From Worst to Best in 9 Months:
Implementing a Drum-Buffer-Rope Solution in Microsoft's IT Department

By David J. Anderson & Dragos Dumitriu, Microsoft Corporation, November 2005

Abstract

This is a case study about implementing common sense changes where they were needed. It's a story not about the brilliance of the Theory of Constraints (TOC) but rather TOC playing a role as permission giver, reinforcing the beliefs of a manager and encouraging him to do the right thing. It's also a story about simplicity – making just a few simple changes, collecting less data, spending less time on overhead and bureaucracy and more on productive tasks.

The XIT Sustained Engineering team is part of one of Microsoft's eight IT groups. The department maintains over 80 applications for internal use worldwide by Microsoft employees. The team completes small change requests (often bug fixes) involving less than 120 hours of development work. The team was considered the worst performing in its business unit at the start of the 2005 fiscal year (July 2004). The backlog of work was exceeding capacity 5 times and it was growing every month. The lead time for a change request was typically 5 months. The due date performance was almost zero. The customers were unhappy. A new program manager stepped in to coordinate the efforts of XIT Sustained Engineering. He wanted to make some changes but was unclear whether they were the right changes and how effective they might be. By performing an analysis using the 5 focusing steps of TOC, David Anderson helped him to understand how his proposals fitted with a drum-buffer-rope and Throughput Accounting implementation. With no new resources, no changes to how the team performed software engineering tasks like design, coding and testing, the changes to how the work was queued and estimated resulted in a 155% productivity gain in 9 months. The lead time was reduced to a maximum of 5 weeks – typically 14 days. Due date performance improved to greater than 90%. The backlog was worked off and the department is no longer seen as an organizational constraint. Customers are delighted.

This study will show that TOC's fundamental 5 focusing steps [Goldratt 1984] and the production flow solution, Drum-Buffer-Rope [Goldratt 1986], show significant value in information technology, software development, without a need to resort to more elaborate TOC solutions such as Critical Chain project scheduling or The Thinking Processes.

Introduction

It is often assumed that the simplest form of the Theory of Constraints – the 5 focusing steps [Goldratt 1984] – cannot readily be applied to knowledge work problems. The production flow solution Drum-Buffer-Rope is overlooked by many practitioners who jump to the assumption that Critical Chain scheduling [Goldratt 1997] and/or the Thinking Processes [Scheinkopf 1999] offer the only solution in this space. This case study will show that the 5 focusing steps and Drum-Buffer-Rope can readily be applied to a knowledge work problem and can show rapid, significant results without significant investment of time, money or resources.

Background

In September 2004, Dragos Dumitriu accepted a new challenge – to take on program management for a team responsible for change requests within one of Microsoft's IT departments (XIT). This team is now known as XIT Sustained Engineering.

During the previous fiscal year the team was entirely located in Redmond, Washington, starting in July 2004, development and testing has been transferred to a subsidiary department in Hyderabad, India. India offers some interesting advantages. Not only are costs lower but often initial quality measured as defects per function (or line of code) can be far superior. The Indian vendors have widely adopted use of the Software Engineering Institute's Capability Maturity Model Integration (CMMI) and the Team Software Process / Personal Software Process (TSP/PSP) as their engineering method. Though often regarded as heavyweight and bureaucratic, these methods do lead to very high quality and reliable output though some might believe that this is true at the expense of productivity and frequent, rapid delivery of customer value.

XIT Sustained Engineering was transitioned to India during the final few months of Microsoft's 2004 fiscal year (ending June 2004). In July 2004, design, development and testing of change requests for XIT had been handed over completely to Hyderabad. This quarter saw the worst ever productivity from the department – only 17 change requests were completed. This had the effect of underscoring the poor performance of the team with regards to producing enough throughput to sustain demand. The backlog of requested changes had been growing steadily.

