Industrial Embedded Systems
- Design for Harsh Environment -

Dr. Alexander Walsch
alexander.walsch@ge.com

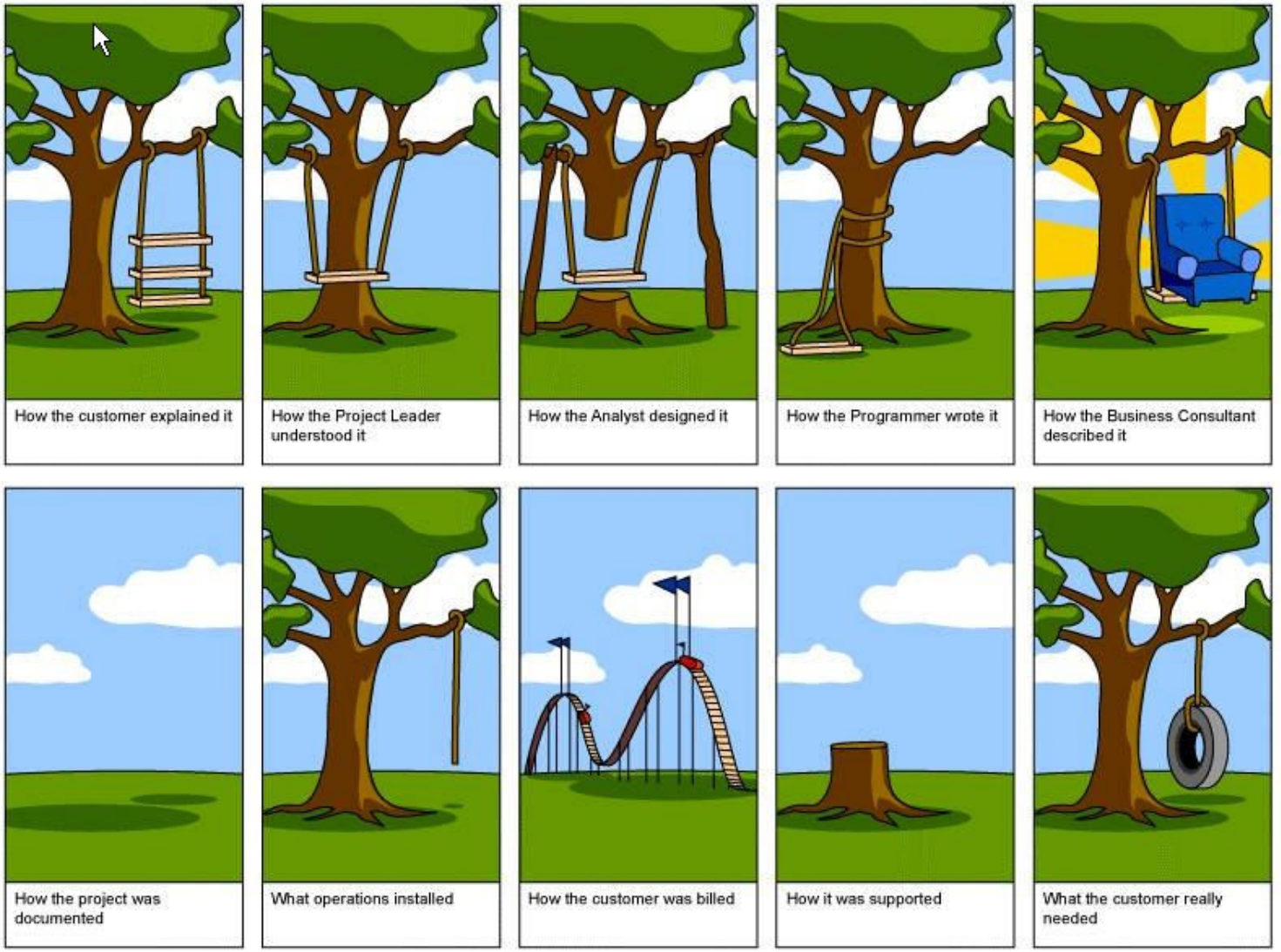
IN2244

Part II – Customer Requirements

WS 2015/16

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 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)
- 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)
- Wrong (e.g. missing, contradicting) information will make us fail at a very cost intensive level → validation (analysis)

Requirements Elicitation

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 Elicitation

- 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 (Analysis)

- 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)?

