eclipse CON 2005

the eclipse way

processes that adapt

Erich Gamma, John Wiegand
Eclipse Project Management Committee Members
IBM

© 2005 by International Business Machines; made available under the EPL v1.0 | February 28, 2005

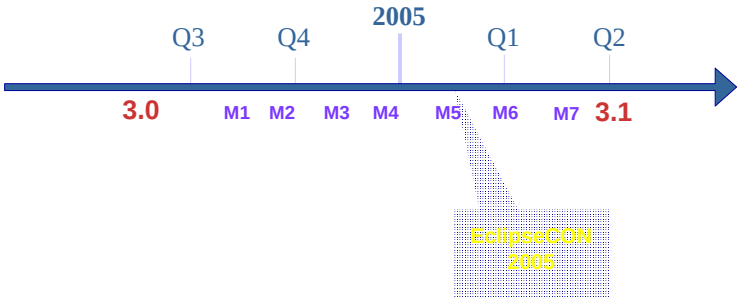
eclipse CON 2005

outline

- on the way to 3.1
- the Eclipse way

eclipse CON 2005

on the way to 3.1…



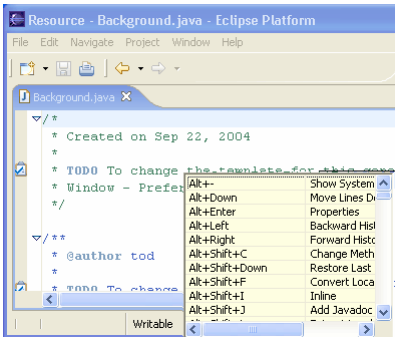
eclipse CON 2005

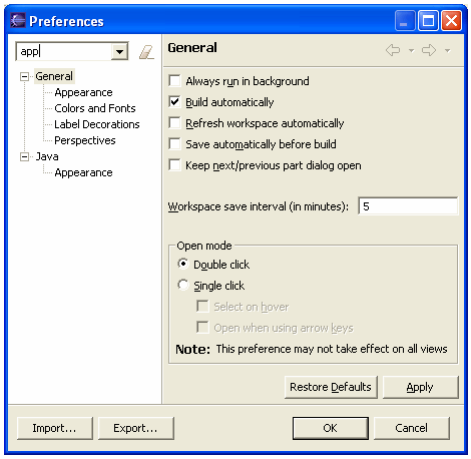
3.1 themes

- scaling up
- enterprise ready
- design for extensibility: be a better platform
- simple to use
- rich client platform improvements
- J2SE 5 support
- broadening the community

