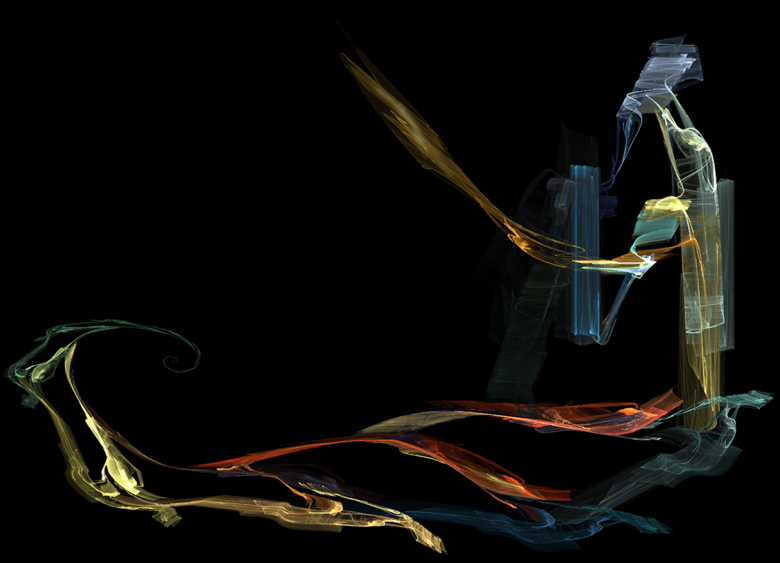
STEVE TENDON & WOLFRAM MÜLLER

TAME THE FLOW

Hyper-Productive Knowledge-Work Management



Tame the Flow

Hyper-Productive Knowledge-Work Management

Steve Tendon and Wolfram Müller

This book is for sale at http://leanpub.com/tame-the-flow

This version was published on 2013-07-01

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Steve Tendon and Wolfram Müller by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just bought Tame the Flow (#tameflow) by @tendon

The suggested hashtag for this book is #tameflow.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search/#tameflow

Contents

Credits

Cover Image: *Fractal Phlegyas on the River Styx*, ©2008, Devin Moore http://www.devinmoore.com/[1]

Acknowledgments

Thanks to Rudi Burkhard for connecting Steve and Wolfram.

Thanks to all reviewers for excellent comments, insights, suggestions, critique and even encouragement. In particular: Michael Burrows, Dimitar Bakardzhiev, Adail Muniz Retamal, Paul Merino, Pascal Van Cauwenberghe.

About the Authors

We live in an every-changing world, you can find out the latest about the authors by checking their online profiles.

Steve Tendon

- Google Plus: https://plus.google.com/u/0/113524936087749568174[2]
- Linked In: http://www.linkedin.com/in/tendon[3]
- Twitter: `@tendon` https://twitter.com/tendon[4]
- Blog: http://chronologist.com[5]
- Site: http://tendon.net[6]

[1] http://www.devinmoore.com/
[2] https://plus.google.com/113524936087749568174?rel=author
[3] http://www.linkedin.com/in/tendon
[4] https://twitter.com/tendon
[5] http://chronologist.com
[6] http://tendon.net

Steve is an experienced consultant, specializing in *organizational hyper-productivity* and *process evolution* for both small and growing businesses, as well as large and established organizations. He has evolved his own unique approach to business management inspired by a diversity of contemporary management theories, and based on how information flows through a modern (computerized) organization, all the while giving due attention to human factors, with their psychological needs and inter-personal interactions.

He holds an MSc. in Software Project Management (University of Aberdeen).

Past clients include: Wolters-Kluwer (The Netherlands), CCH Software (United Kingdom), Norsteds-Juridik (Sweden), and IPSOA (Italy).

Wolfram Müller

- Google Plus: https://plus.google.com/105275040409214636652[7]
- Linked In (English): http://www.linkedin.com/pub/wolfram-m%C3%BCller/14/227/126[8]
- XING (German): http://www.xing.com/profile/Wolfram_Mueller[9]
- Site (German): http://speed4projects.net[10]
- Site (English): http://reliable-scrum.info[11]

Wolfram Müller was born in 1969. He studied Mechatronics and Mechanical Engineering. He worked for two years for BARD/angiomed and gained firsthand experience in development and manufacturing of medical devices. At this time he learned how the tools of the traditional project management work and about their

[7] https://plus.google.com/105275040409214636652
[8] http://www.linkedin.com/pub/wolfram-m%C3%BCller/14/227/126
[9] http://www.xing.com/profile/Wolfram_Mueller
[10] http://speed4projects.net
[11] http://reliable-scrum.info

drawbacks. During his studies he was active as a freelancer in various software development projects and discovered the fun of fast projects.

From 2000 to 2010 he worked as a developer and later as manager of the Project Office of the 1&1 Internet AG and was responsible for more than 500 projects. He used ideas out of the Lean, TOC and Agile realms. He implemented Critical Chain in one business unit of 1&1 with great success. Based on this, he developed add-ons to agile Methods to make them compatible with Critical Chain. His focus is always on speed, throughput, reliability and agility.

Since 2005 Wolfram Müller offers his experience to various interested companies in the form of training, coaching and counseling under the label "Speed4Projects".

Disclaimer

Please notice:

- Neither of the authors are native English speakers, so bear with our command of the language. If you spot errors of any kind, we will be happy to receive your feedback and make the corrections. This book is being entirely self-published. We don't have the dedicated resources of a publisher (like copy-editors and other professionals). We trust you will appreciate more the ideas and the concepts we present, rather than the language we might use or misuse.
- Any quotations are made in good faith and on fair use basis, for the purpose of faithfully representing original thoughts and attributing their authors. If you are a copyright holder and feel you are being cited inappropriately, please contact us to clear the matter.

Introduction

This book is about how to manage knowledge-work, and in particular about how to bring a knowledge organization to a state of *hyper-productivity*. Hyper-productive teams and organizations have always existed, both in the business and military fields. Naturally, they are rare statistical outliers. The main focus of this book is how one can build and manage a *hyper-productive knowledge-work organization*, by taking as a source of inspiration the experience matured in the field of software engineering.

Software development organizations — like Microsoft, Google, Oracle; but also high profile open source organizations like Apache, Ubuntu, Drupal and others — can be considered as *archetypal knowledge-based organizations*, because *the totality of artifacts produced by a software business is purely immaterial.* Software development organizations were the first one to confront the challenges of completely immaterial processes, entirely based on knowledge. They were also the first organizations to experience the socio-technological impact of information technology (especially the development of computer networks) on their internal work processes and organizational structures.

Due to its frantic and exponential pace of innovation, the field of software development also became a testbed for a variety of approaches and methods for managing immaterial knowledge work, typically under the form of software development processes and methodologies. Numerous alternatives evolved over a relatively short period, with a lot of healthy competition in between them; with vehement "holy wars" between proponents of one approach or another; and with many anecdotes about both successes and failures.

Valuable insights can be gained by studying the organizational and

managerial approaches successfully adopted by the industry that produced the information revolution — that is, those organizations that engage first hand in software development — and then by extrapolating those organizational and managerial processes to the more general case of knowledge-based organizations.

Nowadays, the majority of organizations have become knowledge-based. Even the most resilient and traditional "brick-and-mortar" businesses are forced to become knowledge-based organizations. For instance, consider the impact of large-scale 3D printing on the construction industry (*"Contour Crafting"*), where bricks and mortar literally become software. Or nanoscale technologies, where manufacturing at the sub-atomic level becomes software.

The key tenet of this book is that organizational hyper-productivity stems from the two *Noble Patterns* of `Unity of Purpose` and `Community of Trust`. The leaders and the top executives of the organization play a critical role in creating the conditions for these patterns to become effective. Management must play an active role in leading the organization towards these conditions.

In the case of software (and knowledge) organizations, management will have to gain a deep understanding of the nature of software or knowledge work; and that nature is rooted in empiricism. Top executives already have that understanding; though they might not be aware of it, and – unfortunately – do not act consequently. Management must create the conditions for the development of a *learning organization*. Financial responsibility must be exercised differently, and in way that is compatible with the empirical nature of knowledge work. Key roles must be nurtured, while each team needs to develop their own shared vision of their purpose.

Practical help comes from many sources, and we will look specifically into, like: Kanban, Scrum and the Theory of Constraints. The Theory of Constraints nurtures `Unity of Purpose` and `Community of Trust` in many ways. In particular, the Theory of Constraints squarely addresses the need to arrive at common metrics by which

all and everybody are driven; and overcomes the structural impediments (rooted in the use of cost accounting and efficiency metrics for management purposes) which are the sources of divergence of purpose and hidden agendas.

Kanban and Scrum are widely used in modern knowledge work businesses. The Theory of Constraint can extend them in powerful ways, bringing more predictability of behavior of the system as a whole, as well as of the individuals. Their combination becomes a powerful breeding ground for the development of `Unity of Purpose` and `Community of Trust`. Both Kanban and Scrum can be extended with features of the Theory of Constraint, and help creating a hyper-productive organization. The book will present practical ways by which such combination might be realized.

Who Should Read this Book, and Why

The book is about organizational hyper-productivity in knowledge-work. During his consulting assignments, Steve usually works with companies in a holistic way, in order to make them arrive at a state of hyper-productivity. For this to be successful, everybody has to be involved, from the CEO to the latest junior hire. Such work typically progresses in two directions, in parallel: top-down and bottom-up. This explains the structure, targets and purpose of the book.

Parts 1 and 2 are aimed at business owners, CEOs or other C-level executives, and top managers. The concern is mostly about understanding about hyper-productivity, and about what needs to be taken into consideration in order to create a new or evolve an existing business into a hyper-productive organization. The focus is organizational and managerial. The objective is to arrive at spectacular increase in financial throughput. This corresponds to the top-down approach.

Parts 3 and 4 are aimed at middle-managers, project-managers, scrum masters, and team members. In particular: practitioners using

(or intending to use) the Kanban Method (by David Anderson), Scrum (by Jeff Sutherland) or CCPM (by Eliyahu Goldratt). The concern is about how to increase "flow" of knowledge work. The focus is on operational and practical. The Objective is to arrive at spectacular increase in operational throughput. This corresponds to the bottom-up approach.

Structure of this Book

This book is divided into the following parts:

Part I - What and Why: This part of the book starts by describing what hyper-productivity is in a software business, presenting the case of Borland International and the Quattro Pro for Windows team. That team of developers was the most productive ever documented. The history of Borland International is taken as an example of how hyper-productivity is a trait that can be acquired, transferred and lost; and hence sets the foundation for making it possible to bring other organizations into a productive state. Considerations are presented about why it might be worthwhile caring about hyper-productivity in a knowledge organization. Two foundational patterns are identified as `Unity of Purpose` and `Community of Trust`.

