ECE444: Software Engineering
Software Engineering Research
Shurui Zhou
(Competing) concerns in SE...

- **Code**: faster, cheaper, more features, more reliable/secure
- **Developers**: more productive, more skilled, happier, better connected
- **Organizations/communities**: attract/retain contributors, encourage a participatory culture, increase value
“Measuring programming progress by lines of code is like measuring aircraft building progress by weight.”
Contributing graphs considered harmful (Hanselman)

https://www.hanselman.com/
Software Engineering **Design** Space
Research success?
THEORY

PRACTICE
Success practice transfer stories from research

• Automated testing (Facebook)
• Code review tools (Microsoft)
• Software Analytics (Hassan et al.)
DeepCode from ETH

STEP 1: PARSING
This is the only language-specific part of our platform and enables us to add support for any/just a language in a matter of weeks.

STEP 2: SOLVERS
(DECLARATIVE STATIC ANALYSIS)
Our custom language-independent linear-complexity Datalog solvers allow us to analyze huge repositories in a matter of seconds.

STEP 3: ML ALGORITHMS
Our custom Semantic Facts representations allows us to run powerful ML algorithms to understand the structure, function, and intent of the code.
kite from MIT & Stanford


Kite adds AI powered code completions to your code editor, giving developers superpowers.

```python
1 import os
2 import sys
3
4 def count_py_files_in_repos(dirname):
5     #
```
1968 NATO Conference on Software Engineering

- international experts on computer software who agreed on defining best practices for software grounded in the application of engineering.
“Academic software engineering research has been a backwater primarily staffed by those interested in theory, with a tenuous connection to practical software development.”

- Lack of **industrial relevance** (doesn’t scale or solve industry problems) [Briand]
- **Poor replication** of software engineering studies [Menzies et al.]
- **Poor actionability** (practitioners know which modules are buggy...)
- **Perils of mining** software repositories [Kaliamvakou, German et al.]
- Lack of focus on **human/social aspects** [Storey et al.]
What metrics are the best predictors of failures?

If I increase test coverage, will that actually increase software quality?

What is the data quality level used in empirical studies and how much does it actually matter?

Are there any metrics that are indicators of failures in both Open Source and Commercial domains?

I just submitted a bug report. Will it be fixed?

Should I be writing unit tests in my software project?

How can I tell if a piece of software will have vulnerabilities?

Do cross-cutting concerns cause defects?

Is strong code ownership good or bad for software quality?

Does Test Driven Development (TDD) produce better code in shorter time?

Does Distributed/Global software development affect quality?
Mining Software Repository
SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews

The GHTorrent project
Software Documentation Issues Unveiled
Beyond the Code: Mining Self-Admitted Technical Debt in Issue Tracker Systems

Laerte Xavier
ASERG Group - Department of Computer Science
Federal University of Minas Gerais (UFMG)
Belo Horizonte, Brazil
laertexavier@dcc.ufmg.br

Fabio Ferreira
Center of Informatics - Federal Institute of the Southeast of Minas Gerais
Barbacena, Brazil
fabio.ferreira@ifsudestmg.edu.br

Rodrigo Brito
ASERG Group - Department of Computer Science

Marco Tulio Valente
ASERG Group - Department of Computer Science

Refactor :comment_personal_snippet to :create_note

Follow-up to https://dev.gitlab.org/gitlab/gitlabhq/merge_requests/2794

We probably keep this confidential until that MR is merged to master and issue is made public.

The :create_note permission is being checked on the Notable when replying to email notifications. The previous MR adds the :create_note permission to ProjectSnippetPolicy.

This is a duplicate of an existing :comment_personal_snippet permission. We should refactor uses of :comment_personal_snippet to use the common :create_note permission instead.
Requirement
Can a Conversation Paint a Picture?  
Mining Requirements in Software Forums

James Tizard, Hechen Wang, Lydia Yohannes, Kelly Blincoe  
University of Auckland, New Zealand  
{jtiz003,hwan531}@aucklanduni.ac.nz, l.yohannes@web.de, k.blincoe@auckland.ac.nz

