

DEVELOPMENT OF AN AUTOMATED ROSTER GENERATOR USING REQUIREMENT DECLARATIVE MARKUP LANGUAGE

L. S. K. Udugama and G. T. D. Perera

Department of Electrical and Computer Engineering, Open University of Sri Lanka

INTRODUCTION

Timetables are prepared in any organization whether it is a small or a large company. Usually they are prepared manually and it takes a long time to complete the task. Due to various factors such as limited resources and constraints of the organization, the final outcome may not have fulfilled the necessary requirements. Nevertheless it may be prone to human errors as well. Therefore a number of attempts have been made to automate this process (Schaerf, 1999). According to the literature, different methods have been used for the automation of timetabling. The most widely used techniques and algorithms are genetic algorithms, tabu search, graph colouring heuristics, mathematical programming, simulated annealing, network flow models, and constraint programming (Schaerf, 1999, Benli & Botsali).

Preparation of rosters is very much similar to the timetabling problem. A number of publications can be found for roster preparation (Nielsen & Mason, Nurse Rostering). Similarly, techniques and algorithms, as used in timetabling, have been also applied for rostering.

In this paper we present a development of an Automated Roster Generator (ARG) for Sri Lanka Rupavahini Corporation (SLRC). In this development a different technique has been adopted in the place of above mentioned techniques and algorithms. Authors Liyanage & Udugama (2007) proposed Requirement Declarative Markup Language (RDML) specially designed for defining users' requirements. In that paper RDML was used for verifying user requirements after the completion of the faculty timetable. However, in this development RDML is applied differently and it is used for automating the preparation of roster. Although this had been done on an experimental basis, it gave us very satisfactory results.

METHODOLOGY

Intention of the research was to develop a technique to generate a roster automatically using RDML. After analysing the RDML and the design of the Requirement Analysing System presented by Liyanage & Udugama (2007), it revealed that the RDML can be employed for automation of rosters too. For that purpose it is necessary to design a Rules Checker (Figure 1) where all the Requirement Statements (RS) (Liyanage & Udugama, 2007) are verified according to the values passed in. Subsequently Rules Checker will indicate whether the given values have been satisfied the set of RS or not. The Roster Engine generates a set of data for the Rules Checker according to an algorithm already programmed. On the other hand it can act according to the output of the Rules Checker as well, when generating the data set for Rules Checker. After integrating other supportive components ARG can be depicted as in the Figure 1.

In order to use this ARG, first of all it is necessary to analyse the organization requirements and the rules pertain to their staff roster. As the target was to generate a roster for SLRC Master Control Room staff, analysis was done only for the relevant section. Accordingly the constraints, requirements and the manual system of the roster preparation have been studied thoroughly. Eventually objects in the problem domain were extracted. Further, relationships among those objects and the properties of each object were identified. The result of the analysis is illustrated in Figure 2.

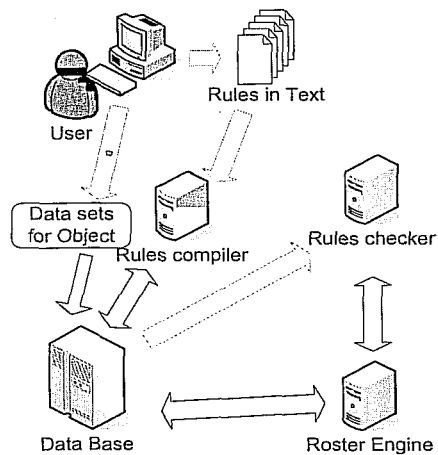


Figure 1. Design of the Automated Roster Generator

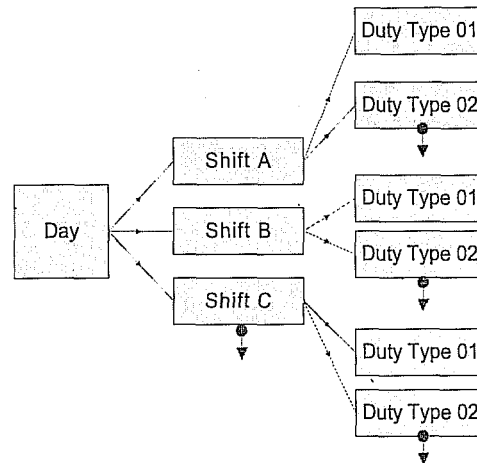


Figure 2. Relationship among objects and properties

As the next step, the set of RS were defined considering all the above objects and the rules of the organization for scheduling a roster. Some of the rules that were considered during the development are:

1. Workers can be assigned to one shift per day only.
 2. Workers can't be assigned on their off days.
 3. Female workers cannot be assigned to an overnight shift.
 4. Always try to assign expert workers for a particular duty type.
 5. Do not assign workers to overnight shifts more than the number defined.
- ... and so on.

