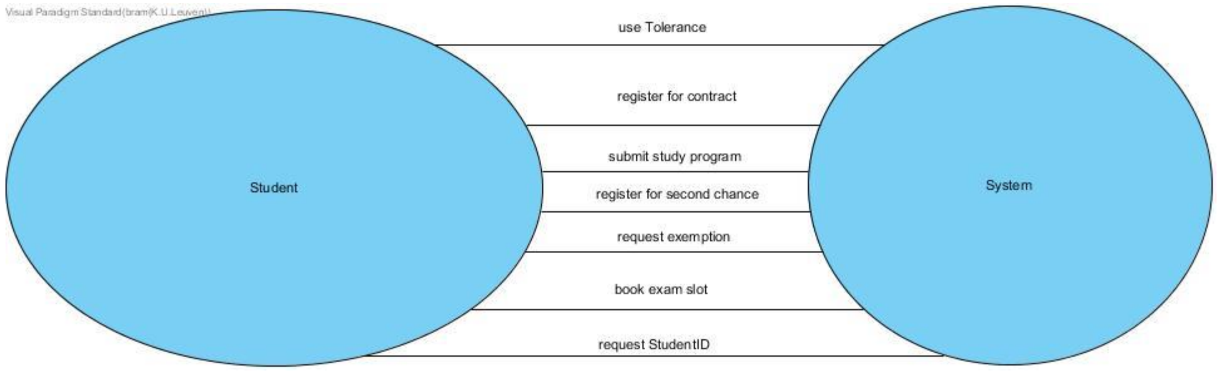
VEREISTENALYSE VOOR COMPLEXE SYSTEMEN

ACADEMIEJAAR:2017-2018

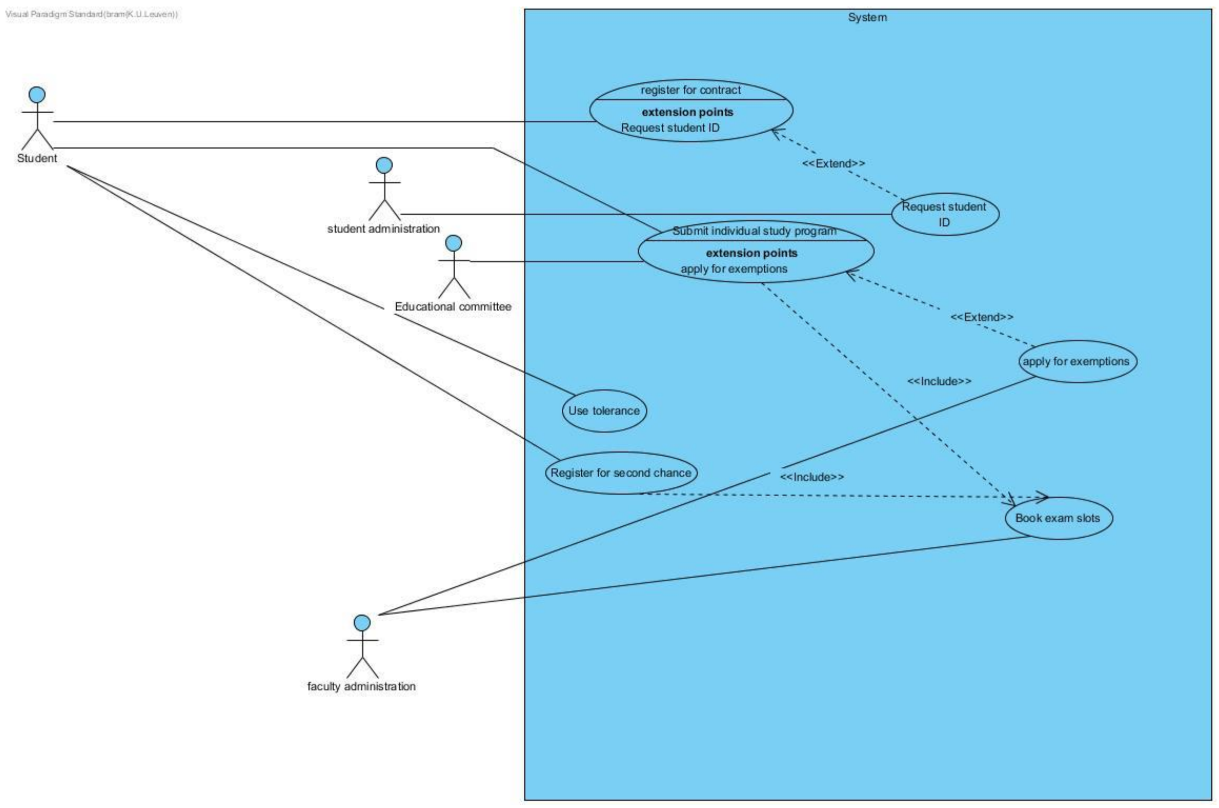
AUTEUR: BRAM BELPAIRE

Inhoudsopgave

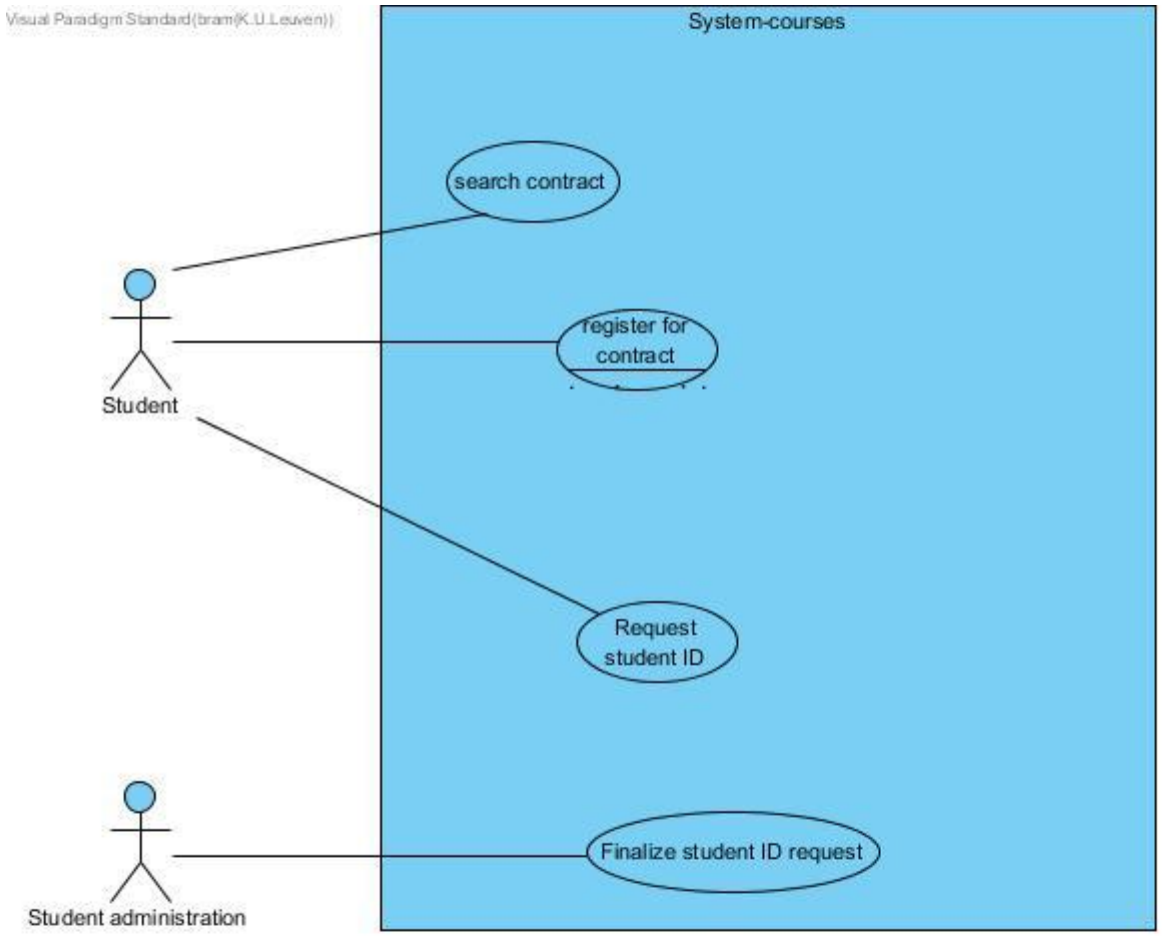
CONTEXT DIAGRAM

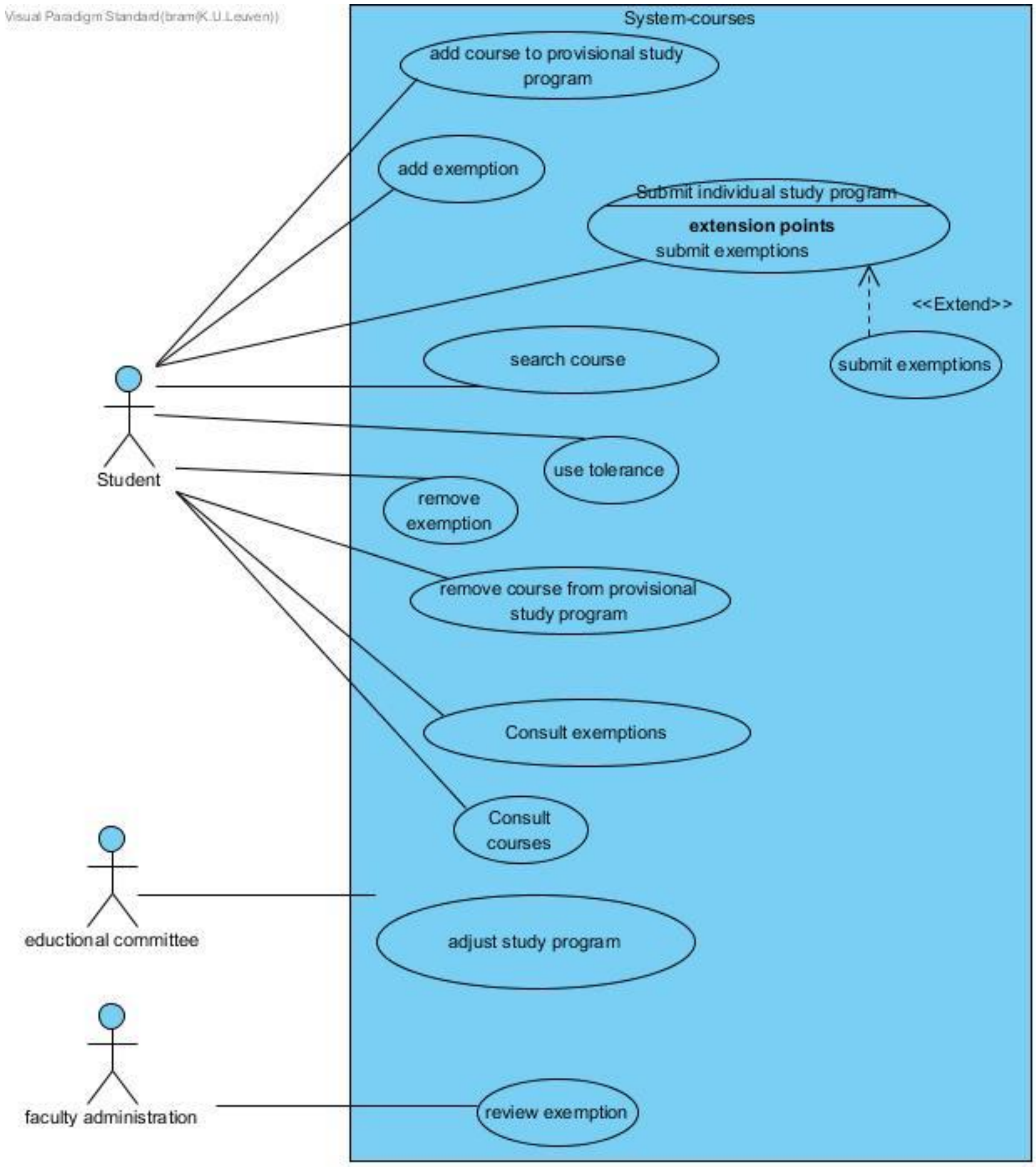


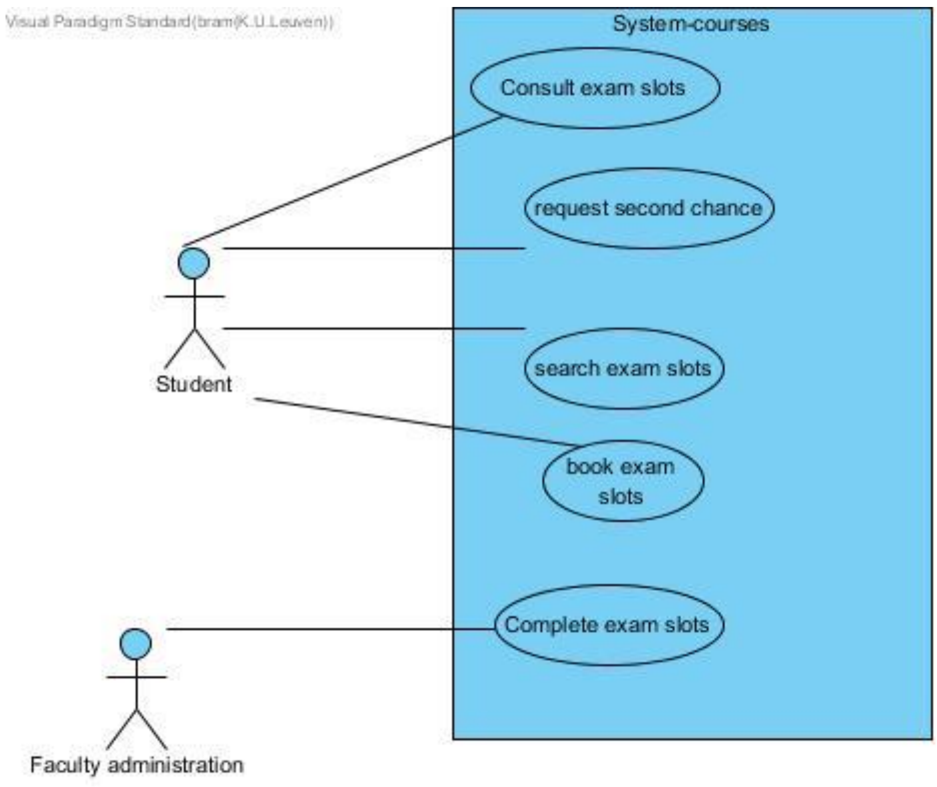
BUSINESS USE CASE DIAGRAM



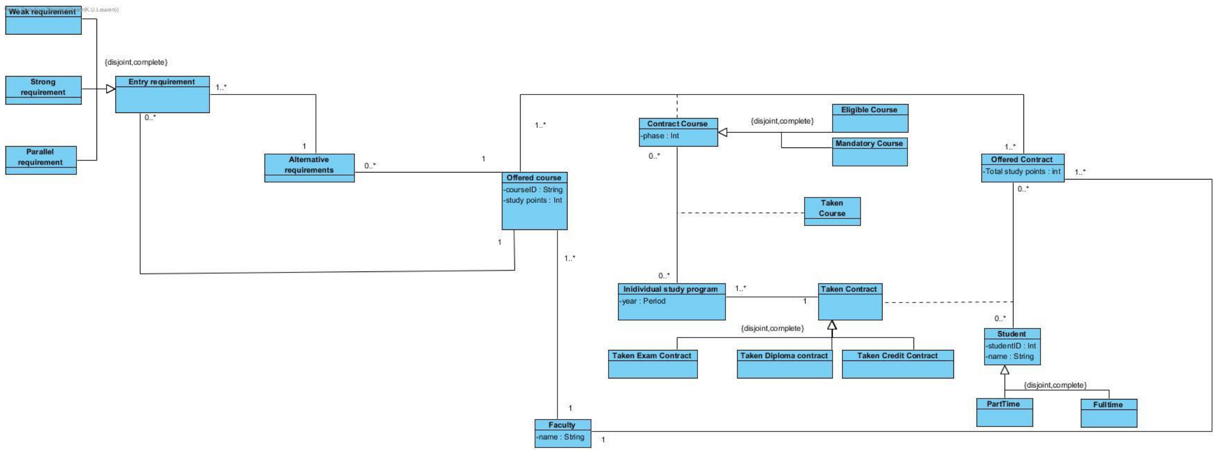
SYSTEM USE CASE DIAGRAMS

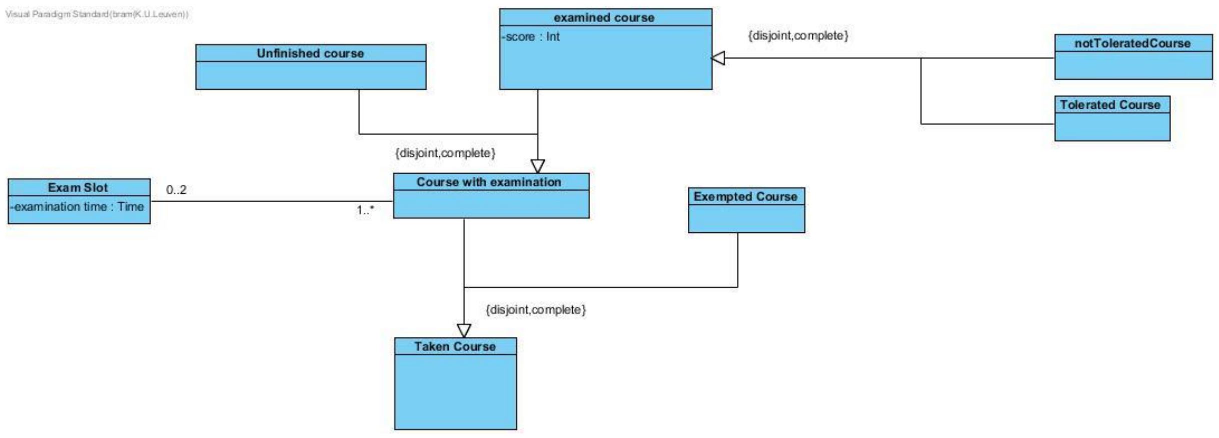



Visual Paradigm Standard(bram(K.U.Leuven))
System-courses
add course to provisional study program
add exemption
Submit individual study program
