S. A. Politsyn, E.V. Politsyna
National Research University — Moscow Aviation Institute, 125993, Volokolamskoe shosse 4, Moscow, A-80, GSP-3,
e-mail: pul_forever@mail.ru, ORCID iD is 0000-0002-0744-6035;
e-mail: kathrin.beaver@mail.ru, ORCID iD is 0000-0002-9313-4766

The Model of the Task Solving Process in Software Engineering

Keywords: project management; resource allocation; time estimation; software engineering process
Development teams are likely to have many difficulties while making more or less accurate estimations on a project: time assumptions, work assessments and required budget. A new approach of solving this problem is described in the article. The approach based on queue theory and a method for using this approach for software engineering process estimations with better accuracy are suggested in this article.

С.А. Полицын Е.В. Полицына
Национальный исследовательский университет «Московский авиационный институт», 125993, Волоколамское шоссе, д. 4, г. Москва, А-80, ГСП-3,
e-mail: pul_forever@mail.ru, ORCID iD is 0000-0002-0744-6035;
e-mail: kathrin.beaver@mail.ru, ORCID iD is 0000-0002-9313-4766

ПЛАНИРОВАНИЕ ПРОЕКТОВ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ПРИМЕНЕНИЕМ МАТЕМАТИЧЕСКОЙ МОДЕЛИ

Ключевые слова: управление проектами, расчет временных затрат, срок окончания проекта, теория массового обслуживания, вероятность
Задача оценки времени и ресурсов для программных проектов обычно решается сильно приближенно. В статье предлагается математическая модель, предназначенная для решения задачи планирования в проектах разработки программного обеспечения. Представлены методика прогнозирования хода процесса разработки программного обеспечения и
метод, позволяющий с вычислять вероятность завершения выполнения проекта в заданный срок.

Introduction

A project plan is essentially important in software development process. Errors in project planning usually cause release delays or overlap available budget allocations. From the one hand, this situation is caused by the complexity of planning and take into account all the risks, obstacles and particularities of the product, customer, development team, budget etc., from the other hand there is the great influence of human factor on the software development process. The current approaches to software development process offer different process models (cascade, iterations, Agile) and there are program tools for project managers supporting each of the desired approaches (Ex. MS Project, Primavera Project Planner, Spider Project etc.)[5]. Usually, the main shortcoming of these systems is the lack of current analysis. Very often, there are only few options for analysis of the calculated plans. The modern software development process is extremely flexible and requirements may change very fast. Thus, a single planning mistake could cause extra costs and/or delays. There are methods for preliminary estimation of the project's plan (ex. PERT, COCOMO)[6] but they usually have comparatively noticeable calculation error or require too many details of a new project. Therefore, development of a new software

estimation model is an important task. Its resolution will allow creating an adoptive project schedule and making forecasts of the project's success probability.[7]

Project development team as a part of a queueing system.

During the development process, a project team can be considered as a part of a queuing system. This system's input has the stream of assigned tasks and output stream of resolved tasks. The state of this System Si is the state when i tasks were
completed. Before actual development, it is necessary for the process to set up the correct order of tasks, their complexity, priority and other factors. Any task could be decomposed into smaller tasks ending with a set of single day tasks. (A single day task is a task that could be resolved by one developer in a single day). Then we can form a queue of these tasks, which should be split for several iterations according to "agile" approaches[6] to software engineering process. An Iteration is a relatively short part of a product ending with a stable build with a subset of implemented requirements.[8] Iteration selection criteria could be either a fixed timeframe or the amount of tasks. Sometimes a project consists of the only one iteration. Queueing system represented by software developers consists of N channels corresponding to the number of developers. The nature of this random process can allow us to presume the probability of any next system state depending only on the current state and unrelated to how the system had come to this state. In general, the probability and the required time for the next tasks do not depend on the exact way of resolution of the previous ones. According to this fact, the software development process could be considered as Markov's process. There are similar tasks in the queue theory. For instance, there are models of anti-aircraft defense systems[1] and models of navy battle[2]. These tasks are also among the applications of reproduction and death models. Despite of the existence of these models we should consider some obstacles during the estimation of software process. First of all, software development tasks could be solved in parallel and the model of the queuing system depends on this. Secondly, there is orientation on time - we should be able to estimate the minimum number of tasks, which could be solved in the exact period to be sure in the iteration's scope.

When $t \to \infty$ we can replace the system of differential equations for the system of linear equations. The common approach in the models[2,3] is calculation of these probabilities at infinite time (so called balanced mode) which cannot be used for the described task because the exact system state is required at the each day of iteration.

Main parts and characteristics of the system.

