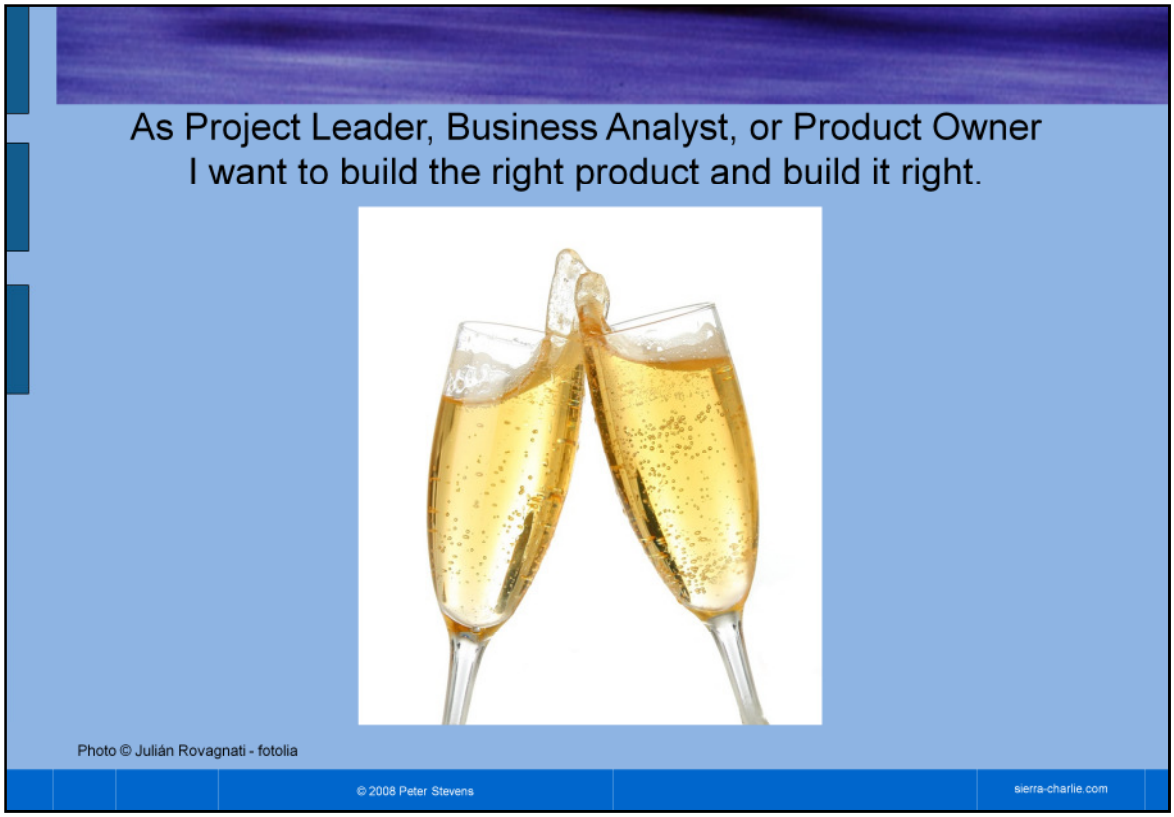

Cash
Concept
A
B
Zürich, April 1, 2009
peter@sierra-charlie.com
© 2009 Peter Stevens
sierra-charlie.com


From Mandate to Acceptance:
How do I ensure that the developers build what I really need?
Peter Stevens
Scrum Breakfast #15
Zürich, April 1, 2009
© 2009 Peter Stevens
May be redistributed under
Creative Commons "by-nd" License
© 2009 Peter Stevens
sierra-charlie.com

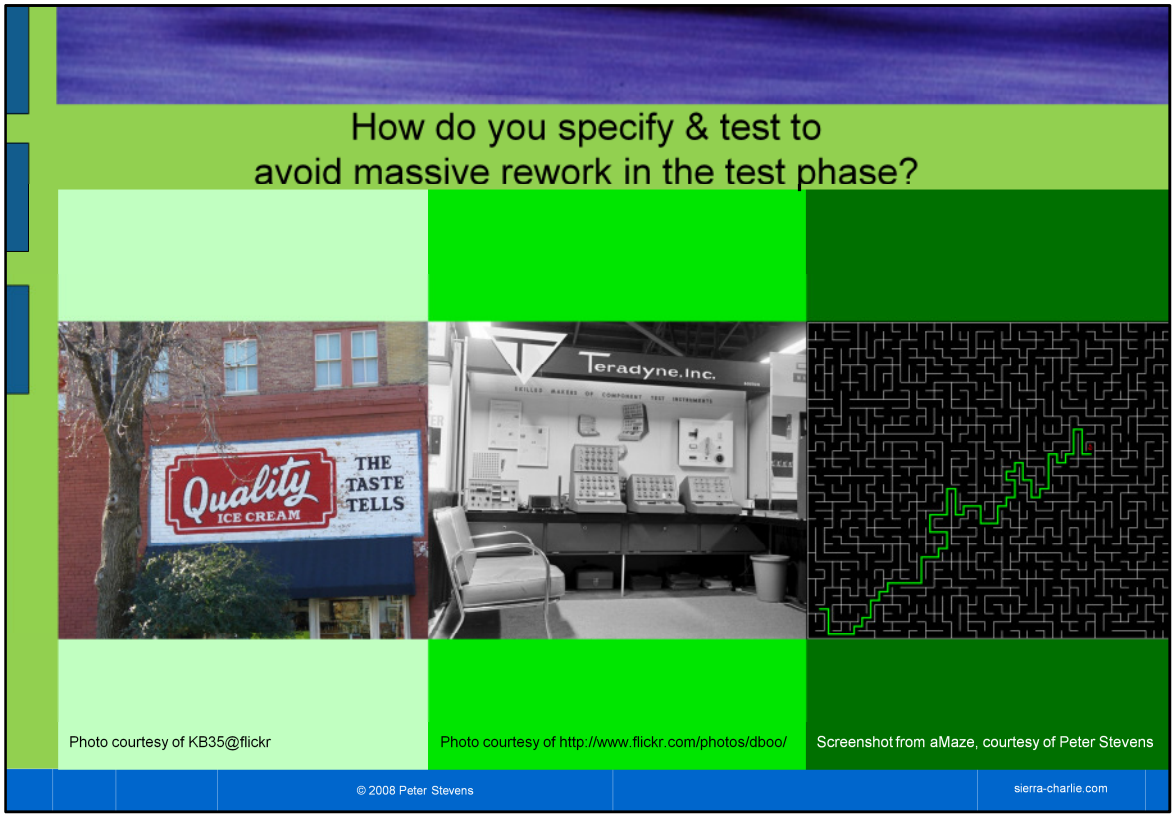


Everyone in the project as their definition of success. Developers was to learn and have fun. The Project Leader wants to satisfy the parameters of the assignment: Scope, Quality, Time, & Cost. The sponsor wants to use the results to achieve some business goal, usually to generate income or save costs. To do that, s/he wants software that meets the needs of the users or customers and works properly.

