I hate Maintenance !

Maintenance SUCKS !

ThoughtWorks®
The art of heavy lifting.℠

Agile Maintenance

Naresh Jain

Agenda of this discussion

- ✓ Glossary
- ✓ Why maintain software
- ✓ Types of Software Maintenance
- ✓ Traditional v/s Agile Maintenance Approach
- ✓ Agile Testing
- ✓ Agile Release Management
- ✓ Why maintenance is so expensive (and can be destructive)
- ✓ Agile Maintenance Best Practices
- ✓ Conclusion
- ✓ Q & A

Glossary

Software maintenance is defined as *the process of modifying a software system or component* ***after delivery*** *to correct faults, improve performance or other attributes, or adapt to a changed environment* [IEEE 1990]

Iterative Agile Development

What's the difference between Maintenance and Support?

ADD MAX VALUE TO THE CLIENT !

Why maintain software?

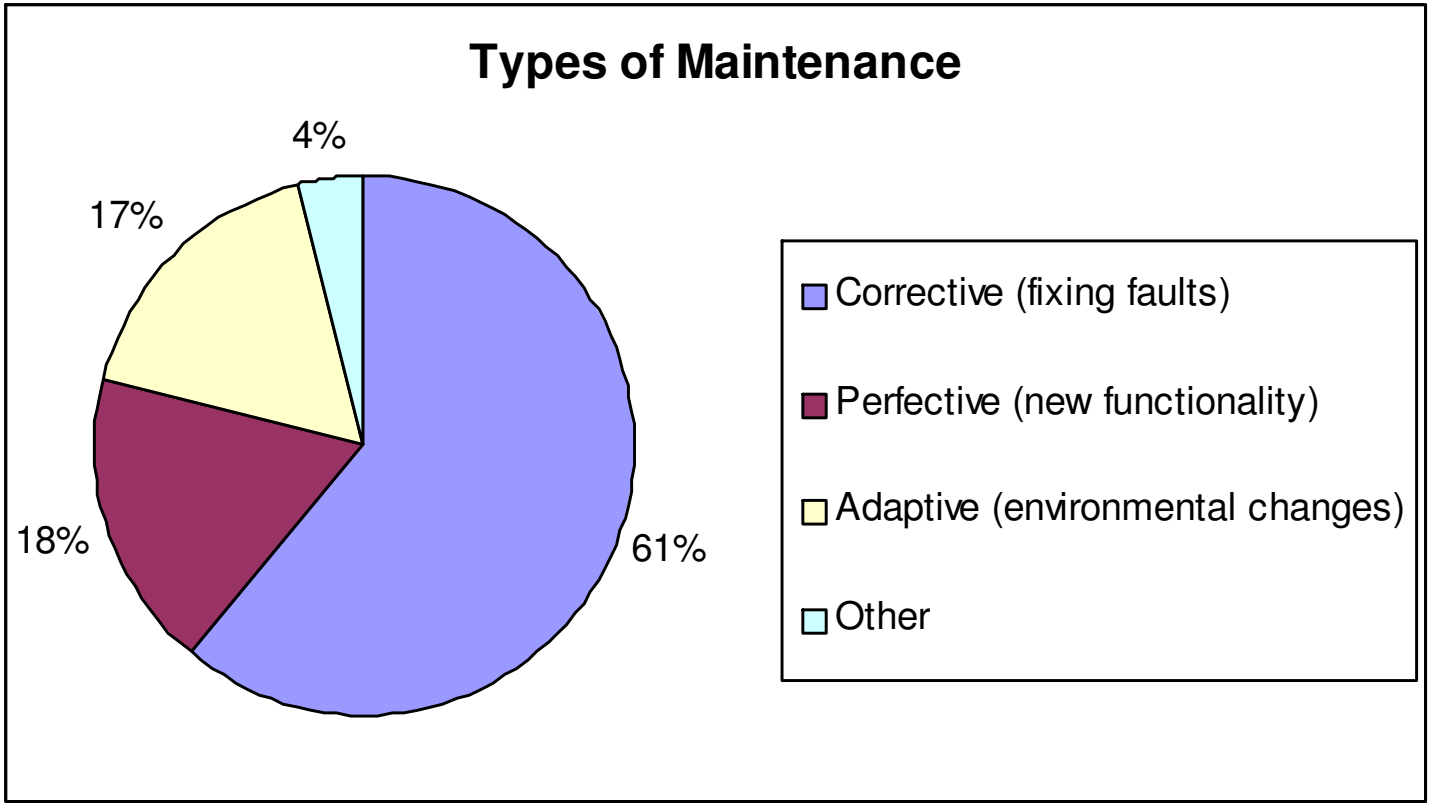
Stephen R Schach summarizes:

- **Model of reality**. As the reality changes, the software must adapt or die.
- **Pressures from satisfied users**, to extend the functionality of the product.
- Software is much **easier to change** than hardware. As a result, changes are made to the software whenever possible.
- Successful software **survives well beyond the lifetime** of the environment for which it was written.

