Ship it!

The entire project team should treat driving the software to a releasable state at the end of each stage as its top priority.

SG, McConnell
Readings

Required:
- “Good enough Quality: Beyond the Buzzword”, James Bach (link on web)

Recommended:
- “Software Project Survival Guide”, Steve McConnell
  - Chapter 16, Software Release
- “Rapid Development”, Steve McConnell
  - Chapters 14 – Feature Set Control
Outline

- When is the product ready to release?
- Are there other issues involved?
- What is a typical release schedule?
When is the product ready?

Consider the criteria for “good enough” quality:

1. The product has sufficient benefits
2. It has no critical problems
3. The benefits sufficiently outweigh the problems
4. In the present situation, and all things considered, further improvement would be more harmful than helpful

Evaluate your project in light of these criteria!
1. **The product has sufficient benefits**

Assess the benefits (from the customer viewpoint):
- Identify them *(feature set)*
- Are they useable?
- How desirable are they?
- Taken as a whole, is there sufficient benefit in this release over the last?

Cray example
2. It has no critical problems

Again, from the customer standpoint:
- Identify the problems (bug database!)
- How likely will the customer experience them?
- How damaging are they if they do arise?
- Are some critical/showstoppers?
- Is the set of noncritical, too much? (rate of defects)

Take it from me, Microsoft begins every project with the certain knowledge that they will choose to ship with known bugs. Bach
Rate of Defects

Survival Guide, McConnell
3. The benefits outweigh the problems

Enough said?

Cray example
4. Is further improvement more harmful?

How could *improvement* be harmful?

Consider:
- Strategy to improve
- Ability to execute
- Cost involved
- Schedule – can it wait for the next release
- Benefits
- Problems
Are there other issues involved?

DEFINITELY
Like what?

- Customer commitments
- Customer expectations
- Wall street expectations
- Revenue needs
- Competitive factors
- Beta release feedback
- Bonus plan...

Can you think of others?
A high-level release schedule

Date A  Feature Freeze
Date B  Code Cutoff

Start release builds and testing

- Daily meeting on commit requests
- Release candidate built daily (don’t break build!)
- Test results posted daily
- Ask “good enough”?

Date C  Create and distribute Beta test package
Date D  Understand/fix Beta problems
Date E  Release Overview, Documentation Done
Date F  Build and expose Final Package
Date G  Production printing/Media duplication
Date H  Release

CSE 403, Spring 2006, Alverson
A departing question…

Is your product “good enough” for the beta release?