



How Microsoft Builds Software

By Mary Cook and Adam Goodman.

Adapted for Powerpoint from an ACM paper by:
A. Cusumano and Richard W. Selby



Presentation format:

- Everyone has their own view (prejudice?) of Microsoft.
- Please keep them to yourself.
- Quick overview of Microsoft as a company.
- Insight into their Software development Processes.
- Conclusions that can be drawn from the paper.



Microsoft – Brief Overview

- Founded in 1975, today Microsoft is one of the largest and most successful companies in the world
- Microsoft employs more than 50,000 workers and its products create millions of jobs for people throughout the world. (source: Yahoo – 03/03)
- Chairman is William H. Gates III
- One of the only companies to have never fallen into debt
- After winning a contract from IBM to provide operating systems for their machines (Windows 95) they have made their system a universal standard, which is constantly updated, and adapted to new technologies and needs.



Microsoft – Brief Overview

- Microsoft Today?
- Designs, Licences and supports a wide range of software products that run on a multitude of platforms.

Microsoft – Brief Overview

- From Heathrow Airports flight scheduling system...



Microsoft – Brief Overview

- To cash machines worldwide...



Microsoft – Brief Overview

- To computer hardware;



Microsoft – Brief Overview

- Question: How does Microsoft manage to have its finger in every pie?



By Creating Products which are "good enough" rather than perfect!





Microsoft – “Good Enough?”

- What does “good enough” actually mean?
- It continually improves its software after release, through fixes, patches and updates and in this way, also allows it to charge for new versions.

How Microsoft Builds Software

- However, not everyone is happy with this method of Incremental software improvement.





Digression – The Paper.

- The authors spent 6 months at Microsoft
 - Interviewing Staff
 - Reading Documentation
 - Observing what went on
-
- They based their paper on their experiences there.



What does Microsoft Build?

- Microsoft builds large, complex software products - requiring large numbers of people working on the same project.
- The problem with many developers working on the same project is communication between team members.



What does Microsoft Build?

- When the company was originally formed, its products were developed in an ad-hoc fashion by one or two people, with few design documents prior to coding.
- But times have changed!
- Example: Windows 95 had over 11 million lines of code!



Small Teams have advantages!

- Microsoft realised that having people working in small teams is advantageous:
- Individuals can be innovative and creative.
- Communication between members of small teams is more efficient than big teams.



How Microsoft Builds Software.

- Microsoft chose to scale up this ‘hacker mentality’ of having small groups working together.
- 3 – 8 people per team.
- Freedom of Expression.
- Each team works on a specific area of a product.



How Microsoft Builds Software

- THE PROBLEM:
- By having artistic freedom and each group working independently, they run the risk of the work becoming incompatible or overlapping.

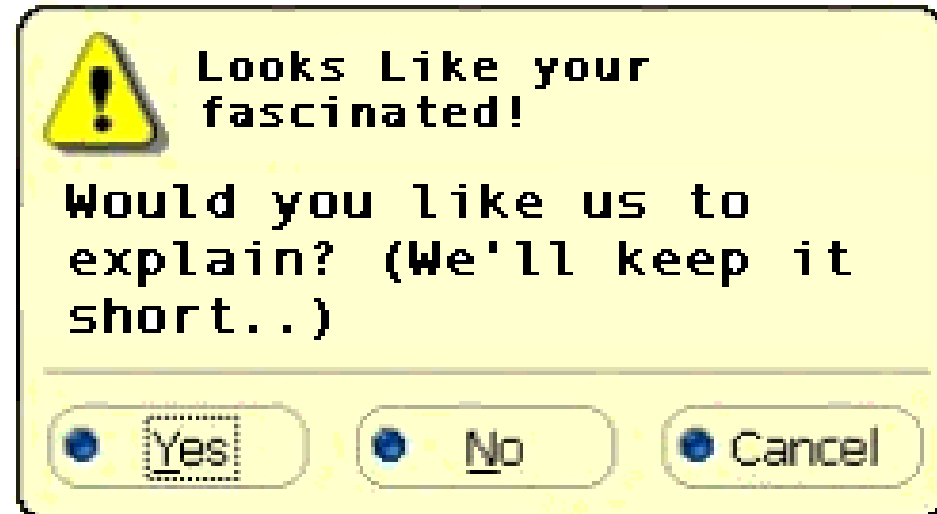


How Microsoft Builds Software

- THE SOLUTION:
- They give the small teams a larger structure to work within.
- Simply: They make each group synchronise the work they have done often.
- They call this the ‘synch-and-stabilise’ method.

Why do they do this?

- They hope that this will achieve their aim of producing large, stable, feature-rich products that are delivered on time.



How Microsoft Builds Software

■ The development phase is done in 3 stages.

- 1) Planning Stage
- 2) Development Stage
- 3) Stabilisation Stage





Planning Phase (1st Phase)

- Starts with production of Vision Statement
- Vision Statement created by marketing managers and contains features deemed important by customer research.
- Used to produce a loose functional specification which will be overviewed briefly.

Functional Specification

- Outlines Architectural Issues
- Outlines Features
- The features are grouped in order of importance for implementing and given an implementation time.
- We believe “Microsoft paperclip” would be high up the list of desirable features.





Why create feature groups?

