

# eliminate waste

in order to

add value to the customer

and

flexibly anticipate  
customer demands

&

# respect for people



J-Fall

12 november 2008 Spant!



## Lean Software Development

# (Agile) Software Development in Perspective

Rob Westgeest - [rob@qwan.it](mailto:rob@qwan.it)

Marc Evers - [marc@qwan.it](mailto:marc@qwan.it)

**QWAN**

*Quality Without A Name*



## Contents

- About us
- Case study
- Seven principles
- History
- Lean vs Scrum, XP, RUP, ...
- Limits



## About us

Independent  
(Agile) software development coaches  
Trainers  
Facilitators  
Consultants  
Developers



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



[www.agileopen.net](http://www.agileopen.net)

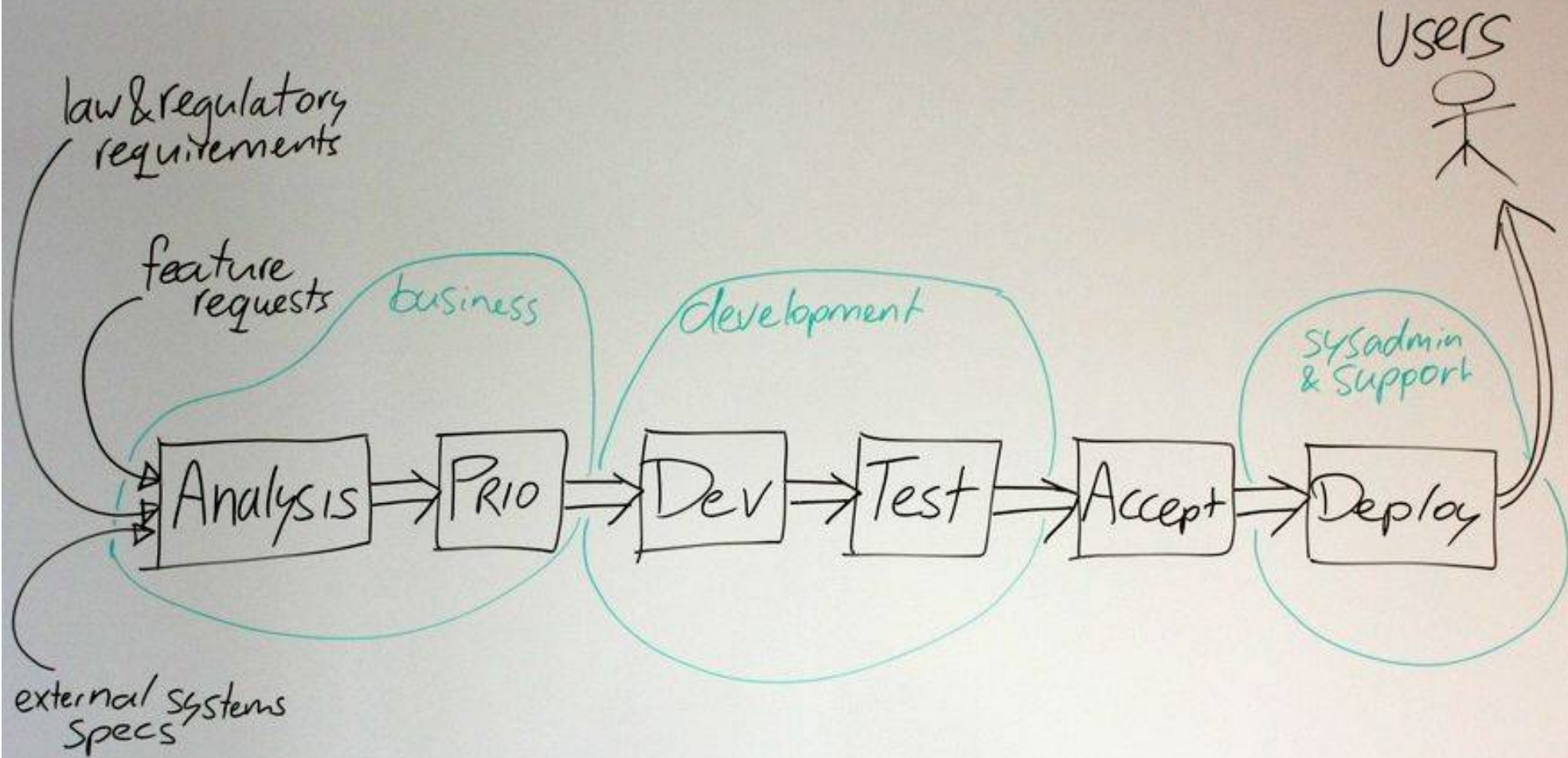
# QWAN

Quality Without A Name

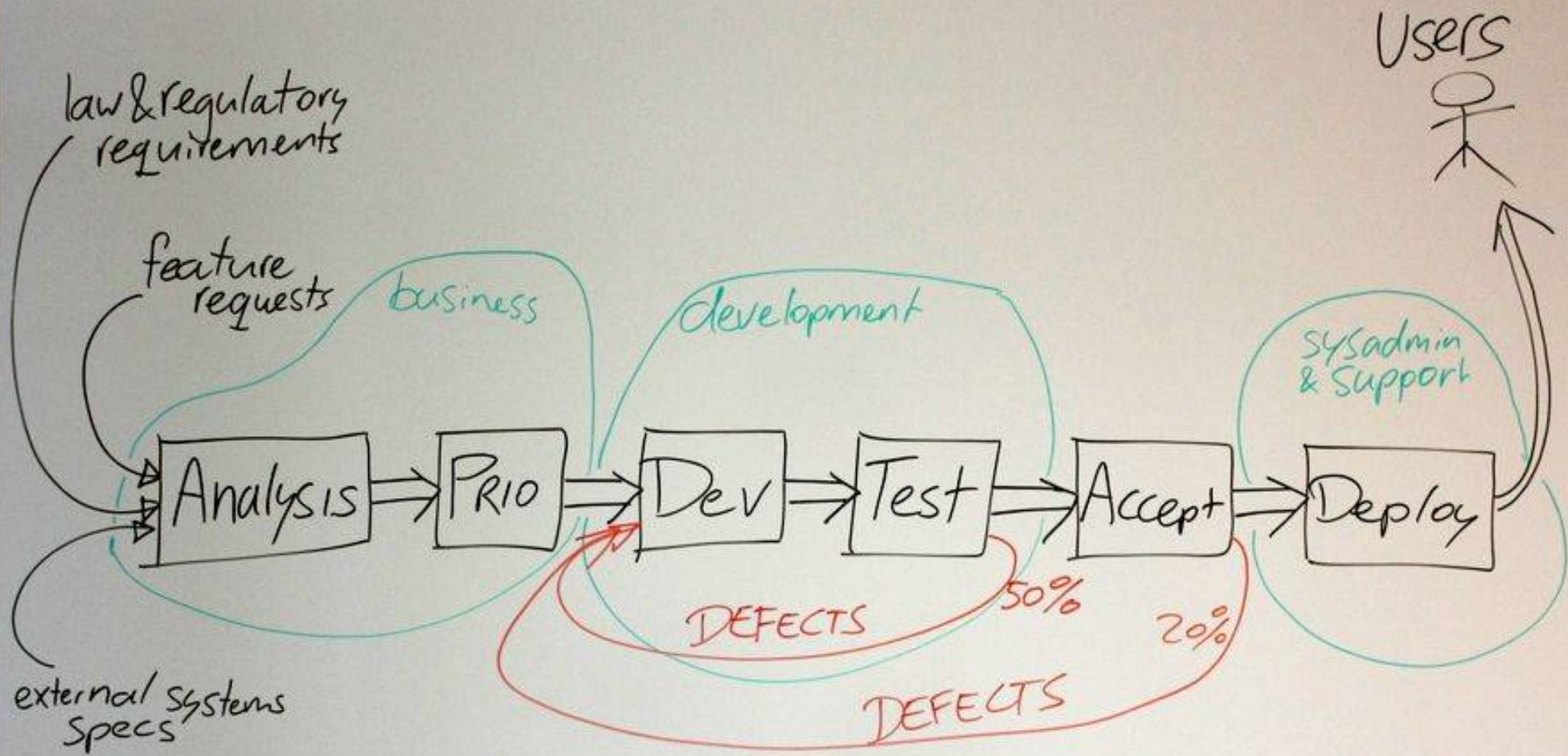
[www.qwan.it](http://www.qwan.it)

