

Italian Agile Day, Bologna,  
November 20, 2009

Peter Stevens  
Scrum Trainer & Coach  
Das Scrum Team  
Zürich – Vienna – Bremerhaven

[www.scrum-breakfast.com](http://www.scrum-breakfast.com)

[peter.stevens@dasScrumTeam.com](mailto:peter.stevens@dasScrumTeam.com)

+41 44 586 6450

Photo courtesy of [vizzzual.com](http://vizzzual.com)@flickr



1982-85 Software Engineer, Microsoft

2005 discovered Scrum

- NZZ Executive
- Publicjobs.ch
- Dynamic-Pricing.ch

April 2008, Independent Scrum Coach

- On-Site Scrum Training E/F/D

Certified Scrum Practitioner

Scrum Evangelist

- Scrum-Breakfast.com
- agilesoftwaredevelopment.com
- Lean-Agile-Scrum SwissICT Group

Peter Stevens, CSM, CSPO, CSP  
DasScrumTeam.com  
peter@sierra-charlie.com  
www.scrum-breakfast.com

© 2008 Peter Stevens

sierra-charlie.com

We are about to start a trip. I am your tour guide. I've done this trip a number of times myself. I want to show you some of the high & low points.

Mostly this talk is about experiences that have worked for me. These tools may also work for you. Try them out. Use them if they work for you. Change them or drop them if they don't. This is about getting results that work for you, your team and your project.

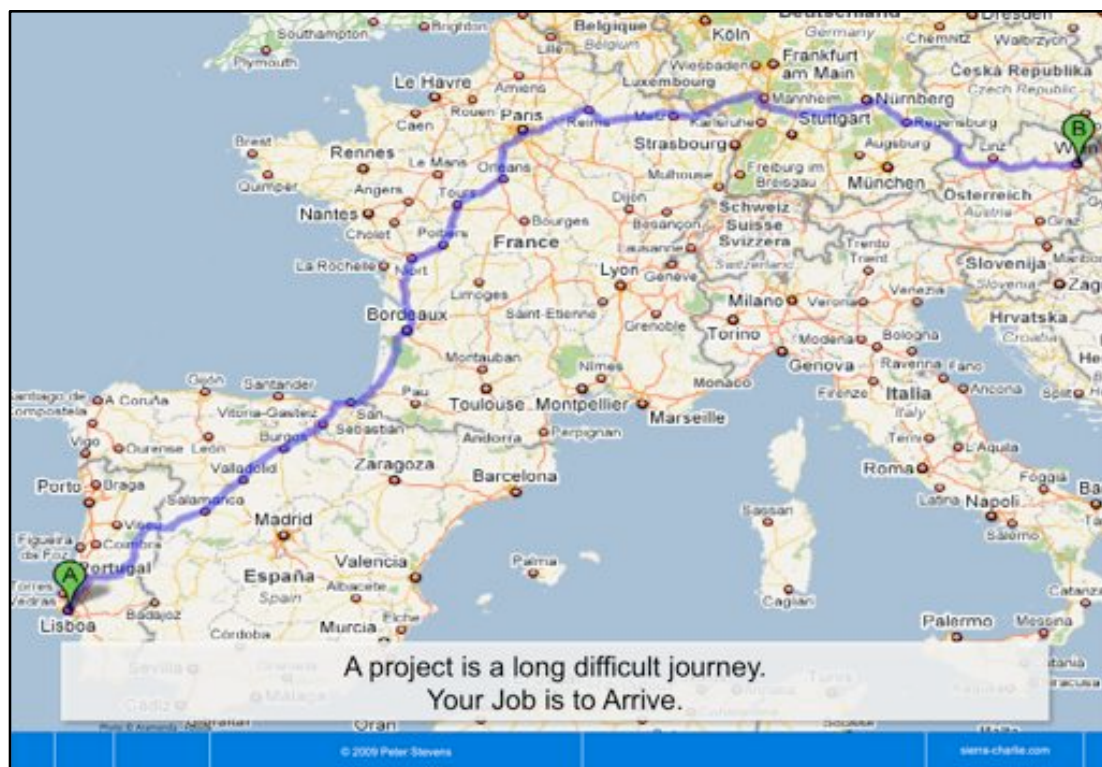
A few words about me:

Peter Stevens  
Scrum Trainer & Coach  
Das Scrum Team  
Zürich – Vienna – Bremerhaven

[www.scrum-breakfast.com](http://www.scrum-breakfast.com)

[peter.stevens@dasScrumTeam.com](mailto:peter.stevens@dasScrumTeam.com)

+41 44 586 6450

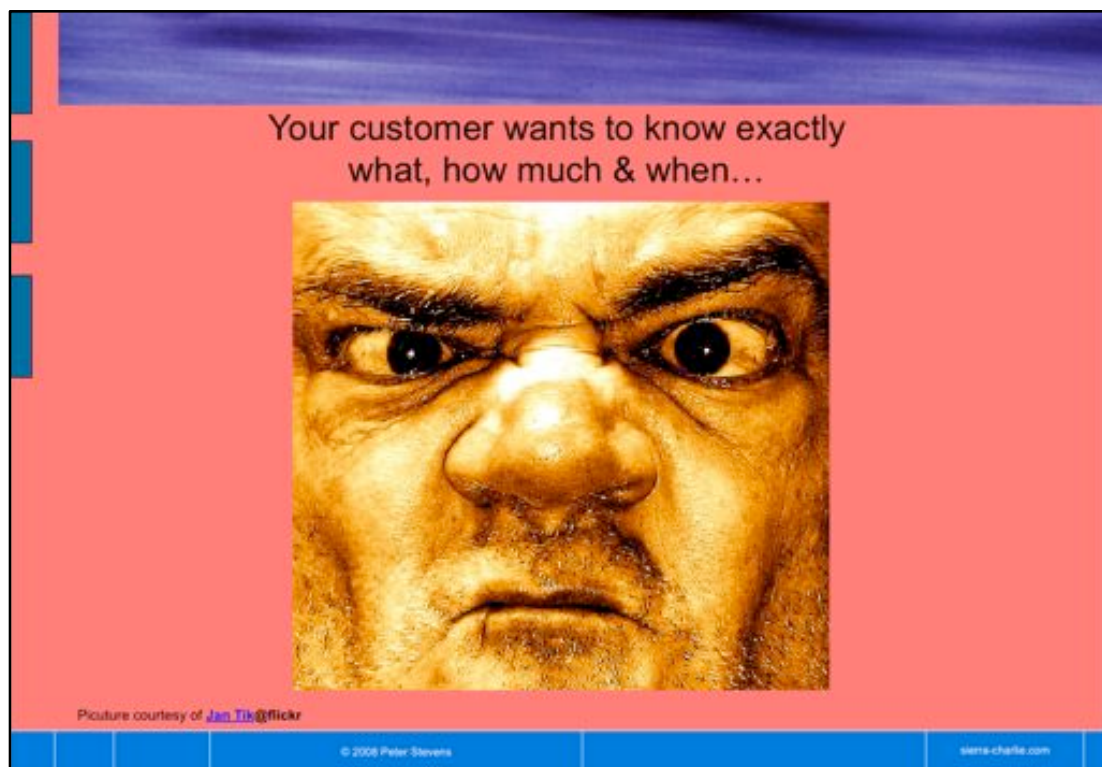


According to a recent study, nearly 2/3<sup>rd</sup>s of all non-agile projects and 30% of all agile projects are considered failures.

In my work as a coach project leader, I have rescued a number of “sinking ships.” I have done so by using Scrum as a management and by encouraging the teams to discover extreme programming.

How many of you are doing Scrum? XP? Kanban? something else agile? Something Non-Agile (“Classical, Robust, Waterfall, Cowboy). OK. I don’t want to tell you what approach to take. I would like to show you what worked from me and maybe you will find patterns which work for you.

Our trip from A to B, Concept to Cash is not easy. Think road movie. How difficult, depends on your team, your “bus”, your customer and you. I want to show you various techniques and practices to increase your chances of arriving,



The interesting questions is “why does the customer want a fixed price, deadline and/or scope.”

The customer may want it for emotional reasons – he doesn’t trust you.

The customer may need it for operational reasons - Legal requirements, some external date is fixed. Would you postpone the FIFA World Championships because the Web Site isn’t ready? The ticketing system? Any IT related issue?

The supplier may want to offer it for competitive advantages – the customer prefers perceived lower risk.



You want to win the bid,  
make money on the project, and have a happy customer.



Photo © Julian Rovagnati - Fotolia

© 2008 Peter Stevens

some-charlie.com

And you want the customer to be happy at the end of the project.

Which is more important? Fulfilling the spec or a happy customer?

Fulfilling the spec or achieving time & budget goals do not guarantee a happy customer!

