Tracking Problems

Andreas Zeller

1

What’s a problem?

- A *problem* is a questionable property of a program run
- It becomes a *failure* if it’s incorrect…
- …a *request for enhancement* if missing…
- …and a *feature* if normal behavior.

It’s not a bug, it’s a feature!

2

Problem Life Cycle

- The user *informs* the vendor about some problem.
- The vendor
  1. *reproduces* the problem
  2. *isolates* the circumstances
  3. *locates* and *fixes* the defect
  4. *delivers* the fix to the user.

3

Vendor Challenges

- How do I organize the life cycle?
- Which problems are currently open?
- Which are the most severe problems?
- Did similar problems occur in the past?

4

4

User Challenges

Solve my problem!

5

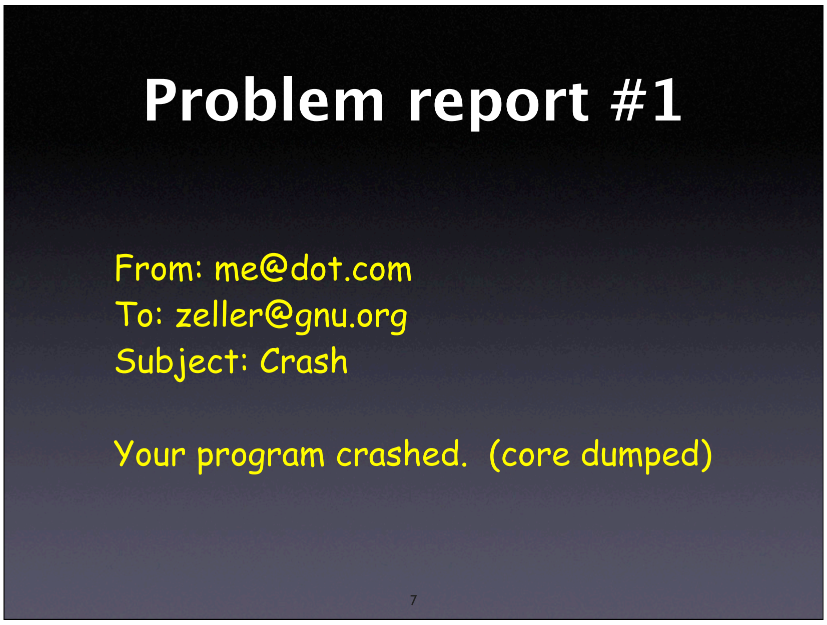
5

Problem Report

- A problem comes to life with a *problem report.*
- A problem report includes all the information the vendor needs to fix the problem.
- Also known as *change request* or *bug report.*

6

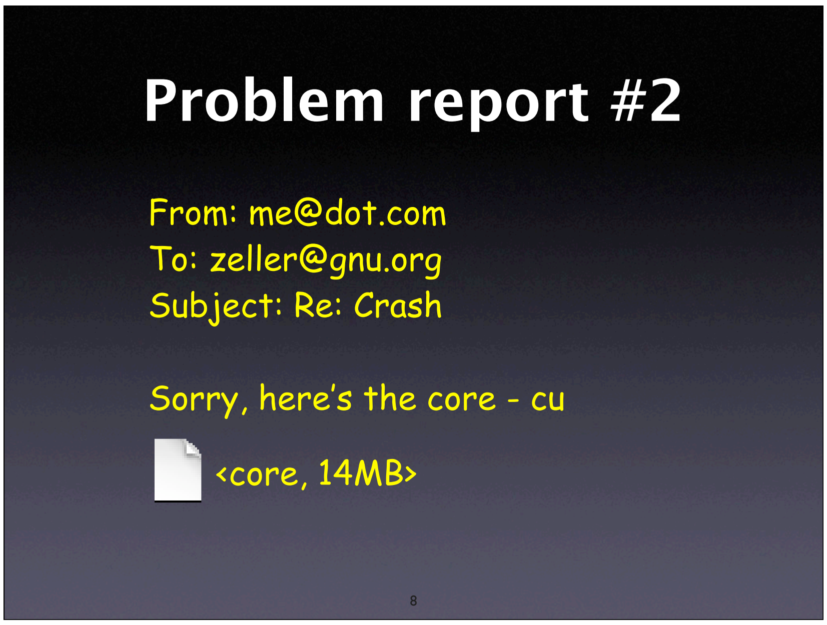
6



Problem report #1

From: me@dot.com
To: zeller@gnu.org
Subject: Crash

Your program crashed. (core dumped)

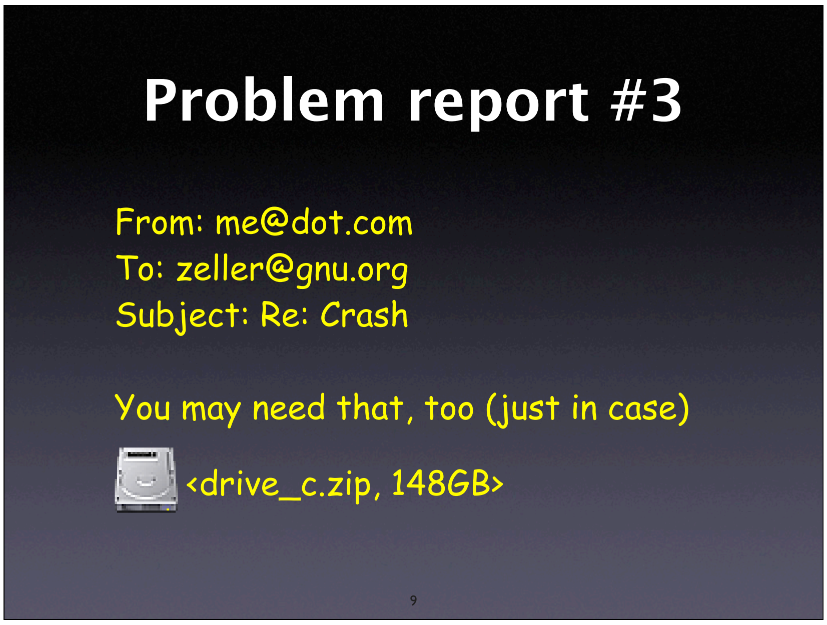


Problem report #2

From: me@dot.com
To: zeller@gnu.org
Subject: Re: Crash

Sorry, here's the core - cu

<core, 14MB>



Problem report #3

From: me@dot.com
To: zeller@gnu.org
Subject: Re: Crash

You may need that, too (just in case)

