CS446
Software Engineering

Richard Bartels

University of Waterloo
Faculty of Mathematics
Department of Computer Science

September 13, 1996

CS446 Software Engineering Issues

Richard Bartels

University of Waterloo
Faculty of Mathematics
Department of Computer Science

September 13, 1996

Outline

- Large-Scale Developement Aspects
- Generic Processes
- Remarks
- Things that Go Wrong
- One Example: ISO 9000

Reference:

This is background material for Chapter 1 of Sommerville

Large Scale Aspects

At least some of the following (and others?):

- System of significant complexity
- Accurate specification critical
- Important to build it correctly
- Many people required
- Management and coordination at demanding levels
- Substantial development time and cost
- Development strategy unavoidably complex
- Development support taxes infrastructure and logistics
- Information storage and retrieval become major issues
- Long expected lifetime of system usage
- Many evolving versions foreseen
- Broad spectrum of platforms to be supported
- Significant issues of documentation and legacy

And It Goes On

- Extreme time and performance pressures on deliverables
- Extensively delivered to a heterogeneous customer base
- Customer service and troubleshooting mandatory
- Competitive products with comparable market share
- Lucrative market being contested
- Safety critical performance imperative

Generic Processes I

- Definition aspect
  - Problem/goal formulation
  - System/requirements analysis
  - Resource/risk analysis
  - Project planning/scheduling
- Development aspect
  - System/software design
  - Implementation
  - Testing/verification
- Maintenance aspect
  - Delivery/service
  - Customer support/training
  - Correction
  - Adaptation
  - Enhancement

Generic Processes II

- Management aspect
  - Resource organisation
  - Team guidance
  - Communication facilitation
  - Schedule tracking
  - Quality monitoring and assurance
  - Risk recognition and avoidance
  - Damage control

Remarks

- Aspects are in evidence at all times to some degree
- Aspects are clearly recognized as important
  - Communality with any other large human project
  - Centuries of experience
  - Little or no religious debate
- How to deal with them is still not clear
  - Software distinct from bridges and invasions
  - Scarce experience
  - Many techniques proposed
  - Sometimes religious wars
- This course will use some techniques
  - **NOT** claimed to be the best or only
- Purpose is to give you exposure to
  - Some of the aspects themselves
  - Some techniques that are used
- Purpose is awareness

Some Things That Go Wrong

- Definition – Formulation
  (What could we do? What shall we do?)
  - Don't precisely know what's needed
  - Don't specify sufficiently to check achievement
- Definition – Analysis
  (How could we do it? How could we fail?)
  - Unable to predict what it will take
  - Unaware of resources/tools
  - Unprepared for difficulties
- Definition – Planning
  (Who will do it? When will they do it?)
  - Milestones poorly mapped
  - Contingency alternatives missing
  - Staffing inadequate or inappropriate

Some Things That Go Wrong

- Development – Design
  (How is it done?)
  - Inappropriate systems/languages/libraries/tools
  - Strong coupling, code scattering, poor abstractions, ...
- Development – Implementation
  (Now we do it!)
  - Unnecessary effort, re-invention
  - Stylistic chaos, inadequate documentation
- Development – Testing/Verification
  (Does it work? Does it do what it was supposed to?)
  - No structure/plan for testing
  - Don't build for testing
  - Insufficient allocation of resources for assessment
  - Spotty or superficial testing

Some Things That Go Wrong

- Maintenance – Provision
  (Can we deliver it? Can the user figure it out?)
  - No marketing/production/supply consideration
  - Poor manuals/tutorials/references
- Maintenance – Aspects of change
  (Do we find out what's wrong?
  Are we in touch with the users?
  Can we fix it?
  What if its environment shifts?
  How can we improve it?)
  - No provision of support nor collection of feedback
  - Design/implementation rigid and confused
  - Developer documentation inadequate
  - No testing legacy to rely upon
  - Implementation strongly bound to environment

Some Things That Go Wrong

- Management – All aspects
  (Can we keep it on track? Are we succeeding? How well?)
  - Poor lines of communication
  - Lack of coordination
  - Inappropriate team/corporate structure
  - No basis on which to judge progress
  - Inattentiveness to detail
  - Problems not recognised in time
  - Inadequate information/archive resources

One Paramount Truth

The earlier you can catch a problem,
the cheaper it is to fix it.

> Specify, test, verify, validate <

- At every point in the process
- For every item, component, or issue

To accomplish this:

Have institutionalised, known, clear procedures in place for

quality assurance

quality assurance management

One Example: ISO 9000

Say what you do
Do what you say
Have clear responsibility
Have comprehensive record keeping
Have complete communication
Use accepted and effective practices

***

Inception

EU desire to eliminate quality issues as an excuse for trade barriers

Goal

A structured process through which quality of goods/services is raised to a target level and then maintained

Elements

pre-audit, process mapping and documentation, any necessary adjustments of existing processes to attain desired level, any introduction of new processes as needed, compliance audit, registration audit, periodic maintenance audits

Guidance

National auditing boards and certified auditors