Dragos decided he wanted to take on the challenge. At the same time, he, like several other managers in XIT, had been reading David's book [Anderson 2003]. They were not aware that David now worked at Microsoft. On October 14th, 2004, Dragos turned up to hear David speak at the Seattle chapter of the American Society for Quality. Afterwards, he approached him and asked if he could help with changes needed in XIT.

The Current Reality (July 2004)

Introduction

In July 2004, the position of Program Manager for XIT Sustained Engineering was advertised internally as vacant. There were few applicants. In recent months, demand for change requests was running at about 1 per day – 85 per quarter – but supply was running around 6.5 per developer [Table 2] for the same period. With the completed transition to Hyderabad, there were only 3 developers on the team. A sustained demand of 85 per quarter had to be met by a team capable of only 20 change requests in the same period. Between July 2004 and October 2004 only 17 change requests [Table 3 and Figure 6] were completed. Typically, requests were taking at least 5 months to process [Figure 7] and that lead time was growing. Customer satisfaction was at an all time low and the team was considered the worst in the business unit. Taking on such a department was, for Dragos, a career risk.

The Job Description

The job description for the open position called for a program manager with ASP skills, SQL Server Admin skills and MS Project Server knowledge in order to provide elaborate reports around work schedules. In other words, it was thought that what was needed was someone who had the skill to code a website which could query a database of work items, combine the results with a scheduling mechanism and forecast how the backlog could be addressed. It was perceived that the problem was "not enough data, and not enough transparency into that data." The situation reflected a general tendency from management to demand more data in hope that it would reveal the root cause of un-reliability.

Customers

XIT Sustained Engineering received change requests from four customer groups [Figure 1]. Each group's interests were represented by a customer Product Manager who was responsible for prioritizing the requests from his group. Each group provided funds for sustained engineering work based on the size of their application portfolio and work forecast.for break/fix issues.

Estimation

When a new change request arrived – typically, one per day – it was sent to the developers and testers for a "rough order of magnitude" (ROM) estimate [Figure 1]. One developer and one tester would pick up the request. ROM estimation lead time was controlled by a service level agreement (SLA) with the customers which stated that all ROMs would be completed within 48 hours. This facilitated scheduling and prioritization of the new requests against the existing backlog and schedule.

The effect of the SLA for ROMs meant that providing ROMs on new change requests was expedited as top priority. With only 3 developers and 3 testers on the team and a total of over 80 applications to maintain, the impact of a ROM request was significant. When a request arrived, one developer and one tester had to checkout the source code, along with the maintenance paperwork and the operations guide. They would read and assess the

impact of the change request. This would typically take 4 hours for each discipline. The productivity impact was one full man day per ROM to estimate both development and test impact. The ROM activity had the effect of sucking about 1 day per change request from available capacity (capacity = 85 * 3 = 255). Calculating ROMs for prioritization and scheduling purposes was sucking up to 40% of available capacity.

Cost Accounting

Using the ROM, and a fixed rate per hour for dev and test activities, the customer would assess the cost of a change request against its value and prioritize it against other requests in the backlog and against other requests from the other 3 customers [Figure 1]. The customer expectation was that the cost of this work could be called off like withdrawing from a bank account. When there were no requests on any given day, week or month, then no withdrawal against the account would be made. The costs of running the department that day would be assigned to an overhead bucket.

Estimates were therefore essential to facilitate both budgeting and prioritization.

