ELEKTROTEHNIŠKI VESTNIK 88(3): 121-128, 2021
IZVIRNI ZNANSTVENI ČLANEK

Improving throughput and due date performance of IT DevOps teams

Tomaž Aljaž

Fakulteta za industrijski inženiring, Šegova ulica 112, 8000 Novo mesto, Slovenija
E-pošta: tomaz.aljaz@fini.unm.si

Abstract. Companies view Information Technology (IT) as a competitive advantage of the future. In recent years, IT departments have proven to have a significant impact on the business process improvement and better customer service, enabling organizations to become more competitive in the global marketplace. To meet these demands, many IT organizations are now transforming their software development and information technology operations to a combined team, called DevOps. However, with only organizational changes and no changes in how the IT DevOps teams manage their tasks, this is not easy to achieve, especially when there is a tension to keep all resources busy and overload them with a task by maintaining a high work-in-progress. As a result, organizations operating in this mode typically experience a significant degradation in their performance.
The paper presents a simulation-based performance assessment that significantly improves the performance of task execution by DevOps teams. An approach is presented to determine how to load and allocate resources and an appropriate allocation of additional (reserve) resources to improve performance. The simulations show a possibility of a 135 percent improvement in the Throughput (the number of completed tasks), work-in-progress reduced to the level of one percent of the completed tasks and the time needed to complete tasks is over 17 times shorter.

Keywords: DevOps, Resource management, Discrete Event Simulation, Constraint Management

Izboljšanje pretočnosti in rokov izvedbe IT DevOps timov

Podjetja na informacijsko tehnologijo gledajo kot na konkurenčno prednost prihodnosti. V zadnjih letih se je izkazalo, da imajo IT-oddelki pomemben vpliv na izboljšanje poslovnih procesov in zadovoljstvo strank ter omogočajo organizacijam konkurirati na svetovnem trgu. Da bi organizacije sledile tem smernicam, zdaj svoje IT-oddelke preoblikujejo v t. i. DevOps time, kjer sta v enem timu združena tako razvoj kakor tudi vzdrževanje programske opreme. Poleg organizacijskih sprememb je treba spremeniti še način vodenja in odobravanja novih nalog, še zlasti v organizacijah, kjer je tradicionalno glavno merilo zasedenost zaposlenih, in ne učinkovitost celotnega tima.
V članku je predstavljen postopek, s katerim lahko bistveno izboljšamo učinkovitost izvajanja nalog DevOps timov. To smo dosegli z upravljanjem količine odobrenega dela, razporeditve zaposlenih in vključitve dodatnih zaposlenih (specialistov) pri nalogah, kjer so se pojavile težave pri izvedbi. Predlagano rešitev smo preizkusili s pomočjo simulacijskega orodja ExtendSim. Simulacije kažejo, da so glede na tradicionalni pristop mogoči približno 135-odstotno izboljšanje pretočnosti (število dokončanih nalog), zmanjšanje nedokončanih nalog na približno 1 odstotek dokončanih nalog in čas, potreben za dokončanje nalog, zmanjšan za faktor 17.

Ključne besede: DevOps, upravljanje virov, diskretne simulacije, upravljanje omejitev

Received 24 February 2021
Accepted 5 April 2021

1 INTRODUCTION

The use of modern software development methods in organizations is becoming more and more prevalent to remain competitive today and in the future in the global market. In order to address their efforts, organizations are introducing agile and lean software development techniques to increase the pace of their software development process and to improve the quality of their software [1]. They use the approach called DevOps [2], that merges the traditional Software development and IT operations to deliver applications and services at a high frequency rate and maintain the quality of deliverables [3].

A the key elements of the DevOps are processes, tools and resources, especially human resources [4]. The processes and tools relate to automating and streamlining the software development and infrastructure management processes, working in very frequent but small software updates. These updates are usually more incremental in their nature than the usual updates performed under traditional release practices. Nevertheless, even software defects can be addressed much faster as changes are smaller.

A DevOps teams take a full responsibility for their applications and services to meet the customer needs, managing their tasks is essential. Traditionally, companies try to heavily load their (DevOps) teams, so that all their resources are always busy, especially the key ones. The assumption behind this is that even if a team (system) is heavily loaded, resources will find a way to get their tasks done. Such approach usually results in a flow of software deliverables degradation, quality decrease and minimization of the system (DevOps team) efficiency. On the other hand, having too little tasks, a team will starve (key) resources and consequently will also reduce the flow of tasks through the system. However, one of the most important factors of any successful team is meeting the requirements of the customer and increasing the Business Value of their activities. The Business Value can be measured with the productivity gain, product quality, customer satisfaction and various profit and market-oriented measures [5]. Unfortunately, this cannot be achieve if the (key) resources are overloaded with tasks. Therefore, a solution is needed to manage the DevOps team tasks and the resources allocate that will increase the predictability and stability of deliverables and the flow of the task development through it.