This talk is about building a product which meets those two goals.

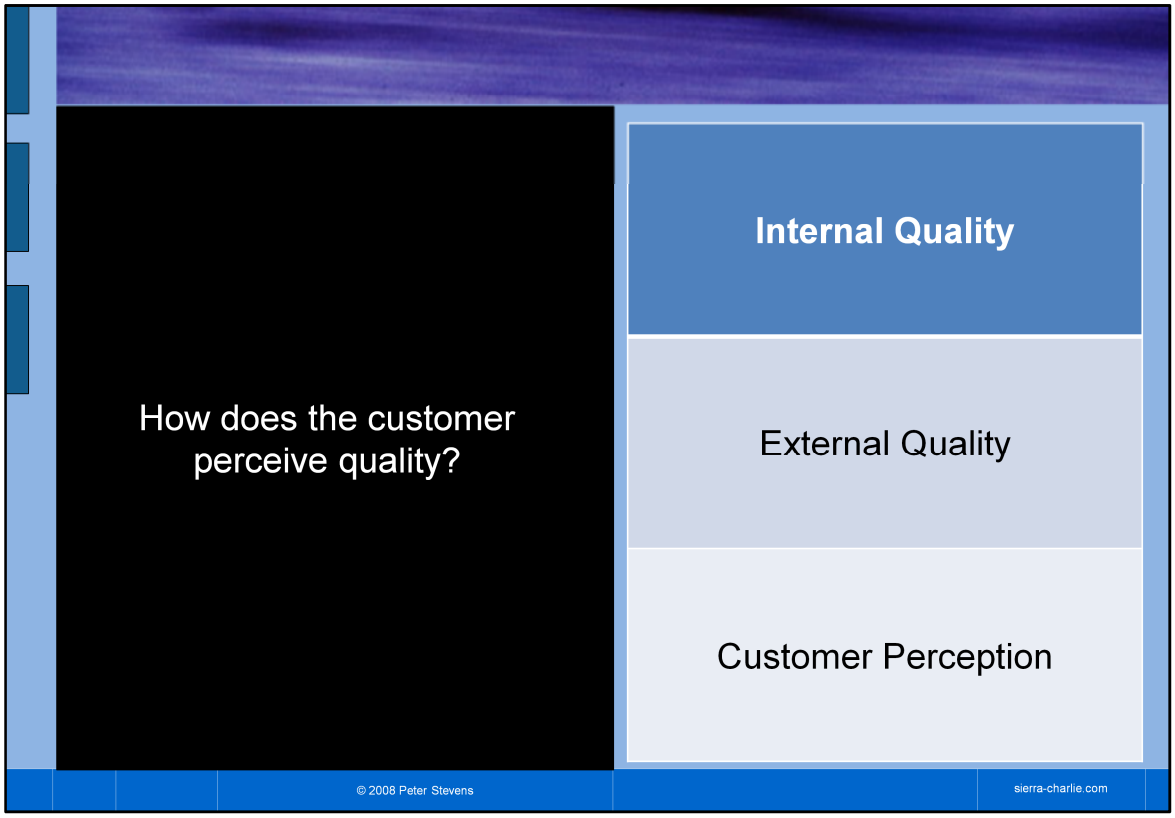


Building a software systems is about fitness for use. Meeting the needs of the users. To build the right product, you need to understand the users. The product must work reliably. In either case, if you build the wrong product or even if you build the right product badly, you will have to fix it, resulting in expensive delays toward the end of the project.



I would like to talk about three things:

- How does the customer perceive quality?
- Acceptance Tests - the interface between development and management
- How to increase the chance of doing it right – (or how to prevent the developers from getting lost en route to the solution).?

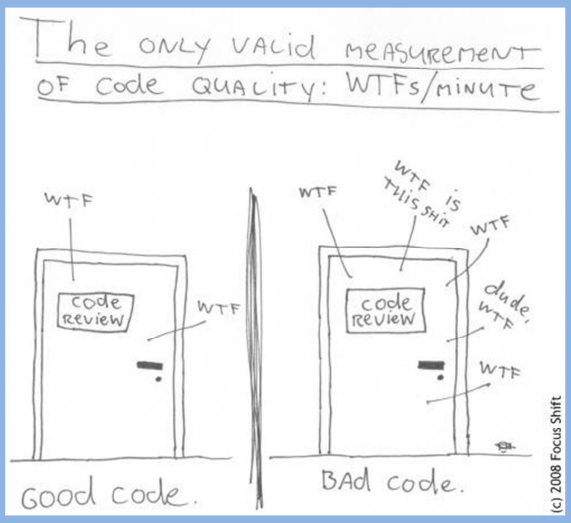


If I buy a car, I could order it with airconditioning, an automatic transmission and cruise control. When I get the car, I check the invoice, confirm that these items are on the bill. I can get in the car, drive it around the block, and check that the components are present, that the a/c blows cold air. These are acceptance tests. The confirm that the product satisfies customer intent.

Was the a/c built to the specifications laid out by ist designers? Will the unit continue to function till the end of the warranty period? There is no obvious way to verify this as the customer, except possibly by reputation of the supplier.

