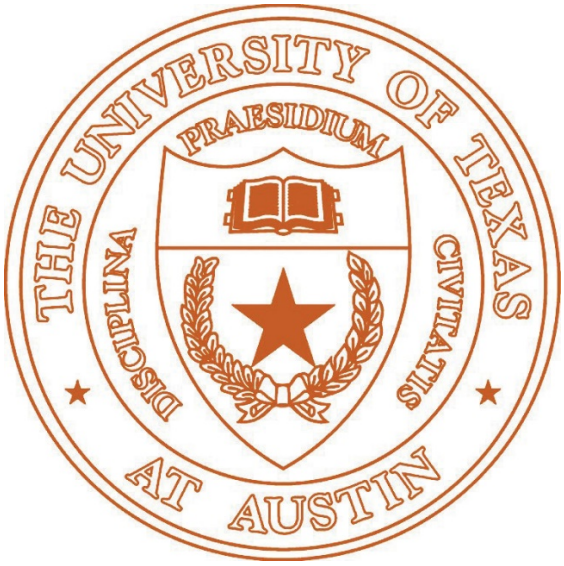
University of Texas at Austin, Cockrell School of Engineering
Requirements Engineering – EE 382C.11

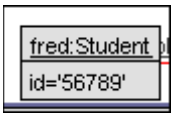


Assignment # 4
Formal Requirements
December 04, 2015

Gabrielson Eapen
EID: EAPENGP

Deliverable 4.1.1

The “University::StudentIdMustBeLength4” invariant specifies that students who are enrolled at a “University” (per the defined association) must have their attribute called “id” (a string value) have a length of exactly 4 characters. In this case the instance of student “fred” shown below has an “id” attribute that is five characters long (length=5) and violates the constraint specified by the invariant.



Deliverable 4.1.2

Added the following invariant to CR1-1.use file per the specification provided:

```
context University
    -- A student's id number must be exactly
    -- four characters long
    inv StudentIdMustBeLength4:
       self.students->forAll(s|s.id.size() = 4)
    -- A student's id number must be unique
    -- in a university
    inv StudentIdMustBeUnique:
        self.students->isUnique(s|s.id)
```

Deliverable 4.1.3

Invariant constraint specified in Deliverable 4.1.2 is working correctly

```
checking invariant (2) `University::StudentIdMustBeUnique': FAILED.
  -> false : Boolean
Results of subexpressions:
  University.allInstances : Set(University) = Set{@ut}
  self : University = @ut
  self.students : Set(Student) = Set{@fred,@sam,@sue}
  s : Student = @fred
  s.id : String = '56789'
  s : Student = @sam
  s.id : String = '1234'
  s : Student = @sue
  s.id : String = '1234'
  self.students->isUnique(s : Student | s.id) : Boolean = false
  University.allInstances->forAll(self : University | self.students->isUnique(s
: Student | s.id)) : Boolean = false
checked 2 invariants in 0.005s, 2 failures.
use>
```

Deliverable 4.1.4

Added the following invariant to CR1-2.use file per the specification provided:

```
context University
    -- A student can be a Graduate student
    -- or an UnderGraduate student but not
    -- both
    inv StudentTypeGradorUndergradNotBoth:
        self.undergraduates->excludesAll(self.graduates)
```

Deliverable 4.1.5

Invariant constraint specified in Deliverable 4.1.4 is working correctly

```
checking invariant (1) `University::StudentTypeGradorUndergradNotBoth': FAILED.
  -> false : Boolean
Results of subexpressions:
  University.allInstances : Set(University) = Set{@ut}
  self : University = @ut
  self.undergraduates : Set(Student) = Set{@sam}
  self : University = @ut
  self.graduates : Set(Student) = Set{@sam}
  self.undergraduates->excludesAll(self.graduates) : Boolean = false
  University.allInstances->forAll(self : University | self.undergraduates->exclu
desAll(self.graduates)) : Boolean = false
checked 1 invariant in 0.004s, 1 failure.
use>
```

