Specifying Software Component Frameworks using UML and OCL

H. Conrad Cunningham
Yi Liu

Software Architecture Research Group
Dept. of Computer & Information Science
University of Mississippi

Project

Title: Acxiom Laboratory for Software Architecture and Component Engineering (ALSACE)

Sponsor: Acxiom Corporation

Type: Curriculum development

Begin Date: January 2002

Personnel:

- Dr. H. Conrad Cunningham (PI)
- Ms. Yi Liu (PhD student)
- Student TBD

Project Goals

- Develop software components course
  - systematic, technology-independent methods
  - Java 2 Enterprise Edition (J2EE) for practical exercises
- Design and implement example software systems
  - simple course registration system?
  - online voting system?
- Lay foundation for further software architecture research and course development

Course Approach

- Emphasize abstraction
- Evolve sequence of system specification models
  - build on object-oriented analysis and design techniques
  - construct system of components
- Employ design patterns and design by contract
- Use standard notations
  - Unified Modeling Language (UML)
  - Object Constraint Language (OCL)
- Apply J2EE implementation and deployment technologies

Software Component

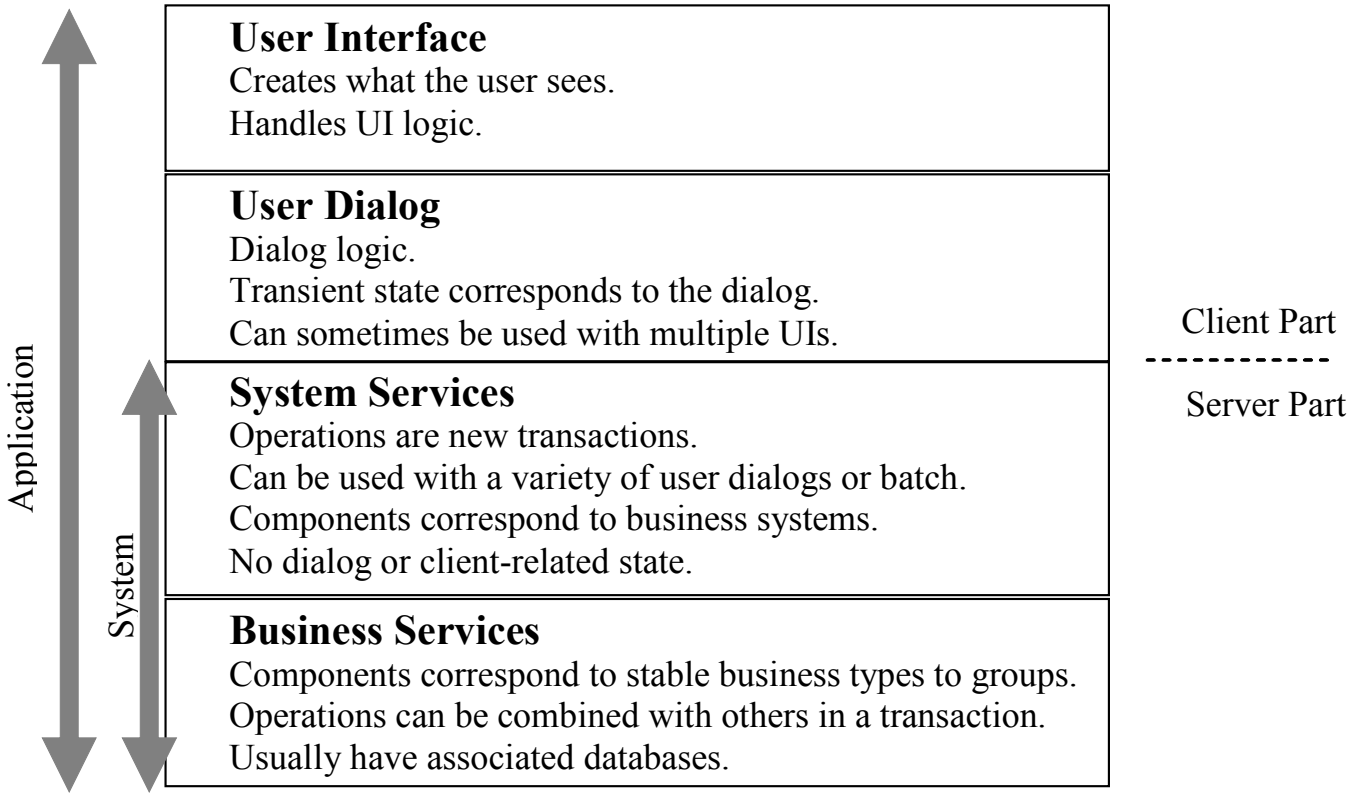
A self-contained software construct with

- conformance to a component standard
- clearly specified functionality
- well-defined runtime interfaces that hide implementation details
- capability for independent deployment
- support for composition and replacement

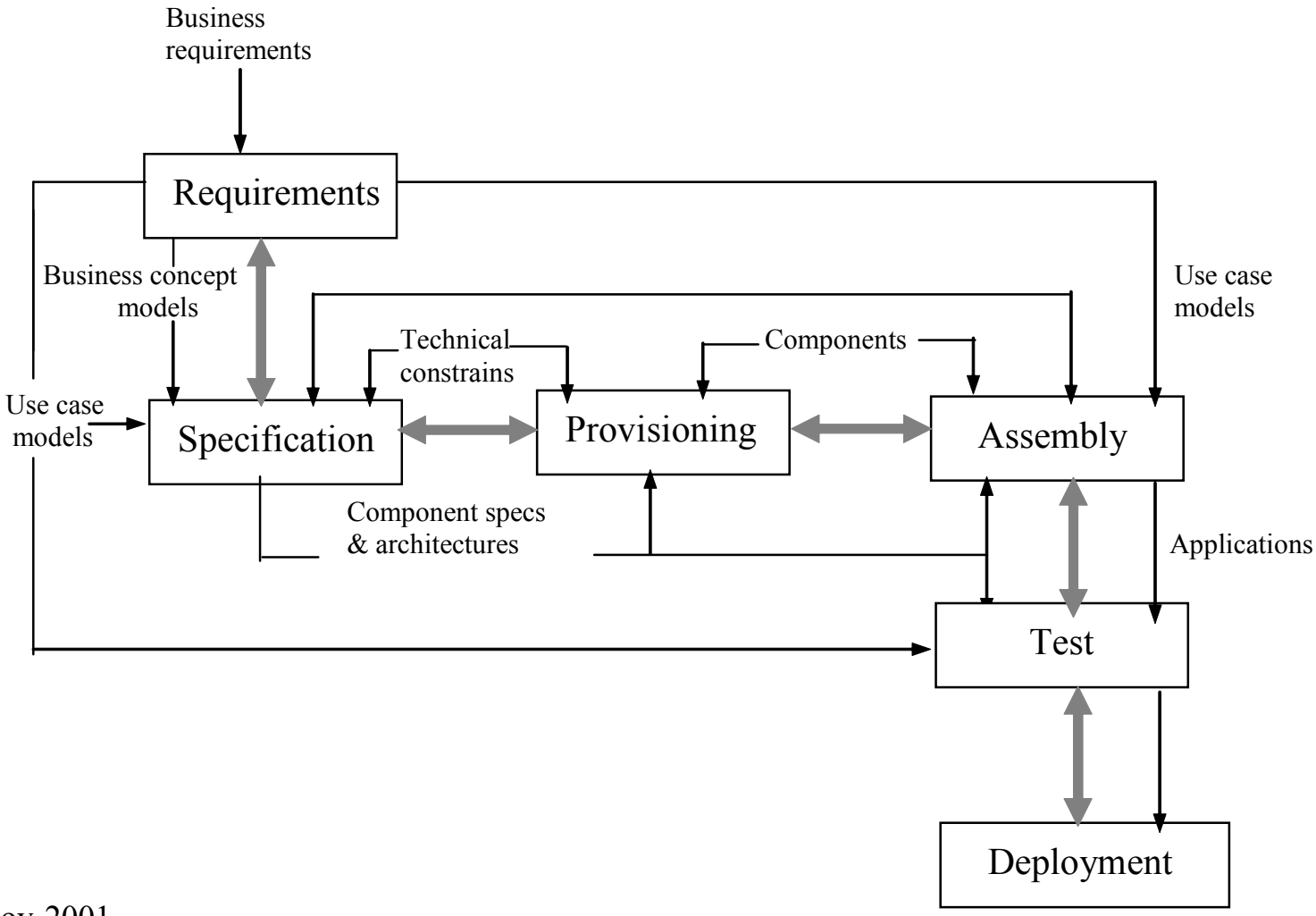
Reference Books

- J. Cheesman and J. Daniels. *UML Components: A Simple Process for Specifying Component-Based Software*, Addison Wesley, 2001.
- J. Warmer and A. Kleppe. *The Object Constraint Language – Precise Modeling with UML*, Addison Wesley, 1999.
- C.T. Arrington. *Enterprise Java with UML,* Wiley, 2001.
- D. Alur, J. Crupi, and D. Malks. *Core J2EE Patterns: Best Practices and Design Strategies*, Prentice Hall, 2001.

