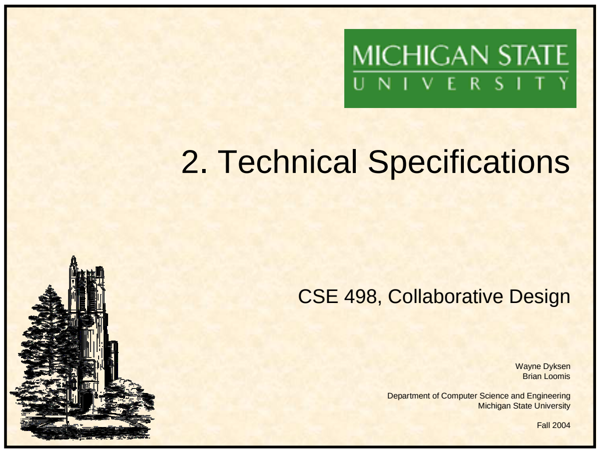


MICHIGAN STATE UNIVERSITY

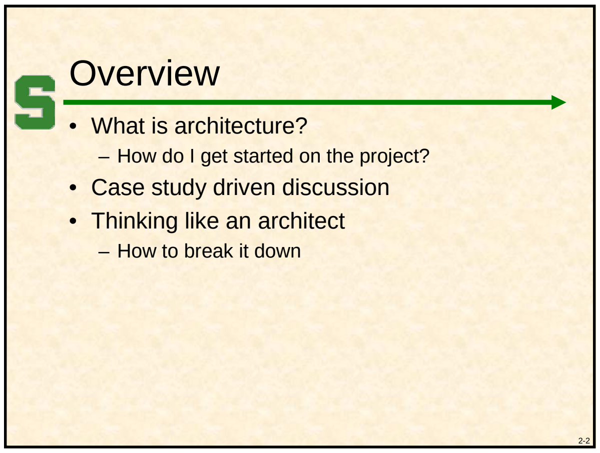
# 2. Technical Specifications

CSE 498, Collaborative Design

Wayne Dyksen
Brian Loomis

Department of Computer Science and Engineering
Michigan State University

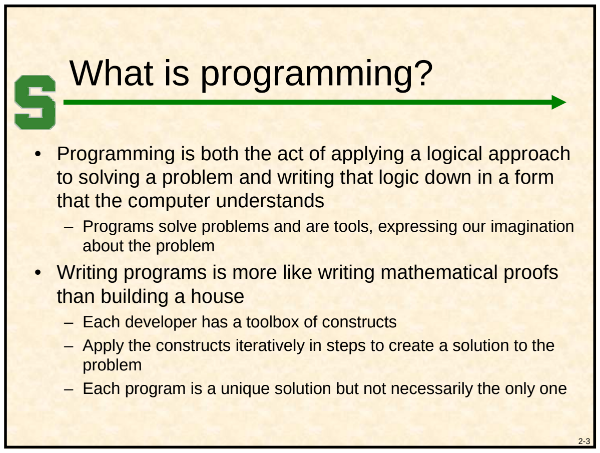
Fall 2004



## 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

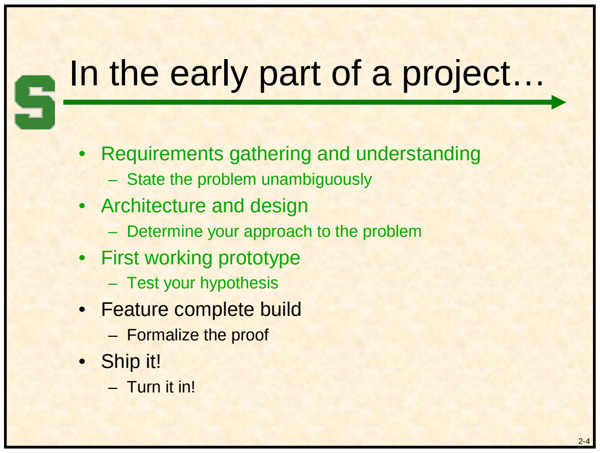
2-2



## 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

2-3

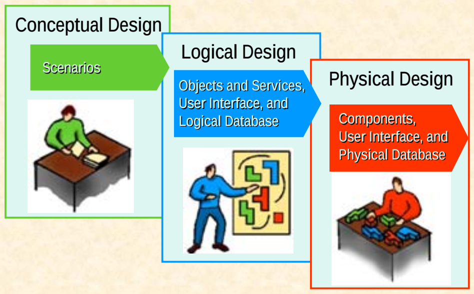


## In the early part of a project…

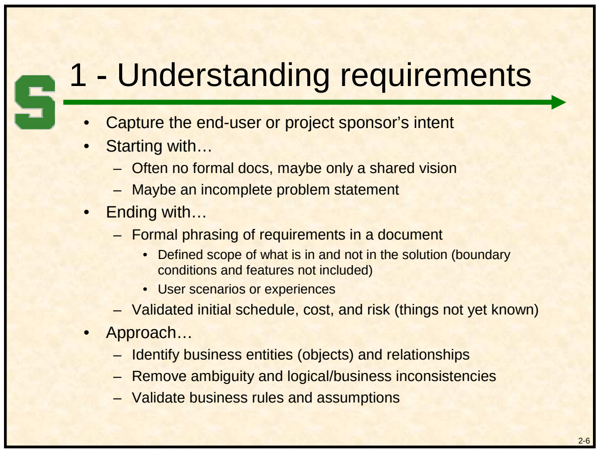
- Requirements gathering and understanding
  - State the problem unambiguously
- Architecture and design
  - Determine your approach to the problem
- First working prototype
  - Test your hypothesis
- Feature complete build
  - Formalize the proof
- Ship it!
  - Turn it in!

2-4

## Design Process Overview


Conceptual Design
Scenarios
Logical Design
Objects and Services, User Interface, and Logical Database
Physical Design
Components, User Interface, and Physical Database