According to the above rules it is necessary to have different data types such as variables, constants etc. After identifying those data types, finally RS can be generated to all corresponding rules. For example,

1. AssOnlyOneShift \diamond Day
 2. OffDay \diamond Day
 3. ((Shift = OverNightShift) AND (Gender = Male)) OR (Shift \diamond OverNightShift)
 4. WorkerAbility = Expert
 5. (AssignedOverNightShift < HavingOverNightShift) OR (Shift \diamond OverNightShift)
- ... and so on.

where Day, Shift, Expert and Gender are variables.

As the RS are ready, a roster can be generated now using the ARG. In this particular example, according to the algorithm used for the Roster Engine, all possible combinations among the elements of each and every object were created for the data set.

RESULTS AND DISCUSSION

For the implementation of the ARG software Visual Basic 6.0 was used as the programming language and MS Access 2003 for database handling. In order to generate the roster, the real data of SLRC were used. The snapshot of the roster generated by the ARG is shown in Figure 3 and a sample of a roster prepared manually is depicted in Figure 4. There were 38 technical staff to be scheduled in 3 working shifts per day and the roster was generated for one week time period. The scheduling was done for 4 different expertise categories of the staff having various expertise levels. They are Transmission (Tx), Digital Editing (DE), Non-Linear Editing (NLE), and Linear Editing (LE). Transmissions Tx1, Tx2, and Tx3 indicate Rupavahini, Eye and NTV respectively. According to the rules of the organization 8 RS were defined and there were 6 hard rules.

To operate the algorithm efficiently, order of assigning a duty point to a particular worker is crucial. Therefore, the most critical duty points, shifts of the day, and workers have to be identified. Moreover, it is necessary to start the rostering from the most critical data. Accordingly the roster (Figure 3) was generated within around two and half minutes on a computer with 3 GHz Core 2 Duo processor, 2 GB memory and running Windows XP operating system. The last column of Figure 3 shows the number of workers required for each duty point in a shift.

