Introduction to Software Analytics

Dongmei Zhang
Software Analytics Group
Microsoft Research

November 26, 2014
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
  - Not much content generation

- Social
  - Huge amount of artifacts generated anywhere anytime

- Collaborative
  - Not much content generation

- Isolated
  - Not much content generation
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

1. Research Topics
2. Target Audience
3. Output
4. Technology Pillars
5. Connection to Practice
Research topics

- 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 and technology pillars

- Quality
- Experience
- Productivity

- Software System
- Software Users
- Software Development Process

Vertical:
- Information Visualization
- Data Analysis Algorithms

Horizontal:
- Large-scale Computing
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 → Task Definition
Various related efforts...

• Mining Software Repositories (MSR)
• Software Intelligence
• Software Development Analytics

http://www.msrconf.org/
Outline

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

XIAO  
Scalable code clone analysis

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

Service Analysis Studio: Incident management for online services
XIAO

Scalable code clone analysis

Code clone research

• Tons of papers published in the past decade

• 8 years of International Workshop on Software Clones (IWSC) since 2006

• Dagstuhl Seminar
  • Software Clone Management towards Industrial Application (2012)
  • Duplication, Redundancy, and Similarity in Software (2006)
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
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 and 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

Online code clone search
Offline code clone analysis
Benefiting developer community

Available in Visual Studio 2012

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

Over hundreds of millions of lines of code indexed across multiple products

Real security issues proactively identified and addressed

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
StackMine

*Performance debugging in the large via mining millions of stack traces*

Shi Han, Yingnong Dang, Song Ge, Dongmei Zhang, and Tao Xie, *Performance Debugging in the Large via Mining Millions of Stack Traces*, in *Proceedings of the 34th International Conference on Software Engineering (ICSE 2012)*, Zurich, Switzerland, June 2012.
Performance issues in the real world

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

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

- Network
- Trace collection
- Pattern Matching
- Problematic Pattern Repository
- Trace Storage
- Bug Database
- Bug update
- Bug filing
- Key to issue discovery
- Bottleneck of scalability
- Trace analysis

Which trace file should I investigate first?
How many issues are still unknown?
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

Systematic analysis of OS trace sets that enables
  • Efficient handling of large-scale trace sets
  • Automatic discovery of new program execution patterns
  • Effective prioritization of performance 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 intertwined

Combination of expertise
• Generic machine learning tools without domain knowledge guidance do not work well
Intuition

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

**Wait callstack**
- ntdll!UserThreadStart
- ntdll!WorkerThread
- ole!CoCreateInstance
- ole!OutSerializer::UnmarshalAtIndex
- ole!CoUnmarshalInterface

**ReadyThread callstack**
- nt!KiRetireDpcList
- nt!ExecuteAllDpcs
- nt!IoIoComplete
- nt!SetEvent

**UI thread**
- Unexpected long execution
  - ntdll!UserThreadStart
  - Browser!OnBrowserCreatedAsyncCallback
  - BrowserUtil!ProxyMaster::GetOrCreateSlave
  - BrowserUtil!ProxyMaster::ConnectToObject
  - rpc!ProxySendReceive
  - wow64!RunCpuSimulation
  - wow64cpu!WaitForMultipleObjects32
  - wow64cpu!CpuPsyscallStub

**Worker thread**
- CPU sampled callstack
  - ntdll! UserThreadStart
  - ntdll! WorkerThread
  - ole! CoCreateInstance
  - ole! OutSerializer::UnmarshalAtIndex

**Underlying Disk I/O**
- nt!IopfCompleteRequest
- nt!SetEvent

**ReadyThread callstack**
- nt!IopfCompleteRequest
- nt!IopfCompleteRequest
- nt!IopfCompleteRequest
- nt!IopfCompleteRequest
Approach

Formulate as a callstack mining and clustering problem

Performance Issues → Problematic program execution patterns → Callstack patterns

Caused by

Mainly represented by

Discovered by mining & clustering costly patterns
Technical highlights

• Machine learning for system domain
  • Formulate the discovery of problematic execution patterns as callstack mining and 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
Service Analysis Studio

Incident management for online services

Motivation

• Online services are increasingly popular and important
• High service quality is the key
• Incident management is a critical task to ensure service quality
Incident management: workflow

1. Alert On-Call Engineers (OCEs)
2. Investigate the problem
3. Restore the service
4. Fix root cause via postmortem analysis
Incident management: characteristics