Part II - Management, Leadership and Organization: This part of the book presents a collection of thoughts about how we can bridge the empirical insight of strategic management with the empirical nature of software development, and examines what kind of management approaches allow to bring empiricism beyond the strategic level, and into the daily exercise of operational management. We will describe a number of management approaches which can contribute in some way to building hyper-productive knowledge organizations. The nature of knowledge work (as software) is examined, as well as how a parallel can be drawn between managing knowledge work and the strategy making activities of

business leaders. The iterative, exploratory, adaptive, and empirical process which is at the root of strategy making, is the same as that which is used in knowledge work; therefore, senior management is at least conceptually well-equipped to manage knowledge work. The key process is a social learning process. Management's responsibility and weaknesses in building a learning organization are highlighted. Known management approaches that help this process are examined, such as: discovery driven planning, beyond budgeting, incremental funding, and throughput accounting. Next, the critical roles of hyper-productive organizations are examined, through the lessons learned from open source software. Also, anti-patterns to hyper-productivity are briefly considered, such as some typical roles of Scrum. The importance of pride in workmanship, fun, and slack are highlighted; as well as practical approaches for creating shared visions at the team level, which then relate to the `Unity of Purpose` at the organizational level.

Part III - In Practice with the Kanban Method: This part of the book goes into practical details about how to manage knowledge work through the Kanban Method. First it is shown how there are strong ties between the Kanban Method and the Theory of Constraints. The Theory of Constraints is one of the most successful approaches for improving the performance of organizations; still, it has yet to be fully exploited in the context of knowledge work. This part of the book intends to show that a combination between the Kanban Method and the Theory of Constraints is actually an excellent one. On the one side, the Kanban Method provides the tools for dealing with knowledge work management. On the other side, the Theory of Constraints provides the focus — especially in terms of determining `Unity of Purpose` — and the leverage that will allow an organization to continuously improve towards ever higher levels of performance. Some fundamental drawbacks of common Kanban implementations are also addressed, and resolved through the application of the Theory of Constraints.

Part IV - In Practice with Scrum: This part of the book, similarly

to the previous one, is of practical nature and shows how the significant improvements in productivity can also be achieved by starting with Scrum rather than with the Kanban Method. Here again, the Theory of Constraints is used as the catalyst that enables continuous performance improvement. Organizational change is considered from the point of view of the Theory of Constraints, and especially how it relates to the flow of work throughout a knowledge-based organization. The nature of constraints is explained, and also where and how to start an improvement initiative, going from major releases to backlogs. Execution of control becomes paramount to balancing resources against demand and achieving due-date performance, ultimately reinforcing the `Community of Trust` that is grown within the organization. Specific tools of the Theory of Constraints — like Drum-Buffer-Rope scheduling and Critical-Chain buffer management — are described in a way that further enhances Scrum.

1 Understanding the Impact of a Constraint

The reason why work state WIP limits are a problem, is that due consideration is not given to the real constraint of the system. One might argue that the approach does indeed produce improvements (simply by hitting the constraint by happenstance, as we saw) and therefore there is no greater reason to change it. However, before jumping to such a conclusion, it might be useful to gain a better understanding of the impact of acting on a constraint or on a non-constraint.

In this chapter we will revisit throughput accounting, or more precisely, look into the significance of **Constraints Accounting** [CASPARI-2004[1]]. The purpose is to gain a deeper understanding of the magnitude of impact that a constraint can have on business operations.

1.1 Choosing between XP and BDD

For the sake of illustration, let us consider a small software house that engages in bespoke custom-software development. The business owner has asked his people to examine their operations and come with suggestions to make improvements. Two proposals are ultimately contending for a positive verdict. One is proposing the adoption of a method called XP and another one is proposing a method called BDD[2]. The business owner is not versed in techni-

[1] http://chronologist.com/bibliography#CASPARI-2004

[2] These abbreviations stand for *eXtreme Programming* and *Behavioral Driven Development*. What the two approaches actually prescribe and how they work is irrelevant in this discussion, as we are focusing only on the business consequences of choosing one or the other.

calities, so those acronyms bear little significance for him.

The two options come with similar investment requirements. Both would need an investment of 5,000 EUR to purchase new equipment (office furniture, workstations, computer hardware, etc.). The question is which option should be taken.

PROPOSED INVESTMENT	Current	XP	BDD
Furniture / Workstation / Hardware		5,000.00	5,000.00

Proposed Investment

1.2 The Lean Perspective

After fierce debates that left the question unsolved, someone from the marketing department asks if it might not be possible to measure and compare one approach against the other; and maybe do some kind of *A/B Testing*. The business owner likes the idea, but wonders how the two options can be measured, without actually carry them out. The suggestion is to call in a *Lean* process expert to quantify the approaches in order to make a significant comparison.

AVERAGE CYCLE TIME (DAYS)	Current	XP	BDD
Days	7.00	6.70	7.30

Average Cycle Time

After doing detailed *Value Stream Mapping* the expert runs *Monte Carlo Simulations* to gain a quantitative understanding of the two alternatives. He comes back with his metrics. He discovered, through historical data, that the current process was exhibiting (on

average) a cycle time of 7 working days to deliver a work item[3]. The runs of several thousands of simulations showed that XP would *reduce* that average cycle time to 6.7 working days, while BDD would *increase* it to 7.3 working days.

The *Lean* process expert wholeheartedly recommends choosing XP over BDD.

1.3 The Accounting Perspective

The business owner, being wise, asked his accountant to look into those figures and confirm that the *Lean* process expert's recommendation was sound.

[3]In this case a "work item" can be considered as a user story, a use case, etc. which — according to Kanban practices — has been classified according to a class of service which is associated with an average delivery time. Naturally the example is a simplification, as it considers only one kind of work item, which is assumed to have a (reasonably) consistent size. In the real world the situation would be different; but the *principles* exposed through this example would still remain valid.

INITIAL DATA	Current
General Expenses / Year	**150,000.00**
Work Items / Year	**75.00**
Average Item Price	**16,000.00**
Average Item Acquisition Cost	**80.00**
Hourly Wage	**22.00**
Work Hours / Day	**8.00**
Work Days / Week	**5.00**
Work Weeks / Year	**48.00**
Work Hours / Year	1,920.00
Work Days / Year	240.00
Daily Wage	176.00

Initial Data

The accountant started by looking at the data he knew about: the overhead (expenses) on a yearly basis, how many work items were delivered (on average) every year, the price of each work item, the average acquisition cost of each item (i.e. the cost of finding a requirement), the hourly wages, and so on.

LABOR AND CHARGING RATES (DAILY)	Current	XP	BDD
Labor	176.00	176.00	176.00
Overhead	625.00	625.00	625.00
Total Daily Labor and Charges	801.00	801.00	801.00

Labor Charging

Then the accountant started making some calculations on the back of an envelope. He was trying to figure out what was most

convenient to do on a daily basis. The daily labor rate of one resource (the process always had exactly one employee working at any given stage) was the hourly wage (22 EUR) times the hours per day (8), giving 176 EUR per day. The overhead charging per day was computed dividing the yearly overhead (150,000 EUR) by the number of working days during the year (240) giving a charge of 625 EUR per day. So the total daily charge, including labor and overhead, was 176 + 625 EUR = 801 EUR per day. These figures were the same also for the two cases, XP and BDD, under examination.

STANDARD UNIT COST	Current	XP	BDD
Acquisition	80.00	80.00	80.00
Direct Labor	1,232.00	1,179.20	1,284.80
Overhead	4,375.00	4,187.50	4,562.50
Standard Unit Cost	5,687.00	5,446.70	5,927.30

Standard Unit Cost

The next step was to determine how much a single work item would cost. It was known that it took an average of 7 working days to deliver a work item. So it had to incur in a labor cost of 176 EUR * 7 = 1,232 EUR. Similarly it had to incur in overhead costs of 625 EUR * 7 = 4,375 EUR. Furthermore, the acquisition cost of 80 EUR had to be considered. Summing these numbers together (1,232 + 4,735 + 80), the standard unit cost was determined as 5,687 EUR. In other words, any single work item would cost the company 5,687 EUR.

The same calculations were done for the two cases, with the difference in the average cycle times. The results showed that XP could be associated with a standard unit cost of 5,446.70 EUR, while BDD came in at 5,927.30 EUR.

STANDARD UNIT COST VARIATION	Current	XP	BDD
Standard Unit Cost Variation		-240.30	240.30

Standard Unit Cost Variation

With these results, the standard unit cost variation, compared to the current situation was easy to find. XP had a standard unit cost of *minus* 240.30 EUR (than the current case), while BDD had the same amount more. (Coincidentally the two cases have the same magnitude because it is the monetary quantification of the *minus* 0.3 days and the *plus* 0.3 days on the cycle time of the two cases; the time variation happens to be of the same magnitude and changes only in sign).

ANNUAL COST VARIATION	Current	XP	BDD
Cost Variation		-18,022.50	18,022.50
Cost Variation First Year		-13,022.50	23,022.50

Annual Cost Variation

Taking the standard unit cost variation (+/- 240.30 EUR) and multiplying by the number of units produced (75), the annual cost variation was determined in -18,022.50 EUR for XP and +18,022.50 EUR for BDD.

Naturally the accountant was quick to discard BDD as it implied an *increase in cost*, while XP was coming with a *cost saving*.

INTERNAL RATE OF RETURN	Current	XP	BDD
Net Cash Flow		18,022.50	
Investment		5,000.00	
Internal Rate of Return		360%	
Payback Period (Days)		101.26	

Internal Rate of Return

To further reinforce his conclusion, the accountant did a quick (approximate) calculation of the *internal rate of return* and of the *payback* period of the investment to adopt XP. The net cash flow corresponded to the cost savings; the investment was 5,000 EUR. The calculation showed an amazing 360% return on investment and the payback period was just over 100 days, more or less 5 months.

The accountant wholeheartedly recommended the adoption of XP.

1.4 The Constraints Management Perspective

The business owner was happy to have received such good advice from the *Lean* process expert corroborated by the numbers provided by the accountant. He broke the news to his senior developer, Herbie, who had been with the company from the very beginning. He could see from his expression, that he was not happy. So he asked why, not after having stated again how good the choice was: it reduced cycle time and provided a real positive impact on the bottom line.

He was even more surprised when Herbie, whom he respected for his great technical skills and past contributions, said that the choice was really throwing away money, rather than making any. So in a patronizing tone, he challenged Herbie to show him that the *Lean* process expert and the accountant were mistaken.

Herbie, having lived through many such "sound business choices" in the past, took the chance to explain his viewpoint, now when the business owner was willing to listen.

AVERAGE CYCLE TIME (DAYS)	Current	XP	BDD
Total Cycle Time	7.00	6.70	7.30
Analysis -- Bluey	**1.80**	**1.50**	**2.40**
Development -- Herbie	**3.20**	**3.40**	**2.70**
Test -- Greenie	**2.00**	**1.80**	**2.20**

Average Cycle Time by Resource

It was true that the XP option had a lower cycle time, but neither the *Lean* process expert nor the accountant had taken into consideration the effect of the system's constraint. By breaking down the cycle time into its constituent parts, the work states of analysis, development and test, Herbie pointed out that in the current state he himself was the constraint. On average a work item took 3.20 days on his desk; while it took only 1.8 upstream in analysis and 2.0 days downstream in test. Obviously he was the constraint.