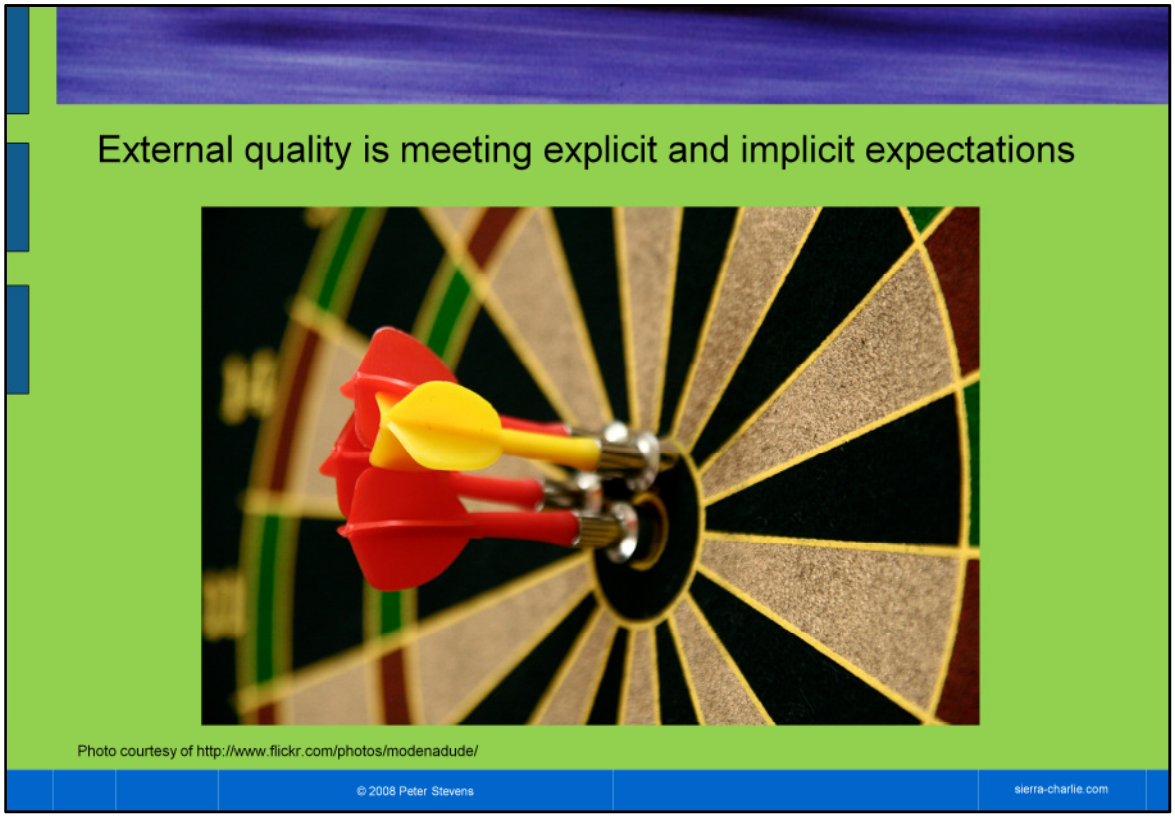
The first purpose of creating a product is ensuring fitness for use. This all about customer perception, and so conceiving the product so that the customer recognizes himself in the product is of paramount importance.

Internal quality is quality of design and construction and is often invisible to the customer.

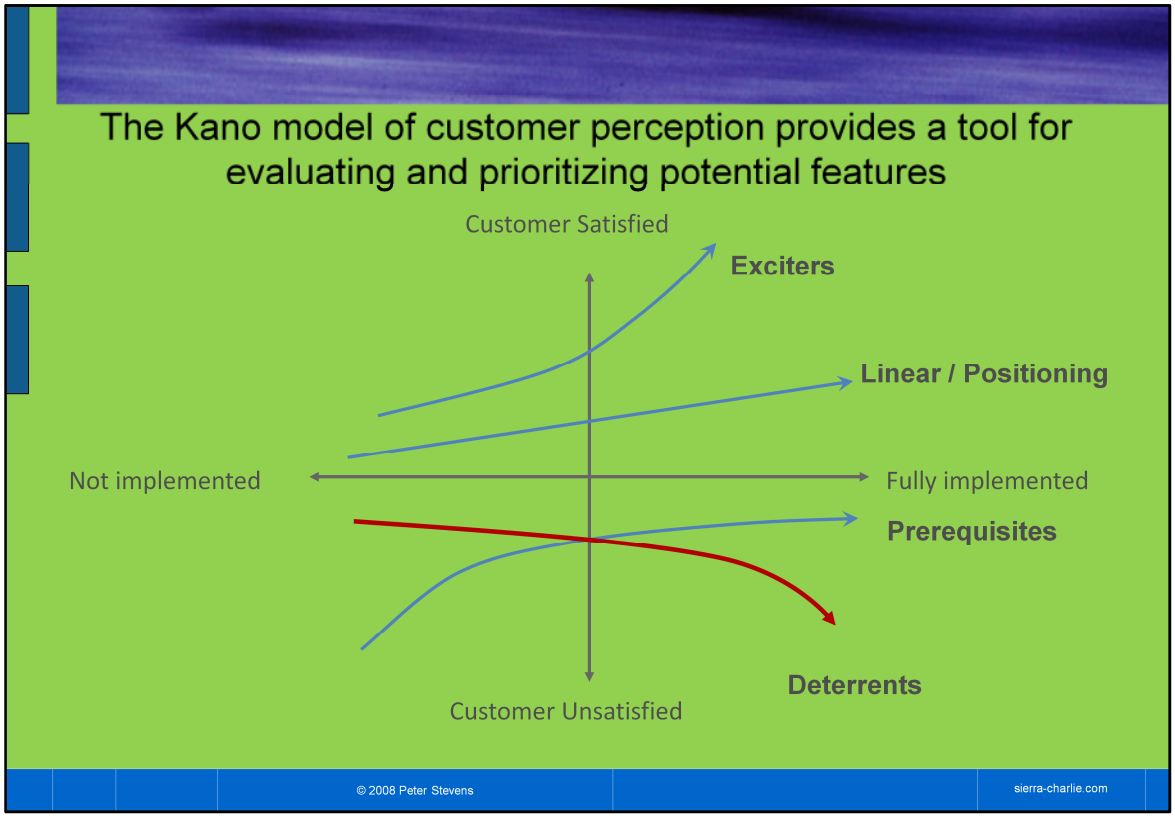


Cartoon courtesy of mstorz@flickr

- Internal Quality is about building it right
- Unit tests by programmers and for programmers confirm that the product's components function properly
- Other tests (smoke tests, security tests, load tests and stress tests) confirm other non functional requirements



- External Quality is about building the right thing
- You can build a well tested, well designed product which is not wanted or not satisfactory for its users.
- How do you ensure that it will be satisfactory to the users? The Kano model of customer satisfaction provides guidance.



According to Kano, there are [three classes of features]:

Prerequisites. If your product doesn't have these features, it will immediately be removed from consideration. Who wants a house without a bathroom?

Linear Features. The more the better, but their higher the cost. A family that wants three bedrooms will not consider a house with one, but they might consider a house with 2 or 4 bedrooms.

