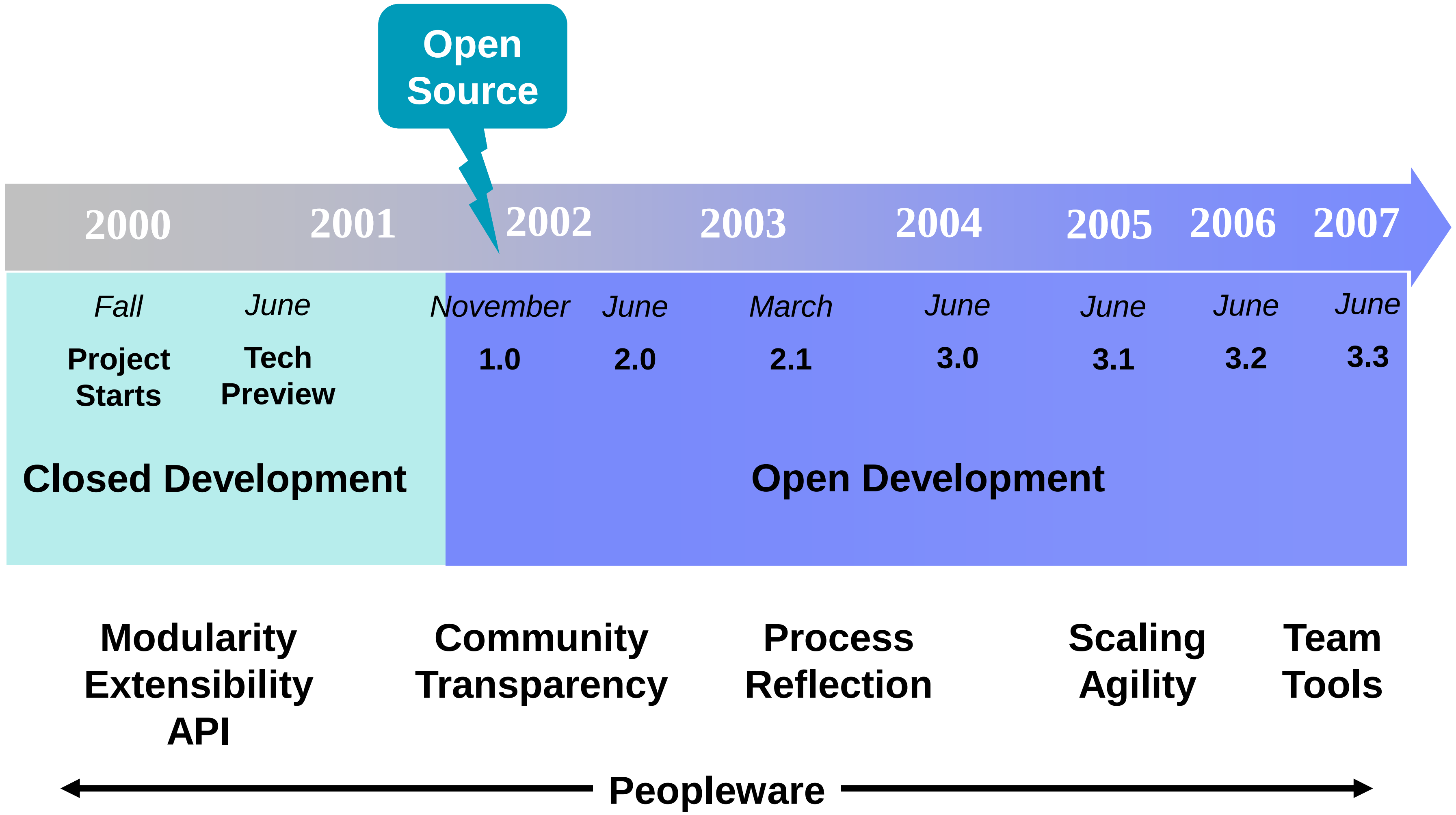
How (7 years of) Eclipse Changed my Views on Software Development

Erich Gamma
IBM Distinguished Engineer
IBM Rational Zurich Research Lab

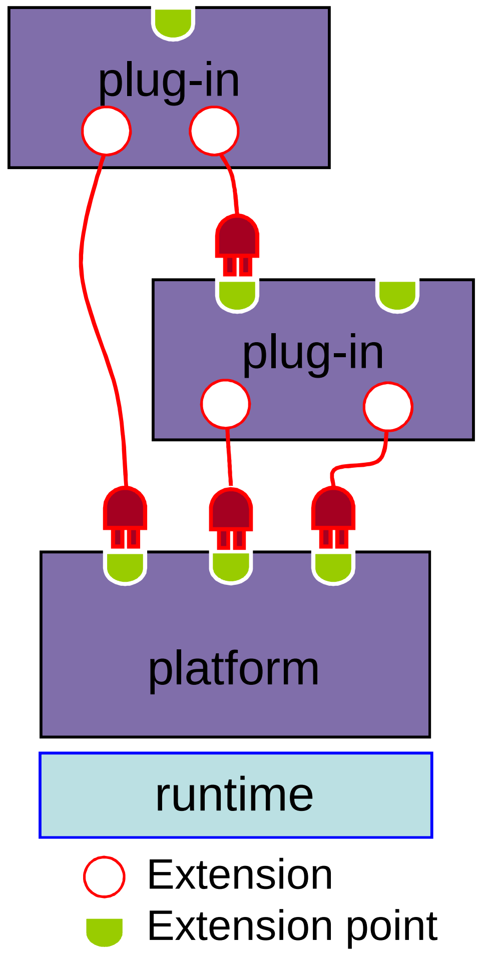
Outline



Open Source
2000
2001
2002
2003
2004
2005
2006
2007
Fall
Project Starts
June
Tech Preview
November
1.0
June
2.0
March
2.1
June
3.0
June
3.1
June
3.2
June
3.3
Closed Development
Open Development
Modularity Extensibility API
Community Transparency
Process Reflection
Scaling Agility
Team Tools
Peopleware

Everything is a plug-in

- **Classes and JARs are not sufficent**
- **plug-in == component**
  - set of contributions
  - smallest unit of Eclipse function
  - details spelled out in plug-in manifest
- **explicit dependencies**
- **explicit hooks for extension**
  - extension points



Key Lessons

- Modularity matters
  - Everything is a plug-in
  - “no exceptions”
- Make it easy to write extensions
  - Plug-in development environment
- Extensibility through extension points
  - Simple but consistent
  - “no exceptions”

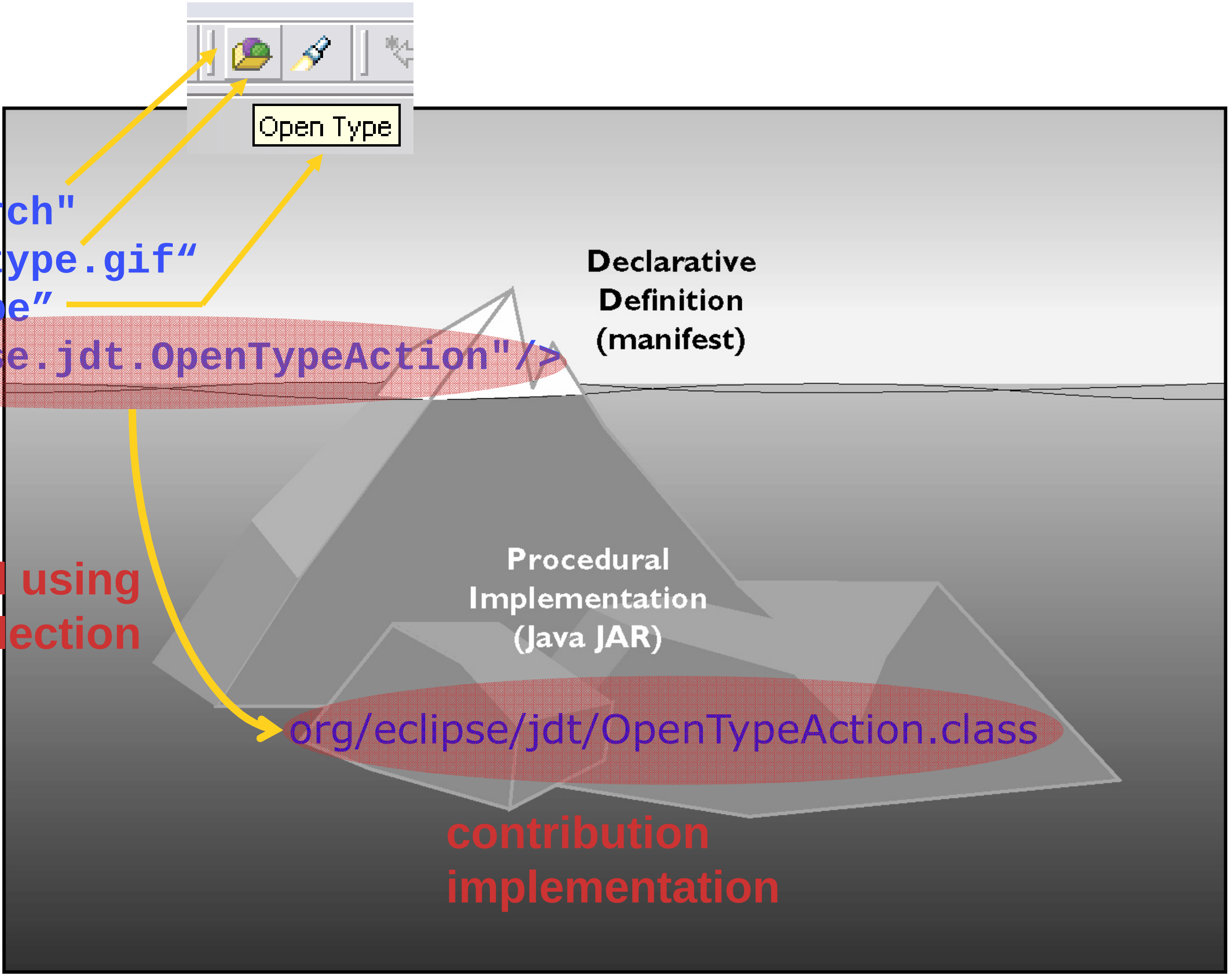
➢ Scalability concerns built in from the beginning $\Rightarrow$ Growth Path

Growth Path...

user visible appearance

```
<action
   toolbarPath="search"
   icon="icons/opentype.gif"
   toolTip="Open Type"
   class="org.eclipse.jdt.OpenTypeAction"/>
```

lazily instantiated using reflection


Open Type
Declarative Definition (manifest)
Procedural Implementation (Java JAR)
org/eclipse/jdt/OpenTypeAction.class

contribution implementation

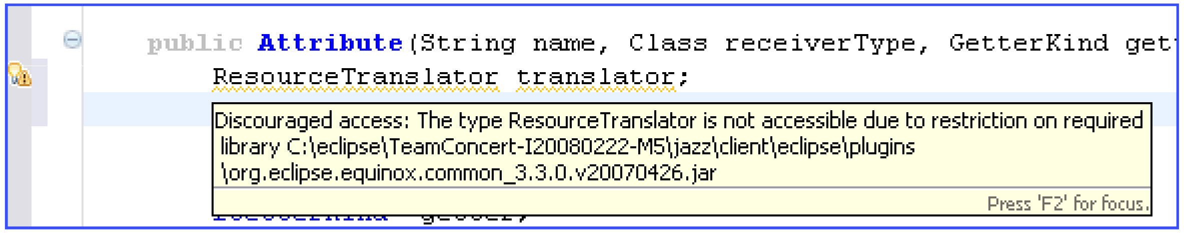
APIs

- **decisions you make today impact what you can do tomorrow**
- APIs matter
  - define consistent, concise API
  - explicit API conventions
  - binary compatibility is highest priority

$\Rightarrow$ **APIs are a huge commitment**
  - we would rather provide less API than desired (and augment) than provide the wrong (or unnecessary) API and need to support it indefinitely
  - API layers…

APIs Tool Support

- Since eclipse 3.1
  - Access restrictions reported as you type

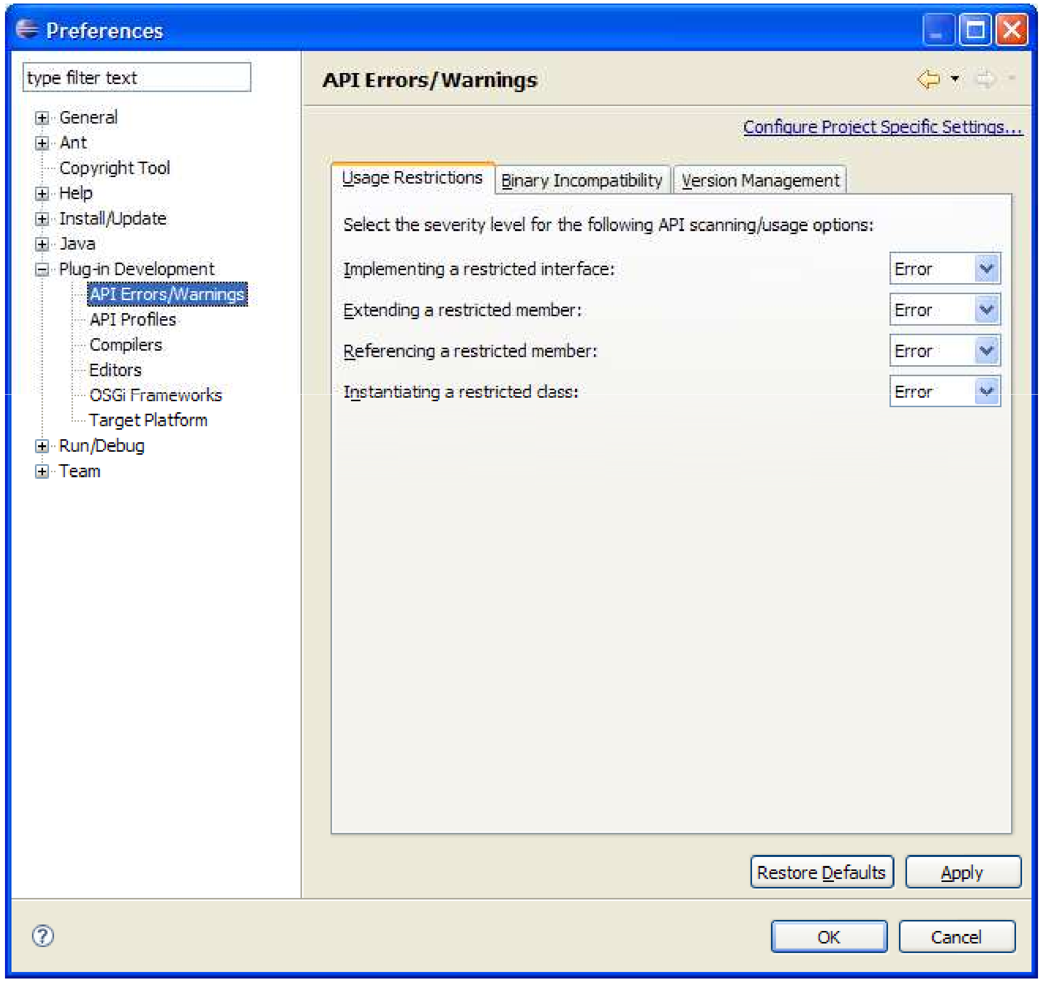


```
public Attribute(String name, Class receiverType, GetterKind get
    ResourceTranslator translator;
```

