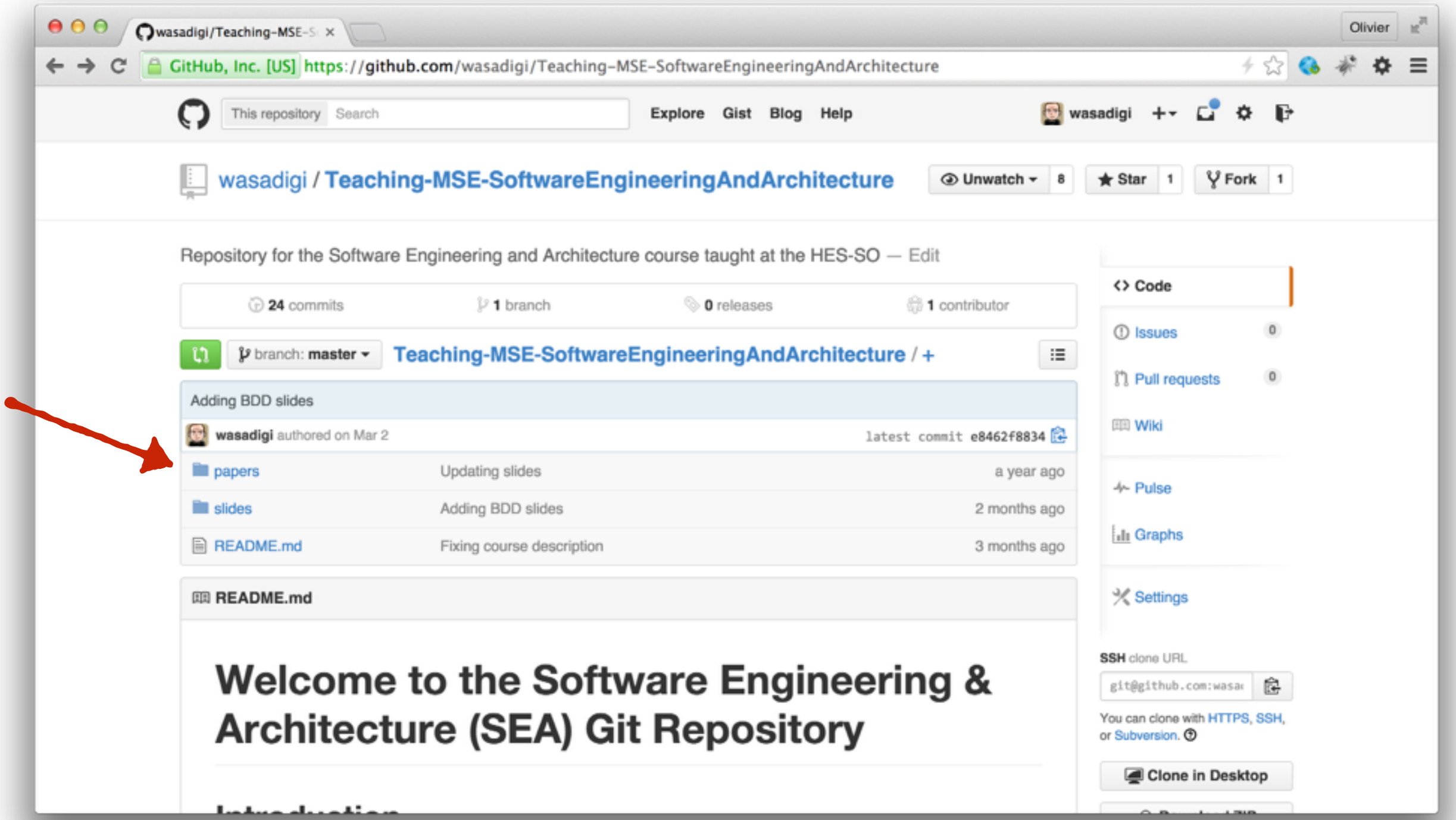
# Introduction to Software Evolution

Olivier Liechti
HEIG-VD
olivier.liechti@heig-vd.ch

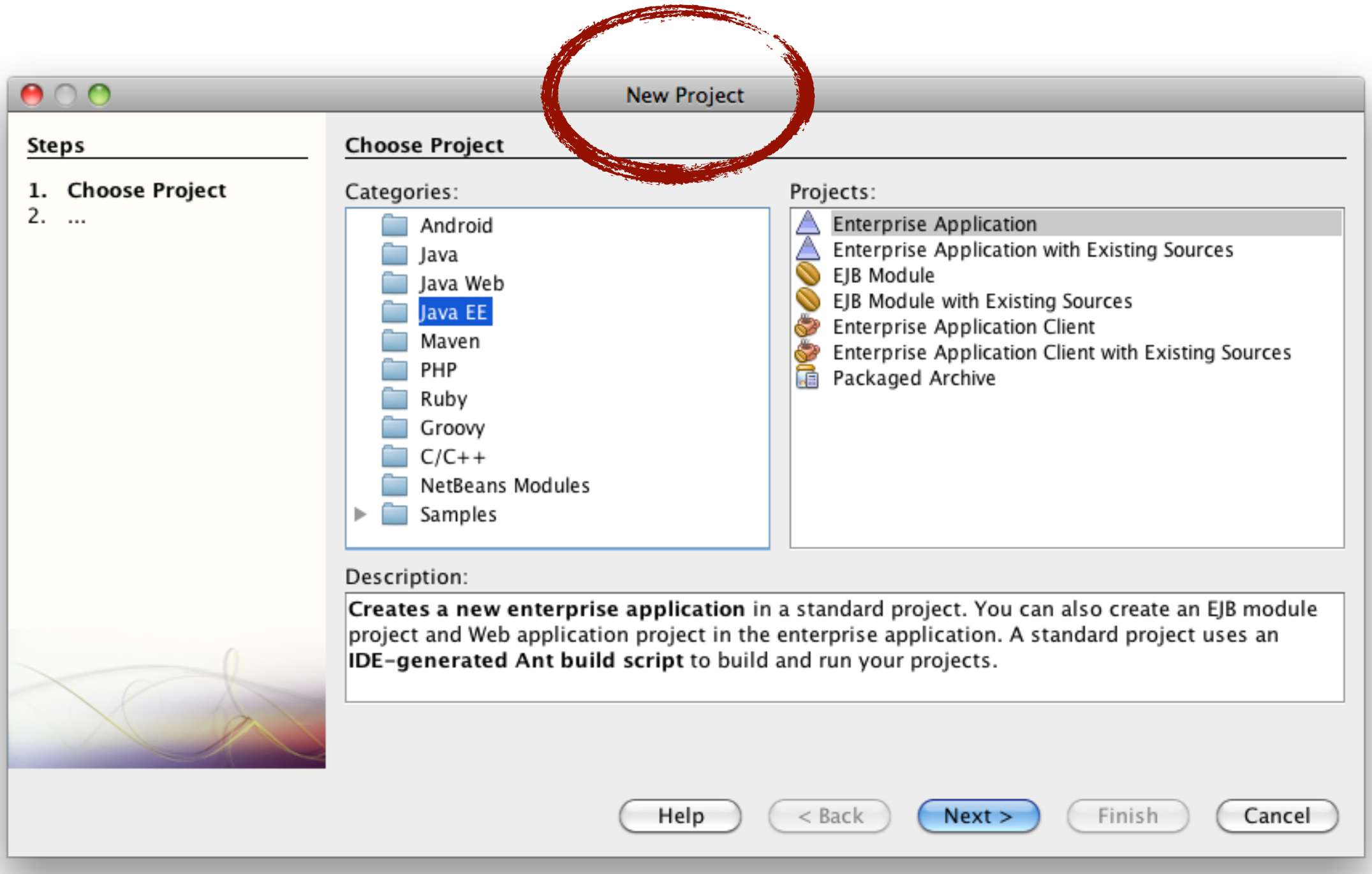
# Schedule

- **Theory**
  - **Introduction**: the scientific and engineering fields of **software evolution**
  - **Foundations**: Laws of Software Evolution, Software Aging
  - **Tools**: code quality tools, software repository mining, etc.
- **Practice**
  - Managing software evolution et code quality with **SonarQube**
  - Integrating SonarQube in your **continuous delivery pipeline**

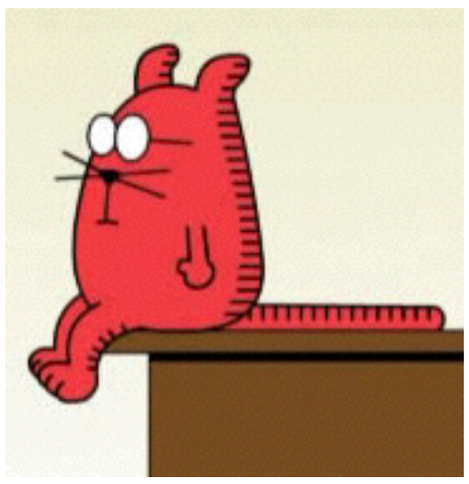

wasadigi/Teaching-MSE-S
Olivier
GitHub, Inc. [US] https://github.com/wasadigi/Teaching-MSE-SoftwareEngineeringAndArchitecture
This repository Search
Explore Gist Blog Help
wasadigi
wasadigi / Teaching-MSE-SoftwareEngineeringAndArchitecture
Unwatch 8 Star 1 Fork 1
Repository for the Software Engineering and Architecture course taught at the HES-SO — Edit
24 commits 1 branch 0 releases 1 contributor
branch: master Teaching-MSE-SoftwareEngineeringAndArchitecture / +
Adding BDD slides
wasadigi authored on Mar 2
latest commit e8462f8834
papers Updating slides a year ago
slides Adding BDD slides 2 months ago
README.md Fixing course description 3 months ago
README.md
Welcome to the Software Engineering & Architecture (SEA) Git Repository
Introduction
Code
Issues 0
Pull requests 0
Wiki
Pulse
Graphs
Settings
SSH clone URL
git@github.com:wasa
You can clone with HTTPS, SSH, or Subversion.
Clone in Desktop

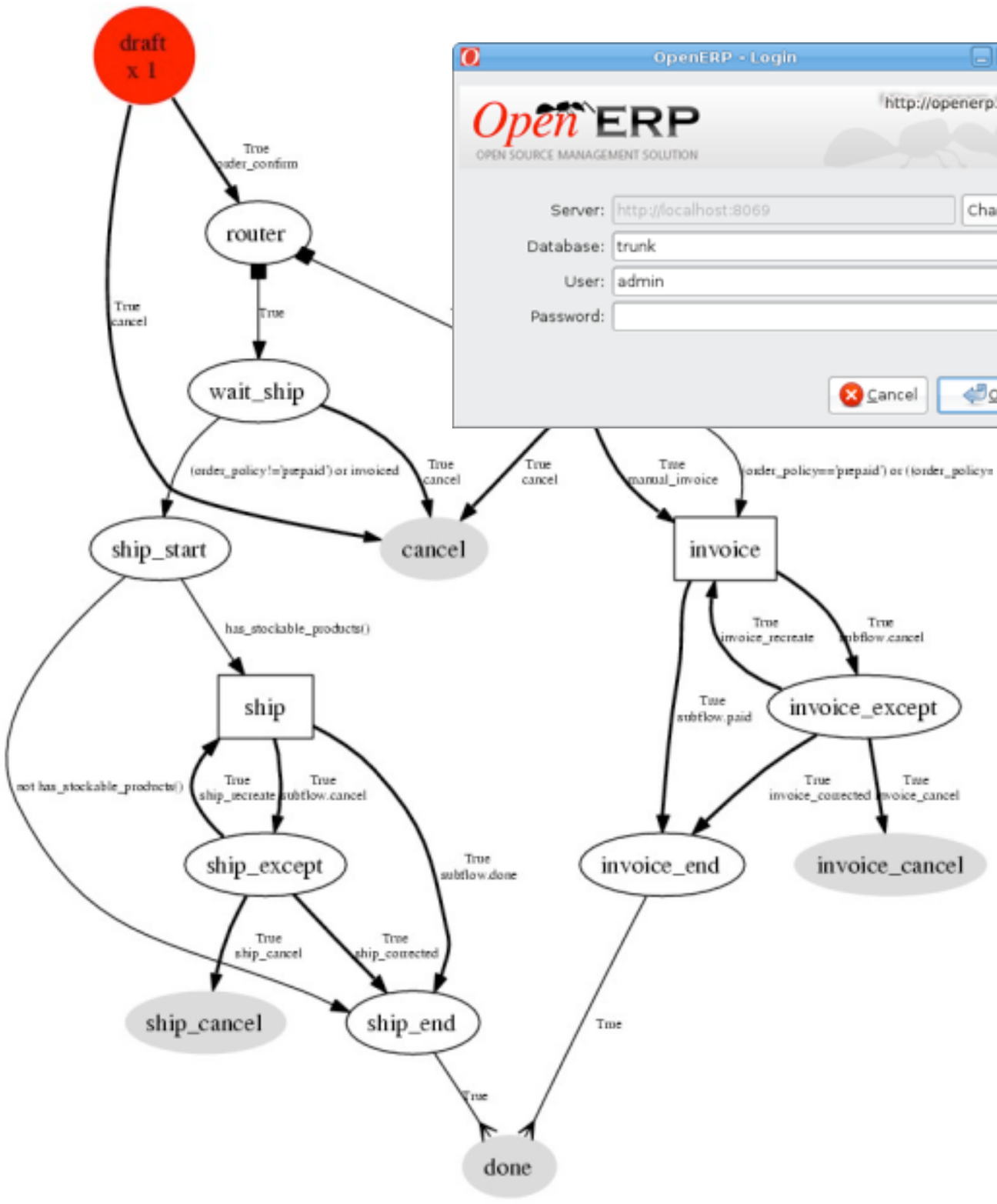
# Before next week

- Please download the PDF version of this book:
- http://scg.unibe.ch/download/oorp/


New Project
Steps
1. Choose Project
2. ...
Choose Project
Categories:
Android
Java
Java Web
Java EE
Maven
PHP
Ruby
Groovy
C/C++
NetBeans Modules
Samples
Projects:
Enterprise Application
Enterprise Application with Existing Sources
EJB Module
EJB Module with Existing Sources
Enterprise Application Client
Enterprise Application Client with Existing Sources
Packaged Archive
Description:
Creates a new enterprise application in a standard project. You can also create an EJB module project and Web application project in the enterprise application. A standard project uses an IDE-generated Ant build script to build and run your projects.
Help
< Back
Next >
Finish
Cancel

I want to be able to send e-invoices to mobile phone users!



