Prerequisites, Overview, and Goals

This course explores leading edge paradigms for automated software design and software modularity from advances in:

- **Generative Programming** — programs are synthesized by other programs
- **Compositional Programming** — programs are assembled by composing prefabricated parts
- **Domain-Specific Languages (DSL)** — domain-specific notations that simplify program specification
- **Model Driven Engineering (MDE)** — programs are specified as and synthesized from models
- **Automatic Programming** — efficient programs are synthesized from declarative specifications
- **Program Refactoring** — common automated restructuring of programs
- **Program Analysis** — tools that evaluate properties of programs

A spectacular example of the integration of most of these areas was realized thirty-five years ago: **relational query optimization (RQO)**. A relational query is specified in SQL, a parser maps it to an inefficient relational algebra expression, a query optimizer optimizes the expression automatically, and an efficient query evaluation program is generated from the optimized expression. SQL is a prototypical **declarative DSL**. Query evaluation programs are specified as compositions of relational algebra operations; relational algebra is a prototype for **compositional programming**. Query optimizers achieve **automatic programming** by rewriting an inefficient expression/program to a semantically equivalent but more efficient expression/program. The cost models that drive expression optimization are examples of **program analysis**. Mapping a relational algebra expression to an efficient program is **generative programming** and is an elementary example of **model driven engineering**.

A "holy grail" of Software Engineering is to replicate the success of RQO in other domains. **Feature Oriented Software Development** is a generalization, and its ideas are at the confluence next-generation research topics in software modularity, program design and program synthesis: OO design, product-lines, program refactoring, model driven engineering, program evolution, and program transformations.
Prior offerings of this course lead to student publications and research degrees (M.Sc. and Ph.D.). Some publications -- not all -- are listed below, the most recent listed last.


### Programming Assignments

All programming assignments will use Java or Prolog. You can pick up what you need in Prolog, if you are unfamiliar with it. We will use the following software, all of which is free to UTCS students, and all of which has been installed in the UTCS public labs. The following is for Windows Platforms. If you use Apple or Linux machines, well, I will try to help as much as I can, but no promises!

- **NetBeans IDE** -- I prefer NetBeans to Eclipse for program development as it is a simpler environment.
  - on Linux boxes, Netbeans AND Eclipse are on the default path, just type "> netbeans" or "> eclipse"
- **Microsoft Visio 2010/2013** -- for drawing UML diagrams. If you are a CS student, you can gain access to Visio via Software Downloads, read the MSDN/AA agreement, click the secure download site link, and follow the instructions. For ECE students, do the same at the ECE-MSDN/AA web page.
- **VLC Media Player** -- to view .MOV videos shown in class. (You can use Windows Media Player, but its video resolution is awful). No need for this software if you are a Mac user.
  - on Linux boxes, there are several media players. **xine** is one and it is installed in /usr/bin
- **SWI Prolog** -- for writing metamodel constraints and model-to-model transformations.
  - on Linux boxes, you can run SWI Prolog as: /usr/opt/swipl-7.2.2/bin/swipl
- **FeatureIDE** -- A useful collection of integrated tools for creating feature-based product lines. I recommend that you download Eclipse with the FeatureIDE plug-in already installed.

Please note that some Java programs will be released and updated during the Semester. These programs are NOT installed on the UTCS public labs. You will have to install these updates yourself. Their links will be posted on assignments.

### Course Prerequisites

Basic familiarity with the following topics are assumed -- all that is needed will be covered, but some prior experience will help greatly:

- Java
- Compilers and Grammars
- Object-Orientation and UML
- Relational Databases

As mentioned above, Prolog will be used in assignments. I expect that you'll pick up what you need.

http://www.cs.utexas.edu/users/dsb/cs392f/#Schedule
Lecture Notes and Texts

Lecture notes will be presented online (after the lecture) as downloadable PPTX files. Links to the lectures are given below in the Course Outline. There is a required text for this course:

- B. Pierce, Basic Category Theory for Computer Scientists

The University Co-op has new copies of this text weighing in at $29. This text is now available as a kindle ebook from Amazon.

Hint: I am unimpressed with today’s ebooks. They are as expensive as hard copies and ebook software sucks. Your call.

Class Grades, Projects, and Homework

Final grades will be determined approximately by the following scheme:

1. Your accumulative programming assignment grade will determine the maximum final grade for the course. Ex: if you get a "B" average across all of your programming projects, your final grade will be no greater than a "B".
2. Final counts 35%; midterm counts 35%; classroom participation 15%; and class presentation counts 15%.

