MICHIGAN STATE UNIVERSITY

2. Technical Specifications

CSE 498, Collaborative Design

Wayne Dyksen
Brian Loomis

Department of Computer Science and Engineering
Michigan State University

Fall 2005

Overview

- What is architecture?
  - How do I get started on the project?
- Case study driven discussion
- Thinking like an architect
  - How to break it down

1-2

What is programming?

- Programming is both the act of applying a logical approach to solving a problem and writing that logic down in a form that the computer understands
  - Programs solve problems and are tools, expressing our imagination about the problem
- Writing programs is more like writing mathematical proofs than building a house
  - Each developer has a toolbox of constructs
  - Apply the constructs iteratively in steps to create a solution to the problem
  - Each program is a unique solution but not necessarily the only one

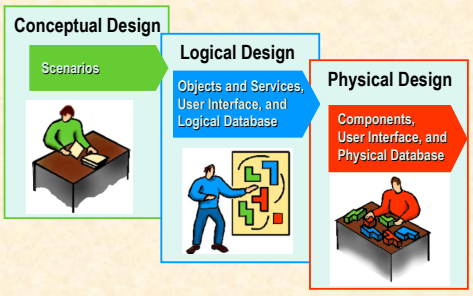
1-3

In the early part of a project…

- Requirements gathering and understanding
  - State the problem unambiguously
- Architecture and design
  - Determine your approach to the problem
- First working prototype
  - Test your hypothesis
- Feature complete build
  - Formalize the proof
- Ship it!
  - Turn it in!

1-4

Design Process Overview



1-5

Starting with something like…

Auto-Owners Insurance: Web-Based Bond Quote Tool

- The project will be to build some web applications that will be used to quote bonds over the internet for our independent agencies. Examples are bonds needed for road construction, notaries, and a bunch of other unrelated items. Another example is jail bonds but we do not write those. Tom Hallead is the person at our company who will work with our MSU team.

1-6

Understanding requirements

- Capture the end-user or project sponsor's intent
- Starting with…
  - Often no formal docs, maybe only a shared vision
  - Maybe an incomplete problem statement
- Ending with…
  - Formal phrasing of requirements in a document
    - Defined scope of what is in and not in the solution (boundary conditions and features not included)
    - User scenarios or experiences
  - Validated initial schedule, cost, and risk (things not yet known)
- Approach…
  - Identify business entities (objects) and relationships
  - Remove ambiguity and logical/business inconsistencies
  - Validate business rules and assumptions

1-7

Possibly a vision document

Content		Purpose
Problem statement	→	Why you want to do it
Vision statement	→	What you want the product to be
Solution concept	→	What you will do
User profiles	→	Who will use the product
Business goals	→	What you want to accomplish
Design goals	→	How you plan to accomplish it

1-8

Another example: Matrix

- The scope of this project is to create an online publication editor that would allow users to associate multimedia objects with their own writing in more complex ways. ***This might include*** such things as embedding objects within text, dynamic slide shows, and time-synced presentations of multiple objects. Although the development platform for this project is ***negotiable***, MediaMatrix at this point utilizes PHP, MySQL, XML, XSL, and SMIL as its development environment.

1-9

Architecture and design

- Start translating the requirements into a plan and logical design that can be implemented as a program to solve the problem
- Starting with…
  - Requirements and user scenarios
- Ending with…
  - **Technical (or functional) specification**
    - Architecture of solution
      - User interface mockup
      - Interfaces to other systems or data formats
      - Entity/object model for system (pseudo-code for business data rules and functions)
      - Data schema
    - Identification of core feature set for the prototype
  - Test plan, schedule, risk analysis
- Approach…
  - Break a big problem into lots of little problems
  - Identify all the moving parts and interactions

1-10

Contents of a technical spec

Content		Purpose
Vision summary	→	What you want the product to be, justification for it, and key high-level constraints
Design goals	→	What you want to achieve with the product
Requirements	→	What you require from the product including "non-functional" requirements like reliability, scalability, security, etc.
Usage summary	→	When the product will be used and who will use it
Features	→	What exactly the product does, user interface mockup, event models, object diagrams (and use cases), data schema
Dependencies	→	Other factors the product depends on (external interfaces and compatibility)
Schedule	→	Key dates and deliverables
Issues	→	What risks might impact the project
Appendixes	→	Network topology, deployment plans, dev environment setup