Discouraged access: The type ResourceTranslator is not accessible due to restriction on required library C:\eclipse\TeamConcert-I20080222-M5\jazz\client\eclipse\plugins \org.eclipse.equinox.common_3.3.0.v20070426.jar

New: Eclipse API Tools

- Support to define an API baseline
  - e.g. Eclipse 3.3 when working on 3.4
- Check access restrictions
  - API javadoc tags: @noimplement, @noinstantiate, @noextend
- Detect binary compatibility violations
- Detect version problems
  - @since
- Problems are reported during builds

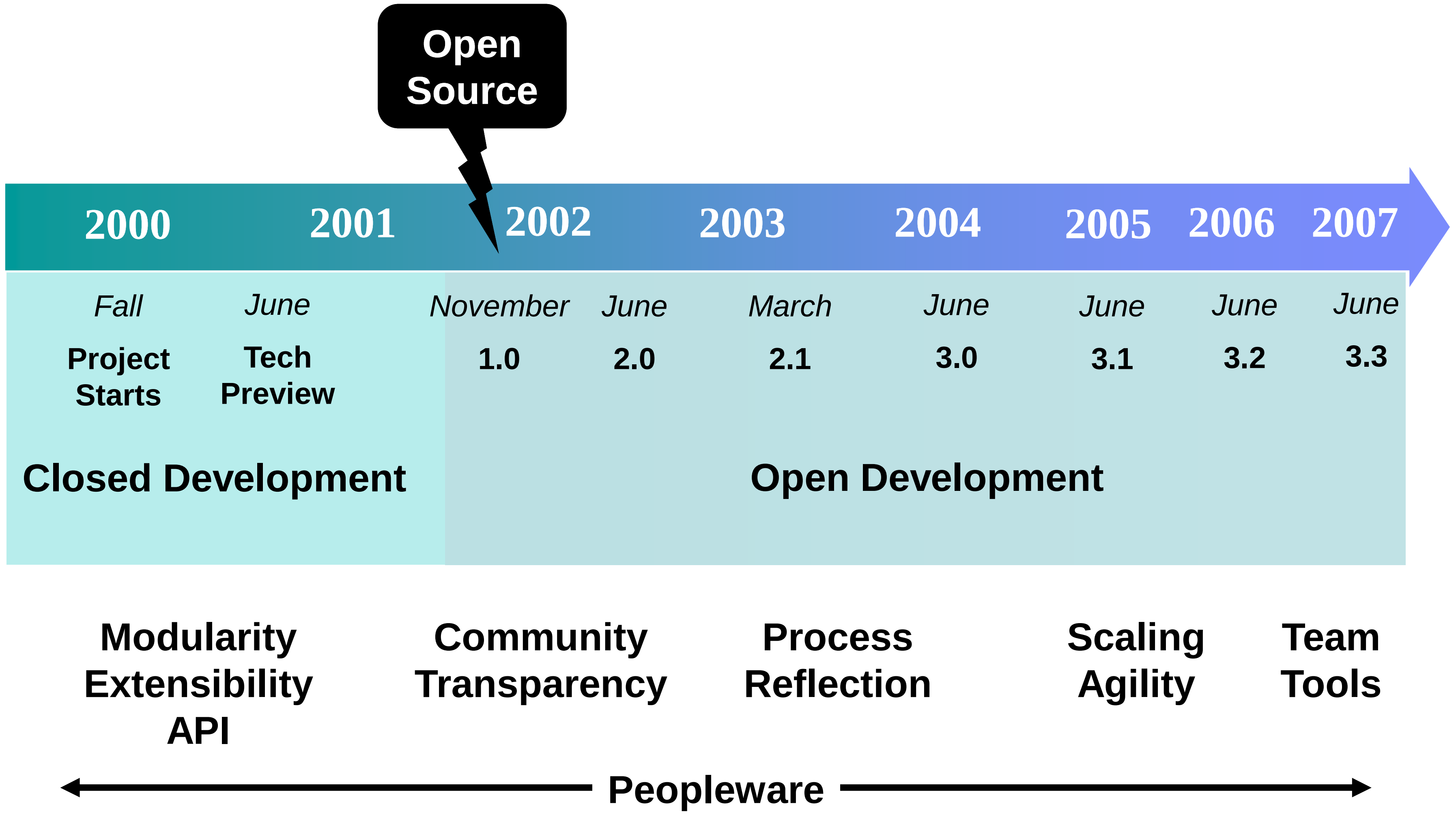


API Errors/Warnings

Select the severity level for the following API scanning/usage options:

Implementing a restricted interface:	Error
Extending a restricted member:	Error
Referencing a restricted member:	Error
Instantiating a restricted class:	Error

Outline


Open Source
2000
2001
2002
2003
2004
2005
2006
2007
Fall
Project Starts
June
Tech Preview
November
1.0
June
2.0
March
2.1
June
3.0
June
3.1
June
3.2
June
3.3
Closed Development
Open Development
Modularity
Extensibility
API
Community
Transparency
Process
Reflection
Scaling
Agility
Team
Tools
Peopleware

Closed development

- The Swiss Bank approach to software development
  - If it hasn't shipped it doesn’t exist
- Strong firewall between developers and customers
- Shipping matters

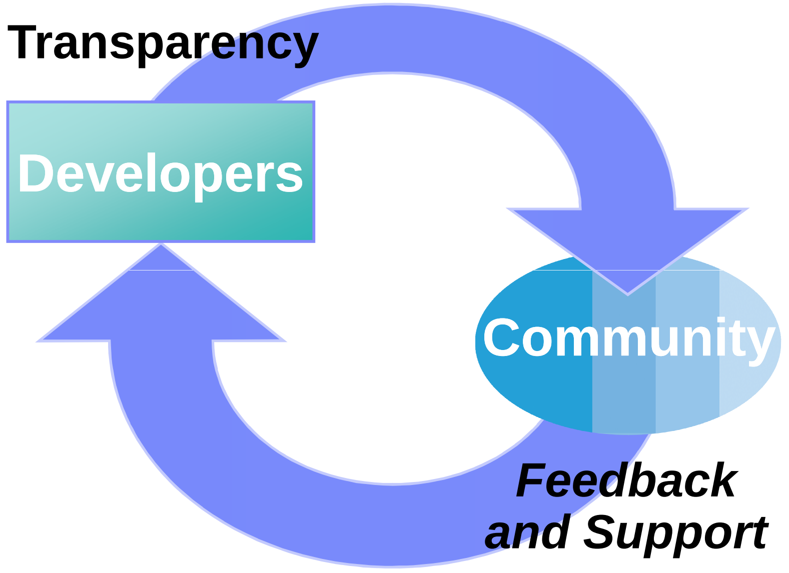
November 2001: “Open Source”
Reaction from the development team

You want us to do what?
Why are we doing this again?
Code in public?
Answer all those dumb questions?
Have technical discussions in public?

Lessons learned

Transparency and predictability enable feedback

- Transparency helps existing development
  - Better understanding of current status
  - Responding to feedback takes time, but pays off
- Use same communication channels inside as outside
  - Helped communication in our globally distributed team



Transparency: “Same Channels”

- Litmus test for transparency:

  Developers and community use the same channels

  - Newsgroups
    - Community and developers ask and answer questions
  - Mailing lists
    - Community and developers subscribe
  - Bugs, dashboards, meeting notes, blogs, wikis
    - Visible to the community
  - Internal builds
    - Downloadable by the community and the team

Open Commercial Development

- Open Commercial Development is more than publishing the source code
- Open, transparent process, from feature requests and planning through delivery
- What can community members do:
  - Download, try out, and provide feedback on betas and incubators, including source code
  - Access, Create, and update work items
  - Access milestone and component iteration plans
  - Access the development wiki
  - Participate in discussions on the development community newsgroups
- Example: www.jazz.net

Milestones Promote Transparency & Accountability

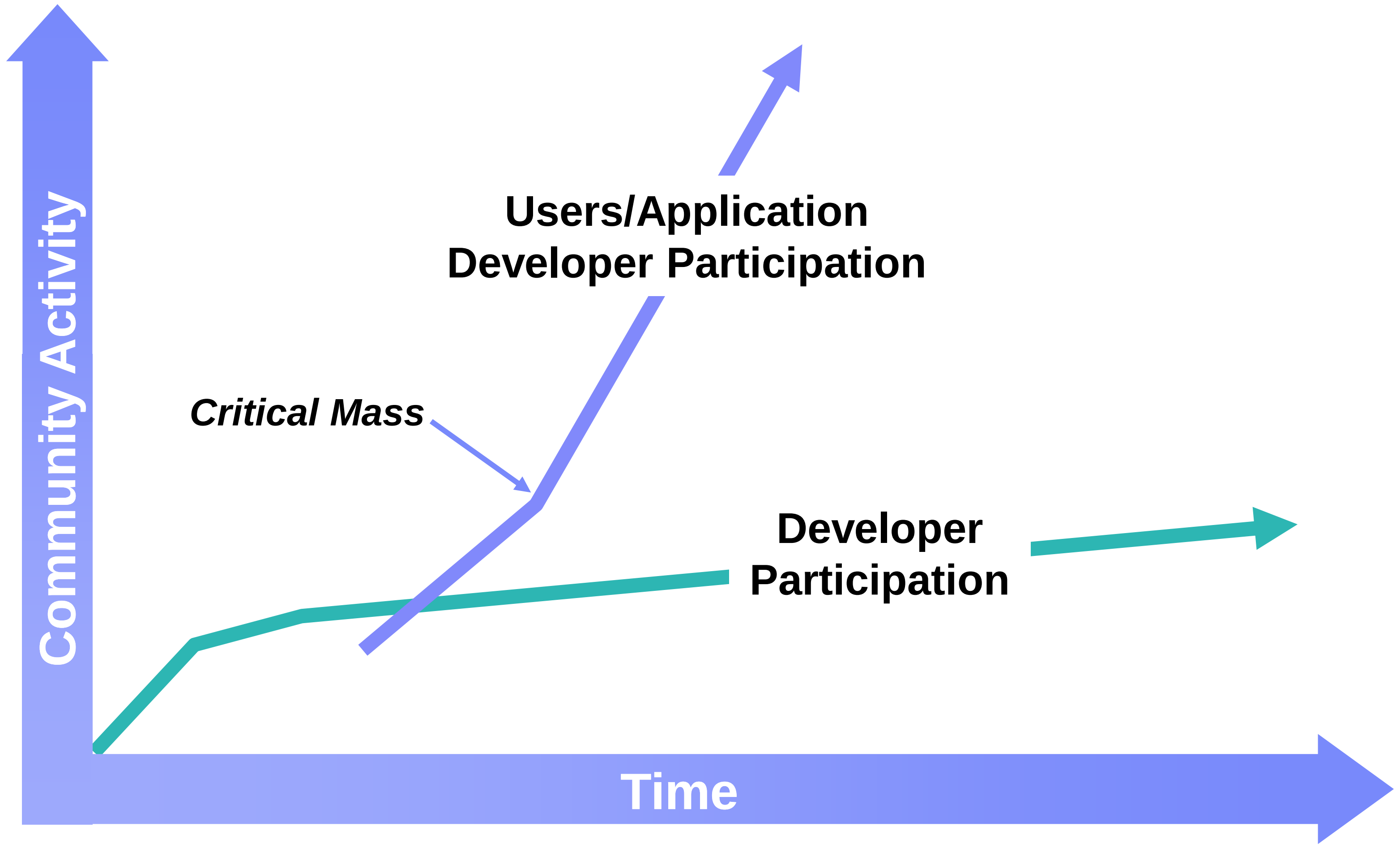
Make it Public:

- Milestones make new work visible to the teams, the community
  - You know people are watching
  - Add incremental value
  - Announce new features New & Noteworthy
- Integration builds are picked up for “self hosting”
  - You know your teammates rely on it working

Result:

- Sense of responsibility
- Accountability
- You learn by shipping, so ship more often…

