Title

Author

August 6, 2012

1 COMP2911 Engineering Design in Computing

1.1 Tutorial - Week 1

Discussion Points

The Good, the Bad and the Ugly (aka Getting to Know You)

1. **The Ugly:**
   - **Name:** Louis Tiao
   - **Nickname:** None
   - **UNSW Program:** Computer Science/Mathematics
   - **Other:**...
2. **The Bad:**
   - ...
3. **The Good:**
   - ...

People As Programmers

1. I see a little bit of myself in most of the programmer predilections from A-Z with respect to the technical inclinations and not so much the personal idiosyncrasies. Specifically, I'm similar to Belligerent Brian and possibly Zealous Zack when it comes to my propensity to "select cutting edge technologies simply for its buzzword compliance, betting that cool acronyms and shiny new methodologies will make me appear progressive and forward-looking." I can also be a bit of a Feature Creep Frank when I indulge in personal projects in that I simply fail to exercise any form of self-restrain from adding a myriad of cool but often frivolous features.

2. Computer Programmers feel that coding is the entirety of their job. It can be said that they are slightly narrow-minded in that they work to close bug tracker issues, while Software Developers have a broad view and deep understanding of the business goals and works to deliver business value and uses technology as a means to an end to solving business problems. It is unlikely that the proverbial "person-in-the-street." *Their intent is to solve a business problem, not just to close an issue in a bug tracking system.*

Software Requirements

1. **What happened to the Mars Polar Lander in 1999?**

   The software triggered the shutdown of the Lander's descent engines, believing that the Lander had already landed on the surface of Mars.

2. **What went wrong?**

   The essential problem was a breakdown in communication of software requirements. The revised requirements never made it into the lower-level requirements. The testing process was also lax in that full regression testing was not performed, so after certain changes were made and bug fixes were done, only a part of the overall system was tested and the touchdown sensor bugs were not detected as a result.

3. **What lessons can you learn about team-based software development from this?**

   The importance of:

   - Communication
   - Requirement Tracking
   - Regression Testing

10 Elements of Good Software Design

1. TBH, I feel like most of the elements listed were slightly esoteric at this stage, only understandable by seasoned Object-Oriented developers. . . But otherwise, everything else was pretty much common-sense and hard not to agree with, e.g.

    - Provides The Necessary Functionality
    - Is As Simple As Current And Foreseeable Constraints Will Allow
    - Eliminates Duplication
    - Is Robustly Documented

2. Yes. I feel the order of important is reasonable, given the important of the thing being designed to meet its purpose, and also being as simple as possible.

3. The above points are consistent with these points from Dieter Rams:

    - Good design makes a product useful
      Synonymous with "Provides The Necessary Functionality".
    - Good design is as little design as possible
      Consistent with being as simple as current and foreseeable constraints will allow.
    - Good design is long-lasting
      A robustly documented product should make the design long-lasting.
    - Good design makes a product understandable
      Simplicity, and robust documentation should make the product understandable.