Kalpa Publications in Computing

Volume 12, 2019, Pages 281–289

Proceedings of 4th International Conference on the Internet, Cyber Security and Information Systems 2019

Cost Implications for an In-House University Timetabling System

Wame Raseonyana, George Anderson and Tallman Nkgau
Department of Computer Science
University of Botswana, Gaborone, Botswana
wraseonyana@gmail.com, georgeganderson@gmail.com,
nkgautz@gmail.com

Abstract

This paper reports on a study on the examination timetabling problem faced in universities, focusing on the problem as it exists at the University of Botswana (UB). Examination timetabling is well researched and many different algorithms have been used in an attempt to produce timetable solutions that meet various objectives of institutions as well as theoretical objectives. Our research tries to produce an optimal examination timetable, which takes into consideration constraints imposed by users of timetables in UB as well as best-practice constraints found in the literature. Most African universities have financial constraints that make cost-effective solutions that take advantage of readily available research results and frameworks critical to their survival and strategic goals. In our paper, we analyse the cost implications of in-house development of an examination scheduling system solution that will produce improved schedules and reduce costs. We take into account the cost of software tools and frameworks we have employed as well as the cost of getting data and actual software engineering of a timetabling solution. Our results suggest that the benefits of this approach are great, since the cost is low while the quality of resulting timetables is good.

1 Introduction

Timetabling is present in many domains, such as, nurse rostering, public transport system, manufacturing industry and educational timetabling. Educational timetabling is one of the most widely studied timetable problems. In an educational setting, the two main types of timetables are course timetabling and exam timetabling. Some universities produce timetables manually which results in an infeasible solution that is time consuming to produce and highly error prone (Chmait & Challita, 2013). The existence of modularity, flexibility in student curricula, increasing student numbers and continued expansion of university departments has led to complex timetables, which require automated timetabling software systems (Arogundade, Akinwale, & Aweda, 2010). There is vast literature on

possible approaches to tackling the timetable problem faced at universities these include: Operational Research (graph coloring, Integer programming, Linear programming, Constraint based techniques), Metaheuristics (Genetic Algorithm, Tabu Search, Simulated Annealing, Great Deluge, Hill Climbing) and finally automated systems. Each one of these approaches comes with advantages and disadvantages therefore organisations need to take in a number of factors into consideration before opting for a particular approach. Some crucial factors to consider are: cost of solution, solution implementation, time requirements and solution technical requirements. In our study, we estimate cost of development based on lines of code in a prototype, and use online sources to source cost of purchasing timetable software suitable for universities, which we compare against in-house development. Our study is significant because universities are faced with funding constraints, and it is important to explore options other than purchase of expensive timetabling software, which is critical for operations. Our paper is focused on exploring cost issues, and not on technical details of producing optimal timetables.

The rest of this paper is organized as follows: Section 2 has a literature review; Section 3 discusses the methodology; Section 4 reports findings; Section 5 has a discussion of findings; and Section 6 has a conclusion.

2 Literature Review

2.1 Approaches to Software Development

A number of options are available to organisations in search of an approach to adopt for developing software solution, these are: In-house development from scratch, purchasing Commercial off-the-shelf (COTS) and adopting and/or customizing Open Source Software (OSS) (Badampudi, Wohlin, & Petersen, 2016). In-house solutions are developed and maintained in an organization using its own resources therefore the solution's Intellectual Property rights are owned by the organization. The main advantage of such solutions is that they are build using the exact requirements of an organization and because in house developers have complete access to source code they can make modification to the solution when the need arises. In-house development is also beneficial because it helps build software development skills internally. In-house development does have some disadvantages though, developing a system from scratch can be very costly and time consuming. Furthermore a well skill in-house software system development team is required in order to ensure successful implementation of the solution. COTS are commercial solutions purchased from an external company or vendor. COTS can quickly adopted for use in organisations without any delays. COTS are tried and tested solutions therefore users can be guaranteed of up to standard functionalities. The problem with COTS is that they are not tailor made to meet aparticular organization's requirements, system functionality is determined by the vendor therefore purchasing such solutions requires solutions to either be customized to fit the organisation's needs or there is need for an organisation to change its requirements to fit the solution. Another disadvantage of COTS is that source code rights are owned by the vendor resulting in restrictions on use and it comes in binary form only restricting users from modifying it therefore there is dependency on vendor for support on issues as well as for updates. COTS tend be costly, costs are incurred during the purchase of the solution as well as from software license fees which allow for the provision of vendor user support. OSS solutions are obtained for free from the OSS community. The meaning of free here is relative, it may mean that the solution source code is available free of charge or it may mean that solution source code is openly available for customization. The open source license allows users to redistribute software, modify source code and charge for code redistribution for as long as changes are publicly available (Riehle, 2007). The use of OSS allows for the development of high quality software solutions that is inexpensive to develop. Developers also have access to the source code therefore they can add desired functionality to the software solution, this also means that there is

reduced level of dependency on software vendors. The use of OSS also help develop software development skills for employees within an organization.