Architectural Layers



Development Workflow



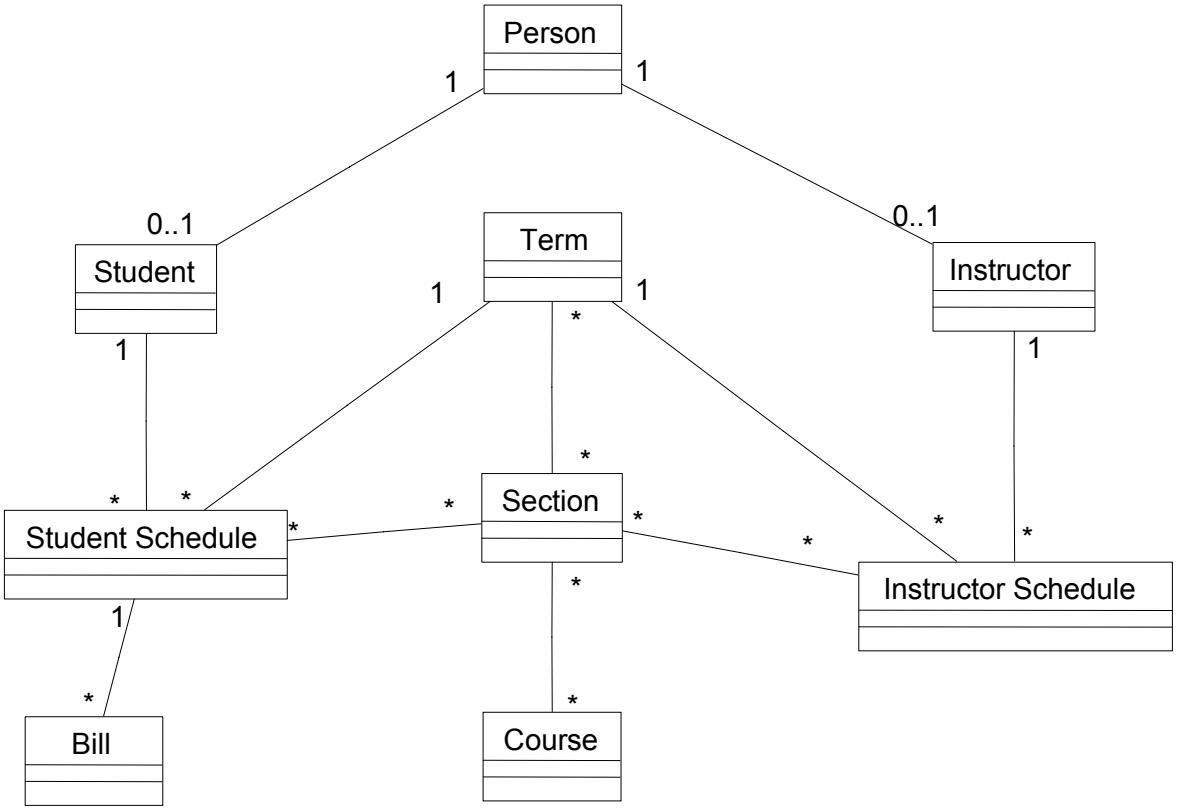
Requirements Definition Phase

- Domain (business concept) model
  - discover classes for real-world entities and related concepts
  - show with UML class diagrams
- Use case model
  - capture requirements by detailing user interactions
  - show with UML use case diagrams and scenarios

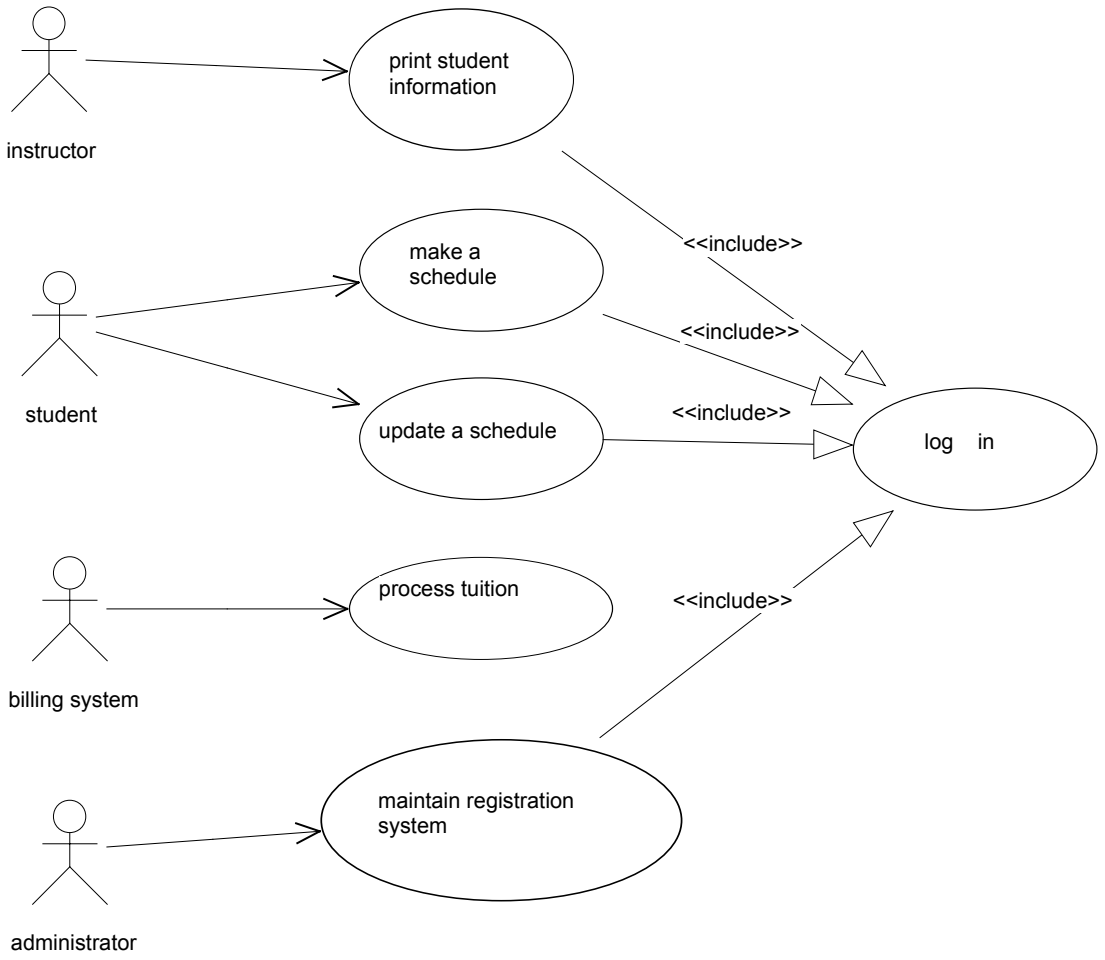
Course Registration System

- Students registering for classes
  - select classes initially
  - modify or display class schedule later
- Instructors
  - display teaching schedule and class rosters
- Billing system
  - generates tuition bills
- System administrators
  - maintain student and instructor lists
  - manage course information

Domain Model



Use Cases


instructor
print student information
make a schedule
student
update a schedule
process tuition
billing system
maintain registration system
administrator
log in
<<include>>
<<include>>
<<include>>
<<include>>

Scenarios

Name: Make a Schedule
Initiator: Student
Goal: Make a new student schedule

Main success scenario:

1. Student logs in the system
2. Student asks to make a new schedule for a term
3. Student selects section
4. Repeat 3 until the student finishes selecting courses
5. Submit the schedule
6. System notifies the billing system
7. Billing system generates tuition bill