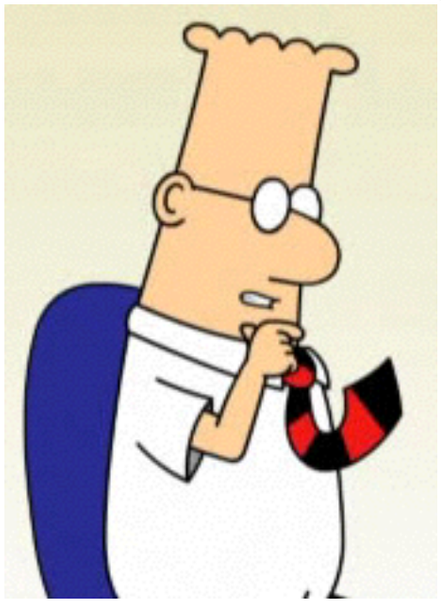


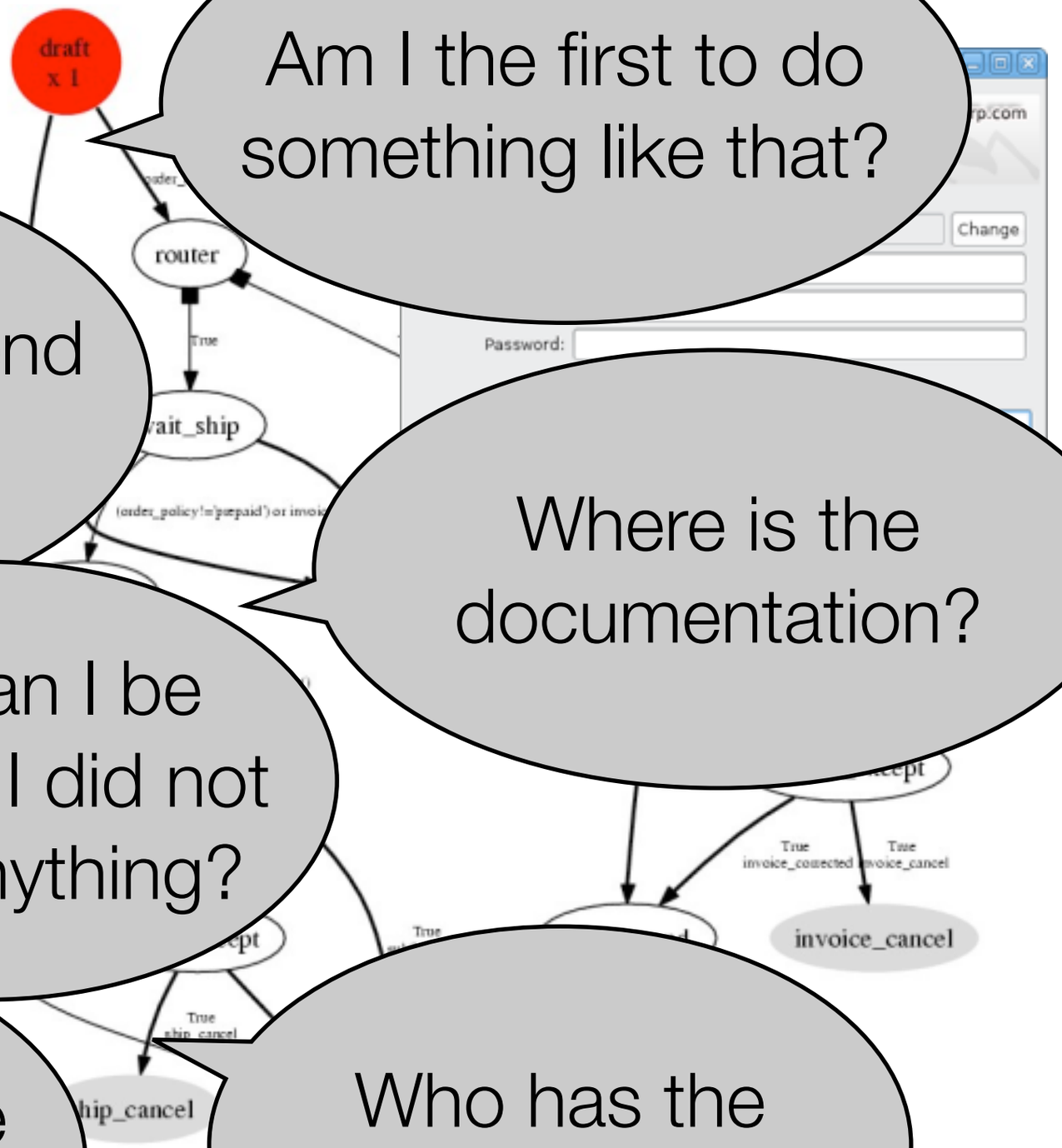

http://doc.openerp.com/developer/2_4_module_development/4_1_module_structure.html

How was the ERP system designed?

How can I extend the ERP?

Am I the first to do something like that?






How can I be sure that I did not break anything?

Where is the documentation?

How do I release my new feature?

Who has the knowledge?

http://doc.openerp.com/developer/2_4_module_development/4_1_module_structure.html

# Introduction

# Research on Software Evolution

Google scholar | software evolution | Search | Advanced Scholar Search | Scholar Preferences

**Scholar** | Articles and patents | anytime | include citations | Results **1** - **10** of about **2,080,000**. (**0.15** sec)

Architecture-based runtime **software evolution** — psu.edu [PDF]
P Oreizy, N Medvidovic, RN Taylor - … conference on **Software** …, 1998 - portal.acm.org
ABSTRACT Continuous availability is a critical requirement for an important class of **software** systems. For these systems, runtime system **evolution** can mitigate the costs and risks associated with shutting down and restarting the system for an update. We present an architecture- ...
Cited by … - Related articles - BL Direct - All 19 versions - Import into BibTeX

[PDF] Programs, life cycles, and laws of **software evolution** — bth.se [PDF]
MM Lehman… - Proceedings of the IEEE, 1980 - ipd.bth.se
PROCEEDINGS OF THE IEEE, VOL. 68, NO. 9, SEPTEMBER 1980 Programs, Life Cycles, and Laws of **Software Evolution** MEIR M. LEHMAN, senior member, ieee Abstract—By classifying programs according to their relationship to the environment in which they are executed ...
Cited by 540 - Related articles - All 2 versions - Import into BibTeX

[PDF] **Evolution** in … **software** … — psu.edu [PDF]
MW Godfrey, Q Tu - 16th IEEE International Conference on **Software** …, 2000 - Citeseer
Most studies of **software evolution** have been performed on systems developed within a single company using tradi- tional management techniques. With the widespread avail- ability of several large **software** systems that have been de- veloped using an "open source" development ...
Cited by 335 - Related articles - View as HTML - All 47 versions - Import into BibTeX

Metrics and laws of **software evolution**-the nineties view — psu.edu [PDF]
… , JF Ramil, PD Wernick, DE Perry, … - … **Software** Metrics …, 1997 - doi.ieeecomputersociety.org
Imperial College of Science, Technology and Medicine London SW7 2BZ tel: +44 (0)171 594 8214 fax: +44 (0) 17 1 594 82 15 e-mail: (mml,jcfl,pdwl)@doc.ic.ac.uk URL:http:Nwww-dse.doc.ic.ac.uk/-mml/feast1/ … M Turski Institute of Informatics Warsaw University Warsaw ...
Cited by … - Related articles - All 36 versions - Import into BibTeX

**Software** maintenance and **evolution**: a roadmap — psu.edu [PDF]
KH Bennett, VT Rajlich - … of the conference on The future of **Software** …, 2000 - portal.acm.org
Keith Bennett has been a full Professor since 1986, and a former Chair, both within the Department of Computer Science at the University of Durham. For the past fourteen years he has worked on methods and tools for program comprehension and reverse engineering, based on …
Cited by 263 - Related articles - All 22 versions - Import into BibTeX

Laws of **software** … **evolution** revisited — psu.edu [PDF]
MM Lehman - Lecture Notes in Computer Science, 1996 - Springer
Abstract. Data obtained during a 1968 study of the **software** process [8] led to an investigation of the **evolution** of OS/360 [13] and and, over a period of twenty years, to formulation of eight Laws o] So]tware Evo- lution. The FEAST project recently initiated (see sections 4 - 6 ...
Cited by 237 - Related articles - BL Direct - All 32 versions - Import into BibTeX

# Research on Software Evolution

Google scholar software aging Search Advanced Scholar Search Scholar Preferences

Scholar Articles and ... anytime include citations Results 1 - 10 of about 1,230,000. (0.15 sec)

**Software aging**
DL Parnas - ... of the 16th international conference on **Software** ..., 1994 - portal.acm.org
ABSTRACT Programs, like people, get old. We can 't prevent **aging**, but we can understand its causes, take steps to limits its effects, temporarily reverse some of the damage it has caused and prepare for the day when the **software** is no longer viable. A sign that the **Software**
Cited by 460 - Related articles - All 12 versions - Import into BibTeX

A methodology for ... **software aging** ncsu.edu [PS]
S Garg, A van Moorsel, K ... - ... on **Software** ..., 1998 - doi.ieeecomputersociety.org
Sachin Garg , Aad van Moorsel Lucent Technologies Bell Laboratories 600 Mountain Avenue Murray Hill, NJ 07974, USA f sgarg,aad g @research.bell-labs.com ... Kalyanaraman Vaidyanathan, Kishor S. Trivedi Center for Advanced Computing & Communication
Cited by 142 - Related articles - All 8 versions - Import into BibTeX

[PDF] Proactive management of **software aging** psu.edu [PDF]
V Castelli, RE Harper, P Heidelberger, SW ... - IBM Journal of ..., 2001 - Citeseer
**Software** failures are now known to be a dominant source of system outages. Several studies and much anecdotal evidence point to "**software aging**" as a common phenomenon, in which the state of a **software** system degrades with time. Exhaustion of system resources, data ...
Cited by 165 - Related articles - View as HTML - BL Direct - All 12 versions - Import into BibTeX

Modeling and analysis of **software aging** and rejuvenation psu.edu [PDF]
KS Trivedi, K Vaidyanathan, K ... - ANNUAL ..., 2000 - doi.ieeecomputersociety.org
**Software** systems are known to suffer from outages due to transient errors. Recently, the phenomenon of "**software aging**", one in which the state of the **software** system de- grades with time, has been reported. To counteract this phe- nomenon,a proactive approach of fault management, ...
Cited by 78 - Related articles - BL Direct - All 18 versions - Import into BibTeX

# Concepts

**Legacy Systems**

**Agile processes**

Life Cycle

**Maintenance**

**Software Aging**

**Software Evolution**

**Reverse Engineering**

Program Comprehension

**Re-engineering**

Program Transformation

**Mining Software Repositories**

Metrics

# Research dimensions

**Empirical studies and validation**

**Basic research**

How do we model a software system and its evolution?

**Industrial, large scale systems**

**Software Evolution**

**Methods and tools-oriented research**

How do we support program comprehension and program transformation?

# Research on Software Evolution

- **Pioneers:**
  - Keith H. Bennett
  - Meir M. Lehman
  - Bennet P. Lientz, Lientz
  - David Lorge Parnas
  - Vaclav T. Raijlich
  - E. Burton Swanson Swanson
- **Some of the people in the community:**
  - Serge Demeyer, Universiteit Antwerpen (Belgium)
  - Stephane Ducasse, INRIA (France)
  - Harald C. Gall, Software Evolution and Architecture lab, University of Zürich (Switzerland)
  - Tudor Girba, did his Ph.D. at the Software Composition Group at University of Bern (Switzerland)
  - Michele Lanza, University of Lugano (Switzerland)
  - Tom Mens, Software Engineering Lab, Université de Mons (Belgium)
  - Oscar Nierstrasz, Software Composition Group, University of Bern.


Publish Date: March 10, 2008
Print ISBN: 3540764399

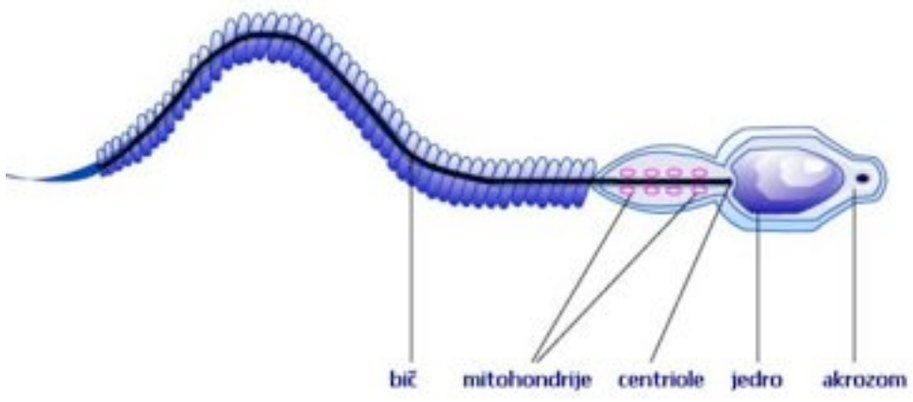
# History and Challenges of Software Evolution

- **1968**: first conference on Software Engineering, organized by the NATO Science Committee, with the goal to establish **sound engineering principles** in order to obtain reliable, efficient and economically viable software.
- **1970**: Royce proposes the **waterfall life-cycle process** for software development. Maintenance is seen as the final phase of the software lifecycle (with only bug fixes and minor adjustments). The model had a strong and long influence on the industrial practice of software development.
- **Late 1970's**: first attempt towards a more evolutionary process model. Identification of new activities, such as impact analysis and change propagation. In the same period, formulation of "**Laws of software evolution**" by Lehman.
- **1990's**: widespread acceptance of software evolution, formalization of evolutionary processes (Gilb's evolutionary development, Boehm's spiral model, Bennet and Rajich's staged model).
- **Software evolution is a crucial ingredient of agile software development (iterative and incremental development, embracing change!)**

Tom Mens, Introduction and Roadmap: History and Challenges of Software Evolution, Software Evolution, ISBN 978-3-540-76439-7

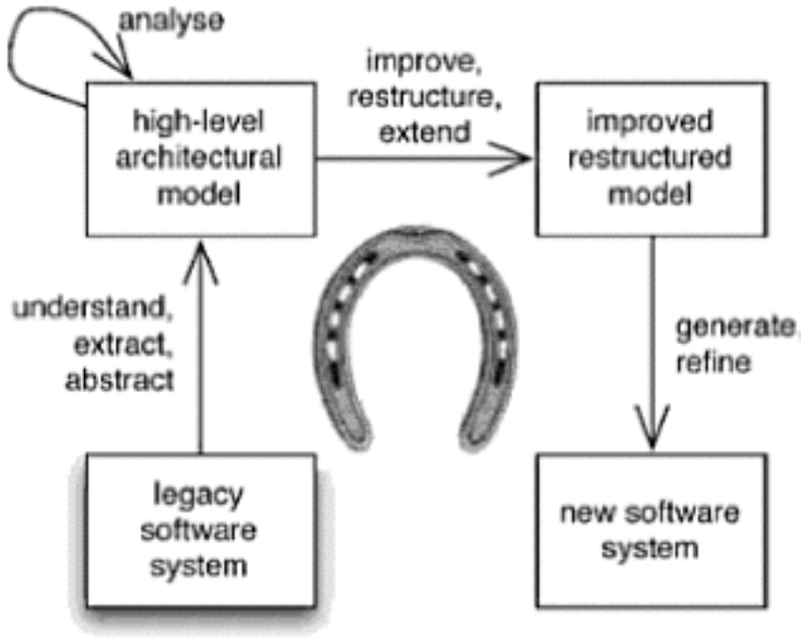
# History and Challenges of Software Evolution

- **Two dimensions of Software evolution**
  - **What and Why?** Software evolution as a **scientific discipline**, which studies the nature of the software evolution **phenomenon**, and seeks to understand its driving factor, its impact.
  - **How?** Software engineering as an **engineering discipline**, which studies more **pragmatic aspects** that aid software developers and project managers in their day-to-day tasks. Focus on **technology**, **methods**, **tools** and activities that provide a means to direct, implement and control software evolution.




# History and Challenges of Software Evolution

- **Reverse engineering**
  - Activity needed when **trying to understand the architecture or behavior** of a large software system, when the only reliable information is the **source code**.
  - Aims at **building higher-level, more abstract, software models** from the source code.
- **Re-engineering**
  - Activity needed when we are confronted with **legacy systems**. In other words, systems that are still **valuable**, but are **difficult to maintain**.
  - Aims at producing a new system that is more evolvable.




