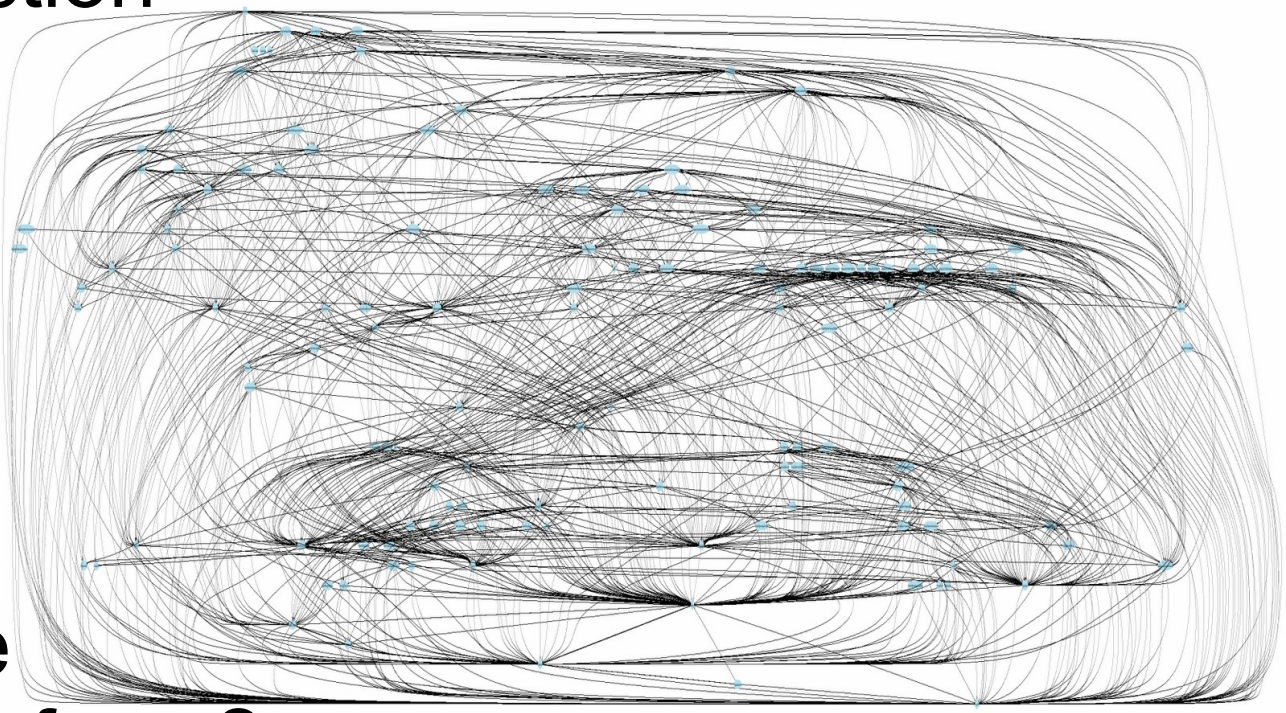
Advanced Programming

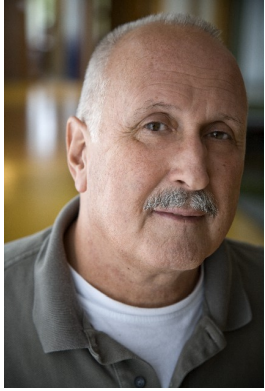
Overview & Introduction



- Nice to Meet You!
- Plan of the Course
- What is expected of you?
- Focus: Meta-programming

Nice to Meet You!

- My background
- Your background
  - Name
  - Some personal background
  - What is your experience with programming?
    - Languages
    - Projects
  - What do you expect to learn in this course?