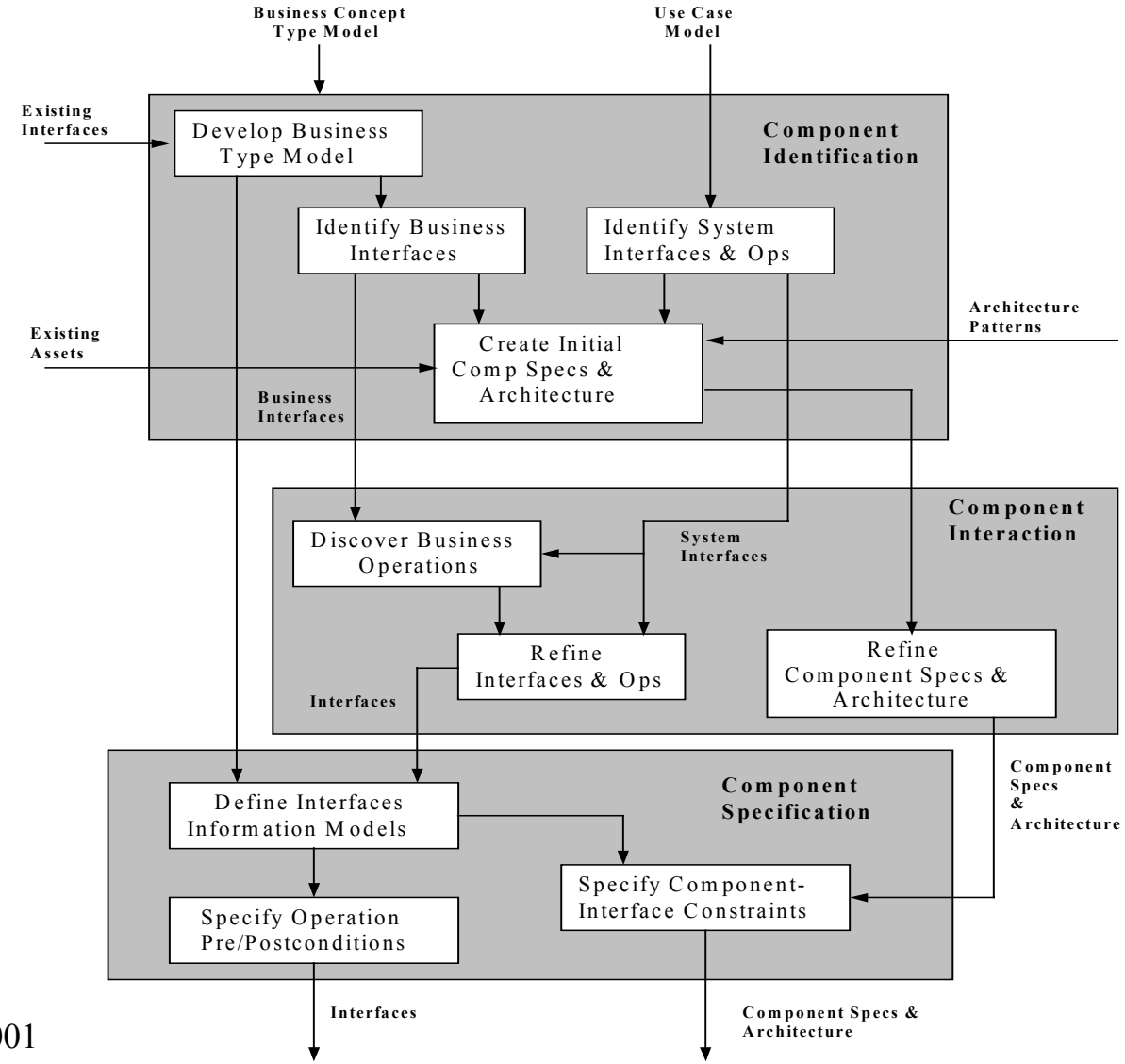
Extensions:

1. Login failed
   a) Fail
2. Section is full.
   a) Fail

3.1 Schedule sections for same time
   a) Fail

3.2 Schedule sections with overlapping times
   a) Fail

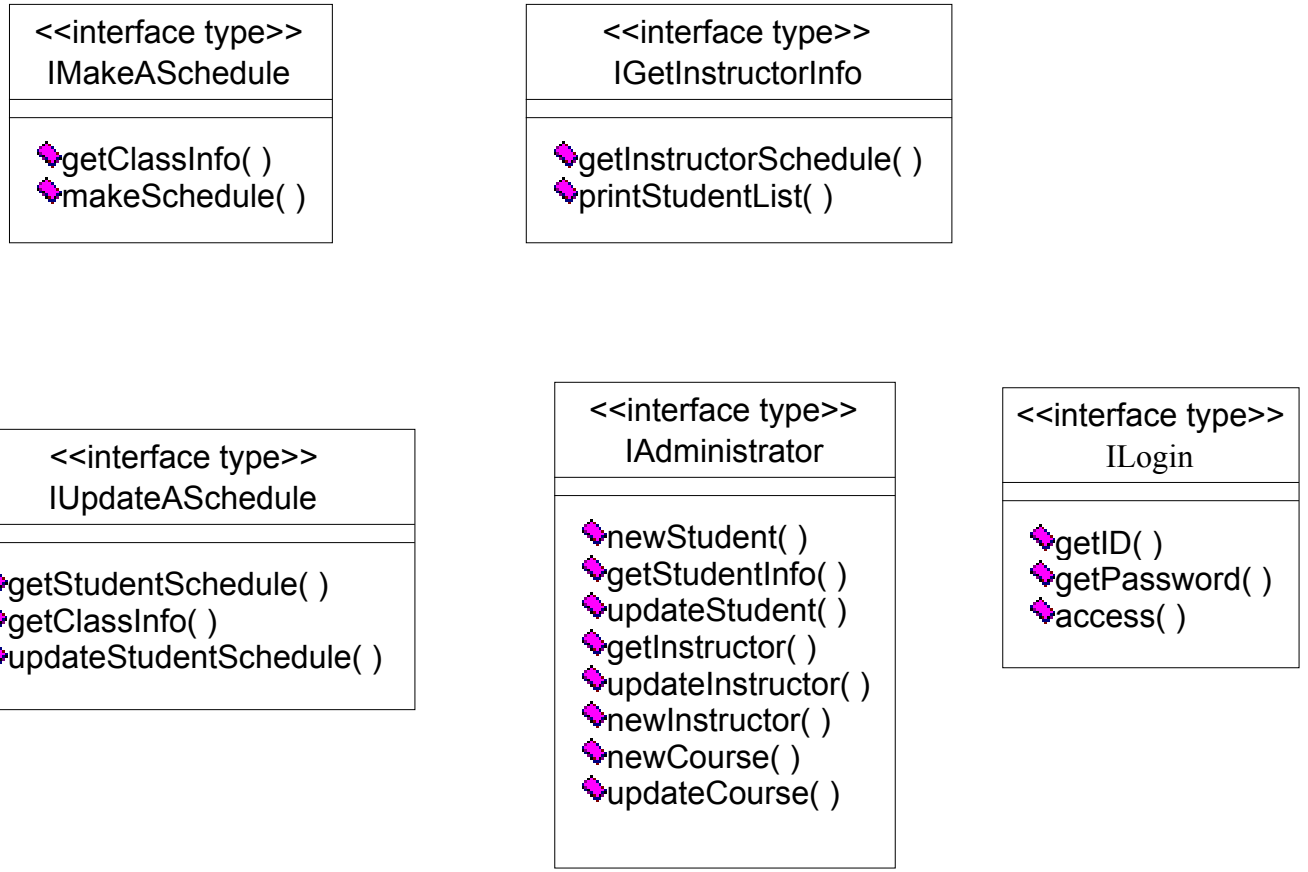
Specification Phase (1)



