Open Source:
Lessons for Research and Industry

Ralph Müller
Director
Eclipse Foundation
@ralph_mueller

May 2014

be nice to nerds …

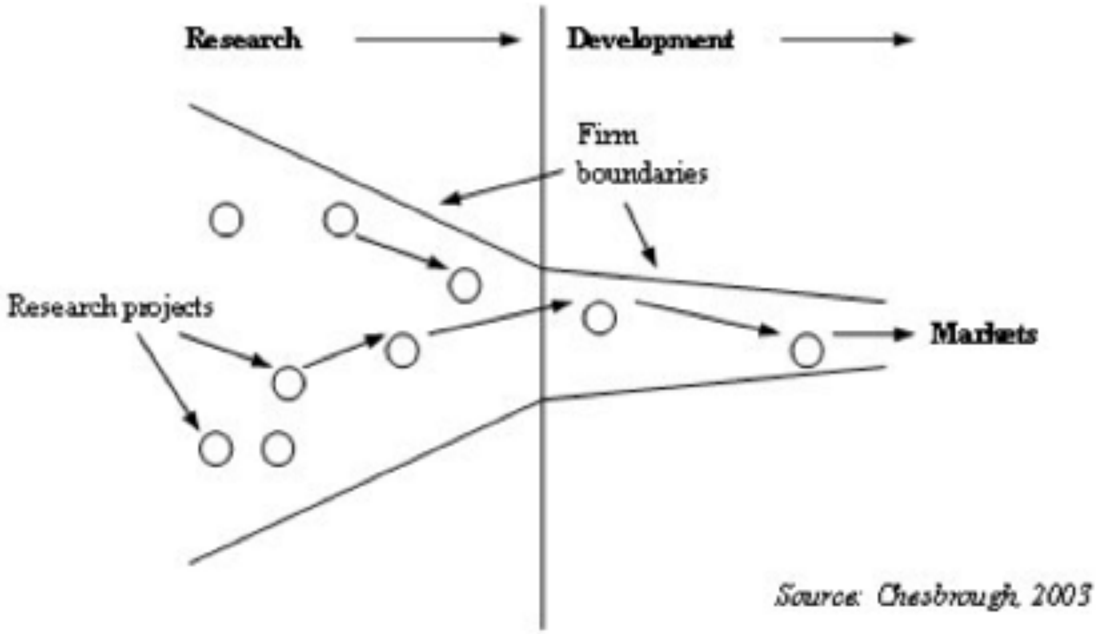
Hardware Age

Game Changer

The Age of the Packages

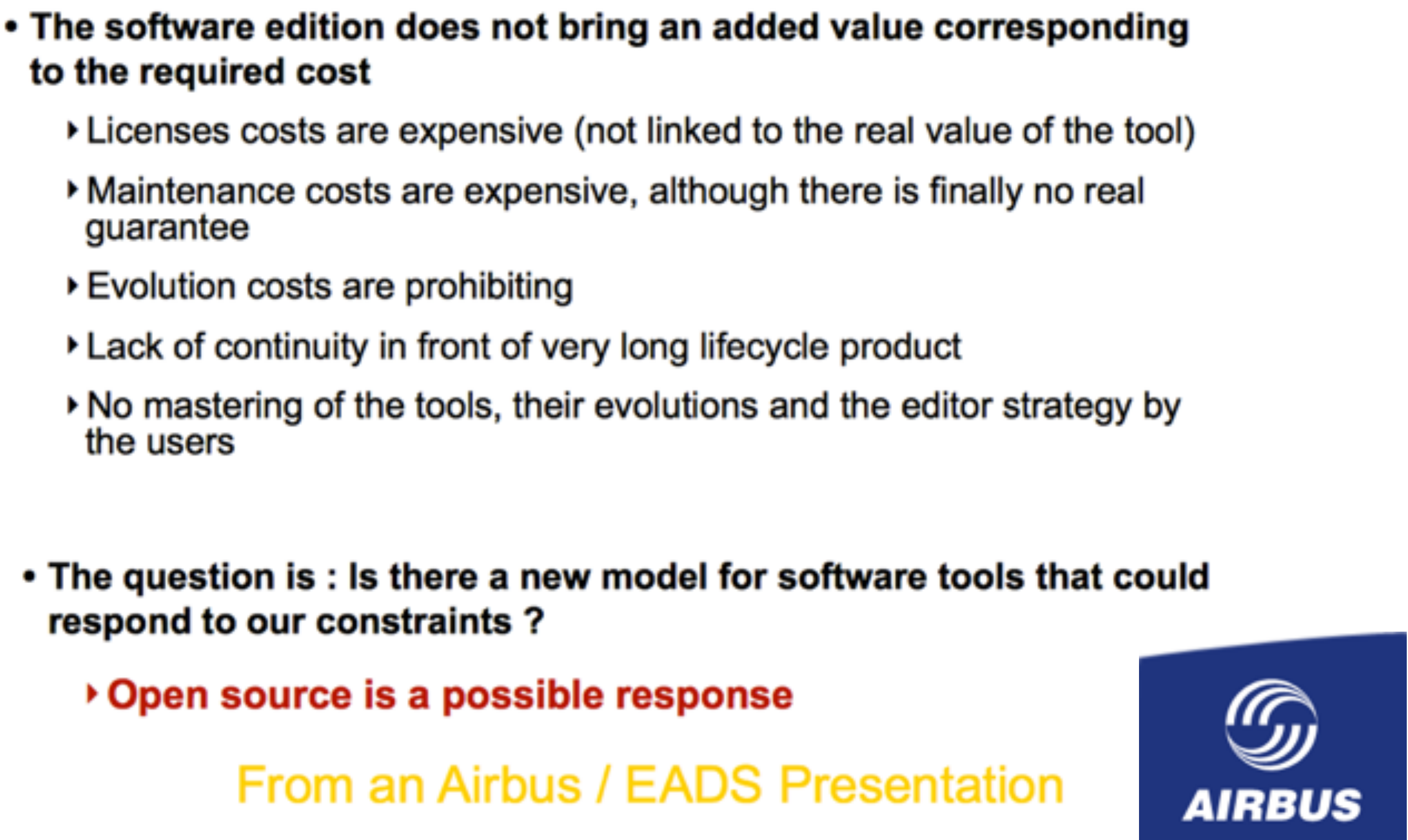
Microsoft
SAP
COREL
software AG
dSPACE
ORACLE
IBM
Rational. software

Ivory Towers



Source: Chesbrough, 2003

Case Study: Airbus

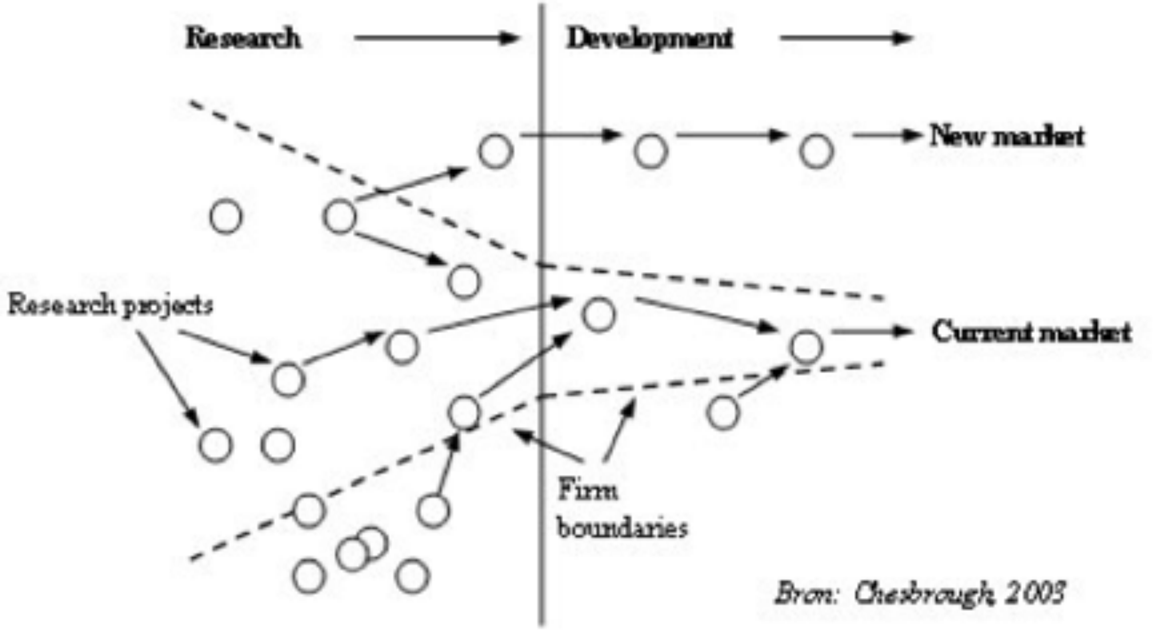


- **The software edition does not bring an added value corresponding to the required cost**
  - Licenses costs are expensive (not linked to the real value of the tool)
  - Maintenance costs are expensive, although there is finally no real guarantee
  - Evolution costs are prohibiting
  - Lack of continuity in front of very long lifecycle product
  - No mastering of the tools, their evolutions and the editor strategy by the users

- **The question is : Is there a new model for software tools that could respond to our constraints ?**
  - **Open source is a possible response**

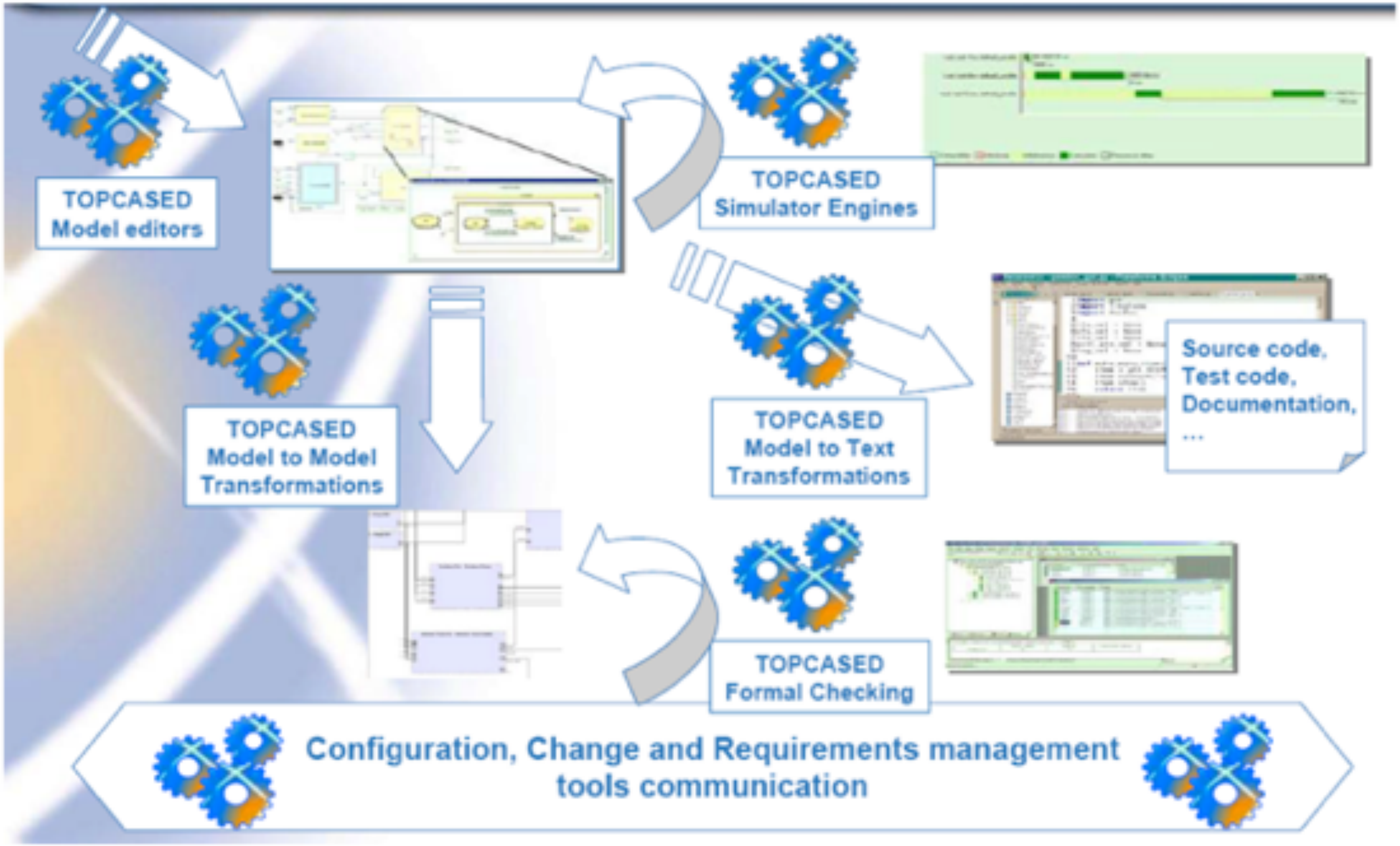
From an Airbus / EADS Presentation

>> Open Up<<



Bron: Chesbrough, 2003

TOPCASED The Open-Source Toolkit for Critical Systems


TOPCASED Model editors

TOPCASED Simulator Engines

TOPCASED Model to Model Transformations

TOPCASED Model to Text Transformations

Source code, Test code, Documentation, …

TOPCASED Formal Checking

Configuration, Change and Requirements management tools communication

Life in an Ecosystem

Industries
AIRBUS
cnes
EADS ASTRIUM
Continental
CS Communication & Systèmes
THALES
Turbomeca SAFRAN Group
Sopra group
Rockwell Collins
Atos Origin
AKKA Technologies
SOGETI High Tech

