Software Quality Management

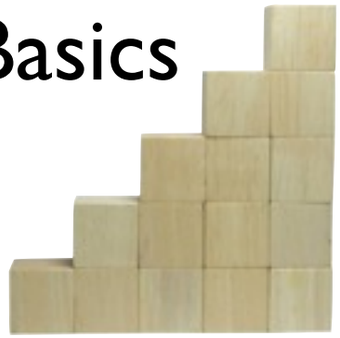
Dr. Stefan Wagner
Technische Universität München

Garching
16 July 2010

Quality management methods

Review of last week's lecture

Basics



Product

Quality

Metrics and

Measurement

Process

Quality

Quality

Management

Certifi-

cation

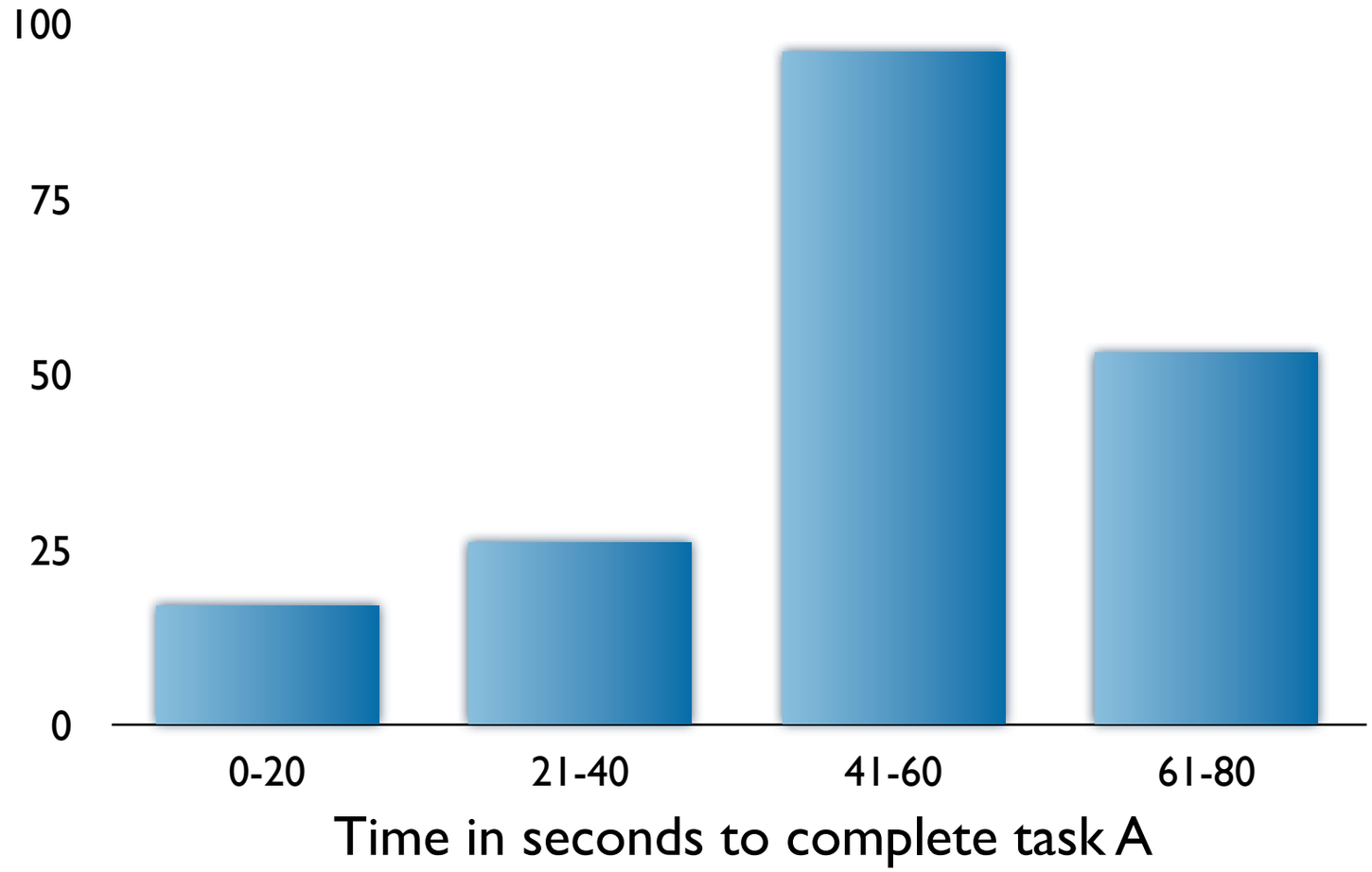
We are in the part "Quality Management".

Quality management methods

Quality planning

Today, we discuss examples of the quality management methods, we had in the group work last week.
In the quality planning part, we especially look at and try out QFD.

Histogram



A histogram is useful for showing the dispersion of data.
Usually, you have to group the data into different buckets to visualise them.
Here, the time that a user needs to complete a task is measured. We see that most users need between 41 and 60 seconds.

