An Analysis Pattern for Course Management

[1]Xiaohong Yuan and [2]Eduardo B. Fernandez

[1]Department of Computer Science, North Carolina A&T State University
Greensboro, NC 27411, USA
xhyuan@ncat.edu

[2]Department of Computer Science and Engineering, Florida Atlantic University
Boca Raton, FL 33431, USA
ed@cse.fau.edu

Abstract

We discuss an analysis pattern for the management of student courses. The pattern describes events such as student registering in course sections, student adding and dropping course sections, grade management, etc. The pattern corresponds to a minimal semantic unit, applicable to a variety of situations. This is a composite pattern that includes two simpler patterns that describe course registration and grade management. The component patterns have their own value and can be used separately.

Introduction

Management of courses is a common process for all colleges, universities, and training centers. It is also needed for distance learning and web-based education programs. We present here an analysis pattern that describes basic aspects of course management. This pattern is an example of what we have called *semantic analysis patterns* [Fer00], because it emphasizes semantic aspects of the application model, as opposed to flexibility. A pattern of this type implements a small set of fundamental use cases and includes a few component patterns. The pattern, Course Management, emphasizes fundamental aspects of course management and leaves out extensions, exceptions, and varieties. This pattern could serve as a starting model for developing course management applications; it must be tailored to the specific application and complemented with other patterns or ad hoc modeling to build a complete application. This is a composite pattern and we present first its two component patterns, Course Registration and Grade Management, which we combine later into Course Management. The component patterns have their own value and can be used separately. The target audience of this paper are software engineers who build applications of this type and software architects and analysts. Another use of this pattern is as a pedagogical tool.

Course Registration

Intent

This pattern describes the process of students registering into courses and keeping track of who is in a course and what courses a student is taking.

Context

An institution that offers courses of any type. This model applies mostly to American universities and may also apply to other countries.

Problem

Students interested in attending courses need to register for them before they start or for some time at the beginning of the course. The students are allowed to add or drop courses during a specific period of time. Registration is needed to allocate instructors and facilities and to collect fees from the students. All this is rather complex and must be done in a convenient and efficient way. There is a variety of ways to perform registration functions but they all have many common aspects; we need to find a generic model for this commonality.

Forces

- A large number of students in a course is hard to manage. We may need to subdivide each course into smaller units (called 'sections' in U.S. universities).
- Students taking a course or faculty teaching a course may have schedule conflicts or live too far from the central campus of the school; we need to help them build appropriate schedules.
- Data organized into standard documents such as enrollment forms, should be directly represented in the model. Only the information in the document is important, not its specific layout or representation.
- To improve student performance in some courses we may want to put constraints about the level of previous knowledge required.
- For convenient management and to improve student performance we may need to put constraints on the enrollment period.

Solution

Divide a course into smaller groups of students (sections), each with a separate instructor and a specific class time. A course may have a different number of sections during different semesters and at different times in a given semester; in this way students can register in their selected sections according to their needs and constraints. Sections may have enrollment limits (maximum and minimum). Students enroll in these sections using specific registration forms. Courses may have prerequisites that are needed to enroll in the course. This corresponds to the following use cases:

(1) *Register a student in a course section*. During the registration period, a student reviews the available course sections and selects a fixed number of course sections for the coming semester. The student also selects a fixed number of alternative course sections in case a course section becomes full or cancelled, or there is a time conflict. The student submits a form (or several forms) that lists these primary courses as well as alternative courses. The system will check whether a course section is full or canceled, whether there is a time conflict, or whether the student satisfies the prerequisites. When the student registers in the class successfully, the system will produce the student's class schedule and the total amount of the tuition the student needs to pay.

(2) *Add/Drop course sections*. During a specific time period, students are able to add or drop courses. A student submits a form listing course sections he wants to add or drop, and the system produces the student's new schedule and the tuition.

Figure 1 is a class diagram for the realization of these use cases. Class **Course** provides a description of the course, typically name, number, summary of contents. The self-association **Prerequisites** of class **Course** reflects the fact that a course may have prerequisites. A course may have several **Sections** during a semester. The association **IsEnrolledIn** between class Section and class Student describes the fact that a student is enrolled in many course sections, and many students are enrolled in a course section. The association class **Schedule** describes a student's schedule of all the course sections in which she is enrolled in a semester. Each student submits one or several **Forms** listing the course sections in which he wants to register. Class **Transcript** is a record of the courses taken by a student and is needed to check the prerequisites of a course at registration.