Types of Software Maintenance

- **Corrective maintenance** *is maintenance performed to correct faults in hardware or software [IEEE 1990]*
- **Adaptive maintenance** *is software maintenance performed to make a computer program usable in a changed environment [IEEE 1990].*
- **Perfective maintenance** *is software maintenance performed to improve the performance, maintainability, or other attributes of a computer program. [IEEE 1990]*
- **Preventative maintenance** *is maintenance preformed for the purpose of preventing problems before they occur [IEEE 1990]*

The Frequency of Each Type of Maintenance



ThoughtWorks®
The art of heavy lifting.℠

Traditional v/s Agile Maintenance Approach

Traditional approach

1. Massive projects with waterfall model
2. CR Form
3. Change control board [CCB]
4. CCB approves and prioritizes the bugs
5. Summary of the bugs handed over to the developers – loss of context
6. Developers jump to the code and fix it
7. Update all the design documents
8. Regression testing
9. Patch release

Agile approach

1. Small projects with agile model
2. Bug reports / story cards
3. Developers estimate
4. Customer prioritizes the bugs
5. Common bug tracking database
6. QAs test and write functional acceptance tests
7. Developers write failing unit tests
8. Developers fix it and run unit & functional tests
9. Regression testing
10. New release

Quick Fix or Iterative Development

- Quick fix is often used in emergency, corrective maintenance. *Emergency maintenance* is *unscheduled corrective maintenance performed to keep a system operational* [IEEE 1998]. Also called Code & Fix

- Iterative development process is based on the Evolutionary development paradigm [Takang and Grubb 1996] or Extreme Programming paradigm.

- Iterative enhancement involves a five-staged cycle:

- ❑ Analysis (simulation)
- ❑ User acceptance tests and Unit tests
- ❑ Redesign and implementation
- ❑ Regression tests
- ❑ Release

Agile Maintenance Testing

- ***Acceptance Testing – xUnit, FIT, …***
- ***Unit Tests – Black box functional unit tests***
- ***Regression testing is selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements [IEEE 1990]***

Agile Release Management

Software release management, also known as software configuration management, is the management of activities surrounding release of one or more versions of software to one or more customers. Release management includes defining acceptable quality levels for release, authority to authorize the release, release procedures, and so on [IEEE 1990].

Best practices:

- Automated release creation and verification process
- Automated management of application specific configuration
- Simulation of target deployment environments and automated testing
- Release creation from the QA environment

Maintenance Smells

- A lot of regression bugs
- Less than 50% of the time is dedicated to refactoring
- Developers not pairing 100%
- Test coverage not improving with time
- Lack of team outings
- People getting stuck on the project for more than a year

ThoughtWorks®
The art of heavy lifting.℠

Why Maintenance Is So Expensive (and Can Be Destructive)?

- Team stability
- Contractual responsibility
- Staff skills
- Transition plan for team members
- Program age and structure
- Stressful nature of work
- Fluctuation in the workload and resourcing problems
- Planning

Agile Maintenance Best Practices

- Automated regression testing
- Continuous integration
- Coding standard
- Developers can focus on maintainability - Refactoring
- Pair programming
- Application logs
- Bug Tracking database – Knowledge repository
- Retrospectives

ThoughtWorks®
The art of heavy lifting.SM

Agile Maintenance Best Practices . . .

- QA – Developer pairing
- *Debugging is parallelizable*
- *Involve your customer – Daily status, Project Wiki in Customer environment, IM conversations, IPM,*
- Constantly improve the test coverage
- Use some acceptance test framework
- QA smoke test on the developer machine for immediate feedback
- Short releases

My 0.02

- Source code is the king
- Untrustworthy documentation
- The bug-tracking database stores knowledge
- Reproduction is essential to obtaining a solution

Retrospective

ThoughtWorks®
The art of heavy lifting.℠

Thank you!

email: njain@thoughtworks.com
blog: http://jroller.com/page/njain