Chapter 2
The System Development Process

Objectives

What are you about to learn?

Knowledge Objectives

- Understand the main elements of a typical real-time systems development process.
- Know about the main artifacts for project management, configuration management, change management, quality management, and system development.
- Understand a typical real-time system requirements engineering sub-process.

Skill Objectives

- Ability to develop a project plan and write a project handbook.
- Ability to work with version control systems.
- Ability to structure files and directories in a project.
- Ability to develop a requirements document (together with the techniques learned in a later chapter on modeling).

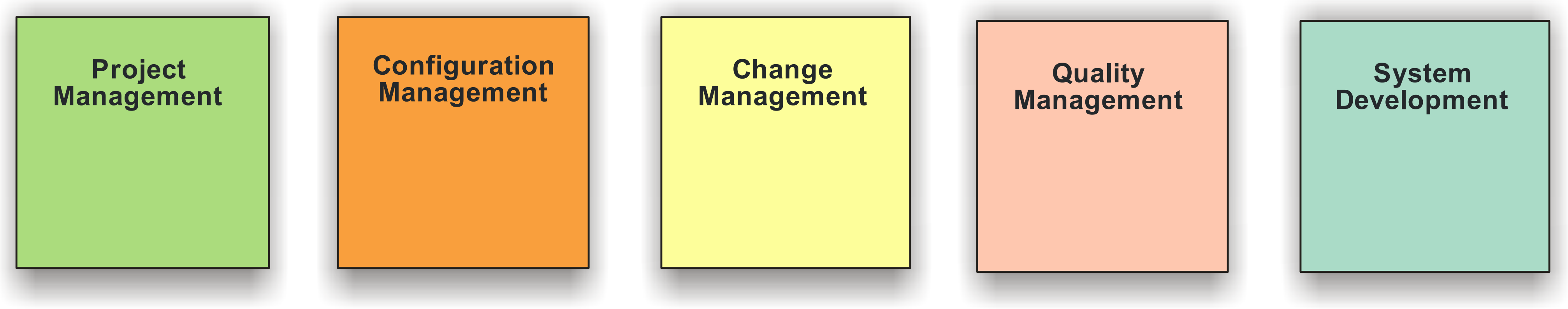
2.1 The V-Model Development Process

Process models describe:

- The organization of people (e.g., via roles)
- The activities including their dependencies
- The structure of documentation and other working products
- The responsibilities for activities and working products

The V-Model XT covers all areas for a real-time system development process. We take it as a basis for our taxonomy and own process.

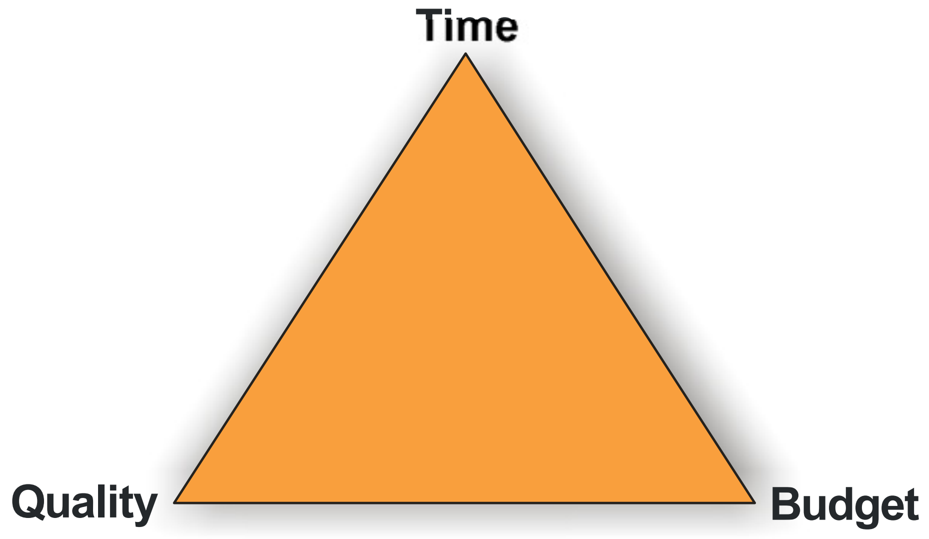
According to this model, all activities and products belong to one of five process areas:



2.2 Project Management

Project management activities lead to "products" that are important for managing the project. The main project management goals are

- Have the project deliver the desired products on time
- Have the project deliver the desired products within a given budget
- Have the project deliver the desired products with the required quality



The main project management activities are carried out by a "project manager". The project manager may have assistants, or delegate responsibilities to other managers or team members.
A typical real-world scenario is that all three parameters are fixed: time, quality, and budget.