The XP solution would actually increase the time required by Herbie from 3.20 to 3.40 days for each work item. In other words, the constraint would become loaded even more. On the other hand the BDD solution would lower Herbie's load from 3.40 to 2.70 days for each work item (yet still remaining the constraint of the system). This fact alone was enough for Herbie to draw his conclusion; but he knew he had to convince the business owner with more hard data and especially he had to convince the accountant with money numbers.

CAPACITY IMPACT / YEAR	Current	XP	BDD
Available Work Days	240.00	240.00	240.00
Time on "Herbie"	3.20	3.40	2.70
Capacity	75.00	70.59	88.89

Capacity Impact

Herbie started to show his boss that the impact of changing had to be examined in terms of how things changed for the constraint. While the number of working days in a year was always the same, the fact that Herbie's time — the constraint's time — on the work items varied, actually changed the *capacity* of the entire system.

The accountant had wrongly assumed that the capacity was a given (75 work items per year) and made calculations accordingly; but capacity changes, because the cycle time on the constraint changes. In the case of XP, capacity decreases from 75 work items per year, to 70.59; that is the 240 days divided by the 3.40 days per work item. Conversely, in the case of BDD, capacity increases to 88.89; that is 240 divided by 2.70.

THROUGHPUT / WORK ITEM	Current	XP	BDD
Work Item Price	16,000.00	16,000.00	16,000.00
Variable Cost	80.00	80.00	80.00
Throughput	15,920.00	15,920.00	15,920.00

Unit Throughput

The throughput in terms of money per work item is the same in all three cases: The work price minus the totally variable cost, or 16,000 EUR minus the 80 EUR for the cost of acquisition, resulting in 15,920 EUR.

TOTAL THROUGHPUT VARIATION	Current	XP	BDD
Capacity Variation		-4.41	13.89
Throughput Variation		-70,235.29	221,111.11

Throughput Variation

The XP option produces 70.59 minus 75.00 that equals 4.41 work items *less* each year. Multiplying those 4.41 work items by the unit throughput of 15,920 the result is 70,235.29 EUR *less* each year.

Conversely the BDD option produced 88.80 minus 75 that equals 13.89 work items *more* each year. Multiplying those 13.89 work items by the unit throughput of 15,920 the result is 221,111.111 EUR *more* each year.

VARIATIONS IN T, I, OE, ROI	Current	XP			BDD	
		First Year	Subsequent		First Year	Subsequent
TP		-70,235.29	-70,235.29		221,111.11	221,111.11
I		5,000.00	no change		5,000.00	no change
OE		no change	no change		no change	no change
Cash Flow (TP-I-OE)		-75,235.29	-70,235.29		216,111.11	221,111.11
ROI ((TP-OE)/I)		-1404.71%			4422.22%	

Variation in Global Measures (TP, I, and OE)

The business owner was astonished; but Herbie continued with the numbers, considering even the effects of the investment and of operating expenses. He showed that the net cash flow on the first year was a *negative* 75.235.29 EUR for the XP option, while it was a *positive* 216.11.11 EUR for the BDD option.

What the accountant had computed as a 360% return on investment for the XP options was in reality a 14 *hundred* percent loss on investment. While the option that had been discarded on the basis of increasing costs was in reality a 44 *hundred* percent return on the investment.

RANGE ON BOTTOM LINE PROFITABILITY	Current	XP	BDD	Range
Cost Accounting		13,022.50	-23,022.50	**36,045.00**
Global Measures (T, I, OE)		-75,235.29	216,111.11	**291,346.41**

Range on Bottom Line Profitability

What was originally seen as a simple decision and minor improvement costing 5,000 EUR returning an annual benefit of approximately 18,000 EUR in reality would cause a net loss of over 75,000 EUR the first year alone, and over 70,000 EUR every year thereafter. Comparing the two choices, we discover a bottom-line profitability effect that spans almost 300,000 EUR. What's worse, is that the cost accounting metrics induces selecting the option that actually makes you *lose* money and will make you discard the option that brings the real profit.

The business owner was astounded, and reconsidered his decision.

> Note: in addition to the financial throughput effect, you should also notice that in this particular example the better choice was the one with the *longer* cycle time. In most cases where Kanban is used, there is often an effort on reducing cycle times. This example highlights that cycle time reduction is not always beneficial, unless it specifically includes cycle time reduction on the constraint. Also note that there might be other reasons for reducing cycle time. One is when service time needs to be improved in order to increase customer satisfaction. Another is when you need to increase the capacity of the non-constraints in order to *maintain* the current constraint in its place.

1.5 Constraints are Archimedean Levers

Naturally the above example must in no way be interpreted as a value statement of either XP or BDD. The two techniques were used only for the sake of illustration. In a real world scenario, the true measures of cycle times would determine which of the two options actually improves the constraint; and the constraint is always dependent on the particular situation and context you are examining.

Similarly, the example should not be deceitfully represented to mean that cycle time reduction is bad and cycle time increase is good. There is no way to tell if a positive or negative cycle time variation is good or bad unless you carefully analyze its impact on the constraint. A variation (whether positive or negative) on total cycle time will have a positive impact on throughput only if it involves a reduction of cycle time on the constraint; as this example illustrates, that can happen even if it means increasing total cycle time.

The point to understand is that the effect of any decision must be measured in terms of its impact on the system's constraint. Constraints are lever points, whereupon one can exercise an Archimedean lever in term of financial result. The above example, even if devised for the purpose of illustration, shows this very clearly.

Constraints Management is Key to Throughput Performance

It follows that it becomes paramount to know exactly where the constraint is in the system. This is the main reason why work state WIP limits are a danger: because they give so many false positives about where the constraint really is. With work state WIP limits it becomes impossible to exercise effective *constraints management* and hence very unlikely to gain the lever effects that would benefit

your decisions otherwise. Work state WIP limits might give the impression of achieving improvements, while in reality they are a random walk into the future. In that randomness, real improvement is coincidental when the real constraint happens to be affected by the intervention. What is worse is that that randomness causes an *artificial system instability*.

This is also one of several reasons why it is preferable to have a well-known constraint in a well-known place, rather than elevating it's capacity so that the constraints move to some other undefined and unidentified place in the system. *As long as the constraint is under control, then it is possible to control and manage flow.* When it moves, you will have an unstable system and you have to start over again (*Step 1: identify the constraint!*); chasing and pinning down the new constraint. This period of instability — between the elevation of an old constraint and the identification of the next one — usually has very bad effects in terms of flow.

Constraints and SLAs

In Kanban, classes of service are naturally associated with average cycle times in order to be able to maintain service level agreements (SLAs) with stakeholders and customers. Because work state WIP limits adversely affect the systems stability they artificially increase the average cycle times which are used as the basis for establishing SLAs. In other words, this artificially induced system instability will make the organization settle for *less competitive SLAs* than what the organization is truly capable of sustaining. Keeping the system stable is paramount for setting aggressive service levels without straining the system and without reducing reliability.

Constraints and Investment Decisions

A strong focus on constraints management will also give clear guidance in terms of *when* an investment should be undertaken. In-

vestments should be aimed exclusively either to establish necessary conditions of operations or to elevate constraints. The first kind of investment are usually taken when a business is starting up. Often they are a one-time investments. The second kind of investment (to elevate the constraint) is taken in the light of achieving a considerable increase in profits, which are inevitably associated with the elevation of a constraint.

It is important to realize the corollary: money should not be spent in order to subordinate to or to exploit the constraint. Subordination and exploitation are simply reconfiguration of current resources and policies, which do *not* require any investment. However, subordination and exploitation can significantly impact the company's profit, because of the increase in the throughput they produce. This is one way that the constraints management improves financial performance *without* requiring new expenses.

When a decision is taken to effectively elevate a constraint, it is often worthwhile to consider elevating the capacity of other resources too, for the *only* purpose of keeping the constraint where it was previously. It is like raising the bar for Herbie, while Herbie still remains the constraint. The intent is to preempt moving the constraint to somewhere else in the system; the purpose is to avoid the instability that comes out of moving the constraint. So the general advice is to aim investments not only at elevating the constraint but also to actively keep the constraint in the same place, while doing so. As long as the system remains stable, then the assumption of Little's Laws can be applied and aggressive service level agreements can be maintained.

When using WIP work state limits, it is never clear if you are confronting the real constraint or not. You might be induced to spend money and invest on elevating some false constraint. You will not only have zero impact on the system's real throughput, but you will also impoverish the company of the spent investment.

So the key point becomes the following: if work state WIP limits are

so counterproductive, what else can be used? How can we handle constraints in a way that is coherent with the visual management paradigm promoted by the Kanban Method; and yet enable all the constraint management techniques for improved organizational performance which are typical of the Theory of Constraints?

2 Improving While In the Flow

In this Chapter we will see how you can improve your work *process* all the while the work *flow* is flowing regularly at a sustainable pace. This technique, which combines ideas from the Kanban Method, the Incremental Funding Method (IFM) and the Critical Chain Project Management (CCPM) of the Theory of Constraints, will also provide the foundation for doing deep root cause analysis (as we will see in the Chapter Root Cause Analysis the TOC Way).

2.1 Minimum Marketable Releases (MMR)

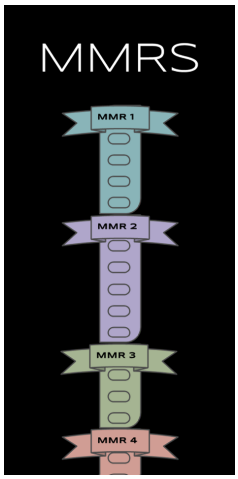
The **Incremental Funding Method** [DENNE-2004a[1]] first introduced the concept of the **Minimum Marketable Feature** (MMF), which is a *minimal* set of features delivering market value.

In Kanban the MMF concept has been revived and is known as a **Minimum-Marketable Release** (MMR). David Anderson [ANDERSON-2010[2]] defines an MMR as *"some atomic unit of value to the market or customer."* For all practical means, the two concepts are equivalent and can be considered as synonyms, because as we will see, the important traits are in *minimality* and *marketability*. Kanban does not prescribe the use of MMRs, but suggests that they *can* be used. At times, because Kanban tries to emphasize continuous flow, the use of MMRs is discouraged because it is perceived as a disruption of flow. Also MMRs might be discouraged as they appear

[1] http://chronologist/bibliography#DENNE-2004
[2] http://chronologist/bibliography/#ANDERSON-2010

to imply *early* commitment to the features they include. However, with respect to the issue of early commitment, if one considers MMRs as *atomic* units of value, then the commitment is about that value, about MMRs as wholes and not about the single work items that may go into an MMR.

If the pool of work to be done is considered as a backlog (like in Scrum and other agile approaches), then MMRs are simply a *partitioning* of that pool into subsets, as illustrated in the figure. It is useful to think about MMRs in this way.



