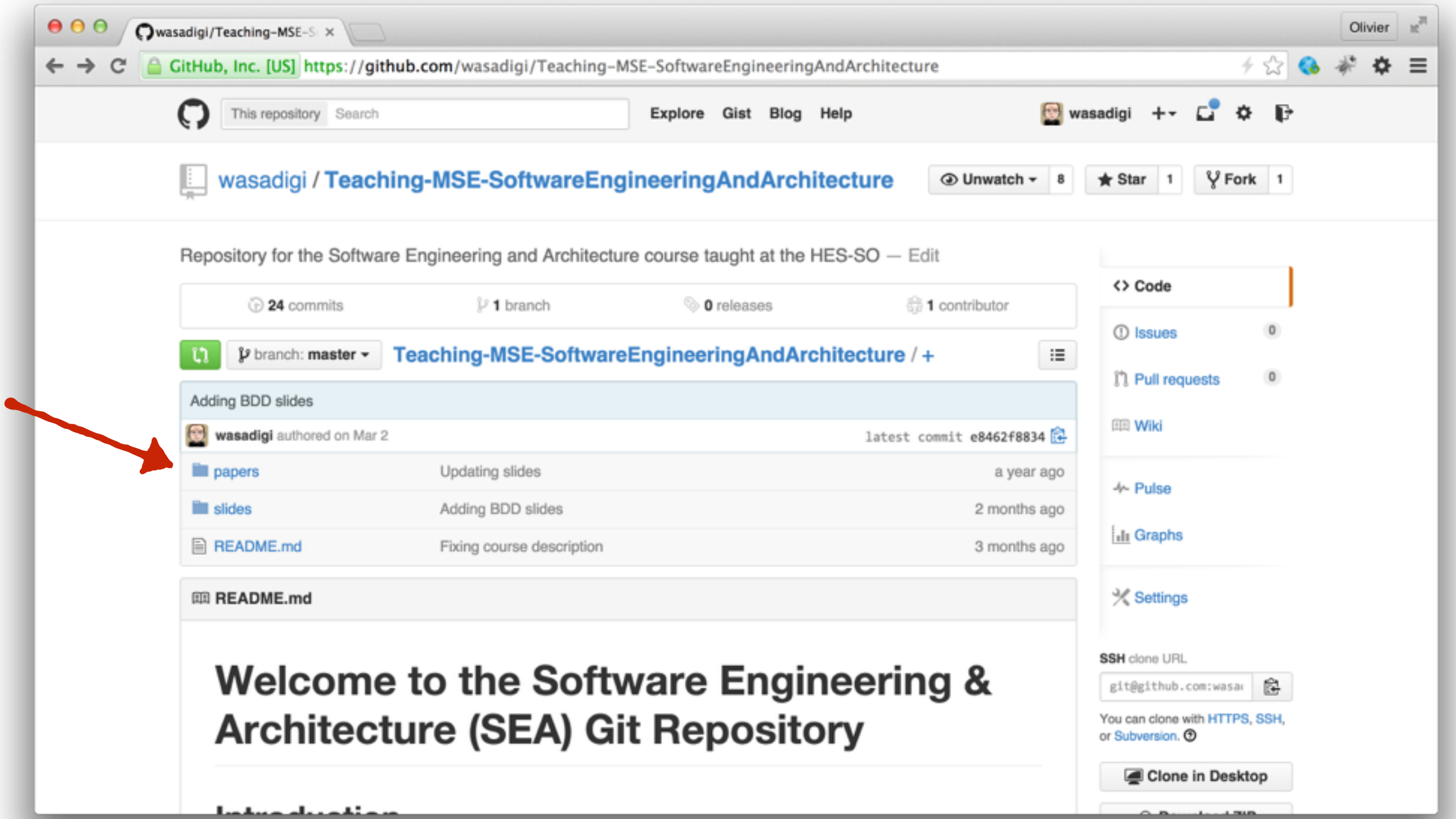
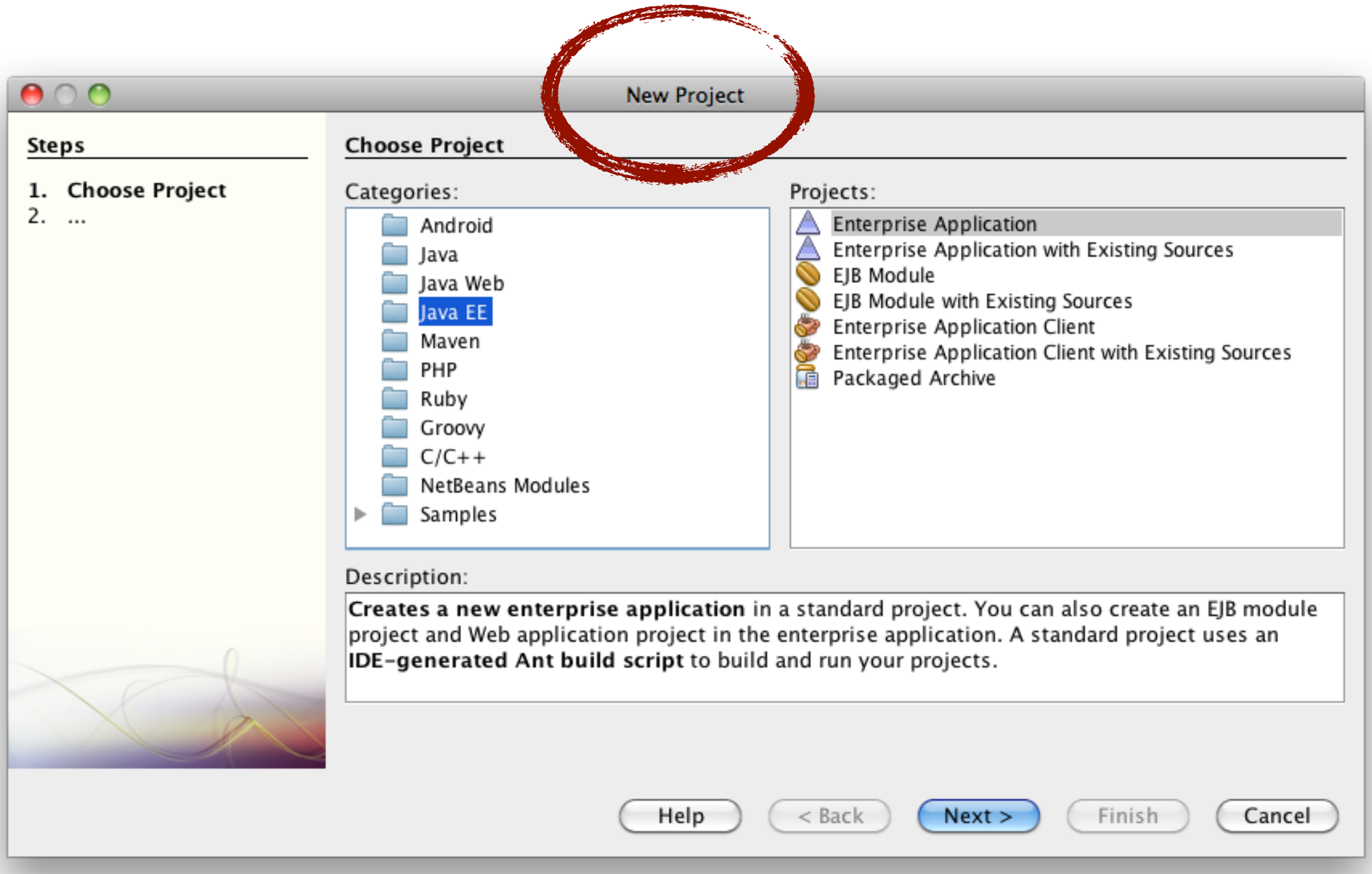
Introduction to Software Evolution

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

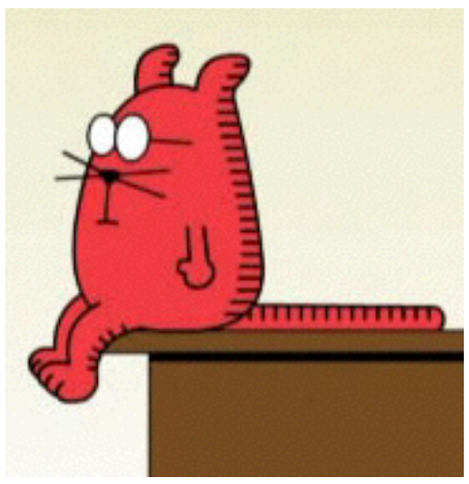
Schedule

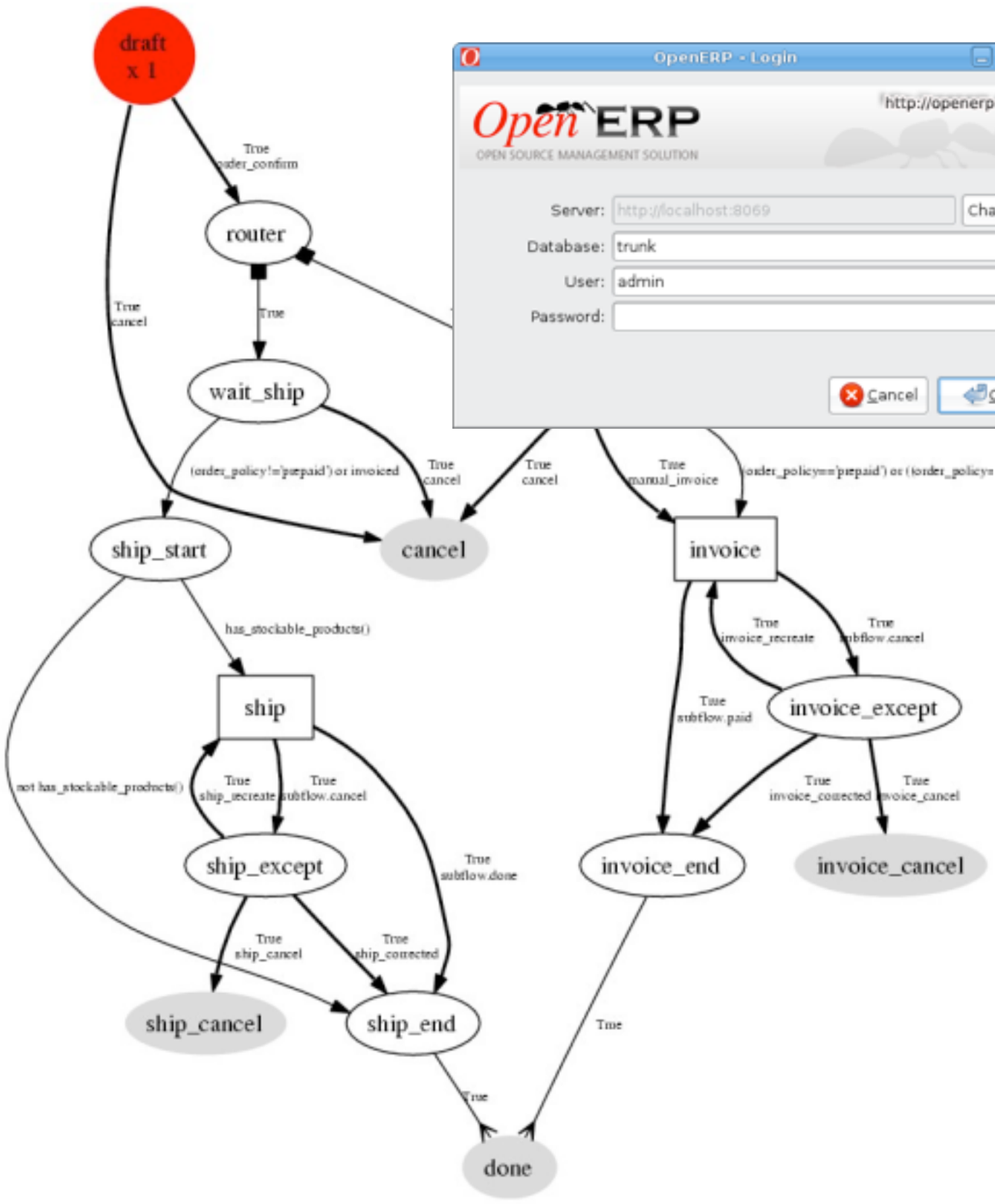
- **Theory**
  - **Introduction**: the scientific and engineering fields of **software evolution**
  - **Foundations**: Laws of Software Evolution, Software Aging
  - **Tools**: code quality tools, software repository mining, etc.
- **Practice**
  - Managing software evolution et code quality with **SonarQube**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


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!



