Model-Based Dynamic Software Project Scheduling

Natasha Nigar
School of Computer Science
University of Birmingham, United Kingdom
n.nigar@cs.bham.ac.uk

ABSTRACT

Software project scheduling, under uncertain and dynamic environments, is one of the most important challenges in software engineering. Recent studies addressed this challenge in both static and dynamic scenarios for small and medium size software projects. The increasing trend of cloud based software solutions *(large scale software projects)* needs agility not only for sustainable maintenance but also for in time and within budget completion. Therefore, this paper formulates software project scheduling problem (SPSP) as an optimization problem under uncertainties and dynamics for hybrid scRUmP software model. In this regard, a mathematical model is constructed with five objectives as project duration, task fragmentation, robustness, cost, and stability.

CCS CONCEPTS

• Software and its engineering → Search-based software engineering;

KEYWORDS

Dynamic software project scheduling; search-based optimization; mathematical model.

ACM Reference Format

Natasha Nigar. 2017. Model-Based Dynamic Software Project Scheduling. In *Proceedings of 11th Joint Meeting of the European Software Engineering conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 2017 (ESEC/FSE'17).*
https://doi.org/10.1145/3106237.3119879

1. INTRODUCTION

In the last two decades, an exponential growth of software companies has resulted in a highly competitive environment where success heavily depends on the faster but within budget completion of software projects. This refers to a scheduling problem (SP) where decisions are made about who does what during project life cycle [1]. The SP becomes more complex and challenging for medium to large scale projects. The key associated challenges are schedule evaluation, handling chaotic behavior, reducing computational burdens, accommodating unpredictability and handling unpredictable cases [2]. In this PhD, we deal with the large-scale software project scheduling problem for a hybrid software engineering model 'scRUmP' [3], which is focused on agility and quality, in a dynamic environment.

Juran [4] defines software quality as an extent to which we comply with the customer requirements and a hybrid software model 'scRUmP' ensures quality by customer involvement. In a typical project scheduling problem (PSP), key focus is on optimally allocating people (employees) to activities (tasks) using automated approaches [1,5,6,7]. In this regard, both traditional and optimized strategies have been used. The most promising traditional methods are program evaluation and review technique (PERT) [8], critical path method (CPM) [9] whereas resource-constrained project scheduling problem (RCPSP) model [10] is classical optimization example. With growing software needs, these are becoming obsolete; hence, there is a rise of interest in new methods to comply with dynamic environment.

Previous works on PSP considers that no disruption occurs during the project life to interrupt the task execution [11] whereas in reality software projects have to go through many uncertain and dynamic changes e.g. new high priority task arrival, new employee addition, employee leave, requirements change during whole project development life cycle. For such scenarios, the optimal schedule may risk within budget and in time completion. The software engineering [13] is defined as an application of systematic, disciplined, quantifiable approach to the development, operations and maintenance of software [12]. Thus, we argue that dynamic software project scheduling approaches that take into account uncertainties and dynamic events to ensure not only within budget and time completion but also to ensure quality and agility in the large-scale software projects must be developed.

Dynamic software project scheduling problem (DSPSP) has attracted many researchers' attraction. In the literature, for static scheduling environment, cost and duration minimization are treated as optimization objectives. While dynamic environment may re-generate a new schedule based on changing software needs. For example, an employee may join the software company after project starts, so allocating him/her on project by

ESEC/FSE'17, September 4–8, 2017, Paderborn, Germany

ACM ISBN 978-1-4503-5105-8/17/09…$15.00
https://doi.org/10.1145/3106237.3119879

replacing an employee with less experience might result in both project cost and time reduction. The tasks are assigned with priorities, so if multiple tasks with same priority arrive at the same time, these must be executed while ensuring project efficiency, which is not trivial. This leads to preemption in our proposed model. Moreover, given uncertainty in DSPSP good schedule should be robust against data variation. Therefore, in the proposed mathematical model, duration, cost, robustness, stability and task fragmentation are modeled as objectives whereas dynamic features (section 3) will be implemented in the algorithm for targeted hybrid software engineering model 'scRUmP'. This is chosen for its inherent ability toward agility and quality, key success factors in a large software project. In this PhD, We are to make the following contributions:

1. Model Validation

Model validation will be done by two methods. First is Qualitative approach. In this approach, surveys/interviews will be done to project managers in different software companies. Another one is Quantitative approach. It includes the collection of data sets from different software houses to validate our model.

