University Grades System Application using Dynamic Data Structure

Nael Hirzallah[1], Dead Al-Halabi[2] and Baydaa Al-Hamadani[3]

[1] Computer Engineering, Fahad Bin Sultan University
Tabuk, Saudi Arabia

[2] What is Next
Amman, Jordan

[3] University of Huddersfield
Oldham, UK

Abstract

Most e-Learning platforms implemented in educational institutes provides a tool for instructors to enter the grades and for students to view them. This tool with the appropriate workflow is considered one of the most sensitive and important applications in any University Information Management System. Consequently, implementing this tool should consider the fact that it must be flexible and adaptable from time to time. This paper focuses on evaluating few different approaches to implement such a tool that belong to a so-called Static Approach. It also discusses the limitations of this approach. The paper then introduces a different, yet dynamic approach to implementing a Grades System Tool. An analytical study of the efficiency of the suggested system is also presented.

Keywords: *Dynamic Grades Tool, Static Grades Tool, Data Structure, Grades System***.**

1. Introduction

Many educational institutes are moving towards implementing a full e-Learning platform that offers state-of-the-art tools for students, academic and administrative staff, as well as the university community at large. Among the tools that are considered important is the virtual driving force for students, which is the one used to manage the Grades. This tool is used to record the worthiness of students' efforts during a semester. Therefore, there is a demand to increase the trust in a university Grades Tool (GT) by most parties involved in the higher education process, such as, Teachers, Students, Managers, and Administrators, [1, 2]. A GT has to be able to adopt new development strategies and accompany the modernization in its background and planning, in order to achieve its objectives, [2, 3]. It has to be built using a strong, efficient, and customized system to enable efficiency in time and efforts to all high education partners. Furthermore, saving time and money by an institution is one of its top priority requirements. Besides, recognizing new techniques and continuous development in the university sub systems indicates the growth of the institution reputation in local and global societies. This is considered by executives as an important marketing feature, [4, 5].

In this paper, two GT running in two different universities will be discussed. These tools are labeled as static due to their inability to adopt new Grading System policies to a certain extent. The paper then introduces a dynamic grades tool approach that depends on linked list structures in its implementation. The advantages of such approach over static GT will be discussed, as well.

The paper is organized as follows: section two summaries the Grading System policies used in Jordanian universities. Section 3 presents a study of two existing GT's that are using the static data structure approach. In section 4, the paper introduces the Dynamic Grades Tool Approach (DGTA) followed by a discussion on the requirements, preparation, implementation, and performance of a DGTA. Finally, in section 6, the paper draws its conclusion.

2. Grading systems

2.1 University's Grading Policy

All Jordanian universities use either percentage or point-for-weight grading systems (i.e. letter grade system) [2, 5]. Table 1, [3], shows the basic transfer scheme from a

percentage grading system to a point grading system that exists in Jordan.

Table 1: Basic grade scheme used in Jordan.

Scale	U.S. Grade Equiv.
80-100	A
70-79	B
50-69	C
0-49	F

The Jordanian Grading Systems (GS) policy states the following issues, which should be considered by any GT [4]:

1. GS applies a numerical grade system in addition to the letter grade.
2. Every instructor is responsible for the following: entering student grades, evaluating students work, judging the course progress for her / his courses, and changing or modifying the final reported grades.
3. Every instructor is responsible for evaluating student's written documents or oral discussions.
4. Instructor (s) can make the grade more specialized, pursuant to his/her rights and authority given by the educational institute s/he works in.
5. All the oral evaluations have to be graded.
6. All sub grades have to be entered into a GT.
7. Course instructor (s) has the responsibility to enter valid data. Data have to be in the range of [minimum, maximum]. Also, s/he must make sure that the data are verified, accurate, and consistent.
8. Grades have to be accepted and verified at the department level and then at the faculty level.

2.2 The Value-Driven Meaning of Grades

GS is changeable and variant from time to time according to a given course specific policy. The transfer between a numeric GS to a letter GS or vice versa is possible and often needed.

There are two major paths for evaluating student's work in terms of percentages: either summative or formative. Both metrics should give indication of the imagination, creativity, and skills necessary for the rapidly changing requirements of modern social life.

