Research Journal of Applied Sciences, Engineering and Technology 12(3): 347-354, 2016
DOI: 10.19026/rjaset.12.2342
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2016 Maxwell Scientific Publication Corp.
Submitted: August 24, 2015 Accepted: September 14, 2015 Published: February 05, 2016

Research Article

Requirement Prioritization and Scheduling in Software Release Planning Using Hybrid Enriched Genetic Revamped Integer Linear Programming Model

[1]Sandhia Valsala and [2]Dr. Anil R. Nair
[1]Karpagam University, Coimbatore,
[2]Scientist Department, Bangalore, India

Abstract: The main objective of this study is to scheduling the prioritized requirements that make the software release in a better way. Software release is a single or a group of change in an already evolved software product that can result in another new product. Therefore, a good planning is essential and a bad plan can always lead to irrelevant features (requirements) being included in the release which in turn can affect the release time of the software. In order to overcome this delay, two things have to be considered such as requirements prioritization and scheduling. Prioritization of requirements means that the significant requirements are released in priority. Second is to schedule these prioritized requirements so as to release the new version on time. If we just do requirement prioritization without making an appropriate time plan, there is a high chance that the project may exceed the release schedule and this probability will grow as the number of dependencies increases. So we have to perform requirement prioritization and scheduling as one model that can minimize the project duration. So the paper consolidates both the processes of software release, prioritization and scheduling, called as Hybrid EGRILP model in order to maximize the revenue and to minimize the project duration. The requirements prioritization is performed using the Enriched Genetic process where the premature convergence problem is overcome and the Revamped Integer Linear Programming (RILP) is introduced with the enriched genetic process. This combination of methods maximizes the profit of the software and minimizes the release time of the software.

Keywords: Modified genetic algorithm, modified heuristic Integer Linear Programming (ILP) model, premature convergence problem, requirements prioritization, requirements scheduling, software release planning

INTRODUCTION

A software release is a collection of new or changed features that can be included in an updated or new version of a software product. At the time of software release planning, the features to be involved in a software release are stable in a way that the budget, technical, risks and resource constraints are met (Rahman and Rokonuzzaman, 2014). Software development is defined as a sequence of actions where the requirements of the users are converted into the final software product. These activities includes converting the user requirements into a model (prototype), progressing the model into real time development (software) and sometimes also includes the maintenance of the delivered software product.

The software release planning has two steps, requirement management and software planning. In requirement management process, the requirements are modified; new requirements are additionally added while software planning phase deals with the way of reaching the goal, processing the risk factors, satisfying the constraints, delivering the final product that promises customer and user satisfaction (Meenakahi, 2014). The requirement selection process should be completed before adding the requirements to the software product. Each considered requirement will not have the same priority and a decision to select the most appropriate requirement is the most vital task of software requirement prioritization.

The next significant process in software development is scheduling these prioritized requirements. Since the selected requirements are having dependencies with each other, scheduling these requirements may have a restriction with time constrained metric (Sandhia and Anil, 2014). Therefore, it is essential to arrange (order) the optimal requirements for the reason of determining the requirements that have to be included in the next version of software release and also it is essential to determine an appropriate time plan to release the software.

An appropriate Software Release Planning includes both requirement prioritization and scheduling. Most of the existing release planning models focuses only on requirement prioritization but our proposed model takes into consideration both these aspects (i.e.,) requirement prioritization and scheduling. Prioritization of

Corresponding Author: Sandhia Valsala, Karpagam University, Coimbatore, India

requirements means ordering the significant requirements according to their priority value. Scheduling these prioritized requirements means developing a project plan so that the new version of the software products released on time. If we just do requirement prioritization without making an appropriate time plan, there is a high chance that the project may exceed the release schedule and this probability will grow as the number of dependencies increases.