2.2 Application Of OSS Development In Educational Timetabling

There are several automated systems developed for timetabling some COTS and others OSS. Example of COTS for timetabling are MIMOSA, CELCAT andSyllabus Plus by Scientia, examples of OSS for timetabling include UniTime, Optaplanner and Free Educational Timetabling (FET). Several universities have resorted to the use of OSS in timetabling in an effort to reconcile tight budgets with the rising cost of technology. The University of Malaysia Perlis adopted the use of FET for course timetabling since 2012, successful implementation was achieved and efficient timetabling was reported. Muhamad et al. (Muhamad, Adnan, Yahya, Junoh, & Zakaria, 2018) also reported that the use of OSS at their university resulted in cost savings and optimal resource usage. Successful implementation of OSS in timetabling was also reported by (Mansor, Arbain, & Hassan, 2013) were FET produced good quality solutions timetabling 300 courses for 10000 students. A common OSS in educational institutions is Moodle course management system, which has been adopted by a number of universities worldwide (Williams van Rooij, 2007). UniTime was deployed to solve the timetable problem at Purdue University.

2.3 Examination Timetabling

The examination timetable problem can be divided into two main categories (Kahar & Kendall, 2010): uncapacitated and capacitated examination timetabling. In an uncapacitated exam timetable problem room capacities are not considered when scheduling whereas incapacitated exam timetabling room capacities are important and they constitute a hard constraint. The examination timetable problem is such that given a set of exams, a fixed number of time slots, and a fix number of rooms, allocate each exam to a time slot in a certain room so that a feasible timetable is produced. A feasible solution is one that does not violate any hard constraints. Hard constraints are those conditions or rules that must not be broken or violated during the production of the timetable. Soft constraints are those rules that it is desirable to satisfy but not essential.

2.4 Software Development Cost Estimation Models

Software cost estimation is the process of determining the effort required to develop a software system. There are many software cost estimation models and each model has advantages and disadvantages. Software cost estimation models can divided into two classes: Algorithmic models and Non Algorithmic models (Keim, Bhardwaj, Saroop, & Tandon, 2014). Algorithmic models use formulas to estimate cost whereas non-algorithmic models do not use formulas. Examples of algorithmic models include COCOMO (Constructive Cost Model) and Function Points (Keim, Bhardwaj, Saroop, & Tandon, 2014). COCOMO can be used for estimating project size, effort, cost, time and quality. The main unit of measure used of COCOMO is SLOC (Source lines of code). COCOMO is made up three models with levels of complexity: Basic, Intermediate and Advanced. The Basic model gives a rough estimate and is usually used in early stages of development. Intermediate model can be used at mature project development level and is more accurate. Advanced is the most accurate model. COCOMO can be used to estimate cost different types of projects. The different types of projects are Organic, Semi-detached and Embedded. Organic projects are small, require less innovation because they are similar to past projects done. Semi-detached projects are the type that fall in between Organic and Embedded projects. Embedded projects have complex constraints and require much effort. Function Points model does not use SLOC as input instead it uses the number of input transactions and the number of unique report (Kemerer, 1987). Some examples of non-algorithmic

models are Top down estimation model, Bottom up estimation model, Estimating by Analogy and Expert judgement (Keim, Bhardwaj, Saroop, & Tandon, 2014). Top down estimating is done by estimating the total cost of the project then partitioning the project into low level components and then estimating the cost of each. Bottom up model does the opposite cost of each low level component is calculated and the costs are combined to estimates total project cost. Historical data from similar projects is used to estimate cost when using the Estimating by Analogy approach, this model is not so accurate because its success depends on data of past projects and it is possible that this data may be unavailable or incomplete. In the Expert by judgment model the experience and knowledge of experts is used to calculate the cost of the project. Moulla (Moulla, 2013) presented work that uses COCOMO to evaluate the cost of adapting TRIADE: an open source e-learning platform at the University of Cameroon. The cost was calculated by estimating the difference in the number of lines between the original TRIADE solution and the modified TRIADE solution. The difference in the number of lines between the two solution was then applied to the COCOMO formula to calculate the cost of the modified solutions. According to Moulla (Moulla, 2013) results showed that OSS does have the advantage of reducing the cost of development compared to other approaches. The approach presented by Moulla (Moulla, 2013) can be applied to our situation we can also estimate cost using lines of code.

