"Requirement Declarative Markup Language" for Timetabling: A Case Study Based Development

Arunasiri K. Liyanage[1], L.S.K. Udugama [2]
[1]Department of Information Technology, Faculty of Information Technology, University of Moratuwa, Sri Lanka
[2]Department of Electrical and Computer Engineering, Faculty of Engineering Technology,
The Open University of Sri Lanka
arunasiri@uom.lk[1] , udugama@ou.ac.lk[2]

Abstract – **Preparation of timetables is one of the challenges in universities. Universities are attempting to automate the timetabling process. However neither the manual nor the automated system may give a 100% accurate timetable. Therefore it is important to verify whether the requirements are satisfactorily fulfilled after generating the timetable. Authors propose a scripting language, Requirement Declarative Markup Language (RDML), for this purpose. Using RDML all user requirements can be defined and verified. Finally authors describe how to use the RDML for verifying a timetable.**

I INTRODUCTION

Scheduling a timetable is one of the major tasks faced by universities. This is not an exception for the Open University of Sri Lanka (OUSL) though it offers courses in distance mode. At the beginning of each academic year the faculty of Engineering Technology of OUSL prepares a suitable timetable for the activities of the whole academic year. For this task a lot of inputs have to be considered. Besides the information for scheduling the timetable, human requirements such as that of lecturers, other staff and students have to be considered during the preparation of the timetable. Accordingly it has to consider a large set of constraints.

Though the faculty produces a timetable without clashes it is not possible to confirm whether the requirements of staff have been fulfilled. It is useful and necessary to inform the person how his/her requirements are satisfied. The author in [1] has also pointed out that quality of the solution (timetable) can be defined as keeping all the stakeholders happy. Therefore it will be appropriate to inform the academics and others when their requirements cannot be satisfied. Subsequently they can make alternative arrangements so that a better timetable could be prepared. On the other hand people also want to check for themselves whether their requirements being fulfilled or not. When considering all these factors this will become a time consuming process in each year.

It is interesting to see how other universities in the world handle this task. During our literature survey related to the problem we have found that most of the universities are attempting to automate the timetabling process [2]. And we hardly found anything to verify the fulfillment of the human requirements after the preparation of the timetable.

We see there is a great necessity to verify whether the staff requirements are made to satisfy them when preparing a timetable for universities. This is because neither the manual nor the automated system may give a 100% accurate timetable. According to [3] automated systems stably provide better solutions for their timetables against manual systems but there is enough room for further development. There are various techniques such as operational research techniques, genetic algorithms, Tabu search, constraint satisfaction, etc. used for solving this problem [4]. Some methods use termination conditions [5], which may lead to generate timetables that are not fully optimized. So it seems there could be situations where some user requirements are opted out in timetables generated in either way, automatically or manually. Therefore it is a necessity to have a mechanism to analyze the timetables for user satisfaction. Though it has been a requirement for OUSL it could be useful for others as well.

In this paper we propose a scripting language, Requirement Declarative Markup Language (RDML), to define requirements of the users for timetabling and verify them. At the beginning we present the background of the problem and then how we analyzed the problem. The model we derived can be found thereafter. We also show how our scripting language can be used. This approach is used at the faculty of Engineering Technology of OUSL and the description of the application developed for the faculty is also presented in the paper.

II BACKGROUND TO THE STUDY

Timetabling process of the said faculty is a much complicated process when compared with other conventional universities. This is owing to the fact that the structure of the programmes offered by the faculty itself and the flexibility of the curricula. On top of that it has to keep in mind the Open and Distance mode of teaching. And also our students are employed and they work anywhere in the country. Traveling to the university has to be minimized

1-4244-1152-1/07/$25.00 ©2007 IEEE.

and convenient. Further it should take care of when scheduling activities at different centers, as they are located at different cities in Sri Lanka. For example traveling to another center for a different activity on the same day is practically impossible.

The faculty offers different programmes. Students have to offer a number of courses in order to gain necessary credits to complete a programme. According to their specialized field they may have to select a set of compulsory courses defined by the faculty. However they need to do additional courses to fulfill the credit requirement for the programme. Here they are free to select any course offered by any department of the faculty.

