Student Alert System Specification

Team C: Andreas, Calin, Jeroen, Timon

Slot 4

System description

The Student Alert System (SAS) represents an initiative to improve the graduation rate amongst students by alerting mentors when the academical progress of a student stops.

Mentors and other interested university staff can register to SAS and provide custom rules which basically define the symptoms of a halt in the student progress. When a rule become active, i.e. its application returns a non-empty result, its subscribers receive an alert.

System description

The advantages of SAS are:

- easy integration with existing applications that keep track of academical progress
- easily customized by changing the rules it applies
- multitude of alert types (email, SMS, etc.)

System specification

We specify the specification by stating properties regarding the entities involved in such a system.

System entities

The main components of the system are:

- users - university staff members which want to receive alerts when certain rules become active.
- rules - expressions which provide meaningful ways to select students according to certain criteria.
- alerts - messages that are sent to every user that is subscribed to them.

System specification

System components

SAS needs to have the following components:

- A user registration component that authenticates and registers users according to this specification.
- A rule management component that provides the following features according to this specification:
  - an intuitive interface to create rules (rule editor)
  - a management tool for deleting and editing rules
  - a way to store and evaluate rules
- An alert component that must send alerts according to this specification.
- A middleware component which has to interact with the university native system and extract information according to this specification.

User specification I

- An user can be either an administrator or a normal user.
- Users can access the system only if they have been securely authenticated.
- A non-user can register to the system if he provides sufficient credentials (e.g. a proof that he is a faculty member); at the registration a user has to provide contact details.

User specification II

- Administrators can manage(create, delete, update) global rules.
- Users can
  - only manage their personal rules.
  - define personal rules.
  - choose the rules from which they want to receive alerts.
- User data needs to be stored in a persistent way.

Rule specification I

- We call rules all well formed expressions defined by the DSL grammar.
- A rule is evaluated according to the semantics of the DSL.
- Rules can be global or personal.
- Global rules are accessible to every user, but personal rules are visible only to the user that defined them.

Rule specification II

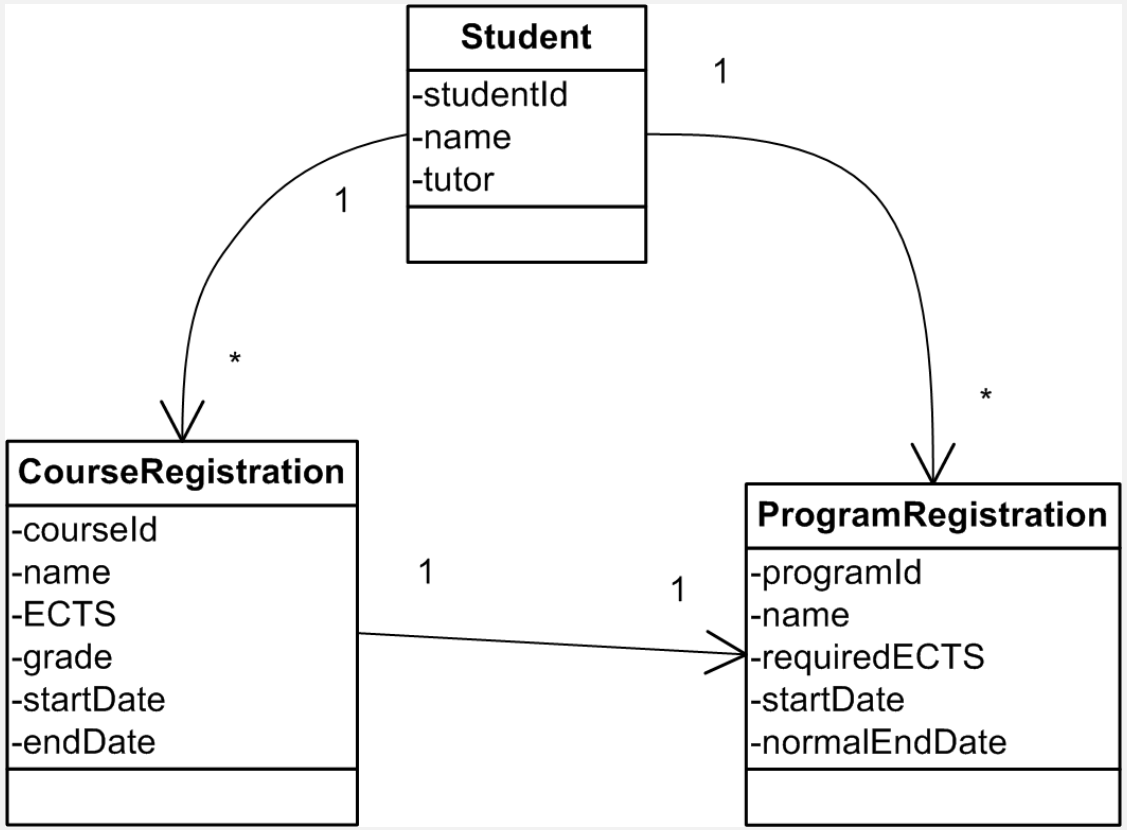
- Only rules that have been registered can be evaluated.
- The result of a rule evaluation is always a subset of the set of all students.
- A rule is active if the result of its evaluation is a non-empty set of students.
- Rules need to be stored in a persistent way.