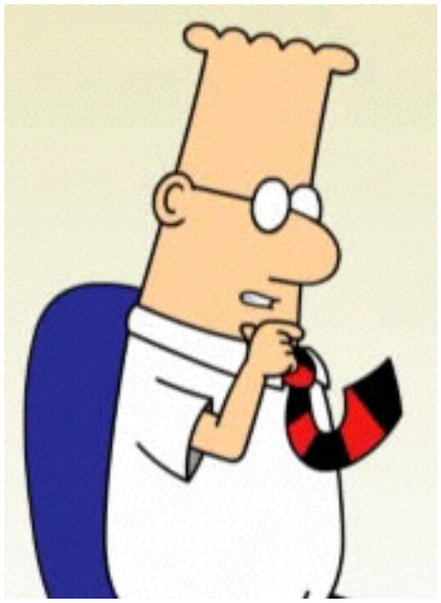


http://doc.openerp.com/developer/2_4_module_development/4_1_module_structure.html

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 [illegible] - 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 [illegible] — 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 [illegible] - 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** [illegible] — 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 patents | 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 detection and estimation of **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?

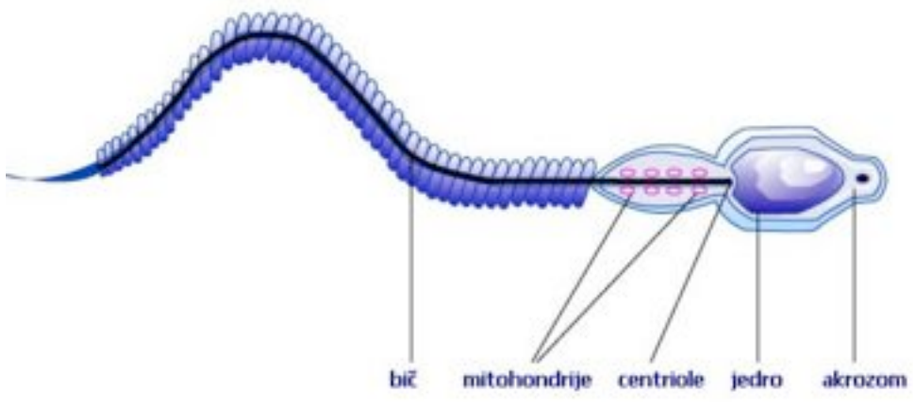
Research on Software Evolution

- **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

- **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

- **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

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.

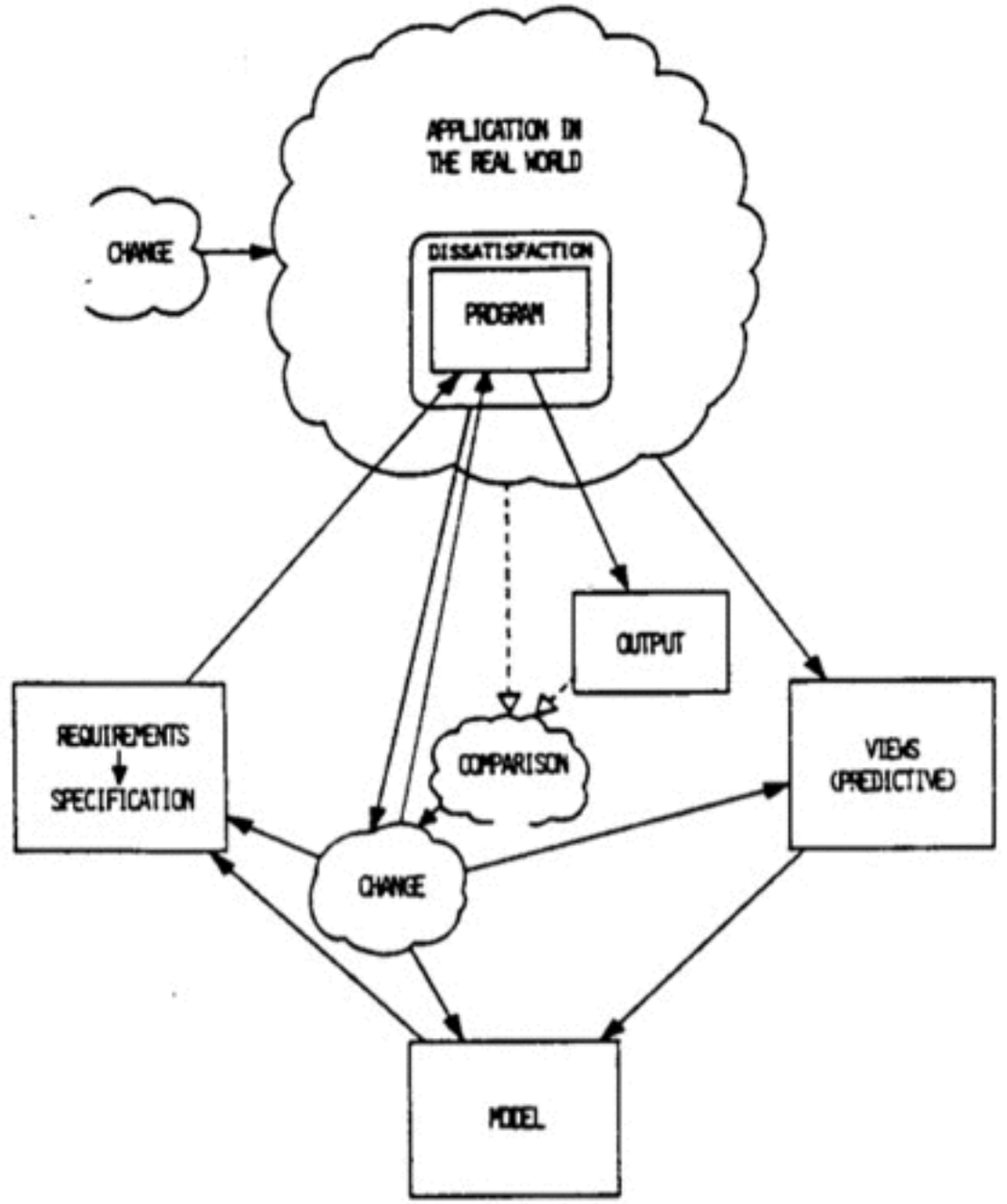
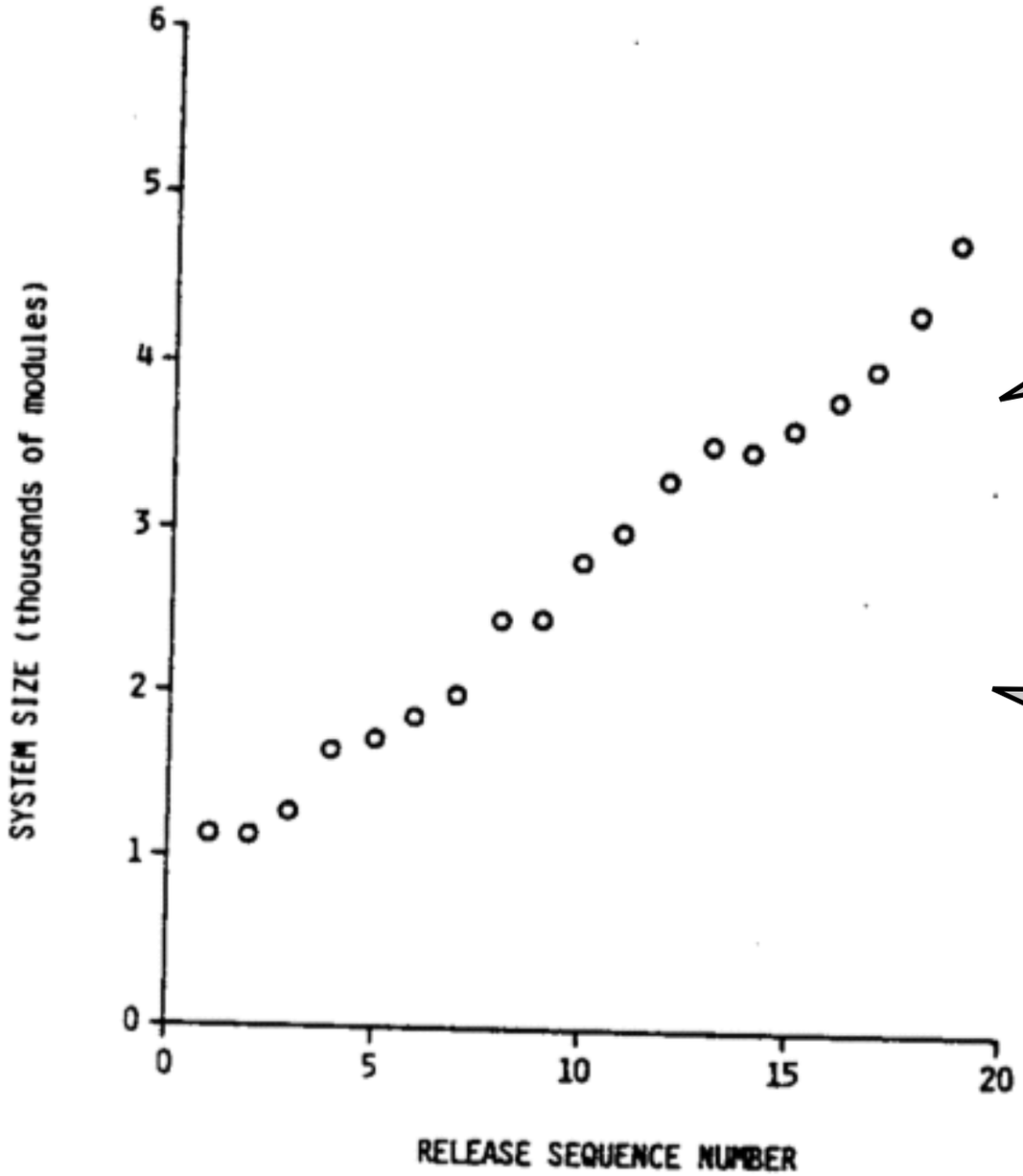


Fig. 4. *E*-programs.