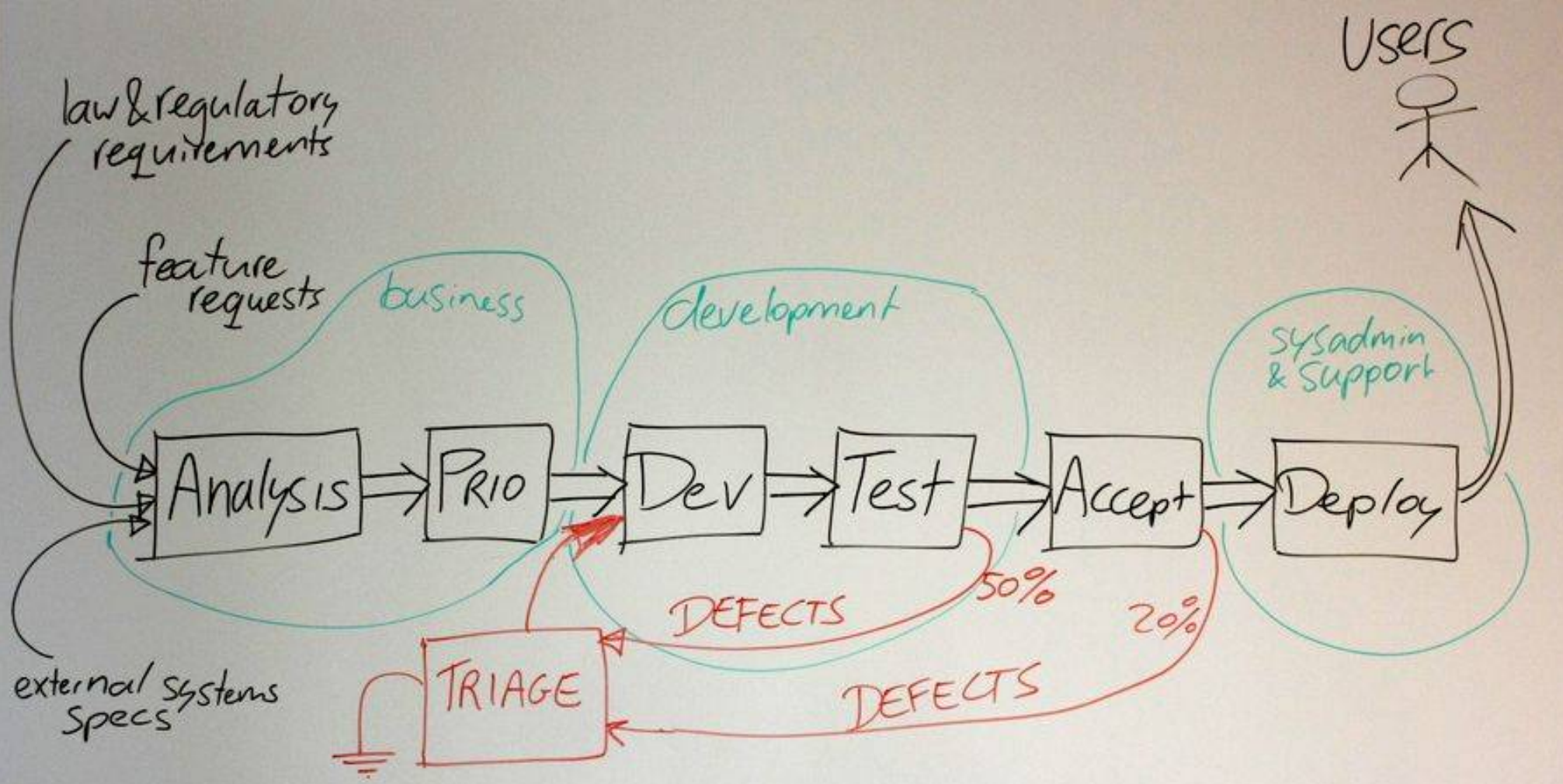
**XP DAYS**  
**2008**  
BENELUX

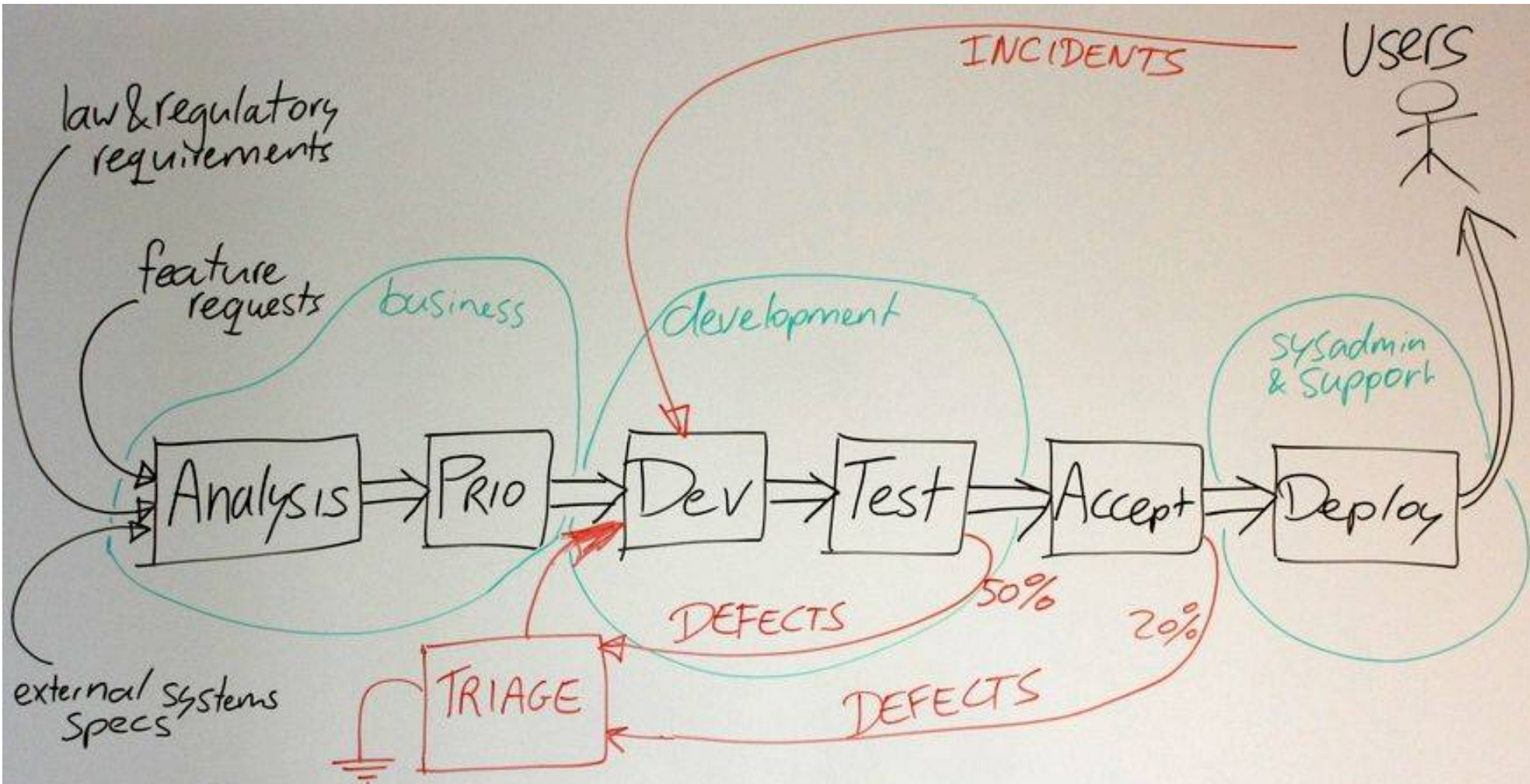
[www.xpday.net](http://www.xpday.net)

