Software Analytics in Practice

Dongmei Zhang  
Software Analytics group  
Microsoft Research Asia

June 6, 2012

Tao Xie  
Computer Science Department  
North Carolina State University
Dongmei Zhang

• Senior Researcher at Microsoft Research Asia (MSRA)
• Founded Software Analytics (SA) Group at MSRA in May 2009
• Research Manager of MSRA SA

• Microsoft Research Asia (MSRA)
  – Founded in November 1998 in Beijing, China
  – 2nd-largest MSR lab with 200+ researchers
• Projects started in 2004 to research how data could help with software development
Tao Xie

• Associate Professor at North Carolina State University, USA
• Leads the ASE research group at NCSU
• PC Co-Chair of ICSM 2009, MSR 2011/2012
• Co-organizer of 2007 Dagstuhl Seminar on Mining Programs and Processes
Outline

• Overview of Software Analytics
• Selected projects
• Experience sharing on Software Analytics in practice
New Era...Software itself is changing...

Software

Services
How people use software is changing... 

Individual  
Isolated  
Not much content generation

Social  

Collaborative  
Huge amount of artifacts Generated anywhere anytime
How software is built & operated is changing...

- Code centric
- In-lab testing
- Experience & gut-feeling
- Centralized development
- Long product cycle
- ... 

- Data pervasive
- Debugging in the large
- Informed decision making
- Distributed development
- Continuous release
- ...
Software Analytics

Software analytics is to enable software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for data-driven tasks around software and services.
Software Analytics

Software analytics is to enable software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for data-driven tasks around software and services.
Five dimensions

- Research Topics
- Target Audience
- Output
- Technology Pillars
- Connection to Practice
Research topics & technology pillars

- Covering different areas of software domain
- Throughout entire development cycle
- Enabling practitioners to obtain insights
Data sources

- Runtime traces
- Program logs
- System events
- Perf counters

- Usage log
- User surveys
- Online forum posts
- Blog & Twitter

- Source code
- Bug history
- Check-in history
- Test cases
Target audience – software practitioners

- Program Manager
- Developer
- Management personnel
- Designer
- Tester
- Support engineer
- Operation engineer
- Usability engineer
Output – insightful information

• Conveys meaningful and useful understanding or knowledge towards completing the target task
• Not easily attainable via directly investigating raw data without aid of analytics technologies
• Examples
  – It is easy to count the number of re-opened bugs, but how to find out the primary reasons for these re-opened bugs?
  – When the availability of an online service drops below a threshold, how to localize the problem?
Output – actionable information

• Enables software practitioners to come up with concrete solutions towards completing the target task

• Examples
  – Why bugs were re-opened?
    • A list of bug groups each with the same reason of re-opening
  – Why availability of online services dropped?
    • A list of problematic areas with associated confidence values
  – Which part of my code should be refactored?
    • A list of cloned code snippets easily explored from different perspectives
Research topics & technology pillars

- Information Visualization
- Data Analysis Algorithms
- Large-scale Computing

Software System
Software Users
Software Development Process

Quality
Experience
Productivity

Vertical
Horizontal
Connection to practice

• Software Analytics is naturally tied with software development practice

• Getting real

Real Data | Real Problems | Real Users | Real Tools
Approach

- Task Definition
- Data Collection
- Analytics Technology Development
- Deployment
- Feedback Collection
Outline

• Overview of Software Analytics
• Selected projects
• Experience sharing on Software Analytics in practice
Selected projects in Software Analytics

StackMine – Performance debugging in the large via mining millions of stack traces

Scalable code clone analysis

Data exploration for Customer Experience Improvement Program (CEIP)
StackMine

Performance debugging in the large via mining millions of stack traces
Performance issues in the real world

- One of top user complaints
- Impacting large number of users every day
- High impact on usability and productivity

High Disk I/O

High CPU consumption

As modern *software systems* tend to get more and more *complex*, given *limited* time and resource *before* software *release*, development-*site* testing and debugging become more and more *insufficient* to ensure satisfactory software performance.
Performance debugging in the large

Pattern Matching

Problematic Pattern Repository

How many issues are still unknown?

Trace Storage

Which trace file should I investigate first?

Bug Database

Bug update

Bug filing

Key to issue discovery

Bottleneck of scalability

Trace analysis

Network

Trace collection

Trace analysis

How many issues are still unknown?

Which trace file should I investigate first?
Software practitioners

• A team of developers as performance analysts
• Highly skillful in single-trace analysis using existing tools
• Knowledgeable about Windows performance issues
• Conducting tool development in addition to analysis
Problem definition

Given operating system traces collected from tens of thousands (potentially millions) of users, how to help domain experts identify the program execution patterns that cause the most impactful underlying performance problems with limited time and resource.
Goal

Conduct systematic program execution pattern discovery & analysis that enables –

• Efficient handling of large-scale trace sets
• Automatic discovery of new patterns
• Effective prioritization of investigation
Challenges

Large-scale trace data
- TBs of trace files and increasing
- Millions of events in single trace stream

Highly complex analysis
- Numerous program runtime combinations triggering performance problems
- Multi-layer runtime components from application to kernel being intertwined

Combination of expertise
- Generic machine learning tools without domain knowledge guidance do not work well
What happens behind a typical UI delay? An example of delayed browser tab creation.

