SOFTWARE ARCHITECTURE

Kevin P Dyer
The goal of this lecture

• To get you to think about your architecture before you start coding
What do we mean by architecture?

- Structure or structures of a system
  - Internal or external
- The relationship between the structures of the system
How do we measure a design?

• Is the customer happy?
  • Understand requirements and communicate with the customer early and often

• But there is more…
How do we measure a design? (cont.)

• Usability
  • Important for a low-level API or a GUI (is your interface intuitive?)

• Integrity
  • Data or resources should not be modified without authorization (does your program require root?)

• Availability
  • All required components of the system should function correctly and respond quickly (is it fast?)

• Confidentiality
  • Data or resources should not be exposed to unauthorized persons (how do you interface with your primitive data structures?)
Case Study #1: An iPhone app

• You are responsible for the development of an iPhone app for a major bank

• Requirements:
  • User must be able to view account balance
  • Bank must be able to send messages to users
  • There should be no ‘write’ functionality
    • For example, the user should not be able to transfer money from their account to an external account
Case Study #1: An iPhone app

- Where do you start?
- What does the infrastructure look like?
- What could go wrong?
- Consider distribution of work between server and client
Case Study #2: GPS

• You are responsible for the development of a GPS application

• What types of questions should we ask about the requirements?
Case Study #2: GPS
Case Study #2: GPS

• Where do you start?
• What does the infrastructure look like?
• What could go wrong?
• Consider distribution of work between server and client
Case Study #3: A High Traffic Website

• You are responsible for the development of a major website

• Requirements:
  • 10M/users a month
  • Users will register and store personal data
Case Study #3: A High Traffic Website

• Where do you start?
• What does the infrastructure look like?
• What could go wrong?
What is important in high level design?

• Understand how the customer will use the software
  • This is hard because the customer may not know

• However, don’t plan too much
  • We can always redeploy software

• Find a balance between the two
In practice: What works?

- Continuous Integration
- Unit Tests
- Prototyping
- Regular communication (internal + external)
- Code reviews
In practice: What doesn’t work?

• A lack of requirements
• Inconsistent coding practices amongst team members
• High level requirements mean nothing if they are ignored on a daily basis (don’t ignore your initial planning!)