Reliable Scrum Guideline 2012

Agility and Reliability through elements out of Critical Chain.

SPEED 4 projects

Situation

Scrum has found his way into many development departments in many enterprises. The consequent control of the work-in-progress within the sprint, the strict building of teams out of the best available experts and the iterative way of working were a major breakthrough and show most of the time good results.

Speed4Projects®

Coaching, training and consulting to implement Critical-Chain and High-Speed project management

Further information you can find on my website www.speed4projects.net

Problem

Normally we work in "big" enterprises and we want to solve "big" problems in form of "big" projects. And then the problems occur.

Scrum is now confronted with the needs of a surrounding organization like:

- The need for reliability. For example if you want to sell a new product in a campaign, a lot of things have to be planned and executed. If the product is not available at this time, a lot of damage is done the company. Therefore the company need a reliable due date when the product is available without reduction in scope.
- The need of transparency. There are always persons who are accountable for the success of a project. They delegate the responsibility of establishing results within a timeframe with given resources. But they stay accountable – and therefore they need to be sure whether the project is going well or whether they have to take some measures to bring it back on line. Therefore the person which is accountable needs objective transparency about the status of the project.

Of course the scrum gurus will say: „we are more reliable than the classic project managers" and „we are absolute transparent". An on the other side they say "it's done when it's done" and "we have this product backlog. It shows after every sprint the current estimated time of finishing. I can tell you after every sprint whether the due date fits or not."

Scrum is good but finally the need for reliability and transparency is not fulfilled. This results in acceptance of Scrum in bigger organizations.

So let's do something against it and make Scrum reliable.

Solution

Out of the project management community we know an approach called Critical Chain. This is a holistic systemic methodology with a lot of success stories all over the world.

Out of this methodology two parts are very interesting for scrum:

- Dealing with uncertainties by aggregating the buffers in the estimations at the end of the project. As a result the due-dates getting very reliable.

- Showing an operational and objective project status by monitoring the progress according the buffer consumption as a easy to read fever curve.

Scrum is good! The Scrum framework has not to be changed at all to get reliable Scrum working.

⚠ One thing is important – reliable Scrum makes just sense in a project environment. If you have already built the product and if you are in this incremental improvement mode, when you deliver after every sprint, then reliable Scrum brings no benefit. Then you are in the production mode and you won't get anything out of project management. But if you are in a mode where you build a new product or if you're working towards a big important release, then you can really benefit from reliable Scrum.

What has to be done to start reliable Scrum?

- Define what part of the release or product is. This should be more than 5 sprints worth – the more the better.

- Find an appropriate due-date for this release or product, where the amount of work in the backlog and the velocity fits together. This due-date has to be calculated in a way, that there is enough buffer in it to have a realistic chance of success.

- Plot after each sprint the new fever curve to get a transparent status.

- Watch out what will happen?

And if you have a portfolio of scrum work streams, than you can do this for all of them. Just put the results together in one diagram and you'll get a perfect overview about all work streams.

What will happen?

Reliable Scrum is new – and I'm sure we don't know everything. But there are experiences out of the first implementations such as:

- It changes the focus from a sprint to sprint basis to a bigger release that delivers real and recognizable value to the customer

- It helps to clarify the backlog in many ways. First of all the stakeholders (there are typically more than one) are identified. The backlog will be completed (as good as possible), the backlog items are qualified whether they are part of the release or not. Based on this Backlog the stakeholders can give their sign off. All stories are estimated (in story points) and big stories are broken down to smaller ones. The result is a clear set of qualified stories adjusted with the stakeholders. Attention: this backlog is not fixed it can change according to the scrum process.

- The team including scrum master and product owner identify the expected variations in the backlog and velocity. They think together about the risks and chances and will make them transparent.
- It helps to find a realistic due date or a realistic amount of backlog for a release. It helps to identify the correct amount of resources needed to fulfill the expectations of the stakeholder. As a result the team gets a realistic probability of success.
- The fever curve will become an excellent tool for the product owner to assess how many stories he can pull into the release or has to leave out. He can adjust the backlog to be always in the yellow zone. In this case there will be always a realistic chance of success. The product owners love reliable scrum.
- The team gets a buffer at the end of the release and a qualified feedback about their performance. So the focus is not so strong on the single sprint. They don't have to protect the sprint results and buffer them. As long as they stay in the green zone they are save. So they can pull as much as possible stories with an optimistic velocity. This prevents the student syndrome and Parkinson and as a result the velocity will improve (more then you can expect right now).
- The stakeholder gets an objective easy to read status report and feel very confident about the reliability of the team. As a result they leave the team alone and work self controlled. The best a team can expect to work as concentrated as possible.
- In big projects with a lot of scrum teams the stakeholders will get an overview about all the steams at ones. The can concentrate on the red ones. Even one more aspect to focus just about the really important issues.

This is just a start – I expect even more positive results ☺

Implementing Reliable Scrum

#1 Define a "major Release"

Reliable Scrum doesn't make sense for scrum teams or products which are in an incremental or maintenance mode. If you can release after each sprint and the customers (and the stakeholder) are happy with this – just leave it as it is.

Out of my experience (with a lot of projects) there are reasons why there always will be "major releases" (formerly known as projects).

First of all – to come into this incremental mode you first have to have a product at all. You have to have this minimum sellable feature set. And this is typically not done in the first sprint – you need several sprints to get there.

Second after having this product you have to decide how to make money with it. Small increments are not sellable without annoying the customer. To really get the focus of a customer you have to have a release with some extraordinary new features in it. And the build these you need more than one sprint – you need a major release.

Just sit down together with the product owner and the stakeholder and define this major release. Define the name, write down the story for the customer, name the value and list the features.

>> The result of this step is a named release with a feature list adjusted with the stakeholders.

If you have a problem doing this, than you have a real problem (that can't be solved with scrum, Kanban, classic project management or even reliable scrum). In this case you're missing a business strategy and you should focus on that.

#2 "Complete" the backlog

You now have now the feature list of this major release and typically you have some kind of a backlog too.

In your current backlog there may be stories that don't belong to this release. So the best way is to tag just the stories that belong to this release and leave the other stories as they are.

If there are stories that are just partly needed for the release – break them apart.

If there are stories missing that are needed to build the features – just add them. Add also conceptual, research or refactoring stories.

>> The result of this step is a list of stories necessary for this release, estimated in story points

The meaning of "complete" is here "best effort". This should not be a way back to "water fall". Important is to reduce the risk in the backlog. If some stories are missing, not estimated or estimated with more than 100 story points – that's o.k. unless it's not too much. A good indicator is that something around 90% of the stories are estimated and below 42 story points.

Normally you can't reach this clarity before the first sprint. Therefore all this is done in parallel to the normal set up of a scrum team. But after 2-3 sprints it should be possible to define a complete backlog - otherwise as I suggested in step 1 you have other bigger problems, that need other ways to solve.