Success means  
reducing the consequences of the risks of the project



© 2008 Peter Stevens

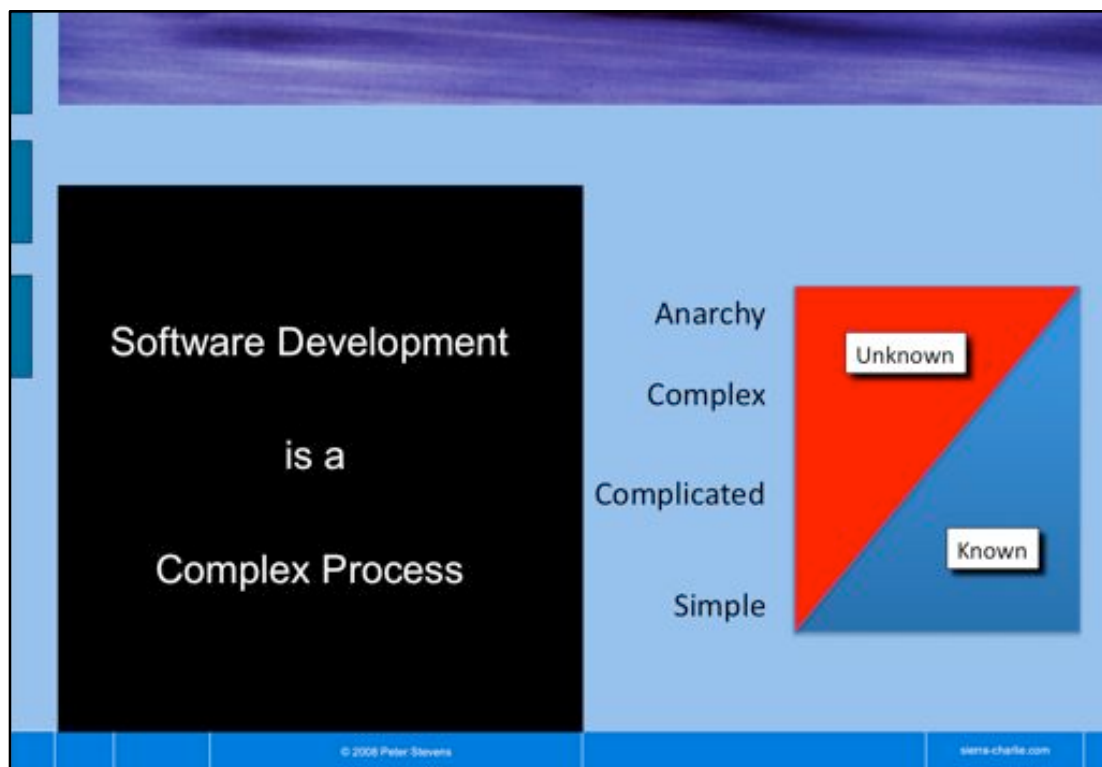
some-thing.com

Some things will go your way, some won't. The things that don't go your way, should not endanger the success of the project.



I want to talk to you about minimizing risks, how to manage the project. First we'll look at how to execute well, then how to plan for good execution, then look at how to combine planning and execution to minimize the risk that individual problems (most like undelivered functionality) will jeopardize the success of the project (or more importantly, the happiness of the customer).

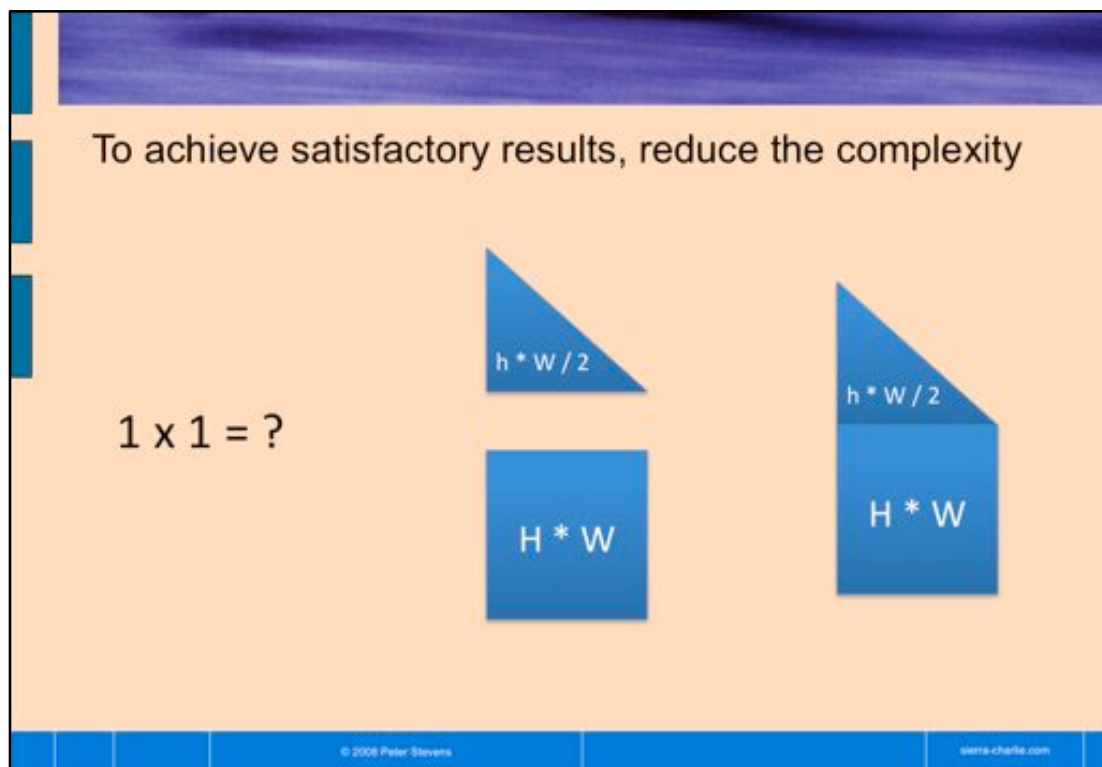




Simple is completely knowable. You don't have to plan to get dressed every morning. You just do it. (OK, maybe deciding which clothes to where is a challenge for some people...)

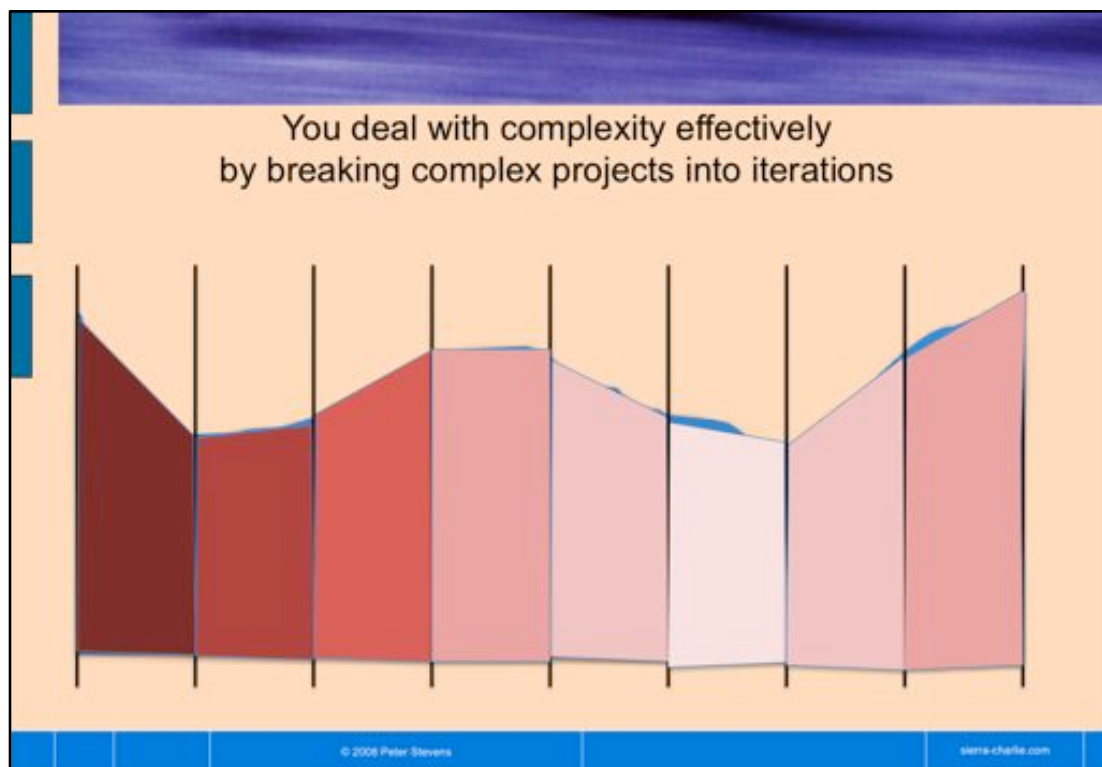
Complicated means 'mostly knowable' Building a car is enormously complicated, but the steps can be defined, the acceptance criteria for a Toyota Corolla with Air Conditioning are clear. So if you build the car (and the 10'000 nearly identical cars) correctly, the customer will be satisfied.