<drive_c.zip, 148GB>

What to report

- The *product release*
- The *operating environment*
- The *problem history*
- *Expected* and *experienced behavior*
- *A* one-line *summary*

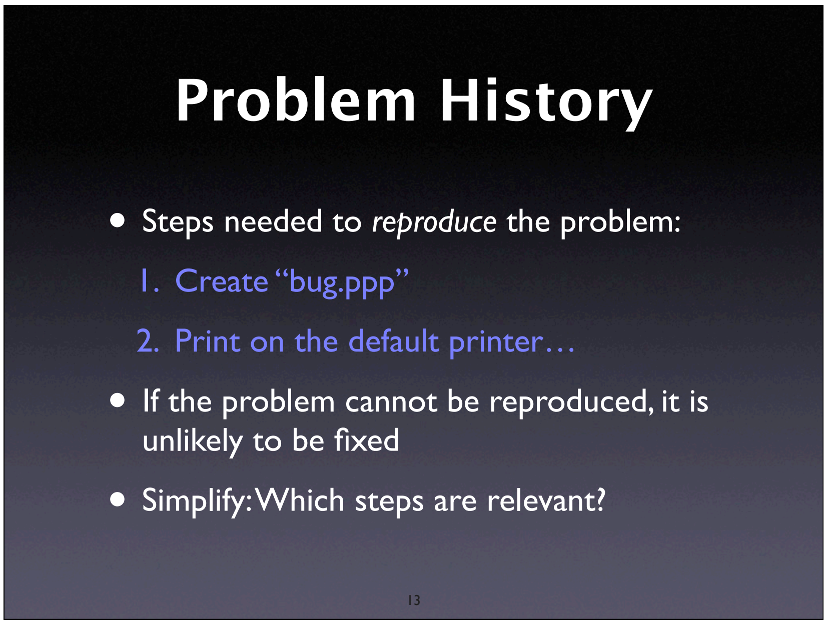
Product Release

- Typically, some *version number* or otherwise unique identifier
- Required to *reproduce the exact version:*

  Perfect Publishing Program 1.1 (Build 7E47)
- Generalize: Does the problem occur only in this release?

Operating Environment

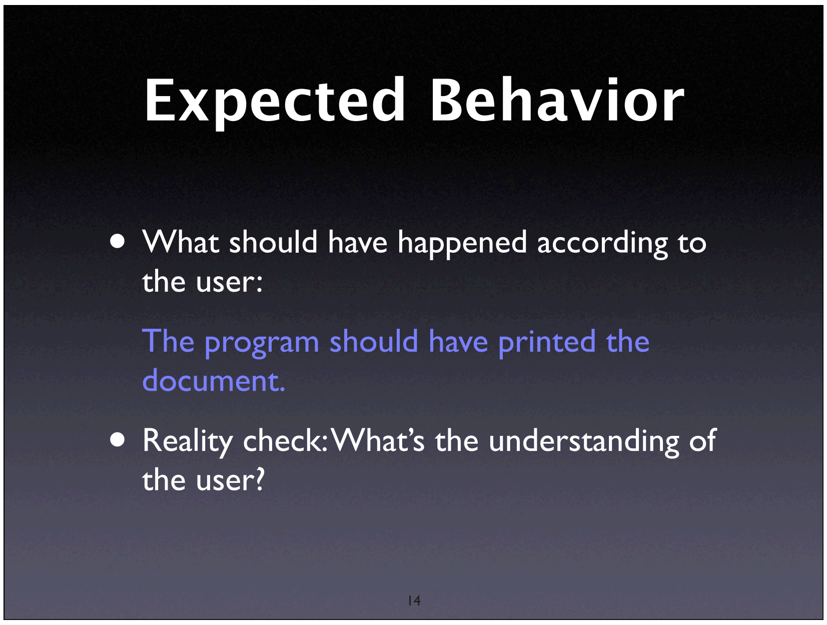
- Typically, *version information* about the operating system
- Can be simple ("Windows 98 SE") or complex ("Debian Linux 'Sarge' with the following packages…")
- Generalize: In which environments does the problem occur?



Problem History

- Steps needed to *reproduce* the problem:
  1. Create “bug.ppp”
  2. Print on the default printer…
- If the problem cannot be reproduced, it is unlikely to be fixed
- Simplify: Which steps are relevant?

13

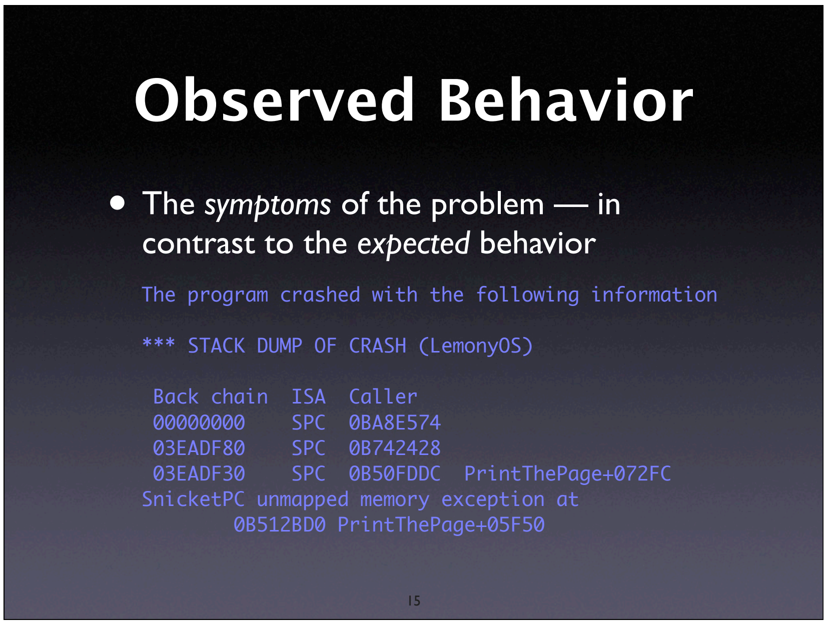


Expected Behavior

- What should have happened according to the user:

  The program should have printed the document.
- Reality check: What’s the understanding of the user?