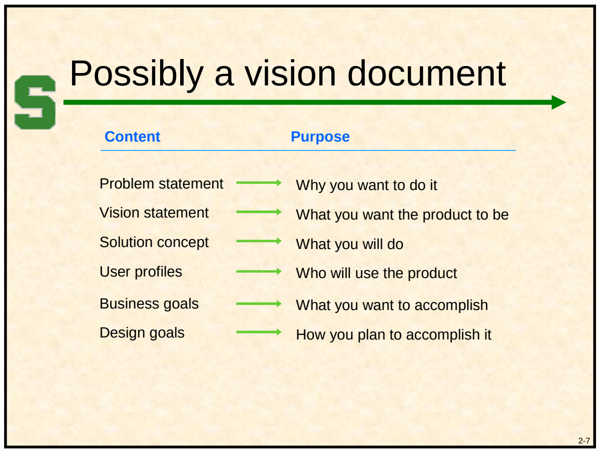
2-5



## 1 - 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

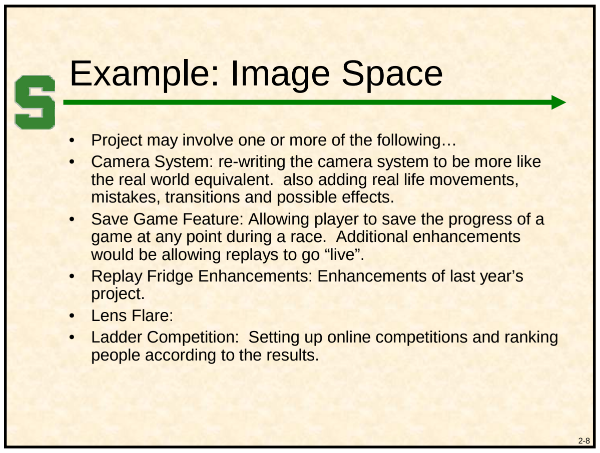
2-6



## 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 |

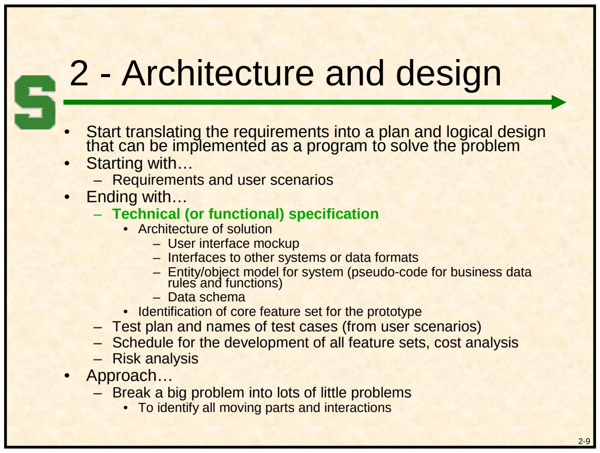
2-7



## Example: Image Space

- Project may involve one or more of the following…
- Camera System: re-writing the camera system to be more like the real world equivalent. also adding real life movements, mistakes, transitions and possible effects.
- Save Game Feature: Allowing player to save the progress of a game at any point during a race. Additional enhancements would be allowing replays to go “live”.
- Replay Fridge Enhancements: Enhancements of last year’s project.
- Lens Flare:
- Ladder Competition: Setting up online competitions and ranking people according to the results.

