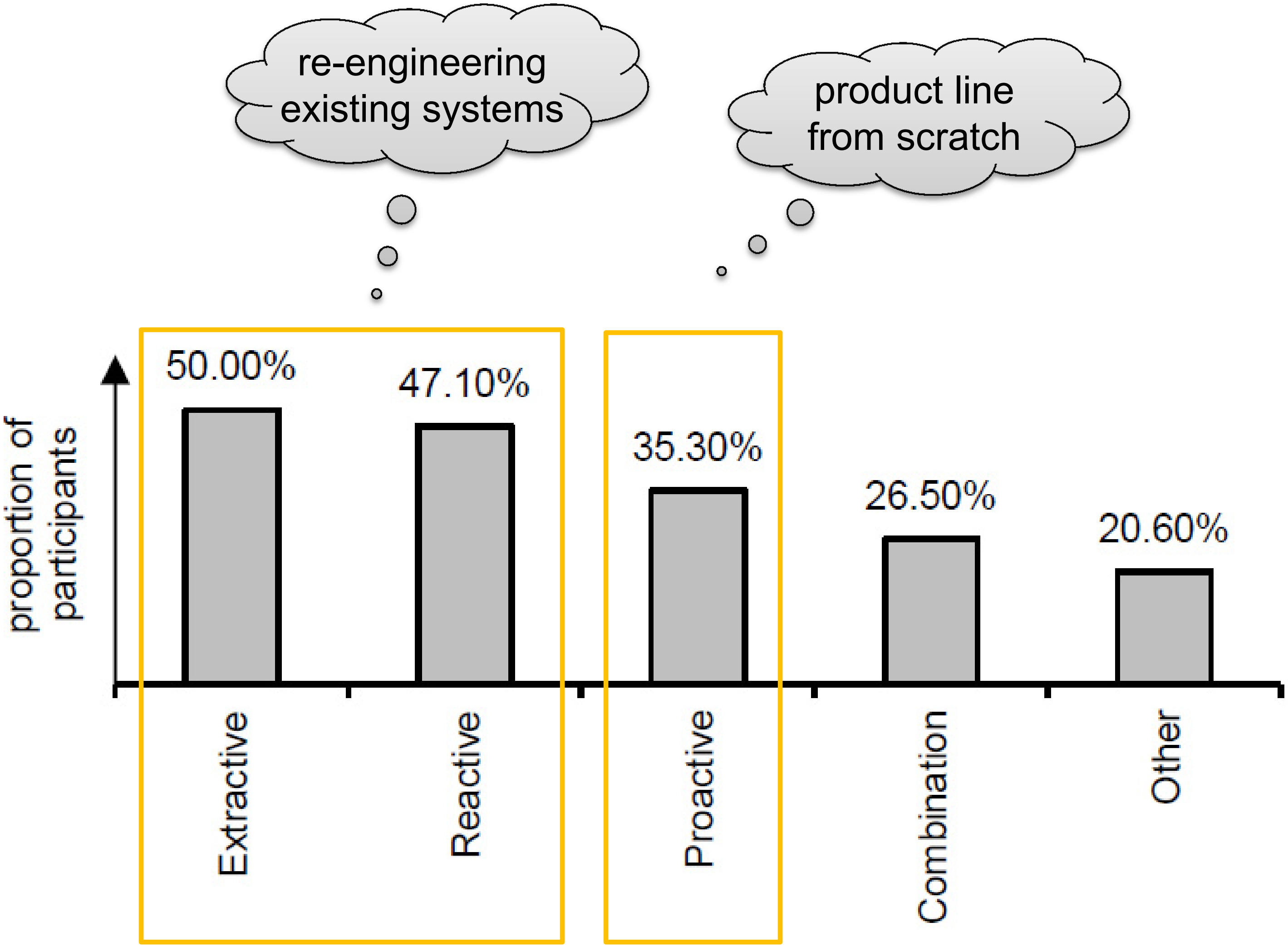
maintaining feature traceability with embedded annotations