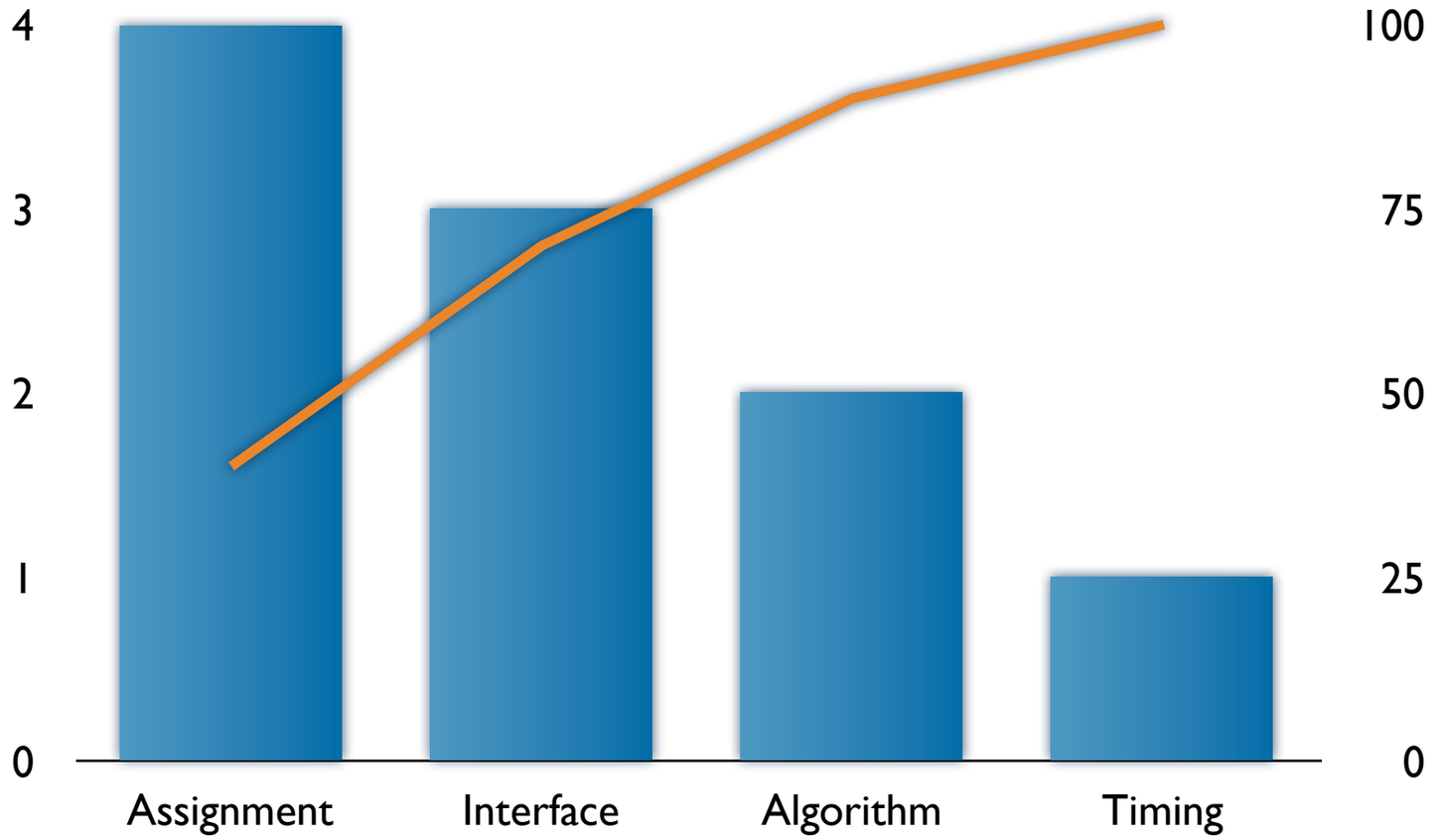
Check sheet

	Component A	Component B	Total
Interface defect	III		3
Assignment defect		IIII	4
Algorithm defect	I	I	2
Timing defect		I	1

A check sheet is a simple way to document counts.

Here, we count the number of defects of different types for the two components A and B.

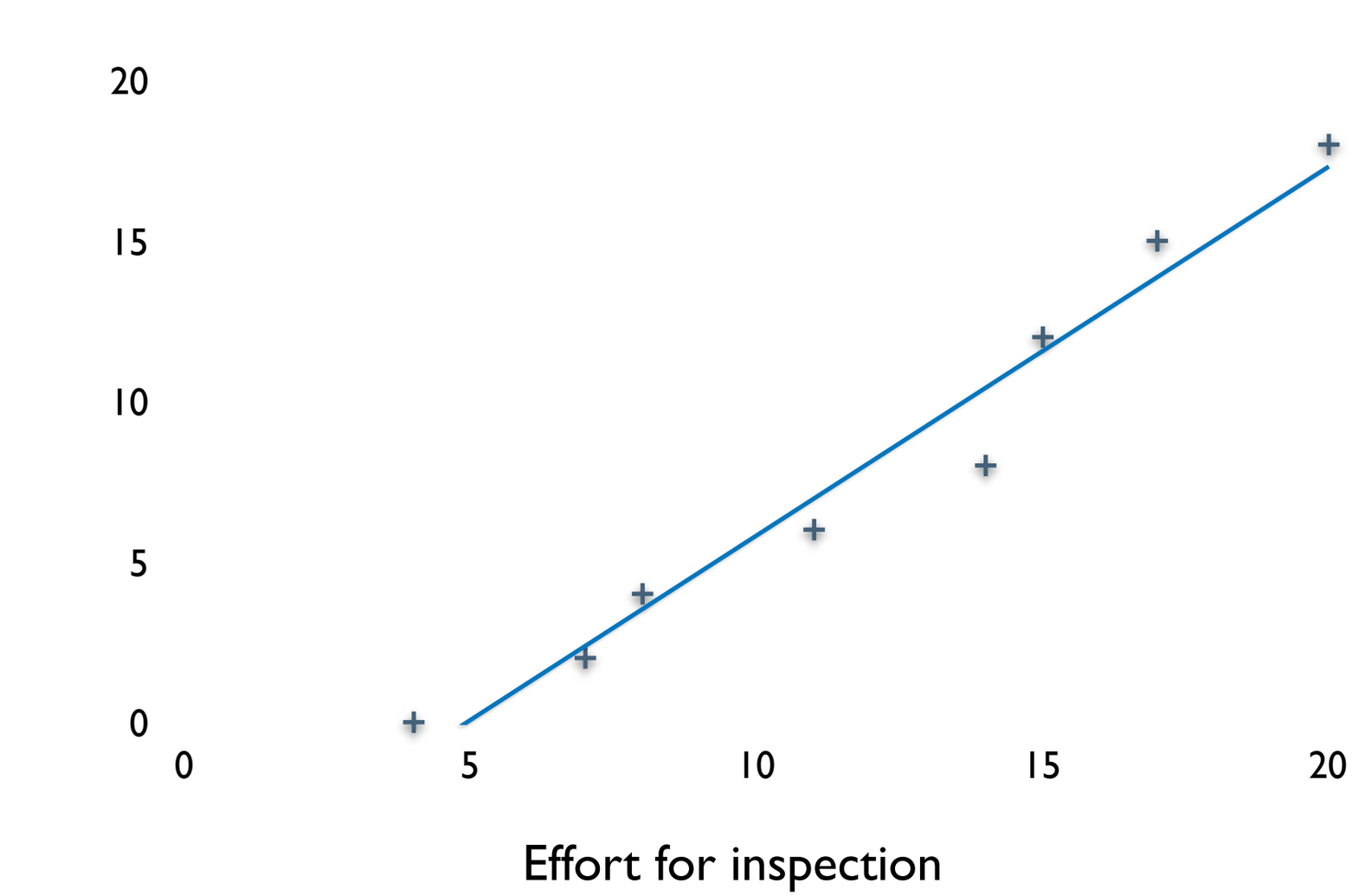
Pareto chart



The check sheet from the last slide is converted here into a Pareto chart.