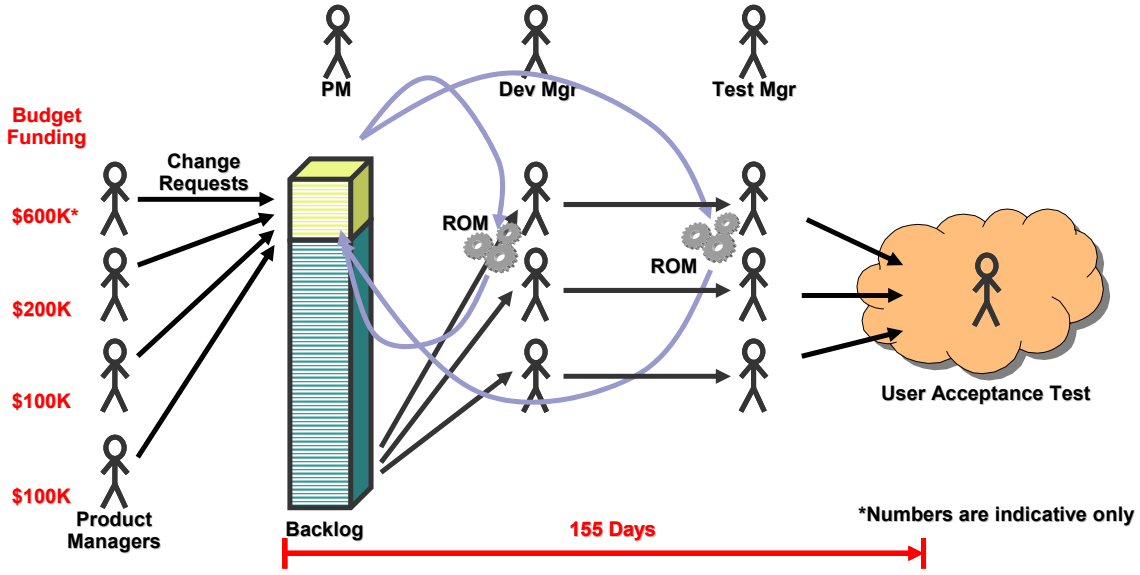

Figure 1. Existing Process showing Estimation and Budgeting

Prioritization

Prioritization of both backlog and any new requests was done via a monthly meeting and involved the 4 customer representatives and managers of the sustained engineering team. During each meeting, the entire backlog was re-prioritized as urgency of requests could change over time, requiring a complete reprioritization. With a backlog exceeding 80 and growing by as much as 10 per month, and a throughput of only 6 or 7 requests per month, the amount of effort spent on prioritizing requests, which would not be worked on within the next month, was considerable. Often, requests would be continually de-prioritized and would languish on the backlog being continually scheduled out later and later. The due date was being missed and then renegotiated time and again.

Actual Effort

Historical data (gathered over at least 9 months) showed that a typical change request took 5 business days to process through development. The low end was 1 day and the high end was 15 days. In theory, anything that was estimated (as ROM) as greater than 15 days was transferred to a different department to be processed as a project.

ROM Efficiency

While the team was providing a ROM for each received request, data showed that only about 50% of requests were actually completed by the team. The other 50% fell into several categories: too big – divert to project; too expensive – no return on investment justification; too slow to implement – overtaken by events, or the application was retired before the request was completed.

ROMs were using 33% - 40% of capacity while only 50% of requests being estimated were actually completed. Around 15%-20% of capacity was being used to estimate work that would never be undertaken.

Perishable Knowledge Work

In theory, much of the analysis from the ROM activity would be reusable when actually doing the development work. However, two things prevented this. Firstly, change requests were estimated often months before implementation. Any knowledge gained during the estimate was lost. Secondly, there was no guarantee that the developer who did the estimate would be the same developer who did the work.

In reality all analysis work needed to create a ROM was waste (muda [Womack 2003]). It's only function was to allow prioritization and facilitate triage of requests that were too big or too expensive.

Productivity Data

FY04	Raw Demand	Productivity	Dev Productivity (per head)	Test Productivity (per head)
Q1	89	53	8.8	17.6
Q2	139	57	9.5	19.0
Q3	83	37	6.2	6.2
Q4	55	39 (India 10)	6.5	6.5

Table 2. XIT Agile Team Fiscal Year 2004

In the second quarter of the 2005 fiscal year (the 4th quarter of calendar 2004), the team was renamed to Sustained Engineering.