Requirements

- There are basically two kinds of requirements from the customer:
  - Non-functional (quality)
  - Functional (operations – IO)
- In addition, 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

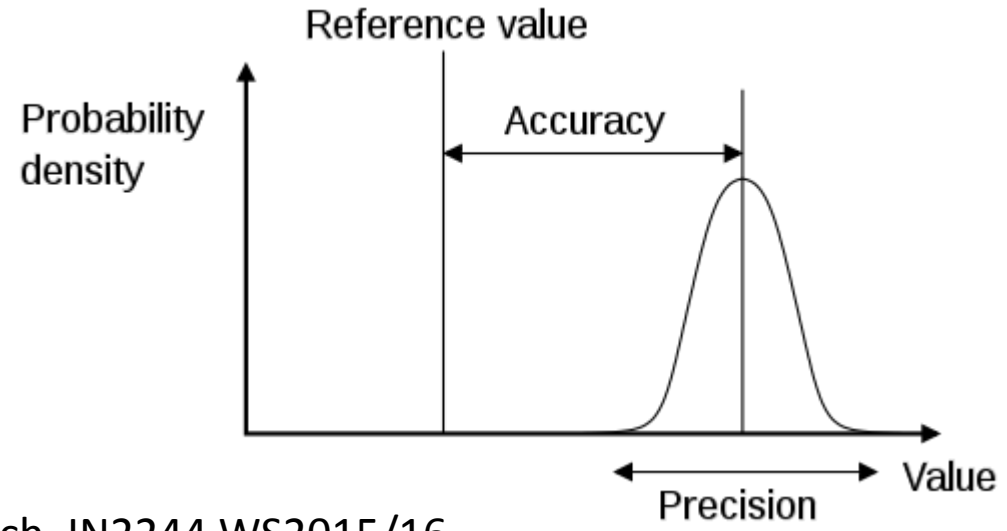
Look at system as black-box and concentrate on a specific use case

- 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)

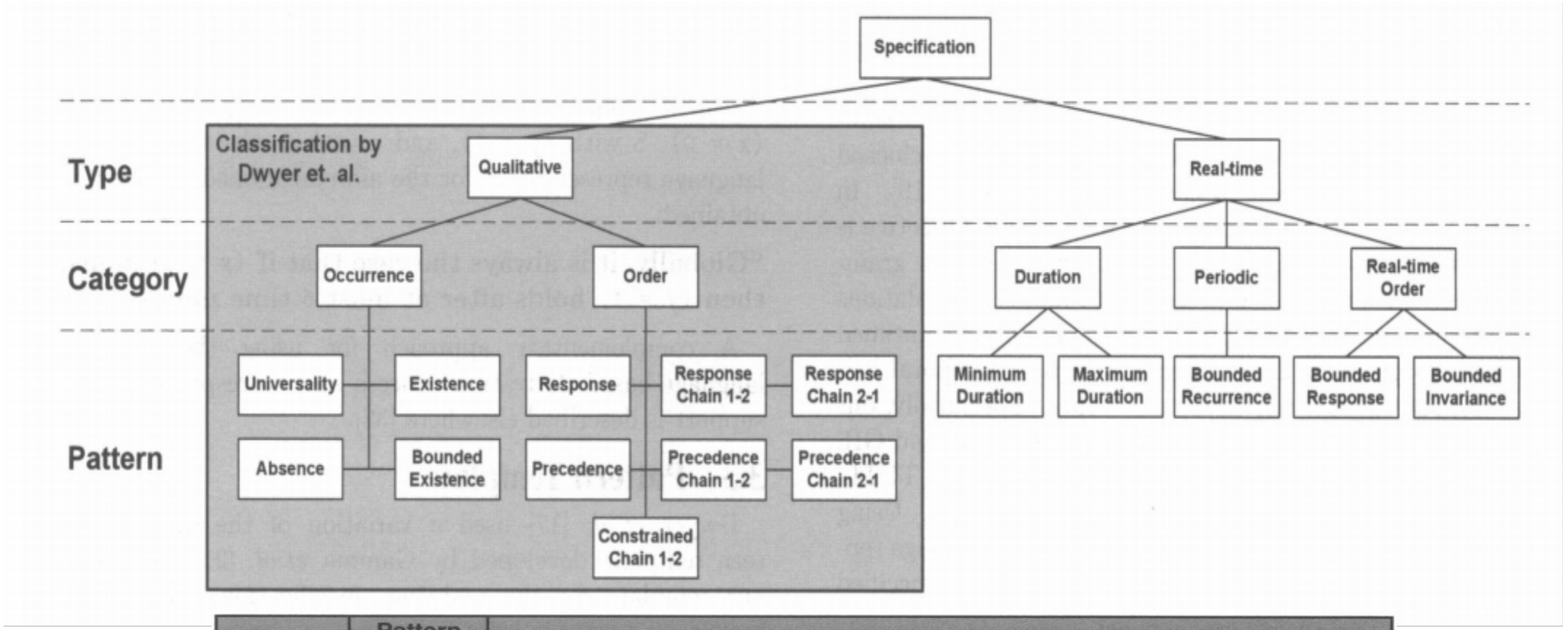
Non-Functional Requirements
- Textual Examples -