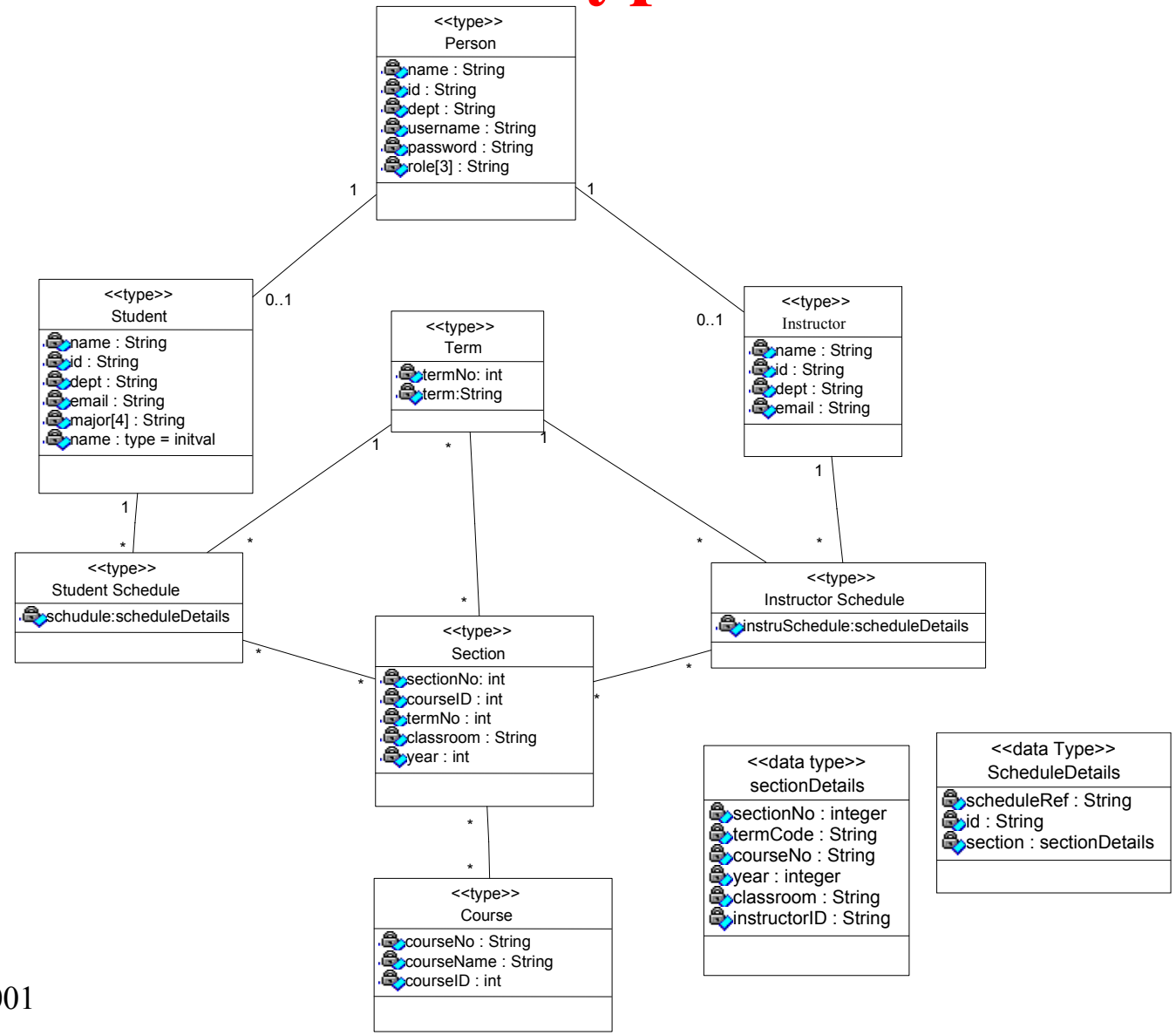
Component Identification Stage

- Specify system interfaces
  - give a system interface for each use case
- Define business type model
  - remove out-of-scope entities from domain model
  - specify business rules
  - identify core business types
  - create business interface for each core type
  - allocate responsibilities for other types and associations
- Identify initial component architecture
  - assign one component for each business interface