⚠ Maybe you've seen that there is no MoSCoW scheme any more. In practice this scheme has proven futile, because everybody understand it a little different, even if it is defined clearly. For reliable Scrum it is not necessary any more. Stories are part of a release or not. The common way in scrum to protect ("buffer") the due-date with should/could/would-Stories is in practice also problematic. The stakeholders just know "I'll get feature X, Y at the due-date Z". They understand "should, could and would as I'll get it" – and they are always disappointed if not. This ambiguity leads to disappointment – this can be avoided with reliable scrum and buffering over time and not over features.

#3 Balance Resources/Backlog and Due-Date

Now to the most important part and essential precondition for reliability – the realistic due-date, the realistic amount of backlog or the correct amount of resources. In the end all three factors have to be balanced to have a realistic chance of success for the team to fulfill the expectations and deliver the release in time, in scope and in budget.

But how can this be established? There are a lot of problems around! We don't know the correct amount of work to be done – the estimations are always wrong and the amount of stories in the backlog is not fixed at all (you remember – it's agile). And on the other hand we have deviations in the velocity. Sometimes everything goes well in the team – but sometimes persons leave or join the team an then the group dynamics starts, or there are vacations and sometimes illnesses too. That's live – that's normal!

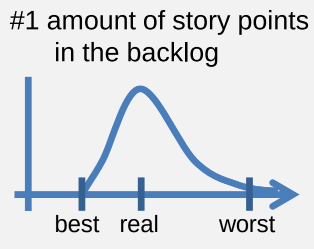
And the solution is just around the corner. There are a few tricks that help us to achieve the goal. (ok not absolutely – there is no absolute security!).

One trick, we have already done, is to aggregate a few sprint together to a release. If something goes wrong in one sprint it can be repaired in the next. If you aggregate uncertainty you get more certainty. If you have more than 6 sprints in the release this effect works pretty well.

The second trick is to make the probability of success transparent and adjust it to and realistic value together with the stakeholders. To do this all we need are some estimations and some mathematics.

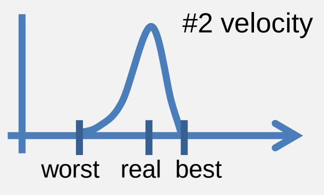
First of all we need an estimation of the amount of work to be done. That is nothing else as the backlog of the release with its story point estimations. This is a base to start from. But estimations are never one point value. They are probability curves with a best-case, realistic case and a pessimistic case estimation.

For the amount of work to be done it's like this. The current estimated story points in the release backlog are the best-case estimation (I've never seen a project that needed less then estimated in the first round). The other two values we just get by team estimation. The team sits together with the product owner and estimates how many more story point they expect in a realistic and in a pessimistic case. This is some kind of a deal and therefor the product owner, as representative of the stakeholders, hast to participate, understand and agree upon these estimations.



The x-axis is the estimated value and on the y-axis you see the relative probability

The other factor in the game is the velocity. Typically the team has done some sprints and therefore knows a little about their velocity. The average of the real velocity is a good value for the realistic case. The best and the pessimistic case are again estimated be the team – and now together with the scrum master. They are responsible for their velocity – no product owner can argue about this. The only thing the product owner can do is to invest some more money to add more resources or even more scrum teams. The deviation out of the last sprints can be a good indicator for this estimation – but the team has to keep in mind the whole release, with all the fluctuations, holidays, illnesses, up and downs.



The same for the velocity

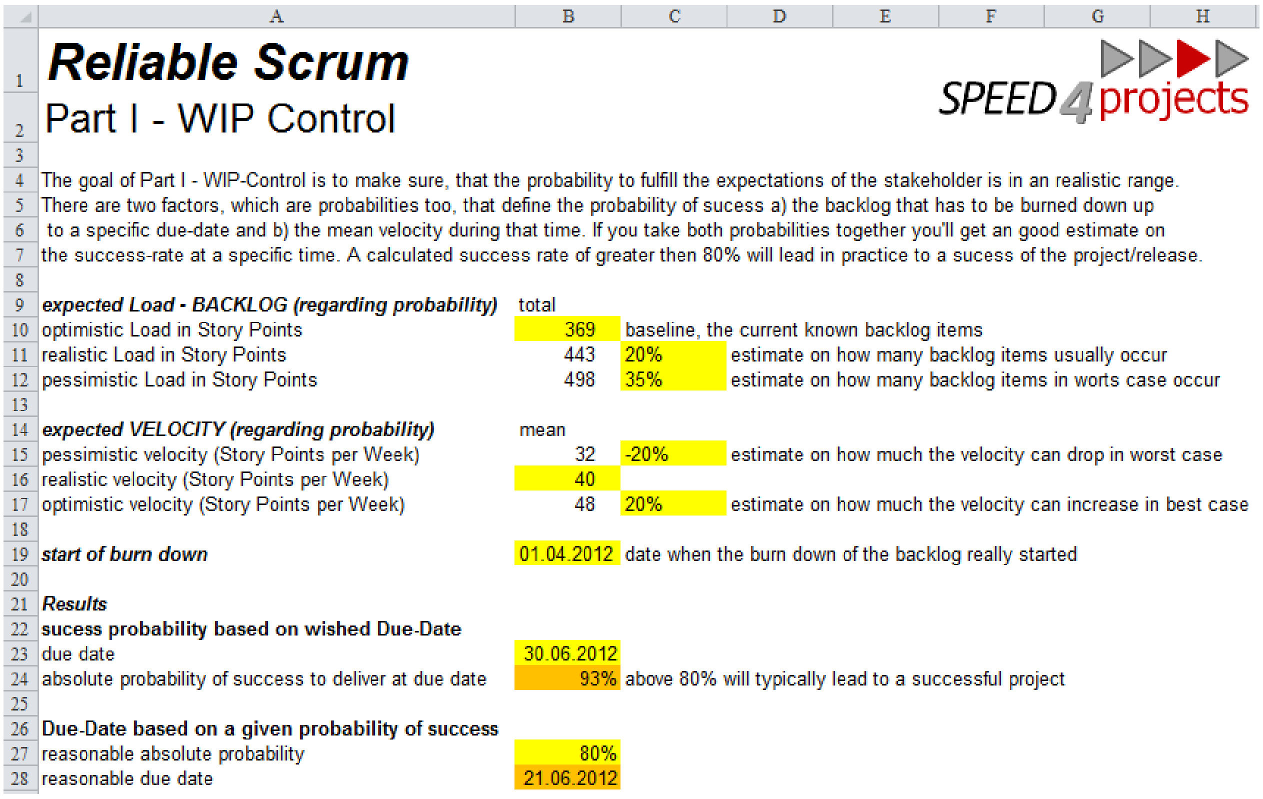
The rest is easy ;-) On these two probability functions the convolution operator can be applied an as result you get the relative probability of success over the time. After this you just have to integrate the curve and you'll get the absolute probability of success over the time – or even an idea of it.

Out of practice and many projects we know that 100% probability is far too much and to too expensive – 80% are absolute ok and a realistic chance of success. All you have to do now is to talk with the product owner and the stakeholder to reduce the amount of stories in the backlog, get more resources or adjust the due-date (number of sprints). That's the real work – but very healthy.

>> The result is a balanced backlog, velocity and due-date and a realistic probability of success for the team and the release.