extension points
submit exemptions
<<Extend>>
submit exemptions
search course
use tolerance
remove exemption
remove course from provisional study program
Consult exemptions
Consult courses
adjust study program
review exemption
Student
eductional committee
faculty administration



DOMAIN MODEL





BUSINESS USE CASES

Register for contract

Business event: the student wishes to register for a contract and take the courses connected to the contract

Trigger: Student goes to the register form with the intent to register for a contract at the university

Preconditions: None

Active Stakeholders: Student, Faculty administration,

Interested Stakeholders: Student administration

Normal business flow:

1. Student walks in to register for a contract at the university
2. If the Student hasn't been registered yet, or returns after several years
   a. The student requests a student id at the administration (extend: Request student ID)
3. Student requests to register
4. Faculty receives a request for the contract
5. Faculty checks whether the student in question is allowed to enrol
6. Faculty sees that there is no problem and lets the student know the registering was successful
7. Student is now registered for the contract

Alternate business flow: Not applicable

Exception business flow:

Exception flow 1: The student has failed 4 examinations previous year without having a CSE above 50%

1.The faculty receives the request

2.faculty lets the student know they cannot enrol for this academic year

Exception flow 2: student failed 6 examinations within 3 academic years in a diploma contract

1.The faculty receives the request

2.faculty lets the student know they cannot enrol for the coming 3 academic years starting from when they failed the last set of examinations

post condition:

The student has been registered for the contract.