A course consists of a lot of activities. Some of these activities are Assignments, Continuous Assessment Tests, Labs, Field Visits, Factory Visits, Design Projects, Presentations, Day Schools and Final Examinations. These activities can be different from course to course. Depending on the activities of a course they can be held in different centers located through out the country. Each course has at least one course coordinator and one academic coordinator.

The most complicated issue arises when considering the requirements of the academic staff and the students. These requirements can occur due to their academic or personal needs of the staff. Nevertheless students' requisites have to be also taken into account when preparing the timetable. Limited resources and how they can be utilized optimally is another brainteaser for the person who is preparing the timetable. When interfaculty activities are considered the timetabling process becomes more complex.

Some of the requirements are given in the Fig.1. The constraints that are common to all courses listed as *General Constraints for Courses* and that are specific to academics and to the students as *Specific requirements of academics and students*.

General Constraints for Courses

- Selected activities of the same course cannot be held on the same day and at the same time. (Minimum days between two activities may be defined)
- Dates of the activities must be in sequence with given constraints. (e.g. order of the assignment should be Assignment #1, #2, #3, …; Final examination comes after all activities etc.)
- Activities of a group of course(s) should not be held on a predefined date. (e.g. activities of all EC courses should not be held on the first working Monday of each month)
- Day School #1 of all courses must be held within 2 weeks of the beginning of the academic year.
- Day Schools should not coincide with the labs, CA Tests, Viva, Presentations etc.
- All activities of the courses (except in the exclusion list) must be over before this date.
- Due date of the Assignment #1 must be one week before the Lab Day #1.
- Day schools should not clash with the day schools of the courses, Course X, Y, Z.

Specific requirements of academics and students

- Some Course coordinators need their Day Schools to be held in different days. But some coordinators wish to have their Day Schools of selected courses on the same day with a specified time gap.
- Course coordinator needs a specific location and time for some activities.
- Most of the students wish to have their activities on non-working days.
- Students want a considerable gap between specified courses in their Examinations.
- Academics want to complete exams in a minimum period.
- Some coordinators wish to complete all activities of specified courses before a given day.
- Some coordinators need to keep a specified period of an academic year without any activities.

Fig. 1. Course Constraints and Human Requirements

According to our experience the final timetable may have considerable overlaps in some activities and some of them can be considered as acceptable. However there may be situations where overlapping should be avoided. Practically it is not possible to overcome all the issues at the timetabling process though it is automated or manual. We have to spend a lot of time to check these overlaps, conflicts of activities and fulfillment of requirements of the academics and the students up to an acceptable level. So we need some sort of concept/technique to solve this problem.

III ANALYZING THE PROBLEM

We looked for available concepts/techniques and methods for solving this problem. We were not able to find any approach for a similar problem. All the published research have been conducted concentrating to automate the timetabling process. In the [6] authors tried to combine automated timetables with user interaction. Gradually the automation has been improved according to its literature, but still it has not given a 100% accurate timetable. Nevertheless, most of the time, the improvement is compared with the manual process [4]. It will not assure whether the timetable is good enough to use or not. On the other hand there are no ways to analyze the timetable whether it derives the expected result or otherwise.

We searched for other existing products. Those software applications are developed to support limited constraints to control the data entry. But we need to change requirements from time to time. In such a case coding of the software or SQL scripting will be impossible to handle the dynamic user requirements. Even SQL does not directly support to write constraints or requirements and cannot be used easily for evaluation.

Therefore we decided to go for our own approach by studying our problem domain. We have done a thorough study on the problem domain in real world situation considering the past academic years. After analyzing a large set of constraints and requirements we found that objects can be extracted. Further these objects have relationships among them. We also identified the properties of these objects that are relevant to the constraints and requirements. The result of the analysis is illustrated in the Fig.2.