FY05	Raw Demand	Productivity	Dev Productivity (per head)	Test Productivity (per head)
Q1	56	17	5.7	5.7
Q2	49	30	10.0	10.0
Q3	51	36	12.0	11.0
Q4	49	44	11.0	22.00

Table 3. XIT Sustained Engineering Fiscal Year 2005

The Future Reality (October 2005)

Introduction

David's first reaction to the described current reality was to see a flow problem which could be improved with a classic Drum-Buffer-Rope [Goldratt 1984] solution in the Theory of Constraints. Dragos had already determined that estimating ROMs was sucking capacity and reducing throughput, however, how to get rid of it? Surely, ROMs were essential to facilitate the cost accounting? Dragos also knew that the cost accounting and calling off work against an account balance did not make sense. However, how to explain this to the customers and agree a suitable alternative? Articulating the problem was a challenge. David saw a solution from the basic rules of Throughput Accounting [Corbett 1998]. Meanwhile, the request to find a manager who was better at administering SQL Server and querying a database looked like a classic case of drowning in the data ocean whilst diving in hope of a solution as described in The Haystack Syndrome [Goldratt 1990].

Productivity Data

FY06	Raw Demand	Productivity	Dev Productivity (per head)	Test Productivity (per head)
Q1	71	56	11.4	24.3
Q2				
Q3				
Q4				

Table 4. XIT Sustained Engineering Productivity Fiscal Year 2006

Performance

Productivity (or throughput) has risen steadily throughout the past year from 17 to 56 change requests per quarter [Table 3, Table 4 and Figure 6]. A new trust exists between the customers and the team where normal committed change requests are delivered for user acceptance testing within 25 business days, typically 12 to 14 business days [Figure

7]. However, the process must continue to accommodate high severity or urgent requests that must be expedited. These are classified as Severity 1 and expedited such that existing work is placed on hold to free up resources. Due date performance on this promise has been greater than 90% despite the possible intervention of "expedite" severity 1 requests. Due to the dramatic improvement in "time to resolve" performance, the percentage of Severity 1 requests has fallen in comparison to total requests. The cost accounting numbers look even better! The cost per change request has fallen from approximately $7,500 to around $2,900 [Figure 6].

Resourcing

There are now 5 developers and 3 testers. This increase happened in Q1 FY06 up from 4 developers and 2 testers in the previous quarter. In Q1 FY05 the ratio was 3 developers and 3 testers.

Capacity Constrained Resource

Development is the designated capacity constrained resource (CCR). Testers have slack capacity. All efforts are made to keep developers fully occupied [Step 2 of the 5 focusing steps – Exploit]. This is achieved through use of a buffer with approximately 7 days of additional work queued

Buffers

Developers now pick change requests for development from a buffer of approved and committed requests. The buffer size has grown slightly with increased resources and is typically about 7 requests – the rope is a total of 12 requests (7 in the buffer and 5 are work in progress).

Testers take work from a "ready for test" buffer which has a size limited to around 8. Some requests go straight in to the "ready for test" buffer without going through development. These are changes made by non-developers such as graphical changes or other data changes which still require testing but do not require changes to source code. Such changes are actually very few in number but they do run the risk of moving the capacity constrained resource from development to test. Hence, the second "ready for test" buffer is required to protect the whole system in the event that the constraint moves to the test department under fire from direct input requests [Figure 5].

Estimates

Developers and testers are no longer required to provide ROMs [Step 3 of the 5 focusing steps – Subordination of other aspects of the system to the decision made to fully exploit developers]. There is no estimating done for the purposes of planning, scheduling, prioritization or budget reasons. All change requests are treated as equal in cost based on the average production rate against the total burn rate of the department.

Actual development and test work is estimated when the resource is ready to commence work and only to facilitate the TSP/PSP process 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 [a classic example of a subordination decision delivering the desired results].

Prioritization

Prioritization is now done in an ad hoc fashion, via email of phone call. When a development buffer slot is vacant, the 4 customers are asked to negotiate a candidate from the backlog to fill it. Choice is made based on urgency and customers decide which request on their backlog is most important for delivery within the next 25 business days.