System Interfaces



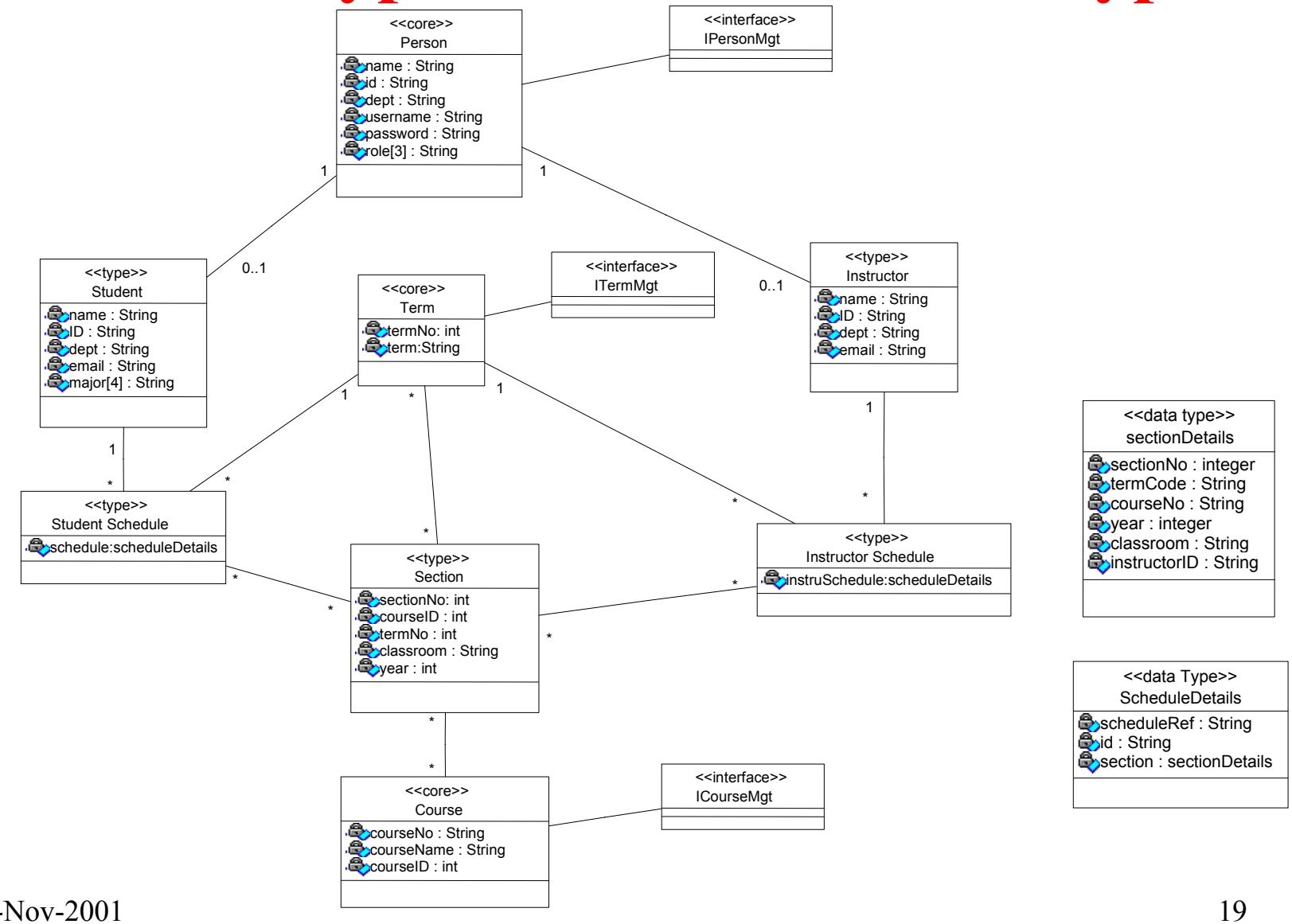
Business Type Model



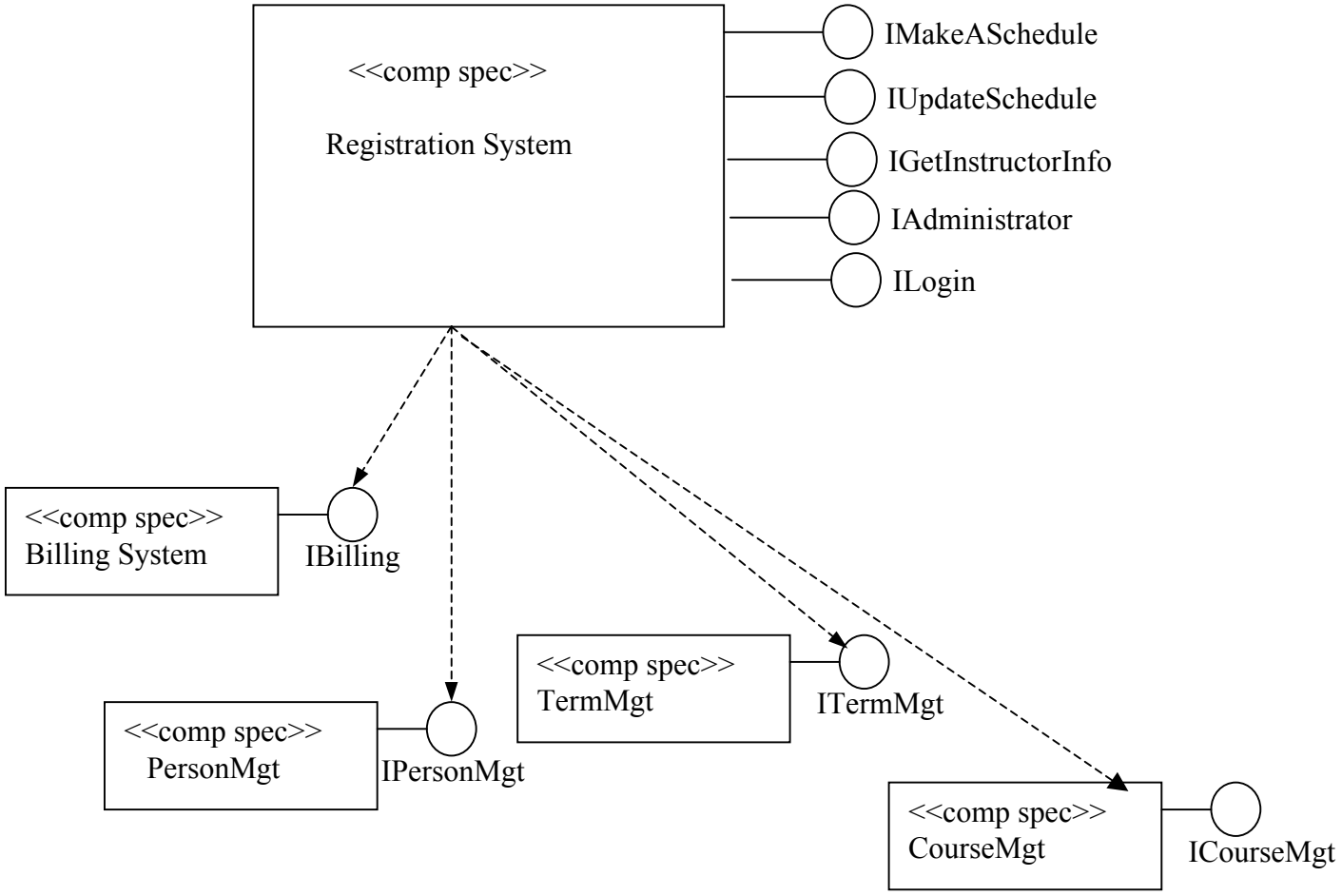
Core Type

- Has business identifier, usually independent of other identifiers
- Exists independently, no mandatory relationships except with categorizing types