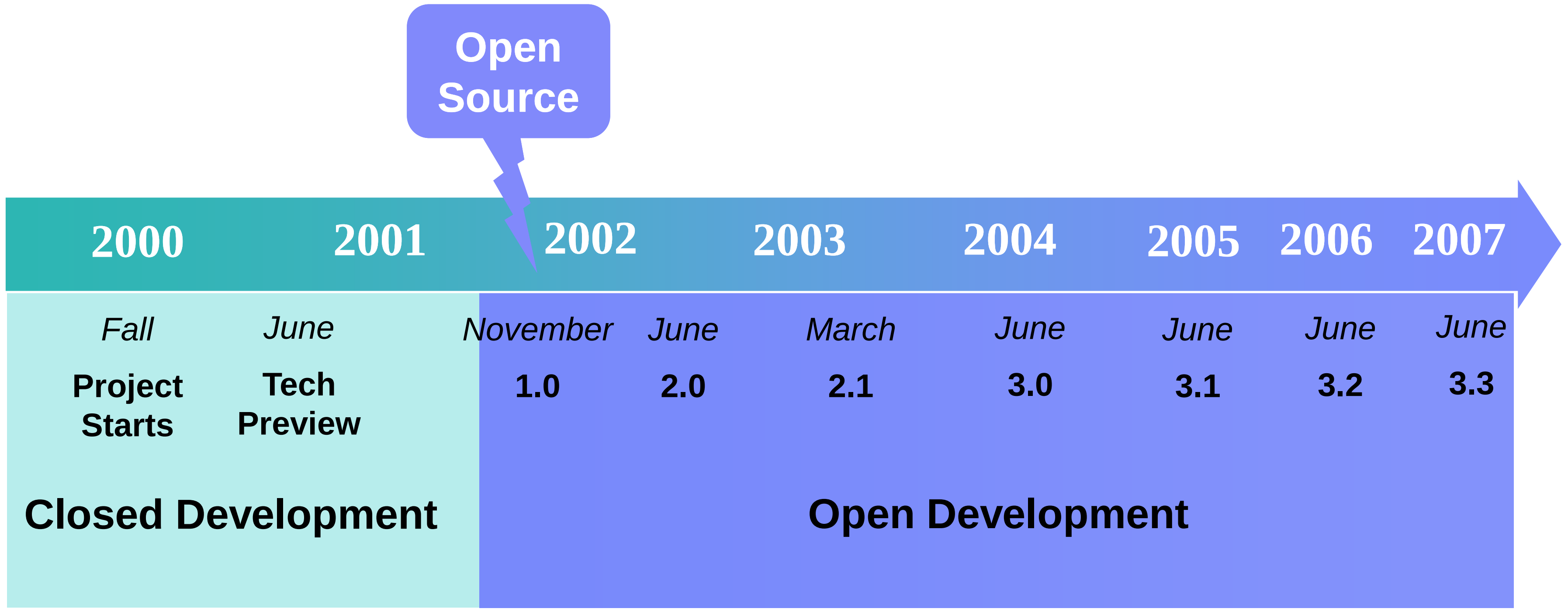
A community reaching critical mass



Lessons learned:
The "village effect"

- A large organization can act like a smaller organization
- Development team becomes a face
- Communication flows are visible for all to see
- Everyone is accountable

Outline



Modularity
Extensibility
API

Community
Transparency

Process
Reflection

Scaling
Agility

Team
Tools

← **Peopleware** →

Our Goal

- We ship on time, every time.
- **Users like** our products and are loyal. Their number grows.
- **Developers are proud** of their products and **enjoy working** on them.

Our Shipping Pattern

- We ship yearly
- Shorter doesn’t give enough time for significant work
- Longer than a year allows too much time to get distracted, go too far off base
- We don’t ship near Christmas
- We don’t ship in the summer
- Thus we **ship in June**

Happy users

- Encourage feedback from users
- Listen to the feedback
    - Let them know that your are listening
- Incorporate the feedback
    - Proof that you listened and understood

- Be predictable

Happy Developers

- Impact
- Responsibility
- Productivity
- Technical challenge
- Acceptable stress levels
- Predictability
- Happy users
- Shipping on time

Consequences

- Decentralize responsibility
  - Allow for highly autonomous groups
  - Everybody feels responsible and accountable

- Ensure transparency across groups

➢ A **collaborative**, **consensus** based development process

Outline

Open Source

2000	2001	2002		2003	2004	2005	2006	2007
Fall	*June*	*November*	*June*	*March*	*June*	*June*	*June*	*June*
Project Starts	**Tech Preview**	**1.0**	**2.0**	**2.1**	**3.0**	**3.1**	**3.2**	**3.3**