Don't panic – all this mathematical stuff is programmed into one excel sheet. This sheet can be found on the www.reliable-scrum.info under "download". ⚠ Pay attention that the macros have be activated.

The functionality you need to balance the workload and due date is found in the sheet "Part I - WIP Control". All you have to do is to fill in the yellow fields and as result you get the brown ones. You get either a probability of success (B24) or a reasonable due-date (B28).



	A	B	C	D
1	***Reliable Scrum***			
2	Part I - WIP Control			
3				
4	The goal of Part I - WIP-Control is to make sure, that the probability to fulfill the expectations of the stakeholder is in an realistic range.			
5	There are two factors, which are probabilities too, that define the probability of sucess a) the backlog that has to be burned down up			
6	to a specific due-date and b) the mean velocity during that time. If you take both probabilities together you'll get an good estimate on			
7	the success-rate at a specific time. A calculated success rate of greater then 80% will lead in practice to a sucess of the project/release.			
8				
9	***expected Load - BACKLOG (regarding probability)***	total		
10	optimistic Load in Story Points	369	baseline, the current known backlog items	
11	realistic Load in Story Points	443	20%	estimate on how many backlog items usually occur
12	pessimistic Load in Story Points	498	35%	estimate on how many backlog items in worts case occur
13				
14	***expected VELOCITY (regarding probability)***	mean		
15	pessimistic velocity (Story Points per Week)	32	-20%	estimate on how much the velocity can drop in worst case
16	realistic velocity (Story Points per Week)	40		
17	optimistic velocity (Story Points per Week)	48	20%	estimate on how much the velocity can increase in best case
18				
19	***start of burn down***	01.04.2012	date when the burn down of the backlog really started	
20				
21	***Results***			
22	**sucess probability based on wished Due-Date**			
23	due date	30.06.2012		
24	absolute probability of success to deliver at due date	93%	above 80% will typically lead to a successful project	
25				
26	**Due-Date based on a given probability of success**			
27	reasonable absolute probability	80%		
28	reasonable due date	21.06.2012		

This sheet is named "WIP Control" cause in the end here the amount of work-in-progress is adjusted to the given velocity and due-date. Doing this is always hard work but it is absolutely necessary for success and helps the team in many ways.

⚠ In some scrum teams stories that are not delivered as promised within the sprint doesn't are not counted. They have to be delivered in the next sprint but they are not counted there either. This makes no sens for reliable scrum. Story points are counted at the moment they are finished.

Execution Control

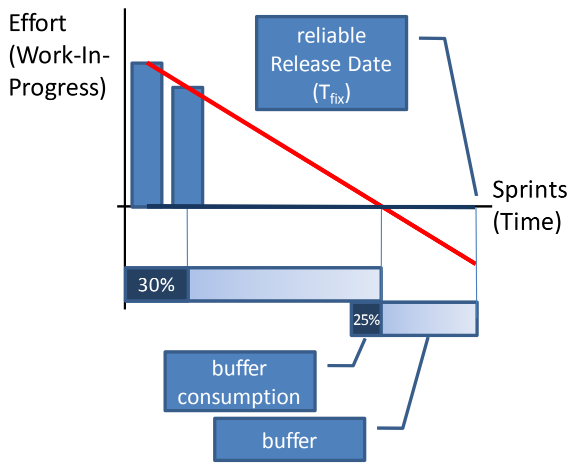
The hard work is already done – now it's time to harvest the results.

Project management and even Scrum is all about trust. In the end the stakeholders stay accountable for the results. They delegate some responsibility to a) the product owner – he has to define the correct stories to build a good product, b) the scrum master – he has to make sure that the team performs and c) the team – the have to deliver. Responsibility comes always with some reporting – the responsible ones have to be able to response when asked. And if they can't give a correct and clear and objective status – they trust is diminishes. As a result µmanagement occurs and the team is disrupted in their work – and so on …

The scrum guys will now say "We are absolute transparent. We have the product burn down chart!" And yes that's right. I appreciate the product burn down chart. It is a very useful and elaborate status reporting.

But what is the message? The stakeholder will get after each sprint a new estimate when the product will be finished – fine. But if he has a due date in mind (and he always have otherwise the product were of no interest for him) there are just two messages a) the new estimated time of finishing is before his due date – then everything is fine, b) the estimated time of finishing is after the due date – and then the project is lost! There is just a win or fail but nothing in between. And he has always in his back of his mind, that the backlog can change every time and a win project can get a fail in every sprint. That builds no trust.

What is the alternative? If we did the exercise of the finding the correct amount of work or a realistic due date we have gained something very important – we got some buffer at the end of the release. We estimated some buffer in the backlog and some in the velocity. If we add the now fixed due date (or reliable release date T_{fix}) in the product burn down chart we will see the buffer.



Just take a look at this example: after two sprints we have some real burn down and an estimated time of finishing (where the red line and the x-axis crosses). That is what is left from the buffer.

Why ist this not the whole buffer anymore? There are some background thoughts to be done here. The goal is to get a objective and really good project status to gain the trus of the stakeholders. Therefore you have to answer the question wheter the project is green (on track) or red (critical). And with the buffer you have a extrem good indicator!

What do you think? If the progress in the release is less than the buffer consumption – is this good or bad? Bad of course - it's red! But if there is some buffer left it is possible to reach the due date – but it is critical. If you have no buffer at all – that is really bad. Than you have no chance to win any more – it's black (a new but meaningful colour in status reporting).

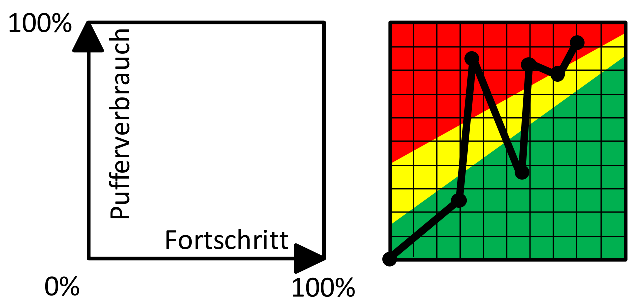
If the progress in the release is better than the buffer consumption – is this good or bad? Yes of course it is good – it's green!

So the best would be if the progress is something near the buffer consumption. Each release starts with progress and buffer consumption 0% and end hopefully with progress 100%. At this time the buffer consumption can be 100% - that would be ok.

With this in mind you understand that after 2 sprints in this example and the shown burn down the progress is something around 30% and therefore the buffer consumption should be nearly the same.

But how big should the buffer be? There is no absolute number – just some rule of thumb. If you set the buffer to 0% of the lead time (duration between start and due date) than this system doesn't work – that's clear. If you set the buffer to very high value (e.g. 50%) than everybody knows there is much buffer and student syndrom and parkinson will occur. Despite of that with such a big buffer it will take a long time an lots of problems until the red flagg will be rised. So the solution should be something in between. In practise (out of thousands of critical chain projects) the value of 30% was found very suitable. It gives enough buffer to cope with murphy and agility – and it reacts fast enough if something is going wrong.