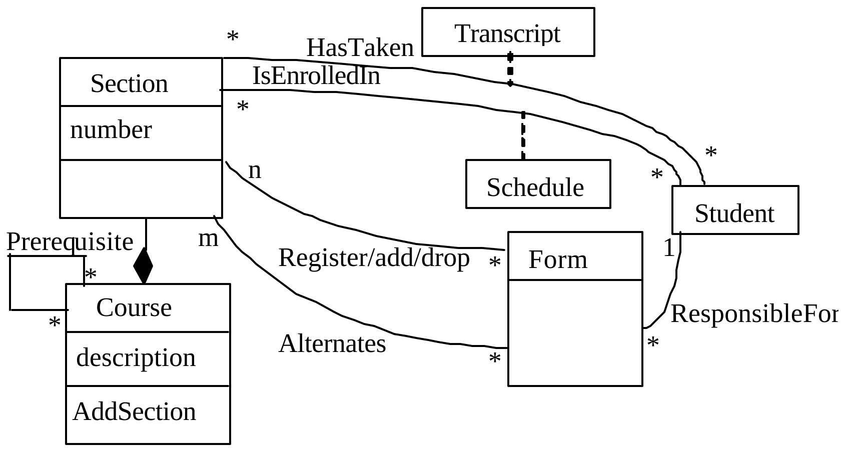

Figure 1 Class diagram for course registration

Figure 2 shows the state diagram for class Section. A Section object could be in the states of *Created*, *Open*, *Filled*, *Canceled* and *Frozen*. A student can register/add/drop a section when this is open. When the total number of students in the section reaches the maximum allowed number, the Section object changes to Filled state. When the number of students registered is less than the minimum allowed number, the section could be canceled. When the deadline for registering/adding/dropping classes expires, the Section object is in Frozen state. When the deadline for registering/adding/dropping classes is extended, the Section object changes to the Open state. The Section object is added to a historic log at the end of a semester.

Figure 3 shows a sequence diagram for registering in course sections. The student submits a form listing the course sections in which he wants to register. Alternative course sections are also listed in the form. The system first checks whether it is within the registration time window, then checks the availability of each section (i.e., whether the section is canceled or full), whether the student satisfies the prerequisites, and whether it has a time conflict with the student's existing schedule. Notice that the student's transcript needs to be checked to verify whether the student satisfies the course prerequisites. If the student fails when registering in a section, the system considers alternative course sections. Finally, the system calculates the tuition according to the number of courses registered. Figure 4 shows a sequence diagram for dropping a course section.

Consequences

The pattern provides the following benefits:

- It describes an abstract course registration process that can be applied to various specific situations.
- Dividing a course into sections provides flexibility, it allows students to register in different sections according to their constraints and faculty to teach the section to their convenience.
- Data in typical documents is represented by classes, for example, Form, Schedule and Transcript.
- Prerequisites in the form of required courses can be specified for each course.
- Specific stages for registration can be defined along the course period.
- Every aspect of the use cases is represented in this pattern, and no implementation aspects are included, leaving open different possibilities.