Ideally, a task schedule would be prepared in advance to assign just enough tasks to resources that need to work on by taking into account their availability, skills and competences. However, in a highly dynamic environment of the software development that we are currently living in, with demand fluctuation, customer behavior, high degree of task duration variability and technology uncertainty, this kind of scheduling is unrealistic. Moreover, an unexpected completion delay of a task can delay on one or more scheduled tasks, likely to result in a domino effect on the remaining tasks.

A traditional approach [6] to these issues is to estimate the workload and to set due-dates for the individual tasks or groups of tasks (e.g. new software release), based on the customer needs or priority. Defining the due dates on the tasks requires that several tasks need to start in parallel. If a task execution involves also a high degree of uncertainty, its estimation is usually inflated in order to meet due dates. To satisfy the last minute customer required changes, priorities of the task execution are changed on a regular basis, sometimes several times within a working day. Morover, the task due dates have also another side effect. A DevOps team will do whatever is takes to complete a task even at a lower quality or functional achievement. This results in a high degree of the DevOps team multitasking, the scheduled work is delayed and the performance and quality of DevOps team is reduced.

Instead of managing every resource in detail, the focus of the methodology presented in the paper reduced on only few (key) DevOps team resources. This ensures a high utilization rate of the (key) resources, while simultaneously leaving some amount of the excess capacity with the other resources without jeopardizing the task(s) due dates. The focus of the second part of the study is on reducing the impact of the high-variability tasks on the performance of the system by ensuring additional (expert) resources. They will not be employed on the most heavily used resources, but only when the existing resources have a trouble in completing their task and requiring help to complete it in time. The paper contributes to the research and practice of DevOps by (i) giving a comprehensive overview and methodology for releasing additional tasks in a DevOps process, (ii) methodology to reduce the impact of high-variability tasks on the system performance, and (iii) by validating the findings by a developed discrete-event simulation model built with ExtendSim.

In Section 2, a brief review is given of the literature our study is based on. Our observation is that there is a lack of analysis of the resource loading mechanisms in the DevOps area in the academic literature. To fill the gap, the traditional approach to managing the resource load is compered by the DevOps team. The problem definition is given in Section 3 and the research methodology in Section 4. In Section 5 are presented the results and the findings are summarized and discussed. Section 6 drows conclusions and gives implications for further research and practice.

2 LITERATURE REVIEW

The DevOps teams are involved in product development, projects, small development and solving software defects and other operational tasks. There is a relatively little work devoted to resource loading systems in DevOps environments and managing the impact of high-variability tasks on the system performance. There is a considerable research work in the field of resource scheduling with constrained resources in the project management area. [7], [8], [9] focus on the resource-constrained project-scheduling problem (RCPSP). The RCPSP research efforts on focus on exact or heuristic algorithms for constructing a schedule and the majority of these papers assume a complete information and a static, deterministic problem setting. This assumption reduces the RCPCP applicability in the today’s environment, where uncertainty of the software development tasks is high. In [10] and [11] the project performance improvements when the number of projects in the system is controlled are evaluated. [12] deals with the resources used multiple times in a single project, or are shared between projects, where any unexpected delay in a single task can cause a significant domino effects, delaying one or more projects. With the complicated and interrelated schedules that exist in a project environment, an attempt to tightly schedule projects does not produce satisfactory result in general. In [13][15] the concept of a Constant Work-In-Process (CONWIP) in a multi-project environment is simulated. Two control mechanisms are described, i.e. the constant number of projects in process (CONPIP) and the constant time of projects in process (CONTIP). The CONPIP mechanism restricts the number of projects and the CONTIP mechanism limits the total processing time off the projects that are active in the system. A potential drawback of the CONWIP protocol, including CONTIP and CONPIP, is researched in [14][14], suggesting that when the bottleneck is in the upstream direction, the focus should be on WIP leading to the bottleneck and not so much to the work required after the bottleneck resource. This is similar to the Drum-Buffer-Rope (DBR) methodology [15], [16], [17] used widely in manufacturing. In our study, it is used as a scheduling mechanism.