*Horseshoe model: reverse engineering, restructuring, forward engineering*

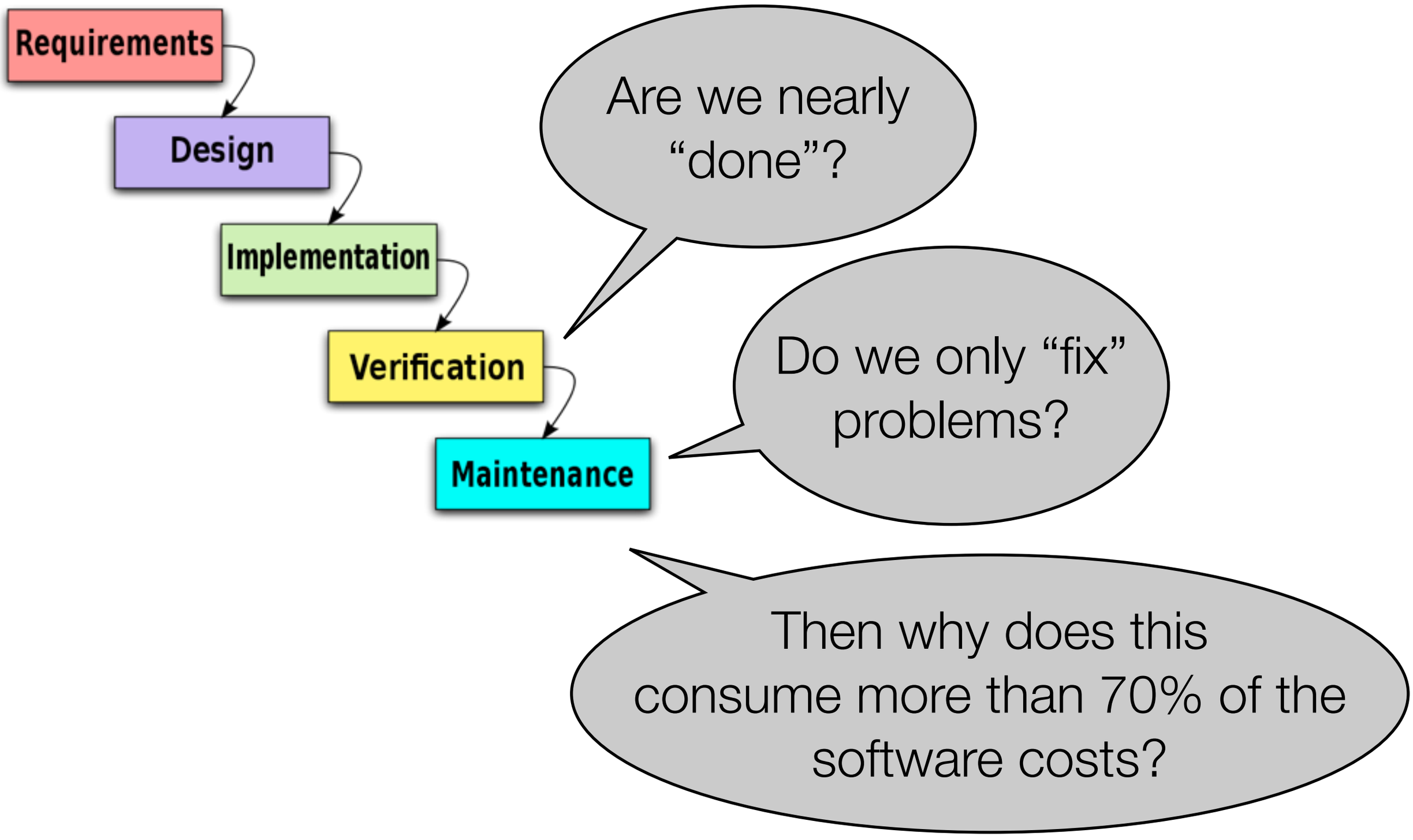
# Software Maintenance

- In many engineering disciplines, **physical products** are designed, built and delivered to users.
- Think about a **bridge**, a **building**, a **car** or a **vacuum cleaner**.
- Maintenance is about making sure that the product **stays operational** over the years.
- Since we are talking about physical products, we are thinking about fixing defects, changing **worn-out components**, do some cleaning, adding oil, etc.
- When we talk about maintenance, the **functionality of the product stays the same**.

http://www.flickr.com/photos/laenulfean/474217855/sizes/m/#cc_license

*Is software maintenance only about fixing defects?*

# What does it mean to “maintain” software?



# Software Maintenance

- Maintenance is **costly** - between 70% to 80% of the software costs are spent on maintenance.
- Classification of software maintenance activities:
  - **Corrective**: errors need to be fixed.
  - **Preventive**: prevent problems in the future (fix design issues).
  - **Adaptive**: something has changed in the environment.
  - **Perfective**: improve system qualities, e.g. performance.




Management Applications
H. Morgan Editor

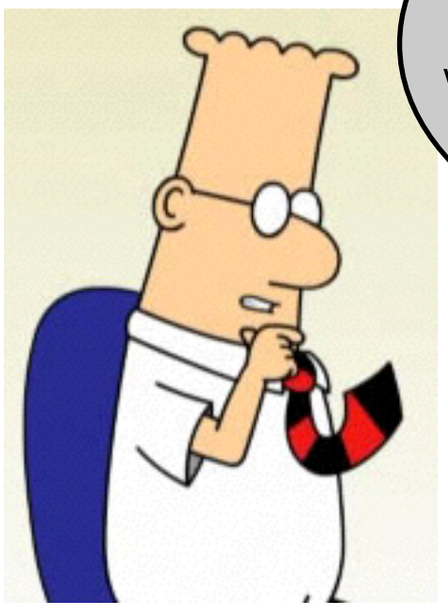
## Characteristics of Application Software Maintenance

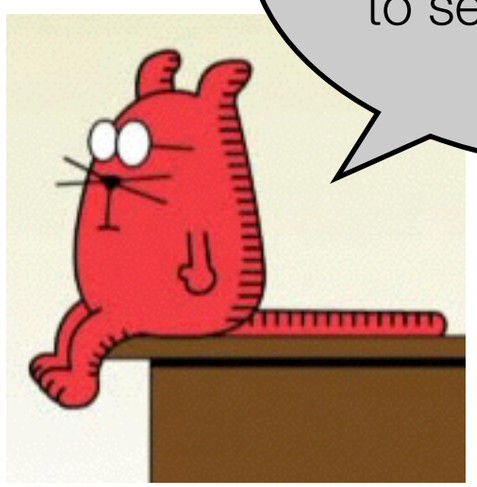
B. P. Lientz, E. B. Swanson,
and G. E. Tompkins
University of California at Los Angeles

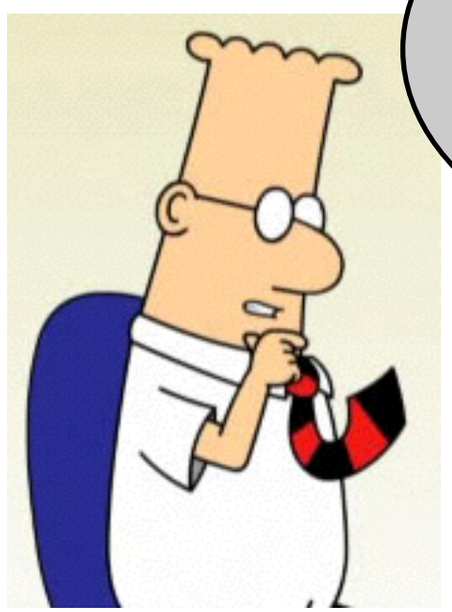
**Maintenance and enhancement of application software consume a major portion of the total life cycle cost of a system. Rough estimates of the total systems and programming resources consumed range as high as 75–80 percent in each category. However, the area has been given little attention in the literature. To analyze the problems in this area a questionnaire was developed and pretested. It was then submitted to 120 organizations. Respondents totaled 69. Responses were analyzed with the SPSS statistical package. The results of the analysis indicate that: (1) maintenance and enhancement do consume much of the total resources of systems and programming groups; (2) maintenance and enhancement tend to be viewed by management as at least somewhat more important than new application software development; (3) in maintenance and enhancement, problems of a management orientation tend to be more significant than those of a technical orientation; and (4) user demands for enhancements and extension constitute the most important management problem area.**

Communications of the ACM
June 1978 Volume 21 Number 6

# Software Maintenance

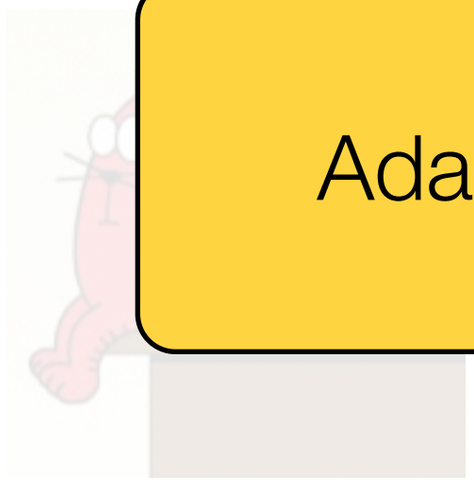



# Software Maintenance

Corrective

Preventive

Adaptive



Perfective

# Lehman's "Laws" of Evolution

- Propose a **theoretical model** to reason about software systems and their interaction with the socio-economic **environment.**
- Maintenance is costly, but maintenance is not only about bug fixing!
- Propose a classification of software: S-Programs, P-Programs and **E-Programs.**
- Conduct **quantitative studies** on large scale industrial projects, (collect metrics)
- Derive "**Laws of Evolution**" that are generally applicable.

THE TOTAL U.S. expenditure on programming in 1977 is estimated to have exceeded $50 billion, and may have been as high as $100 billion. This figure, which represents more than 3 percent of the U.S. GNP for that year, is already an awesome figure. It has increased ever since in real terms and will continue to do so as the microprocessor finds ever wider application. Programming effectiveness is clearly a significant component of national economic health. Even small percentage improvements in productivity can make significant financial impact. The potential for saving is large.

Economic considerations are, however, not necessarily the main cause of widespread concern. As computers play an ever larger role in society and the life of the individual, it becomes more and more critical to be able to create and maintain effective, cost-effective, and timely software. For more than two decades, however, the programming fraternity, and through them the computer-user community, has faced serious problems in achieving this [1]. As the application of microprocessors extends ever deeper into the fabric of society the problems will be compounded unless very basic solutions are found and developed.

"Programs, Life Cycles, and Laws of Software Evolution", Proceedings of the IEEE, Vol. 68, No. 9, September 1980.

# Different Types of Software Systems

- **S-Programs**. Programs that can be completely and formally specified.
  - e.g. a program that sorts an array.
- **P-Programs**. Programs that can be completely specified, but which makes an approximation of the real world.
  - e.g. a program that plays chess against a human player.
- **E-Programs**. Programs that mechanize a human or societal activity. The program becomes a part of the world it models!
  - e.g. an ERP system.

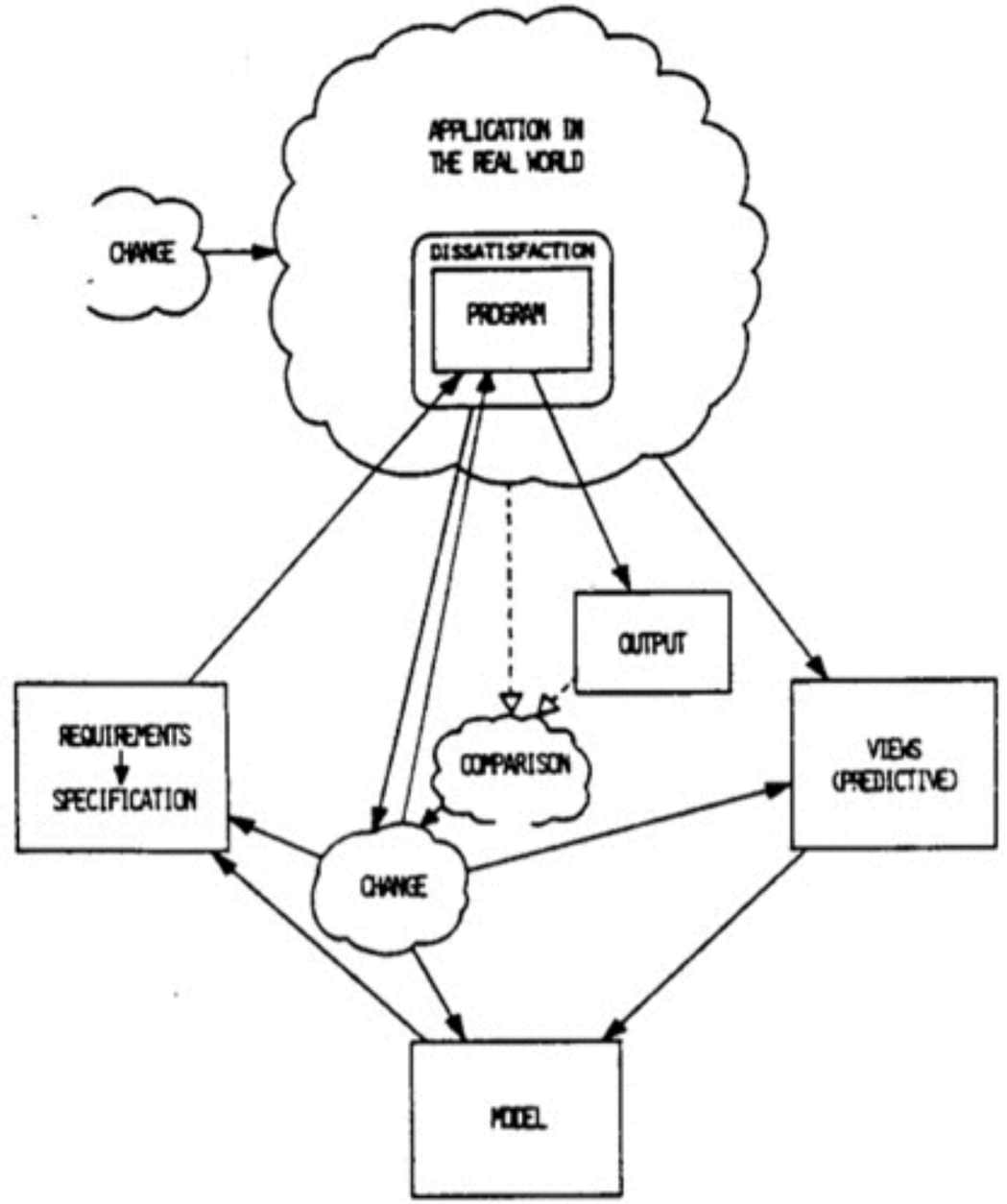
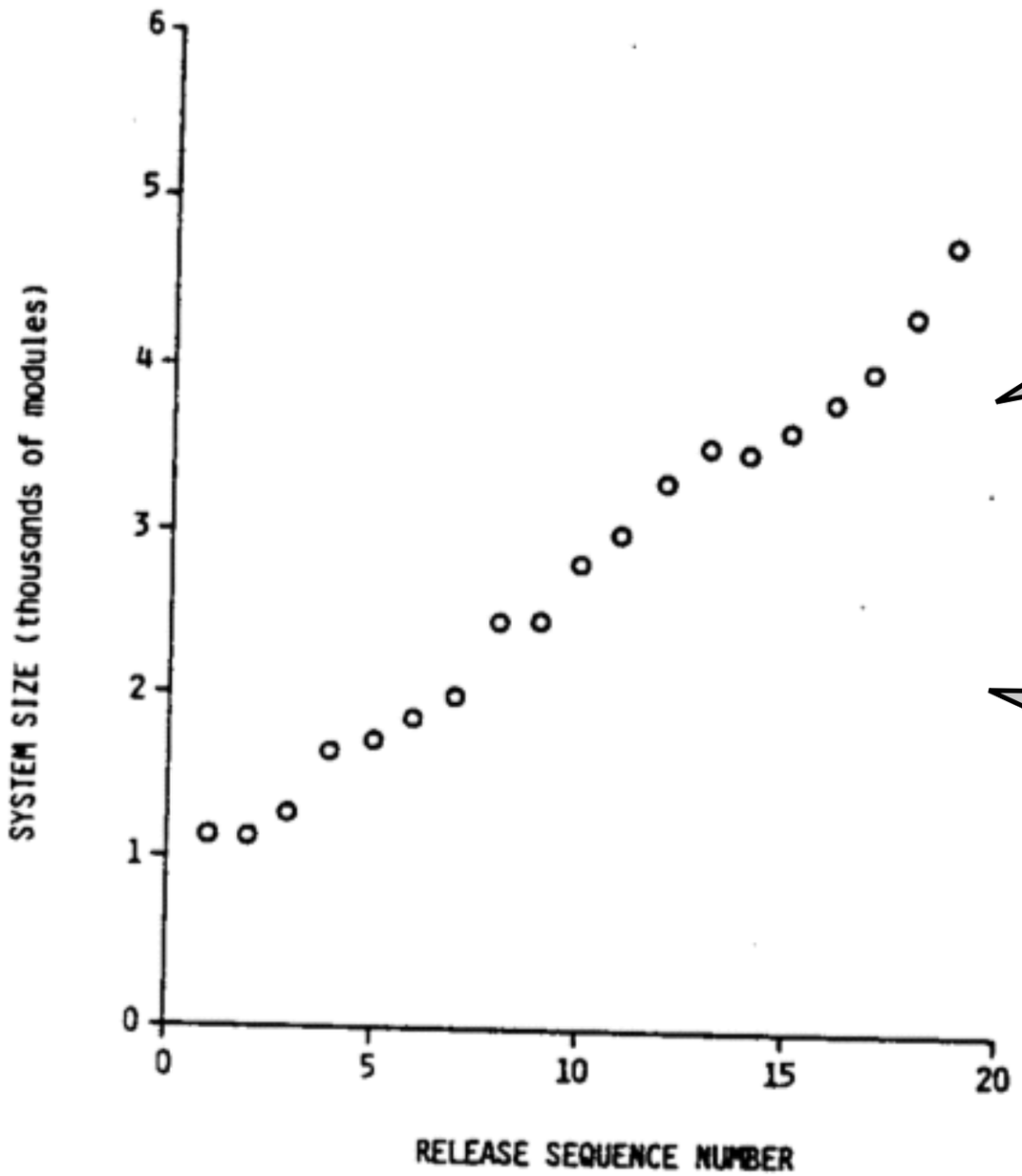



