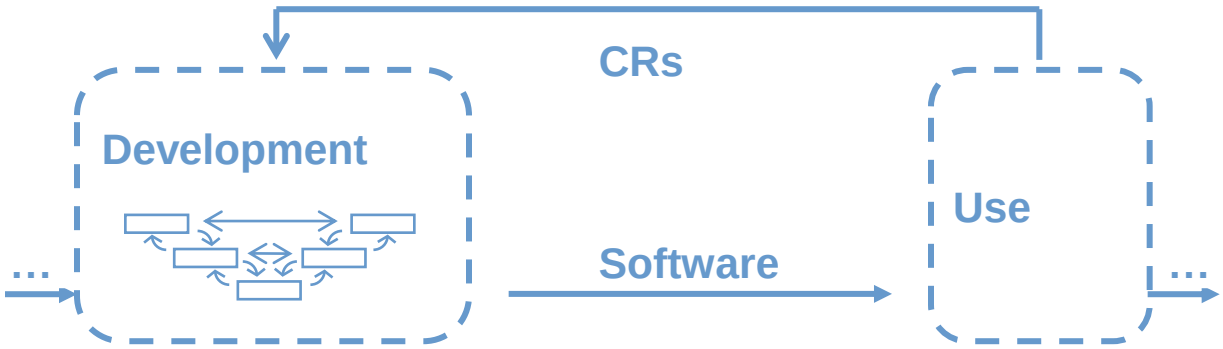
Feature Profiling for Evolving Systems

Elmar Juergens, Martin Feilkas,
Markus Herrmannsdoerfer, Florian Deissenboeck

Munich RE

Rudolf Vaas, Karl-Heinz Prommer

Motivation



- Do I need to fix this code, or it obsolete?
- Which code to test most?
- How many users are impacted if I change this?
- …

This Talk

Feature profiling for accurate, up-to date usage information

Context

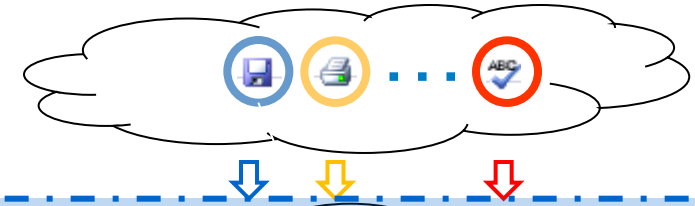
- Business information systems
- Custom Software

Outline

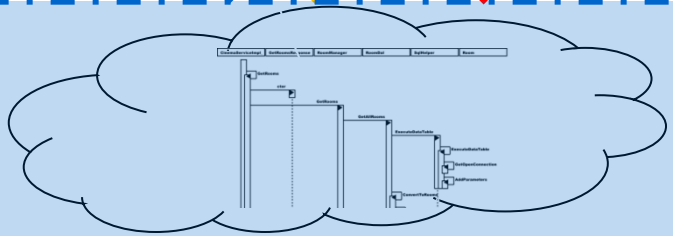
- Feature Profiling Approach
- Industrial Case Study

Feature Profiling Activities

1) Feature Modeling

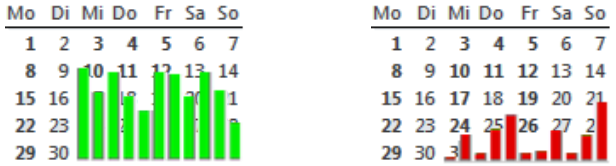


2) Feature Location



3) Execution Monitoring

4) Usage Evaluation



Terms

Feature (Eisenbarth et al. ‘03)

- Realized functional requirement
- User-visible behavior
- Can be triggered by user