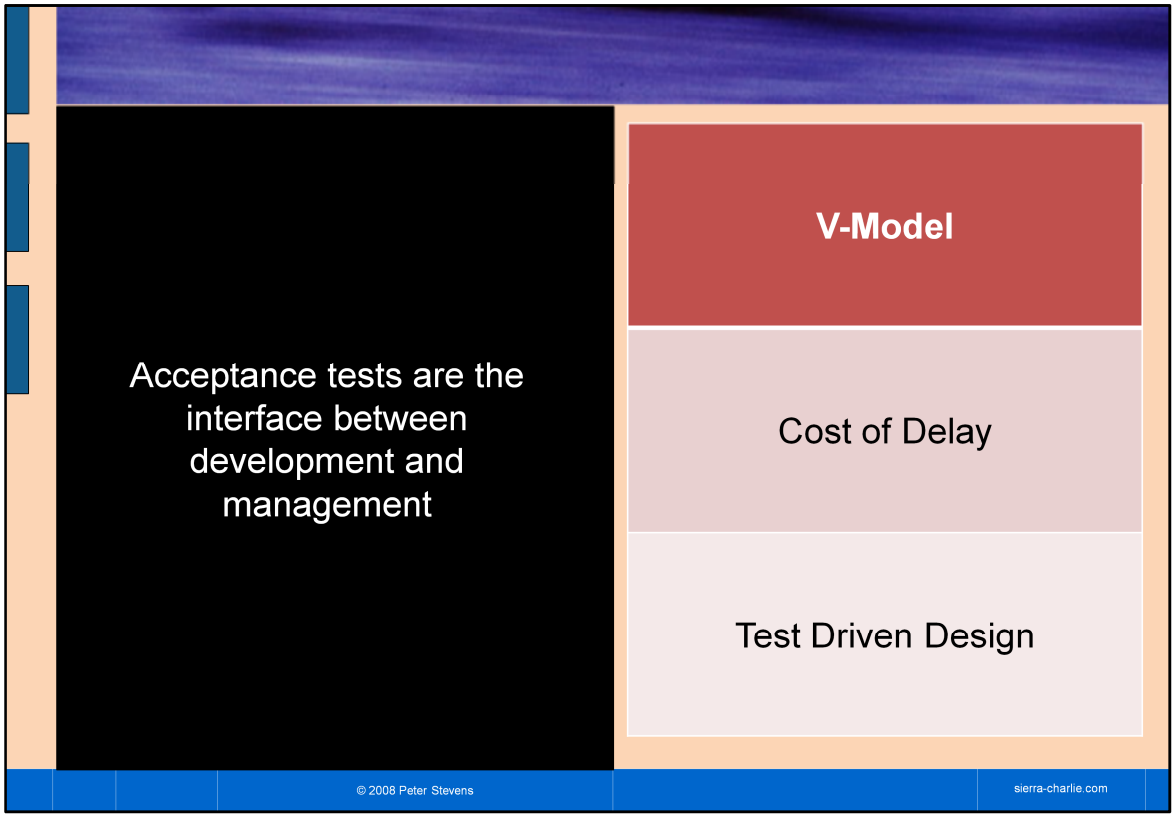
Exciters. Unique features which only this product has and which convince the customer to say "Yes! I want it! This one and no other!"

I like to talk about a fourth class, which is not explicitly charted on most Kano diagrams: Deterrents.

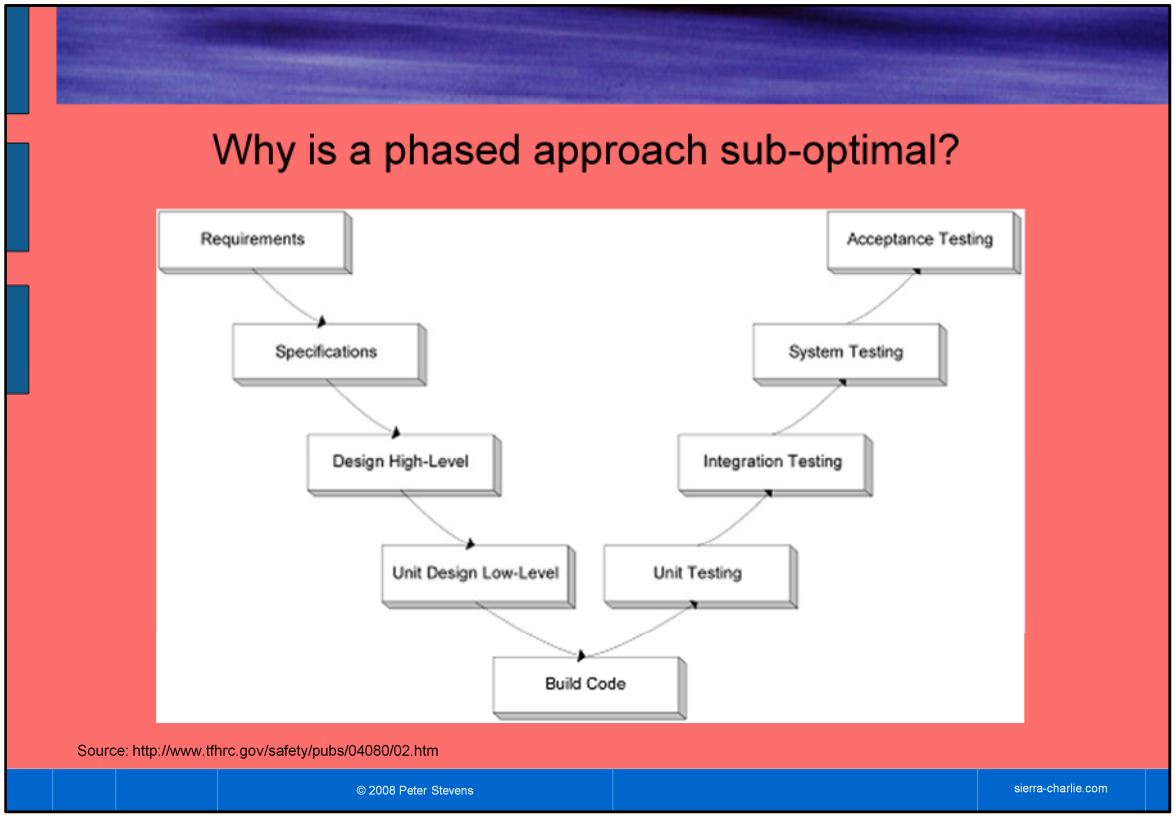
Deterrents are features which actually turn off your customers. For a house, proximity to shopping can be an exciter for some and deterrent for others (due to associated noise and traffic). Trying to scale often causes deterrents. The flexibility required for large companies creates a need to configure which is a pain for SMEs or individuals.