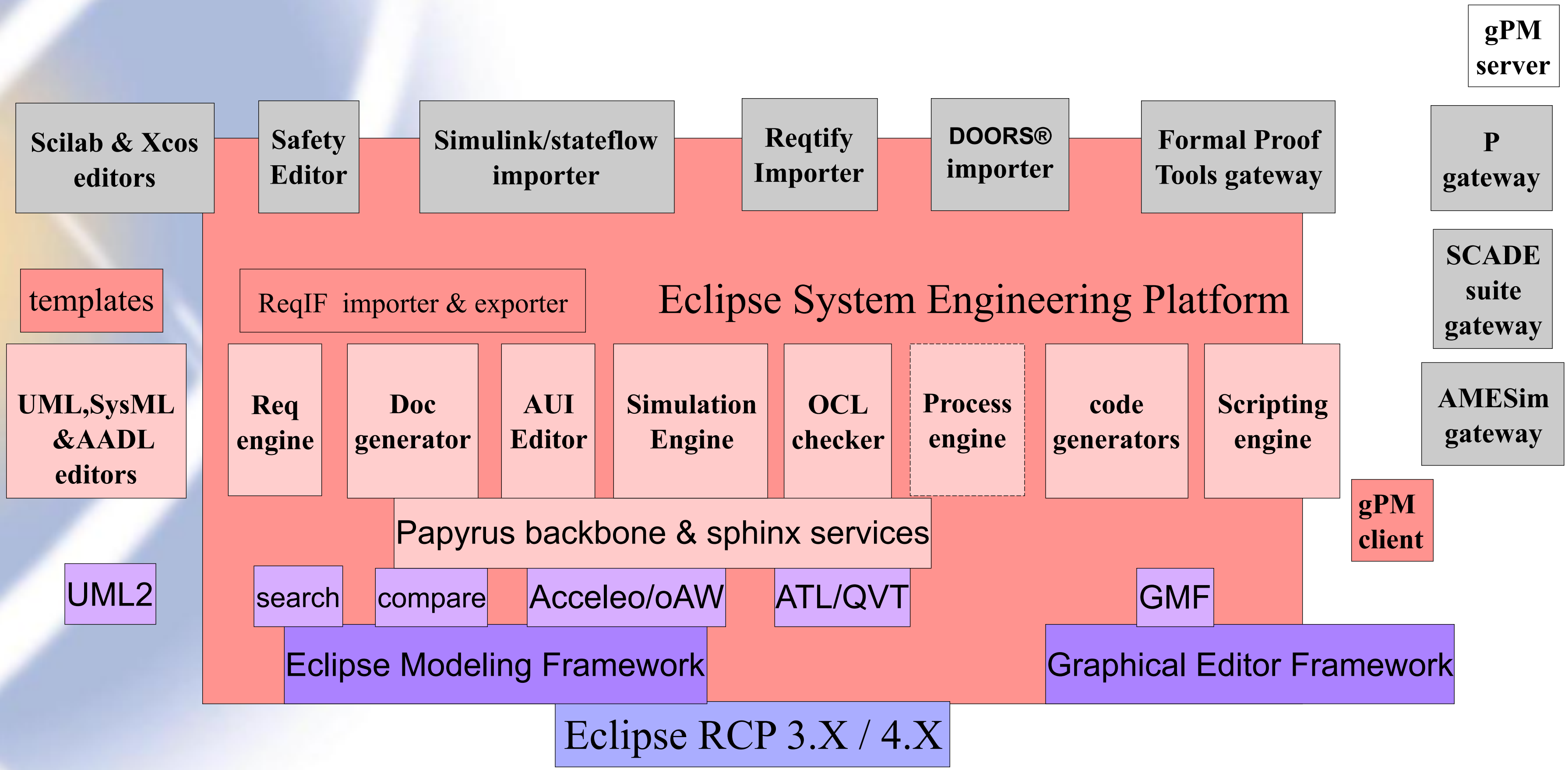
SMEs
Tectosages
anyware technologies
SODIFRANCE L'inspiration technologique
Ellidiss Technologies
MICOUIN Systems Engineering
AdaCore The GNAT Pro Company
geensys
Obeo Model Driven Company
CLEARSY System Engineering

INP ENSEEIHT
Universidade Federal de Santa Catarina
ENSIETA
mips
ESEO
UNIVERSITÉ PAUL SABATIER TOULOUSE III
INSA Toulouse
UNIVERSITÉ DE TOULOUSE LE MIRAIL
UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR
TELECOM ParisTech
School/Universities

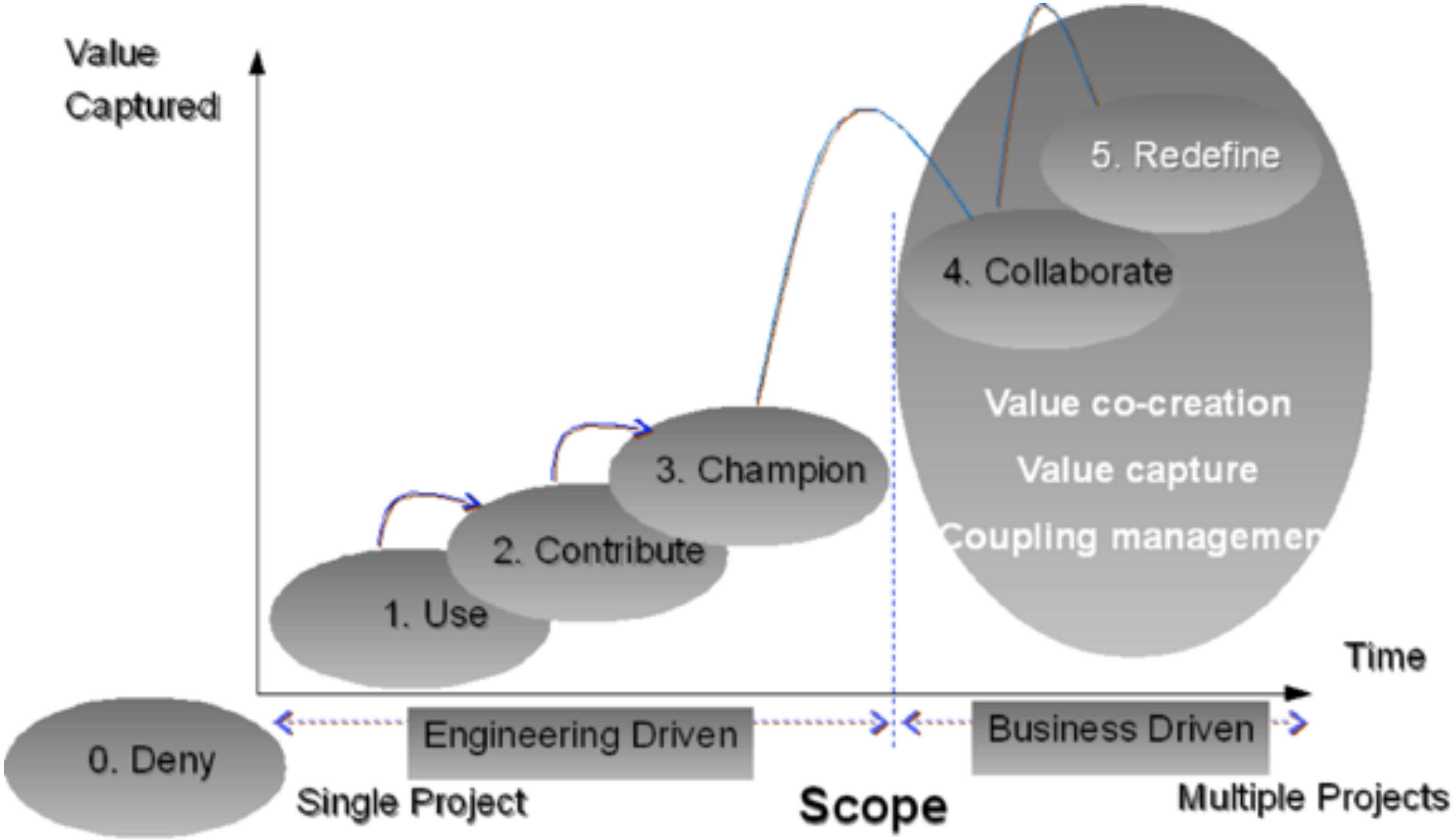
INRIA
VASY
Atlas
ONERA
IRISA
Triskel
FéRIA Fédération de Recherche en Informatique et Automatique
LAAS
IRIT
Carnegie Mellon Software Engineering Institute
cea Cesta
cea list
LIP6
Laboratories

Eclipse contribution roadmap – mid term (2012-2014)

- Contribution to the Eclipse foundation
  - Propose most components to the Eclipse foundation
  - Propose TOPCASED as “Eclipse System Engineering Platform”



Bizz Model


Value
Captured
5. Redefine
4. Collaborate
Value co-creation
Value capture
Coupling managemen
3. Champion
2. Contribute
1. Use
Time
0. Deny
Engineering Driven
Business Driven
Single Project
Scope
Multiple Projects

THE HANSEN REPORT
ON AUTOMOTIVE ELECTRONICS

Vol. 21, No. 6 ◆ www.hansenreport.com ◆ July/August 2008

Progress on Integrated Software Development Tools

Carmakers Consider Open Source and Commercial Solutions

Software is the means by which a carmaker can stand apart from other carmakers. Carmakers who get software development right are able to bring new and better features and functions to the market faster and at lower costs than those who don't. And as software com-

BMW Leads Open Source Infotainment Initiative

Daimler On Board; GM and PSA Will Probably Join

Already in hypercompetitive mode, the infotainment industry is facing further upheaval following BMW's recent announcement that it has joined with Intel, Wind River and others in an effort to promote an open source infotainment and communications system platform based on Linux. A viable open source platform will force infotainment suppliers to change the

Because it takes way too much time to develop new infotainment features and functions, embedded systems today tend to be not only much more expensive than they should be, but also lacking in the most up-to-date technologies.

Embedded infotainment systems, with their big displays and driver-safe user interfaces, need to be periodically updated to keep pace with the latest portable consumer electronics they are supposed to

Dürfen wir uns zivile Sicherheitssysteme mit Closed Source Software überhaupt noch leisten?

Denkansätze am Beispiel der ETCS-Migration bei der Eisenbahnsignaltechnik

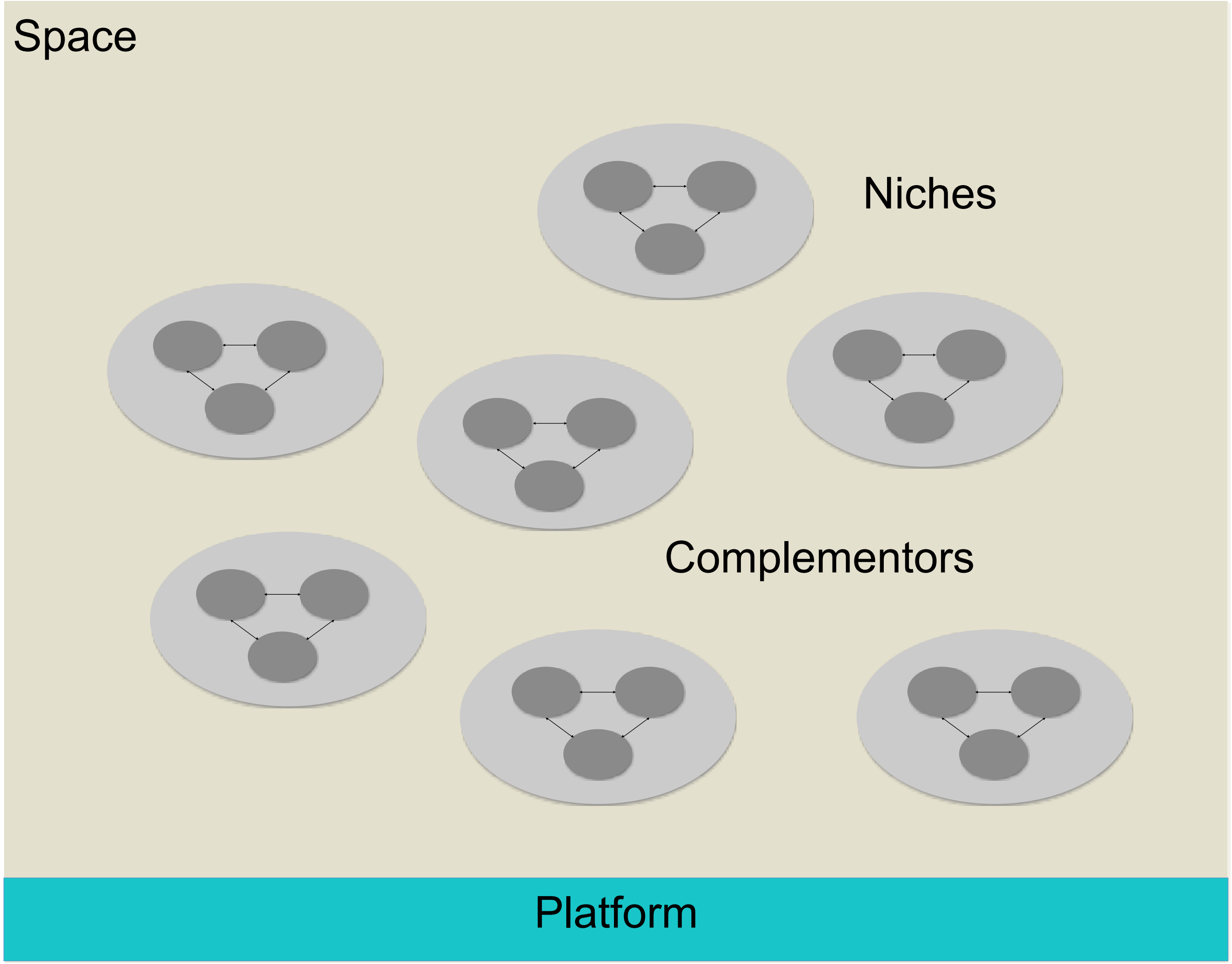
Deutsche Bahn AG

Klaus-Rüdiger Hase

Technik, Systemverbund und Dienstleistungen

Braunschweig, 03.12.2009

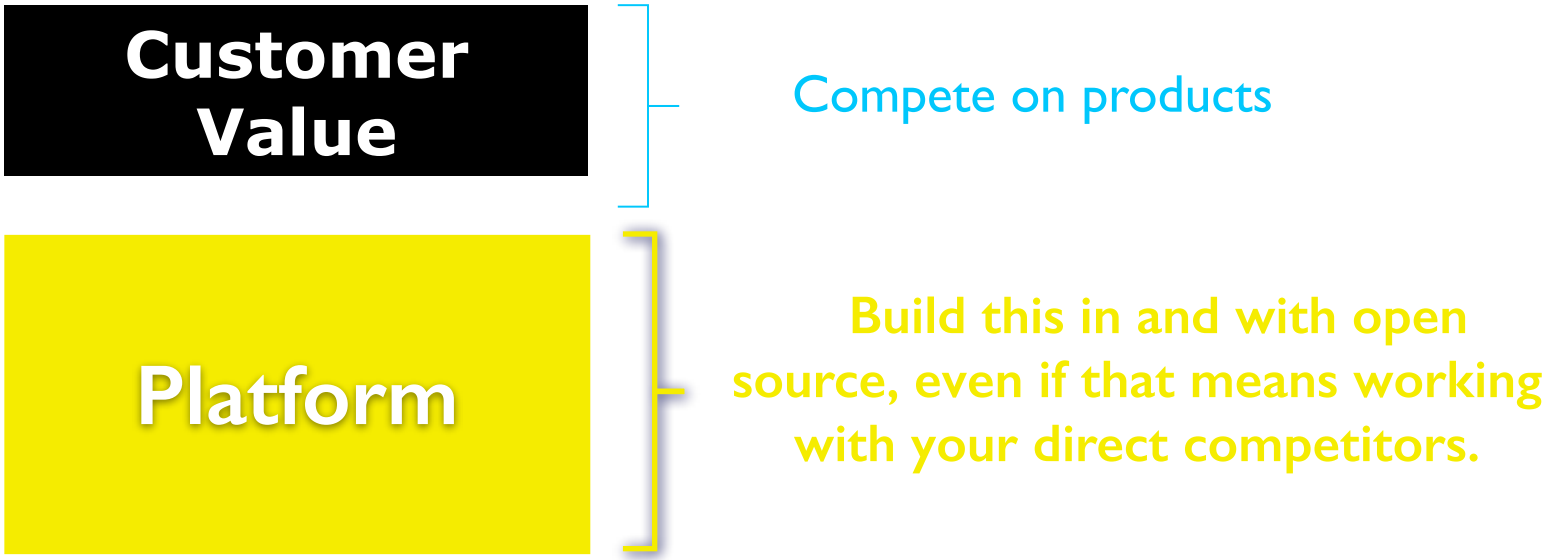
Ecosystems



Some Success Stories

Speaking: Henry Chesbrough

- Innovate using an open platform
- Play poker, not chess



Customer Value

Compete on products

Platform

Build this in and with open source, even if that means working with your direct competitors.