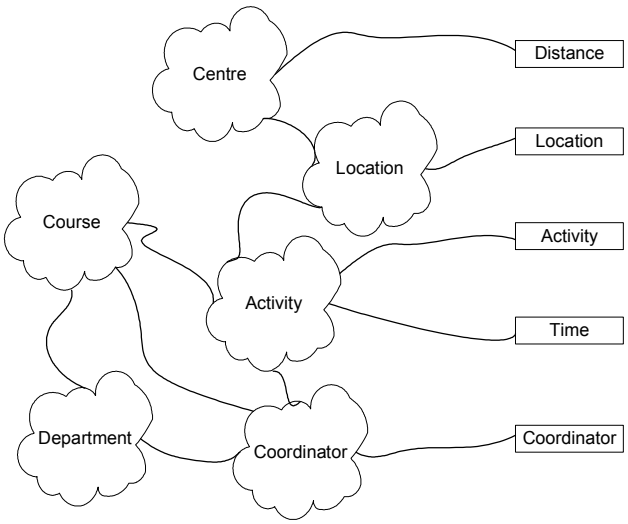

Fig. 2. Relationship among objects and significant properties

Accordingly the activities of courses, time of the activity, location of the activity, distance between the locations (centers) and persons conducting the activity are identified as most significant properties in the domain. This result is very much similar to the elements required for a general timetable as described in [7]. However in our situation we have an additional component, distance. This is needed when scheduling activities at different centers, which are located at a considerable distance where students or academics have to spend much of the time for traveling.

Though we have simplified the problem to a few numbers of properties the next difficulty is how to use them to formulate the constraints and the requirements. We have analyzed the pattern as to how the properties can behave in constraints and requirements. We deduced that all properties have arithmetic, condition and/or logic operations among them. The Fig. 3 demonstrates how to derive basic formula from constraints and requirements.

Constraints/ Requirements

1. Due date for the Assignment #1 of the course ECX2330 must be one week before the Lab Day #1
2. Course coordinator (ECX2330) needs a specific location (Colombo Block 19) and time (9.00 a.m.) for some activities (Lab 1)

Basic Formulas derived from Constraints/ Requirements

1. (Assignment 1 **due date** of ECX2330) **<** (Lab **day 1** of ECX2330) -7
2. ((Lab1 **location** of ECX2330) **= Colombo Block 19**) ***AND*** ((Lab1 **time** of ECX2330) **= 9.00 a.m**)

Fig. 3. Example for deriving basic formulas from constraints and requirements

From this basic formula we can form a common set of formulas varying the properties. For example, if we need to expand the requirement 1 in the Fig. 3 for different courses we should generate similar formulas for each course. Similarly in place of the property, course, we can use any other property, such as activity, location or coordinator. The

Constraints/ Requirements

1. Due date for the Assignment #1 of the courses in the CourseList must be one week before the Lab Day #1
2. Course coordinators in the CoordList need a specific location (Colombo Block 19) and time (9.00 a.m.) for the activities in the ActList.

Common formulas derived from Constraints/ Requirements

1. (Assignment 1 **due date** of CourseList) **<** (Lab **day 1** of CourseList) -7
2. ((ActList **location** of courses of CoordList) **= Colombo Block 19**) ***AND*** ((ActList **time** of courses of CoordList) **= 9.00 a.m**)

Fig. 4. Example for common formulas relevant to the basic formulas given in the Fig. 3

Fig. 4 shows how the requirements in the Fig. 3 can be extended into common formulas. Once we write a common formula it should generate a set of basic formulas. So we do not need to write basic formulas for each instant. Finally this would be the methodology that can be used for analyzing the user requirements in timetable scheduling.

The next task is to find out a suitable approach for modeling our methodology. The approach should be simple for the user as many of them are not professionals in the area of computing. On top of that the selected approach or the system should be flexible enough to use without modifying the application each time for different user requirement analysis.

According to the analysis the solution for the problem would be a scripting language model. This language should be able to define constraints and requirements. According to [8] there are three modeling languages for timetabling problems. According to the paper, however, these languages have no advantages to the user and do not simplify the modeling process. Also in the paper [8] the authors proposed a new modeling framework for Examination Timetabling Problem. Still it requires a fare amount of programming skills from the user. So we propose a new model/scripting language and we called it as Requirement Declarative Markup Language.

