Jim Fawcett
CSE681 – Software Modeling and Analysis
Spring 2010

SOFTWARE SYSTEM TAXONOMY

Agenda

- Taxonomy – An organization or catalogue
- Software Systems Taxonomy – parts of a catalogue of models:
  - System Structure
  - Software Studio Examples
  - CSE681 Project #5 Examples
  - MS Thesis Research Examples

The gap between theory and practice
   in theory
   is nowhere near as big as
the gap between theory and practice
   in practice

Structuring Paradigms

- Computational
  - Example - Scientific computing
  - Focus on answers and views
  - May be distributed by function but probably not by machine
- Event-Driven
  - Examples - User Interfaces, Servers, Security
  - Focus on state and state changes
  - User Interfaces and (semi) Real Time systems
- Service Oriented
  - Examples - Communication, Business services
  - Focus on reliability and performance
  - Usually network or web based

Software System Structures

- Client-Server
- Three-Tier
- N-Tier
- Layered
- Peer-to-Peer
- Collaborative
- Service Oriented
- Agent Based

Client-Server

- Client initiates, server responds
  - Servers are passive
  - Only provide replies to specific request types
- Server provides a service
  - Web server, file server, network storage
- Examples:
  - Web sites – Amazon, ecs.syr.edu
  - Web services – google maps
  - http://www.ecs.syr.edu/faculty/fawcett/handouts/CSE681/Presentations/IntroductionToWeb.ppt

Three-Tier (really four)

- Presentation
  - What user sees, may have many distinct views
  - Initial rendering determined solely by server
  - Client (Javascript and Ajax, for example) can provide subsequent local as well as server activity
- (Often Implicit) Control
  - Responds to user inputs
  - Routes events to handler actions
- Application (object-based)
  - Implements a model of what the user views and manipulates
- Data (usually table-based)
  - Manages creation, retrieval, update, delete (CRUD)

Presentation and Control

- Desktop
  - Windows forms, WPF, Java Swing, gtk+
- Web Application
  - Asp, Asp.net, Asp.net MVC, Silverlight with WPF, Java with Servlets, …
- Mobil Application
  - Thin versions of the Web Application technologies
- Ajax
  - Round-trip data transmission mechanism, orthogonal to the above.

WinForms, WPF, Asp.Net

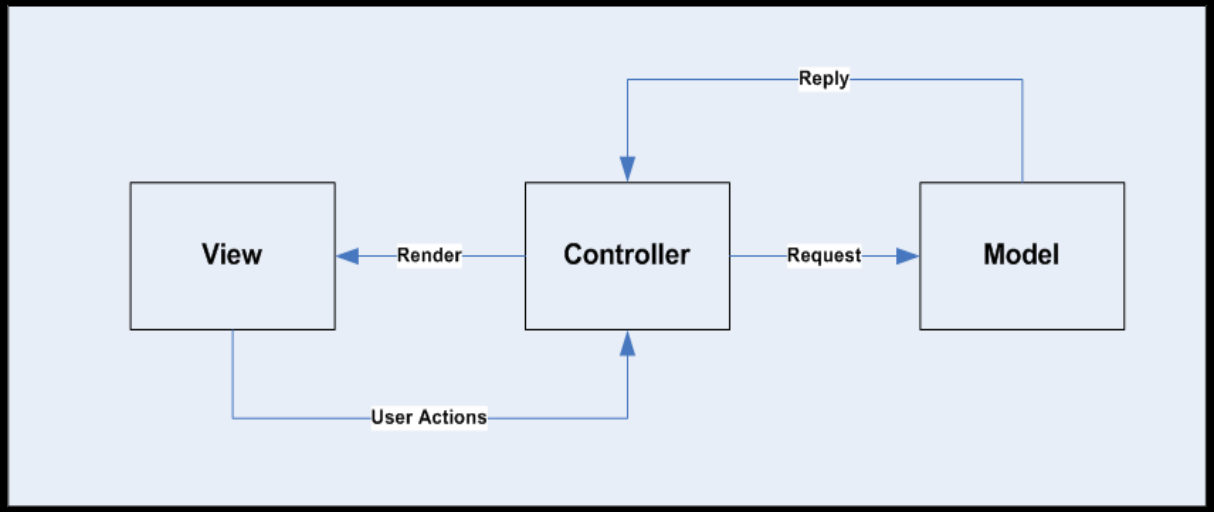
- Make control implicit, and encourage tight binding
- Two kinds of binding
  - Bind view tightly to event handling – one to one correspondence between events and handlers.
    - Different views may need the same event handling, but it is hard to share event handlers across views.
  - Bind directly to data in event handlers.
    - WPF has a lot of infrastructure to support binding controls to data.
- But we may have many views, application modes, and data sources.
  - Tight binding makes it hard to avoid repeating code.