14



Observed Behavior

- The *symptoms* of the problem — in contrast to the *expected* behavior

```
The program crashed with the following information

*** STACK DUMP OF CRASH (LemonyOS)

 Back chain  ISA  Caller
 00000000    SPC  0BA8E574
 03EADF80    SPC  0B742428
 03EADF30    SPC  0B50FDDC  PrintThePage+072FC
SnicketPC unmapped memory exception at
        0B512BD0 PrintThePage+05F50
```

15

A one-line summary

- Captures the essential of the problem

  PPP 1.1 crashes when printing

Things to avoid

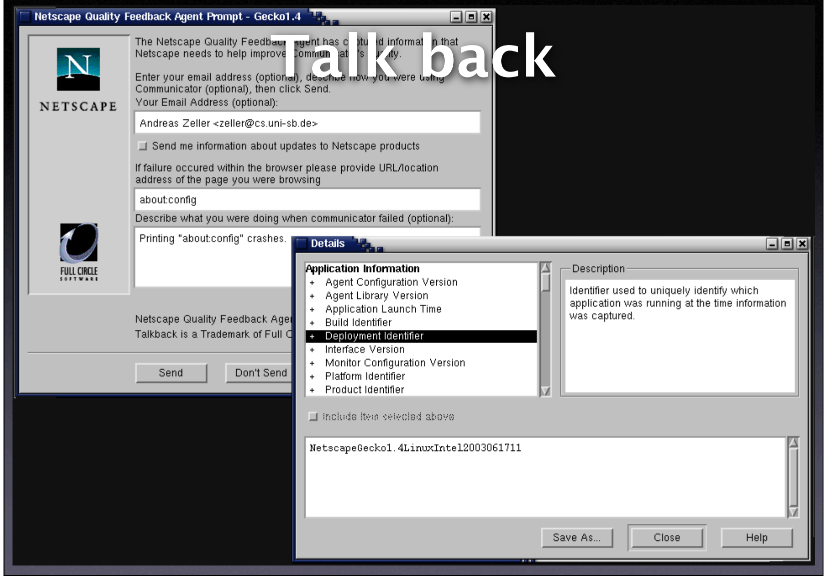
- Humor

  PPP (oops, gotta go to the restroom :-) …
- Sarcasm

  Here's yet another "never-to-be-fixed" bug
- Attacks

  If you weren't too incompetent to grasp…

Talk back



Netscape Quality Feedback Agent Prompt - Gecko1.4

The Netscape Quality Feedback Agent has c ptu d informat n that Netscape needs to help improve Commun at 's ity.

Enter your email address (optional), desc how you were using Communicator (optional), then click Send.

Your Email Address (optional):

Andreas Zeller <zeller@cs.uni-sb.de>

Send me information about updates to Netscape products

If failure occured within the browser please provide URL/location address of the page you were browsing

about:config

Describe what you were doing when communicator failed (optional):

Printing "about:config" crashes.

Netscape Quality Feedback Age

Talkback is a Trademark of Full C

Details

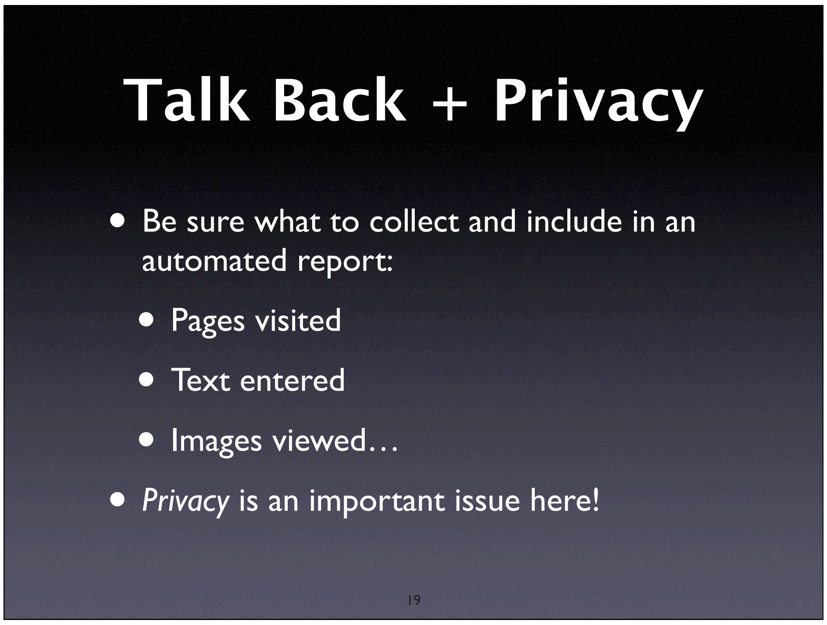
Application Information
- Agent Configuration Version
- Agent Library Version
- Application Launch Time
- Build Identifier
- Deployment Identifier
- Interface Version
- Monitor Configuration Version
- Platform Identifier
- Product Identifier

Description

Identifier used to uniquely identify which application was running at the time information was captured.

Include item selected above

```
NetscapeGecko1.4LinuxIntel2003061711
```

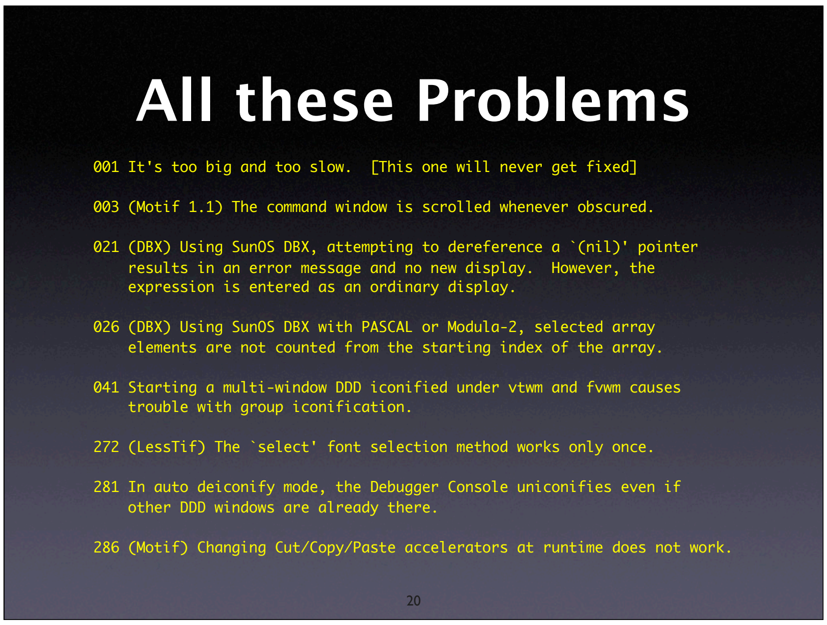


Talk Back + Privacy

- Be sure what to collect and include in an automated report:
  - Pages visited
  - Text entered
  - Images viewed…
- *Privacy* is an important issue here!

