

**GEORGE MASON UNIVERSITY  
DEPARTMENT OF COMPUTER SCIENCE**

SWE 721 / IT 821 - Reusable Software Architectures  
Spring 2008                      Wednesday 4:30 – 7:10 PM

Innovation Hall 222

Dr. Hassan Gomaa

703-993-1652

Science & Technology 2 Room 330

E-mail: hgomaa(at)gmu.edu

Fax: 703-993-1638

Office Hours: Wednesday 3:00-4:15 PM, by appointment, phone & e-mail

GTA: Erika Olimpiew <eolimpie(at)gmu.edu>

**Prerequisite:** SWE 620 and SWE 621, or permission of instructor.

**Course Description:**

This course investigates the software concepts that promote reuse of software architectures. The influence of object technology on software design and reuse is studied. Domain Modeling methods, which model the application domain as a software product family from which target systems can be configured, are investigated. The course also covers reusable software patterns including architecture patterns and design patterns, software components, and object-oriented frameworks.

**Course Content:**

Overview of software reuse. Designing reusable software, reusable components, Reusable Software Architectures: Software libraries, generic architectures, software product lines, modeling single systems and product lines with UML, model driven architecture.

Object-Oriented Software Life Cycle for Software Product Lines; Object-Oriented Requirements Modeling; Object-Oriented Analysis Modeling, Object-Oriented Design Modeling, Incremental software construction and integration.

Requirements Modeling for Software Product Lines. The use case modeling approach for defining functional requirements. Kernel, optional, and alternative use cases and actors. Modeling variability with use case parameterization, variation points, and extension points.

Feature Modeling for Software Product Lines. Feature as a reusable requirement. Common, optional, default, and alternative features. Feature dependencies. Feature groups – mutually exclusive, one and only one, one or more of a group. Feature Modeling with UML. Modeling features with use cases; relationship between features and use cases. Feature conditions.

Analysis Modeling for Software Product Lines. Static modeling: objects, classes, and relationships. Object and class structuring; class categorization using stereotypes. Kernel, optional, and variant classes.

Dynamic interaction modeling for Software Product Lines. Developing object interaction models for kernel, option, and alternative use cases. Developing interaction models for different scenarios addressing use case variability. Kernel First Approach. Software Product Line Evolution Approach.

Statecharts for Software Product Lines. Kernel, optional and variant statecharts. Mutually exclusive variant statecharts and co-existing variant statecharts.

High-level statechart to capture generalization of multiple variants. Modeling variability in statecharts through inheritance and parameterization.

Feature/class dependency modeling. Modeling commonality/variability with abstract classes and parameterized classes. Using inheritance to support variant classes. Feature-based impact analysis. Feature/class dependencies.

Architectural Patterns for Software Product Lines. Software architectural structure patterns, software architectural communication patterns, software architectural transaction patterns, documenting software architectural patterns, applying software architectural patterns in product lines.

Software Component-based Architectural Design for Product Lines. Developing the overall software architecture. Separation of concerns in component design. Component-based structuring criteria. Software components and connectors: ports, provided and required interfaces.

Software Application Engineering. Deriving individual members of the product line from the OO software product line architecture and components. Using the product line feature model to derive the requirements, analysis, and design models for the application.

Software Product Line Case Studies: Microwave Oven, Distributed Factory Automation, Electronic Commerce.

**Required Course Text:**

Hassan Gomaa, “Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures”, Addison-Wesley Object Technology Series, 2005.

<http://www.awprofessional.com/title/0201775956>

<http://www.aw-bc.com/catalog/academic/product/0,1144,0201775956-PRE,00.html>

**Recommended Course Text:**

F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, “Pattern Oriented Software Architecture: A System of Patterns”, John Wiley & Sons, 1996.

**Assignments:**

Term project and term paper.