Separation of Concerns

- Except for the simplest of applications it's not a good idea to bind presentation, control, and data together.
  - There often are many views, more than one application mode, many sources of data.
  - If we bind these all together we get spaghetti
    - Very hard to test, hard to maintain, hard to document.

Model-View-Controller

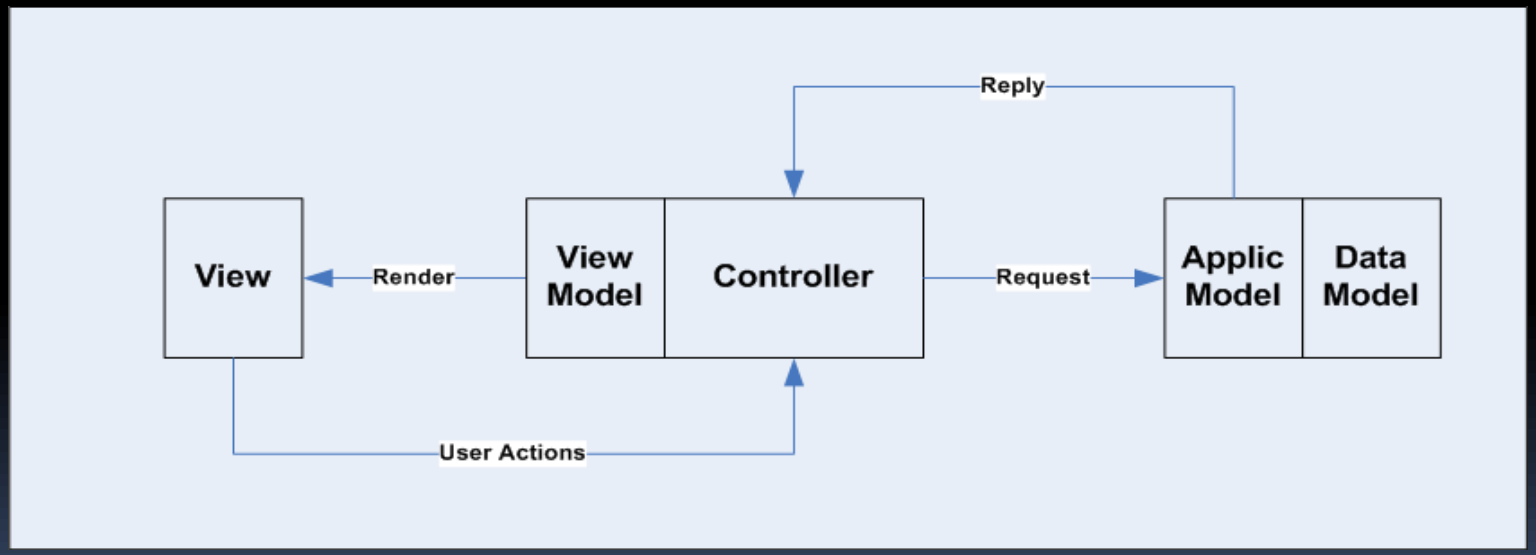
- MVC Separates concerns:



- Directly supports multiple views and multiple application scenarios
- Users request actions, not resources
  - Give me this view into model, not this web page

MVC – More Realistic

- Views and Models usually have some substructure, e.g.:



View – View Model

- A view is what gets rendered
- A view model is an abstraction that:
  - Defines resources that many be used in several places.
  - Defines styles that may be used in several places
  - Defines an object model for the application to manipulate
- In some implementations of MVC:
  - Controller updates the model
  - View subscribes for update events from the model.

Application vs. Data Models

- Application model
  - Defines classes for all the entities a user knows and cares about, e.g., orders, customers, products, etc.
- Data model
  - Defines wrapper classes for tables and stored procedures
  - Manages connections
- Object to Relational Mapping
  - Relationships between application objects and data objects.