Alert specification

- An alert is a message that needs to be delivered to users.
- An alert consists of at least an email message.
- A triggered alert is delivered to every user who is currently subscribed to it.
- An alert is triggered:
  - only when the rule is active.
  - if a rule is active and some user has subscribed to it.
  - only once in a certain period of time(specified in the system configuration).

DB diagram

The middleware should provide means for extracting the following information.



DSL specification I

$$
\begin{array}{rcl}
\text{Rule} & = & \text{Date CompOp Date} \\
 & | & \text{Num CompOp Num} \\
 & | & \text{Id EqOp String} \\
 & | & \text{NOT Rule} \\
 & | & \text{AND Rule Rule} \\
 & | & \text{OR Rule Rule} \\
\text{CompOp} & = & \text{EqOp} \\
 & | & <, >, \leq, \geq \\
\text{EqOp} & = & =, \neq \\
\text{Op} & = & +, -
\end{array}
$$

DSL specification II

```
Date  =  Date Int Int Int -- dd/mm/yyyy
      |  CurrentDate
      |  ProgramRegisteredDate Id
      |  CourseRegisteredDate Id
      |  CourseCompletedDate Id
      |  LastCourseRegisteredDate
      |  LastCourseCompletedDate
      |  Date Op NumVal -- days
      |  EmptyDate
```

DSL specification III

$$
\begin{array}{lcl}
\text{Num} & = & \text{Num Int} \\
 & | & \text{Date} - \text{Date} \\
 & | & \text{PointsSinceDate Date} \\
 & | & \text{NumVal Op NumVal} \\
\text{Id} & = & \text{Program} \\
 & | & \text{Tutor} \\
 & | & \text{StudentId}
\end{array}
$$

DSL examples

- Not registered since a certain date:

  LastCourseRegisteredDate $\leq$ (CurrentDate $-$ (Num 90))

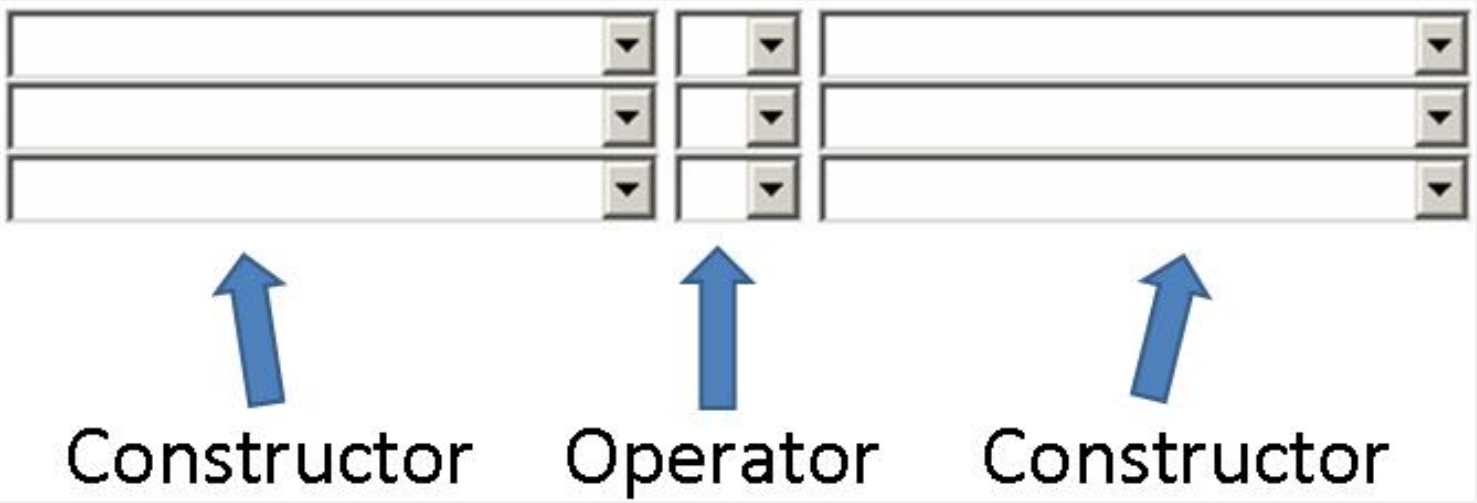
- Less than a number of points since a date:

  (PointsSinceDate (CurrentDate $-$ 180)) $\leq$ (Num 30)

- Started in a year with a program and tutor that haven't completed a certain course:

  ((ProgramRegisteredDate "Software Technology") $>$ (Date 1 9 2007))
  AND (Tutor $=$ "Meertens")
  AND (EmptyDate $=$ (CourseCompletedDate "SWS")
  AND (CurrentDate $>$ (CourseEndDate "SWS"))

Rules management: GUI



- Fields for constructor parameters appear when a constructor is selected
- Rules view: tree structure