizations record feature traceability information during the development of the features (the *eager* strategy), or they retroactively recover such information when needed (the *lazy* strategy).

In the eager strategy, developers record the location of features when they perform tasks related to these features or shortly thereafter, when the knowledge is still fresh in their memory. In the lazy strategy, if memory has deteriorated, developers recover the location of features by reading the code or by applying semi-automated feature location techniques.

When choosing the eager strategy, organizations face another ... *feature traceability information?"*

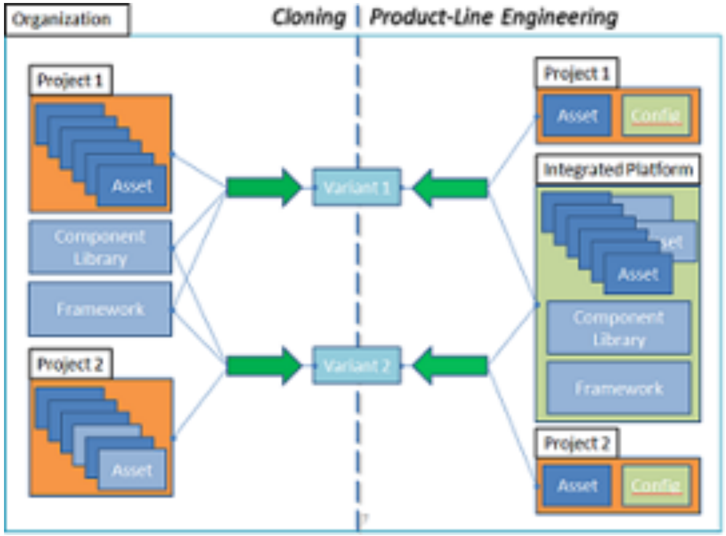
Conclusions

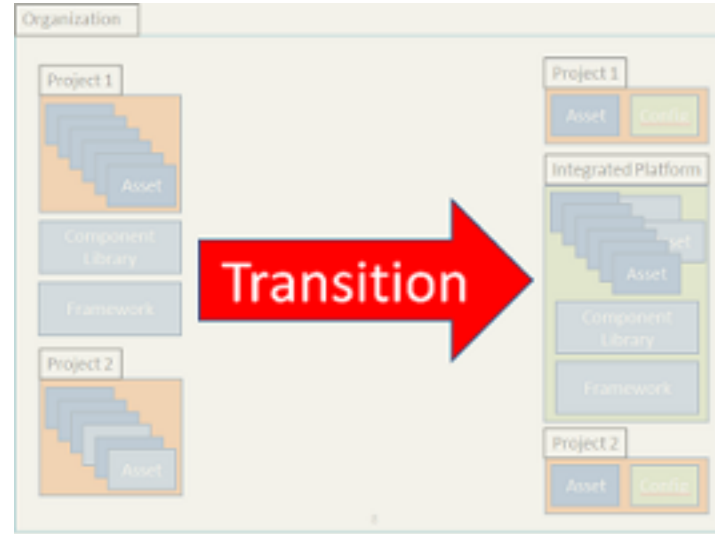
A roadmap for organizations

- Justifiable effort / expected benefits
- Ability to scale up reuse

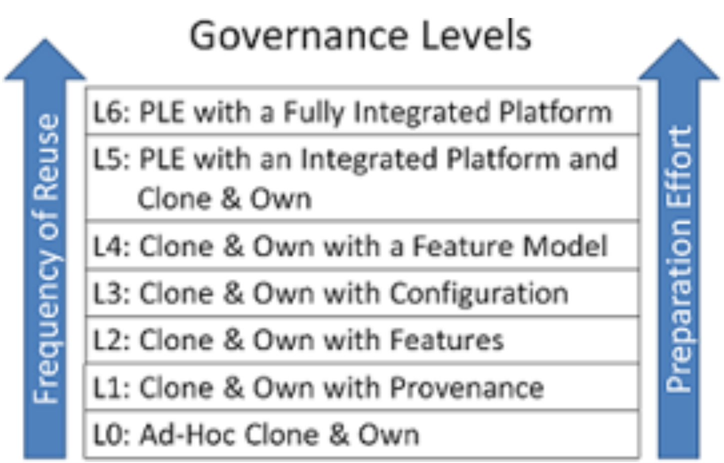
A way to achieve some benefits of PLE by SMEs

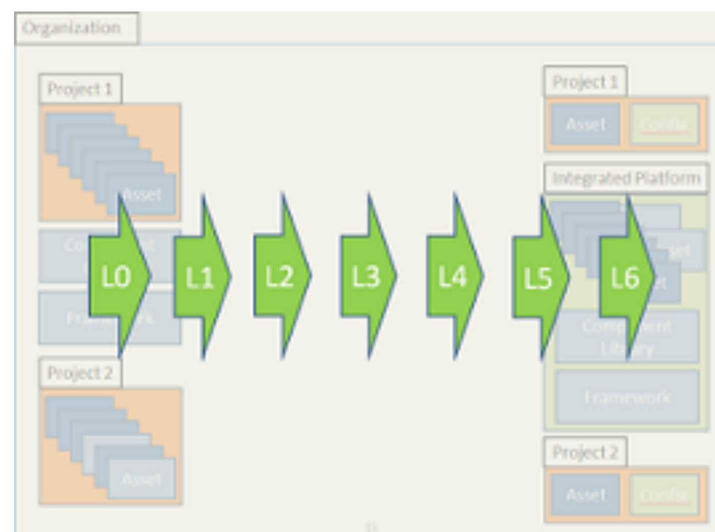
- Feature-oriented development
- Proactively or retroactively


Organization
Cloning | Product-Line Engineering
Project 1
Asset
Component Library
Framework
Project 2
Asset
Variant 1
Variant 2
Project 1
Asset
Config
Integrated Platform
Asset
Component Library
Framework
Project 2
Asset
Config


Transition

Thank You!


Governance Levels
Frequency of Reuse
Preparation Effort
L6: PLE with a Fully Integrated Platform
L5: PLE with an Integrated Platform and Clone & Own
L4: Clone & Own with a Feature Model
L3: Clone & Own with Configuration
L2: Clone & Own with Features
L1: Clone & Own with Provenance
L0: Ad-Hoc Clone & Own


L0
L1
L2
L3
L4
L5
L6