Wenbin Ji, Thorsten Berger, Michal Antkiewicz, Krzysztof Czarnecki

University of Waterloo, Canada

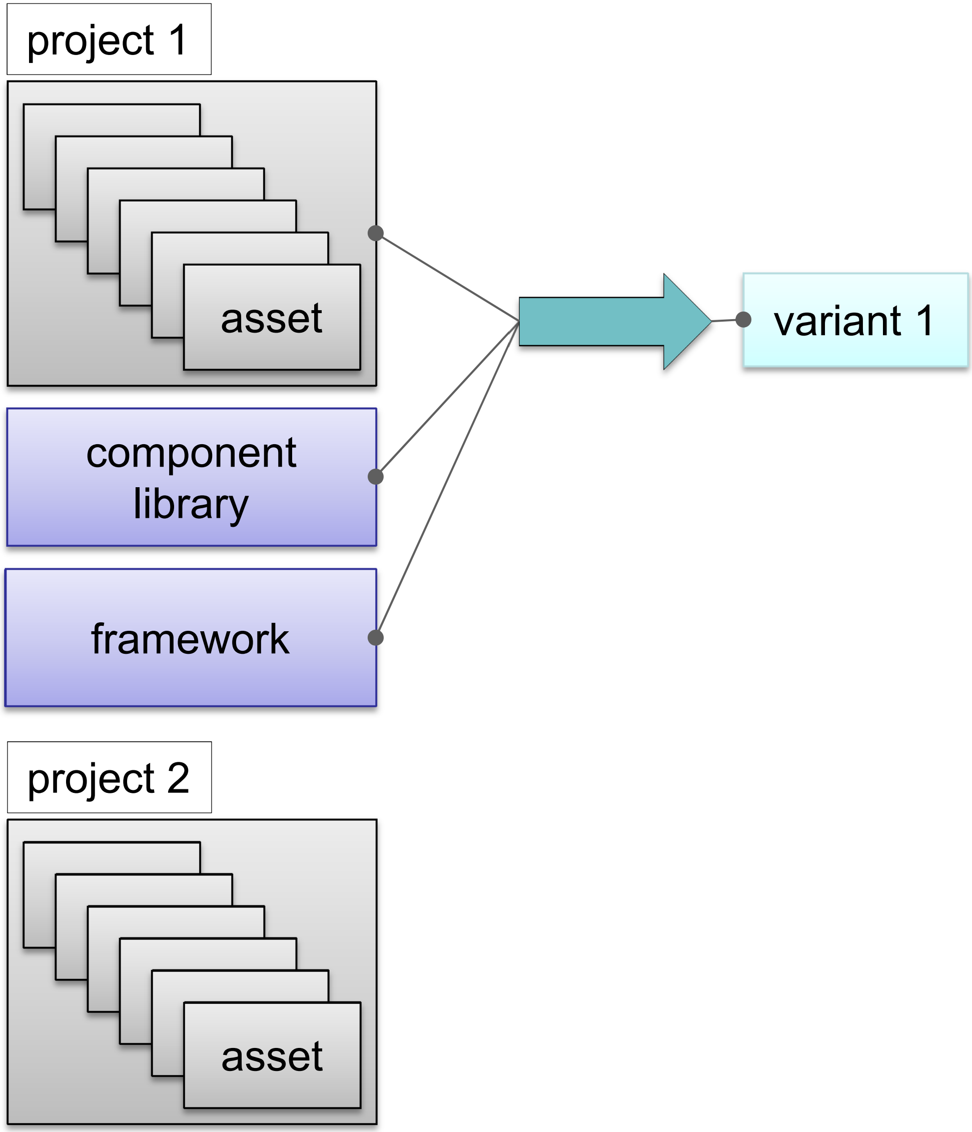
adoption of software product lines



re-engineering existing systems

product line from scratch

Berger et al. “A survey of variability modeling in industrial practice,” in VaMoS, 2013.