19

19

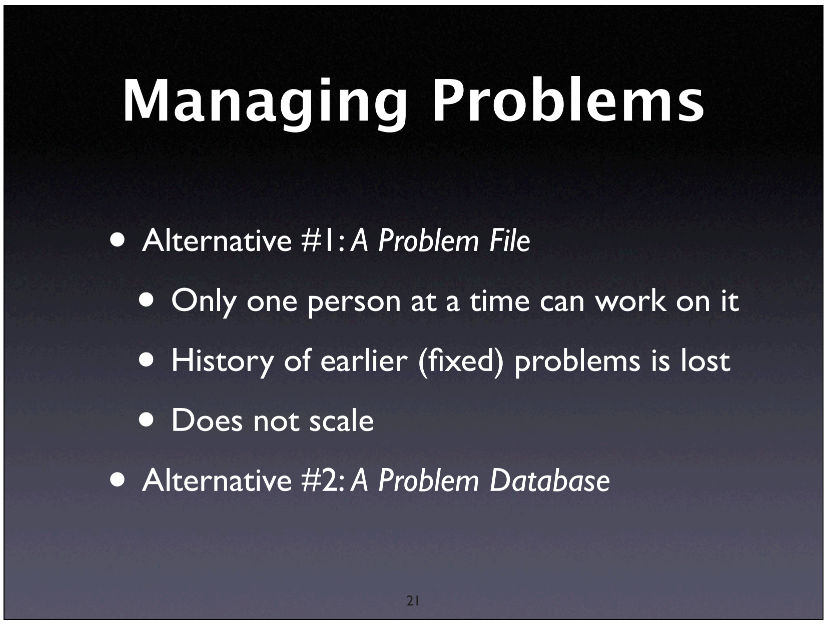


All these Problems

```
001 It's too big and too slow.  [This one will never get fixed]

003 (Motif 1.1) The command window is scrolled whenever obscured.

021 (DBX) Using SunOS DBX, attempting to dereference a `(nil)' pointer
    results in an error message and no new display.  However, the
    expression is entered as an ordinary display.

026 (DBX) Using SunOS DBX with PASCAL or Modula-2, selected array
    elements are not counted from the starting index of the array.

041 Starting a multi-window DDD iconified under vtwm and fvwm causes
    trouble with group iconification.

272 (LessTif) The `select' font selection method works only once.

281 In auto deiconify mode, the Debugger Console uniconifies even if
    other DDD windows are already there.

286 (Motif) Changing Cut/Copy/Paste accelerators at runtime does not work.
```

20

20



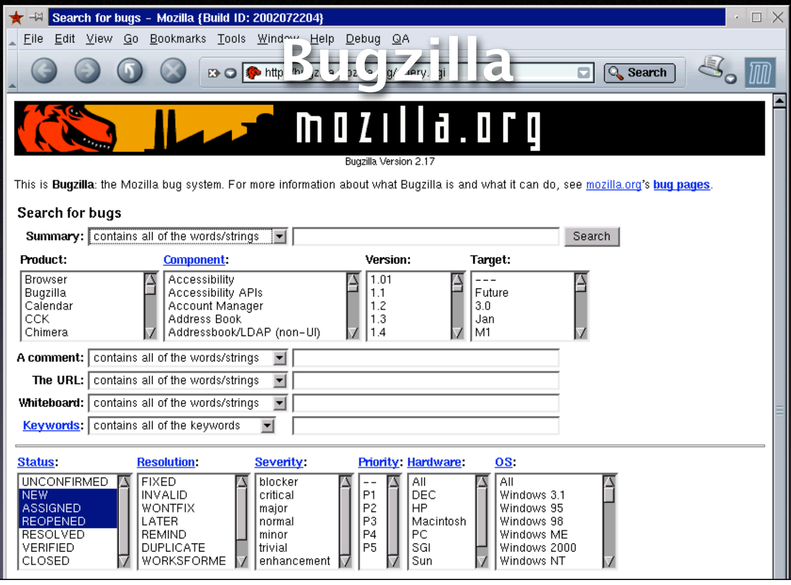
Managing Problems

- Alternative #1: *A Problem File*
  - Only one person at a time can work on it
  - History of earlier (fixed) problems is lost
  - Does not scale
- Alternative #2: *A Problem Database*

21

21

Bugzilla


Search for bugs - Mozilla {Build ID: 2002072204}
File Edit View Go Bookmarks Tools Window Help Debug QA
Search
mozilla.org

Bugzilla Version 2.17

This is **Bugzilla**: the Mozilla bug system. For more information about what Bugzilla is and what it can do, see mozilla.org's **bug pages**.

Search for bugs

Summary:	contains all of the words/strings		Search

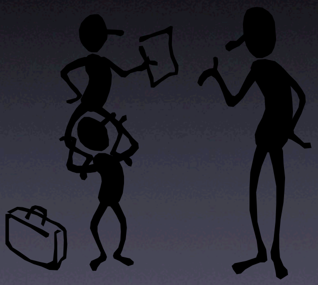
Product:	**Component:**	**Version:**	**Target:**
Browser	Accessibility	1.01	---
Bugzilla	Accessibility APIs	1.1	Future
Calendar	Account Manager	1.2	3.0
CCK	Address Book	1.3	Jan
Chimera	Addressbook/LDAP (non-UI)	1.4	M1

A comment:	contains all of the words/strings	
The URL:	contains all of the words/strings	
Whiteboard:	contains all of the words/strings	
Keywords:	contains all of the keywords	

Status:	**Resolution:**	**Severity:**	**Priority:**	**Hardware:**	**OS:**
UNCONFIRMED	FIXED	blocker	--	All	All
NEW	INVALID	critical	P1	DEC	Windows 3.1
ASSIGNED	WONTFIX	major	P2	HP	Windows 95
REOPENED	LATER	normal	P3	Macintosh	Windows 98
RESOLVED	REMIND	minor	P4	PC	Windows ME
VERIFIED	DUPLICATE	trivial	P5	SGI	Windows 2000
CLOSED	WORKSFORME	enhancement		Sun	Windows NT

Classifying Problems

- Severity
- Priority
- Identifier
- Comments
- Notification



Severity

Enhancement. A desired feature.