eclipse CON 2005

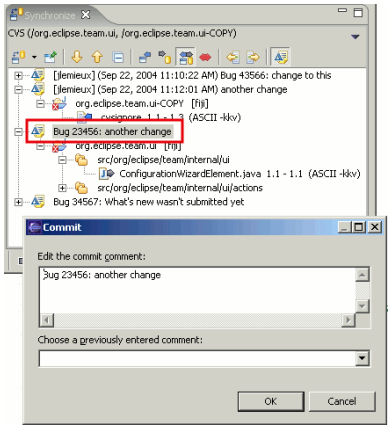
UI

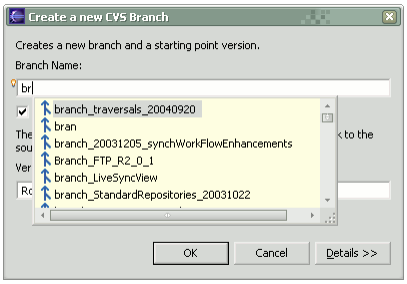




eclipse CON 2005

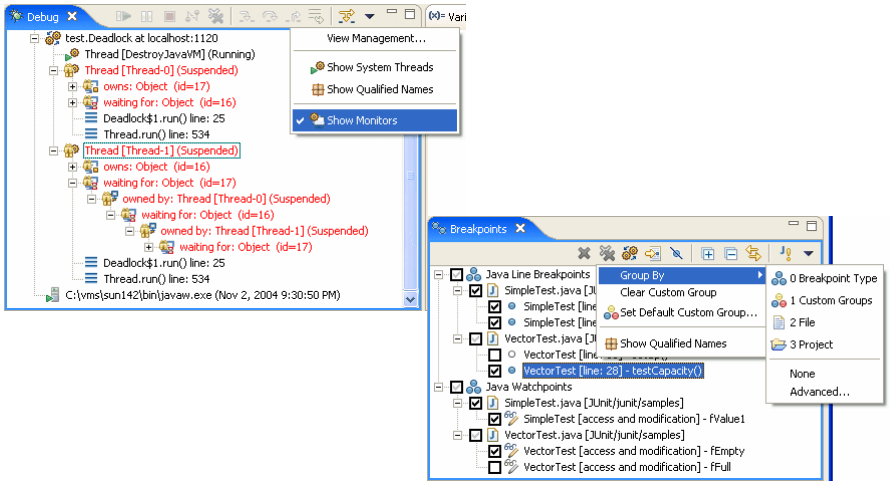
CVS





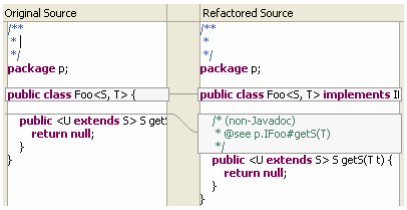
eclipse CON 2005

Debug



eclipse CON 2005

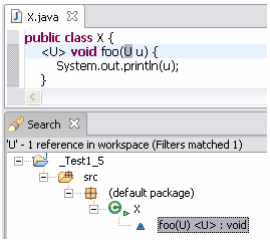
J2SE 5

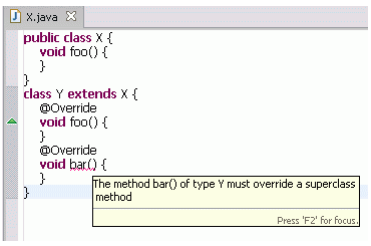



```
enum TestStyle { CORRECTNESS, PERFORMANCE }

@Target(METHOD)
@interface Test {
    TestStyle style() default CORRECTNESS;
}

@Test(style=PERFORMANCE)
public void randomAccess() {
    List<E> vector= fFull;
    long before= System.currentTimeMillis();
    E one= vector.get(42);
    long after= System.currentTimeMillis();
    Long maxAccessTime= new Long(100);
    assertTrue(after - before < maxAccessTime);
}
```





eclipse CON 2005

JDT Quick Fixes

```
public void conditionsCheck(boolean a, boolean b) {
    if ((!a) && b) {
    }
}
```

- Add else block
- Inverse conditions
- Remove extra parenthesis
- Split && condition

```
...
public void conditionsCheck(boolean a, boolean b) {
if ((a) || !b) {
//then-block
...
```

```
if (element instanceof IAdaptable) {
```

- Introduce new local with casted type

```
public void conditionsCheck(boolean a, boolean b) {
    if (a || b) {
        //then
    } else {
        //else
    }
```

- Inverse 'if' statement
- Rename in file

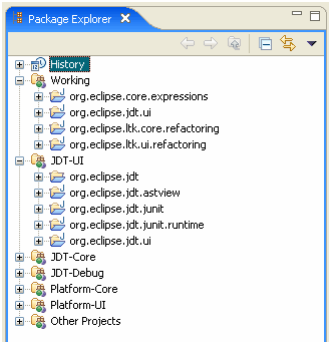
```
...
public void conditionsCheck(boolean a, boolean b
if (! a && ! b) {
//else-block
} else {
//then-block
}
...
```

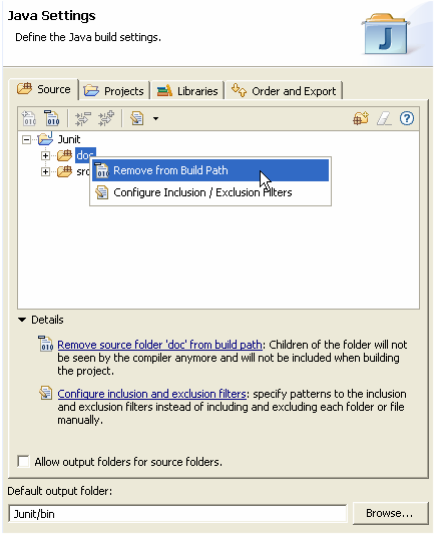
```
String[] names= {"Calvin", "Hobbes"};
for (int i= 0; i < names.length; i++) {
    String name= names[i];
    System.out.println(name);
}
```

- Convert to enhanced for loop
- Remove surrounding 'for' statement

eclipse CON 2005

JDT UI





eclipse CON 2005

JDT NLS support

```
alid_range=Invalid range argument
alid_visible_region_1=Invalidx visible region argument
```

Change to 'Invalid'
Change to 'Invalids'
Always ignore 'Invalidx'
Add 'Invalidx' to dictionary

```
String s;
s= Messages.getString("test");
s= Messages.getString("t
s= Messages.getString("t
s= Messages.getString("s
```