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Req. Workers
00.00 - 08.00	A3-Tx1, B7-Tx1, A5-Tx1, A6-Tx2, A11-Tx2, B19-Tx2, A20-Tx3,	B3-Tx1, B4-Tx1, B5-Tx2, B9-Tx1, A9-Tx2, A13-Tx3, A18-Tx2,	B1-Tx2, A3-Tx1, B6-Tx2, A5-Tx1, A6-Tx1, A10-Tx2, B16-Tx3,	B2-Tx2, B7-Tx1, B9-Tx1, A8-Tx2, A11-Tx1, B17-Tx3, A21-Tx2,	B1-Tx2, B3-Tx1, A3-Tx1, A4-Tx2, A6-Tx1, A10-Tx2, B12-Tx3,	B7-Tx1, A5-Tx1, B9-Tx1, A8-Tx2, B10-Tx3, A11-Tx2, B21-Tx2,	B1-Tx1, B3-Tx1, B4-Tx1, A4-Tx2, B8-Tx3, A9-Tx2, B12-Tx2,	DE=0, LE=0, NLE=0, Tx1=3, Tx2=3, Tx3=1,
08.00 - 16.00 OR 08.30 - 16.30	A4-Tx1, B8-Tx3, C2-Tx1, A17-LE, B12-Tx2, B14-NLE, A15-NLE, B15-NLE, A16-DE, B18-LE, B20-LE, B21-Tx2,	B8-Tx1, A7-LE, C2-Tx1, A17-LE, B14-NLE, A15-NLE, B15-NLE, A16-DE, B18-LE, B19-Tx2, B20-Tx2, A19-Tx3,	B4-Tx1, B5-Tx2, A7-DE, A9-Tx2, C2-Tx1, A17-LE, B14-NLE, A15-NLE, B15-NLE, B19-Tx3, B20-LE, A19-LE,	B5-Tx1, B6-Tx2, A7-DE, B10-Tx2, C2-Tx1, B14-NLE, A15-NLE, B16-LE, B19-Tx3, B20-LE, A19-NLE, A20-LE,	B2-Tx1, B6-Tx2, A7-DE, B11-NLE, C2-Tx1, A13-LE, A15-NLE, B16-LE, B17-LE, A16-Additional, B18-NLE, A21-Tx2, A20-Tx3,	B2-Tx2, B6-Tx3, A7-DE, B11-NLE, C2-Tx1, A10-Tx1, A13-LE, B14-Additional, B15-NLE, B17-LE, B18-NLE, A10-LE, A21-Tx2,	B2-Tx2, A8-Tx1, B10-Tx2, B11-NLE, B14-NLE, A15-NLE, A16-DE, B18-LE, A18-LE, A19-LE, A21-Tx3, B21-Tx1,	DE=1, LE=3, NLE=3, Tx1=2, Tx2=2, Tx3=1,
14.30 - 22.30	B2-LE, A8-LE, B10-LE, B11-DE, B17-NLE, A21-NLE,	A1-LE, B10-NLE, B11-DE, B12-LE, A20-NLE, B21-LE,	B8-LE, B11-DE, A13-NLE, B18-NLE, A18-LE, B21-LE,	B4-LE, B8-LE, A9-LE, A17-DE, B15-NLE, A16-Additional, A18-NLE,	B5-LE, A17-DE, B15-NLE, B19-LE, B20-LE, A19-NLE,	B5-LE, A17-DE, B16-LE, A16-Additional, B20-NLE, A19-NLE, A20-LE,	B6-LE, A7-DE, A10-LE, A13-NLE, B16-LE, B17-NLE,	DE=1, LE=3, NLE=2, Tx1=0, Tx2=0, Tx3=0,
16.00 - 24.00	B3-Tx1, B4-Tx1, B5-Tx2, B9-Tx1, A9-Tx2, A13-Tx3, A18-Tx2,	B1-Tx2, A3-Tx1, B6-Tx2, A5-Tx1, A6-Tx1, A10-Tx2, B16-Tx3,	B2-Tx2, B7-Tx1, B9-Tx1, A8-Tx2, A11-Tx1, B17-Tx3, A21-Tx2,	B1-Tx2, B3-Tx1, A3-Tx1, A4-Tx2, A6-Tx1, A10-Tx2, B12-Tx3,	B7-Tx1, A5-Tx1, B9-Tx1, A8-Tx2, B10-Tx3, A11-Tx2, B21-Tx2,	B1-Tx1, B3-Tx1, B4-Tx1, A4-Tx2, B8-Tx3, A9-Tx2, B12-Tx2,	A3-Tx1, B7-Tx1, A5-Tx1, A6-Tx2, A11-Tx2, B19-Tx2, A20-Tx3,	DE=0, LE=0, NLE=0, Tx1=3, Tx2=3, Tx3=1,
OFF	B1, B6, A7, A10, B16, A19,	B2, B7, A8, A11, B17, A21,	B3, A4, B10, B12, A16, A20,	A5, B11, A13, B18, B21,	B4, B8, A9, B14, A18,	A3, A6, A15, B19,	B5, B9, C2, A17, B15, B20,	

Figure 3. Snapshot of the roster automatically generated by the ARG

According to the final result, none of the rules had been violated. However the manual roster (Figure 4) itself had errors when it was published including violation of some of the rules. Those are indicated by circles and boxes in the figure itself. However, these kinds of faults/errors could not be found in the automated roster.

