Out of chaos in 12 months - improving lead time of sprint projects in software development implementing Drum Buffer Rope Solution

doc. dr. Tomaž Aljaž[1]

[1] *Šolski center Celje, Pot na Lavo 22,3000 Celje*
E-pošta: tomaz.aljaz@gmail.com

Abstract

This is a case study about implementing common sense changes in managing software development area. It's an example how Theory of Constraints (TOC) applications can reinforcing the beliefs of management and encourage them to do the right thing – making just a few simple changes, collecting less data, spending less time on overhead and administration and do more tasks that benefit the whole development process (and organization).
The software development team is developing and maintaining more the 70+ applications. The sprint project on average takes 5 working days of work, but lead time is almost 8 times longer. The backlog of sprint projects was constantly increasing, the due date performance for all non-mandatory development request was very poor. The development requesters were unhappy. Last year, supported by top management, activities were started in order to break existing situation. By performing analysis using TOC tools and applications understanding what needs to be done was identified, especially how production Drum-Buffer-Rope solution can be used. With almost 20% reduction of resources, no changes to how the team performed software development tasks like design, coding and testing, the changes to how the work was queued and estimated resulted in a reduction of lead-time by 40%. The backlog was reduced by more the 50%, improving investment for development by 40% and improved satisfaction of development requesters.
This paper shows how tools and application of TOC, especially Drum-Buffer-Rope solution, provides meaningful improvements in (software) development area, without a need to make changes in technology, but focusing on the management, planning, scheduling and queuing of development tasks.

1 Introduction

The company has internal software development team that is responsible for developing solutions for 70+ applications on two major functional areas. Majority of development is requested by internal users, mainly trigger by regulatory requirements (strict and short deadlines). The development activities are divided into three areas: (1) sprint projects, (2) projects and (3) maintenance and support. The same resources are working in both development and also in maintenance & support tasks. There are also different decision bodies, with different participants, managing software development. In the reminder of the document, we will focus on the management of sprint projects.
Up to mid 2013, all sprint projects were approved on monthly meeting, with »sooner we approve the development requests, sooner it will be finished« manner. There was no commitment on delivery date (except on mandatory deadlines imposed by regulator), on best effort principle (if we will have time, we will do it) and without any global prioritization criteria. There were also "short-cuts" for development approvals, expediting line, done by approval of IT director or personally with direct interaction between internal user and development team (each developer had big backlog of development requests). Also there was no strict authorization policy implemented in software development change management tool to limit the state changes in workflow – there was "trust" that no one will take advantage or to expedite own development or maintenance requests.
Last year's situation on sprint project shows that development tasks are unequally balanced with high backlog of sprint projects. The majority of tasks are stuck in software development department, as shown in bellow.

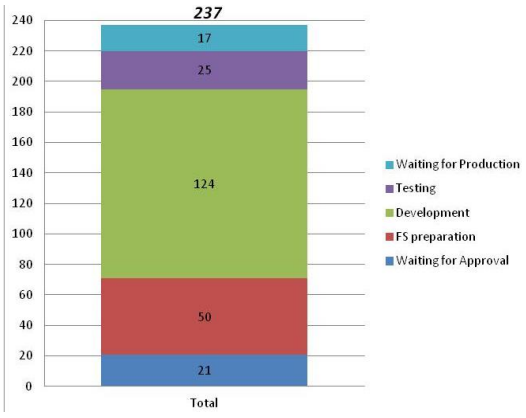

Figure 1: Number of sprint projects based on dev. phase

Clearly, a lot of specifications were prepared (and workload estimation) usually far before development has started, thus spending valuable time of resources

preparing functional specifications for a tasks that will probably needs to be done all over again. There is big change that information system will be changed on application and specifications / design (and workload estimation) will need to be done again, with no guarantee that the resource that did the estimate would be the same resource that did the work before.
Historical data (gathered over at least 12 months) showed that a typical sprint project took 37 business days to process through development. The low end was 1 day and the high end was 232 days. Furthermore, analysis was done on average lead time of sprint projects, where average lead-time was compared to reported workload. We identify opportunity window:

- Average lead time was 37 working days; and
- Average reported workload was 5 days.

$$Opp.window = 1 - \frac{Av.rep\_time}{Av.lead\_time} = 1 - \frac{5}{37}$$

$$Opp.window \cong 86\%$$

Cleary, the existing solution of managing development requests have big opportunities for improvements. Several (main) undesirable effects were identified, using Theory of Constraints Thinking Process [2]: lack of common management / view of development requests, unclear responsibilities, different approval bodies, no common prioritization criteria, lack of common view on resources, spending time on overhead activities, etc.