Externalized Value:
Hello World
Press 'F2' for focus.

```
TextViewer.error.bad_location.shift_2= TextViewer.shif
TextViewer.error.bad_location.verifyText= TextViewer.v
TextViewer.error.invalid_range= Invalid range argument
TextViewer.error.invalid_visible_region_1= Invalid vis
TextViewer.error.invalid_visible_region_2= Invalid vis
```


eclipse CON 2005

Ant

```
<target name="init">
    <echo>start</echo>
</target>

<!-- ================================
     target: base
     ================================ -->
<target name="default" depends="init" descript
    <javac srcdir="src"
     destdir="bin"
     classpath="test.jar"
```

Debug
test build.xml [Ant Build]
c:\Documents and Settings\luc\Desktop\target\test\build.xml
Thread [Ant Build] (Suspended (breakpoint at line 4 in build.xml))
build: echo line: 4
C:\Program Files\Java\jre1.5.0\bin\javaw.exe (Dec 14, 2004 10:53:11 PM)

build.xml

```
<project default="build">
  <target name="build">
    <echo>begin</echo>
    <echo>end</echo>
  </target>
</project>
```

```
<target name="default" depends="init" description="-
    <javac srcdir="src"
     destdir="bin"
     classpath="tes
     debug="on"
  />
  </target>

</project>
```

Javac Task - Microsoft Internet Explore
File Edit View Favorites Tools Help
Address http://ant.apache.org/manual/CoreTasl

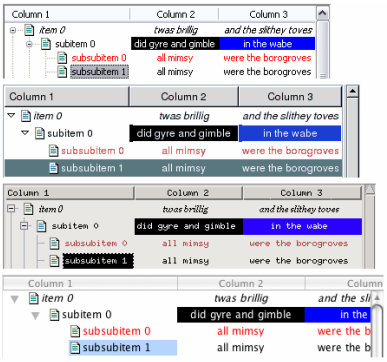
Javac

Description

Compiles a Java source tree.

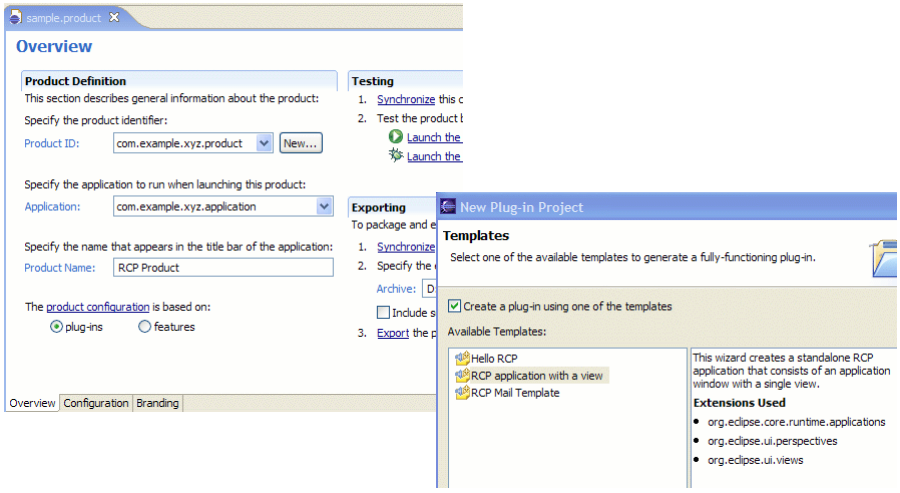
eclipse CON 2005

SWT



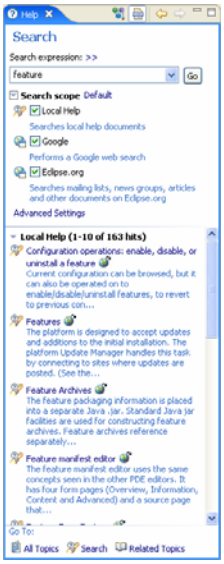
eclipse CON 2005

PDE/RCP



eclipse CON 2005

user assistance



eclipse CON 2005

outline

- on the way to 3.1
- the Eclipse way

eclipse CON 2005

why “the eclipse way”?

- our process has evolved over time
  - stay aware, adapt, change
- our process has helped us (the platform) to achieve:
  - predictability
  - quality delivery on time
- share what we have learned
  - influence “by example” – similar as JDT

eclipse CON 2005

“I've been following the development of Eclipse for some time and I'm continually amazed as the extended team hits each projected milestone and ship date with precision. Until this experience, I was convinced that such a feat was impossible for a software project of any size and complexity.