project 1

asset

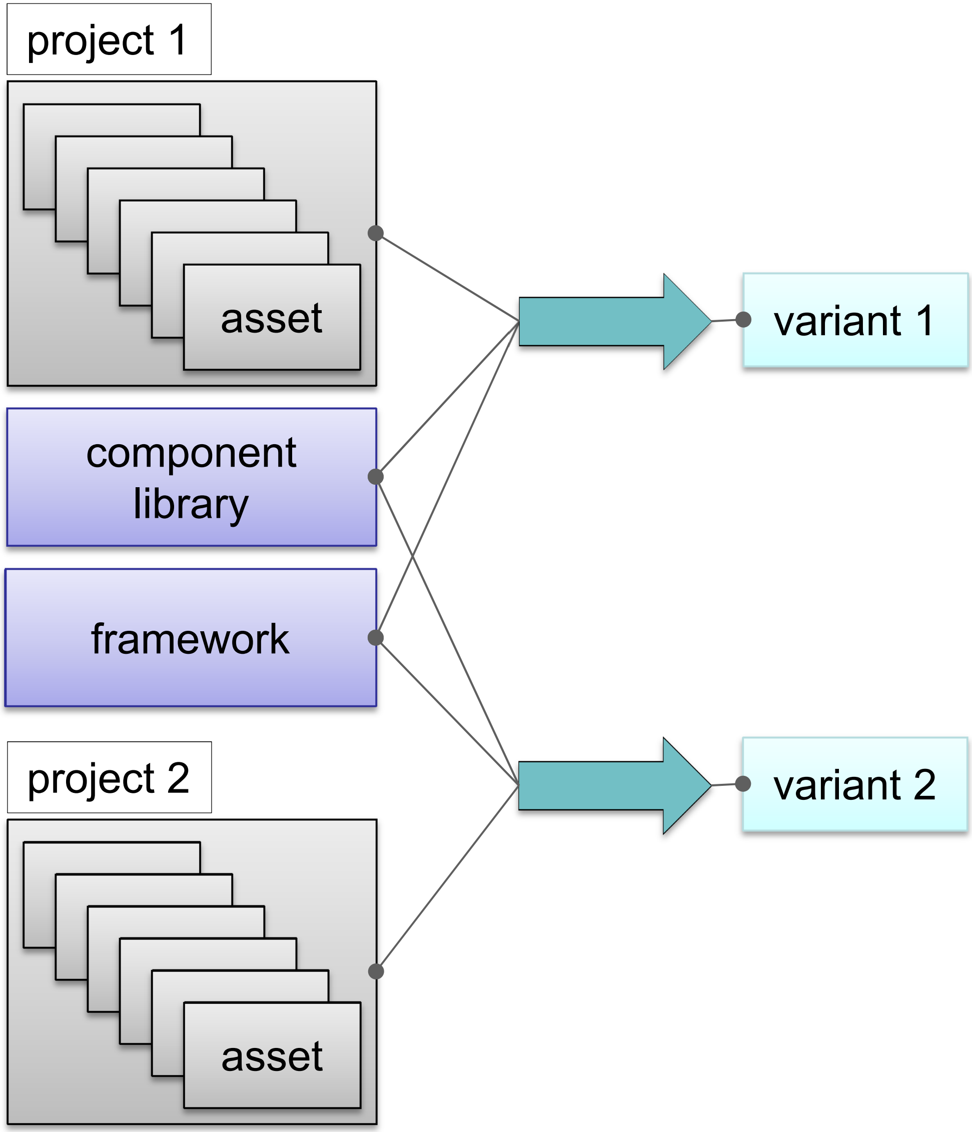
component library

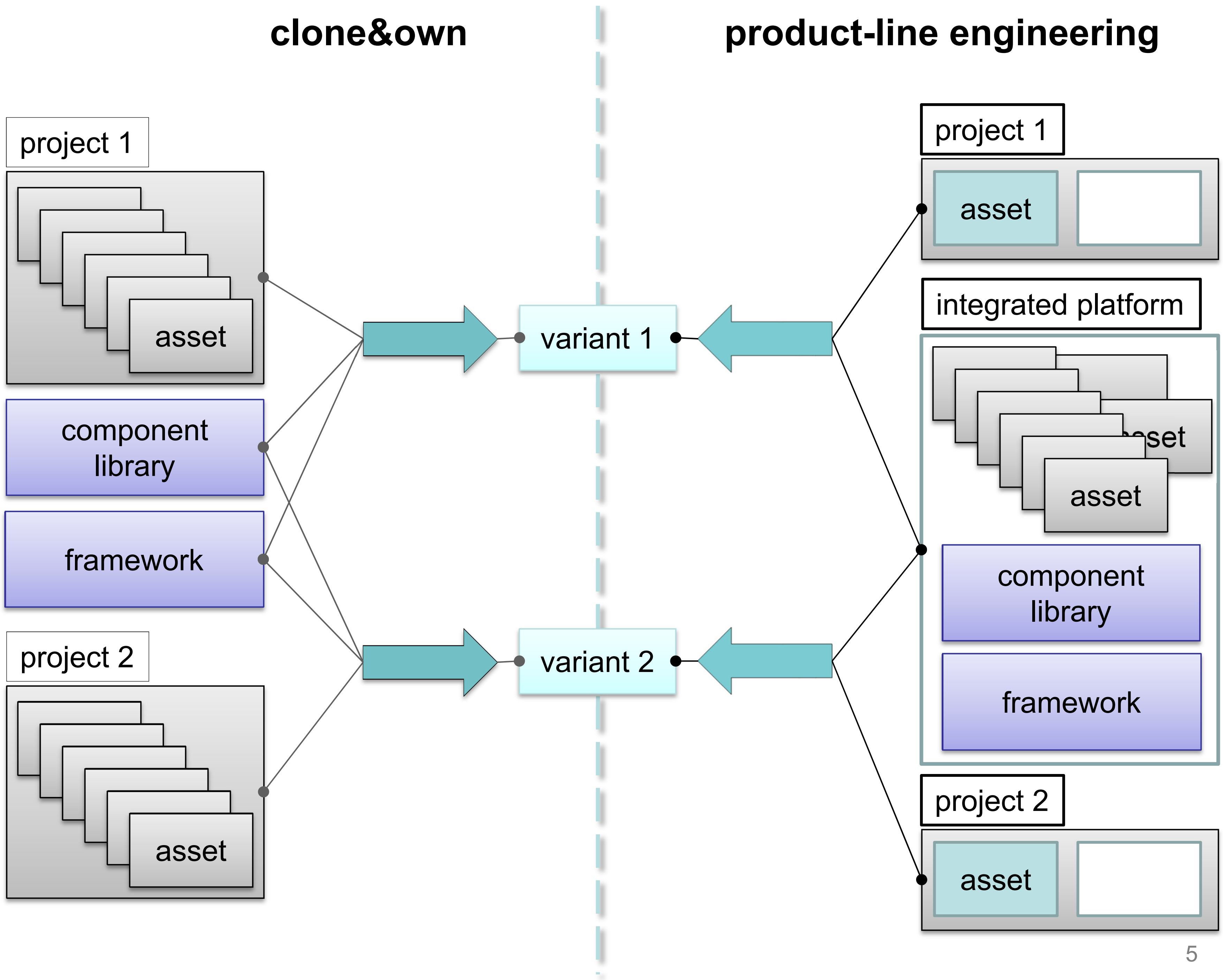
framework

variant 1

project 2

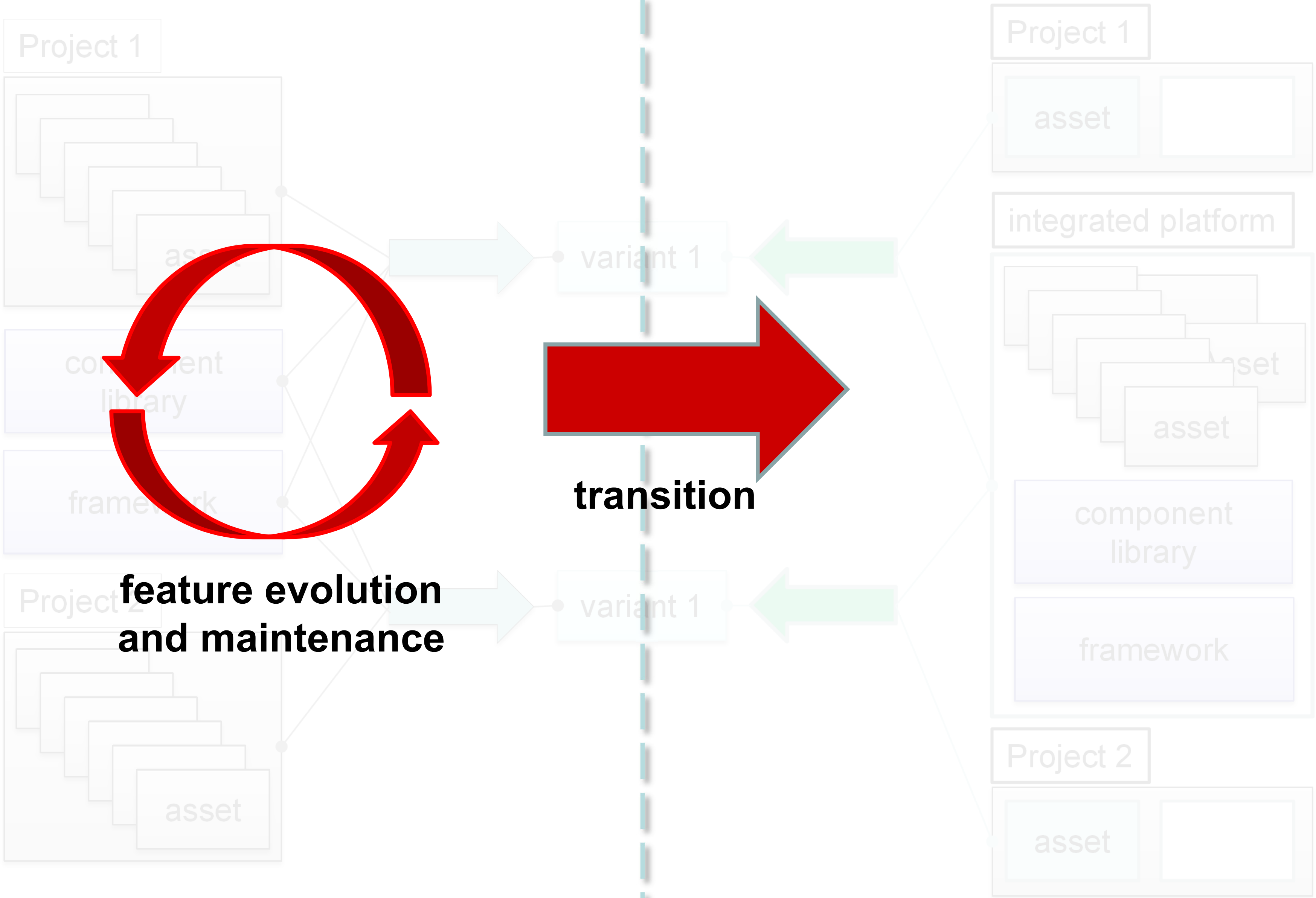
asset


project 1
asset
component library
framework
project 2
asset
variant 1
variant 2


clone&own
product-line engineering
project 1
asset
component library
framework
project 2
asset
variant 1
variant 2
project 1
asset
integrated platform
asset
asset
component library
framework
project 2
asset

clone&own

product-line engineering



feature evolution and maintenance

transition

challenge

feature location

…in development teams

hypotheses

Introducing and maintaining features, feature models, and traceability **early** eases feature maintenance and product-line adoption.

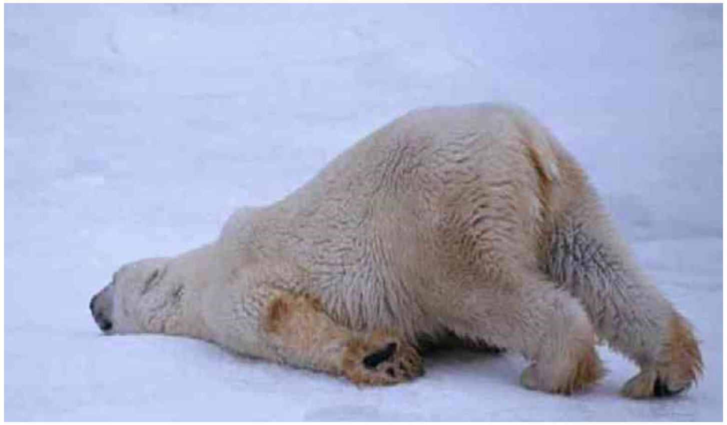
Resulting costs are amortized by the benefits.

how to maintain traceability?

TWO QUESTIONS

how to maintain traceability effectively?

recover traceability
when needed



lazy strategy

record and edit
traceability during
development

eager strategy

how to store traceability information?

store in a database

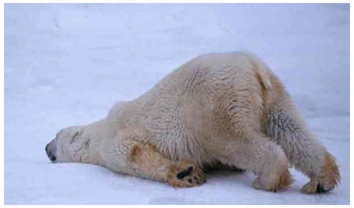
external storage

embed traceability into assets

internal storage

feature-location recovery

expensive and tools required



lazy + external

survey [Rubin et al. ’13]

tools have low precision and require high manual effort

experiments [Wang et al. ‘13]

systems with 73k, 2k, 43k, 19k LOC

average location time: 15min

Rubin, Chechik, “A survey of feature location techniques,” in *Domain Engineering*, 2013.

Wang et al., “How developers perform feature location tasks: a human-centric and process-oriented exploratory study,” *Journal of Software: Evolution and Process*, 2013.

embedded feature annotations?

eager + internal

SIMULATION CASE STUDY

research questions

invested cost

RQ1: What are annotation recording/editing costs?

annotation mistakes,
forgotten annotations,
incomplete knowledge

RQ2: How many annotation recordings/edits still required feature-location recovery?

recall

RQ3: How much of the invested cost prevented feature-location recovery?