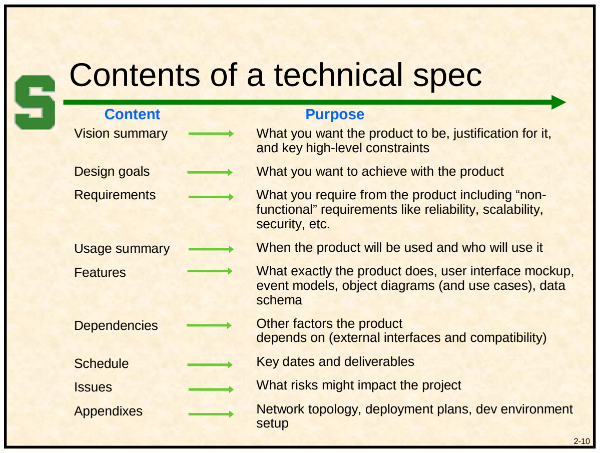
2-8



## 2 - 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 and names of test cases (from user scenarios)
  – Schedule for the development of all feature sets, cost analysis
  – Risk analysis
- Approach…
  – Break a big problem into lots of little problems
    - To identify all moving parts and interactions

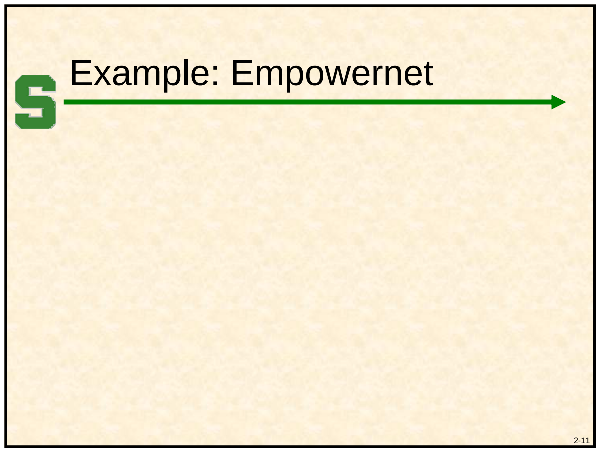
2-9



## Contents of a technical spec

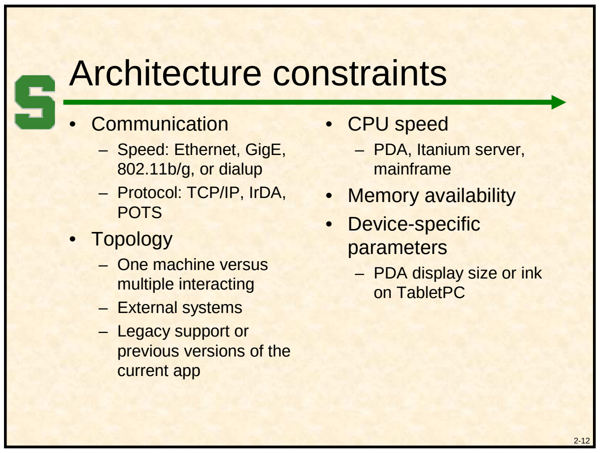
| Content | Purpose |
|---|---|
| Vision summary | What you want the product to be, justification for it, and key high-level constraints |
| Design goals | What you want to achieve with the product |
| Requirements | What you require from the product including “non-functional” requirements like reliability, 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 |