Closed Development | **Open Development**

Modularity
Extensibility
API

Community
Transparency

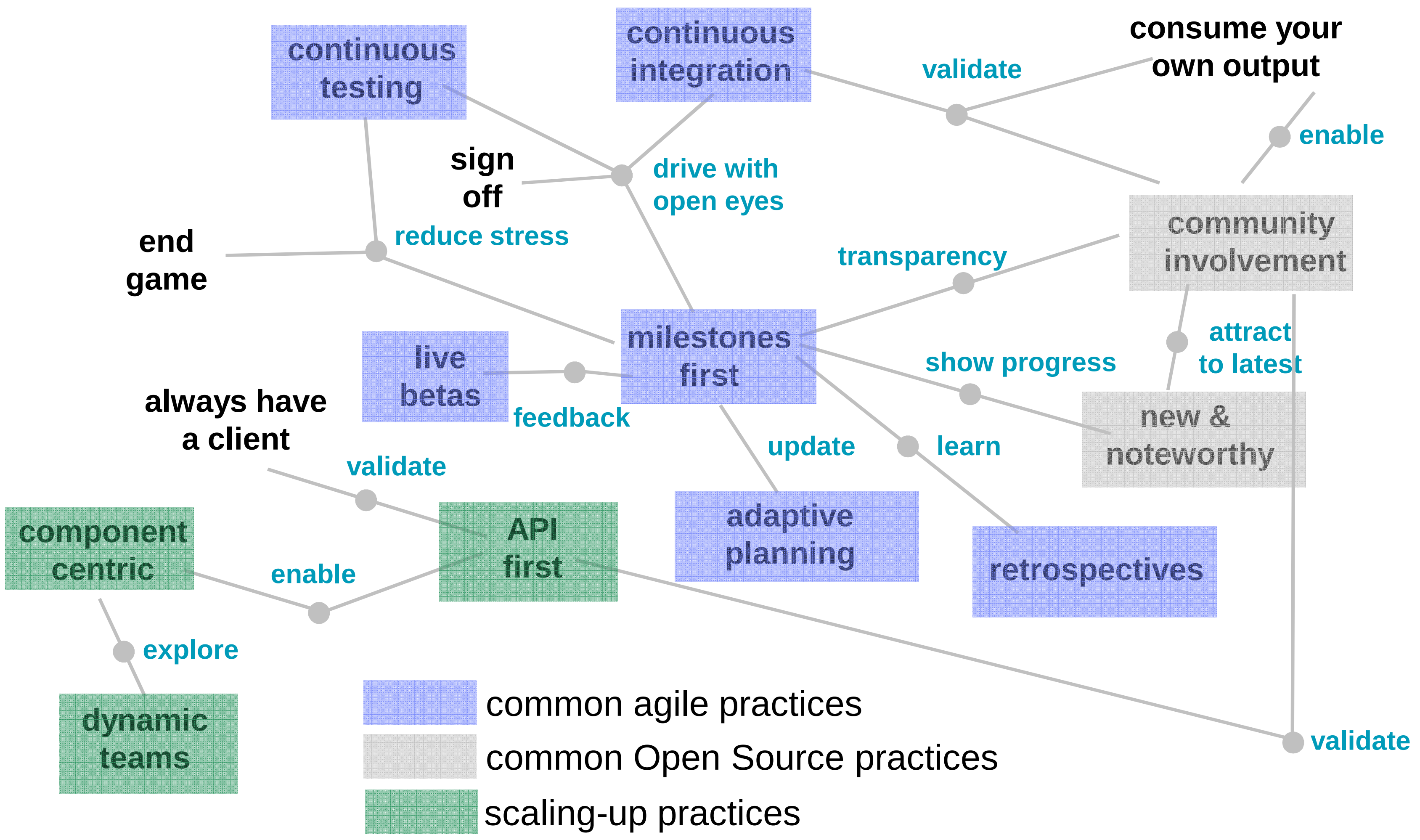
Process
Reflection

Scaling
Agility

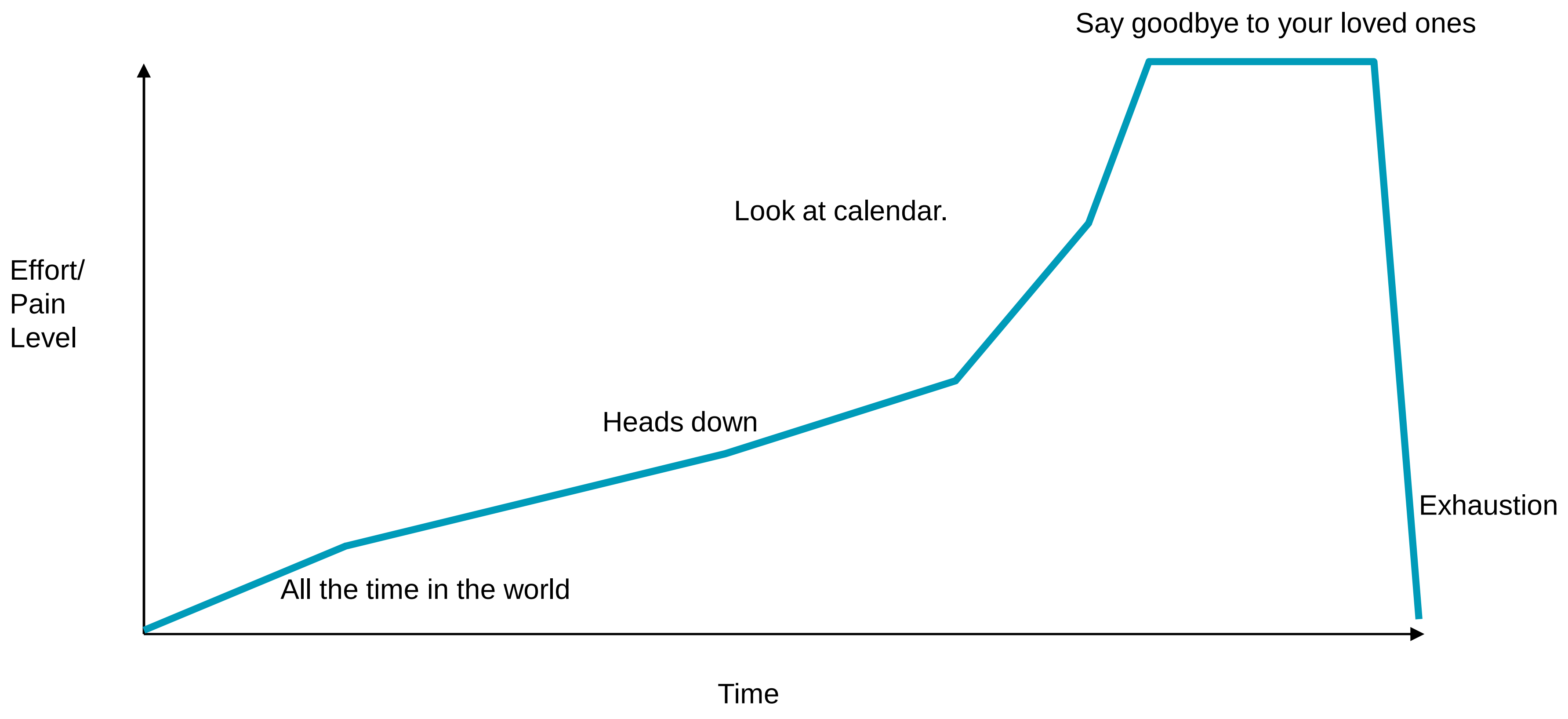
Team
Tools

← **Peopleware** →

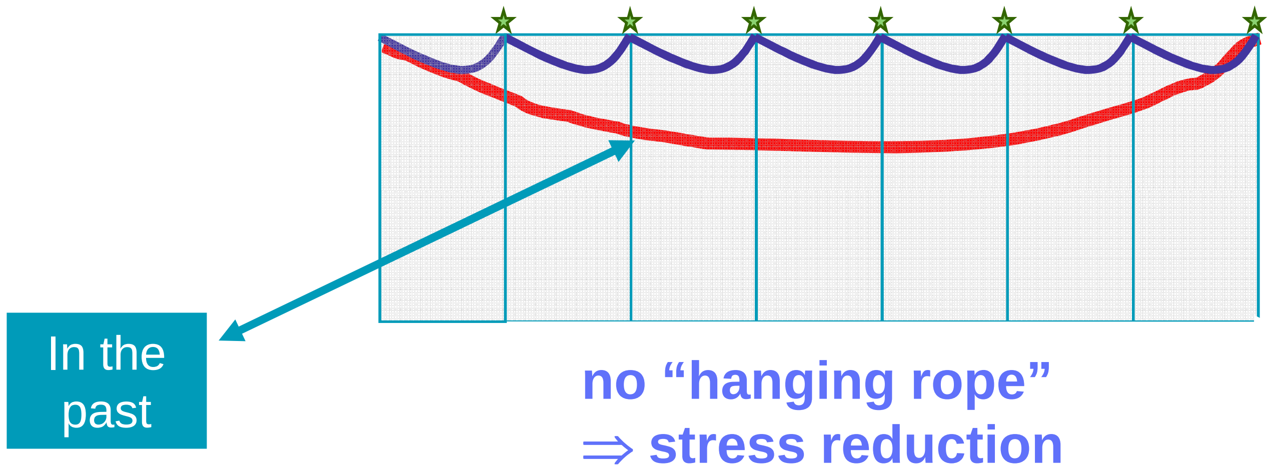
The Eclipse Way Practices



In the Past…

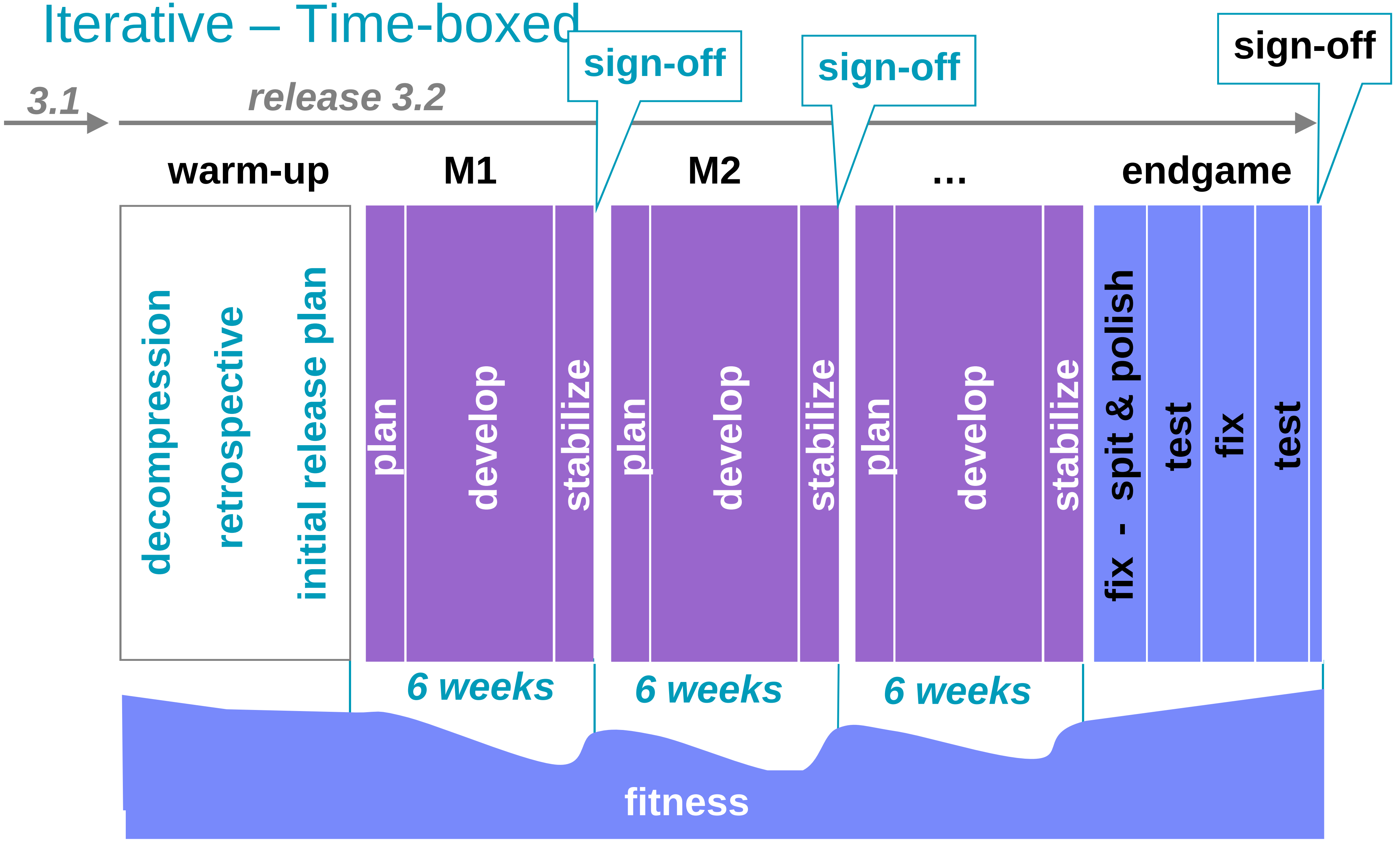

Say goodbye to your loved ones
Look at calendar.
Effort/ Pain Level
Heads down
Exhaustion
All the time in the world
Time

Iterative – No hanging rope



no “hanging rope”
⇒ stress reduction

Iterative – Time-boxed

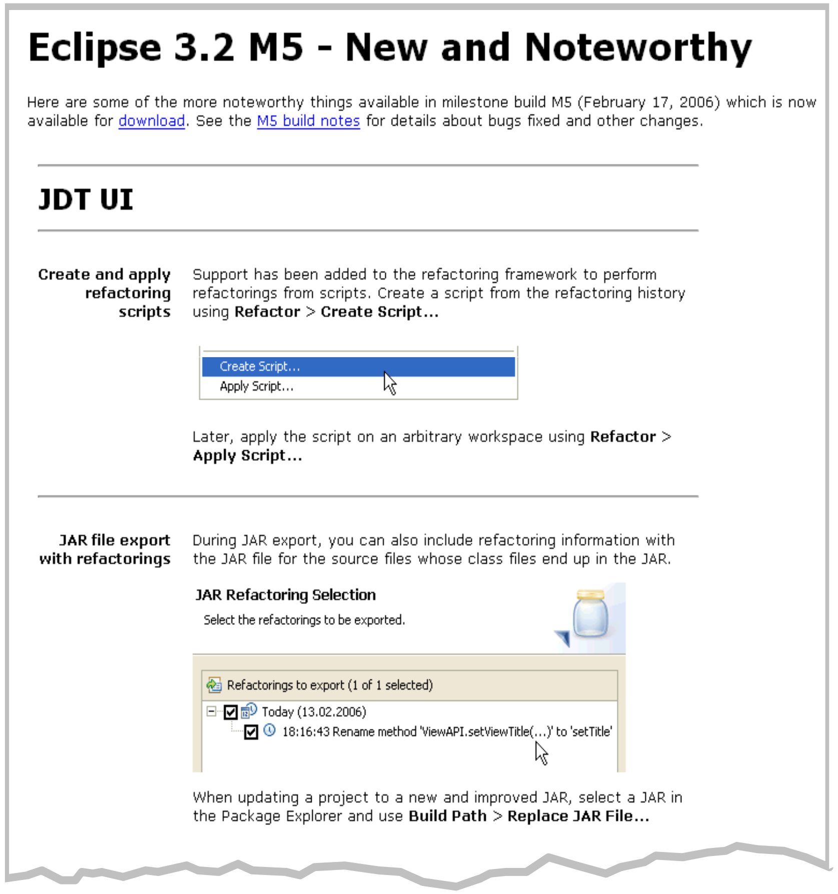


Iterative **and** Incremental

- **make iteration results visible**
  - we need **feedback** on our latest!
  - reduce stale defect reports

- **incremental-ness** enables **feedback**

new & noteworthy



Eclipse 3.2 M5 - New and Noteworthy

Here are some of the more noteworthy things available in milestone build M5 (February 17, 2006) which is now available for download. See the M5 build notes for details about bugs fixed and other changes.

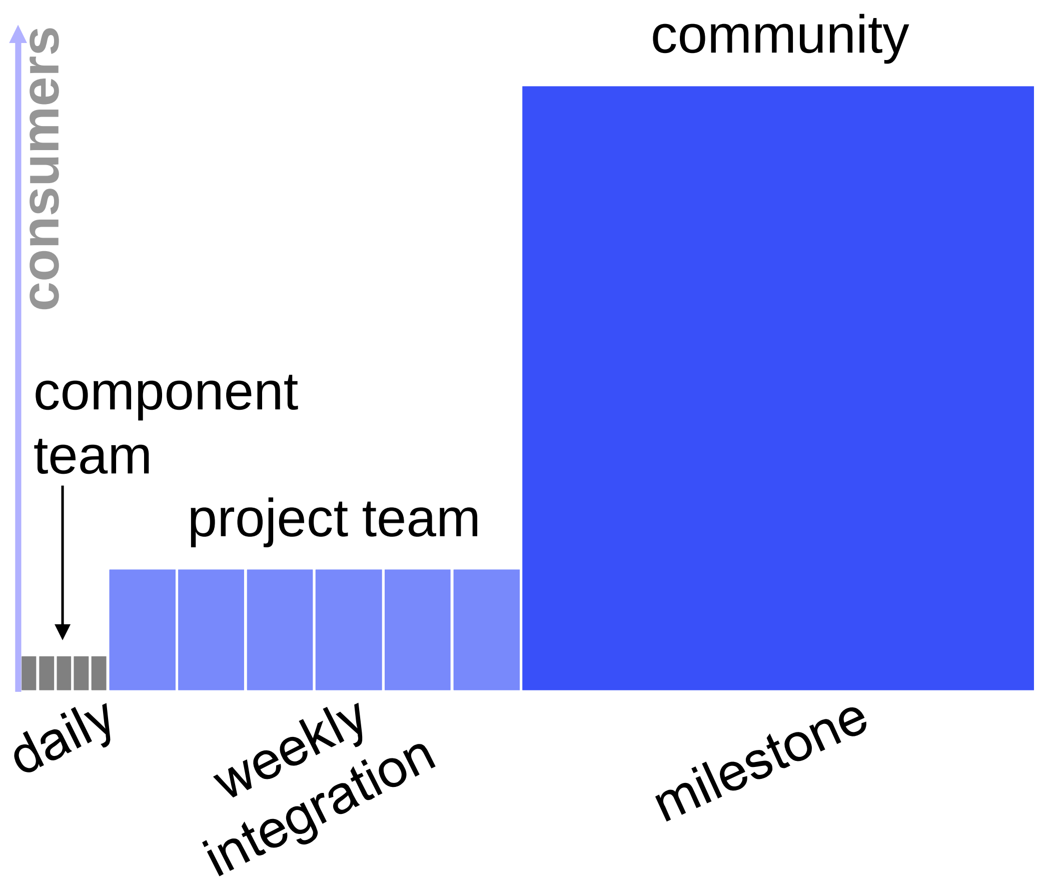
JDT UI

Create and apply refactoring scripts	Support has been added to the refactoring framework to perform refactorings from scripts. Create a script from the refactoring history using **Refactor > Create Script...** Later, apply the script on an arbitrary workspace using **Refactor > Apply Script...**
JAR file export with refactorings	During JAR export, you can also include refactoring information with the JAR file for the source files whose class files end up in the JAR. When updating a project to a new and improved JAR, select a JAR in the Package Explorer and use **Build Path > Replace JAR File...**

Create Script...
Apply Script...
JAR Refactoring Selection
Select the refactorings to be exported.
Refactorings to export (1 of 1 selected)
Today (13.02.2006)
18:16:43 Rename method 'ViewAPI.setViewTitle(...)' to 'setTitle'

Builds

- continuously **consumable**
- continuously **interesting**
- continuous **listening**
  - users have influence
  - encourages feedback



➢ we continuously ***consume our own output***

Summary

- **It is about being continuous**
  - **Continuous** iterative and adaptive planning
  - **Continuous** design/refactoring
  - **Continuous** integration/testing
  - **Continuous** delivering/demos
  - **Continuous** feedback
  - **Continuous** learning

