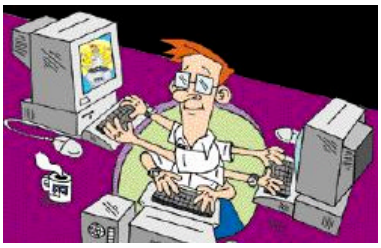
CONCEPT
HEIDELBERG

GMP Compliance for
Computerized Systems Validation
January 16 - 17, 2003 at Istanbul, Turkey

Requirements - the Key for System Validation

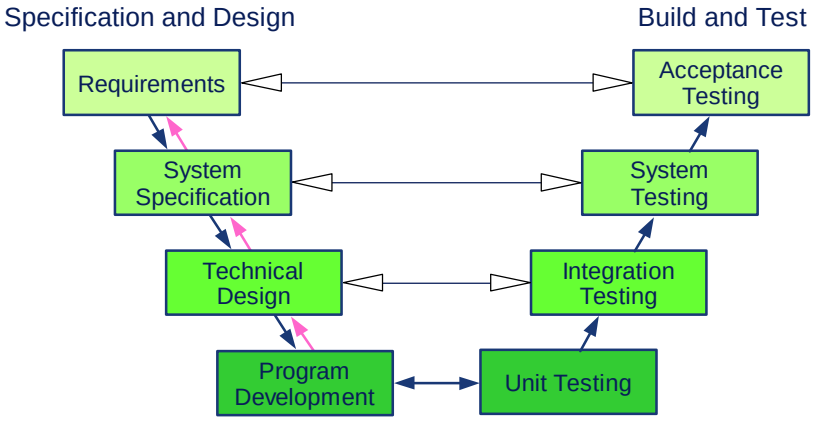
Dr.-Ing. Guenter Generlich
guenter@generlich.de



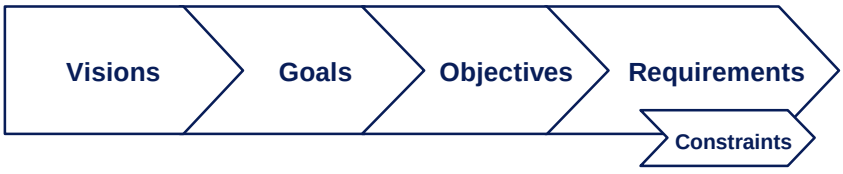
CSV Musts

- Requirements trigger **all** Validation activities
- Requirements must be documented
- All requirements need to be met and can be traced to the appropriate desigtn elements
- All requirements are verified and can be traced to a test or verification activity that shows that the requirement has been met
- The system is validated after the successful conclusion of acceptance testing

The Well Known V Scheme



The Requirements Hierarchy



Common Requirements Misconceptions

- We must start with firm requirements
- Requirements come from the users
- If it passes test, it must be o.k.
- The (Validation) problems are technical
- Validation is different
- We need more and better people / Validation specialists

Structured Approach I

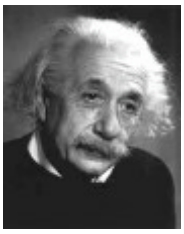
- Plan the work
- Follow up and maintain the plan
- Split the work into indepent parts
- Exactly define deliverables for each part
- Control dependencies of all parts
- Understand the system validierung as learning process

Structured Approach II

- Recognize knowledge gaps
- Close big gaps between knowledge and capabilities before proceeding
- Organize, monitor and review the work
- Fully concentrate on the job
- Update the plan with inproved validation experience

Simplify !

" Things should be as simple as possible -
but not simpler "



Albert Einstein

Reality

"Only the measurable is real"

Max Planck

Always the Same?