Why an Open Source Platform?

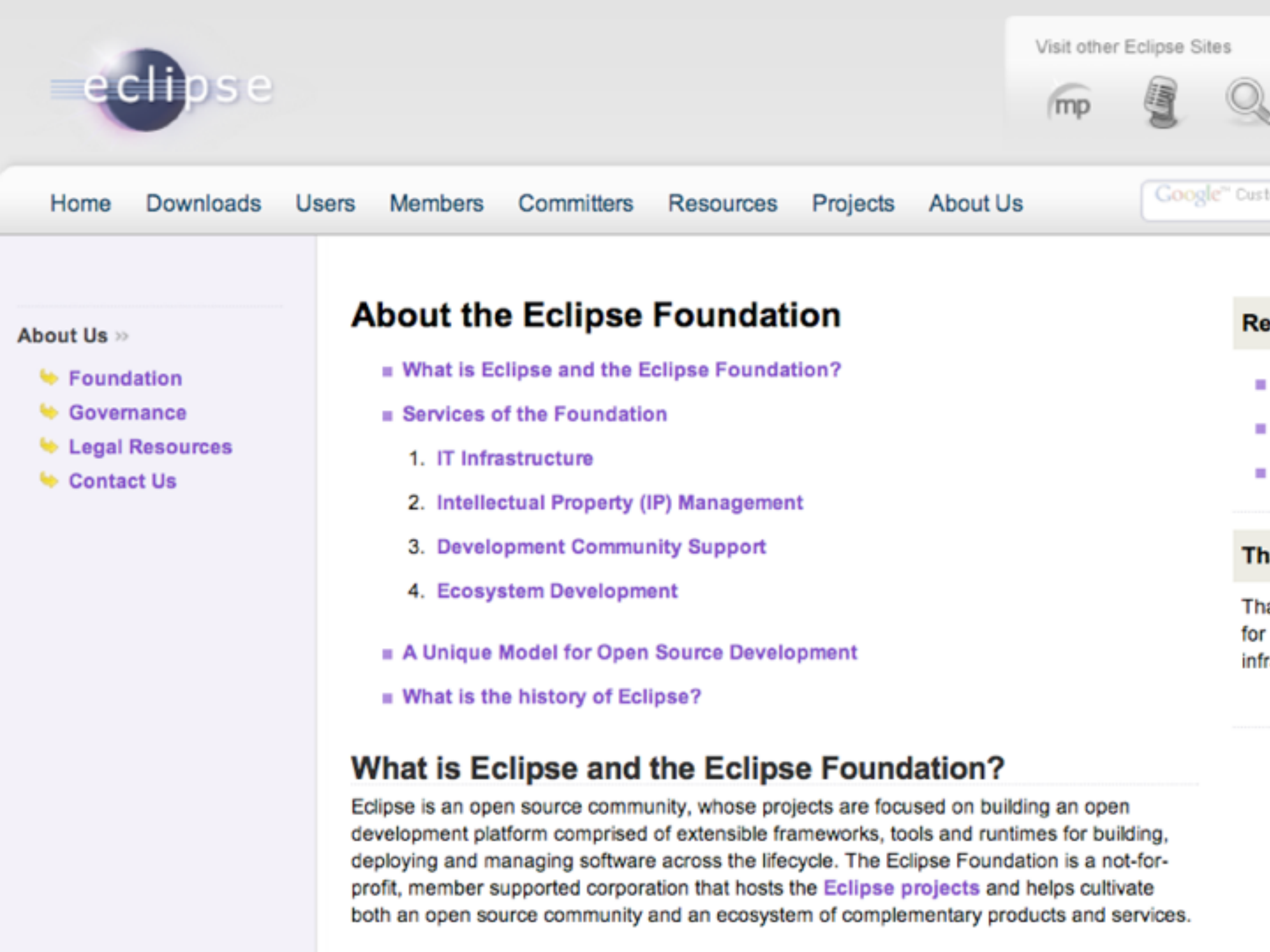
- Open Source development model encourages open innovation
  - Openness, Transparency, Meritocracy
  - Vendor neutrality
- Open Source licensing allows competitors to collaborate on shared platforms
  - No requirement for royalties.
  - No single control point of intellectual property
- Open Source business model encourages rapid adoption of technology
  - It is free and easy to access
- Open Source can allow companies to disrupt the business models of their competitors
- Open Source can allow companies to disrupt supply chain issues

FREE
LUNCH
$10.00
$
SCHWADRON

Open Source Questions

- Is Open Source chaotic?
- How does development *really* work?
- What about the Open Source community?
- How do you manage community contributions?
- How do you plan in Open Source?

- Did Heartbleed demonstrate that OpenSource doesn’t work?



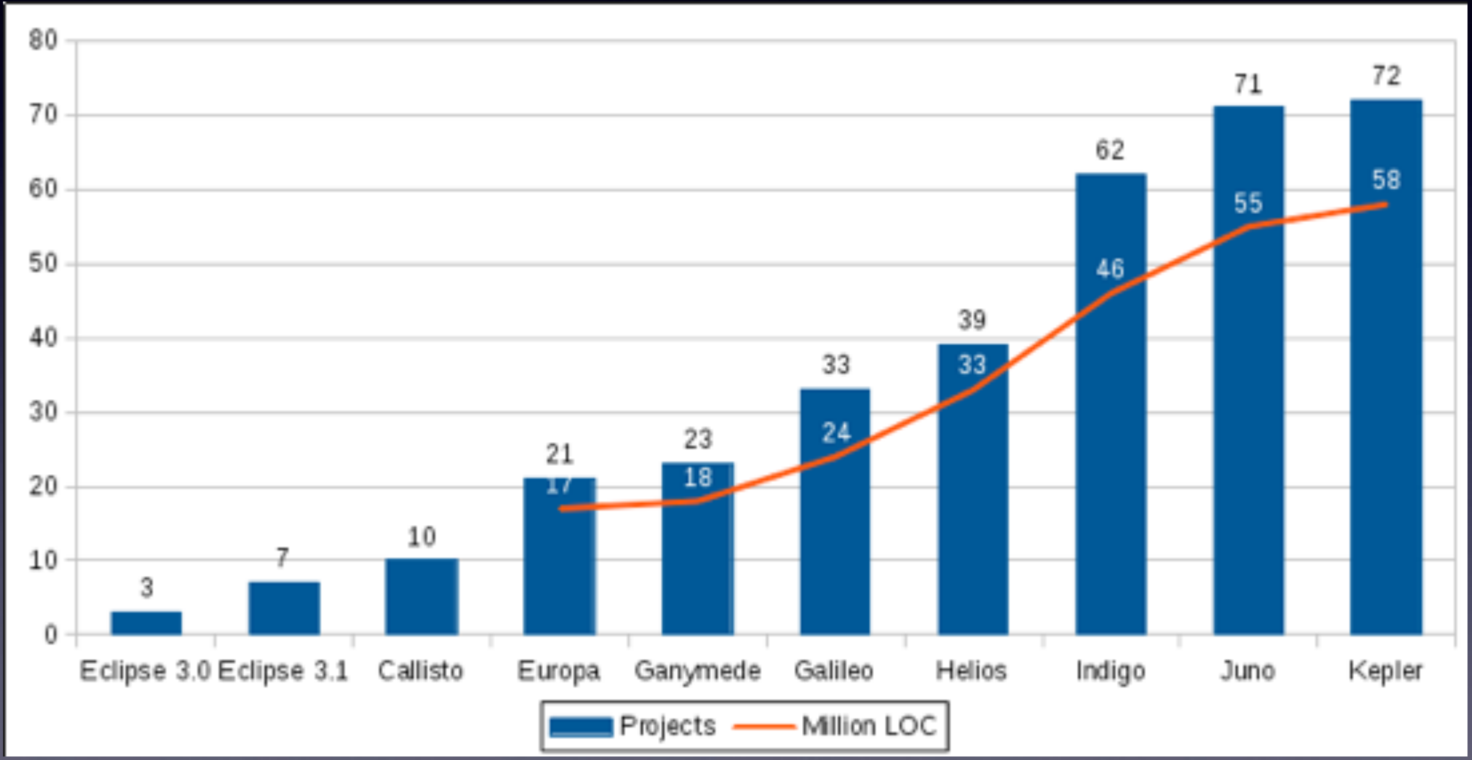
About the Eclipse Foundation

- What is Eclipse and the Eclipse Foundation?
- Services of the Foundation
  1. IT Infrastructure
  2. Intellectual Property (IP) Management
  3. Development Community Support
  4. Ecosystem Development
- A Unique Model for Open Source Development
- What is the history of Eclipse?

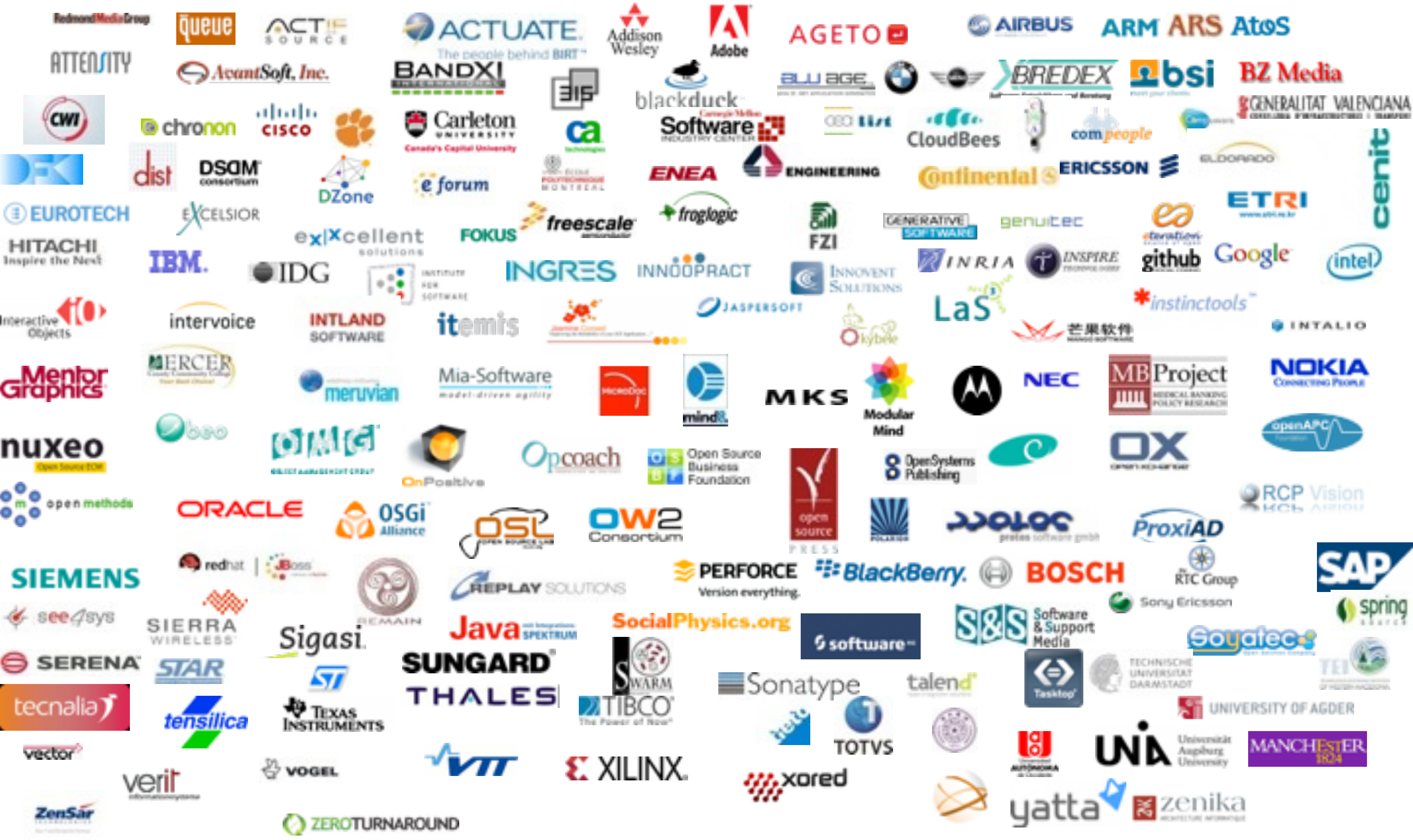
What is Eclipse and the Eclipse Foundation?

Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. The Eclipse Foundation is a not-for-profit, member supported corporation that hosts the Eclipse projects and helps cultivate both an open source community and an ecosystem of complementary products and services.

72 Projects, 58 MLOC



Members of Eclipse



Business Ecosystems are defined as intentional communities of economic actors whose individual business activities share in some large measure the fate of the whole community.