2 The need for new development process

The need for new "development process was identified, where the main goal (strategy) was defined.

> The speed needed to develop a solution is the number one consideration. The target is not how many development tasks are started, but how many development tasks are completed (in time and within the approved budget).

As "the flow" is the major consideration, the applications of Theory of Constraints (TOC), in particular it was identified that Drum Buffer Rope solution can be used.
The TOC assumes that each organization is represented by number of processes, which are interconnected and interdependent. Therefore, we can compare organization with the power of "chain", where the power of the whole chain is limited by the strength of the weakest link. In the case of organizations, this means that its results depend on the "performance" (speed, quality) of the weakest link. Moreover, the weakest link in organization represents system limitation and restricts it to achieve better results. Consequently, this means that any improvements on the link, which is not the weakest, (usually) do not provide meaningful improvements - may cause more negative consequences (e.g., increasing inventory level, stock of uncompleted work). The TOC defines the "weakest link" in an organization as a constraint.
In order to achieve the most of current organization, five focusing steps are defined by the TOC [3]:

- Identify the system's constraint(s) (that which prevents the organization from obtaining more of the goal in a unit of time)
- Exploit the system's constraint(s) (get the most out of the constraint, e.g. avoid unnecessary idle time, farm out work to other resources where possible)
- Subordinate all other resources to the constraint (align the whole system or organization to support the constraint's operation, e.g. prioritize repair and maintenance, change process batch size on non-constraints.)
- Elevate the system's constraint(s) (make other major changes needed to increase the constraint's capacity, e.g. buy a new machine)
- Warning! If in the previous steps a constraint has been broken, go back to step 1, but do not allow inertia to cause a system's constraint.

Additionally, The Drum-Buffer-Rope (DBR) [6] is powerful and robust TOC solution that is intended to manage the flow of work through a (development) process rather than managing the capacity of resources. It is designed to protect against general cause variation that can't be removed from the system and some special cause variation (e.g., Murphy). As basis for its work it uses first three steps of five focusing steps defined by TOC: Identify the system constraint, decide how to exploit the system constraint and Subordinate everything else to the above decisions. Basic principle of DBR is shown in Figure 22.

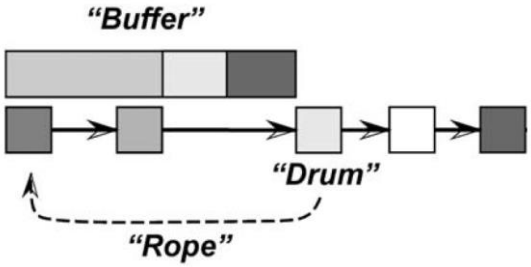

Figure 2: Basic principles of DBR

Based on the new findings, existing development process was slightly changed, mainly at initial part of it and accordingly implemented within existing Change management tool. Also additional control was done on roles that allow development request classification and approval, providing strict control for releasing new tasks in development process.
All development requests are initiated by internal users using "proposal" form and sent for approval. For development requests it is required to fulfill related Key Performance Indicators, which are used for ranking purposes. The development request is then classified, defined application that will be upgraded / used, rank

(prioritize) and estimated workload (cost). The rank can only be modified by top management, using "correction" factor. If the workload (or cost) exceeds the number for sprint projects, the request is moved to Project Management office and top management for approval – separate stream and is out of this document scope. As result, central repository of all development requests was build with global ranking and application classification.

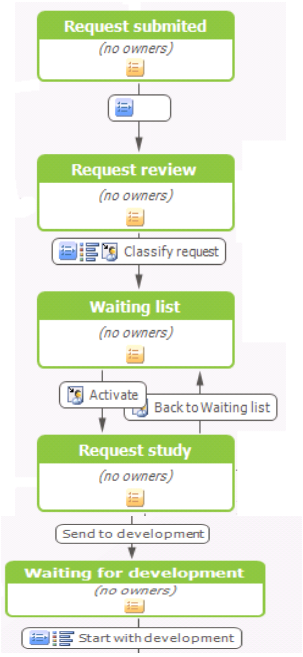

Figure 3: Strict control for releasing new tasks

3 How to improve speed of sprint projects

In order to build initial situation on development area, first step was checking the status of all "active" sprint projects. Based on analysis, all sprint projects that were already in "active" development (development has started, but not finished) remained in the system and other moved to "waiting" area or canceled the obsolete one (defined by internal users). The goal is to remove at least 20% [4] of "active" development requests, reducing high level of work in progress, as shown in Figure 4.