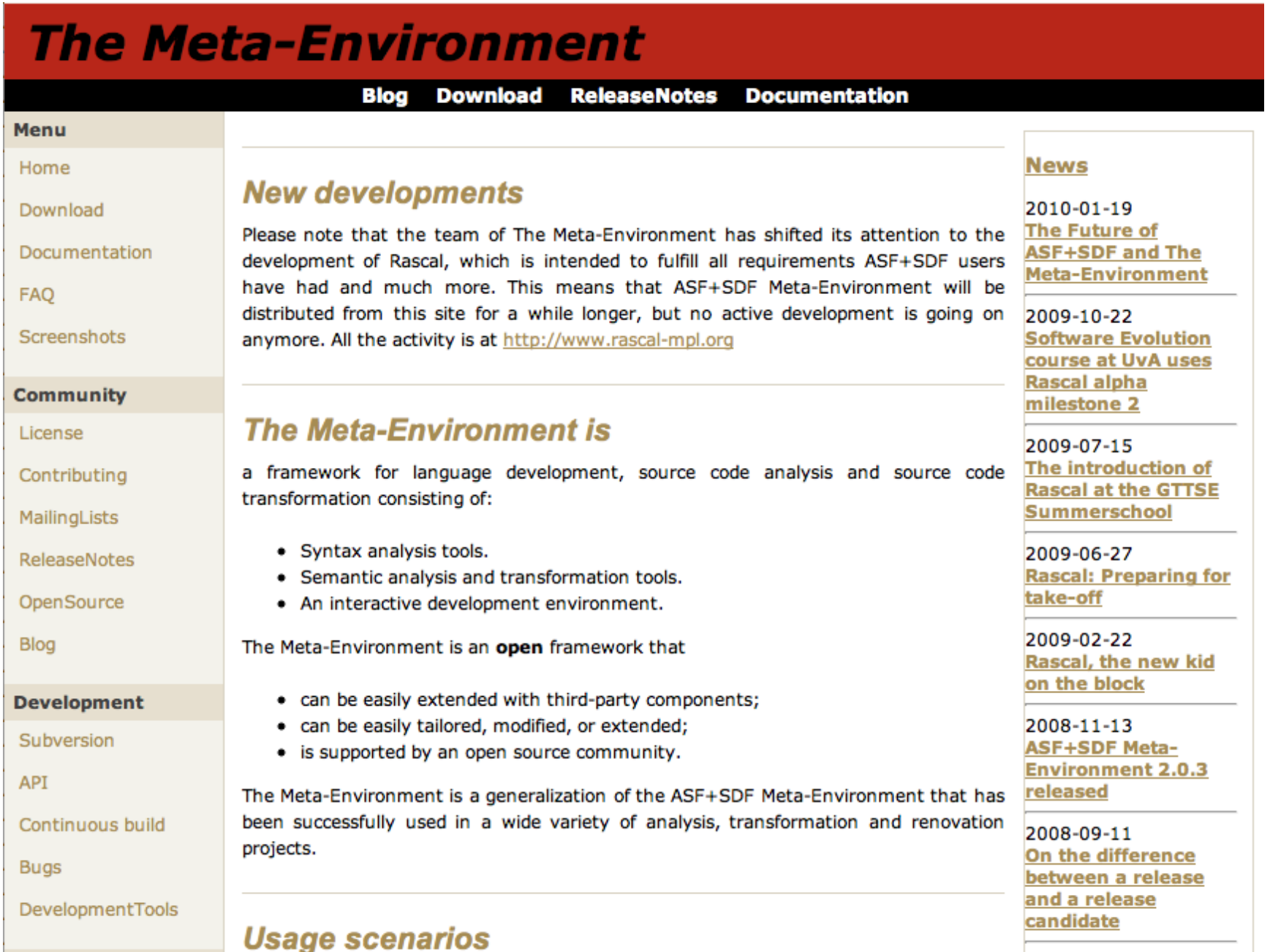
My background

- Paul Klint (see www.cwi.nl/~paulk)
- Professor in Software Engineering, UvA
- Former Head of Software Engineering, Centrum Wiskunde & Informatica (CWI); Now: research fellow
- Director Master Software Engineering, UvA
- Visiting professor Univ. of London (Royal Holloway)
- Project leader ATEAMS, INRIA, France

I am interested in

- Meta-programming
- Generic language technology
- Domain specific languages (DSLs)
- Analysis/transformation/renovation of software
- Interactive learning
- Patents
- Innovation (created 3 companies, most succesful: Software Improvement Group, www.sig.eu)

www.meta-environment.org



The Meta-Environment

Blog Download ReleaseNotes Documentation

Menu

Home
Download
Documentation
FAQ
Screenshots

Community

License
Contributing
MailingLists
ReleaseNotes
OpenSource
Blog

Development

Subversion
API
Continuous build
Bugs
DevelopmentTools

New developments

Please note that the team of The Meta-Environment has shifted its attention to the development of Rascal, which is intended to fulfill all requirements ASF+SDF users have had and much more. This means that ASF+SDF Meta-Environment will be distributed from this site for a while longer, but no active development is going on anymore. All the activity is at http://www.rascal-mpl.org

The Meta-Environment is

a framework for language development, source code analysis and source code transformation consisting of:

- Syntax analysis tools.
- Semantic analysis and transformation tools.
- An interactive development environment.

The Meta-Environment is an **open** framework that

- can be easily extended with third-party components;
- can be easily tailored, modified, or extended;
- is supported by an open source community.

The Meta-Environment is a generalization of the ASF+SDF Meta-Environment that has been successfully used in a wide variety of analysis, transformation and renovation projects.

Usage scenarios

News

2010-01-19
The Future of ASF+SDF and The Meta-Environment

2009-10-22
Software Evolution course at UvA uses Rascal alpha milestone 2

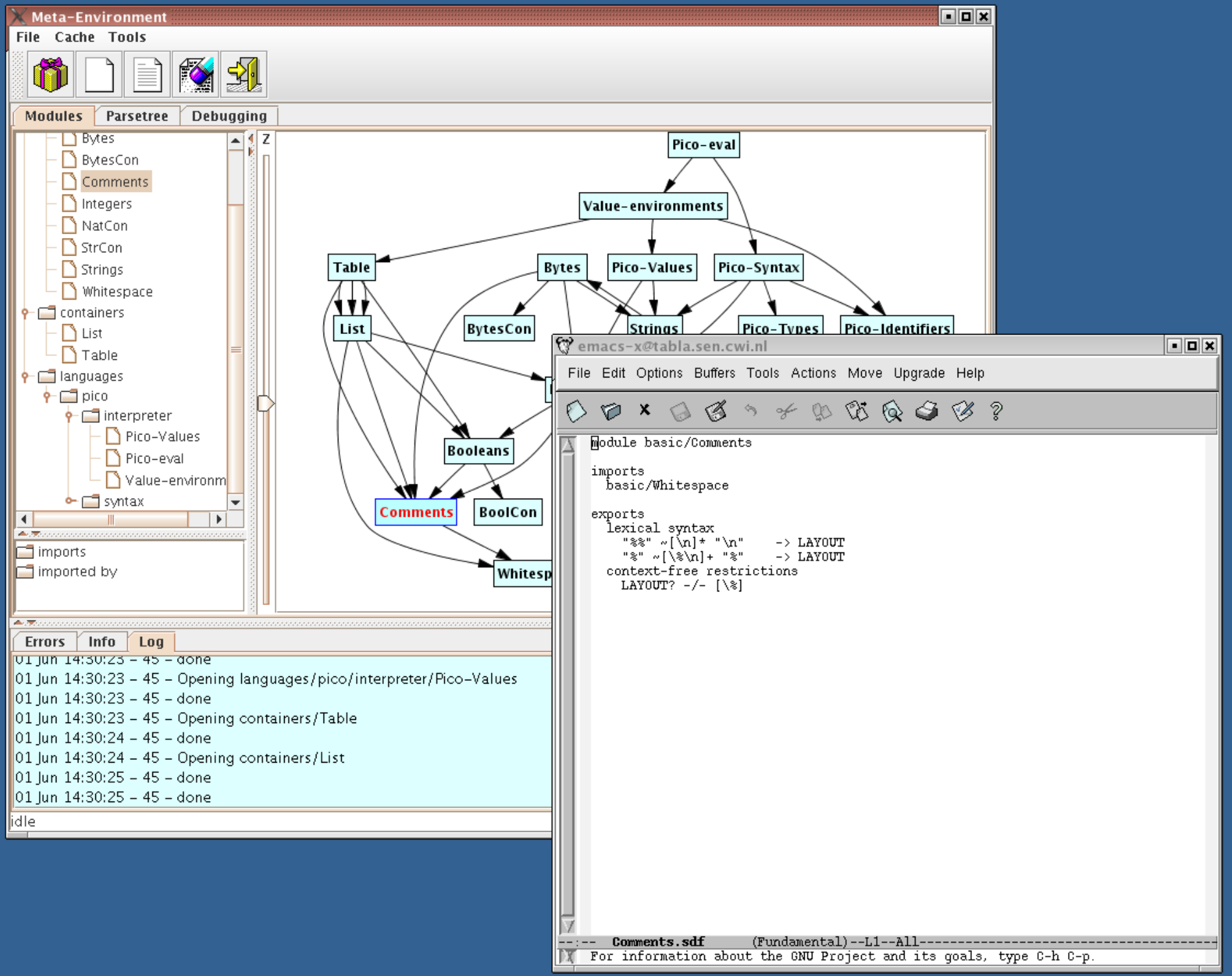
2009-07-15
The introduction of Rascal at the GTTSE Summerschool