So... the question is how is this really accomplished? And the immediate follow-up of course is, how can this process be replicated elsewhere?

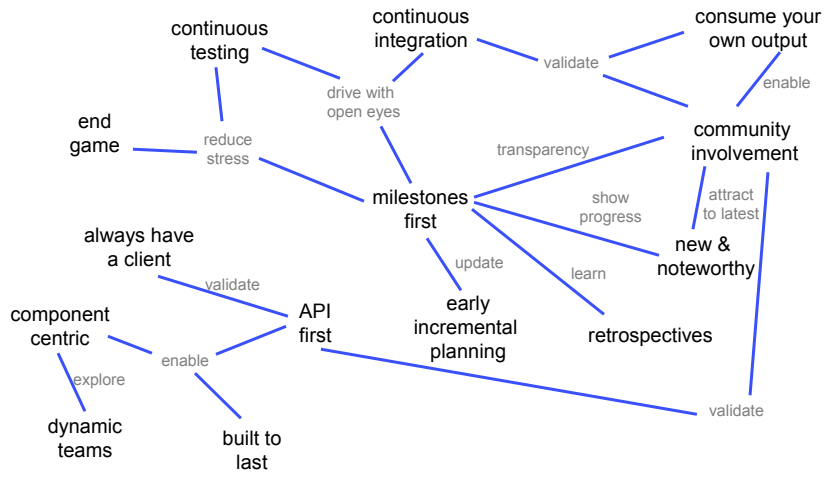
- a post to the eclipse developer’s mailing list

The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

eclipse practices

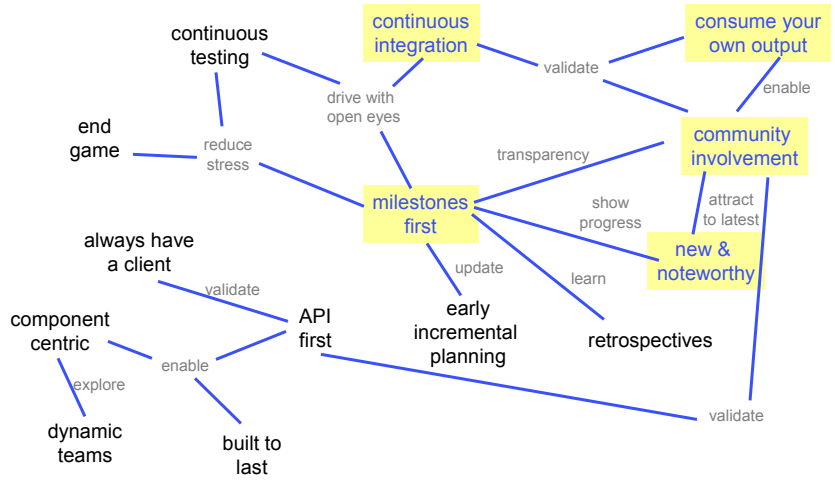
it's all about feedback!