Business Ecosystems and the View from the Firm
James F. Moore, Antitrust Bulletin, Fall 2005

Meritocracy

Transparency

Andrew Magill – flickr.com

Openness

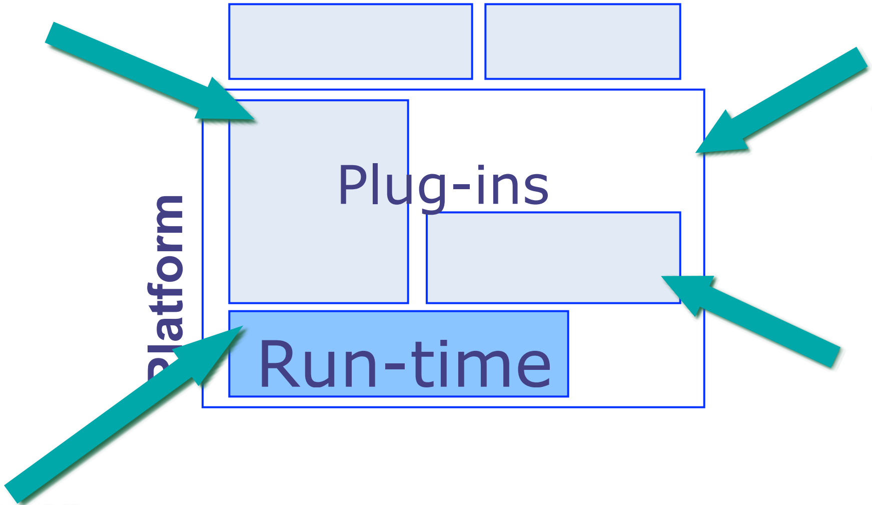
Chris J. Fry – flickr.com

Key Success Factors

- Architecture
- Governance
- Process

Platform Modularity: The Eclipse Experience

Ease of Integration and Extensibility Spurs Innovation



New Plug-ins are First Class Citizens – same footing for everyone

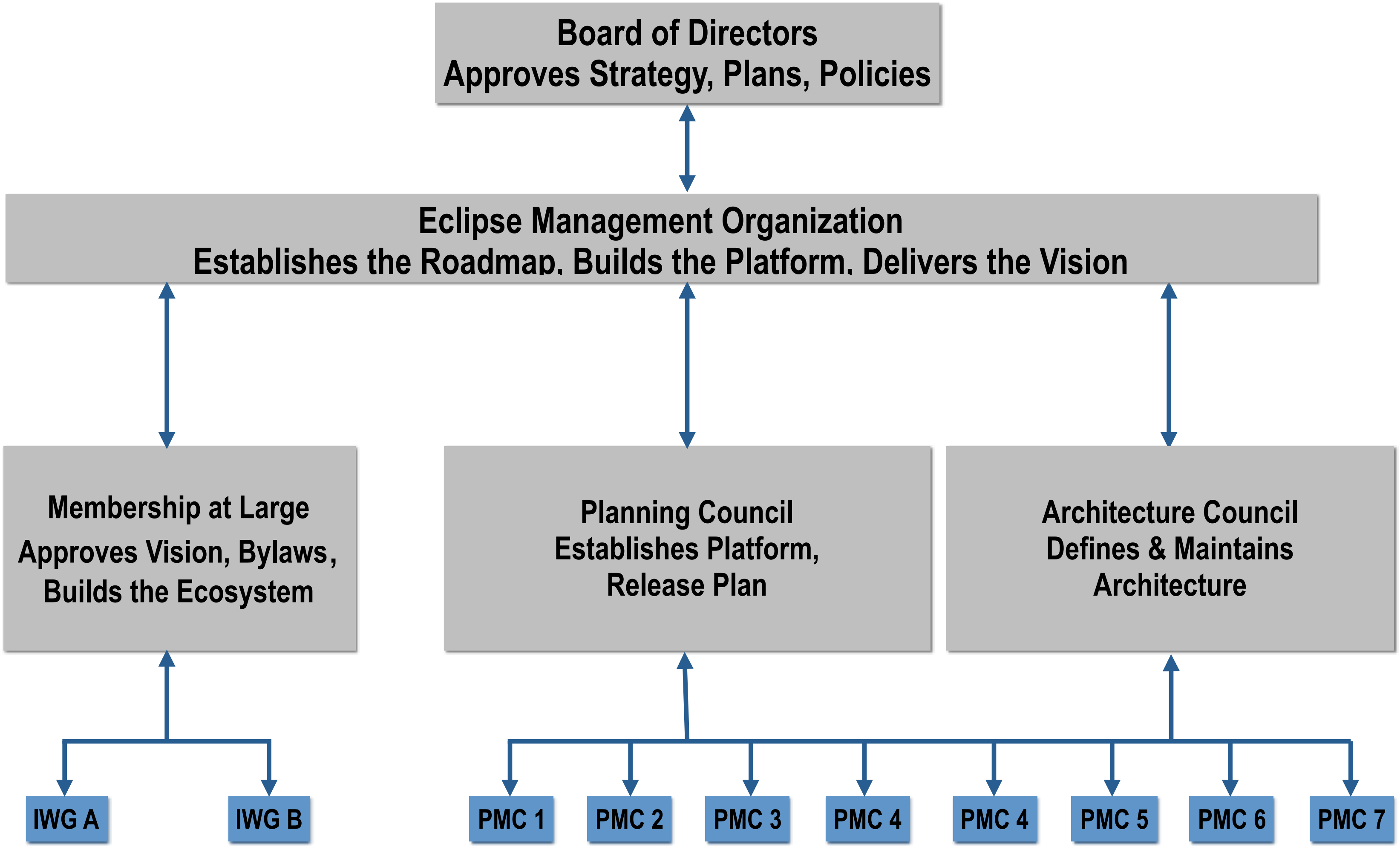
Open API and commercially friendly licensing – Low barriers to Entry

Competition can take place on implementations – users decide winners

Successful Ecosystems are built on this model!

Governance ≠ Management

Eclipse Governance Structure



Board of Directors
Approves Strategy, Plans, Policies

Eclipse Management Organization
Establishes the Roadmap, Builds the Platform, Delivers the Vision

Membership at Large
Approves Vision, Bylaws,
Builds the Ecosystem

Planning Council
Establishes Platform,
Release Plan

Architecture Council
Defines & Maintains
Architecture

IWG A

IWG B

PMC 1

PMC 2

PMC 3

PMC 4

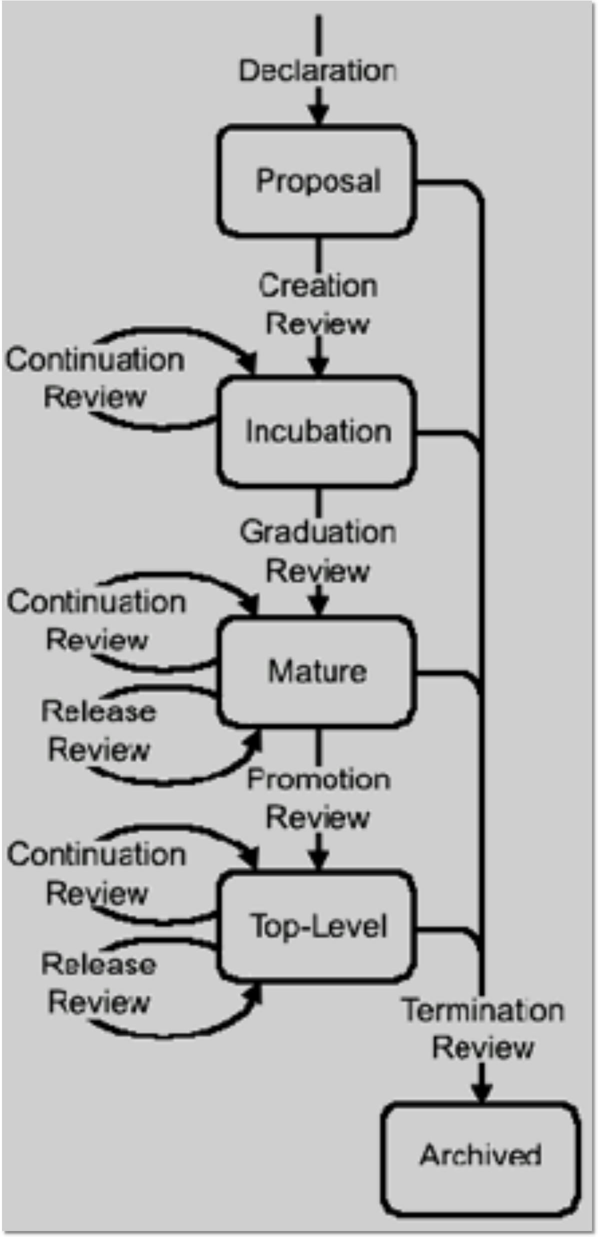
PMC 4

PMC 5

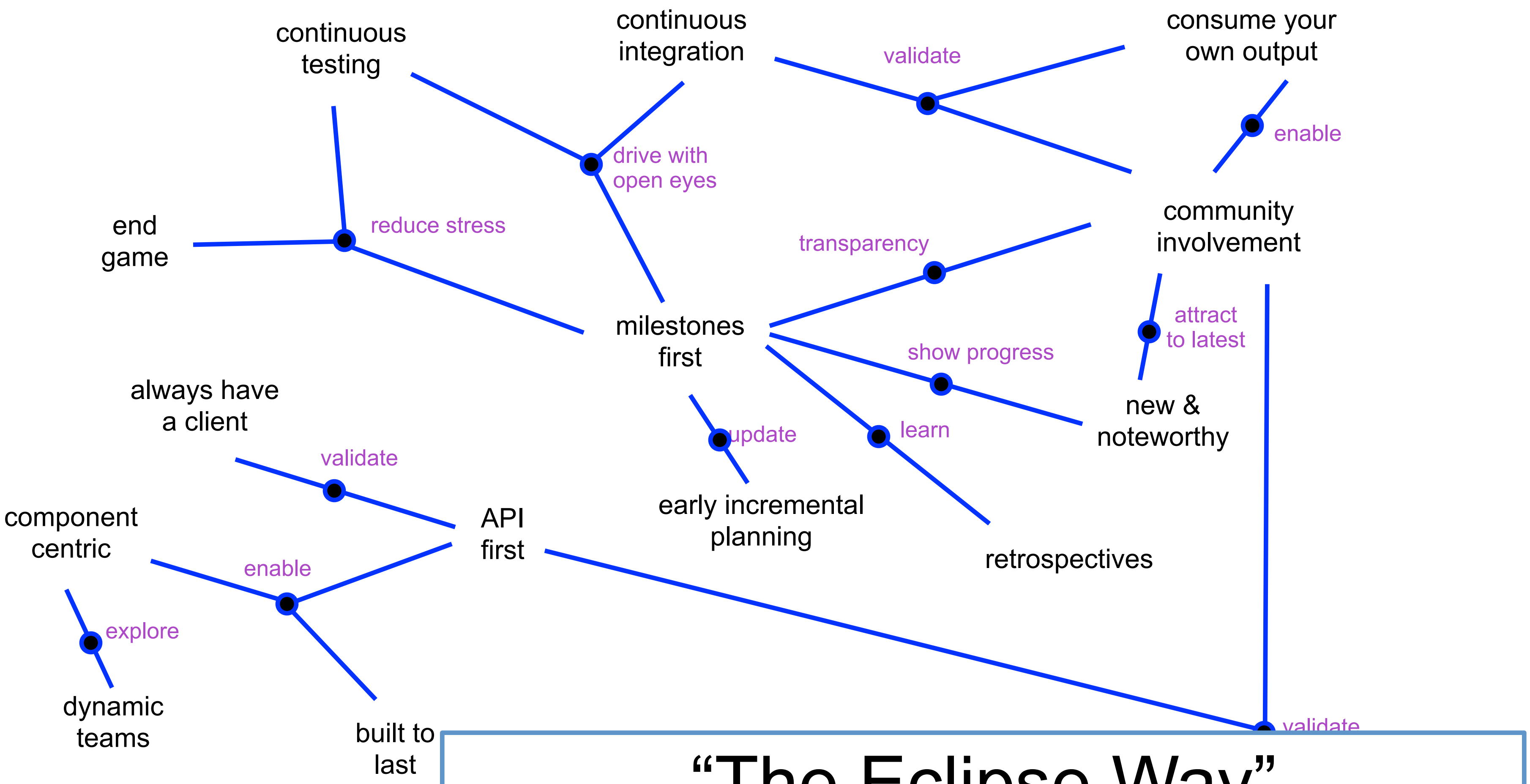
PMC 6

PMC 7

Governance:
The Project Lifecycle



How is the Development Done?



continuous testing

continuous integration

validate

consume your own output

drive with open eyes

enable

reduce stress

end game

community involvement

transparency

milestones first

attract to latest

show progress

always have a client

update

learn

new & noteworthy

validate

early incremental planning

API first

component centric

retrospectives

enable

explore

dynamic teams

built to last

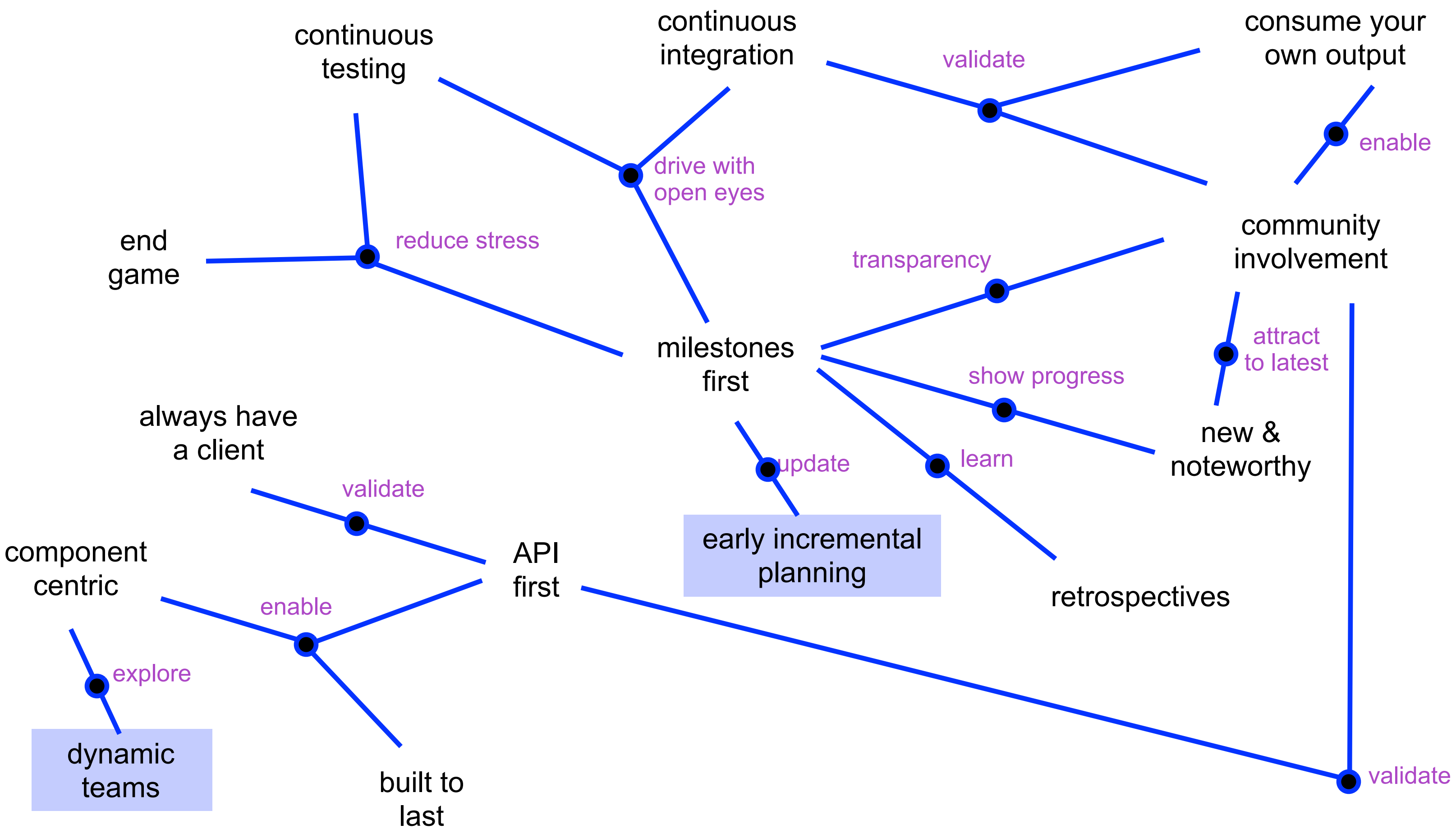
validate

“The Eclipse Way”
Erich Gamma and John Wiegand

Open Source Rules

- OS projects are highly structured
  - explicit rules (more than in most closed source projects)
  - Who may change the source code?
  - Who is responsible for delivering?
  - Who decides about the architecture?
  - …
- Commit rights: public "meritocracy"
  - only a small number of developers can modify the source code: committers
  - key architecture defined by a small team of lead developers
  - peer pressure among committers – continuous reviewing
  - continuous review and feedback by the community
  - contributions from outside have to be reviewed by committers