Trivial. Cosmetic problem.

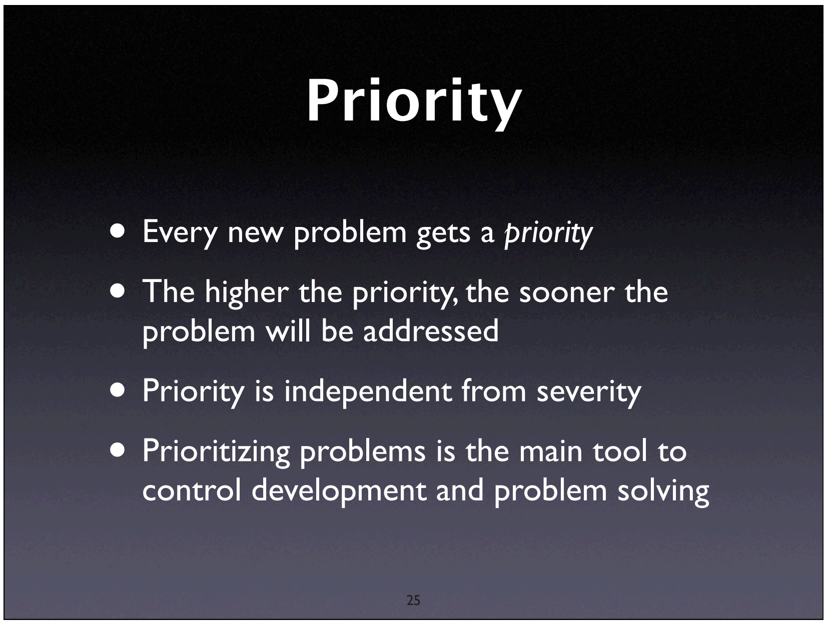
Minor. Problem with easy workaround.

Normal. "Standard" problem.

Major. Major loss of function.

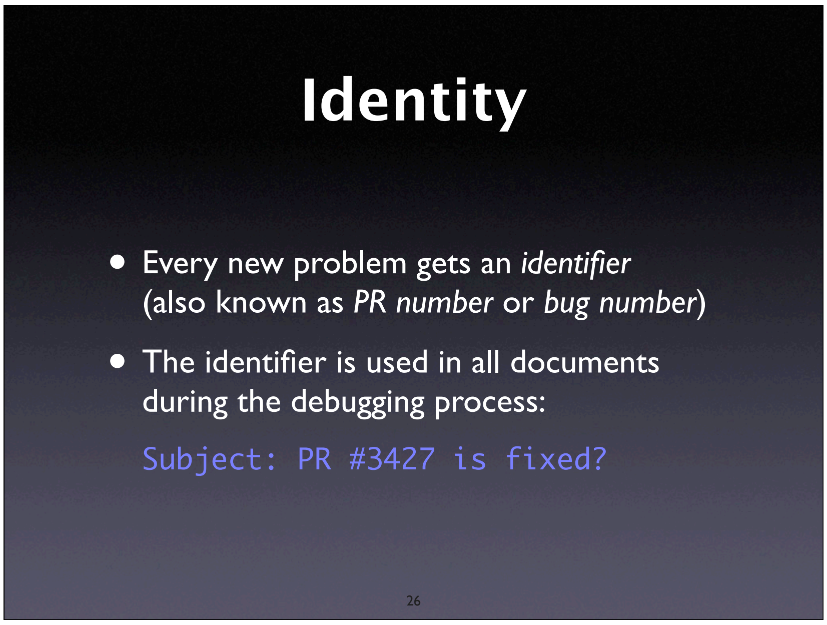
Critical. Crashes, loss of data or memory

Showstopper. Blocks development.



Priority

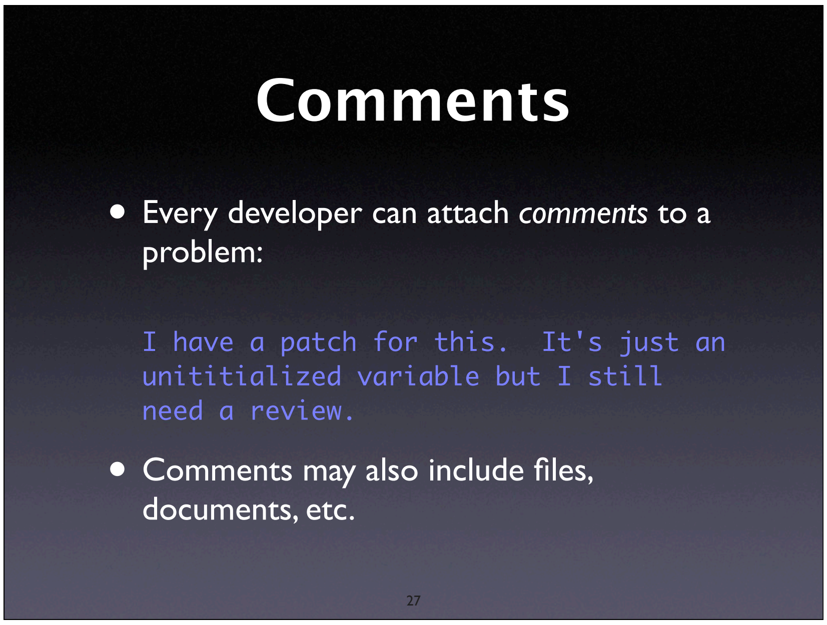
- Every new problem gets a *priority*
- The higher the priority, the sooner the problem will be addressed
- Priority is independent from severity
- Prioritizing problems is the main tool to control development and problem solving



Identity

- Every new problem gets an *identifier* (also known as *PR number* or *bug number*)
- The identifier is used in all documents during the debugging process:

```
Subject: PR #3427 is fixed?
```



Comments

- Every developer can attach *comments* to a problem:

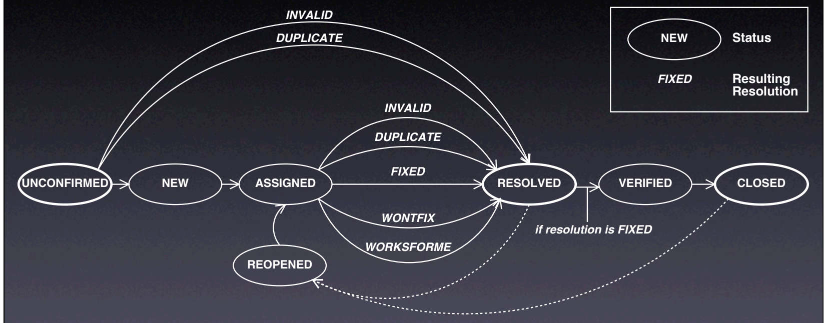
```
I have a patch for this.  It's just an
unititialized variable but I still
need a review.
```

- Comments may also include files, documents, etc.

