University Example Use Cases

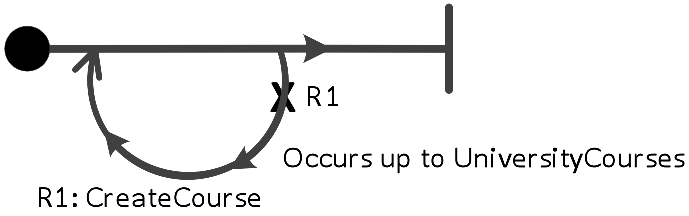
In the University example we can identify the following use cases which are mapped to the scenarios in the ACL University example contracts.

Use Case	Actor/s	Input	Output
Create Courses	University	Number of courses	A list of courses.
Create Students	University	Number of students, Student type (part-time/full-time).	A list of students.
Create a term	University	Number of courses, number of students, student type.	List of courses, list of students.
Register for courses	Student, University, Course	Student type, selected course/s, number of courses allowed to register in.	List of courses which student registered in.
Take courses	Student, Course	Term started, number of assignments if required, number of midterms if required, is project required, is final required, selected courses to drop.	List of courses taken, list of courses dropped.
Report marks	Student, Course, University	List of students, marks.	Student marks.

Therefore we will create a UCM for each scenario in the University example

Scenarios UCMs

1. CreateCourses



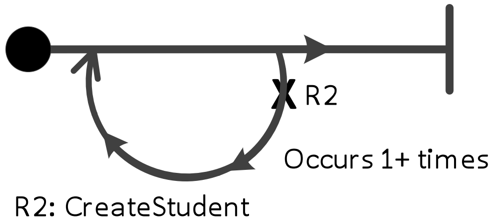
   a. Operational Variables required for this scenario:

Variable Name	Motivation
UniversityCourses	This variable defines the number of courses that can be created at the university; ranges from 1 to 100, that is a university must have at least 1 course and no more than 100 courses. We will use the min/mx values of this variable to flatten the loop.

b. Path sensitization variables testing table:

Variant	Operational Variables	Expected Results
	UniversityCourses	
Path.1.1	1	1 course will be created.
Path.1.2	100	100 courses will be created.

2. CreateStudents



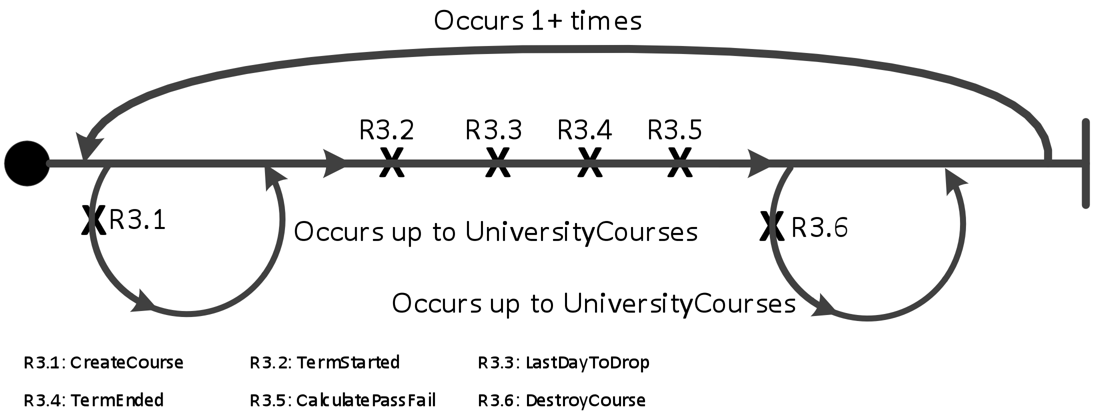
a. Operational Variables required for this scenario:

	Variable Name	Motivation
	NumberOfStudents	We will need to create this SPV to control the number of students created in university.

b. Path sensitization variables testing table:

Variant	Operational Variables	Expected Results
	NumberOfStudents	
Path.1.1	Min (i.e. 1).	At least 1 student is created.
Path.1.2	Max.	This should be provided by user.

3. Term



This is the most important scenario as it will trigger all other scenarios on all contracts in the University example, however looking at the scenario grammar solely without considering the responsibilities called from within the scenario we can identify one PSV.

