

# *Preliminary Research Goal Statement*

To develop and/or improve mechanisms that aid in the extraction of unnecessary complexities involved in the development and maintenance of software.



# *Some Motivation*

Technology changes at a rapid pace. Old technologies fade, new technologies emerge, and existing technologies change. Software development methodologies that were once thought of as state of-the-art are now obsolete and new methodologies are considered industry standards.

---

---

## *Some Motivation*

Hence, it may be a difficult task for software developers and architects to manually keep abreast of new technologies and methodologies. Even if a developer is abreast of a new technology it is not positive that the developer will have adequate knowledge to apply the technology correctly. Questions remain open as to where and how to apply the new technology and to which situations, etc.

---

---

# Research Interests

- (Semi-)Automatic refactoring (in general)
    - “*A lot for very little*”
    - Cheaper to renovate than rebuild.
    - Guaranteeing system integrity.
  - Aspect mining/discovery of legacy systems.
    - Improving the accuracy of aspect mining techniques through formal specifications.
      - Pattern contracts/subcontracts?
    - Lexigraphical analysis of functional requirements documentation to identify crosscutting system functionality.
- 
-

# Research Interests

- Usage of aspects in regards to:
    - Persistence
    - Distribution
  - (Modular) Reasoning of aspect oriented programs
  - Aspect-oriented frameworks
  - Aspect polymorphism
    - (as a language mechanism)
  - Automated OO persistence and transaction management using garbage collection algorithms
- 
-

# Motivation of Aspect Mining of Legacy Systems

- Crosscutting concerns cannot be well modularized in object oriented software due to its inherent limitations.
    - Result is poor program comprehension.
  - Patterns attempt to separate concerns (i.e., *roles*)
    - can not completely do so.
  - Systems even with a moderate level of complexity must deal with crosscutting concerns.
    - Code is thus scattered/tangled.
    - System is difficult to explore, understand, maintain, and reuse.
- 
-

# *Motivation of Aspect Mining of Legacy Systems*

A system that is better modularized in which concerns are neatly separated is more likely to contain components that can be extracted for reuse in other systems.

---

---

# *Goal of Aspect Mining of Legacy Systems*

The aim is to migrate these systems to AOP and to do so “automatically.”



# Realization of Aspect Mining

- Source mining of legacy (non-AOP) systems.
    - Not clear if the process can be completely automated.
    - Some existing techniques to discover tangled and/or scattered code that may (should) be transformed into aspects:
      - Static Analysis
        - Fan-in
        - Identifier
      - Dynamic Analysis
        - Pattern matching in execution traces
    - All techniques attempt to *guess* if code implements a cross-cutting concern.
      - Human intervention may be required for refinement.
    - None utilize system specifications.
- 
-