IV PROPOSED MODEL: REQUIREMENT DECLARATIVE MARKUP LANGUAGE

We have analyzed the problem domain and all constraints and requirements have been simplified to the simple formulas as given in Fig. 3 and 4. We use the same results to produce this language model. Accordingly it is possible to write statements for constraints or requirements using operands. Such operand of a statement can be written in the following fashion.

CourseList1.Activity1.Location1.Date1

The statements can be constructed using operands with arithmetical and/or conditional operators. As an example,

CourseList1.Activity1.Location1.Date1 <> CourseList1.Activity2.Location1.Date1

Further these statements can be combined and defined into more complicated requirements by using logical operators.

(CourseList1.Activity1.Location1.Date1 <> CourseList1.Activity2.Location1.Date1) AND (CourseList1.Activity1.Location2.Date1 <> CourseList1.Activity2.Location2.Date1)

Finally all statements give true or false value which means, it says, either requirement is satisfied or not. Subsequently requirement statements (RS) can be written according to the formula given in Fig. 5. Though the RS gives one result and looks like one formula it consists of a set of statements, each of them has been generated for accessing a single possible value from the database.

Further there can be different possible combinations to generate statements when we use sets of values for properties in an operand. On one hand it will give flexibility to the user to write in single RS to generate various statements. On the other hand it may generate unnecessary statements as well. However this can be avoided by using with simple RS. The user can indicate the way he/she wants to generate statement by using square brackets ([ ]) in the RS. These brackets indicate that the section within the brackets will generate all possible instances only once. If you omit the bracket it will generate all combinations considering all the properties in the operand.

In addition to the declaration of requirement statements there should be a mechanism to define variables (objects) and constants as well. Moreover it has to provide a facility to map the key fields in the data tables with the properties of the operands. This declarative section can come at the very beginning of the script. In the same section user may define the way to give the results either as a message or as a log file.

V REQUIREMENT ANALYZING SYSTEM

Eventually we can develop a system for RDML. The system, we call as Requirement Analyzing System (RAS), will analyze the requirements defined by the user using RDML. The Fig. 6 illustrates the working principle of the RAS.

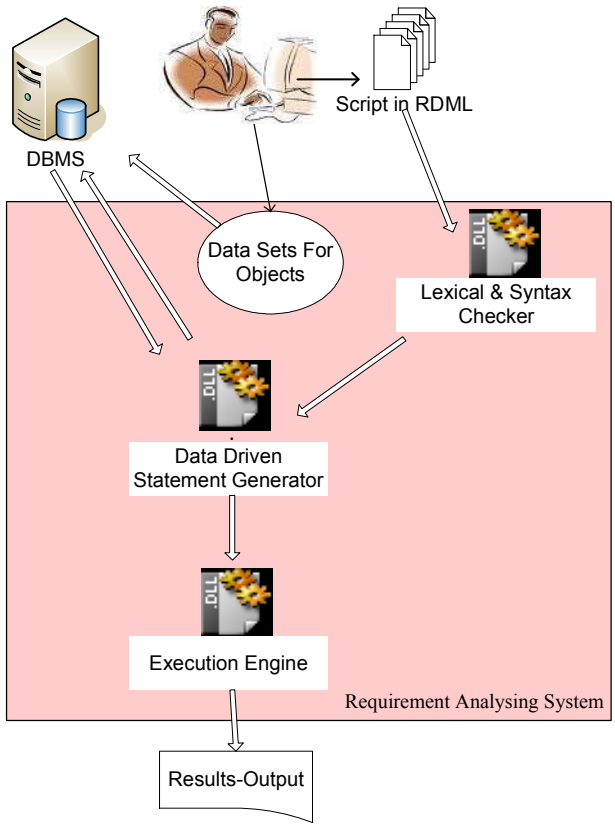

Fig 6. Graphical representation of the Requirement Analysing System

$$RS \equiv \bigotimes_{l,\,k=1}^{p} \left(X_k\right) \qquad l \in \{1,2,3\}$$

where

$$\otimes_1 \equiv AND \qquad \otimes_2 \equiv OR \qquad \otimes_3 \equiv XOR$$

$$X_k = property_{n_k}\left(\left(object_i\right)_{i=0}^{n_k}\right)\left[\oplus Constk_1\right]\Theta\ property_{m_k}\left(\left(object_j\right)_{j=0}^{m_k}\right)\left[\oplus Constk_2\right],$$