In complex projects, there is more known than unknown and feedback loops arise. Each software project is unique. The customer doesn't really know what he wants until he has seen it. Fitness for use is (or should be) the most important criteria for judging the results.



How do you prevent complexity from becoming chaos?

Back in 3<sup>rd</sup> Grade, you learned to multiply. Later you learned to use multiplication to calculate the area of simple shapes. You could even combine the simple shapes to calculate more complicated shapes.



Later, perhaps in college, you look at even more complicated shapes: How do mathematicians calculate the area of these shapes?

By slicing them into smaller shapes that resemble the simple forms we saw in the previous slide. The total area is simply the sum of the individual slices.

This is how Scrum, XP and other agile frameworks attack the software development process.



In each iteration (or sprint), only a small piece of the total functionality is implemented. Functionality should be 'done' (or 'done-done') by the end of the sprint.

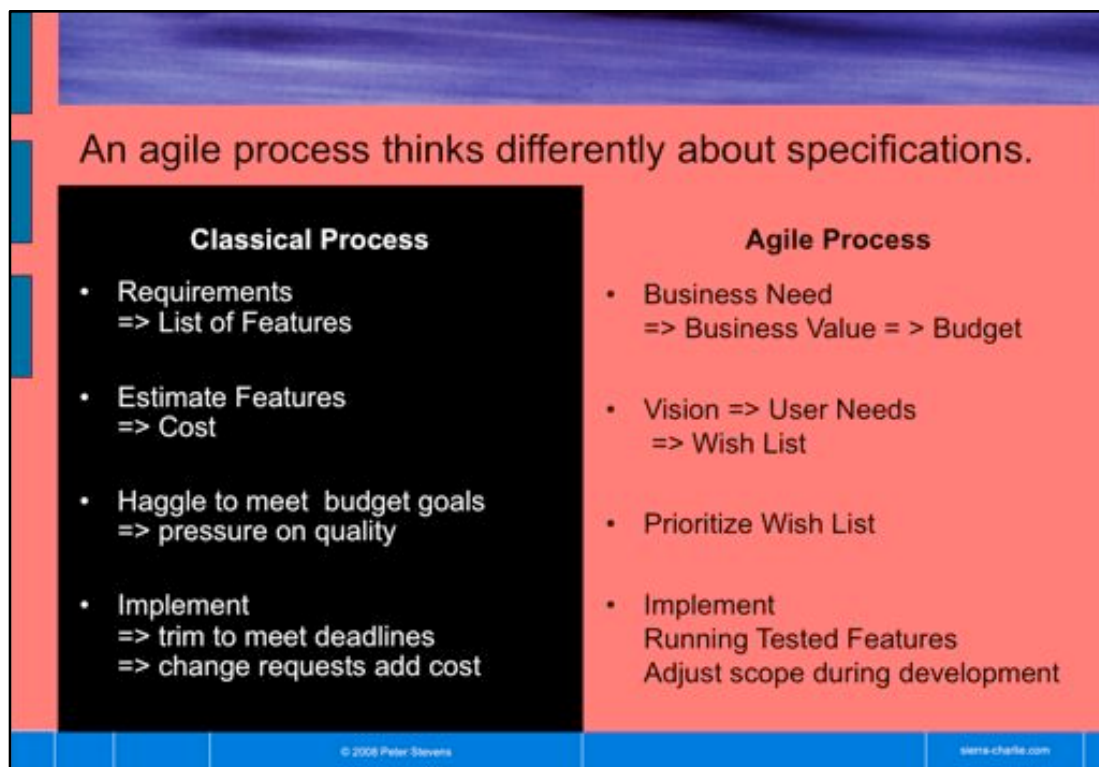
A small subset of the functionality is selected for implementation in sprint. All of the parameters of the project become frozen for the duration of the sprint.



I have found the metaphor of a “Sprint Contract” to be helpful in understanding (and sometimes enforcing!) the relationship between product owner and implementation team. This is not really a commercial contract, but simply the agreement between the Product Owner and the Team for one sprint.

The implementation team agrees to do its best to deliver an agreed on set of features (scope) to a defined quality standard by the end of the sprint. (Ideally they deliver what they promised, or even a bit more.) The Product Owner agrees not to change his instructions before the end of the Sprint. So a Scrum project simply is a series of mini projects with fixed parameters: Time (Sprint Length), Scope (Sprint Backlog), Quality (Definition of Done) and Cost (Team Size\*Sprint Length).

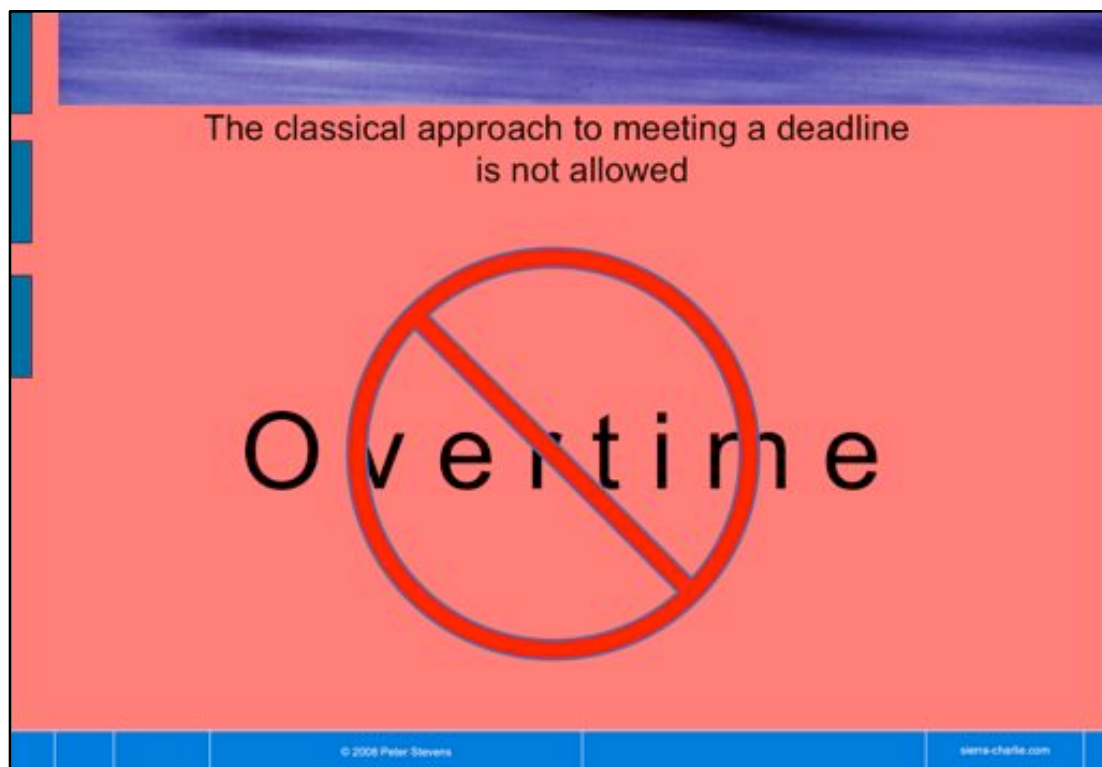
BTW – Quality is often overlooked in the traditional approach. “We don’t talk about quality, we just do it!?” But do you? Under Scrum and XP, there is an agreed upon definition of done. This defines the quality and technical standards of the team. As the project progresses and the team gets better, the definition of done should get more demanding, so that a higher level of quality is achieved.



Culture conflict number 1: Agile Development is driven by Business Value. The question of how much is the project worth is answered at the beginning of the process.

In a classical specification process, all requirements are created equal. There is no basis to prioritize. Hitting a target is about reducing expectations to hit the target. Scope and quality suffer unpredictably. The former may make the product unusable to its intended users and latter drives up the support and maintenance costs.





Cultural conflict Nr 2: Overtime is bad for quality, which raises maintenance costs and lowers productivity.

Sustainable pace discourages overtime


Quality is constant

Scope is variable

How do you deliver a fixed price contract in this context?

Expect Conflict! Better: define Must Features and Other Features. Only commit to the musts.

Work with the customer  
to identify  
what is really important



© 2008 Peter Stevens  
some-charlie.com

IF this customer is asking you for a fixed quote, you should understand why. What is really important to him and why? For instance, time & minimal functionality may be critical, budget of secondary importance.

Or he may only have \$XXX available.

By understanding what's really important to the customer, you can adjust you strategy accordingly.



Working with the customer is more important than contract.

The contract is important, and there are probably better contract forms than fixed price fixed scope.

Remember, the Manifesto says the items in both columns are important. Just the left column is more important.