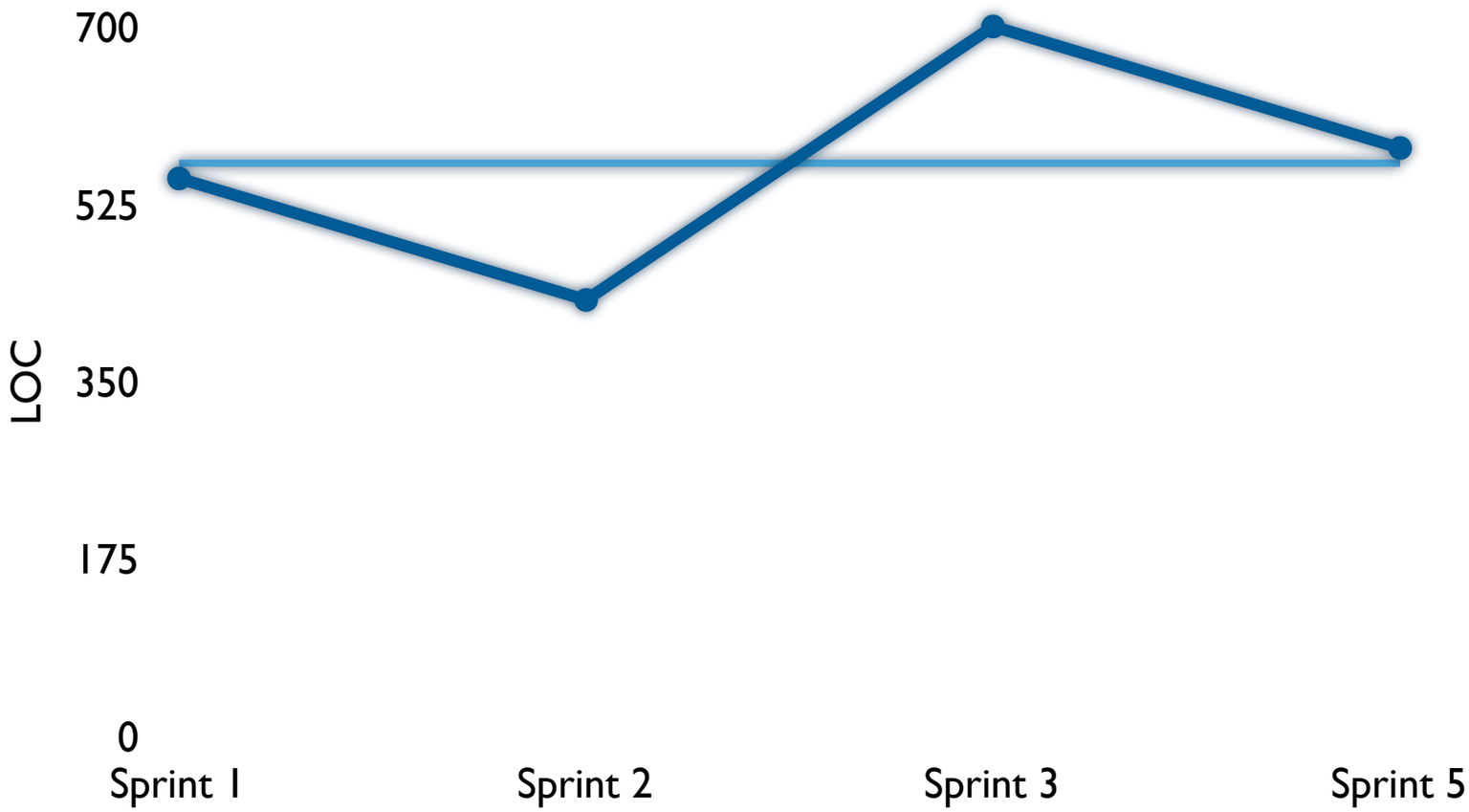
One part of the Pareto chart is similar to a histogram, but we order the data starting with the largest.
In addition we have a line that shows how the data adds up to the total (usually in percentages).
If this line is very straight, the data is equally distributed. If not, then we have a more typical Pareto distribution
with most defects in of one type, for example.

Scatter plot



A scatter plot shows the relationship between any two variables of interest.
It shows you, for example whether a higher investment in inspections paid off.
In this case, it seems that more effort for inspections results in a higher number of detected defects.
The straight line shows the possible linear relationship.