where

$$property_{n_k}\left(\left(object_i\right)_{i=0}^{n_k}\right) \equiv object_1.object_2.\cdots.object_{n_k}.property_{n_k},$$

n_k - last object of the first operand in the k^{th} statement,

m_k - last object of the second operand in the k^{th} statement,

[ ] - optional,

$\Theta \in \{=, >, <, >=, <=, <>\}$,

$\oplus \in \{+, -\}$.

Fig. 5. General formula for requirement statements

The system has 3 Engines: Lexical and Syntax Checker, Data Driven Statement Generator, and Execution Engine. Using the RDML we can write all constraints and requirements as a program. The Lexical and Syntax Checker checks the script for lexical and syntax errors. If there are no errors it will pass to the Data Driven Statement Generator where generates statements to access data from the database. Thereafter Execution engine fetches the data according to the Data driven statements. Execution starts with the arithmetical calculations. Thereafter it executes condition operations in the order of the appearance in the statement. Finally it will do the logical operations. This will be repeated for the whole set of data. After executing all requirement statements Execution Engine generates Messages and/or Log files as requested by the user.

VI HOW RDML AND RAS CAN BE USED

Here we briefly describe how to use the RDML and RAS. The first step is to identify how the data being organized at the application level. This includes a good knowledge of the database structure of the application. The next step is analyzing the requirements of the user and formulating them into requirement statements. Once these statements are ready it is possible to define the constants and variables to be used for RDML. And also map the key fields with the properties of the operands. Finally all requirement statements can be converted into RDML statements. Then the RAS will do the rest.

The RAS can be used at any point of the application either during the data entry or after completing the data entry. In our case we have used after generating the timetable i.e. with the final set of data.

VII CONCLUSION AND FUTURE WORK

This paper introduces a scripting language, Requirement Declarative Markup Language, which can be used for timetabling to analyze user requirements and find the fulfillment of them. It can also be converted to a clash-checking program in the manner the requirement statements are written. The system RAS developed for the use of RDML gave accurate results when tested for the faculty timetable data. The simple structure of the RDML is a major advantage and it can be easily used with relational databases.

In the future we hope to study further to use the RDML for analyzing all kinds of databases for different applications. The RDML, as it is, will be a good tool for the research community in the area of timetable automation.

REFERENCES

[1] Barry McCollum, "University Timetabling: Bridging the Gap between Research and Practice", *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*, ISBN 80-210-3726-1, pp. 15-35, 2006.

[2] The International Series of Conferences on the Practice and Theory of Automated Timetabling (PATAT), http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml.

[3] Keith Murray, Tomáš Müller, "Automated System for University Timetabling", *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*, ISBN 80-210-3726-1, pp. 536-541, 2006.

[4] A.Schaerf, "A Survey of Automated Timetabling", *Artificial Intelligence Review*, 13/2, pp. 87-127, 1999.

[5] Tomáš Müller, "Constraint-based Timetabling" - Summary of Ph.D. Thesis, http://www.smas.purdue.edu/research/papers/phdab05.pdf, 2005.

[6] Tomáš Müller, Roman Barták, "Interactive Timetabling", *Proceedings of the ERCIM Workshop on Constraints,* 12 pages, Prague, June 2001. http://arxiv.org/ftp/cs/papers/0109/0109022.pdf.

[7] M. Radaideh, S. Horani, M. Raseen, "WBSGA: A Web-based Tool for Course Timetabling and Scheduling", *International Journal of Computers and Applications*, Vol. 28, No. 1, pp. 74-83, 2006.

[8] David Ranson, Samad Ahmadi, "An Extensible Modelling Framework for the Examination Timetabling Problem", *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*, ISBN 80-210-3726-1, pp. 281-292, 2006.