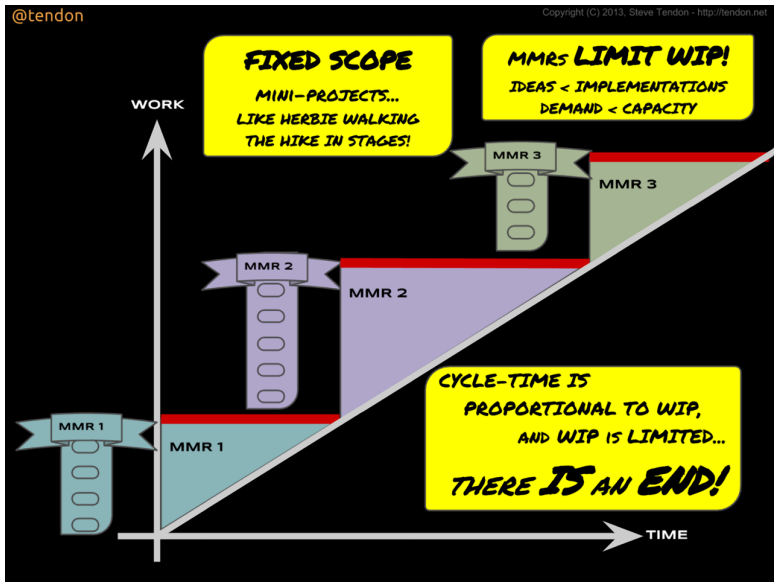
Minimum Marketable Releases

IFM teaches how to find the *optimal sequence of implementation* of such subsets in order to maximize the *net present value* of a project. While we might apply the techniques of the IFM, we consider such sequencing techniques as optional. In fact, in Kanban, pull selection is more appropriately handled with consideration to cost of delay and other risk factors. The focus of the present discussion, however, is not about *pull selection strategies*. It is about the traits of MMRs that render them invaluable from a risk management and process improvement perspective, when combined with other elements of the TOC.

2.2 Minimum Marketable Release as a WIP-Limiting Unit of Work

The first reflection is that an MMR is a *unit of work* which is a collection — a batch, effectively — of work items. The single work items can belong to different classes of service, as necessary. The key tenet is that all work items belonging to an MMR must be delivered

together, as a single unit. We will advantageously consider an MMR primarily as a *unit of work that is released into the work flow* and secondarily as a *unit of value that is released to the market* once finished.



MMRs Limit WIP!

The second reflection is about the notion that an MMR represents the *minimal amount of functionality* that can successfully be offered to the market. *Minimality* and *marketability* together have a very important consequence: an MMR becomes like a small *fixed scope project.*

By definition of MMR, you cannot take work items away from it, or it would lose its marketability. Analogously, you cannot add items to an MMR, or it would no longer be minimal. Therefore, since you cannot take away nor add items, the MMR is similar to a fixed scope mini project. A MMR can be considered as a *fixed scope unit of work with marketable value,* which must be delivered in its entirety. No more, no less.

A MMR is like a Fixed Scope Project

We put a strong emphasis on **Fixed Scope**: you cannot change an MMR's scope, lest the very purpose of defining and identifying the MMR in the first place becomes vain. (Naturally, if there was an error in the initial definition of the MMR, then it has to be changed; but this is another issue all together. Let's assume that the scope definition is appropriate and correct.)

A MMR Limits Work In Process

On a cumulative work flow diagram, the MMR appears as a demand line that is horizontal. In other words, *work in process is limited from the outset.* The capacity line grows progressively until it intercepts the demand line. At that point all work has been performed and can be delivered.

In virtue of Little's Law, we can assume that there is a *finite cycle time* for the delivery of the MMR. In other words, because the MMR interrupts an (ideal) continuous flow of work, we have a future time point when the MMR will be delivered. (We will see shortly how to arrive at that time point.)

Notwithstanding that an MMR interrupts continuous flow of work, it brings the significant advantage of *limiting* work in process; and limiting work in process is a fundamental control mechanism that favors *better* flow.

2.3 Manage Risk by Varying Time, Not Scope

Kanban does not require explicit risk management because there is a lot of risk handling capabilities that are "built-in," and come

"for free" even with the most shallow adoption Kanban. The event-driven risk management that is enabled by work state WIP limits of Kanban (notwithstanding their many disadvantages mentioned several times in this work) or by the signals of Hyper-Kanban will adequately take care of most *special cause variation* risks.

While not prescribing risk management, Kanban strongly suggests that explicit risk management should be performed nonetheless. On the other hand, the Incremental Funding Method, where MMFs where first conceptualized, *mandates* explicit risk management precisely because of the nature (fixed scope!) of MMFs.

It follows to reason that if you use Kanban with MMRs — which are fixed scoped too — then explicit risk management becomes mandatory.

Furthermore, IFM mandates explicit risk management, but does not prescribe or define how to realize it. Hence it becomes interesting to see how risk can be managed.

"Cutting the Backlog" Does Not Cut It

One should notice, at this point, that the most common and popular agile risk management approach of *"cutting the backlog"* cannot be applied. Cutting the backlog does not work with a fixed scope MMR, because of the properties of minimality and marketability, as already explained. Risk must be controlled in some other way.

Lessons from Critical Chain Project Management

If we cannot vary the scope because of the nature of the MMR, we have to get better at managing *time*. In fact, it is possible to *deal with risk by considering time*! This is where we can be inspired by the Critical Chain Project Management approach of the Theory of Constraints. Let's get familiar with CCPM.

Critical Chain Project Management (CCPM) is characterized by a network of project activities, just like the traditional Critical Path Method. The *Critical Path Method* (CPM) is basically the method that is supported by the majority of project management software packages like Microsoft Project.

Unlike the Critical Path Method, CCPM gives explicit attention to *resource dependencies* in addition to *task dependencies*. (Without resource contention, then CCPM and CPM would essentially be the same, as long as the project network is concerned.)

In CCPM the **Critical Chain** is defined by [SULLIVAN-2012[3]] as

> *The longest sequence of dependent events through a project network considering both task and resource dependencies in completing the project. The critical chain is the constraint of a project.*

One key insight here is the following: no matter what kind of process or methodology is employed, at the end of the project all work must have traversed a *"network of tasks executed by resources."* An undeniable fact remains: to get to the final delivery, the project network must be traversed and therein the Critical Chain is still the constraint limiting the project's capability to deliver, no matter what methodology or approach is used. Even when using Kanban.

In CCPM, the project network is used for *planning* the project, just like in the Critical Path Method; but the logic used to schedule the activities is different. For our purposes, *the project network of CCPM is uninteresting*. We simply assume that the activities involved in delivering an MMR are equivalent to the activities represented inside such a project network.

The other major distinguishing difference between CCPM and CPM is that in CCPM a single **Project Buffer** is placed at the end of

[3] http://chronologist.com/bibliography#SULLIVAN-2012

the project network. In CPM "padding" typically occurs at the task level.

This project buffer is a *time buffer*. In Kanban we are much concerned about time aspects: ordinarily you use lead time and cycle time metrics for many purposes, not the least for establishing the service level agreements associated with classes of service. Shortly we will see how we will make use of this observation.

TOC teaches us to identify, protect and elevate the constraint. In the project network the constraint (i.e. the critical chain), is protected and managed by constantly monitoring the project buffer. A superficial (and wrong) explanation of why the project buffer is needed, states that it is there in order to absorb any variability that might adversely affect project delivery. As we we will see, that is not the case because there is more to it. The project buffer's main benefit comes from continuously monitoring in order to get *leading* and *operational* signals about oncoming trouble that might affect the health of the project.

So the essential part that we preserve from CCPM is the project buffer at the end of the project network; we do not care about the scheduling aspects of CCPM and the actual building of the project network. Unlike the network that is used during *planning*, the project buffer is used during the *execution* of the project. We will execute all planning activities as suggested by Kanban. What we will transfer over from CCPM to Kanban is the interpretation and use of the project buffer. We will use the project buffer, exactly in the same way as it is done in CCPM and thus getting the well documented advantages typical of CCPM project execution.

Such advantages include:

- Leading indicators of imminent risk materialization
- Just-in-time risk registry compilation
- Frequency Analysis to identify sources of common cause variation

- Root-cause analysis identifying the source of those problems that are most expensive or frequent
- Process of ongoing improvement driven by focusing and leveraging solutions to those expensive or frequent problems

The most significant outcome of this approach is that it will allow you to improve the process you are using, resulting in increased throughput. You will increase the capability of the team and/or process, with improvements that will be beneficial not only for the project you are working on at the moment, but also for all other, future projects you will undertake. *The improvements are systemic.*

The Best of Two Worlds

It is important to notice that the improvements induced by this approach are not replacing those that you achieve with Kanban or Hyper-Kanban alone, because the two are complementary. One completes the other in a significant way. More specifically, you achieves the benefits of both like this:

- The event-driven risk management typical of Kanban or Hyper-Kanban that allows you to react very quickly to special cause variation (either through work state WIP limits or through Hyper-Kanban signals); and
- The buffer-monitoring driven risk management typical of CCPM that allows you not only to identify sources of common cause variation, but to discern and select those which are adversely affecting the team's and/or the process's capabilities and start improvement initiatives around those.

Likewise, this proposal can be seen from the Theory of Constraints perspective. Software development has seen a proliferation of Agile methods in the last 15-20 years, which are now being adopted more

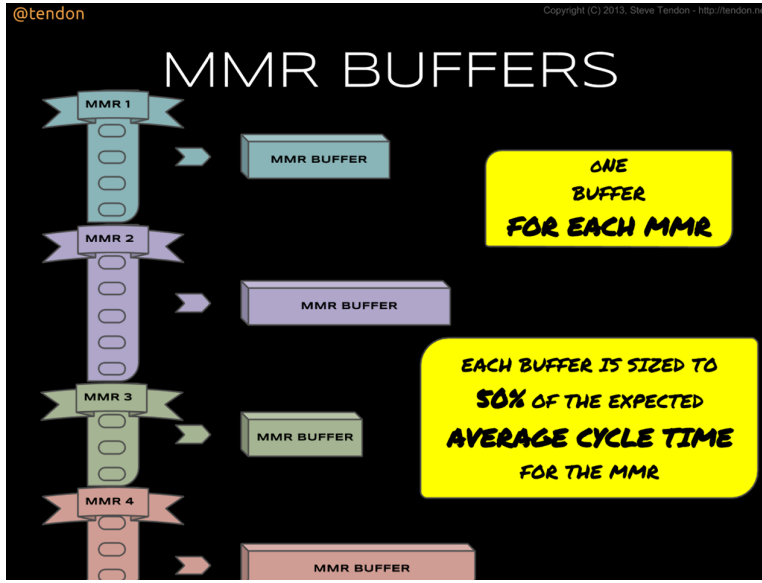
generally by all sorts of knowledge workers. CCPM is in many ways anchored to approaches which cannot benefit from the advances seen with Agile methods or other modern management methods. Therefore, this approach can be seen as an enhancement that introduces Kanban or Hyper-Kanban within the Theory of Constraints' operational practices for project execution (Buffer Management).

TOC practitioners will gain the complementary advantages offered by Kanban / Hyper-Kanban which are most significant in their event-driven risk management capabilities. Moreover, TOC practitioners will recognize and appreciate that the principles, practices and values of Kanban are surprisingly aligned with what Eliyahu Goldratt defined as the *Ever Flourishing Company*.

2.4 The MMR Buffer

As CCPM protects the activities on the critical chain with a project (time) buffer, likewise, we will protect the implementation of an MMR by adding a buffer to its end and constantly monitoring it.

