Industrial Embedded Systems
- Design for Harsh Environment -

Dr. Alexander Walsch
alexander.walsch@ge.com

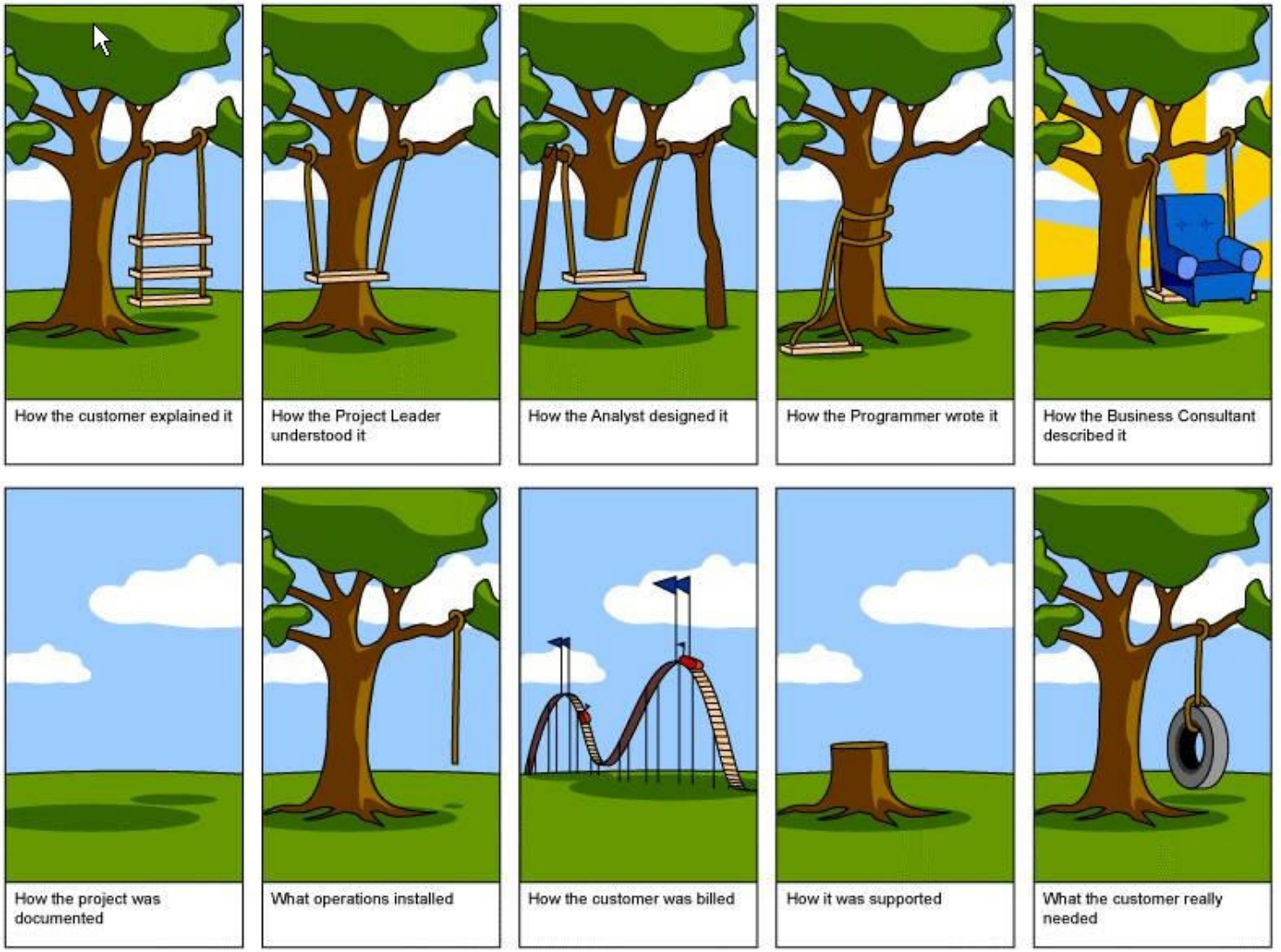
IN2244

Part II – Customer Requirements

WS 2014/15

Technische Universität München

Motivation
- What the Customer really needed -



Source: http://www.bowdoin.edu/~disrael/what-the-customer-really-needed/what-the-customer-really-needed.jpg