Cost Accounting

Cost accounting is no longer used. Customers agreed that the cost of running the XIT Sustained Engineering department is fixed. [The reality is that a 12 month agreement is made with the vendor. The costs are essentially fixed for a whole year. A Throughput Accounting interpretation of the operations of the business unit.]

Allocation of capacity is based on percentage of total funding provided by each customer group. Allocation of empty buffer slots is passed around amicably based on the promise of fair and even buffer allocation. The 80 applications have been sorted into buckets for each customer. Each of them has been asked to prioritize their top three applications. This helps facilitate the amicable cross-team discussions and choice of candidates for any given vacant spot. A record is kept of delivered requests against the budgeted capacity for each customer.

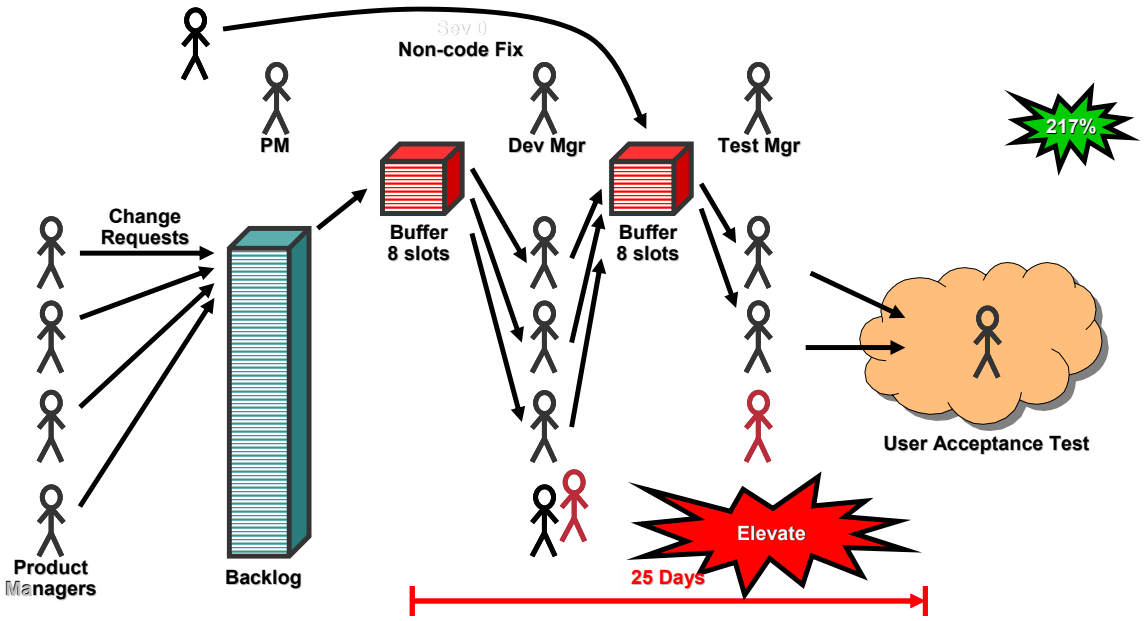

Figure 5. The new process with buffers and additional resources

Job Description

There is clearly no longer a need for a program manager who has skills in deep sea diving on an ocean of data in the warehouse. Dragos now finds that the sustained engineering process runs like clockwork with the customers negotiating buffer allocation by email

directly with a local manager in India. This has freed up his time to look for new management challenges and address new problems in XIT.

Management Interventions

The transition from the current reality of September 2004 to the future reality of October 2005 was achieved with 3 management interventions (or Injections to use the TOC vernacular). Two of these seemed like common sense; TOC merely helped underscore their importance and gave Dragos the confidence to go and carry them out. The third intervention was the creation of buffers. This wasn't immediately obvious but it was the mechanism which facilitated the other two interventions. Without the generic Drum-Buffer-Rope solution for flow problems in TOC, the entire set of changes may not have worked as explained here…

Intervention 1 – Buffers

Before buffers were added to the process, the team had no control over their ability to deliver according to customer expectations. New requests could arrive stochastically and in far greater numbers than could reasonably be processed. They sucked capacity through the need for a ROM estimate within 24 hours. New requests pushed the existing schedule out further and forced rescheduling and reprioritization which involved senior management and sucked yet more capacity from the team.