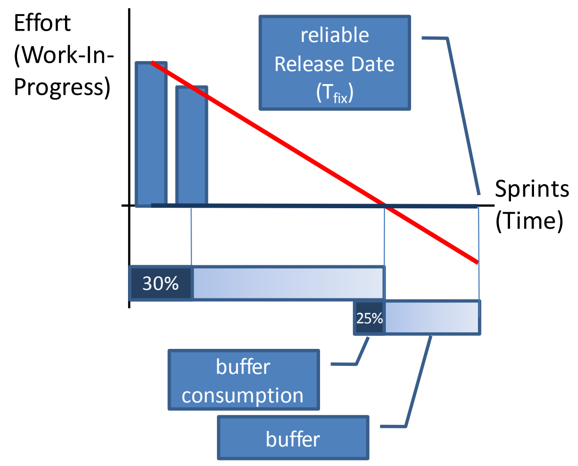
Putting all together you'll get the following diagram – called fever curve.



On the x-axis you'll see the progress and on the y-axis the buffer consumption. Each release starts in the left lower corner and ends in the upper right. In the diagram you see the colours – with an additional yellow zone. The perfect release will always stay at the border between yellow and green. Each point is the end of a sprint. That is pretty easy to understand (even for stakeholder :-)

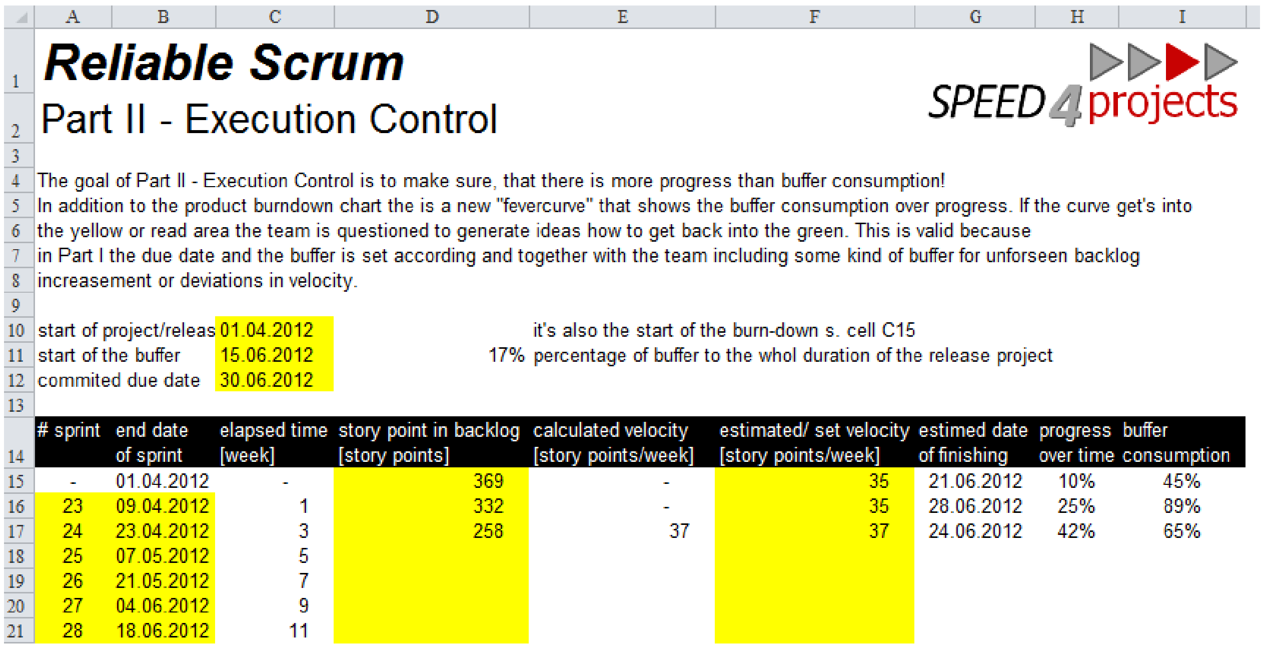
>> The result is an objective very easy to understand release status.

Progress an Buffer consumption are calculated as shown in the product burn down chart.



Don’t panic – all this stuff is also programmed into the same excel sheet. This sheet can be found on the www.reliable-scrum.info under “download”. ⚠ Pay attention that the macros have be activated.

The functionality you need to draw the fever curve is in the sheet “Part II - Execution Control”. All you have to do is to fill in the yellow fields and as result you’ll get the fever curve.



1 **Reliable Scrum**
2 Part II - Execution Control
3
4 The goal of Part II - Execution Control is to make sure, that there is more progress than buffer consumption!
5 In addition to the product burndown chart the is a new "fevercurve" that shows the buffer consumption over progress. If the curve get's into
6 the yellow or read area the team is questioned to generate ideas how to get back into the green. This is valid because
7 in Part I the due date and the buffer is set according and together with the team including some kind of buffer for unforseen backlog
8 increasement or deviations in velocity.
9

	A	B	C	D	E	F	G
10	start of project/releas	01.04.2012			it's also the start of the burn-down s. cell C15		
11	start of the buffer	15.06.2012		17%	percentage of buffer to the whol duration of the release project		
12	commited due date	30.06.2012					
13							

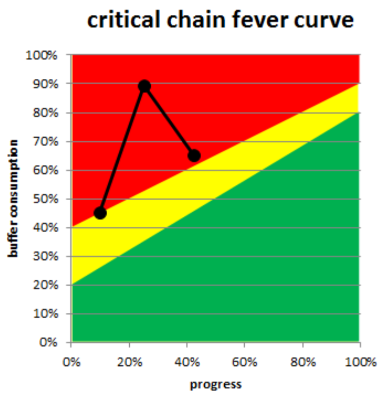
	# sprint	end date of sprint	elapsed time [week]	story point in backlog [story points]	calculated velocity [story points/week]	estimated/ set velocity [story points/week]	estimed date of finishing	progress over time	buffer consumption
15	-	01.04.2012	-	369	-	35	21.06.2012	10%	45%
16	23	09.04.2012	1	332	-	35	28.06.2012	25%	89%
17	24	23.04.2012	3	258	37	37	24.06.2012	42%	65%
18	25	07.05.2012	5						
19	26	21.05.2012	7						
20	27	04.06.2012	9						
21	28	18.06.2012	11						

Ok there ist a little more data to enter but you have all at hand. In line 10 to 12 you enter the timing. And as you see in this real example we worked with just 17% buffer – that’s pretty tough but works.

In the table below you will find the burn down data. Line 15 is the start of sprint 1 and mainly important for the amount of initial story points in the release backlog. After each sprint one new line is added. In this example the team worked over a long time with scrum und was already in sprint 23 when they applied reliable scrum. As a result the velocity information was also very stable.

After each sprint the remaining amount of story points are entered in column “D”. Out of this the sheet calculates a velocity (linear regression) – but this is just a hint. More important is the estimated velocity for the reast of the relase. This is provided by the team and the scrum master. In this case the calculated and the estimated velocity are equal – but this in normally not the case.

Based on this the sheet calculates the estimated time of finishing, the progress and the buffer consumption – and shows this as a fever curve. In this case – caused by the small buffer the release is in the red zone but with a clear tendency to the yellow and green. The team regains buffer!