Fig. 4. *E*-programs.

# Quantitative Studies

- **Collection of data** on a large scale, industrial project (IBM OS 360)
- **Analysis over**
  - Elapsed time
  - Release numbers
- **Metrics**
  - System size (number of modules)
  - Incremental growth
  - Modules changed (indicates complexity)
  - Interval
- **Statistical analysis shows trends in metrics evolution.**

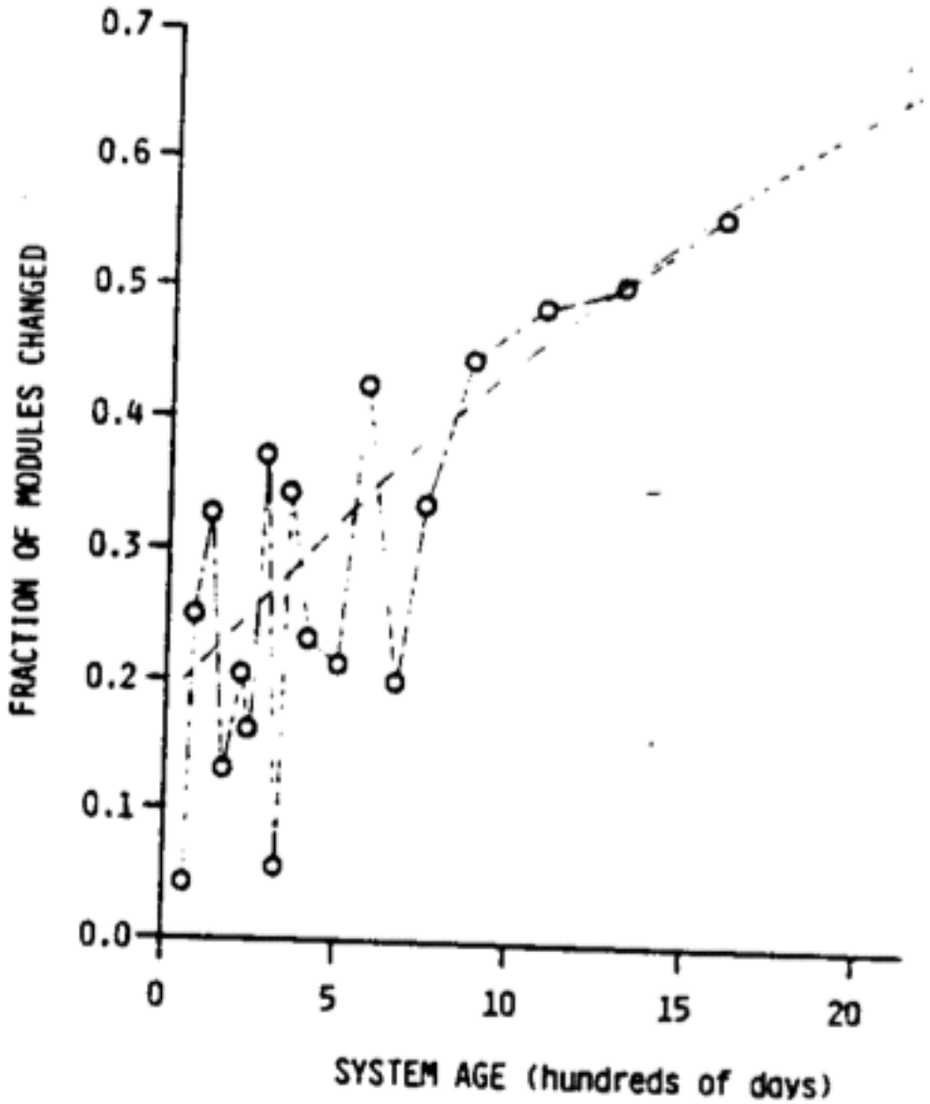
# Quantitative Studies




The system grows in a predictable way. This reflects the constant evolution!

So... we're not only fixing bug then?

# Quantitative Studies




The number of modules that must be changed for a release reflects the complexity of the system.

So... it becomes more and more complex then?

# Lehman's "Laws" of Evolution

Software systems have to **evolve**, otherwise they gradually become useless.

If you don't take specific actions, software will become more **complex** and more difficult to adapt.

TABLE I
LAWS OF PROGRAM EVOLUTION

I. *Continuing Change*

A program that is used and that as an implementation of its specification reflects some other reality, undergoes continual change or becomes progressively less useful. The change or decay process continues until it is judged more cost effective to replace the system with a recreated version.

II. *Increasing Complexity*

As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it.

III. *The Fundamental Law of Program Evolution*

Program evolution is subject to a dynamics which makes the programming process, and hence measures of global project and system attributes, self-regulating with statistically determinable trends and invariances.

IV. *Conservation of Organizational Stability (Invariant Work Rate)*

During the active life of a program the global activity rate in a programming project is statistically invariant.

*Conservation of Familiarity (Perceived Complexity)*

During the active life of a program the release content (changes, additions, deletions) of the successive releases of an evolving program is statistically invariant.

# Lehman's "Laws" of Evolution (nineties view)

If you don't take specific actions, software **quality** will decline!

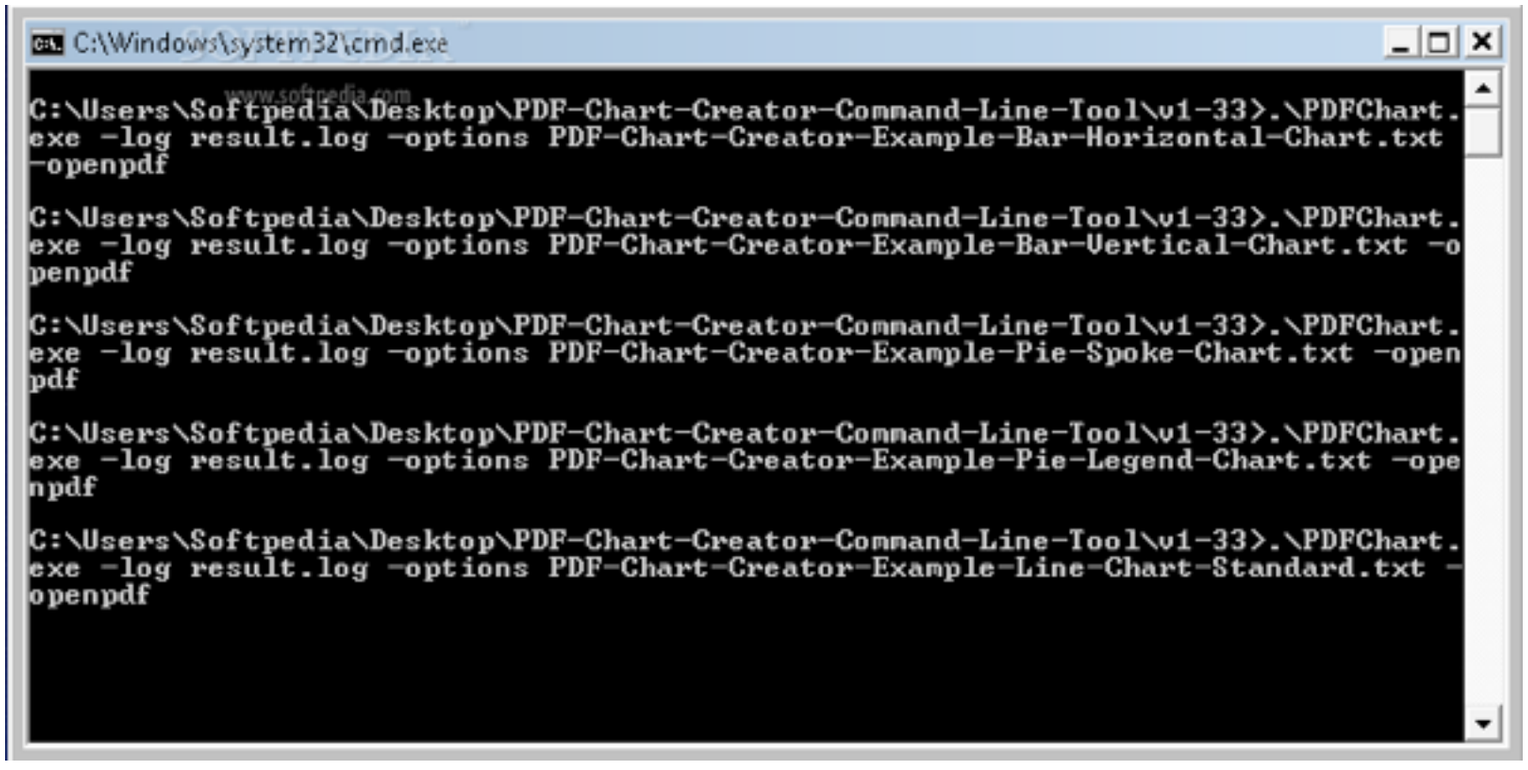
| No. | Brief Name | Law |
|---|---|---|
| I 1974 | Continuing Change | *E*-type systems must be continually adapted else they become progressively less satisfactory. |
| II 1974 | Increasing Complexity | As an *E*-type system evolves its complexity increases unless work is done to maintain or reduce it. |
| III 1974 | Self Regulation | *E*-type system evolution process is self regulating with distribution of product and process measures close to normal. |
| IV 1980 | Conservation of Organisational Stability (invariant work rate) | The average effective global activity rate in an evolving *E*-type system is invariant over product lifetime. |
| V 1980 | Conservation of Familiarity | As an *E*-type system evolves all associated with it, developers, sales personnel, users, for example, must maintain mastery of its content and behaviour [leh80a] to achieve satisfactory evolution. Excessive growth diminishes that mastery. Hence the average incremental growth remains invariant as the system evolves. |
| VI 1980 | Continuing Growth | The functional content of *E*-type systems must be continually increased to maintain user satisfaction over their lifetime. |
| VII 1996 | Declining Quality | The quality of *E*-type systems will appear to be declining unless they are rigorously maintained and adapted to operational environment changes. |
| VIII 1996 | Feedback System (first stated 1974, formalised as law 1996) | *E*-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable base. |

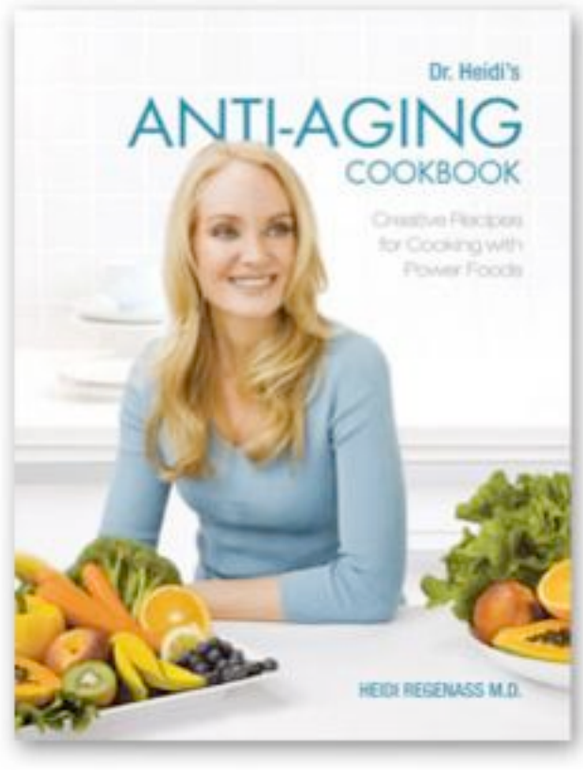
# Software Aging

*“Programs, like people, get old. We can’t prevent aging, but we can understand its causes, take steps to limit its effects, temporarily reverse some of the damage it has caused, and prepare for the day when the software is no longer viable.”*

David Lorge Parnas, 1994

# Software Aging



# The causes of software aging

- **Lack of movement**
  - Caused by the failure of the product's owners to modify it to meet **changing needs**.
  - Unless software is frequently updated, its users will **become dissatisfied** and they will change to a new product as soon as the benefits outweigh the costs of retraining and converting.
- **Ignorant surgery**
  - Caused by the **changes** that are made to software.
  - Changes made by people who do not understand the original design concept almost always cause the **structure of the program to degrade**.
  - Software that has been repeatedly modified in this way becomes very expensive to update.