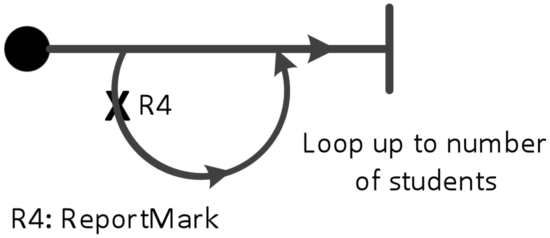
a. Operational Variables required for this scenario:

Variable Name	Motivation
UniversityCourses	This variable defines the number of courses that can be created and destroyed at the university; ranges from 1 to 100, that is a university must have at least 1 course and no more than 100 courses. We will use the min/mx values of this variable to flatten the loop.

b. Path sensitization variables testing table:

Variant	Operational Variables	Expected Results
	UniversityCourses	
Path.1.1	1	1 course will be created and later on destroyed.
Path.1.1	100	100 courses will be created and later on destroyed.

4. ReportMarks



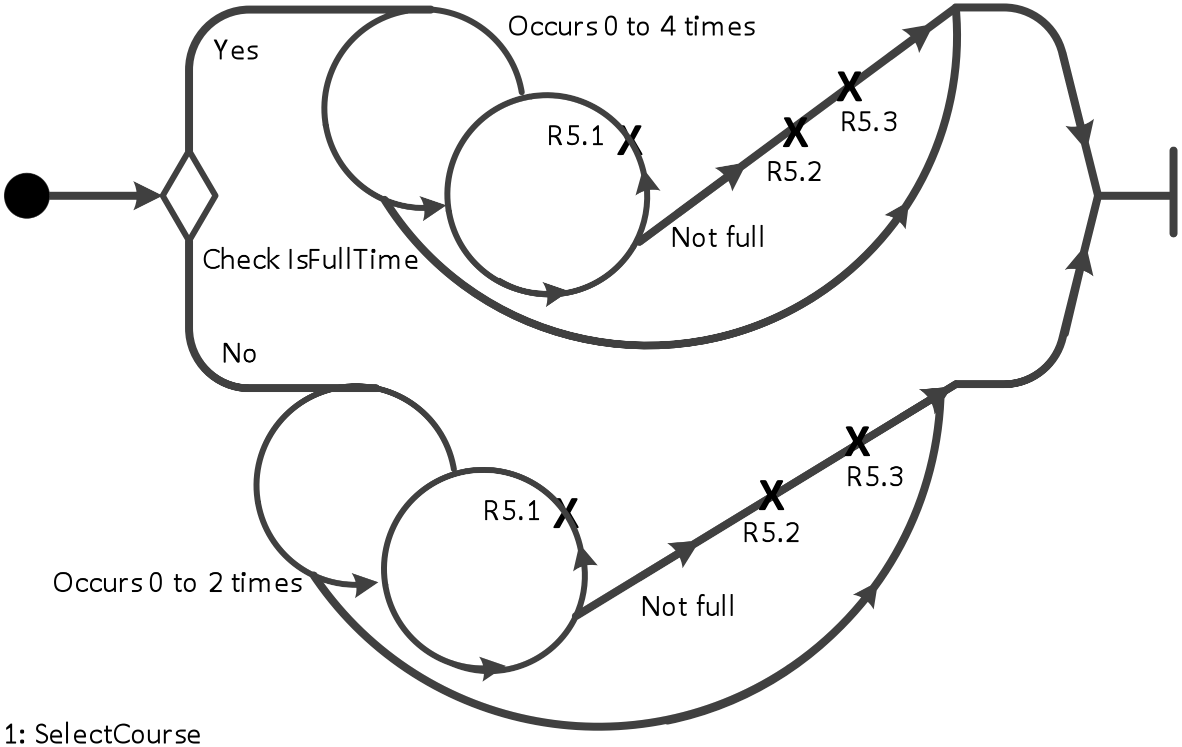
a. Operational Variables required for this scenario:

Variable Name	Motivation
NumberOfStudents	This variable holds the number of students registered in a course (i.e. Students().Length()). The range of this variable value is 1 up to course capsize. We will use the min/mx values of this variable to flatten the loop.

c. Path sensitization variables testing table:

Variant	Operational Variables	Expected Results
	NumberOfStudents	
Path.1.1	1	ReportMark will be reported for 1 student in this course.
Path.1.2	Capsize	ReportMark will be reported for all students (capsize) in this course.