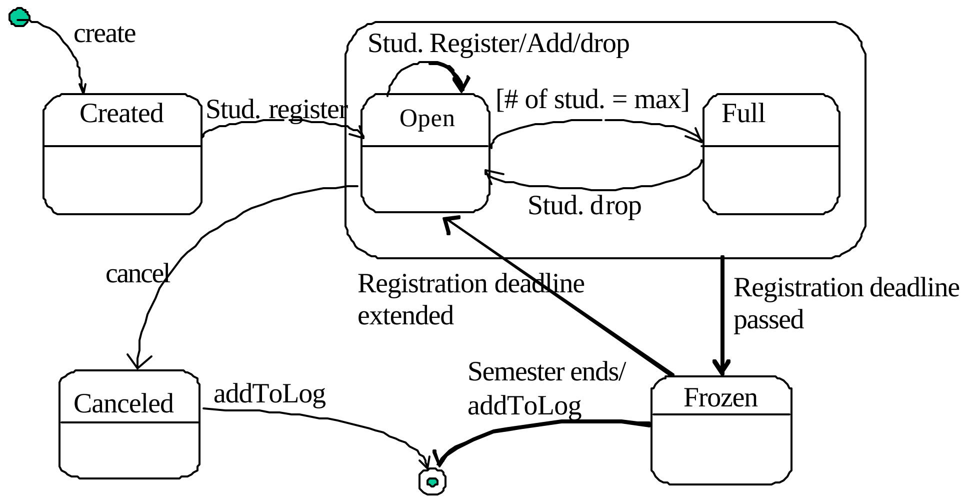

Figure 2 State diagram for class Section

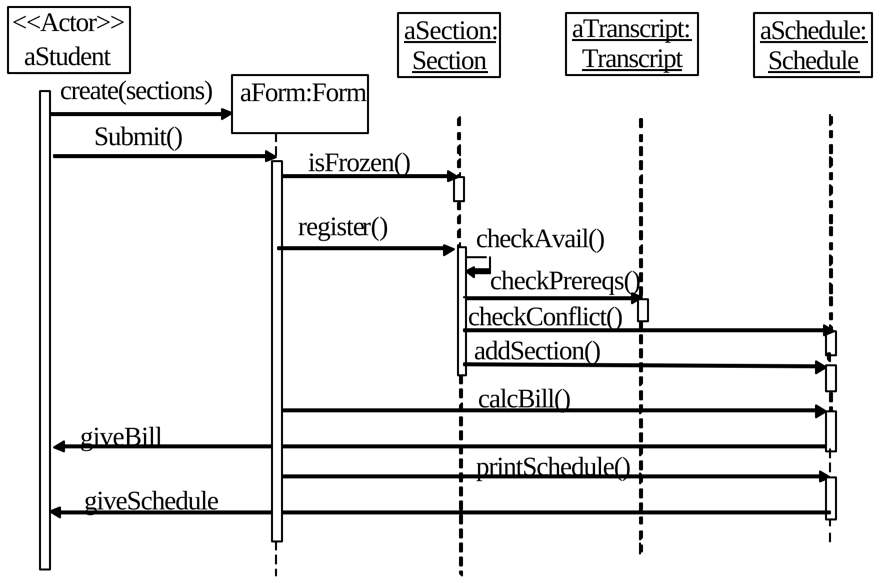

Figure 3 Sequence diagram for registering in course sections

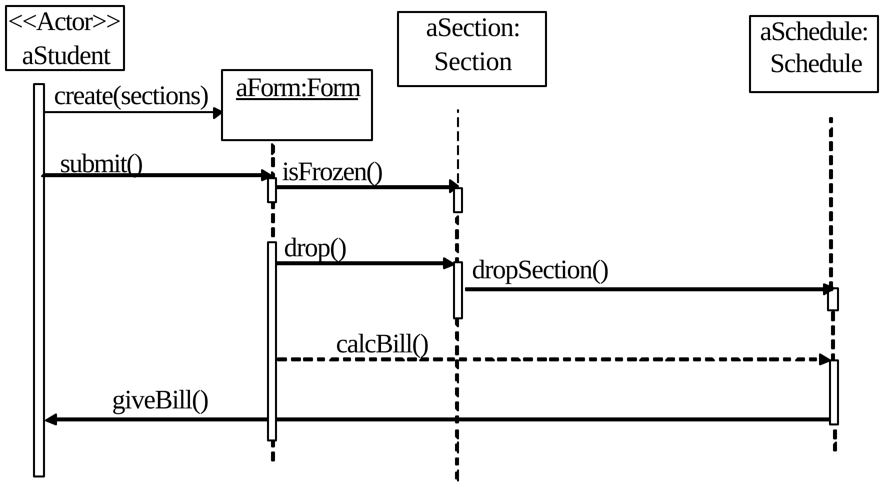

Figure 4 Sequence diagram for dropping course sections

Not all the situations described by this pattern are exactly alike, which means that the pattern needs to be tailored for specific applications:

- There are a variety of ways to register in course sections. In some universities, students fill in registration forms listing course sections they want to register in as well as alternative course sections, and then give the forms to the registrar or their advisors, who will register the student in classes. Some universities have a web site for students to register on line, in which case the registration forms are created and submitted on line. Some universities allow students to register through phone calls and a registration form does not exist explicitly. Several or all of these ways of registration may be provided in a given university.

- Some places do not use sections but in many places a course may have several sections in a semester. Some sections may have special restrictions. For example, an honors section only allows honors students to register in. Some sections only allow students of certain majors to register in.
- The university may place a hold in a student's file, for example, a health insurance hold. A student is not allowed to register if his holds are not removed.
- Some universities may place restrictions on the maximum and minimum number of credit hours a student can take in a semester.

