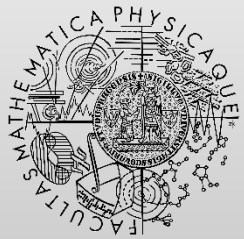
Monitoring: Runtime Behavior & Software Development Process

http://d3s.mff.cuni.cz

Pavel Parízek

parizek@d3s.mff.cuni.cz



FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Monitoring runtime behavior

Monitoring runtime behavior

- Goals
  - Recording information about program behavior
  - Notification about specific important events
- Information: performance, security, exceptions
- Target domain: long-running programs
  - Application servers (JBoss, Tomcat, WebSphere, ...)
  - Network servers and daemons (Apache, Sendmail)
- Alternative name: **tracing**

Department of Distributed and Dependable Systems D3S

Basic approaches

- Manual implementation of logging commands
- Using tools for automated runtime monitoring

Tools

- Unix-like platforms
  - Syslog, strace, ltrace, DTrace
- Java ecosystem
  - Log4j 2, Java Logging API, VisualVM, JVM TI
- Windows/.NET
  - Log4net, NLog, Process Explorer
- Other (multiplatform)
  - Log4cxx, ng-log, spdlog

- Events: custom messages, system calls, library calls
- Output: text log files (off-line inspection), GUI

Log4j

- Popular logging framework for Java platform
  - http://logging.apache.org/log4j/2.x/
- Features
  - Hierarchy of loggers based on class names
  - Filtering messages based on logging levels
  - Dynamically updateable configuration (XML)
  - Multiple output destinations (console, file)
  - Formatting log messages (printf-style, HTML)

Log4j API: example

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;


// get a Logger object with a particular name
Logger logger = LogManager.getLogger("cz.cuni.mff");

logger.warn("Running out of disk space");
...
logger.error("File {} not found", f.getName());
...
logger.info("Something normal happened");
```

Department of Distributed and Dependable Systems

Using Log4j

- Levels
  - TRACE < DEBUG < INFO < WARN < ERROR < FATAL
- Logger objects
  - Identified by logical names (e.g., Java class names)
  - They make a hierarchy based on the name prefixes
    - Logger named “cz.cuni” is a parent for the Logger “cz.cuni.mff”
    - Inheriting configuration (levels, appenders, formatting pattern)
    - Root Logger always exists at the top of any custom hierarchy
- Configuration: XML, programmatic
  - Default file name `log4j2.xml` (must be on classpath)

Configuration: example

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
  <Appenders>
    <Console name="konzole" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss} %-5level %c{36} - %m%n"/>
    </Console>
    <File name="logfile" fileName="test.log">
      <PatternLayout pattern="%d{HH:mm:ss} %-5level %c{36} - %m%n"/>
    </File>
  </Appenders>
  <Loggers>
    <Logger name="cz.cuni.mff" level="info">
      <AppenderRef ref="konzole"/>
    </Logger>
    <Root level="error">
      <AppenderRef ref="logfile"/>
    </Root>
  </Loggers>
</Configuration>
```

Appenders

- Responsible for writing log messages to actual target destinations

- Supported targets
  - Console (`stdout`, `stderr`)
  - File (buffered, appending)
  - Database (via JDBC)
  - SMTP (sending emails)
  - Network socket (TCP, UDP)
  - Unix/Linux syslog service

Layout

- Purpose: formatting messages
- Available layouts
  - Pattern
    - `%m` // message text
    - `%n` // line separator
    - `%-5level` // level, justified to the right, width five chars
    - `%d{HH:mm:ss}` // current datetime with pattern
    - `%c{20}` // logger name with the maximal length
    - `%C %M %L` // class name, method name, line number
    - `%t` // thread name
  - HTML, XML, Syslog

Tracing control flow

```
public Object doSomething(int arg1) {
   logger.entry(arg1);
   try {
      ...
      Object res = ...
   }
   catch (Exception ex) {
      logger.catching(ex)
   }
   logger.exit(res);
}
```

Log4j: other features

- Filtering messages
  - markers, regular expression, time
- Automatic reconfiguration
  - if you update the XML configuration file at runtime