2. Novel Algorithm development

A novel algorithm will be developed to deal with the five objectives of our model. There is a non-existing solution for the proposed scheduling problem.

This paper is organized as follows. The section 2 presents related work whereas sections 3 presents mathematical model of DSPSP for 'scRUmP' software hybrid model. This paper concludes in section 4 with future work.

2. RELATED WORK

The literature review is divided into static and dynamic software project scheduling problems.

2.1 Static Software Project Scheduling (SSPS)

During last decade, search-based approaches became a promising way for software project scheduling problem. Alba and Chicano [1] presented a basic model to tackle SPSP and solved many different software project scenarios. Minku et al. [5] proposed an improved algorithm based on the model presented in [1] whereas Xiuli et al. [21] solved it in a better way. Xiao et al. [19] based on mathematical model in [1] assigned tasks to the humans.

Chang et al. [15] proposed software project management net (SPMnet) model to find near optimal solutions. An improved version of Chang et al. [17] work, introduced a 3D matrix representation, specifying the work load assignment of each employee for each task on each time period. Hanne and Nickel [16] have proposed multi-objective evolutionary algorithm for scheduling and inspection planning. Xiao et al. [18] allocated human resources to multiple projects under resource requirements, constraints. Tavana et al. [20] deal with three conflicting objectives. Chen and Zhang [22] handled both task scheduling and human resource allocation where employees leave/return were regarded as events. Studies in [17,22] treat some dynamics but have been designed for static environment.

2.2 Dynamic Software Project Scheduling (DSPS)

Dynamic scheduling has been of researchers' interest. Gueorguiev et al. [23] introduced a search based approach to software project robustness under uncertainty using proactive scheduling method with a multi-objective evolutionary algorithm (MOEA). Ge [24] proposed a rescheduling method with GAs under uncertainty. In this approach both efficiency and stability were handled as a single objective function by weighted sums. Xiao et al. [25] have proposed resource management under disruption prone environment. But there exist some limitations for their work such as continuous rescheduling. Antoniol et al. [26] developed the scheduling method which combines GA and queue simulation. Although this method realizes scheduling under some uncertainties, issues such as stability are not considered. Chicano et al. [27] present a novel formulation for SPSP which considers productivity of the employees at performing different tasks. Their wok also provides robust solution under analysis of the inaccuracies in task-cost estimations. Shen et al. [11] proposed a mathematical model with four objectives. Still more dynamics can be added in their work.

The above literature highlights that static approaches do not consider uncertainty and unpredictability which has strong impact in large scale software projects efficiency, particularly in this competitive environment. Contrary, the dynamic approaches consider rescheduling. However, none of the existing approaches consider continuous rescheduling as well as duration, cost, stability, task fragmentation and robustness as competing objective functions. Moreover, the dynamic features inclusion is also limited to employees leaving and returning as events. Besides there are features, identified in [11] as addition of new tasks, removal of tasks, change in task precedence and addition of new employee, which can make rescheduling problem highly challenging for large software projects.

3. MATHEMAICAL MODEL

3.1 Employees' Attributes

Let's suppose that there are N employees for a software project from {1,2,.....,N}. Each employee has associated attributes. An employee set is $\{e^{id}, e^{skills}, e^{exp}, e^{basic\_salary}, e^{overwork\_salary}, e^{perhour\_salary}, e^{normalhours}, e^{maxhours}\}$. Table 1 presents the detailed description of employee's attributes.