Planning



continuous testing

continuous integration

consume your own output

validate

enable

drive with open eyes

reduce stress

end game

community involvement

transparency

milestones first

attract to latest

show progress

new & noteworthy

always have a client

update

learn

validate

early incremental planning

API first

component centric

enable

retrospectives

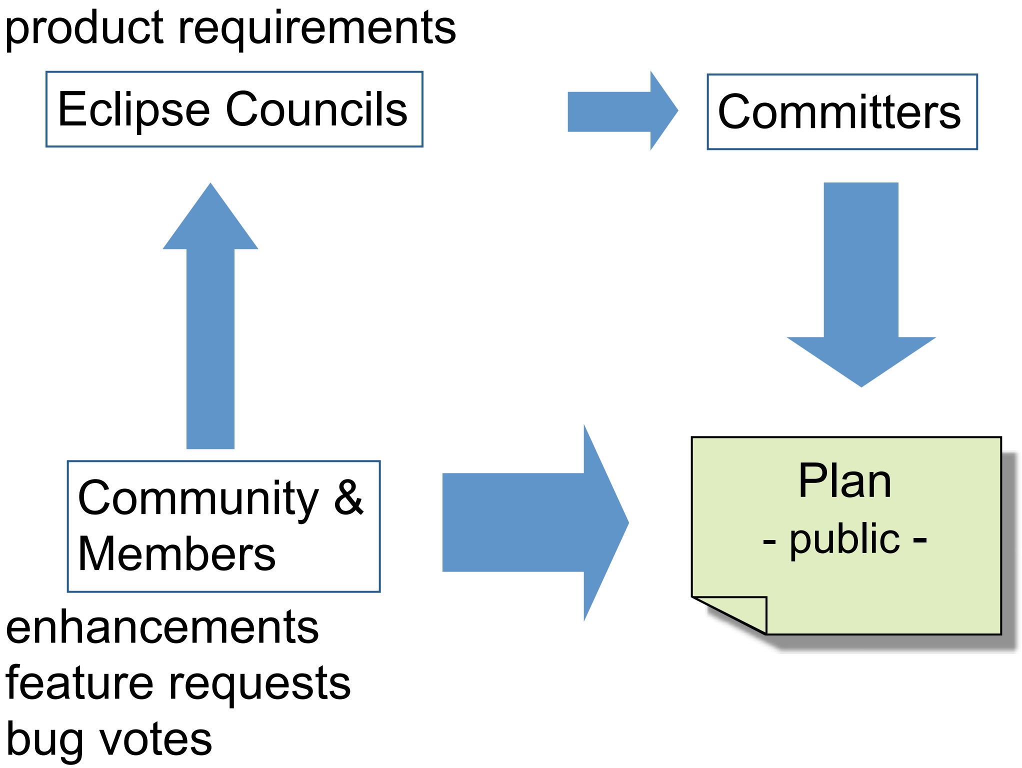
explore

dynamic teams

built to last

validate

Forces of Influence



product requirements

Eclipse Councils

Committers

Community & Members

enhancements
feature requests
bug votes

Plan
- public -

suggest improvements
commit to plan

Planning Council
posts draft plan

- plans start in embryonic form and are revised throughout the release cycle
- milestones/time boxes are fixed early on

Planning

- Release themes establish big picture
  - Community input
  - Planning council new source of input
- Component teams define component plans
- PMC collates initial project plan draft
  - Tradeoff: requirements vs. available resources
  - committed, proposed, deferred
- Plan initially spells out
  - themes
  - milestones
  - compatibility (contract, binary, source, workspace)
- Plan is alive

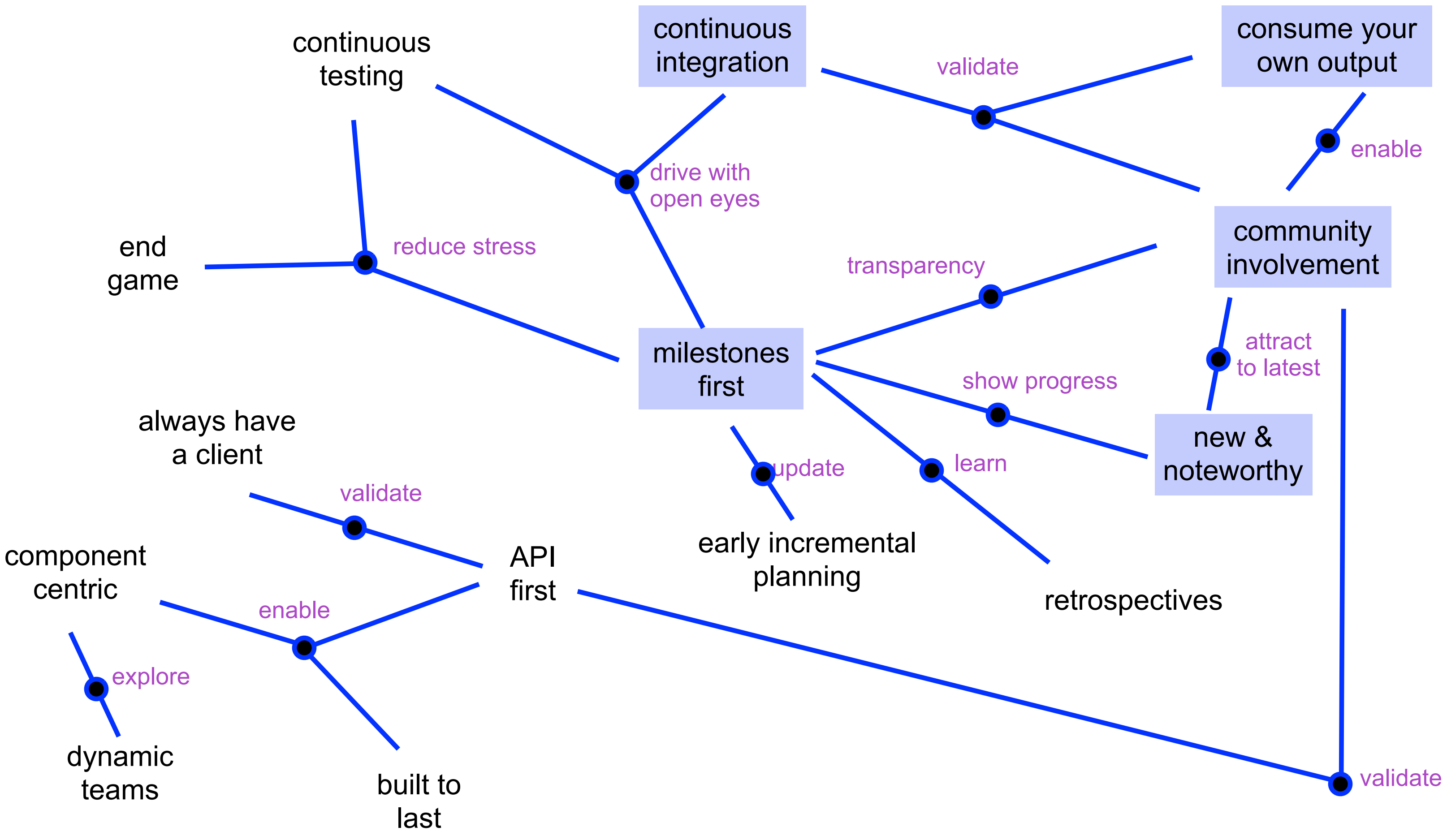
Ongoing Risk Assessment

- Address high risk items and items with many dependencies early
- Maintain schedule by dropping items (if necessary)
  - we will drop proposed items
  - we hate to drop committed items
  - prefer fewer completed items than more items in progress
- High risk items are sandboxed to reduce risk to other items
  - prefer to serialize highest risk items (to minimize integration pain)

Collective Ownership

- Planning team meets at least once a week
  - status
  - planning
  - identification of cross-component issues
  - meeting notes posted to the developer mailing lists
- Dynamic teams are established for solving cross-component issues
  - one cross-component issue per dynamic team
  - members are key developers from all effected components
  - find, implement, and roll-out solution of the assigned cross component issue
  - represented in the weekly planning calls

Project Rhythm

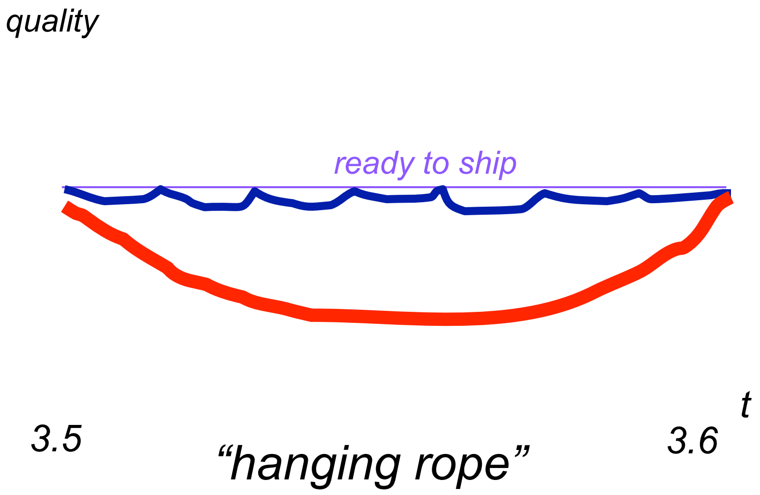


Milestones

- break down release cycle into milestones
  - We use 6 weeks
- milestones are a miniature development cycle
  - Plan, execute, test, retrospective
- milestone builds are good enough to be used by the community

➢ **milestones reduce stress, keep quality high**

- before/after



"hanging rope"

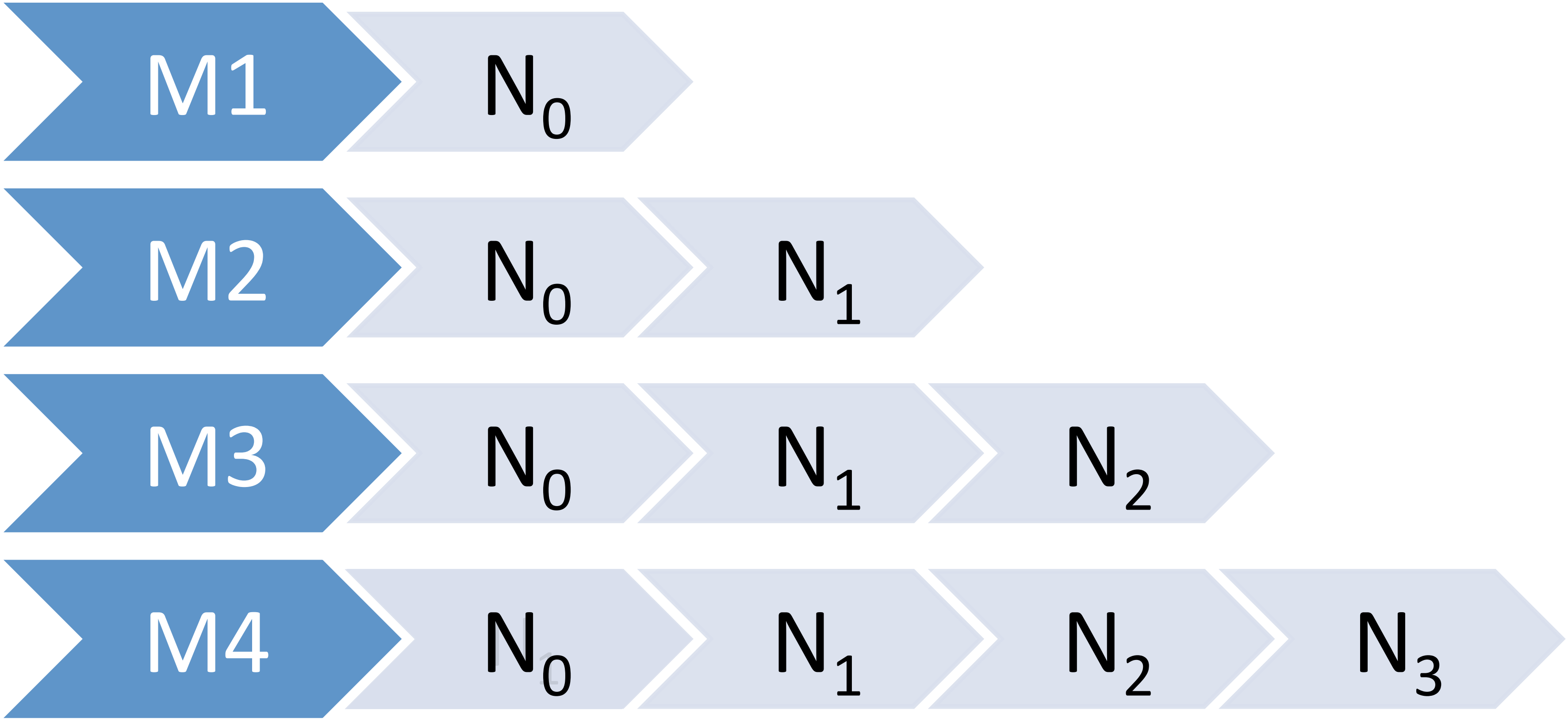
Continuous Integration

- Fully automated build process
- Build quality verified by automatic unit tests
- Staged builds
  - nightly builds (some projects even more frequently)
    - discover integration problems between components
  - weekly integration builds
    - all automatic unit tests must be successful
    - good enough for our own use
  - milestone builds
    - good enough for the community to use