We can consider the task solving process in software development as the model of the queuing system, which consists of several main parts: input stream, output stream, processing system and rules of the processing.

Some investigations made before introducing the system allowed determining the most typical characteristics of the queuing system: type of the input stream, structure of data processing, and the most important calculated values. In our case, the probability of solving the estimated number of tasks during iteration or the project in general was chosen as the main characteristic of the system (the probability of solving the exact number of tasks by the exact day of a project).

The development team of a project consists of several members. If we speak about agile approach, we can postulate that all the members have almost equal skills. The intensiveness or team velocity of solving the tasks can be calculated as a sum:

$$\lambda = \lambda_{a1} + \lambda_{a2} + \ldots + \lambda_{aN}, \quad (1)$$

where λ_{old} – the current team velocity, $\lambda_{newcomer}$ – the newcomer velocity or skill level, N_{old} – the current number of developers.

Basing on team velocity λ we can calculate λ_1, λ_2, ..., λ_N – the pace of solving tasks by developer depending on the number of people in the team.

Software development process can be modelled as a queuing system with waiting and without losses. Finally, all the proposed mandatory tasks must be resolved. Inquiries or tasks solved as a queue formed out from the project's scope taking priority, task complexity and dependencies.

The main characteristic of this system is the probability of solving the suggested number of planned tasks for the given time. The other characteristics could be (the list is not full because you may require some more for your own purposes):

1. The probability of solving *m* tasks in the day number *s*, where *m* is any number of tasks in queue, *s* - any desired day.

2. The number of tasks, which could be resolved in *an* iteration with the fixed iteration success probability.

3. The number of days required for *an* iteration when the volume of work is fixed and cannot be changed using the provided success probability.

The main mathematical instrument of the research is equations of the system states probabilities. In the queuing system, the number of solved tasks describes each state. When the task is resolved, the system changes its state to a new one. The velocity of state changes is determined by the average team velocity and the number of team members.

As the team consists of N developers some or all of them could complete their tasks in a chosen period of time (usually a day) and the system could shift from the current state S_k to any of the states S_{k+1} S_{k+n}, where $n \leq N$. The model of the system should include all the possible state changes.

This allows us to draw a highlighted graph of system states and form a system of equations that links all the probabilities of the possible states.

The solution of this system defines the probability of solving the exact number of tasks at any moment of time that allows calculation all the necessary system characteristics.

System states description

Before forming the graph of queuing system states for any large team some examples are provided for smaller teams consisting of one, two and three developers. The queuing system can be considered as the dynamic model because the exact form of the system is calculated upon project details.

The system could be in one of the possible states independent from the number of developers:

S_0 – the start point at the beginning of the iteration (no resolved tasks).

S_1 – one task is resolved.

S_2 – two tasks are resolved.

...

S_m – m tasks are resolved, i.e. all tasks assigned for the iteration.

S_{m+1} – m+1 tasks are resolved, i.e. all tasks assigned for the iteration plus one extra task.

...

S_K – K tasks are resolved, where K – any number of tasks because the queue of tasks in any software development project could be unlimited.

At the beginning of the iteration, the system is in state S_0. At any moment, any team member could complete his or her task. In this case the system changes its state to the nearest state – S_{k+1}; but if no tasks were completed the system stays in the current state. If there are several developers then the velocity of states changing is determined by a team velocity and could be obtained as an average value of state changes. For example, the velocity of changes from state S_i to state S_{i+2}, i.e. finishing two tasks at the same time is $\lambda_2=\lambda/2$. The same approach is used for further jumps. During the transition from state S_i to state S_{i+3} when there are three or more engineers, the velocity of changes is $\lambda_3=\lambda/3$.

Project independent and most common graph for N developers is shown on Fig. 1. During a short period Δt the queuing system could solve from 0 to N tasks, where N is the number of developers in the team. It means that the possible state switches are among S_i, and S_{i+N}. The velocity of state changes for each possibility λ_1, λ_2, ..., λ_N is calculated by the formula: $\lambda_i=\lambda/i$. This proposed approach is the simplest way of velocity estimation, which nevertheless shows adequate efficiency and accuracy in the range of analyzed projects.

With the help of this state graph, we can form the system of equations for the probability of each state.