Submit individual study program

Business event: the student decides he wants to send an individual study program

Trigger: Student goes to the submit form with the intention to submit his individual study program along with his possible exemptions

Preconditions: registered for a contract

Active Stakeholders: Student, educational committee, Faculty administration

Interested Stakeholders: Not applicable

Normal business flow:

1. Student decides to submit his study program
2. Student registers an exam slot for each course (include:register exam slots)
3. If Student also has exemptions to send
    a. Student submits all exemptions for courses (extend: apply for exemptions)
4. Student submits individual study program together with any exemptions
5. The faculty administration checks whether the program is valid
6. The faculty administration finds that the program is valid and sends back feedback that it is approved
7. Student receives the message that the study program is valid

Alternate business flow: Not applicable

Exception business flow:

Exception flow 1: The administration finds that the program sent is not valid for a certain reason or multiple reasons

1.The faculty administration sends that the input given is not valid to the student and why
2. the student has to resubmit the study program, making sure it is now valid

Exception flow 2: if the student hasn't given a valid study program in three weeks, then the committee decides the study program for the student

1. Committee decides the study program and approves it
2. Committee lets the student know they decided the study program in his stead

Exception flow 3: The administration does not approve certain exemption(s)

1. Faculty notices an exemption request it does not approve
2. Faculty sends for each exemption an explanation to the student, letting them know why the exemption was not approved
3. Student has to resubmit his study program, as it is possible that the exemptions were needed for entry requirements to be satisfied

Exception flow 4: if the student hasn't booked an exam slot for each course

1. The administration selects exam slots
2. The administration lets the student know they decided the exam slots

post condition: Student has submitted an individual study program that is valid and has been approved and all sent exemptions have been approved (if any were sent)

additional information:

an individual study program is valid:

-if the student is a first-year student, he takes all courses of the first phase, unless he has exemptions for these

-the student has between 40 and 75 study points and is a full-time student

-the student has between 0 and 30 study points and is a part-time student

-the student has taken all mandatory classes that he has failed and not tolerated of previous years

-the student has taken a course and satisfies the entry requirements

-a student can only take courses, if he has taken all the mandatory courses of the previous phases

-students may not have several credit contracts or several exam contracts for courses under supervision of the same faculty.

- no unfinished course can be included in different contracts at the same time

Request student ID

Business event: student decides he is going to get a student ID

Trigger: student goes up to student administration to request the student ID

Preconditions: none

Active Stakeholders: Student, student administration

Interested Stakeholders: Not applicable

Normal business flow:
1.student walks up to the student administration
2.student requests a student ID
3.the student administration requests personal information
4.student delivers personal information
5.student administration checks if the personal information is valid
6.student administration gives the student his new unique ID along with an e-mail address

Alternate business flow:

Alternate business flow 1: If at step 6 the administration realizes the student has already been registered before:

1.the student administration reuses the student ID the student used before

<u>Exception business flow:</u>

Exception flow 1: if the student doesn't deliver all the information needed:

1.student administration notices that the student didn't send enough information
2.lets the student know they should add the information they forgot

<u>post condition</u>: The student now has received an e-mail address and his student ID

Register for second chance

<u>Business event</u>: The student wishes to retake exam(s)

<u>Trigger</u>: Student goes to the form to register for a second chance

<u>Preconditions</u>: The courses the student retakes are part of his study program

<u>Active Stakeholders</u>: Student, faculty administration

<u>Interested Stakeholders</u>: Not applicable

<u>Normal business flow:</u>

1. Student goes to the form for second chances
2. Student decides which courses to retake
3. Student books exam slots for all the courses he wishes to retake (include: register exam slots)
4. Student goes away with his exam slots booked

Alternate business flow: Not applicable

<u>Exception business flow:</u>

Exception flow 1: if the student hasn't booked an exam slot for each course

1. The administration selects exam slots
2. The administration lets the student know they decided the exam slots in his stead

<u>post condition</u>: The student has registered for a second chance for certain courses, and has booked exam slots for them

SYSTEM USE CASES

Register for contract

<u>Short description</u>: A registered student can register for a contract

<u>Preconditions</u>: the student has a student ID and a contract has been selected

<u>Primary actor</u>: Student

<u>Secondary actor</u>: Not applicable

<u>Normal business flow:</u>
1.student decides to register for a selected contract
2.the system responds to the student's request, asking for confirmation along with information about the contract
3.The student confirms
4. The system:
 a.registers the contract
 b.notifies the student the contract has been successfully registered
 c.brings the student back to where he started before registering

Alternate flow: Not applicable

<u>Exception flow:</u>
Exception flow 1: The student cancels the registration
1.system brings the student back to where he started before registering

<u>post condition</u>: The student has been registered for a contract

Request second chance

<u>Short description</u>: The student can register for a second chance, specifying the courses he wants to retake, after which he can select the exam slots for his courses he retakes

Search exam slot

<u>Short description</u>: The student can search the possible open exam slots for a given course

Book exam slot

<u>Short description</u>: The student can book an open exam slot for a given course

Consult exam slots

<u>Short description</u>: The student can consult the booked exam slots for his courses

Complete exam slots

<u>Short description</u>: The Faculty administration completes the exam slots for students that did not specify them

<u>Preconditions</u>: There is a study program where not all exam slots have been specified

<u>Primary actor</u>: Faculty administration

<u>Secondary actor</u>: Not applicable

<u>Normal flow:</u>

1.Faculty administration requests a study program where not all courses have an exam slot
2.System provides one

3.Faculty administration chooses an exam slot for every course that does not yet have one, and decides to confirm it
4.System asks for confirmation
5.Faculty administration confirms
6.The system lets the student know that his courses now have exam slots

Alternate flow:

Exception flow:

Exception flow 1: Faculty administration cancels the session
1.System brings them back to where they started

Exception flow 2: Faculty administration selected an exam slot that was full
1.System lets them know the exam slots for certain courses are not correct
2.The faculty administration fixes these exam slots and tries again

post condition: The faculty administration has completed a study program by giving all courses an exam slot

Search contract
Short description: The student can search for a given contract

Request student ID
Short description: The student can request a student ID

Finalize student ID request
Short description: The student administration approves for a student ID to be sent to the student applying for one

Add exemption
Short description: The student can add an exemption in his study program for a given course

Remove exemption
Short description: The student can remove an exemption in his study program for a given course

Consult exemptions
Short description: The student can consult the exemption in his provisional study program for a given course

Search course
Short description: The student can search for a given course

Add course to provisional study program
Short description: The student can add a selected course to a study program

Remove course from provisional study program
Short description: The student can remove a selected course from a study program

Consult courses

Short description: The student can consult the courses he has in a provisional study program

Submit exemption

Short description: the student sends all the exemption requests he has for a given individual study program

Use tolerance

Short description: The student can tolerate a course he has been examined for

Preconditions: The course has been examined

Primary actor: Student

Secondary actor: Not applicable

Normal flow:

1.The student decides to tolerate a course
2.The system gives the student information about the course and asks for confirmation
3.The student confirms

Alternate flow: Not applicable

Exception flow:
Exception flow 1: The student cancels the toleration
1.The system brings the student back to where he started

Exception flow 2: The student cannot tolerate the course for any/multiple reasons
1.The system lets the student know they cannot tolerate the course and cancels the request

post condition: The selected examined course has now been tolerated

additional information:

tolerating a course is possible when:

-score must be 8 or 9 and the course is part of a diploma contract
-where less than 10% of the non-exempted courses can be tolerated

-maximum 50% of the possible tolerance points can be spent on the first phase of a diploma contract

Adjust study program

Short description: the educational committee finishes a study program that wasn't yet completed in time

Preconditions: There is a study program that has been sent to the educational committee and has not been adjusted

Primary actor: Educational Committee

Secondary actor: Not applicable

Normal flow:

1.Educational committee decides to complete a study program
2.System sends them a study program to complete
3.The educational committee adjusts the study program and decides it is completed
4.The system asks for confirmation, while displaying all information about the study program
5.The education committee confirms
6.System sends a reply to the student to which the study program belongs that it has been adjusted

Alternate flow: Not applicable

Exception flow:

Exception flow 1: The educational committee decides to send a study program with courses that would violate the rules at the university

1.System sends a request to the committee to change the study program, along with feedback about the rules they didn't follow
2.Educational committee acknowledges the request
3.System brings the committee to the place where they can change the courses

Exception flow 2: educational committee cancels the adjusting of the study program
1.The system brings the committee back to the starting place

post condition: The educational committee has adjusted a study program that is now valid

Review exemption

Short description: The faculty administration reviews incoming exemptions

Preconditions: There is an exemption to be reviewed

Primary actor: Faculty administration

Secondary actor: Not applicable

Normal flow:

1.The faculty administration requests an exemption to review
2.The system provides one along with the arguments
3.The faculty administration reviews the exemption and approves it
4.The system asks for confirmation
5.The faculty administration confirms
6.The system sends a reply to the student informing them the exemption has been approved

Alternate flow:
Alternative flow 1: The Faculty does not approve the exemption
1.Faculty administration does not approve the exemption
2.The system asks for a motivation
3.The faculty administration provides the motivation
4.The system asks for confirmation
5.The faculty confirms

6. The system sends a reply to the student informing them the exemption has not been approved along with the motivation

Exception flow: The faculty administration decided to cancel the current review
1.system brings the faculty administration back to where they started

post condition: The faculty administration has reviewed an exemption

Submit individual study program

Short description: The student submits his individual study program, along with eventual exemptions

Preconditions: Student has selected a contract

Primary actor: Student

Secondary actor: Not applicable

Normal flow:

1.Student decides to send his courses as well as his exemptions if he has any
2.System responds with a confirmation request, as well as with a list of courses he will send
3.Student confirms his choice
4.System confirms that the student's individual study program has been sent

Alternate flow:

Exception flow:
Exception flow 1: Student decides to send a study program with courses that would violate the rules at the university:
1.System sends a request to the student to change his study program, along with feedback about the rules he didn't follow
2. student acknowledges the request
3.System brings the student to the place where he can change his courses

Exception flow 2: Student cancels the submitting of the study program
1.The system brings the student back to the starting place

Exception flow 3: Student hasn't sent his study program in three weeks:
1. System sends the study program to the educational committee, and lets the student know he was too late

post condition: Student has sent his individual study program, along with eventual exemptions

User stories

Tolerating courses

Title: As a student I can tolerate a selected course

Rational: So that the student can quickly tolerate a course, having a large amount of students means, that filling paperwork and reviewing paper work, takes a very long time, automated forms are much faster

Context: Students manually request for a specific, already examined, course a toleration, a task that only happens during summer holidays, and practically are not requested in other times of the year

Normal flow:

-The student decides to tolerate a course

-The system

Looks up the details of the course

Validates whether it can be tolerated

Prepares to change the selected course to a tolerated course

Asks the student for confirmation

-The student confirms his decision

-The system:

persists the change from examined course to tolerated course

Notifies the student the change has gone through

Brings the student back to the where the flow started

Exception Flow: If the course is not able to be tolerated

-The system

notifies the student it cannot tolerate the course, along with the reason why and gives the student possibility to:

go back to the beginning of the flow

or

cancel the session

Tests:

Is the student notified when it is successful? YES

Is all the data from the examined course copied to the tolerated course? YES

Is the student notified when it is not successful? YES

Is the student able to cancel the toleration when asked for confirmation? YES

Can the student see the details of the course to be tolerated at any point? YES

Is the student back at the beginning screen at the end of the flow? YES

Is the data persisted before confirmation? YES

-Can the system confirm the toleration in less than a minute? YES

-Can the system support multiple thousands of students tolerating courses? YES

Submitting Courses

Title: As a student I can submit courses for an individual study program request

Rational: So that the student can quickly submit his courses, having a large amount of students means that doing easy checks by humans is slow, and doing it through an automated system is fast

Context: Students can submit courses when submitting a study program along with exemptions, in this story we only focus on sending the courses. This task is something that happens in the first three weeks of each semester, thus any method should probably be able to withstand high congestion/work loads.

Normal flow:

-Student decides to send his list of courses

-System

Looks up the details of the courses

System prepares to commit the courses to the study program

Responds with the details of the study program, along with a confirmation request

-The student confirms

-The system:

Persists the study program with the courses

Notifies the student that the study program has been sent

Sends the student back to where they were before the flow started

Exception Flow: The student has courses that are not possible to be part of the study program

-The system responds:

"This combination of courses is impossible because of " <Reason(s)>

Gives possibility to:

bring the student back to a page where he can change his courses
or
cancel the session, but saving the provisional courses so the student can easily change the courses later

Tests:

-is the student notified on success? YES

-is the student notified on failure? YES

-does the student know why his provisional courses where denied? YES

-does the student get the opportunity to cancel the current session while saving the state? YES

-can the student request the specifics of a course (course ID, entry requirements)? YES

-is the student at the end of the flow back to the starting position? YES

-Is the data persisted before confirmation? YES

-Can the system tolerate multiple thousands of students submitting courses? YES

OCL

1. that fulltime students registering for a study program must take at least
all the courses of the first phase

context FulltimeStudent

inv:

//select all diploma contracts

self.takencontract->select(oclIsTypeOf(DiplomaContract)).individualstudyprogam

//select the individual study program with the lowest academic year, thus the first year of a conctract
->select(individualstudyprogam |individualstudyprogam.year= individualstudyprogam.takenContract.
individualstudyprogam ->min(year))

//select all courses in phase one
->forAll(individualstudyprogam|individualstudyprogam.contractCourse
->select(phase=1))

//these courses must contain all the courses of the first phase, that the offered contract contains
->containsAll(individualstudyprogam.takencontract
->select(oclIsTypeOf(DiplomaContract)).offeredcontract.contractCourse->select(phase=1)))

2.

that students cannot take courses for which they do not satisfy stated entry requirements

context: student

inv :

//all entry requirements of a course are either empty or

self.takencontract.individualstudyprogam.contractcourse.offeredcourse
->forAll(offeredCourse|offeredCourse.alternativerequirement->isEmpty()

or

```
//at least one combination of alternative requirements is fulfilled

offeredCourse.alternativerequirement
->exists(alternativerequirement| alternativerequirement.entryrequirement

//strong requirements require that you passed the course or exempted it
->forAll(entryreq|entryreq
->oclIsTypeOf(strong requirement)
and
self.takencontract.individualstudyprogam.takencourse
->select(oclIsTypeOf(exempted course) or oclIsKindOf(examined course)
and score>=10).contractCourse.offeredCourse
->contains(entryreq.offeredCourse)

Or

//a weak requirement requires you to have followed the course, or have it exempted it

entryreq->oclIsTypeOf(weak requirement) and
self.takencontract.individualstudyprogam.takencourse
->select(oclIsTypeOf(exempted course)or
oclIsKindOf(examinedcourse)).contractCourse.offeredCourse->contains(entryreq.offeredCourse)

or

//the parallel requirement means that you follow(ed) the course or exempted it

entryreq
->oclIsTypeOf(parallel requirement) and
self.takencontract.individualstudyprogam.contractCourse.offeredCourse
->contains(entryreq.offeredCourse)

))
```

3.that a student can not take courses of some phase if he/she has not taken all

courses of all preceding phases,

```
context TakenContract
def: MaxPhase()

post result = self
->select(oclIsTypeOf(DiplomaContract)).individualstudyprogam.contractCourse->max(phase)

context: Student

//every takencontract contains all phases of the offeredcontract, that are lower than the maximum
//phase currently taken
```

```
Self.takencontract->forAll(takencontract|takencontract
->select(oclIsTypeOf(DiplomaContract)).individualstudyprogam.contractCourse
->containsAll(takencontract.offeredcontract.contractcourse->select(phase<
takencontract.MaxPhase())))
```