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.**

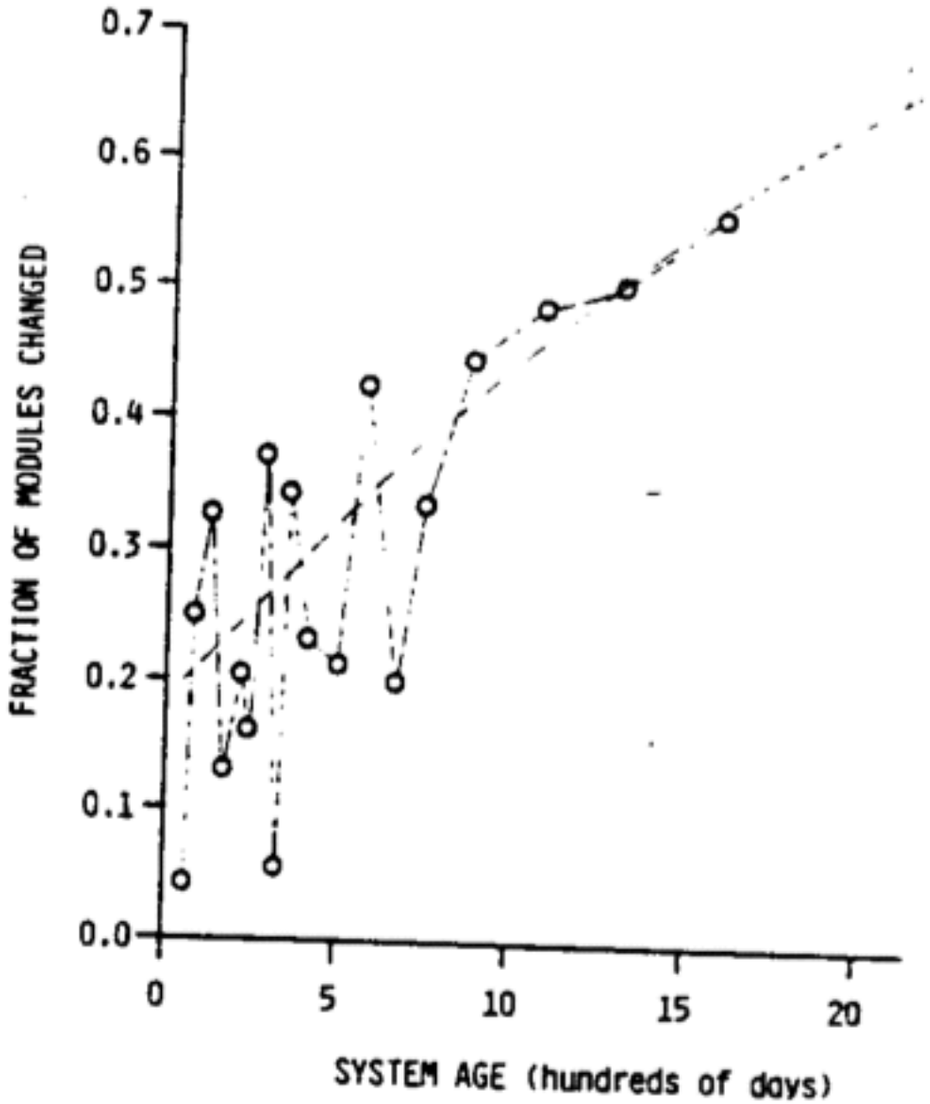
Quantitative Studies



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

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 Maintenance

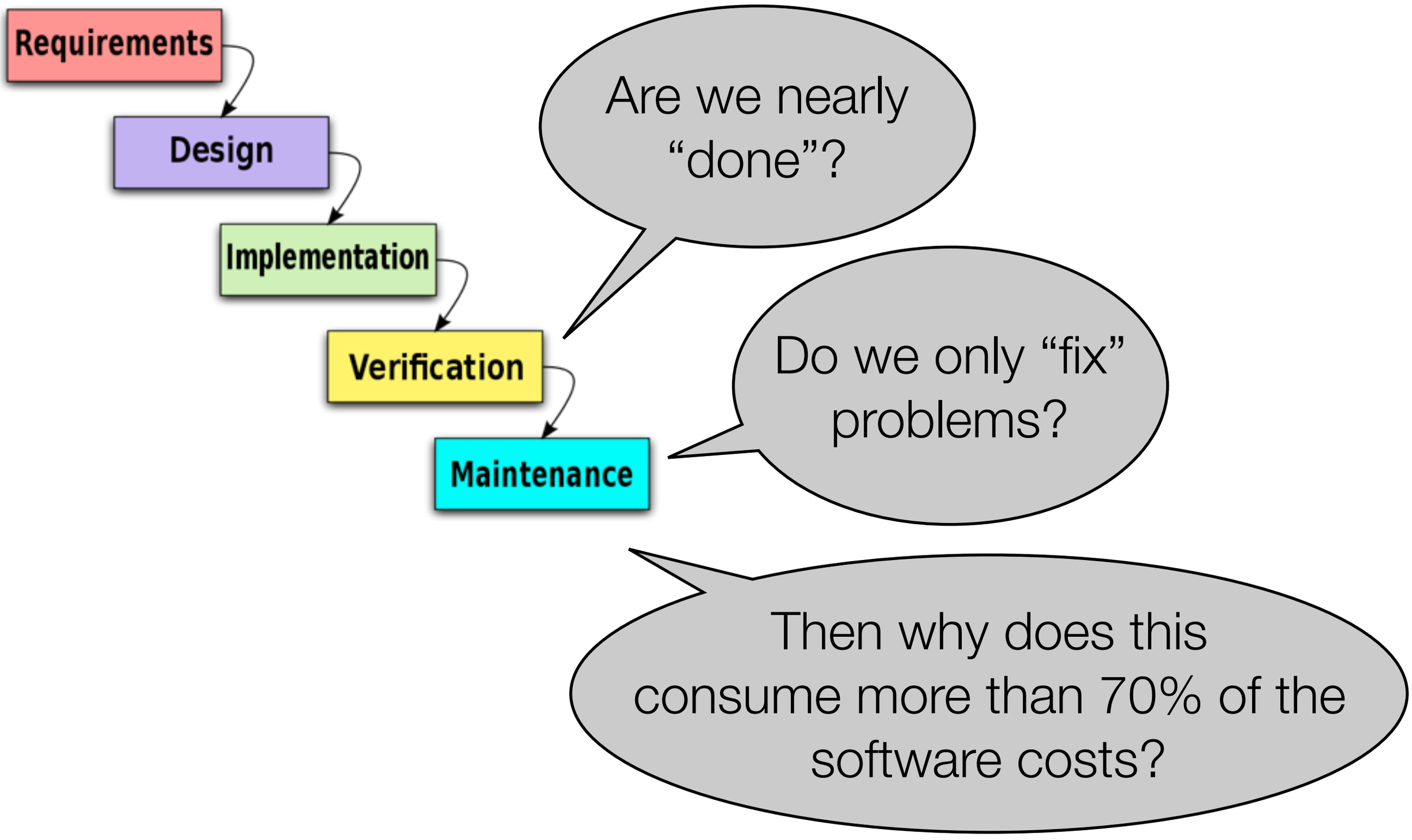
Software Maintenance

- 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

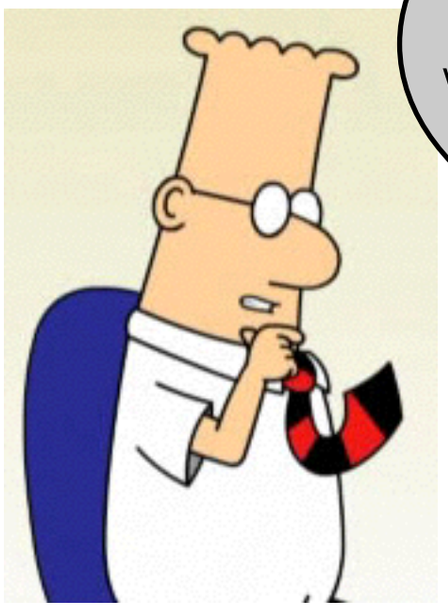
Characteristics of Application Software Maintenance

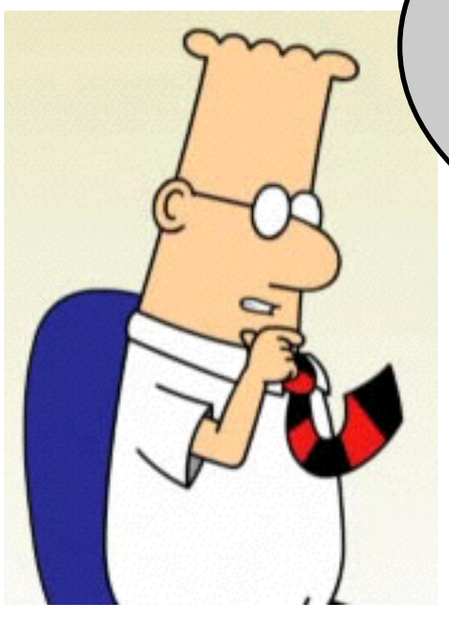
B. P. Lientz, E. B. Swanson,
and G. E. Tompkins
University of California at Los Angeles

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

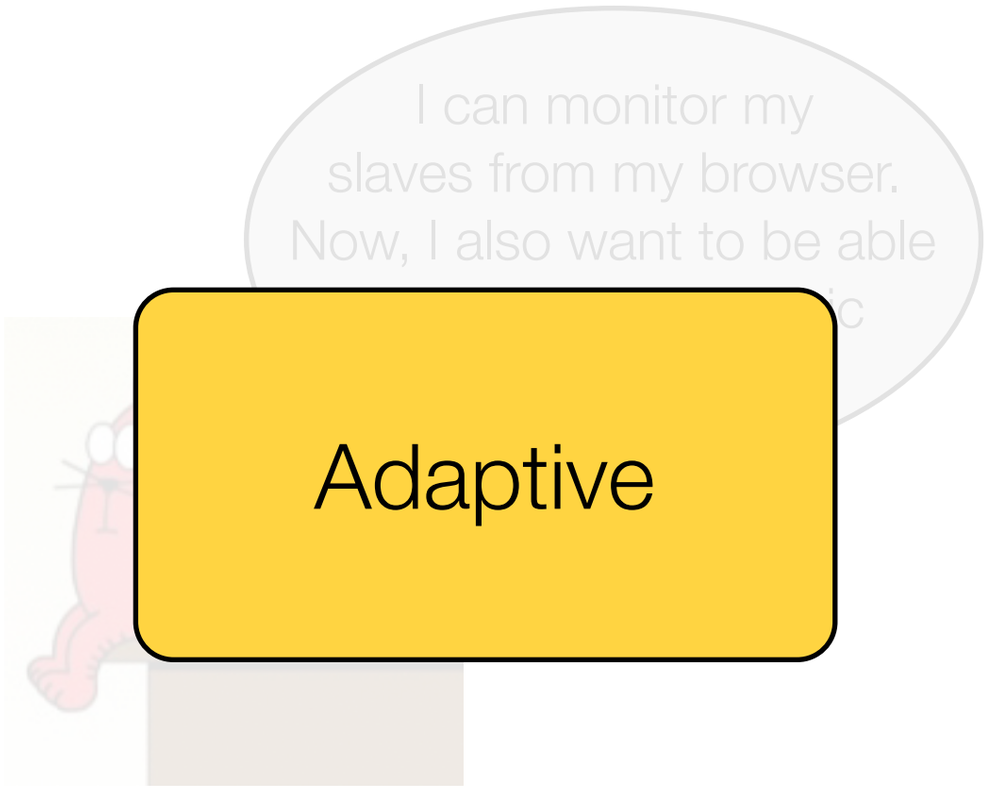




Software Maintenance

Corrective

Preventive



Adaptive

Perfective

Legacy Systems Technical Debt & Re-engineering

Great book for your future consulting activities!

- Open source version of this book:
- http://scg.unibe.ch/download/oorp/

Dealing with Legacy Systems

New or changing requirements will gradually degrade original design…
…unless extra development effort is spent to adapt the structure.

New Functionality

Quick Hack?

Remember Lehmans's laws of Software Evolution?

- Duplicated code
- Complex conditionals
- Abusive inheritance
- Large classes and methods

*Take a **loan** on your software*

- Refactor
- Restructure
- Reengineer

***Invest** for the future*

Technical Debt

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.*

- http://www.youtube.com/watch?v=pqeJFYwnkjE
- http://martinfowler.com/bliki/TechnicalDebt.html

Technical Debt

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.*

- http://www.youtube.com/watch?v=pqeJFYwnkjE
- http://martinfowler.com/bliki/TechnicalDebt.html

Technical 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.

- http://www.youtube.com/watch?v=pqeJFYwnkjE
- http://martinfowler.com/bliki/TechnicalDebt.html

Technical Debt

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.*

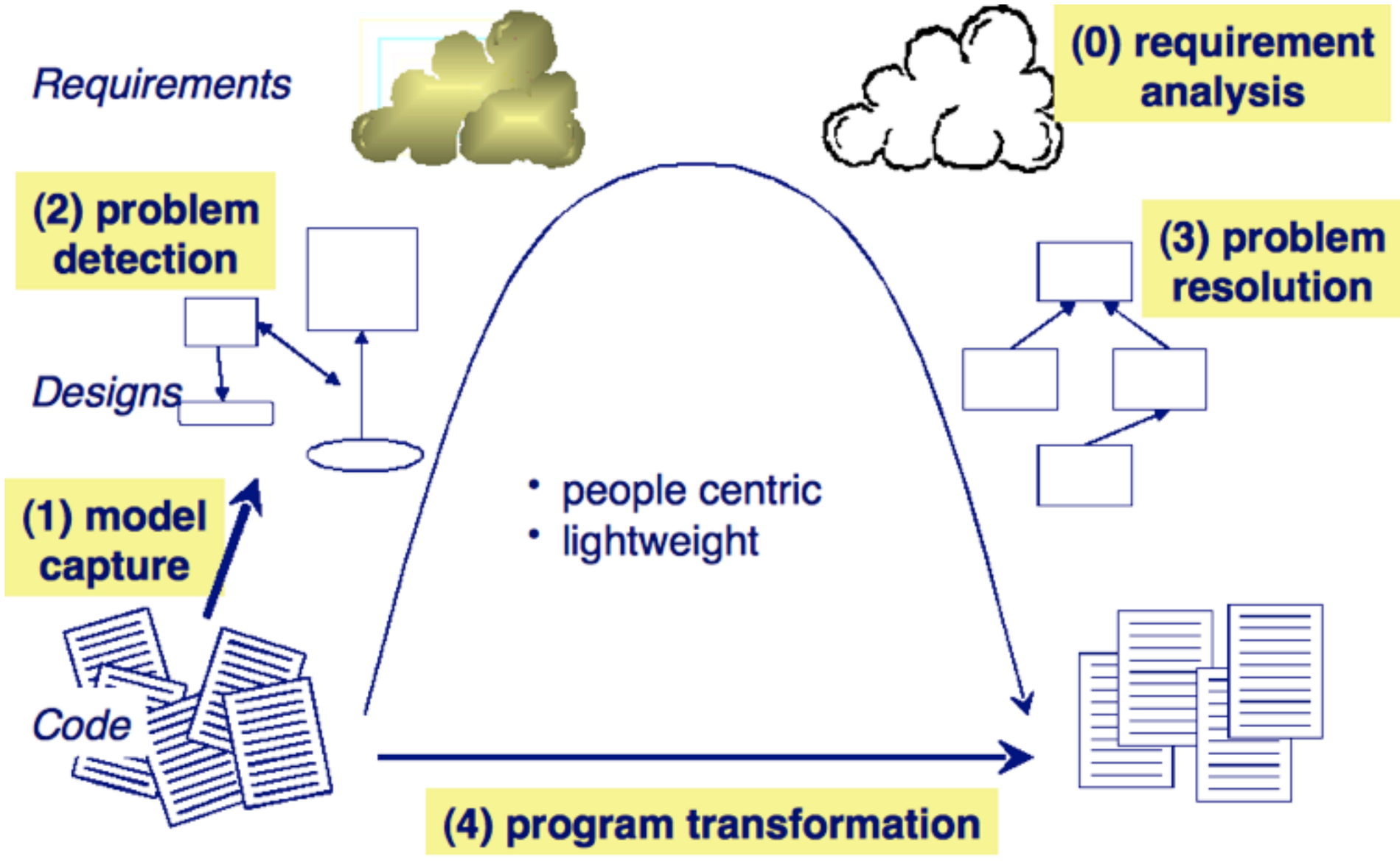
- http://www.youtube.com/watch?v=pqeJFYwnkjE
- http://martinfowler.com/bliki/TechnicalDebt.html