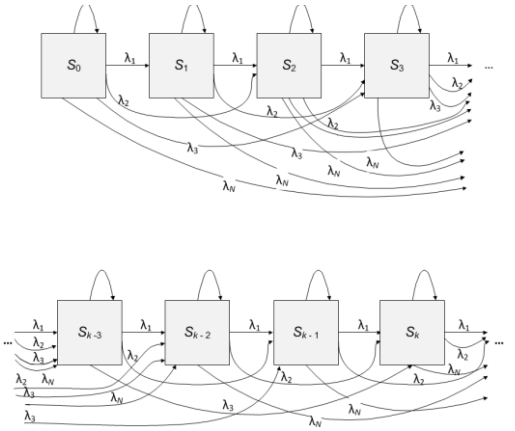


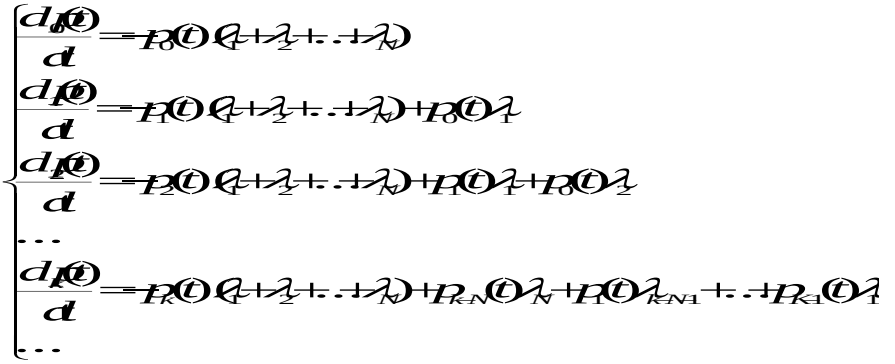
Fig. 1. Graph of system states of a system with N developers

Equations of system states

Based on the sate graph we can obtain probability equitation for each state. In general case, when team consists of N developers (Fig. 1) the probability of staying in the state S_0, is defined as probability of no shifts to any of the other states S_1, .., S_N. The probability of S_1 state in a short time frame Δt will include the probability of a change from the state S_0 to the state S_1 and probability of the fact if system was already in S_1 and this state wasn't changed:

(2)

Finally, for all states we can obtain Kolmogorov – Chapman system of equations:



(3)

After calculations and transformations:

$$\begin{cases} \frac{dp_0(t)}{dt} = -p_0(t)\sum_{i=1}^{N}\lambda_i; \\ \frac{dp_1(t)}{dt} = -p_1(t)\sum_{i=1}^{N}\lambda_i + p_0(t)\lambda_0; \\ \ldots \\ \frac{dp_k(t)}{dt} = -p_k(t)\sum_{i=1}^{N}\lambda_i + \sum_{i=1}^{N} p_{k-i}(t)\lambda_i; \\ \ldots \\ \sum_{k=0}^{K} p_k = 1. \end{cases} \quad (4)$$

Probabilities of the system states

The solution of this system of equations determines the dependence of the queuing system states on time. For the complex projects with many developers and many tasks assigned for each iteration, numeral solution of this system is preferred but not mandatory.

After solving this system of equations and defining the probability of each system state after the time period *t*, i.e. the probability of solving exact number of tasks by the team it is required to find the probability of solving the "at least" number of tasks during the selected period of time.

$P_i(t)$ is the desired probability of a successful project or iteration. $P_i(t) = P_i(t) + P_{i+1}(t) + P_{i+2}(t) + \ldots$ or $P_i(t) = 1 - \sum_{j=0}^{i-1} P_j(t)$. With the help of the same equations, we can calculate the probability of solving any number of tasks by the exact day of iteration or calculate the probability of success of the whole iteration.

If the calculated probability is less than required, some options may be proposed. If we keep the number of days in the iteration, we can choose the number of tasks, which will satisfy this probability of success. Alternatively, if the amount of work is important and it

is impossible to reduce it we can choose the number of days, which will meet this criterion.

The suggested probability limit chosen for accepting a project plan is 0.9. If the calculated success probability of a project's plan is lower than this value then the less number of tasks may be chosen or the duration of iteration may be increased to meet this requirement.

A task estimation example of a project's iteration

An example of modelling a queuing system and calculation of its characteristics is given below. Considered test project includes thee developers and six tasks for a short iteration. The duration of iteration is three days and the team's velocity is two tasks per day. Graph of states for this project is shown on Fig. 2.

