Curriculorum: A Computer Science Approach to Curriculum Management

Bob Lewis

School of EECS
Washington State University

October 10, 2008

Alternate Title

When Computer Scientists Confront the Dark Side (i.e., become administrators).

Outline

Motivation

- In 2005, I became Program Coordinator for Computer Science at the WSU Tri-Cities campus.
- I have many of the responsibilities of a Director (whom I report to on the main campus in Pullman), but none of the authority. (Seriously, I do like the job.)
- I needed to manage over time a moderately-sized amount of curriculum-related data from a variety of sources for planning and reporting.
- My approach: Treat the whole thing as a software design problem.

A Clarification on Terminology

- A *class* is a set of attributes and behaviors assigned to an object in an object-oriented system.
- A *course* is a unit of instruction which is (or was) listed in a catalog.
- A *session* is an instance (but not in an object-oriented sense) of a course. (It "has-a" course. In fact, it may have several.)

My Goal Today

I'll talk about the system I developed, but what I really want to emphasize is *how* I developed it rather than the system itself, so that, as computer scientists, you can build your own if the need arises.

So let's start with the requirements...

Provide Reports to...

Students	What are the course offerings and instructors for next semester?
The Bookstore	What instructors are teaching what classes?
The Registrar	What instructors are teaching what classes when?
Payroll	What adjunct instructors are to be paid how much?
Webmaster	What are the course offerings for the next two years?
Accreditors	What courses have been taught for the last seven years?

and *make sure they're all consistent*!

Planning Functionality

Some information is for my own use in planning:

- inform the (EECS-wide) Curriculum Committee
- associate adjuncts with courses they can teach
- maintain a record of instructor performance

Correct for Limitations of My University's DBMS

WSU has a fairly standard online course management system ("RONET" – Registar's Office NETwork), but...

- I have read-only access. (This is probably a good thing.)
- There's no provision for planning future semesters.
- Their schema doesn't quite do what I need:
  - missing information on adjuncts, degree programs, other institutions
  - one instructor listed per course
  - same telecourse treated as separate sections on separate campuses
  - differing special topics treated as separate sections
- The course catalog is sometimes wrong or out-of-date.

Unusual Requirements

Compare to the usual student-course-instructor-department example given in a lot of database books (to name a few):

- All special topics courses have the same number.
- Prerequisites are problematical:
  - may be at different institutions (i.e. community colleges).
  - may allow "one of" prerequisites.
- Some instructors are regular faculty, some are adjuncts.
- Each course has a "coordinator", which may be a regular faculty member or a committee.
- Multiple courses may be taught at the same time in the same place.
- Some data may be missing at a given time (e.g. during planning).

Additional Requirements

- help me learn about the curricula (hence the name *curriculorum*)
- allow for incremental year-to-year modifications
- provide a way to check for inconsistencies (typos, misspellings, etc.)

Is There a Commercial Product That Does This?

There are course management systems (e.g. Blackboard^{TM}) and larger, university-wide (usually custom) systems exist, but I wanted something in between.
Such a commercial system is unlikely (see above requirements) but this was irrelevant, because:

- I wanted to learn and design the schema, not adapt someone else's (assuming that was even possible).
- I was not in a position to evaluate how well such a product would fit my schema when I started.

Why Not Use a DBMS?

- DBMS's (the ones I know) focus on the wrong thing: tables with fields of (fixed-width, usually) strings.
- None of the classes I envision (instructors, courses, etc.) have more than about 100 instances.
- No GUI required. (Text editors don't scare me!)
- No client-server architecture required. (One user: me.)

This was more of a data structuring problem than a database problem.

Why Python?

- I knew it and wanted to get better at it.
- Lots of handy features:
  - object orientation (classes, inheritance, and polymorphism)
  - defaultable arguments to functions and methods
  - built-in sequences (lists and tuples)
  - excellent string operations (arbitrary length!)
  - very readable (as we'll see)
  - name errors (objects, keywords, etc.) detected by interpreter
  - comments (!)
- No reason why your favorite language (Java, C#, C++, Lua, Perl, Ruby, etc.) couldn't work as well. Try it!

The curriculorum Module

This ∼650-line Python module contains no objects, only class definitions:

- curriculum-related classes:

`Campus`	`Department`	`Instructor`	`Session`
`Course`	`GraduateArea`	`Season`	`Staff`
`Degree`	`Institution`	`Semester`	`Timeslot`