2009-06-27
Rascal: Preparing for take-off

2009-02-22
Rascal, the new kid on the block

2008-11-13
ASF+SDF Meta-Environment 2.0.3 released

2008-09-11
On the difference between a release and a release candidate

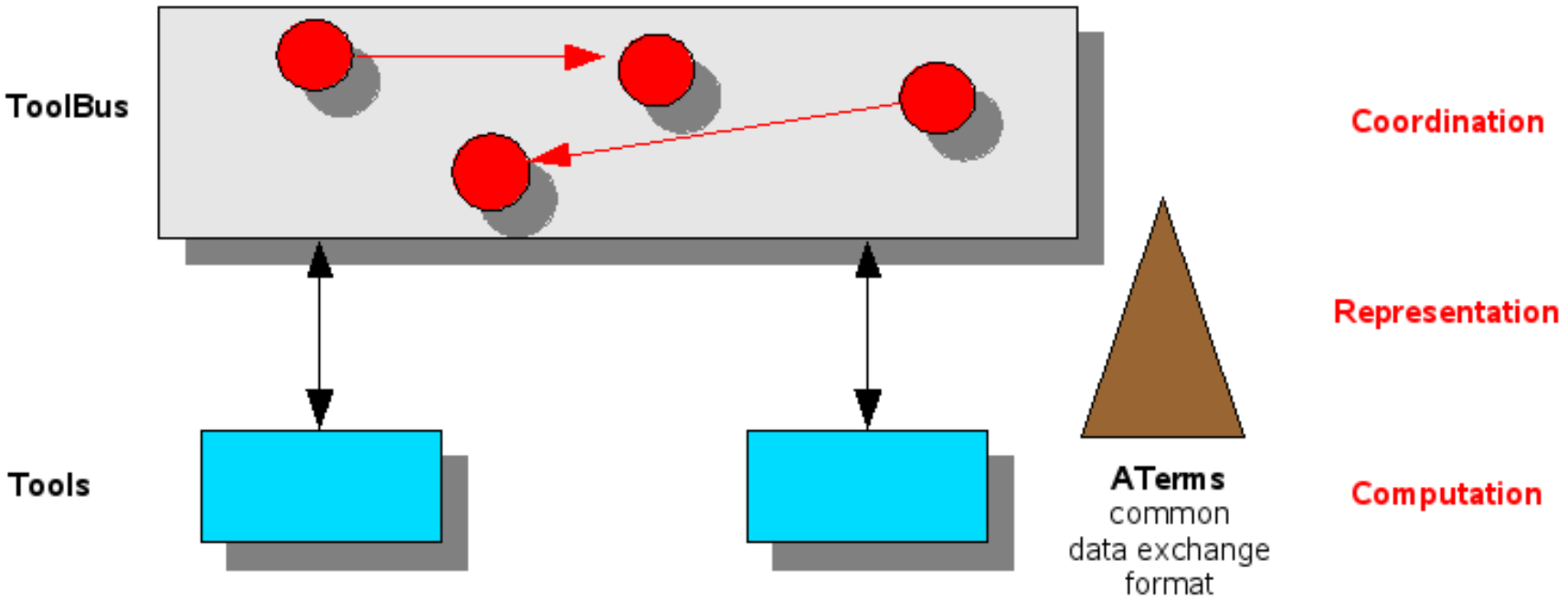

Meta-Environment
File Cache Tools
Modules Parsetree Debugging
Bytes
BytesCon
Comments
Integers
NatCon
StrCon
Strings
Whitespace
containers
List
Table
languages
pico
interpreter
Pico-Values
Pico-eval
Value-environm
syntax
imports
imported by
Pico-eval
Value-environments
Table
Bytes
Pico-Values
Pico-Syntax
List
BytesCon
Strings
Pico-Types
Pico-Identifiers
Booleans
Comments
BoolCon
Errors Info Log
01 Jun 14:30:23 – 45 – done
01 Jun 14:30:23 – 45 – Opening languages/pico/interpreter/Pico-Values
01 Jun 14:30:23 – 45 – done
01 Jun 14:30:23 – 45 – Opening containers/Table
01 Jun 14:30:24 – 45 – done
01 Jun 14:30:24 – 45 – Opening containers/List
01 Jun 14:30:25 – 45 – done
01 Jun 14:30:25 – 45 – done
idle
emacs-x@tabla.sen.cwi.nl
File Edit Options Buffers Tools Actions Move Upgrade Help
module basic/Comments
imports
basic/Whitespace
exports
lexical syntax
"%%" ~[\n]* "\n" -> LAYOUT
"%" ~[\%\n]+ "%" -> LAYOUT
context-free restrictions
LAYOUT? -/- [\%]
--:-- Comments.sdf (Fundamental)--L1--All--
For information about the GNU Project and its goals, type C-h C-p.