Basing on this graph we can derive the equations system of the queuing system states. Transforming this system we can obtain the following Kholmogorov-Chapman system of equations:

$$\begin{cases} \frac{dp_0(t)}{dt} = p_0(t)(\lambda_1+\lambda_2+\lambda_3) \\ \frac{dp_1(t)}{dt} = p_1(t)(\lambda_1+\lambda_2+\lambda_3)+p_0(t)\lambda_1 \\ \frac{dp_2(t)}{dt} = p_2(t)(\lambda_1+\lambda_2+\lambda_3)+p_1(t)\lambda_1+p_0(t)\lambda_2 \\ \frac{dp_3(t)}{dt} = p_3(t)(\lambda_1+\lambda_2+\lambda_3)+p_2(t)\lambda_1+p_1(t)\lambda_2+p_0(t)\lambda_3 \\ \frac{dp_4(t)}{dt} = p_4(t)(\lambda_1+\lambda_2+\lambda_3)+p_3(t)\lambda_1+p_2(t)\lambda_2+p_1(t)\lambda_3 \\ \frac{dp_5(t)}{dt} = p_5(t)(\lambda_1+\lambda_2+\lambda_3)+p_4(t)\lambda_1+p_3(t)\lambda_2+p_2(t)\lambda_3 \\ \frac{dp_6(t)}{dt} = p_6(t)(\lambda_1+\lambda_2+\lambda_3)+p_5(t)\lambda_1+p_4(t)\lambda_2+p_3(t)\lambda_3 \end{cases} \quad (5)$$

Team velocity of tasks solving could be calculated as average velocity: $\lambda=2$ and $N=3$: $\lambda_1=2/1=2$, $\lambda_2=2/2=1$, $\lambda_3=2/3=0.66$.

To define the probability of solving k tasks for the iteration of d duration it is required to solve the system of $k+1$ equations. This system is solved numerically by Runge- Kutta method. The calculated results show the probability of each system state at each day of the iteration, Table 1. Knowing the model of system's states, we can calculate the probability of completing the selected iteration in time. The results are shown in Table 2.

Table 1. The probability of each system state

	$p(t)$						
t, days	0	1	2	3	4	5	6
1	0.1599	0.1599	0.1599	0.1599	0.1199	0.0879	0.0613
2	0.0256	0.0511	0.0767	0.1022	0.1150	0.1175	0.1115
3	0.0041	0.0123	0.0245	0.0409	0.0582	0.0742	0.0866
4	0.0007	0.2614	0.0065	0.0131	0.0222	0.0335	0.0458

Table 2. The probability of completing the selected iteration in time

t, days	*P*(*t*)					
	1	2	3	4	5	6
1	0.6802	0.5203	0.3604	0.2405	0.1526	0.0931
2	0.9234	0.8467	0.7445	0.6295	0.5120	0.4005
3	0.9836	0.9591	0.9183	0.8600	0.7858	**0.6992**
4	0.9967	0.9902	0.9771	0.9549	0.9214	0.8756
5	0.9992	0.9985	0.9954	0.9877	0.9791	0.9556

Table 2 shows that the probability of completing the selected iteration in time is lower than the chosen probability (0.6992 < 0.9). In this case, it is necessary to find alternative options: either the number of tasks, which could be solved in the iteration with the chosen probability of success or, as it is often requested in practice, the number of days required to solve all the tasks.

For our example, all the tasks could be resolved with the probability about 70 %, either we can chose 3 tasks for the iteration or extend the iteration time limits to 5 days to resolve 5 tasks with 95.5 percent probability.

Conclusion

The article proposes the mathematical model of software development process based on queue theory. Specialists resolve incoming tasks, each system state is characterized by the number of resolved tasks. The velocity of changing states is calculated as the average velocity of the team and the number of team members. In addition, it is taken into account that the average velocity of the team should be based on the velocity of historical projects and individual characteristics of employees. The average velocity is re-calculated after every iteration or in case of developers' number changing.

The suggested model will allow projects managers to change all the characteristics of a project and automatically estimate the probability of solving the selected number of tasks for any period or iteration. This should make a software development process more evident and successful.

REFERENCES:

1.E.S. Ventsel, Ovcharov L.A. *Theory of casual processes and its engineering application*. Moscow, "Higher school" ("VISSHAYA SHKOLA"), 2000. 480 p. L. N. Byzov *Modelling of random processes*. Spb: BGTU named after D.F. Ustinov "Voenmekh", 1998. 125 p.

2.Gnedenko B.V. *The course of probability theory*, 8th edition. Moscow: Editorial, 2005, 448p

3.Samarov K. L. *Queue theory basics* M: Resolventa, 2009. 340 p

4.Kultin N.B. "*Project management utilities*" Sbp: Politekhnika, 2002. 216 p.

5.Aliev H.P. "*Effective model of software development projects estimation*" Developed in Russia vol. 11 2008, pp. 338-364.

6.Politsyn S.A "*Approaches to time estimation in sowtware development process*" Program engineering Moscow, vol. 7. 2008, pp 9-15

7.Cohn M. "*Succeeding with Agile: Software Development Using Scrum*" Addison-Wesley Professional, 2009, 576 p.