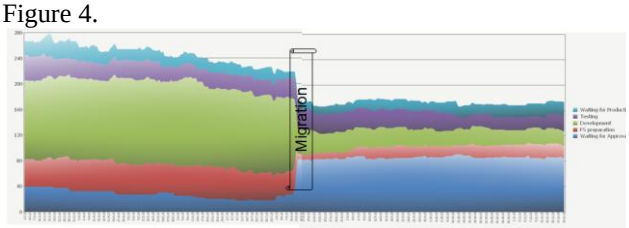

Figure 4: Number of active sprint projects

Moreover, the development team (analysts, developers and testers) are no longer required to provide workload estimation, thus moving their work to special dedicated team, which roughly estimate needed workload. Actual development work is estimated when the resource is ready to do work. As a result, estimates are never wasted – the analysis work involved in making the estimate is used immediately and performed by the same developer or tester. This classic example of subordination decision delivering the desired results, and free their capacity for about 20%.

As it was already mentioned, the constrain resource (Drum) determines the speed of development activities. The selection of the buffer in front of development constrain resource must be defined in order to prevent his / her work starvation. Instead of trying to make schedule of each and every resource using sophisticated tools to prevent starvation of constrain resource, taking account also Murphy, we can manage development of development requests in more pragmatic way. We can release new development requests into development based on the rate (Rope) that constrain resource (Drum) can consume, while at the same time protecting it from starvation (buffer - number of development request that are waiting in front of him / her). For each application, constrain development resource is defined (selected) and his / her backlog for 1 month of work defined.

The "rope" ensures that development requests enters the development process at a rate that is synchronized with the capacity of the constrain resource. Consequently, the number of (development) requests on non-constrain resource is regulated - not to overload the constrain resource. Additional development requests remain outside the development process (in waiting area) until constrain resource is "free". The buffer shows the number of (development) requests that are waiting to be done by constrain resource and provides insurance that there is always enough work to. Monitoring the status of buffer (number of request) will enable quick reaction of possible starvation of constrain resource, due to disruption caused by "last minute" changes in development tasks or Murphy. Also ensures integrity of the scheduled work - all none-constrain resources have excess capacity so they will be able to fulfill the possible gaps in buffer.

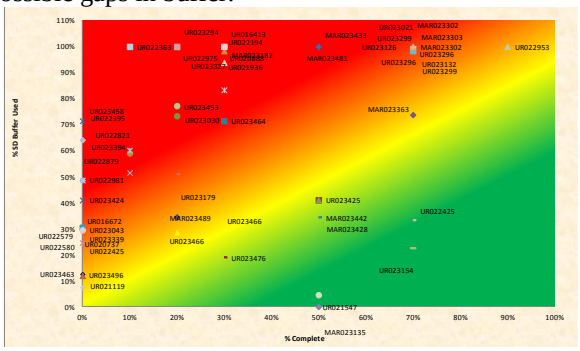

Figure 5: Example of Fever Chart for sprint projects

In software development process it is important to know which development requests needs to be done first. It should be clarified the difference between rank for releasing new development requests into the development process and priority of development work. The release of new development request is done based on the business related Key Performance Indicators, workload (cost) and resource availability, especially constrain resources. The development priority is based on the committed deadline, thus protecting the cost and scope of development requests. In order to be able

defining work priority, the Fewer Chart was used, as shown in Figure 5.
The Fewer Chart show work in progress in linear relationship based on the development phase and its estimated / committed deadline. The "X" axis represents the phase of development, mapped into the %, while the "Y" axis represents the time "buffer" (how much time in % is still left to fulfill the requested / committed deadline of development request). The % is calculated based on the approved date, requested / committed deadline of particular development request and current date.
In the example in Figure 55, it can be seen many sprint projects in "red", indicating that are already late in respect of defined deadline. As the main goal is to deliver sprint projects on time (thus protecting scope and cost), the need to start working on sprint projects that are in the "red" area is identified, while monitoring the "yellow" and "green" ones.