Run chart

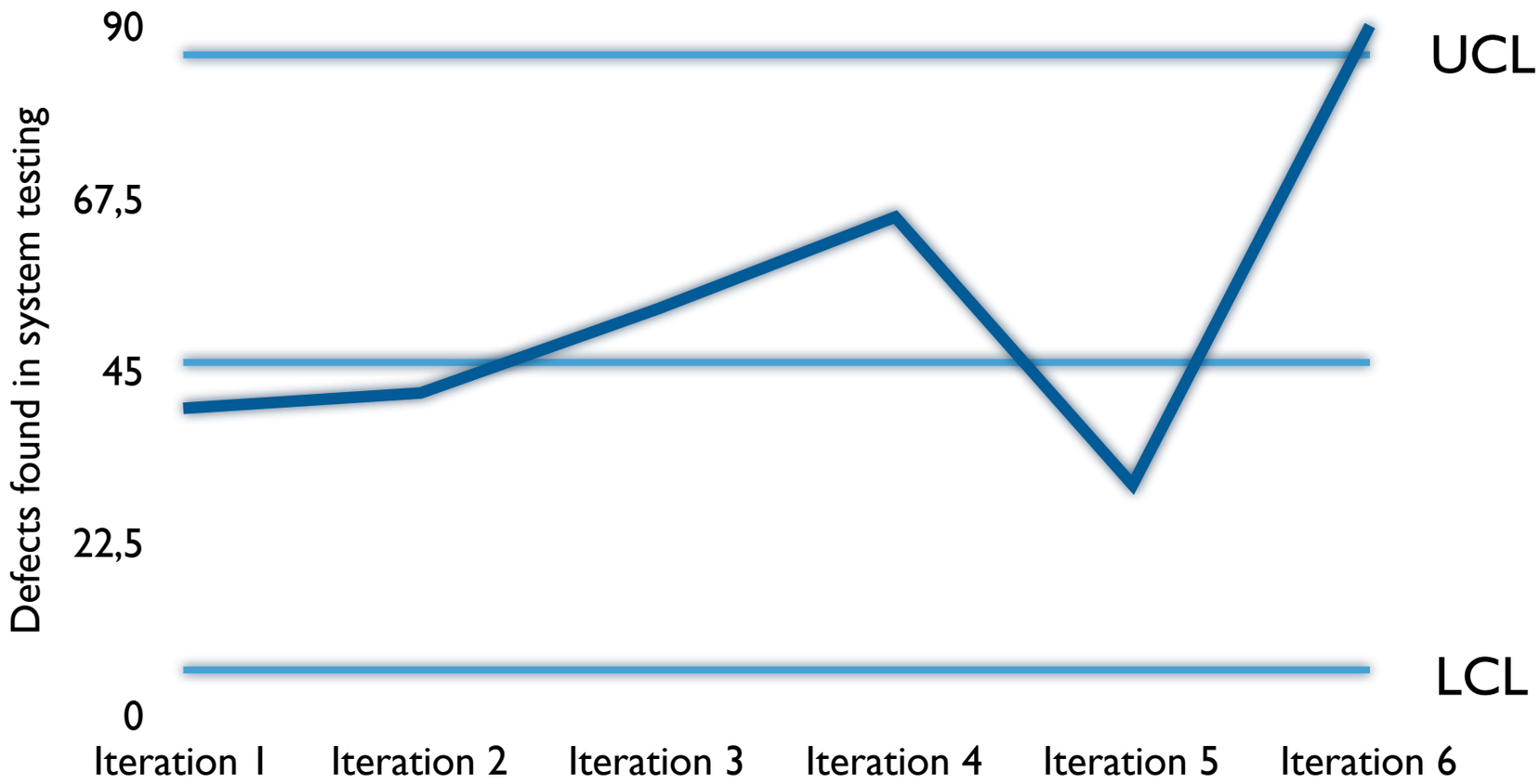


A run chart shows the change of a measure, usually over time. It is therefore similar to a trend chart.
In addition a run chart adds the average (shown as light blue here).
The example depicts the LOC that we produced in each sprint.

An extension of run charts can be used to detect problems in your process. What is a problem in the process?
A process varies over time. We cannot produce the same amount of LOC in each sprint, because of various factors.
For example, the requirements are always different, we use off-the-shelf components, or different people are involved.
These are all common cause variations.
In everyday life, a common cause for being a little bit late to work is a traffic jam. Sometimes there is more traffic, sometimes
there is less traffic. But it is normal.

If your car burnt down, however, you would not call that normal. This is a special cause for being late. Moreover,
you probably will be much later than with the delay from a traffic jam.
A burnt car is easy to detect. Strong and therefore special cause variations in development process are harder to detect.