Terminology

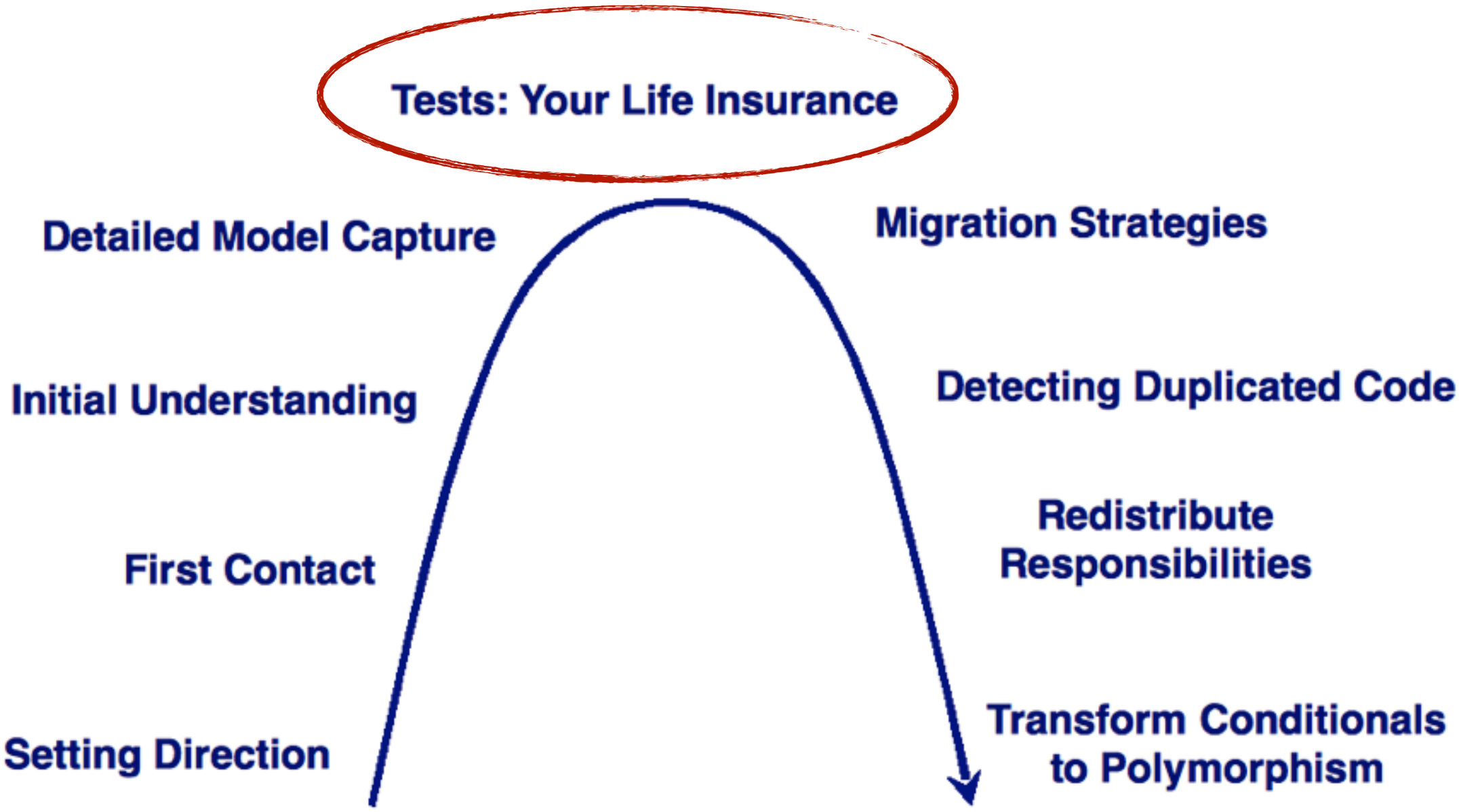
- **Forward Engineering**
  - traditional process of moving **from high-level abstractions** and logical, implementation-independent designs **to the physical implementation** of a system.
- **Reverse Engineering**
  - process of analyzing a subject system to identify the system’s components and their interrelationships and **create representations of the system** in another form or at a higher level of abstraction.
- **Reengineering**
  - examination and alteration of a subject system to **reconstitute it in a new form** and the subsequent implementation of the new form.

Chikofsky and Cross, 1993

The Reengineering Life-Cycle

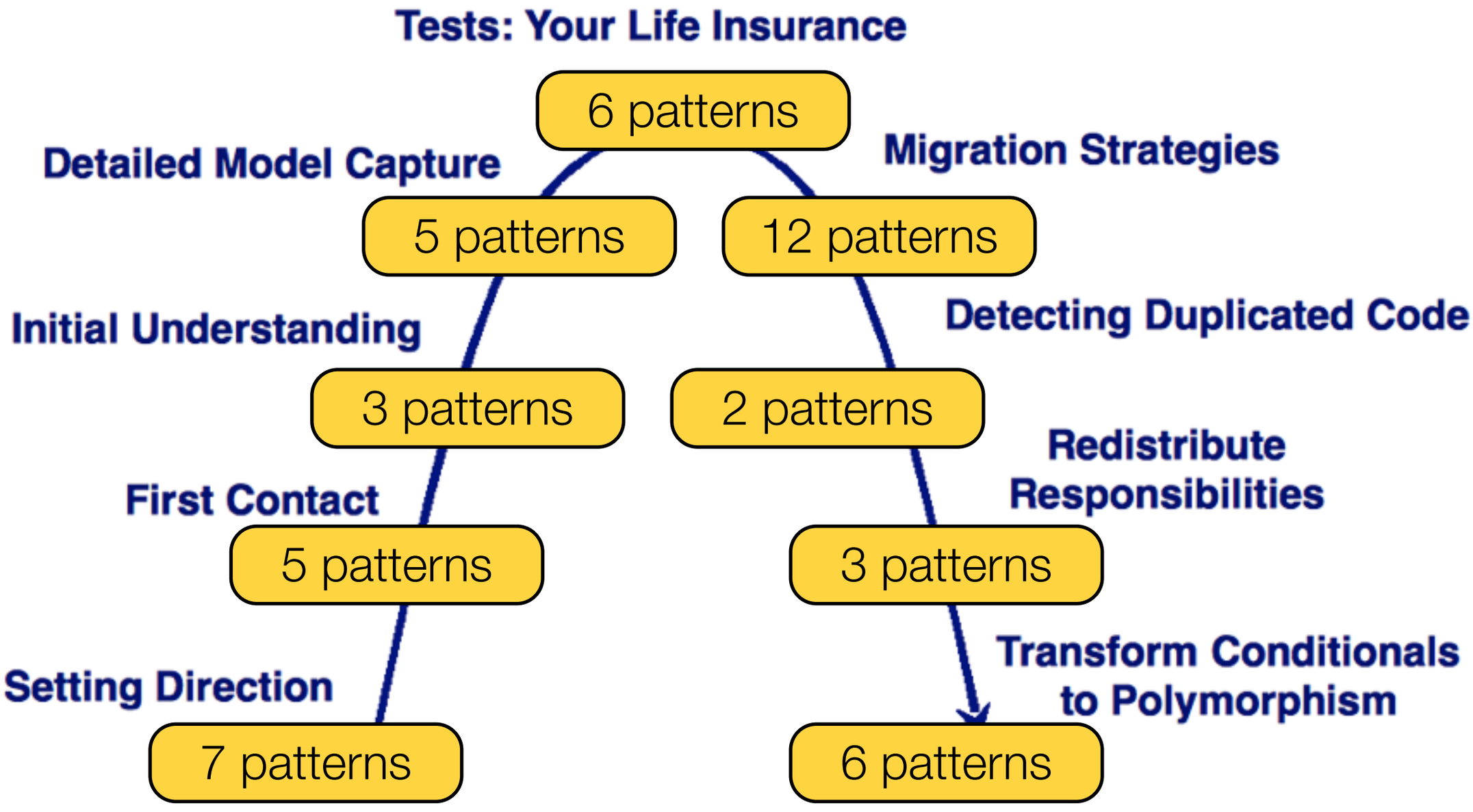


A Map of Reengineering Patterns



Based on https://www.iam.unibe.ch/scg/svn_repos/Lectures/OORPT/

A Map of Reengineering Patterns



Based on https://www.iam.unibe.ch/scg/svn_repos/Lectures/OORPT/

Reverse Engineering Patterns

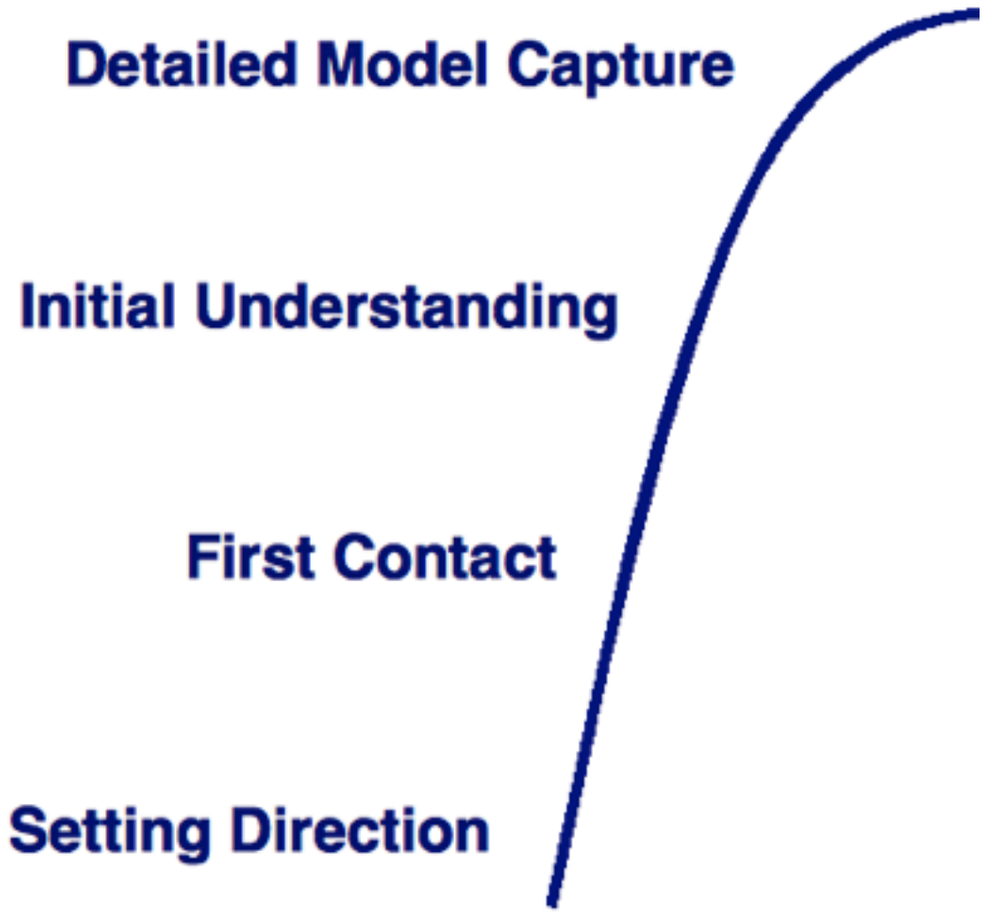
Reverse engineering patterns encode expertise and trade-offs in ***extracting design*** *from* ***source code***, ***running systems*** *and* ***people***.