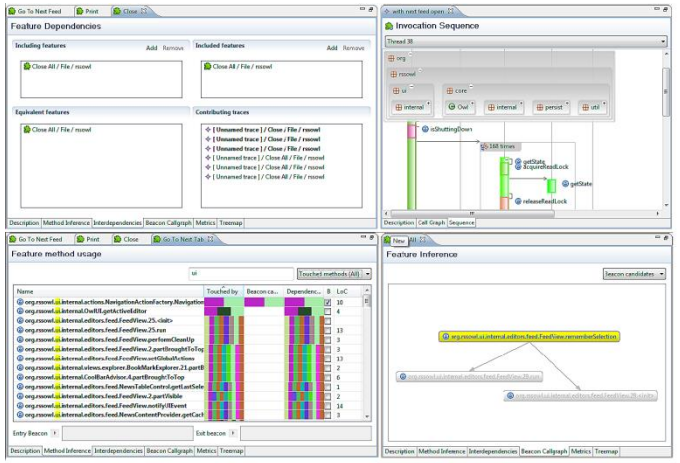
Characteristic Method

- Executed during every feature execution
- Not ever executed for other features

Feature Location

Steps

1) Trace feature executions
2) Mine characteristic methods
3) Manually select feature beacon from characteristic methods



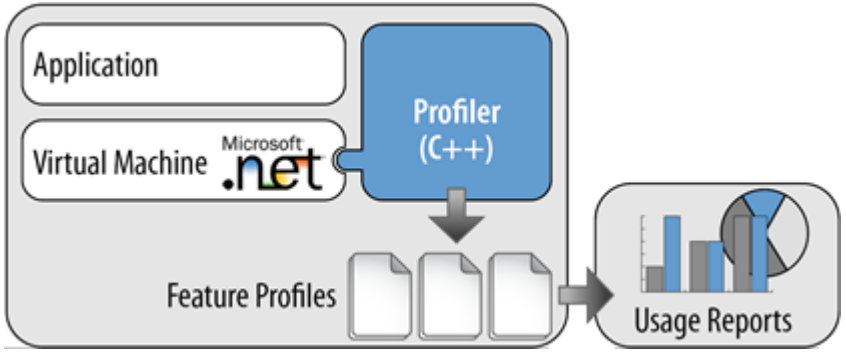
Tradeoffs

+ No changes to application required
- For some features, no characteristic methods might exist

Execution Monitoring

Coverage Profiler

- .NET Profiling interface
- Ephemeral profiling
- Daily method usage reports



Tradeoffs

+ Minimal impact → usable in production
+ Can be disabled without redeployment
– Binary information per feature per day, no invocation counts

Case Study

Research Questions

- Do different engineers have a consistent expectation of feature usage?
- Do actual and expected usage differ?

Study object

- C#, 360 kLOC, client – server application
- 8 years in production, 9-16 engineers, different contractors
- 150 expert users, 10 countries, 4 continents
- Still under active maintenance