1-11

Examples

- Quantum, Empowernet
- Last semesters…

- Every good spec is different but similar
  - Make sure you are complete
    - All tiers of the architecture – DB, classes, UI
    - Topology
    - Standards
  - Make sure you use two models at least
    - Network diagram and class descrips
    - Use cases
  - Ask yourself if you could be the tester on this system

1-12

Architecture constraints

- Communication
  - Speed: Ethernet, GigE, 802.11b/g, or dialup
  - Protocol: TCP/IP, IrDA, POTS
- Topology
  - One machine versus multiple interacting
  - External systems
- CPU speed
  - PDA, Itanium server, mainframe
- Memory availability
- Device-specific parameters
  - PDA display size or ink on TabletPC
- Legacy support or previous versions of the current app

1-13

Architecture tradeoffs

- Complexity
  - Number of technologies in use
  - Design patterns vs. execution speed
  - Number of tiers or subsystems
- Fully-custom, semi-custom, or off-the-shelf
  - Platform (OS, servers, SDKs, ++)
  - Language and compiler choice
  - Project type choice
- Appropriate technology
  - Reusable modules
  - Special-purpose languages
  - Community support
- Tools and process
  - How automated a process do you need?
  - How do you communicate designs? (UML, ORM, etc.)

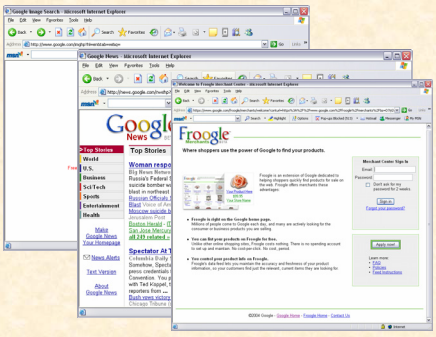
1-14

A Basic Modeling Process

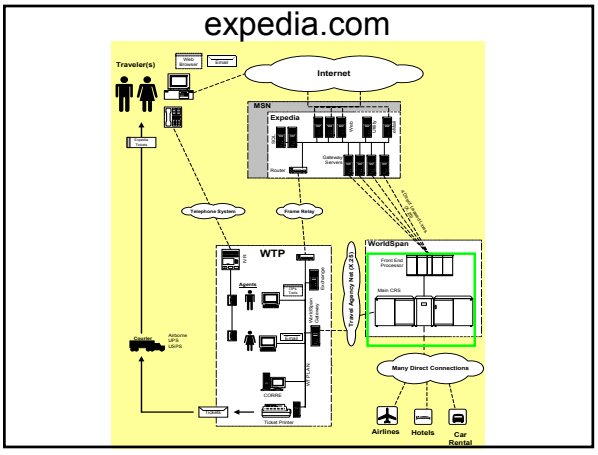
1. Identify logical entities (the "things" and their operations) from the requirements statements
2. Make these into variables with specific data types
3. Identify starting point for problem (what do I want to solve?)
4. Determine what entities and logical operations you need from this starting point and relationships between objects
5. For each logical operation, determine inputs and outputs and types of looping constructs (pseudo code)
6. Analyze entities for non-functional requirements – exceptions, security… per coding standards

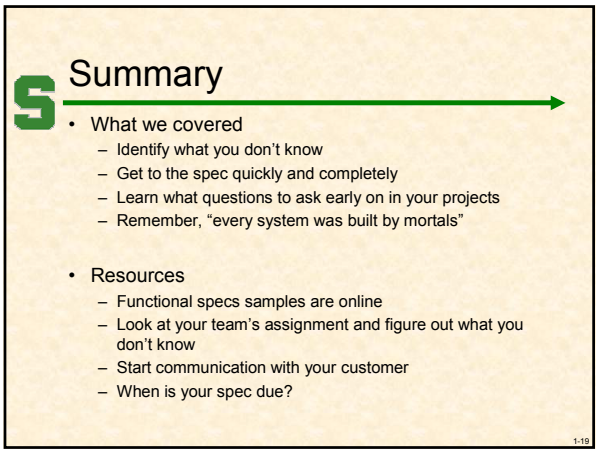
1-15

Bigger examples: Google



1-16




Summary
• What we covered
– Identify what you don't know
– Get to the spec quickly and completely
– Learn what questions to ask early on in your projects
– Remember, "every system was built by mortals"
• Resources
– Functional specs samples are online
– Look at your team's assignment and figure out what you don't know
– Start communication with your customer
– When is your spec due?
1-19