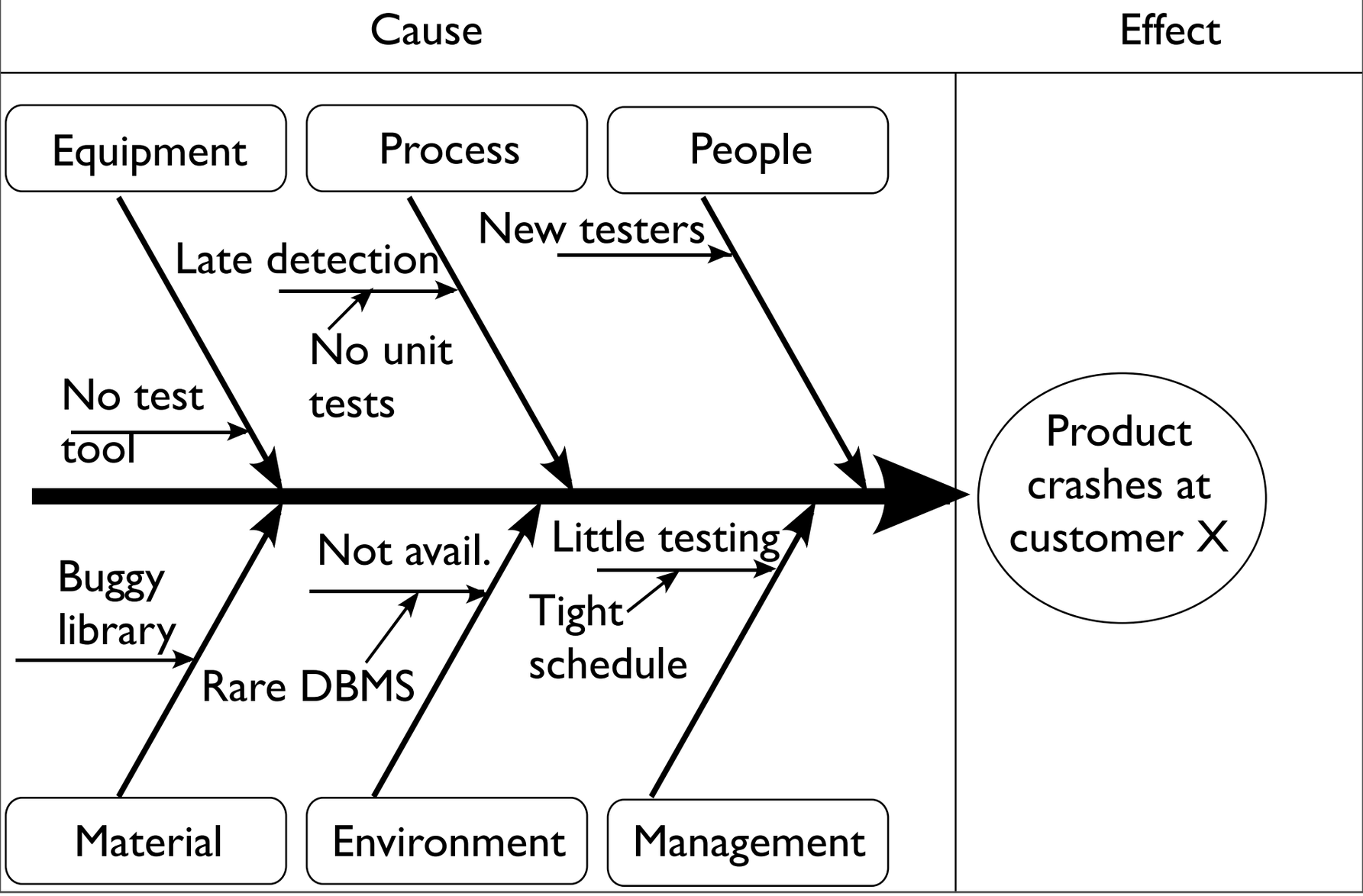
Control chart



Therefore, we use control charts to control the process and distinguish common cause and special cause variations.
We define control limits (UCL=upper control limit, LCL=lower control limit), which denote the borders of what we
call "common".
Often, this is 3 standard deviations from the mean (3σ). In the example, I calculated the mean and standard
deviations for iteration 1 to 5 to analyse whether iteration 6 crosses the control limits. It crosses UCL and therefore
we need to analyse the cause for this.

Control charts are the most important tools for statistical process control. It is highly disputed to what extent
it can be applied to software development for which we have far less data and more variation that is common.

Ishikawa fishbone diagram

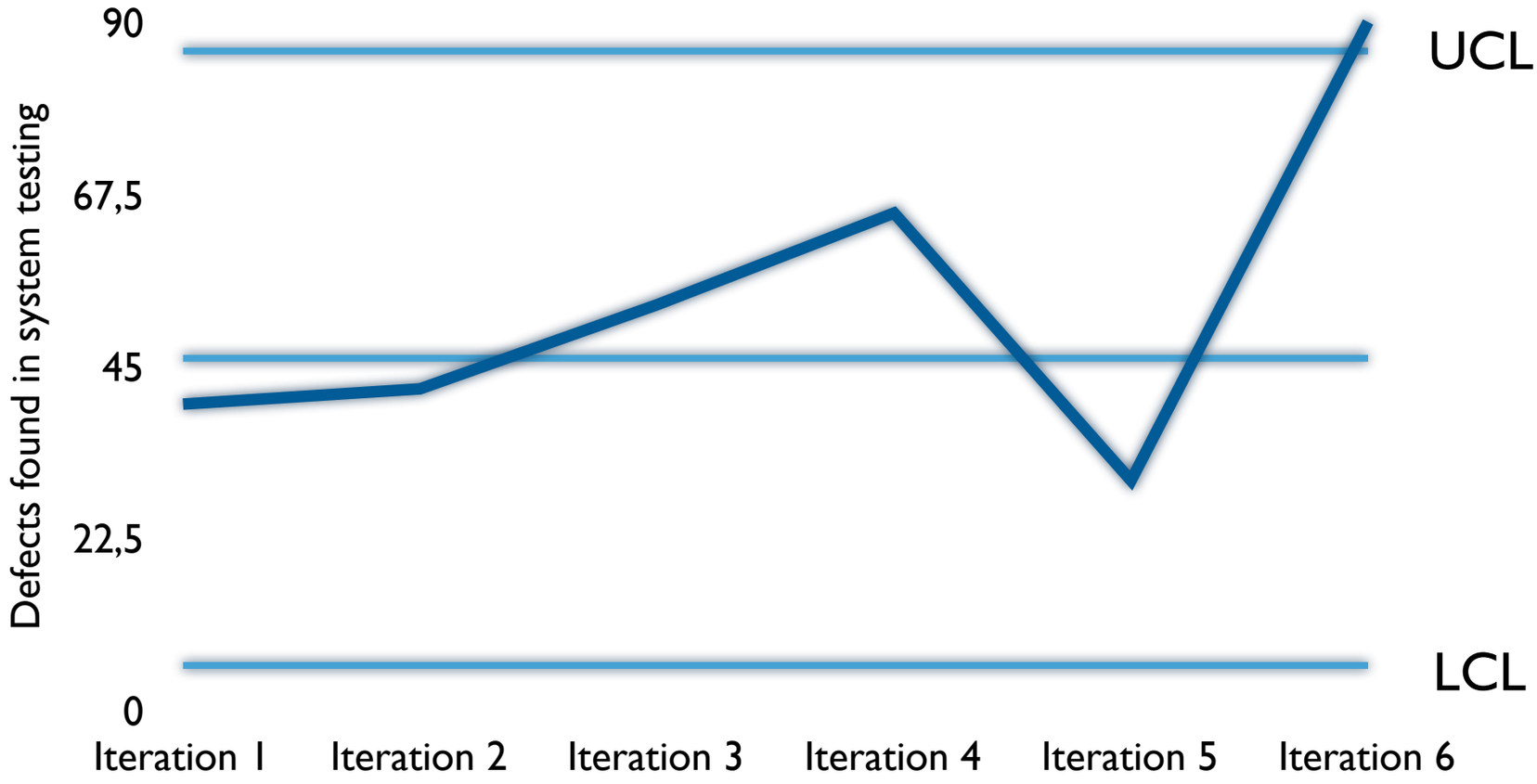


The cause-effect diagram can then be used to analyse, if we broke a control limit.

In the example, we want to find the causes why our product crashes at customer X. We use the standard topics
equipment, process, people, material, environment, and management to think about possible causes and how
they interrelate.