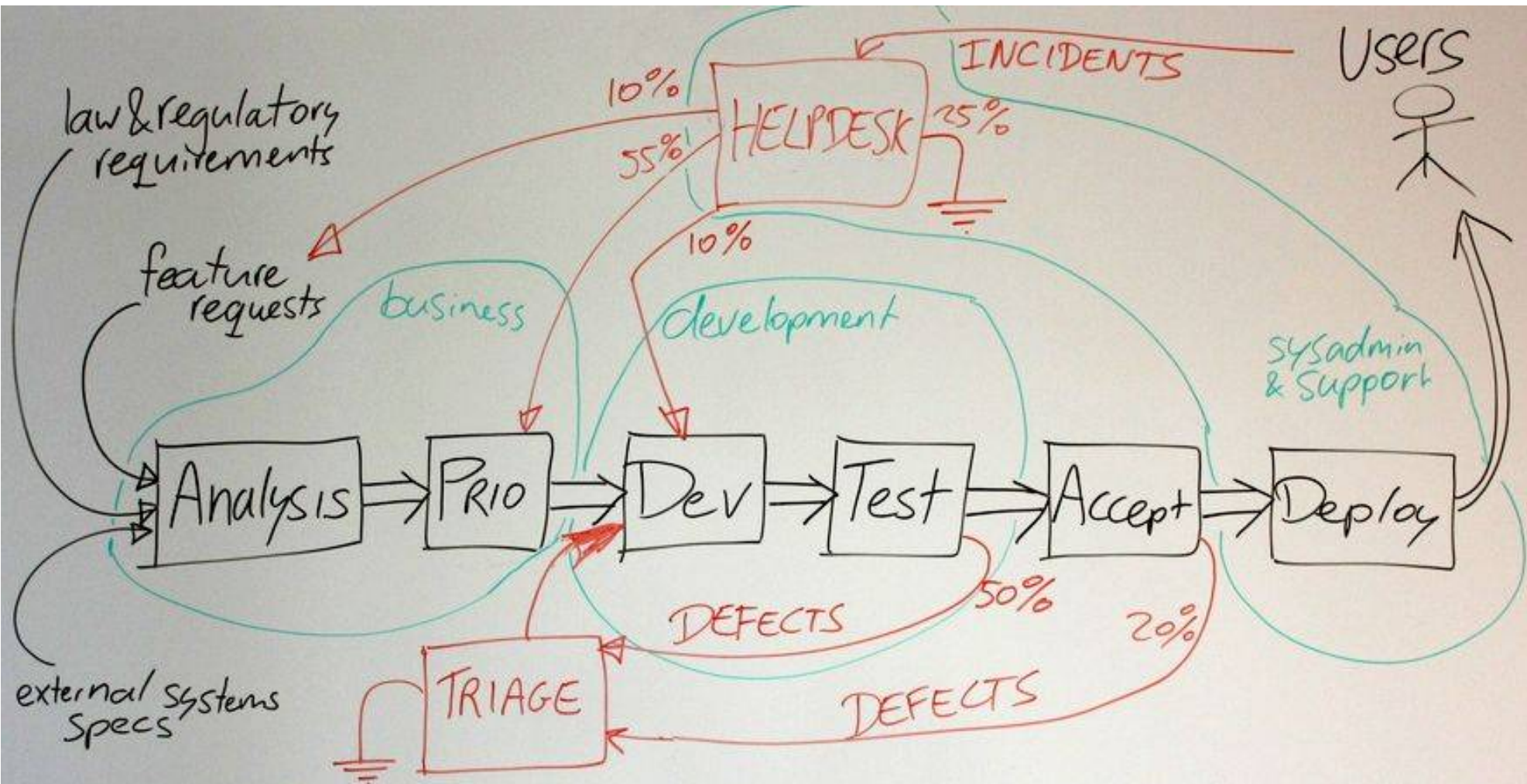













A Lean Perspective...

# 7 Principles

- 1 Eliminate waste
- 2 Create knowledge
- 3 Defer commitment
- 4 Deliver fast
- 5 Respect people
- 6 Build quality in
- 7 Optimize the whole

# 1 Eliminate waste

A yellow bulldozer is shown in a landfill, surrounded by large piles of waste. The bulldozer is positioned in the center-right of the frame, facing left. The waste consists of various materials, including plastic bags, metal scraps, and other debris. In the background, there are green hills under a clear sky. The text "Strive to remove anything from your process that does not add value to the customer" is overlaid on the bottom half of the image.

Strive to remove anything from your process that does not add value to the customer