Usage Expectation Consistency

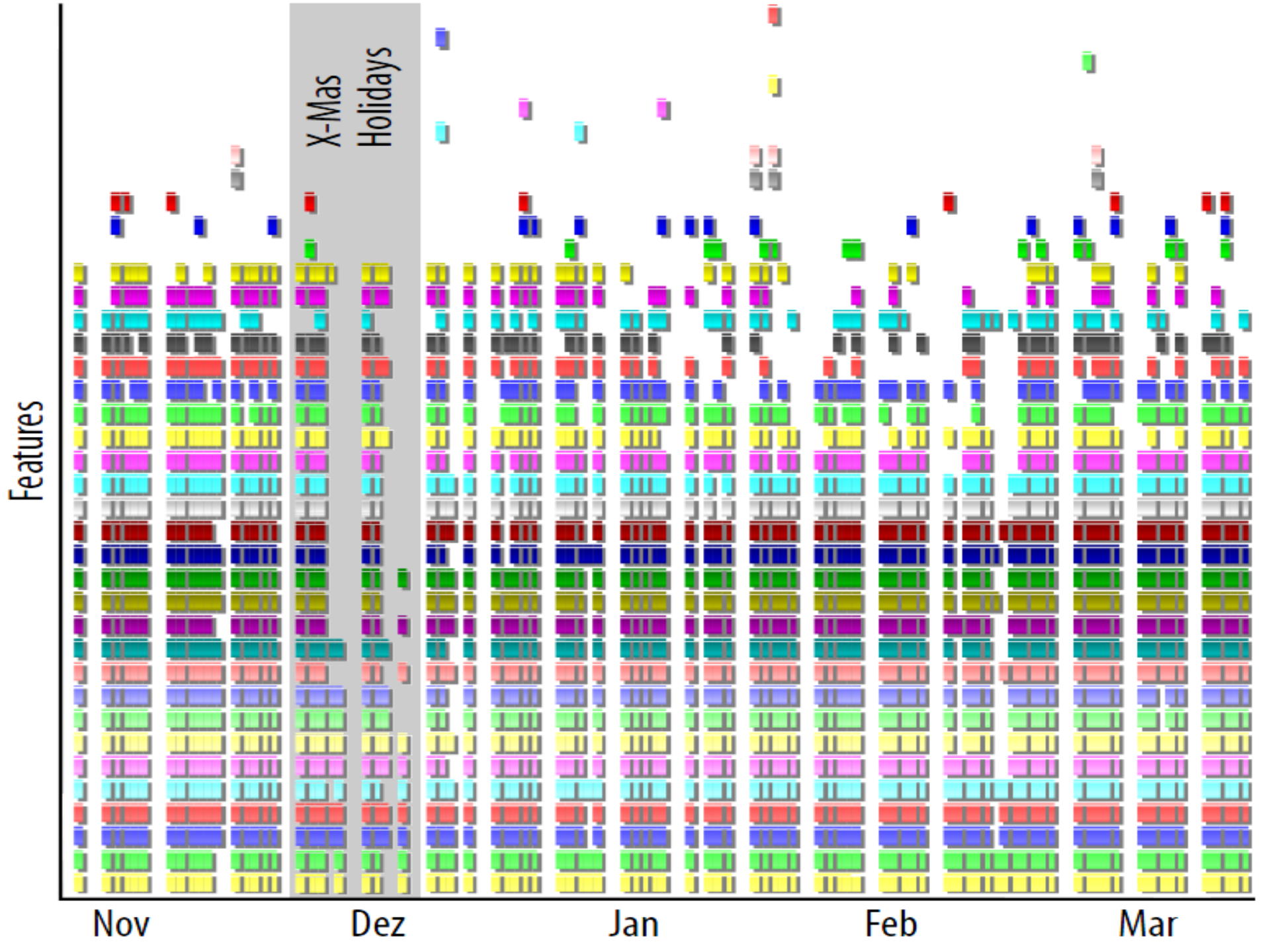
Design & Execution

1) Interviewed product manager, internal developer, external developer
2) Selected expected usage for 76 features from scale:

   1d, 2d, 3d, 4d, 5d 2w, 3w, 1m 2m, 3m, 6m 9m, 1y, 2y
3) Cohen's Kappa to determine agreement (0 ~ none, 1 ~ perfect)