Introducing a buffer and only committing to a delivery date when a request was slotted into a buffer had several effects [Figure 5]. Firstly, it enabled management to actually manage the process and to set customer expectations in a manner that was controllable and could be delivered against. Secondly, it had the effect of stemming off the released demand at a rate that the team could consume it. Doing this helps to reveal the slack resources in the chain. Thirdly, it reduces lead time by insuring that resources are single-tasking and focused on moving their current request through to the next stage. Finally, buffers focus the customer attention on what is most important at that moment in time. With the increased trust that exists with a predictable system, the customer can make a rational and clear decision about which request best deserves to be assigned to an empty buffer spot.

The use of the buffer system was sold to the customers using common sense arguments. The theory behind Drum-Buffer-Rope was not introduced. Instead it was argued that it simply didn't make sense to keep prioritizing and scheduling requests that wouldn't be worked on for months. Everyone found the monthly prioritization meeting to be painful. Hence, the answer was to stop doing it.

Intervention 2 – Stop Estimating

Estimation was sucking up to 40% of capacity. Removing estimating produced a big and immediate productivity improvement [Table 3]. However, permission to stop estimating required a change in mindset from customers and internal management. They needed to stop the cost accounting for prioritization and budgeting. This was presented to them using more common sense arguments.

(1) There is no such thing as an account

The budget transfer that happened on the first day of the fiscal year was not an account the customer could draw off against. The funds were allocated in full to the vendor and drawn off in monthly tranches against the agreed burn rate.

(2) ROMs are too expensive

The act of calculating a ROM was much more trouble than it was worth. Up to 40% of capacity was used to calculate ROMs. The customer was asked whether they'd prefer more throughput and simply treat all requests as equal in cost.

There were a couple of edge cases that needed consideration. Some requests were too big. They should be diverted as projects. It was agreed that these could be queued in the buffer as normal and that a developer would alert the team as soon as it was feasible to classify something as too big. There was a risk that occasionally a change slightly bigger than 15 days would slip through but this was worth the risk.

As Deming [1994] pointed out, it is better to manage for the common cause and treat the exceptions as exceptional rather than legislate for every eventuality.

Also there are times when a change may be too expensive. This would be treated slightly differently. If it was worth scheduling for delivery within the next 25 business days, it would be delivered – regardless of its cost! If it wasn't really worth doing then it shouldn't have been selected to fill a precious buffer slot.

(3) All changes would be treated equally for cost purposes

Based on the available data, the average request took 5 working days to process through development. In this respect, all changes would be treated equally, the real question being, what is most important to have delivered within 25 business days? And not how much did it cost?

Intervention 3 – Reallocation of Resources

Historical data suggested that the correct ratio of developers to testers ought to be 2:1. With the team following the TSP/PSP process, initially quality ought to be high and re-work and re-testing very low. This would also indicate that a 2:1 ratio might be appropriate. In Q1 of fiscal year 2005, that ratio was 1:1, 3 developers and 3 testers. The testers appeared to be fully utilized.

The combination of the introduction of a Drum-Buffer-Rope solution which stemmed off released demand at the rate at which the capacity constrained resource (at this point unidentified within the black box for development and testing) could consume, and the removal of the need to ROM estimate new requests (freeing a further 33% of capacity), it was thought that the testers should have slack time. To verify this, Dragos had to visit the

team in India and actually observe them working. This direct observation of the team for two weeks did indeed reveal that there was slack capacity in test.

The resultant intervention was to ask the vendor to reallocate one resource from test to development giving a ratio of 4:2 [Figure 5]. This produced further improvements as seen in Table 3.

Elevate for more Throughput

Given the success of the team in fiscal year 2005 ending in June 2005, it was recognized that Dragos had done a great job improving this team. However, some additional capacity was needed to fully meet customer expectations.