Requirements Engineering

- The requirements elicitation and analysis phase of embedded system development is about:
  - Getting all system functions together
  - Showing scope, usage, and constraints (performance, environment, regulation, threats, etc.) of the proposed system
  - Get a good understanding on effort and system architecture (risk reduction)
- Wrong (e.g. missing, contradicting) information will make us fail at a very cost intensive level → validation
- Once all information are available and validated the requirements are translated into a requirements specification which is a technical document for further development (metrics and defined format on all requirements)

Requirements Elicitation and Analysis

How do we get all these requirements?

- Involves technical staff working with customers or users to find out about the application domain (field technicians), the services that the system should provide and the system's operational constraints.
- May involve end-users, our customers, managers, engineers involved in prior development and/or maintenance, domain experts, certification bodies, etc. These are called stakeholders.
- Also non-functional requirements can be discovered in a systematic way (QFD, FTA, RBD, PHA, ...)

Challenges in Requirements Analysis

- Stakeholders don't know what they really want.
- Stakeholders express requirements in their own terminology – maybe not precise.
- Different stakeholders may have conflicting requirements.
- Political factors may influence the system requirements (e.g. disasters).
- The requirements change during the analysis process.
- Some requirements might be common sense and not explicitly mentioned.

Requirements Validation

- Validity
  Does the system provide the functions which the customer expects?
- Consistency
  Are there any requirements conflicts?
- Completeness
  Are all functions required by the customer included? Are more functions included?
- Realism
  Can the requirements be implemented given available budget and technology -> feasibility?
- Verifiability
  Can the requirements be shown to be implemented correctly (e.g. tested)?

Non-functional Requirements

- There are basically two kinds of requirements:
  - Non-functional (quality)
  - Functional (operations – IO)
- We will look into tools that help to gather requirements for
  - Safety: hazard analysis, fault trees (FTA), risk assessment (quality)
  - Reliability: (failure mode and effect) FMEA, FTA
- There are more non-functional requirements which will not be covered.

Non-Functional Requirements

Non-Functional Requirement

Constraints on implementation – How should the system be?

Includes

- Global constraints that influence system as a whole (shock, vibration, temperature, cost…)
- Function performance (response time, repeatability, utilization, accuracy)
- The “-ilities” (reliability, availability, safety, security, maintainability, testability, …)
- Other quality (ease of configuration and installation, …)

Non-Functional Requirements Capture

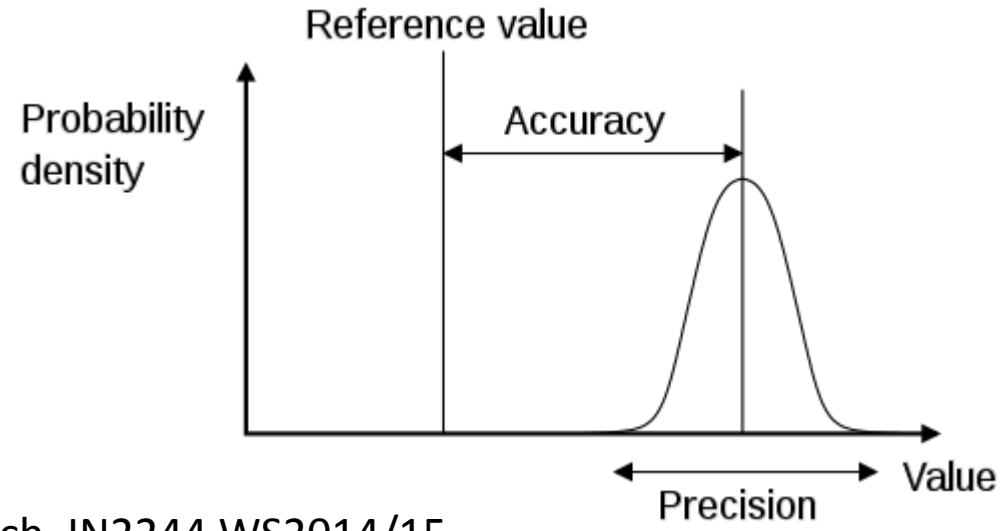
Look at system as black-box and concentrate on a specific use case