Each group of students will complete an approved project by the end of the class. A classroom presentation on every project is expected. Details on the projects will be announced later.

All programming projects are to be submitted through Canvas. A PDF file must be included on all assignments -- it lists your name and email addresses that are hyperlinked so that I can easily send my comments of your assignment back to you.

Extenuating Circumstances

If you have difficulty meeting the requirements of this course, fail to hand in an assignment, or miss an exam because of extenuating circumstances, please advise the instructor in writing at the earliest possible date so that your situation can be discussed. If you encounter an unexpected medical or family emergency or a random act of Nature that causes you to miss the due date for homework or miss a quiz or exam, you must present suitable documentation in writing to the instructor before special consideration will be given. A file of all written correspondence will be kept by the instructor and decisions regarding them will be made at the end of the semester.

Schedule

Numbers in [brackets] indicates the estimated number of lectures on a topic. The number indicated is a lower-bound, as there will be class room discussions to work on problems and review of homework assignments. Papers that are listed below are required readings and are accessible via its web link. The order in which topics are presented might be changed as the class progresses.

The syllabus on the first day of class is here (to be posted after the first class), as I do modify this page as the course progresses, like posting new assignments, readings, and lectures.

<table>
<thead>
<tr>
<th>Topics</th>
<th>Written Assignments</th>
<th>Programming Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>[# of lectures + days of discussion]</td>
<td></td>
<td>PreCourse Survey</td>
</tr>
<tr>
<td>Precourse Survey</td>
<td></td>
<td>Due Tuesday, Sept 1</td>
</tr>
</tbody>
</table>

http://www.cs.utexas.edu/users/dsb/cs392f/#Schedule
# Introduction to Model Driven Engineering [4]
1. UML class diagrams, MDE models and metamodels, model-2-text xforms
2. Metamodel constraints, OCL, model-to-model xforms, ATLAS, prolog
3. Building Domain-Specific Tools with MDE
4. Bootstrapping MDE Tools

**Readings**
- [Teaching Model Driven Engineering from a Relational Database Perspective](#)
- [Wikipedia Model Driven Engineering Article](#)

# Feature Models and Software Product Lines [3]
1. Feature models, attribute grammars, propositional formulas, SAT solvers
2. Feature model analyses, reasoning about edits
3. Next generation feature models, feature replication, multi product lines

**Readings**
- [Automated Analysis of Feature Models 20 Years Later: A Literature Review](#)
- [Mining Configuration Constraints: Static Analyses and Empirical Results](#)
- [CVL-Revised-Submission (Sections 7.2 and 7.3 only)](#)

# Feature Oriented Software Development (a.k.a. Feature Modularity) [2-3]
- Layered designs, OO virtual machines, collaborations, extensions, mixins, mixin-layers
- FOSD, principles of uniformity and scalability, generalized modularity, composition

**Readings**
- [AHEAD](#)
- [FeatureHouse](#)
- [Delta-Oriented Programming](#)

# Feature Interactions and Product Lines of Product Lines [2]
- Feature interactions, formal models of feature modules and interactions
- Software product lines of rank $n>1$, examples, and open problems

**Readings**
- [Feature Interactions, Products, and Composition](#)
- Draft of Product Lines of Product Lines

# Midterm

## 5. Refactorings [3]
- Review of Refactorings and Design Patterns
- Reflective Refactoring

**Readings**
Refactoring using Type Constraints

6. Design By Transformation [3]
- Map reduce, Liskov and Perry substitution principles, refinement, optimization, correct-by-construction
- Parallel architectures, product lines of streaming applications
- Product lines of streaming applications

Readings
- Modeling in Event-B (1st 2 chapters)


☑ score card for categories
- Domains, instances, arrows and MDE
- Pragmatics, inheritance, SPLs, functors, examples, commuting diagrams
- Products, product families, products of categories, pushouts
- Applications: portlet synthesis, geodesics, lifting, homomorphisms, test generation

Readings
- Pierce’s Basic Category Theory Text (pgs 1-27, 36-39, 41-47)
- Generative Programming for Embedded Systems

8. Extras [1] (if there is time)
- aspect-oriented programming (AOP) pointcuts, advice, event-based programming, feature-extensions of interpreters

9. Class Project Presentations [2-3]
Review [1]
Final