Table 1: Employees' Attributes

Attribute	Description
e^{id}	Each employee has a specific id.
e^{skills}	Employee's skills in which he is proficient.
e^{exp}	An employee has associated experience between [0,1]. 1 means that employee is an experienced employee, 0 means employee is fresh; and don't have much knowledge.
e^{basic_salary}	Each employee has a basic salary per month.
$e^{overwork\_salary}$	Monthly salary of an employee for overtime.
nhours	Normal working hours of an employee.
maxhours	Monthly maximum allowed working hours.
$e^{perhour\_salary}$	Per hour salary of an employee.

3.2 Tasks' Attributes

Let's suppose, there are M tasks for a software project from {1,2,…..,M}. Each task has associated attributes. A task set is {T^{id}, T^{skills}, T^{status}, T^{premp}, T^{prio}, $T^{man\text{-}hours}$}. When 'K' new tasks are added to the project, total tasks in the project are {1,2,…..M, M+1, M+2,……,M+K}. Table 2 is the detailed description of task attributes.

Table 2: Tasks' Attributes

Attribute	Description
T^{id}	Each task has a specific id.
T^{skills}	Skills required to accomplish the task.
T^{status}	Task has associated status. 1 means task is active. 0 means task has been cancelled.
T^{premp}	Each task has a preemption associated with it. 0 means task is high priority task and can't be interrupted. 1 means task can be preempted.
TaskList	Each task belongs to a task list. TaskList is a list of tasks with priorities of execution. {VeryHigh, High, Medium}
T^{prio}	Each task has a priority as mentioned in task list.
$T^{man\text{-}hours}$	Number of man-hours required to complete task.

3.3 Objectives

There are five objectives to be optimized namely project duration, cost, task fragmentation, robustness and stability. There are four phases 'p' and 'iter' number of iterations depending on project size. If teams have no or less experience as compared to specified percentage then a penalty factor is added in project duration and cost. This may include time and cost for team training etc.

$$\mathbf{F} = [f_1, f_2, f_3, f_4, f_5]$$

3.3.1 Duration. Project duration is the maximum time required to complete the project. T^{start_time}, T^{end_time} is the start and end time of a task respectively. This e^{exp} attribute differentiates our hybrid model with other models presented in literature.

$$\min f_1(t) = \sum_{1=1}^{p} duration_p \tag{1}$$

if e^{exp} =1 then duration is

$$= \sum_{1}^{p} \left(\sum_{j=1}^{iter} \left(max\left(T^{end\_time}\right) - min\left(T^{start\_time}\right) \right) \right) \tag{2}$$

elseif e^{exp} =0 then duration is

$$= \sum_{1}^{p} \left(\sum_{j=1}^{iter} \left(max\left(T^{end\_time}\right) - min\left(T^{start\_time}\right) \right) \right) + \sum Penalty_j \tag{3}$$

3.3.2 Cost. Project cost depends on number of iterations.

$$\min f_2(t) = \sum_{1=1}^{p} cost_p \tag{4}$$

if maxhours ≤ nhours

$$\text{cost} = \sum_{j=1}^{n} e_j{}^{perhour\_salary} * nhours + e_j{}^{basic\_salary} \tag{5}$$

elseif maxhours > nhours

$$\text{cost} = \sum_{j=1}^{n} e_j{}^{perhour\_salary} * nhours + e_j{}^{basic\_salary} + (maxhours - nhours) * e_j{}^{over\_salary} \tag{6}$$

3.3.3 Task Fragmentation. f_3(t) represents task fragmentation performance. It measures dependency of one task on another. Objective is to avoid the task schedule fragmentation. If more tasks are dependent on a task and due to some reason that task is delayed so that other tasks are not affected and whole completion time is not delayed. This objective differentiates our hybrid model with other models presented in literature.

$$\min f_3(t) = task\ fragmentation = \left(\alpha * DS(t) + \beta * UDS(t)\right) \tag{7}$$

where α and β are control parameters for direct successors (DS) and undirect successors (UDS) respectively.