Thus, in this study consolidates requirements prioritizing and scheduling in a software release so as to maximize the revenue and minimize the project span. The requirements prioritization is performed using the Enriched Genetic Algorithm (EGA) where the premature convergence problem is overcome and the an heuristic Revamped Integer Linear Programming (RILP) model is introduced for the purpose of scheduling these prioritized requirements that makes the software release in a better way.

LITERATURE REVIEW

In Chen *et al*. (2010) introduces two integer linear programing models that integrate time scheduling to software release planning. First model proposes two separates ILP's, one ILP to perform requirement prioritization and another ILP to perform requirement scheduling so as to minimize the project duration. The second model integrates scheduling into requirement selection process (Li *et al*., 2007). This model not only maximizes the revenue, but also promises an on time project schedule and project delivery. Additionally, the author presents Scrum Methodology that can simplify the dynamic adaptation for under or overestimation of processing time or revenues. The simulation results show that the requirement dependency is closely linked with requirement selection and scheduling process.

The requirement engineering process involves requirement prioritization as a major step. It also helps to make essential decisions about requirements selection. The requirement prioritization process aims to define those candidate requirements involved in software development process that should be included in a certain release. For this purpose various methods are utilized. These methods use various approaches and also considers various factors for prioritization like benefit, value, risk, cost, etc., Thus, in Iqbal *et al*. (2009) author describes an evaluation technique utilized for requirement prioritization.

In Arup *et al*. (2011) proposes an approach for test case prioritization utilizing a simple mathematical prioritization method. This method has identified a number of generic parameters under database, GUI, Networking and has taken into consideration a number of projects under these domains. From these it uses experts view to classify the level of user requirements regarding the parameters. At the first instance, on a six scale basis, the information for all the tables is joined to create a Project Specific Base Table (PSNT). Whenever a new project comes under the same category, the corresponding priority levels are assigned.

Software release planning, requirements catalogues are often not homogeneous and complete. Current release planning method, assume such details of commitments and thus, in Samuel and Susanne (2012), the author proposes a method on how to perform software release planning efficiently. At the same time reducing the release planning time and increasing the decision making flexibility is also analyzed. The selected features like REQUIRES and OR relationship between requirements are captured. The selected requirements structure can be utilized to support abstraction and to hide incompleteness. Additionally, this study describes the methods of decision-making, trust with an industrial case and effort.

Software release Management is a significant vital technology for the distributing the product or project to the customer. The success factor of a software product is based on how gracefully the project is released to the customer. The process of planning, testing, building, deploying software and hardware, storage of software and version control are all coming under release management process. There are a number of methods to estimate release planning and software development. But there are no such methods that gather all the capabilities and resources in one shot. Thus, in Prabhat and Ashish (2012) proposes an intelligent scheduling method to estimate the software release by feeding as inputs the starting requirement, capabilities of the resources and availability. These data are considered as the training set. Finally, the requirements are analyzed for the process of decision making.

Each requirement is having its own importance based on its priority and this priority changes over time and over projects. So the arranging the requirements in order (i.e.,) requirements prioritization is an important task of software release. The result of prioritization is to implement these selected requirements. Many methods are there to do this prioritization with their own advantages and disadvantages. The author suggests a clear cut method to solve this prioritization process and examines various methods used previously. From this investigation, a new prioritization framework is introduced which overcomes the drawback of the existing methods.

MATERIALS AND METHODS

Hybrid enriched genetic revamped integer linear programming model:

The Enriched Genetic Algorithm (EGA): In software release process, the software requirements have to be prioritized initially and then the prioritized requirements are scheduled in order to make the