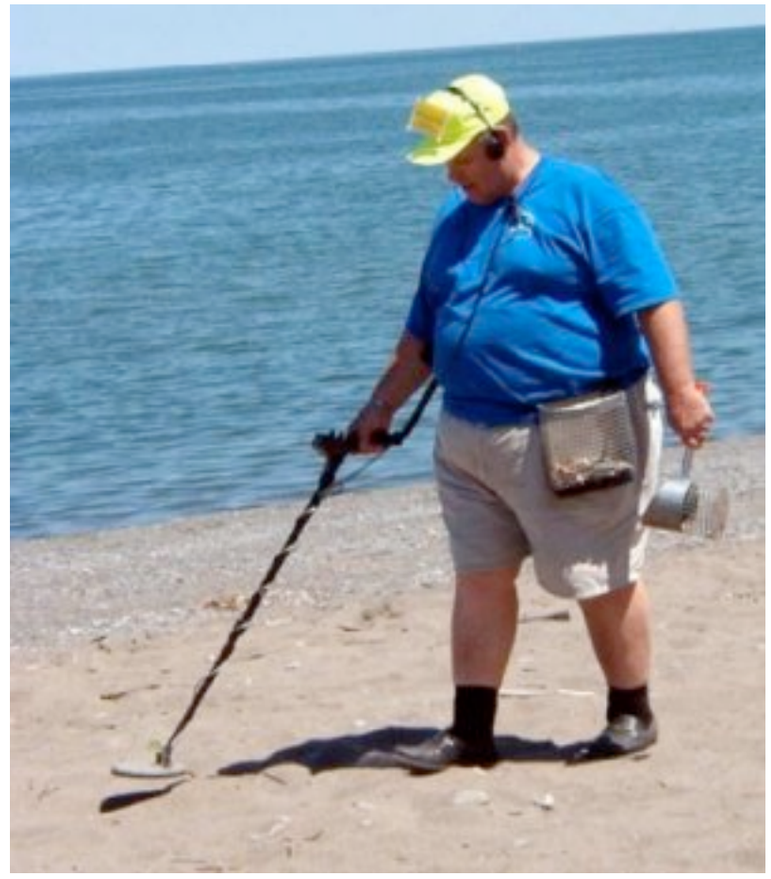
# The costs of software aging

- The **symptoms** of software aging mirror those of human aging:
  - Owners of aging software find it increasingly **hard to keep up** with the market and lose customers to newer products (“weight gain”)
  - Aging software often **degrades in its performance** as a result of a gradually deteriorating structure.
  - Aging software often becomes “**buggy**” because of errors introduced when changes are made.

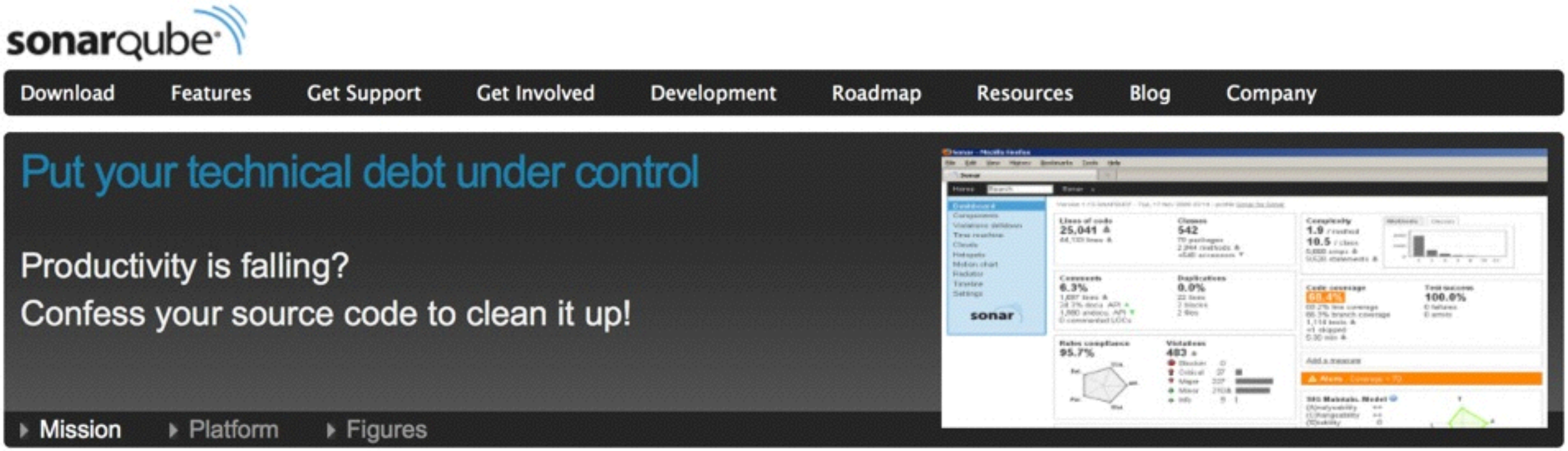
# Reducing the costs of software aging

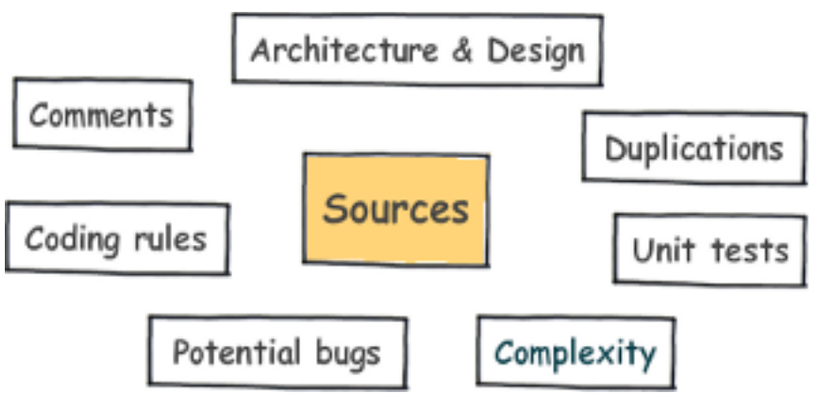
- **Preventive medicine**
  - **What can we do to delay the decay and limit its effects?**
  - Design for change
  - Keep records - documentation
  - Second opinion - reviews
- **Software geriatrics**
  - **What can we do to treat software aging that has already occurred?**
  - Stopping the deterioration
  - Retroactive documentation
  - Retroactive incremental modularization
  - Amputation
  - Major surgery - restructuring

# Software evolution & code quality in practice



# Tool: SonarQube





http://www.sonarqube.org/

# SonarQube: technical debt

- Concept invented by Ward Cunningham, also described by Martin Fowler
  - http://www.youtube.com/watch?v=pqeJFYwnkjE
  - http://martinfowler.com/bliki/TechnicalDebt.html

*You have a piece of functionality that you need to add to your system.*

***You see two ways to do it**, one is **quick** to do but is **messy** - you are sure that it will make further changes harder in the **future**. The other results in a **cleaner** design, but will take **longer** to put in place.*

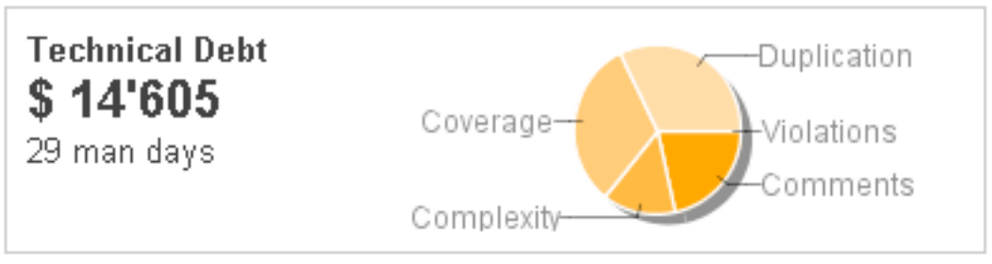
*Technical Debt is a wonderful **metaphor** to help us think about this problem. Doing things the quick and dirty way sets us up with a technical debt, which is **similar to a financial debt**.*

*Like a financial debt, the technical debt incurs **interest payments**, which come in the form of the extra effort that we have to do in future development because of the quick and dirty design choice. We can choose to continue paying the interest, or we can **pay down** the principal by **refactoring** the quick and dirty design into the better design. **Although it costs to pay down the principal, we gain by reduced interest payments in the future.***

# SonarQube: technical debt

- Concept invented by Ward Cunningham, also described by Martin Fowler
  - http://www.youtube.com/watch?v=pqeJFYwnkjE
  - http://martinfowler.com/bliki/TechnicalDebt.html

**Debt(in man days) =** cost_to_fix_duplications + cost_to_fix_violations + cost_to_comment_public_API + cost_to_fix_uncovered_complexity + cost_to_bring_complexity_below_threshold




***The metaphor also explains why it may be sensible to do the quick and dirty approach.***

*Just as a business incurs some debt to take advantage of a **market opportunity** developers may incur technical debt to hit an important deadline. The all too common problem is that development organizations let their debt get **out of control** and spend most of their future development effort paying crippling interest payments.*

**Duplications** = cost_to_fix_one_block * duplicated_blocks

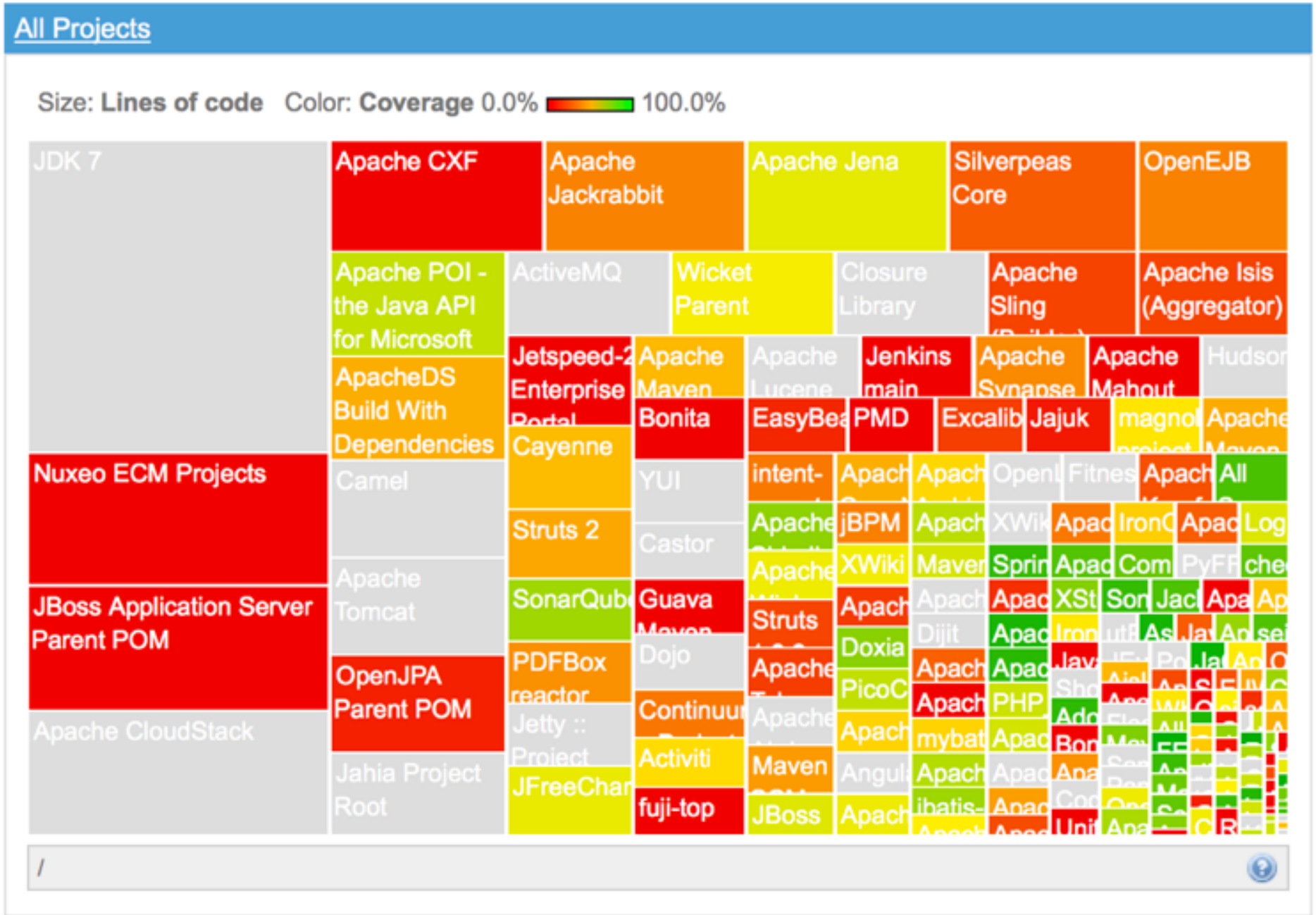
**Violations** = cost_to fix_one_violation * mandatory_violations

**Comments** = cost_to_comment_one_API * public_undocumented_api

**Coverage** = cost_to_cover_one_of_complexity * uncovered_complexity_by_tests (80% of coverage is the objective)

**Complexity** = cost_to_split_a_method * (function_complexity_distribution >= 8) + cost_to_split_a_class * (class_complexity_distribution >= 60)

# SonarQube on open source projects



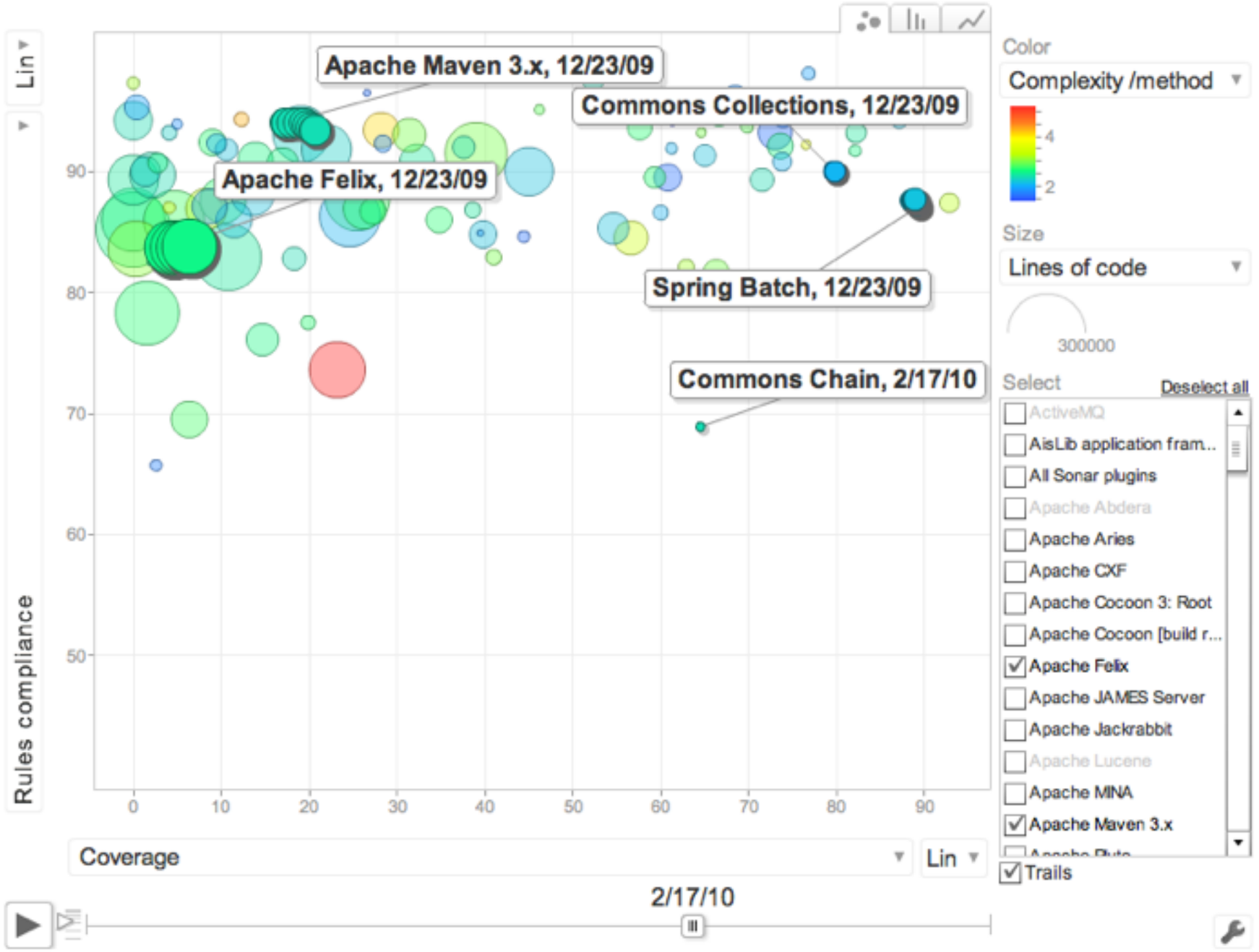

http://nemo.sonarqube.org/

# SonarQube on open source projects

| | Name | Lines of code | Technical Debt ratio | Coverage | Duplicated lines (%) | Build time |
|---|---|---|---|---|---|---|
| | AisLib application framework | 12,187 | 9.6% | 38.6% | 1.1% | 2010-03-13 |
| | Tapestry 5 Project | 57,213 | 6.7% | 51.8% | 0.2% | 2010-03-15 |
| | Commons Collections | 20,901 | 10.1% | 79.8% | 3.3% | 2010-03-14 |
| | MicroEmulator | 28,344 | 11.7% | | 2.3% | 2010-03-14 |
| | Apache Jackrabbit | 206,604 | 16.6% | 26.4% | 4.6% | 2010-03-14 |
| | Unitils | 18,585 | 14.9% | 2.8% | 3.8% | 2010-03-15 |
| | javagit | 6,127 | 11.1% | 61.2% | 0.3% | 2010-03-14 |
| | Wicket Parent | 104,703 | 9.3% | 44.7% | 1.1% | 2010-03-15 |
| | Commons Chain | 3,901 | 14.4% | 64.5% | 21.9% | 2010-03-14 |
| | Commons IO | 6,779 | 5.2% | 82.0% | 2.3% | 2010-03-14 |
| | Commons SCXML | 7,331 | 9.8% | 69.8% | 7.2% | 2010-03-14 |
| | Sonar | 31,558 | 6.2% | 68.6% | 0.0% | 2010-03-17 |

