Software Engineering
Best Practices

Practical things we can all do

David Loo
IMS Health
CUSEC 2002
About IMS Health

- In Canada from 1960
- Offices in over 100 countries (HQ in US)
- Canada’s leading supplier of health information
  - Pharmaceutical consumption rates/patterns
  - Prescription estimates
  - Disease & treatment patterns
About me

- BSc, MSc McGill (Physics)
- 16+ years in software industry
- Held positions as software architect, software engineer, project manager
- Matrox, Clinidata, IMS Health
- In charge of development standards & methodology at IMS Health
Agenda

- Introduction
- Caveats
- Best practices
- Where to get more information
- Questions and Answers
Introduction

- What is Software Engineering?
- What is a Best Practice (BP)?
- How these BPs are chosen
What is Software Engineering?

"The intelligent application of proven principles, techniques, languages, and tools to the cost-effective creation and maintenance of software that satisfies users’ needs." [Dav95]
What is a Best Practice (BP)?

“A principle, technique, or rule about Software Engineering that is applicable regardless of the development methodology, language, or application domain.” [Dav95]
How these BPs are chosen

- They’re proven in the field
- Personal experience
- Practical, easy to implement
Agenda

- Introduction
- Caveats
- Best practices
- Where to get more information
- Questions and Answers
Caveats

- List is not exhaustive
- Not everything has to be followed
- Not everything works all the time
- Not every BP is compatible with each other
- Not necessarily for project managers or leads only
Agenda

- Introduction
- Caveats
- Best practices
- Where to get more information
- Questions and Answers
Best Practices

- General BPs
- BPs for Construction
- BPs for Test
- BPs for Documentation
Best Practices

- General BPs
- BPs for Construction
- BPs for Testing
- BPs for Documentation
Quality First

- Users won’t tolerate product of poor quality, no matter how quality is defined
- Quality cannot be “retrofitted”
- It’s better to have poor efficiency than poor reliability
- High quality = low productivity
- Most every BP ultimately traces to ensuring Quality in software products
Triage

“A system used to allocate a scarce commodity, such as food, only to those capable of deriving the greatest benefit from it.” [You97]

Major application is in requirements management
Triage (cont’d)

- Classify requirements using **MoSCoW** rule: “**Must have**”, “**Should have**”, “**Could have**”, “**Wish to have**” [Sta97]

- Focus only on “**Must have**” requirements first

- Must actively & continuously manage this with all stakeholders (Users, development, QA, Marketing, management)
Top 10

- List of 10 most serious risks to project
- Include status, context, possible resolutions
- Update every week
- Raises awareness of project risks
- More timely solutions possible
- Improves visibility of progress
Configuration management

- Use CM tool (SourceSafe, etc.)
- Archive all versions of all intermediate Artefacts (specifications, code, test-plans, user manuals, etc.)
- Assign name/version number
- Have a baseline as early as possible
- Control ALL changes to baseline
- Track every change
Defect tracking

- Log of all defects found in all stages of life cycle
- Include also suggestions
- Must have version number from CM tool
- Easier post-mortem analysis
- Easier metrics gathering
Have standards

- Promotes discipline
- Ensures uniformity of artefacts
- Use of common language
- Easier maintenance
- Availability of tools
Don’t follow standards

- Good methodology is one that makes sense to the company
- Don’t have to follow every single step
- Adopting new methodology often accompanied by huge productivity drop
- Fix fundamental problems first
- Be careful when following trends
Miniature milestones

- Milestones that are achievable in 1-2 days
- Good for crisis or project recovery
- Good for team motivation
- Increased status visibility
Avoid Waterfall methodology

- Trying to freeze requirements, design,…
- Trying to plan details from beginning to end
- Not suitable for new, unfamiliar projects
- Not adaptable to changes
- Pushes risks to tail-end of project
- Testing way too late
- Integrating way too late
- *
Use Iterative methodology

- Project goes through many iterations
- Each iteration includes usual 4 stages
- Each iteration refines requirements, design, build, tests, …
- Each iteration adds more and more features
- Users see value with each iteration
Iteration

1. Business modeling
2. Requirements
3. Analysis & design
4. Implementation
5. Test
6. Deployment

Start Iteration

End Iteration
Iterative methodology

- Embraces changes, not resists them
- Emphasis on working software, not documents
- Creates working system as early as possible
- Continuous testing, continuous integration
- Risk driven: greatest risks handled first
Release early

- Define & create release process and actual release deliverables as early in project life-cycle as possible (iterations)
- Minimizes integration problems
- Avoids “end of build” rush
Framework group

- A group to define company-wide common architecture, development tools, & methodology
- Promotes reuse of common components
- Decreases risks
Do a post-mortem

- At project end, key players analyze every problem that occurred in project
- Goal is to document, analyze, and learn from mistakes
- Metrics can also be gathered
- Usually takes 3-5 days
- Great benefits to future projects
Best Practices

- General BPs
- BPs for Build
- BPs for Testing
- BPs for Documentation
Captain’s Log

- Keep a log of what was done
- Can be written or electronic
- Especially important for build
- Easier recall of what or why
- Easier post-mortem analysis
Daily build

- System completely rebuild every day (usually overnight)
- Treat it as heartbeat or synch pulse of project [Cus95]
- Fixing broken build is given top priority
- Minimizes integration risk
- Easier defect diagnosis
Daily smoke tests

- A set of tests that exercises major functional areas of system
- Should be automated
- Run it after each Daily Build
- Update the set as more functionality are added to each iteration
Fix bugs as soon as found

- Don’t wait “until later”; “later” never comes
- Fix while code is still fresh in memory
- No new features until bugs are fixed
- Greatly increases quality of product
Best Practices

- General BPs
- BPs for Build
- BPs for Testing
- BPs for Documentation
Single-step through every line of code

- Step through code in debugger
- Step through EVERY SINGLE line of code that was added, changed, moved
Branch, path, and other things

- Test every branch
- Test every path
Do Inspections

- Catches high percentage of bugs
- Promotes good coding practices
- Promotes use of coding standards
- Great for people new to project or code
Best Practices

- General BPs
- BPs for Build
- BPs for Testing
- BPs for Documentation
Create a glossary & index

- Every document should have glossary & index
- Especially important for User documents (requirements specs, user manuals)
Be boring

Which is better?

- “There are three types of special commands. Regular commands come in four varieties.”

- “There are three types of special commands. There are four types of regular commands.”

[Dav95]
Speak the reader’s language

- Don’t put technical terms in documents destined for Users
- Use the reader’s terminology
Agenda

- Introduction
- Caveats
- Best practices
- Where to get more information
- Questions and Answers
Where to get more information

- Bibliography lists 31 other books, papers, and web sites.
Questions and Answers