When partitioning a project into MMRs, we can consider each MMR as a fixed scope project. For any single one of those MMRs, we know in advance how much work we have to perform. By adding up the average cycle times of the work items that belong to an MMR, we get an expected average cycle time for the corresponding MMR.

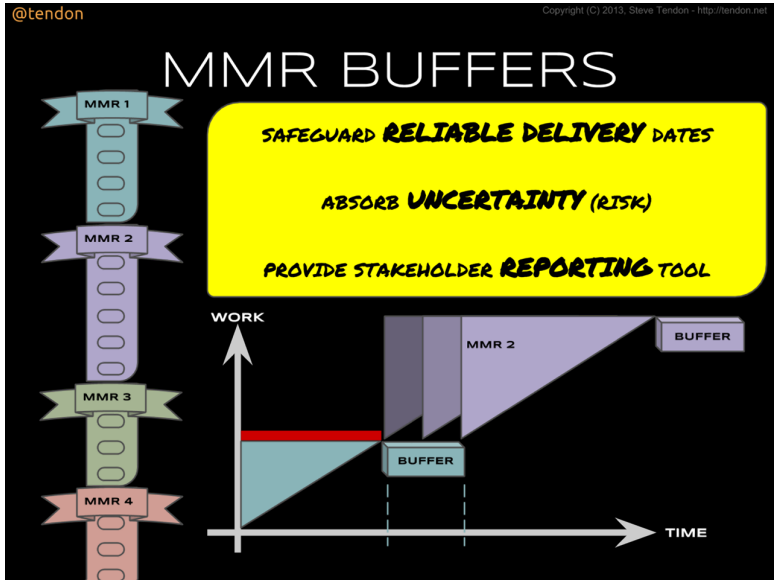


A Buffer is Associated with Each MMR

Naturally, because MMR vary in size, each one will have a buffer of different size. This also means that we are actually expecting the start time of work on subsequent MMRs to vary, depending on whether we finish an MMR sooner or later.

Notice that this extra buffer is *not* a kind of *padding* that we add "just-in-case" something goes wrong. Reinertsen [REINERTSEN-2009[4]] refers to this as principle *"V11: The Buffer Principle: buffers trade money for variability reduction"*. It is typical of CPM, where "padding" (deliberate or not) is used to "buy safety."

[4] http://chronologist.com/bibliography#REINERTSEN-2009



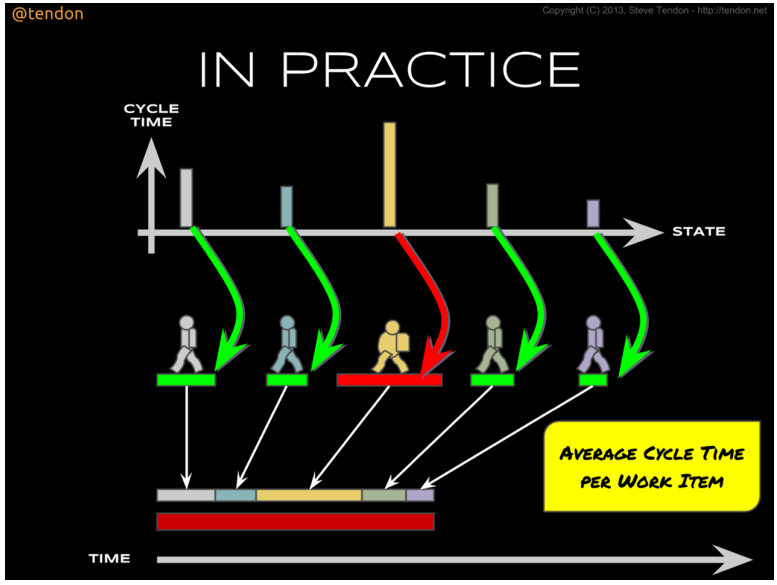
MMR Cycle Time with Buffer

In CCPM (and in our approach), the buffer is not there for this purpose, although it is often presented like this and can even be (mis)used that way. The primary purpose of the MMR buffer is to provide an operational monitoring tool that gives significant leading signals of oncoming risk materialization. The Critical Chain is the constraint and the constraint has to be exploited to the most; the objective in a CCPM project or in our approach is to deliver any work item *as soon as possible*. The buffer is not used to *reduce variability*, as in the case of padding, but to *detect unfavorable variability* early enough to be able to take any necessary counter-measure in time before it affects the whole project adversely. The buffer is sized in a deliberate manner; and, more important, it is *actively used* to steer the execution of the project and get *signals* of oncoming trouble. In order to size the buffer, we take the simple heuristic of dividing the expected cycle time in half. The reason why this makes sense is explained by CCPM and we will look into the rationale of it in the following section.

This buffer is important from a hyper-productivity perspective too: it is not only about absorbing uncertainty or managing risk, but also about building of a `Community of Trust` by making project delivery more reliable.

2.5 Buffer Sizing

You must choose the size of the buffer carefully. To start, consider the expected average cycle time from historical flow metrics. Conceptually, sizing the MMR buffer is like going in the opposite direction of what happens in Critical Chain Project Management. In CCPM, you halve the Critical Path time, to get a 50% due date probability; then you add the project buffer to the end of that.

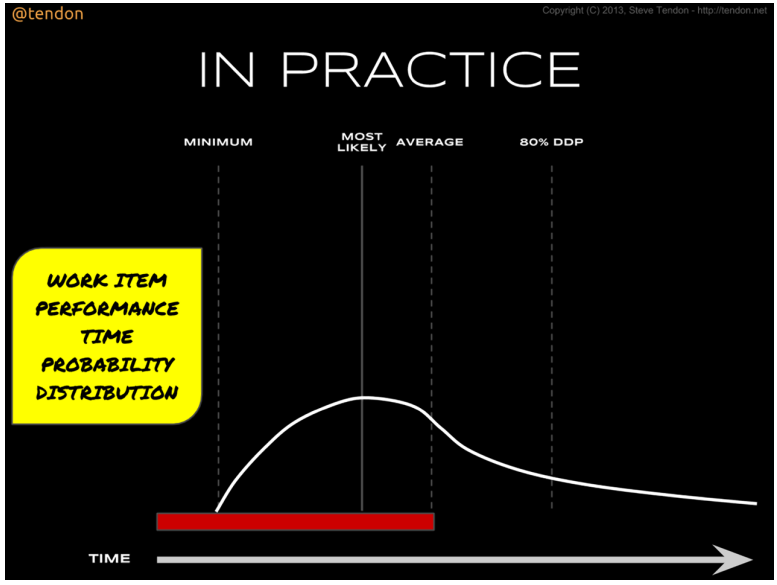


Average Cycle Time for Work Item

In the case of the MMR Buffer, we start with the historical data of average cycle times. Let's consider one single work item that

might be included in the MMR. The work item will belong to a predetermined class of service and consequently it has an expected average cycle time (the average cycle time associated with that class of service).

The probability distribution of cycle times is often skewed (as illustrated in the figure). The "average" time is not the most likely time. In most cases, you will need less than the average cycle time to deliver any work item. We then have a few work items that take "much longer" than the average time. That's why we get the skewed distribution.

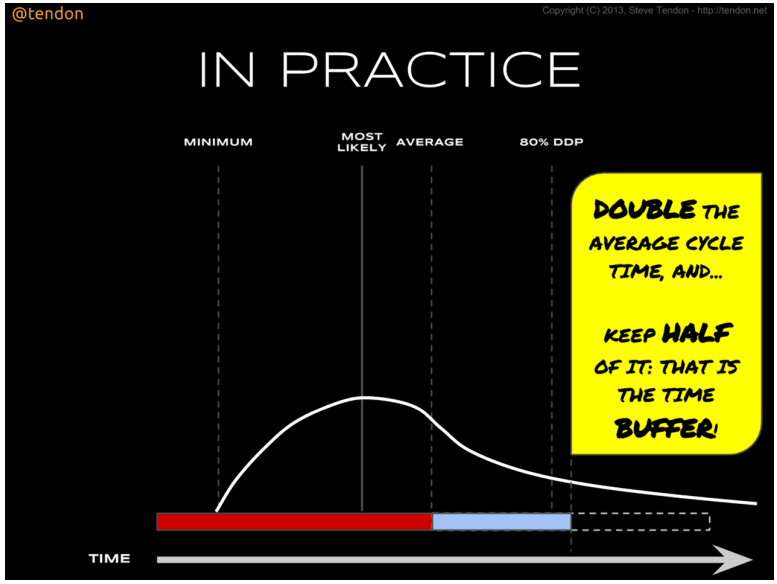


Cycle Time Probability Distribution

Another consequence of the fact that the distribution is skewed is that you will have a higher probability of due date performance the more you progress along the time axis. The idea is to add a buffer to the average cycle time in order to reach a time point where we have very high due date performance.

As a first approximation we can size the buffer to half of that average expected cycle time. By adding half the average cycle time, it is very likely that we are already reaching into the areas of very high probability of on time delivery, in the order of 80% due date performance or more.

We will see how this buffer (like the buffer in Hyper-Kanban) will be used to give significant steering signals. It can be worthwhile to experiment with different sizing techniques for the MMR buffer, because the size of the buffer is critical for the effectiveness of the signals.



Buffer Reaches into High Due date Performance Probability

If the buffer is too big, signals come too late. If the buffer is too small, signals come too frequently, with many "false positives." Eventually, one can use the advanced dynamic buffer sizing technique of the Theory of Constraints to get the optimal buffer size with respect to your project's unique characteristics.

Determining the size of the buffer is of critical importance; and it can be sized in many ways.

For example: [GEEKIE-2006[5]] reviews seven common methods. [FALLAH-2010[6]] uses uncertainty for sizing the buffer. [COX-2010[7]] describes *dynamic buffer sizing*, where the size of the buffer is changed dynamically during the due course of the project.

While knowing about advanced buffer sizing techniques can be very useful, it is also out of scope of (and inconsequential to) the presentation of the concept here. What really matters is the approach that the buffer *enables* in terms of *risk management*.

In more practical terms, the buffer must be *"appropriately sized"* — The *"slashing in half"* mentioned above, is just a gross simplification, used primarily as a simplifying heuristic, that works surprisingly well in most cases.

> **Aggressive Buffer Sizing with Hyper-Kanban**. If you are using Hyper-Kanban, you are already tracking the average cycle times of all work states because we use those averages to find "Herbie" (the constraint) in the work flow. In this case, rather than taking the average cycle time of the work flow, you can take half of the average cycle time *on the constraint*. This will be a much smaller buffer; and it might be too small to be workable. However, it gives you a good heuristic. You can take the halves of both cycle times and consider them as extreme points for determining your buffer sizing. In other words, you might find a good buffer size somewhere in between.