Mission Impossible

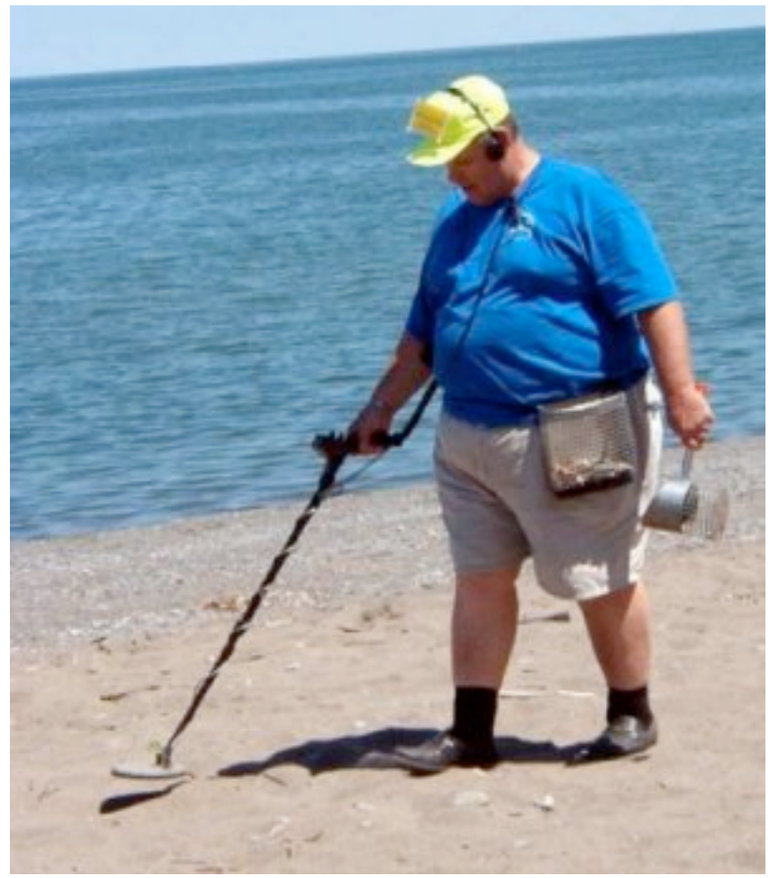
- **“Allo, PK Consulting”?**
  - “Yes”
- **“We have got a small problem with our fancy social networking site...”**
  - “Yes?”
- **“As soon as there are 20 concurrent users, it implodes...”**
  - “Yes...”
- **“Can you help us?”**
  - “Yes.”

Reverse Engineering Patterns

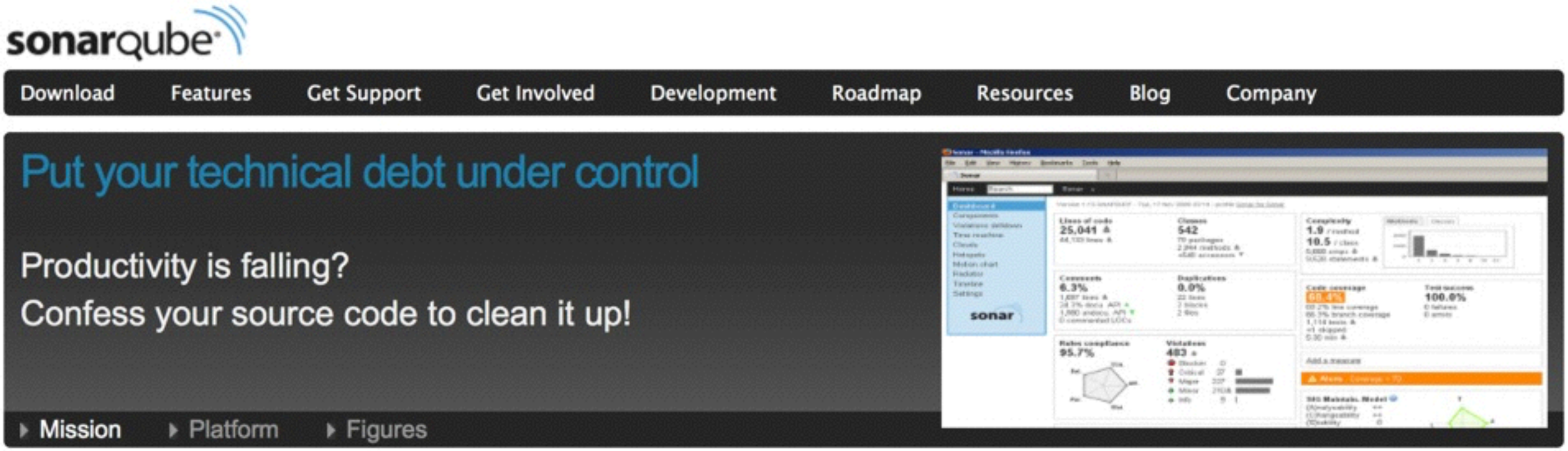
- When you start a reengineering project, you will be pulled in **many different directions**, by management, by the users, by your own team.
- How do you **set the direction** of the reengineering effort, and how do you **maintain direction** once you have started?