5. RegisterForCourses



R5.1: SelectCourse
R5.2: RegisterStudentForCourse
R5.3: RegisterCourse

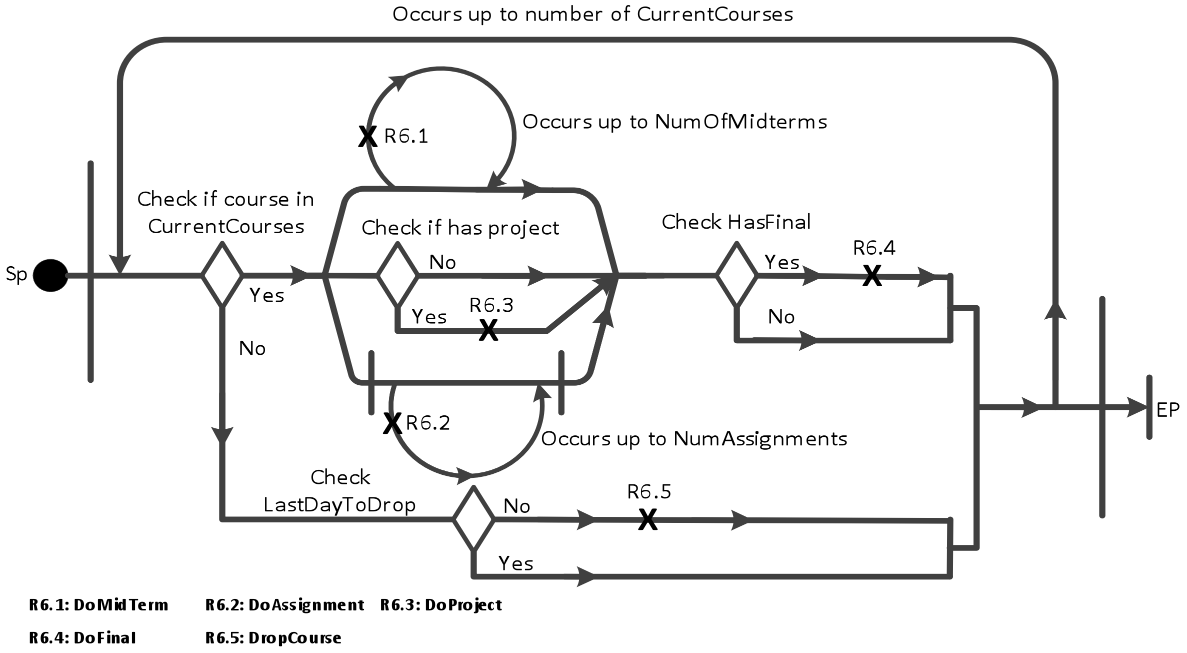
a. Operational Variables required for this scenario:

Variable Name	Motivation
IsFullTime	This variable is a Boolean indicates whether a student is full time or part time and therefore it sets two different paths.
NumberOfCourses	This variable holds the number of courses successfully registered in for a student, the variable value has a constraint on the type of student (i.e. full time or part time) which will set the maximum number of courses allowed for this student to register in. We will use the min/mx values of this variable to flatten the loop.
NumberOfRedoIterations	This variable holds the number of iterations of the redo statement to find a non-full course.

b. Path sensitization variables testing table:

Variant	Operational Variables			Expected Results
	IsFullTime	NumberOfCourses	NumberOfRedoIterations	
Path.1.1 (SP → EP)	Yes	0	0	Full time student is not registered in any course.
Path.1.2 (SP→R5.1→R5.2→R5.3)	Yes	4	Courses immediately available.	Full time student registered in the maximum number of courses allowed for her/him in a term and all selected courses were immediately available (i.e. not full).
Path.1.3 (SP→R5.1→R5.2→R5.3)	Yes	4	Coursers selected only after the maximum number of retries has been attempted.	Full time student registered in the maximum number of courses allowed for her/him in a term and all selected courses were selected after the maximum number of retries has been attempted (i.e. full courses).
Path.2.1 (SP → EP)	No	0	0	Part time student is not registered in any course.
Path.2.2 (SP→R5.1→R5.2→R5.3)	No	2	Courses immediately available.	Part time student registered in the maximum number of courses allowed for her/him in a term and all selected courses were immediately available (i.e. not full).
Path.2.3 (SP→R5.1→R5.2→R5.3)	No	2	Coursers selected only after the maximum number of retries has been attempted.	Part time student registered in the maximum number of courses allowed for her/him in a term and all selected courses were selected after the maximum number of retries has been attempted (i.e. full courses).