Anything that does

*not*

add value to the customer

Waste

Anything that

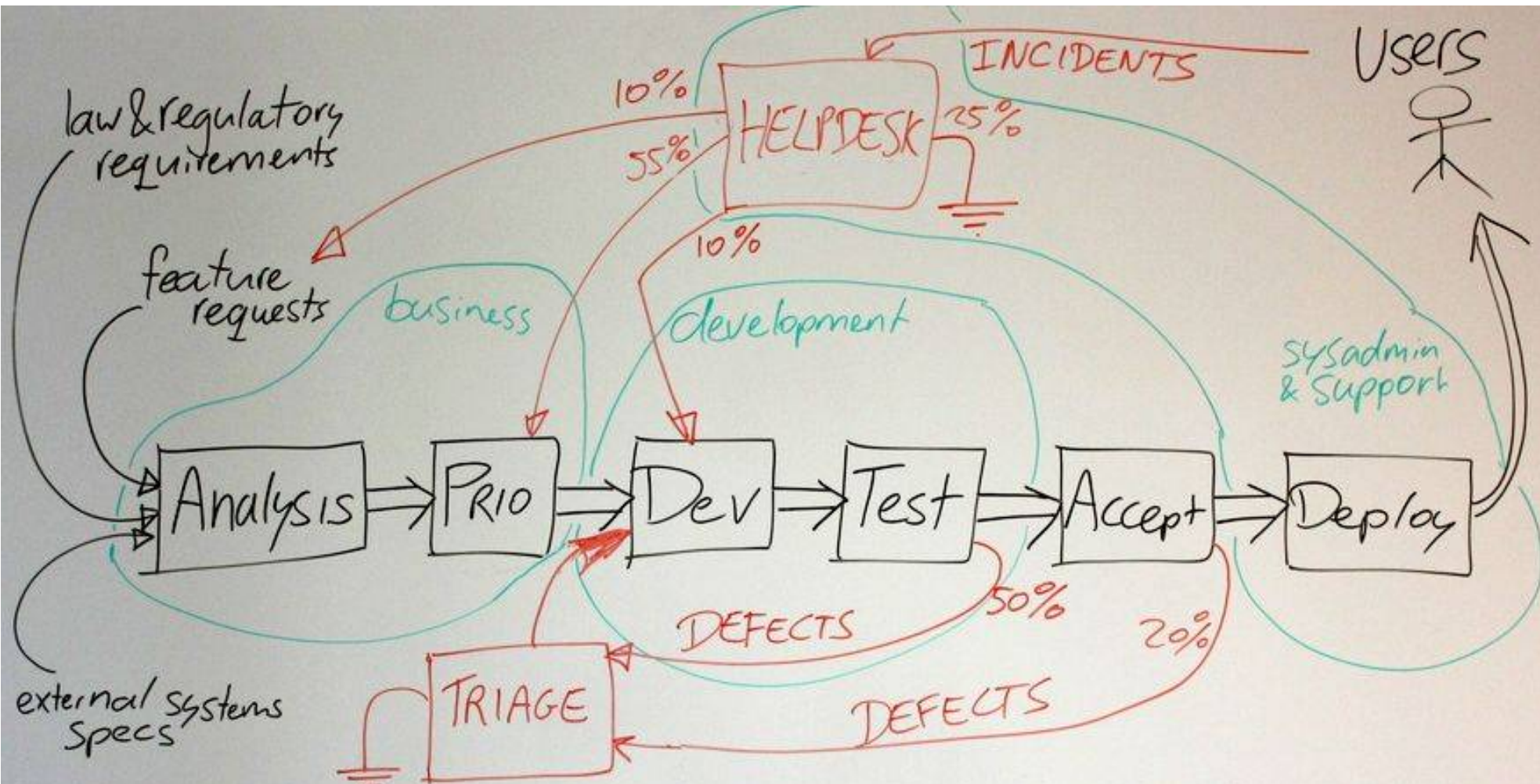
*delays*

value to the customer

# Seven Wastes



- Defects
- Extra features
- Handoffs - crossing organizational boundaries
- Waiting & delays
- Partially done work
- Task switching
- Relearning - reinventing wheels



value	4 h	1 h	2 d	1 d	1 h	1 h	v: 4 d		
waste	16 h	2 w	7 h	2 w	2 w	1 w	2 w	7 h	w: 61 d
CYCLE TIME: 65 d									
EFFICIENCY: 6%									

