Assignment 3, Part 1, Specification

SFWR ENG 2AA4, COMP SCI 2ME3

March 23, 2021

This Module Interface Specification (MIS) document contains modules, types and methods used to support the CEAB (Canadian Engineering Accreditation Board) graduate attributes process. An accredited program needs to measure the learning outcomes in all of the courses and show that the measures adequately satisfy the graduate attributes specified by the CEAB.

For every learning outcome in every course in a given program the number of students is counted in each of the following categories: below expectations, marginal, meets expectations and exceeds expectations. This data for each learning outcome needs to be aggregated to express the results using the CEAB's graduate attributes. We will use the same categories when summarizing the attributes, but rather than using student numbers, below, marginal, meets and exceeds will be expressed as percentages.

Each attribute is divided into a set of indicators. Each program consists of a set of courses and each course will cover a set of indicators. For each indicator in a course there will be a set of learning outcomes. The learning outcomes are course specific, but the indicators and the attributes are determined by the CEAB and the Faculty of Engineering, respectively.

There are many choices on how to aggregate the data, depending on at what point the data is normalized. These modules allow this choice to be modified at run-time so that the different possibilities can be dynamically explored. This is enabled by the use of the Norm abstract object module. At one extreme the data for a program is only normalized after summing all of the student counts for each learning outcome for each indicator for each attribute in each course. At the other extreme all of the data is normalized at each step, starting with the measures of the learning outcomes themselves. A comparison of the different schemes can be found in the sample spreadsheet available in the assignment folder.

IndicatorT Module

Module

IndicatorT

Uses

None

Syntax

Exported Constants

None

Exported Types

IndicatorT = {
math, *#Competence in mathematics*
specEngKnow, *#Competence in specialized engineering knowledge*
assumpt, *#Ability to identify reasonable assumptions*
suitableFund, *#Ability to identify suitable engineering fundamentals*
recogTheory, *#Able to recognize and discuss applicable theory knowledge base*
modelSelect, *#Selects appropriate model and methods*
estOutcomes, *#Estimates outcomes, uncertainties and determines data to collect*
desProcess, *#Recognizes and follows an engineering design process*
desPrinciples, *#Recognizes and follows engineering design principles*
openEnded, *#Proposes solutions to open-ended problems*
ideaGeneration, *#Employs appropriate techniques for generation of creative ideas*
healthSafety, *#Includes appropriate health and safety considerations*
standards, *#Determines and employs applicable standards and codes of practice*
tools, *#The ability to use modern/state of the art tools*
engInSoc, *#Demonstrates an understanding of the role of the engineer in society*
awarePEO; *#Aware of PEO and role of licensing*
}

Exported Access Programs

None

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Considerations

When implementing in Java, use enums (as shown in Tutorial 07 for ElementT).

Attributes Module

Template Module

AttributeT

Uses

IndicatorT

Syntax

Exported Constants

None

Exported Types

AttributeT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new AttributeT	String, seq of IndicatorT	AttributeT	
getName		String	
getIndicators		seq of IndicatorT	

Semantics

State Variables

name : String
s : set of IndicatorT

State Invariant

None

Assumptions

None

Access Routine Semantics

new AttributeT($attribName, indicators$):

- transition: $name, s := attribName, \cup(i : \text{IndicatorT} | i \in indicators : \{i\})$
- output: $out :=$ self
- exception: none

getName():

- output: $out := name$
- exception: none

getIndicators():

- output: $out := \langle i : \text{IndicatorT} | i \in s : i \rangle$ # *order in the sequence does not matter*
- exception: none

Measures Interface Module

Interface Module

Measures

Uses

IndicatorT, AttributeT

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
measures		seq of $\mathbb{R}$	UnsupportedOperationException
measures	ind: IndicatorT	seq of $\mathbb{R}$	UnsupportedOperationException
measures	att: AttributeT	seq of $\mathbb{R}$	UnsupportedOperationException

Considerations

The interface provides several signatures for the measures method. The modules that inherit Measures should provide all versions. The UnsupportedOperationException is included because not all signatures are valid for every subclass.

The entries in the output sequence at indices 0 to 3 correspond respectively to the number of measures: below expectations, marginal, meets expectations, exceeds expectations.