The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

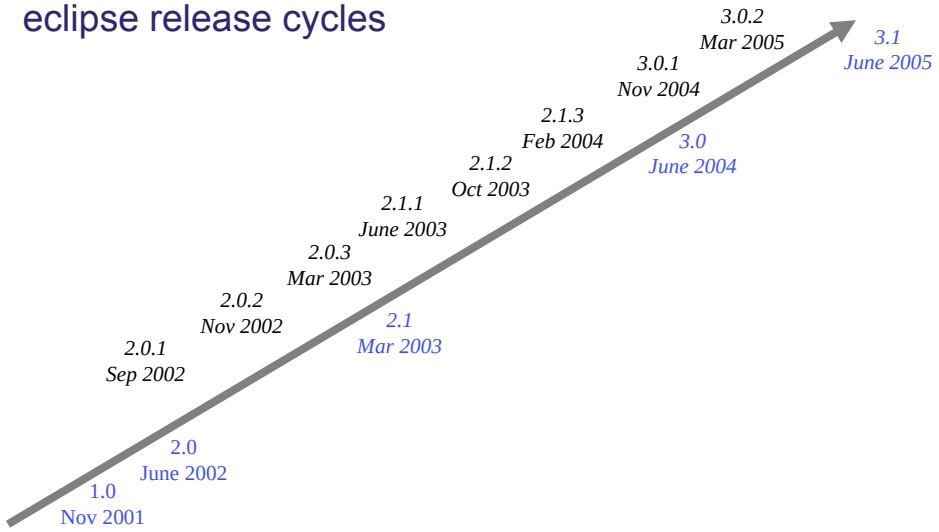
project rhythm



The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

eclipse release cycles



The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

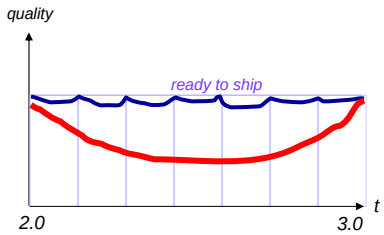
eclipse CON 2005

milestones first

- break down release cycle into milestones
  - we currently use 6 weeks
- each milestone is a miniature development cycle
  - plan, execute, test, retrospective
  - teams refer to the release plan when creating milestone plans
    - assign plan items to a milestone
    - milestone plans are public
- result of a milestone
  - milestone builds: good enough to be used by the community
- at the end of each milestone we do a retrospective
  - what went well, what did not?

➢ milestones reduce stress!

- before/after



The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

continuous integration

- fully automated build process
- build quality verified by automatic unit tests
- staged builds
  - nightly builds
    - 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
- reality: build failures occur
  - rebuild to create acceptable integration, milestone builds.

© 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

always beta

- each build is a release candidate; we expect it to work
- results of the build process and the automatic tests
  - indicate where we are
- as tool makers we use our own tools
  - component team – use the latest code daily
  - project team – use integration builds (weekly)
  - community – use milestone builds

➢ continuously *Consume Our Own Output*

© 2005 by International Business Machines; made available under the EPL v1.0

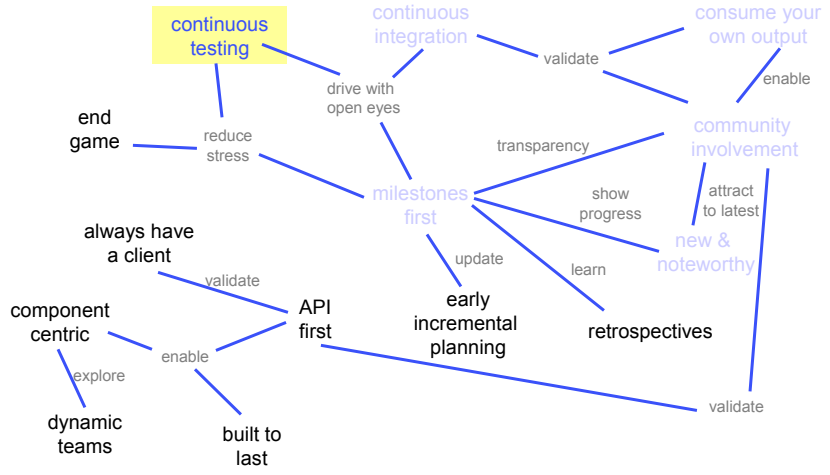
eclipse CON 2005

community involvement

- requires transparency
  - community needs to know what is going on to participate
- requires open participation
  - We value the contributions of the community
- we are the community
- *problem*: no one knew what was in a milestone,
  - so there was no incentive to move to milestone builds
  - so we received minimal feedback
    - more stale defect reports
  - quality suffered
- *solution*: publish new and noteworthy
  - advertise what we have been doing