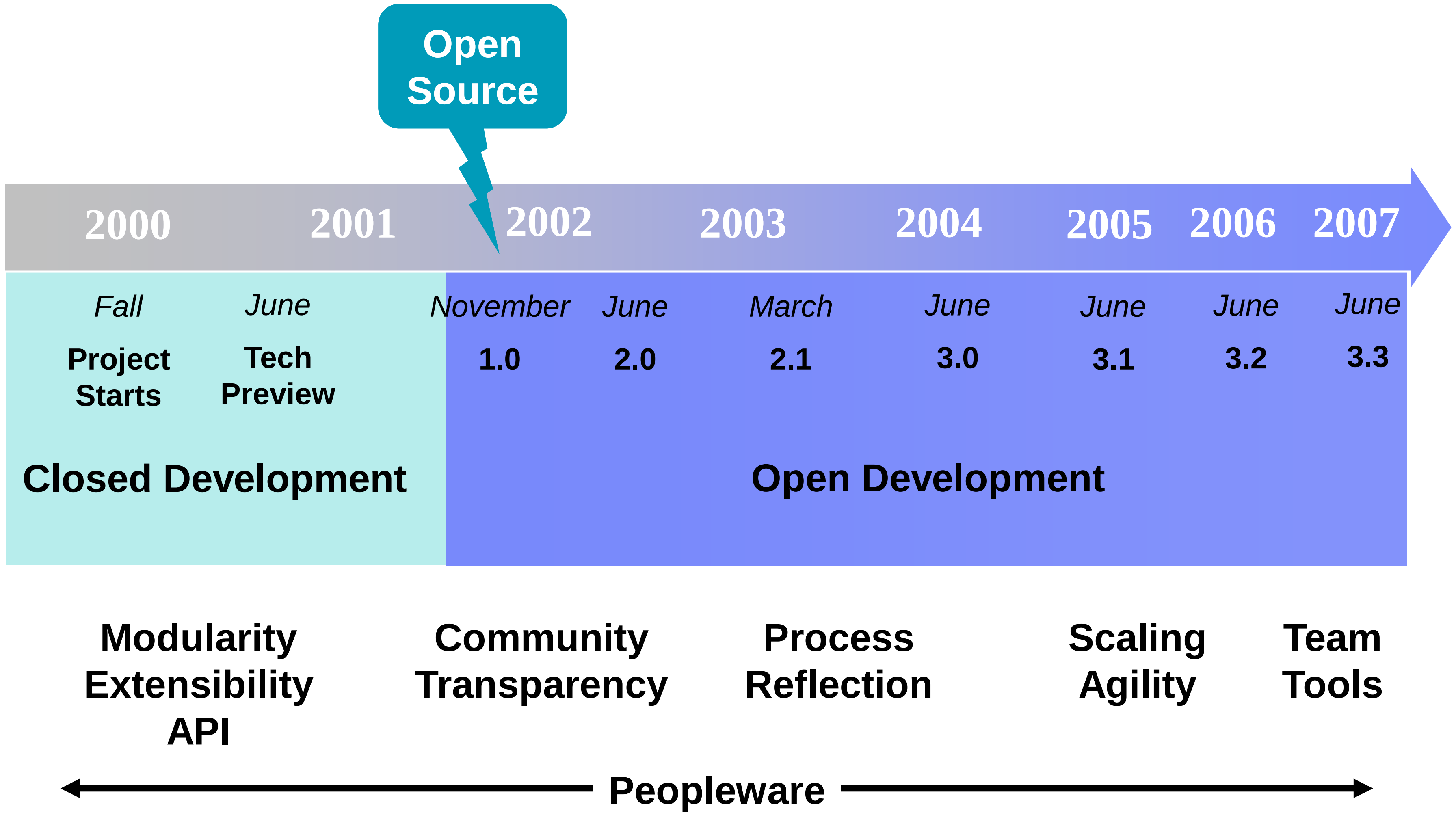
➢ **Continuous health**

➢ **Many effective teams work like this**

What is behind the Eclipse Way

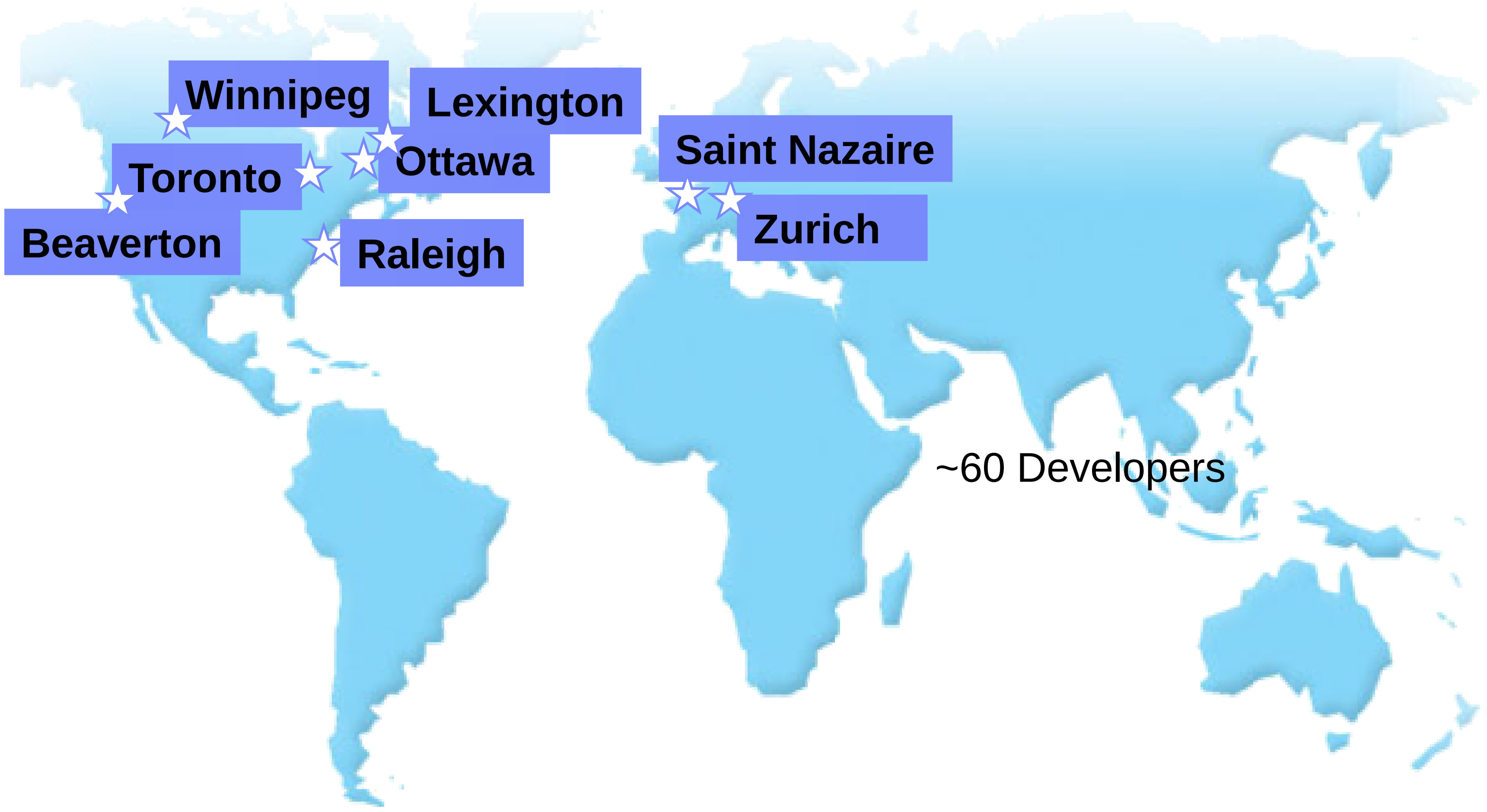
- Practices underpinned with **values**
  - ship quality on time
- **Used**, **developed** and **improved** over time
  - A mix of practices that worked for us
  - Another mix of practices works for others
- Practices are from all kinds of sources
  - XP, Scrum, Crystal Clear, RUP, …
  - Patterns - Organizational Patterns of Agile Software Development – Coplien
- It is **not low ceremony**
  - Approvals, verifications, reviews
- It is **agile**: incremental, iterative, collaborative, transparent, customizable
  - And it scales up

Outline



Open Source
2000
2001
2002
2003
2004
2005
2006
2007
Fall
Project Starts
June
Tech Preview
November
1.0
June
2.0
March
2.1
June
3.0
June
3.1
June
3.2
June
3.3
Closed Development
Open Development
Modularity
Extensibility
API
Community
Transparency
Process
Reflection
Scaling
Agility
Team
Tools
Peopleware

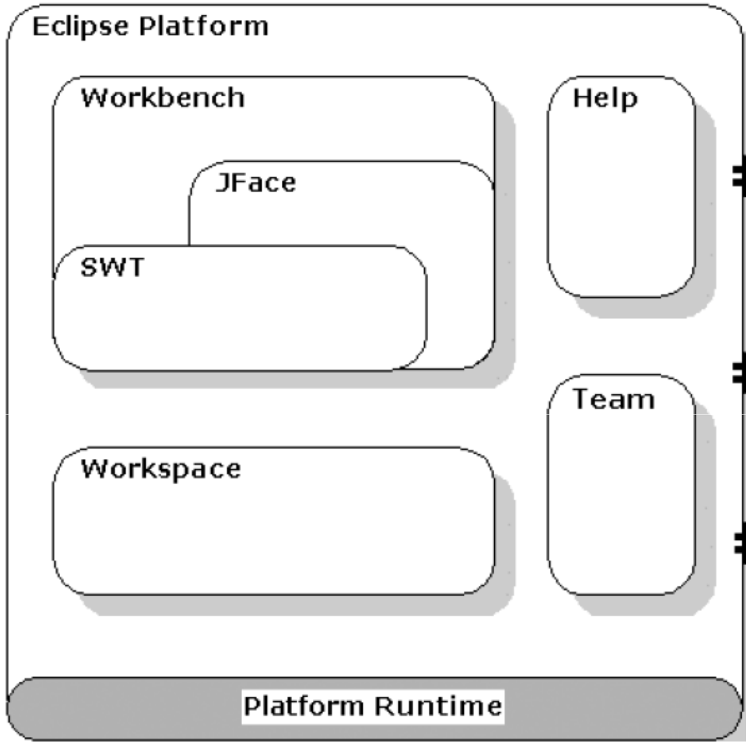
Scaling-up Agility



~60 Developers

Component Based Development

- **Component** based development
  - a **team** is responsible for one or more **component** at one **site**
  - **"architecture follows organization"**
  - dependencies through **APIs**
- **API first**



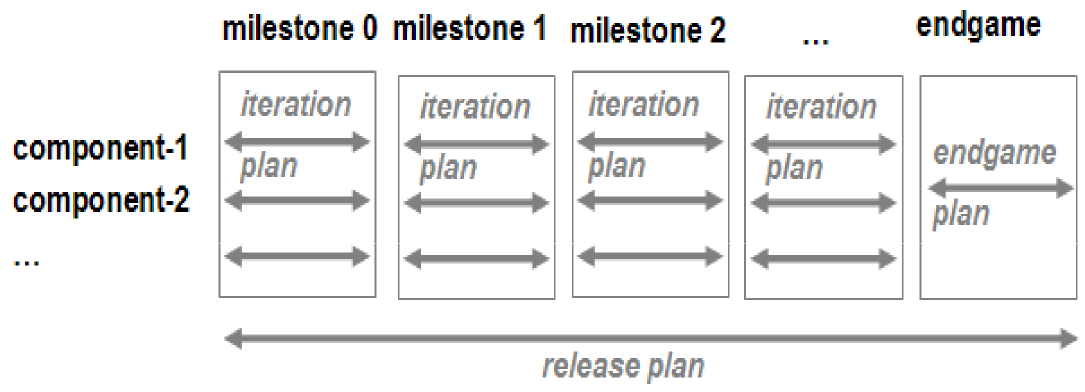
Eclipse Components

Team First

- Teams own a component
- Teams empowered to make decisions and owns:
  - Plan
  - Build
  - Test
- Each Team is different
  - Team has its own process and constantly tunes it
  - All teams agree on core practices
- Teams are self organized, interdisciplinary
  - Team member play different roles
    - developer, architect, releng, tester

Planning and Tracking Iterations

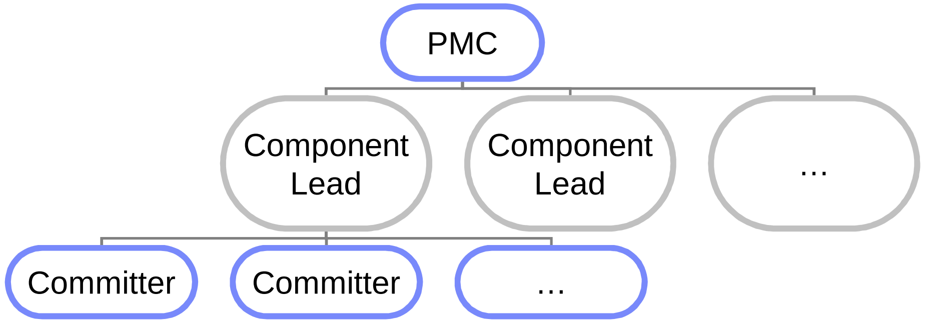
- **Same rhythm** across teams
- **Release plan** defines
  - rhythm
  - themes and features
  - ⇒ coarse grained



- **Iteration plan**
  - per team/component
  - defines
    - stories, tasks, enhancements, defects
  - ⇒ fine grained
- Project leadership team (PMC) defines themes and stories

Organization

- **Project leadership team**
  - Accountable for release plan
  - Themes
  - Facilitator, coordinator
    - encourages participatory decisions
      - e.g. top 5 architectural issues
- **Component lead**
  - Accountable for
    - iteration plan
    - test plans
    - component's architecture, UI, quality
- **Developer**
  - Accountable for code, tests



Scaling up Continuous Builds

- Continuous build for all components
  - used to sense integration issues
  - **rarely** green
- Each component has its own continuous build
  - **always** green
- Weekly integration of component baselines
  - **stabilized until** green

Collaboration Events

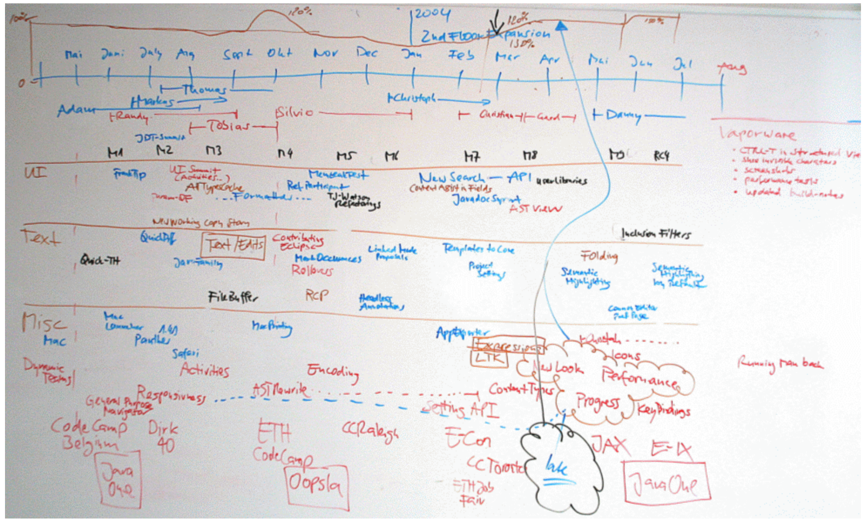
- Bi-weekly coordination calls with all component leads
- Daily stand-ups per team
- We all sign-off on deliverables