Services Module

Module

Services

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
normal	seq of $\mathbb{R}$	seq of $\mathbb{R}$	

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

normal(v):

- output: $out := \langle i : \mathbb{N} | i \in [0..|v| - 1] : v_i / \text{sum}(v) \rangle$
- exception: none

Local Functions

sum: seq of $\mathbb{R} \rightarrow \mathbb{R}$
$\text{sum}(x) \equiv +(i : \mathbb{N} | i \in [0..|x| - 1] : x_i)$

Norm Aggregation Algorithm Configuration Module (Abstract Object)

Module

Norm

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
setNorms	$\mathbb{B}$, $\mathbb{B}$, $\mathbb{B}$		
getNLOs		$\mathbb{B}$	
getNInd		$\mathbb{B}$	
getNAtt		$\mathbb{B}$	
setNLOs	$\mathbb{B}$		
setNInd	$\mathbb{B}$		
setNAtt	$\mathbb{B}$		

Semantics

State Variables

$normLOs$: $\mathbb{B}$
$normInd$: $\mathbb{B}$
$normAtt$: $\mathbb{B}$

State Invariant

None

Assumptions

The state variables will be set before they are used.

Access Routine Semantics

setNorms($nLOs$, $nInd$, $nAtt$):

- transition: $normLOs, normInd, normAtt := nLOs, nInd, nAtt$
- exception: none

getNLOs( ):

- output: $out := normLOs$
- exception: none

getNInd( ):

- output: $out := normInd$
- exception: none

getNAtt( ):

- output: $out := normAtt$
- exception: none

setNLOs($nLOs$):

- transition: $normLOs := nLOs$
- exception: none

setNInd($nInd$):

- transition: $normInd := nInd$
- exception: none

setNAtt($nAtt$):

- transition: $normAtt := nAtt$
- exception: none

Learning Outcomes Module

Template Module inherits Measures

LOsT

Uses

Measures, IndicatorT, AttributeT, Services, Norm

Syntax

Exported Constants

None

Exported Types

LOsT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new LOsT	String, $\mathbb{Z}$, $\mathbb{Z}$, $\mathbb{Z}$, $\mathbb{Z}$	LOsT	IllegalArgumentException
getName		String	
equals	LOsT	$\mathbb{B}$	

Semantics

State Variables

$name$: String
n_blw: $\mathbb{N}$
n_mrg: $\mathbb{N}$
n_mts: $\mathbb{N}$
n_exc: $\mathbb{N}$

State Invariant

None

Assumptions

None

Access Routine Semantics

new LOsT($topic, nblw, nmrg, nmts, nexc$):

- transition: $name, n\_blw, n\_mrg, n\_mts, n\_exc := topic, nblw, nmrg, nmts, nexc$
- output: $out := self$
- exception: exc := $((nblw < 0) \vee (nmrg < 0) \vee (nmts < 0) \vee (nexc < 0) \Rightarrow$ IllegalArgumentException$|(nblw = 0) \wedge (nmrg = 0) \wedge (nmts = 0) \wedge (nexc = 0) \Rightarrow$ IllegalArgumentException)

getName():

- output: $out := name$
- exception: none

equals(o):

- output: $out := name = o$.getName()
- exception: none

measures():

- output: *# output should also be converted from natural numbers to reals*

  $out := (\neg$Norm.getNLOs() $\Rightarrow [n_blw, n_mrg, n_mts, n_exc]|$True $\Rightarrow$ normal($[n\_blw, n\_mrg, n\_mts, n\_exc]$))
- exception: none

measures(*ind*: IndicatorT):

- output: none
- exception: exc := UnsupportedOperationException

measures(*att*: AttributeT):

- output: none
- exception: exc := UnsupportedOperationException

Courses Module

Template Module inherits Measures

CourseT

Uses

Measures, IndicatorT, AttributeT, Services, Norm

Syntax

Exported Constants

None

Exported Types

CourseT = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new CourseT	String, seq of IndicatorT	CourseT	
getName		String	
getIndicators		seq of IndicatorT	
getLOs	IndicatorT	seq of LOsT	
addLO	IndicatorT, LOsT		
delLO	IndicatorT, LOsT		
member	IndicatorT, seq of LOsT	$\mathbb{B}$	

Semantics

State Variables

name : String
m: set of MapInd2LOsT # *mapping between indicators and set of learning outcomes*

State Invariant

None

Assumptions

The programmer will not use CourseT until the indicators have been populate with learning outcomes. The constructor starts with the set of learning outcomes being empty; therefore, addLO needs to be called at least once for each indicator for the course. (# *Initially a measureDone flag was part of the state variables for the module to track this, but in the interest of keeping things simpler, it was removed.*) If learning outcomes are removed, it is assumed that not all of them will be removed before the Course is used in an aggregation calculation.