http://nemo.sonarsource.org/

# SonarQube on open source projects

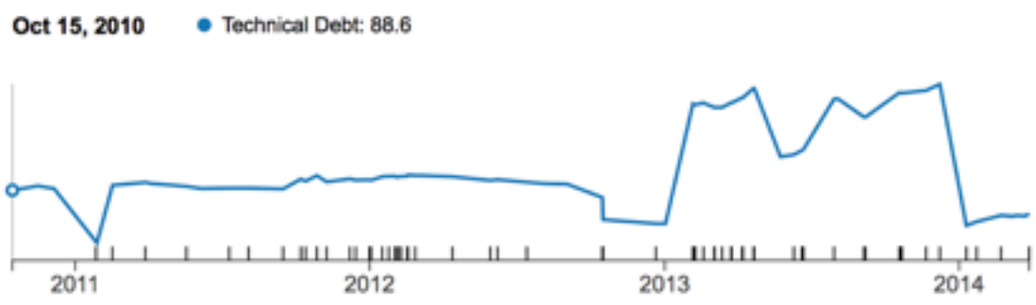



http://nemo.sonarqube.org/

# SonarQube on SonarQube

**Lines of code**
**101,386**
176,640 lines
37,155 statements

**Files**
**1,818**
459 directories
1,909 classes
12,382 functions
1,362 accessors

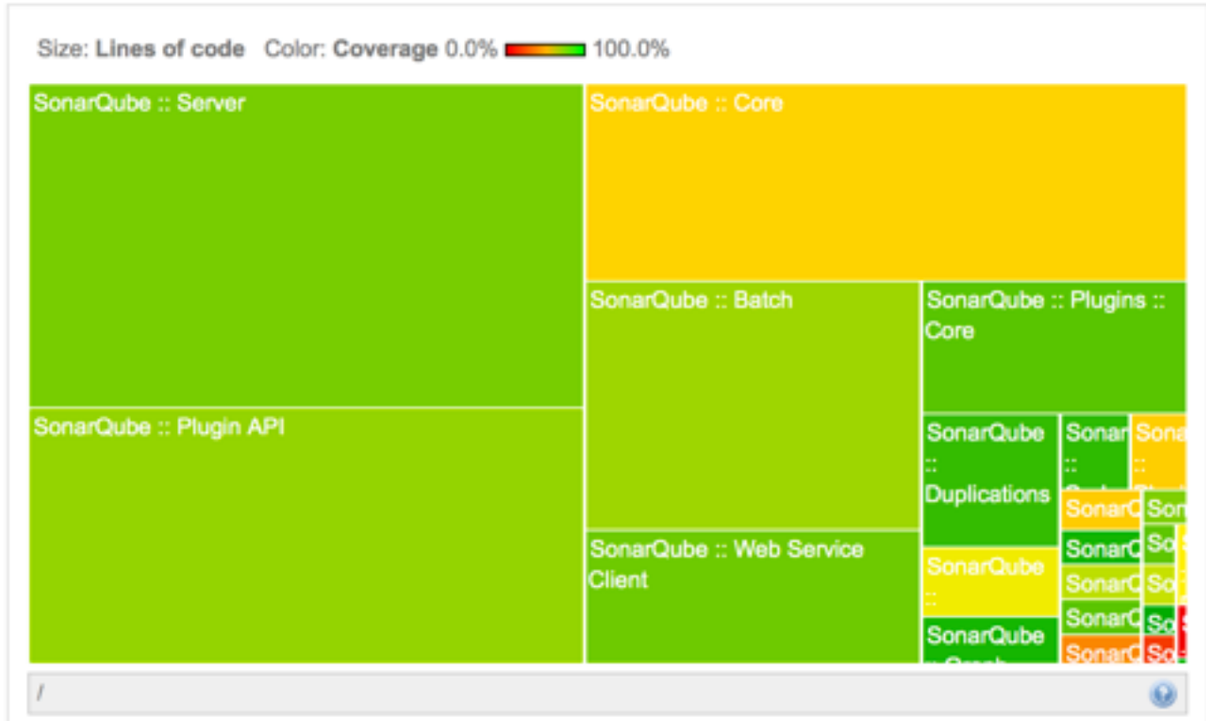



**Unit Tests Coverage**
**68.4%**
68.4% line coverage
68.4% branch coverage

**Unit test success**
**89.3%**
1 failures
521 errors
4,873 tests
35:30 min

**Documentation**
**25.5%** docu. API
9,063 public API
6,754 undocu. API

**Comments**
**7.3%**
7,942 lines




**Issues**
**1,226**

**Technical Debt**
**66.6** days

| | |
|---|---|
| Blocker | 2 |
| Critical | 58 |
| Major | 781 |
| Minor | 385 |
| Info | 0 |

⚠ **Alerts** : Unit tests success (%) < 100, Unit tests duration > 600000.

# SonarQube on SonarQube

**Severity**

| | |
|---|---|
| Blocker | 2 |
| Critical | 58 |
| Major | 781 |
| Minor | 385 |
| Info | 0 |

**Rule**

| | |
|---|---|
| Exception handlers should provide some context and preserve the original exception | 49 |
| Methods named "equals" should override Object.equals(Object) | 6 |
| System.exit(...) and Runtime.getRuntime().exit(...) should not be called | 1 |
| "equals(Object obj)" should be overridden along with the "compareTo(T obj)" method | 1 |
| String literals should not be duplicated | 1 |

| | |
|---|---|
| SonarQube :: Server | 22 |
| SonarQube :: Plugin API | 11 |
| SonarQube :: Core | 10 |
| SonarQube :: Testing Harness | 2 |
| SonarQube :: Web Service Client | 2 |
| SonarQube :: Batch | 1 |

| | |
|---|---|
| src/main/java/org/sonar/server/issue | 10 |
| src/main/java/org/sonar/api/utils | 4 |
| src/main/java/org/sonar/core/plugins | 2 |
| src/main/java/org/sonar/server/charts/deprecated | 2 |
| src/main/java/org/sonar/server/debt | 2 |
| src/main/java/org/sonar/core/persistence | 2 |

| | |
|---|---|
| InternalRubyIssueService.java | 10 |
| DateUtils.java | 2 |
| PluginClassloaders.java | 2 |
| LocalizedMessages.java | 2 |
| WebServiceEngine.java | 2 |
| DebtModelXMLExporter.java | 2 |

SonarQube / SonarQube :: Core

src/main/java/org/sonar/core/plugins/PluginClassloaders.java

Coverage | Duplications | **Issues** | Metrics | Source — Raw

**10** issues — Blocker: **0** — Critical: **2** — Major: **3** — Minor: **5** — Info: **0** — Technical Debt: **0.3 days**

Full source | Time changes... | Exception handlers should provide some context and preserve the original exception (2)

```
2011-06-10 simon.brandhof@gmail.com  211      throw new IllegalStateException("Plugin classloaders are not initialized");
                                     212    }
2010-10-15 mandrikov@gmail.com       213    try {
2010-11-24 mandrikov@gmail.com       214      return world.getRealm(key);
2010-10-15 mandrikov@gmail.com       215    } catch (NoSuchRealmException e) {
```

**Either log or rethrow this exception along with some contextual information.**

Open | Debt: 10 minutes | Author: mandrikov@gmail.com

```
                                     216      return null;
                                     217    }
                                     218  }
                                     219
```

# References

- **Mining Software Repositories community**
  - http://2014.msrconf.org/
  - http://msr.uwaterloo.ca/msr2010/
- **Tools**
  - **Sonar**: http://sonar.codehaus.org
  - **StatSVN**: http://www.statsvn.org
  - **Bugzilla**: http://www.bugzilla.org
- **Visualization**
  - http://code.google.com/apis/charttools
  - http://prefuse.org, http://flare.prefuse.org
  - http://processing.org

# Lab

# Tools

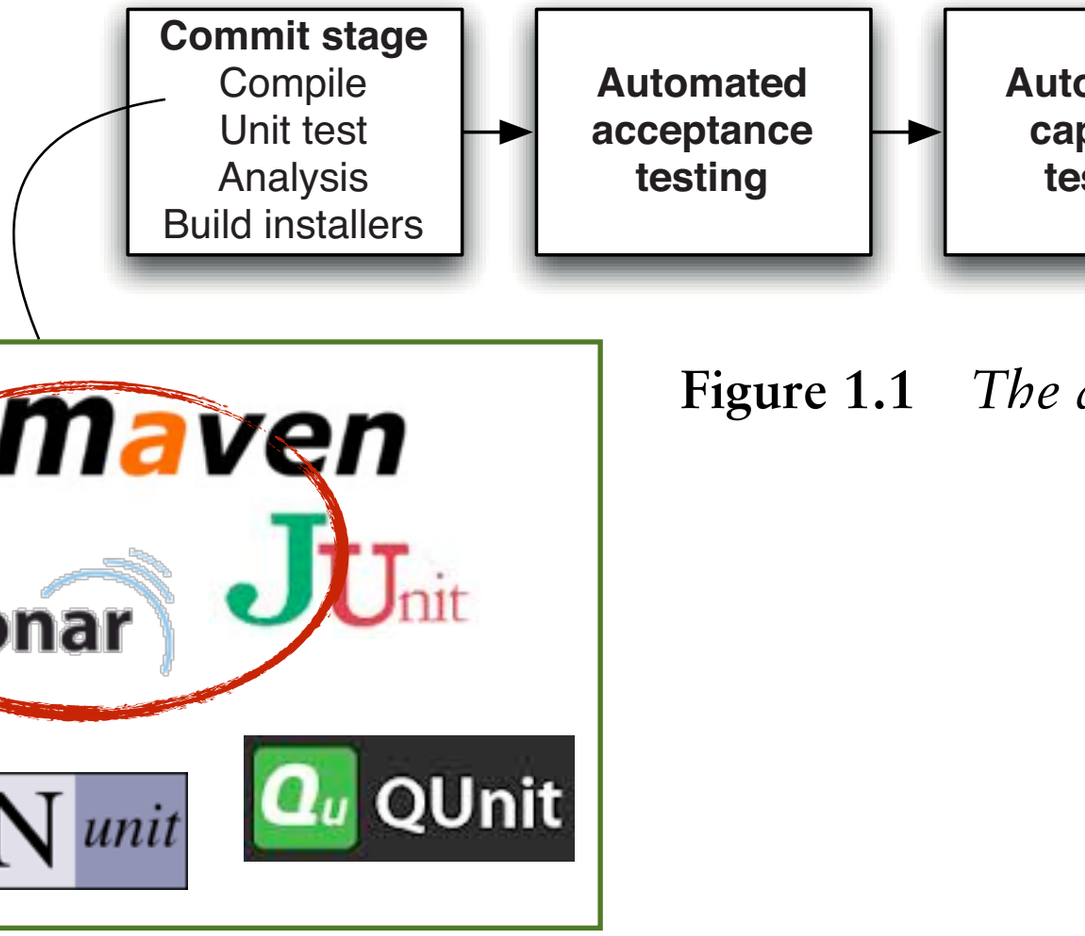
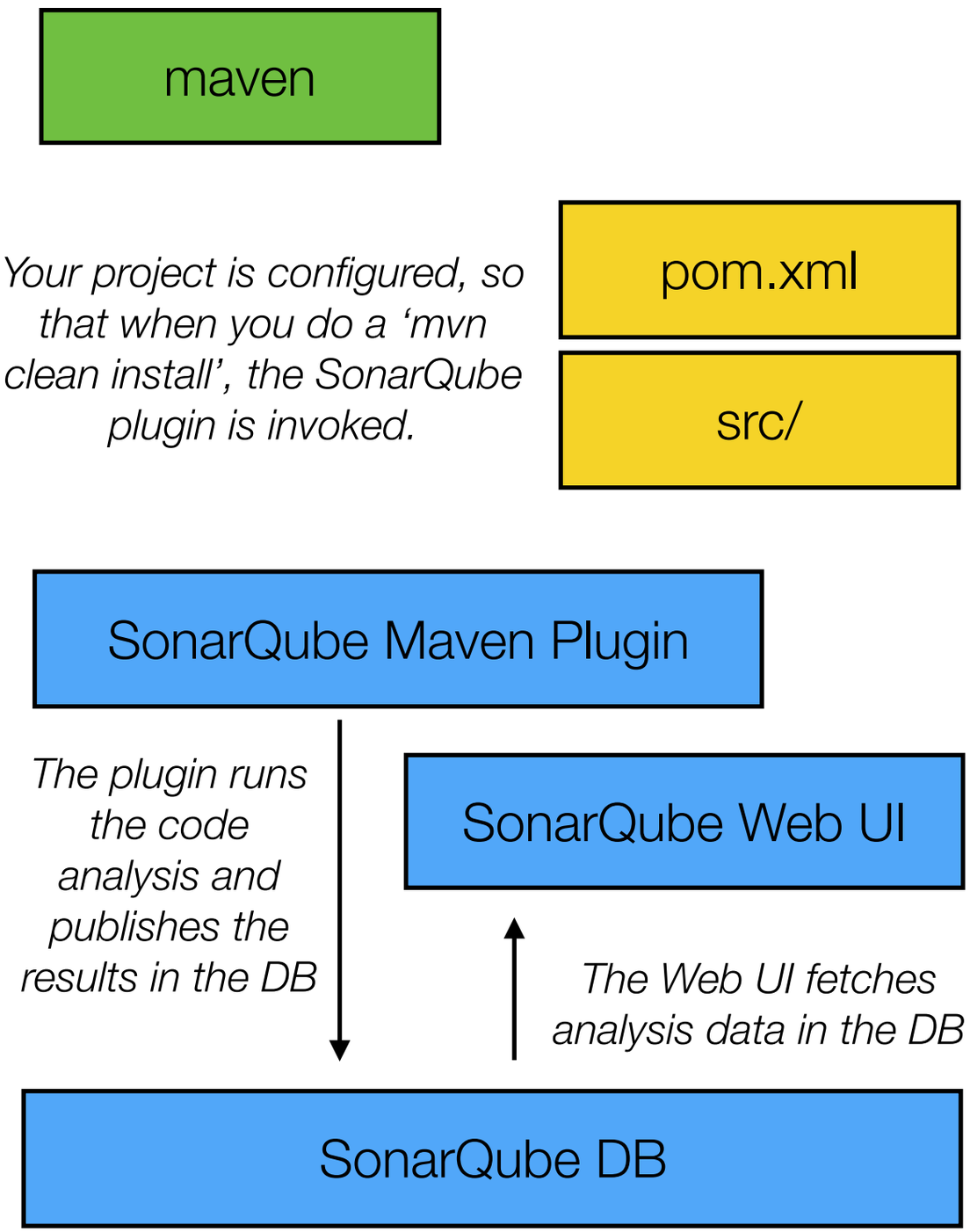