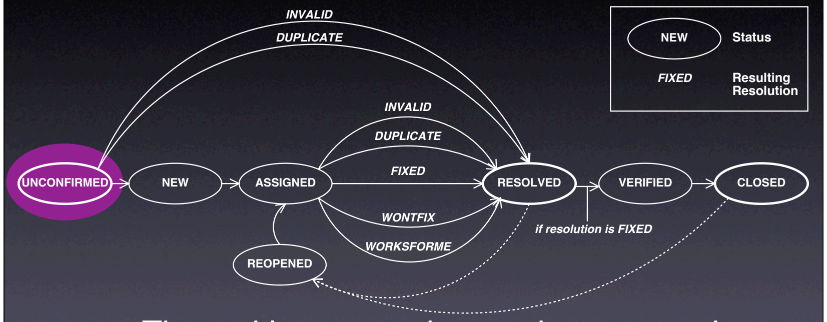
Notification

- Developers can attach an e-mail address to a problem report; they will be notified every time the report changes.
- Users can do so, too.

The Problem Lifecycle

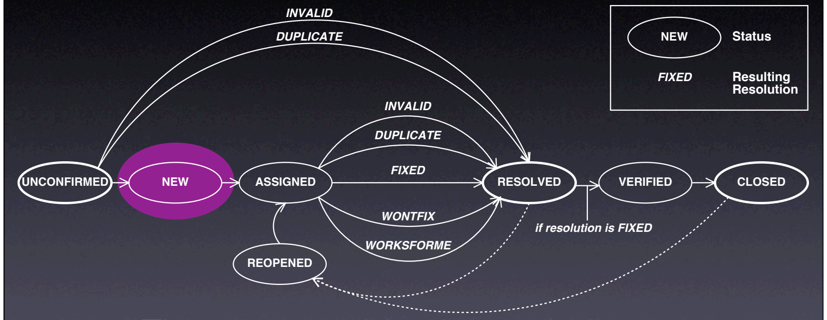


Unconfirmed Problem



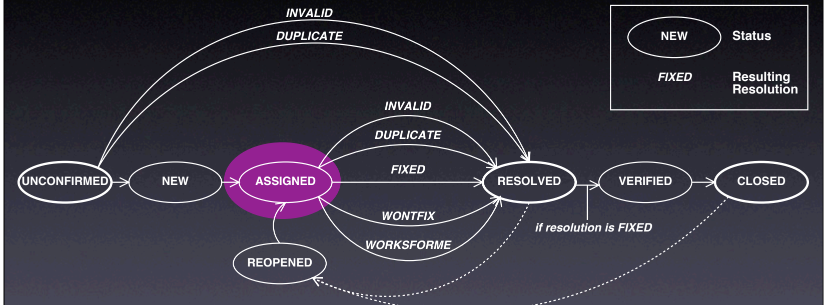
- The problem report has just been entered into the database

New Problem



- The report is *valid* and not a *duplicate*. (If not, it becomes *resolved*.)

Assigned Problem

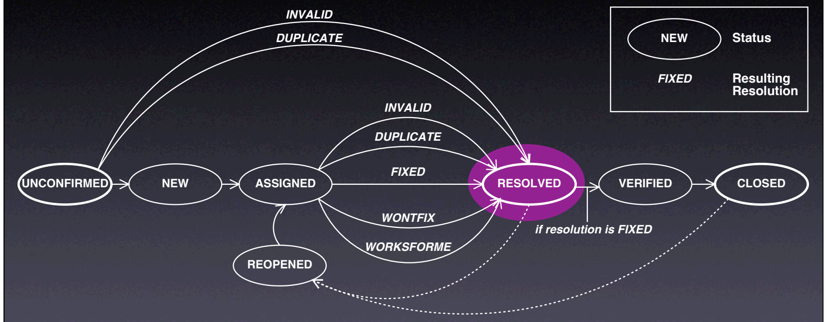


- The problem is assigned to a developer

Resolution

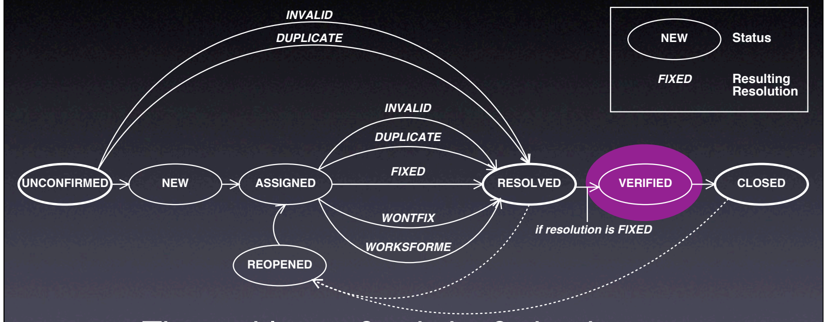
- FIXED: The problem is fixed.
- INVALID: The problem is not a problem.
- DUPLICATE: The problem already exists.
- WONTFIX: Will never be fixed (for instance, because the problem is a feature)
- WORKSFORME: Could not be reproduced.

Resolved Problem



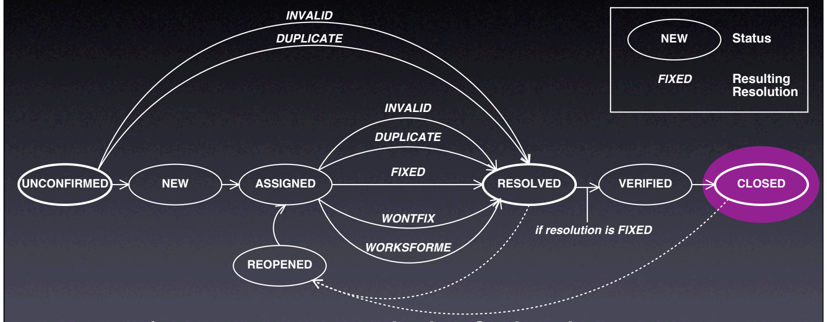
- The problem report has been processed.