eclipse CON 2005

testing

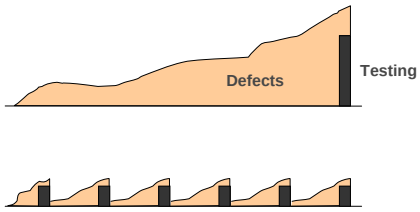


The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

testing

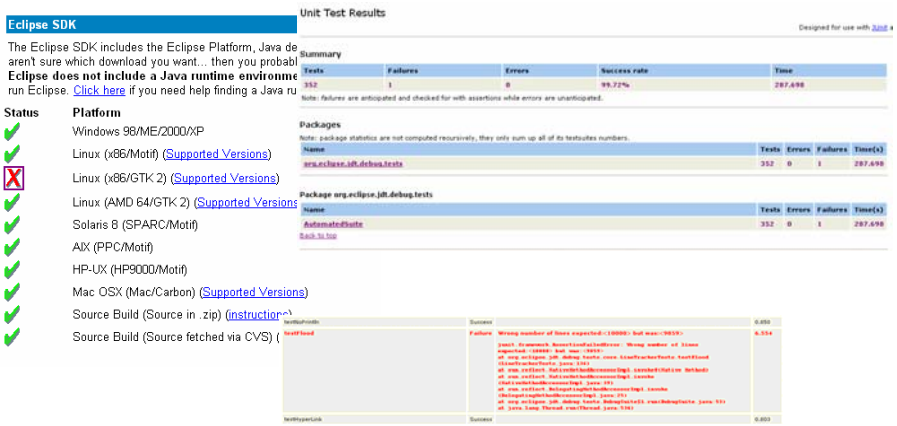
- innovate with confidence
  - continuous incremental design
- > 20,000 JUnit tests
  - tightly integrated into the build process
    - tests run after each build (nightly, integration, milestone)
  - integration builds are only green when all tests pass
- test kinds
  - **correctness** tests
    - assert 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

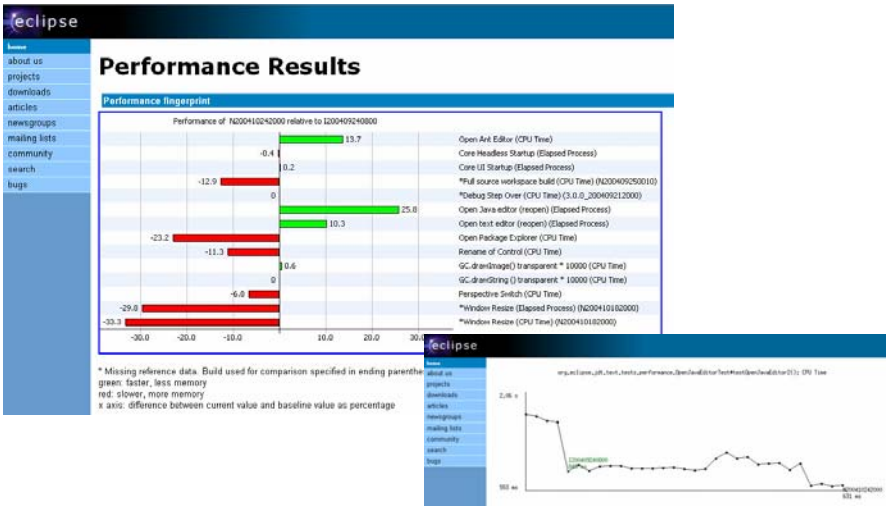
eclipse CON 2005

test report



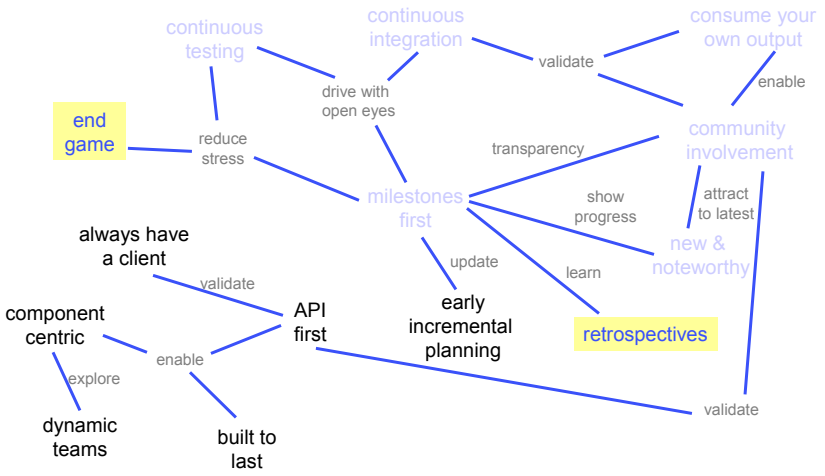
eclipse CON 2005

performance test report



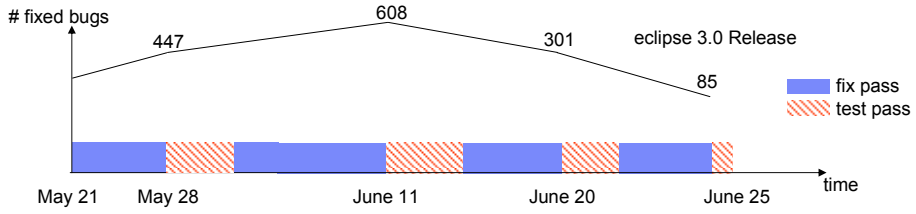
eclipse CON 2005

finishing



endgame

- convergence process applied before release
  - sequence of test-fix passes
    - community event
  - 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



The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

endgame

- endgame endurance
  - we are only effective for so long
  - distribute Quality/Polish effort throughout the release
- shared responsibility and commitment
  - we all Sign off

The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

[platform-releng-dev] PDE status for I-20040625-1208

- "An enthusiastic GO from PDE" - Cherie
- "The best build of the year!" - Dejan
- "An Eclipse build that the whole family can enjoy" - Wassim

- Your PDE team.