M.C.R. S'AFF ROSTER - SEPTEMBER 2010 (From 01 to 30 SEPTEMBER 2010)

hrs.	SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
0800 - 0800	A5, A11, B11	A6, B10, A9, A11	B3, B1, B7, A10, A3	B9, A4, A6, A3	B4, B6, B16, B17	B12, A3, B7, A11	B3
0800 - 1600	A17, B16, A18 B20, A4 A10, B17, B12, A19	B11, A4, A7, A15 B14, A17, B16, A18 A8	B10, B4, B6 B10, B14, B18 A13, A16, B11 A9, A8	A9, A10, A13, A19 A17, B18, B19, B20 A8, B21	B8, B5, B10, A16 A15, B21, A19, A13 A8	A7, A18 B14, A20 B18, B20 A8, A19	B2, B3, B14 B15, B16, A16 B18, B21, B19, B20 A20, A15
0800 - 1600 / ROT		A11, A19, B19, B20		A7, A10, A13, A19 A17, B18, B19, B20	B8, B5, B10, A16 A15, B21, A19, A13	A7, A18 B14, A20 B18, B20 A8, A19	B2, B3, B14 B15, B16, A16 B18, B21, B19, B20 A20, A15
0830 - 1630							
0830 - 1630 / ROT							
0900 - 1900							
1430 - 2230	A21, B15 A7	A13, A21, B2, B2 B4, A16 B9, B18, B17	A15, A19, A20, A7 B2, B19, B15 A17	B3, B5 B10, A16, B12, A15 B14	B1, A20, A18 A7, B20, B15 B9, A17, B19	B2, B5, B6 B7, B1, A21, B15 B21, B19	A21, B21
1600 - 2400	A11, A20, B21, A6 A3, A9, B10	A5, B1, B5, B7 B3, B12, A10	B9, A3, A6, A18 A4, A11, A21	B3, B4, B11, B2 B16, B6, B17	A5, A3, A6, B12 B7, A10, A11	B9, B17, A9, A4 B3, B4, B8	A5, B1, B7, B6 A10, A11, B11
OFF	A8, B2, B4, B5 B9, B11, B19 A6, B1, B12	B6, B15 B20	B21, B17 B16, A15	A5, A20 B1, B7	A21, A7 A9, B5	B11 A13	B10, A3, A6, A7, A8 A8, A10, A17, A18

SHEET - A

A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21

This post has been removed as per S.I.C request.

SHEET - B



B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
B12
B13
B14
B15
B16
B17
B18
B19
B20
B21

This post has been removed as per S.I.C request.

SHEET - C

C1
C2

1. T.OO in "A" shift will be relieved by the respective T.OO's in "B" shift. This should be very strictly adhered to.
2. During the week ends (Saturdays & Sundays) T.OO who are detailed to report for duty at 0800 hours should continue till 2200 hours however OIC MCR may release any officer after completing the duties assigned to him.
3. Please note that no programme can be cancelled due to lack of staff. Duty roster will be reviewed at short notice if necessity arise.

DE(Ops) cc:Actg.DDG(E) DDE(MCR), ADE(MCR), AO(E)

Figure 4. Sample of a roster prepared manually

In addition to that, ARG software has a facility to make duty changes at anytime. The system will allow any change in accordance with RS.

CONCLUSIONS/RECOMMENDATIONS

The results of the new approach indicate that RDML can be used for automation the roster preparation. As there were no errors found, the approach can be also employed for other similar kind of scheduling tasks. However the algorithm used in the Roster Engine needs further improvements in order to get quick results with a large set of data. The algorithm operates efficiently when the data are organized according to their priority, which depends on the behaviour of the organization concerned.

REFERENCES

Benli Omer S. & Botsali A. Reha. An Optimization-Based Decision Support System for a University Timetabling Problem: An Integrated Constraint and Binary Integer Programming Approach. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.9323&rep=rep1&ctype=pdf>

Liyanage Arunasiri K. & Udugama L.S.K. (2007). "Requirement Declarative Markup Language" for Timetabling: A Case Study Based Development. *Second International Conference on Industrial and Information Systems, ICIIS 2007, Sri Lanka*, pp 249-253.

Nielsen David & Mason Dr Andrew. Commercial development of an optimisation-based roster engine. Retrieved from www.orsnz.org.nz/conf33/papers/p87.pdf

Nurse Rostering Papers 2007-2008. Retrieved from www.asap.cs.nott.ac.uk/watt/resources/NR_2008_REFS.pdf

Schaerf A. (1999). A Survey of Automated Timetabling, *Artificial Intelligence Review*, 13/2, pp. 87-127.