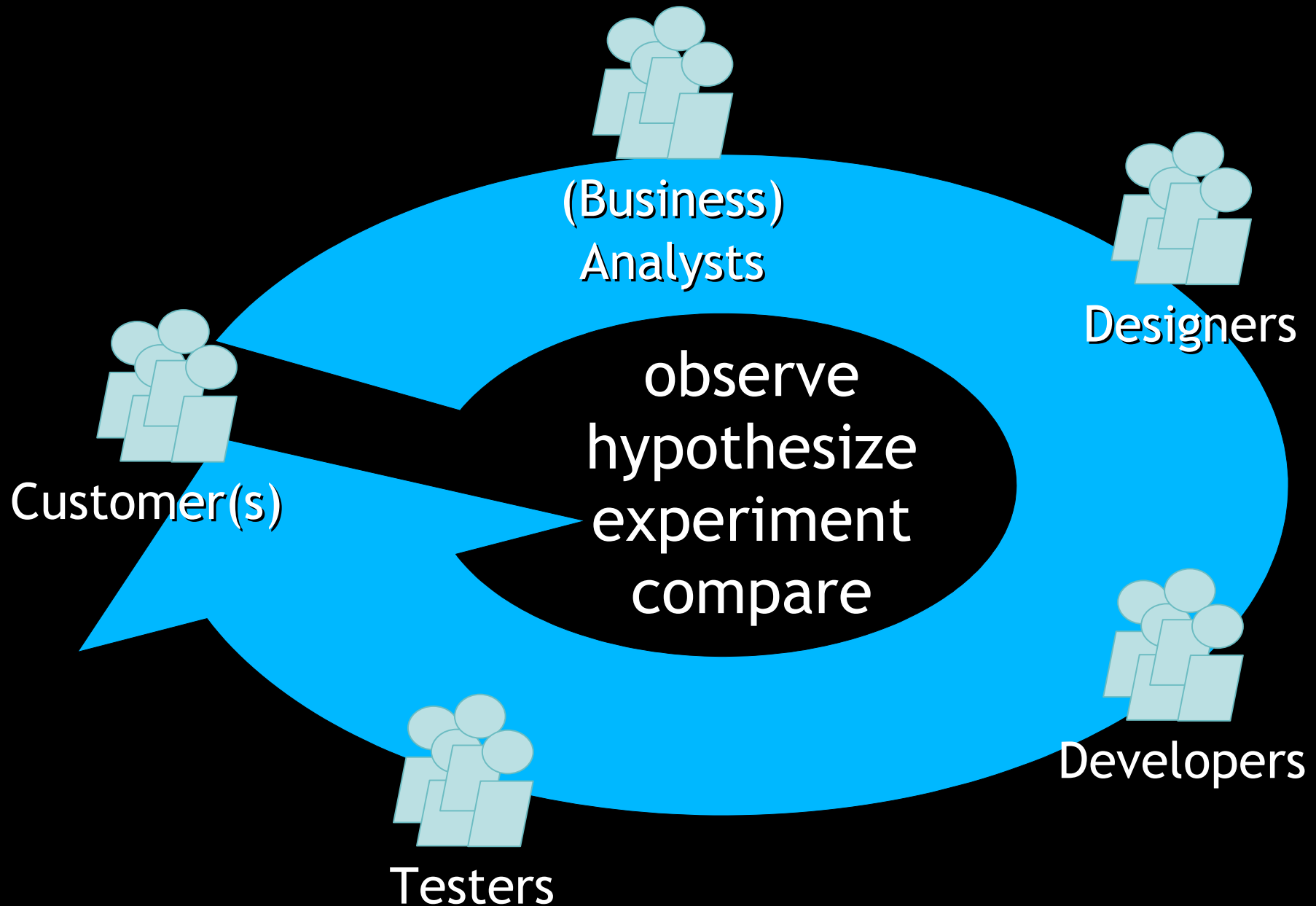
## 2 Create knowledge



If learning is what  
we do for a living  
then amplify that

collaboration and invention = learning

# Programming as Theory Building \*)



\*) P. Naur - reprinted in *Agile Software Development* (Cockburn)

# Grow project methodology

A methodology is a base set of conventions, supported by everyone



# 3 Defer commitment

A close-up photograph of a gold-colored quartz clock face. The clock has Roman numerals for the hours and the word "QUARTZ" printed on the dial. The hands are black and the clock is set against a blurred background.

Make decisions as late as possible - right before the decision is made for you

# Delay decisions

- Decisions are less important than we believe
- Tracer bullets
- Example techniques:
  - Simple design - *You Aren't Gonna Need It*
  - Short iterations
  - Set based development

Until...



# The Last Responsible Moment

The moment you have 'perfect knowledge'  
Right before options disappear

# Tool: Set based development



- Ill defined problems
- Decide slowly by consensus, implement rapidly
- Several solutions for the same problem

create three graphics designs  
cross-platform development  
spike solution

# Tool: Set based development

- Ill defined problems
- Decide slowly by consensus, implement rapidly
- Several solutions for the same problem

## Expensive ?

create three graphics designs  
cross-platform development  
spike solution

# 4 Deliver fast



Build a culture around delivering value  
to the customer  
as fast as possible

# Concurrent development

Start with implementation and design simultaneously



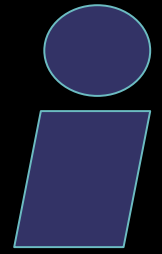
Acceptance test and story simultaneously

UI design and implementation simultaneously

On site customer

# Pull Systems

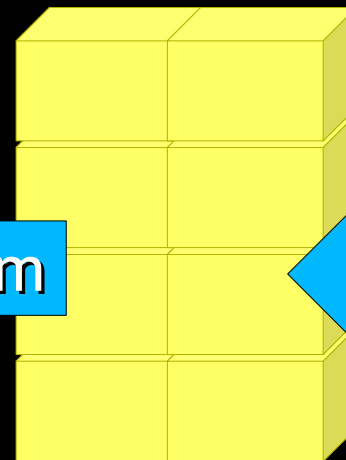
- Kanban
- Supermarket as inspiration



Customer



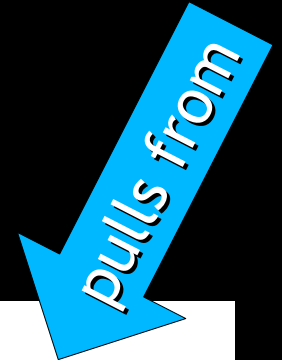
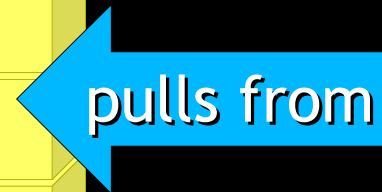
Manufacturer