## Let's Schedule the Daily Scrum

- Problem: Missing the Daily Scrum is expensive!
- Scrum Master travel times to Team Meeting:

9 Min

Tram Leaves	Tram Arrives
7:15	7:27
7:22	7:34
7:28	7:40

5 Min

Train Leaves	Train Arrives
7:10	8:20
7:39	8:54
8:10	9:20

5 Min

© 2008 Peter Stevens some-charlie.com

The team wants to have the daily scrum as early as possible. The ScrumMaster has 1:45 or so to get to the meeting and likes to sleep in. The penalty for arriving late is expensive (10 €), so he doesn't want to miss it!

When do you schedule the Daily Scrum?  
 Which train? Which Tram? When do you leave the house?  
 What happens if you miss a tram or if the tram/train is late.  
 Which steps can you influence?

My choice (this was a real situation). Daily Scrum at 9.30. Tram at 7.22, Train at 7.39, arrive at 8.54. Travel with Trotinette (scooter) which reduces walking time by factor of 2 or 3. If I miss the tram at 7:22, I will probably catch the train at 7.39 (because it is often late). Even if I don't, a 9.20 arrival leaves margin to make the Daily Scrum at 9.30, even if the train is delayed slightly.

Execution

How do you deliver  
what the customer  
asked for?




Photo courtesy of [thecmahoe@flickr](#)

© 2008 Peter Stevens

some-chaire.com

As team, it is your job to put presents under the Christmas tree.

A development team should produce running tested features (RTFs).

As a passenger, I want to be cool in the summer	OK
All seals correct	OK
Condenser Temperature < XXX°C	OK
Fan throughput = X l/min in low setting	OK
Fan throughput = Y l/min in high setting	OK
Cooled Air Temperature = Ambient - 10±2°C	OK

© 2008 Peter Stevens [www.charlie.com](http://www.charlie.com)

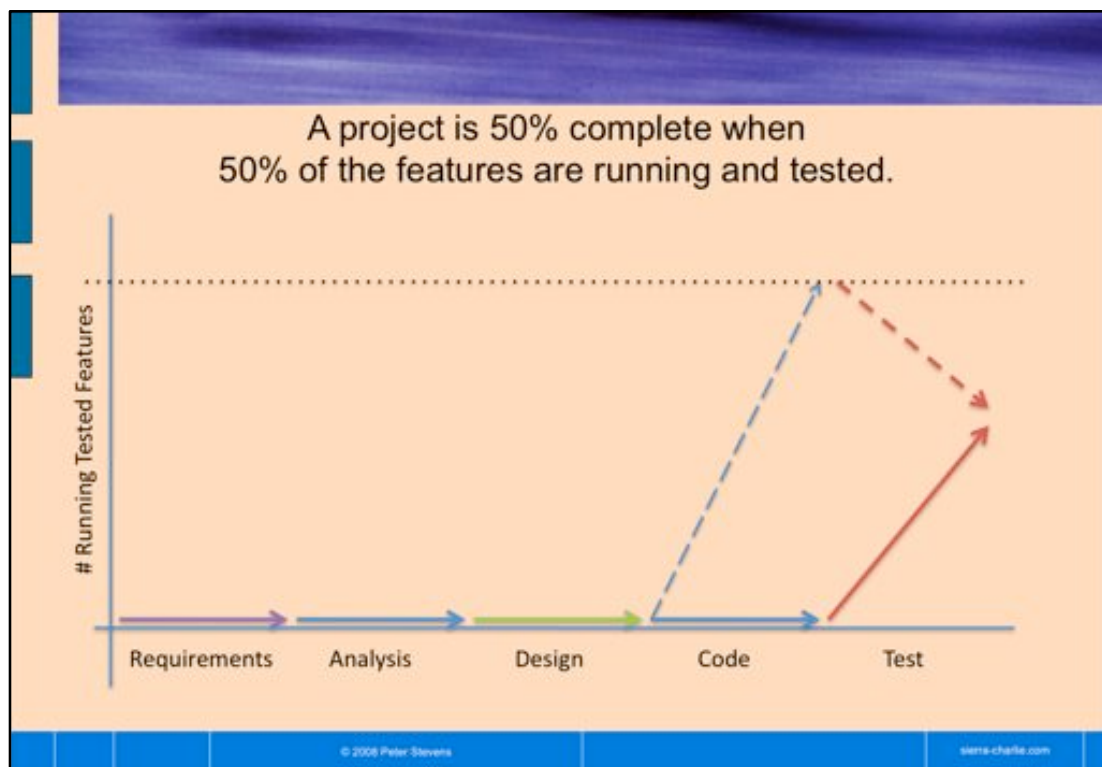
A feature tested during the first sprint should still be tested after the 6<sup>th</sup> sprint. So automation is essential.

Define Acceptance Tests before writing code to give the developers the right target to aim for. “Build the right thing.”

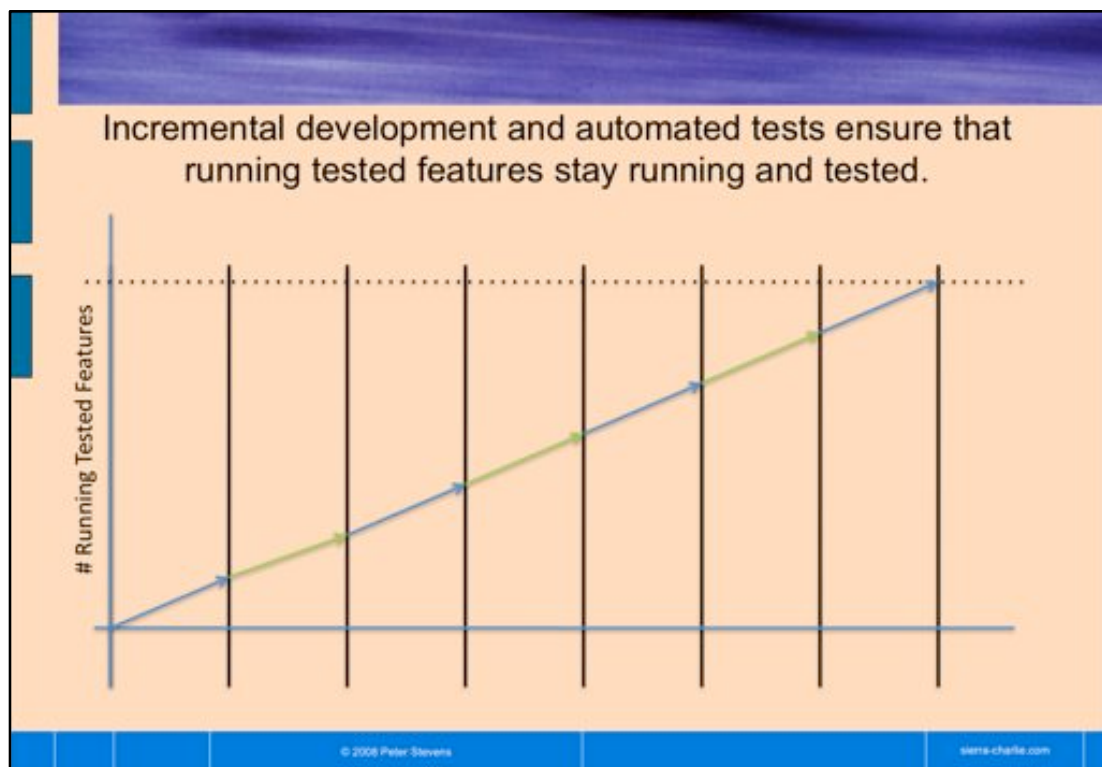
“Unit Tests” ensure conformity with developer intent and represent a safety net. “Build it right”. As developers, you need to do both: build the right thing and build it right.

A complete suite of automated tests helps ensure that running tested features stay running and tested, even when you create new versions of the software.