6. TakeCourses



a. Operational Variables required for this scenario:

Variable Name	Motivation
NumMidterms	This variable holds the number of midterms in a course, the value ranges from 0 to 2 and the default is 1, we will use the min/mx values of this variable to flatten the loop.
NumAssignments	This variable holds the number of assignments in a course, the value ranges from 0 to 5 and the default is 1, we will use the min/mx values of this variable to flatten the loop.
HasProject	This variable is a Boolean that indicates whether this course has a project or not. This variable sets two different paths on the UCM.
HasFinal	This variable is a Boolean that indicates whether this course has a final exam or not. This variable sets two different paths on the UCM.
NumberOfCurrentCourses	This variable holds the number of course instances (i.e. CurrentCourses().Length()). This is equivalent to the maximum number of courses a student can take in a term (i.e. 4 for full time

	students and 2 for part time students).
NumberOfConcurrentInstances	This variable holds the number courses can have their requirements be concurrent.
LastDayToDrop	This variable creates 2 different paths UCM.
CourseFound	This variable creates 2 different paths UCM.

b. Path sensitization variables testing matrix: for this scenario I will split the test scenarios in multiple tables:

   i. One instance test results.

Variant	**Operational Variables**						**Expected Results**
	CourseFound	LastDayToDrop	NumMidterms	NumAssignments	HasProject	HasFinal	
Path.1 SP→EP	Yes	No	Min (i.e. 0)	N/A	N/A	No	No midterms and no project.
Path.1.1 SP→R6.4→EP	Yes	No	Min (i.e. 0)	N/A	N/A	Yes	No midterms but have a project.
Path.1.2 SP→R6.3→EP	Yes	No	Max (i.e. 2)	N/A	N/A	No	Have 2 midterms and no project.
Path.1.3 SP→R6.1→6.4→EP	Yes	No	Max (i.e. 2)	N/A	N/A	Yes	Have 2 midterms and a project.
Path.2 SP→EP	Yes	No	N/A	N/A	No	No	Have no project and no final.
Path.2.1 SP→R6.4→EP	Yes	No	N/A	N/A	Yes	No	Have project but no final.
Path.2.2 SP→R6.3→R6.4→EP	Yes	No	N/A	N/A	Yes	Yes	Have project and a final.
Path.3 SP→EP	Yes	No	N/A	0	N/A	No	Have 0 assignments and no final.
Path.3.1 SP→R6.4→EP	Yes	No	N/A	0	N/A	Yes	Have 0 assignments and a final.
Path.3.2 SP→R6.2→EP.	Yes	No	N/A	Max (i.e. 5)	N/A	No	Have 5 assignments and no final.

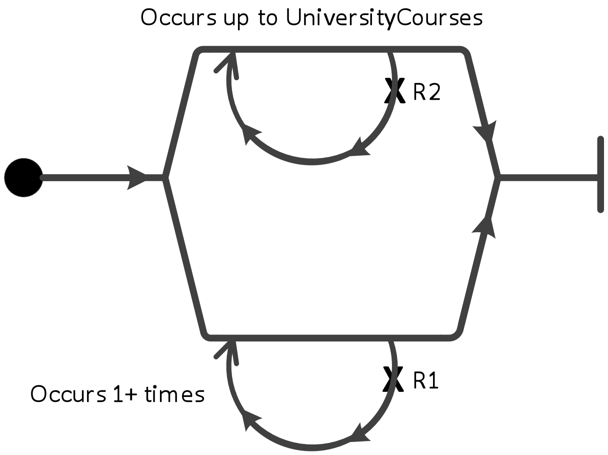
Path.3.2 SP→R6.2→R6.4 →EP.	Yes	No	N/A	Max (i.e. 5)	N/A	Yes	Have 5 assignments and a final.
Path.4 SP→EP	No	No	N/A	N/A	N/A	N/A	Course not found in students list of courses and do not drop the course
Path4.1	No	Yes	N/A	N/A	N/A	N/A	Course not found in students list of courses and drop the course

c. The Parallel block: here the whole combination in the table above will be used in each combination of this table.