So as you create stories and as you implement the product, you need to ensure you've got enough of the prerequisites to keep the customers from jumping off and enough exciters so that the customers buy the product.

Agile, particularly the work of Mike Cohn, provides guidance for deriving the needed functionality based on understanding user needs. Identify roles, then their goals, then identify the features to support these roles.



Acceptance testing should confirm that functional requirements have been met.



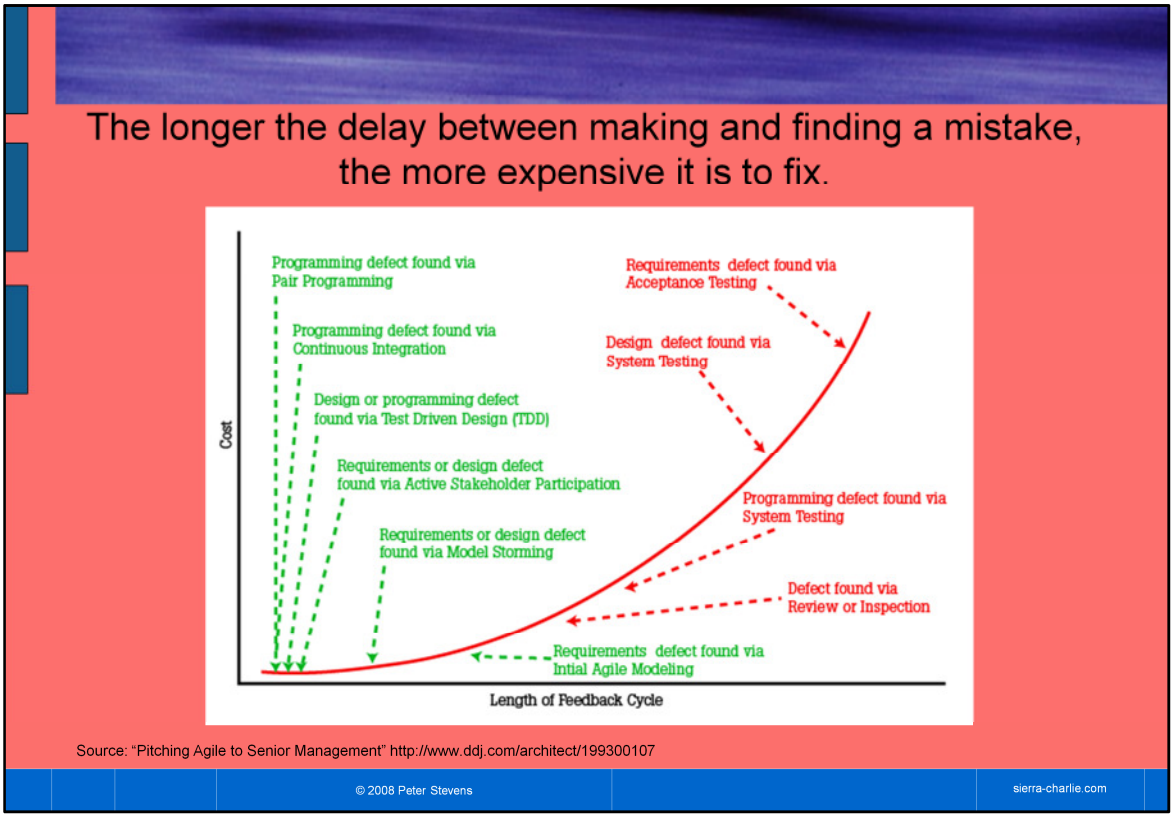
Does anyone here use the V-Modell?

For each implementation level, there is a corresponding test level.

When do the testers write/write down tests?

Do the testers work work with or even talk to the developers? “We don‘t want the developers guesses about the specs to influence the testers guesses”

What happens when **acceptance** tests are written at the end of the development process?



Learning is what happens when we make a mistake and then correct it.

Learning should happen as quickly as possible.

The cost of an error increases exponentially with the delay before the error is recoginized.

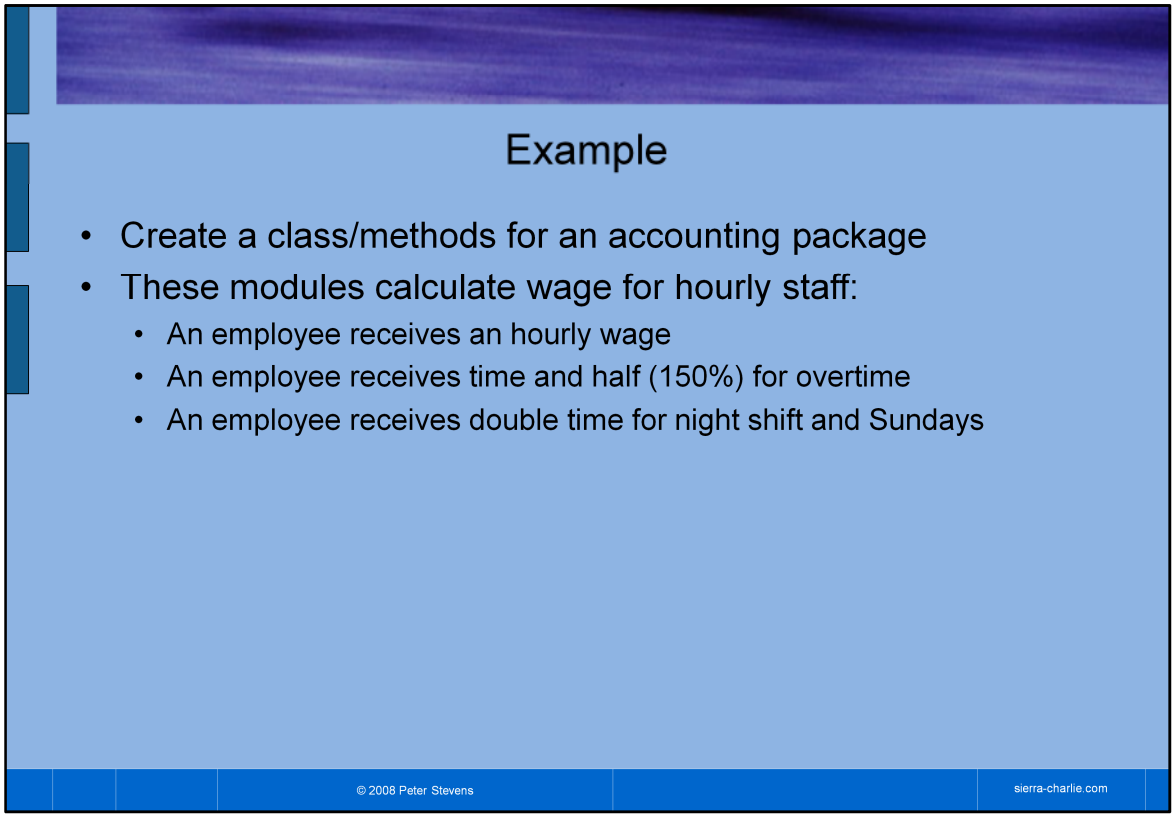
The ultimate is case is discovering that a prerequisite function is missing after having deployed the application. The costs to repair are the highest and the opportunity costs are a multiple of the direct losses.