Thus, any assessment criteria should guarantee the following:

1. Fairness
2. **Validity**
3. **Reliability**

The summative assessment is based on the overall summation of sub-activities that had occurred during a semester for a particular course. It involves written paper, such as 1st, 2nd, and Mid exams, assignments, essays, tutorials, quizzes, self reading materials, and class projects. On the other hand, the formative assessment is a self-reflective process for a student. It is based on class discussions, questions, and seminars. The Final grade can be a mix between summative and formative; its assessment shall be at the end of the semester [8].

3. Existing Grads Tools

The observations discussed in this section are based on the experience gained by working on various systems in different institutions as a user with the role of an instructor. Static GT (SGT) is a client/ server application. Client sends and receives the required class information that belongs to an instructor. The user screen will be filled with the required information by opening a channel with the server.

There are three types of universities in Jordan: public, private, and distance higher education, [5]. The discussion will focus on two of these universities, labeled in this paper as “A” and “B”. University “A” is a public university and “B” is a private one. University “A” uses letter grades, while “B” uses percentage grades. University “A” has two policies for obtaining the final grade. The first one states that the final grade is divided into: 1st Exam, 2nd Exam, course-work, and a Final exam. While in the second policy, the final grade is divided into: a Mid exam, course-work, and a Final exam. On the other hand, University “B” has only one policy to obtain the final grade: 1st exam, 2nd exam, course-work, and a Final exam.

The detailed weight distribution for each sub grade was left flexible and usually set by either the department or the instructor. Different course weight division may exist. One example is (15, 25, 10 and 50), while another one has (20, 20, 10 and 50).

In University “A”, which uses letter grades system, the course weight distribution may be changed from one semester to another.

Figure 1 shows a snapshot of a weight distribute of one course offered by University “A” in one semester. The screen is divided into two blocks; the above one acts as a list of templates. One may select a course template then fill the sub weights values in the second block below it.

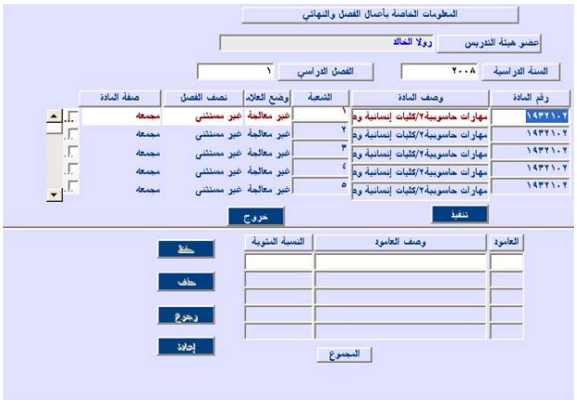
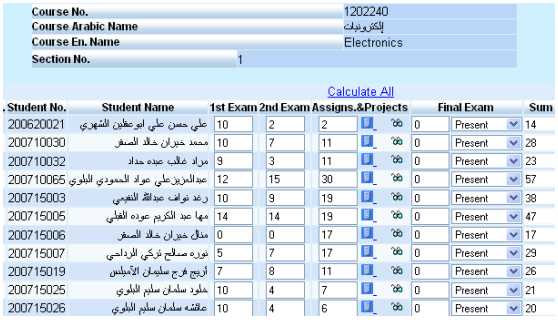


Figure 1: distribute grades weight in “A” under SGT.

Figure 2 shows another snapshot of an SGT screen from university “A” that allows instructors to enter grades for a specific course section. The screen illustrates the relationship between the course section and the student list that belongs to that section. The screen includes the following information: the teacher name, academic year, course number, section number, lecture room, semester number, and lecture time. It also includes student number, student name, 1st, 2nd, and 3rd grades, course work, and the Final exam. The last two columns are for the total grade, one in percent and the other in letters.

Figure 2: university “A” SGT entering grade screen.

University “B” uses percentages for evaluating the work of students in a semester. Sub grade weights are to be figured out from the data rather than from the system. There was no checking for exceeding the sub grade limit weight, if any, except at the end, when computing for the total grade out of 100. For example, if all entered grades of one exam are between [0, 20] then one may conclude that the maximum grade for that exam is 20. Figure 3 shows a snapshot of a screen of the SGT that allows instructors to enter grades for a course section in a semester.