2.2 Project Management

Project management products

Project management activities lead to “products”. Products are all documents which other people rely on for their own work and which are versioned. The main project management products are:

- A project handbook
- A project plan
- Project status information

The project manual contains the description of the actual development process. It defines

- which kind of roles are defined
- which kind of documents are to be developed
- which kind of activities are to be performed
- which kind of tools are to be used

It does not define the actual products and activities, only the generic ones: (e.g., “system architecture”, but not “system architecture of component B”).

The project plan defines

- the product structure (as far as it is known)
- the actual documents and products to be developed
- who is responsible for developing a particular document or product
- who is assuming which role(s)
- who is responsible for which activities, and when they are carried out

2.2 Project Management

The project status defines

- for each activity, how much work has been performed for it
- for each product, if it has been published or delivered
- all open points, problem reports, and change requests
- all open risks that could jeopardize the project goals

The project manual is rather static; it serves well as a reference for new project team members to get them briefed on processes and tools.

The project plan develops as the project progresses and new and more accurate information becomes available.

The project status is the most dynamic document, as it should be updated on a daily basis or at least once a week.

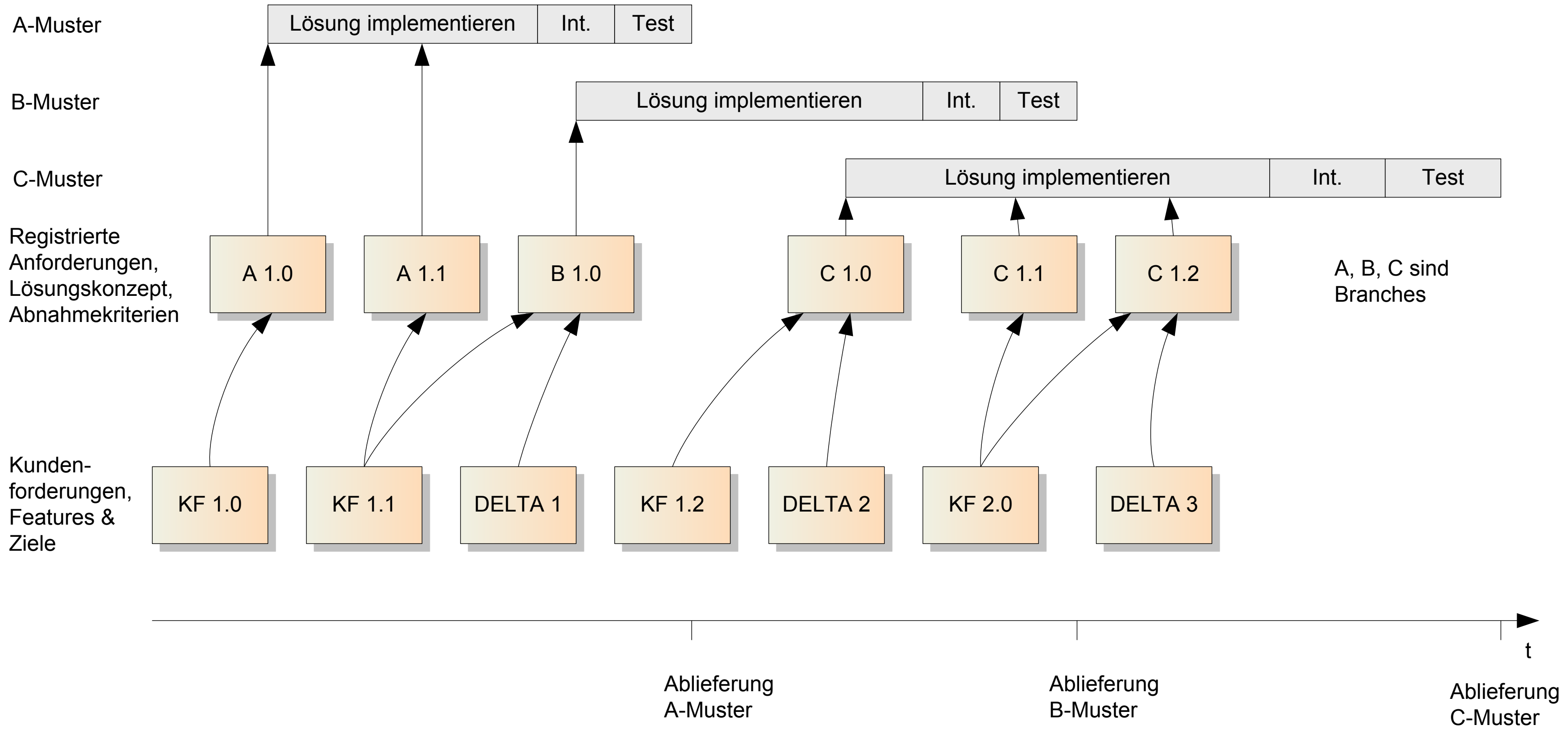
In our project, we use the issue tracking tool Track+ (http://www.trackplus.com) to maintain both, the project plan and the project status.