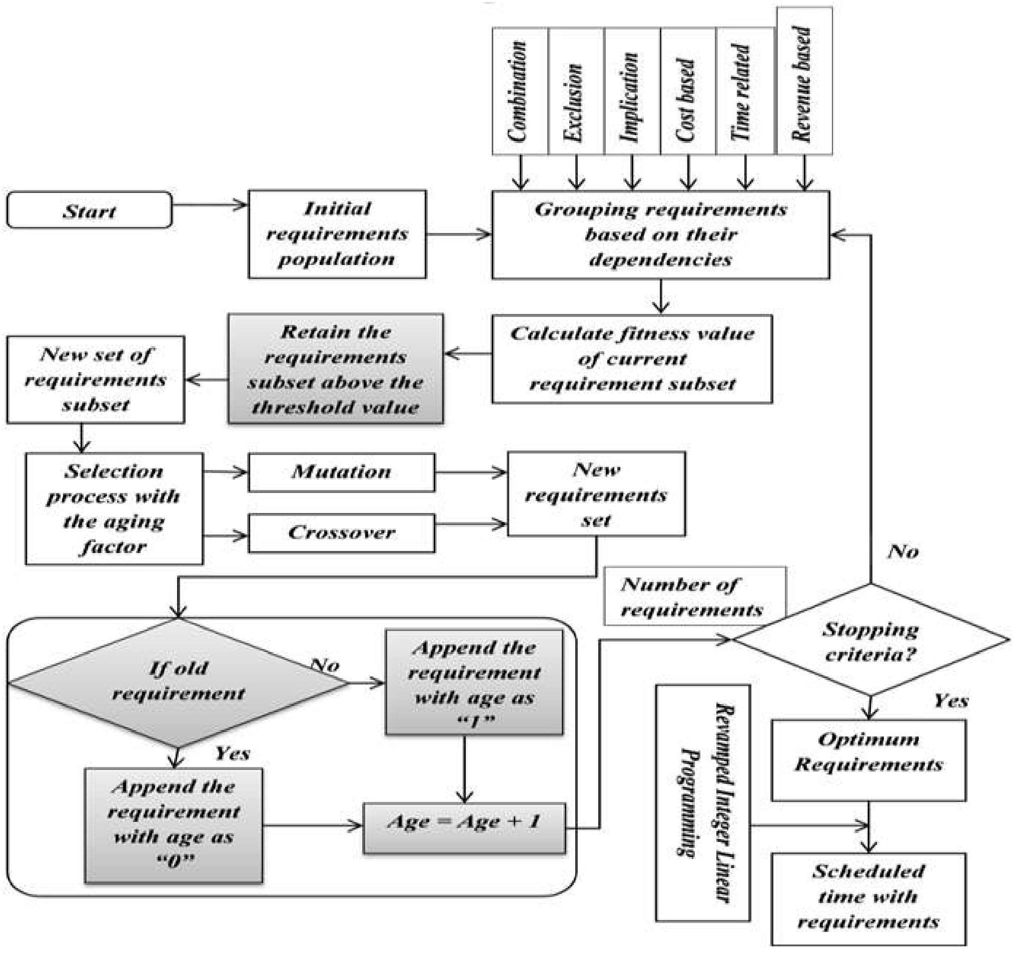
Combination
Exclusion
Implication
Cost based
Time related
Revenue based
Start
Initial requirements population
Grouping requirements based on their dependencies
Calculate fitness value of current requirement subset
Retain the requirements subset above the threshold value
New set of requirements subset
Selection process with the aging factor
Mutation
Crossover
New requirements set
If old requirement
No
Yes
Append the requirement with age as "1"
Append the requirement with age as "0"
Age = Age + 1
Number of requirements
Stopping criteria?
No
Yes
Optimum Requirements
Revamped Integer Linear Programming
Scheduled time with requirements

Fig. 1: Proposed software release planning framework

software release process successful. So at first, the Enriched Genetic Algorithm (EGA) is proposed in order to prioritize the requirements and some additional parameters are considered for the purpose of making the prioritization process more efficient in terms of computation, robustness and reliability. The proposed framework is shown in Fig. 1.

Initially the stakeholders are involved in a discussion session and they gather "m" number of requirements that are needed to be developed for a software product. Mostly these requirements are interdependent with each other and some requirements themselves are singular without any dependencies. Some requirements will have dependencies over other requirements (Aasem *et al*., 2010). Henceinorder to do prioritization of these requirements, dependencies are fixed between them based on six dependency factors (Ma and Krings, 2008). The dependency factors are combination, exclusion, implication, cost-based, revenue based and the time based dependencies.