Course No.	1202240
Course Arabic Name	إلكترونيات
Course En. Name	Electronics
Section No.	1

Calculate All

Student No.	Student Name	1st Exam	2nd Exam	Assigns.&Projects		Final Exam		Sum
200620021	علي حسن علي ابوعطيون الشهري	10	2	2		0	Present	14
200710030	محمد خيران خالد المسفر	10	7	11		0	Present	28
200710032	مراد غلاب عبده حداد	9	3	11		0	Present	23
200710065	عبدالعزيزعلي عواد الحمودي البلوي	12	15	30		0	Present	57
200715003	رعد نواف عبدالله الفهمي	10	9	19		0	Present	38
200715005	مها عبد الكريم عوده الغلبي	14	14	19		0	Present	47
200715006	مذلل خيران خالد المسفر	0	0	17		0	Present	17
200715007	نوره صالح تركي الردادحي	5	7	17		0	Present	29
200715019	أريج فرج سليمان الأتيلس	7	8	11		0	Present	26
200715025	حلود سلمان سليم البلوي	10	4	7		0	Present	21
200715026	عاقشه سلمان سليم البلوي	10	4	6		0	Present	20

Figure 3: snapshot of screen of SGT of university “B”

The screen in Figure 3 is like that of Figure 2. It includes the following information: semester number, course number, section number, and the credit hours for the given course. It also contains: students’ numbers, students’ names, first and second grades, course work, final exam and total grade. The second last column describes the students situation in the course in terms of Withdraw, Absent from Final Exam, or Denied.

Generally speaking, static data structure implementation is easy to deal with and fast to implement. Its data access is a straight forward process; only a direct location is needed to obtain the data, such as the index value. There is no time wasted and an indexing schema can be used to organize its access time.

The main disadvantage is the waste of unused memory. Take for example the following scenario: if a course has its evaluation metric (Exam1, Exam2, Course-Work, and a Final exam) for 80 students, this means that there is a need for four columns multiplied by 80 records, which equals to 320 memory fields. Assume another course that is distributed as: mid-exam, Course-work, and a Final exam, for 80 students. This would need 3 columns multiplied by 80 records which equals to 240 fields. Therefore, if the system is set to have statically 320 fields, this would results in 80 wasted fields. In other words, the system creates k-columns even if the number of needed ones is less than k, in order to accommodate the worst case scenario. This is of course for one course. Now, assume that you have N-courses, then there will be 80 X N wasted fields.

Moreover, a waste in memory would result in delays in the access times when retrieving data under heavy load conditions. This would affect application ranking as it considers strongly page-load speeds.

4. Suggested proposal for DGT

4.1 Dynamic features

The theory behind dynamic allocation is based on the following statement: only what is necessary to build will be built. Thus, there is no need for unusable storage to be created, and no memory to waste.

Access and retrieving times are the most important features to be considered. The difficulty in implementing a system based on dynamic approach comes from analyzing and considering the risks that may occur due to scenarios that are rare to occur.

Dynamic approach takes in its consideration time and storage factors. Storage retrieving mechanism should exist, in addition to focusing on time scheduling.

4.2 DGT Procedures

The main two features concerned in dynamic approach that depend on each other are space and time. Figure 4 illustrate this relationship. Assume the x-axis represents time and the y-axis represents memory size allocated. The figure shows ascending relationship between them. That is, by time the memory space needed increases. Yet there is no fixed rhythm for dynamic approach as it is in the case of the static approach. Note that the memory size needed by the end of a semester for the same class in both approaches are the same. However, in the static approach the memory gets allocated at early times, while in the dynamic approach, it gets allocated by time. This will have a good effect on the complexity and access time.