Agile processes seek to minimize the delays in learning about the system in development.



How to increase the chances of doing it right.

Test Driven Design helps the developers understand the requirements before they start to write code. It is not the same as Test Driven Development.

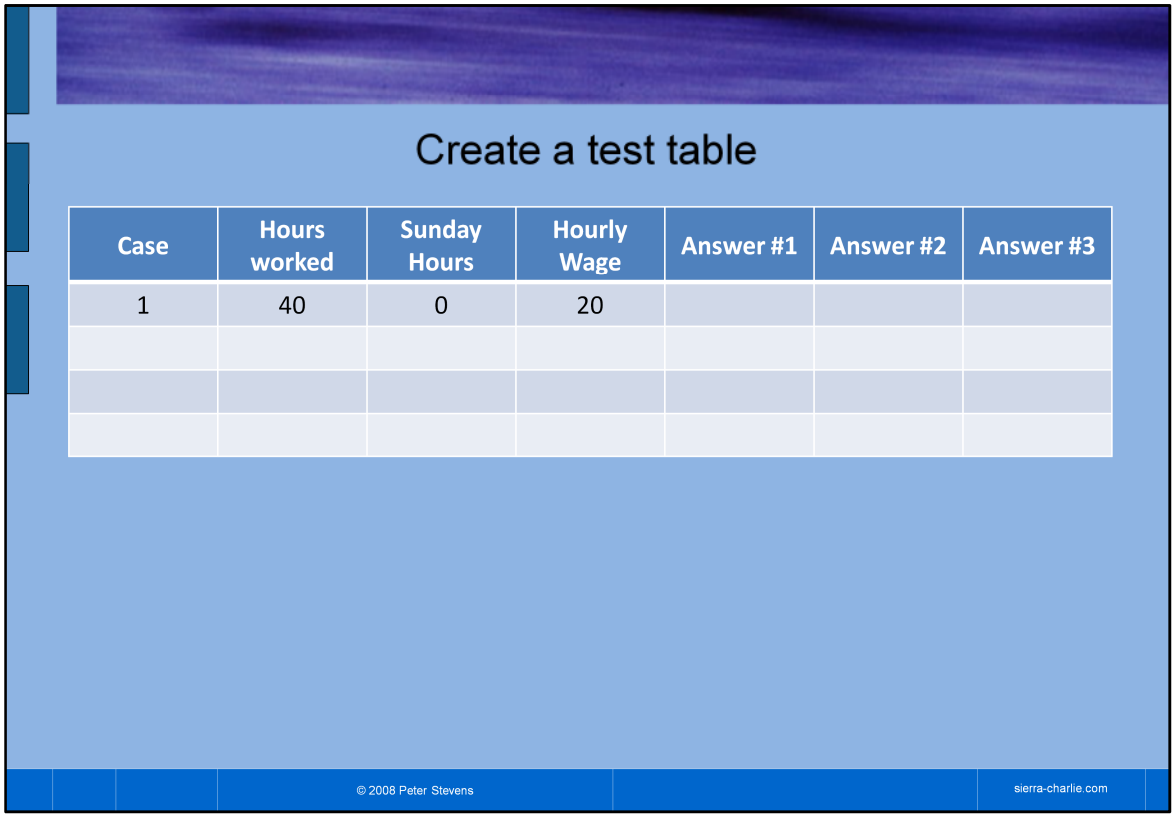


If this were a class, I would ask 8 people to leave the room. The I would invite one to come in. I would give him or her this specification – verbally. The he would pass it on to the person, etc. until…

Until what? What do you think happens as the specs get passed on?

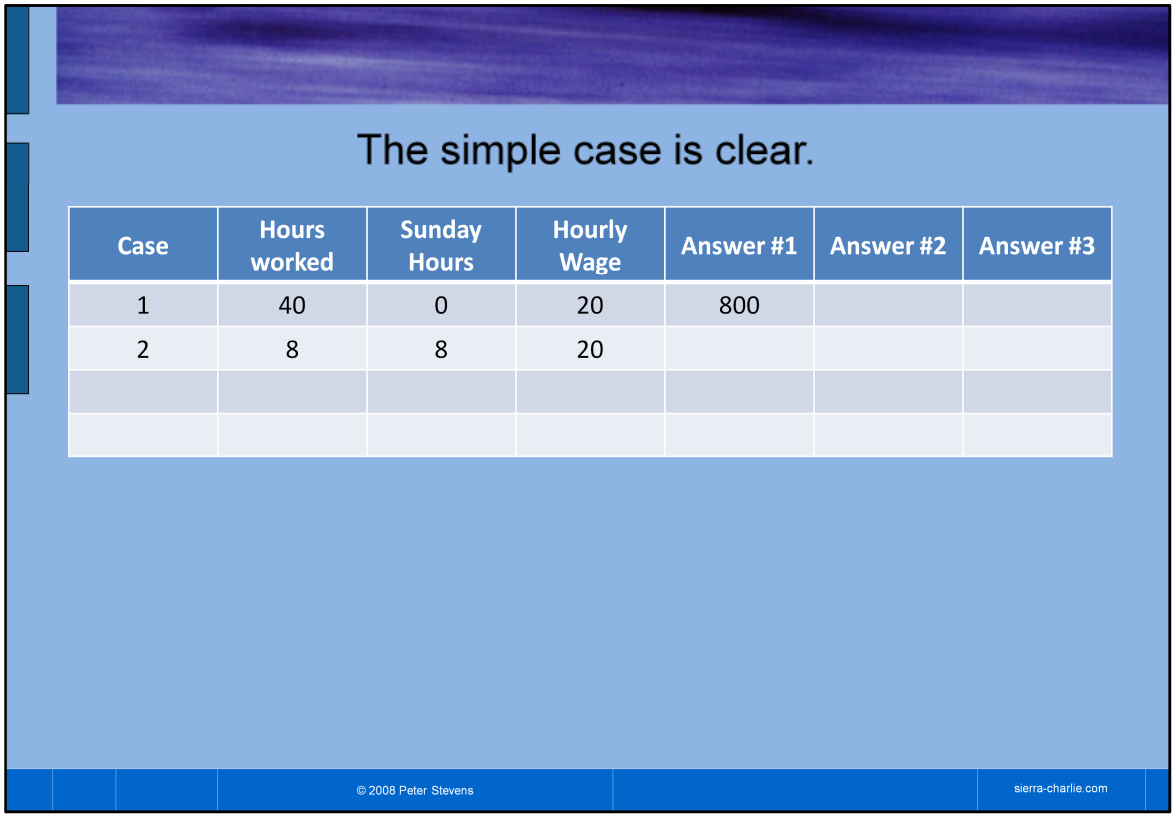
After 4 or 5 hops, the spec is pretty unrecognizable.