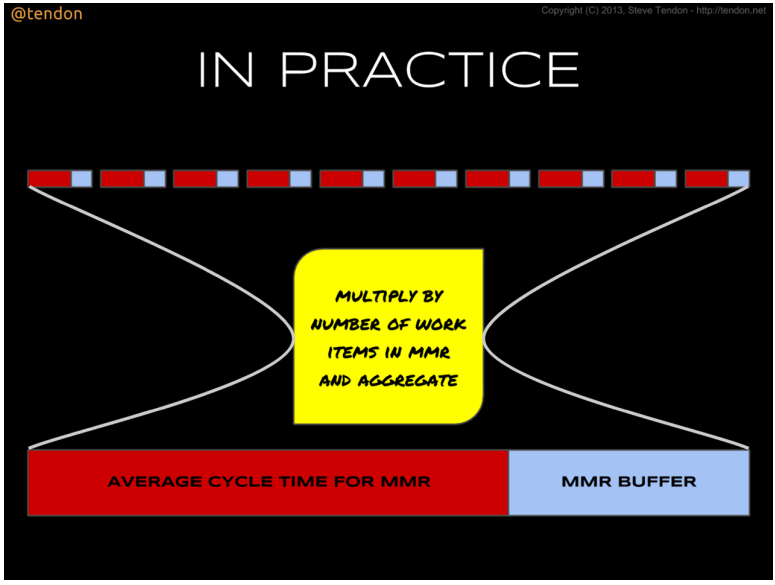
Once the average cycle times and buffers of all work items in

[5] http://chronologist/bibliography#GEEKIE-2006
[6] http://chronologist/bibliography#FALLAH-2010
[7] http://chronologist/bibliography#COX-2010

the MMR have been determined, they are aggregated to find the corresponding cycle time and buffer for the MMR.

Remember that work items can be of different sizes and belong to different classes of service; all of which might be associated with their own average cycle time. You must take this into account when sizing the buffer.

(Note though that, for the sake of simple illustration, the figure shows work items belonging to just one and same class of service and therefore they are all showing the same cycle times!)



MMR Buffer is an Aggregate

Again, it is important to stress that this buffer is not created for the purpose of "padding" the delivery time. It is an operational tool that will be used during execution. Next we will see how to use and interpret the buffer for this purpose.

2.6 Buffer Management, Usage, and Interpretation

The **buffer management** techniques that we are going to explore now are an integral part of TOC in general and of CCPM in particular (in the context of project management). The project buffer is a safety instrument. It protects the project (i.e. the implementation of an MMR in our case) from disruptions that might happen when the work activities (on the Critical Chain/in the MMR) are performed.

The buffer is *constantly monitored* to detect unfavorable variability.

An appropriately sized buffer will give good *leading indicators* of oncoming trouble. Just like the DBR buffer on the Kanban board gives signals about problems that are happening upstream of Herbie *before* they affect Herbie's own work, the signals from the MMR time buffer are reliable clues that problems are happening *before* they actually negatively impact reliable delivery.

The buffer gives a signal *while there still is time* to recover. Of course, one must also act and take the opportunity to do so.

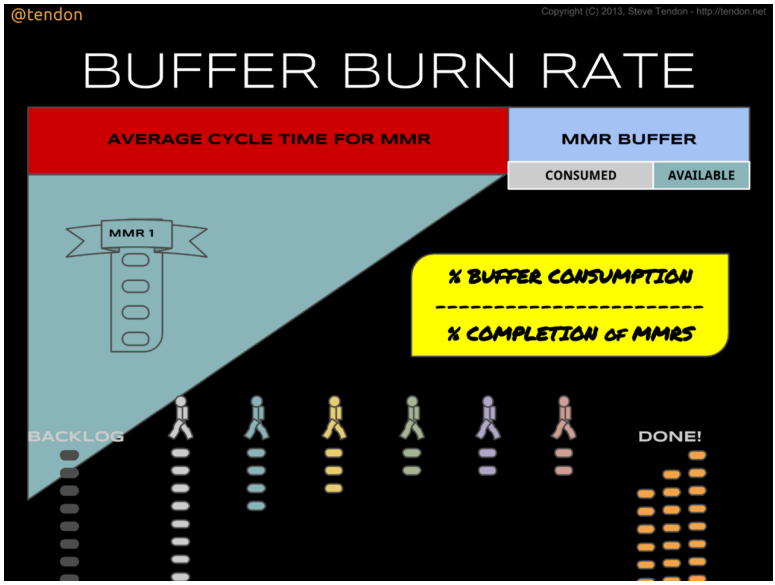
Buffer Burn Rate

Buffer management plays a critical role and the key is the concept of **Buffer Burn Rate** (also known as *Buffer Consumption*). The Buffer Burn Rate is defined by [SULLIVAN-2012[8]] as follows:

> *The rate at which the project buffer is being consumed [...] The rate is calculated as the ratio of the percent of penetration into the project buffer and percent of completion of the critical chain.*

[8] http://chronologist.com/bibliography#SULLIVAN-2012

For our purpose, the completion of the critical chain corresponds to the amount of work items in the MMR that have been completed. The percentage of buffer consumption can be computed in virtue of Little's Law. You can determine when the MMR will be delivered by using the current throughput and then compare that time point to the start and end points of the buffer. (An example of this will be given later.) High buffer consumption relative to completion is a sure sign that something is wrong.



The Buffer Burn Rate

In CCPM the buffer is used to coordinate resources on the Critical Chain (and have them ready when needed) and to prioritize work. Monitoring the buffer is the reason why the buffer is created. It is the premier activity to know if the project is healthy. [WOEPPEL-2005[9]] describes the role of the buffer very clearly:

Project execution is THE most important part of achiev-

[9] http://chronologist/bibliography#WOEPPEL-2005

> *ing success [...] Monitoring and responding to the condition of the buffers is the key to that. Rather than responding to individual tasks, the project team responds to the condition of the buffers. [...] The [buffer burn] ratio tells us when a project is in danger of not being completed on time. [...] By identifying which tasks are creating the highest buffer burn ratio, the project manager knows which tasks to focus on right now.*

Any task consuming the buffer is given the highest priority. In our case, since we are not dealing with tasks but with work items in an MMR, when the buffer gives signals, we will look at the configuration of the Kanban board to know where to intervene and identify the work items that *might* be having problem. If the Kanban board is not giving any special signals (i.e. everything is still "in the flow"), then we will use the techniques explained later to find ways to counteract *common cause variation.*

Monitoring the buffer is an *operational activity* performed during the project's execution. The Critical Chain is used for *planning* the project; but the buffer is used for *managing execution* of the project. This is why we forgo the Critical Chain's planning methods (we use Kanban or Hyper-Kanban instead) and take advantage of the buffer for what it gives in terms of managing execution. The buffer prompts self-expediting, assigns priority of resources and actions and solicits management actions when necessary.

The ability to have a *leading* indicator — the buffer consumption ratio — is the strongest contributions of CCPM. This indicator *does not report the amount of work done.* This is very different from most other project management methods, which tend to report project status in terms of work done (*"We are 90% done!" "Yeah! Right!"*). This indicator represents *work done in relation to how much time has been set aside (the buffer) to absorb unforeseen problems.* The extent to which this margin is consumed is an indication of the

project's health. Consequently, there are many operational advantages, which all enable improved risk management. For example, one such advantage relates to frequency of reporting. [LEACH-2004[10]] observes:

> *For buffer management to be fully useful, the buffer monitoring time must be at least as frequent as the shortest task duration.*

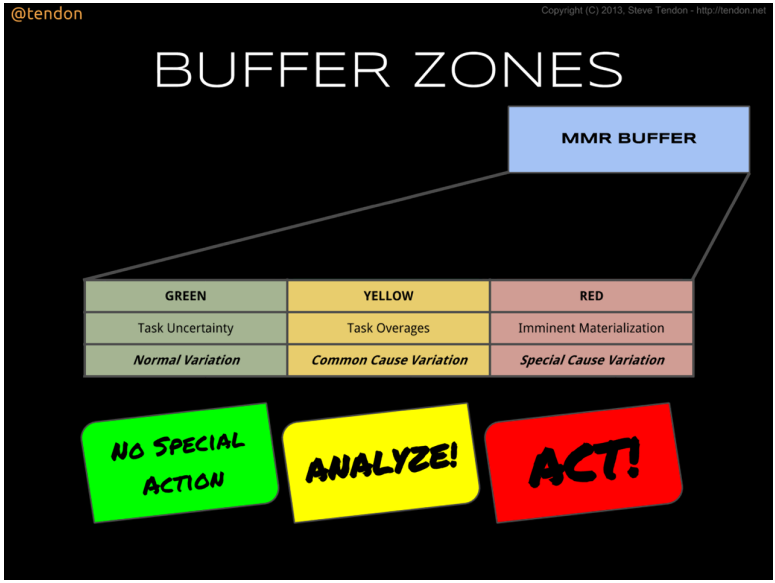
We can equate the "shortest task duration" to the shortest average cycle time of any class of service present in the MMR. If we monitor the buffer so often, then reporting frequency can be much higher. If the shortest planned task is measured in hours, then reporting can be made on an hourly basis — rather than weekly as is typical in most project environments — giving even earlier signals about problems.

Buffer Zones

The project buffer is divided into three zones. The zones are often represented in Green, Yellow and Red. Typically these zone are sized to one third of the buffer; though relative sizes may be changed dynamically in the more advanced applications (this relates to how to *appropriately size* a buffer.)

The three zones give a more granular control about knowing when to act. C. Spoede Budd and J. Cerveny (in [COX-2010[11]]) offer a crucial insight: the three zones are representative, respectively, of *Expected Variation*, *Normal Variation*, and *Abnormal Variation*.

[10] http://chronologist/bibliography#LEACH-2004
[11] http://chronologist/bibliography#COX-2010



Buffer Zones

Monitoring buffer consumption with respect to the three zones gives visible and actionable signals:

- **Expected Variation** (Green Zone): Everything is working *"according to plan."* The green zone absorbs inherent task uncertainty. **No special action** is required. In fact any interference in this zone is most likely counter-productive, as it would produce what [DEMING-1993[12]] referred to as "tampering:" a waste of productive time that causes loss of focus.
- **Normal Variation** (Yellow Zone): Everything is under control; but you must **investigate and prepare for action**. The yellow zone absorbs the inherent uncertainty in task duration prediction. Time is consumed to cover task overages: prepare plans to recover lost time, *but take no action yet* (avoid

[12] http://chronologist/bibliography#DEMING-1993

tampering). Focus on understanding what is causing time consumption and what can be done.

- **Abnormal Variation** (Red Zone): When the red zone is reached, **you must act**. Implement the plans prepared while buffer consumption was in the yellow zone. Most likely, unique events outside the normal course of the project's operations have caused the problem.

> **Common and Special Causes**: The TOC terminology can be mapped to the customary common cause and special cause variation. The *Normal Variation* and *Abnormal Variation* can be thought of, respectively, as *Common Cause Variation* and *Special Cause Variation*. The distinction will be made clearer later. Since buffer zones do not always represent this faithfully, we will be using other tools for distinguishing between common and special causes. What matters most is the operational valence of the three buffer zones.