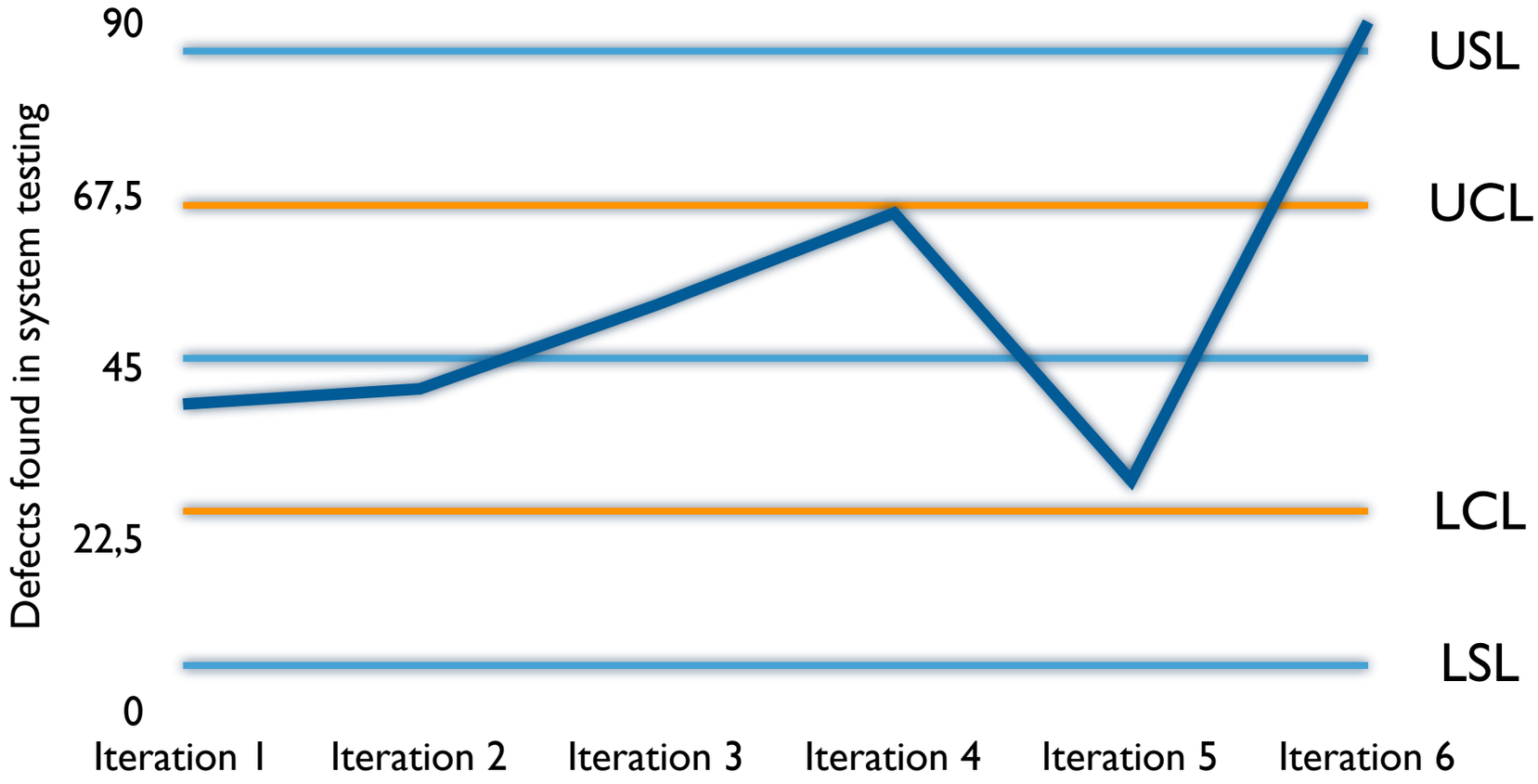
This can then be the start for a process improvement measure (maybe: introduce unit tests).

Control chart



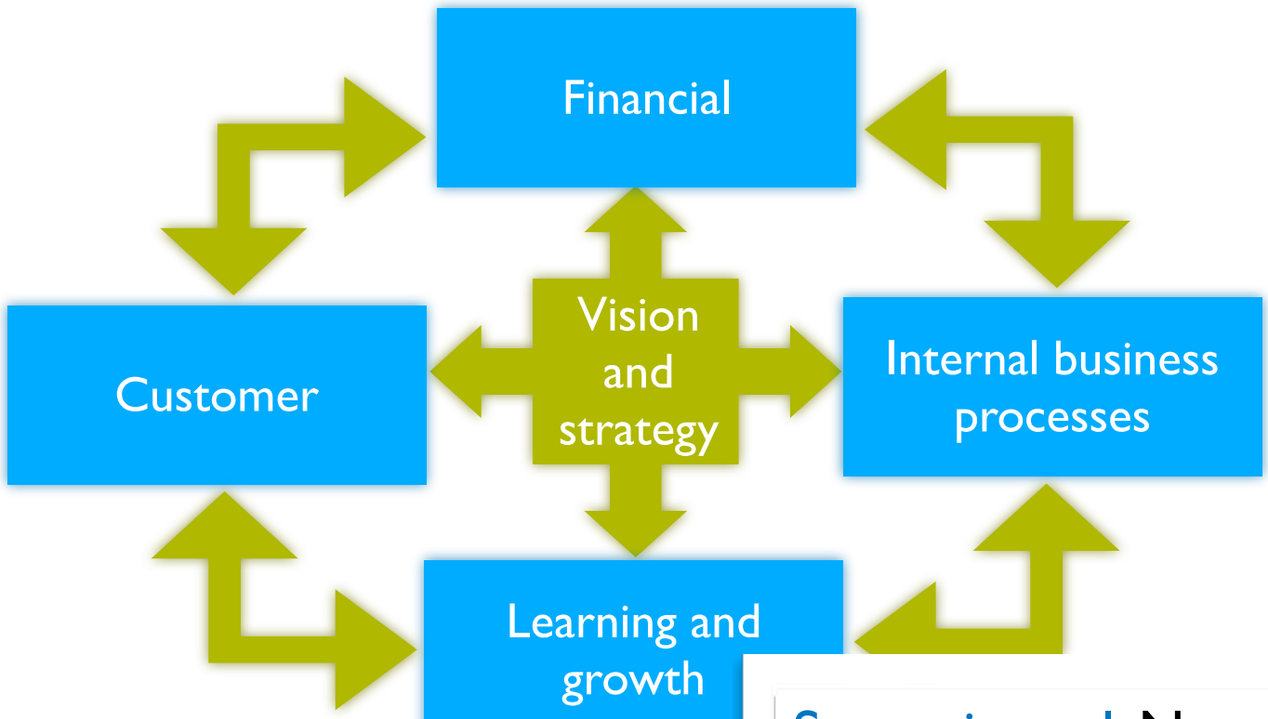
The control chart uses 3σ for the UCL and LCL.