**Figure 1.1** *The deployment pipeline*

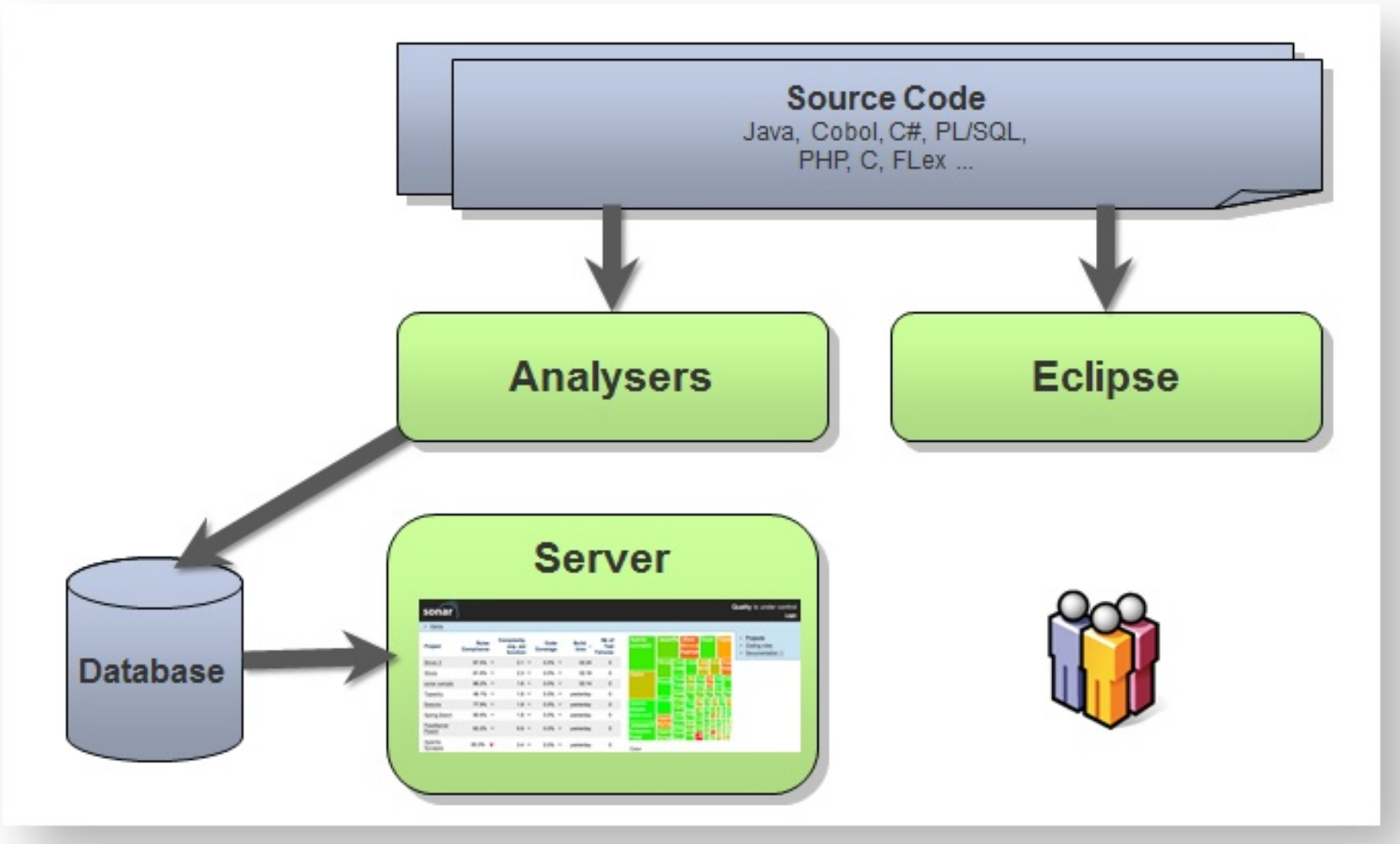



# Lab activities

- Firstly, we need a **sample project** to build, test and evaluate (“system under test”). We will use **apache maven** to drive the build process.
- Secondly, we need to **install SonarQube**.
- Thirdly, we need to **configure maven** (pom.xml file), so that the SonarQube analyser is invoked during the build process.
- Finally, we can see the results of the code analysis in the **SonarQube web UI** (dashboard).



http://docs.sonarqube.org/display/SONAR/Installing

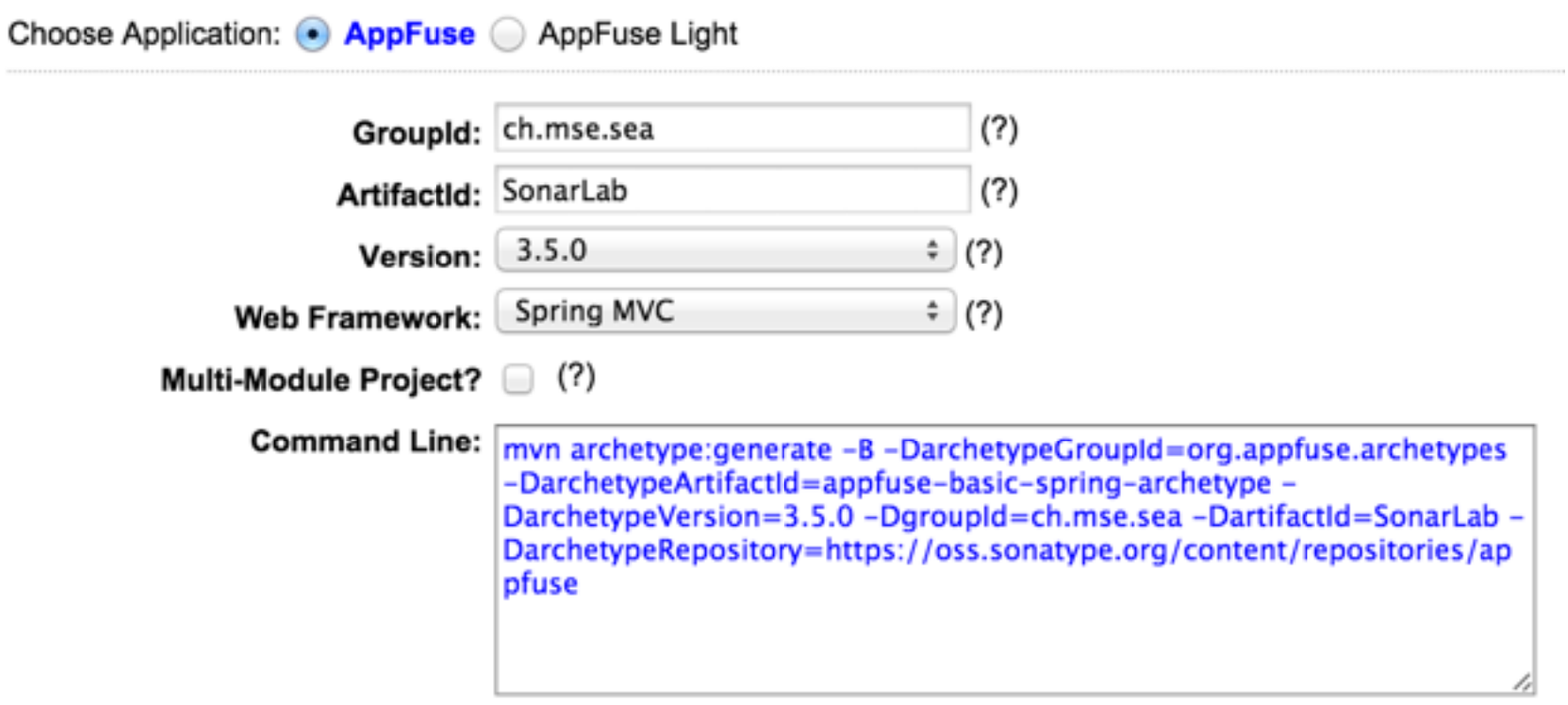


# Activity 1: Create a test project

- **You could use any project** (and you should try to use SonarQube for your semester projects!). Note that Java and maven are not mandatory!
- But let's use a nice feature of maven to create a sample project: **archetypes**:
  - A maven archetype is a **skeleton** for a certain type of project.
  - Archetype definitions can be **shared** via repositories.
  - **Creating you own archetype is a very efficient way to create a SDK.**
  - **AppFuse** is an open source project that uses archetypes as a way to pre-integrate several open source frameworks (with a layer on top of them).
- AppFuse is not a new framework. It was one of the first **scaffolding tools**, which have become very popular (e.g. Yeoman, Spring Boot, Java Hipster, etc.)

# Activity 1: Create a test project

- Create a directory for your project and move inside
- Visit http://appfuse.org/display/APF/AppFuse+QuickStart
- Fill out the form and copy the the content of the Command Line field. Paste it in your command line and execute it.
- Make it a git repository (`git init`, `git add *`, `git commit -m`)



# Activity 1: Create a test project

- You should be able to do a **`mvn clean install`** and to build your project.¨
- You should be able to do a **`mvn jetty:run`** to start the web app and to access it via http://localhost:8080/
- Notes:
  - This assumes that you have a MySQL server running on the standard port (3306) and that the root user can connect to it without password.
  - If that is not the case, then you need to adjust the maven configuration. Read “Changing database settings” in http://appfuse.org/display/APF/AppFuse+QuickStart.

Login | AppFuse

Olivier

localhost:8080/login

AppFuse Login

## Sign In

Username

Password

Remember Me

Login

Not a member? Signup for an account.

Forgot your password? Have your password hint e-mailed to you.

Request a password reset e-mailed to you.

Version 1.0-SNAPSHOT

Home | AppFuse

Olivier

localhost:8080/home

Do you want Google Chrome to save your password?

Never for this site | Save password

AppFuse | Home | Edit Profile | Administration | Logout

- View Users
- Current Users
- Reload Options
- Upload A File

# Welcome!

Congratulations, you have logged in su ve logged in, you have the following options:

- Edit Profile
- Upload A File

Version 1.0-SNAPSHOT | Logged in as: admin

© 2003-2015 Your Company Here

localhost:8080/home#

# Hints

- You can use different databases with the demo app and can use an in-memory app (H2):
  - See: http://raibledesigns.com/rd/entry/database_profiles_in_appfuse_2
  - Use the profile of your choice, such as: `mvn jetty:run -Ph2`
- You will certainly need to increase the memory:
  - `export MAVEN_OPTS="-Xmx1024M -XX:PermSize=512m"`
- Installation guide for SonarQube:
  - http://docs.codehaus.org/display/SONAR/Installing
  - http://docs.codehaus.org/display/SONAR/Installing+and+Configuring+Maven

# Activity 2: Install SonarQube

- The SonarQube server is implemented in Java. You could deploy it in a Java EE application server, but you can also run it with its **embedded server**.
- Installation is extremely simple. For this lab, we do not even have to setup a proper database and can live with the **embedded database (h2)**.
- Follow these instructions (only steps 1 and 2) to download and start SonarQube:
  - http://docs.sonarqube.org/display/SONAR/Setup+and+Upgrade
- You should be able to connect to the server via **http://localhost:9000**

Home | AppFuse | SonarQube | SonarQube | Olivier

localhost:9000

sonarqube | Dashboards | Issues | Measures | Rules | Quality Profiles | Quality Gates | More | Log in

Home

**Welcome to SonarQube Dashboard**

Since you are able to read this, it means that you have successfully started your SonarQube server. Well done!

If you have not removed this text, it also means that you have not yet played much with SonarQube. So here are a few pointers for your next step:

- Do you now want to run analysis on a project?
- Maybe start customizing dashboards?
- Or simply browse the complete documentation?
- If you have a question or an issue, please visit the Get Support page.

**PROJECTS**

| QG | NAME | VERSION | LOC | TECHNICAL DEBT | LAST ANALYSIS |
|---|---|---|---|---|---|
| No data | | | | | |

**PROJECTS**

No data

SonarQube™ technology is powered by SonarSource SA
Version 5.1 - LGPL v3 - Community - Documentation - Get Support - Plugins - Web Service API

# Activity 3: Analyze code during build process

- At this point, the SonarQube DB is empty (no project).
- There are different ways to feed the DB (different ways to trigger the code analysis). In this lab, we will use a **maven plugin provided by SonarQube**.
- This plugin is invoked by typing:
  - `mvn clean install`
  - `mvn sonar:sonar`
- The documentation is available here:
  - http://docs.sonarqube.org/display/SONAR/Analyzing+with+Maven
- When you have run the code analysis, go back to the SonarQube Web UI. You should now see your project!

Home | AppFuse | SonarQube | SonarQube | Analyzing Source Code - S | Olivier

localhost:9000/dashboard/?did=4

Do you want Google Chrome to save your password? Never for this site | Save password

sonarqube | Dashboards | Issues | Measures | Rules | Quality Profiles | Quality Gates | Settings | More | Administrator

Home

Configure widgets

**Welcome to SonarQube Dashboard**

Since you are able to read this, it means that you have successfully started your SonarQube server. Well done!

If you have not removed this text, it also means that you have not yet played much with SonarQube. So here are a few pointers for your next step:

- Do you now want to run analysis on a project?
- Maybe start customizing dashboards?
- Or simply browse the complete documentation?
- If you have a question or an issue, please visit the Get Support page.