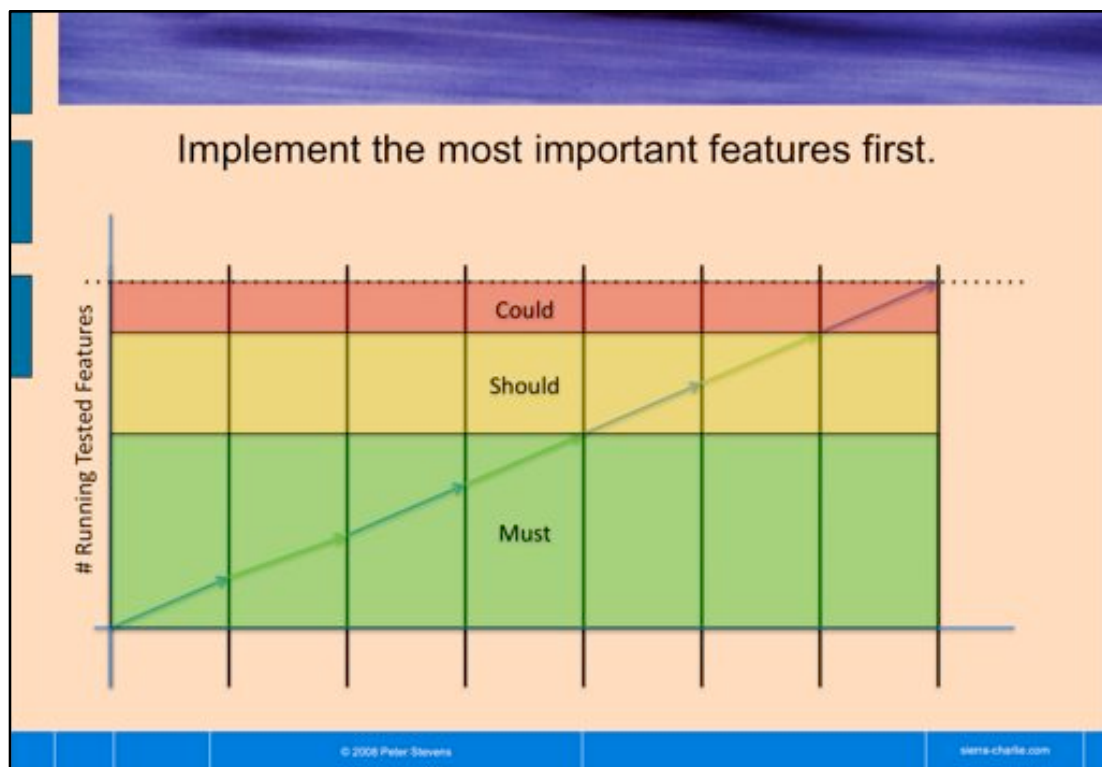




A classical approach only produces tested features at the end of a long development project. It is not easy to predict which features will be correctly implemented and which will be “prioritized” away in the pressure to get a release out.



Each sprint produces features which are of value for the customer or user.



First priority: achieve usability

Second priority: provide options, customizations, improve efficiency

Thirds priority: anything that could be considered luxury.

Get the system into a state where it achieves its business goal. Probably once the business goal is achieved, and maybe some optimizations and customizations are available, the customer will decide the ship the product, rather than work on low value nice-to-haves.

How do you plan a fix price project effectively?

Prerequisites for a credible quote

Techniques for better project plans

Putting it together: Step by Step to a good, agile project plan.

© 2008 Peter Stevens

[www.charlie.com](http://www.charlie.com)

What are your no go criteria for a fixed price bid?



Photo courtesy [Tony the Misfit](#)

© 2008 Peter Stevens

[some-charlie.com](#)

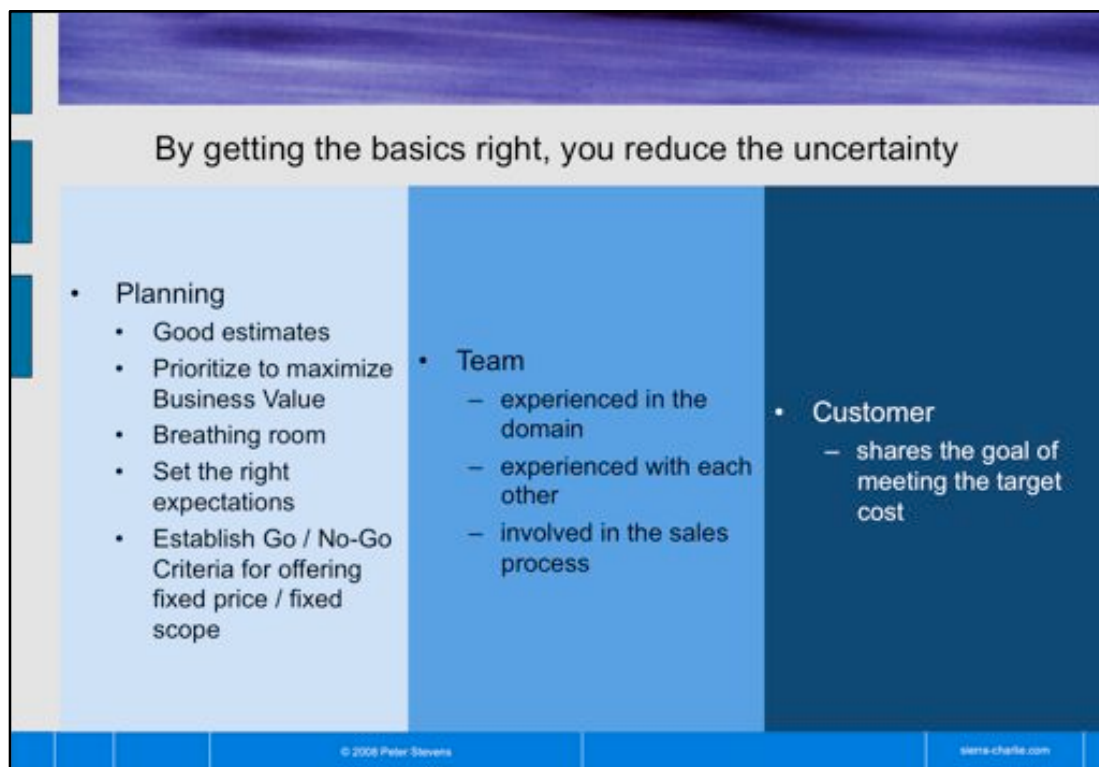
Are you willing to renounce business if your criteria are not met?

The courage to say no starts in the sales process.

If the risks are too high – you don't trust the customer, you don't know the domain or the technology, the margin is too low. Don't bid!

Better to end in a disaster than a disaster without end.

Better still, don't start something you know will end in a disaster!



Planning is about managing uncertainty. The more these items are on your side, the less risk you have.



## Techniques for planning a fixed price project

- Pre-Project: Specify requirements as user stories
- Involve the whole team in the sales process
- Use existing velocity "Yesterday's weather" to calibrate your estimates
- Use Scheduling and Feature Buffers to give your breathing room
- Prioritize to minimize the risks if things turn against you.

I have done projects to an aggressive cost ceiling. This is how I did it. This is how I would do it again if I were in the same situation.

## Specify functional requirements as user stories

As a <user> I want (to)	so that ( I can / to )
Sailor take a shower	be among people
Sailor surf the net	plan my shore leave
Sailor seasickness medications	work in bad weather
Soldier send email	coordinate my home leave
Soldier order care packages	enjoy life
Soldier post pictures on YouTube	pass time
Tailor cut fabric	make clothes
Tailor wash clothes	earn extra money
Tailor order fabric	make clothes
Tinker wash my hands	eat
Tinker download manuals	fix things
Tinker oil	silence squeaky wheels

© 2008 Peter Stevens

some-charlie.com

As a <user> I can do <something>

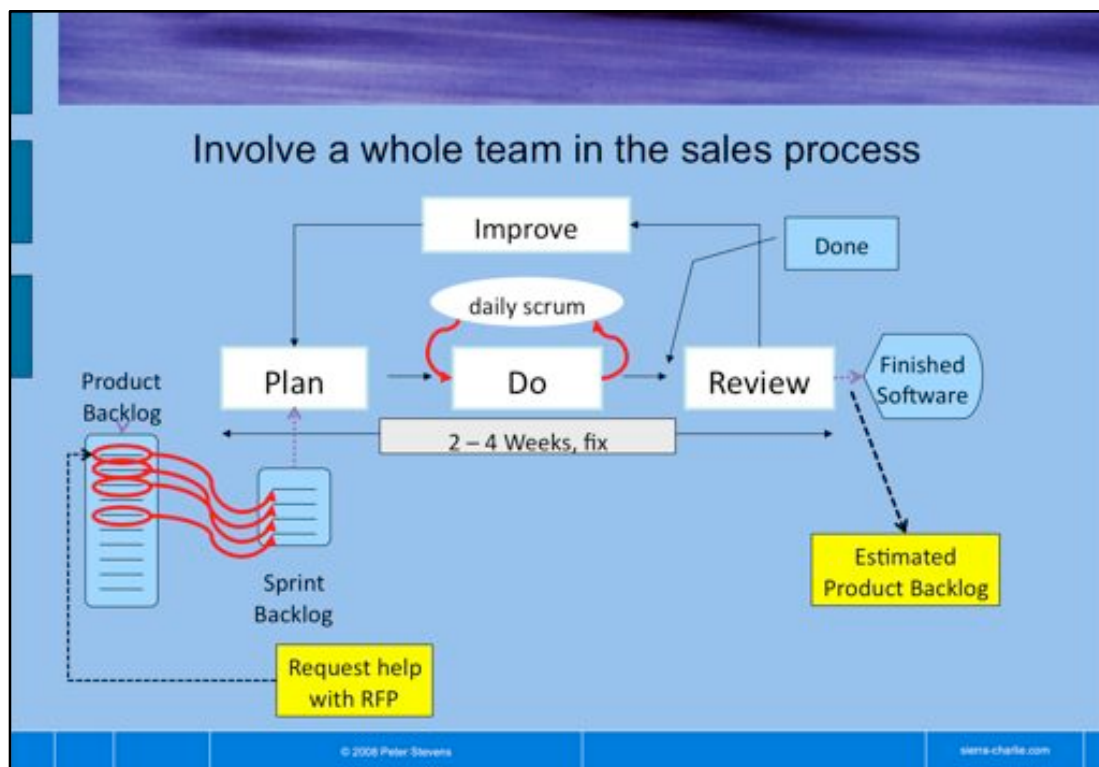
The N in INVEST means negotiable. User stories are relatively easy to understand. You can and should define acceptance criteria for each story (as you go).