Operational Variables		**Expected Results**
NumberOfCurrentCourses	NumberOfConcurrentInstances	
Min (i.e. 1)	Min (i.e. 1)	Student has only one course and therefore there is no concurrency.
Min (i.e. 1)	Max (i.e. 1)	Student has only one course and therefore there is no concurrency.
Max for part time student (i.e. 2)	Min (i.e. 1).	Part time student has max number of courses (i.e. 2) and the minimum number (i.e. 1) of concurrent courses.
Max for part time student (i.e. 2)	Max (i.e. 2)	Part time student has max number of courses (i.e. 2) and the max number (i.e. 2) of concurrent courses.
Max for full time student (i.e. 5)	Min (i.e. 1).	Full time student has max number of courses (i.e. 5) and the minimum number (i.e. 1) of concurrent courses.
Max for full time student (i.e. 5)	Max (i.e. 5)	Full time student has max number of courses (i.e. 5) and the max number (i.e. 5) of concurrent courses.

Relations UCMs

1. Relation: Creation



R1: CreateCourse

R2: CreateStudent

a. Operational Variables required for this relation:

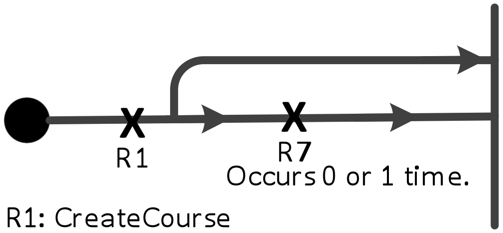
Variable Name	Motivation
UniversityCourses	This variable defines the number of courses that can be created at the university; ranges from 1 to 100. We will use the min/mx values of this variable to flatten the loop.
NumberOfStudents	We will need to create this SPV to control the number of students can be created in university.

b. Path sensitization variables testing table:

Variant	Operational Variables		Expected Results
	UniversityCourses	NumberOfStudents	
Path.1	1	N/A	1 course will be created.

SP→R2→EP			
Path.1.2 SP→R2→EP	100	N/A	100 courses will be created.
Path.2 SP→R1→EP	N/A	1	1 student is created.
Path.2.1 SP→R1→EP	N/A	Max	This should be provided by user.

2. Relation: Cancellation



R1: CreateCourse

R7: CancelCourse

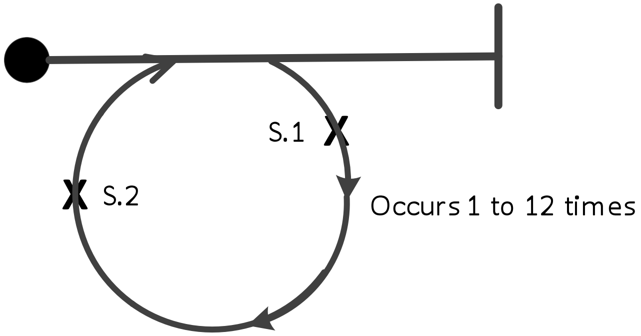
a. Operational Variables required for this relation:

Variable Name	Motivation
N/A	

b. Path sensitization variables testing table:

Variant	Operational Variables	Expected Results
Path.1 SP→R1→EP	N/A	Course created but not canceled.
Path.2 SP→R1→R7→EP	N/A	Course created and canceled.

3. Relation: Students



S1: RegisterForCourses

S2: TakeCourses

a. Operational Variables required for this relation:

Variable Name	Motivation
NumTermsToComplete	This variable defines the number of iterations to run the relation. The variable value ranges from 1 to 12. We will use the min/mx values of this variable to flatten the loop.

b. Path sensitization variables testing table:

Variant	Operational Variables	Expected Results
	NumTermsToComplete	
Path.1 SP→S1→S2→EP	Min (i.e. 1)	The relation to run 1 time, the Student's scenario “RegisterForCourses” will run one time, Student's scenario “TakeCourses” will run 1 time.
Path.1.1 SP→S1→S2→EP	Max (i.e. 12)	The relation to run 12 times, the Student's scenario “RegisterForCourses” will run one time, Student's scenario “TakeCourses” will run 12 times.

Completeness and inaccuracies

Scenarios give only a partial view of the system as they depict the system in black-box view and in user's perspective. Scenarios testing model is incomplete since specifications are not complete and iteratively augmented throughout the lifetime of the system. Scenario based testing provides only partial results.

The test cases derived from scenarios UCMs in the ACL University example provides partial results since they are only positive scenario test cases which address the model base requirements of the system and only valid PSV values relevant for testing these scenarios can be used.

When compared to a university registration system, the ACL University example courses do not span more than one term, however in a university registration system some classes span more than one term. Relation "Canceling" creates a course however it may not cancel the course. University contract defines a variable to control the number of courses created but it does not provide a variable to control the number of the students to be created at university (i.e. there is no cap on this).