2.3 Configuration Management

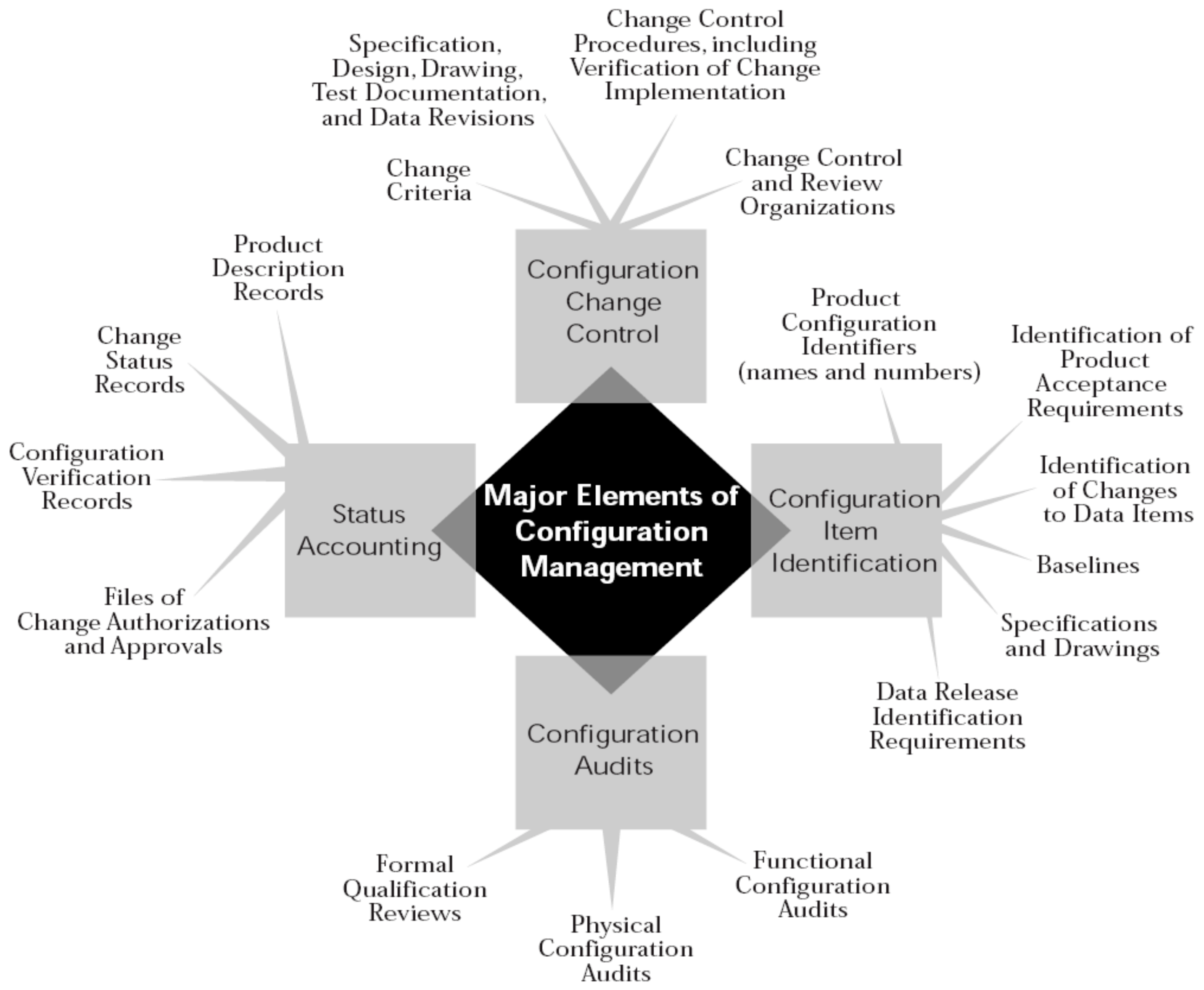
Configuration management (CM) bundles all activities that are related to the management of "versions" and "configurations" of products.