In a number of papers, the DBR systems are simulated to estimate their parameters, such as the time buffer and some others compare the DBR' effectiveness with the systems such as Lean or CONWIP. The application of the Theory of constraints [18] should not be limited only to the production environment. In [19] the DBR scheduling is applied to any type of the organization and service-oriented or manufacturing at the same degree of success. In manufacturing, DBR is used to schedule the machinery and in services, DBR can be used to schedule resources within an organization and appointments for customers, or to predict lead-times for customers [16].

The paper offers a simple methodology for managing tasks and resources by DevOps teams and a discrete-event simulation model developed to evaluate the research using ExtendSim.

3 PROBLEM DEFINITION

The study is based on an example given in [19][18]. A Microsoft IT XIT Sustained Engineering team maintains several applications for an internal use worldwide. The team completes small change requests and solves software defects taking less than 120 hours of work and involving mainly the software development and testing. The work backlog exceedes its capacity five times and is growing every month. The lead-time to complete a development request is typically 5 months. The due-date of the performance is almost zero. The customers are unhappy.

The task of the simulation is to get answer on the following questions:

- What are the average task execution times (flow time) for the traditional task management (to be used as a baseline)?
- What is the impact of the policy to do workload estimations for each task on the overall resource availability?
- What is the impact of the resource allocation on the same process on the overall task completion times?
- What is the impact of the management of the workload of a few of the key resources on the same process on the overall task completion times?
- Which methodology gives the best results in terms of the average task completion time, standard deviation for the task completion time, and lowest number of active tasks / inventory (Work In progress – WIP)?

4 SIMULATION SET-UP

Following the example from [19], a DevOps team receives development requests from different customer groups, responsible to prioritize the requests from their group. The average number of the demand is one per day. As shown in Figure 1, each request is sent to a DevOps team for a Rough Order of Magnitude (ROM) as the most important task. The Service Level Agreement with a customer defines that each ROMs estimate needs to be completed within 48 hours. When a request arrives, a developer and a tester assess the content of the request and provide a workload estimate. The customer assesses the cost of a request against its value and prioritizes it against other requests in the backlog. Estimates are therefore essential to facilitate both budgeting and prioritization.

Unfortunately, the team completes only 50 percent of the development requests. The other 50 percent are dealth in the projects as they are too big to be done by the DevOps team, too expensive and with no return on the investment, too slow to implement, or, the application is retired before the request is completed.

Therefore, the simulation setup consists of a Demand process and three workplaces, named ROM estimates, the Development and Testing. The Workplaces are ordered in a linear sequence. In the Demand process, there is an unlimited supply of new customer development requests and the completed requests are assumed to be all accepted from the end of the testing workplace. Each workplace has a WIP (Work in Process) storage, where completed tasks from the previous workplace are stored. Each workplace has a certain number of resources and duration. To simulate the task variability, the time to complete each task is based on the lognormal distribution [20]. On average the ROM estimates take four hours of work, involving both the Developer and Tester, and must be done within 48 hours, as described above. When there are no free resources for the ROM estimates, they are taken from the Development and / or Testing workplace. On average each development request on average takes 15 days, involving the Development and Testing resources, named Developers and Testers. The average execution time for the Developers is ten days (with a low end three days and whith a high end 20 days). The average completion time for Testers is five days (with a low end one day and with a high end ten days).

The simulation process for such case consists of the following four workplaces (see Figure 1):

- Demand (customer requests) – on average one per day,
- ROM estimates – average duration is 0.5 day; involving one Development resource and one Testing resource;
- Development time – the average duration is ten days; involving one Development resource;
- Testing time – the average duration is five days, involving one Testing resource.

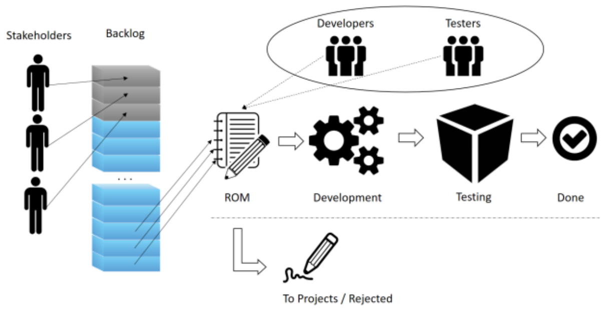

Figure 1. Simulation process setup (during a sample simulation in a simulator).