3 Methodology: Using Unitime OSS To Solve The Examination Timetable Problem At UB

Unitime is a Java OSS educational scheduling system for universities. It offers Curriculum-based and Enrollment-based Course Timetabling. Examination Timetabling and Student Sectioning. UniTime is used in several universities including Purdue University in the USA where it was originally developed and is used in scheduling large student enrollments (around 40,000 students). MIT first introduced the use of UniTime in 2014 with the aim of expanding the functionality and flexibility of their scheduling process. UniTtime replaced their outdated system enabling support for the definition of different institutional periods, integration of scheduling with the curricular-review process and complex subject configurations (Flessner-Filzen, 2015). A number of researchers have used UniTime in an attempt to improve timetabling at different universities and (Čupić & Franović, 2011) calls it the most state-of-the-art implementation of an automatic scheduler. (Lukáš, 2019) implements a set of new features on UniTime to cater for course timetabling requirements necessary at the Faculty of Education and the Faculty of Arts at Masaryk University in Czech Republic where the system has been in use since 2010. Following the addition of this feature on UniTime feasible timetables were produced. We tackle the examination timetable problem at UB using UniTime. The UniTime solver is based on local search algorithms (Tabu Search, Simulated Annealing, Hill Climbing and Great Deluge). We customize the Unitime solver by introducing Genetic Algorithm into it and creating hybrids of this algorithm with local search algorithms. We also customized the software by introducing certain timetabling requirements which are unique to the examination timetable problem in UB. Timetable solutions produced by each hybrid algorithm are evaluated and compared with others and the best hybrid for UB dataset is identified and recommended. Our proposed solution produced better quality compared to the current system used at the university.

4 Software Cost Estimation for Implementing UniTime to Solve Examination Timetabling Problem at the University of Botswana

The estimated cost of deploying UniTime to tackle the examination timetable at UB was calculated using the COCOMO estimation model. This model is a good fit for our project because according to (Keim, Bhardwaj, Saroop, & Tandon, 2014) it is suitable for early cost estimation. Our project is at an advanced stage of development, therefore we have an accurate estimate of code size. In this approach cost estimation is performed using globally available project properties. We considered the following factors when estimating the cost of software: effort (usually in person months), project duration (in Calendar time) and cost (in monetary value). The effort estimate can be converted to monetary value by calculating an average salary per unit time of staff involved and then multiplying this by the estimated effort required. The most determining factor of cost estimation of software development is human effort and most cost estimation methods use this factor to give estimates in terms of person-months. Accurate software cost estimates are important, underestimating costs may result in poor quality solutions that exceed the budget and set time whereas overestimating cost may result in excess resources being allocated to a project. The estimated cost is calculated as follows.

$$Estimated\ cost = \frac{Person\ Salary}{12} \times Person\ months$$

We used the Government C2 salary scale highest notch to estimate the Average Person Salary of a developer assigned the task of using Unitime to develop an exam timetable system for the university. The highest notch is BWP 147 252.00 per annum.

$$Average\ Person\ Salary\ per\ Month = \frac{147252}{12} = 12271.00$$

Person months expresses effort a person devotes to a specific project. To below formula shows how person months is calculated using COCOMO using the main unit of measure: thousands lines of code. Our proposed solution is estimated to have 2000 lines of codes. The breakdown of the estimate appears in Table 1.

Table 1. Breakdown of estimate of number of lines of code

Class/Module/Algorithm Implementation	Number of lines of code
Genetic algorithm implementation	600
Hybrid algorithm implementation	100
Exam model implementation	100
Data pre-processing	200
Creation of system input files	150
Creation of reports (output files)	200
Timetabling constraints	300
Other class modifications	350

This number of lines of codes is estimate is fairly accurate, due to the advanced stage of development. Several scheduling algorithms have already been implemented. The thousands lines of code has to be converted to Kilo lines of code, therefore $2000=2$ kilo lines of code.

$$Person\ Months = a \times (KLOC)^b = 3.6 \times 2^{1.20} = 8.27$$

Where $KLOC$ is Kilo Lines of Code and a and b are COCOMO constants. The Project duration is given by the number of Person months that is going to be spent on the project. This value was derived taking into consideration the time required to learn and become familiar with Unitime OSS. The cost of project in the calculated Project Duration is:

$$Duration\ in\ Months = c \times (Person\ Month)^d = 2.5 \times 8.27^{0.32} = 4.9$$

Where c and d are COCOMO constants. The cost estimation of our project using the values calculated above is

$$Estimated\ cost = Duration\ in\ Months \times Average\ Person\ Salary\ per\ Month$$
$$= 4.9 \times 12271.00 = BWP\ 60{,}27.90$$