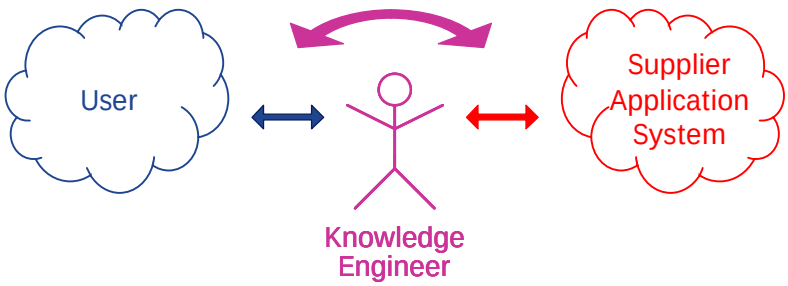
- User wishes
- User needs
- Business Requirements
- Business needs

If all you have is a hammer, everything looks like a nail

Requirements

- Requirements cannot be firm, because we cannot anti-cipate the ways the tasks will change when automated
- The user/user management doesn't generally know what is needed and neither does anyone else
- The initial requirements are (therefore) often wrong and will change
- Someone familiar with the application system need to be involved, and work together until the job is completed
- The larger and more complex the system, the more you will need expert domain knowledge

Requirements Development



Biases

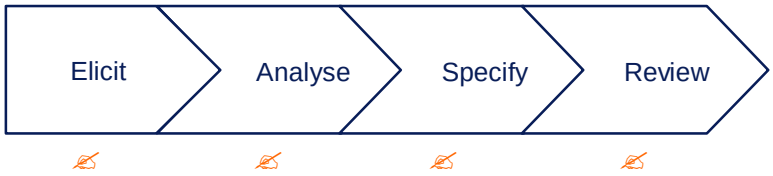
User	education many years ago experience present system, present work
Supplier	big sales expectations development for tomorrow´s market
Consultant	payed according to effort not accontable

Key Roles of Users

Users and their management must

- understand the requirements
- (actively) agree with and commit to the requirements
- be willing and capable to realize the requirements

The Requirement Process



Requirements Elicitation

- Write a Projekt Charter (Vision, Scope, Resources, Risks)
- Define a requirements development procedure
- Identify user classes and their characteristics
- Select product champion for each user class and focus groups of typical users
- Hold JAD sessions
- Identify Use Cases
- Determine quality attributes and other nonfunctional requirements
- Reuse requirements across projects

... **and document all this !**

Requirement Types

- Function
- Performance
- Operation
- Maintenance
- Safety / Security
- Control
- Quality
- Ergonomy
- Prototyping
- Look and Feel
- Standards
- Constraints
- Law / Regulations
- Interfaces
- Documentation
- Data Management
- Resources
- . . .

Anforderungen sammeln

- Interviews
- Brainstorming
- Requirements Workshop (special: JAD)
- Prototyping
- Use Cases
- Storyboards (paper prototypes)
- Problem reports of current systems
- Interface analyses
- Modelling (FOD ? P)
- Performance- and Capacity Analyses

Joint Application Development

Participants (good example):

- Facilitator — can act as Moderator
- End user — 3 - 5, always present
- Developer — 2 or 3, to clarify open questions
- Tie Breaker — Senior Manager, normaly not present
- Observer — 2 oder 3, don´t speak
- Domain Experts — improve knowledge and understanding

Prototyping Benefits

- Clarify and complete requirements
- Correct musunderstandings before build
- Avoid surprises to developer and user
- Save programming / reprogramming time
- Develop alternatives
- Gain user confidence
- Ease transition to operative system

Use Cases

- A Use Case descibes a user scenario of system communication in order to reach a certain goal or to perform a certain task
- Use Cases are successfully implemented for
  – requirements management
  – design of user - system interfaces (GUI)
  – definition of test cases

Requirements Analysis

- Draw a context diagram of the system
- Create user interface prototypes
- Analyse requirement feasibility
- Prioritize the requirements
- Model the requirements
- Match requirements and business goals and objectives
- Create a data dictionnary
- Apply Quality Function Deployment

Requirements Language

- Understandable for all stakeholders
- Formally correct
  - obligatory ? must, has to, shall
  - recommended ? should
  - desirable ? can
- Stimulate a dialog

Requirements Spezification

- Adopt an SRS template
- Label each requirement
- Identify sources of requirements
- Collect requirements attributes
- Record business rules
- Create a requirements traceability matrix

Requirements Attributes

- Business, user benefit and justification
- Realization effort
- Priority
- Status
- Source
- Responsibility
- Logical explanation
- Date
- Version
- Relation to other requirements
- History

Why Requirements Tracking

- All requirements are met and can be traced to the appropriate design elements
- All requirements are verified and can be traced to test or verification activity that shows that the requirement has been met
- Confirms there are no „extra“ system features, not shown as requirements
- Understanding for requirement changes

Sample Requirements Traceability Matrix

Requirement	Source	ID	GxP Impact	Funct. Spec.	Techn. Spec.	Program Module	Test Spec.	....	Remarks
Objective 1: Security									
The software product shall have three user access levels with capability to add new access levels in the future	Conference record May 2, 02	SEC1-0	Y	FS02.3.1	DS_Doc 1.4.5	PT68-C	SY-Test 24#5-7; Acc-Test 04-54a	....	See also minutes May 2, 02
								....	
								....	