Verified Problem



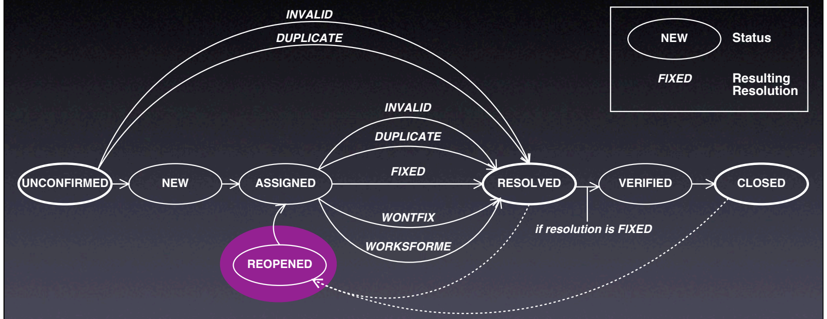
- The problem is fixed; the fix has been successful.

Closed Problem



- A new version with the fix has been released.

Reopened Problem



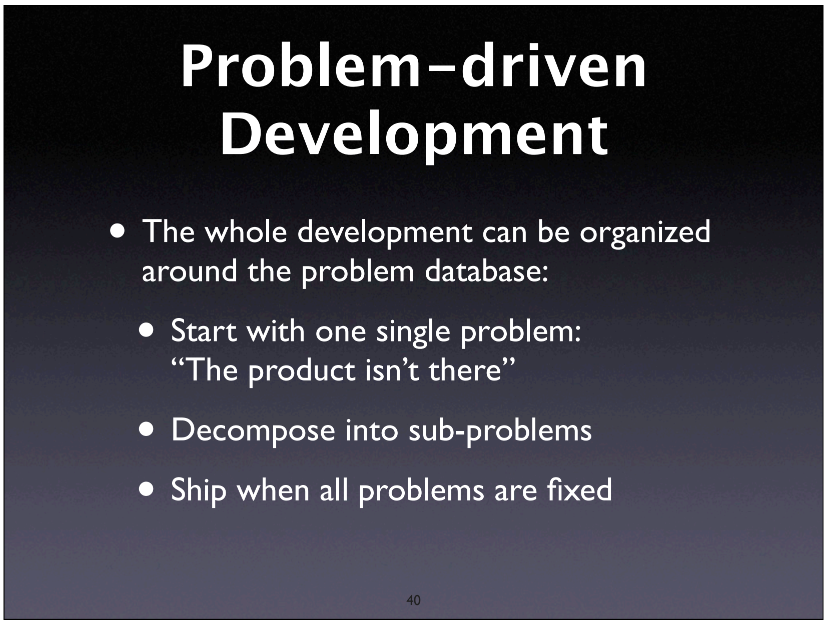
- Oops – there we go again :–(

Management

- Who *enters* problem reports?
- Who *classifies* problem reports?
- Who sets *priorities*?
- Who takes care of the problem?
- Who *closes* issues?

The SCCB

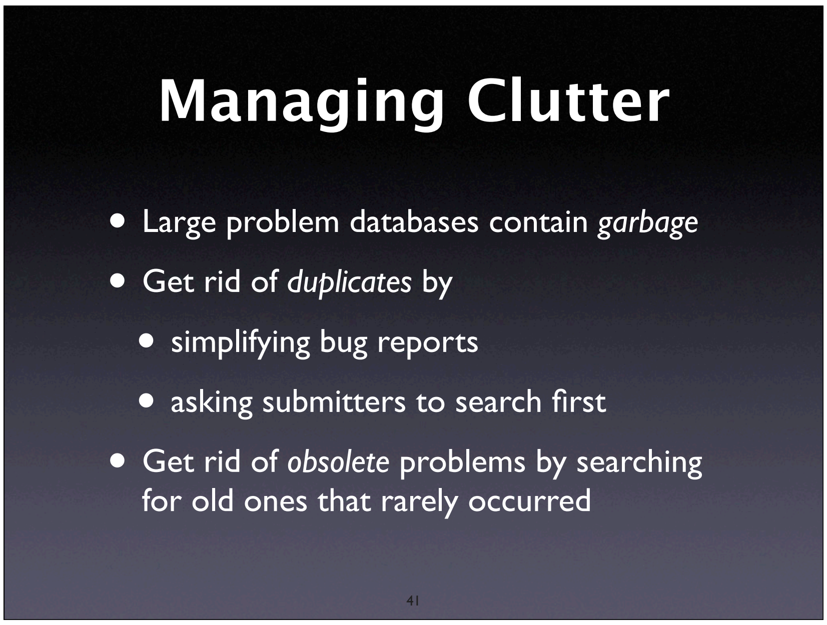
- At many organizations, a *software change control board* is in charge of these questions:
  - Assess the *impact* of a problem
  - Assign tasks to developers
  - Close issues…



Problem-driven Development

- The whole development can be organized around the problem database:
  - Start with one single problem: "The product isn't there"
  - Decompose into sub-problems
  - Ship when all problems are fixed