When any two requirements are dependent on each other and one cannot be implemented without another, then the dependency is named as combination dependency. Taken two requirements in which software development process requires neither of them or either of them or do not require bot, this dependency is termed as exclusion. The implication dependency needs a requirement that support another requirement to function (Shinto and Sushama, 2013), but, not in vice versa. It gives importance to the logical connection between the requirements more than the precedence relation. Next is cost based and revenue based dependencies. When the cost and revenue of a requirement affects another requirement, then these dependencies exists.

Due to the time constraint (time dependency), it is very difficult to process all the requirements at the time of the prioritization process. Hence, while considering the prioritization (i.e., requirement selection), In this proposed work the process requirements are changed based on the time dependency rather than the constraint based. The constraints played an important role in finding both the relative cost-benefit trade-offs among techniques and cost effectiveness of prioritization techniques. As the project size is very huge means, the processing system consumes huge amount of time. This issue of software systems can propose by utilizing a good requirement prioritization technique. A

Table 1: Matrix formation to calculate the requirement quality value and stake holders priority

Requirements set based on dependencies	Req set 1	Req set 2	…	Req set k
Req set 1	$\sum Req\ set\ 1 / \sum Req\ set\ 1$	$\sum Req\ set\ 2 / \sum Req\ set\ 1$	…	$\sum Req\ set\ k / \sum Req\ set\ 1$
Req set 2	$\sum Req\ set\ 1 / \sum Req\ set\ 2$	$\sum Req\ set\ 2 / \sum Req\ set\ 2$	…	$\sum Req\ set\ k / \sum Req\ set\ 2$
⋮	⋮	⋮	⋮	⋮
Req set k	$\sum Req\ set\ 1 / \sum Req\ set\ k$	$\sum Req\ set\ 2 / \sum Req\ set\ k$	…	$\sum Req\ set\ k / \sum Req\ set\ k$

requirement prioritization technique schedules the execution with the EGA Aging factor, thus the higher priority processed before lower priority.

The fitness value for each requirement subsets is calculated by using the following process.

Fitness value calculation: Since the fitness value calculation is based on the stake holders priority level and the requirements quality value, both the measures are calculated by a matrix where the rows and the columns are $requirements\ X\ requirements$. The input of the stake holder's priority level matrix is stake holder's priority and the input to the requirements quality value is requirement quality category. The matrix formulation for the "k" number of requirements subset is shown in the Table 1.

The computation of effect of a grouped requirement subset is as:

$$Effect = \sum_{i=1}^{k} compeer * weight \quad (1)$$

where, the requirement's effects are calculated by multiplying the weight of the requirements and the compeer value of the requirements.

The compeer value is resoluted by comparing the grouped requirement subset with original requirement set. If they both are matched, then the compeer value defined as "1", else it is defined as "0".

Then the delta value is computed by taking the modulo of difference between the effect and the stakeholders priority value and is given as:

$$\Delta = \left| Effect - \sum \begin{matrix} stakeholders\ priority\ level\ based\ on \\ the\ requirement\ for\ a\ requirement \\ subset \end{matrix} \right| \quad (2)$$

The requirement quality is characterized as rating and the penalty value computed as follows:

$$penality = \frac{\Delta * \sum rating}{100} \quad (3)$$

Since the summation of the penalty and the fitness value of a requirement subset equals "1", the fitness value can be written as:

$$fitness = 1 - penalty \quad (4)$$

This method of removing the requirements may lead to a situation where the selected number of requirements will be higher than the number of requirements in the starting stage. If this issue occurs, then from the list of chosen requirements subsets the one with less fitness will be discarded. Therefore, the computational efficiency is enhanced and the computational complexity is minimized.