**Shrink-Wrapped Software Debugging**
- Root Cause and Fix
- Debugger
- Controlled Environment

**Online Service Incident Management**
- Workaround
- No Debugger
- Live Data
Incident management: challenges

- Large-volume and noisy data
- Highly complex problem space
- Knowledge scattered and not well organized
- Few people with knowledge of entire system
## Data sources

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Performance Indicators (KPI)</td>
<td>Measurements indicating the major quality perspectives of an online service</td>
<td>Request failure rate, average request latency, etc.</td>
</tr>
<tr>
<td>Performance counters and system events</td>
<td>Measurements and events indicating the status of the underlying system and applications</td>
<td>CPU, disk queue length, I/O, request workload, SQL-related metrics, and application-specific metrics, etc.</td>
</tr>
<tr>
<td>User requests</td>
<td>Information on user requests</td>
<td>Request return status, processing time, consumed resources, etc.</td>
</tr>
<tr>
<td>Transaction logs</td>
<td>Generated during execution, recording system runtime behaviors when processing requests</td>
<td>Timestamp, request ID, thread ID, event ID, and detailed text message, etc.</td>
</tr>
<tr>
<td>Incident repository</td>
<td>Historical records of service incidents</td>
<td>Incident description, investigation details, restoration solution, etc.</td>
</tr>
</tbody>
</table>
Service Analysis Studio (SAS)

• Goal
  Given an incident in an online service, effectively helping service engineers reduce Mean Time To Restore (MTTR).

• Design principals
  • Automating data analysis
  • Handling heterogeneous data sources
  • Accumulating knowledge
  • Supporting human-in-the-loop (HITL)
Data analysis techniques

- Mining suspicious execution patterns from transaction logs
- Mining resolution solutions from historical incidents
- Identifying incident beacons from system metrics

Data-driven service analytics
Impact

Deployment

• SAS deployed to worldwide datacenters of Service X in June 2011
• Five more updates since first deployment

Usage

• Heavily used by On-Call Engineers of Service X for about 2 years
• Helped successfully diagnose ~76% of service incidents
Lessons learned

- Understanding and solving real problems
- Understanding data and system
- Handling data issues
- Making SAS highly usable
- Achieving high availability and performance
- Delivering step-by-step
Understanding and solving real problems

• Working side-by-side with On-Call Engineers
• Targeting at reducing MTTR
• Focusing on addressing challenges in real-world scenarios
Understanding data and system

Techniques

Practical Problems
Handling data issues

Data issues
(1) Missing/duplicated
(2) Buggy
(3) Disordered

Approach
(1) Preprocessing
(2) Designing robust algorithms

Experience
Data preprocessing cannot be perfect. Robust algorithms are in great need.
Making SAS highly usable

There is an internal server error related issue.

Datacenter: DC1
Start time: 9/4/2012 3:48:00 AM    End time: 9/4/2012 3:58:00 AM

Impact:

<table>
<thead>
<tr>
<th>Influenced requests</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Influenced end users</td>
<td>100</td>
</tr>
</tbody>
</table>

Diagnosis:
This issue is a problem of “Credential loss”. The source of the issue mainly locates at Front End Server—“FE001”.

Here are similar previous occurrences of the issue:

- Incident ID 91236: 3/14/2012 10:49:00 AM [see detail]
- Incident ID 91271: 7/25/2012 14:25:00 AM [see detail]

See also:

- Malfunctioned Frontend Servers: 973 of 1000 failed requests related to FE001.
- Malfunctioned SQL Servers: No malfunctioned SQL servers detected.
- Suspicious Metrics: No highly correlated metrics found.
- Suspicious Execution Patterns: 1 major pattern in the logs covers 973 of 1000 failed requests.

Suggested actions based on similar past incident (ID 91236):
Reset the IIS service on the front end server FE001.
Achieving high availability and performance

• SAS is also a service
  • To serve On-Call Engineers at any time with high performance
  • Critical to reducing MTTR of services

• Auto recovery
  • Continuously monitored
  • Check-point mechanism adopted

• Backend service + On-demand analysis
Delivering step-by-step

• Demonstrating value and building trust
  • Deployment in production has cost and risk
  • In-house → dogfood → one datacenter → worldwide datacenters

• Getting timely feedback
  • Requirements may not be clear early on and requirements may change
  • Gaining troubleshooting experiences from On-Call Engineers
  • Understanding how SAS was used
  • Identifying direction of improvement
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

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
Together let us walk the exciting journey to make great impact!
Q & A

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