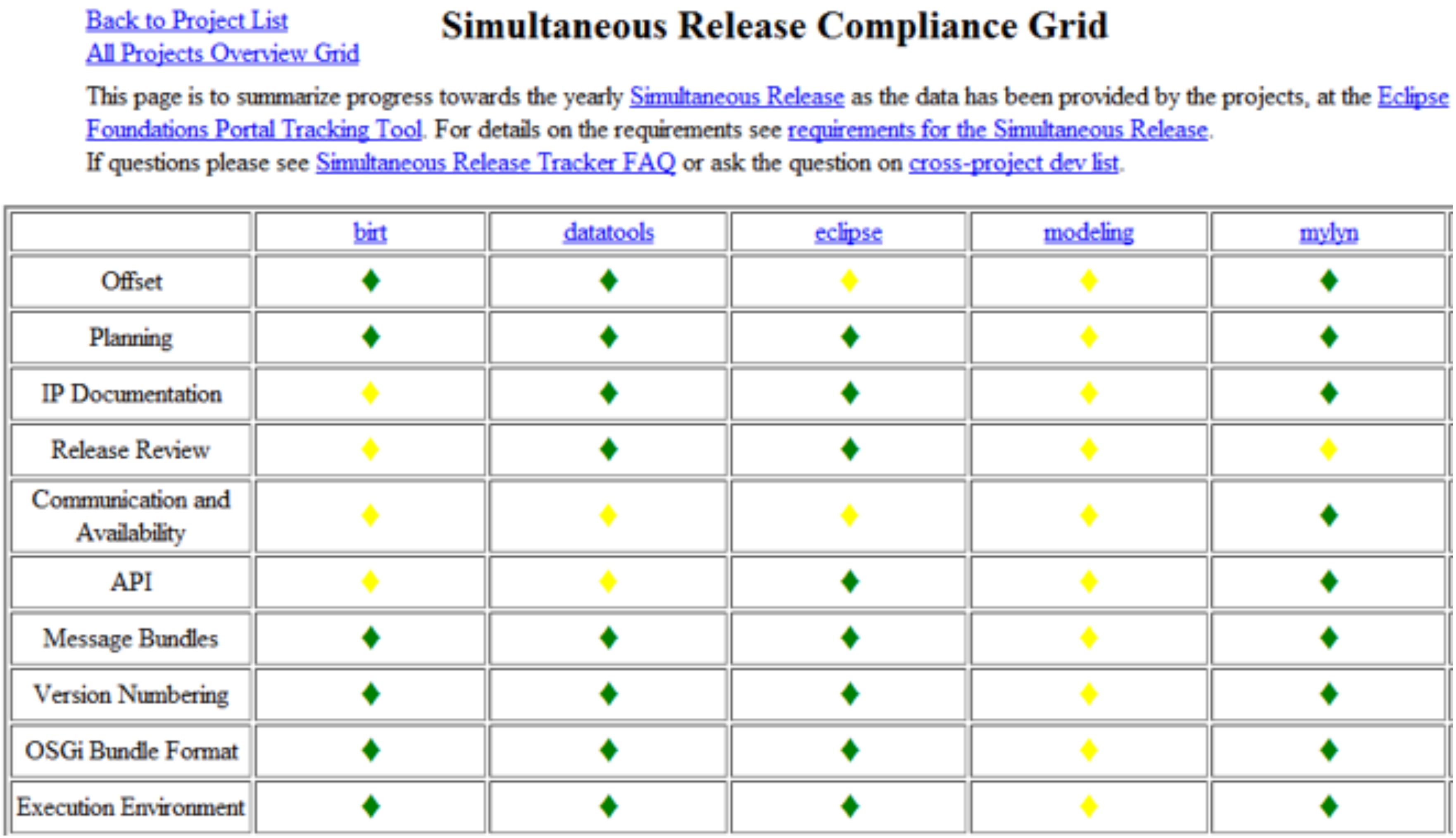
Practice Makes Perfect

- 7 milestones, 4 release candidates
  - 11 chances to practice releasing
- Projects denoted N_0, N_1, N_2, N_3
  - Build in order of dependencies
  - Early builds takes days, later builds take hours
- Build to shared repository, make everything available to the community for feedback and testing

Getting on the Train



Constant Public Status Reporting



Back to Project List
All Projects Overview Grid

Simultaneous Release Compliance Grid

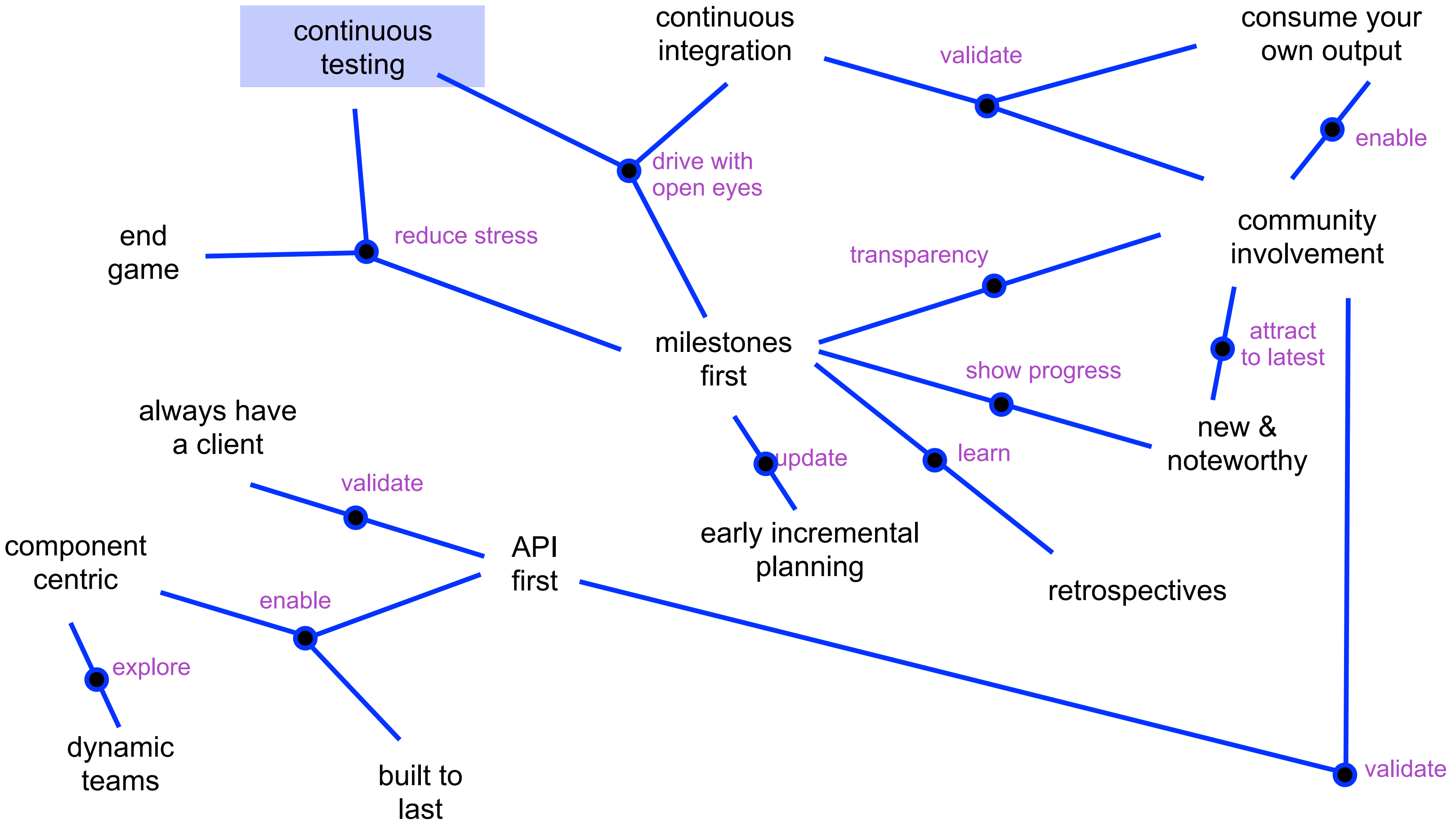
This page is to summarize progress towards the yearly Simultaneous Release as the data has been provided by the projects, at the Eclipse Foundations Portal Tracking Tool. For details on the requirements see requirements for the Simultaneous Release.
If questions please see Simultaneous Release Tracker FAQ or ask the question on cross-project dev list.

	birt	datatools	eclipse	modeling	mylyn
Offset	🟩	🟩	🟨	🟨	🟩
Planning	🟩	🟩	🟩	🟨	🟩
IP Documentation	🟨	🟩	🟩	🟨	🟩
Release Review	🟨	🟩	🟩	🟨	🟨
Communication and Availability	🟨	🟨	🟨	🟨	🟩
API	🟨	🟨	🟩	🟨	🟩
Message Bundles	🟩	🟩	🟩	🟨	🟩
Version Numbering	🟩	🟩	🟩	🟨	🟩
OSGi Bundle Format	🟩	🟩	🟩	🟨	🟩
Execution Environment	🟩	🟩	🟩	🟨	🟩

Community Involvement

- An active community is the major asset of an OSS project
- OSS project gives and takes:
  - OSS developer gives:
    - listen to feedback and react
    - demonstrate continuous progress
    - transparent development
  - OSS developer takes:
    - answer user questions so that developers do not have to do it
    - report defects and feature requests
    - validate technology by writing plug-ins
    - submit patches and enhancements
- Give and take isn't always balanced
  - community isn't shy and is demanding

Testing



continuous testing

continuous integration

validate

consume your own output

enable

drive with open eyes

reduce stress

end game

community involvement

transparency

milestones first

attract to latest

show progress

new & noteworthy

always have a client

update

learn

validate

API first

early incremental planning

component centric

enable

retrospectives

explore

dynamic teams

built to last

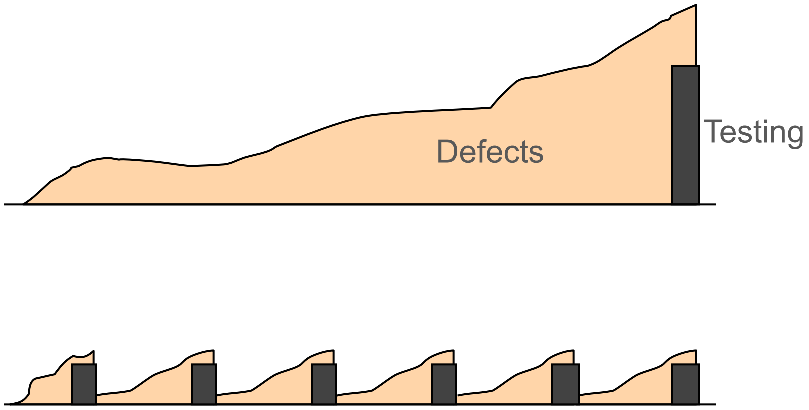
validate

Testing

- Innovate with confidence

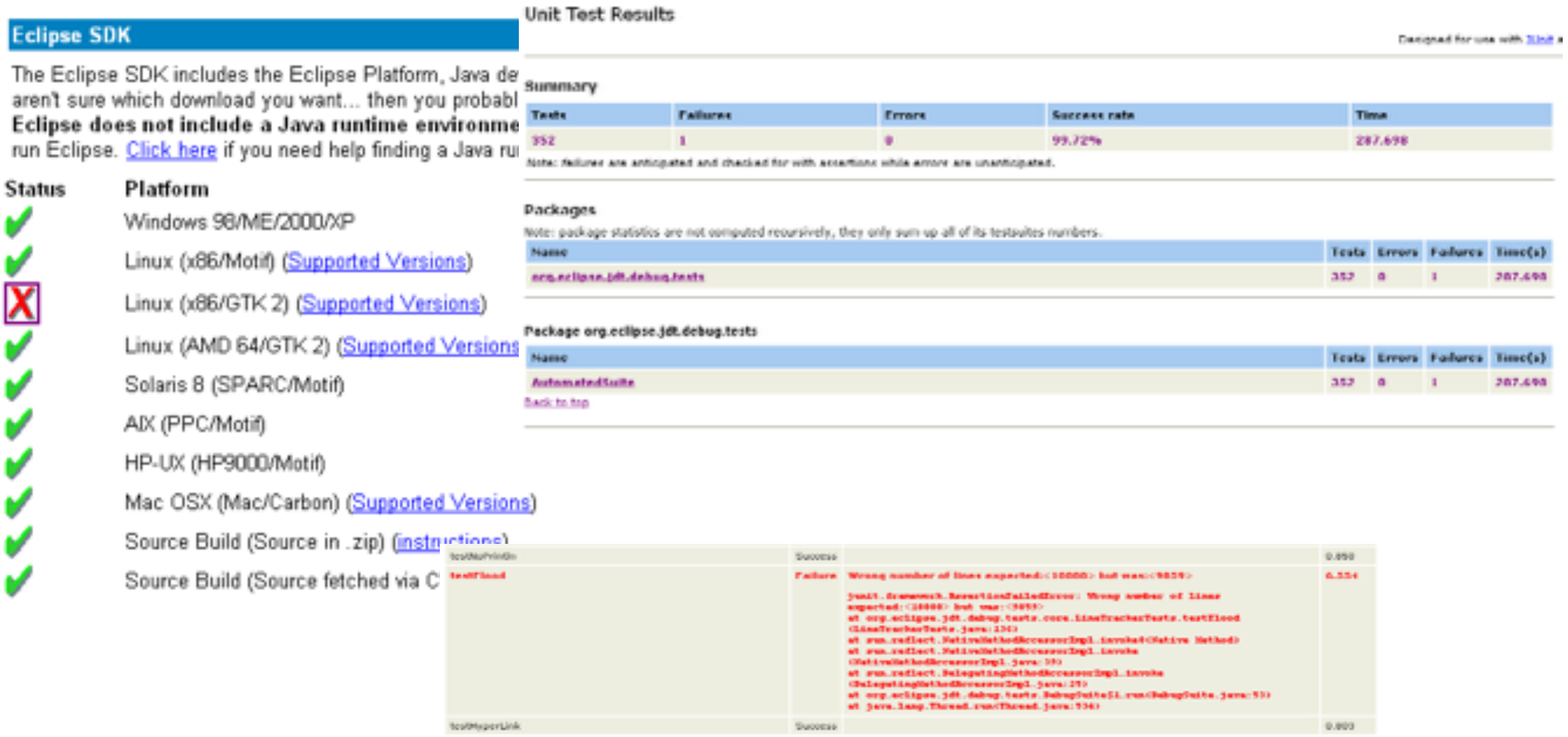
- Tests run after each build

- Test kinds
  - **correctness** tests
    - assert correct behavior
  - **performance** tests
    - assert no performance regressions
      - based on a database of previous test run measurements
  - **resource** tests, leak tests
    - assert no resource consumption regressions



Kent Beck – JUnit handbook

Unit Test Report



Eclipse SDK

The Eclipse SDK includes the Eclipse Platform, Java de
aren't sure which download you want... then you probabl
Eclipse does not include a Java runtime environme
run Eclipse. Click here if you need help finding a Java ru

Status	Platform
✓	Windows 98/ME/2000/XP
✓	Linux (x86/Motif) (Supported Versions)
☒	Linux (x86/GTK 2) (Supported Versions)
✓	Linux (AMD 64/GTK 2) (Supported Versions
✓	Solaris 8 (SPARC/Motif)
✓	AIX (PPC/Motif)
✓	HP-UX (HP9000/Motif)
✓	Mac OSX (Mac/Carbon) (Supported Versions)
✓	Source Build (Source in .zip) (instructions)
✓	Source Build (Source fetched via C

Unit Test Results

Designed for use with JUnit a

Summary

Tests	Failures	Errors	Success rate	Time
352	1	0	99.72%	287.698

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Packages

Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers.