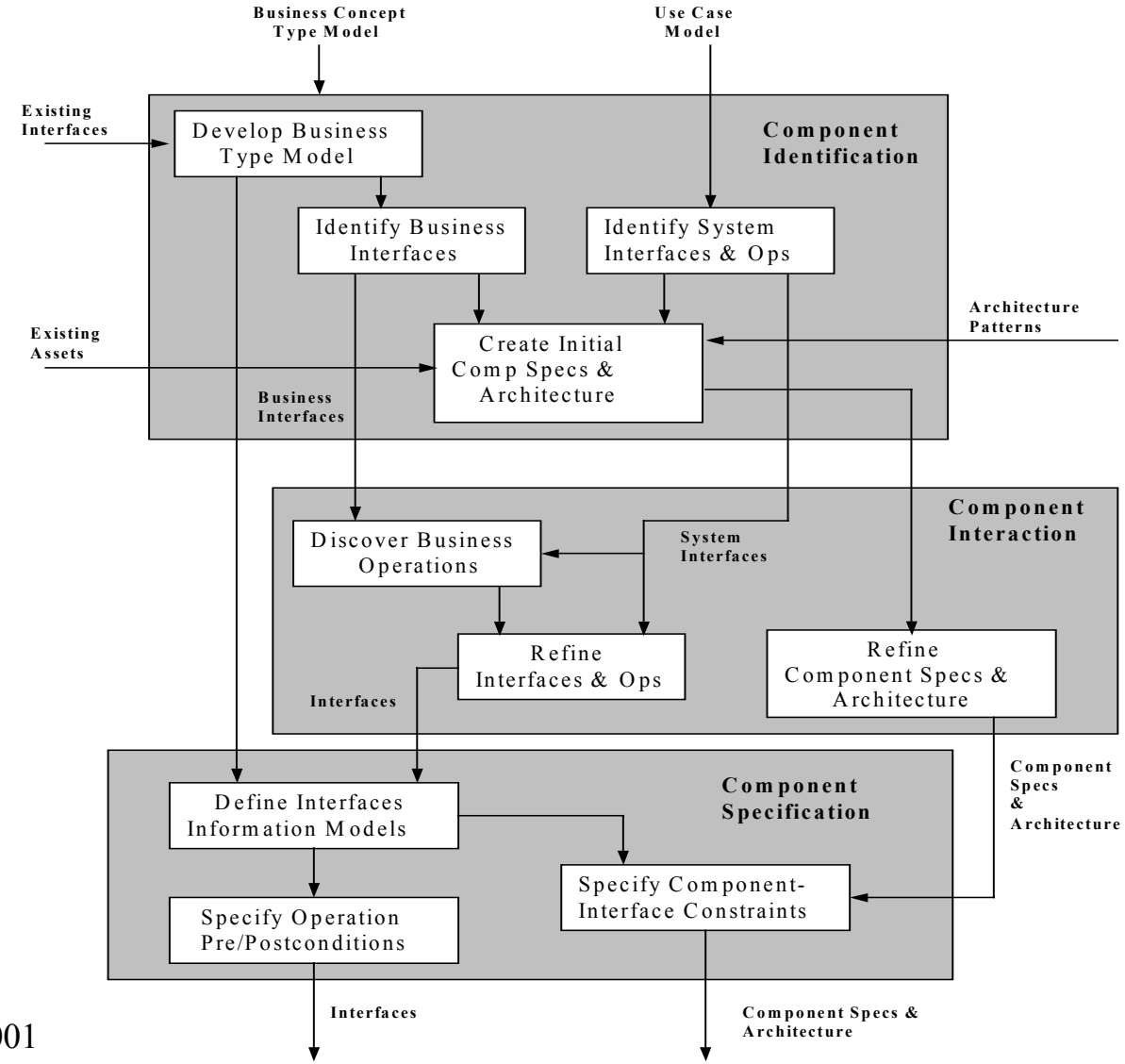
Business Type Model with Core Types



Initial Component Architecture



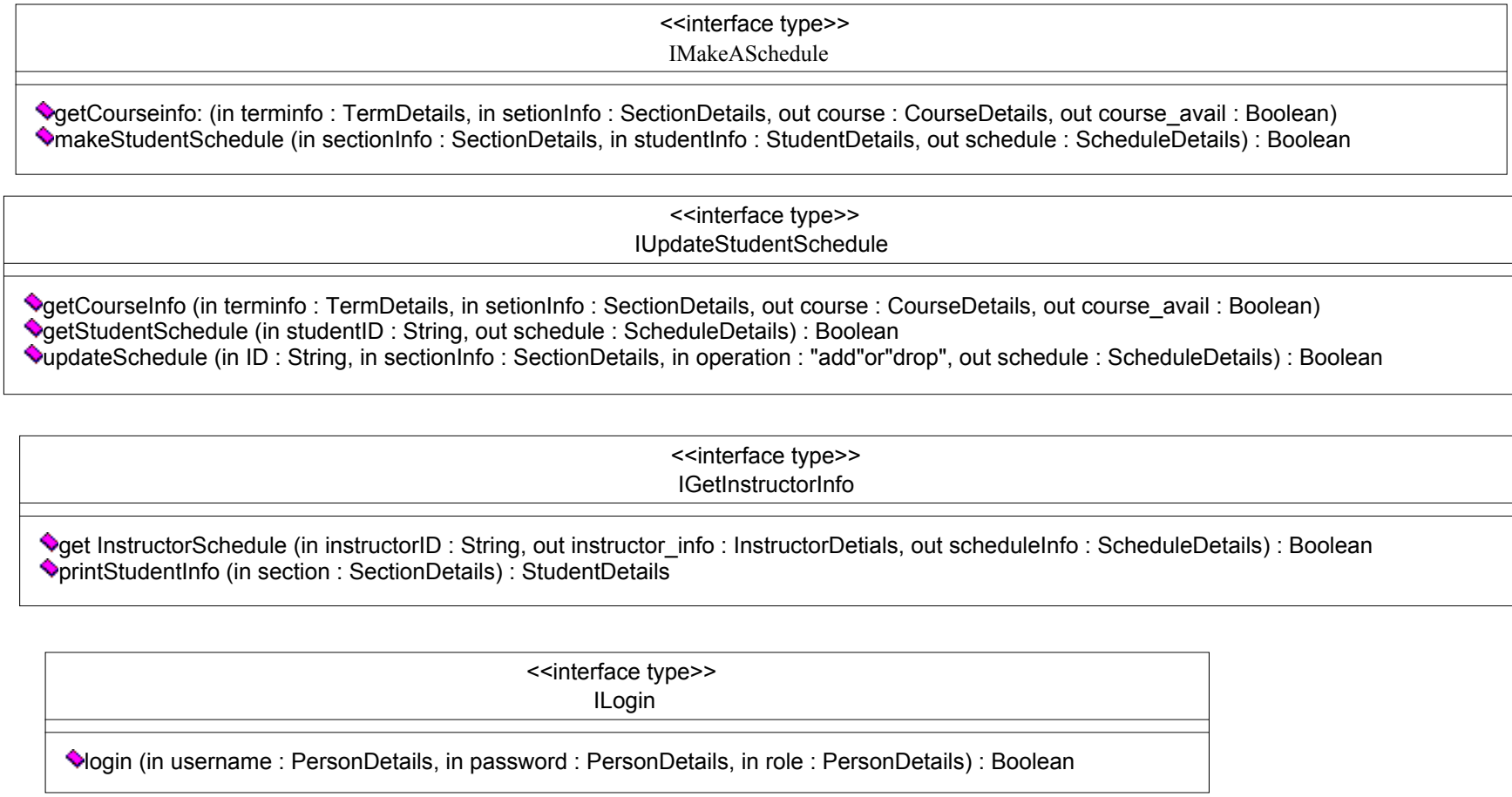
Specification Phase (2)



Component Interaction Stage

- Develop interaction model for each system interface operation by walking thru scenarios
- Discover business interface operations and their signatures
- Define any needed component object architecture constraints
- Refine and factor the interfaces

System Interface Operation Signatures



Business Operation Signatures

<<interface type>>
IPersonMgt

- addNewStudent (in student : StudentDetails) : Boolean
- updateStudentInfo (in studnet : StudentDetails, out student : StudentDetails) : Boolean
- getStudentDetails (in ID : studentID) : studentDetails
- makeStudentschedule (in sectionInfo : SectionDetails, in studentInfo : StudentDetails, out schedule : ScheduleDetails) : Boolean)
- updateStudentSchedule (in ID : String, in sectionInfo : SectionDetails, in operation : "add"or"drop", out schedule : ScheduleDetails) : Boolean
- addNewInstructor (in instructor : InstructorDetails) : Boolean
- updateInstructorInfo (in instructor : InstructorDetails, out instructor : details) : Boolean
- getInstructorDetails (in ID : InstructorID) : instructorDetails
- makeInstructorschedule (in sectionInfo : SectionDetails, in instructornfo : InstructorDetails, out schedule : ScheduleDetails) : Boolean)
- updateInstructorSchedule (in ID : String, in sectionInfo : SectionDetails, in operation : "add"or"drop", out schedule : ScheduleDetails) : Boolean

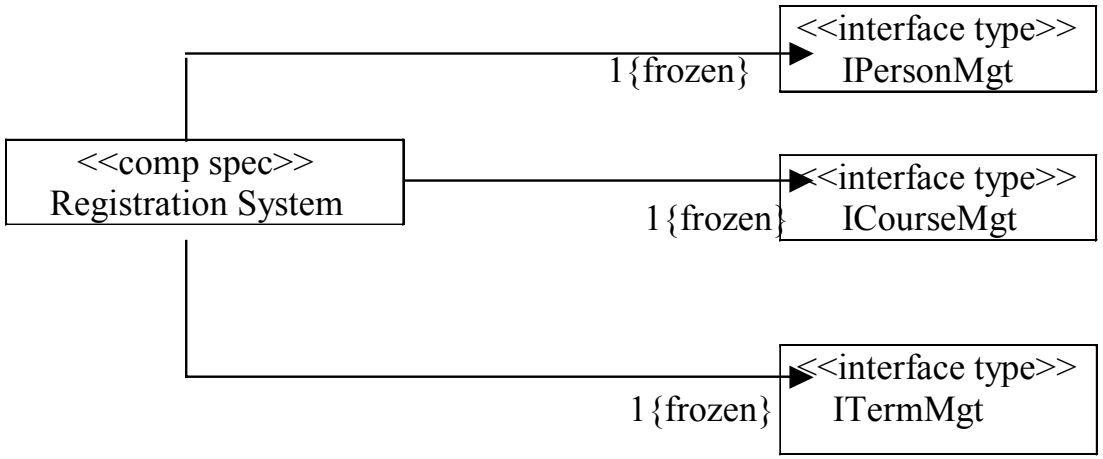
<<interface type>>
ITermMgt

- getTermInfo (out term : TermDetails)
- newSection (in termC : termNo, in course : CourseDetails, in sectionNo, in classroom, out sectionInfo : SectionDetails) : Boolean
- updateSection (in sectioninfo : SectionDetails, out sectionInfo : SectionDetails) : Boolean