<table>
<thead>
<tr>
<th>ARdoc Classes</th>
<th>Mapped Forum Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Discovery</td>
<td>Apparent bug</td>
</tr>
<tr>
<td>Feature Request</td>
<td>Feature request</td>
</tr>
<tr>
<td>Information Seeking</td>
<td>Question on application</td>
</tr>
<tr>
<td></td>
<td>Help seeking</td>
</tr>
<tr>
<td></td>
<td>Requesting more information</td>
</tr>
<tr>
<td></td>
<td>Question on background</td>
</tr>
<tr>
<td>Information Giving</td>
<td>Application guidance</td>
</tr>
<tr>
<td></td>
<td>User setup</td>
</tr>
<tr>
<td></td>
<td>Praise for application</td>
</tr>
<tr>
<td></td>
<td>Dispraise for application</td>
</tr>
<tr>
<td></td>
<td>Application usage</td>
</tr>
<tr>
<td></td>
<td>Attempted solution</td>
</tr>
<tr>
<td></td>
<td>Acknowledgement of resolution</td>
</tr>
</tbody>
</table>
Detecting Bad Smells in Use Case Descriptions

1. Overview
   Actor withdraws money from a bank account via bank ATM.

2. Event flow
   2.1 Precondition
      Actor has a bank account and an ATM card.
   2.2 Postcondition
      Specified amount of money is withdrawn from the actor's account.
   2.3 Basic flow
      1. System displays the message "Enter your PIN number" on the ATM screen.
      2. Actor inserts a PIN number.
      3. Actor enters a PIN number.
      4. System checks the PIN number and displays the message
         "Enter the amount of money" on the ATM screen.
      5. Actor enters the amount of money that is being withdrawn.
      6. System checks the balance of the account, removes the withdrawn amount
         from the balance and puts the withdrawn amount
         to the output port of the ATM.
      7. Actor takes the money from the output port.
      8. System prints the details and puts together with the ATM card.
      9. Actor takes the.
     10. The use case finishes.

2.4 Alternate flows
   A1 The case of inserting a bankbook (at Basic flow 1)
      A1.1 Actor inserts a bankbook to the ATM.
      A1.2 Return to Basic flow 2.
   A2 The case of inserting a bankbook (at Basic flow 8)
      A2.1 System enters transaction amount to the bankbook.
      A2.2 Return to Basic flows 8.
   A3 The case of mistaking a PIN number (at Basic flow 4)
      A3.1 System displays the message "PIN number is mistaken" on the ATM screen.
      A3.2 Return to Basic flow 3.

2.5 Exception flow
   E1 The case of mistaking a PIN number at three times (at Basic flow 4)
      E1.1 System displays the message "As you mistook the PIN number
         at three times, please contact the person in charge" on the ATM screen.
      E1.2 Finish the use case.
   E2 The case of insufficient balance (at Basic flow 6)
      E2.1 System displays the message "Balance is insufficient" on the ATM screen.
      E2.2 Return to Basic flow 8.

Fig. 6. Example result of the smell detection.
# NERO: A Text-based Tool for Content Annotation and Detection of Smells in Feature Requests

Fangwen Mu\(^{1,2}\), Lin Shi\(^{1,2*}\), Wei Zhou\(^1\), Yuanzhong Zhang\(^1\), Huixia Zhao\(^1\)

## TABLE II
EXPLANATIONS AND DETECTION METHODS OF SMELLS

<table>
<thead>
<tr>
<th>Smell Category</th>
<th>Smell Name</th>
<th>Explanation</th>
<th>Detection method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ambiguous</strong></td>
<td>Vagueness</td>
<td>Vagueness occurs whenever a statement admits borderline cases, such as appropriate, clear, significant, etc.</td>
<td>Keyword glossary, Lemmatization</td>
</tr>
<tr>
<td></td>
<td>Weakness</td>
<td>Weakness occurs when the feature requests use words with weak semantic content and little emotional color, such as could, may, might, etc.</td>
<td>Keyword glossary, Lemmatization</td>
</tr>
<tr>
<td></td>
<td>Generality</td>
<td>Generality occurs when the sentence contains words that identify a certain type of object, and no modifiers limit its scope, such as flow, access, data, interface, etc.</td>
<td>Keyword glossary, Lemmatization, Dependency parsing</td>
</tr>
<tr>
<td></td>
<td>Coordination ambiguity</td>
<td>Coordination ambiguity occurs when the use of coordinating conjunctions leads to multiple potential interpretations of a sentence.</td>
<td>POS tagging, Regular expression</td>
</tr>
<tr>
<td></td>
<td>Referential ambiguity</td>
<td>Referential ambiguity occurs when an anaphor (e.g. it, that, which, etc.) can take its reference from more than one element, each playing the role of the antecedent.</td>
<td>POS tagging, Regular expression</td>
</tr>
<tr>
<td></td>
<td>Passive voice</td>
<td>Passive voice occurs when the passive voice is used in the feature requests.</td>
<td>Dependency parsing, Regular expression</td>
</tr>
<tr>
<td><strong>Incomplete</strong></td>
<td>Missing condition</td>
<td>Missing condition occurs when the sentence contains an if clause expressing the condition, but there is no corresponding else/otherwise clause.</td>
<td>Keyword glossary, Lemmatization</td>
</tr>
<tr>
<td></td>
<td>Missing description</td>
<td>Missing description occurs when the sentence contains omitted-meaning words, such as as defined, to be completed, to be determined, etc.</td>
<td>Regular expression</td>
</tr>
<tr>
<td><strong>Unintelligible</strong></td>
<td>Unreadability</td>
<td>Unreadability occurs when the sentences in one feature request are too long or not smooth.</td>
<td>GPT2 LM, Coleman-Liau formula</td>
</tr>
<tr>
<td></td>
<td>Partial Content</td>
<td>Partial Content occurs when the feature requests lack any of the five semantic annotations (except Trivia) mentioned in the content annotation. We assume that feature requests with more different content annotations will deliver more diverse information.</td>
<td>Weighted analysis</td>
</tr>
</tbody>
</table>
NERO: A Text-based Tool for Content Annotation and Detection of Smells in Feature Requests