3.3.4 Robustness. f_4(t) represents robustness, the schedule's ability to cope with small increases in the time duration of some tasks. It is defined as task's slack time by which a task can be delayed without delaying the whole project. In Equation (8) 'S' represents the slack time of a task. 'NSucc' is number of immediate successors of a task. So, robustness is maximized for our problem.

$$\max\ f_4(t) = robustness = \sum_{i=1}^{p} \left(\sum_{j=1}^{iter} \sum_{k=1}^{m} S_k * NSucc_k \right) \tag{8}$$

3.3.5 Stability. f_5(t) objective measures deviation between new and original schedules. t′ represents new schedule time and t represents old schedule time. A penalty is attached for preventing employees for being shuffled around too much.

$$\max\ f_5(t) = stability = \sum_{i=1}^{m} \sum_{j=1}^{n} |t'_{i,j} - t_{i,j}| + Penalty_{i,j} \tag{9}$$

3.4 Constraints

Listed below are soft constraints for our DSPSP.

1. Task Due-date
   Task should not delay from its due date, it is defined:
   $$\text{Taskdelay} = \varphi_t\,(d_n - DD_n)$$
   where
   d_n, task finish time
   DD_n , Due date of task n
   $\varphi_t=1$ *if* $(d_n - DD_n) >0$; = 0 otherwise.

2. Task headcount
   There is limit for number of employee to work on a task. Each task has maximum number of headcount of employees.
   $$\forall\, T_j,\ T_j^{\,no\_of\_emp}(t') \le T_j^{\,maxheadcount}$$

Here $T_j^{\,maxheadcount}$ is already defined for our problem.

4. CONCLUSIONS

This work introduces a novel idea for dynamic software project scheduling in hybrid software model 'scRUmP' for medium to large scale projects in agile way. We develop a mathematical model to summarize five objectives. The project constraints and dynamic features have been identified. Future work will be the design of algorithms to deal with this multi-objective optimization problem. A new search algorithm will be either developed or an existing evolutionary algorithm such as ant colony optimization (ACO) or particle swarm optimization (PSO) will be adapted. Currently there is an issue for our model validation. We will validate our proposed model in future through surveys/interview to project managers and collection of data sets from different software companies.

ACKNOWLEDGMENTS

I would like to thank Professor Xin Yao and Dr. Miqing Li for their guidance on this work. This work was supported by the FDP Fund for Young Scholars under Grant no. Estab/D-412(331)/2014/2278.

REFERENCES

[1] E. Alba and J. F. Chicano. Software project management with gas. *Inf. Sci.*, vol. 177, no. 11, pp. 2380–2401, 2007.

[2] Parunak and H. Van Dyke. Characterizing the manufacturing scheduling problem. *Journal of manufacturing systems*, vol.10, no. 3, pp. 241-259, 1991.

[3] E. del Nuevo , M. Piattini, and F. J. Pino. Scrum-based methodology for distributed software development. In *Proceedings of 6th Internaional Conference on Global Software Engineering*, Aug. 2011, pp. 66-74.

[4] J.M. Juran, *Juran's Quality Control Handbook*, McGraw-Hill, 1988.

[5] L. L. Minku, D. Sudholt, and X. Yao. Improved evolutionary algorithm design for the project scheduling problem based on runtime analysis. *IEEE Trans. Softw. Eng.*, vol. 40, no. 1, pp. 83–102, Jan. 2014.

[6] C. K. Chang, M. J. Christensen, and T. Zhang. Genetic algorithms for project management. *Ann. Softw. Eng.*, vol. 11, pp. 107–139, 2001.

[7] F. Luna, D. González-Álvarez, F. Chicano, and M.A.Vega-Rodríguez. The software project scheduling problem: A scalability analysis of multi-objective metaheuristics. *Appl. Soft Comput.*, vol. 15, pp. 136–148, 2014.

[8] J. D. Wiest, and F. K. Levy, *A Management Guide to PERT/CPM: with GERT/PDM/CPM and Other Networks.*, NJ, USA: Prentice-Hall. 1977.

[9] D. Golenko-Ginsburg and A. Ganik. Stochastic network project scheduling with non-consumable limited resources. *Int'l. J. Production Econ.*, vol. 48, pp. 29–37, 1997.

