Section 06: Discussion Points

(covered in 45 minutes)

(Continuing on the theme from "*The Joel Test: 12 Steps to Better Code*")

Which of the following steps does your team currently take?

- Automated testing
  - Unit tests (JUnit, NUnit)
  - Acceptance (end-to-end) tests: following use cases from start to finish
  - Coverage metrics (very important for the final release)
  - Results: 1 of 6 teams currently have some unit and acceptance tests.
- Documenting along with code development (rather than at the last moment)
  - User documentation (how can a customer use your product)
  - Technical documentation (major design decisions, assumptions, updated design docs, etc.)
  - Results: 2 of 6 teams have been doing it.
- Zero-feature release
  - Typically includes unit tests and documentation (and preferably a little demo the user can play with), not just oodles of source code.
  - If you leave it for later, it'll hurt you (but you probably won't believe me till that happens).

When will your software be ready to ship?

- Without ongoing automated tests, documentation, and daily builds -- it's anybody's guess, so it's a big risk.
- With all the above done concurrently, plus real-time coverage metrics (how do you know that it works?), it's ready to ship at any moment, so the risk is minimal to non-existent. From a business standpoint this is a huge advantage, because unexpected things do happen and a company in a competitive environment must be able to react to those as quickly and as predictably as possible.
  - The two questions of management:
    - When is it going to be done?
    - How much will it cost?

What is a beta release?

- It's a contract between a customer and a provider (i.e., you).
  - The 5 components of a contract:
    - *What*: The deliverable
    - *Who*: The customer and the provider
    - *By when*: The due date (and time)
    - *How*: The set of quality criteria that the deliverable must satisfy
    - *Success indicator*: how to tell if the contract is/isn't completed
- Includes the latest sources and binaries, unit and acceptance tests, user and technical documentation, a (small) demo, and release notes. The user must be able to find their way through without assistance from developers.

What if I asked you today to deliver tomorrow? Do you have readiness?
How can you respond to such a request?

- Speech Acts:
  - 4 possible ways to respond to requests (see handout):
    - *Yes*
    - *No*
    - *Counter-offer*
    - *Commit to commit*
  - All of the above are essential; inability to take one (or more) of these paths when a request is made indicates a communication problem and a threat to evolve into a product delivery problem.