Applications of MVC

- Asp.Net MVC (Web application, .Net environment)
  - Released as part of a service pack for .Net 3.5
  - Is an official part of .Net 4.0 framework with project wizard available in Visual Studio 2010.
- Microsoft Composite UI Application Block (Desktop, .Net)
  - http://richnewman.wordpress.com/2008/02/23/model-view-controller-explained-introduction-to-cabscsf-part-22/
  - http://msdn.microsoft.com/en-us/library/aa480450.aspx
- JavaEE6 (Web applications, java environment)
  - http://java.sun.com/javaee/
  - http://programmaremobile.blogspot.com/2009/01/mvc-design-pattern-in-java-ee-eng-ver.html
- http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

N-Tier Structure

- So, the three tier MVC has morphed into a five tier V-VM-C-AM-DM
  - View – what gets rendered
  - View Model – an abstraction of the view
  - Controller – routes View events to handlers in the Application Model
  - Application Model – classes that model the "business" logic
  - Data Model – models data storage tables
    - Database, XML file, custom data structures

Layered Structure

- Provides a structure based on:
  - System Services – things the user doesn't think about
    - Communication, storage, security, file caching, …
  - User Services – things the user manipulates as part of the use of the system
    - Input, Display, Check-in/Check-out, …
  - Ancillary – Things that are not part of the system mission but are necessary
    - Logging, extension hooks, test hooks, …