- Look at real-time aspects
  response time, sampling rate
- Data quality
  accuracy, precision
- Refine functional requirements – make more specific and testable
- Look at comparable systems (prior art, competitors)
- Safety (new laws or regulation) and reliability
- Hardware constraints (memory, CPU, IO)

Non-Functional Requirements
- Textual Examples -

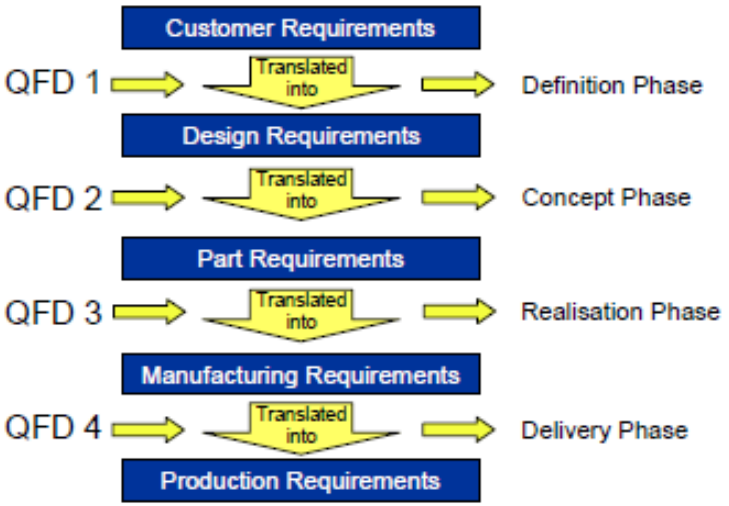
“Pressure samples shall be taken every 1s.”
“The response time for pressure measurement shall be less than 10ms.”
“Reliability: 1000 FIT”
“The measurement shall have an accuracy of 2%.”
“The measurement shall be repeatable with a precision not less than 0.5%.”
“The system shall meet the safety criteria according to [std.].”

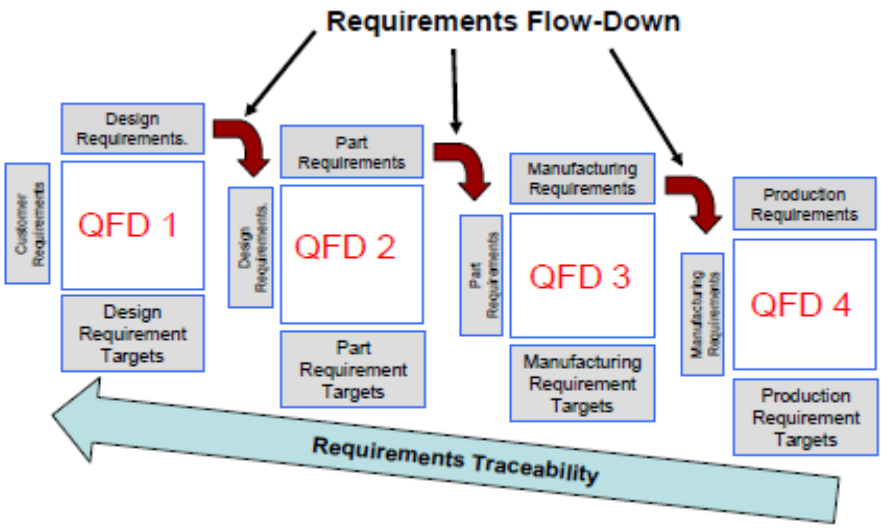
Source: wikipedia



Quality Function Deployment (QFD)

- QFD is a systematic way to correlate the relationship between (ambiguous) customer requirements and (precise) technical requirements.
- QFD is based on a sequence of matrix charts.
- QFD has been introduced for quality planning in manufacturing (Akao 1960s) but is a general methodology that can also be applied to computer system design.