Dividing the project buffer into three zones provides a powerful tool for anticipating and acting on risks. Other project management practices don't detect the problems until later. [LEACH-2004[13]] concludes:

> *Through this mechanism, buffer management provides a unique anticipatory project-management tool with clear decision criteria.*

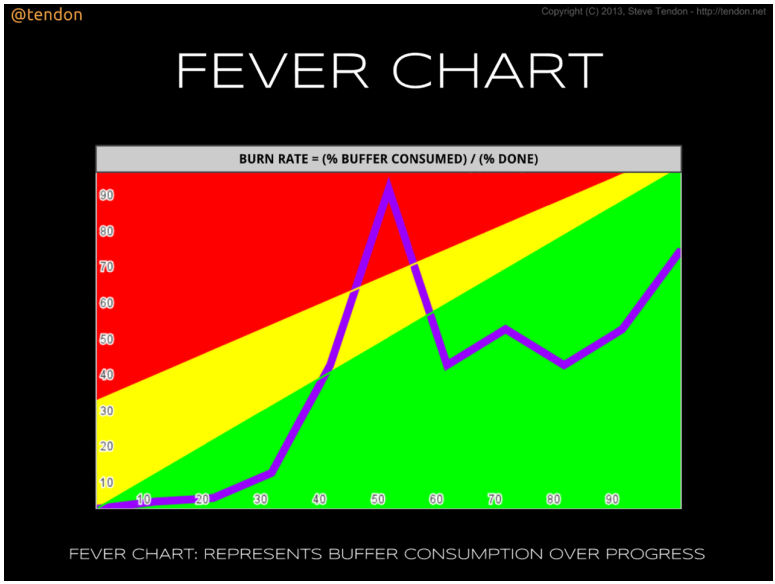
[13] http://chronologist/bibliography#LEACH-2004

2.7 Buffer Charts

Anytime we have quantitative data it is natural to use diagrams to gain insight into what the numbers really mean. The burn rate can be used as the basis for plotting two significant diagrams: the buffer fever chart, and the buffer control chart.

Buffer Fever Charts

To visualize the status of a buffer, you typically use a **Fever Chart**, where you plot the buffer consumption (as a percentage) towards the project completion (again, as a percentage). For instance, you could have a diagram that looks like the one illustrated in the figure.



Buffer Fever Chart

In the figure, the blue line represents the progress of the project's execution. You can see it started off in the green zone, progressed

into the yellow zone and then ran into problems when it penetrated into the red zone. Problems were addressed, execution went back into the green zone, and remained there until the end of the project, meaning that due date performance was acceptable ("green").

(Note: this figure is only illustrative. The actual placement and slope of the two threshold lines depend on how you have sized the buffer and, especially, how you have sized the three zones.)

The buffer fever chart is great for reporting the project health to stake holders. It can also be used in multi project environments, where the status of all projects can be represented on a single chart.

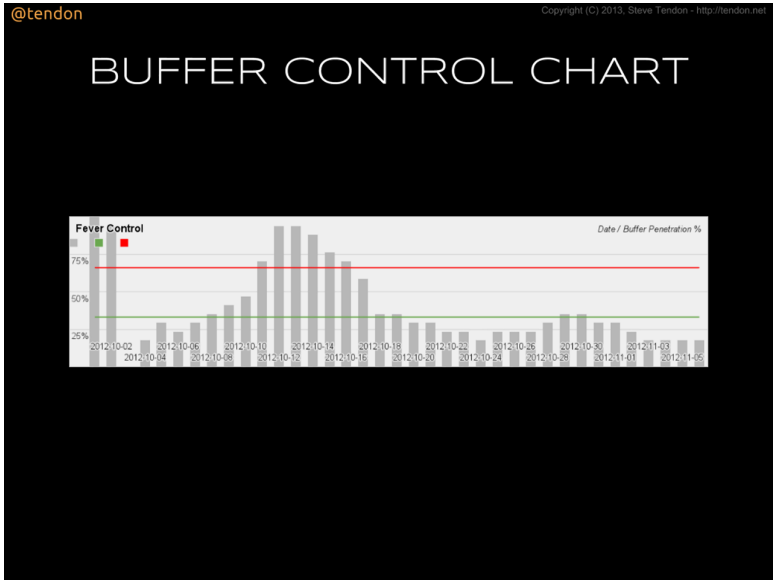
Buffer Control Charts

You can realize further refinements via **Control Charts**[14]. In fact, [LEACH-2004[15]] suggests this:

> *Plot trends of buffer utilization. The buffer measure then becomes, in essence, a control chart and can use similar rules. [...] trending buffer data preserves the time history of the data and shows the trend of buffer consumption vs. project time.*

This kind of chart certainly helps improving control (of work vs. time). A control chart gives very clear indication about the project's health with respect to (real) calendar time, rather than with respect to the buffer consumption. It can be used to get an idea if the project is healthy day by day and can be used as basis for insight and reflection during the daily meetings (like stand-up meetings) and also at any moment when project retrospections are performed.

[14] http://en.wikipedia.org/wiki/Control_chart
[15] http://chronologist/bibliography#LEACH-2004



Buffer Control Chart

As we shall see shortly, because the buffer control chart shares the same time line that can be used with cumulative flow diagrams, a powerful visualization is the combination of the two.

Thresholds and Signals

The chart is used like any control chart, with two control limits set by the thresholds between the green-yellow and the yellow-red zones.

The transition across the first threshold signals that you need to start investigate about potential problems. You should not act yet at this point, but simply start thinking about what might be a problem. The root cause analysis techniques that will be presented in the next Chapter will be very helpful, as they will uncover even sources *common cause variation* and not only sources of *special cause variation.*

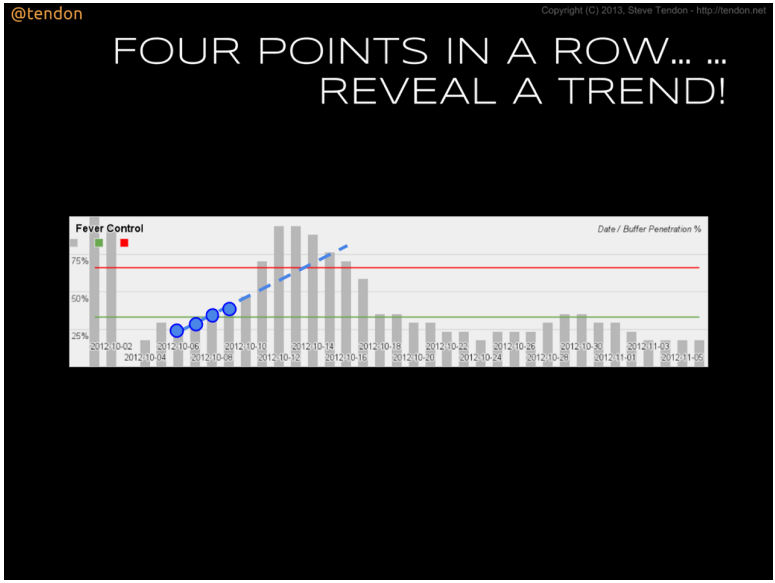
Fever Signals

The transition across the second boundary signals that there is a real problem and that action must be taken immediately. There is still time to recover when this threshold is passed, but there is no time to waste! Action should be quick and resolute. The nature of the action should already have been decided while the project was going through the yellow zone.

Trends

You can refine the use of the buffer control chart through the use of **Trend Analysis**.

For instance, four points in a row trending towards a threshold might be enough to take action. You detect the oncoming trouble *even earlier*. Use judgment when deciding whether to act upon such signals or not; but at least you now have the possibility of considering such early signals.



Trends

As noted by [SHEWHART-1986[16]], if your process is *not* under statistical control, then the use of trending data is even more so important.

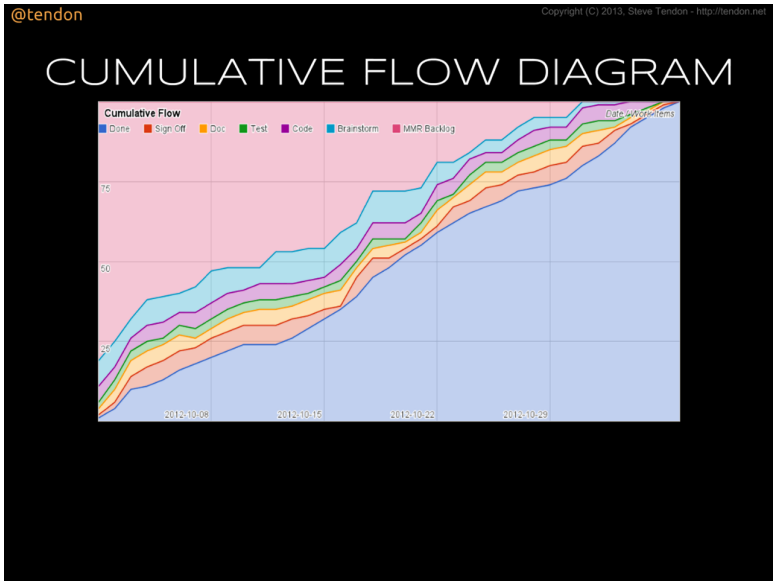
Empirical processes are seldom under statistical control; and as we have seen in an earlier chapter, software development and knowledge work can be considered as a dynamic complex adaptive systems that can only be controlled through empirical processes. Spotting trends in diagrams might be hard and might require experience; but they are a powerful tool that can provide additional and especially early and leading, insights.

Cumulative Flow Diagrams

It is not coincidental that Scrum uses Burn Down charts and that Kanban employs Cumulative Flow diagrams as the one illustrated

[16] http://chronologist/bibliography#SHEWHART-1986

in the figure. Both of these kinds of charts reveal trends in an empirical process. All such charts can be used to reveal emerging trends and help in taking decisions based on such trends.



Cumulative Flow Diagrams

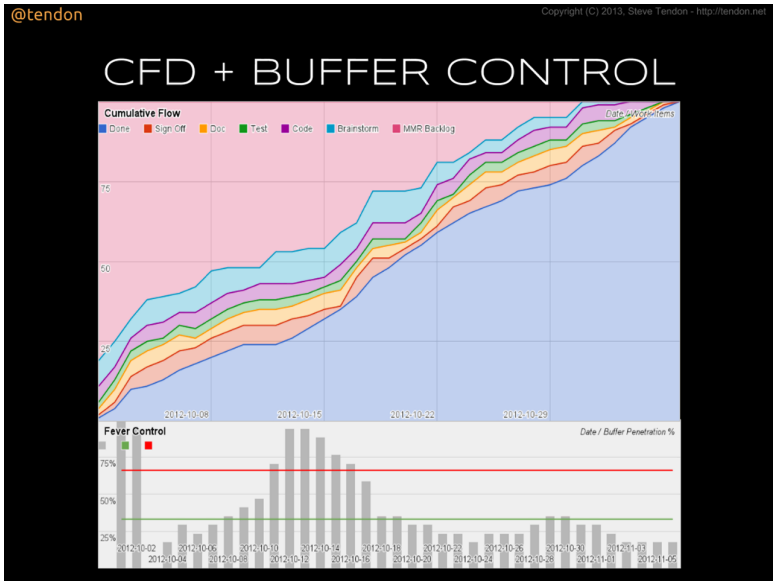
With the buffer management techniques of the Theory of Constraints (and the further techniques we will explore in the next chapter), such decisions can be taken with better insight about the nature of the problems that you might be facing, especially with respect to *common cause variation* or *special cause variation*.