ToolBus

Description

The **ToolBus** is a software application architecture that utilizes a scripting language based on process algebra to describe the communication between software tools. A **ToolBus** script describes a number of processes that can communicate with each other and with tools living outside the **ToolBus**. A language-dependent adapter that translates between the internal **ToolBus** data format and the data format used by the individual tools makes it possible to write every tool in the language best suited for the task(s) it has to perform.

The global architecture is as follows:



The processes inside the **ToolBus** (red circles) take care of the coordination of the whole application. ATerms are used to represent the data that is shipped between **ToolBus** and tools. The tools (blue boxes) carry out the actual work. They may be implemented in different languages (C, Java, Perl, ASF+SDF, ...) and may also run on different computers.

The **ToolBus** comes with a viewer that allows the debugging of large applications. This screenshot illustrates the **ToolBus** viewer in action. It shows the Meta-Environment that has justed started up and a user has requested to open a module via the **MetaStudio** user-interface.

www.rascal-mpl.org

Rascal - Meta Programming Language

Home

Download
GettingStarted
Documentation
Q&A Forum
Bugreporting
Teaching
Stories
Facts

Developers

SVN
Contributing

Welcome to Rascal

Rascal is a domain specific language for source code analysis and manipulation a.k.a. meta-programming. It is currently being developed and tested at CWI. No formal release has been made yet, but we do provide alpha quality "previews" on the **Download** page.

For information please contact:
mailto:Paul.Klint@cwi.nl,Jurgen.Vinju@cwi.nl,Tijs.van.der.Storm@cwi.nl

Download it to try it!

Rascal is powered by:

- Java
- Itself
- Eclipse IMP

Edit | Attach | Ref-By | Printable | Prefs | Stats | Menu | News | Header | More | Changes |

Copyright © 2010, CWI; This site is maintained using TWiki

Available online at www.sciencedirect.com
ScienceDirect
Science of Computer Programming 64 (2007) 264–285
Science of Computer Programming
www.elsevier.com/locate/scico

About "trivial" software patents: The IsNot case☆

Jan A. Bergstra[a,b], Paul Klint[c,a,*]

[a] *Informatics Institute, University of Amsterdam, Netherlands*
[b] *Faculty of Philosophy, University of Utrecht, Netherlands*
[c] *Centrum voor Wiskunde en Informatica (CWI), Software Engineering Department, Netherlands*

Available online 30 October 2006

Abstract

So-called "trivial" software patents undermine the patenting system and are detrimental for innovation. In this paper we use a case-based approach to get a better understanding of this phenomenon. First, we establish a baseline for studying the relation between software development and intellectual property rights by formulating a life cycle for the patenting system as well as three variations of the software life cycle: the defensive patent-aware software life cycle that prevents patent infringements, the more offensive patent-based software life cycle that aims both at preventing infringements and at creating new patents, and the IPR-based software life cycle that considers all forms of protection of intellectual property rights including copyright and secrecy.

Next, we study an application for a software patent concerning the inequality operator and a granted European patent on memory management. We also briefly mention other examples of trivial patents. These examples serve to clarify the issues that arise when integrating patents in the software life cycle.

In an extensive discussion, we cover the difference between expression and idea, the role of patent claims, software patents versus computer implemented inventions, the role of prior art, implications of software patents for open source software, for education, and for government-funded research. We conclude the discussion with the formulation of an "integrity axiom" for software patent authors and owners and sketch an agenda for software patent research.

We conclude that patents are too important to be left to lawyers and economists and that a complete reinterpretation of the patenting system from a software engineering perspective is necessary to understand all ramifications of software patents. We end with explicit conclusions and policy recommendations.

Keywords: Software patents; Trivial patents; Intellectual property rights; Software engineering; Patent life cycle; Software engineering life cycle; Open source software; Prior art; Patent claims; Patent policy

Paul Klint: Selected Publications

Disclaimer

2012

- Paul Klint and Atze van der Ploeg: Compositional 2D Graphics for Free, in preparation.
- Mark Hills, Paul Klint and Jurgen Vinju: Exploring PHP Feature Usage for Static Analysis, submitted for publication.
- Mark Hills, Paul Klint and Jurgen Vinju: Meta-Language Support for Type-Safe Access to External Resources, accepted for publication in Software Language Engineering 2012 (SLE 2012), 2012.
- Mark Hills, Paul Klint and Jurgen Vinju: Scripting a Refactoring with Rascal and Eclipse, Fifth Workshop on Refactoring Tools 2012 (WRT 2012), To be published in the ACM Digital Library, 2012. pdf.
- Mark Hills, Paul Klint, and Jurgen J. Vinju: Program Analysis Scenarios in Rascal, 9th International Workshop on Rewriting Logic and Its Applications (WRLA 2012), Invited Paper, To be published in LNCS, 2012, pdf
- Mark Hills, Paul Klint, Tijs van der Storm, Jurgen J. Vinju: A One-Stop-Shop for Software Evolution Tool Construction. ERCIM News 2012(88). html