Modern logging frameworks

- Simple Logging Facade for Java (SLF4J)
  - General unified API for logging frameworks
  - Supported backends: Log4j, logback, ...
  - http://www.slf4j.org/
- Logback
  - Replacement for Log4j
  - Implements SLF4J API
  - http://logback.qos.ch/

Logging for .NET (C#, VB) and other

- Log4net
  - http://logging.apache.org/log4net/index.html
- NLog
  - http://nlog-project.org/
  - https://github.com/NLog/NLog/wiki
- Log4cxx
  - https://logging.apache.org/log4cxx/latest_stable/index.html
- Features
  - Configuration: file (XML), programmatic (API)
  - Multiple targets (file, database, console, email)
  - Layouts (plain text, CSV, XML, JSON)

Exercise

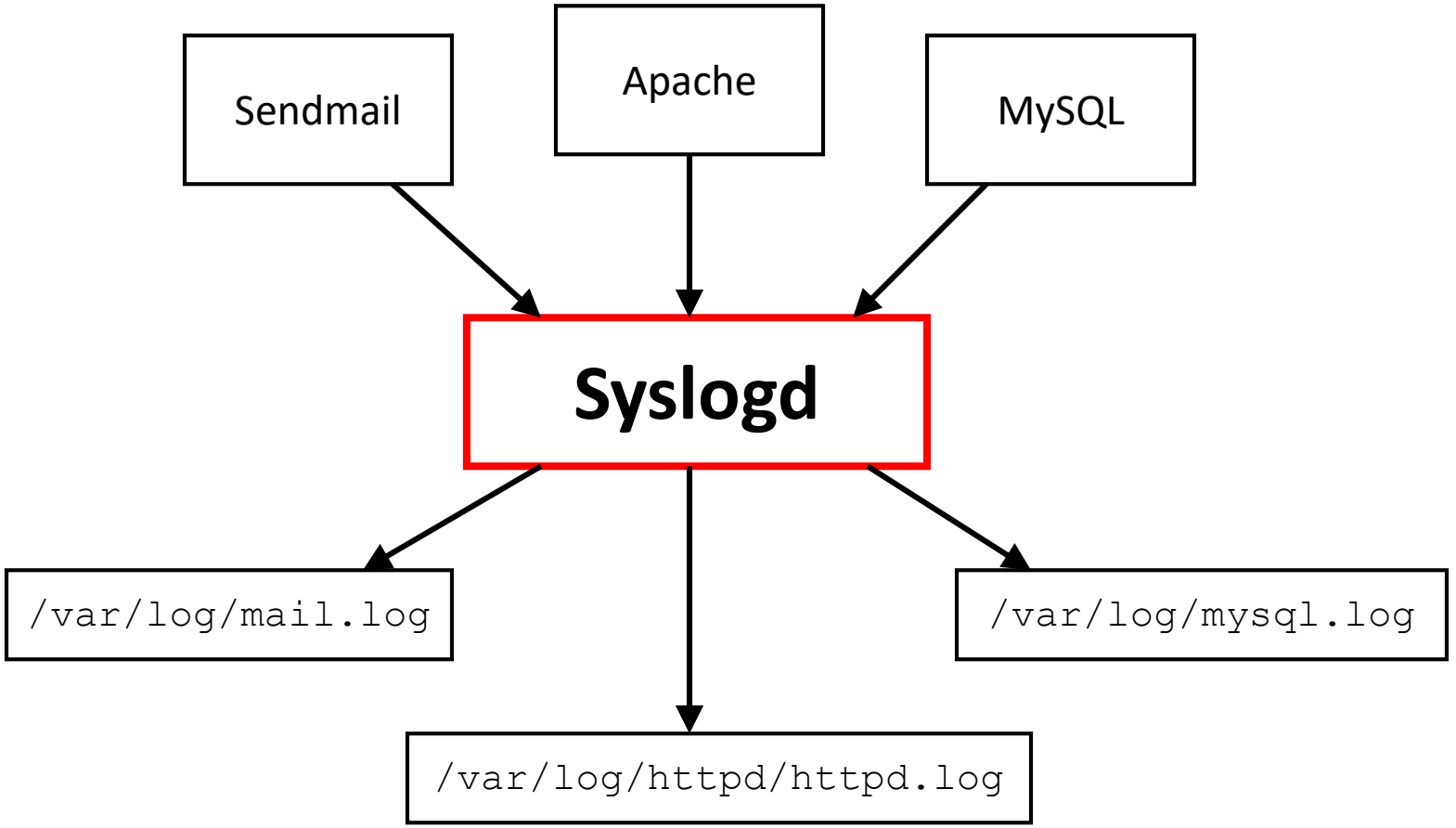
- Download Log4j/Log4net from the web
  - http://logging.apache.org/log4j/2.x/
    - Only important JAR files: core, api
  - http://logging.apache.org/log4net/
- Write simple program in Java or C#
  - You can also take some existing program (anywhere)
- Try important features of the particular logging framework
  - Use several Loggers
  - Different log levels
  - Configuration (XML)
  - Tracing control flow
- Check the output (console, log files)