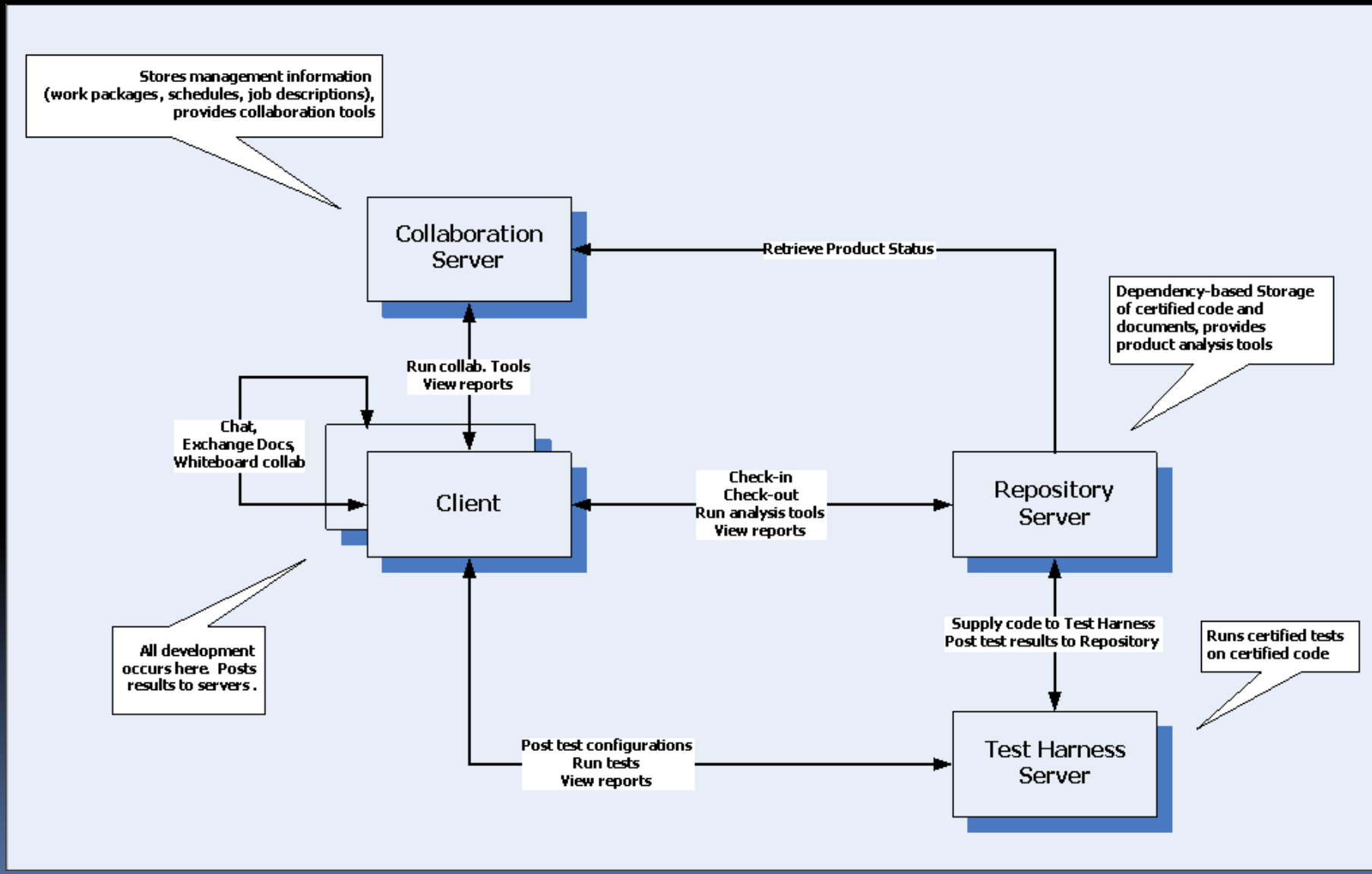
Peer-to-Peer

- Distribution of parts that cooperate on a mission by sending each other commands and messages.
  - Parts may or may not be identical, but probably have identical layered system services
  - Usually part of a collaboration system
  - May have a "distinguished" peer
  - Development attempts to provide one set of core services and build peer personalization on top of that
- Example:
  - Software Matrix, Gosh M.S. Thesis, http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/softwarematrix.htm

Collaboration System

- System that focuses on sharing of processes and products among peers with a common set of goals.
  - Primary focus is organizing and maintaining some complex, usually evolving, state:
    - Software development baseline
    - Set of work plans and schedules
    - Documentation and model of obligations
    - Communication of events
- Example:
  - Collab – CSE784, Fall 2007, http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/CServ.htm

Example Collaboration System



Service Oriented

- System composed of
  - Set of autonomous services
  - Software glue that binds the services together
- Focus on
  - Reliability, availability, compos ability
- Example:
  - VRTS – CSE784 Project, Fall 2008, http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/Vrts.htm

Agent-Based

- System uses Software Agents
  - Semi-autonomous, mobile, task oriented software entities
  - May be scheduled
  - Provide scriptable user specific services
    - Collect information from a large set of data
    - Perform analyses on changing baseline and report
    - Conduct specific tests
    - Make narrowly specified modifications to baseline
- Example:
  - CSE681 Project #5, summer 2009, http://www.ecs.syr.edu/faculty/fawcett/handouts/CSE681/Projects/Pr5Su09.doc

Enterprise Computing combines Structures

- Enterprise computing binds together a business with its partners, suppliers, and customers.
- May integrate many functions:
  - Inventory control, order processing, product disclosure, product design collaboration.
- Likely to be peer-to-peer with "distinguished" peer that coordinates activities.
  - Partners work together through a collaboration subsystem.
- Uses web-based service oriented architecture.

Project #5

- Peer-to-peer?
  - May initiate analyses from client
  - May schedule analyses and notify users of results
- Collaborative?
  - QA, Management, Developers, and Architects all care about the analyses and results.
  - How do we overtly support collaboration?
- Service Oriented?
  - Communication and Notification are probably service-based
- Layered?
  - If we extend by sending libraries to remote machines to be run from tool holster, we may want to have the holster provide execution services – a sandbox – to enhance security
- Agent-based?
  - We probably want to schedule tests, tailored to specific users, e.g. QA, team lead, architect.

Software Studio Examples

- All of these were designed, built, and delivered by CSE784 classes.
  - VRTS – Virtual Repository Testbed Servers, Fall 2008, http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/Vrts.htm
  - Cserv – Collaboration Server, Fall 2007 http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/CServ.htm
  - RSA – Remote Software Assistant, Fall 2006, http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/RSA/rsa2006.html
  - RTBS – Repository Testbed System, Fall 2005, http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/RepoTB.htm