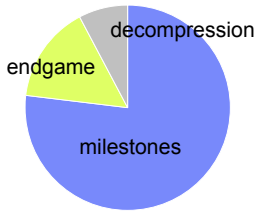
decompression

- recover from release
- retrospective of the last cycle
  - achievements
  - failures
  - process
  - cross-team collaboration
- start to plan the next release and cycle

eclipse CON 2005

where the time goes

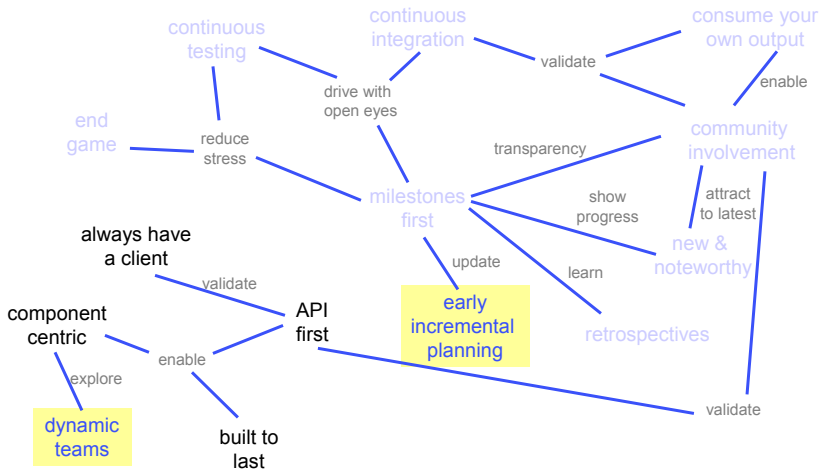
- release cycle 12-16 months
  - milestones - 9 months
  - endgame – 1-2 months
  - decompression – 1 month



The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

planning



The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

early planning

- release themes establish big picture
  - community input
  - requirements council new source of input
- component teams define component plans
- PMC collates initial project plan draft
  - tradeoff: requirements vs. available resources
  - Committed, Proposed, Deferred

The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

the plan is alive

- the project plan is updated quarterly to reflect
  - progress on items
  - new items
  - input from the community
- ➢ becomes final at the end of the release
- before: static plans
  - accurate once, but no early feedback: non-existent until late in the cycle.

The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

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)

© 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

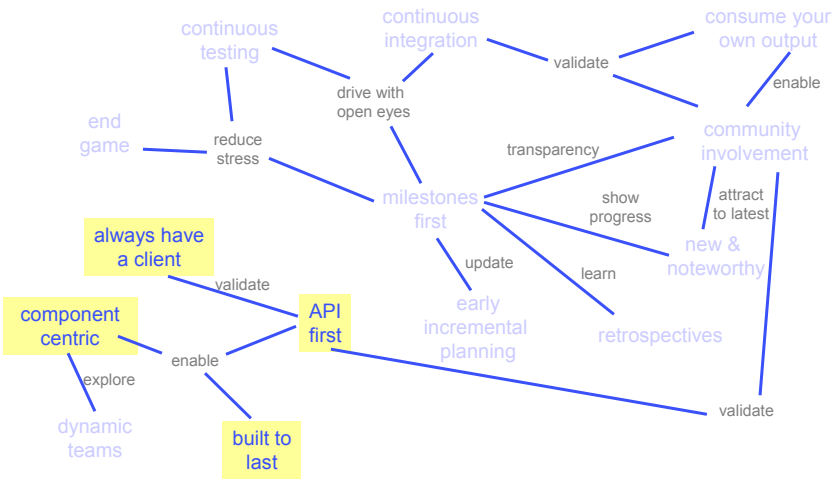
collective ownership

- PMC meets at least once a week
- all component leads and the PMC meet for a weekly planning call
  - 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

© 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

built to last



eclipse CON 2005

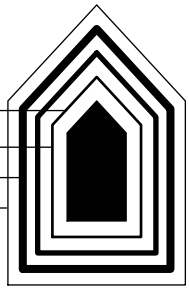
built to last

- deliver on time, every time
  - decisions in this release impact what we can do next release
  - must preserve architectural integrity
- deliver quality
  - innovation with continuity
  - need to have a solid foundation
    - scalable
    - performant
    - stable

eclipse CON 2005

how buildings "survive"

