**Implementation Design**

**Objective**

To compute the new ability α of the target person after each level in the course

**Needs**

The marks of the person = marks [nques] [nlevels]

The relevance of questions to the user = relevance [nques] [nlevels]

The current ability value for user = α [nlevels + 1]

Where nques = no. of questions

**Theory**

New ability is calculated by the formula:

*α[i+1] = α[i] × k + αcomp × (1-k) ------------------- Usage In Static Algorithm*

Where *αcomp* = computed ability from marks obtained in test.

*k* =constant for adaptability.

*i є nlevels*

|  |
| --- |
| **ComputeAbility(**  **Input:** marks[nques], relevance[nques], α, nques  **Output:** αcomp  **)**  begin  αcomp = 0;  for i in nques begin  αcomp = αcomp + Ques\_LO[i] \* marks[i];  end for  return αcomp  end function |

**Objective**

To compute the tolerance class for a student after each level in the course

**Given**

1. The fitness function of the paths after level **i** for all the previous students who had already completed this course. It is given by ***pathValuei***.
2. The ability *α* of past students after level **i** given by ***αi***.
3. Time remaining after completion of a particular topic **i** for previous students given by ***timei***.
4. The learning aim of the user which is taken as an input for the user. It is constant throughout the course, and it is given by ***aim***.

**Theory**

The tolerance sets for each of the above stated entities is being computed using the **T ()** function and then the following equation is used to obtain the set which contains the set of students which are similar to the target user after level **i**.

{Ttotal} = (({T*pathValue*}  {T*time*})  {T*α*})  {T*aim*}

Where, Ttotal = Total tolerance

TpathValue = Tolerance based on fitness function of path

T*α* = Tolerance based on current Learning ability

Ttime = Tolerance based on time spent on the course

Taim = Tolerance based on aim of a person

|  |
| --- |
| **ComputeTolerance (**  **Inputs:**αcurrent, pathValuecurrent, timecurrent, aimcurrent  **Outputs:** {Ttotal} (Set of students similar to current student)  **)**  begin  Total\_tolerance = {}  aim\_tolerance = computePrioritySet(nstudents, aimcurrent)  for i in aim\_tolerance:  if T(αi,αcurrent) or  T(pathValuei, pathValuecurrent) and  T(timei, timecurrent) then  Total\_tolerance = Total\_tolerance  i  end if  end for  return Total\_tolerance  end function |
| **T (**  **Inputs:** x1 (value to test), x2 (base value), *kx* (limit of variation)  **Outputs:** value\_of\_tolerance  **)**  begin  return ***max*** (( ***max*** (0, ***min*** (x1, x2)) + *k*x – ***max*** (x1, x2) ) / *k*x, 0)  end function |
| **ComputePrioritySet (**  **Inputs:** nstudents (number of students), aimcurrent (aim of current user)  **Outputs:** {Taim} (Set of Students with same aim as current student)  **)**  begin  return {x | x є nstudents, aimx = aimcurrent}  end function |

**Objective**

To provide a recommendation to the user about the next perspective he should take

**Given**

1. The paths taken by all the previous users stored in the following template:-

**nodepath = (idlevel, idperspective[idlevel]), maskLO[idlevel][idperspective]**

where,

idlevel = index of the topic taken

idperspective = index of the perspective in idlevel

maskLO = bit string showing LOs taken by the user on (idlevel, idperspective)

1. The set of students which are similar to the current user after completing level **i** in the course.
2. The final Scores of the students which shows their complete performance on completing the course.

**Approach**

The solution that has been devised is based on the fact that the user has a number of perspectives to take on the next level, out of which one has been suggested to him based on the static prediction. However, the other perspectives might help him perform better in the course and improve his scores in upcoming levels. The success rate of various perspectives is being computed and the perspective that has maximum success rate is being recommended by the function to the user.

|  |
| --- |
| **RecommendPerspective (**  **Inputs:** idlevel, αcurrent  **Outputs:** idperspective[idlevel+1]  **)**  begin  passed\_studs = setGoodStudents()  for each in passed\_studs  begin  if R(LAeach, LAavg)  begin  LAcompmax += LAeach  \* exp (αavg .timeavg);  LAcount++;  end if  end for  LAcompmax = LAcompmax / LAcount;  LAmax = k .e-βt (LAcompmax – LAexpertmax) + LAexpertmax.  return LAmax |

|  |
| --- |
| **Algorithm for the function R**  **function** R(xi, xavg)  begin  if (T(xi, xavg, κ) )  {0, 1}  return **true**;  else  return **false**;  end if  end  where, T is same function as the one used for computing tolerance  κ is value of threshold, computed based on the improvement in grades. |

|  |
| --- |
| **function** setGoodStudents ()  return set of students having a passing grade in course. |