EM Algorithm: Since the requirements specified by the stakeholders may be huge in number, it is necessary to determine an efficient way to reduce the number of requirements subsets. Thus, the Expectation-Maximization (EM) algorithm is used for this purpose. The different stakeholders present their requirements in different ways and some of the requirements may have similar outcomes. This process reduces the number of requirements in an efficient way and supports the proposed Enriched Genetic Algorithm (EGA) to be performed with reduced number of requirements subset.

The threshold value is fixed through which all the requirements subsets from the EM algorithm compares their fitness value with the threshold value is as follows:

$$threshold\ value = fitness\ average(1 - fitness\ average) \quad (5)$$

The threshold value using the fitness value is found. Then the each requirement subset compares its fitness value with the threshold value; remove the lowest fitness value requirements subset and preserve the highest fitness value requirements subset for the next process. The higher fitness value requirements subset is reproducing the next generation process. Initially, the crossover operation is performed so as to mate two requirements subset to generate the new requirement subset. Then the mutation process attempts to enhance the requirement subset with dynamic selectiveness that produces the efficient requirement subset.

Aging factor: During the genetic process of crossover and mutation, the searching process of requirements restricts the system to local minima. The process catches initially grouped requirement subsets in its memory compares the newly generated requirement subset with the memory. If both are matched, then the requirement subset age is set as 1 else 0. It is described as:

$If\ requirement\ subset(i)$
$=\ requirement\ subet(i)\ in\ memory$
$then\ requirement\ subset\ age\ = 0$
$else$
$requirement\ subset\ age = 1$

The process of calculating the fitness value continuous until the number of requirements becomes eight which are optimized requirement set and prioritized according to their fitness value. (i.e.,) after the aging factor concept, the requirement subset reveals its requirements and checks the number of requirements if it is equal to eight. If not, the requirements are again taken to the grouping requirement process to get the optimal eight requirements.

Revamped Integer Linear Programming (RILP): The requirement prioritization is the most fundamental process of software release planning where the incoming requirements of a specific software are optimized and processed to fulfill all the stakeholders' requirements (Maglyas and Fricker, 2014; Praveen *et al*., 2013). The process of requirements selection is done by utilizing Enriched Genetic Algorithm. Hence, the proposed RILP model is used to solve the requirement scheduling problem. The proper ordering of these requirements should reduce the final software release delay. Once the prioritization process of requirement is over, the chosen requirements are required to be scheduled within the (fixed) static time interval. Normally, the scheduling requirements are taking two types of constraints such as the limited available resources and precedence constraints. This RILP model based requirement scheduling process is to overcome the RCPSP problem.

In RILP, two types of metrics are used such as Requirement dependencies and time span representation of the project. The requirements and their dependencies between them are characterized as $RS = \{(R_a, R_{a*}) | R_a \leftarrow R_{a*}\}$. The requirement set having the requirements including R_a and R_{a*}, where the requirement R_{a*} depends on the requirement R_a. In time span representation of a project, let $Time_{max}$ is defined as the time required to complete the whole project is calculated as follows:

$$Time_{max} = \sum_{j=1}^{n} max\left(d_a | J_a \in X(R_j)\right) \quad (6)$$

where, d_a be the developing time, J_a is defined as processing job, R_j is defined as job j belongs to the requirement R. Then the earliest start time of the job "a" is calculated as follows:

$$Est_a = Time(project\ start, J_a) \quad (7)$$

It means that the time between the project start (virtual job) and the real job. After that, the latest start time of the job J_a is computed as:

$$Lst_a = Time(Time_{max}, project\ end) \quad (8)$$

It means that the time between the completion of project time and project end (virtual job). In RILP model, the basic steps of the common Integer Linear Programming (ILP) for the RCPSP problem are considered and in addition some constraints are also added that are used in formulating the ILP (Chen *et al*., 2010). Therefore, this proposed approach is termed as Revamped ILP. This proposed RILP model minimizes the project cost and also minimizes the project time span.