Fangwen Mu\textsuperscript{1,2}, Lin Shi\textsuperscript{1,2*}, Wei Zhou\textsuperscript{1}, Yuanzhong Zhang\textsuperscript{1}, Huixia Zhao\textsuperscript{1}
\textsuperscript{1}Laboratory for Internet Software Technologies, Institute of Software Chinese Academy of Sciences, Beijing, China.
\textsuperscript{2}University of Chinese Academy of Sciences, Beijing, China.
{fangwen, zhouwei, yuanzhong, huixia}@itechs.iscas.ac.cn, {shilin}@iscas.ac.cn

Fig. 1. The Overview of NERO
Documentation
On Automatically Generating Commit Messages via Summarization of Source Code Changes
Luis Fernando Cortés-Coy\textsuperscript{1}, Mario Linares-Vásquez\textsuperscript{2}, Jairo Aponte\textsuperscript{1}, Denys Poshyvanyk\textsuperscript{2}

Automatic Documentation Generation via Source Code Summarization of Method Context
Paul W. McBurney and Collin McMillan
AREN\textsuperscript{A}: An Approach for the Automated Generation of Release Notes
Laura Moreno, Member, IEEE, Gabriele Bavota, Member, IEEE, Massimiliano Di Penta, Member, IEEE, Rocco Oliveto, Member, IEEE, Andrian Marcus, Member, IEEE, and Gerardo Canfora
Traceability
A Novel Approach to Tracing Safety Requirements and State-Based Design Models

Mounifah Alenazi*
University of Cincinnati
Cincinnati, Ohio
alenazmh@mail.uc.edu

Nan Niu
University of Cincinnati
Cincinnati, Ohio
nan.niu@uc.edu

Juha Savolainen
Danfoss Drives A/S
Gråsten, Denmark
juha.savolainen@danfoss.com

Establishing Multilevel Test-to-Code Traceability Links

Robert White
University College London
London, UK

Jens Krinke
University College London
London, UK

Raymond Tan
University College London
London, UK

Improving the Effectiveness of Traceability Link Recovery using Hierarchical Bayesian Networks

Kevin Moran
William & Mary
Williamsburg, VA, USA
kpmoran@cs.wm.edu

David N. Palacio
William & Mary
Williamsburg, VA, USA
dnaderp@cs.wm.edu

Carlos Bernal-Cárdenas
William & Mary
Williamsburg, VA, USA
cebernal@cs.wm.edu

Daniel McCrystal
William & Mary
Williamsburg, VA, USA
dmc@cs.wm.edu
Code Review
Mitigating Turnover with Code Review Recommendation: Balancing Expertise, Workload, and Knowledge Distribution

Ehsan Mirsaeeedi  
Department of Computer Science and Software Engineering  
Concordia University, Montréal, Québec, Canada  
s_irsaeed@encs.concordia.ca

Peter C. Rigby  
Department of Computer Science and Software Engineering  
Concordia University, Montréal, Québec, Canada  
peter.rigby@concordia.ca

Who Should Review My Code?  
A File Location-Based Code-Reviewer Recommendation Approach for Modern Code Review

Patanamon Thongtanunam*, Chakkrit Tantithamthavorn*, Raula Gaikovina Kula†, Norihiro Yoshida‡, Hajimu Iida*, Ken-ichi Matsumoto*  
*Nara Institute of Science and Technology, †Osaka University, ‡Nagoya University, Japan  
{patanamon-t, chakkrit-t, matumoto}@is.naist.jp, iida@itc.naist.jp, raula-k@ist.osaka-u.ac.jp, yoshida@erl.t.u
Helping Developers Help Themselves: Automatic Decomposition of Code Review Changesets
Deployment
FastLane: Test Minimization for Rapidly Deployed Large-scale Online Services