Six sigma



The six sigma approach allows only smaller σ and then requires an analysis of 6σ. The 3σ still denote the UCL and
LCL, but we add an upper specification limit (USL) and a lower specification limit (LSL) at the 6σ level.
We want to detect variation as special cause much earlier.
In the example, iteration 4 already is at the border of the UCL.

Balanced score card



Strategic goal: New customers
Measure: New/total
Frequency: 3 months
Target: Increase by 4%
Initiative: Marketing campaign

The balanced score card is on a much higher level of abstraction. It is mostly a management method.
It is a structured approach to balance your strategy with respect to financial, customer, internal business processes, and
learning and growth aspects.
What comes out of it is usually a set of strategies with goals, measures, targets, and initiatives.

Quality management methods

Quality planning

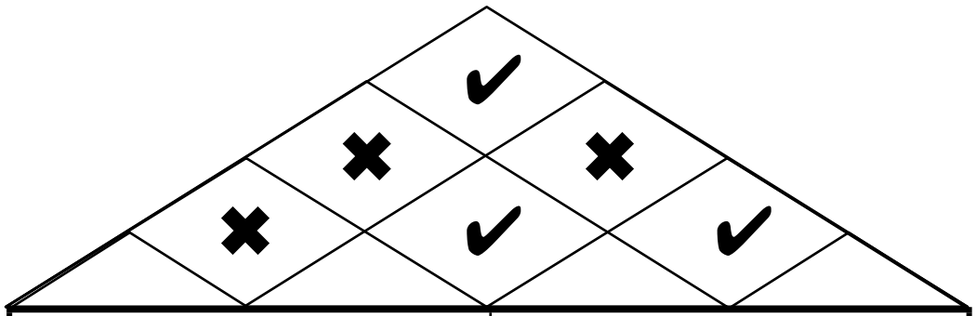
For quality planning, we use Quality Function Deployment.
Quality planning is the management task, which has the aim to find out what quality the customers want and
how we can reach that quality.

Front end of the QFD process

Ficalora, Cohen (2010)

The front end of Quality Function Deployment (QFD) is a requirements engineering process.
We gather the Voice Of the Customer (VOC) by standard means like interviews or questionnaires.
These VOC are then analysed by grouping and merging them.
Then they are prioritised and validated. This can be done together with the customer in workshops or
by building prototypes and user tests.
Finally, the work on the House Of Quality (HOQ) begins to add technical realisations to the customer needs.

House of quality



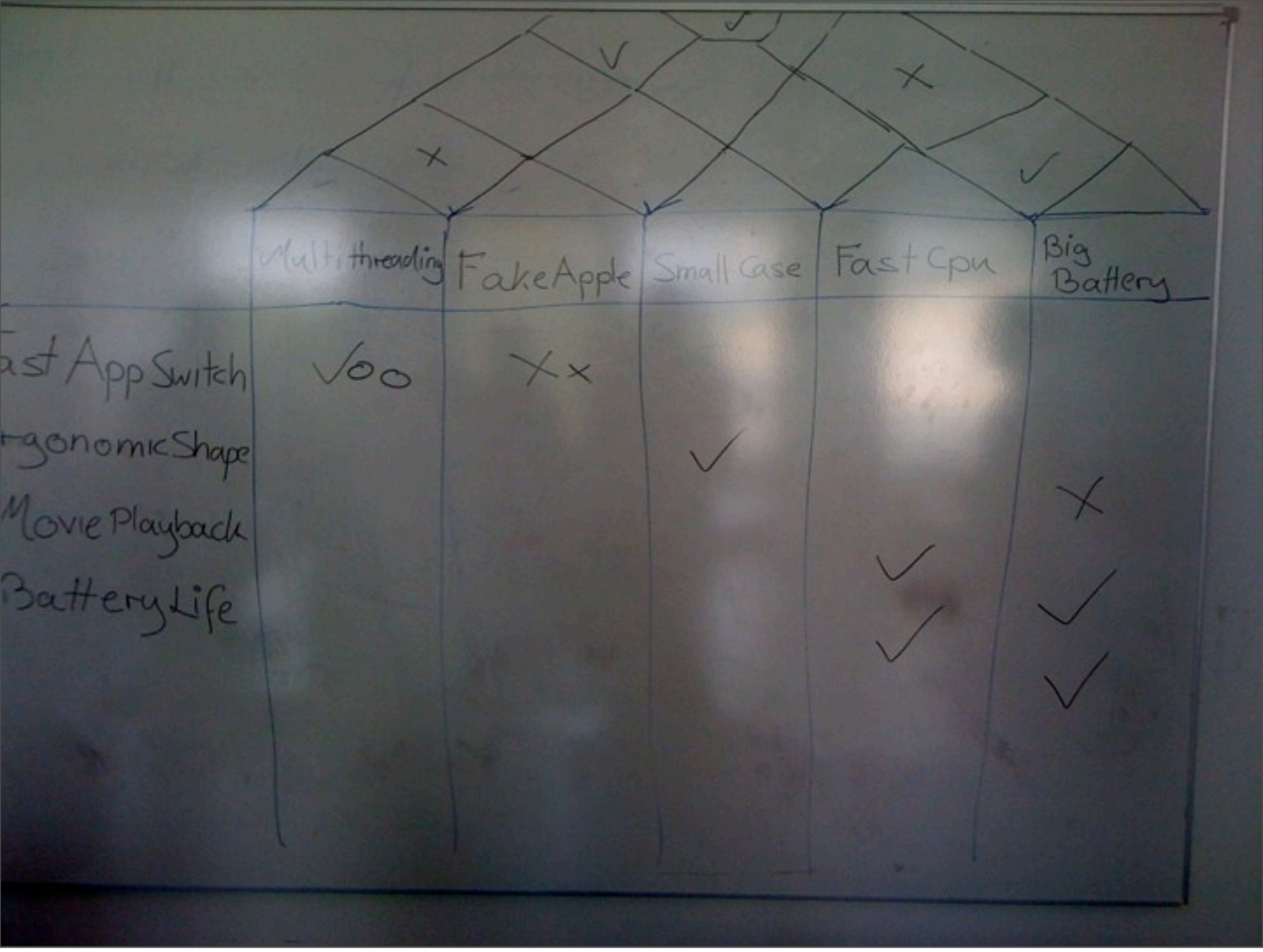
		GUI		Progr. Lang.	
		Native	Swing	Java	C
Easy to use	Quick learning	✔	✖		
	Intuitive control	✔	✖		
Reli-able	No total crashes			✔	✖
	Seldom malfunct.			✔	✖