The liabilities of the pattern are in the fact that some aspects are not represented in the pattern, including:

- Billing and payment policies. How to handle a payment failure before its due date. In some universities, all the course sections in which the student has enrolled are dropped automatically if the student fails to pay before the deadline. The student needs to make the payment and then register for courses all over again.
- Time conflicts. When a student cannot register in a course section because of a time conflict, she can choose an alternative course section. Sometimes the student needs to drop some of the course sections in which he registered and then choose other course sections.
- Policies about course or section cancellation. These vary widely between institutions.
- Waiting lists for courses. When a student drops a course section, a student on the waiting list for the course section will be able to enroll in this section.

Known uses and related patterns

The following are examples of uses of this pattern:

- A registration system in a college or university. Most American universities use similar models, e.g., Florida Atlantic University, North Carolina A&T State University, U.C.L.A.
- An on-line registration system in a college or university. Again, most American universities offer this option.
- A registration system for a web-based distance learning program. An example is the University of Phoenix, AZ, which offers a large variety of web courses.
- A registration system for industrial training centers. An example of this is the Lucent Training Department that schedules several sections of a course at different times and in different locations.

This pattern includes a Collection pattern [Bus96] (A course has a collection of sections).

Grade Management

Intent

To allow an instructor to evaluate the performance of his students in a course.

Context

An institution that gives courses where the students need to be evaluated for their performance in the courses they take. This model applies mostly to American universities and may apply to other countries.

Problem

When students take courses in a teaching institution they may want to have a proof that they have acquired a given degree of knowledge. This may be needed to satisfy degree requirements, for searching for employment, or just for personal satisfaction. Sometimes the evaluation is just reduced to a proof of attendance. This means that an instructor who teaches a course section needs ways to evaluate the performance of the students registered in the course section and to keep records of these evaluations.

Forces

- We need a standard way of evaluating performance in a course.
- The instructor needs convenient ways to keep records of her evaluations.
- Because of the need for proof, data in standard documents such as Report Cards and Transcripts should be directly represented in the model

Solution

The most common way to evaluate performance in courses is by giving the student a *grade*. Two types of grade records are kept by the course instructor. She keeps track of partial grades, e.g., grades in homework, tests, projects, etc.; this is the *Report Card*. She uses these to calculate the final course grade. *Transcripts* are historical records of all the grades in the courses the student has taken.

Figure 5 shows the class diagram for Grade Management. Classes **Student** and **Instructor** define these two basic participating entities. Class **AcadPerson (Academic Person)** generalizes Student and Instructor. An instructor advises many students, and may teach several course sections. The association class **ReportCard** describes a student's grade information (homework grades, test grades and final grade) in a specific course section. The association class **Transcript** describes information of a student's final grades in all the course sections he has taken so far.

Figure 6 shows the sequence diagram for an instructor giving a grade to a student, calculating the average grade made by a student, and adding the final grade to the transcript.