2-10



## Example: Empowernet

2-11



## Architecture constraints

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

2-12

## 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.)

2-13

## A Basic Modeling Process

1. Identify logical entities (the "things" and their operations) from the requirements statements
   - "Noun analysis"
   - And implied entities; also group the entities if there are a large number (for namespaces later)
2. Make these into variables with specific data types (integer, long, boolean, custom object)
3. Identify starting point for problem (what do I want to solve?)
4. Determine what entities and logical operations on each the starting one needs to complete the solution
   - Identify relationships between entities (1:1, 1:N, "is-a")
5. For each logical operation, determine inputs and outputs and types of looping constructs (pseudo code flow)
   - If I have all the inputs and know the output, then I should be able to compute the small problem
6. Analyze entities for non-functional requirements – exceptions, security… per coding standards

2-14

## Building a prototype

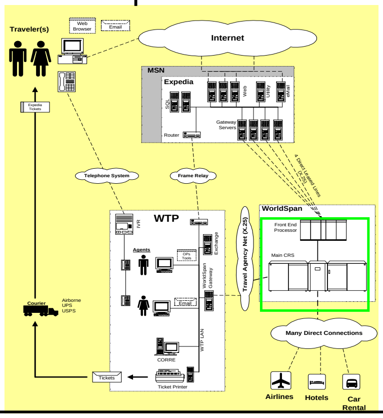
- Define initial scope of prototype
  - Which user scenarios are high risk and need more definition?
  - Which user scenarios exercise most of the entities?
- Build the prototype
  - Start all of the assemblies (shells)
  - User scenarios converted to screen layouts
  - Business entities converted to code objects and data schema elements
- Complete the specification
  - Objects replace entities, now have methods and members
    - Relationships added
    - User interface elements completely defined for all scenarios – UI has methods (events) and members (controls) and relationships (redirection)
  - Database schema added
  - Design standards agreed on

2-15

## Bigger examples: Google

2-16

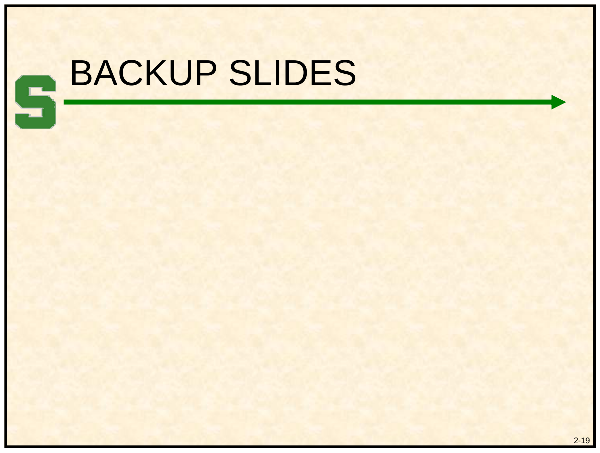
## expedia.com



## Summary

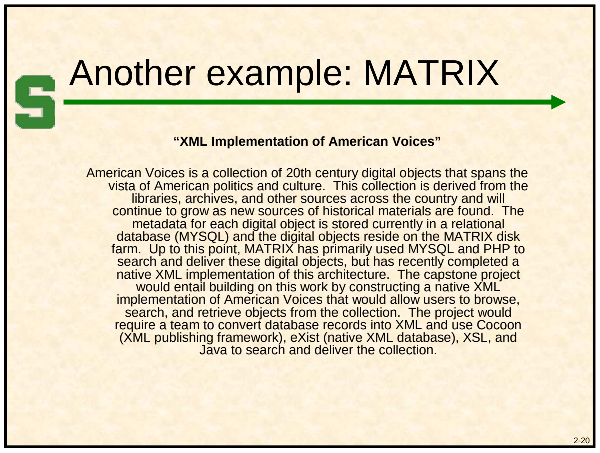
- What we covered
  - Identify what you don't know
  - Get to the spec quickly and completely
  - Every system was built by mortals
  - What questions to ask early on in your projects
- Why its important
- Resources
  - Functional specs samples are online
  - Look at your team assignments and figure out what you don't know
  - Start communication with your customer and building a spec