Deliverable 4.1.6

Added the following invariant to CR1-3.use file per the specification provided:

```
context University
    -- A student cannot be registered for more
    -- than maxApprovedSemesterHours
    -- Assume all courses 3 hours
    inv StudentDoesNotExceedmaxApprovedHours:
        self.students->forAll(s|s.takingCourses->size *3 <= s.maxApprovedSemesterHours)
```

Deliverable 4.1.7

Invariant constraint specified in Deliverable 4.1.4 is working correctly

```
checking invariant (1) `University::StudentDoesNotExceedmaxApprovedHours': FAILE
D.
  -> false : Boolean
Results of subexpressions:
  University.allInstances : Set(University) = Set{@ut}
  self : University = @ut
  self.students : Set(Student) = Set{@sam}
  s : Student = @sam
  s.takingCourses : Set(Course) = Set{@BUS311,@CS306,@E306,@EE302,@EE323,@EE338,
@EE379K}
  s.takingCourses->size : Integer = 7
  3 : Integer = 3
  (s.takingCourses->size * 3) : Integer = 21
  s : Student = @sam
  s.maxApprovedSemesterHours : Integer = 18
  ((s.takingCourses->size * 3) <= s.maxApprovedSemesterHours) : Boolean = false
  self.students->forAll(s : Student | ((s.takingCourses->size * 3) <= s.maxAppro
vedSemesterHours)) : Boolean = false
  University.allInstances->forAll(self : University | self.students->forAll(s :
Student | ((s.takingCourses->size * 3) <= s.maxApprovedSemesterHours))) : Boolea
n = false
checked 1 invariant in 0.003s, 1 failure.
use>
```

Deliverable 4.2.1

Revised contents of CR2.use file with required preconditions and post conditions as well as and additional invariant which I took as implied that every student must be registered for at least one course.

```
-- OCL constraints

constraints

context University
    -- A student must be registered for atleast one class (implied not required)
    inv AllStudentsRegisteredAtleastOneCourse:
        self.students->forAll(s|s.takingCourses->size >= 1)

context Student::drop(c:Course)
    --PRE Conditions
        -- Pre-Condition 1: Student has previously registered
        --                  for course to be dropped
        pre dropPre1:
               self.takingCourses->includes(c)

        -- Pre-Condition 2: Before the drop is processed, the student
        --                  must be registered for more than one class
        pre dropPre2:
               self.takingCourses->size > 1

    --POST Conditions
        -- Post-Condition 1: Student no longer registered for dropped
        --                   course and that is only course dropped
        post dropPost1:
               self.takingCourses->excludes(c) and
                   self.takingCourses@pre->excluding(c) = self.takingCourses

        -- Post-Condition 2: the course is no longer considered full
        post dropPost2:
               c.isFull = false


        -- Post-Condition 3: Student remains enrolled at university
        --                   after drop is processed
        post dropPost3:
               self.isEnrolledAt@pre = self.isEnrolledAt

        -- Post-Condition 4: Only this student is removed from the
        --                   course
        post dropPost4:
               c.studentsEnrolled@pre->excluding(self) = c.studentsEnrolled
```