2011

- P. Klint, B. Lisser and A. van der Ploeg, Towards a One-Stop-Shop for Analysis, Transformation and Visualization of Software, Proceedings Solftware Language Engineering, SLE2011. In: Proceedings of the Fourth International Conference on Software Language Engineering (SLE 2011), Editors: Saraiva, J. and Aßmann, U. and Sloane, A.M.. Springer, 1-18, 2011. pdf
- P. Klint, Tijs van der Storm, Jurgen Vinju, EASY Meta-programming with Rascal, in João M. Fernandes, Ralf Lämmel, Joost Visser and João Saraiva (eds), Generative and Transformational Techniques in Software Engineering III International Summer School, GTTSE 2009, Braga, Portugal, July 6-11, 2009. Revised Papers, LNCS Volume 6491, 2011, 222--289. bib, pdf, [Springer]
- Basten, H.J.S. - Klint, P. - Vinju, J.J.: Ambiguity Detection: Scaling to Scannerless In: Proceedings of the Fourth International Conference on Software Language Engineering (SLE 2011), Editors: Saraiva, J. and Aßmann, U. and Sloane, A.M.. Springer, 303-323, 2011. pdf
- Bos, J. van den - Hills, M.A. - Klint, P. - Storm, T. van der - Vinju, J.J.: Rascal: From Algebraic Specification to Meta-Programming In: Proceedings Second International Workshop on Algebraic Methods in Model-based Software Engineering (AMMSE 2011), Series: Electronic Proceedings in Theoretical Computer Science, Vol. 56, pp. 15 - 32, Editors: Rusu, V and Durán, F., 2011 pdf
- Klint, P. - Hills, M.A. - Storm, T. van der - Vinju, J.J.: A case of visitor versus interpreter pattern In: Proceedings of International Conference on Objects, Models, Components and Patterns 2011 (49), 228-243,2011. pdf
- Klint, P. - Vinju, J.J. - Hills, M.A.: RLSRunner: Linking Rascal with K for program analysis In: Proceedings of International Conference on Software Language Engineering 2011, 344-353, Springer, 2011. pdf
- Hills, M.A. - Izamaylova, A. - Klint, P. - Ploeg, A., van der - Storm, T. van der - Vinju, J.J.: The Rascal meta-programming language - a lab for software analysis, transformation, generation & visualization In: Proceedings of ICT.Open 2011, 353--358, 2011. pdf

2010

- P. Klint, T. van der Storm, J.J. Vinju: On the Impact of DSL Tools on the Maintainability of Language Implementations In: Proceedings of Workshop on Language Descriptions, Tools and Applications 2010. ACM, 2010. bib, pdf, [ACM]

Your Background

- Name
- Some personal background.
- What is your experience with programming?
  - Languages
  - Projects
- What do you expect to learn in this course?

Plan for this Course

“Better Software via Meta-Programming”

Programme (1/2)

- Overview
- Quick intro to Rascal
- Quick intro to Visualization
- Lists
- Sets
- Relations
- Datatypes: Trees and Algebraic Datatypes

Programme (2/2)

- Syntax and Parsing
- A simple expression language
- A Lisp interpreter
- Type checking
- Compilation
- A software engineering perspective on meta-programming

Tools and Background Material

- Rascal website: http://www.rascal-mpl.org/
- Downloading and installing: http://www.rascal-mpl.org/Rascal/EclipseUpdate
- All Tutor courses: http://tutor.rascal-mpl.org/
- The concepts discussed in this course: http://tutor.rascal-mpl.org/Courses/Rascalopedia/Rascalopedia.html
- Ask all your questions about Rascal: http://ask.rascal-mpl.org
- The blackboard site: http://blackboard.auc.nl, course 20112012Advanced Programming

What is expected of You?

- Attend and actively participate in classes
  - Ask questions
  - Participate in discussions
  - Present your results
- Install Rascal
- Turn in home work
- Participate in “photo wall”

Computational Thinking

- See Jeannette Wing's paper: http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf
- Key Idea: relate computer science ideas to daily life.
- Objective: experiment to explore whether this helps us to understand concepts.