But you say wait, we need the spec in writing. OK, here it is in writing. Everyone understand?



1: 40 x 20 = 800

Anybody get a different answer?



2a: 8 * 40 = 320
2b: 8 *20 + 8 * 40 = 480

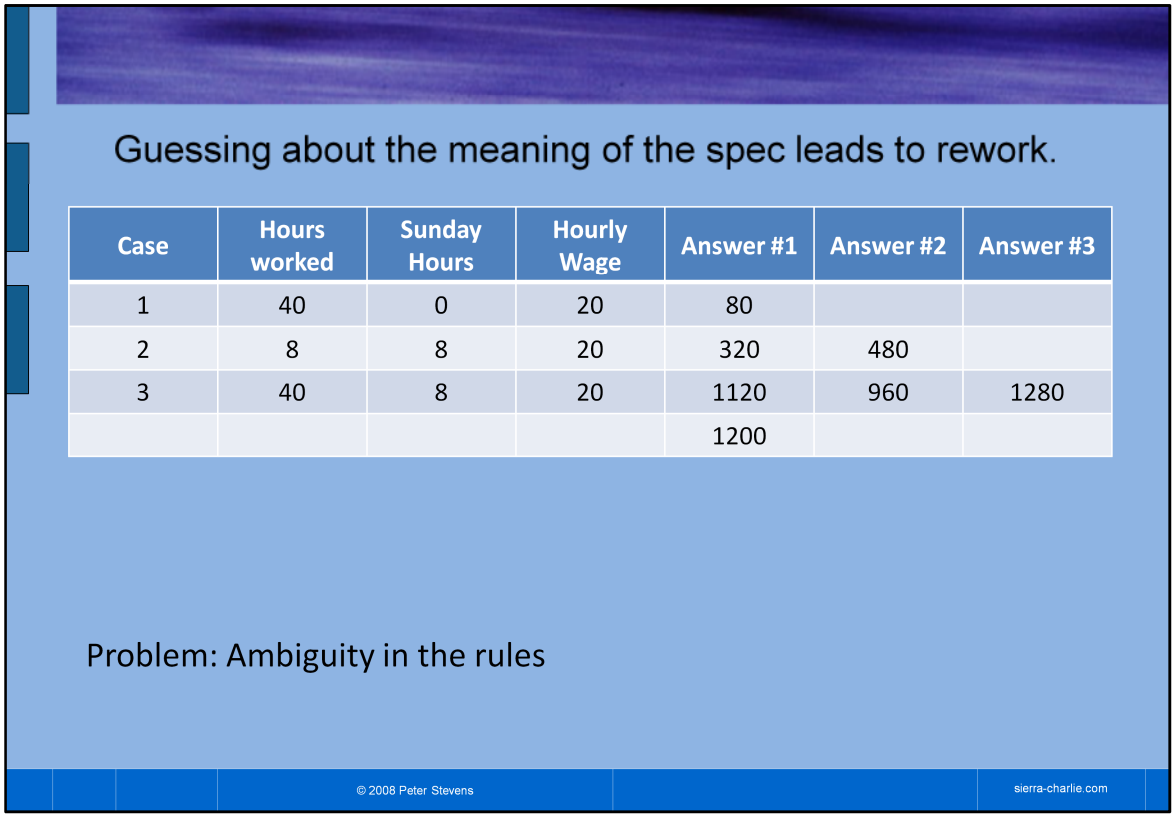
Ambiguity in the interface is a source of error

Case	Hours worked	Sunday Hours	Hourly Wage	Answer #1	Answer #2	Answer #3
1	40	0	20	80		
2	8	8	20	320	480	
3	40	8	20			

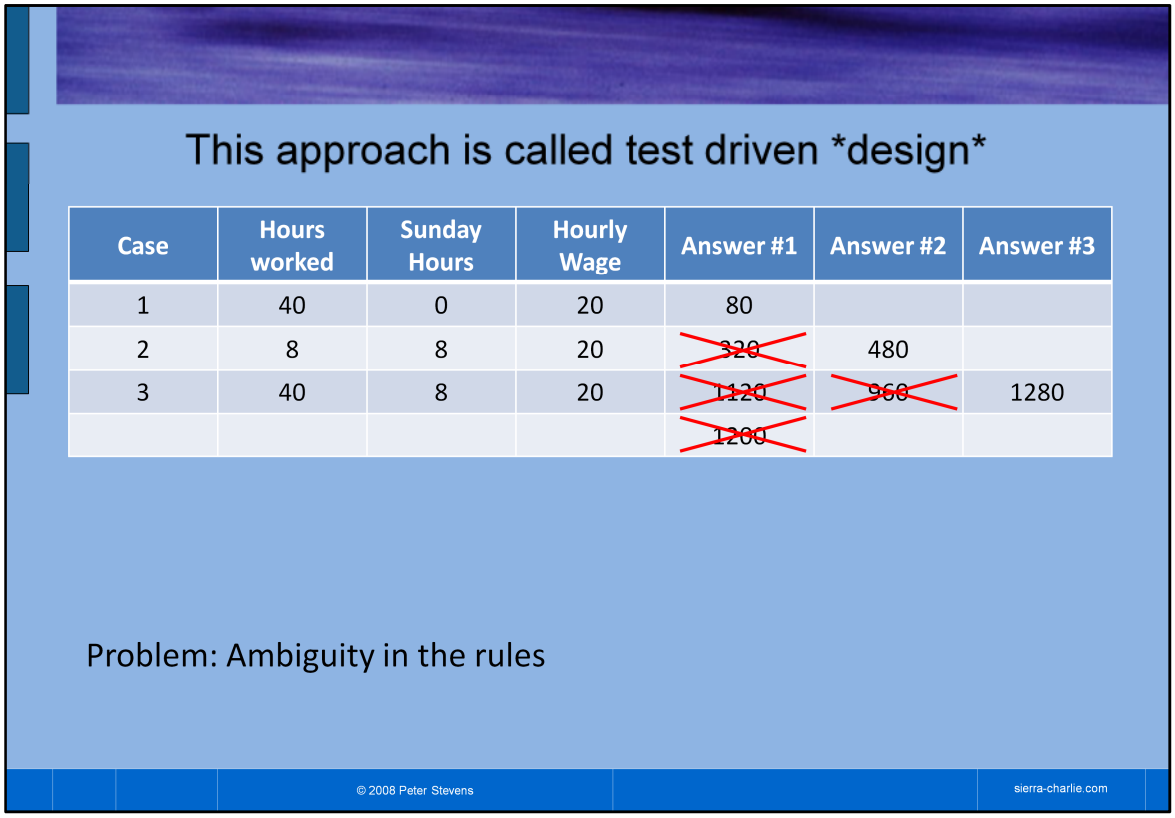
Problem: Ambiguity in the meaning of the parameters