- **Stewart Brand: how buildings learn – what happens after they're built**
  - stuff: furniture
  - services: electrical, plumbing (7-15y)
  - structure: foundation, load bearing walls (30-300y)
  - site: geographical setting (forever)



Site

- **layers:**
  - evolve at different rates during the life of a building
  - shear against each other as they change at different rates
  - an adaptive building must allow slippage
  - ➢ a building that lasts is adaptive and can change over time

The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

structure foundation

- the eclipse plug-in architecture
- everything is a plug-in
  - simple and consistent

The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

services plumbing: APIs

- APIs matter
  - define consistent, concise API
  - don't expose the implementation
  - develop implementation and client at the same time
- define APIs for stability
  - binary compatibility is highest priority
  - we would rather provide less API than desired (and augment) than provide the wrong (or unnecessary) API and need to support it indefinitely

© 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

component centric

- component centric development
  - a team is responsible for one or more component
- dependencies through APIs
  - ensures high velocity development inside a component
  - eclipse 3.1 provides tools support to check for API access violations
  - define producer/consumer relationships among components
    - tension among components is healthy for coming up with good component interfaces/APIs

© 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

APIs first

- APIs don't just happen; we need to design them
- specifications with precisely defined behavior
  - what you can assume (and what you cannot)
  - it works ≠ API compliant
- must have at least one client involved, preferably more
- need an API advocate
  - we all care about having sustainable APIs
  - need someone who lives and breathes APIs

© 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

practical extensibility

- extensible in ways that are known useful
  - needed by us, requested by others
- we don't provide hypothetical generality - we want to be extensible in ways that matter
  - don't over generalize

© 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

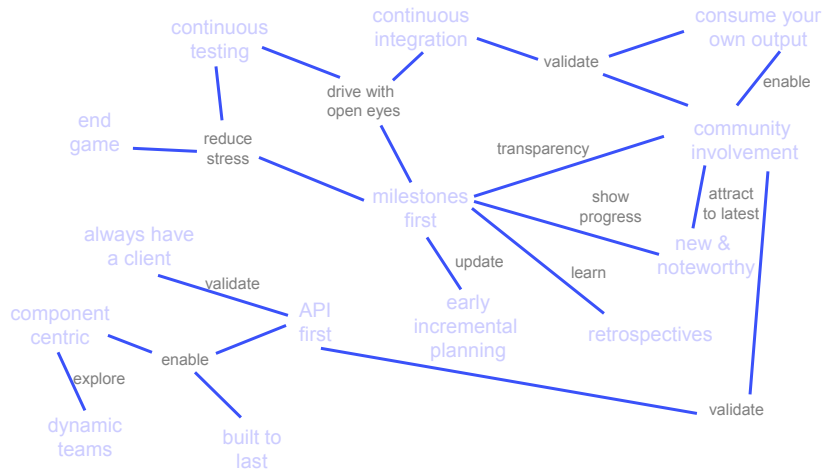
API tension

- conflict
  - APIs need to be implemented and used to be right
    - requires iteration
    - needs external client
  - stable API necessary
    - for widespread adoption
- resolution
  - don't commit API before its time
    - APIs can (and should) change during the release to accommodate new requirements, experience

The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

summary



The Eclipse Way | © 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

confessions

- ripples - a change in a lower layer that propagates up
  - we often underestimate the impact on upper layers!
    - consequence: end game swirls and stress
  - ground rule:
    - you are not done unless the upper most layer has digested the ripple
- dynamic teams – not all efforts are successful
  - require leadership, feeling of responsibility
  - require even more detailed planning
- we drop features!
  - but we hate to drop committed items

© 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

summary: how extreme is eclipse?

- ✓ *testing early, often and automated*
- ✓ *incremental design*
  - Yes but API stability is important to us
- ✓ *daily deployment*
- ✓ *customer involvement*
  - even better we have an active community
- ✓ *continuous integration*
  - nightly, weekly, milestone
- ✓ *short development cycles*
  - 6 week cycles
- ✓ *incremental planning*
  - start with embryonic plan
  - refine after each milestone

© 2005 by International Business Machines; made available under the EPL v1.0

eclipse CON 2005

conclusion

- the team makes the process work
- the team defines and evolves the process

© 2005 by International Business Machines; made available under the EPL v1.0