List

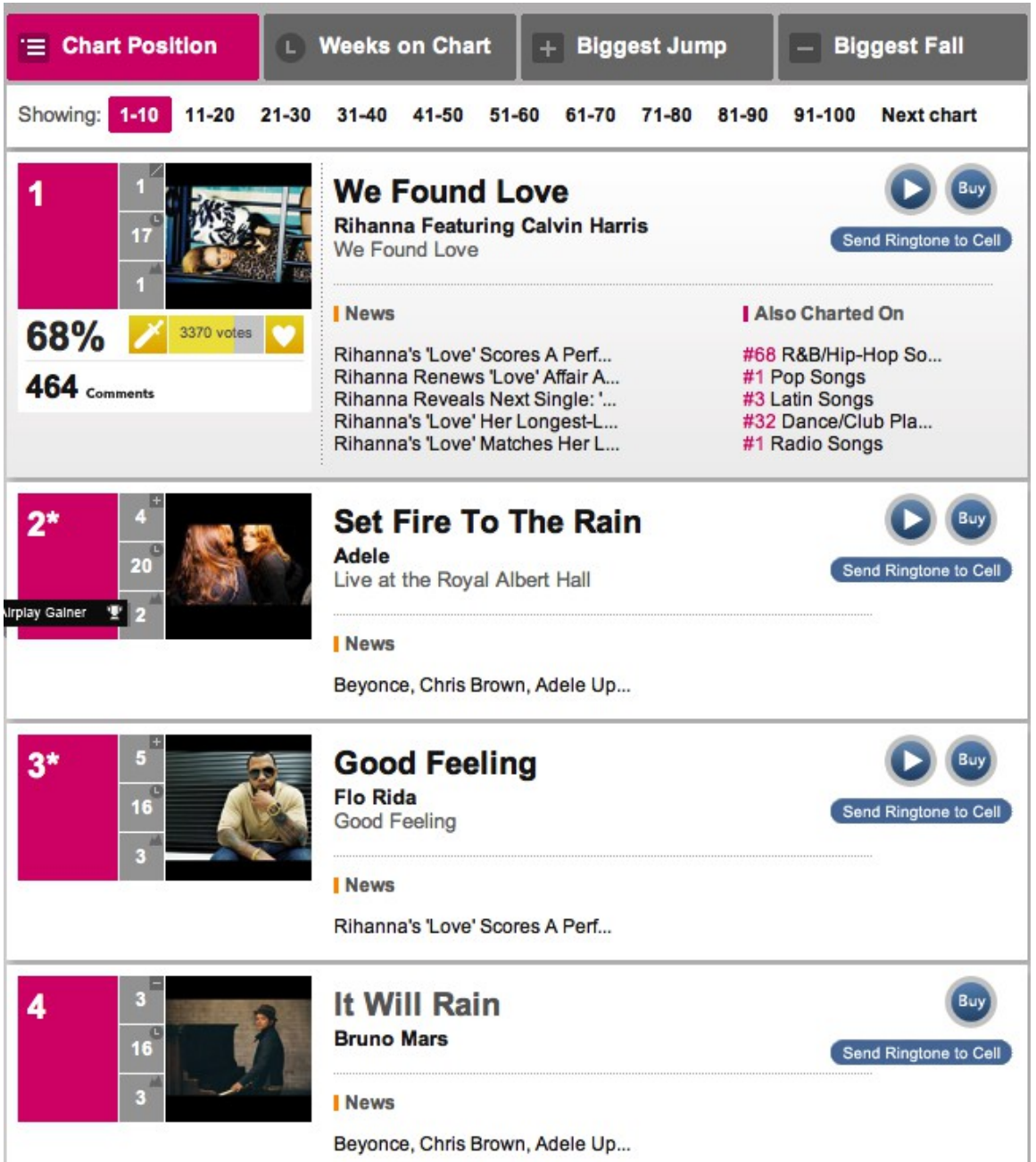
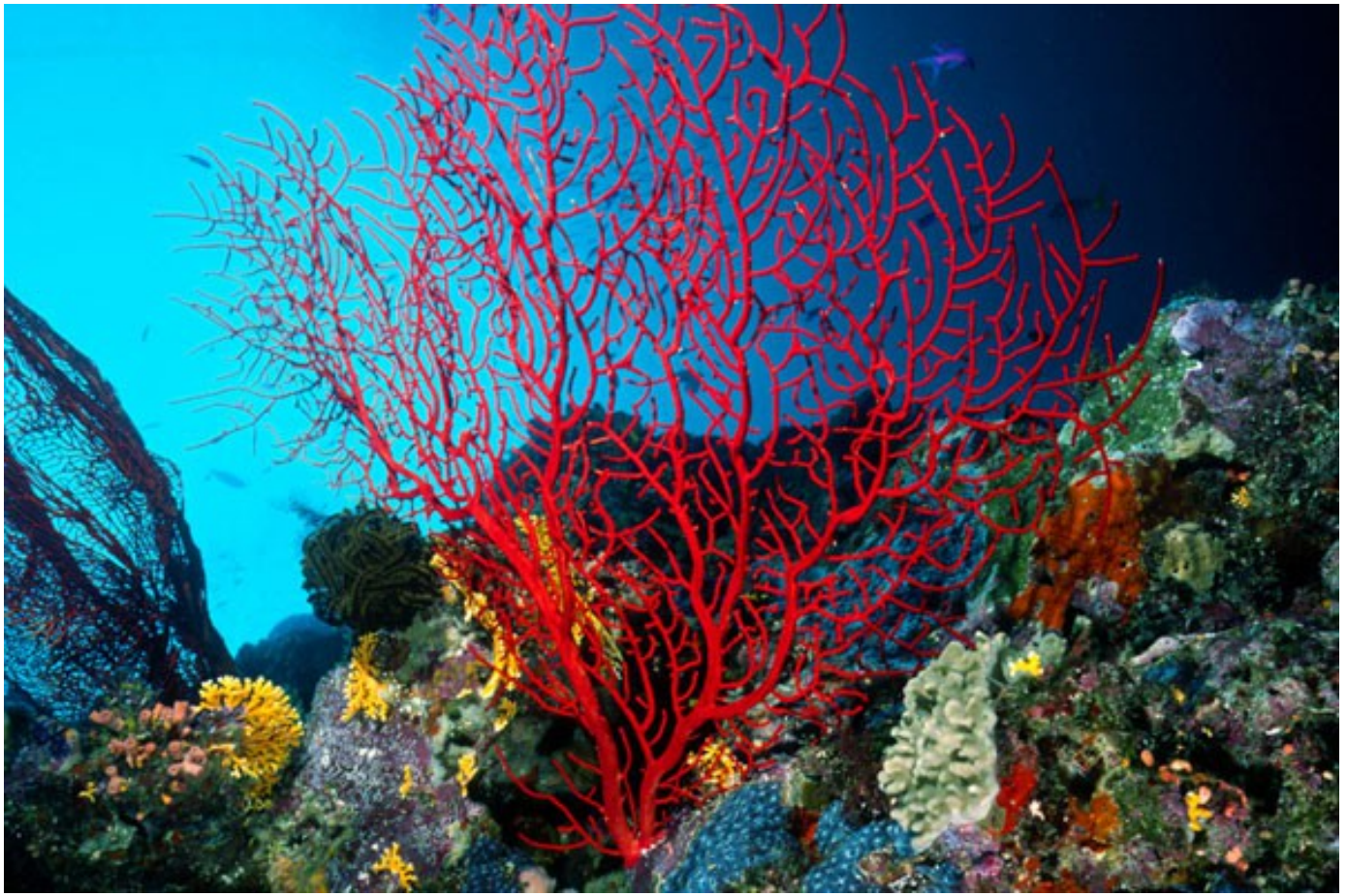