[10] W. Herroleon , B. D. Reyck, and E. Demeulemeester. Resource-constrained project scheduling: A survey of recent developments. *Comput. Oper. Res.*, vol. 25, no. 4, pp. 279–302, 1998.

[11] X. Shen, L. L. Minku, R. Bhasoon, and X. Yao. Dynamic software project scheduling through a proactive-rescheduling method. *IEEE Trans. Softw. Eng.*, vol. 42, no. 7, pp. 658-686, July. 2016.

[12] R. Pressman, *Software Engineering: A Practioner´ıs Approach*, 6th ed.New York, NY, USA: McGraw Hill, 2005.

[13] M. Harman. The Current State and Future of Search Based Software Engineering. In *Proc of Future of Software Eng*, May. 2007, pp. 342-357.

[14] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing.* vol. 53, Heidelberg: springer, 2003.

[15] C. K. Chang, M. J. Christensen, C. Chao, and T. T. Nguyen. Software Project Management Net: A New Methodolog on Software Management. In *Proceedings of 22nd Annual Int'l Conf. on Computer Soft and App.*, 1998.

[16] T. Hanne and S. Nickel. A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects. *Europen Journal of Operational Research*, vol. 167, no.3, pp. 663-678, 2005.

[17] C. K. Chang, H. Jiang, Y. Di, D. Zhu, and Y. Ge. Time-line based model for software project scheduling with genetic algorithms. *Inf. Softw. Technol.*, vol. 50, pp. 1142-1154, 2008.

[18] J. Xiao, Q.Wang, M. Li, Q. Yang, L. Xie, and D. Liu. Value-based multiple software projects scheduling with genetic algorithm. In *Proceedings of Internaional Conference on Software Process*, May. 2009, pp. 50-62.

[19] J. Xiao, X. T. AO, and Y. Tang. Solving software project scheduling problems with ant colony optimization. *Computers & Operations Research*, vol. 40, no. 1, pp. 33-46, 2013.

[20] M. Tavana, A. R. Abtahi, and K. Khalili-Damghani. A new multi-objective multi-mode model for solving preemptive time–cost–quality trade-off project scheduling. *Expert Systems with App*, vol. 41, pp. 1830-1846, 2014.

[21] X. Wu, P. Consoli, L. Minku, G. Ochoa, and X. Yao. An Evolutionary Hyper-heuristic for the Software Project Scheduling Problem. In *Proceedings of Int'l Conf. on Parallel Problem Solving from Nature*, 2016, pp. 37-47,

[22] W. N. Chen and J. Zhang. Ant colony optimization for software project scheduling and staffing with an event-based scheduler. *IEEE Trans. Softw. Eng.*, vol. 39, no. 1, pp. 1-17, Jan 2013.

[23] S. Gueorguiev, M. Harman, and G. Antoniol. Software project planning for robustness and completion time in the presence of uncertainty using multi-objective search based software engineering. In *Proceedings of 11th Annual Conference on Geneic Evol. Compu.* 2009, pp. 1673–1680.

[24] Y. Ge. Software Project Rescheduling with Genetic Algorithms. In *Proceedings of Internaional Conference on Artificial Intelligence and Computational Intelligence*, Nov. 2009, pp. 439-443.

[25] J. Xiao, L. J. Osterweil, Q. Wang, and M. Li. Dynamic resource scheduling in disruption-prone software development environments. In *Proceedings of Internaional Conference on Fundamental Approaches to Software Engineering, March.* 2010, pp. 107–122, doi: 10.1007/978-3-642-12029-9_8.

[26] G. Antoniol, M. Di Penta, and M. Harman. A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty. In *Proc. of Int'l Symp. Software Metrics*, 2004, pp. 172–183.

[27] F. Chicano, A. Cervantes, F. Luna, and G. Recio. A novel multiobjective formulation of the robust software project scheduling problem. *Applications of Evolutionary Computation.* New York,USA: Springer, 2012, pp. 497–507.