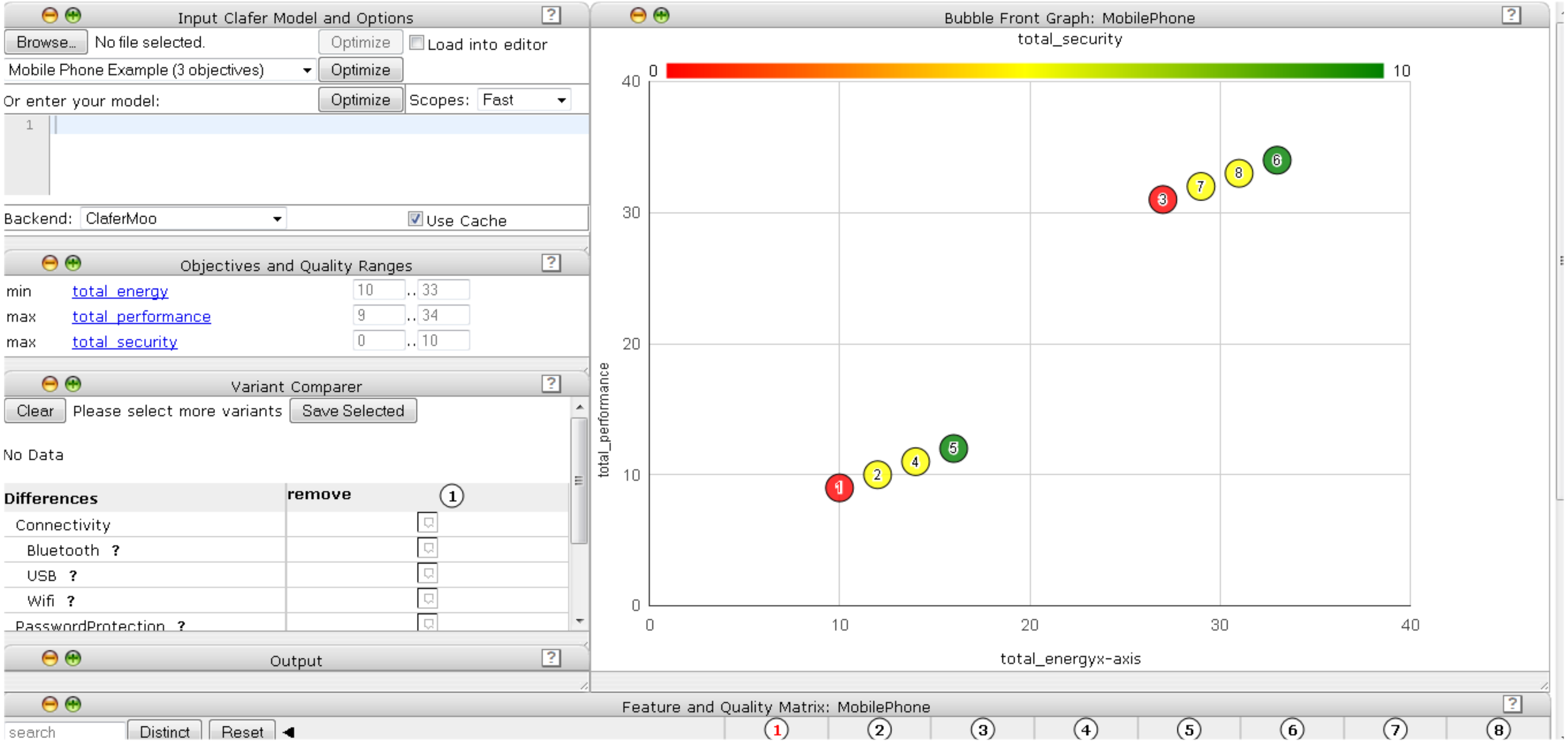
precision

RQ4: How much feature location-recovery cost could be avoided?

subject: Clafer Web Tools

ClaferMooVisualizer, ClaferConfigurator, ClaferIDE

- set of projects that share assets
- developed in JavaScript using clone&own



Input Clafer Model and Options

Browse... No file selected. Optimize □ Load into editor

Mobile Phone Example (3 objectives) Optimize

Or enter your model: Optimize Scopes: Fast

Backend: ClaferMoo ☑ Use Cache

Objectives and Quality Ranges

min	total_energy	10	.. 33
max	total_performance	9	.. 34
max	total_security	0	.. 10