“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



Requirement Patterns



Category	Pattern Name	Description
Duration	**Minimum Duration**	Describes the minimum amount of time a state formula has to hold once it becomes true. (Ex.: Engine starter system "The system has a minimum 'off' period of 120 seconds before it reenters the cranking mode.")
	Maximum Duration	Captures that a state formula always holds for less than a specified amount of time. (Ex.: Engine starter system "The system can only operate in engine cranking mode for no longer than 10 seconds at one time.")
Periodic	**Bounded Recurrence**	Denotes the amount of time in which a state formula has to hold at least once. (Ex.: ABS system "The ABS controller checks for wheel skidding every 10 milliseconds.")
Real-time Order	**Bounded Response**	Restricts the maximum amount of time that passes after a state formula holds until another state formula becomes true. (Ex.: ABS system "From direct client input, detection of and response to rapid deceleration must occur within 0.015 seconds.")
	Bounded Invariance	Specifies the minimum amount of time a state formula must hold once another state formula is satisfied. (Ex.: Engine starter system "If Error 502 is sent to the Driver Information System, then the braking system is inhibited for 10 seconds.")

Source: Konrad and Cheng, Real-time Specification Patterns

Tools

	AF3	RAT/RATSY	STIMULUS	Embedded Specifier
supplier	Fortiss	Graz University of Technology and Fondazione Bruno Kessler, Trento, Italy	Argosim	BTC
contact (e.g. web site)	http://af3.fortiss.org/	http://rat.fbk.eu/ratsy/	http://argosim.com/	http://www.btc-es.de/
tool type (e.g. Commercial, TRL)	free, high TRL	open -LGPL, medium TRL (academic)	commercial	commercial
tool description	Tool suite from requirements engineering to deployment (auto test case generation, schedulability, etc.). Provides capability to enter text based requirements.	RATSY (Requirements Analysis Tool with Synthesis) is includes the capability to synthesize reactive systems from their temporal specification (input language). Furthermore, it includes a game-based approach for debugging specifications.	STIMULUS enables you to express natural language requirements using formalized language templates, state machines, and block diagrams (also with auto test case generation capability)	Starting with informal textual requirements, a formal and machine-readable specification is intuitively derived step-by-step. This machine-readable specification furthermore can be used for fully automated formal verification as recommended by automotive standard ISO 26262.
input (e.g. natural language)	natural language using templates	PSL: http://www.mel.nist.gov/msidlibrary/doc/nistir6459.pdf	natural language using templates	textual requirements, universal pattern
functionality (e.g. formal analysis, simulation)	Structure, analyze and verify requirements. Formal validation of requirements (NuSMV model-checker and Yices SMT solver)	Property Simulation, Property Assurance, Property Realizability and Synthesis, and Property Debugging using Games	user extendable templates, graphical entry (state machines, blocks), detect errors in requirements by simulation, include environmental assumptions	formal specification used to allow for verification and automatic test case generation