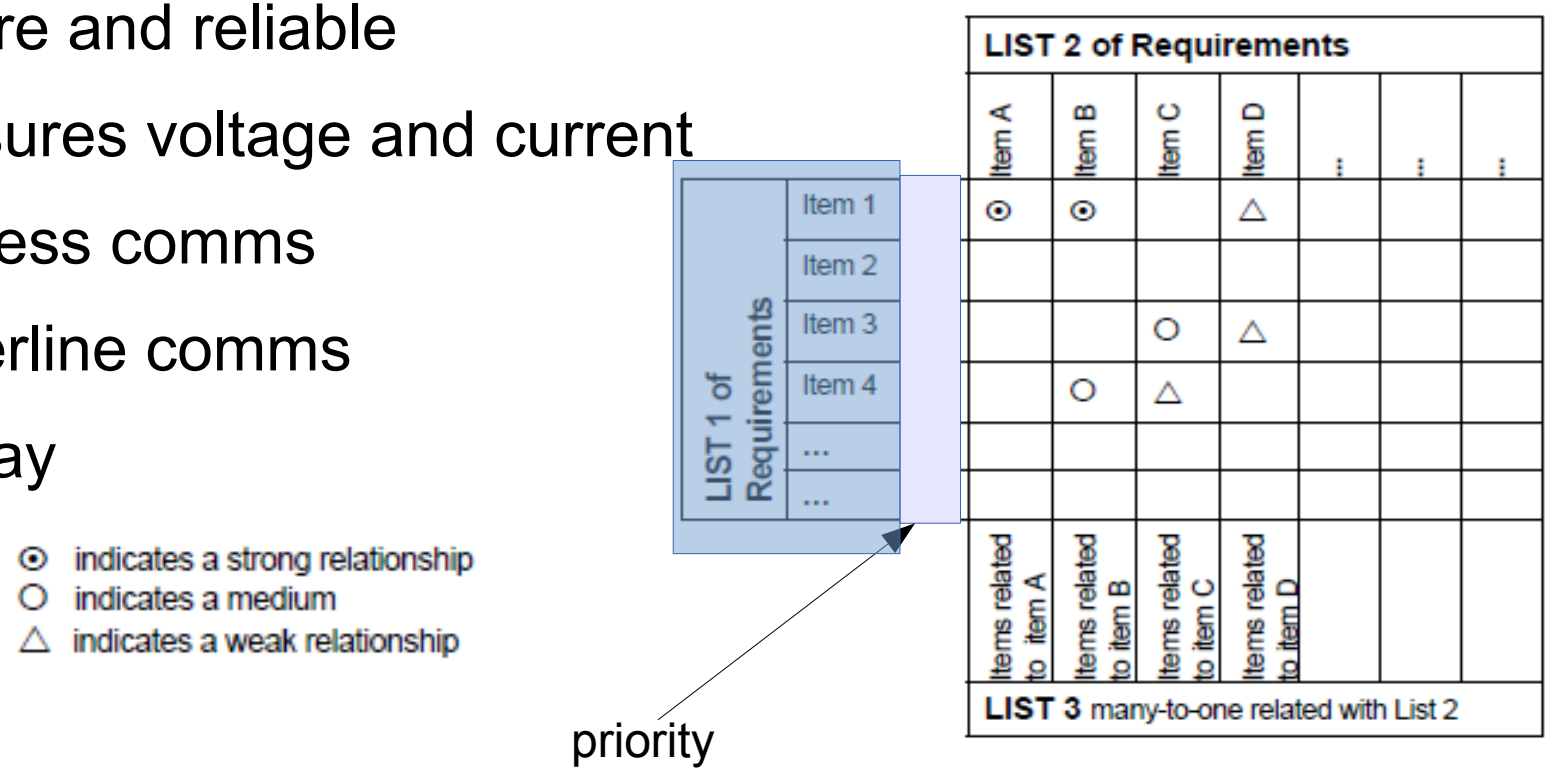
Source: Burge, S
A functional approach to QFD

QFD for Software

- QFD needs to be adjusted to reflect embedded software development.
- Customer requirements – requirements as received from customer
- Design requirements → Software technical requirements (functions)
- Part requirements → Architectural requirements (coarse design, larger entities, look at cohesion and coupling)
- Manufacturing requirements → Detailed Design (functions, classes, algorithm, data)
- Production requirements → Implementation (coding details, code quality)

QFD Example
- See Whiteboard -

- Development of a smart meter
- Customer requirements:
  - Inexpensive
  - Secure and reliable
  - Measures voltage and current
  - Wireless comms
  - Powerline comms
  - Display