Variant Comparer

Clear Please select more variants Save Selected

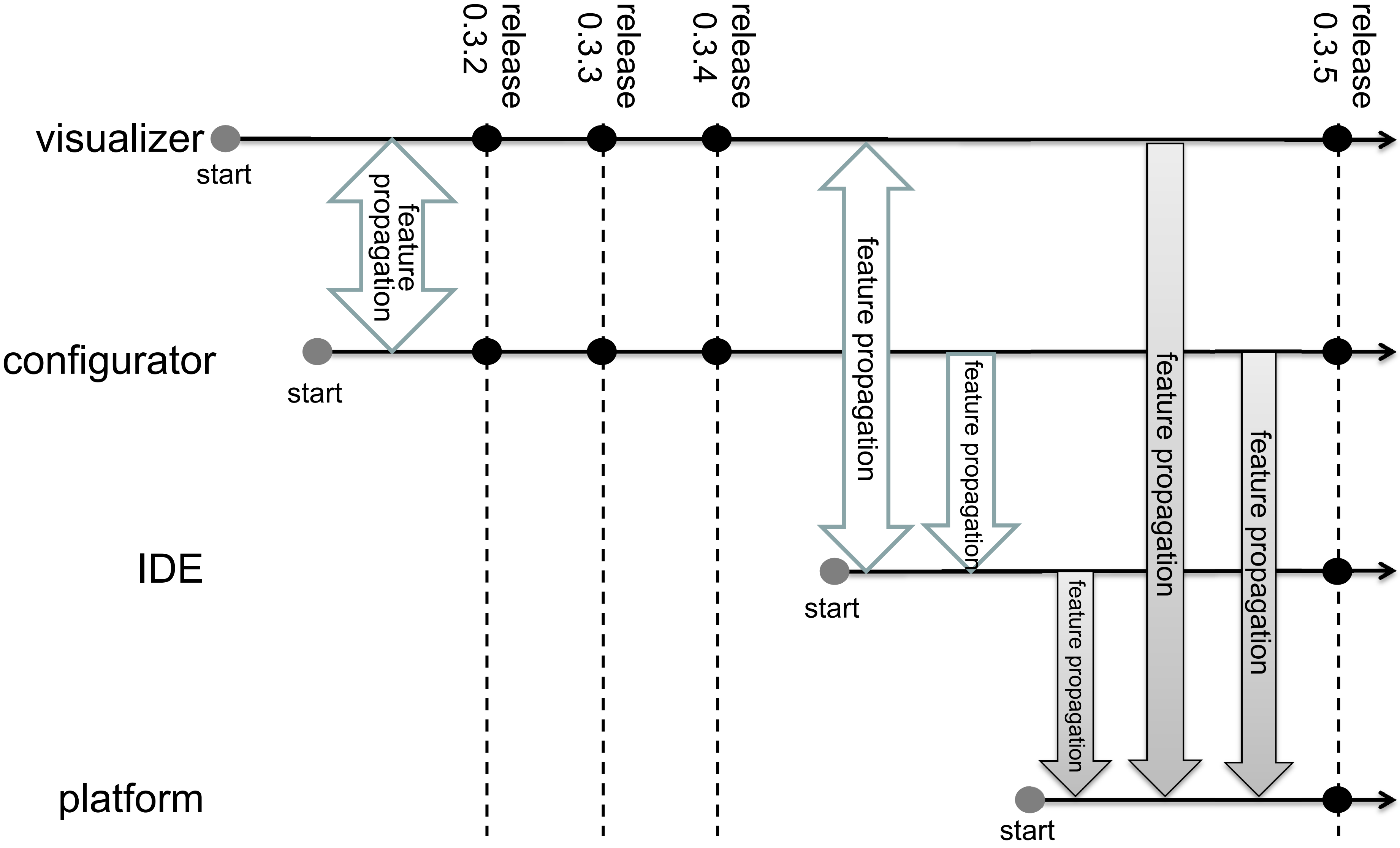
No Data

Differences	remove	1
Connectivity		□
Bluetooth ?		□
USB ?		□
Wifi ?		□
PasswordProtection ?		□

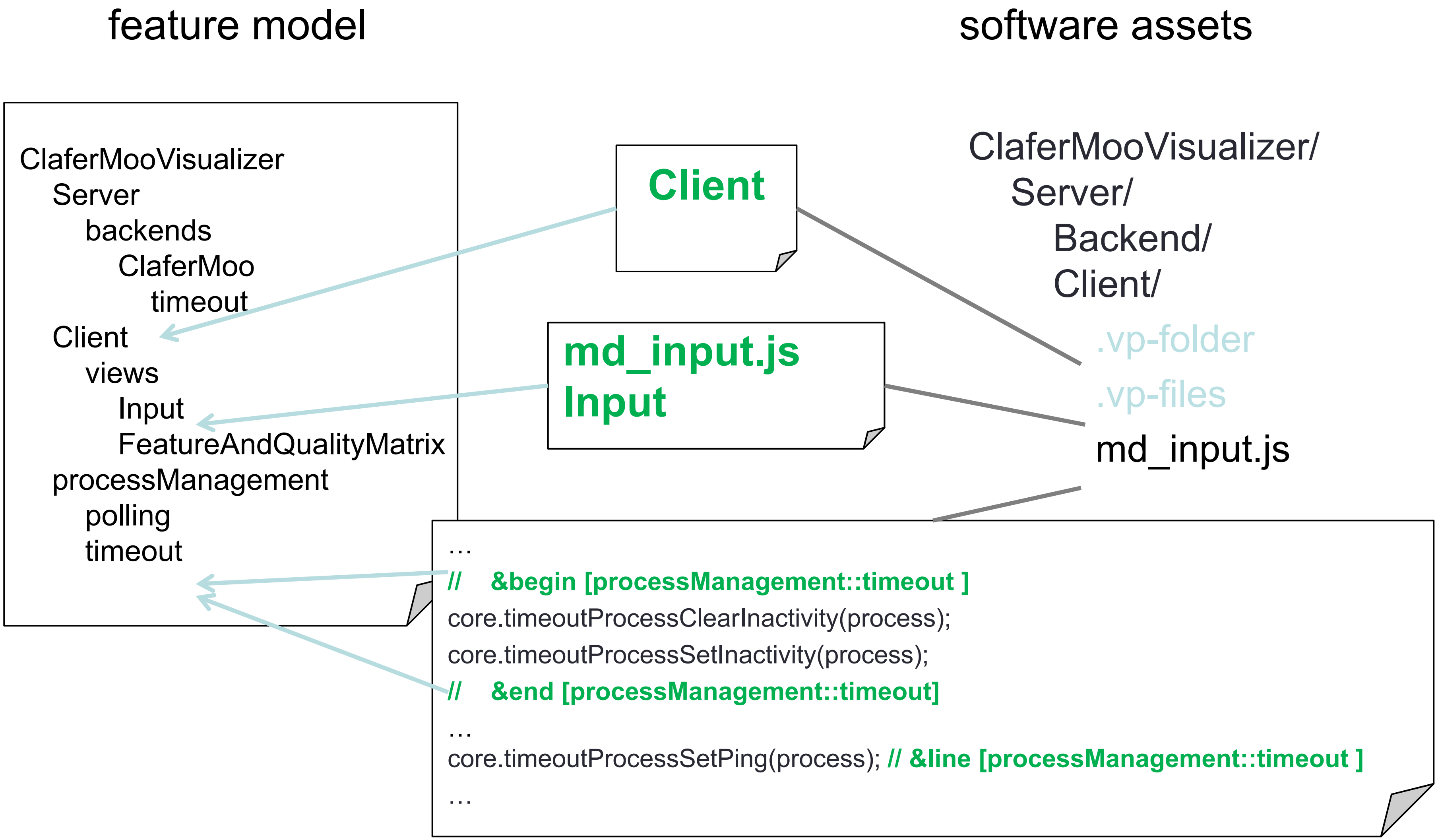
Output

Bubble Front Graph: MobilePhone

Feature and Quality Matrix: MobilePhone


release 0.3.2
release 0.3.3
release 0.3.4
release 0.3.5
visualizer
start
feature propagation
configurator
start
IDE
start
platform
start
feature propagation
feature propagation
feature propagation
feature propagation
feature propagation

