The Challenge

- Large programs contain functionality that resists clean modularisation, also referred to as cross-cutting concern (CCC)
- Typical examples include logging, debugging, and resource management (lock/unlock)
- Such scattered functionality is a weakness of current systems: it makes them hard to maintain, extend or change
- Aspect-oriented programming (AOP) tries to remedy that by factoring this out into code-entities called aspects
- For existing code to benefit from AOP, aspects have to be identified first, a task also called aspect mining

The Idea – Step 1

Analysis of Version Archives

- Analyse CVS transactions for code additions
- For each added method call in a code location make a cross in a table labelled with methods/locations

The Idea – Step 2

Formal Concept Analysis

- Find maximal blocks in a table as they represent CCCs or so-called aspect candidates
- Each block is also a concept in a lattice, which can be computed efficiently using formal concept analysis

Results

Mining Results and Statistics (Eclipse 3.2M3)

- High precision
- Scalability to industrial-sized systems
- Fast and efficient (on average 1 msec per transaction)

What’s next?

- Tool support: build Eclipse plugin
- Deployment by programmers: is technique useful?
- Extension to find refactorings: include code deletions
- Automatic tracking of (evolution of) CCCs: avoid re-mining after code changes
- Understanding concerns over time: how do they evolve?

And What Do You Think?

Your opinion counts! Feel free to give your feedback below, or leave your email address and I’ll get back to you.

Status (1 year into PhD)

- Demo & Prototype "HAM" (History-Based Aspect Mining)
- Publications
  - Mining Eclipse for Cross-Cutting Concerns. S. Breu, T. Zimmermann, and C. Lindig. 3rd International Workshop on Mining Software Repositories at ICSE (MSR).