Name	Tests	Errors	Failures	Time(s)
org.eclipse.jdt.debug.tests	352	0	1	287.698

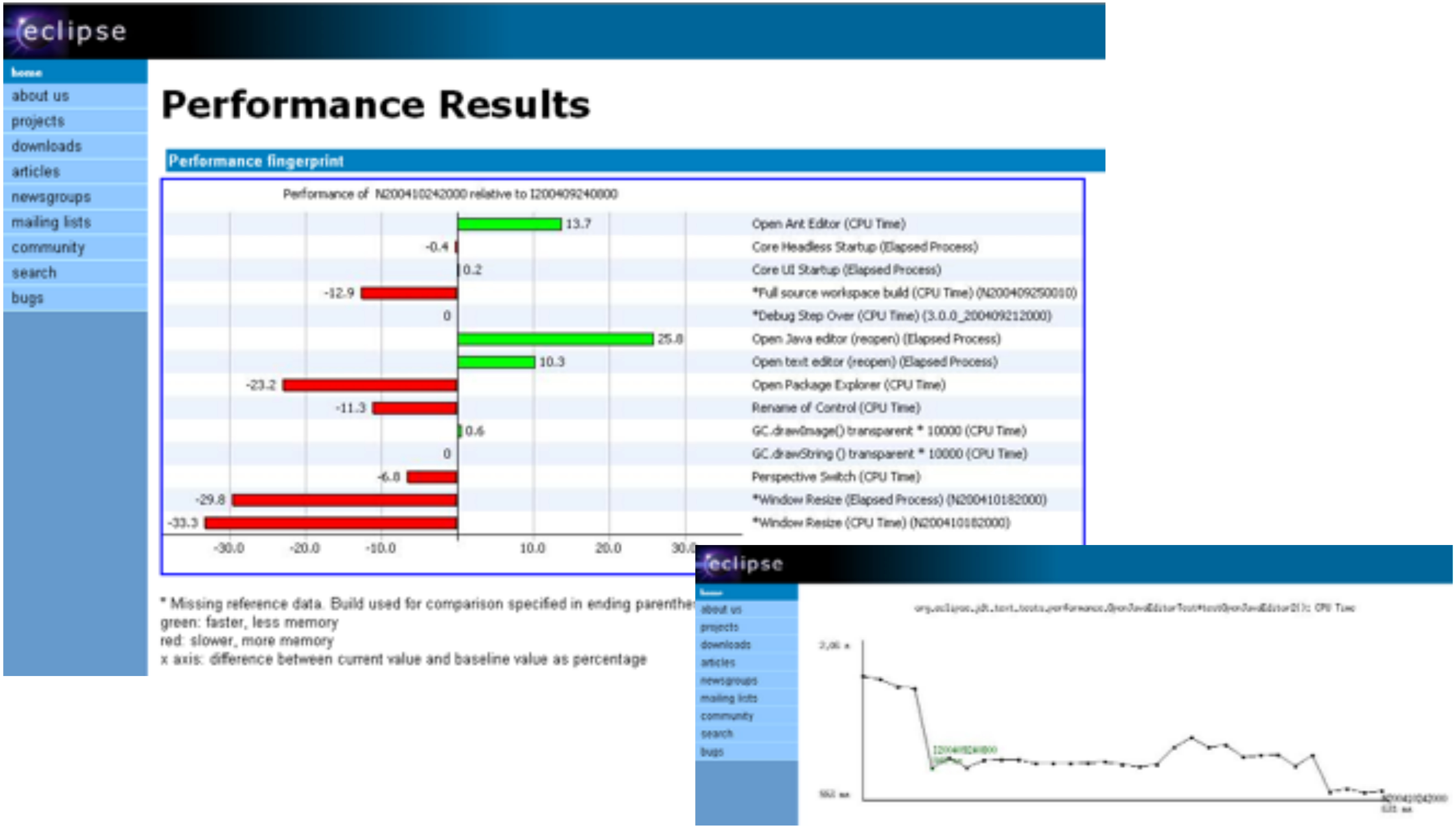
Package org.eclipse.jdt.debug.tests

Name	Tests	Errors	Failures	Time(s)
AutomatedSuite	352	0	1	287.698

Back to top

testNoPrintln	Success		0.050
testFlood	Failure	Wrong number of lines expected:<10000> but was:<9859> junit.framework.AssertionFailedError: Wrong number of lines expected:<10000> but was:<9859> at org.eclipse.jdt.debug.tests.core.LineTrackerTests.testFlood (LineTrackerTests.java:134) at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke (NativeMethodAccessorImpl.java:39) at sun.reflect.DelegatingMethodAccessorImpl.invoke (DelegatingMethodAccessorImpl.java:25) at org.eclipse.jdt.debug.tests.DebugSuite$1.run(DebugSuite.java:53) at java.lang.Thread.run(Thread.java:534)	6.334
testHyperLink	Success		0.003

Performance Test Report


eclipse
home
about us
projects
downloads
articles
newsgroups
mailing lists
community
search
bugs

Performance Results

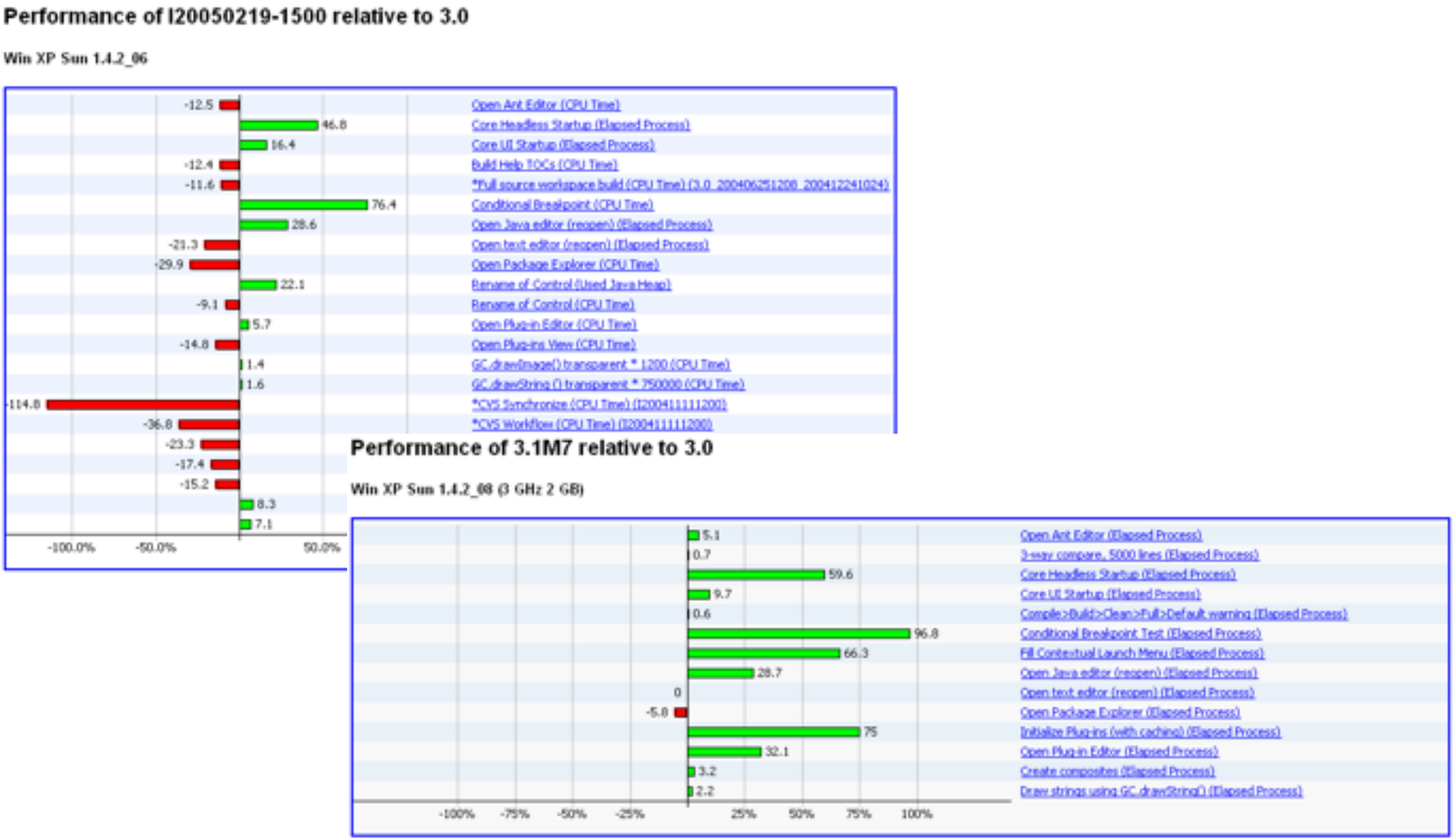
Performance fingerprint

Performance of N200410242000 relative to I200409240800
13.7
-0.4
0.2
-12.9
0
25.8
10.3
-23.2
-11.3
0.6
0
-6.8
-29.8
-33.3
Open Ant Editor (CPU Time)
Core Headless Startup (Elapsed Process)
Core UI Startup (Elapsed Process)
*Full source workspace build (CPU Time) (N200409250010)
*Debug Step Over (CPU Time) (3.0.0_200409212000)
Open Java editor (reopen) (Elapsed Process)
Open text editor (reopen) (Elapsed Process)
Open Package Explorer (CPU Time)
Rename of Control (CPU Time)
GC.drawImage() transparent * 10000 (CPU Time)
GC.drawString () transparent * 10000 (CPU Time)
Perspective Switch (CPU Time)
*Window Resize (Elapsed Process) (N200410182000)
*Window Resize (CPU Time) (N200410182000)
-30.0
-20.0
-10.0
10.0
20.0
30.0

* Missing reference data. Build used for comparison specified in ending parenthe
green: faster, less memory
red: slower, more memory
x axis: difference between current value and baseline value as percentage

eclipse
home
about us
projects
downloads
articles
newsgroups
mailing lists
community
search
bugs
org.eclipse.jdt.text.tests.performance.OpenJavaEditorTest#testOpenJavaEditor2(): CPU Time
2,06 s
553 ms
I200409240800
N200410242000
531 ms

Before (M5) – After (M7)



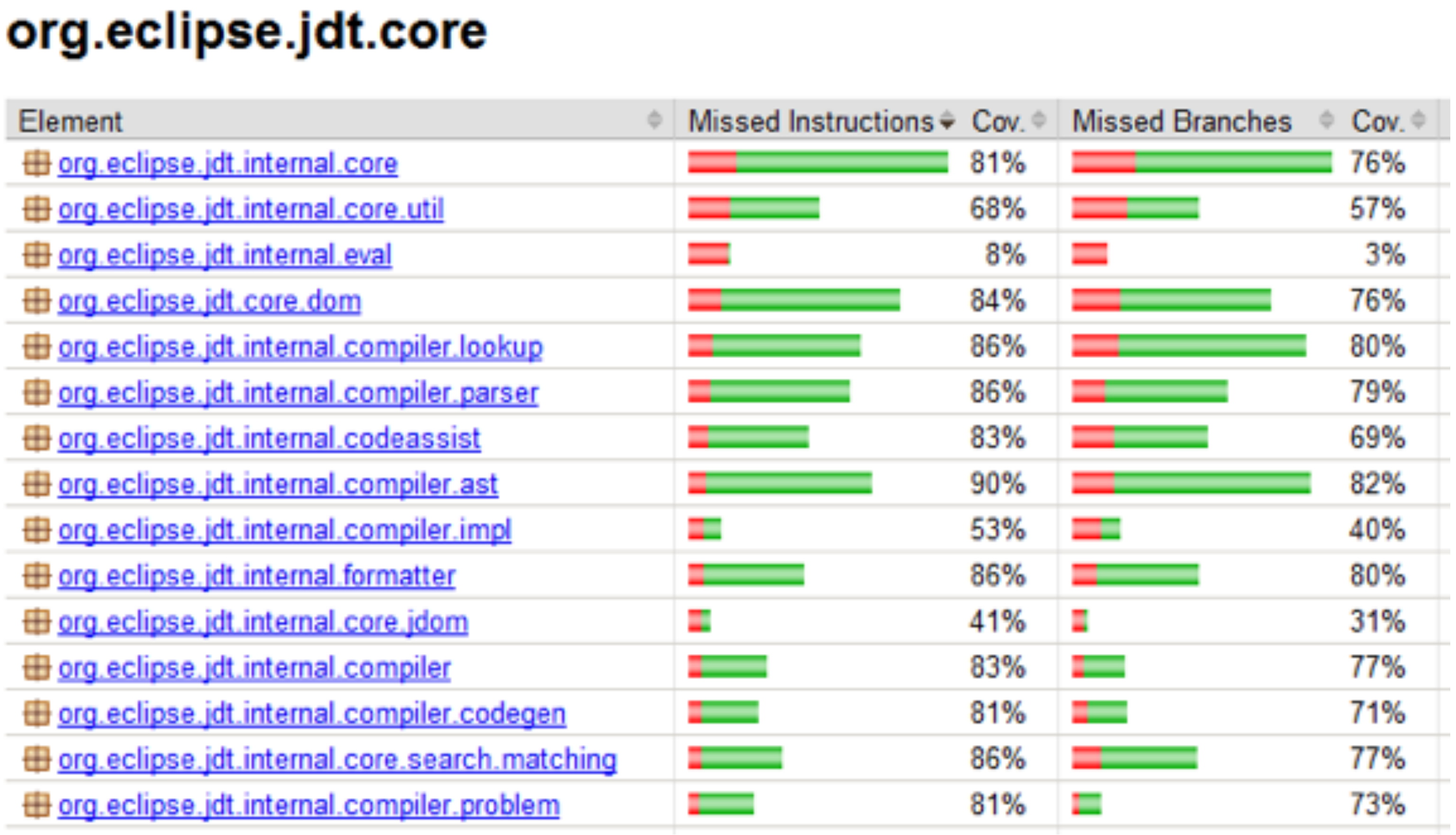
Performance of I20050219-1500 relative to 3.0

Win XP Sun 1.4.2_06

Performance of 3.1M7 relative to 3.0

Win XP Sun 1.4.2_08 (3 GHz 2 GB)