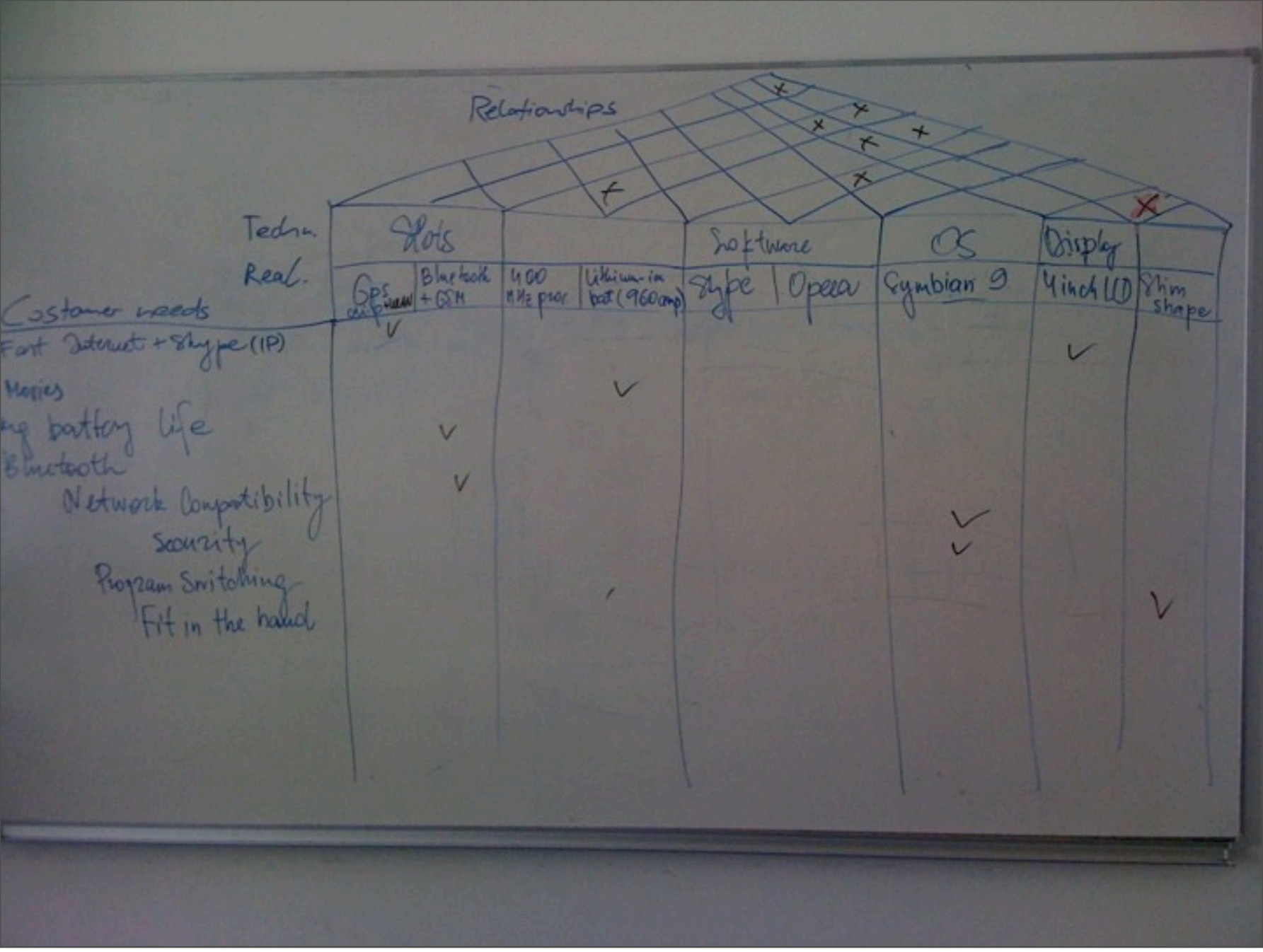
On the left, we have some example customer needs for a software system. The columns are possible technical realisations.
The check marks and crosses show whether a technical realisation is important for the need or is even obstructive.
The roof of the house shows similar relationships between the technical realisations.

From here, prioritise and resolve trade-offs. Then a second house can be built that shows design alternatives.

Group work

- Build a HOQ!
- 3 groups
- 20 minutes
- Customer statements on work sheet


Multithreading
FakeApple
Small Case
Fast Cpu
Big Battery
ast App Switch
gonomic Shape
Movie Playback
Battery Life


Relationships
Techn.
Real.
Customer needs
Fast Internet + Skype (IP)
Memories
Long battery life
Bluetooth
Network Compatibility
Security
Program Switching
Fit in the hand
Slots
GPS chip
Bluetooth + GSM
400 MHz proc
Lithium-ion batt (960 amp)
Software
Skype | Opera
OS
Symbian 9
Display
4 inch LCD
Slim shape

Quality management methods

Quality planning