UI thread

CPU

Wait

Ready

CPU

Wait

Ready

CPU

Underlying Disk I/O

Unexpected long execution

Worker thread

CPU sampled callstack

What happens behind a typical UI delay? An example of delayed browser tab creation.
Approach

Formulate as a callstack mining and clustering problem

Caused by:
- Performance Issues
- Problematic program execution patterns

Mainly represented by:
- Callstack patterns

Discovered by mining & clustering costly patterns
Technical highlights

• Machine learning for system domain
  – Formulate the discovery of problematic execution patterns as callstack mining & clustering
  – Systematic mechanism to incorporate domain knowledge

• Interactive performance analysis system
  – Parallel mining infrastructure based on HPC + MPI
  – Visualization aided interactive exploration
Impact

“We believe that the MSRA tool is highly valuable and much more efficient for mass trace (100+ traces) analysis. For 1000 traces, we believe the tool saves us 4-6 weeks of time to create new signatures, which is quite a significant productivity boost.”

Highly effective new issue discovery on Windows mini-hang

Continuous impact on future Windows versions
For details...

Wednesday June 6, 2012, 10:45am
ICSE Research Session 4: Performance Analysis

Performance Debugging in the Large via Mining Millions of Stack Traces

Shi Han, Yingnong Dang, Song Ge, Dongmei Zhang, and Tao Xie
(Microsoft Research, China; North Carolina State University, USA)
XIAO

Scalable code clone analysis
Code clone research

• Tons of papers published in the past 10 years
• 6 years of International Workshop on Software Clones (IWSC) since 2006
• Dagstuhl Seminar
  – Software Clone Management towards Industrial Application (2012)

Source: http://www.dagstuhl.de/12071
XIAO: Code Clone Analysis

• Motivation
  – Copy-and-paste is a common developer behavior
  – A real tool widely adopted internally and externally

• XIAO enables code clone analysis in the following way
  – High tunability
  – High scalability
  – High compatibility
  – High explorability

[Visual Studio]

Code Clone Detection Experience at Microsoft
Yinghong Dang, Song De, Ray Huang and Dongmei Zhang
Microsoft Research Asia
yidang.song@rayhuang.dongmei@microsoft.com

ABSTRACT
Cloning source code is a common practice in the software development process. In general, the number of code clones is on the rise, which brings various challenges to project managers, engineering managers and software developers. Code clones are usually considered to be a kind of redundant source code that can be removed at some point in time. In this paper, we study clone detection in a real development project. We also focus on optimizing and validating XIAO, a code clone detection tool developed at Microsoft. We explore the efficiency and quality of code clone detection in a real development environment by using XIAO both as an exploratory tool and development setup.