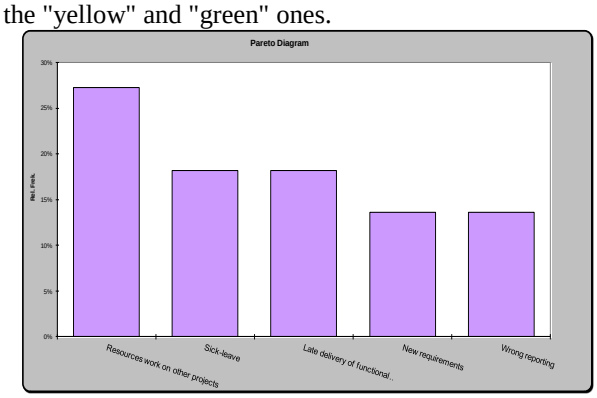

Figure 6: Number of active sprint projects

The reasons for delays must be identified and collected as a basis for further analysis and improvement, e.g., Pareto diagram [6], as shown in Figure 46.

4 Results after 12 months

The productivity or throughput has risen steadily throughout the past year thus decreasing lead-time for development request from 37 working days to 21 working days – 40% improvement.

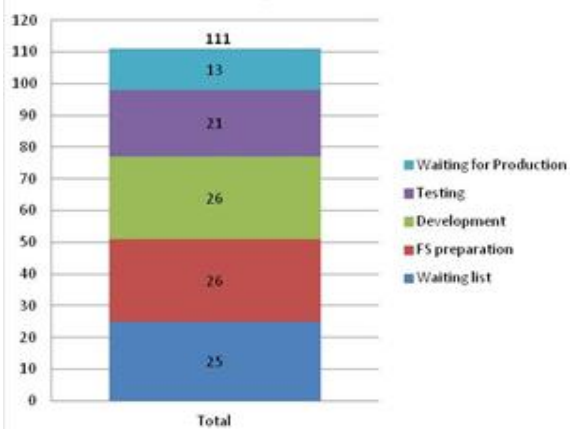

Figure 7: Results after 12 months

There is still high dispersion of lead time, ranging from low end of 1 day to high end of 109 days. This indicates that there is still too much work in progress based on the capacity of development team (reduced for 20% compared to last year), which needs to be addressed in the near future.
As can be seen in Figure 7, the reduction of work in progress, from 237 active sprint projects to 111, was done in one year. This resulted in more than 40 % reduction of "inventory" and improving investment life cycle (quicker amortization of developed requests).

5 Conclusion

In today's constrain environment where there is constant pressure "to do more with existing resources", the TOC application tools and techniques can be used to address these needs. The basic philosophy of TOC and their five focusing steps can be applied in software development area. As the main constrain of development process are human resources, the proposed solutions are focused on identifying them, exploit them and subordinate all others to prevent starvation of constrain resources. Several principles can be used in software development process, like building "central warehouse" of all development requests, implementing Drum-Buffer-Rope solution for improved and stable delivery rates, "replenishment" and "inventory control", Fever Chart and Pareto diagram to improve performance of resources involved and to provide Process Of On-Going Improvements.
Finally, the example presented in this document shows that productivity of (software) development team in not related to the development tools but to the management, planning, scheduling and queuing of development tasks. Without adding resources or changing any of the development tools, it was possible to decrease lead-time of sprint projects by more than 40% (improve productivity), with 20% less resources.

References

[1] Eliyahu M. Goldratt, Jeff Cox. The Goal: A Process of Ongoing Improvement, North River Press, 2004

[2] Lisa J. Scheinkopf. Thinking for a Change Putting the TOC Thinking Processes to Use, CRC Press LLC, 1999

[3] James F Cox III, John Schleier. Theory of Constraints Handbook, McGraw-Hill Professional, 2010

[4] Mark Woeppel. Projects in Less Time, TOCICO 2009 Conference presentation

[6] Tomaž Aljaž, Lidija Grmek Zupanc, Branka Jarc Kovačič, Gabrijela Krajnc, Mateja Demšar. Kako s pomočjo Teorije omejitev narediti več a delati manj, Dnevi slovenske informatike, april, 2014