Deliverable 4.2.2

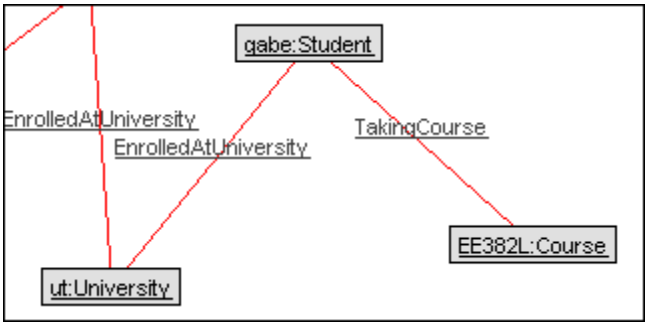
Scenario file with Thread of Execution plus some additional operations to validate the pre/post conditions.

```
-- Define minimum needed objects
!create ut : University
!create EE302 : Course
!create EE380 : Course
!create EE382N : Course
!create EE382L : Course
!create E306 : Course
!create EE338 : Course
!create EE323 : Course

!create gabe : Student
!insert (gabe,ut) into EnrolledAtUniversity
!insert (gabe,EE382N) into TakingCourse
!insert (gabe,EE382L) into TakingCourse

!create tom : Student
!insert (tom,ut) into EnrolledAtUniversity
!insert (tom,EE382N) into TakingCourse
!insert (tom,EE380) into TakingCourse
!insert (tom,E306) into TakingCourse

!set EE382N.isFull := true

--Thread of execution
-- 1: Should Succeed
!openter gabe drop(EE382N)
!delete (gabe,EE382N) from TakingCourse
!set EE382N.isFull := false
!opexit

-- 2: Should fail pre-condition as no longer registered
!openter gabe drop(EE382N)
!opexit

-- 3: Should fail pre-condition as only registered in one course now
!openter gabe drop(EE382L)
!opexit
```

Deliverable 4.2.3

Invariant constraints specified in Scenario files working correctly. Also has outputs beyond thread of execution and the optional invariant validation I put in 4.2.1

```
use> open CR2.use
use> read CR2.cmd
CR2.cmd> -- Define minimum needed objects
CR2.cmd> !create ut : University
CR2.cmd> !create EE302 : Course
CR2.cmd> !create EE380 : Course
CR2.cmd> !create EE382N : Course
CR2.cmd> !create EE382L : Course
CR2.cmd> !create E306 : Course
CR2.cmd> !create EE338 : Course
CR2.cmd> !create EE323 : Course
CR2.cmd>
CR2.cmd> !create gabe : Student
CR2.cmd> !insert (gabe,ut) into EnrolledAtUniversity
CR2.cmd> !insert (gabe,EE382N) into TakingCourse
CR2.cmd> !insert (gabe,EE382L) into TakingCourse
CR2.cmd>
CR2.cmd> !create tom : Student
CR2.cmd> !insert (tom,ut) into EnrolledAtUniversity
CR2.cmd> !insert (tom,EE382N) into TakingCourse
CR2.cmd> !insert (tom,EE380) into TakingCourse
CR2.cmd> !insert (tom,E306) into TakingCourse
CR2.cmd>
CR2.cmd> !set EE382N.isFull := true
CR2.cmd>
CR2.cmd> --Thread of execution
CR2.cmd> -- 1: Should Succeed
CR2.cmd> !openter gabe drop(EE382N)
precondition `dropPre1' is true
precondition `dropPre2' is true
CR2.cmd> !delete (gabe,EE382N) from TakingCourse
CR2.cmd> !set EE382N.isFull := false
CR2.cmd> !opexit
postcondition `dropPost1' is true
postcondition `dropPost2' is true
postcondition `dropPost3' is true
postcondition `dropPost4' is true
CR2.cmd>
CR2.cmd> -- 2: Should fail pre-condition as no longer registered
CR2.cmd> !openter gabe drop(EE382N)
precondition `dropPre1' is false
precondition `dropPre2' is false
CR2.cmd> !opexit
Error: Call stack is empty.
CR2.cmd>
CR2.cmd> -- 3: Should fail pre-condition as only registered in one course now
CR2.cmd> !openter gabe drop(EE382L)
precondition `dropPre1' is true
precondition `dropPre2' is false
CR2.cmd> !opexit
Error: Call stack is empty.
CR2.cmd>
use> check -v
checking structure...
checking invariants...
checking invariant (1) `University::AllStudentsRegisteredAtleastOneCourse': OK.
checked 1 invariant in 0.002s, 0 failures.
use>
```


gabe:Student
EnrolledAtUniversity
EnrolledAtUniversity
TakingCourse
EE382L:Course
ut:University