More columns make often sense, e.g.
- test verification
- req´mts dependencies
- req´mts changes
.....

Requirements Verification

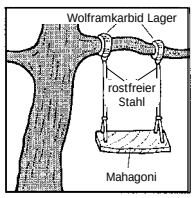
- Inbolve all stakeholders - team approach
- Inspect requirements documentation
- Define acceptance criteria
- Write test cases from the requirements
- Write a user manual

Requirements Characteristics

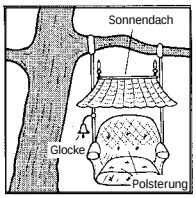
- business related
- complete and correct
- exact and unambiguous
- actual and/or future directed
- verifiable and testable ✍ validatable
- annotated and justified
- internally consistent
- prioritized and timed
- necessary and important
- below 100%

- uniquely identified
- traceable
- feasible
- manageable
- free of unwarranted design details
- modifiable
- implementation-free
- understood and accepted

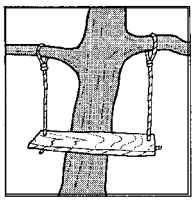
Product Development

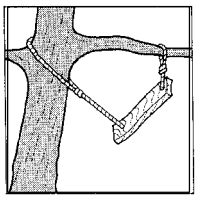


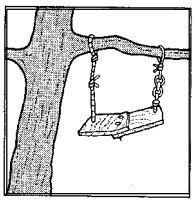
What the user asked for

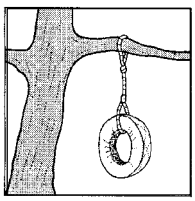


What the analyst promised

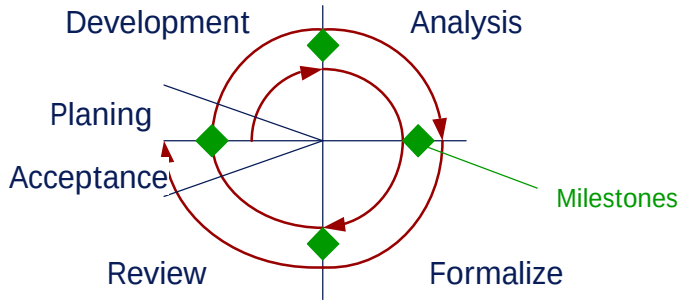

What the designer specified


What the programmers produced


What went into production


What the "business actually needed

The Requirements Spiral



Requirements Management Tools

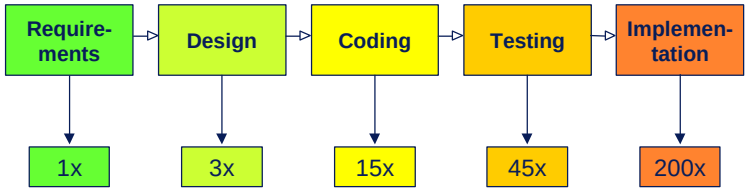
- Manage versions and changes
- Store requirements attributes and annotations
- Ease traceability
- Facilitate impact analysis
- Track requirements status
- Control access
- Communicate with stakeholders
- Reuse requirements

Keys for Good Requirements

- Train users, developers and management (✍ stakeholders)
- Develop co-operation user - developer
- Understand different requirement types
- Recognize different requirement sources
- Iterative aproach
- Standardize Templates
- Review documents formally and informally
- Write test plans / test cases early
- Manage requirement changes

Defect Elimination Costs

Cost Faktors for Defect Elimination



Barry Boehm 1981: Software Engineering Economics

Epilogue

- Problems can be solved only at begin of development in a simple and cost-efficient manner
- Just 15% (with restrictions 40%) of all IT projects are a success
- **50%** of failures are caused by deficient, inadequate, etc. requirements
- Without „good“ requirements the system validation can get very expensive