The capacity constrained resource of 4 developers had been fully exploited using a buffer. Throughput had settled at around 11 change requests per developer per month. The rest of the system had been subordinated to the decision to fully exploit the developers and keep them working optimally on coding change requests, by abandoning ROM estimates and scheduling only on buffer slot allocation. SLA's had been renegotiated and use of cost accounting for budgeting and prioritization had been dropped. All of this had produced a 155% throughput improvement without additional money or resources. However, in order to further improve it was finally necessary to make an investment to elevate the constraint.

2 more staff were added, one in development and one in test, taking the ratio to 5:3 [Figure 5]. The result on throughput of this extra capacity can be seen in Table 4. The productivity per resource clearly shows stabilization at around 11 requests per developer per quarter.

The final piece of good news is that the whole team is no longer capacity constrained. They reduced the backlog from 80 to under 10 requests. Customer service from XIT Sustained Engineering is better than ever.

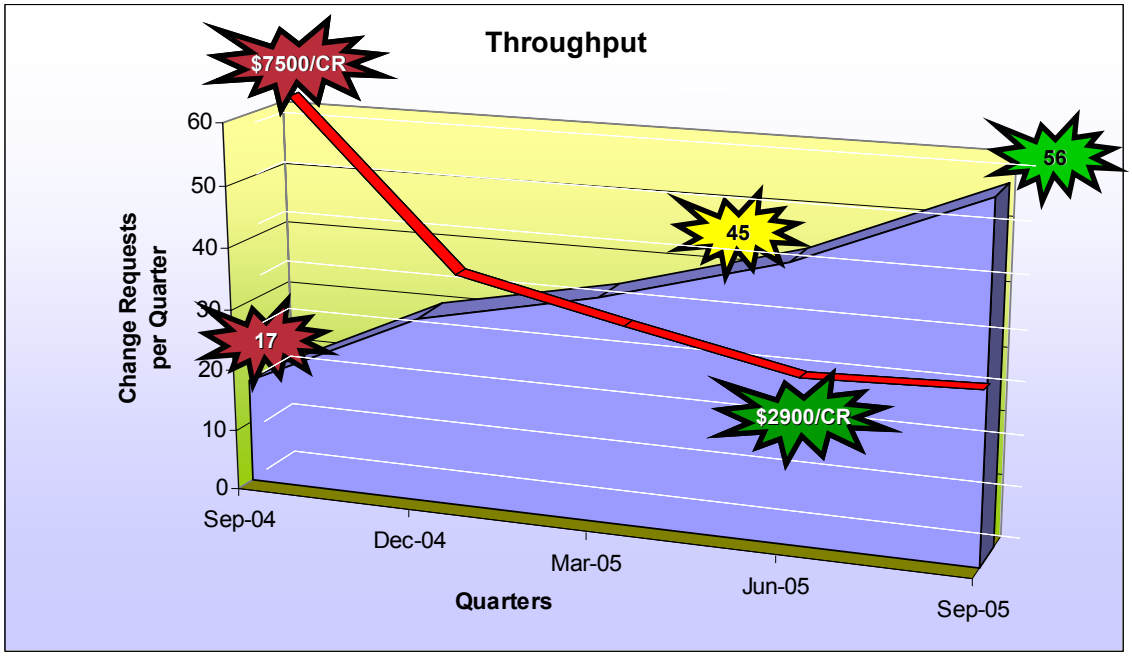

Figure 6. Throughput and Cost per Change Request

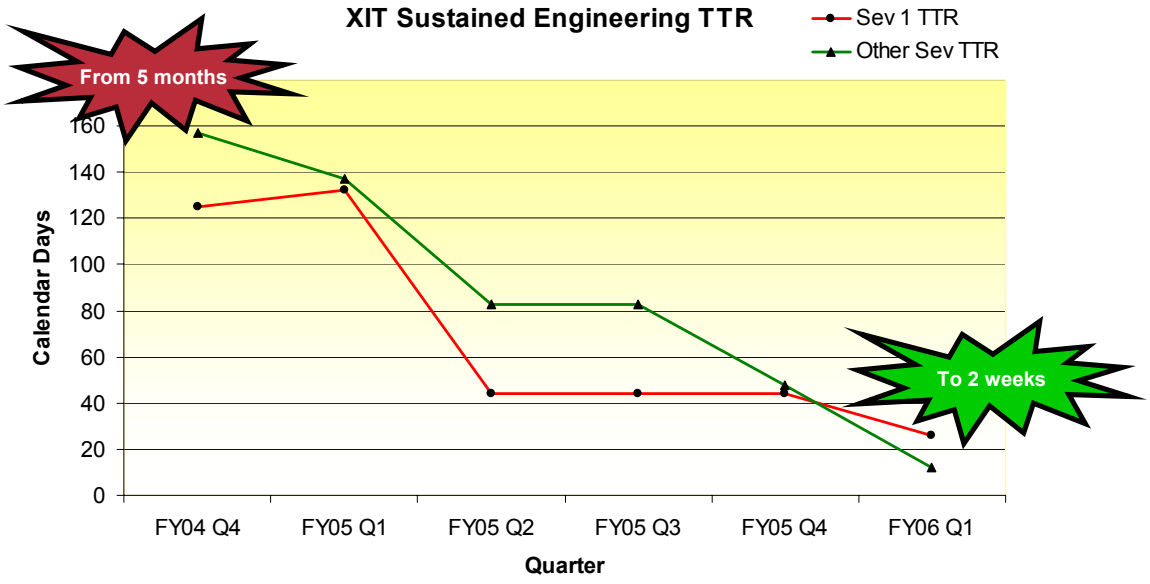

Figure 7. (TTR) Time To Resolve (official Lead Time metric)

Conclusions

The Theory of Constraints is highly relevant to knowledge work problems. Its basic philosophy of 5 focusing steps and the production flow solution, Drum-Buffer-Rope, can be applied in Information Technology software maintenance and development situations. This specific example shows that much of what constrains the productivity of software engineers in not related to the method of engineering but to the management, planning, scheduling and queuing of work. Without adding resources or changing any of the engineering method, it was possible to increase productivity by more than 100%.