That's it – that's reliable scrum.

No of course not – there is more!

Reliable Scrum the hero for the product owners

Coming soon …

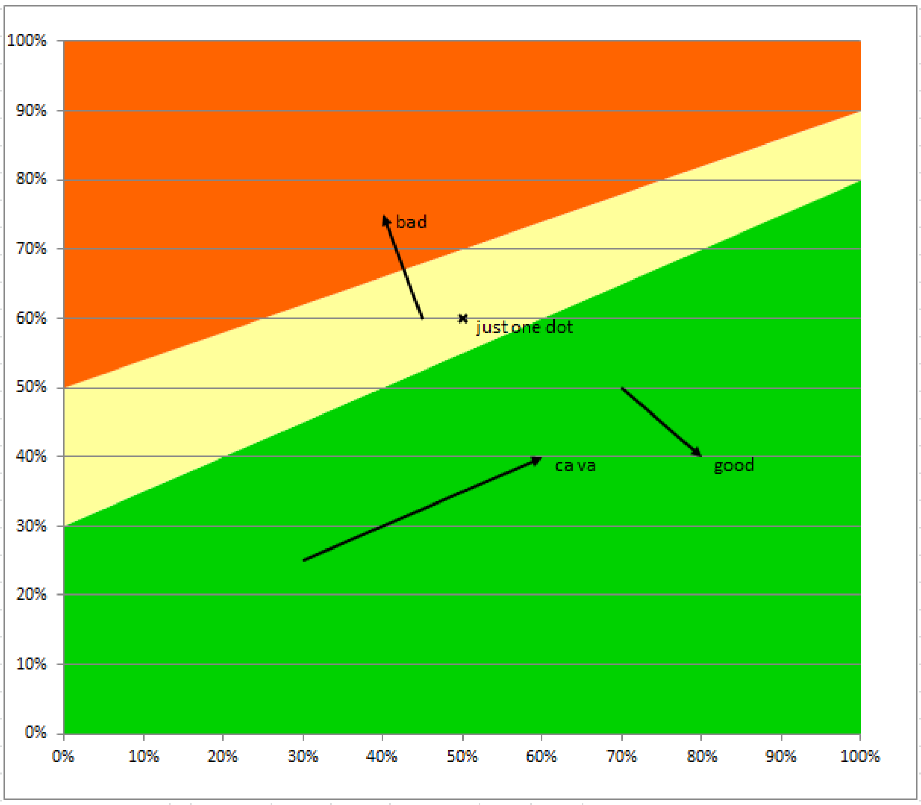
The portfolio overview

Coming soon …

Here just a small preview. You can draw more than one scrum stream into the same diagram – then you'll get the portfolio overview.

Demo-Portfolio

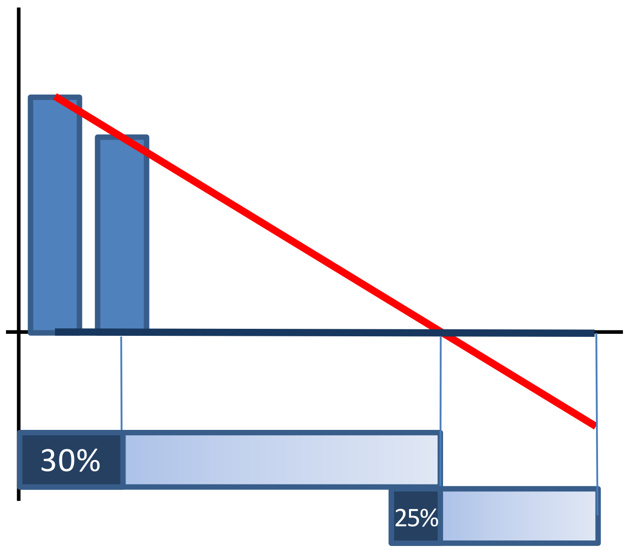
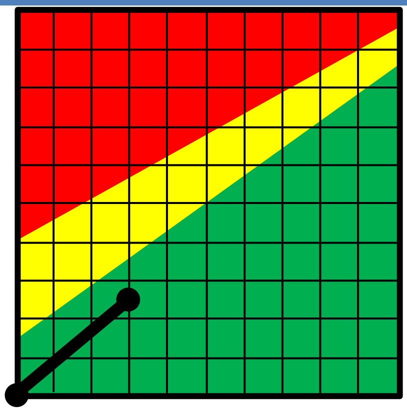
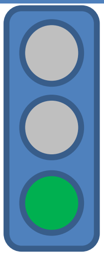
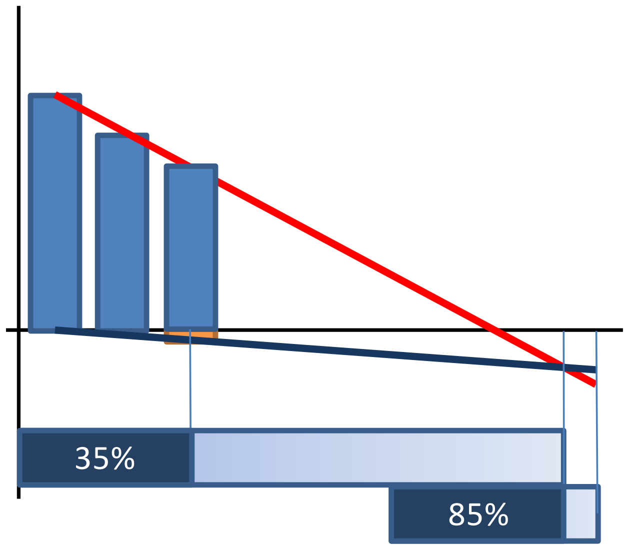
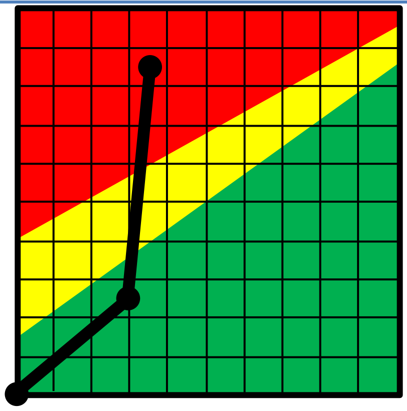
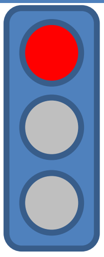
Report Date	10.06.12