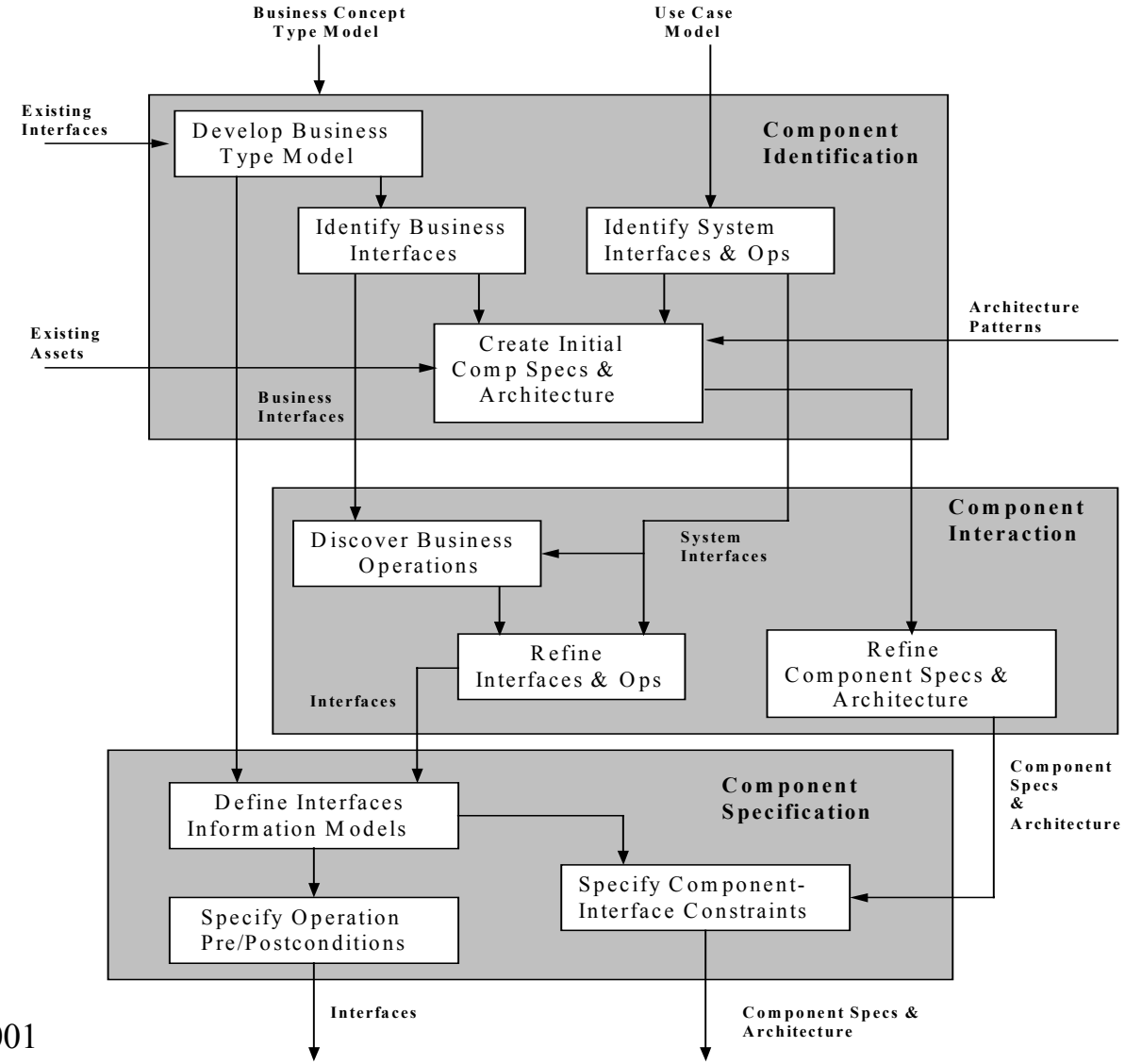
<<interface type>>
ICourseMgt

- getCourseInfo (out couseinfo : CourseDetails)
- newCourse (in course : CourseDetails) : Boolean
- updateCourse (in courseCode : CourseNo, out courseInfo : CourseDetails) : Boolean

Constraints on Component Objects



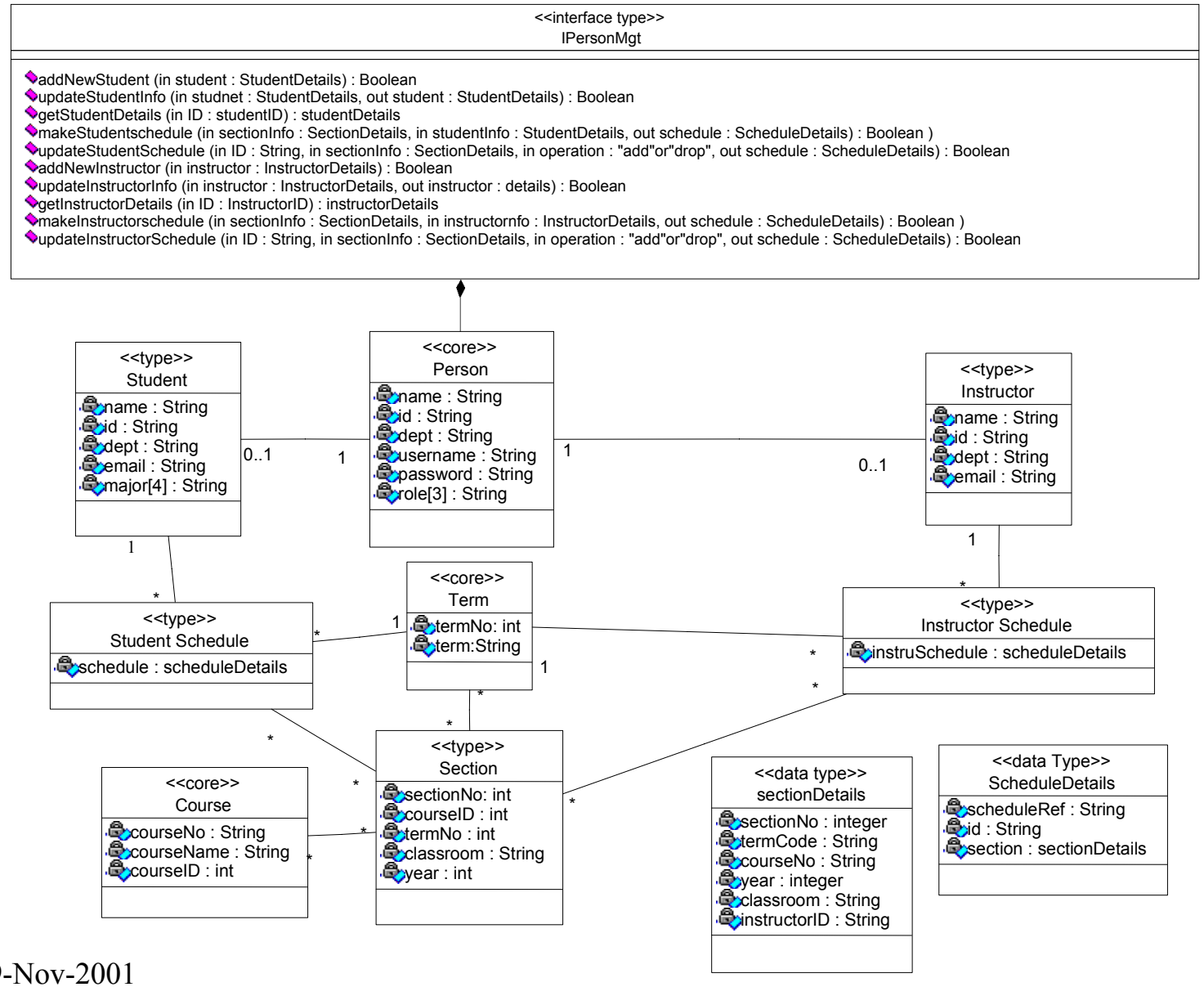
Specification Phase (3)



Component Specification Stage

- Refine each interface specification
  - define interface information model (subset of business type model)
  - specify component invariants using OCL
  - specify operation preconditions and postconditions using OCL
- Add implementation constraints to components
- Define component interaction constraints

Interface Information Model



OCL Operation Specification

IPersonMgt::makeStudentSchedule(in sectioninfo:sectionDetails, in studentinfo:studentDetails, out schedule:scheduleDetails):Boolean

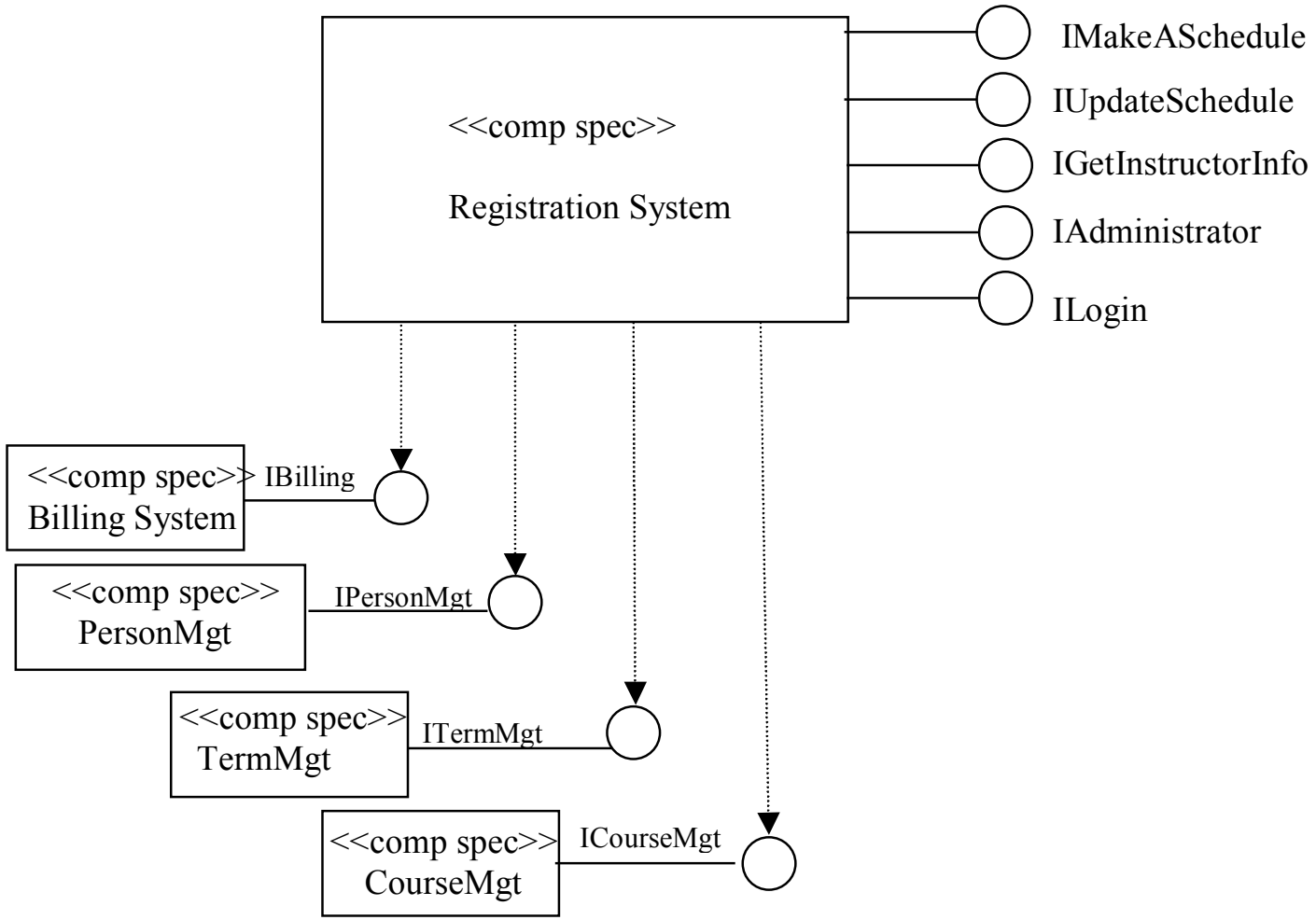
Pre:

```
----------------section and student information are valid
Course ->exists(c|c.id = sectioninfo.courseId)     and
Term->exists(t|t.termNo = sectioninfo.termNo)   and
Section -> exists (se|se.sectionNo = sectioninfo.sectionNo)  and
Person->exists(z|id = studentinfo.studentID) and
Student ->exists(y|id = studentinfo.studentID)
```