Syslog

- Standard logging framework for Unix-like systems
- Service
  - Collecting messages from different sources (applications)
  - Writing received messages to various output destinations
    - log files (`/var/log`), another computer over network
  - Configuration: `/etc/syslog.conf`, `/etc/rsyslog.conf`
  - Log rotation: `/var/log/messages`, `/var/log/messages.1`, …
- Protocol
  - Format of data exchanged between applications and the service
  - Message: content (plaintext, < 1024 bytes), priority
  - Supported priorities (low to high)
    - debug, info, notice, warning, error, critical, alert, emergency
  - Definition: RFC 3164, 3195

Department of Distributed and Dependable Systems D3S

Configuration: example



Syslog API: example

```
#include <syslog.h>

openlog("myprog", LOG_CONS | LOG_PID, LOG_USER);

syslog(LOG_NOTICE, "Program runs for %d hours", 2);
syslog(LOG_ERROR, "File %s does not exist", fname);

closelog();
```

strace

- Tool for monitoring interactions with the operating system kernel
  - System calls performed by the given program
  - Signals received by the given program from OS
- Available for Unix-like platforms

- Usage: `strace <program>`
  - Attaching to a running process: `strace -p <pid>`

- Output: list of system calls and signals

```
open("/etc/passwd", O_RDONLY) = 3
open("/etc/passwords", O_RDONLY) = -1 ENOENT (No such file)
```

Exercise / Demo

- Try using
  - `strace` (syscalls)
  - `ltrace` (libraries)
- Check output

VisualVM

- Download: https://visualvm.github.io/
- Key features
  - Provides useful information
    - CPU usage, memory consumption, threads
  - Nice graphical interface
  - Connection to remote JVM
- How to run it: `visualvm`
- Live demo

Department of Distributed and Dependable Systems

Monitoring tools for C#/.NET

- .NET Memory Profiler
  - https://marketplace.visualstudio.com/items?itemName=SciTechSoftware.NETMemoryProfiler
- dotMemory
  - https://www.jetbrains.com/dotmemory/

Windows Sysinternals

- Process Explorer
  - https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer
  - Displays information about running processes
- Process Monitor
  - https://learn.microsoft.com/en-us/sysinternals/downloads/procmon
  - Displays some live (real-time) process activity

Other monitoring tools

- Instrumentation (binary, source code)
- Notification about specific events
  - accesses to object fields and variables
  - locking (acquisition, release, attempts)
  - procedure calls (e.g., user-defined list)
- Pin: dynamic binary instrumentation tool
  - https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-dynamic-binary-instrumentation-tool.html
- JVM Tool Interface
  - https://docs.oracle.com/en/java/javase/17/docs/specs/jvmti.html
- .NET Profiling API
  - https://learn.microsoft.com/en-us/dotnet/framework/unmanaged-api/profiling/profiling-overview
- Valgrind: heavyweight dynamic binary translation
- DiSL (https://disl.ow2.org/bin/view/Main/)
- SharpDetect (https://github.com/acizmarik/sharpdetect)

Department of Distributed and Dependable Systems D3S

Log analysis tools

- Elasticsearch + Logstash + Kibana (ELK stack)
  - https://www.elastic.co/
- Azure Monitor (Application Insights)
  - https://azure.microsoft.com/en-us/products/monitor/
- Prometheus: https://prometheus.io/
- Sentry: https://sentry.io/