Adithya Abraham Philip, Ranjita Bhagwan, Rahul Kumar, Chandra Sekhar Maddila and Nachiappan Nagappan
Microsoft Research

Fig. 1. The integrated algorithm and FastLane test prediction flow. This includes all three types of predictions FastLane performs (green boxes): 1. commit risk prediction, 2. test outcome-based correlation, and 3. runtime-based outcome prediction. The blue box captures the learning functionality used to continuously train FastLane.
Productivity
Characterizing Software Developers by Perceptions of Productivity

Fig. 1: Comparing the clusters with respect to words that developers associate with *productive* (left, Q1) and *unproductive* work days (right, Q2). Terms in *turquoise* are related to Cluster 1, *orange* to Cluster 2, *purple* to Cluster 3, *pink* to Cluster 4, *green* to Cluster 5, and *gold* to Cluster 6. The size of a term corresponds to the difference between the maximum relative frequency and the average relative frequency of the word across the six clusters.
Do Developers Discover New Tools On The Toilet?

To increase awareness and adoption of software tools and practices, Google uses a technique called “Testing on the Toilet”, or TotT for short.

Evaluation of the effectiveness of TotT

Hypothesis: Testing on the Toilet increases usage of advertised developer tools.

Case Study: CausalImpact, a Bayesian statistical technique that was developed to evaluate the impact of advertising on website traffic.
FLOSS Participants’ Perceptions about Gender and Inclusiveness: A Survey

Investigating the Effects of Gender Bias on GitHub

Nasif Intiaz, Justin Middleton, Joymallya Chakraborty, Neill Robson, Gina Bai, and Emerson Murphy-Hill

1Department of Computer Science, North Carolina State University
2Google, LLC

{siintiaz, jomiddl, jschakr, plrobnso, gbai2}@ncsu.edu, emersonm@google.com

Engineering Gender-Inclusivity into Software: Tales from the Trenches

Claudia Hilderbrand, Christopher Perdriau, Lara Letaw, Jillian Emard, Zoe Steine-Hanson, Margaret Burnett, Anita Sarma

Oregon State University
Corvallis, Oregon, USA

{minic, perdriac, letawl, emardj, steinehz, burnett, sarmaa}@oregonstate.edu
Clone Detection
Code Clone Categorization

• Type-1 clones – Identical code fragments but may have some variations in whitespace, layout, and comments
• Type-2 clones – Syntactically equivalent fragments with some variations in identifiers, literals, types, whitespace, layout and comments
• Type-3 clones – Syntactically similar code with inserted, deleted, or updated statements
• Type-4 clones – Semantically equivalent, but syntactically different code
Key points of Code Clone

• Pros
  – Increase performance
    • Code inlining vs. function call
  – Increase program readability

• Cons
  – Increase maintenance cost
    • If one code fragment contains a bug and gets fixed, all its clone peers should be always fixed in similar ways.
  – Increase code size
Detecting Strategies

- Text matching
- Token sequence matching
- Graph matching
Collaboration

The ACM Conference on Computer-Supported Cooperative Work (CSCW) is the premier international forum for the latest in interdisciplinary research on the design, evaluation, and impact of technologies that support human collaboration. This year, CSCW is being held online for the first time.

Our theme for CSCW 2021 is "Beyond the Cloud: New Opportunities for Collaboration in the Cloud and Beyond." This gives new meaning to "designing for the cloud" when the "cloud" itself is no longer in the sky.

Please see the Program for information about the virtual conference.
Predicting Developers’ Negative Feelings about Code Review

Carolyn D. Egelman¹, Emerson Murphy-Hill¹, Elizabeth Kammer¹, Margaret Morrow Hodges²,
Collin Green¹, Ciara Jaspan¹, James Lin¹

How Software Practitioners Use Informal Local Meetups to Share Software Engineering Knowledge

Erik H. Trainer, Arun Kalyanasundaram, Chalalai Chaithirunarn, James D. Herbsleb
Institute for Software Research
Carnegie Mellon University

Scaling Open Source Communities: An Empirical Study of the Linux Kernel

Xin Tan
Department of Computer Science and Technology, Peking University

Minghui Zhou*
Department of Computer Science and Technology, Peking University

Brian Fitzgerald
Lero—the Irish Software Research Centre, University of Limerick