FIRST-CLASS PROGRAM TRANSFORMATIONS

Danny Dig
Programming as Done Today

Change is the heart of software development:
- add features, fix bugs, support new hardware/UI/OS

Today’s program development is very crude
- change carried manually, through low-level edits
- changes are almost never reused
- versioning tools focus on changes to lines of code

Change is too ad-hoc making software development error-prone, time-consuming, and $$$
- 2/3\textsuperscript{rd} of software costs due to software evolution, some industrial surveys claiming 90%
My View of Tomorrow: Programming is Program Transformation

Change needs to move to a higher-level of abstraction

Program transformations as first-class:
- most changes carried through automated program transformations
- even manual edits become transformations
- programs as sequence of program transformations

Q1: **Analyze** what software changes occur in practice?
Q2: How can we **automate** them?
Q3: Can we **represent** programs as transformations? **Archive**, **retrieve**, and **visualize** them?
Q4: Can we **infer** higher-level transformations?
Overview of My Research Results

Program Transformations (Refactorings)

- **Inferring**
  - ECOOP13, ASE09a
  - ECOOP06

- **Archiving & Retrieving**
  - TSE08, ICSE07

- **Testing**
  - ICST13, TSE10, ASE09b, ISSTA08, FSE07

- **Automating**
  - ICST13, ICSM12, ICSE11, SOFTW11, OOPSLA09, ICSE09, ICSE08

- **Understanding**
  - ECOOP12, FSE12, IWMSE11, JSME06, ICSM05
# Our Refactorings for Parallelism

## Refactorings for thread-safety
- make class immutable [ICSE'11]
- convert to Atomic* classes [ICSE'09]
- use concurrent collections [ICSE'09]
- infer region annotations [ASE'09]
- atomic check-then-act operations [ICST’13]

## Refactorings for throughput
- parallel recursive divide-and-conquer [ICSE'09]
- loop parallelism via ParallelArray [OOPSLA’10:demo]
- loop parallelism via lambda-enabled functional operators [Submitted]

## Refactorings for scalability
- Atomic*, concurrent collections [ICSE'09]
Practical Impact of My Research

Program Transformations (Refactoring)

Inferring
- used at Google
- dozen labs

Automating
- ship
- ship/ongoing

Understanding
- influenced Java and .NET official concurrency libraries
- learnparallelism.net
30,000 visitors in 6 mo

Testing
- testing infra at Oracle

founded Workshop on Refactoring Tools, HotSWUp

first open-source refactoring
17,000 downloads
Change-Oriented Programming Environment (COPE)

Inferring
- unknown, frequent, online
- static & dynamic

Composing

Algebra:
- commutativity
- distributivity
- conflict
- dependency

Automating
- Scripting custom transformations

Understanding
- perceive, recall, communicate change

Archiving/Retrieving
integrate with version control

Recently funded by NSF

SHF:Large
Interactive Refactoring for Parallelism on Mobile/Web Applications

New converging forces that reshape computing
- end users spend most time on mobile and web apps
- mobile devices are all going multicore

What are the new transformations we need to automate?
- inspiration from explorative studies

Examples of transformations:
- adding parallelism in JavaScript apps via ParallelArray
- candidate programs with trade-offs between performance & power consumption
- adaptation to different display technologies
- split functionality between the device and cloud
Acknowledgements: Funding $$$

NSF

Microsoft

IBM

Intel

BOEING

ILLINOIS
Science and Tools for Change

“Change is the only guaranteed constant”

To foster a revolution in software technology, we need to raise the level of abstraction for changes

Interactive, automated transformation more effective than manual

Many of our tools ship with official release

Today's brand new programs are tomorrow's legacy programs
- continue to teach students how to change large code
- software evolution becomes the primary paradigm of software development