  "An enthusiastic GO from PDE" - Cherie

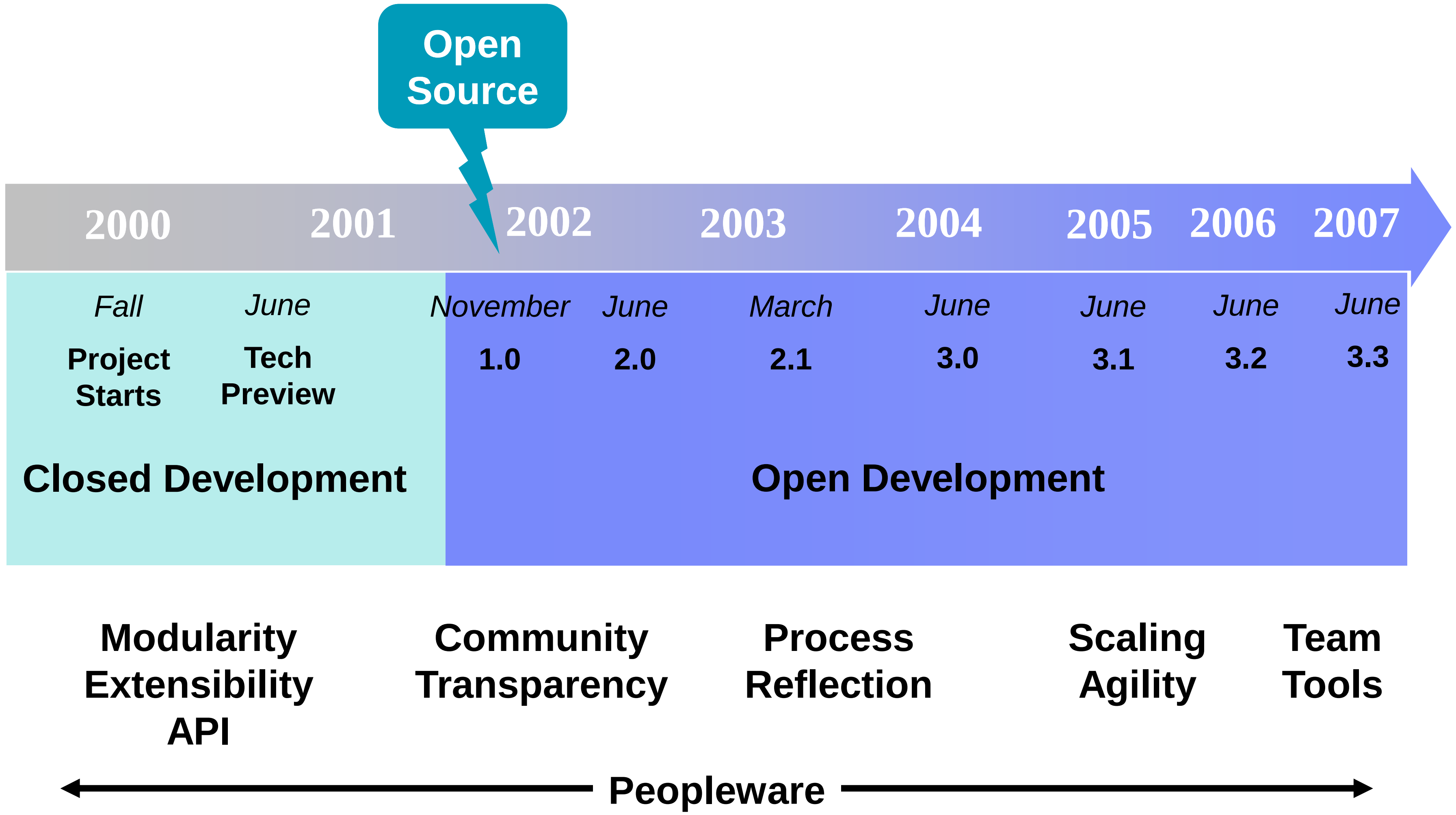
  "The best build of the year!" - Dejan

- Retrospectives/reflection at the end of each iteration and release
  - Steering committees aggregates

“GO from SWT"



Outline


Open Source
2000
2001
2002
2003
2004
2005
2006
2007
Fall
Project Starts
June
Tech Preview
November
1.0
June
2.0
March
2.1
June
3.0
June
3.1
June
3.2
June
3.3
Closed Development
Open Development
Modularity
Extensibility
API
Community
Transparency
Process
Reflection
Scaling
Agility
Team
Tools
Peopleware

But… there are Pain Points…

- joining a team
- get my environment configured to be productive
- what is happening in my team
- collecting progress status
- following the team’s process
- ad hoc collaboration/sharing of changes
- starting an ad hoc team

Collaboration

- is the fix in the build?
- what will be in the next build?
- tracking a broken build
- Avoid breaking a build/personal build
- why is this change in the build?
- reconstructing a context for a bug/build failure

Development

- creating, tracking iteration plans
- interrupting development due to a high priority bug fix
- working on multiple releases concurrently
- tracking the code review of a fix
- referencing team artifacts in discussions
- how healthy is a component?
- collecting project data/metrics?

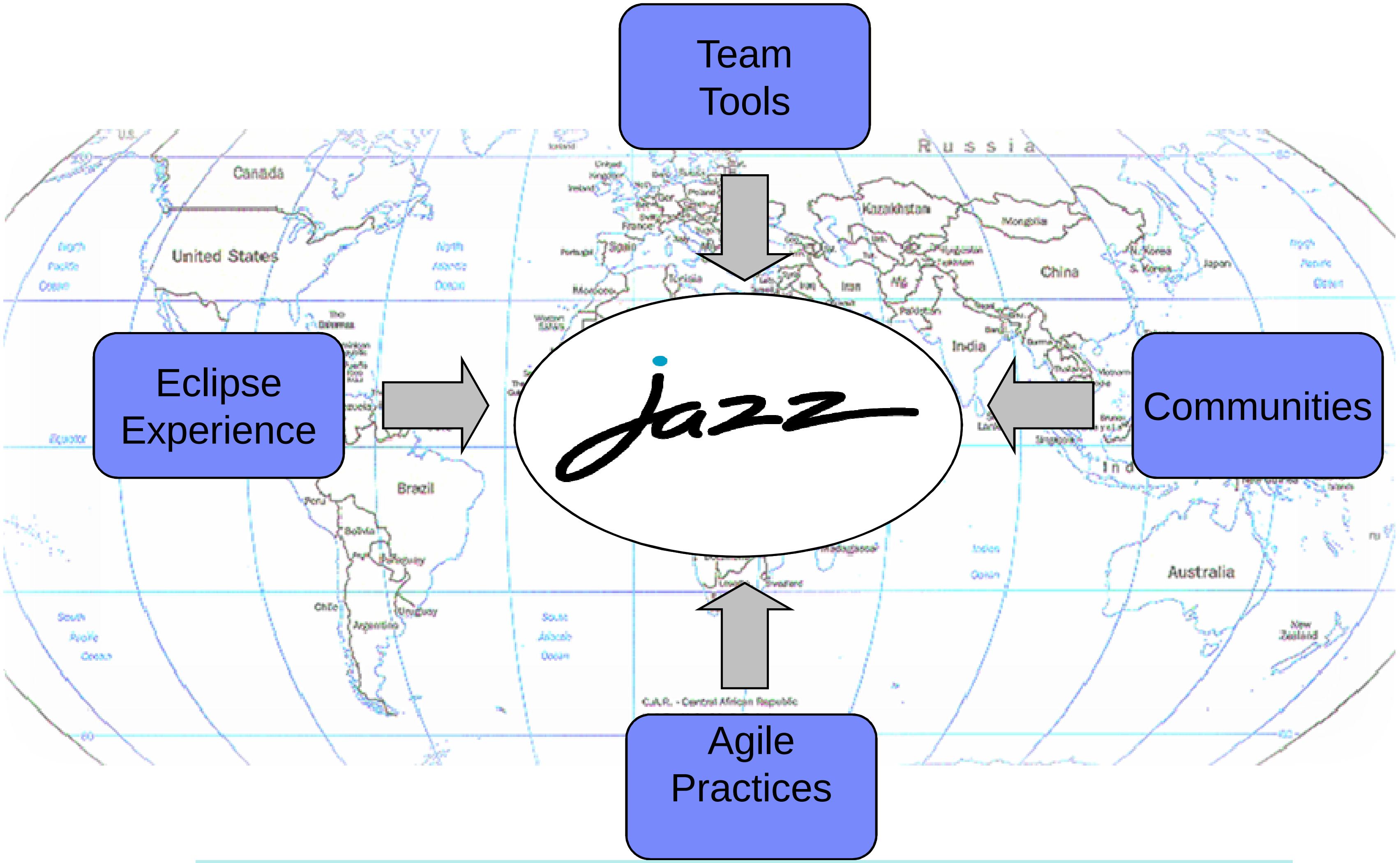
Project Management

Boring and painful

Affected Development Tools

	Work Items	SCM	Build	Reports	Project Mgt.
joining a team	X	X	X		
get my environment configured to be productive	X	X	X		
what is happening in my team?	X	X	X	X	X
collecting progress status	X		X	X	X
following the team's process	X	X	X		
ad hoc collaboration/sharing of changes	X	X	X		
starting an ad hoc team					X
is the fix in the build?					
run a personal build		X	X		
tracking a broken build	X		X		
why is this change in the build?	X	X	X		
reconstructing a context for a bug/build failure		X	X		
interrupting current work due to a high priority bug fix	X	X			
Snapshot of changes without sharing	X	X			
working on multiple releases concurrently		X	X		X
tracking the code review of a fix	X	X			
referencing team artifacts in discussions	X				X
how healthy is a component?			X	X	X

⇒ **integrated** tool set

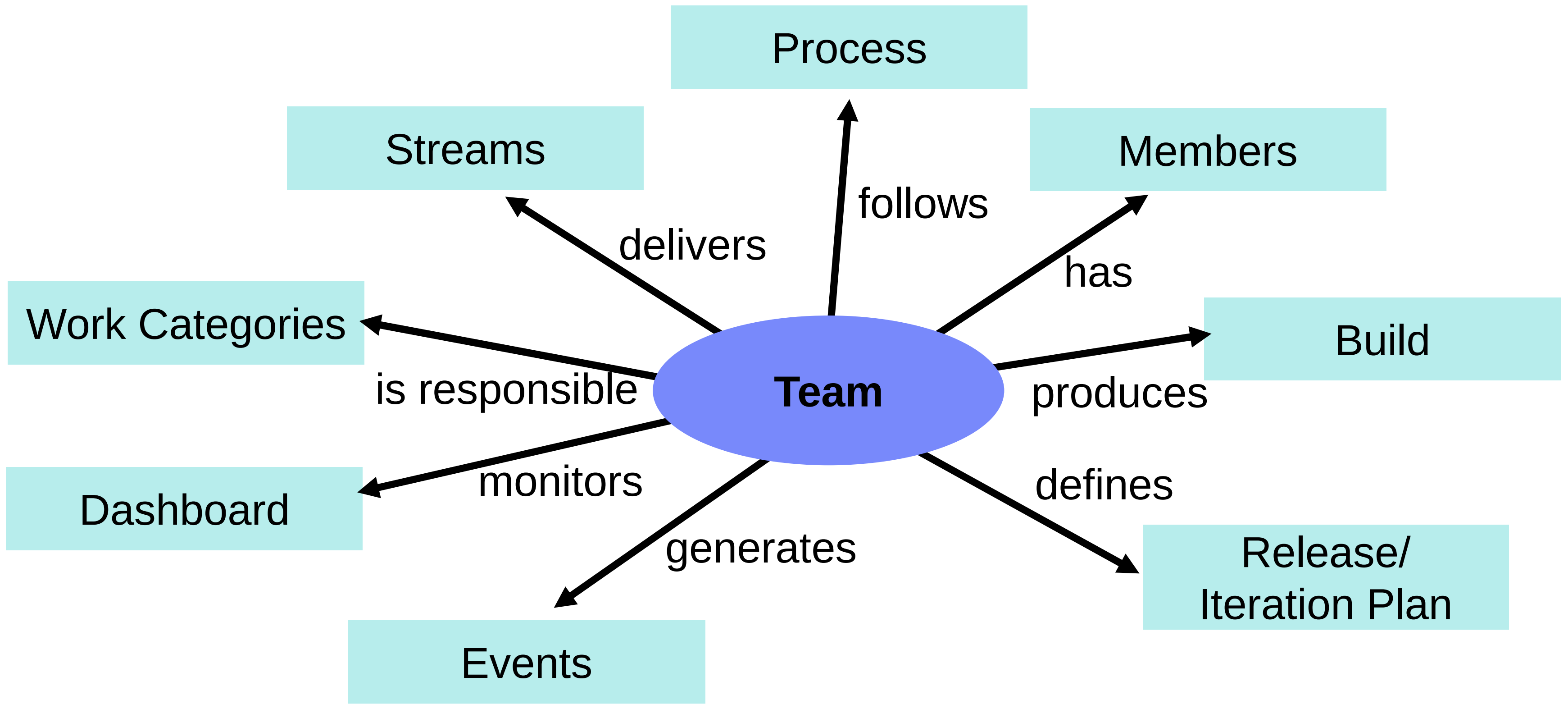


Goal: a scalable, extensible team collaboration platform
for seamlessly integrating tasks across the software lifecycle.