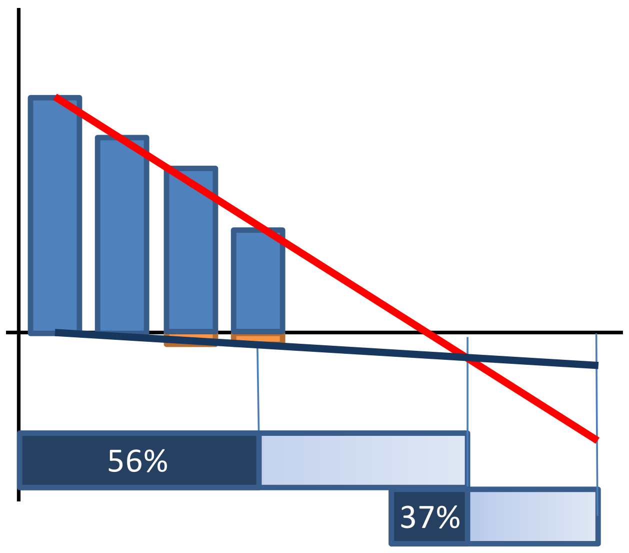
Stream		last Status			current Status			
(Release)		Date	%LCC	%BC	Date	%LCC	%BC	most penetrating task
good	g	01.05.12	70%	50%	15.05.12	80%	40%	none
bad	r	01.05.12	45%	60%	16.05.12	40%	75%	waiting on decision of pbb
ca va	g	02.05.12	30%	25%	10.05.12	60%	40%	waiting on fire wall
missing data	?	02.01.00			02.01.00			Warning: Data missing!
just one dot	y	02.01.00			19.05.12	50%	60%	first shot

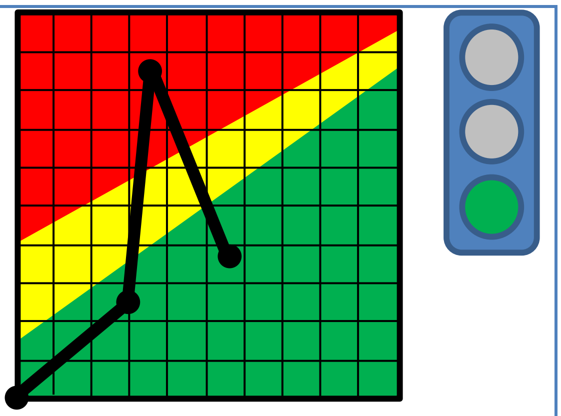
Complete example set of fever curves for a release

The following example is fictive and shows just the principle. Below you'll find some real ones and on the realiable-scrum.info website on the page "In Practice" you'll find even more and current ones.

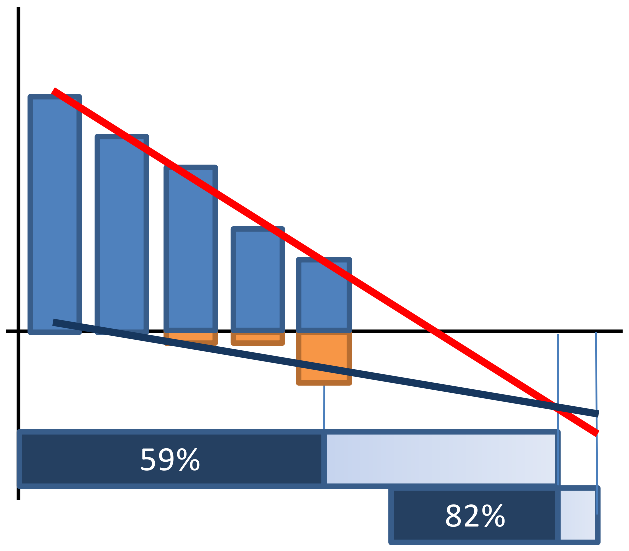
Enhanced Product Burndown Chart	Fever Curve	Traffic Light
Sprint 2 		
Sprint 3 		