Supermarket  
Inventory

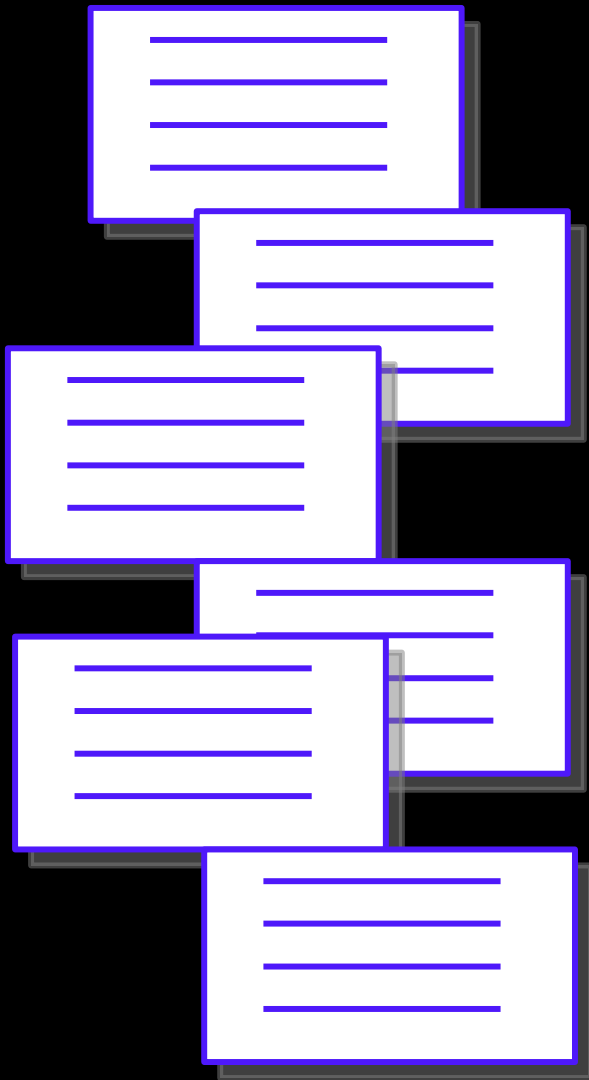


Supermarket  
Shelf

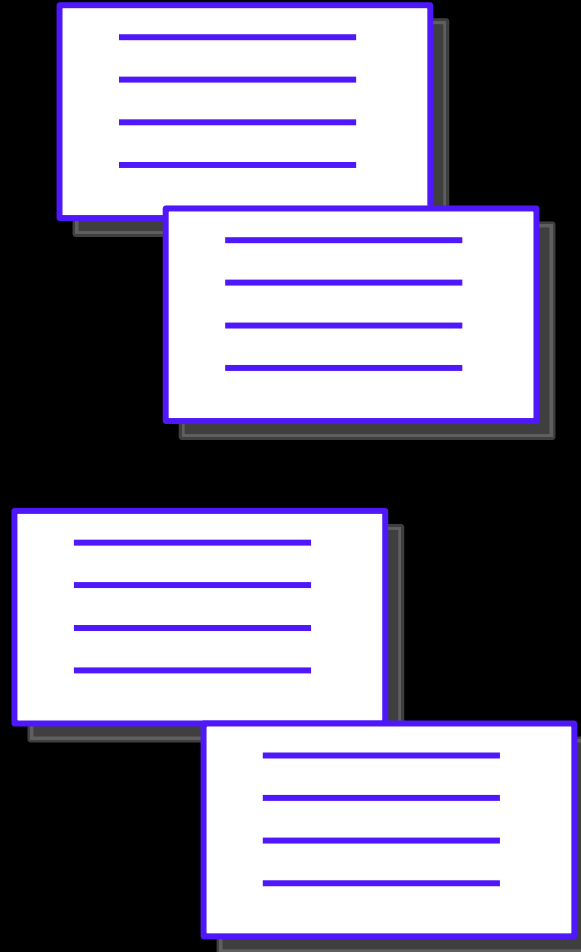


# Software Pull Sample

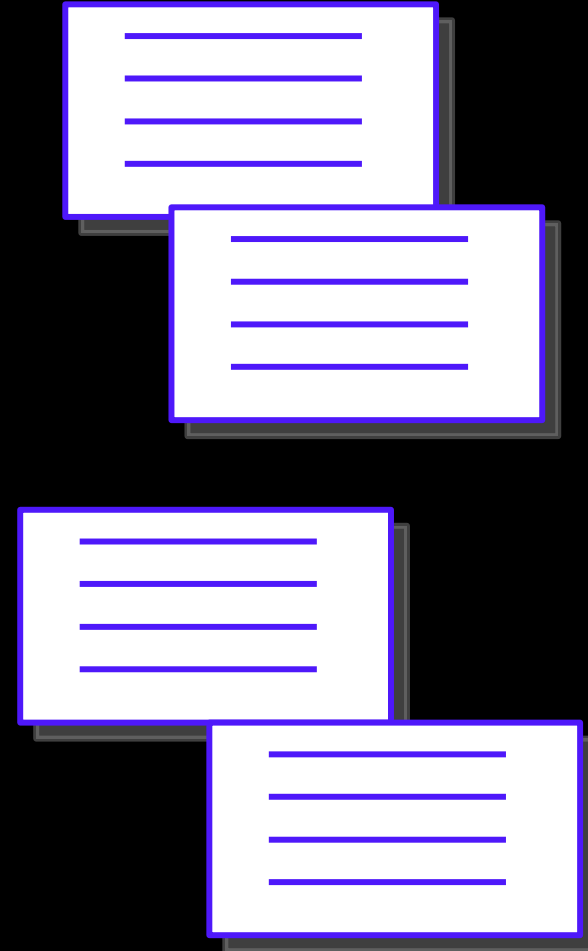
Todo



Developing



Tested





## 6 Build quality in

Build and maintain internal and external integrity in the product you are building