5 Discussion: Cost Analysis Of OSS Vs COTS Vs In-House Development

OSS is a tradeoff between COTS and In-house development. OSS has become a subject of much interest resulting in the resolution of some core issues in software development such as long development periods and budget constraints. An investigation of cost analysis of these 3 approaches reveals that the advert of OSS has resulted in lower software cost compared to those of In-house development and COTS (Riehle, 2007). OSS development achieves cost cutting by allowing organisations to save money that would have otherwise been used on license fees for COTS (Williams van Rooij, 2007). COTS may also be viewed as expensive because they tend to be complex and may require specialized resources which are expensive to maintain. OSS development is also considered to be much cheaper than In-house development because with OSS the software is not developed from scratch therefore it can be completed with less resources in a reasonable amount of time in contrast In-house development is perceived to require more time for development and this raises the cost of development. Several organisations have deployed OSS as a cost saving measure. According to (Williams van Rooij, 2007) OSS development leads to lower cost of ownership. Adoption of OSS as a cost saving measure at Beaumont Hospital can be seen in (Glynn, Fitzgerald, & Exton, 2005) where information systems were re-deployed using OSS. Villano (Villano, 2006) further supports this claim by stating that Campus Technology saved 20% in annual license fees by switching from commercial software to OSS. Different vendors have different prices for their software solutions, depending on the features available a vendor can have different packages. Table 2 indicates package prices of different COTS software solutions. Since we are focused on a problem in Botswana, we have converted US dollar and Euro amounts into BWP (Botswana Pula) amounts using an exchange rate of 1 USD = BWP 10.72, and 1 Euro = BWP 12.18.

The COTS cost for Syllabus Plus was not available at the time of writing this article. We therefore decided to use the cost of other timetabling COTS as seen in Table 2 and Table 3 to analyse how they compare to the use of UniTime OSS for exam timetabling in UB. The comparison clearly shows that UniTime costs less than other solutions over a period of 20 years. Out of the other solutions, Mimosa is the only on with features for universities, as well as schools; Prime Timetable and ASC Timetables are targeted at schools. They therefore do not have all the necessary features to support UB requirements. Prime Timetable appears to be close in cost to UniTime OSS, but the quoted cost is for a license which supports not more than 100 teachers. UB has a much higher number of lecturers. Another issue is that the other solutions require separate support contracts, which would drive the cost even

Table 2. Prices for COTS used in Timetabling.

Software Solution	Package Type	Cost (BWP)	Package Features
Mimosa	Single License	6,085.00	Installation on one computer only. Free updates, email and phone support.
Mimosa	Site License	9,736.00 (< 800 students) 97,440.00 (8000 students x 12.18	Installation on one computer only. Free updates, email and phone support.
Prime Timetable	Basic plan	1,597.28/year	Up to 40 teachers
Prime Timetable	Premium plan	3205.28/year	Up to 100 teachers
ASC Timetables	Primary schools	1,929.60	
ASC Timetables	Standard	2,394.40	Unlimited scheduling capabilities
ASC Timetables	Premium	5,360.00	Unlimited scheduling capabilities. Timetabling user support.
ASC Timetables	Pro	21,386.40	Generates individual student-specific schedules.

higher. UB has a student population of over 12000 students and over 1300 therefore we need a software solution that is scalable. The cost of Mimosa is very high compared to the cost of deploying Unitime, and this is representative of university-grade timetabling software. Mimosa is a state of the art software and compares well with our proposed system. Our cost is slightly greater than that of Prime Timetable and the reason for this may be that our system has more features. The cost of using our proposed system is cheaper because payment is done one time unlike the above COTS software which require year subscriptions, overtime their costs will surpass that of our system, this is can be seen in Table 2. Figure 1 shows growth of cost for the solutions considered in Table 3, on a log-scale. The UniTime OSS is clearly cheaper.

Table 3. Comparison of cost of COTS versus cost of UniTime OSS deployment in UB.