Team First: What if your tools knows more about the team…

- … about **your teams**
- … about **your teams artifacts** and **linkages**
- … rules under which circumstances **code** can be **delivered**
  - Code quality, traceability, test runs, intellectual property
- … how to **bootstrap** a **project**
- … **how to** help new team members get **started**
- … your important **work item types** and their **state transitions**
- …

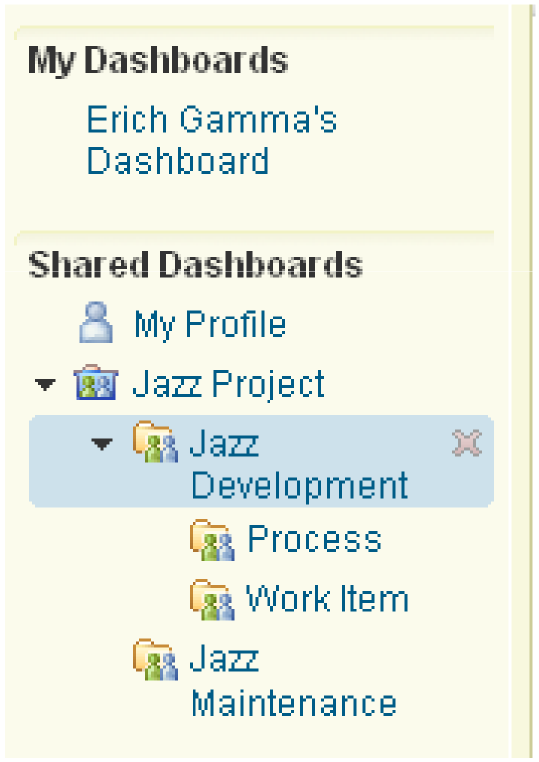
Team First



Process
follows
Streams
delivers
Members
has
Work Categories
is responsible
Team
Build
produces
Dashboard
monitors
generates
defines
Release/ Iteration Plan
Events

Team First: Scaling-up

- **Contributor**
  - Repository workspace
  - Private builds
  - My events
- **Team**
  - Team stream
    - Sharing change sets
  - Continuous build
  - Team events
- **Teams of Teams**
  - Integration/stabilization streams
    - Sharing baselines
  - Integration builds



My Dashboards

Erich Gamma's Dashboard

Shared Dashboards

My Profile

Jazz Project

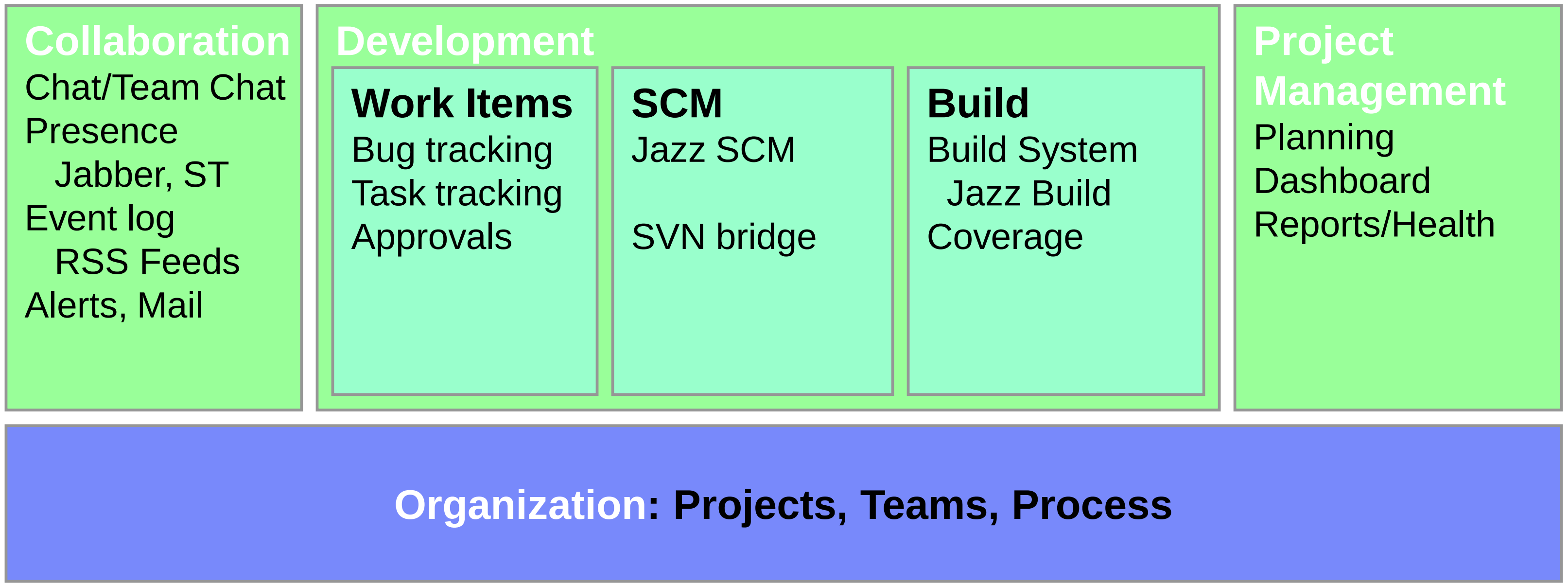
Jazz Development

Process

Work Item

Jazz Maintenance

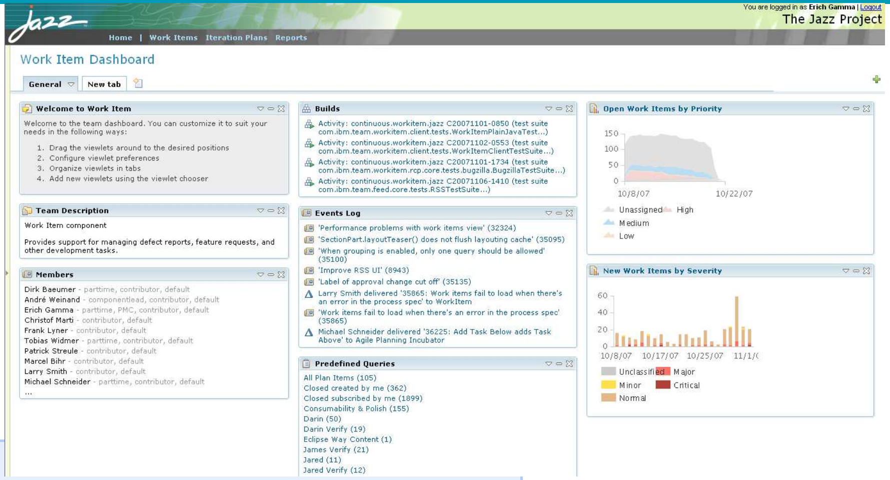
Integrated Tool Set



“a frictionless surface for development by eliminating or automating many of the daily activities of the team” (Grady Booch)

Transparency

- transparency in planning
  - dynamic plans
- transparency in development
  - automatic linking
  - build results/reports
  - dashboard
- transparency in the end game
  - code reviews
  - verification
- transparency in process
  - team structure
  - team roles


You are logged in as **Erich Gamma** | Logout

The Jazz Project

Home | Work Items Iteration Plans Reports

Work Item Dashboard

General | New tab

Welcome to Work Item

Welcome to the team dashboard. You can customize it to suit your needs in the following ways:

1. Drag the viewlets around to the desired positions
2. Configure viewlet preferences
3. Organize viewlets in tabs
4. Add new viewlets using the viewlet chooser

Team Description

Work Item component

Provides support for managing defect reports, feature requests, and other development tasks.

Members

Dirk Baeumer - parttime, contributor, default
André Weinand - componentlead, contributor, default
Erich Gamma - parttime, PMC, contributor, default
Christof Marti - contributor, default
Frank Lyner - contributor, default
Tobias Widmer - parttime, contributor, default
Patrick Streule - contributor, default
Marcel Bihr - contributor, default
Larry Smith - contributor, default
Michael Schneider - parttime, contributor, default
...

Builds

Activity: continuous.workitem.jazz C20071101-0850 (test suite com.ibm.team.workitem.client.tests.WorkItemPlainJavaTest...)
Activity: continuous.workitem.jazz C20071102-0553 (test suite com.ibm.team.workitem.client.tests.WorkItemClientTestSuite...)
Activity: continuous.workitem.jazz C20071101-1734 (test suite com.ibm.team.workitem.rcp.core.tests.bugzilla.BugzillaTestSuite...)
Activity: continuous.workitem.jazz C20071106-1410 (test suite com.ibm.team.feed.core.tests.RSSTestSuite...)

Events Log

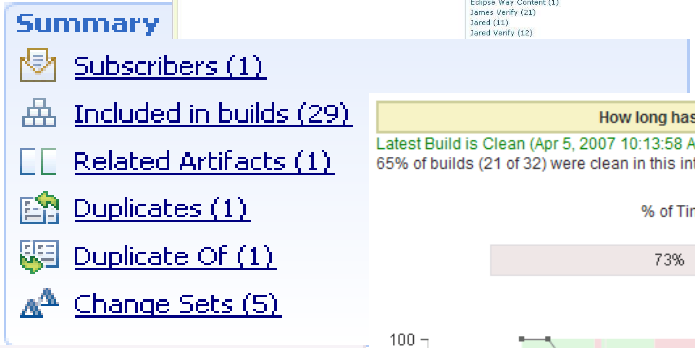
'Performance problems with work items view' (32324)
'SectionPart.layoutTeaser() does not flush layouting cache' (35095)
'When grouping is enabled, only one query should be allowed' (35100)
'Improve RSS UI' (8943)
'Label of approval change cut off' (35135)
Larry Smith delivered '35865: Work items fail to load when there's an error in the process spec' to WorkItem
'Work items fail to load when there's an error in the process spec' (35865)
Michael Schneider delivered '36225: Add Task Below adds Task Above' to Agile Planning Incubator

Predefined Queries

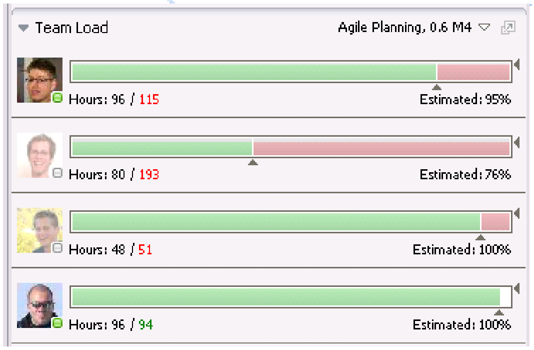
All Plan Items (105)
Closed created by me (362)
Closed subscribed by me (1899)
Consumability & Polish (155)
Darin (50)
Darin Verify (19)
Eclipse Way Content (1)
James Verify (21)
Jared (11)
Jared Verify (12)

Open Work Items by Priority

New Work Items by Severity


Summary

Subscribers (1)
Included in builds (29)
Related Artifacts (1)
Duplicates (1)
Duplicate Of (1)
Change Sets (5)

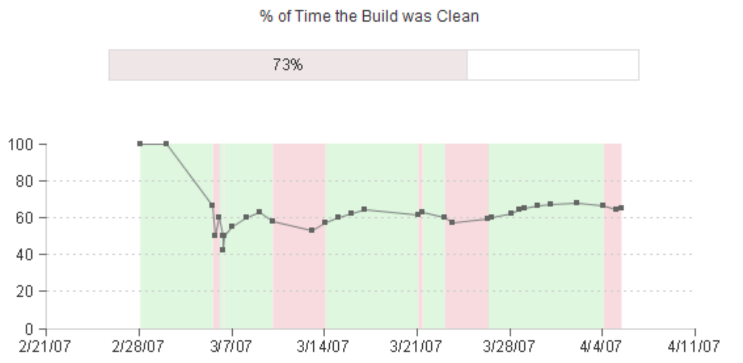

Team Load | Agile Planning, 0.6 M4

Hours: 96 / 115	Estimated: 95%
Hours: 80 / 193	Estimated: 76%
Hours: 48 / 51	Estimated: 100%
Hours: 96 / 94	Estimated: 100%

How long has it been since a clean build?

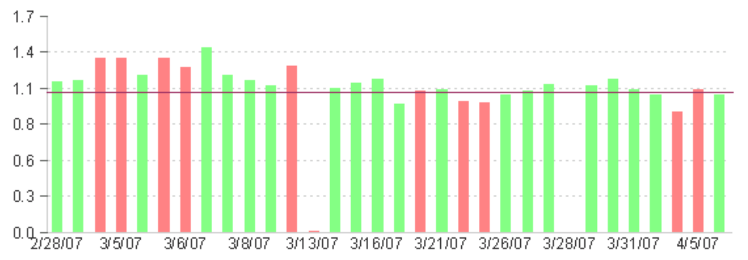
Latest Build is Clean (Apr 5, 2007 10:13:58 AM)

65% of builds (21 of 32) were clean in this interval

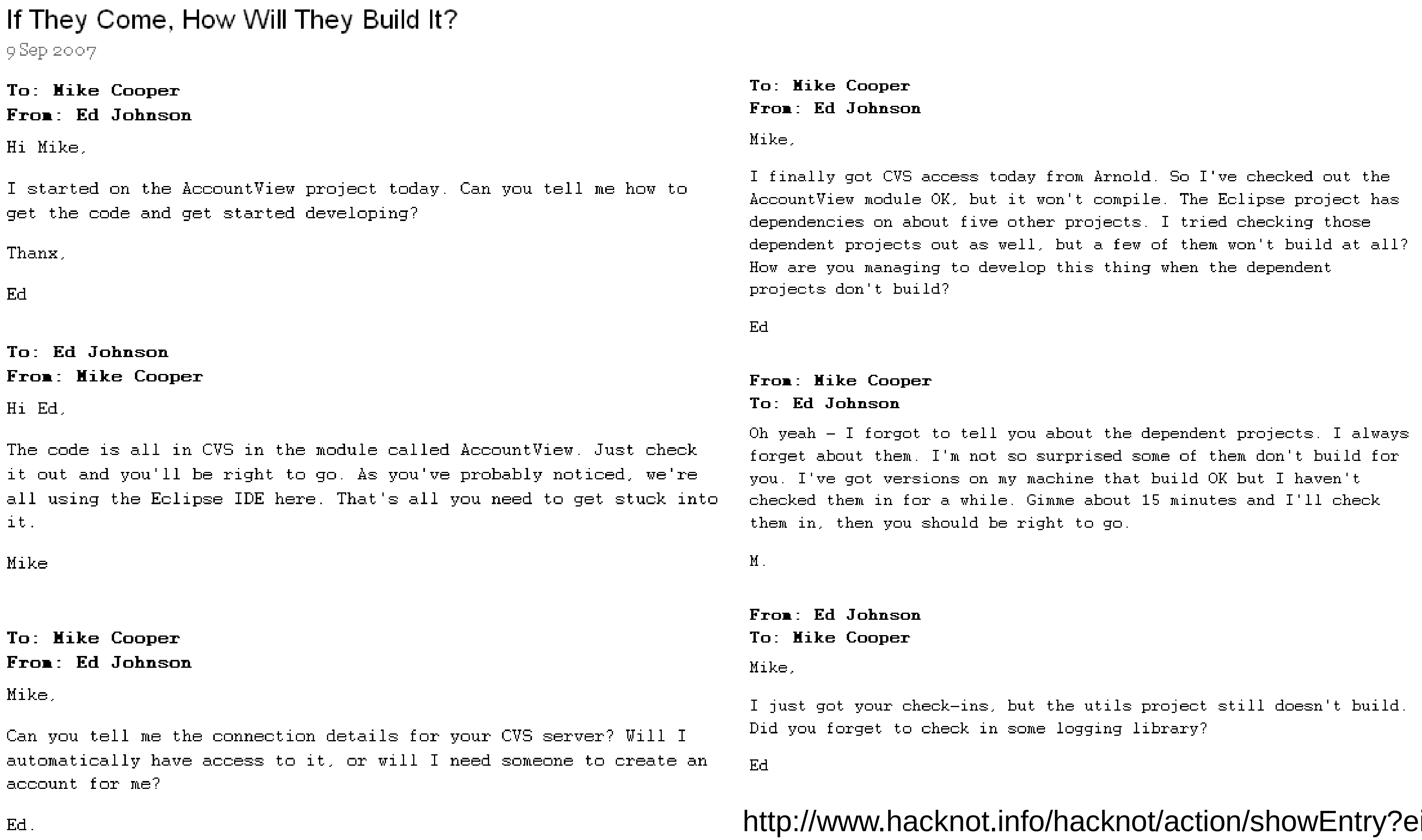


How Long does the Build Take?