This is further confirmation that more elaborate TOC solutions such as Critical Chain project scheduling [Goldratt 1997, Leach 2005] and use of The Thinking Processes [Scheinkopf 1999] are not required to make significant and lasting improvement in information technology software development work.

References

[Anderson 2003] Anderson, David J., ***Agile Management for Software Engineering – applying the Theory of Constraints for Business Results***, Prentice Hall Professional Technical Reference, 2003
[Corbett 1998] Corbett, Thomas, ***Throughput Accounting***, North River Press, 1998
[Deming 1994] Deming W. Edwards, ***The New Economics – For Industry, Government and Education*** (2nd Edition), MIT Press, 1994
[Goldratt 1984] Goldratt, Eliyahu M., ***The Goal***, The North River Press, 1984
[GOldratt 1986] Goldratt, Eliyahu M., Robert E. Fox, ***The Race***, North River Press, 1986
[Goldratt 1990] Goldratt, Eliyahu M., ***The Haystack Syndrome – Sifting Information Out of the Data Ocean***, North River Press, 1990
[Goldratt 1997] Goldratt, Eliyahu M., ***Critical Chain***, The North River Press, 1997
[Leach 2005] Leach, Lawrence P., ***Critical Chain Project Management***, 2nd Edition, Artech House, 2005
[Scheinkopf 1999] Scheinkopf, Lisa J., ***Thinking for a Change – Putting the TOC Thinking Processes to Use***, APICS 1999
[Womack 2003] Womack, James P. and Daniel T. Jones, ***Lean Thinking – Banish Waste and Create Wealth in Your Corporation, Revised and Updated***, Free Press, 2003

Contact Details

david.anderson@microsoft.com
ddumitri@microsoft.com