Access Routine Semantics

new CourseT(*courseName*, *indicators*):

- transition: $name, m := courseName, \{i : \text{IndicatorT} | i \in indicators : \langle i, \{\} \rangle\}$
- output: $out :=$ self
- exception: none

getName():

- output: $out := name$
- exception: none

getIndicators():

- output: $out := \langle s : \text{MapInd2LOsT} \mid s \in m : s.ind \rangle$
- exception: none

getLOs(*indicator*):

- output: $out := (\langle indicator, LOs \rangle \notin m \Rightarrow [] | \text{True} \Rightarrow \text{set\_to\_seq}(LOs) \text{ where } \langle indicator, LOs \rangle \in m)$
- exception: none

addLO(*indicator*, *outcome*):

- transition:

  $m := \{s : \text{MapInd2LOsT} | s \in m : (s.ind = indicator \Rightarrow \langle s.ind, s.LOs \cup \{outcome\} \rangle | \text{True} \Rightarrow s)\}$

- exception: none

delLO(*indicator*, *outcome*):

- transition:

  $m := \{s : \text{MapInd2LOsT} | s \in m : (s.ind = indicator \Rightarrow \langle s.ind, s.LOs - \{outcome\}\rangle | \text{True} \Rightarrow s)\}$

- exception: none

member(*indicator*, *outcomes*):

- output:

  $out := \exists(s : \text{MapInd2LOsT} | s \in m : s.ind = indicator \wedge \forall(x : \text{LOsT} | : x \in s.LOs \leftrightarrow x \in outcomes))$

- exception: none

measures():

- output: none
- exception: exc := UnsupportedOperationException

measures(*ind*: IndicatorT):

- output: $out := (self.\text{getLOs}(ind) = [] \Rightarrow [0,0,0,0] | \text{True} \Rightarrow (\text{Norm.getNInd}() \Rightarrow \text{normal}(measureInd) | \text{True} \Rightarrow measureInd))$
  where $measureInd \equiv \text{sumMeas}(o : \text{LOsT} | o \in self.\text{getLOs}(ind) : o.\text{measures}())$
  # sumMeas is a binary operator, so it can be used with our normal notation.
- exception: none

measures(*att*: AttributeT):

- output: $out := (att.\text{getIndicators}() = [] \Rightarrow [0,0,0,0] | \text{True} \Rightarrow (\text{Norm.getNAtt}() \Rightarrow \text{normal}(measureInd) | \text{True} \Rightarrow measureInd))$
  where $measureInd \equiv \text{sumMeas}(i : \text{IndicatorT} | i \in att.\text{getIndicators}() : self.\text{measures}(i))$
- exception: none

Local Functions

set_to_seq : set of LOsT $\rightarrow$ seq of LOsT
set_to_seq(s) $\equiv \langle x : \text{LOsT} | x \in s : x \rangle$ # *Return a seq of all of the elems in the set s*

sumMeas : seq [4] of $\mathbb{R} \times$ seq [4] of $\mathbb{R} \rightarrow$ seq [4] of $\mathbb{R}$
sumMeas$(a, b) \equiv [a_0 + b_0, a_1 + b_1, a_2 + b_2, a_3 + b_3]$

Local Types

MapInd2LOsT = tuple of (*ind*: IndicatorT, *LOs*: set of LOsT)

ProgramT Module

ProgramT

Template Module inherits Measure, HashSet(CourseT)

Uses

Measures, IndicatorT, AttributeT, Services, Norm, CourseT, HashSet

Syntax

Exported Constants

None

Exported Types

ProgramT = ?

Exported Access Programs

Routine name	In	Out	Exceptions

Semantics

State Variables

s: set of CourseT *# inherited from HashSet(CourseT)*

State Invariant

Assumptions

The set of courses will not be empty and at least one course will have non-zero measures for each attribute.

Access Routine Semantics

measures():

- output: none

- exception: exc := UnsupportedOperationException

measures(*ind*: IndicatorT):

- output: none
- exception: exc := UnsupportedOperationException

measures(*att*: AttributeT):

- output: *out* := normal(sumMeas(c : CourseT|$c \in s$: c.measures(*att*)))
- exception: none

HashSet Module

Generic Template Module inherits Set(E)

HashSet(E)

Considerations

Implemented as part of Java, as described in the Oracle Documentation