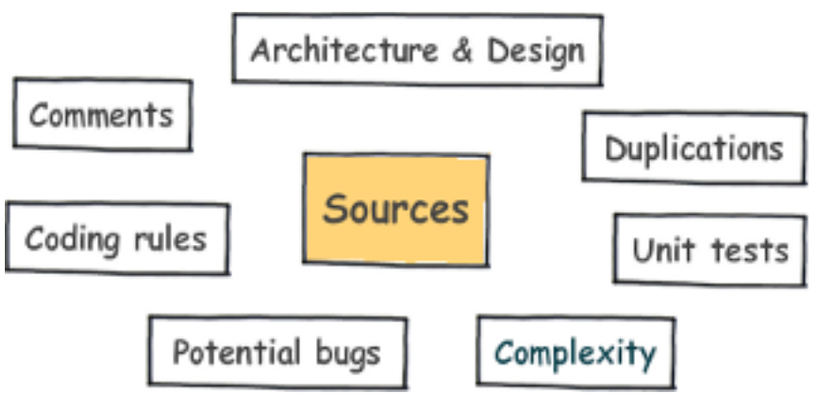


Software evolution & code quality in practice



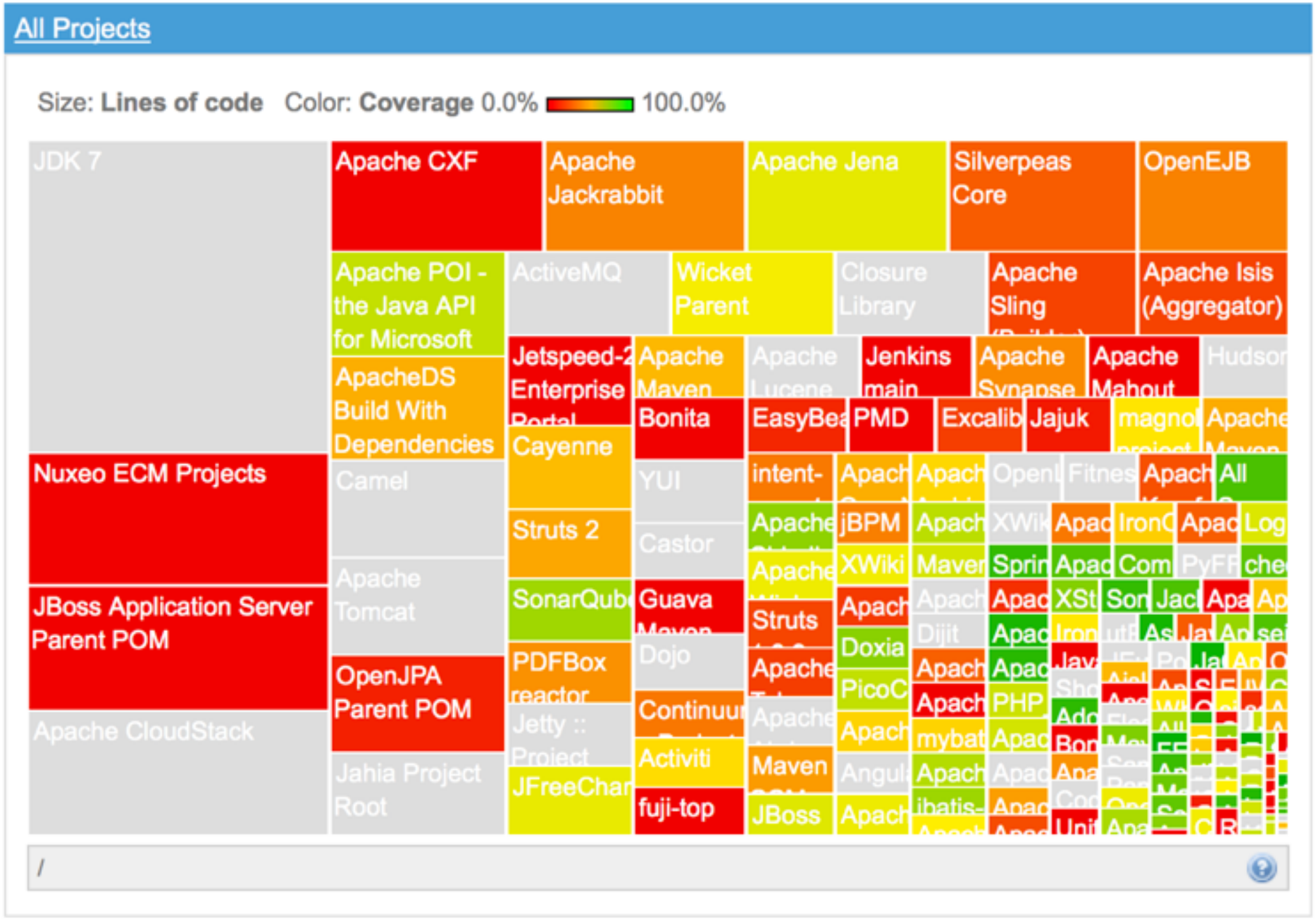
Tool: SonarQube





http://www.sonarqube.org/

SonarQube on open source projects



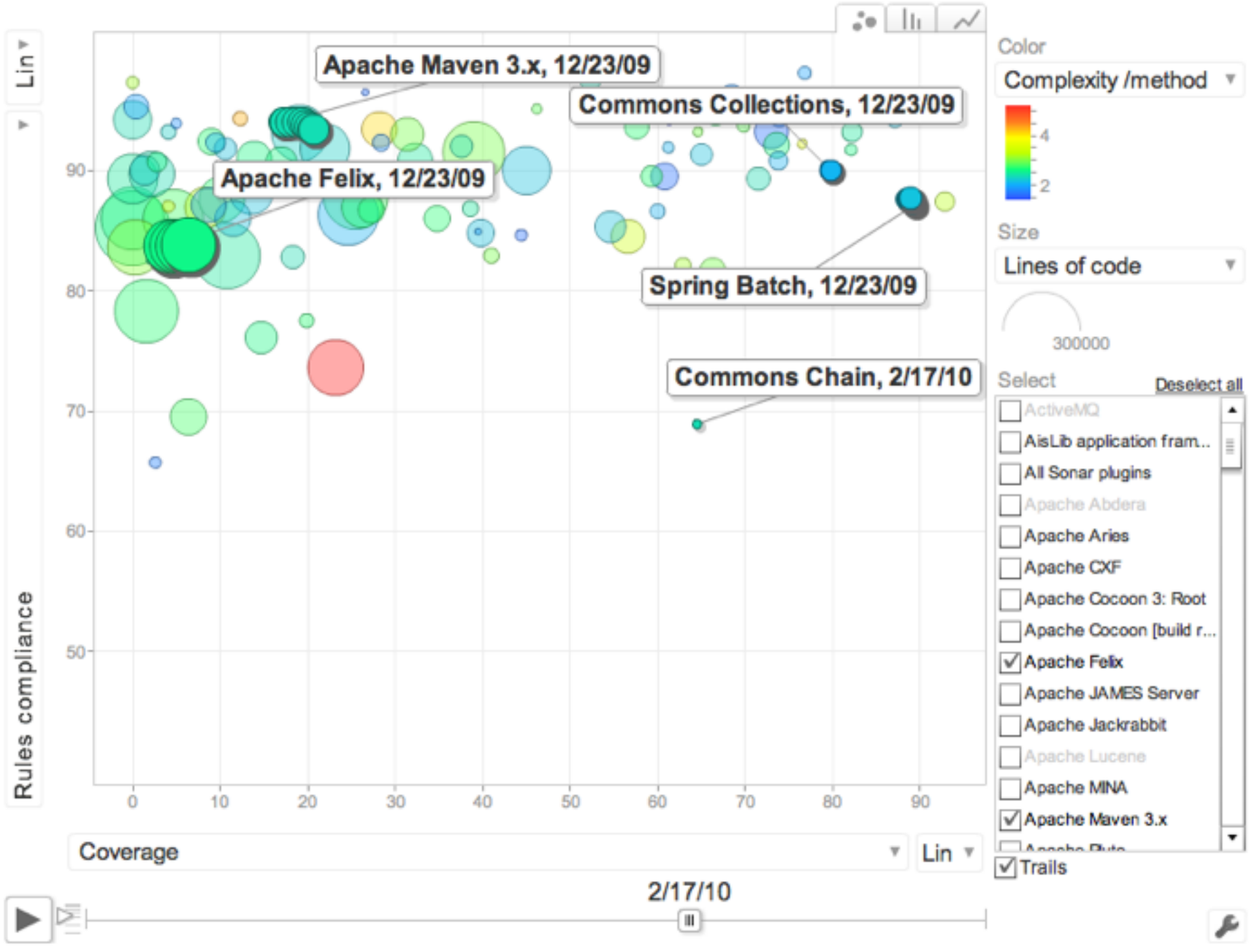
http://nemo.sonarqube.org/

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

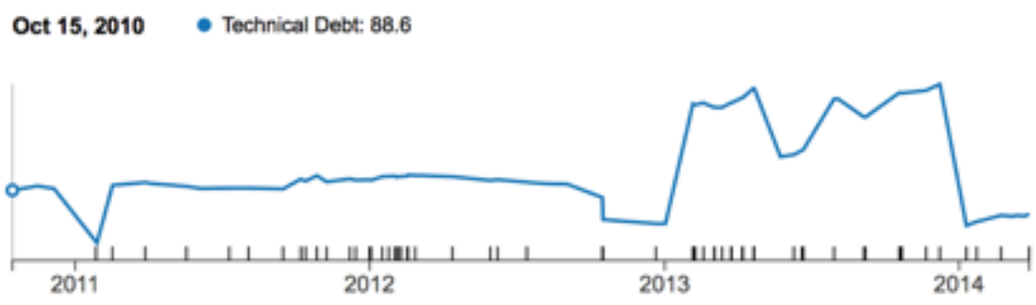


http://nemo.sonarqube.org/

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

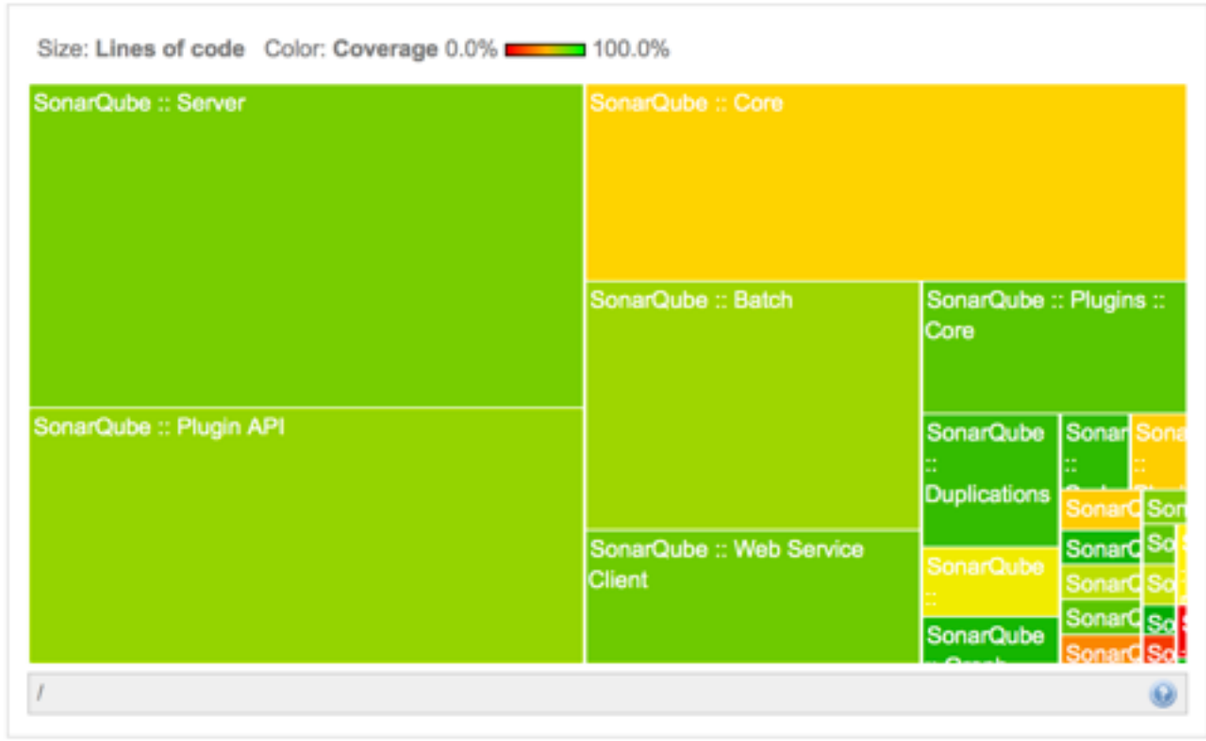


Unit Tests Coverage
68.4%
68.4% line coverage
68.4% branch coverage

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
```

Back to our CD pipeline!

Where did we leave it?

- **We have implemented 2 micro-services:**
  - The “clock” micro-service
  - A “CRUD” REST API (the “beers” micro-service”)
- **We have defined 2 docker “topologies”:**
  - One for the CD pipeline itself: for now, it consists of a Jenkins service
  - One for the testing environment: it consists of one container for every micro-service.
- **We have updated the definition of our pipeline (Jenkinsfile):**
  - Each time we trigger a build, Jenkins will: 1) fetch the source code, 2) tell maven to build executable jars, 3) tell Docker to package the jars into images and 4) tell Docker to “up” a topology (with docker-compose).
- **At the end of the build process, we have a runtime environment with the last version of our micro-services. We can now test and validate them.**

What do we want to do now?

- **Firstly, in the commit stage, we want to assess the quality of our code (and compute our technical debt):**
  - We have seen that **SonarQube** is a very popular solution. Let's see if we can integrate it in our pipeline!
  - We could go quite far. SonarQube defines the notion of "quality gates" and exposes the information via REST APIs. So, we could "**break the build**" if SonarQube tells us that the code quality is below a given threshold.
    - **We will do something simpler**: ask SonarQube to do the evaluation and make the information available for **human inspection**.
- **Secondly, in the validation stage, we want to launch a Docker container that will execute the API tests written with the "supertest" Javascript framework.**
  - We will initially not break the build if tests fail.

Task: assess code quality with Sonar

Step 1: get familiar with SonarQube

- **Remember what we did a looooong time ago to discover Jenkins?**
  - We paid a visit to Docker Hub
  - We looked if there was an (official) image wrapping the Jenkins service
  - With a single “docker run” command (and some bandwidth), we had a working server to play with within minutes.
- **Let’s apply the same strategy for SonarQube!**

ImageLayers.io 0 B / 20 Layers

For more information about this image and its history, please see the relevant manifest file (`library/sonarqube`). This image is updated via pull requests to the `docker-library/official-images` GitHub repo.

For detailed information about the virtual/transfer sizes and individual layers of each of the above supported tags, please see the `sonarqube/tag-details.md` file in the `docker-library/docs` GitHub repo.

What is SonarQube?

SonarQube is an open source platform for continuous inspection of code quality.

> wikipedia.org/wiki/SonarQube

sonarqube

How to use this image

Run SonarQube

The server is started this way:

```
$ docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube
```

To analyse a project:

```
$ On Linux:
mvn sonar:sonar

$ With boot2docker:
mvn sonar:sonar -Dsonar.host.url=http://$(boot2docker ip):9000 -Dsonar.jdbc.url="jdbc:h
```

https://hub.docker.com/_/s | SonarQube | Olivier

192.168.99.100:8000

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 Go Support page.

PROJECTS

QG	NAME	VERSION	LOC	BUGS	VULNERABILITIES	CODE SMELLS	LAST ANALYSIS
No data							

PROJECTS

No data

admin/admin

Embedded database should be used for evaluation purpose only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA

Version 5.5 - LGPL v3 - Community - Documentation - Get Support - Plugins - Web API

Step 2: Invoke SonarQube from the pipeline

```
/*-----------------------------------------------------------------------*/
stage 'Commit'
/*-----------------------------------------------------------------------*/

echo "Build the clock micro-service and create a Docker image"
// Build an executable jar of Game Dock with maven and build a corresponding Docker image
dir('microservices/ClockService/') {
     sh "mvn clean package"
}
sh "mkdir -p docker-images/service-clock/tmp"
sh "cp microservices/ClockService/target/*SNAPSHOT.jar docker-images/service-clock/tmp/"
dir('docker-images/service-clock') {
     sh 'docker build -t sea/service_clock .'
}

echo "Ask SonarQube to run a static analysis of the code; results will be available in Sonar Web UI"
dir('microservices/ClockService/') {
     sh "mvn sonar:sonar -Dsonar.host.url=${SONAR_URL}"
}
```

https://hub.docker.com/_/s | SonarQube | Olivier

192.168.99.100:9000/dashboard/index?did=2

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	BUGS	VULNERABILITIES	CODE SMELLS	LAST ANALYSIS
✓	beers	0.0.1-SNAPSHOT	64	1	0	1	10:06
✓	clockservice	0.0.1-SNAPSHOT	32	1	0	1	10:06

2 results

PROJECTS

Size: Lines of code Color: Coverage

Embedded database should be used for evalua

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is

Resources should be closed × Utility classes should not hav... × BeersApplication.java ×

https://hub.docker.com/_/s | beers | Olivier

192.168.99.100:9000/overview?id=10

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

beers

June 2, 2016 12:06 PM Version 0.0.1-SNAPSHOT

Issues | Measures | Code | Dashboards

Quality Gate **Passed**

Bugs & Vulnerabilities

1 E	0 A	Leak Period: since previous version started 2 hours ago	
Bugs	Vulnerabilities	0 New Bugs	0 New Vulnerabilities

Code Smells

1 A	30min	0	0
Code Smells	Debt	New Code Smells	New Debt

started 2 hours ago

Duplications

0.0%	0	+0.0%
Duplications	Duplicated Blocks	Duplications

Structure

Java 100.0%	64	+0
	Lines of Code	Lines of Code

Demo project for Spring Boot

Home
Sources

Key
ch.heigvd.sea:beers

Quality Gate
(Default) SonarQube way

Quality Profiles
(Java) Sonar way

Events All

Version: 0.0.1-SNAPSHOT
June 2, 2016

Embedded database should be used for evaluation purpose only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is

Resources should be closed × | Utility classes should not hav... × | BeersApplication.java ×

https://hub.docker.com/_/s × | beers × | Olivier

192.168.99.100:9000/component_issues/index?id=ch.heigvd.sea%3Abeers#resolved=false

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

beers

June 2, 2016 12:06 PM Version 0.0.1-SNAPSHOT

Issues | Measures | Code | Dashboards

Issues | Effort

▲ 1 / 2 ▼ Reload | New Search

Type

Bug	1
Vulnerability	0
Code Smell	1

Resolution

Unresolved	2	Fixed	0
False Positive	0	Won't fix	0
Removed	0		

- Severity
- Status
- New Issues
- Rule
- Tag
- Module
- Directory
- File
- Assignee
- Author
- Language

beers | src/main/java/ch/heigvd/sea/BeersApplication.java

Add a private constructor to hide the implicit public one. ...
2 hours ago L7
Code Smell | Major | Open | Not assigned | 30min effort
design

Close this "ConfigurableApplicationContext". ...
2 hours ago L10
Bug | Blocker | Open | Not assigned | 5min effort
cert, cwe, denial-of-service, leak

Embedded database should be used for evaluation purpose only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA

Version 5.5 - LGPL v3 - Communi

Resources should be closed × | Utility classes should not hav... × | BeersApplication.java ×

https://hub.docker.com/_/s × | beers × | Olivier

192.168.99.100:9000/component_issues/index?id=ch.heigvd.sea%3Abeers#resolved=false

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

beers

June 2, 2016 12:06 PM Version 0.0.1-SNAPSHOT

Issues | Measures | Code | Dashboards

Issues | Effort

Type

Bug	1
Vulnerability	0
Code Smell	1

Resolution

Unresolved	2	Fixed	0
False Positive	0	Won't fix	0
Removed	0		

- Severity
- Status
- New Issues
- Rule
- Tag
- Module
- Directory
- File
- Assignee
- Author
- Language

Return to List | beers | BeersApplication.java | 1 / 2 | Reload | New Search

beers
src/main/java/ch/heigvd/sea/BeersApplication.java

13 Lines | 2 Issues

```
1  olivi…  package ch.heigvd.sea;
2
3          import org.springframework.boot.SpringApplication;
4          import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6          @SpringBootApplication
7          public class BeersApplication {
```

Add a private constructor to hide the implicit public one. …
Code Smell | Major | Open | Not assigned | 30min effort — 2 hours ago | L7 | design

```
8
9                  public static void main(String[] args) {
10                         SpringApplication.run(BeersApplication.class, args);
```

Close this "ConfigurableApplicationContext". …
Bug | Blocker | Open | Not assigned | 5min effort — 2 hours ago | L10 | cert, cwe, denial-of-service, leak

```
11         }
12 }
13
```

Resources should be closed × | Utility classes should not hav… × | BeersApplication.java ×

Task: run our API tests

Step 1: prepare a “validation” Docker image

- **What we have:**
  - After packaging the executable .jars in Docker images, our pipeline uses “docker-compose” to launch a “runtime topology”.
  - Every micro-service is running in its container and is accessible via a URI.
  - We have a set of API tests written in JavaScript. We can run them on our host (with node).
- **What we have to do:**
  - For every new build, we want to get the last version of the API tests.
  - We want to package them in a Docker image.
  - We want to run a container, which will execute the tests (i.e. make HTTP calls), produce an output and die.
  - We have to be careful: we can only run the tests when the micro-service has been fully started (docker tells us that the boot process has started... but not that the API is available).

Step 2: Invoke SonarQube from the pipeline