Simulation Basic rules:

- Each workplace uses a lognormal distribution to define the task completion time;
- One day of completion is one day, which is eight hours;
- For each workplace fixed number of resources is defined;
- Demand process draws the material from an unlimited supply (on a customer demand);
- The tasks completed in the previous workplace go to the next workplace;
- When there are more than one task at a resource, they are serviced using the First-In-First-Attended (FIFA) service discipline.
- The simulation starts with a zero task before each workplace (WIP = 0).
- Each simulation is run for 20 interactions, each interaction is run for 4000 simulation days.

Other simulation assumptions:

- These are simulations run that are modeled as "machines", no human-behavior issues are modeled such as the Student syndrome, Parkinson law, Multitasking, sick leave, etc., that could produce an additional delay and impact the overall performance of the result;
- There are no problems with the resources and logistics:
  - No lead time at the beginning of each task;
  - No tasks prioritizations;
  - Independent process – no artificial delays;
  - All customer requests are available on demand;
  - No technical disruptions;
  - Customers accept each completed tasks immediately.

4.1 *Traditional methodology*

The simulation setup for the Traditional process is done in different configurations, initially with three Developers and three Testers. The methodology starts by estimating the workload for a task placed on each resource, namely the Developers and Testers during a finite period. The methodology takes into account the available resources and based on the resource availability and on the change in the number of resources, Developers and / or Testers, are made for each task.

The simulation serves as a baseline for a comparison with other approaches, for i.e. Resource loading and Expert bench methodology.

4.2 *Resource loading methodology*

To test the Resource loading methodology, a different policy and resource allocation is simulated. The simulation setup provides the necessary information to decide whether to release a new customer request into the DevOps process. The methodology does not consider an exact work schedule, nor does it consider a unique timing situation that could overload even the best-planned schedule. Rather, it considers the backlog of work for the workplace.

The methodology provides a capability in a DevOps process, by which the impacts of new task releases on system performance can be compared and makes decisions whether to release new tasks into the system. Therefore, for each new task ready to be released into the system, the analysis evaluates the impact of this release on the resource workload at the key workplace, and thereby on the system performance, before a release of a task actually occurs.

The simulation model assumes that all task times are measured in days and uses a release mechanism that evaluates, on a daily basis, whether or not to release a new task into the system. A new task is released into the system only if the current resource workload for the most heavily used workplace is at or below a defined threshold level (two times the number of the available resources of the most heavily used workplace). In such scenario, the most heavily used workplace is Development, except in the scenarios with four Developers and two Testers (Resource Load 2) where the Testers workplace is the most heavily used resource.

4.3 *Expert Bench methodology*

The third simulation investigates the use of an additional expert resource in a task execution. Additional resources are added to reduce the impact of high-variability tasks on the performance of the system rather than adding more capacity to the most heavily used resources.

In a (typical) development process, tasks are subject to a high degree of uncertainty and it is very difficult to predict how long a task will take, especially in new subject areas. This variability can occur with any resource, not just the most heavily used resource. Therefore, a method is needed to minimize the impact of occasional task durations that are well above the median, with a minimum overhead.

An effective solution comes in the form of an Expert Bench, the resources that are not involved in other tasks and can work on any task, thus helping any resource that has a problem in completing a task. Such a situation exists when a resource has worked on a task at least 2/3 of the mean time [21]. The Expert Bench is used when a resource working on a task has problems completing it, otherwise it is idle. When the Expert Bench comes to help the resource with a problem in completing a task, the probability of completing a task doubles. The Expert Bench resource and the resource with problems in completing a task now work together to complete a task. After completing a task, the Expert Bench resource is returned to an idle state and is available to be called by any other resource.

5 RESULTS AND DISCUSSION

Table 1 summarizes the results of the 4000 simulation days for the different approaches, considered in our study.

Table 1. Summary of 4.000 simulation days

	Results				Resource	DevOps Team		
Methodology	WIP	Tasks	Flow Time	St.Dev.	loading	Developers	Testers	Exp.bench
Traditional 1	1260,8	727,0	851,9	725,5	No	3	3	No
Traditional 2	1302,8	791,9	965,1	720,9	No	4	2	No
Traditional 3	874,4	1103,1	655,0	595,2	No	4	3	No
Traditional 4	522,0	1478,4	443,4	426,3	No	5	3	No
Res. Loading 1	8,5	829,8	57,4	34,2	Yes	3	3	No
Res. Loading 2	15,8	1119,0	98,4	99,4	Yes	4	2	No
Res. Loading 3	8,1	1195,1	43,9	23,4	Yes	4	3	No
Res. Loading 4	9,3	1555,6	44,2	22,3	Yes	5	3	No
Expert Bench 1	8,7	1106,4	43,7	16,8	Yes	3	3	Yes
Expert Bench 2	12,5	1206,4	60,0	40,8	Yes	4	2	Yes
Expert Bench 3	9,2	1497,6	34,8	12,9	Yes	4	3	Yes
Expert Bench 4	9,9	1736,0	33,0	11,9	Yes	5	3	Yes