How to implement is negotiable, presence is binary.

At the end, it is quite binary if someone has added a new user story.

What if the requirements are not user stories? Difficult problem, because the customer wants to see his specification in your quote. You can convert, but there is the usual problem of document chains. Better is if you can reach the customer before he has written the spec and convince him/her to write the spec as user stories.

If the spec cannot not be brought into a form where you can deliver functionality incrementally, consider not doing Scrum.



Team Estimates are better than individual estimates

Ideally, the request should come to the team through the normal sprint planning mechanism. You might not know which team will do the work.

By getting the team involved, you also get commitment to the estimates.

How much effort?

Rough Guess:

- ½ day to read RFP,
- ½ to 1 day to brainstorm stories,
- ½ to 1 day for planning poker
- = Total Calendar time, \* Team Size = Effort.
- = 1½ to 2½ days \* Team Size

You might consider using a subset of the team, but at least half, and all functions should be represented.

## The Team estimates the project

As a <user>	I want (to)	so that ( I can / to )	Size
Sailor	take a shower	be among people	8
Sailor	surf the net	plan my shore leave	8
Sailor	seasickness medications	work in bad weather	13
Soldier	send email	coordinate my home leave	8
Soldier	order care packages	enjoy life	3
Soldier	post pictures on YouTube	pass time	13
Tailor	cut fabric	make clothes	8
Tailor	wash clothes	earn extra money	8
Tailor	order fabric	make clothes	5
Tinker	wash my hands	eat	13
Tinker	download manuals	fix things	13
Tinker	oil	silence squeaky wheels	8
			<b>108</b>

You or better, the customer presents the stories. Use Planning Poker.

Why should the team estimate? Commitment to estimate. It's not (just) about whether the estimate is right or wrong (it will be wrong), but about how the team feels about the estimates and the process which created them.

## Prioritize Based on Business Value

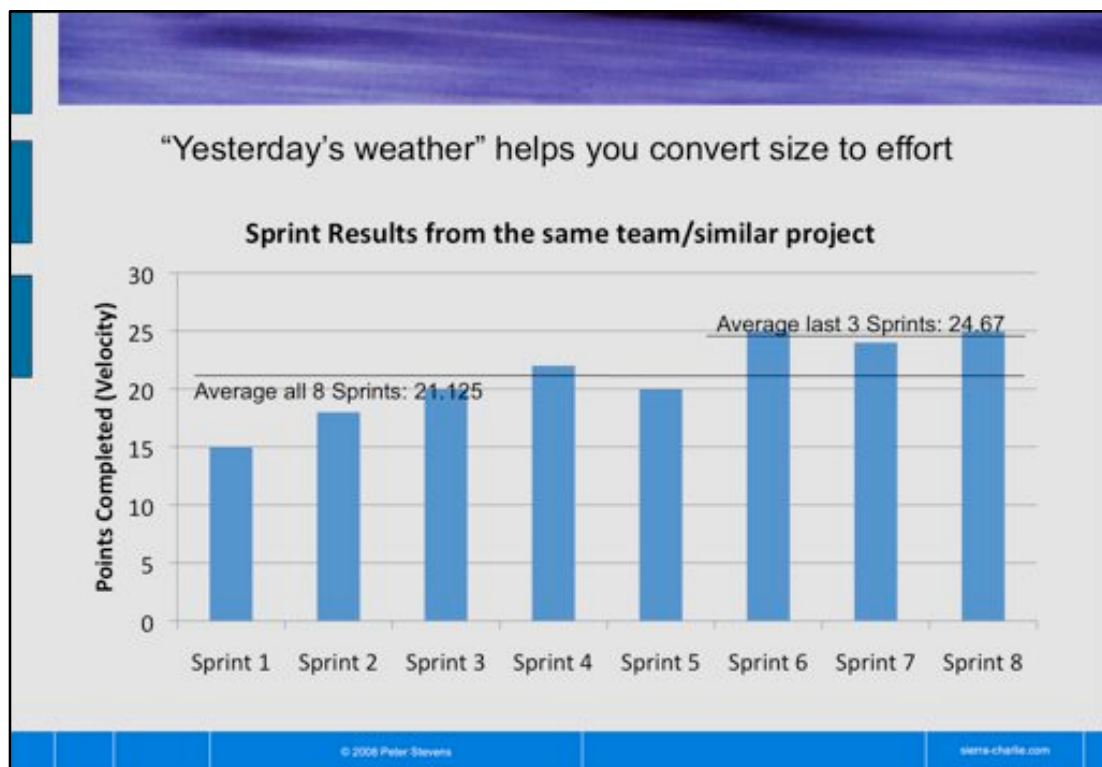
As a <user>	I want (to)	so that (I can / to )	Business Value	MoSCoW	Size
Tinker	download manuals	fix things	200	Must	13
Tailor	cut fabric	make clothes	150	Must	8
Tailor	order fabric	make clothes	150	Must	5
Tinker	silence squeaky wheels	to silence squeaky wheels	100	Must	8
Sailor	seasickness medications	work in bad weather	100	Must	13
Soldier	send email	coordinate my leave	80	Should	8
Tinker	wash my hands	eat	60	Should	13
Sailor	surf the net	plan my shore leave	50	Should	8
Sailor	take a shower	be among people	40	Could	8
Soldier	receive care packages	enjoy life	40	Could	3
Soldier	post pictures on YouTube	pass time	20	Could	13
Tailor	wash clothes	earn extra money	10	Won't	8
			1000		108

© 2008 Peter Stevens

www.charlie.com

MoSCoW:

- Must
- Should
- Could
- Won't (well, would be nice)
- Yellow & Green (in this example) represent 70% of work, 89% of value



- working with an experienced team lowers the uncertainty
- how much uncertainty factor to apply for a new team?
- Velocity is also known as focus factor



## Calculate estimates in Sprints based on previous velocity

I want (to)	Size	Cum	Number of Sprints by Velocity		
			21	25	14
download manuals	13	13	1	1	1
cut fabric	8	21	1	1	2
order fabric	5	26	2	1	2
silence squeaky wheels	8	34	2	2	3
seasickness medications	13	47	3	2	4
send email	8	55	3	3	4
wash my hands	13	68	4	3	5
surf the net	8	76	4	3	6
take a shower	8	84	4	4	6
receive care packages	3	87	5	4	7
post pictures on youtube	13	100	5	4	8
wash clothes	8	108	6	5	8
	108				

### Given

- Size of team = 5
- Sprint Duration = 2 Weeks (10 Working Days)
- Number of Sprints

### Calculate

- Estimated Duration = number of sprints \* sprint duration
- 6 sprints \* 2 weeks / sprint = 12 Weeks
- 12 Weeks \* 5 days / week \* 5 people = 300 person days
- Maybe deduct 15% for vacations, sickness, holidays, etc

How do you  
quantify and minimize  
the risk in the project?