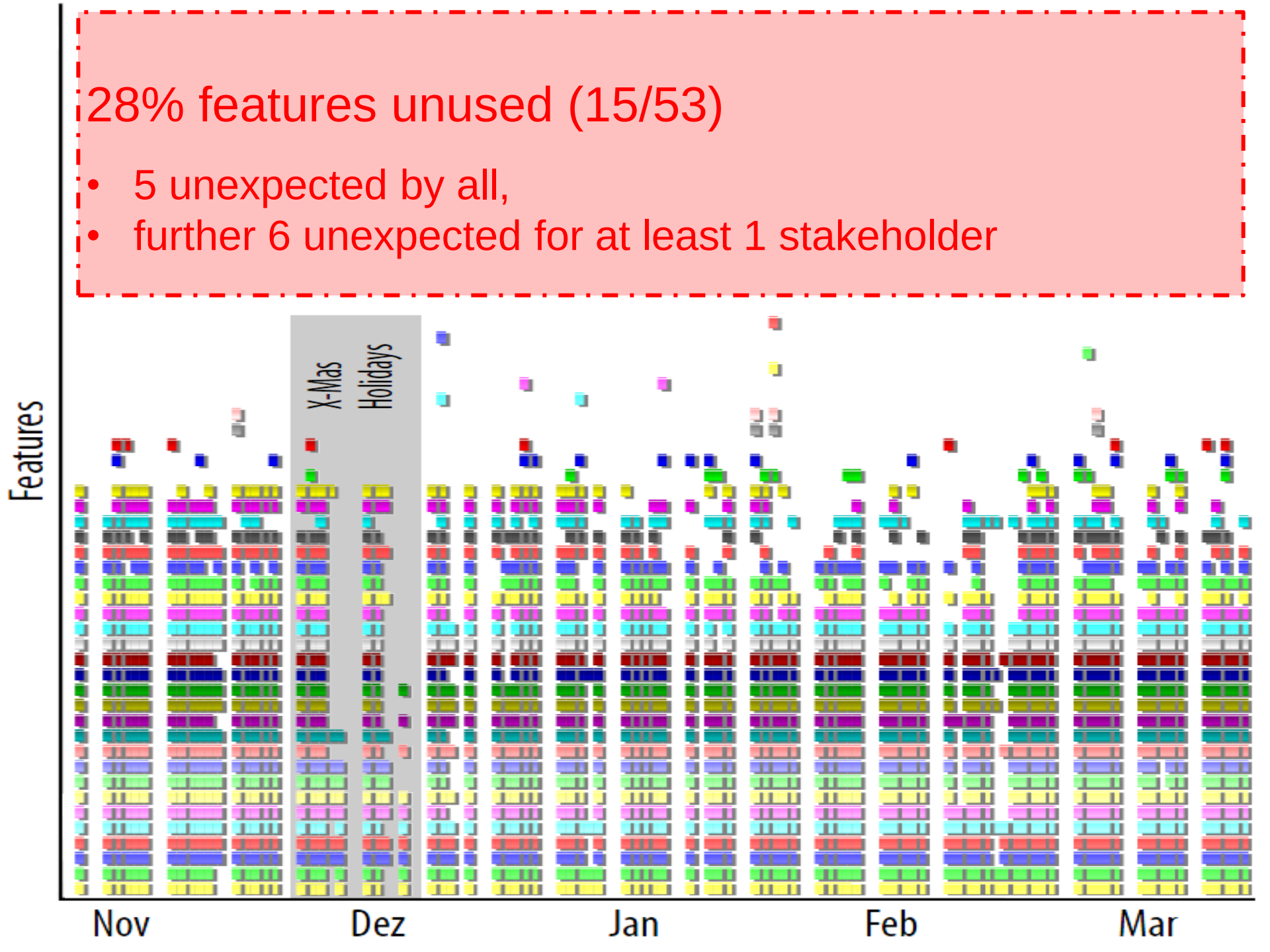
Results

- 10% (8/76) rated identical; Kappa 0,21 → slight agreement
- Aggregated: Kappa 0,59 → still no strong agreement


X-Mas
Holidays
Features
Nov
Dez
Jan
Feb
Mar

28% features unused (15/53)

- 5 unexpected by all,
- further 6 unexpected for at least 1 stakeholder



Actual vs. Expected Usage

Computation of deviation

- Computed longest interval ***i*** without use
- Deviation, iff ***i < ½ e*** or ***i > 2e***

Results

- Product manager (internal) 43%, Internal developer 40%
- External developer 55%
- Majority: system used ***less*** than expected

No stakeholder had accurate expectation of actual usage

Participants found results helpful to improve maintenance

Threats to Validity

	Threat	Mitigation
Construct	• Threshold for deviation computation debatable	• Consistent with intuition of participants. Absolute values unclear though (40% vs 43%?)
Internal	• Not all features profiled	• No systematic error, results representative for system
	• Ephemeral profiling	• Future work: Usage counts of feature beacons
External	• Only a single system measured	• Generalizability unclear – future work on other systems required

Conclusions

Feature profiling: dynamic analysis for accurate, up-to-date usage data

Industrial case study demonstrated that it is

- Feasible in production
- We cannot expect usage expectation to be consistent or correct

Gained information considered helpful by software engineers

Future Work

- More systems: Already running on 2nd; 3rd under way
- Make features explicit in code during forward engineering
- Make feature profiles easily accessible by engineers