Course Sub-Topics

Getting started with Research

◆ Finding Resources
  – Wayne Powell on using Tisch Resources
  – Internet resources
    » Google
    » Citeseer
    » Researcher home pages
    » Wikipedia only for ideas and leads
      ◆ never for reference
      ◆ Lifetime of Web references not generally guaranteed
Project Guidelines

- May be hands-on or theoretical
- If to be used as MS project then must
  - Be hands-on
    » (unless your job includes software development)
  - Include extended project report
- Topic
  - Related to course goals
  - Submitted to Judy and Rick before February 3
- Project Proposal to class on 2/6 or 2/13
- Class Presentation/Demo on 4/16 or 4/23
- See project guidelines for more details

Project Proposal

- **Description of the type of project**
  - Hands-on or Theoretical
- **What problem are you investigating – what do you want to learn?**
- **The end goal?**
- **How you will show that the end goal has been achieved – How can you convince the class that you have learned what you wanted to?**
Paper Presentation Guidelines

◆ For Presenter
  – Read assigned paper two weeks before presentation date
  – Create question to use as a starting point for class discussion
    » have it posted to website by Judy or Rick one week in advance
    » should be answerable in no more than 250 words
  – Explore topic more deeply with focus on how it relates to course goals
  – Prepare 30 minute presentation for class
    » Overview of assigned reading
    » Your view of how it relates to course goals
    » Discussion of general topic area in as much as it relates to course goals
  – Present to Class
  – Lead Class Discussion on Topic
◆ For rest of class
  – Read the paper and be prepared to answer question

Separation of Concerns and Modularity

Meeting 5
Separation of Concerns

“Identify, encapsulate, and manipulate only those parts of software that are relevant to a particular concept, goal, or purpose”

- An underlying goal in modern software engineering
- Not everything is important all the time
  - Importance depends on perspective
  - Often it is better to ignore some details
- Separate concerns are related
- For instance, topics in this class

Modularity

- Early attempt at achieving separation of concerns
- Programming-in-the-large vs. programming-in-the-small
  - Decompose
    » Divide system into concerns to be implemented
  - Structure
    » Create systems by connecting concerns (e.g., module interconnection languages)
- Enables additional forms of expression and verification
- Applies to both logical and physical entities
Relevance

- Desire for separation of concerns created need for new abstractions and modeling
  - Modularity tried to address that need
  - Aspect-oriented programming
  - Component Frameworks and Containers
- Still not resolved
  - OSGi, JSR 277, & JSR 294 for Java
  - .NET has Assemblies, but still not sufficient

Product Line Architectures, Configuration Management, and Software Deployment

Meeting 6
Product Lines

“A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.”

- Supports improvements in time to market, cost, productivity, quality, and other business drivers.
- Enables rapid market entry and flexible response.
- Provides a capability for mass customization.

Products

- Configuration of one or more products from a collection of related product families
  - Examples
    » 32” Flat screen, HD TV with...  
    » 19” Tube TV/DVD player
- When is code configured
  - Compile time?
  - Product ship time?
  - Ever...?
- Run-time Configuration not finalized until set up in consumer’s home
Product Line Architectures

- **Elements of PLA**
  - Components
  - Connectors
  - Variation points
  - Binding times
- **Support rapid product customization**

Configuration Management

- **Managing evolution in software systems**
- **Goal:**
  - Keep software system in a well-defined state
  - Support long term maintenance and evolution
- **More than versioning**
  - Models system components and structure
  - Manages development, people, and process
Software Deployment

◆ A system isn't worth much if we don't get it in the hands of the customer
  – CM manages valid system configurations
  – Deployment makes a valid configuration usable
◆ More than just getting the software there
  – Managing evolution of deployed configuration
◆ Requires system modeling from a different perspective
  – Potentially the end user's perspective

Relevance

◆ Lots of models being created
  – Separate, but related concerns
  – What the relationships among them?
  – How do we avoid research islands?
Architecture Description and Architectural Styles

Meeting 7

Architecture Description

◆ ADLs and other notations support description of
  – Elements
  – Configurations
  – A given ADL supports capturing a specific system abstraction

◆ Documentation
  – Description
    » Elements
    » Configurations
  – Rationale
  – Related doc
Koala PL-ADL from Philips

Architecture Documentation

- **Describe**
  - Abstractions useful to ensure requirements will be met by any system that implements the architecture
    - Code (Module view(s))
    - Run-time (C&C view(s))
    - Allocation

- **Rationale**
  - Why where the elements selected?
  - Why are they configured the way they are?

- **Pointers to Related docs (requirements, standards, etc.)**

- **Dictionary of terms and acronym list**
Architectural Styles

- Known solutions to recurring problems
- Describe
  - Types of elements
  - Types of relations
  - Constraints on their interrelationships
  - Description of which quality attributes it is good for and which it is bad for
  - Example of its use
- Ex: Layered Style (in Module category)
  - Elements are layers
  - Relations are “allowed to use”
  - Constraints
    » A module is allowed to use modules at lower level
  - Supports evolution, maintainability
Software Components

- **Software components**
  - Commercial
  - Adapted from previously used software
  - In-house development

- **Issues**
  - Who’s in control
  - When is component bound
  - Mismatched assumptions
    » By component about environment
    » By environment about component

Relevance

- **What different types of software components exist?**
- **What are the distinctions among them and what is the importance of the distinctions?**
  - Architectural components
  - Commercial Components
  - Services
Service-Oriented Architecture

Meeting 9

- Commonly viewed as a specific approach
  - Generically, it is a style
    » Provider, consumer, possibly a broker
- Functionality is abstracted into "services"
  - Units of functionality available for others to use
- Provides
  - Implementation substitutability
    » Based on requirements
  - Loose coupling
  - Late binding
  - Location transparency
Relevance

◆ Another model
  – For discovery and interaction
◆ Implementation separation of concerns
◆ Fits nicely with components and dynamism
◆ Interface-based approach has popularized inversion of control/dependency injection
  – e.g., Spring, PicoContainer, HiveMind, EJB3
  » Trend toward POJOs

Autonomic Computing,
Self-Healing Systems,
and
Context Awareness

Meeting 10
Autonomic and Self-Healing Systems

◆ Manage themselves
  – Self-configuring
  – Self-optimizing
  – Self-healing
  – Self-protecting
  – Self-*
◆ Operate given only high-level objectives
  – Specify what is desired, not how to achieve it

Context Awareness

◆ Ubiquitous computing – computers everywhere, all the time
◆ Adapting to meet the needs/desires/intents of the user at any given time with respect to
  – User task, environmental conditions, social conditions
◆ High rates of dynamism
◆ Requires sophisticated modeling
  – Dependent on semantics and behavior
Relevance

- What are the correct models?
- How do these models help the system self-evaluate?
- How do these models fit with existing modeling techniques?

Project Brain-Storming

- Online doc building on german proto
- Deployment of dynamic application
- Hierarchical analysis
- Data-oriented IoC framework
- Data modeling withing the V&B approach
- Graphical view of a dynamic system
- Product line architecture for biomedical research
- Agile and architecture
- Open source and architecture
What’s Next

◆ Send top three paper preferences to Judy (jas@cs.tufts.edu). Assignment will be made as they come in.
◆ Send project ideas to Rick and Judy by Feb 3 (So we can exert our veto power if necessary)
◆ Prepare project proposal