Cumulative Flow Diagrams contain a lot of information and provide visual cues about the load of WIP through each work state. The formation of queues and starvation are easy to spot. The problem, though, is such signals usually come *late*; they are an indication that risk has already materialized.

Buffer signals are *leading* indicators; and their signaling can be reinforced if there is a trend developing in the cumulative flow

diagram. Therefore, the best results can be achieved when the Cumulative Flow Diagram is combined with a Buffer Control Chart, as described here next.

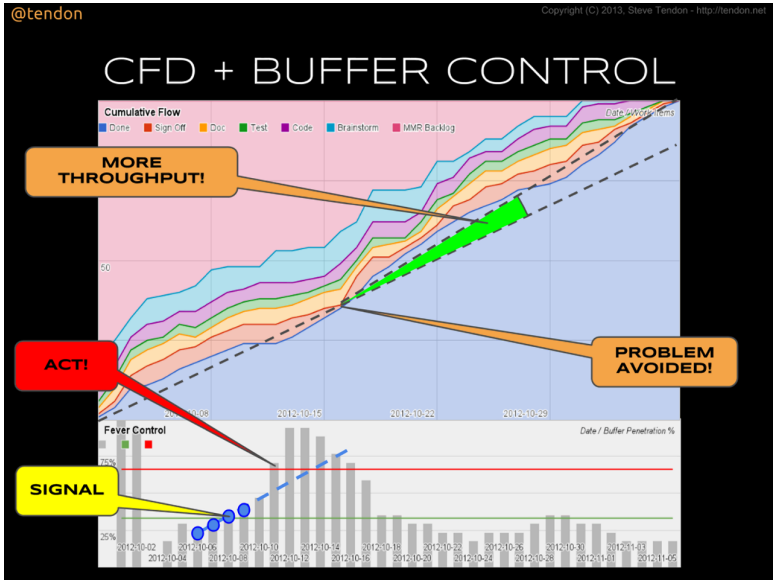
Combining Diagrams and Charts



Cumulative Flow Diagrams with Buffer Control Chart

A particularly powerful representation of the projects progress is the combination of a Cumulative Flow Diagram with a Buffer Control Chart. Both share the same horizontal axis, the time line.

With this combination the *leading signals* provided by the buffers become very apparent. In the figure you can see this combination. The two diagrams represent the same situation, on the same time-line.



Evidence of Leading Signals

Trends developing on the cumulative flow diagram can be used to pinpoint where problems are and to confirm the early signals of the buffer control chart. For instance, in the figure you can see that when the control chart signals that buffer consumption has entered into the red zone, the WIP of the "Sign Off" state is progressively getting lower and lower: it is trending towards starvation.

Without the leading signal of the buffer control chart, this trend in the cumulative flow diagram would go undetected until much later.

It is evident how the cumulative flow diagram reveals a problem that is solved more or less in the middle of the time line. However, if you observe the corresponding buffer chart, you can see that a yellow signal was raised seven days earlier and the red signal four days earlier. Six days earlier the buffer chart exhibited four points in a row revealing a trend towards the red zone.

In other words, compared to a situation without the buffer chart, in this case we have between four and six days advance notice of

oncoming trouble.

The effect of the intervention that was made in virtue of these early signals can also be read off the cumulative flow diagram, where a noticeable increase of throughput is evident after that mid-point hiccup.

A general rule of thumb is that you don't need to look at the cumulative flow diagram unless you are receiving signals from the buffer control chart. As soon as such signals appear, watch out for any trend that you see developing on the cumulative flow diagram to understand which work state(s) might hide some problem.

Signal Reaction Handled by Normal Kanban Policies

Whenever the MMR buffer raises a signal, you must identify the Root Cause (in the next Chapter we will see how to do this) that is the source of the problem. If the resolution requires action by the team, then you create a corresponding Issue/Impediment card. Since we are managing work flow with Kanban, you must assign a class of service to the impediment card. Hence the impediment card will be handled through normal Kanban class of service policies.

2.8 How to Build and Monitor an MMR Buffer

We have already seen that sizing a buffer for an MMR is simple. You add up the cycle times of all work items and take half of it. What might not be so clear, at this point, is how to use the buffer *in practice*.

For the sake of illustration, let's assume that we have `100` work items in an MMR and that the average cycle times adds up to `34` days. In

other, words, we are expecting to deliver the 100 work items in 34 days with a confidence level of 50%.

We divide 34 in two and we get a buffer of 17 days. So, for instance, if the project starts on October 1, the expected end date would be 34 days later, on November 4. (In this illustration we do not consider weekends or holidays; though, naturally, in a real case all dates would need to be adjusted accordingly.) The buffer would extend for another 17 days from November 5 to November 21.

Let's examine the situation on October 7. Suppose that 18 work items have been delivered. The percentage of work done is 18% (we had 100 work items to start with).

To understand if we are in a healthy or unhealthy state, we need to figure out the buffer status. This is were we will invoke Little's Law. We know that the time elapsed is 7 days. So actual throughput up to this date is the delivered WIP of 18 divided by the cycle time of 7, resulting in 2.57 work items per day.

We now assume that this is the *sustainable* throughput performance of the team and that it will be kept until full delivery. Let's look at how much time we would need, with that actual throughput, to deliver all 100 work items in the MMR. Again we invoke Little's Law. The time needed is the original WIP of 100 divided by the actual throughput of 2.57, which gives us 39 days (rounding up to whole days.)

The expected 39 days are 5 days more than the 34 days established through our historical cycle time. If the actual throughput remains unchanged we would consume 5 days of our 17 day buffer. Our buffer penetration is 5/17 or 29%. This is less than 33% so we are in the *green zone* of the buffer. There is nothing to worry about.

Let's now suppose that on the following day, October 8, two more work items are delivered for a total of 20. Elapsed time is 8 days. Actual throughput is 20 work items divided by 8 days, or 2.5 work items per day. The estimated time to complete the 100 work items

in the MMR is the original WIP of 100 work items divided by latest actual throughput of 2.5, or 40 days. Now 40 days are 6 days more than the 34 days from our historical average cycle time. Buffer penetration is 6/17 or 35%. This is above the 33% threshold. We have just passed into the *yellow zone* of the buffer. We need to investigate what might be adversely affecting throughput performance.

We have just illustrated twice the basic calculation. Take advantage of Little's Law and calculate the expected delivery date based on the original WIP and the measured actual throughput. Look where that date falls with respect to the buffer; and if it is later than the starting day of the buffer, calculate the percentage of penetration. Once you have a spreadsheet set up with the calculations it is no harder than inputting the total amount of WIP deliverd up to date.

The numbers used in this example are those that can be seen in the figures with the cumulative flow diagram and the buffer control chart presented earlier above.

Little's Law and the Assumption of Steady/Ideal State of Flow

Naturally, what we are doing is relying heavily on Little's law to make both the sizing of the buffer and the subsequent calculations meaningful. Little's law is significant only when the system is in a steady state; and this means that the input is balanced to the output.

We have seen how we can achieve dynamic balancing through the Drum-Buffer-Rope, Kanban tokens, and Replenishment tokens used in Hyper-Kanban. The Hyper-Kanban approach is more stable than the original one used in Kanban, with work state WIP limits. This can literally be seen when comparing cumulative flow diagrams of Kanban projects with those of Hyper-Kanban projects. Kanban projects present a cumulative flow diagram that is much more jagged, with many stop-and-goes, flats-and-spikes; these are all consequences of the nature of work state WIP limits. A cumulative

flow diagram of a Hyper-Kanban project (like the one shown above) is much smoother, with slower accelerations/decelerations of the flows, that are detected and acted upon earlier both because of the DBR mechanism and even more so because of the MMR buffer management techniques presented here.

As we have mentioned many times, work state WIP limits introduce artificial disruptions to the flow of work. Artificial disruption of flow makes the application of Little's Law less reliable. The implication is that, even though the MMR buffer technique works very well with Kanban (in fact, it was first devised of with Kanban, before Hyper-Kanban came about), it works much better with Hyper-Kanban, because Hyper-Kanban strives to interfere with Little's Law as little (pardon the pun!) as possible.

Little's Law and the Conditions of Maximum Sustainable Pace

The method works even when the team is working at its *maximum sustainable* pace. Notice that a steady state (that is, when Little's Law is applicable) might be achieved even at a level of throughput that is *less* then the maximum sustainable one. This happens, typically, when flow is balanced to average throughput exhibited by Herbie, rather than to Herbie's real capacity, as was shown in the Section The Mirage of Balancing the Flow in the Chapter on Kanban. It is very easy to fall into this trap when one seeks to "balance the flow." And it is a situation that the drum-buffer-rope mechanism will avoid.

Even if you are most likely not getting anywhere close to the team's maximum sustainable pace with Kanban (though you would with Hyper-Kanban), the peculiarity to notice is that when the team is at its *maximum* sustainable pace, then one would not be able to apply the normal Kanban strategy to increase the WIP limits in order to gain more throughput. That would effectively introduce a special

cause variation disruption into the workflow. Herbie will trip over and fall — so to say — and halt the whole troop's progress.

Because we do not want to lose *sustainability*, the only way to improve throughput is to decrease cycle time — and, as we have learned in the earlier chapters, such a cycle time reduction must involve Herbie, the constraint.

This is an important point. You cannot increase throughput indefinitely by increasing WIP. There comes a point when by increasing WIP limits, the team's ability to cope with the additional work load will become unsustainable. It is at this point that any further attempt of increasing throughput by increasing WIP limits will fail. Such attempts will be "tampering" with the WIP limits and "thrashing" the system.

The failure will certainly show up on the Kanban Board, but the application of the five focusing steps will only reveal the special cause reason why the greater pace could not be sustained (additional WIP). It will not reveal the inherent reason why the troop cannot walk any faster, notwithstanding that Herbie remains still the overall constraint (why capacity is limited). Finding out that you are going slower *because* you have increased the load does not give you any more information. You added that extra load yourself. Instead, finding out why capacity is limited can bear much greater fruits. If you can increase capacity, *then* you will be able to increase the load too.

It is like an athlete practicing endurance sports. An increase in WIP limit will just result in a *sprint* (may the readers who are into Scrum pardon the pun here!), that will make everybody run out of breath after a short while. But we want to run marathons, not sprints. We want to keep a sustainable pace for as long as possible; and increase that pace without falling to the ground breathless.

When you are at the *maximum* sustainable and steady state, WIP limits should be kept untouched in order to *maintain* that level of sustainable performance. It is in this situation that we will use

buffer management to reveal what is preventing the team from being even more productive. The improvement that will result will affect the entire process. The improvement will not just remove one temporary (and maybe artificial) impediment from the current project or work flow. The improvement will be of benefit to all *future* projects too, because it will be a lasting improvement to the *process*.

The focus of any improvement attempt has to shift from the individual resources, to the entirety of the team, seen as a system. We want to find what prevents the team from increasing their maximum sustainable pace, all the while we keep the process in the ideal/steady state! This is what is meant by improving *while in the flow*! but to achieve this, we must start hunting for sources of common cause variation. Reacting to special cause variation alone will not allow you to make such break through improvements. The nature of these improvements are truly breaking systemic constraints and they are Archimedean levers that will increase performance to much higher levels.