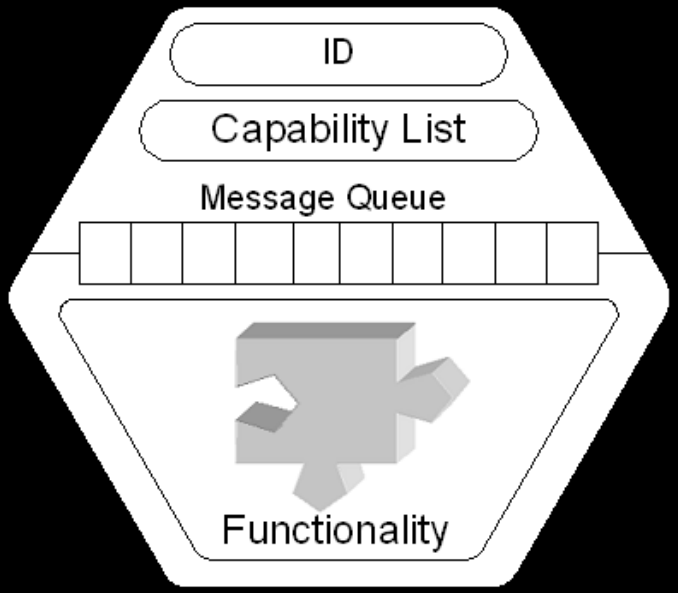
CSE681 Project #5 Examples

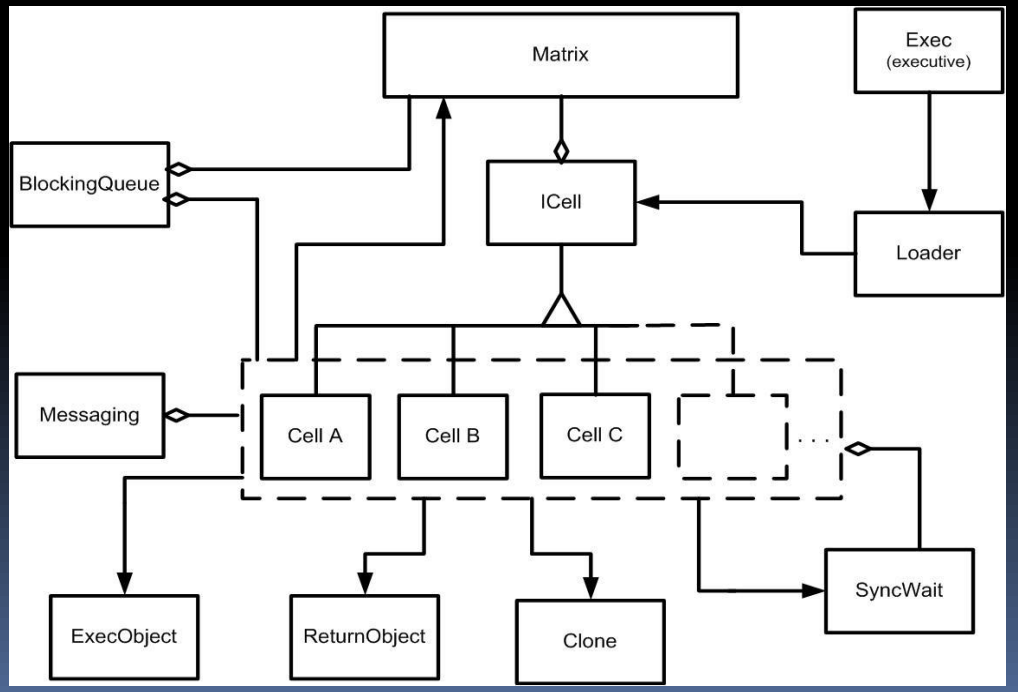
- **ABSA** – Agent Based Software Assistant, Su09
  - Agents running on Repository, Testbed, and Tools
- **VDS** – Virtual Display System, Fo8
  - Large display system driving Repository, Testbed, Collaboration servers
- **ABQATS** – Agent Based Quality Analysis and Test System, Su08
  - Agents supporting Test System
- **Cserv** – Collaboration Server system, Fo7
  - Collaboration server with repository and testbed in context.
- **ADSCS** – Agent based Distributed Software Collaboration System, Su07
  - Agent based support for collaboration

Master's Thesis Research Examples

- The following are all based on Software Matrix structure – Autonomous cells often used with mediator
  - Software Matrix – Gosh, 2004
  - Self Healing Systems – Anirudha, 2005
  - Cross Platform Development – Appadurai, 2007
  - Model-Driven Development – Patel, 2007
- http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/research.htm

Software Matrix Concept





The End