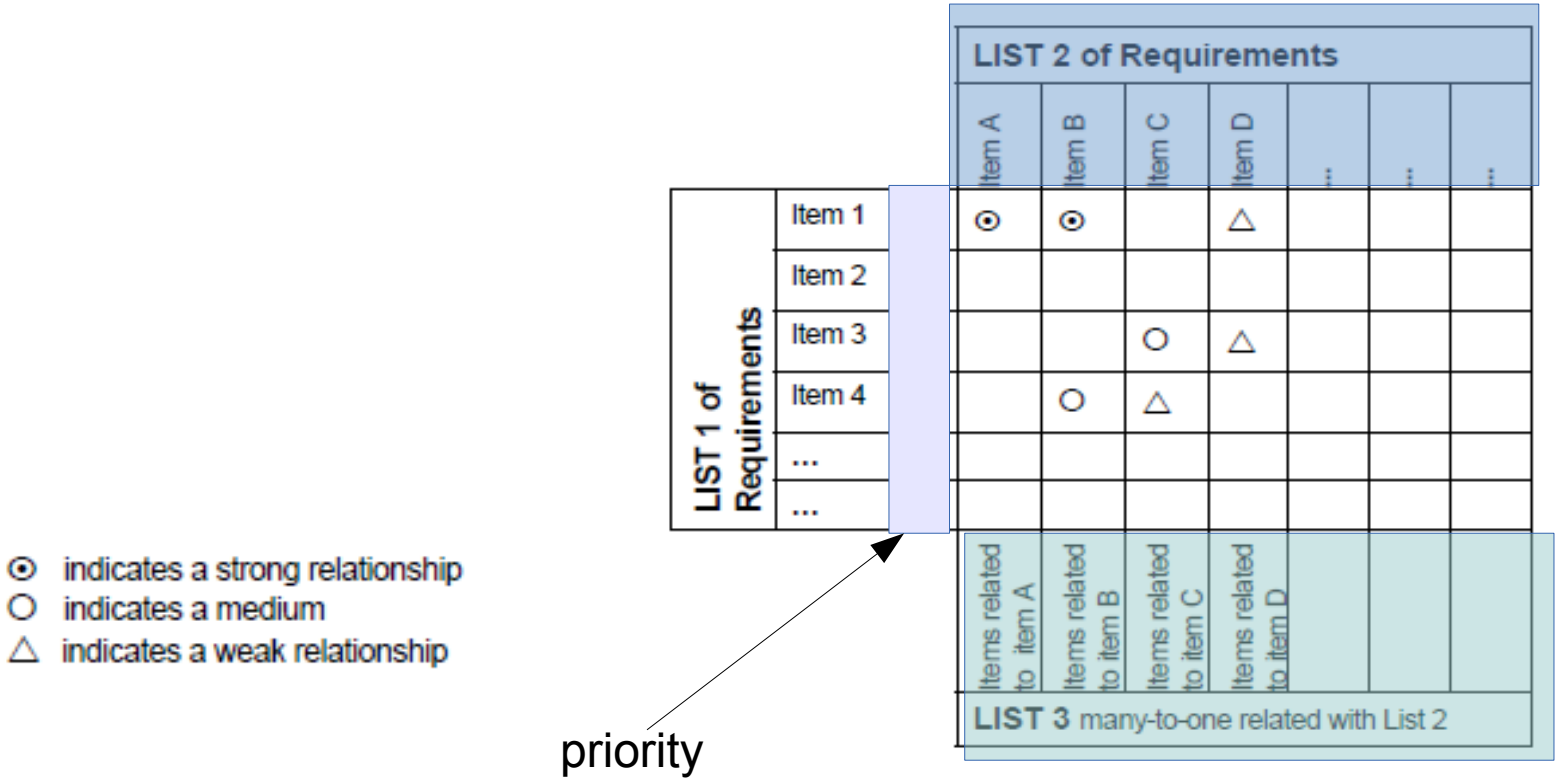


⊙ indicates a strong relationship
○ indicates a medium
△ indicates a weak relationship

Source: Burge, S
A functional approach to QFD

QFD Example
- See Whiteboard -

- Technical requirements (design requirements)
- Targets for technical requirements (non-functional)



Source: Burge, S
A functional approach to QFD