Monitoring development process

Issue tracking systems

- Typically part of a project management system
  - https://github.com/
  - https://www.gitlab.com/
  - https://bitbucket.org/
- Popular systems
  - Bugzilla, Trac, JIRA, YouTrack
- Issue = reported bug, feature request, other task
- Components
  - Some database of known issues
  - User interface (WWW, desktop)

Issue characteristics

- Time of reporting
- Product (module)
- Version of the product
- Severity of the bug / Priority of the feature
  - blocker, critical, major, normal, minor, enhancement
- Platform (OS, HW, SW)
- Textual comments
- Current status
  - new, unconfirmed, assigned, fixed, wontfix, resolved
- Assigned to
  - Who should do it (fix the bug, implement the feature)

Lifecycle of an issue (bug)

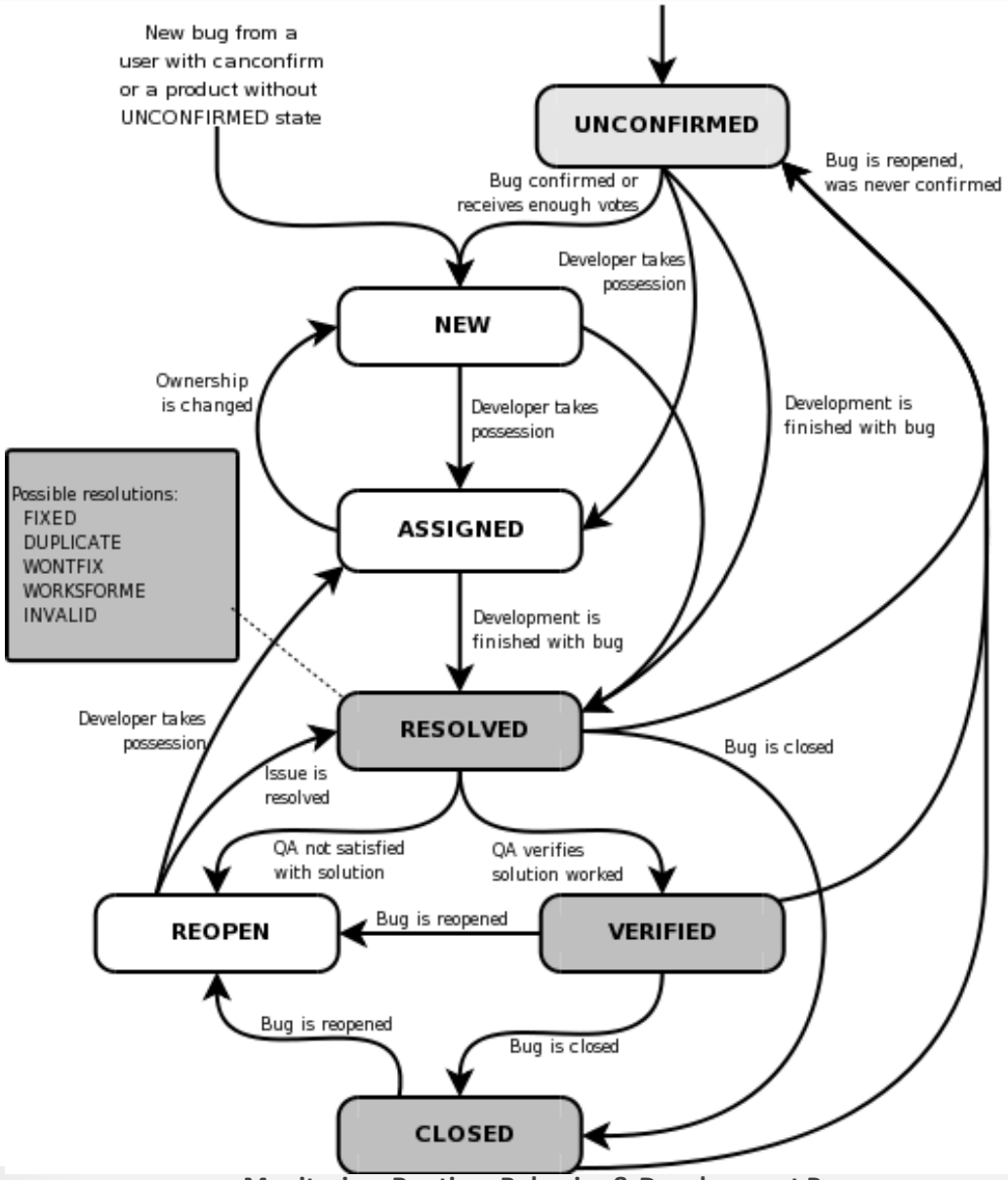


Figure taken from
http://www.bugzilla.org/docs

Common actions

- Developer
  - Entering new issues (bug reports)
  - Search for assigned tickets (issues)
  - Changing status of a specific ticket
- Manager
  - Inspecting overall statistics
  - Look for unresolved bugs
  - Assign priorities to features

Bugzilla

- Web-based tool
  - http://www.bugzilla.org
- SW requirements
  - Database (MySQL, PostgreSQL)
  - Perl 5 with specific modules
  - Web server (e.g., Apache httpd)
- Features
  - Advanced queries
    - Boolean operators (and, or, not)
  - Saved search
  - Cloning of bugs

Trac

- Project management system
  - http://trac.edgewall.org/
- Features
  - Tracking issues (bugs, feature requests)
  - Good integration with version control
    - Supported tools: Subversion, Mercurial, Git
    - Links from bug reports to source code files
  - Source code browser (version control)
  - Wiki pages (e.g., for documentation)

Test coverage

- Criteria: statement, branch, path
- Mutation testing
  - Detects missing tests
- Fault injection

- Practice: achieving 100% coverage is hard

- Resources
  - https://en.wikipedia.org/wiki/Mutation_testing

Test coverage – tools

- Mutation testing and fault injection