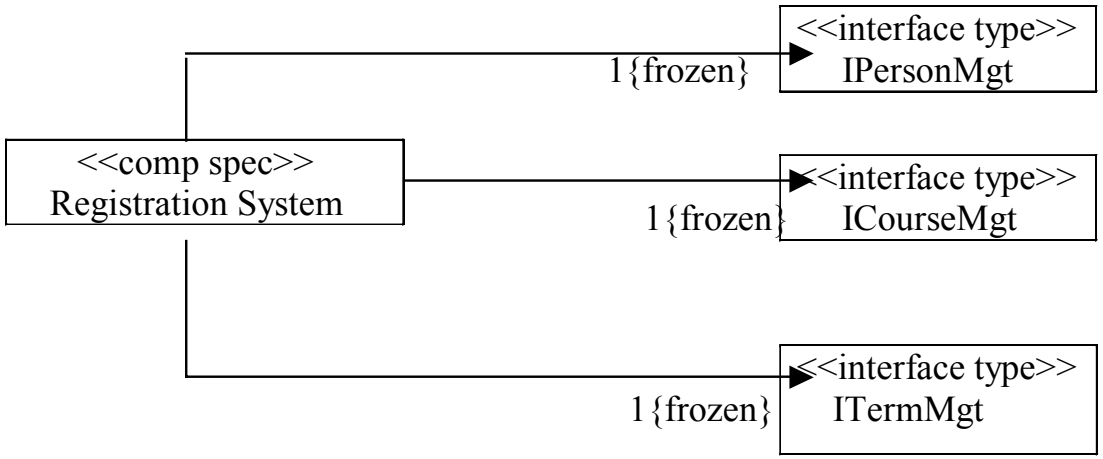
Post:

```
Result implies
  StudentSchedule@pre->forall(x|x.scheduleRef <> schedule.scheduleRef)
  and
  let s = ( StudentSchedule – StudentSchedule@pre)->
                asSequence->first in
        s.schedule.scheduleRef = schedule.scheduleRef      and
        s.schedule.id = schedule.id   and
        schedule.id = studentInfo.studentID   and
        s.schedule.section = schedule.section    and
        schedule.section = sectioninfo.section
```

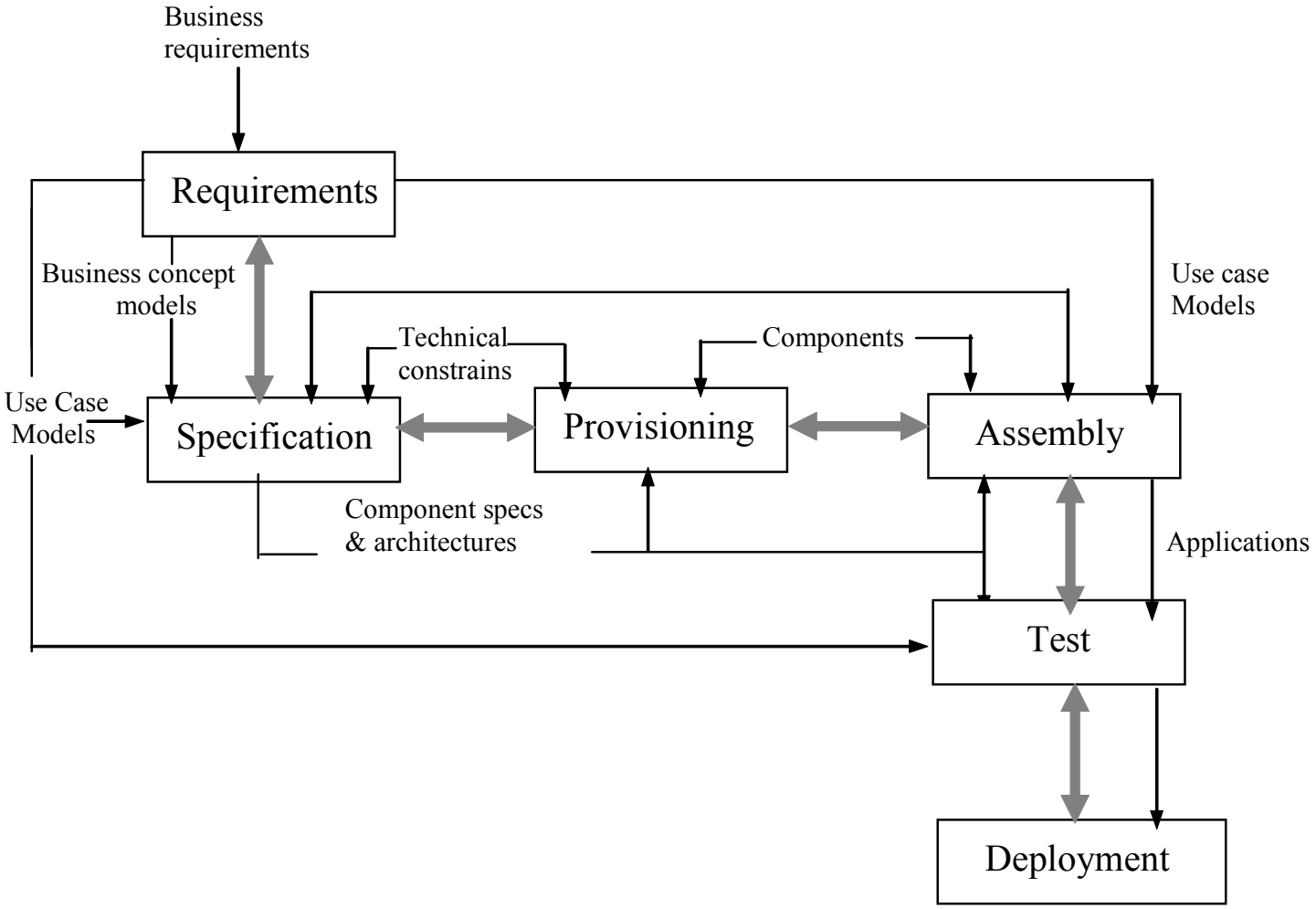
Component Specification Diagram



Component Specification Constraints



Development Workflow Revisited



Summary

- Develop a software components course
  - emphasizing rigorous, systematic methods
  - using standard notations UML and OCL
  - applying J2EE technologies
- Reinforce course with example systems
  - illustrating faithful application of the methods
  - supporting extension and modification by students
  - providing means for investigation of development methods