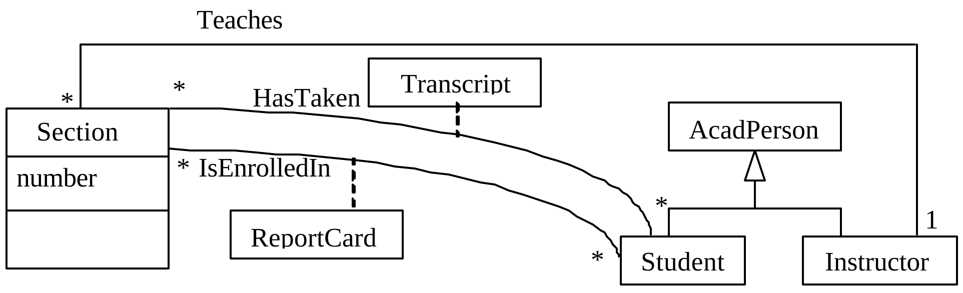

Figure 5 Class diagram for grade management

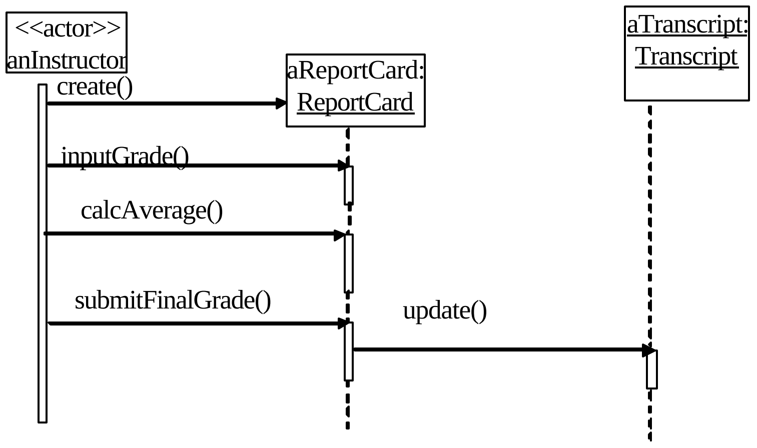

Figure 6 Sequence diagram for grade management

Consequences

- This pattern describes the general aspects of grade management for a course.
- The grades are recorded in standard documents represented by classes ReportCard and Transcript.
- For courses that just require attendance the ReportCard becomes AttendanceRecord.

Aspects not represented in the pattern include:

- Sometimes a course section is also assigned student teaching assistants (TAs). In that case the TAs may give grades to students, or the professor and the TA together assign grades to the students.
- There are different grade schemes. For example, in the USA universities use “A”, “B”, “C”, “D”, “F”; some use “P” (pass) and “F”. In Europe and South America numerical scales are used (0 to 5, 0 to 7, 0 to 15). All of these have in common that they are measures of quality, going from a lower to an upper bound.

Known uses and related patterns

The following are examples of uses of this pattern:

- A grade management system for a university. The examples for the first pattern are still valid.
- An on-line grade report system of a university. The Omni Middle school in Boca Raton, FL, uses an on-line grade reporting system accessible to its students and their parents.
- A grade management system for a web-based distance learning program.
- Some industrial training centers give grades to their students. For example, Lucent requires instructors to evaluate the performance of the employees who take their courses.

This pattern includes the Role pattern [Bau00] (The same academic person may appear in the roles of student and professor).

Course Management

Intent

This pattern describes the conceptual aspects of course management, including registration and evaluation aspects. It also handles common aspects such as adding or deleting courses, assigning instructors to sections, and keeping track of advising.

Context

An institution that offers courses of any type. This model applies mostly to American universities and may apply to other countries. .

Problem

There are institutions where we need to combine the functions of registration and evaluation as well as aspects that are related to both of these functions. These include setting up lists of courses every semester, assigning instructors, and keeping track of advising.

Forces

In addition to the forces of the component patterns we also have:

- Instructors devise new courses that need to be added to the list. Courses not being taught anymore must be removed.
- Instructors have a range of courses they can teach and they usually state their preferences about what they want to teach.
- It is important that students know who their advisors are and vice versa.

Solution

Combine the solutions of the two component patterns Course Registration and Grade Management and add ways to keep track of students advised, to assign instructors to sections, and to add or remove courses. The following additional use cases are implemented:

- Add/delete/update courses. The department chair adds/deletes course sections or updates course description.
- Add/remove instructors. Instructors may be added/removed from departments.
- Assign instructors to courses. Instructors are assigned to teach specific sections of a course.
- Find out who advises a specific student.

This class model is a combination of the two previous class models with the addition of a **Department** class with which instructors are associated. Association **Advises** indicates the students advised by a given instructor. Figure 7 shows the class diagram for course management.