40

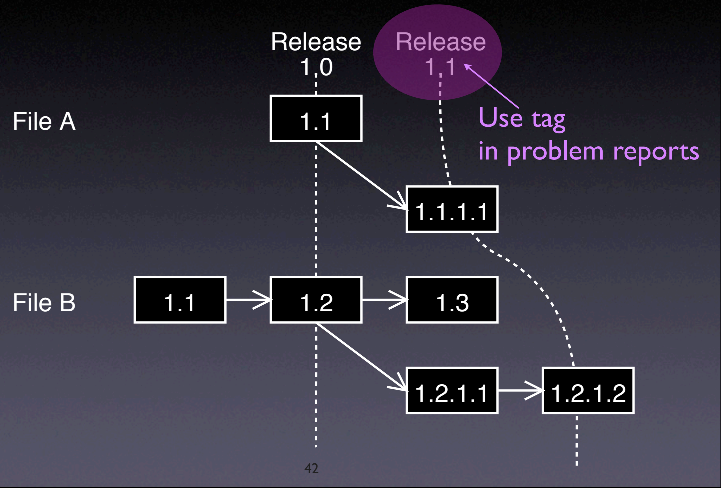


Managing Clutter

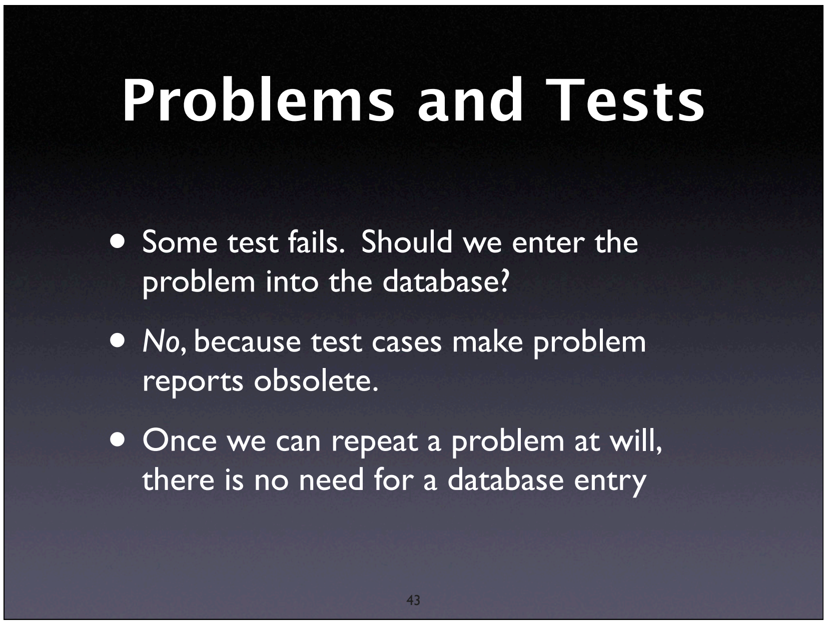
- Large problem databases contain *garbage*
- Get rid of *duplicates* by
  - simplifying bug reports
  - asking submitters to search first
- Get rid of *obsolete* problems by searching for old ones that rarely occurred

41

Problems and Fixes



42

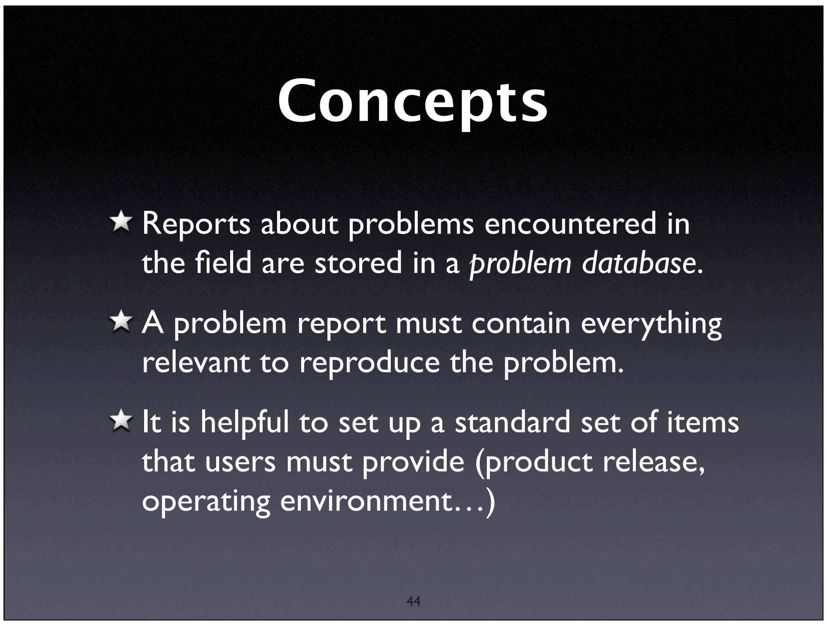


Problems and Tests

- Some test fails. Should we enter the problem into the database?
- *No*, because test cases make problem reports obsolete.
- Once we can repeat a problem at will, there is no need for a database entry

43

43



Concepts

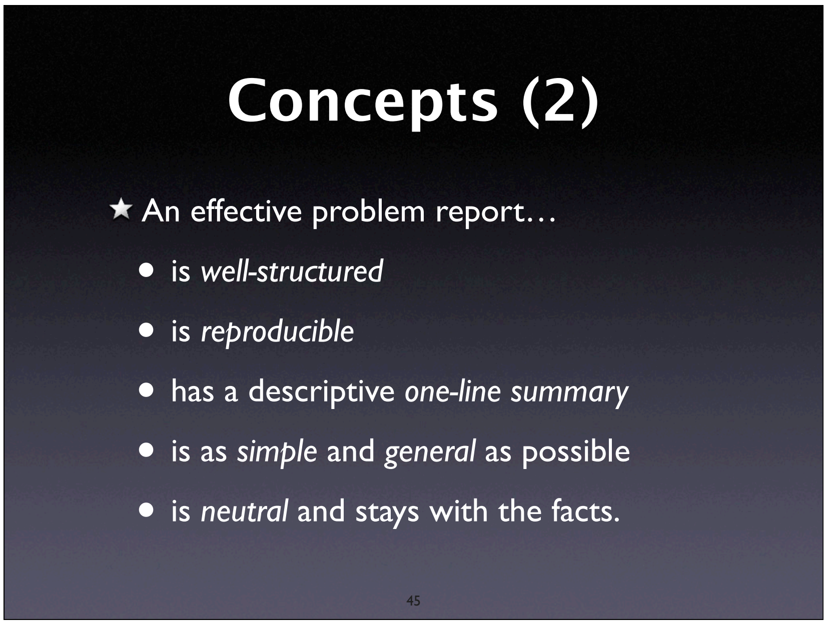
★ Reports about problems encountered in the field are stored in a *problem database.*

★ A problem report must contain everything relevant to reproduce the problem.

★ It is helpful to set up a standard set of items that users must provide (product release, operating environment…)

44

44



Concepts (2)

★ An effective problem report…

- is *well-structured*
- is *reproducible*
- has a descriptive *one-line summary*
- is as *simple* and *general* as possible
- is *neutral* and stays with the facts.

45

45

Concepts (3)

- A typical problem life cycle starts with an *unconfirmed* status
- It ends with a *closed* status and a specific *resolution* (such as *fixed* or *worksforme*)
- Typically, a *software change control board* organizes priorities and assignments

46

Concepts (4)

- Use *version control* to separate fixes and features during development.
- Establish conventions to relate *changes* to *problem reports* and vice versa.
- Make a problem report *obsolete* as soon as a test case exists.

47

48