  - Jumble (http://jumble.sourceforge.net/)
  - PIT (http://pitest.org/)
  - Major (http://mutation-testing.org/)

- Coverage analysis
  - Cobertura (http://cobertura.sourceforge.net/)
  - Clover (http://www.atlassian.com/software/clover/)
  - dotCover (https://www.jetbrains.com/dotcover/)
  - JaCoCo (https://www.eclemma.org/jacoco/)
  - Support in Visual Studio (Test Explorer)
  - Coverlet (for C#/.NET and MS Test)
    - https://github.com/coverlet-coverage/coverlet
    - https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-code-coverage?tabs=windows
    - https://victormagalhaes-dev.medium.com/test-coverage-analysis-with-coverlet-in-net-2e38df3c6ed7

Department of Distributed and Dependable Systems

Continuous Integration (CI)

- Frequent merge, building, and test execution
  - https://en.wikipedia.org/wiki/Continuous_integration
  - https://martinfowler.com/articles/continuousIntegration.html
  - https://www.atlassian.com/continuous-delivery/continuous-integration

- Jenkins (https://jenkins.io/)
- Cruise Control (http://cruisecontrol.sourceforge.net/)
- TeamCity (http://www.jetbrains.com/teamcity/)
- Travis CI (https://travis-ci.org/)
- AppVeyor (https://www.appveyor.com/)

- Azure DevOps (https://azure.microsoft.com/en-us/products/devops)

- GitLab CI/CD (https://docs.gitlab.com/ee/ci/)
- GitHub Actions (https://github.com/features/actions)

Department of Distributed and Dependable Systems

Other links

- Syslog: http://www.gnu.org/software/libc/manual/html_node/Syslog.html

- Log4cxx: https://logging.apache.org/log4cxx/latest_stable/index.html
- ng-log: https://github.com/ng-log/ng-log
- spdlog: https://github.com/gabime/spdlog

- DTrace: http://dtrace.org/blogs/about/

- Swiss Java Knife: https://github.com/aragozin/jvm-tools

- YouTrack: https://www.jetbrains.com/youtrack/
- JIRA: https://www.atlassian.com/software/jira

Homework

- Assignment
  - ReCodEx: group associated with this course
  - Web: http://d3s.mff.cuni.cz/files/teaching/nswi154/ukoly/
- Deadline
  - 23.4.2025

Department of Distributed and Dependable Systems