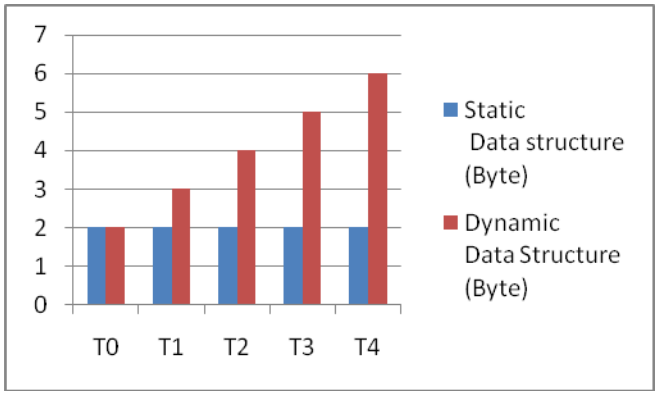


Figure 4: data size for static and dynamic structures

The main steps to populate a DGT with data are as follows:

1. Preparation: Identify the essential data and build the corresponding data structure.
2. Grading: For sub-grade 1, build the dynamic data structure associated with it and fill it with the proper data.
3. Repeat step two for subsequent sub-grades, say second exam, first quiz, first assignment, and so on, till the Final exam.

For step one, prior to a semester, students register in a section for a course. The course coordinator usually sets the weight of each sub-grade. For example the first exam gets a weight of W1, the second exam gets W2, and so on up to Wk , where k is the number of sub-grades. One scenario could be as follows: (W1=10, W2= 20, W3=5, W4=15 and W5= 40) where k =5.

Step two can be accomplished automatically at its previously assigned time, as stated in the course syllabus, or manually by the course coordinator.

Figure 5 shows how the data structure of such a system would look like. It has an array of pointers that has the size of N, where N represents the number of students registered in the class. Each pointer links the array with a structure that contains the student's number and a pointer to a link list for the student's grades. In what follows, the C++ notations will be used.

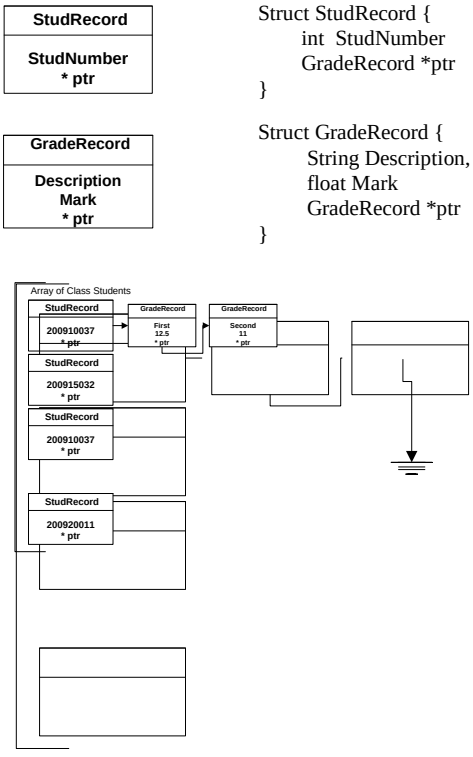


Figure 5: DGT data structure

Step two procedure is illustrated in Figure 6. The procedure gets the number of sub-grade to be entered and assigns it to i. For all the students in the class (1 to N) the sub-grade value is entered separately. Each value is checked against its maximum value, which is Wi. If it is validated, a new node of type GradeRecord will be created and initialized.

4.3 Performance Analysis

The rapid changes in the GS policy have to be reflected in the tool or application. A classical evaluation of the student work, that is, using three written exams namely: first, second and final, would be probably better implemented if using a GT that uses static data structures, SGT. This suitability comes from the ascending relationship between the mostly fixed measurements requirements and static data structures, [5]. The problems arise when there are many student works evaluation schemes rather than just one or two. For example, TMA (Tutor-Marked Assignment) is a vital example in student work evaluation environment that usually varies in number and weight from one section to another, from one course to another, and from one semester to another [7].