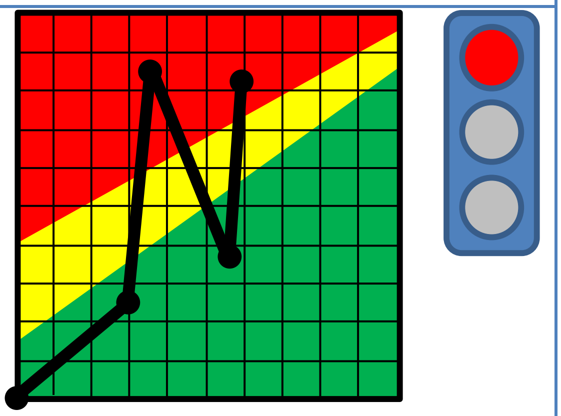
Sprint 4



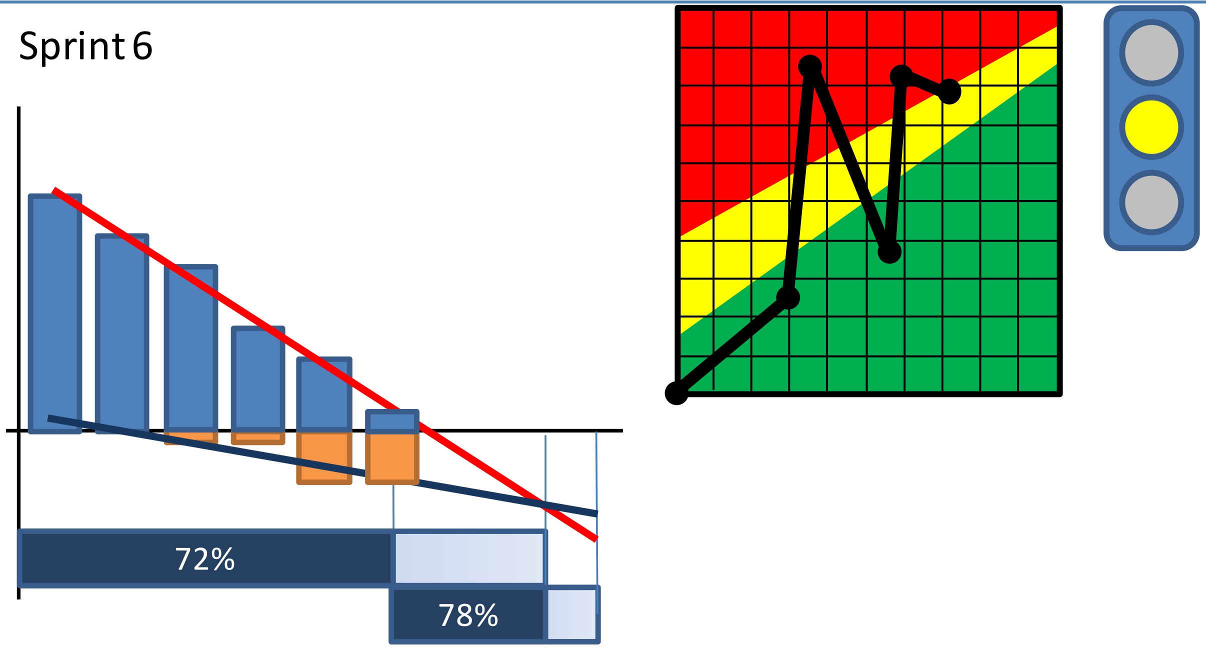


Sprint 5

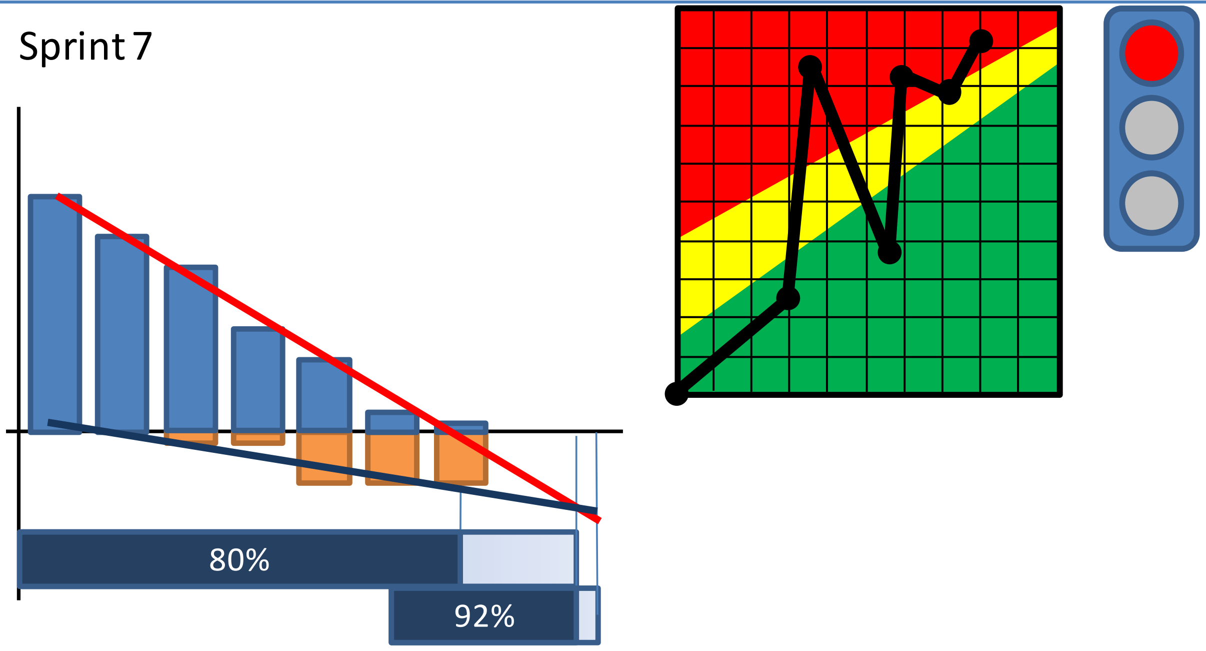




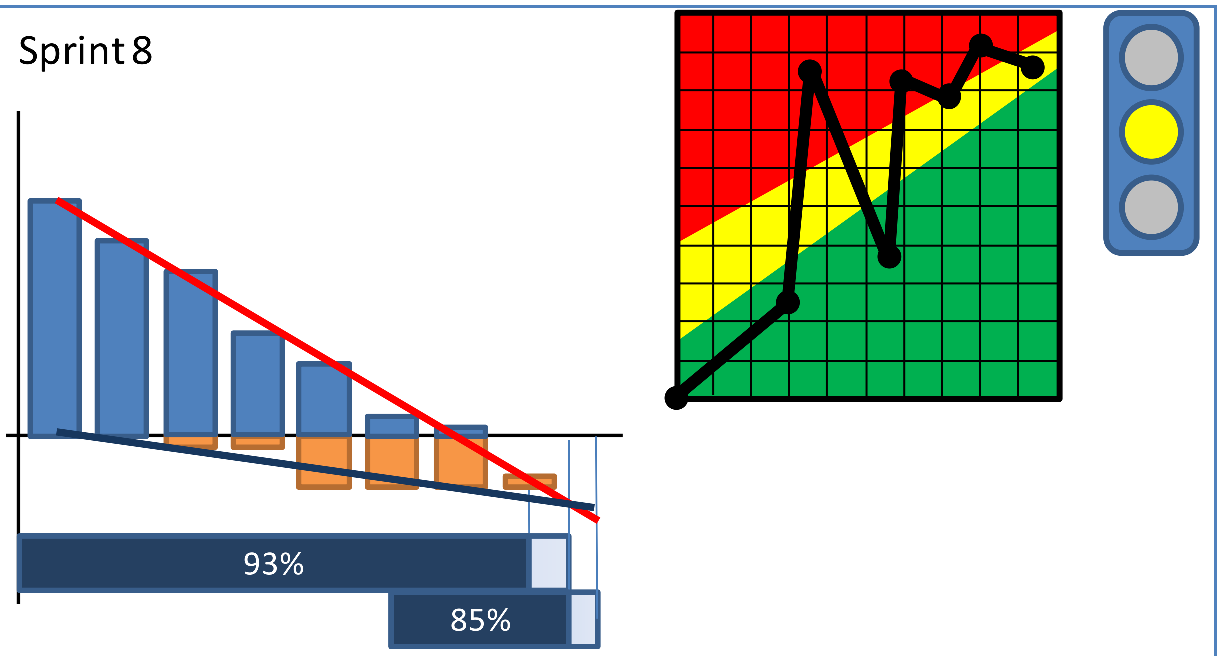
Sprint 6



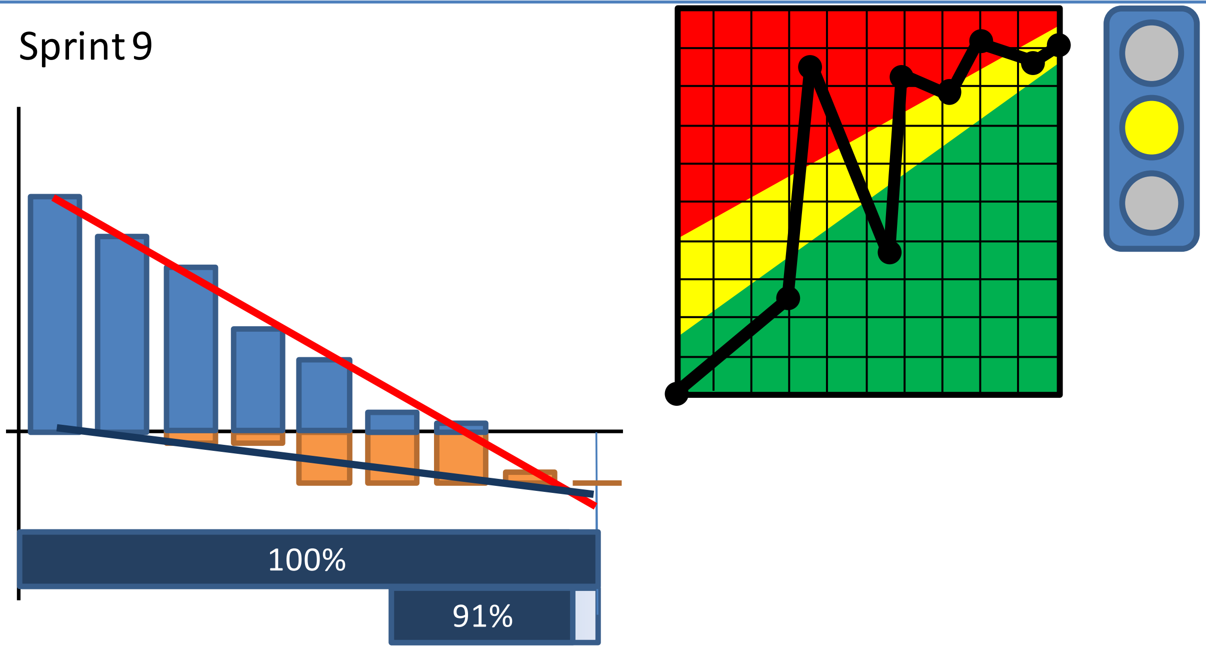
Sprint 7



Sprint 8

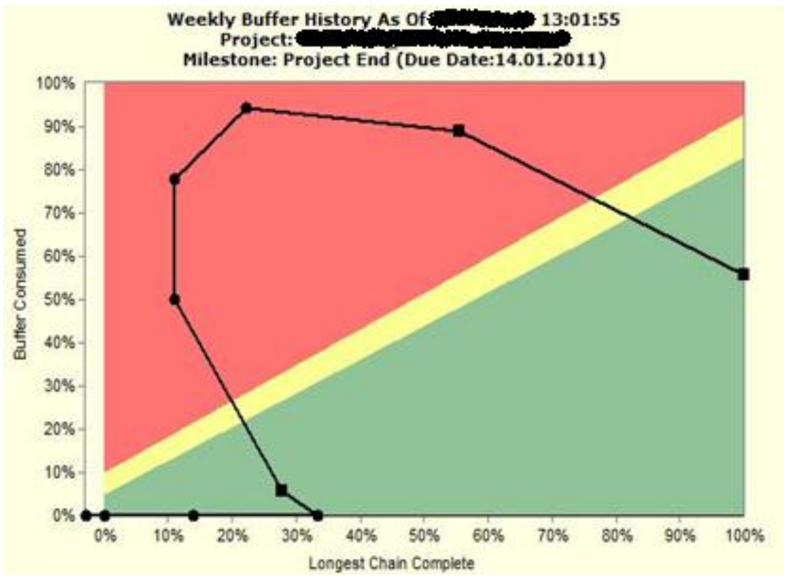


Sprint 9

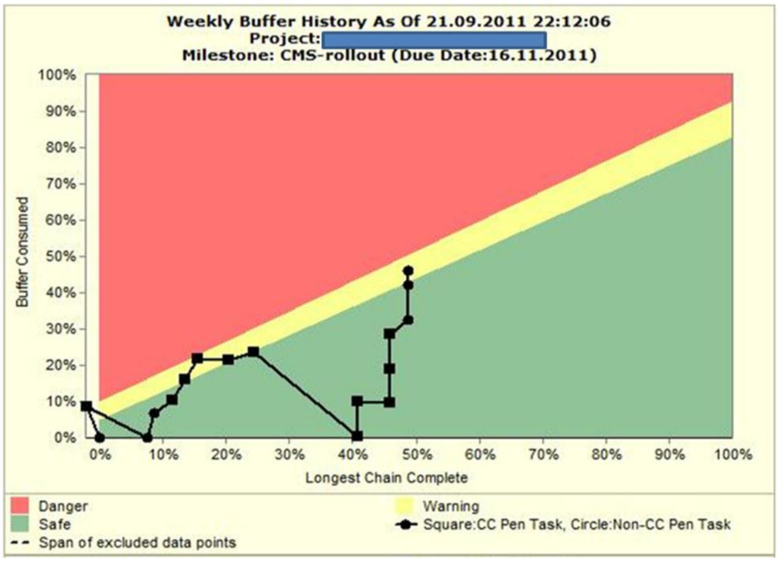


Some real examples

Out of the critical chain world.



Ein recht typisches Projekt in der Lernphase. Zuerst wird vermeintlich Fortschritt erzeugt, ohne Pufferverbrauch. Danach kommen die Probleme – wobei das Team lernt damit umzugehen und im Verlauf Wege findet die Geschwindigkeit zu erhöhen um das Gesamtergebnis zu sichern.



Ein Team/Release mit deutlich mehr Erfahrung. Der große Sprung in die grüne Zone ist keiner netto Geschwindigkeitssteigerung geschuldet hier wurde Architekturentscheidungen getroffen, die die Implementierung vereinfachten und das Backlog entsprechend angepasst.

Über die Speed4Projects.Net

Unternehmensgeschichte

Die Speed4Projects wurde im Jahr 2006 von Wolfram Müller gegründet und bietet seither Beratung zur Einführung von Critical-Chain-Projektmanagement und High-Speed-Projektmanagement an.

Wolfram Müllers langjährige Tätigkeit im Bereich Entwicklung und Projektmanagement in unterschiedlichsten Unternehmen hatte ihn mit einer Vielzahl unterschiedlicher Ansätze des Projektmanagements in Berührung gebracht. Viele dieser Ansätze hat er scheitern sehen und erst die Methoden rund um Critical Chain und High-Speed-Projektmanagement brachten sichtbare Erfolge. In Zusammenarbeit mit der Firma VISTEM konnte Wolfram Müller diesen Ansatz in vielen Unternehmen einführen. Im Bereich Erfolgsgeschichten finden Sie Stimmen der Kunden, die dies bestätigen.

Philosophie