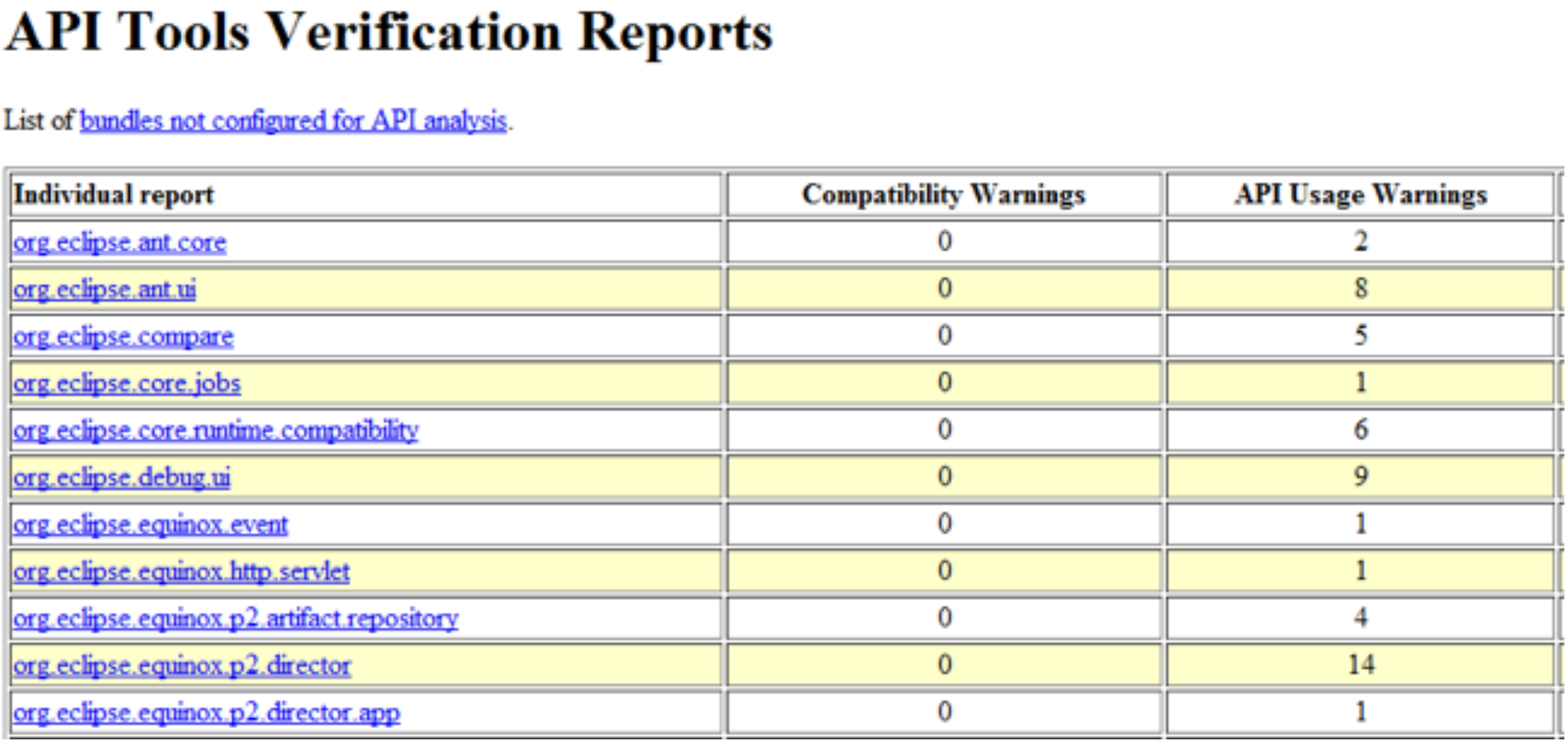
Code Coverage



org.eclipse.jdt.core

Element	Missed Instructions	Cov.	Missed Branches	Cov.
org.eclipse.jdt.internal.core		81%		76%
org.eclipse.jdt.internal.core.util		68%		57%
org.eclipse.jdt.internal.eval		8%		3%
org.eclipse.jdt.core.dom		84%		76%
org.eclipse.jdt.internal.compiler.lookup		86%		80%
org.eclipse.jdt.internal.compiler.parser		86%		79%
org.eclipse.jdt.internal.codeassist		83%		69%
org.eclipse.jdt.internal.compiler.ast		90%		82%
org.eclipse.jdt.internal.compiler.impl		53%		40%
org.eclipse.jdt.internal.formatter		86%		80%
org.eclipse.jdt.internal.core.jdom		41%		31%
org.eclipse.jdt.internal.compiler		83%		77%
org.eclipse.jdt.internal.compiler.codegen		81%		71%
org.eclipse.jdt.internal.core.search.matching		86%		77%
org.eclipse.jdt.internal.compiler.problem		81%		73%

API Conformance Testing

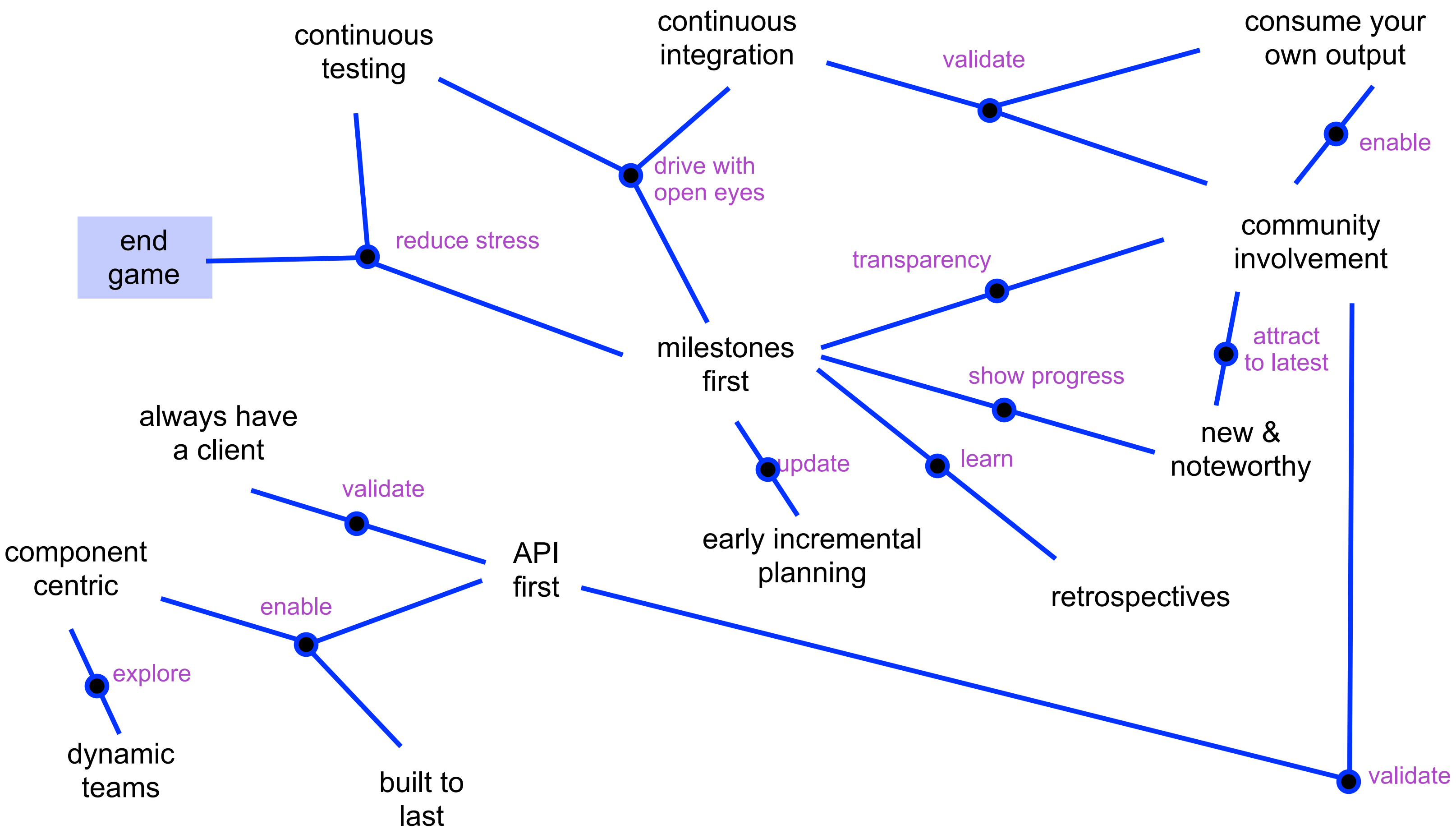


API Tools Verification Reports

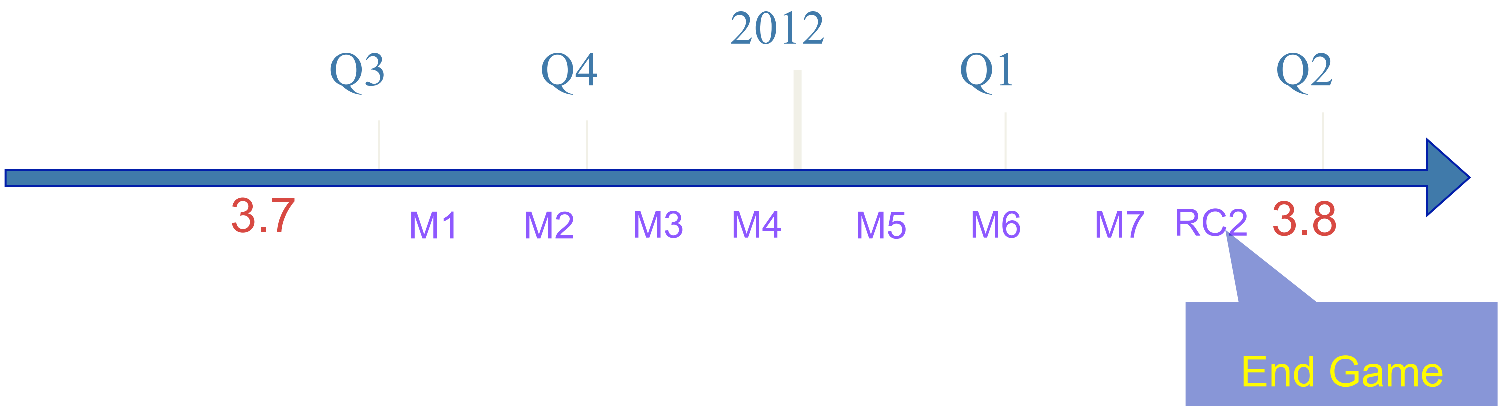
List of bundles not configured for API analysis.

Individual report	Compatibility Warnings	API Usage Warnings
org.eclipse.ant.core	0	2
org.eclipse.ant.ui	0	8
org.eclipse.compare	0	5
org.eclipse.core.jobs	0	1
org.eclipse.core.runtime.compatibility	0	6
org.eclipse.debug.ui	0	9
org.eclipse.equinox.event	0	1
org.eclipse.equinox.http.servlet	0	1
org.eclipse.equinox.p2.artifact.repository	0	4
org.eclipse.equinox.p2.director	0	14
org.eclipse.equinox.p2.director.app	0	1

End Game


continuous testing
continuous integration
validate
consume your own output
enable
drive with open eyes
reduce stress
end game
community involvement
transparency
milestones first
attract to latest
show progress
new & noteworthy
always have a client
update
learn
validate
API first
early incremental planning
retrospectives
component centric
enable
explore
dynamic teams
built to last
validate

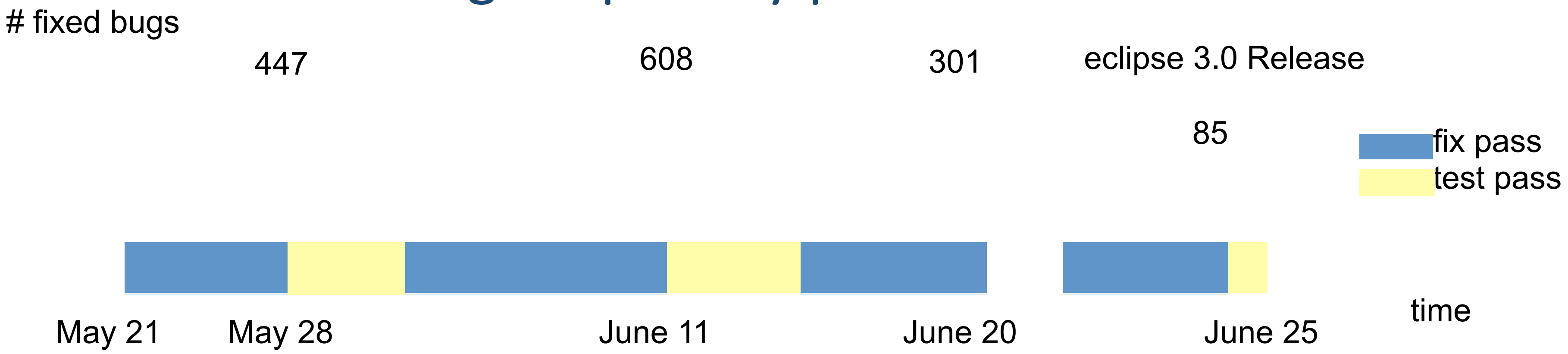
The Annual Schedule



- convergence process applied before release
- sequence of test-fix passes
  - it is a community event!

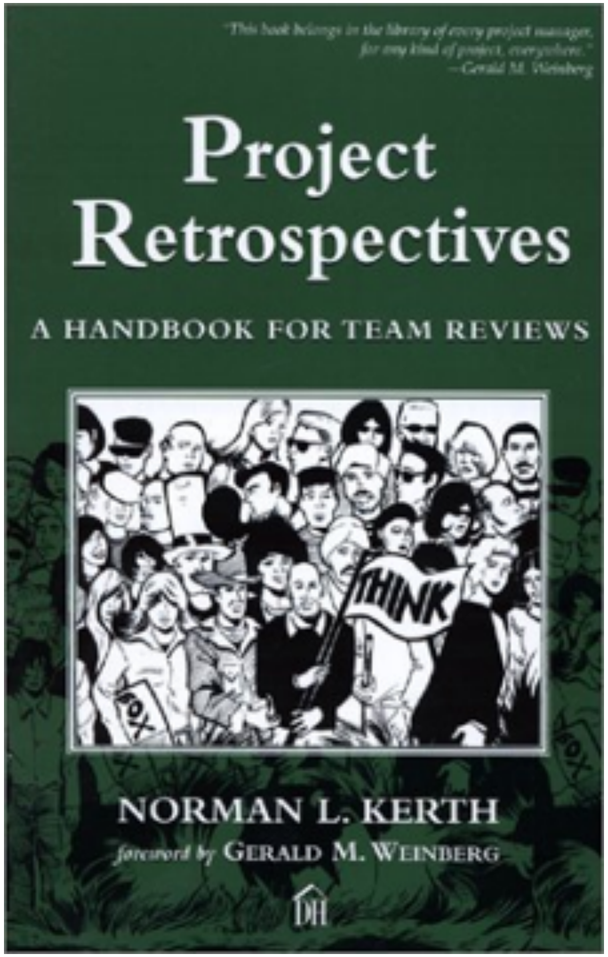
End Game Convergence

- with each pass the costs for fixing are increased
  - higher burden to work on fix for a problem
  - higher burden to release a fix for a problem
  - focus on higher priority problems



Decompression

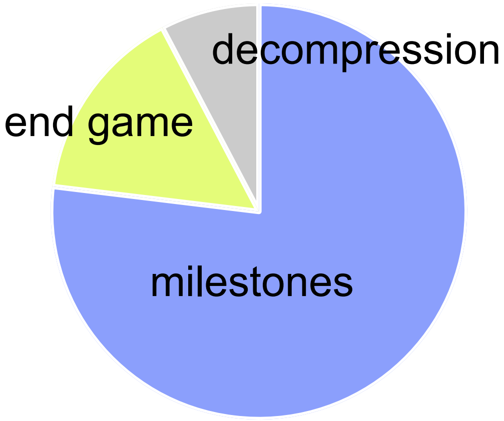
- recover from release
- retrospective of the last cycle
  - learn from the last cycle
    - achievements
    - failures
  - "stay aware, adapt, change"
  - define retrospective actions


"This book belongs in the library of every project manager, for any kind of project, everywhere."
—Gerald M. Weinberg
Project Retrospectives
A HANDBOOK FOR TEAM REVIEWS
THINK
NORMAN L. KERTH
foreword by GERALD M. WEINBERG
DH

- start to plan the next release and cycle

Where the Time Goes

- release cycle 12 months
  - milestones – 9 months
  - endgame – 2 months
  - decompression – 1 month



Conclusions

- Open source uses highly rigorous and disciplined processes
- Chose your platform carefully
- Adopt these principles:
  - Meritocracy
  - Openness
  - Transparency

Lessons learned

- Open Source is like any other software
- It needs proper funding
- Quality needs to be tested, it is not a given
- The advantages of the approach still weigh out the disadvantages

A8 connect
A8L 3.0 T quattro

DB
ICE
63
6

Thank You!

Questions?

ralph.mueller@eclipse.org

the talk is based on materials by:
Erich Gamma, John Wiegand
Mike Milinkovich