```
/*-----------------------------------------------------------------------------*/
stage 'Validation'
/*-----------------------------------------------------------------------------*/

echo "Create a fresh test environment, by UP'ing a Docker Compose topology"
dir('docker-topologies/topology-runtime') {
        sh "docker-compose down || true"
        sh "docker-compose up &"
}

waitUntil {
        def appIsReady = false
        try {
                echo "Checking Spring Boot status page via ${CLOCK_MICROSERVICE_URL}"
                sh "set +e; curl -f -sL -w \"%{http_code}\\n\" ${CLOCK_MICROSERVICE_URL}/health -o /dev/null; echo \$? > springBootAppStatus; return 0"
                def status = readFile('springBootAppStatus').trim()
                echo 'status: ' + status
                appIsReady = (status == '0')
        } catch (e) {
                echo 'exception: ' + e
                appIsReady = false
        }
        echo 'return appIsReady'
        return appIsReady == true

}
echo "The clock micro-service have been started and is ready"

dir('validation') {
        sh './build-image.sh'
}
sh "docker run -e CLOCK_API_PREFIX=${CLOCK_MICROSERVICE_URL} sea/validation-api-tests"
echo "API tests have been run."
```

https://hub.docker.com/_/j | SEA Build Pipeline [Jenkins | Olivier

192.168.99.100:18080/job/SEA%20Build%20Pipeline/

Jenkins | search

Jenkins › SEA Build Pipeline › | ENABLE AUTO REFRESH

- Back to Dashboard
- Status
- Changes
- Build with Parameters
- Delete Pipeline
- Configure
- Move
- Full Stage View

Build History | trend

find

- #4 Jun 2, 2016 10:05 AM
- #3 Jun 2, 2016 9:40 AM
- #2 Jun 2, 2016 9:37 AM
- #1 Jun 2, 2016 8:55 AM

RSS for all | RSS for failures

Pipeline SEA Build Pipeline

add description

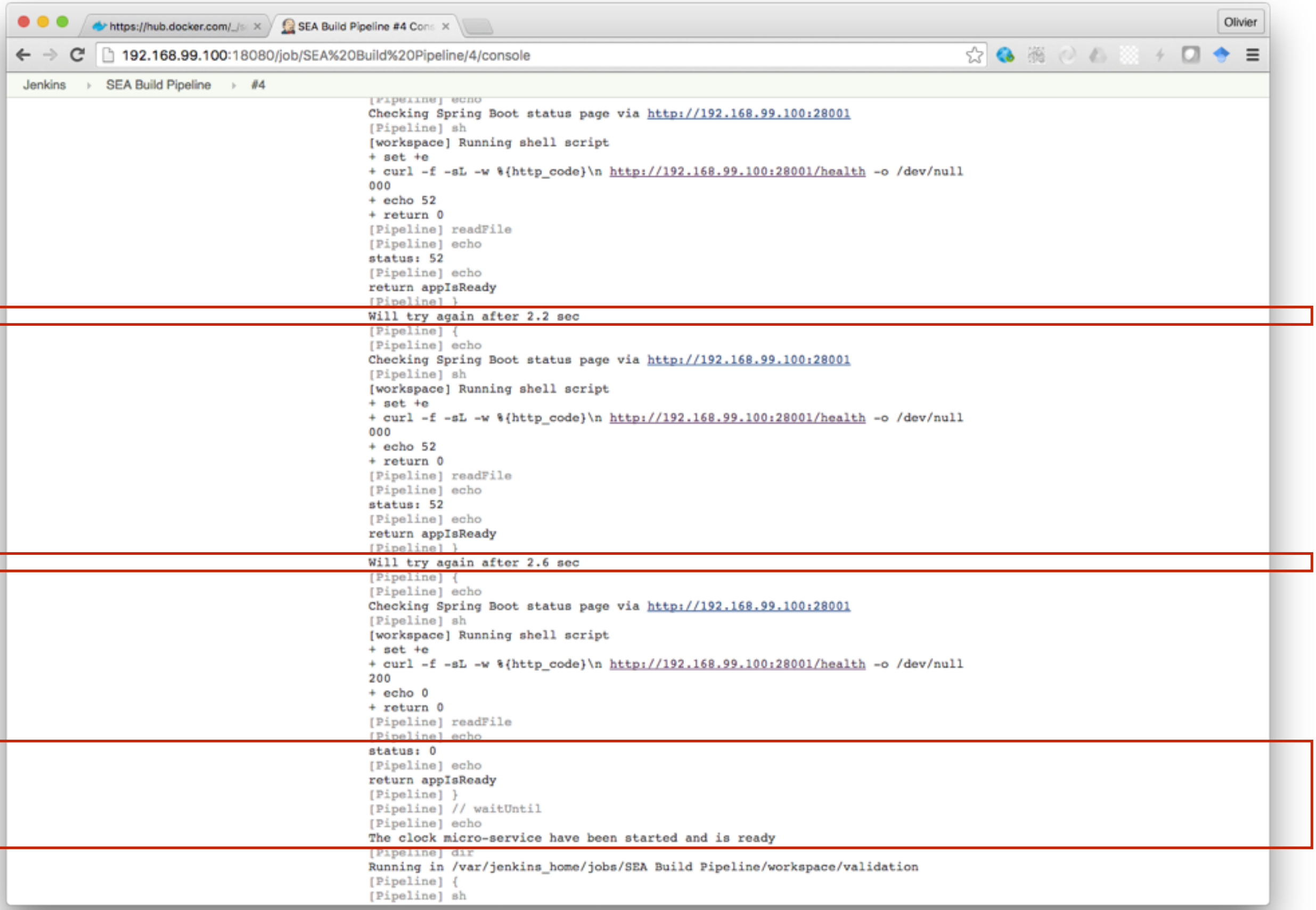
Recent Changes

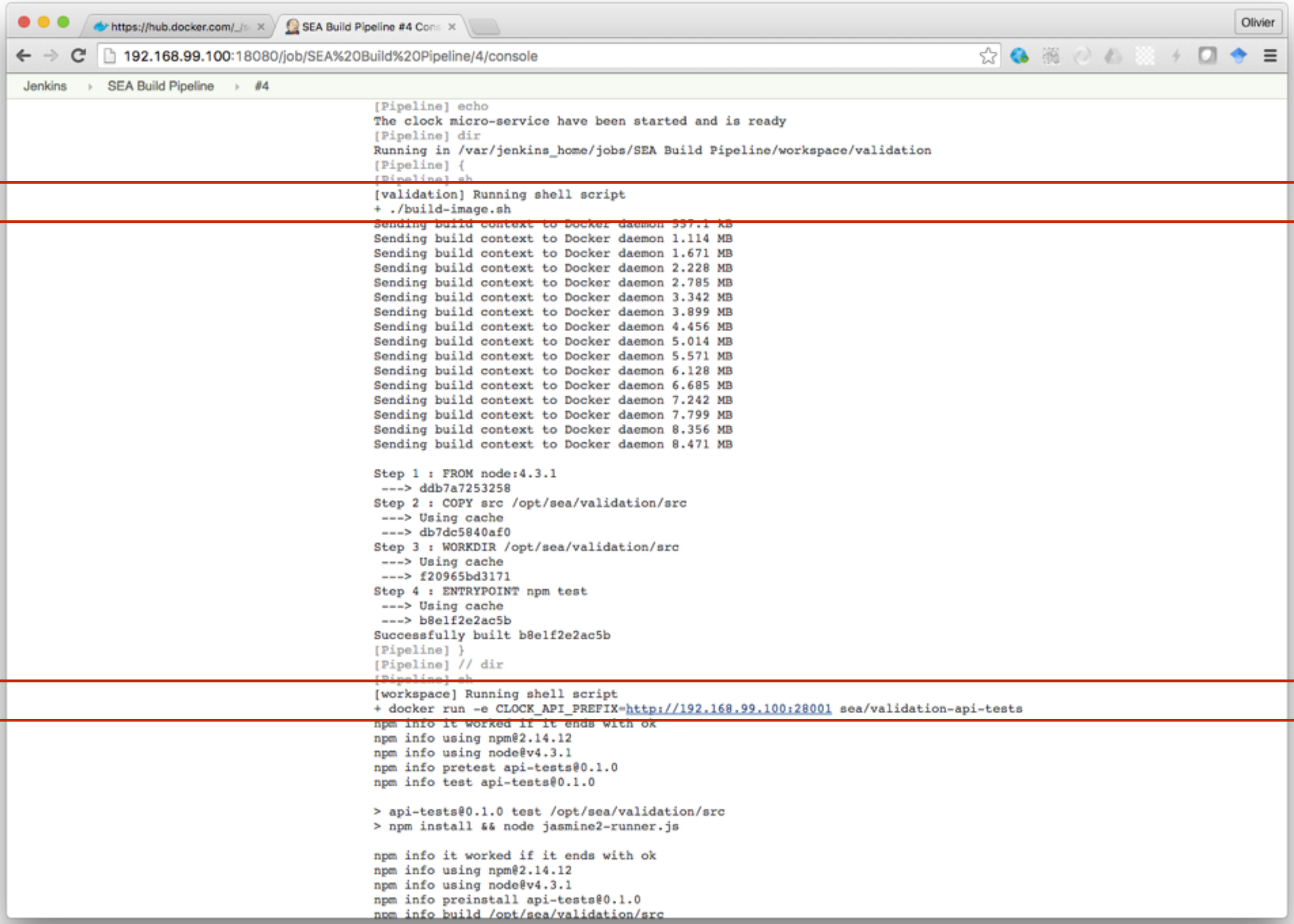
Stage View

	Setup	Commit	Validation	End
Average stage times: (Average full run time: ~2min 32s)	31s	1min 25s	33s	21ms
#4 Jun 02 12:05 — 1 commits	1s	1min 3s	27s	21ms
#3 Jun 02 11:40 — 1 commits	2s	1min 3s	30s	14ms
#2 Jun 02 11:37 — 2 commits	4s	1min 7s	1min 13s aborted	
#1 Jun 02 10:55 — No Changes	1min 58s	2min 27s	1s	30ms

Permalinks

- Last build (#4), 1 hr 10 min ago
- Last stable build (#4), 1 hr 10 min ago
- Last successful build (#4), 1 hr 10 min ago
- Last unsuccessful build (#2), 1 hr 39 min ago
- Last completed build (#4), 1 hr 10 min ago


https://hub.docker.com/_/
SEA Build Pipeline #4 Cons
Olivier
192.168.99.100:18080/job/SEA%20Build%20Pipeline/4/console
Jenkins
SEA Build Pipeline
#4
[Pipeline] echo
Checking Spring Boot status page via http://192.168.99.100:28001
[Pipeline] sh
[workspace] Running shell script
+ set +e
+ curl -f -sL -w %{http_code}\n http://192.168.99.100:28001/health -o /dev/null
000
+ echo 52
+ return 0
[Pipeline] readFile
[Pipeline] echo
status: 52
[Pipeline] echo
return appIsReady
[Pipeline] }
Will try again after 2.2 sec
[Pipeline] {
[Pipeline] echo
Checking Spring Boot status page via http://192.168.99.100:28001
[Pipeline] sh
[workspace] Running shell script
+ set +e
+ curl -f -sL -w %{http_code}\n http://192.168.99.100:28001/health -o /dev/null
000
+ echo 52
+ return 0
[Pipeline] readFile
[Pipeline] echo
status: 52
[Pipeline] echo
return appIsReady
[Pipeline] }
Will try again after 2.6 sec
[Pipeline] {
[Pipeline] echo
Checking Spring Boot status page via http://192.168.99.100:28001
[Pipeline] sh
[workspace] Running shell script
+ set +e
+ curl -f -sL -w %{http_code}\n http://192.168.99.100:28001/health -o /dev/null
200
+ echo 0
+ return 0
[Pipeline] readFile
[Pipeline] echo
status: 0
[Pipeline] echo
return appIsReady
[Pipeline] }
[Pipeline] // waitUntil
[Pipeline] echo
The clock micro-service have been started and is ready
[Pipeline] dir
Running in /var/jenkins_home/jobs/SEA Build Pipeline/workspace/validation
[Pipeline] {
[Pipeline] sh


https://hub.docker.com/_/
SEA Build Pipeline #4 Cons
Olivier
192.168.99.100:18080/job/SEA%20Build%20Pipeline/4/console
Jenkins
SEA Build Pipeline
#4
[Pipeline] echo
The clock micro-service have been started and is ready
[Pipeline] dir
Running in /var/jenkins_home/jobs/SEA Build Pipeline/workspace/validation
[Pipeline] {
[Pipeline] sh
[validation] Running shell script
+ ./build-image.sh
Sending build context to Docker daemon 557.1 kB
Sending build context to Docker daemon 1.114 MB
Sending build context to Docker daemon 1.671 MB
Sending build context to Docker daemon 2.228 MB
Sending build context to Docker daemon 2.785 MB
Sending build context to Docker daemon 3.342 MB
Sending build context to Docker daemon 3.899 MB
Sending build context to Docker daemon 4.456 MB
Sending build context to Docker daemon 5.014 MB
Sending build context to Docker daemon 5.571 MB
Sending build context to Docker daemon 6.128 MB
Sending build context to Docker daemon 6.685 MB
Sending build context to Docker daemon 7.242 MB
Sending build context to Docker daemon 7.799 MB
Sending build context to Docker daemon 8.356 MB
Sending build context to Docker daemon 8.471 MB
Step 1 : FROM node:4.3.1
---> ddb7a7253258
Step 2 : COPY src /opt/sea/validation/src
---> Using cache
---> db7dc5840af0
Step 3 : WORKDIR /opt/sea/validation/src
---> Using cache
---> f20965bd3171
Step 4 : ENTRYPOINT npm test
---> Using cache
---> b8e1f2e2ac5b
Successfully built b8e1f2e2ac5b
[Pipeline] }
[Pipeline] // dir
[Pipeline] sh
[workspace] Running shell script
+ docker run -e CLOCK_API_PREFIX=http://192.168.99.100:28001 sea/validation-api-tests
npm info it worked if it ends with ok
npm info using npm@2.14.12
npm info using node@v4.3.1
npm info pretest api-tests@0.1.0
npm info test api-tests@0.1.0
> api-tests@0.1.0 test /opt/sea/validation/src
> npm install && node jasmine2-runner.js
npm info it worked if it ends with ok
npm info using npm@2.14.12
npm info using node@v4.3.1
npm info preinstall api-tests@0.1.0
npm info build /opt/sea/validation/src

https://hub.docker.com/_/j × | SEA Build Pipeline #4 Cons × | Olivier

192.168.99.100:18080/job/SEA%20Build%20Pipeline/4/console

Jenkins › SEA Build Pipeline › #4