Mit pragmatischen Ansätzen schnelle sichtbare Erfolge erzielen, das ist der Grundgedanke von Speed4Projects. Nicht von ungefähr kommt es, dass die Wahl des passenden Instruments auf die Methode Critical-Chain und High-Speed-Projektmanagement fiel. Viele Projektmanagement-Methoden setzen voraus, dass eine Firma ihre gesamten Prozesse umstellt und sich zusätzlich noch teure Software anschafft. Speed4Projects ermöglicht es Ihnen mit Ihrer bestehenden Projektmanagement-Infrastruktur weiter zu arbeiten und durch gezielte punktuelle Optimierung Ihre Projektsteuerung deutlich zu verbessern.

Geschäftsführung

Wolfram Müller, Jahrgang 1969, beschäftigte sich als Dipl.-Ing. Mechatronik und Dipl.-Ing. Maschinenbau zunächst mit Themen der Entwicklung, Fertigung und Prozessoptimierung. Die Werkzeuge des klassischen Projektmanagements lernte er im Rahmen seiner Tätigkeit als Projektmanager in der Medizintechnik kennen. Seit 1987 und parallel zum Studium entdeckte er als Freelancer in zahlreichen Software-Entwicklungsprojekten den Spaß an schnellen Projekten. Bis heute konnte er beide Seiten erst als Entwickler und später als Manager des Project Office der 1&1 Internet AG (mit ihren Marken Schlund+Partner, GMX sowie web.de) ausleben. Seit 2006 steht seine Erfahrung im Critical-Chain und High-Speed-Projektmanagement in Vorträgen, Veröffentlichungen und vor allem in Form von Beratung zur Einführung jedermann, zu Verfügung.

Erfahrung aus über 530 Projekten in Bereichen:
- > Softwareentwicklung
- > Medizintechnik
- > IT-Projekte
- > Organisationsprojekte

Für namhafte Unternehmen:
- > BOSCH
- > AT&T
- > BARD/angiomed
- > Porsche
- > S+P, web.de, GMX, 1&1

Wenn sie Fragen haben:

Telefon: **+49 171 565 1821**

eMail: Wolfram.Mueller@Speed4Projects.Net