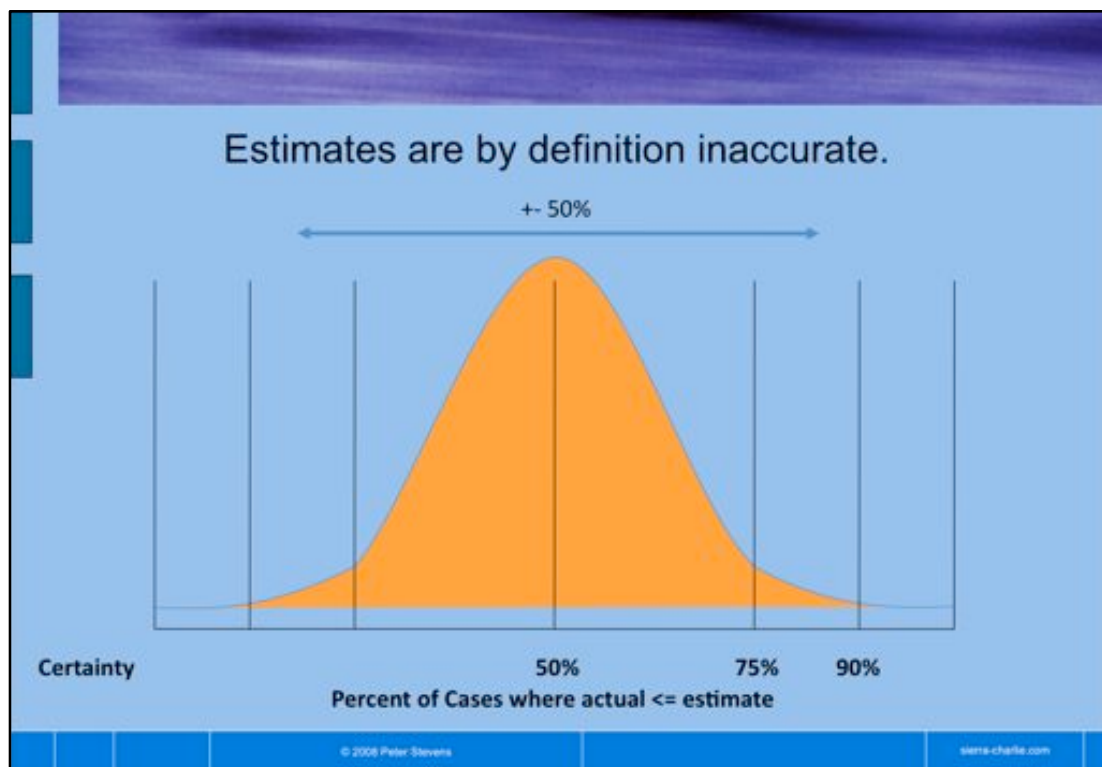
An estimate offers a 50/50 chance.

Improve your odd of finishing the project  
within budget by considering worst  
cases.

Use buffers and priorities to minimize  
risks

© 2008 Peter Stevens

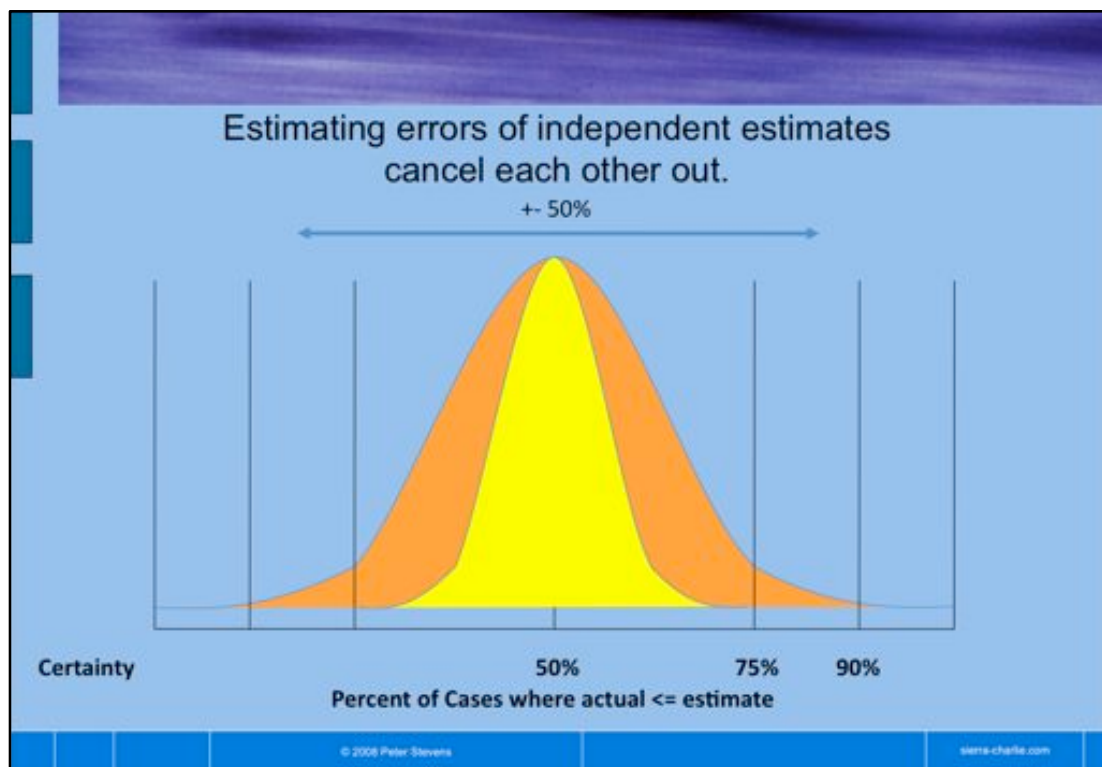
some-charlie.com



Assuming the developers understand the subject matter and the technology....

Average case has a 50/50 chance of being completed within the estimated time.

Worst case has 90% chance of being completed in less than the expected time.



If you sum 20 or more individual estimates at  $\pm 50\%$ , the resulting project error is  $\pm 20\%$

Conclusion -> break down the stories into smaller chunks.

Use Worst Case Estimates to Calculate Project Worst Case				
I want (to)	Size E(50)	Worst Case E(90)	Delta E(90) - E(50)	Delta ^ 2
download manuals	13	20	7	49
cut fabric	8	13	5	25
order fabric	5	8	3	9
silence squeaky wheels	8	13	5	25
seasickness medications	13	20	7	49
send email	8	13	5	25
wash my hands	13	20	7	49
surf the net	8	8	0	0
take a shower	8	13	5	25
receive care packages	3	5	2	4
post pictures on youtube	13	20	7	49
wash clothes	8	13	5	25
	108	166		334
			Risk Buffer	18 =SQRT(334)
			+ Basic Estimate	108
			= Project Estimate	126

If a worst case estimate give you 90% probability on the story, how do you get 90% probability on the project?

Planning Poker usually gives one value per story even though the team will often want to specify a range. Remember, each estimate is  $\pm 50\%$ , so if a 5 turns out to be either a 3 or and 8, it was correct. Alternatives for the worst case:

1. After two or three attempts, write down middle and high estimate from the team
2. Let the team converge on one number, then use the next value in the Fibonacci / Cohn Scale as worst case.

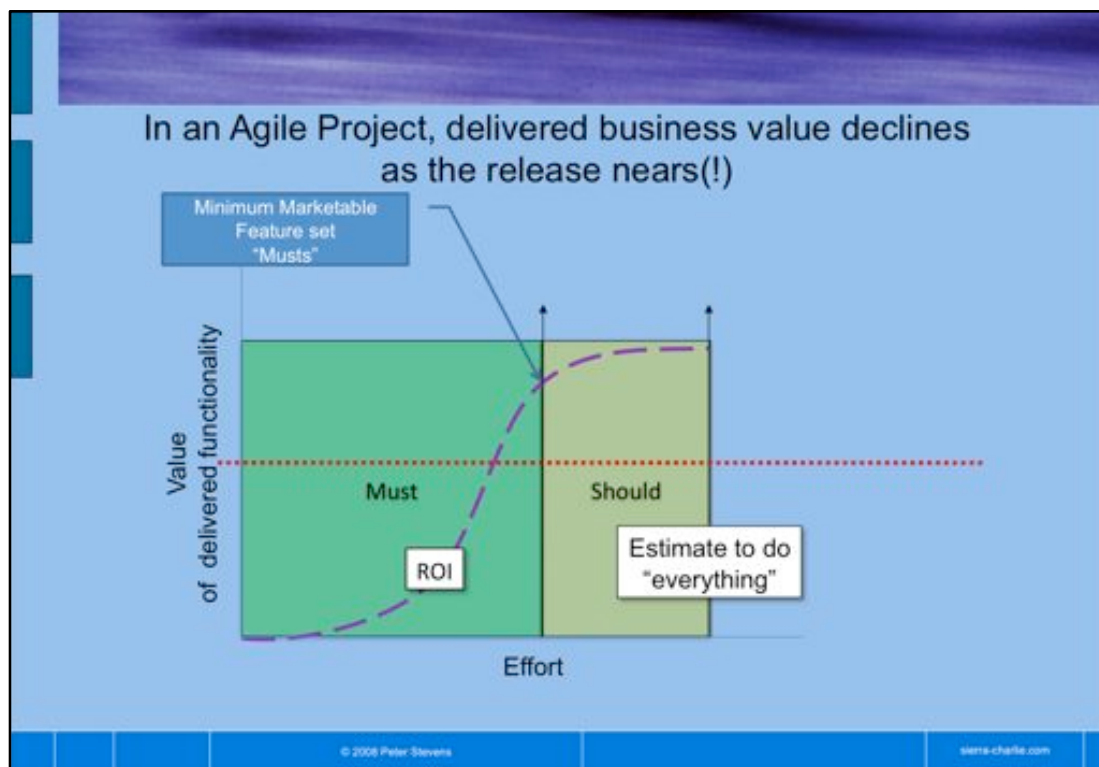
Simply summing the worst case estimates is (usually) too pessimistic. According to statistics,  $\text{SQRT}(\text{Sum}(\text{delta}^2))$  should be 90% reliability for the project.