Consider the variable δ_{at} occur in the time interval between early start time and latest start time. This variable occurs for each job J_a and the time t in the following RILP formulation characterizes the possible time for the particular job to start. Each job development is started with the already given deadline d_j and the release date r_j. Also, a decision variable γ_{jt} is introduced and is equal to "1" only if the particular job j processed in time t. In some cases, the jobs have been developed in the specified time or take overtime to develop. So OT_{nt} denotes the volume of resources that are available during the overtime and HOT_{nt} denotes the volume of resources that are hired in overtime. Then formulation of RILP is shown as follows:

$$Minimize: \sum_{t=Es_{END}}^{t=Ls_{END}} t.\delta_{ENDt},$$

$$\sum_{n=1}^{K}\left[\sum_{t\in T^0} c_{nt}^{H} H_{nt} + \sum_{t\in T\setminus T^0}\left(c_{nt}^{O} OT_{nt} + c_{nt}^{HO} HOT_{nt}\right)\right] \quad (9)$$

Subject to:

$$\sum_{t=Est_a}^{t=Lst_a} \delta_{at} = 1, for\ all\ J_a \in X' \quad (10)$$

$$\sum_{t=Est_a}^{t=Lst_a} t.\delta_{at} + dt_a \le \sum_{t=Est_{a*}}^{t=Lst_{a*}} t.\delta_{ta*}, for\ all\ (J_a, J_{a*}) \in JS \quad (11)$$

$$\sum_{J_a\in X(G_i)} \sum_{\mu=\rho(t,a)}^{t} \delta_{at} \le 1, for\ all\ t \in \{0,1,\dots,Time_{max}\}, i \in \{1,\dots,m\} \quad (12)$$

$$\sum_{j=1}^{n} q_{jn}\gamma_{jt} \le OT_{nt} + HOT_{nt}, \forall n, t \in T^0; \quad (13)$$

$$\sum_{t=r_j}^{t=d_j-1} \gamma_{jt} = pt_j, \forall j; \quad (14)$$

$$p_i(1 - F_{j,t}) \ge \sum_{\bar{t}\ge t} \gamma_{j\bar{t}}, \forall t \in T; \quad (15)$$

$$\delta_{at} \in \{0,1\}\ for\ all\ t \in [Est_a, Lst_a], J_a \in X' \quad (16)$$

$$F_{j,t} = 0, \forall t\{r_j, \dots, d_j - p_j\}; \quad (17)$$

$$\gamma_{jt} = 0, \forall t\{r_j, \dots, d_j - 1\}; \quad (18)$$

$$F_{j,t}, \gamma_{jt} \in \{0,1\}, \forall j, t \in T; \quad (19)$$

Table 2: Scheduling results of model1

Req id	Team A (Start day)	Team A (End day)	Team B (Start day)	Team B (End day)	Team C (Start day)	Team C (End day)	Duration in days
12	0	8			0	6	8
34	9	15	0	5	7	10	15
35	16	28			11	17	28
66			6	12	18	23	23
63	29	36	13	18			36
25	37	45			24	32	45
43	46	50	19	26			50
67	51	54					54

Table 3: Scheduling result of model 2 (Hybrid EGRILP)

Req id	Team A (Start day)	Team A (End day)	Team B (Start day)	Team B (End day)	Team C (Start day)	Team C (End day)	Duration in days
67	0	4			0	6	6
34	5	14	0	5	7	10	14
66	15	25	6	13			25
35	26	37			15	27	37
12			14	23	28	32	32
43	38	40			33	36	40
25	41	49			37	44	49
63	50	53					53