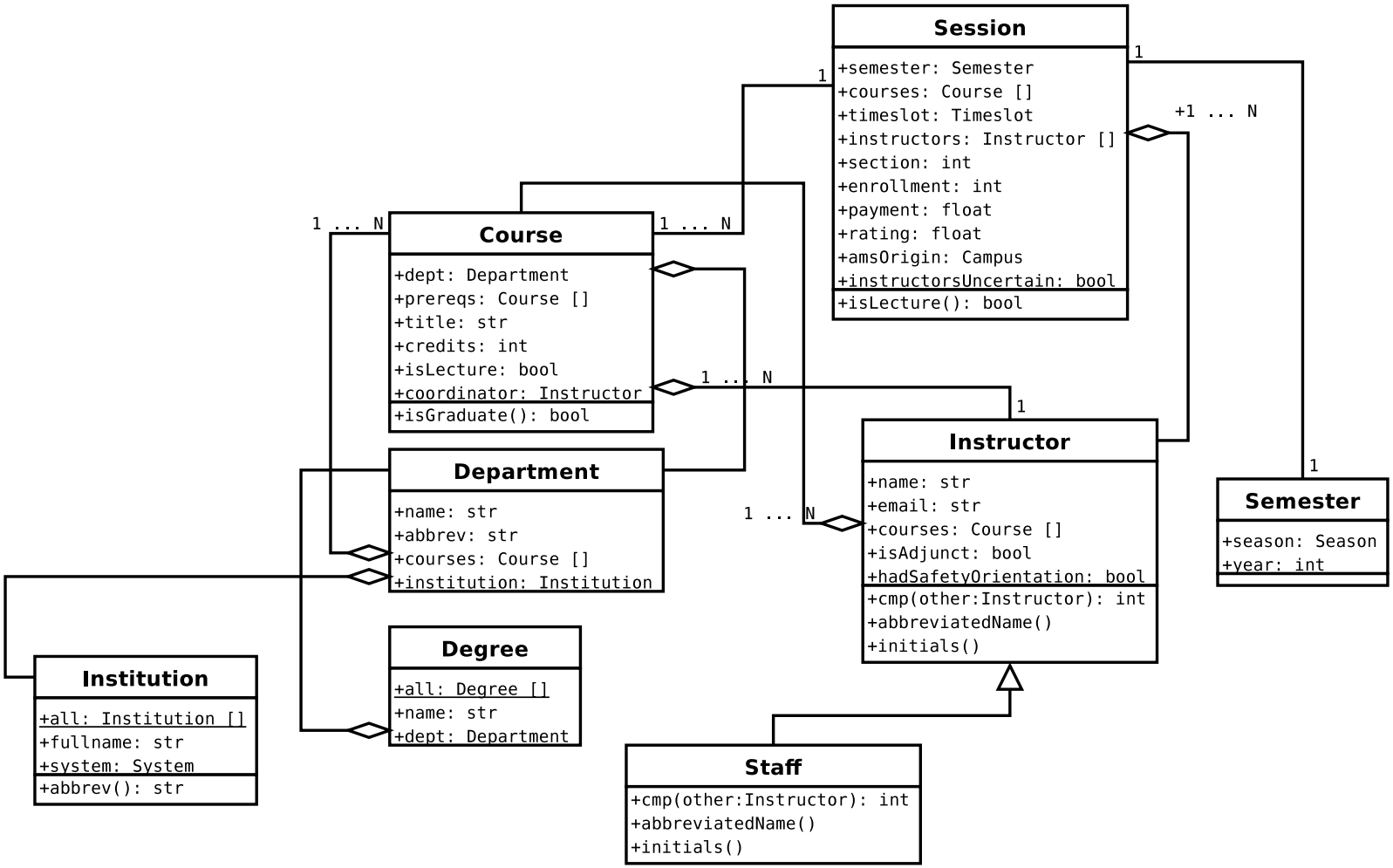
- curriculum-related exceptions:

`CourseIsNotGraduate`	`NotQualified`	`ImproperPayment`

- and a simple (LaTeX) table generation package:

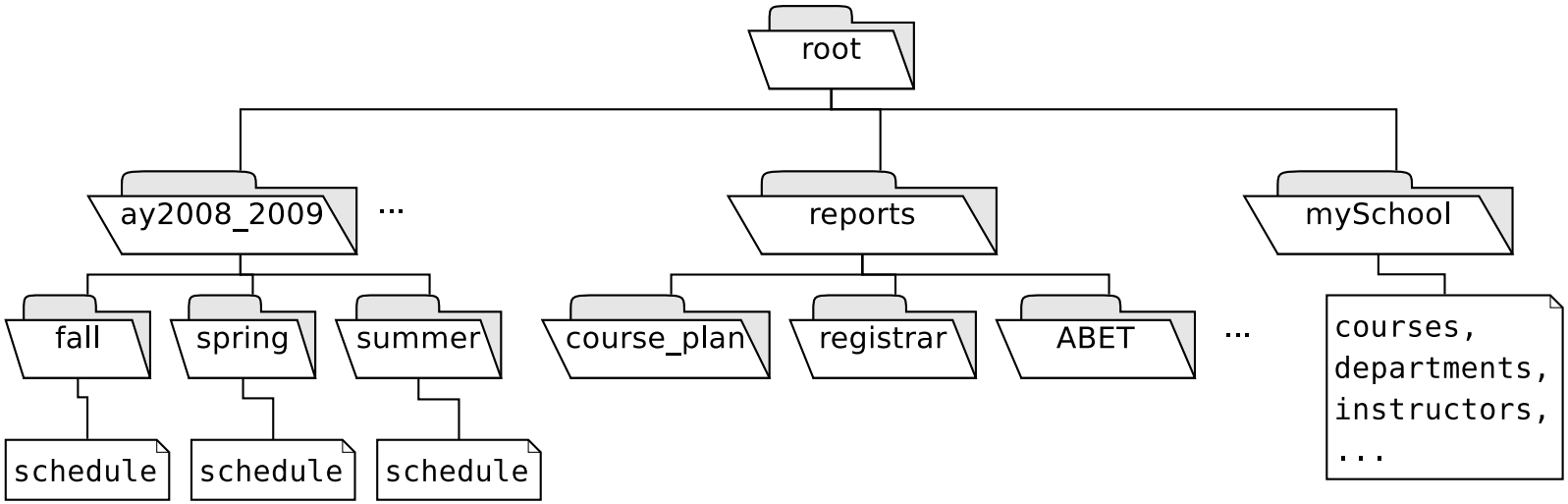
`Table`	`Column`

UML Class Diagram



Directory Layout

An important part of the implementation is getting the directory structure right...



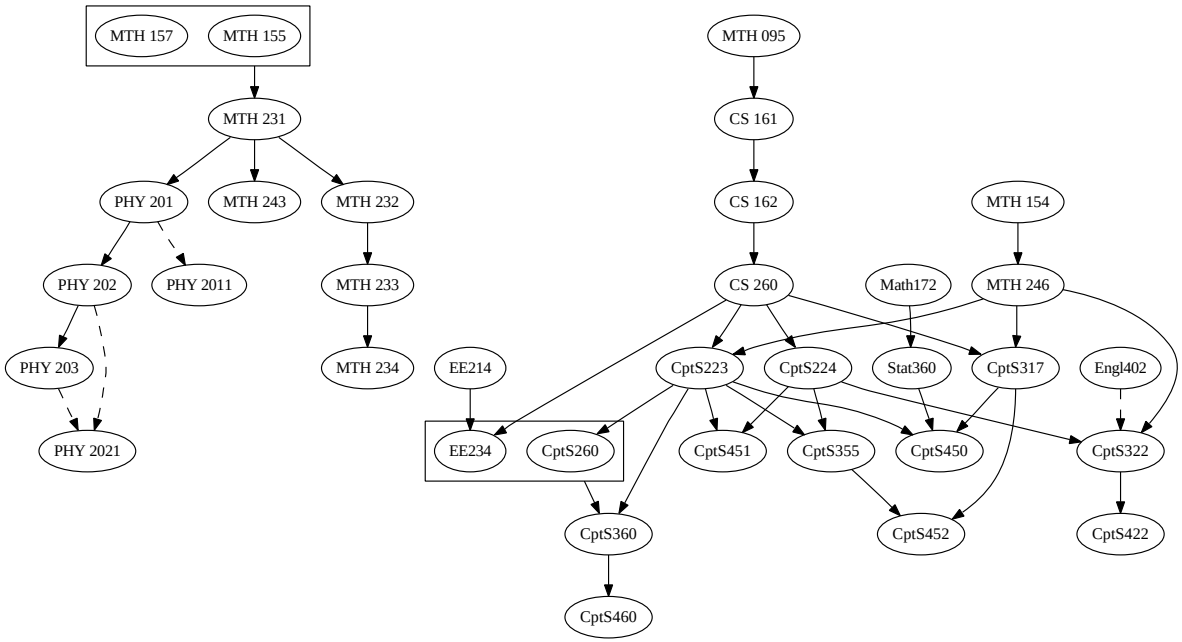
curriculorum is installed as a Python module and contains no institution-specific data (mostly classes).

Report Generation

- *curriculorum* generates (LaTeX) tables.
- Tables may be included in LaTeX documents (see example).
- *curriculorum* can also generate the GraphViz "dot" format (e.g.) for prerequisite dependency graphs.

Example: Dependency Graph

Here is the dependency tree (DAG, to be precise) *curriculorum* automatically derives from prerequisite data:



This includes courses at both WSU and Columbia Basin (Community) College.

A Guided Tour

Let's examine some of the files first-hand...

Conclusions

- *Curriculorum* has more than justified its development time.
- It has been in place since 2005, and has adapted to curricular changes very well.
- Side effect: Developing *curriculorum* has made me a better programmer.

Future Work

- documenting student specializations (e.g. games, networks, software engineering) and showing recommended schedules
- additional community college transfer equivalencies (i.e. new `Institutions`)
- HTML output (esp. for student perusal)