**MY FAVOURITES**

| QG | NAME | LAST ANALYSIS |
|---|---|---|
| No data | | |

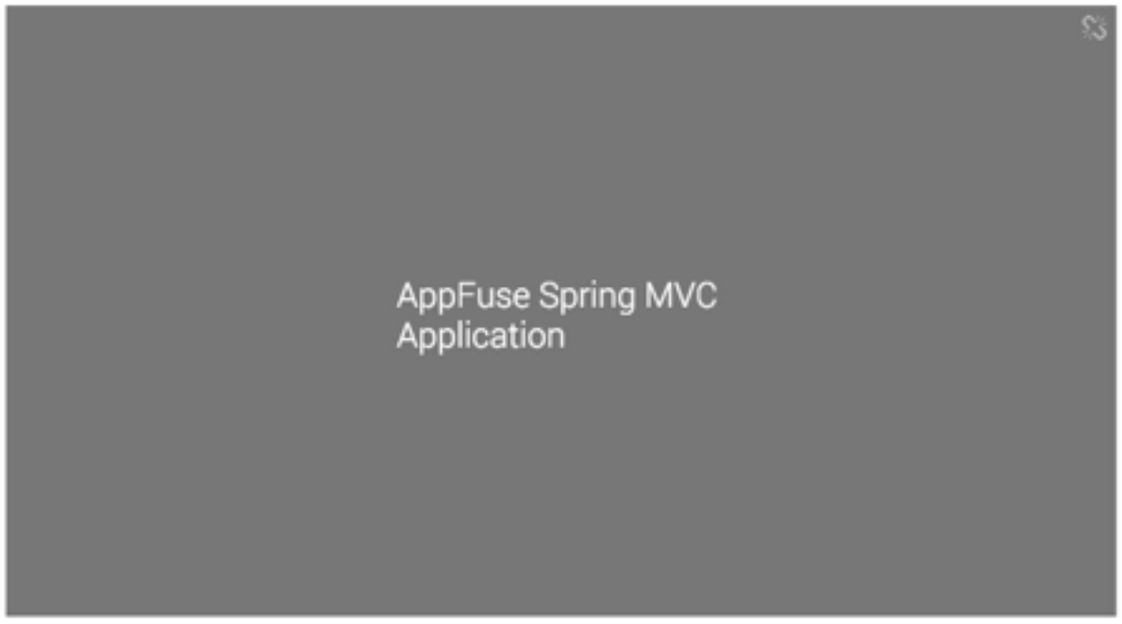
**PROJECTS**

| QG | NAME | VERSION | LOC | TECHNICAL DEBT | LAST ANALYSIS |
|---|---|---|---|---|---|
| ☆ | AppFuse Spring MVC Application | 1.0-SNAPSHOT | 1,717 | 2d 4h | 09:44 |

1 results

**PROJECTS**

Size: Lines of code Color: Coverage




SonarQube™ technology is powered by SonarSource SA

Home | AppFuse × SonarQube × SonarQube - AppFuse Spri × Analyzing Source Code - S × Olivier

localhost:9000/dashboard/index/196

sonarqube Dashboards Issues Measures Rules Quality Profiles Quality Gates Settings More Administrator

AppFuse Spring MVC Application

Version 1.0-SNAPSHOT / May 11 2015 9:44 AM

Overview Components Issues Settings More

Main Dashboard

Configure widgets

Lines Of Code
1,717
Java

Files
24
Directories 10
Lines 2,877

Functions
105
Classes 25
Statements 732
Accessors 20

SQALE Rating
A

Technical Debt Ratio
2.3%

Debt
2d 4h

Issues
135

| | |
|---|---|
| Blocker | 0 |
| Critical | 31 |
| Major | 79 |
| Minor | 24 |
| Info | 1 |

Duplications
0.0%
Lines 0
Blocks 0
Files 0

Directory Tangle Index
0.0%
Cycles
> 0

Dependencies To Cut
Between Directories 0
Between Files 0

Complexity
340
/Function 3.2
/Class 13.6
/File 14.2

60

Unit Tests Coverage
-

Unit Test Success
100.0%
Failures 0
Errors 0
Tests 28
Execution Time 29.1 sec

Home | AppFuse | SonarQube | AppFuse Spring MVC Appl | Analyzing Source Code - S | Olivier

localhost:9000/component_issues?id=ch.mse.demo%3ASonarDemo#resolved=false|severities=CRITICAL

sonarqube | Dashboards | Issues | Measures | Rules | Quality Profiles | Quality Gates | Settings | More | Administrator

# AppFuse Spring MVC Application

Version 1.0-SNAPSHOT / May 11 2015 9:44 AM

Overview | Components | Issues | Settings | More

**Severity**

| | | | |
|---|---|---|---|
| Blocker | 0 | Minor | 24 |
| Critical | 31 | Info | 1 |
| Major | 79 | | |

**Resolution**

| | | | |
|---|---|---|---|
| Unresolved | 31 | Fixed | 0 |
| False Positive | 0 | Won't fix | 0 |
| Removed | 0 | | |

Status
New Issues
Rule
Tag
Module
Directory
File
Assignee
Reporter
Author
Language
Action Plan

1 / 31 | Reload | New Search | Bulk Change

AppFuse Spring MVC Application src/main/java/ch/mse/demo/webapp/controller/BaseFormController.java

Make "List" serializable or don't store it in the session. — 2 minutes ago L80
Critical | Open | Not assigned | Not planned | 20min debt | Comment | bug

Make "List" serializable or don't store it in the session. — 2 minutes ago L92
Critical | Open | Not assigned | Not planned | 20min debt | Comment | bug

AppFuse Spring MVC Application src/main/java/ch/mse/demo/webapp/controller/PasswordHintController.java

Either log or rethrow this exception. — 2 minutes ago L92
Critical | Open | Not assigned | Not planned | 10min debt | Comment | error-handling

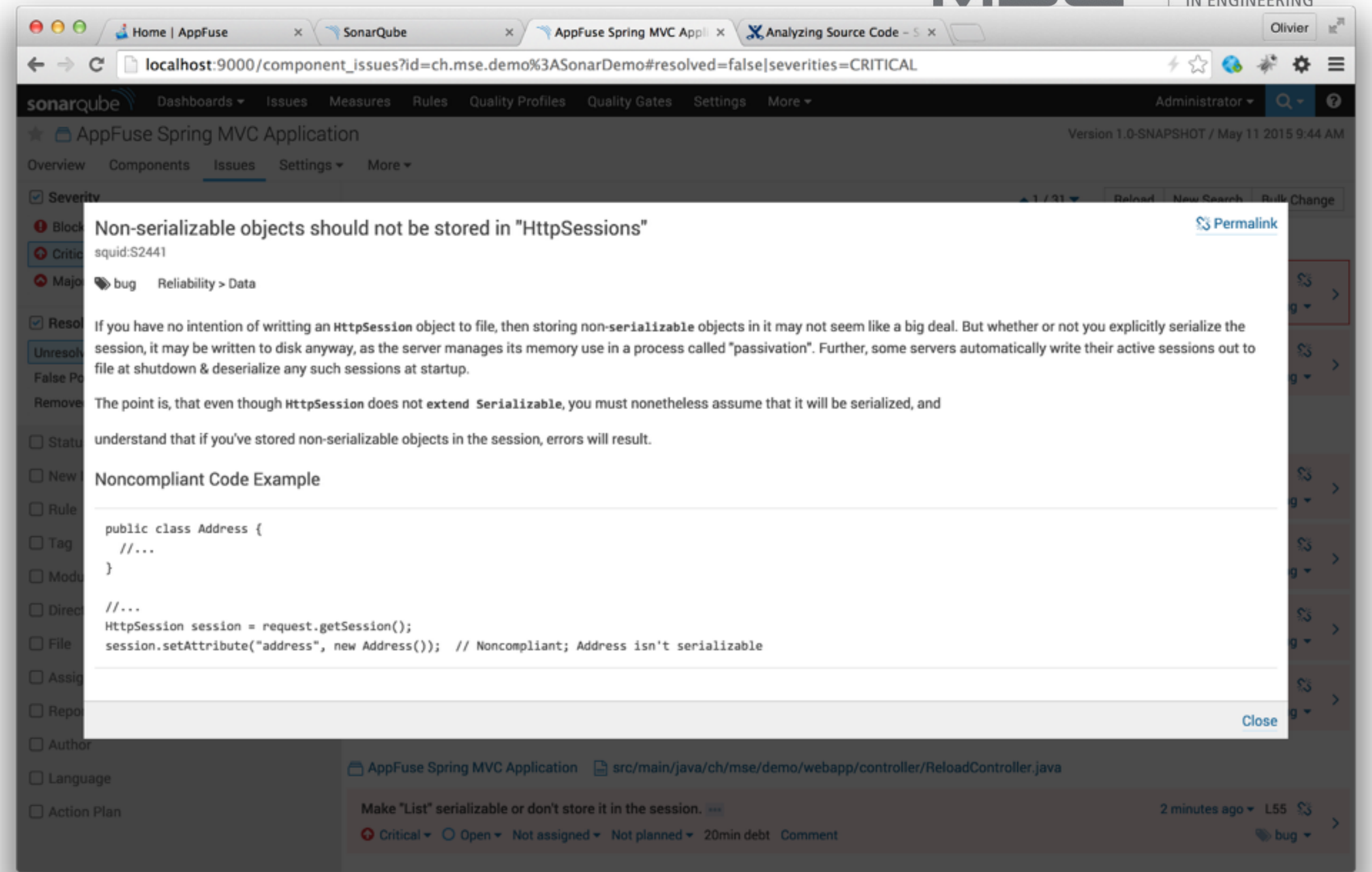
Either log or rethrow this exception. — 2 minutes ago L95
Critical | Open | Not assigned | Not planned | 10min debt | Comment | error-handling

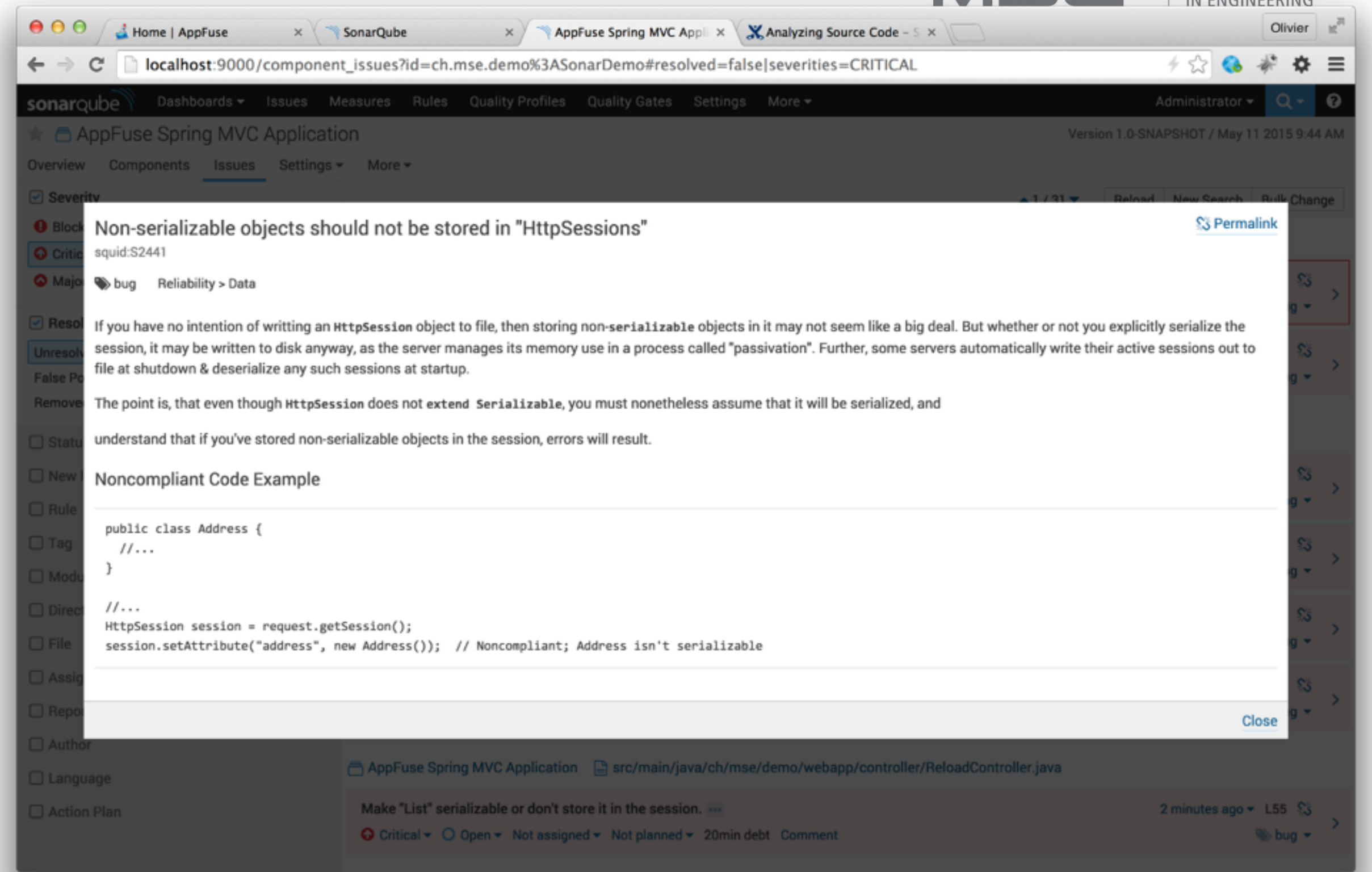
Make "List" serializable or don't store it in the session. — 2 minutes ago L110
Critical | Open | Not assigned | Not planned | 20min debt | Comment | bug

Make "List" serializable or don't store it in the session. — 2 minutes ago L121
Critical | Open | Not assigned | Not planned | 20min debt | Comment | bug

AppFuse Spring MVC Application src/main/java/ch/mse/demo/webapp/controller/ReloadController.java

Make "List" serializable or don't store it in the session. — 2 minutes ago L55
Critical | Open | Not assigned | Not planned | 20min debt | Comment | bug


Home | AppFuse
SonarQube
AppFuse Spring MVC Appli
Analyzing Source Code - S
Olivier
localhost:9000/component_issues?id=ch.mse.demo%3ASonarDemo#resolved=false|severities=CRITICAL
sonarqube
Dashboards
Issues
Measures
Rules
Quality Profiles
Quality Gates
Settings
More
Administrator
AppFuse Spring MVC Application
Version 1.0-SNAPSHOT / May 11 2015 9:44 AM
Overview
Components
Issues
Settings
More
Severity
Reload
New Search
Bulk Change
Non-serializable objects should not be stored in "HttpSessions"
Permalink
squid:S2441
bug
Reliability > Data
If you have no intention of writting an HttpSession object to file, then storing non-serializable objects in it may not seem like a big deal. But whether or not you explicitly serialize the session, it may be written to disk anyway, as the server manages its memory use in a process called "passivation". Further, some servers automatically write their active sessions out to file at shutdown & deserialize any such sessions at startup.
The point is, that even though HttpSession does not extend Serializable, you must nonetheless assume that it will be serialized, and
understand that if you've stored non-serializable objects in the session, errors will result.
Noncompliant Code Example
public class Address {
//...
}
//...
HttpSession session = request.getSession();
session.setAttribute("address", new Address());  // Noncompliant; Address isn't serializable
Close
Rule
Tag
Module
Directory
File
Assignee
Reporter
Author
Language
Action Plan
AppFuse Spring MVC Application
src/main/java/ch/mse/demo/webapp/controller/ReloadController.java
Make "List" serializable or don't store it in the session.
2 minutes ago
L55
Critical
Open
Not assigned
Not planned
20min debt
Comment
bug


Home | AppFuse
SonarQube
AppFuse Spring MVC Appli
Analyzing Source Code - S
Olivier
localhost:9000/component_issues?id=ch.mse.demo%3ASonarDemo#resolved=false|severities=CRITICAL
sonarqube
Dashboards
Issues
Measures
Rules
Quality Profiles
Quality Gates
Settings
More
Administrator
AppFuse Spring MVC Application
Version 1.0-SNAPSHOT / May 11 2015 9:44 AM
Overview
Components
Issues
Settings
More
Severity
Reload
New Search
Bulk Change
Non-serializable objects should not be stored in "HttpSessions"
Permalink
squid:S2441
bug
Reliability > Data
If you have no intention of writting an HttpSession object to file, then storing non-serializable objects in it may not seem like a big deal. But whether or not you explicitly serialize the session, it may be written to disk anyway, as the server manages its memory use in a process called "passivation". Further, some servers automatically write their active sessions out to file at shutdown & deserialize any such sessions at startup.
The point is, that even though HttpSession does not extend Serializable, you must nonetheless assume that it will be serialized, and
understand that if you've stored non-serializable objects in the session, errors will result.
Noncompliant Code Example
public class Address {
//...
}
//...
HttpSession session = request.getSession();
session.setAttribute("address", new Address());  // Noncompliant; Address isn't serializable
Close
Rule
Tag
Module
Directory
File
Assignee
Reporter
Author
Language
Action Plan
AppFuse Spring MVC Application
src/main/java/ch/mse/demo/webapp/controller/ReloadController.java
Make "List" serializable or don't store it in the session.
2 minutes ago
L55
Critical
Open
Not assigned
Not planned
20min debt
Comment
bug