[IWSC’11 Dang et.al.]
High tunability – what you tune is what you get

- Intuitive similarity metric
  - Effective control of the degree of syntactical differences between two code snippets
- Tunable at fine granularity
  - Statement similarity
  - % of inserted/deleted/modified statements
  - Balance between code structure and disordered statements

```c
for (i = 0; i < n; i++) {
    a ++;
    b ++;
    c = foo(a, b);
    d = bar(a, b, c);
    e = a + c;
}
```

```c
for (i = 0; i < n; i++) {
    c = foo(a, b);
    a ++;
    b ++;
    d = bar(a, b, c);
    e = a + d;
    e ++;
}
High scalability

• Four-step analysis process

- Pre-processing
- Coarse Matching
- Pruning
- Fine Matching

• Easily parallelizable based on source code partition
High compatibility

- Compiler independent
- Light-weight built-in parsers for C/C++ and C#
- Open architecture for plug-in parsers to support different languages

- Easy adoption by product teams
  - Different build environment
  - Almost zero cost for trial
High explorability

1. Clone navigation based on source tree hierarchy
2. Pivoting of folder level statistics
3. Folder level statistics
4. Clone function list in selected folder
5. Clone function filters
6. Sorting by bug or refactoring potential
7. Tagging

1. Block correspondence
2. Block types
3. Block navigation
4. Copying
5. Bug filing
6. Tagging
Scenarios & Solutions

**Quality gates at milestones**
- Architecture refactoring
- Code clone clean up
- Bug fixing

**Post-release maintenance**
- Security bug investigation
- Bug investigation for sustained engineering

**Development and testing**
- Checking for similar issues before check-in
- Reference info for code review
- Supporting tool for bug triage
Benefiting developer community

Available in Visual Studio 2012 RC

Searching similar snippets for fixing bug once

Finding refactoring opportunity
More secure Microsoft products

Code Clone Search service integrated into workflow of Microsoft Security Response Center

Hundreds of million of lines of code indexed across multiple products

Real security issues proactively identified and addressed
Example – MS Security Bulletin MS12-034


3 publicly disclosed vulnerabilities and seven privately reported involved. Specifically, one is exploited by the Duqu malware to execute arbitrary code when a user opened a malicious Office document

Insufficient bounds check within the font parsing subsystem of win32k.sys
Cloned copy in gdiplus.dll, ogl.dll (office), Silver Light, Windows Journal viewer

Microsoft Technet Blog about this bulletin

“However, we wanted to be sure to address the vulnerable code wherever it appeared across the Microsoft code base. To that end, we have been working with Microsoft Research to develop a “Cloned Code Detection” system that we can run for every MSRC case to find any instance of the vulnerable code in any shipping product. This system is the one that found several of the copies of CVE-2011-3402 that we are now addressing with MS12-034.”
Three years of effort

Prototype development
- Problem formulation
- Algorithm research
- Prototype development

Early adoption
- Algorithm improvement
- System / UX improvement

Tech transfer
- System integration
- Process integration
Data Exploration for CEIP Data
Customer Experience Improvement Program

• Multiple channels for user feedback
  – Usability test, surveys, focus groups & other field studies
  – Limited customer base

• CEIP
  – Providing all customers with ability to contribute to the design and development of MS products
  – Voluntary participation
  – Anonymous
CEIP data

• Usage
  – How software is used
  – Examples: general feature usage, commands on Ribbon, actions taken in wizards, etc.

• Reliability and performance
  – Whether software performs as expected
  – Examples: assertions for logical inconsistency, measuring execution speed, etc.

• Hardware/software configuration
  – Providing context for data interpretation
  – Example: long document loading time only on machines with low RAM or a particular processor speed?

Source: CEIP at http://www.microsoft.com
Questions answered by usage data

• Command usage
  – How frequently is it used? [Prominence on UI]
  – How many people use it? [Impact]
  – What is the most frequent way of accessing it? [Ease of access]
  – Does this command occur as part of a clear workflow? [Better support]

• Feature usage
  – How many files contain a Table? [Impact]
  – How big is the average Table? [Optimization choice]
  – What are the most frequently used Table styles? [Design choices]
  – What other features are used in files containing Tables? [interaction with other features]

Source: CEIP at http://www.microsoft.com
Software practitioners

• Userbility engineers
• Program managers
• Developers
• Testers
• Managers
Different levels of data analysis

Report consumers
- Getting information from dashboards & canned reports
- Little knowledge of raw data

Report creators
- Creating various reports for different analysis needs
- Some knowledge of raw data

Data expert
- Conducting special and in-depth analysis not covered by reporting
- Good knowledge of raw data
Goal

How to unlock the insights in the data collected via the Customer Experience Improvement Program (CEIP) at scale for software practitioners of different disciplines?
Challenges

• Deep learning curve on data schema and data query
• Difficult to conduct in-depth analysis due to data complexity
• Different levels of knowledge and skills among practitioners
• Huge amount of data
Approach

• Building infrastructure for large-scale data storage and computing

• Collaborating with product teams to create interactive reporting portal to answer the most commonly asked questions across different feature teams

• Building flexible and easy-to-use visual query tools to enable ad hoc multi-dimensional analysis
Analytics related techniques

• Large-scale data processing and computing
• Highly interactive data visualization controls
• Sequence mining
• Pattern matching
Impact

• CEIP data analysis service #1 has been used across company for about four years

• CEIP data analysis service #2 was used in an important business division for two and half years
Outline

• Overview of Software Analytics
• Selected projects
• Experience sharing on Software Analytics in practice
Analytics is the means to the end

Interesting results vs. Actionable results vs. Problem hunting vs. Problem driven
Beyond the “usual” mining

- Mining vs. matching
- Automatic vs. interactive
- Researchers vs. practitioners
Keys to making real impact

• Engagement of practitioners
  - Solving their problem
  - Timing
  - Champions in product teams
  - Culture

• Walking the last mile
  - Targeting at real scenarios
  - “It works” is not enough
  - Trying out tool has cost
  - Getting engineering support

• Combination of expertise
  - Research capabilities
  - Visualization & design
  - Engineering skills to build systems
  - Communication
Suggested actions for academia

• Get research problems from real practice
• Get feedback from real practice
• Collaborate across disciplines
• Collaborate with industry
Conclusion

• Mission of Software Analytics
  – Utilizing data-driven approach to help create highly performing, user friendly, and efficiently built & operated software and services

• Software Analytics in practice
  – Insightful & actionable
  – Collaboration with practitioners
Q & A

http://research.microsoft.com/groups/sa/