Tool: Jidoka

Stop the production line

# Tool: Jidoka



Fix problems

Ask for help

Find root causes

continuous integration

test driven development

spontaneous stand-up meetings





# Tool: Refactoring

Evolve and maintain integrity in design

Design quality in

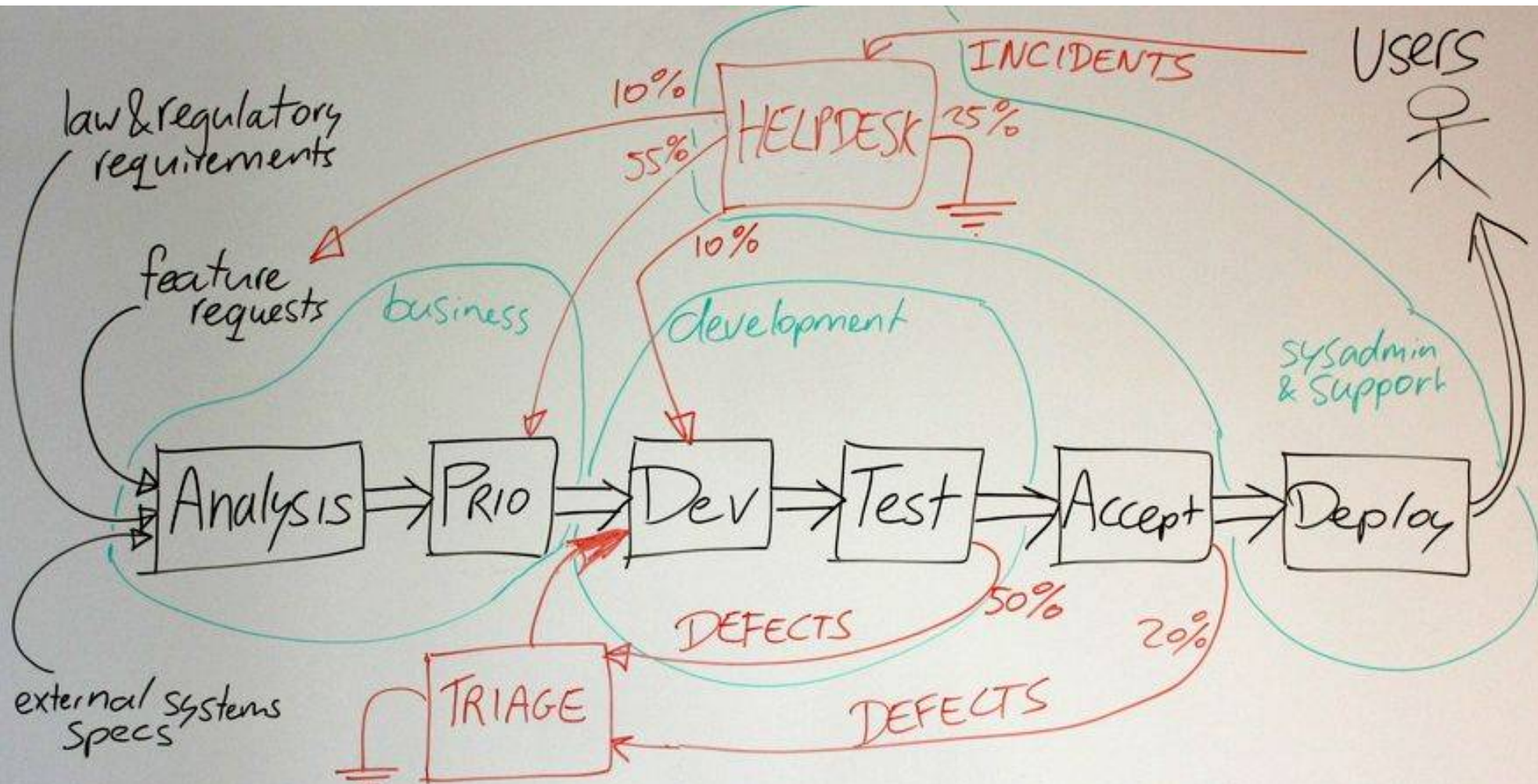
Part of Test Driven Development Practice

# 7 Optimize the whole

Let all stakeholders see the system as a whole



Prevent local optimization



value	4 h	1 h	2 d	1 d	1 h	1 h	v: 4 d		
waste	16 h	2 w	7 h	2 w	2 w	1 w	2 w	7 h	w: 61 d
CYCLE TIME: 65 d									
EFFICIENCY: 6%									

# 5 Respect people

A photograph of four people in a meeting room. Three men and one woman are gathered around a whiteboard. The whiteboard is covered with colorful sticky notes and diagrams. The room has a drop ceiling with recessed lights. The people are dressed in casual business attire.

Team decides on way of working

Developers sign up for tasks

Facilitating & coaching

- Set the context for self organization
- Simple, generative rules

Respect your partners

# Grow project methodology

A methodology is a base set of conventions, supported by everyone



# Hansei - Kaizen

Relentless reflection

&

Continuous improvement

改善

# History

Toyoda Automatic Loom Works (late 1800)

Toyota Motor Corporation (1930)

W. Edwards Deming teaches Japanese companies (1950-60)

Japanese cars are special (1980)

Toyota cars even more (1990)

*The New New Product Development Game* (1986)

inspiration for Scrum

Lean manufacturing (1990's)

Lean software development (2003+)

Kanban approach (2007)

# Lean vs Scrum, XP, RUP, ...

*Defer commitment*  
iterations

*Deliver fast*  
increments

*Pull*  
scrum boards, feature based delivery

*Create knowledge*  
retrospectives, iterations, quick experiments

*Eliminate waste*  
simple design

*Build quality in*  
test driven development, refactoring

# Lean is different

Principles vs practices

It is not a methodology,  
it is a set of thinking tools

# Limits and failure modes

## Short sighted view on waste

- Refactoring is waste
- Unit testing is waste

## Value fetish

# Credits

the daily standup © PDAgrl

<http://flickr.com/photos/heathershacienda/194577079/>



Earth, courtesy Apollo 17, and probably the most reproduced image of all time ©

woodleywonderworks

<http://flickr.com/photos/wwworks/2222548359/in/photostream/>



path to knowledge © fictures

<http://flickr.com/photos/fictures/8594544/>



books in a stack (a stack of books) © austinevan

<http://flickr.com/photos/austinevan/1225274637/>



the monster that buries waste © nicolas.boullosa

<http://flickr.com/photos/faircompanies/2201897208/>



e-Waste © ÇP

<http://flickr.com/photos/techbirmingham/345897594/>



Sagrada Spiral © david.nikonvscanon

<http://flickr.com/photos/nikonvscanon/1464203380/>



Pizza in Delhi (c) tracyhunter

<http://flickr.com/photos/tracyhunter/289011309/>



# References

Mary & Tom Poppendieck

- *Lean software development, an Agile toolkit* (2003)
- *Implementing Lean software development* (2007)

Ken Schwaber, *Agile Software Development with Scrum* (2001)

Alistair Cockburn, *Agile Software Development* (2002, 2006)

Fred Brooks, *The Mythical Man Month* (1975, 1995)

David J. Anderson, *Agile Management for Software Engineering* (2003)

Peter M. Senge, *The Fifth Discipline*, (1992)

Gerald M. Weinberg, *Quality Software Management* vol. 1-4 (1991-1997)

Jeffrey K. Liker, *The Toyota Way* (2003)

Womack & Jones, *Lean Thinking* (2003)

Allen C. Ward, *Lean Product and Process Development* (2007)