embedded annotations



feature model

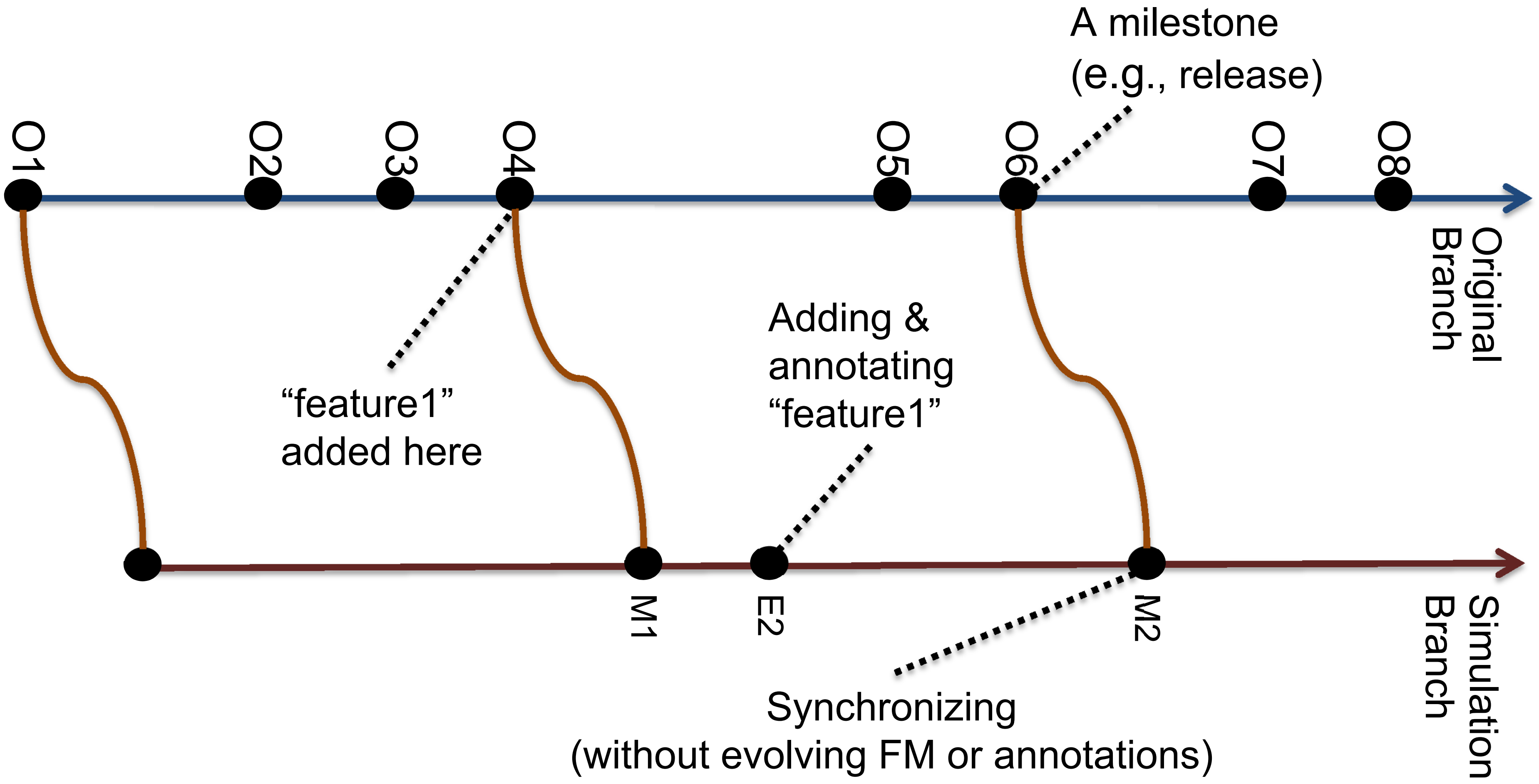
```
ClaferMooVisualizer
    Server
        backends
            ClaferMoo
                timeout
    Client
        views
            Input
            FeatureAndQualityMatrix
    processManagement
        polling
        timeout
```

software assets

```
ClaferMooVisualizer/
    Server/
        Backend/
        Client/
            .vp-folder
            .vp-files
            md_input.js
```

```
…
//    &begin [processManagement::timeout ]
core.timeoutProcessClearInactivity(process);
core.timeoutProcessSetInactivity(process);
//    &end [processManagement::timeout]
…
core.timeoutProcessSetPing(process); // &line [processManagement::timeout ]
…
```