Average Build Time is 1.08 hours



Joining a team episode



If They Come, How Will They Build It?

9 Sep 2007

To: Mike Cooper
From: Ed Johnson

Hi Mike,

I started on the AccountView project today. Can you tell me how to get the code and get started developing?

Thanx,

Ed

To: Ed Johnson
From: Mike Cooper

Hi Ed,

The code is all in CVS in the module called AccountView. Just check it out and you'll be right to go. As you've probably noticed, we're all using the Eclipse IDE here. That's all you need to get stuck into it.

Mike

To: Mike Cooper
From: Ed Johnson

Mike,

Can you tell me the connection details for your CVS server? Will I automatically have access to it, or will I need someone to create an account for me?

Ed.

To: Mike Cooper
From: Ed Johnson

Mike,

I finally got CVS access today from Arnold. So I've checked out the AccountView module OK, but it won't compile. The Eclipse project has dependencies on about five other projects. I tried checking those dependent projects out as well, but a few of them won't build at all? How are you managing to develop this thing when the dependent projects don't build?

Ed

From: Mike Cooper
To: Ed Johnson

Oh yeah - I forgot to tell you about the dependent projects. I always forget about them. I'm not so surprised some of them don't build for you. I've got versions on my machine that build OK but I haven't checked them in for a while. Gimme about 15 minutes and I'll check them in, then you should be right to go.

M.

From: Ed Johnson
To: Mike Cooper

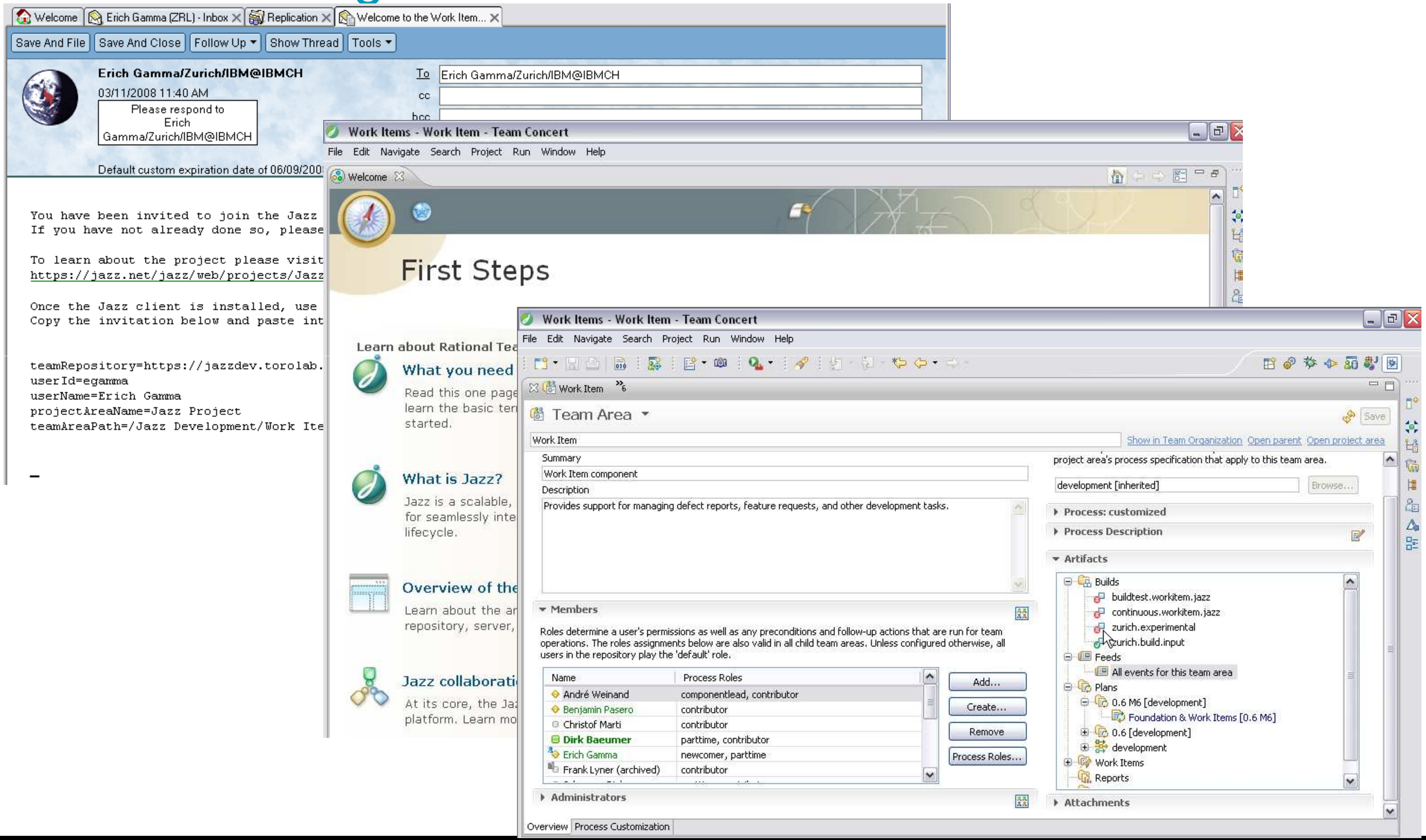
Mike,

I just got your check-ins, but the utils project still doesn't build. Did you forget to check in some logging library?

Ed

http://www.hacknot.info/hacknot/action/showEntry?eid=97

Demo: Joining a Team



Erich Gamma/Zurich/IBM@IBMCH
03/11/2008 11:40 AM
Please respond to Erich Gamma/Zurich/IBM@IBMCH

To: Erich Gamma/Zurich/IBM@IBMCH
cc
bcc

Default custom expiration date of 06/09/200

```
You have been invited to join the Jazz
If you have not already done so, please

To learn about the project please visit
https://jazz.net/jazz/web/projects/Jazz

Once the Jazz client is installed, use
Copy the invitation below and paste int


teamRepository=https://jazzdev.torolab.
userId=egamma
userName=Erich Gamma
projectAreaName=Jazz Project
teamAreaPath=/Jazz Development/Work Ite
```

Read this one page
learn the basic ter
started.

Jazz is a scalable,
for seamlessly inte
lifecycle.

Learn about the ar
repository, server,

At its core, the Jaz
platform. Learn mo

Work Item

Summary: Work Item component

Description: Provides support for managing defect reports, feature requests, and other development tasks.

project area's process specification that apply to this team area.

development [inherited]

Process: customized

Process Description

Artifacts

- Builds
  - buildtest.workitem.jazz
  - continuous.workitem.jazz
  - zurich.experimental
  - zurich.build.input
- Feeds
  - All events for this team area
- Plans
  - 0.6 M6 [development]
    - Foundation & Work Items [0.6 M6]
  - 0.6 [development]
  - development
- Work Items
- Reports

Members

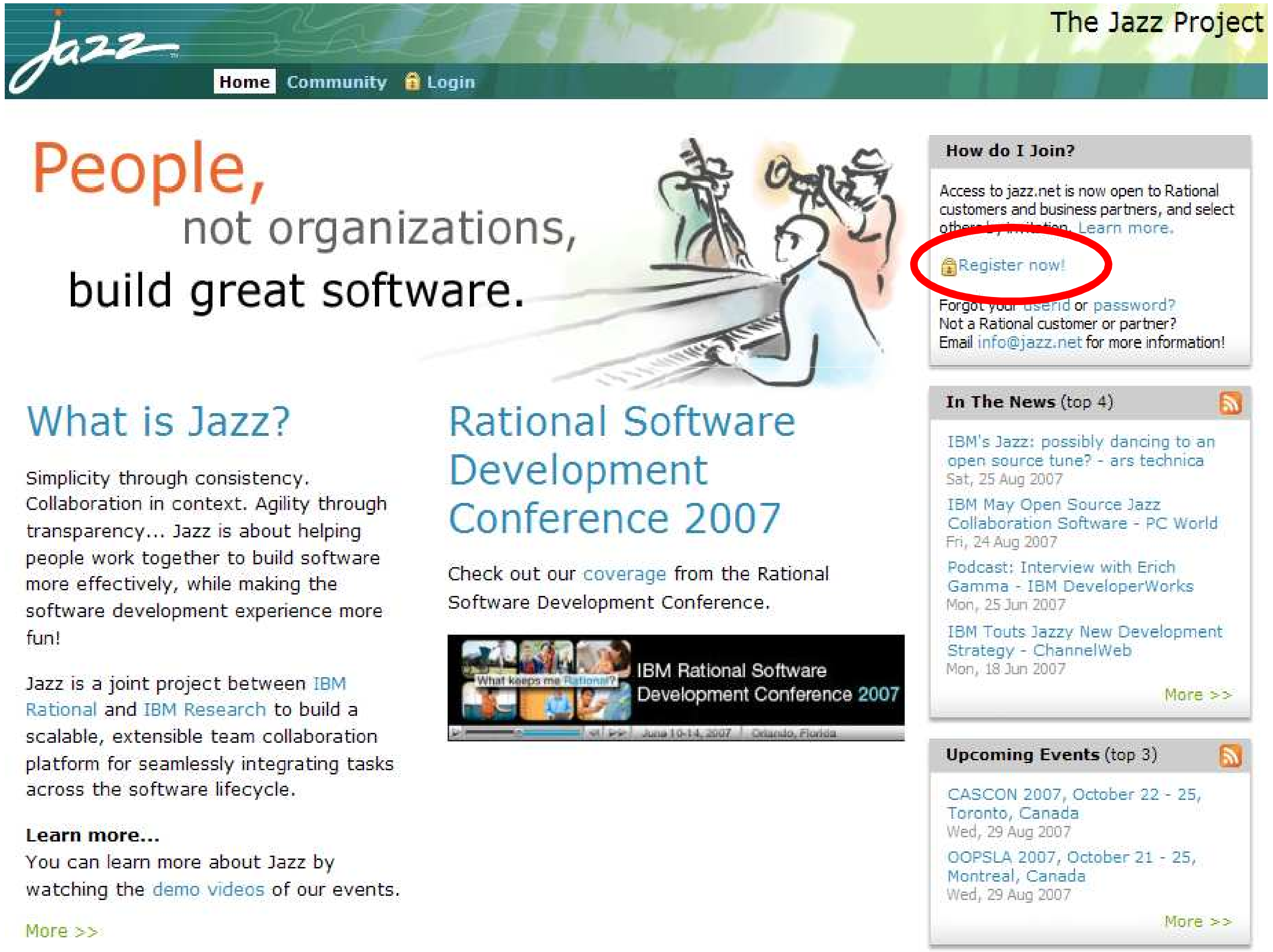
Roles determine a user's permissions as well as any preconditions and follow-up actions that are run for team operations. The roles assignments below are also valid in all child team areas. Unless configured otherwise, all users in the repository play the 'default' role.

Name	Process Roles
André Weinand	componentlead, contributor
Benjamin Pasero	contributor
Christof Marti	contributor
Dirk Baeumer	parttime, contributor
Erich Gamma	newcomer, parttime
Frank Lyner (archived)	contributor

Administrators

Attachments

Try it yourself on www.jazz.net



People,
not organizations,
build great software.

What is Jazz?

Simplicity through consistency. Collaboration in context. Agility through transparency... Jazz is about helping people work together to build software more effectively, while making the software development experience more fun!

Jazz is a joint project between IBM Rational and IBM Research to build a scalable, extensible team collaboration platform for seamlessly integrating tasks across the software lifecycle.

Learn more...
You can learn more about Jazz by watching the demo videos of our events.

More >>

Rational Software Development Conference 2007

Check out our coverage from the Rational Software Development Conference.

How do I Join?

Access to jazz.net is now open to Rational customers and business partners, and select others by invitation. Learn more.

Register now!

Forgot your userid or password?
Not a Rational customer or partner?
Email info@jazz.net for more information!

In The News (top 4)

IBM's Jazz: possibly dancing to an open source tune? - ars technica
Sat, 25 Aug 2007

IBM May Open Source Jazz Collaboration Software - PC World
Fri, 24 Aug 2007

Podcast: Interview with Erich Gamma - IBM DeveloperWorks
Mon, 25 Jun 2007

IBM Touts Jazzy New Development Strategy - ChannelWeb
Mon, 18 Jun 2007

More >>

Upcoming Events (top 3)

CASCON 2007, October 22 - 25, Toronto, Canada
Wed, 29 Aug 2007

OOPSLA 2007, October 21 - 25, Montreal, Canada
Wed, 29 Aug 2007

More >>

Thank You