© 2008 Peter Stevens

3a: 40 * 20 + 8 * 40 = 1120 (Sunday rule overrides overtime rule)
3b: 32 * 20 + 8 * 40 = 960 (40 hours is total hours worked, so no overtime)
3c: 40 * 20 + 8 * 60 = 1280 (Sunday is overtime and Sunday time)
3d: 8 * 40 + 32 * 20 + 8 * 30 = 1200 (Sunday is first day of week, so not overtime)

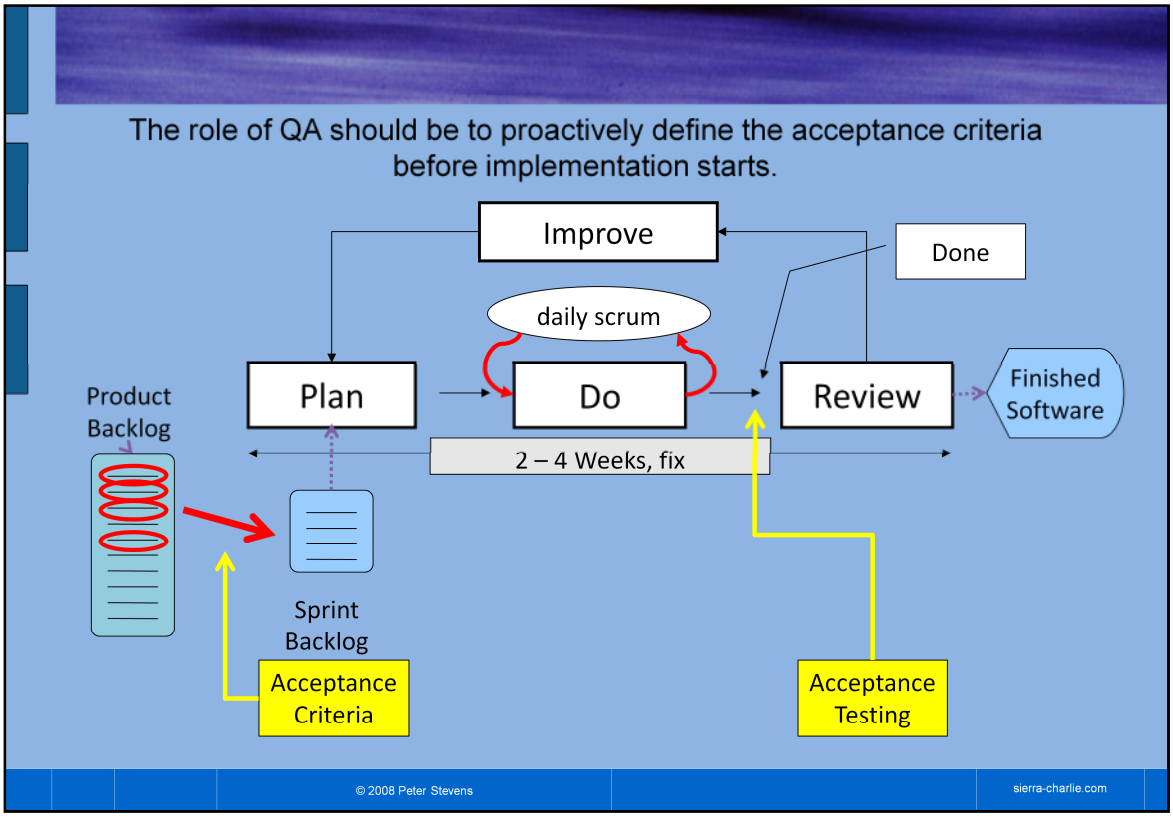


3a: 40 * 20 + 8 * 40 = 1120 (Sunday rule overrides overtime rule)
3b: 32 * 20 + 8 * 40 = 960 (40 hours is total hours worked, so no overtime)
3c: 40 * 20 + 8 * 60 = 1280 (Sunday is overtime and Sunday time)
3d: 8 * 40 + 32 * 20 + 8 * 30 = 1200 (Sunday is first day of week, so not overtime)



Test Driven Design – write tests to specify the product and focus the developers on writing the right thing.

BTW - This result is obvious to anybody who had worked for an hourly wage in North America. What does that say about the problems of offshoring?



Testers are part of the implementation team.

QA prepares the acceptance criteria for each story before it is accepted into a sprint.

QA is transformed from detecting bugs to preventing bugs.

Testing still happens at the end. But it should be a confirmation and one does not expect to find (many) bugs.

Basic Process of Test Driven Design

- Understand your users and what motivates them
- Create & Prioritize User Stories
- Per Sprint:
  - Create Acceptance Test Cases
  - Accept the acceptance tests
  - Write the code
  - Verify that the code passes the acceptance tests
  - Submit to customer for acceptance

© 2008 Peter Stevens

Questions



Photo © leonardo viti - fotolia

© 2008 Peter Stevens

Further Reading

- Fit for Developing Software
  - Rick Megridge & Ward Cunningham

- FitNesse
  - enables customers, testers, and programmers to learn what their software should do, and to automatically compare that to what it actually does do. It compares customers' expectations to actual results
  - http://fitnesse.org/

© 2008 Peter Stevens



Scrum Training:

http://tinyurl.com/Scrum-Kurse

Blog:

http://www.scrum-breakfast.com