On the vertical axis, the following parameters are displayed for each process, based on 4.000 simulation days:

- Traditional methodology – initial simulation scenario with a different DevOp resource allocation:
  - Traditional 1: three Developer resources and three Testing resources;
  - Traditional 2: four Developer resources and two Testing resources;
  - Traditional 3: four Developer resources and three Testing resources;
  - Traditional 4: five Developer resources and three Testing resources;
- Res.loading – adding a Resource loading methodology into simulation:
  - Resource Loading 1: three Developer resources and three Testing resources;
  - Resource Loading 2: four Developer resources and two Testing resources;
  - Resource Loading 3: four Developer resources and three Testing resources;
  - Resource Loading 4: five Developer resources and three Testing resources;
- Expert Bench. – adding an Expert Bench into a simulation scenario:
  - Expert Bench 1: three Developer resources, three Testing resources and two Expert Bench resources;
  - Expert Bench 2: four Developer resources, two Testing resources and two Expert Bench resources;
  - Expert Bench 3: four Developer resources, three Testing resources and two Expert Bench resources;
  - Expert Bench 4: five Developer resources, three Testing resources and two Expert Bench resources.

The initial analysis is made with a more complex configuration environment including additional resources and a different resource allocation. However, increasing the number of resources and a different resource allocation deviates from a real-life situation and gives no additional information that would justify the increased complexity and the scale of the simulation.

On the horizontal axis, the following parameter values are displayed:

- WIP – WIP after the end of the simulation;
- Tasks – quantity of the completed tasks after simulation;
- Flow Time – Average Flow time of the task completion through the DevOps process at the end of simulation time;
- Standard deviation – standard deviation of Flow time of task execution through the DevOps process of the simulation.

5.1 *Rough Order of the Magnitude (ROM)*

The activity of estimating ROMs is treated as a top priority for the DevOps team. Therefore, if there is a request for ROM to execute task, the Developer and Tester need to stop working on the existing task and move their activities to do a ROM estimate. Uncompleted tasks at the Developer or Tester, if any, are returned to the head of the queue and wait to have resources available to continue with their tasks completion.

With the Traditional 1 scenario, having three Developers and three Testers on the DevOps team, the impact of a ROM request is significant. When a request arrives, one Developer and one Tester take a request (simulating a real case where a DevOps team checks-out the source code, maintains the necessary paperwork and associated operations guide) and assess the impact of the development request. The duration of the ROM estimate for both the development and testing is on average half a day. Performing a ROM estimation activity takes about one day per a ROM request from their available capacity. Having in mind that in 2021 there are 261 weekdays available and the resources have on average some 40 days of leave (vacations, training, sick leave, …) the ROM estimates reduce Development and Testing resources by some 40% of their available capacity (see Table 2).

Table 2. Impact of the ROM estimates on the resource availability

	DevOps Team			Reduced
	Developers	Testers	Effort per ROM	availability
1	3	3	1 man-day	39%
2	4	2	1 man-day	44%
3	4	3	1 man-day	34%

Table 2 shows a similar calculation for the scenarios with four Developers and two Testers, four Developers and three Testers, and five Developers and three Testers.

5.2 *Completed tasks*

Every completed task provides benefits to the stakeholders and thus ensures a Business Value. Therefore, the more tasks completed, the more benefits expected.

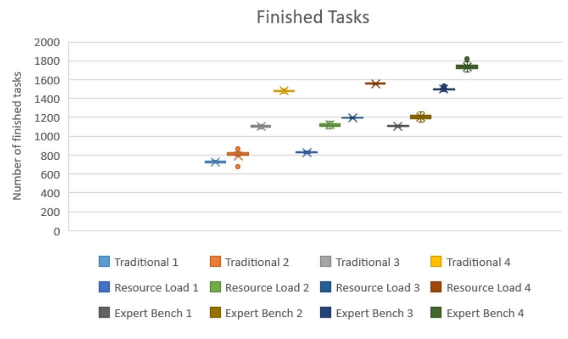

Figure 2. Quartile chart of the completed tasks at the end of different simulation scenarios.