For more details, see Source: Mike Cohn, Agile Planning & Estimating

## Use Feature & Schedule Buffers to Give Breathing room

I want (to)	Size E(50)	Worst Case E(90)	Delta E(90) - E(50)	Delta ^ 2	
download manuals	13	20	7	49	Commit
cut fabric	8	13	5	25	
order fabric	5	8	3	9	
silence squeaky wheels	8	13	5	25	
seasickness medications	13	20	7	49	
send email	8	13	5	25	Feature Buffer
wash my hands	13	20	7	49	
surf the net	8	8	0	0	
take a shower	8	13	5	25	Feature Buffer
receive care packages	3	5	2	4	
post pictures on youtube	13	20	7	49	
wash clothes	8	13	5	25	
	108	166		334	
			Risk Buffer	18	=SQRT(334)
			+ Basic Estimate	108	
			= Project Estimate	126	

Feature buffers are not margin that you can compromise on to win the project!



Rule of thumb: Musts should be no more than 60 to 70% of the project.

Rule of thumb: If you have problems delivering everything, your customer will be happier with you if he's got what he really needs and the rest is seen as less important.



If you're going to fall short,  
fall short on unimportant functionality.

The graph illustrates the relationship between the value of delivered functionality and the effort required to deliver it. The y-axis represents the 'Value of delivered functionality', and the x-axis represents 'Effort'. A horizontal red dotted line serves as a threshold. The area under the 'Musts' curve to the left of this threshold is green and labeled 'Musts  $\leq 70\%$ '. The area under the 'Features' curve to the right of this threshold is light green and labeled 'Should'. A box labeled 'Should's' with arrows points to the 'Should' region. The x-axis is divided into 'Effort' (left of red line) and 'Estimate to do "everything"' (right of red line).

© 2008 Peter Stevens  
www.charlie.com

For each user story, you defined acceptance criteria.  
Use them to declare the project 'done'

I want (to)	Acceptance Tests	
	defined	status
download manuals	yes	passed
cut fabric	yes	passed
order fabric	yes	passed
silence squeaky wheels	yes	failed
seasickness medications	yes	passed
send email	yes	passed
wash my hands	yes	passed
surf the net	yes	passed
take a shower	yes	failed
receive care packages	yes	not run
post pictures on YouTube	no	n/a
wash clothes	no	n/a

Acceptance criteria for each story should be defined shortly before implementation is to begin.

For example: As a sailor, I can take a shower.

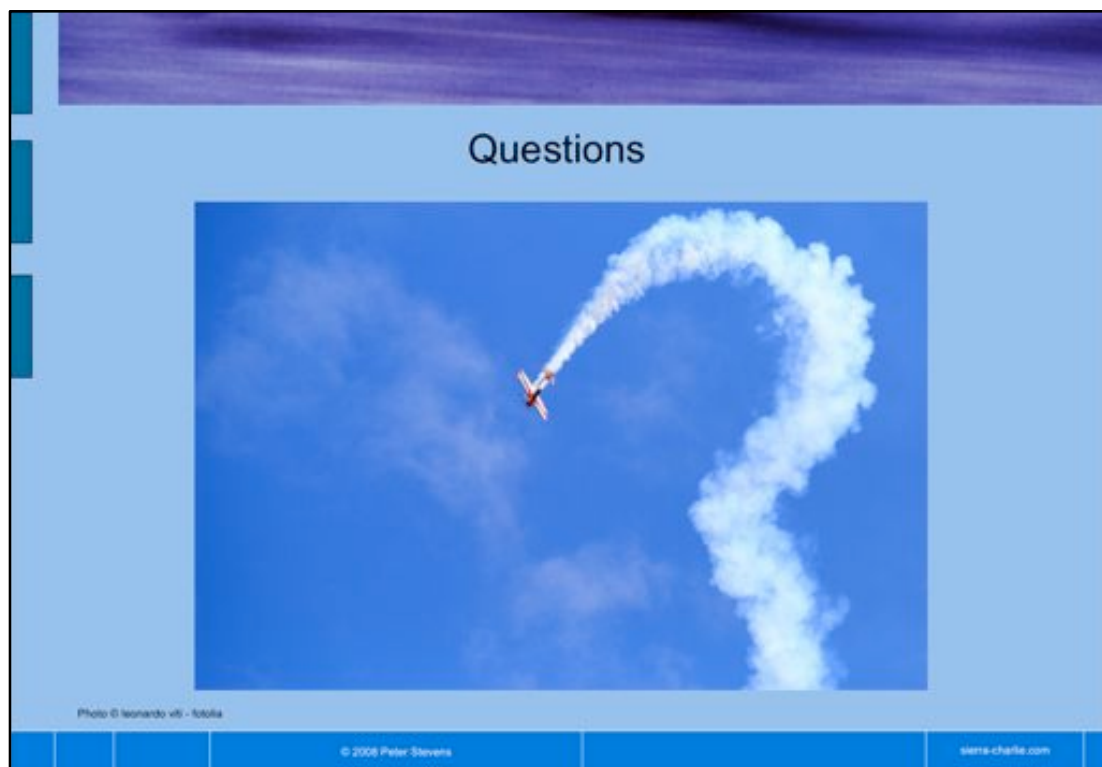
Acceptance criteria

- Water comes down from above my head
- Hot and cold water
- Shower curtain for privacy
- Drain for collected water

What about a detachable shower head? If it did not get included in the acceptance criteria, it becomes a new story. Depending on your contract, this might be a new story or something else might have to be removed.

## Summary & Suggestions

- Work with an experienced team
- Make sure the customer shares the goal
- Look for wiggle room / consider more cooperative models
- Pre-Project to define project in user stories.
- Estimate "normally"— and add buffers (time / scope)
- Deliver quality (automated testing & delivery)
- Deliver top priority business value first
- Ensure transparency about cost & progress
- Raise issues early
- Work with the customer



This talk was based on experience from several projects.

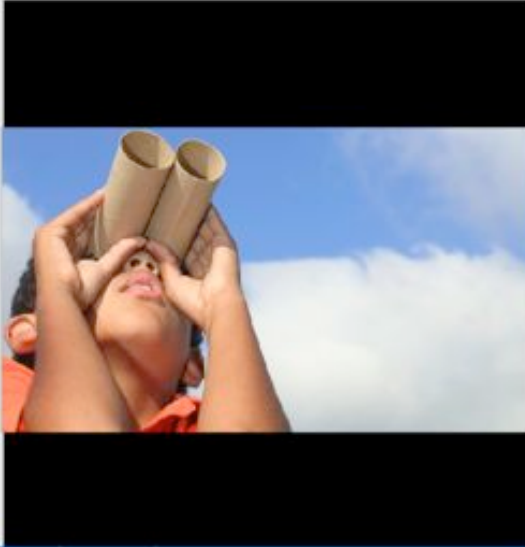
In the last of these projects, the customer first asked us 'what would it mean to modernize our [www.XYZ.com](http://www.XYZ.com) site with a map based search?'. In the pre-project, we looked at 2 aspects: 1) what is the competition doing? Features, information, Architecture. 2) we used User Centered Design to identify and prioritize potential users, their goals, and the features they need to achieve those goals. We gave the customer a list of user stories with 3 colors: green – you must have these to be competitive in the market (Cost 100). Purple – with these you will be considered a strong player. Blue – your site will be a technology leader (cost 140).

The customer went back to talk to his customer, came back a few months later, and said, we'd like to do the green stories, but we don't have 100. We have 80. Can you do it? It was a challenge, but we had worked long together as a team and the customer. We trusted our estimates (1 SP ~ 2.5 Person Days) and knew we could count on the customer to work constructively to stay within the cost ceiling.

User stories proved surprising useful for navigating the change request game. It was clear when the customer wanted a new feature and there were no 'et ceteras' for sneaking in extra features without paying for them. We finished the project on the day we had committed to 6 months before. When the project was finished, the actual cost was closer to 90 (due to additional stories which the customer wanted), and when his customer saw the product, they had some additions as well. These were all extra costs.

The customer and customer's customer were both very happy. The project met it's business goals, and the small cost overruns were not an issue. I met the customer's customer (VP of E-Commerce for a major Swiss Newspaper). Total satisfaction! Our sub-project was the only one that finished on time and we had completely understood the business, so it was a great product.

## Further Reading



- Mike Cohn
  - Agile Estimating and Planning
- Mary & Tom Poppendieck
  - Lean Software Development: An Agile Toolkit
- Blogs
  - Peter Stevens  
<http://scrum-breakfast.com>  
[agilesoftwaredevelopment.com](http://agilesoftwaredevelopment.com)
  - Jeff Sutherland  
<http://tinyurl.com/cuddvn>

Bon Voyage!



Peter Stevens  
peter@sierra-charlie.com  
+41 44 586 6450

© 2008 Peter Stevens

sierra-charlie.com