$$OT_{nt}, HOT_{nt} \geq 0, \forall n, t \in T \setminus T^0; \qquad (20)$$

The goal of RILP is defined in the statement (9), which reduces the development cost of requirement and project development time. So as to minimize this objective function of the proposed method, the subsequent constraints are taken into consideration. The constraint in (10) denotes that each job starts to process only once, examine when two requirements are executed by two variant teams. If the starting requirement relies on second requirement, then the second process is executed first after this process the first process will be executed second. These processes cannot be executed either in different order or at the same time. These requirements dependencies are presented in the constraint (11). Similarly, a single development team examines only on a single requirement at a time and is explained in (12). This constrain used for reducing the technical interdependencies among modules also reduces the interdependencies among the tasks at particular time. In certain cases, the job development procedure may exceed the given time, in such cases during preprocessing stage; it may use the available volume of resources and also lease the resources. The constraint (13) maintains that the needed amount of resources must not go beyond the amount of available resources. The constraint (14) is a situation for checking the software development time needed for a requirement should be between the release date and deadline. The constraint (15) makes sure that the specific job begins and ends in specific time. Constraint for all the variables is defined as {0, 1} in the equation (16). Finally the constraints (17) and (18) create all the irrelevant variables to zero and the next constraints (19) and (20) are defined as variable domain.

RESULTS AND DISCUSSION

A successful software release planning involves two main process of requirements prioritization (optimal requirement selection) and scheduling these requirements to be released on time. Individually, the Enriched Genetic Algorithm (EGA) is responsible for the selecting the optimized requirements and the Revamped Integer Linear Programming (RILP) schedules these prioritized requirements so as to minimize the project span.

We are comparing the simulations of our proposed two models of prioritization and scheduling i.e., the first model that does requirement prioritization and scheduling using Enriched Genetic algorithm and Revamped ILP algorithm. Our first model does not take into consideration the time dependency factor and our second model the Hybrid EGRILP considers time dependency constraint while selecting and scheduling the requirements of arelease. Our simulations prove the process of software requirement prioritization and scheduling performed using HybridEGRILP model yields a very optimal solution. In our implementation process, the requirement prioritization process begins with 99 requirements given as input to the prioritization process and the generated final eight optimal requirements are given for scheduling.

Table 2 shows the scheduling results of our first model and Table 3 shows the scheduling results of our second model that considers time dependency constraint during selection and scheduling process.

The results shows that the scheduling results of our Hybrid ERILP models yield better results than our first model.

In order to test the relation between varying dependency and its effect on scheduling another set of simulations were carried on these models. The main

Table 4: Varying dependency and scheduling results on model 1

Req set	Dependency ratio	No of dependencies	Project span Max days	Min days	Average days	No of delay (out of 100 runs)	Average revenue
Small set (8 req, 60 days)	10	1	72	54	63	9	1587.95
	20	2	84	54	60	29	1238.95
	30	3	98	54	72	41	1029.55
	40	4	108	54	81	49	889.95

Table 5: Varying dependency and scheduling results on model 2 (Hybrid EGRILP)

Req set	Dependency ratio	No of dependencies	Project span Max days	Min days	Average days	No of delay (out of 100 runs)	Average revenue
Small set (8 req, 60 days)	10	1	66	53	59.5	5	1657.75
	20	2	78	53	65.5	19	1413.45
	30	3	84	53	68.5	30	1221.5
	40	4	102	53	77.5	40	1047.0

aim of this simulation was to check if the models schedule the prioritized requirements always on schedule even when we change the number of dependency between the requirements. The important research question to be answered here was the relationship between the number of dependencies and the probability of the project running out of time.

As the requirements selected for scheduling is 8 in number, the possibility of number of dependencies are 8*7/2 = 28. In the proposed system simulation, the requirements are prioritized bot the models to prepare the project planning. The procedure is repeated 100 times continuously and the minimum, maximum and the average time span of the project is calculated. The Table 4 illustrates the simulation results of model1 and Table 5 illustrates the simulation results of Hybrid EGRILP model.

Our simulations prove the process of software requirement prioritization and scheduling performed using HybridEGRILP model yields a very optimal solution and the proposed software release planning system works better in terms of project span with minimum delay and maximum revenue. These simulations also show that considering the time dependency constraint during scheduling and prioritizations yield better results.