- It is hoped they will evolve a “horizontal program architecture”.
- This means the most important features are common across many of Microsoft’s products.
- Examples include web-integration and help features.



Why Create Feature Groups?

- Next layer of functionality is product specific and is build upon previous layer.
- Example: Calculation logic for Excel
- This strategy means that Microsoft products are compatible with each other as early as possible.

Functional Specifications

- Also used to organise groups timeplan.
- Emphasis is put on creating a good product but at the same time putting pressure on the developers to have a working product ready by deadline.



Development Phase (2nd Stage)

- Consists of 3 “sub-projects” each focussed on implementing a group of features.
- Sub-project 1 are the most important features, with sub-project 3 being the least.
- Last for 2-4 months with 20% ‘buffer time’ for unforeseen circumstances.
- Each sub-project encompasses design, coding, feature integration, usability and code stabilisation (more on this later)

Development Phase (2nd Phase)

- Teams may change features/design as they see fit – thus allowing for innovation.
- Features are allocated to teams who code separately
- Teams collaborate when features are integrated into the product (synchronised)



Development Phase (2nd Phase)

- Code deemed 'bug free' by a development team is 'checked-into' a central code set.
- This code set is compiled by product managers who oversee a single project and is again tested for bugs.
- If bugs are found the problem code is 'rolled-back' to the last bug free version.



Development Phase (2nd Phase)

- All O/S versions (pc/mac etc) and all language versions of the project are compiled simultaneously.
- This means that a shippable version of the project is available as early as possible.



Why synchronise frequently?

- Teams working on separate functions continually combine their code – Why?
- If there is overlap between teams it can be rectified as early as possible.

Why synchronise frequently...

- Frequent synchronisation means that all parts of the project are known to work together correctly....
- But as we have all experienced; things may slip through the net...



Usability Labs



- Continuously conducted through stage 2 (development phase)
- Allow prospective users to test the software and give their opinions of it.
- Allows the developers to adapt the software to the ever changing user needs and market climate.
- If a prospective user wants a feature it can be added during the development phase.



Stabilization Phase (3rd Phase)

- Occurs at the end of the development process.
- Prepares the product for release by attempting to remove as many bugs as possible.



Highlights of Microsoft's Method

- Aims to produce large, feature rich, stable products on time.
- Large products are achievable using the small team methodology as they make up one large team due to the daily code builds.
- Feature rich products are possible because user feedback is constantly evolving the final product.

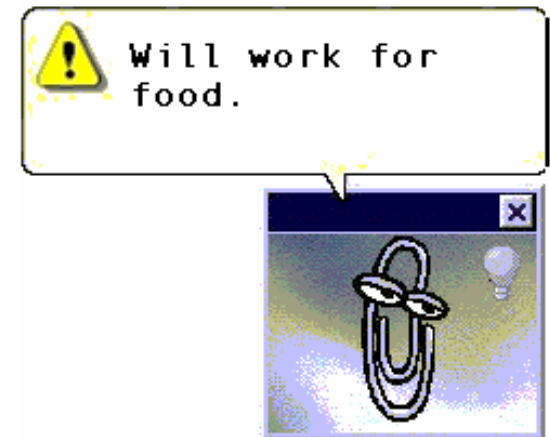


Highlights of Microsoft's Method

- Small development teams are allowed to innovate thus improving the product using their own ideas.
- Bug-free products are possible because of the periodic stabilisations

Highlights of Microsoft's Method

- Products can be delivered on time due to:
- synchronisation of different parts of the project is done continually, rather than at the end of the project.
- if a release date is deemed to be impossible to meet, features are dropped.
- Which could unfortunately mean.....





Highlights of Microsoft's Method

- Microsoft tries to standardise as many methods as possible while still allowing innovation and agile product evolution.
- Coding styles, development tools, team schedules, project schedules, testing methods and quantitative decision-making are all very tightly controlled
- BUT: developer teams still have the freedom to guide their own work



Evaluation of Microsoft's Method

- Microsoft's sub-project structure contrasts with traditional development models.
- The "waterfall" model, for example, is a linear progression of stages, from outlining software requirements to testing and does not allow for an evolving product specification

Evaluation of Microsoft's method

- If separate parts of the project are not compatible, or there are major bugs, they will not be discovered until the end.
- Meaning added expense and delay to the release.
- "Synch-and-Stabilise" approach aims to avoid this through constant integration and testing, so any problems become apparent very early on.

Points to ponder....



- If the product is continuously and thoroughly tested why do we get bugs?
- Is 'good' really good enough??
- Approach scales down well because the technique is proven for small groups.
- Scales up reasonably well, though really large projects (like Windows) tend to be more buggy than they should be.
- Is it acceptable to wait for fixes to become available?

Paper Evaluation

- Offers an insight into Microsoft's software processes and has been used as a de-facto referenced by many other papers.
- The authors state objectives such as “the company needs to pay more attention to.” without explaining themselves.
- Testing details not explained (so don't ask when it comes to questions!)

WARNING!

The system is either busy or has become unstable. You can wait and see if it becomes available again, or you can restart your computer.

- * Press any key to return to Windows and wait.
- * Press CTRL+ALT+DEL again to restart your computer. You will lose unsaved information in any programs that are running.

Press any key to continue _