Building Software at Microsoft

David Greenspoon

Microsoft Corporation

Agenda

- Basics of development at Microsoft
- Challenges of large scale development
- Development tools at work

Microsoft "Development" Job Functions

Program Manager – collects requirements, writes specifications, handles some project management

Developer – architects, designs, writes code, fixes bugs

Tester – develop test plans, write test code and automation, manual testing

Microsoft Development Cycle

- Milestone Zero
- Milestones 1-N
- Code Complete
- Beta(s)
- Release Candidate(s)
- Release to Manufacturing/Web

Build Room

- Every major product or suite has a build lab where the product is put together:
  - Compile and link various modules
  - Performance tuning is performed
  - Collect files and build setup
  - Run basic verification tests

Build Room Problems

- Builds are too painful because of build breaks and integration problems
- No matter how many additional procedures we try and use to improve the quality of the builds, new problems always surface
- We try and introduce better process into the build room but those changes results in new and different problems

Build Room Evolution Example

- Problem: Too many blocking bugs were coming out of the builds
- Solution: Include a suite of automated tests in the build process
- New Problem: Automated test failures now hold up the build
- Solution: Have developers run automated tests before checking in
- New Problem: Developers blocked from checking in changes when tests need to be updated

Bug Trends

Sources of large numbers of bugs:

- UI layout and behavior (especially after localization)
- Setup and configuration
- Integration of code from different developers/groups
- Changing unfamiliar code (large number of regressions from bug fixes)
- Extensible architectures (multiple drivers, API's, object model) hard to test

The "Worst" Bugs We Can Ship

- Data loss or corruption bugs
- Break existing documents or applications
- Security holes
- Reliability problems

What Happens When a Customer Finds a Bug?

- Customer notifies Microsoft product support
- If new and serious problem, product support hands off QFE to product team
- Product team makes fix and tests it
- Microsoft releases QFE to customer
- If it's a serious problem (e.g. security hole), Microsoft notifies customers and posts fix on the web
- Fix gets rolled up into service packs and future releases

This is expensive for Microsoft and our customers

Challenges of Large Scale Development

- Quantity and complexity of code
- Sharing code and compatibility
- Aligning schedules
- Sharing development

Quantity and Complexity of Code

- Microsoft has numerous products with millions of lines of code that work closely together
- C++, OLE, and COM make it very hard to look at, and sometimes walk through, code and understand what is happening
- Frequently code will stay in the product longer then the developer who wrote it

Sharing Code and Compatibility

- A large part of Window's success derives from the ability to share code
- Key examples:
  - DLL's for sharing code (all of Office 2000 uses common code in mso9.dll)
  - Advanced Windows controls available in Windows 98 and 2000 (riched20.dll, mshtml.dll)

Code Sharing Dilemma

- Windows/IE4 make mshtml.dll available to ISV's for HTML development
- Outlook 98 and other applications choose to use mshtml.dll for HTML rendering and editing instead of duplicating effort
- Windows/IE5 makes huge advances in HTML capabilities – does the team produce a new mshtml.dll or mshtml2.dll?

Code Sharing Dilemma (cont.)

- Advantages of mshtml.dll:
  - Releasing mshtml.dll allows Outlook 98 and other applications to take advantage of new functionality
  - Only one instance of mshtml on disk and in memory
- Advantages of mshtml2.dll:
  - Releasing mshtml2.dll will likely not affect Outlook 98 or existing applications
  - Allows new applications (like Outlook 2000) to only test with one version of mshtml.dll

Aligning Schedules

- Microsoft now ships big suites of software (e.g. Office 2000, Visual Studio 6)
- Multiple large, complex, and inter-dependant applications need to be ready to ship at the same time
- Management challenge in trading off being aggressive and including important features vs. limiting risk of one or two pieces holding up the entire product

Sharing Development

- When one product is so big (e.g. Office) it is worthwhile to clearly identify the common code that needs to be written and create teams to build that common code
- Best way to form these teams is with people from the products who will use the shared code
- Shared feature team developers have the added complexity of working on multiple, unfamiliar code bases with varying levels of cooperation from the product teams

Development Tools at Work

- Typical productivity tools
- Common development tools
- Advanced development tools
- Specific examples

Typical Productivity Tools

- Outlook & Exchange for collaboration
- Word or FrontPage for specifications
- Excel or Project for schedules

Common Development Tools

- Microsoft Visual Studio for compiler, linker, sometimes development environment
- Code editor (various)
- Source Control System (VSS or other for large scale)
- Bug tracking tool
- Visual Test

What Language Do We Speak?

- Most code at Microsoft is written in C or C++
- My personal beef with C++: Exception Handling causes bugs and performance degradation
- We have tried Java but it is not a big improvement over C++:
  - Fewer small memory corruption errors
  - Performance clearly worse then C++
- Still searching for the new paradigm that will really make a difference to how we write code

Advanced Development Tools

- Performance tuning tools
- Tools to help scan source code for bugs
- Tools to automatically test products

Specific Tool Examples

- Source Insight™
- Discover®
- BugTrapper™

Note: Microsoft products exceed the limits on many tools we evaluate and hope to use

Source Insight™

- An example of an external tool success at Microsoft
- Highly targeted editor/browser enables developers to navigate a big, complex code base
- Tool is small and fast
- Index files are small and kept up-to-date (without building)

Discover®

- An example of an external tool that didn't succeed at Microsoft
- Discover parses code into tree that you can parse further and do static analysis on using their rich model
- Expensive and slow (especially if you have lots of pre-compiled headers)
- Biggest failing is the lack of an advanced UI to allow developers to really walk through the code

BugTrapper™

- An example of an external tool that shows promise
- Lets you select what source code you want to watch
- Records function parameters and return values
- Allows developers to step backwards and forwards through the executed code to learn or debug
- Trace size is large (5MB for a small trace without symbols)
- Needs to be integrated better into the debugger to be more useful

Summary

- Skill, discipline and management play a large role in product development
- Quantity, complexity and compatibility are hurting us
- Tools are pervasive today in the development process, but …
- More advanced tools to handle the complexity of large scale development are badly needed

Questions