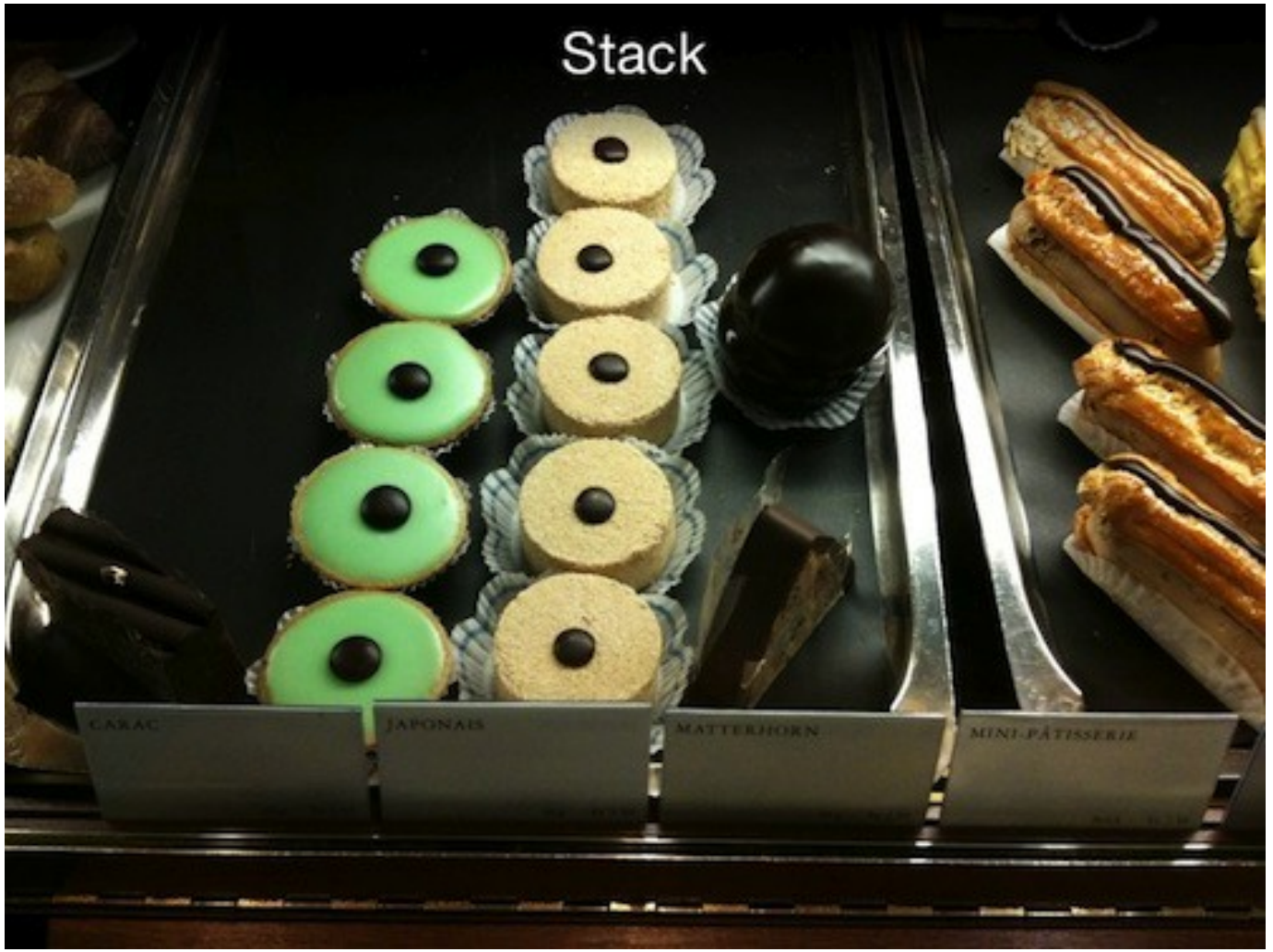
Chart Position
Weeks on Chart
Biggest Jump
Biggest Fall
Showing: 1-10 11-20 21-30 31-40 41-50 51-60 61-70 71-80 81-90 91-100 Next chart
1
1
17
1
68%
3370 votes
464 Comments
We Found Love
Rihanna Featuring Calvin Harris
We Found Love
Buy
Send Ringtone to Cell
News
Rihanna's 'Love' Scores A Perf...
Rihanna Renews 'Love' Affair A...
Rihanna Reveals Next Single: '...
Rihanna's 'Love' Her Longest-L...
Rihanna's 'Love' Matches Her L...
Also Charted On
#68 R&B/Hip-Hop So...
#1 Pop Songs
#3 Latin Songs
#32 Dance/Club Pla...
#1 Radio Songs
2*
4
20
2
Airplay Gainer
Set Fire To The Rain
Adele
Live at the Royal Albert Hall
Buy
Send Ringtone to Cell
News
Beyonce, Chris Brown, Adele Up...
3*
5
16
3
Good Feeling
Flo Rida
Good Feeling
Buy
Send Ringtone to Cell
News
Rihanna's 'Love' Scores A Perf...
4
3
16
3
It Will Rain
Bruno Mars
Buy
Send Ringtone to Cell
News
Beyonce, Chris Brown, Adele Up...

Tree



Set


NIGERIA
SILVER JUBILEE
REPUBLIQUE FRANCAISE
MAGYAR POSTA
PERU
AIR MAIL



Credit: Tudor Girba

Queue

Credit: Tudor Girba

Questions about using computational thinking

- Do you find this an interesting approach?
- Do you want to try it?
- How can we organize it?
  - In some exercises you are asked to find pictures that illustrate a concept.
  - You are at the constant lookout for related pictures.
  - We maintain a photo wall somewhere (Facebook? A Volunteer?)

Problem: The Software Volcano

Mt. Etna, Sicily, Italy

The Software Volcano: Languages

Distribution of languages in use, worldwide

Language	Used in % of total
COBOL	30
Assembler	10
C	10
C++	10
550 other languages	40

- For mainframe applications 80% is COBOL!
- Figures taken from Capers Jones (Software Productivity Research)

Software Volcano: Volume

- The total volume of software is estimated at
  $7 * 10^9$ *function points*
- 1 FP = 128 lines of C or 107 lines of COBOL
- The volume of the volcano is
  - 750 Giga-lines of COBOL code, or
  - 900 Giga-lines of C code

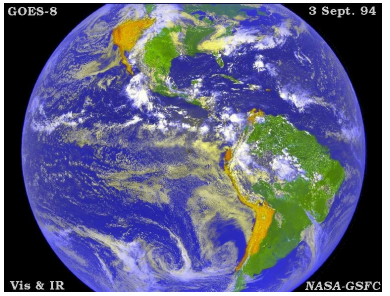
Printed on paper we can wrap planet Earth 9 times!