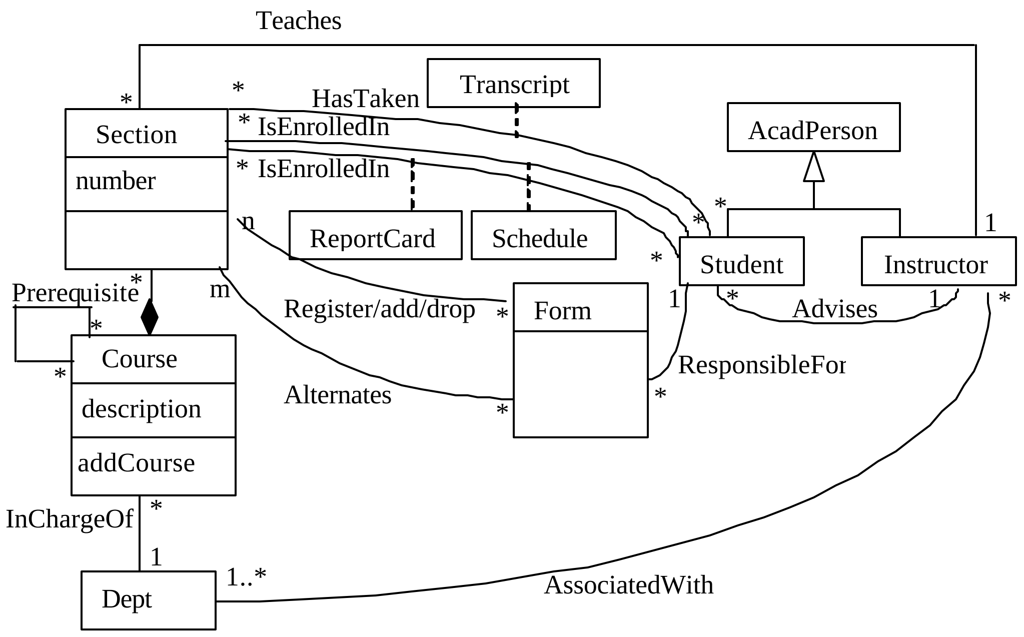

Figure 7 Class diagram for course management

Consequences

This pattern combines the consequences of its component patterns plus:

- It is easy to add new courses or remove existing courses.
- Instructors can be conveniently assigned to courses.
- Students can keep track of their advisors and faculty of their advisees.

Some additional aspects that could be added include:

- Access rights. The registration system may be only accessible by the registrars, or/and by the instructors. Students may also have access to the system if they are allowed to register through phone system or on-line. Only instructors can input grades and only for the sections they are teaching. This can be done using the patterns in [Fer01].
- How to assign classrooms and time slots to course sections.

Known uses

The following are examples of uses of this pattern:

- A course management system in a university. Most universities in the US, including the ones mentioned earlier.
- A course management system for a web-based distance learning program.
- Course management for industrial training centers. Again, Lucent uses this model.

[D’S97] presents generic frameworks that can specify the requirements for a service company that markets and delivers seminars related to resource allocation, materials production and market trends. Ambler [Amb01] uses a student registration application as a running example. Kimovski et al. [Kim01] describe a virtual learning system that has a web-based interface. They present an activity diagram that shows the student and professor registration sequences. The student and the professor will follow different branches; for example, the professor can log on and update new courses and preview information about student attending a certain course. The student can log on and view information about the course or his grade information. In [Vad99], an object model for an application involving student, course and lecturer is mapped into a table model. This object model has a class TA to model the fact that a student could be TA for a course offering, and uses object Enrollment to record the enrollments of a course offering.

Related patterns

As indicated earlier, this pattern includes two other patterns: a Collection pattern [Bus96], and a Role pattern [Bau00]. Frameworks for resource allocation in [D’S97] could complement our analysis pattern to cover such aspects as allocating classrooms and time slot to course sections, and allocating instructors and TAs to course sections. In fact, schedule management, advising, and others, could be separate patterns if we add enough details. As indicated, the authorization patterns described in [Fer01] can complement this pattern to describe authorization and access control.

Acknowledgements

We thank our shepherd Till Schuemmer provided very valuable suggestions that considerably improved this paper. The participants in the Writers' Workshop at EuroPLoP'03 also provided some useful comments.

References

[Amb01] S.W.Ambler, *The object primer* (*2nd. Ed*.), Cambridge University Press, 2001.
[Bus96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal., *Pattern-oriented software architecture*, Wiley 1996.
[Bau00] D. Baumer, D. Riehle, W. Siberski, and M. Wolf, "Role Object", Chapter 2 in *Pattern Languages of Program Design 4* (N. Harrison, B. Foote, and H. Rohnert, Eds.). Also in Procs. of PLoP'97, http://jerry.cs.uiuc.edu/~plop/plop97
[D'S97] D. D'Souza. "Framework: Java to UML/Catalysis", *Journal of Object-Oriented Programming* (*JOOP)*, September 1997, 10-13.
[Fer99] E. B. Fernandez, "Good analysis as the basis for good design and implementation of object-oriented systems". http://www.cse.fau.edu/~ed/Goodanalysis.pdf
[Fer00] E. B. Fernandez and X. Yuan, "Semantic analysis patterns", *Procs. 19th Int. Conf. on Conceptual Modeling (ER2000)*, October, 2000. 183-195.
[Fer01] E. B. Fernandez, "A pattern language for security models", http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions/
[Kim01] G. Kimovski, V. Trajkovic and D. Davcev. "Virtual learning system", *Procs. 2001 IRMA International Conference*, 427-430.
[Vad99] K. Vadaparty. "Mapping objects to tables", *Journal of Object-Oriented Programming (JOOP)*, July/August 1999, 45-47.