4.that a student cannot subscribe for the same course in different

Contracts

context: Student

inv:

```
//Every course that hasn't been examined yet, has a unique courseID
self.takencontract.individualstudyprogam.takencourse
->select(oclIsTypeOf(Unfinished Course)).contractCourse.offeredCourse
->isUnique(courseID)
```

5. that a student is blocked for 1 year on a diploma contract if he/she has failed 4 times for

some course and his/her CSE is below 50%

context takencontract

```
//definition of CSE

def CSE():

post result = (self.individualstudyprogram.takencourse
->select(score>=10&oclIsTypeOf(notToleratedCourse)).contractCourse.offeredCourse)
-> sum(studypoints)
/ (self.individualstudyprogram.takencourse
->select(oclIsKindOf(examinatedCourse))).contractCourse.offeredCourse -> sum(studypoints)
```

context Student

inv:

```
//every diplomacontract with a CSE lower than 0.5
self.takencontract
->select(oclIsTypeOf(DiplomaContract))
-> forAll (takencontract|takencontract.CSE()<0.5 and
//and where a student has failed an exam 4 times
takencontract.individualstudyprogam.takenCourse
->forAll(takenCourse | takenCourse
->oclIsTypeOf(notToleratedCourse)and score<10) and takenCourse.examSlots->count()=4

//can not have a study program in the following year
 implies takencontract.individualstudyprogram.year
->excludes(takenCourse.individualstudyprogam->max(year)+1))
```

6. that a student can only tolerate courses for which he/she had a score of 8 or 9 and the total number of tolerated study points cannot exceed 10% of the total amount of study points for the program at stake

```
context Student


inv
//in all diploma contracts
self.takencontract->select(oclIsTypeOf(DiplomaContract))
->forAll( takencontract|takencontract.individualstudyprogam.takencourse
//every course that is tolerated has a score of 8 or 9
->select(takencourse| takencourse->oclIsTypeOf(toleratedCourse))
->forAll( toleratedCourse| toleratedCourse.score=8 or toleratedCourse.score=9) and
//and the total study points tolerated, is less than the study points of the program at stake minus the
// study points of courses that have been exempted
(takencontract.individualstudyprogam.toleratedCourse.contractCourse.offeredCourse)
->sum(studypoints)
/
(takencontract.offeredContract.studypoints-(takencontract.individualstudyprogam.takencourse
->select(oclIsTypeOf(Exempted Course)).contractCourse.offeredCourse
->sum(studypoints) )<=0.1))
```

Business Use Case Name

Apply for exemptions

variables:

```
        myStudent:Student

        myIndividualstudyprogam:IndividualStudyProgram

        myExemptedCourses: collection(ContractCourse)
```

pre:

```
        //the courses are not part of the study program and the study program belongs to the
        student in question and the courses are part of the offered contract

        myStudent.takencontract.individualstudyprogam->contains(myIndividualstudyprogam) and

        individualstudyprogam.takencontract.offeredContract.contractCourse
        ->containsAll(myExemptedCourses) and

        individualstudyprogam.contractcourse->excludesAll(myExemptedCourses)
```

post:

```
        //the individual study program now contains the exempted courses, and all the new courses,
        //are exempted courses

        individualstudyprogam.contractcourse ->containsAll(myExemptedCourses) and

        individualstudyprogam.takenCourse ->containsAll(takencourse@pre) and

        individualstudyprogam. contractcourse ->count(oclIsNew())=myExemptedCourses->size() and

        individualstudyprogam. contractcourse ->forAll(oclIsNew() implies takenCourse
        ->oclIsTypeOf(ExemptedCourse))
```

EXTRA 16+:

```
context offeredCourse

def getCoursesFromRequirements():

post

result=self.alternativerequirement->isEmpty()

        or


 result= self.alternativerequirement.entryreq.offeredCourse
```

```
context offeredcourse

inv:

        //either a course has no requirements, or the closure of the requirements of the course does
        //not contain itself

        allcourses= self->closure(getCoursesFromRequirements()) ->excludes(self)


Context DiplomaContract

def getValidAmountCourses():


//result of helper definition, getValidAmountCourses, it is the collection of all the collections of
//courses, that contain enough study points to graduate and that are part of the offeredContract

post: result->forAll(singleResult|singleResult->sum(studypoints)>=

(self.offeredContract.totalstudypoints-
->self.individualstudyprogam.takencourse
->select(takenCourse|takenCourse->oclIsKindOf(ExaminatedCourse) and takenCourse.score>=10 or
takenCourse->oclIsTypeOf(Exempted)).contractCourse.offeredCourse
 ->sum(studypoints))
//are part of the offeredcontract

and self.offeredContract.offeredCourse->containsAll(singleResult))

//none of the courses suggested are already part of the study program

and result-> forAll(singleResult|singleResult->excludesAll(self.individualstudyprogam.takencourse
->select(takenCourse|takenCourse->oclIsKindOf(ExaminatedCourse) and takenCourse.score>=10 or
takenCourse->oclIsTypeOf(Exempted)).contractCourse.offeredCourse))



context DiplomaContract

def getSmallestAmountCourses

// the collection of possible courses, that contain enough study points to graduate, while being
//minimal in size

result= self.getValidAmountCourses()->any(min(size()))
```