maintaining feature traceability with embedded annotations
adoption of software product lines

clone&own

Project 1
asset
component
library
framework

Project 2
asset
component
library
framework

variant 1

transition

product-line engineering

feature evolution and maintenance

variant 1

Project 1
Config
asset
integrated platform

Project 2
Config
asset
component library
framework

asset
challenge

feature location

...in development teams
Introducing and maintaining features, feature models, and traceability \textbf{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

record and edit traceability during development

lazy strategy

eager strategy
how to store traceability information?

- **external storage**: store in a database
- **internal storage**: embed traceability into assets
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


embedded feature annotations?

eager + internal
SIMULATION CASE STUDY
research questions

RQ1: What are annotation recording/editing costs?

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

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

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

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

...
simulation method

1. Original
2. Branch
3. O2
4. O3
5. O4
6. O5
7. O6
8. O7
9. O8
10. M1
11. E2
12. M2
13. A milestone (e.g., release)
14. "feature1" added here
15. Adding & annotating "feature1"
16. Synchronizing (without evolving FM or annotations)

O: Original, M: Merge, E: Evolution
RESULTS
EVOLUTION PATTERNS

qualitative results

costs (annotation-related activities) and benefits
## evolution patterns

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Frequency</th>
<th>Sub-pattern</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cost</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1: Adding or extending a feature</td>
<td>62</td>
<td>P1.1</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P1.2</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P1.3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P1.4</td>
<td>4</td>
</tr>
<tr>
<td><strong>benefit</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2: Removing or disabling a feature</td>
<td>7</td>
<td>P3.1</td>
<td>4</td>
</tr>
<tr>
<td>P3: Structural change within a feature</td>
<td>7</td>
<td>P3.2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P3.3</td>
<td>2</td>
</tr>
<tr>
<td><strong>cost</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4: Adjusting file or folder mapping</td>
<td>9</td>
<td>P5.1</td>
<td>6</td>
</tr>
<tr>
<td>P5: Evolving the model and the annotations in isolation</td>
<td>16</td>
<td>P5.2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P5.3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P5.4</td>
<td>4</td>
</tr>
<tr>
<td>P6: Fixing an asset annotation</td>
<td>11</td>
<td>P6.1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P6.2</td>
<td>9</td>
</tr>
<tr>
<td><strong>benefit</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P7: Cloning a project</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P8: Propagating a feature</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P9: Evolving annotated assets</td>
<td>210</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**

- P8: `f_1` and `f_2` propagating to a larger structure.
quantitative results (cost model)

COST AND BENEFIT
costs

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

\[ C_{\text{pattern}}(p_i) = C_{\text{mdl}}(p_i) + C_{\text{annot}}(p_i) \]

RQ1: annotation recording and editing cost
- recording cost \( C_{\text{rec}} = 317 \) lines
- editing cost \( C_{\text{ed}} = 339 \) lines

RQ2: cost of annotation omissions
- arose in three patterns
  - identifying a new feature
  - fixing missing annotations
  - propagating a feature
- \( C_{\text{ao}} = 75 \)
benefit

RQ3: annotations that prevented feature-location recovery (recall)
identified 55 feature propagations (cloned or moved)
121 annotation markers matched

RQ4: feature-location recovery costs saved (precision)
14 annotations missed (two features)
135 annotation markers involved in feature propagations
break-even point?

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

break-even point based on actual costs?

investment < benefit

\[( C_{\text{rec}} + C_{\text{ed}} ) \cdot \text{AR} < B_{\text{prop}} \cdot \text{AL} \cdot (1 - B_{\text{dim}}) \]

we assume \( \text{AL} = 10\text{min} \) (based on [Wang et al. ‘13])

then \( \text{AR} < 1.85\text{min} \)

clear benefit of eager embedded annotations in our case study

\( \text{AR} \) ... avg. annotation-recording cost
\( \text{AL} \) ... avg. feature-loc. recovery cost (per annot.)
summary and future work

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

ccontributions
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