```
 ---> f20965bd3171
Step 4 : ENTRYPOINT npm test
 ---> Using cache
 ---> b8e1f2e2ac5b
Successfully built b8e1f2e2ac5b
[Pipeline] }
[Pipeline] // dir
[Pipeline] sh
[workspace] Running shell script
+ docker run -e CLOCK_API_PREFIX=http://192.168.99.100:28001 sea/validation-api-tests
npm info it worked if it ends with ok
npm info using npm@2.14.12
npm info using node@v4.3.1
npm info pretest api-tests@0.1.0
npm info test api-tests@0.1.0

> api-tests@0.1.0 test /opt/sea/validation/src
> npm install && node jasmine2-runner.js

npm info it worked if it ends with ok
npm info using npm@2.14.12
npm info using node@v4.3.1
npm info preinstall api-tests@0.1.0
npm info build /opt/sea/validation/src
npm info linkStuff api-tests@0.1.0
npm info install api-tests@0.1.0
npm info postinstall api-tests@0.1.0
npm info prepublish api-tests@0.1.0
npm info ok
Using API prefix: http://192.168.99.100:28001
Started
[32m.[0m[32m.[0m[32m.[0m


3 specs, 0 failures
Finished in 0.255 seconds
Validation completed.
npm info posttest api-tests@0.1.0
npm info ok
[Pipeline] echo
API tests have been run.
[Pipeline] stage (End)
Entering stage End
Proceeding
[Pipeline] echo
We have been through the entire pipeline
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

That's not a very practical way to check the status of API tests...

Help us localize this page

Page generated: Jun 2, 2016 11:16:33 AM UTC REST API Jenkins ver. 1.651.1

Future Work...

How could we improve our pipeline? (1)

- **At the moment, we only break the build if...**
  - The maven process fails (compilation, unit tests)
  - There is an issue with Docker
- **Often, we want the build to break if...**
  - The code quality is not good enough (we could query SonarQube and fail the build)
  - The integration / API tests fail (we could check the output generated by the Docker container and fail the build if an error as been raised).

How could we improve our pipeline? (2)

- **At the moment, we generate new Docker images for every build:**
  - but we don’t tag them (with versions). What would happen if multiple builds were started at the same time?
  - We can configure Jenkins to only run one build at the time, but we could also think about a versioning/tagging strategy.
  - We need to check disk usage...

How could we improve our pipeline? (2)

- **We could add other types of tests:**
  - If we have a Web UI on top of our micro-services, we might run functional tests with Selenium.
  - We could use a framework like **webdriver.io**, which allows us to write these tests in a “BDD” style (we would use automated acceptance tests in Gherkin).
  - We could run performance/load tests. For instance by controlling JMeter from our pipeline.
  - We could integrate a security assessment tool and again trigger it from our pipeline.

How could we improve our pipeline? (2)

- **We could make the results of our tests easier to analyze:**
  - In our implementation, we have to go in the console output and read what was generated by our supertest script. Not good.
  - Jenkins has a concept of “reporter” and there are many plugins that provide some kind of reporting capability. We could explore this ecosystem.
  - Or we could use **Probe Dock** (as demonstrated in the webcast provided at the beginning of the semester).

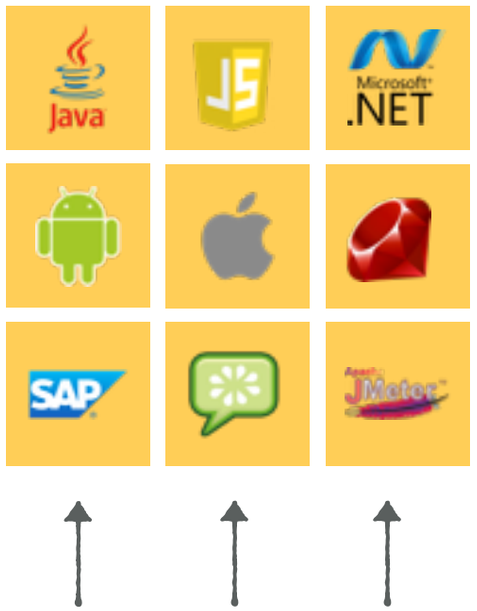
PROBE DOCK

Test & QA analytics for agile teams

MSE | MASTER OF SCIENCE IN ENGINEERING

PROBE DOCK

Test Probes



Dashboards, Displays & Devices

Test Results Collector

Reporting

Data Modeling & Storage

Gamification

Rule Engine

Analytics

Continuous Delivery Pipeline

https://hub.docker.com/_/s × | Probe Dock > probedock > × | Olivier

https://demo.probedock.io/probedock/Game-Dock-Engine

Probe Dock | Probe Dock ▾ | Dashboard | Reports | Projects | Help ▾ | Admin ▾ | oliechti | Sign out

Game Dock Engine

Tests by category

Jasmine2 99.2% / 235 tests

JUnit 0.8% / 2 tests

Details

8foj75gxe0p3

The Gamification Engine of the Probe Dock platform

Number of Tests
237
Number of Test Runs
478
Creation Date
Mon, Feb 22, 2016 5:32 PM
Last Update
Thu, May 19, 2016 4:40 PM

Contributors

User	Tests	Categories
yibnl	209	Jasmine2
oliechti	28	JUnit Jasmine2

Recent activity

ProbeDockJenkins ran some tests 14 days ago
1.0.0 JUnit
1

ProbeDockJenkins ran some tests 17 days ago
1.0.0 Jasmine2
207

ProbeDockJenkins ran some tests 17 days ago
1.0.0 JUnit
1

ProbeDockJenkins ran some tests 19 days ago
1.0.0 Jasmine2
207

ProbeDockJenkins ran some tests 19 days ago

Project health

One result in the report

Runner: ProbeDockJenkins

Version: 1.0.0

Run on: Thu, May 19, 2016 4:40 PM

Filter by version: 1.0.0

Filter by Runner: ProbeDockJenkins

Testing activity

No new tests were run over the last 20 days.

1

Probe Dock v0.1.28

https://hub.docker.com/_/s × | Probe Dock > probedock > × | Olivier

https://demo.probedock.io/probedock/reports/k73gnbk31zal

Probe Dock | Probe Dock | Dashboard | Reports | Projects | Help | Admin | oliechti | Sign out

Probe Dock

Latest Reports | May 16 20:32 by ProbeDockJenkins

Test Run Report - Mon, May 16, 2016 8:32 PM

Game Dock Engine 1.0.0 | Jasmine2

Details

- 207 test results
- Run in 12s 12ms

Summary

Health

GET applications should not return an empty array when applications do exist

Duration: 35ms

Filter by result

Filter by name

Any name

Filter by category

All categories

Filter by tag

All tags

Filter by ticket

All tickets

Probe Dock v0.1.28

https://hub.docker.com/_/ | Probe Dock > probedock | Olivier

https://demo.probedock.io/probedock/reports/k73gnbk31zal

Probe Dock | Probe Dock | Dashboard | Reports | Projects | Help | Admin | oliechti | Sign out

GET application should return an organization with expected point scales after having POSTed one

Game Dock Engine 1.0.0 | Jasmine2 | GET-application | Organizations-REST-endpoint

Run: May 16, 2016 8:32 PM **By:** ProbeDockJenkins **Duration:** 47ms

GET applications should be possible to limit items

Game Dock Engine 1.0.0 | Jasmine2 | Applications-REST-endpoint | GET-applications

Run: May 16, 2016 8:32 PM **By:** ProbeDockJenkins **Duration:** 79ms

GET applications should be possible to select an item page

Game Dock Engine 1.0.0 | Jasmine2 | Applications-REST-endpoint | GET-applications

Run: May 16, 2016 8:32 PM **By:** ProbeDockJenkins **Duration:** 70ms

GET applications should not return an empty array when applications do exist

Game Dock Engine 1.0.0 | Jasmine2 | Applications-REST-endpoint | GET-applications

Run: May 16, 2016 8:32 PM **By:** ProbeDockJenkins **Duration:** 35ms

GET applications should return a 200 HTTP status code in case of successful GET processing

Game Dock Engine 1.0.0 | Jasmine2 | Applications-REST-endpoint | GET-applications

Run: May 16, 2016 8:32 PM **By:** ProbeDockJenkins **Duration:** 99ms

GET applications should return an array

Probe Dock v0.1.28

https://hub.docker.com/_/ Probe Dock > probedock Olivier

https://demo.probedock.io/probedock/reports/64cs5bfieyz9

Probe Dock | Probe Dock | Dashboard | Reports | Projects | Help | Admin | oliechti | Sign out

Probe Dock

Latest Reports | May 16 20:32 by ProbeDockJenkins | May 14 06:53 by ProbeDockJenkins

Test Run Report - Sat, May 14, 2016 6:53 AM

Game Dock Engine 1.0.0 | Jasmine2

Details

- 207 test results
- Run in 11s 846ms

Summary

Health

PUT application should return a 404 HTTP status code in case of non existing application with a given id

Duration: 7ms

Filter by result

Filter by category

All categories

Filter by ticket

All tickets

Filter by name

Any name

Filter by tag

All tags

Probe Dock v0.1.28

https://hub.docker.com/_/ | Probe Dock > probedock | Olivier

https://demo.probedock.io/probedock/reports/64cs5bfieyz9

Probe Dock | Probe Dock | Dashboard | Reports | Projects | Help | Admin | oliechti | Sign out

Filter by name

Any name

All tags

List of results

Authorization should be refused in case of missing credentials

Game Dock Engine 1.0.0 | Jasmine2 | Levels-REST-endpoint | GET-levels | Authorization

Run: May 14, 2016 6:53 AM **By:** ProbeDockJenkins **Duration:** 4ms

```
1 failed expectation(s):
- Failed: Cannot read property 'status' of undefined

Stack trace (1st failed expectation):
TypeError: Cannot read property 'status' of undefined
    at Test._assertStatus (/opt/gamedock/bdd/src/node_modules/supertest/lib/test.js:229:10)
    at Test._assertFunction (/opt/gamedock/bdd/src/node_modules/supertest/lib/test.js:247:11)
    at Test.assert (/opt/gamedock/bdd/src/node_modules/supertest/lib/test.js:148:18)
    at assert (/opt/gamedock/bdd/src/node_modules/supertest/lib/test.js:127:12)
    at /opt/gamedock/bdd/src/node_modules/supertest/lib/test.js:124:5
    at Test.Request.callback (/opt/gamedock/bdd/src/node_modules/supertest/node_modules/superagent/lib/node/index.js:687:12)
    at ClientRequest.<anonymous> (/opt/gamedock/bdd/src/node_modules/supertest/node_modules/superagent/lib/node/index.js:639:10)
    at emitOne (events.js:77:13)
    at ClientRequest.emit (events.js:169:7)
```

DELETE application should be possible to delete an application with a linked organization

Game Dock Engine 1.0.0 | Jasmine2 | Applications-REST-endpoint | DELETE-application

Run: May 14, 2016 6:53 AM **By:** ProbeDockJenkins **Duration:** 3ms

```
1 failed expectation(s):
```

Probe Dock v0.1.28