Figure 2 displays the area of values for the completed tasks taking into account the average value for each interactions. The traditional approach has the lowest values (from 727 to 1478) compared to the Resource Load approach (from 829 to 1555) and Expert Bench approach (from 1106 to 1736). Dispersion of the results is very similar for all scenarios, except for the Resource Load 2 and Expert Bench 2 approach where it is higher due to the unstable conditions of the most heavily used workplace (all available Testers and Developers are fully utilized). From the number of the completed tasks, the Resource Load gives a much better result compared to the traditional approach (on average around 115 percent), while the Expert Bench gives the best result (on average by 135 percent better than the Traditional approach and on average around 118 percent better than the Resource Load approach).

5.3 *Work In Progress*

WIP in a DevOps process refers to a partially completed task that is waiting to be completed. This means that an uncompleted task brings no Business Value, but acts as a buffer. This means that the WIP levels are ideal at low values, but when they are too low this may cause starvation of the most heavily used workplace (resources), thus providing unnecessary delays in the task completion.

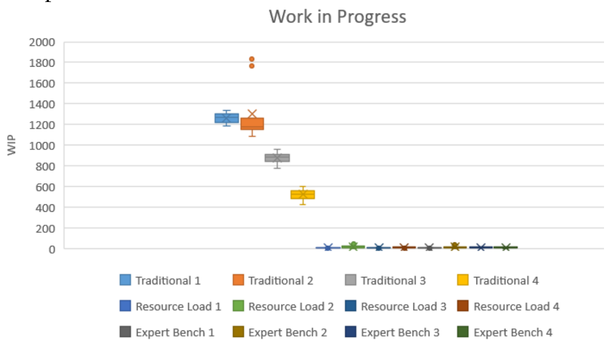

Figure 3. Quartile chart of the WIP results at the end of a simulation for different simulation scenarios.

Figure 3 displays the area of the WIP values for each simulation scenario. The WIP levels are the highest using the Traditional approach of managing tasks (on average from 521 to 1260) and this process has the highest result dispersion. The Resource Load (on average from 8 to 15) and Expert Bench (on average from 9 to 12) scenarios have a significantly lower and similar size with the lower result dispersion. Resource Load 2 and Expert Bench 2 scenarios have a higher result dispersion due to unstable process conditions, described in Chapter 5.2. Results show that compered to the WIP levels, the Resource Load and Expert Bench scenarios give a better result (excluding Resource Load 2 and Expert Bench 2 scenarios), i.e. only 1 percent of a completed tasks.

5.4 *Average Flow time*

The Flow time, i.e. the time taken to complete a task, is important for the stakeholder of the DevOps process. Consequently, lower Flow time results are more desirable, thus monetizing investment to Business Value. Moreover, a low standard deviation of a Flow times is wanted to meet the stakeholders expectation of having stable and predictable delivery dates.

Figure 4 displays the area of values for the average task Flow times for different scenarios. These values are the highest for the Traditional approach scenario (on average from 443 to 852 days) with the highest dispersion. The results for the Resource Load scenario (on average from 44 to 57 days) and Expert Bench scenario (on average from 33 to 43 days), excluding Resource Load 2 scenario (on average 98 days) and Expert Bench 2 scenario (on average 60 days) approaches are significantly lower (on average 17 times) and of low a dispersion (on average 41 times) compared to the Traditional approach.

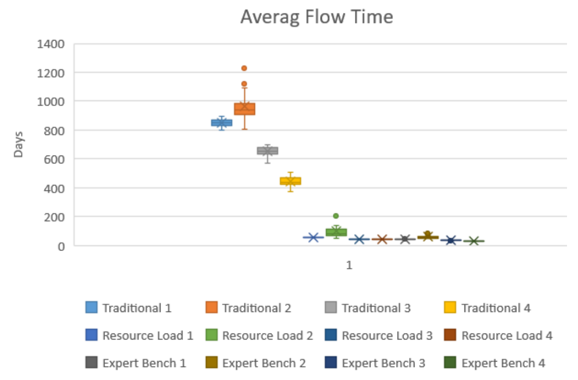

Figure 4. Quartile chart of an average task Flow time at the end of a simulation for different simulation scenarios.

The second criteria of the previous parameter is a standard deviation of the task Flow time. Figure 5 shows that predicting the task Flow time is the most difficult for the Traditional approach where the standard deviation is on average from 426 to 715 with an average of 617. The standard deviation of the Resource Load approach is from 22 to 34 with an average of 26, excluding the Resource Load 2 approach (from 30 to 228, an average of around 100). The standard deviation of Expert Bench approach is from 12 to 17, with an average of around 14, excluding Expert Bench 2 approach (from 27 to 75, an average of around 41).