2-18



## BACKUP SLIDES

2-19



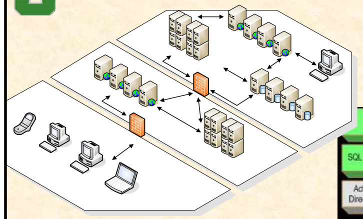
## Another example: MATRIX

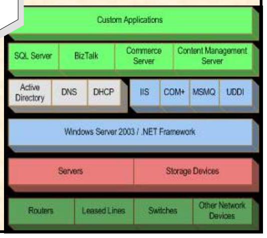
**"XML Implementation of American Voices"**

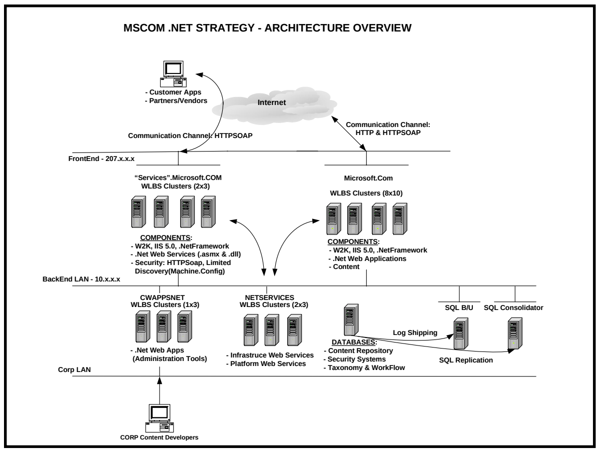
American Voices is a collection of 20th century digital objects that spans the vista of American politics and culture. This collection is derived from the libraries, archives, and other sources across the country and will continue to grow as new sources of historical materials are found. The metadata for each digital object is stored currently in a relational database (MYSQL) and the digital objects reside on the MATRIX disk farm. Up to this point, MATRIX has primarily used MYSQL and PHP to search and deliver these digital objects, but has recently completed a native XML implementation of this architecture. The capstone project would entail building on this work by constructing a native XML implementation of American Voices that would allow users to browse, search, and retrieve objects from the collection. The project would require a team to convert database records into XML and use Cocoon (XML publishing framework), eXist (native XML database), XSL, and Java to search and deliver the collection.

2-20

## Two Views of Enterprise Apps






MSCOM .NET STRATEGY - ARCHITECTURE OVERVIEW
- Customer Apps
- Partners/Vendors
Internet
Communication Channel: HTTPSOAP
Communication Channel: HTTP & HTTPSOAP
FrontEnd - 207.x.x.x
"Services".Microsoft.COM WLBS Clusters (2x3)
Microsoft.Com
WLBS Clusters (8x10)
COMPONENTS:
- W2K, IIS 5.0, .NetFramework
- .Net Web Services (.asmx & .dll)
- Security: HTTPSoap, Limited Discovery(Machine.Config)
COMPONENTS:
- W2K, IIS 5.0, .NetFramework
- .Net Web Applications
- Content
BackEnd LAN - 10.x.x.x
CWAPPSNET WLBS Clusters (1x3)
NETSERVICES WLBS Clusters (2x3)
SQL B/U
SQL Consolidator
Log Shipping
- .Net Web Apps (Administration Tools)
- Infrastruce Web Services
- Platform Web Services
DATABASES:
- Content Repository
- Security Systems
- Taxonomy & WorkFlow
SQL Replication
Corp LAN
CORP Content Developers

Web Servers

Application Servers

Web Servers

Interna
Clients

Databa