Software Volcano: Defects

- Observation:
  - on average 5 errors (bugs) per function point
  - includes errors in requirements, design, coding, documentation and bad fixes
- The software volcano, world-wide, contains

$5 * 7 * 10^9$ Bugs = 35 Giga Bugs

This means 6 bugs per human being on planet Earth!



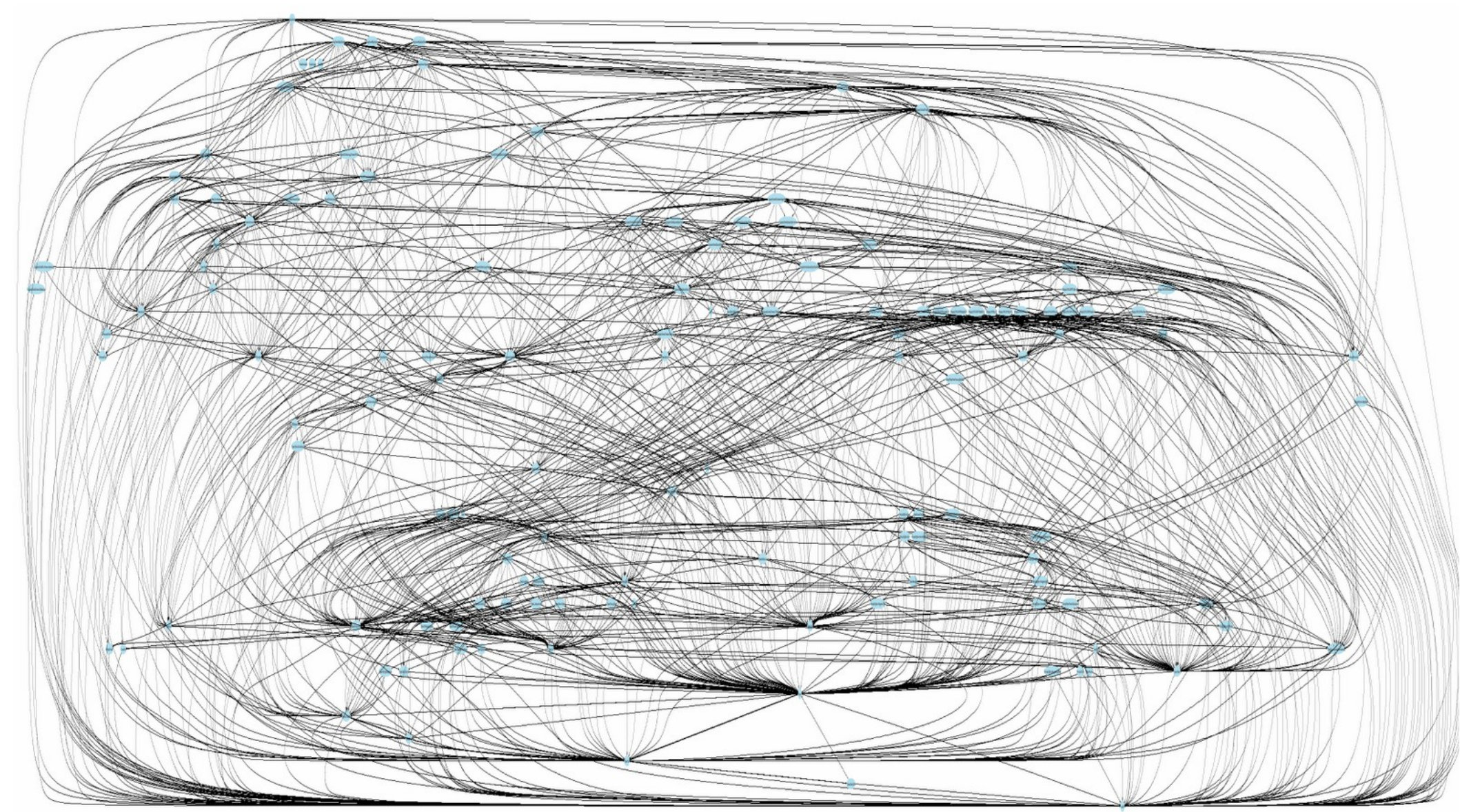
Work distribution of programmers

Year	New projects	Enhancements	Repairs	Total
1950	90	3	7	100
1960	8,500	500	1,000	10,000
1970	65,000	15,000	20,000	100,000
1980	1,200,000	600,000	200,000	2,000,000
1990	3,000,000	3,000,000	1,000,000	7,000,000
2000	4,000,000	4,500,000	1,500,000	10,000,000
2010	5,000,000	7,000,000	2,000,000	14,000,000
2020	7,000,000	11,000,000	3,000,000	21,000,000

Now: 60% of the programmers work on enhancement and repair

In 2020: only 30% of all programmers will work on new software

A “real” system



Message

- When an industry approaches 50 years of age it takes more workers to perform maintenance than to build new products (*ex:* automobile industry)
- Maintenance and renovation of existing software become more and more important: avoid that the software volcano explodes
- Meta-programming is the technique of choice to improve upon this situation.

Solution: Meta-Programming

- Normal programs:
  - Read data
  - Produce data
- Meta-programs:
  - Read programs (and other info related to programs such as version repositories, test data, ...)
  - Produce other programs (*refactoring*) or data about those programs (*metrics*, *bug locations*, *visualizations*)

Assignments Week 1
due Friday 7, 16:00 via Blackboard

- Install Eclipse and Rascal
- Search for different definitions of “meta-programming”; argue which one is best.
- Search for at least 3 different applications of meta-programming; summarize each and relate to your favorite definition.
- Explain advantages and disadvantages of meta-programming
- Search for pictures that illustrate meta-programming.
- **We will discuss your solutions in class**