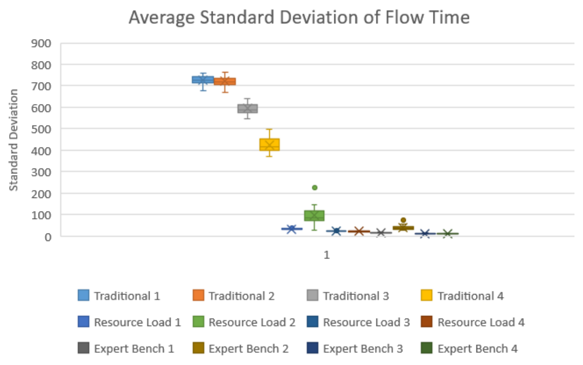

Figure 5. Quartile chart of the standard deviation of the task Flow times at the end of a simulation for different simulation scenarios.

Combining the two values, the following task Flow times are estimated, the exeption being the approach with four Developers and two Testers (i.e., Traditional 2, Resource Load 2 and Expert Bench 2):

- The average task Flow time of the Traditional approach is 650 days and on average standard deviation of 582. So, the development times from 68 to 1232 days are expected (note that the task Flow time is constantly growing).
- The average task Flow time of Resource Load is 48 days and the average standard deviation is 28. So, the expected development times are from 20 to 76 days.
- The average task Flow time of the Expert Bench is 37 days and the average standard deviation is 14. So, the expected Flow times are from 23 to 51 days.

6 CONCLUSION

Researchers in the resource management and scheduling are urged by various organizations to find optimal ways of working. Researches critically evaluate processes to determine how effective they are in delivering a maximum value and strive to incorporate methods that significantly reduce the Flow time, lower the costs, and improve the quality of their products and solutions or the services they provide. The study analyzes and formalizes the results of using an ExtendSim simulator by changing the resource allocation, reducing the workload on resources, and providing experts to help in the software development process called DevOps when needed.

The simulations show that some simple steps proposed in the study reduce the task completion time and the number of uncompleted tasks, and increase the number of completed tasks. The traditional approach does not consider the resource availability of the DevOps process. Specifically, the number of new requests / tasks is greater than a DevOps process can handle, resulting in a high number of incompleted tasks (Work in Progres) and by far the longest and most unpredictable task completion time (flow time). Moreover, the ROM process takes 40+ percent of the available resource capacity. Managing the load on the most heavily used resources as well as invoking expert resources (when needed) shows a stable and relatively low number of incompleted tasks, but a higher number of finished tasks and a shorter task completion time compared to the traditional approach. Moreover, the simulations show that the resource allocation (i.e., the number of the Developers versus the Testers) significantly effects the effectiveness of the overall DevOps process.
With no new resources and no changes to the way a DevOps team performs the software development tasks, such as the design, coding, and testing, the task completion time is over 13 times shorter, the work in progress is over 104 times reduced and the increase in the number of the completed tasks is on the level of 110 percent. This is achieved by managing the utilization of the most heavily used resources, thus providing the means to manage the entire DevOps process. Also, this

releases new tasks in a DevOps process at a rate acceptable by the resources. By adding resources in strategic places and experts to help when needed, the time to complete a tasks is now reduced by over 17 times, the amount of the work in progress is reduced by over 98 times and the number of completed tasks is increased at least by 135 percent. To sum up, increasing the number of resources improves the Business Value of the activities done by a DevOps team at a certain cost, of course, which is left to be decided upon by the organization itself.

The first important conclusion of the study is that controlling the number of incompleted tasks in the DevOps process (Work In Progress) is more important than a continuous flow of new tasks (i.e., the development request) as it significantly reduces the task Flow time through a lower number of uncompleted tasks (WIP). The second important conclusion is that introducing expert resources to help when needed offers a better result in the overall DevOps process for allowing an easy resource management and simultaneously leaving room for improvement by introducting the Critical's Chain Project Management Fever Chart to prioritize tasks that are already in a system. Finally, eliminating the ROM estimation process with a policy change ensures a considerable and immediate improvement, freeing 40+ percent of the resource availability.

LITERATURE

[1] F. M. A. Erich, C. Amrit, M. Daneva. (2017). A qualitative study of DevOps usage in practice". Volume29, Issue6 Special Issue: Recent Advances in Agile Software Product Development

[2] Lianping Chen. (2018). Microservices: Architecting for Continuous Delivery and DevOps". in IEEE International Conference on Software Architecture. Seattle, USA: IEEE

[3] Bass L, Weber I, Zhu L. (2015). DevOps: A Software Architect's Perspective". Addison-Wesley Professional

[4] Jha, Pratibha & Khan, Rizwan. (2018). A Review Paper on DevOps: Beginning and More To Know. International Journal of Computer Applications.