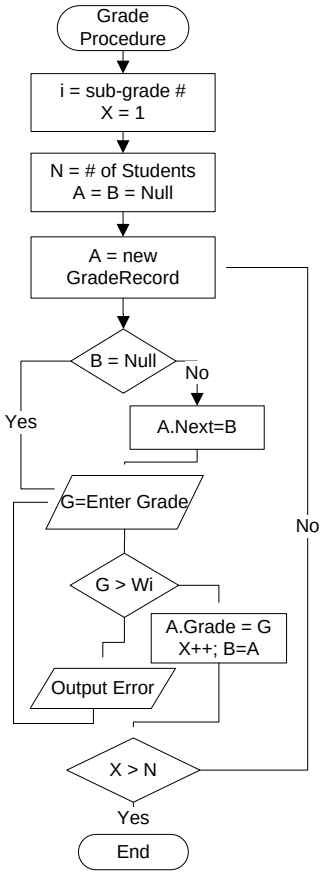


Figure 6: Flow chart of Grading procedure

Therefore, the GT is a changeable tool that has to provide a capability to understand the new non-functional and functional requirements in order to be able to support efficiency, reliability, portability, usability, performance and space, in addition to validation, accuracy, and consistency of data (grades - functional requirements that depend on the system domain). For that DGT is a more suitable solution under the requirements changing condition.

In this section we present an analytical module to study the worthiness of implementing the suggested solution methodology of adopting dynamic data structure in GT in terms of complexity (big-O and memory size).

Big-O analysis depends on the run time of the application. DGT is a client-server tool or application, and it is assumed that the computation for a DGT is done on the server side. It is also assumed that the network infrastructure is well built to eliminate communication negative factors, as well as has negligible page-load timings.

For the analysis, the following specific assumptions were considered:

- A course has four weights, k=4, that is W1, W2, W3 and W4.
- All algorithms had been run and the computation of big-O is based on the fact that the procedures have reached the final exam. (i.e. complete course evaluation).

The O (F(N)) = Big-O for preparation algorithm added to it the Big-O for the mid-term (W1) grade, added to it the Big-O for course work (W2) grade, and so on. The preparation procedure will pass on every cell in the array and fill it with the student number and a null pointer for the grade list. This process will take O(N). While the Grading procedure will pass on every student in the list and add the corresponding grade as required, (such as adding item in a linked list). This process will take O(i) where i is a constant to indicate the number of sub-grades for one student to evaluate. For N students, this will yield O(N). Thus, O(F(N)) will result in the following:

$$O(F(N)) = O(N)$$

A brief comparison was made between SGT (existing system) and DGT (suggested solution) based on the processing time and memory size. In SGT with respect to memory, it remains the same throughout the application life time. Thus, the memory size is of the order of Big-O (c X N), where c is a constant that represents the number of fields to be entered (5 in our earlier assumption).

While in the suggested solution, the amount of memory used depends on time. As an example, the number of fields to be created for the first exam at time t1 will be N. Later, another N fields will be created for the second exam at time t2.

Memory complexity in DGT is observed to be reduced. Also there is an obvious reduction in the processing time as well, due to less memory being used.

5. Conclusion

The paper has presented the importance of the right implementation to a Grades Tool. Couple of real implementation examples was discussed that belong to traditional static programming habit (array of records). Such approach was labeled Static Grades Tool (SGT), and its limitations were presented. The paper then presented the use of dynamic data structures (array of link-lists) in such applications, labeling them as Dynamic Grades Tool, (DGT). A reduction in storage and processing times were the driving factors. Moreover, an analysis on the run time using big-O method gave good indicators on the superiority of DGT over SGT.

References

[1] http://www.ju.edu.jo/units/registration/Home.aspx, Last visit 20-4-2010

[2] http://www.uop.edu.jo/admission/Default.aspx?lang=en&location=admission, last visit 20-4-2010

[3] http://www.wes.org/gradeconversionguide/

[4] http://www.uop.edu.jo/Admission/Grading.aspx?lang=en&location=FS, last visit 20-4-2010

[5] Software & Systems Requirements Engineering: In Practice, Brian Berenbac, daniel j. paulish, juergen kazmeier, arnold rudorfer, Mnc raw hell, 2009.

[6] Data Structures and Algorithms in Java, Michael T. Goodrich, Roberto Tamassia, John Wiley & Sons, 4TH edition, 2006

[7] http://www.open.ac.uk/assessment/pages/tma-submission-methods.php, last visited 30-5-2010

[8] http://www.nmsa.org/Publications/WebExclusive/Assessment/tabid/1120/Default.aspx, last visit 30-5-2010

[9] Distributed systems: Principles and Paradigms, Andrew S. Tanenbaum and Maarten van steen, prentice Hall, 2nd edition, 2002

[10] Principles of Distributed database systems, M. Tamer Ozsu and Patrick Valduriez, 2nd edition, 1999