CONCLUSION

In this study, the requirement prioritization process and requirement scheduling in software release planning is described and also proposed is an Enriched Genetic Algorithm (EGA) with Revamped Integer Linear Programming (RILP) which considers time dependency constraint while selecting and scheduling requirements. In EGA the aging factor introduced for eliminating premature convergence problem and the dynamic population improves the computational efficiency and decreases the computational complexity. The Revamped Integer Linear Programming (RILP) model is enriched from the previously utilized Integer Linear Programming (ILP) relies upon the process of scheduling the requirement, not only minimizing the cost of the project, additionally it minimizes the project span by adding additional resourceful constraints with the general ILP formulation. Simulations are performed on two models one that does not take into consideration the time dependency factor and the other model the Hybrid EGRILP model that considers time dependency constraint added to the EGA and RILP algorithm. Our simulations proves that the process of software requirement prioritization and scheduling performed using Hybrid EGRILP model yields a very optimal solution and the proposed software release planning system works better in terms of project span with minimum delay and maximum revenue. These simulations also show that considering the time dependency constraint during scheduling and prioritizations yield better results.

REFERENCES

Aasem, M., M. Ramzan and A. Jaffar, 2010. Analysis and optimization of software requirements prioritization techniques. Proceeding of the IEEE International Conference on Information and Emerging Technologies, pp: 1-6.

Arup, A.A., B. Goutam and P. Namita, 2011. A novel approach for test case prioritization using priority level technique. Int. J. Comput. Sci. Inform. Technol., 2(3): 1054-1060.

Chen, L., M. van den Akker, S. Brinkkemper and G. Diepen, 2010. An integrated approach for requirement selection and scheduling in software release planning. Requir. Eng., 15(4): 375-396.

Iqbal, A., F.M. Khan and S.A. Khan, 2009. A critical analysis of techniques for requirement prioritization and open research issues. Int. J. Rev. Comput., pp: 8-18, E-ISSN: 2076-3336.

Li, C., J.M. van den Akker, S. Brinkkemper and G. Diepen, 2007. Integrated requirement selection and scheduling for the release planning of a software product. In: Sawyer, P., B. Paech and P. Heymans (Eds.), REFSQ, 2007. LNCS 4542, Springer-Verlag, Berlin, Heidelberg, pp: 93-108.

Ma, Z. and A.W. Krings, 2008. Dynamic populations in genetic algorithms. Proceedings of the 23rd Annual ACM Symposium on Applied Computing, pp: 1807-1811.

Maglyas, A. and S.A. Fricker, 2014. The preliminary results from the software product management state-of-practice survey. In: Lassenius, C. and K. Smolander (Eds.), ICSOB, 2014. LNBIP 182, Springer International Publishing, Switzerland, pp: 295-300.

Meenakahi, T., 2014. Reengineering a software process by using unified foundation. Int. J. Invent. Comput. Sci. Eng., 1(3), ISSN: 2348-3431.

Prabhat, G. and O. Ashish, 2012. A resource oriented intelligent scheduling scheme to estimate software release. Int. J. Adv. Res. Comput. Sci. Softw. Eng., 2(9): 149-154.

Praveen, R.S., S. Subrahmanyan and P. Pushkar, 2013. Optimal software release policy approach using test point analysis and module prioritization. MIS Rev., 18(2): 19-50.

Rahman, T. and M. Rokonuzzaman, 2014. A noble methodology for users' work process driven software requirements for smart handheld devices. Int. J. Softw. Eng. Appl., 5(4): 21-38.

Samuel, F. and S. Susanne, 2012. Release planning with feature trees: Industrial case. In: Regnell, B. and D. Damian (Eds.), REFSQ, 2012. LNCS 7195, Springer-Verlag, Berlin, Heidelberg, pp: 288-305.

Sandhia, V. and R. Anil, 2014. Review and analysis of software release planning models. Int. J. Eng. Adv. Technol., 3(5): 1-7.

Shinto, K.G. and C.M. Sushama, 2013. An algorithm for solving integer linear programming problems. Int. J. Res. Eng. Technol., 2(7): 107-112.