[5] Guido Schryen (2013) Revisiting IS business value research: what we already know, what we still need to know, and how we can get there, European Journal of Information Systems, 22:2, 139-169

[6] Spalek, Seweryn. (2016). Traditional vs. Modern Project Management Methods. Theory and Practice. Smart and Efficient Economy: Preparation for the Future Innovative Economy, 21st International Scientific Conference

[7] Ballestín, F. and Leus, R. (2009). Resource-Constrained Project Scheduling for Timely Project Completion with Stochastic Activity Durations. Production and Operations Management, 18: 459-474.

[8] Creemers, S. (2015). Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. J Sched 18, 263–273

[9] Amer Fahmy, Tarek M. Hassan, Hesham Bassioni. (2014). Improving RCPSP solutions quality with Stacking Justification – Application with particle swarm optimization, Expert Systems with Applications, Volume 41, Issue 13 , Pages 5870-5881, ISSN 0957-4174

[10] Anavi-Isakow, S., & Golany, B. (2003). Managing multi-project environments through constant work-in-process. International Journal of Project Management, 21(1)

[11] Bendavid, I., Golany, B. (2011). Predetermined intervals for start times of activities in the stochastic project scheduling problem. Ann Oper Res 186, 429–442

[12] Pritsker, AAB, Waiters, LJ, Wolfe, PM. (1969). Multiproject scheduling with limited resources: a zero-one programming approach. Manag. Sci. 16(1), 93–108

[13] Hübl A. (2018). Conwip. In: Stochastic Modelling in Production Planning. Springer Gabler, Wiesbaden.

[14] Nasim Nahavandi. (2009). CWIPL II, a mechanism for improving throughput and lead time in unbalanced flow line, International Journal of Production Research, 47:11, 2921-2941

[15] Bhardwaj, & Gupta. (2010). Drum-Buffer-Rope: The Technique to Plan and Control the Production Using Theory of Constraints. World Academy of Science, Engineering and Technology, Article 19, page 103

[16] Tomaž Aljaž. (2014). Out of chaos in 12 months - improving lead time of sprint projects in software development implementing Drum Buffer Rope Solution, ERK

[17] Zupancic, D., Buchmeister, B., & Aljaz, T. (2017). Reducing the Time of Task Execution with Existing Resources – Comparison of Approaches. International Journal of Simulation Modelling, 16, 484-496.

[18] Moos. (2007). Improving Service Quality with the Theory of Constraints. Journal of Academy of Business and Economics, 4(3) 1-15

[19] D. J. Anderson, D. Dumitriu. (2005). From Worst to Best in 9 Months: Implementing a Drum-Buffer-Rope Solution in Microsoft's IT Department, TOC ICO World Conference November 2005

[20] Trietsch, D., L. Mazmanyan, L. Gevorgyan, and K. R. Baker. (2012). Modeling activity times by the Parkinson distribution with a lognormal core: theory and validation, European Journal of Operational Research, v 216, pp. 386-396.

[21] Oswald, A., Muller, W. (eds.). (2017). Management 4.0 – Handbook for Agile Practices. Books on Demand, Norderstedt

Tomaž Aljaž is having over 22 years of professional experience in the area of Information & Telecommunication. He is employed in Spar Slovenija where is managing IT projects with a particular focus on improving performance of the project team, establishing and maintaining an optimal use of resources and reducing the operational risks. He is also a faculty member of the Faculty of Industrial Engineering Novo mesto, Slovenia. His past experiences are related to the R&D environment where he worked as a Resource, Project, Product and Solution manager. He has published several papers on the information technology and telecommunication area, resource management, project management and process improvements using Theory of the Constraints methodology. He is a holder of a Ph.D. degree in Electrical Engineering received from the Faculty of Electro Engineering and Computer Science of Maribor and has completed courses in Constraint Management at the Washington State University, USA. For over 11 years he has been teaching at a graduate and postgraduate level the topics related to performance improvement of organizations, project management, information technology and telecommunication. In 2018 and 2019 he was granted a Certified Scrum Master (CSM) and Certified Scrum Product Owner (CSPO) certificate and in 2014 a Jonah certificate, by the Theory Of Constraints International Certification Organization (TOCICO).