Software	Cost/2 years	Cost/5 years	Cost/10 years	Cost/20 years
UniTime OSS	60,127.90	60,127.90	60,127.90	60,127.90
Mimosa	194,880.00	487,200.00	974,400.00	1,948,800.00
Prime Timetable	6,410.56	16,026.40	32,052.80	64,105.60
ASC Timetables	42,772.80	106,932.00	213,864.00	427,728.00

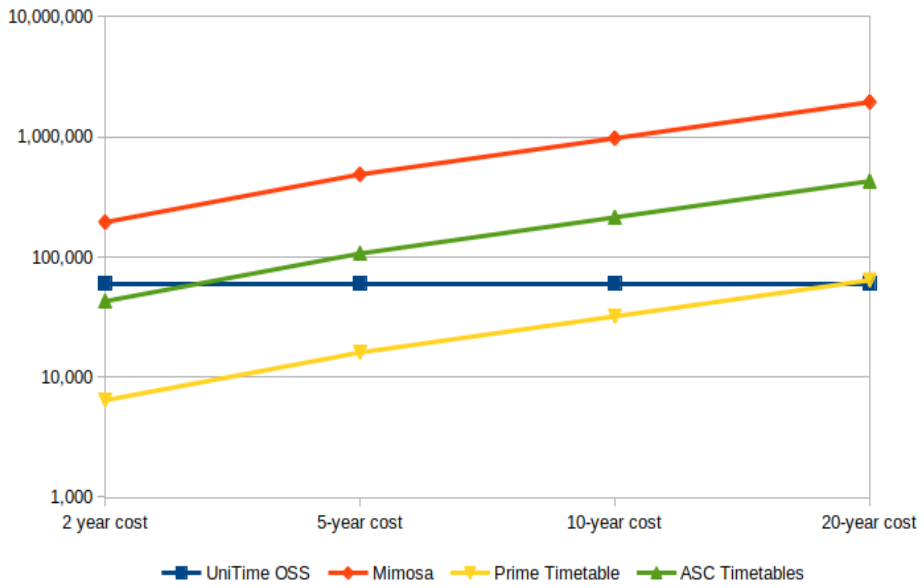

Figure 1 Growth of Cost for UniTime vs COTS (2-20 years) on a Log Scale.

6 Conclusion

The use of OSS has gained traction over the years because of its technical and cost saving measures. When comparing our proposed approach with state of the art timetable software it clear that using an in-house approach with OSS is more cost effective. The successful implementation of OSS in other universities as highlighted by our literature review is an indication that OSS can provide an alternative for solving the university timetable problem in institutions that are trying to cut costs.

References

Arogundade, O. T., Akinwale, A. T., & Aweda, O. M. (2010). A genetic algorithm approach for a real-world university examination timetabling problem. *International Journal of Computer Applications, 12*, 975-8887.

Badampudi, D., Wohlin, C., & Petersen, K. (2016). Software component decision-making: In-house, OSS, COTS or outsourcing-A systematic literature review. *Journal of Systems and Software, 121*, 105-124.

Chmait, N., & Challita, K. (2013). Using simulated annealing and ant-colony optimization algorithms to solve the scheduling problem. *Computer Science and Information Technology, 1*, 208-224.

Čupić, M., & Franović, T. (2011). Scheduling problems at a university: a real-world example. *International Journal of Knowledge and Learning, 7*, 51-69.

Flessner-Filzen, J. (2015). *Improving MITś scheduling system*. Retrieved from http://news.mit.edu/2015/unitime-improving-mit-scheduling-system-0122

Glynn, E., Fitzgerald, B., & Exton, C. (2005). Commercial adoption of open source software: an empirical study. *2005 International Symposium on Empirical Software Engineering, 2005.*, (pp. 10--pp).

Kahar, M. N., & Kendall, G. (2010). The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution. *European journal of operational research, 207*, 557-565.

Keim, Y., Bhardwaj, M., Saroop, S., & Tandon, A. (2014). Software cost estimation models and techniques: A survey. *International Journal of Engineering Research and Technology, 3*, 1763-1768.

Kemerer, C. F. (1987). An empirical validation of software cost estimation models. *Communications of the ACM, 30*, 416-429.

Lukáš, M. (2019). Course timetabling at Masaryk University in the UniTime system.

Mansor, Z., Arbain, J., & Hassan, M. A. (2013). Implementation of fet application in generating a university course and examination timetabling. *Second International Conference on Onformation Technology and Business Application. Anais.*

Moulla, D. K. (2013). COCOMO model for software based on Open Source: Application to the adaptation of TRIADE to the university system.

Muhamad, W. Z., Adnan, F. A., Yahya, Z. R., Junoh, A. K., & Zakaria, M. H. (2018). Solving university course timetabling problems using FET software. *AIP Conference Proceedings, 2013*, p. 020052.

Riehle, D. (2007). The economic motivation of open source software: Stakeholder perspectives. *Computer, 40*, 25-32.

Villano, M. (2006). Open source vision. *Campus Technology, 19*, 26.

Williams van Rooij, S. (2007). Perceptions of open source versus commercial software: Is higher education still on the fence? *Journal of Research on Technology in Education, 39*, 433-453.