simulation method



O: Original, M: Merge, E: Evolution

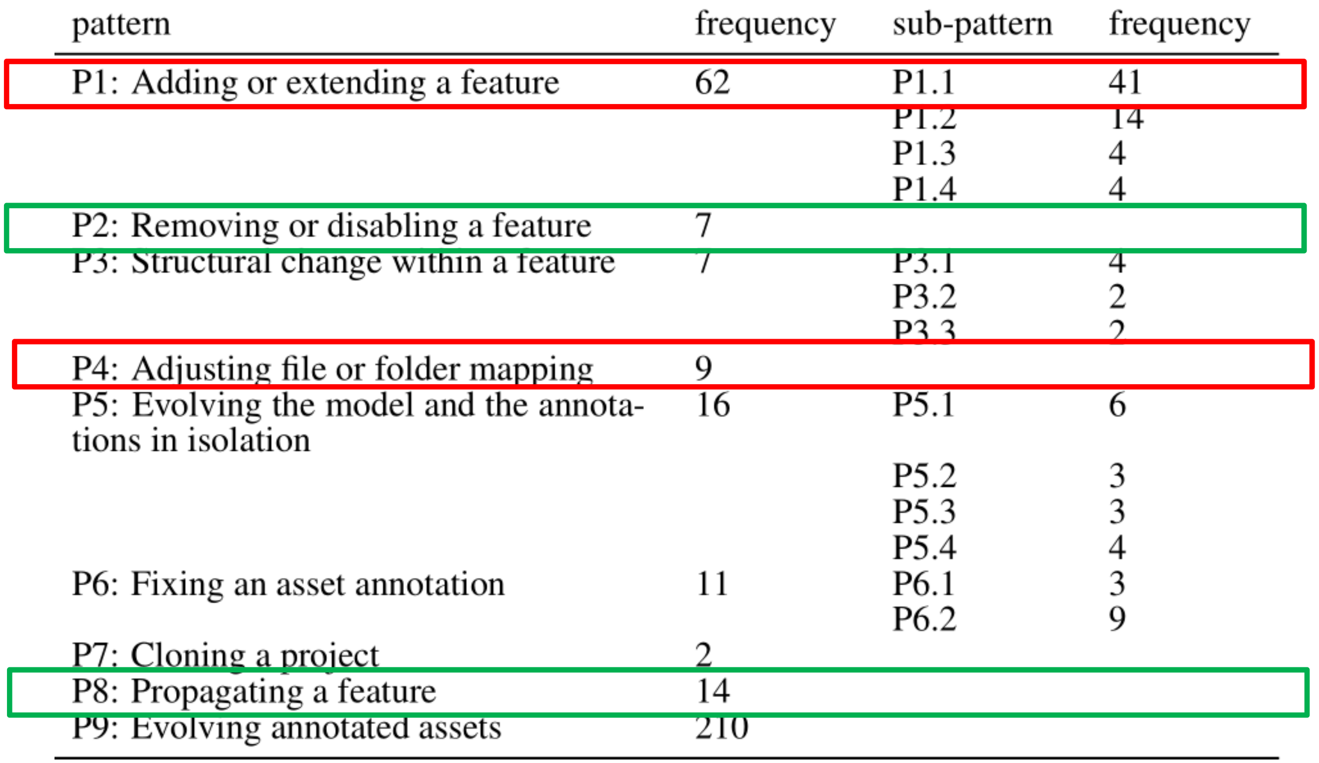
RESULTS

costs (annotation-related activities) and benefits

qualitative results

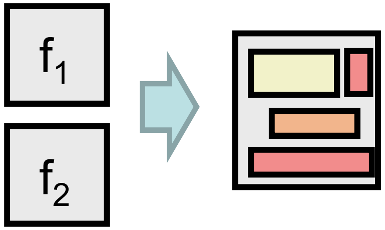
EVOLUTION PATTERNS

evolution patterns



	pattern	frequency	sub-pattern	frequency
cost	P1: Adding or extending a feature	62	P1.1	41
			P1.2	14
			P1.3	4
			P1.4	4
benefit	P2: Removing or disabling a feature	7		
	P3: Structural change within a feature	7	P3.1	4
			P3.2	2
			P3.3	2
cost	P4: Adjusting file or folder mapping	9		
	P5: Evolving the model and the annotations in isolation	16	P5.1	6
			P5.2	3
			P5.3	3
			P5.4	4
	P6: Fixing an asset annotation	11	P6.1	3
			P6.2	9
	P7: Cloning a project	2		
benefit	P8: Propagating a feature	14		
	P9: Evolving annotated assets	210		

P8:



quantitative results (cost model)

COST AND BENEFIT

costs

metric: **annotation markers** (lines) in model and annotations

$C_{pattern}(p_i) = C_{mdl}(p_i) + C_{annot}(p_i)$

RQ1: annotation recording and editing cost

recording cost C_{rec} = 317 lines

editing cost C_{ed} = 339 lines

traceability information naturally co-evolves with assets

RQ2: cost of annotation omissions

arose in three patterns

- identifying a new feature
- fixing missing annotations
- propagating a feature

C_{ao} = 75

feature-location recovery required for 12% of annotation-related activities

benefit

RQ3: annotations that prevented feature-location recovery (recall)

- identified 55 feature propagations (cloned or moved)
- 121 annotation markers matched

18% of annotation recording
and editing costs saved
feature-location recovery costs

RQ4: feature-location recovery costs saved (precision)

- 14 annotations missed (two features)
- 135 annotation markers involved in feature propagations

90% of feature-location
recovery costs saved

break-even point?

18% of the invested costs saved 90% of feature-location recovery costs

break-even point based on actual costs?

investment < benefit

$(C_{rec} + C_{ed}) \cdot \mathbf{AR} < B_{prop} \cdot \mathbf{AL} \cdot (1 - B_{dim})$

AR ... avg. annotation-recording cost
AL ... avg. feature-loc. recovery cost
(per annot.)

we assume **AL** = 10min (based on [Wang et al. ‘13])

then **AR** < 1.85min

clear benefit of
eager embedded annotations
in our case study

summary and future work

good news!

summary

- simple annotation system was surprisingly beneficial
- traceability maintenance reduced effectively
- realistic break-even point

contributions

- evolution patterns showing application of an annotation system
- empirical data on its cost/benefit
- repositories with a history of feature annotations

future work

- realize a feature dashboard
- support annotations with recommender systems
- evaluate in larger industrial setting

thanks for your time!

maintaining feature traceability with embedded annotations

Wenbin Ji, Thorsten Berger, Michal Antkiewicz, Krzysztof Czarnecki