Each product mentioned in the project plan has to be managed by a configuration management tool (e.g., ClearCase, Perforce, Subversion).

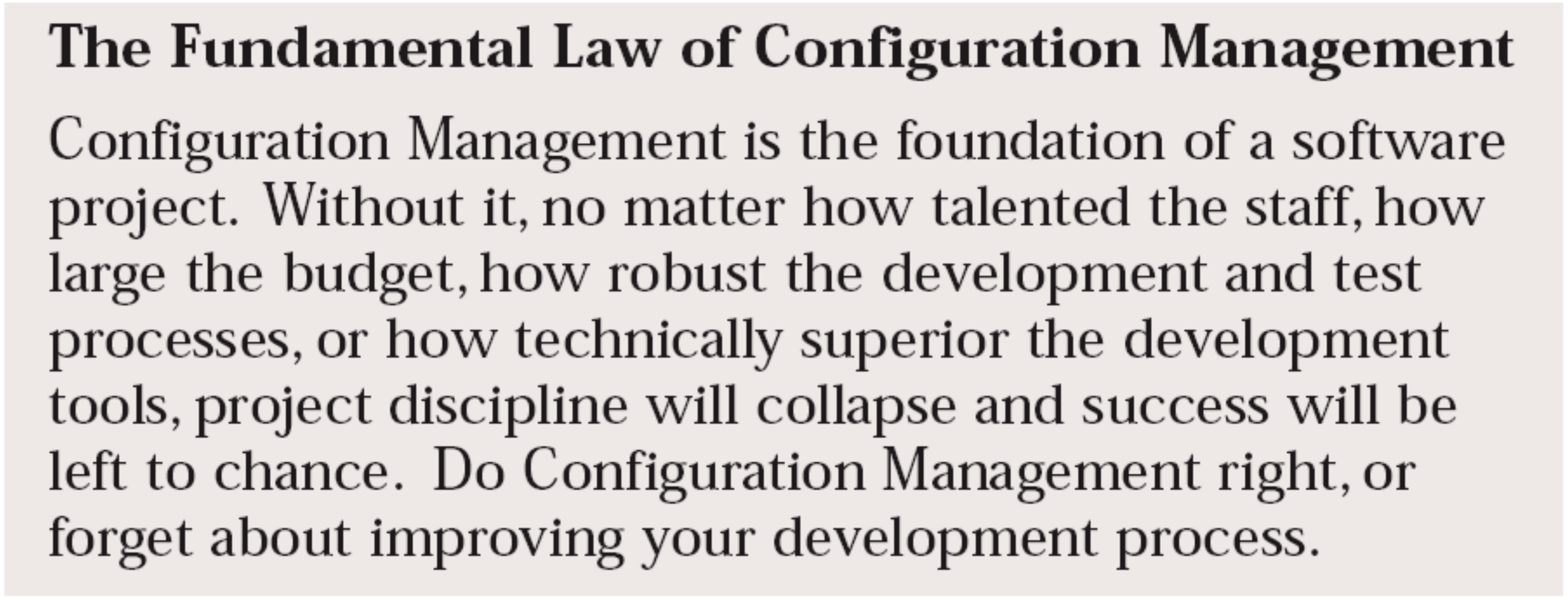
In our project, we use Subversion, http://www.subversion.tigris.org.



2.4 Configuration Management


Major Elements of Configuration Management
Configuration Change Control
Change Criteria
Specification, Design, Drawing, Test Documentation, and Data Revisions
Change Control Procedures, including Verification of Change Implementation
Change Control and Review Organizations
Configuration Item Identification
Product Configuration Identifiers (names and numbers)
Identification of Product Acceptance Requirements
Identification of Changes to Data Items
Baselines
Specifications and Drawings
Data Release Identification Requirements
Configuration Audits
Formal Qualification Reviews
Physical Configuration Audits
Functional Configuration Audits
Status Accounting
Product Description Records
Change Status Records
Configuration Verification Records
Files of Change Authorizations and Approvals

2.4 Configuration Management


The Fundamental Law of Configuration Management

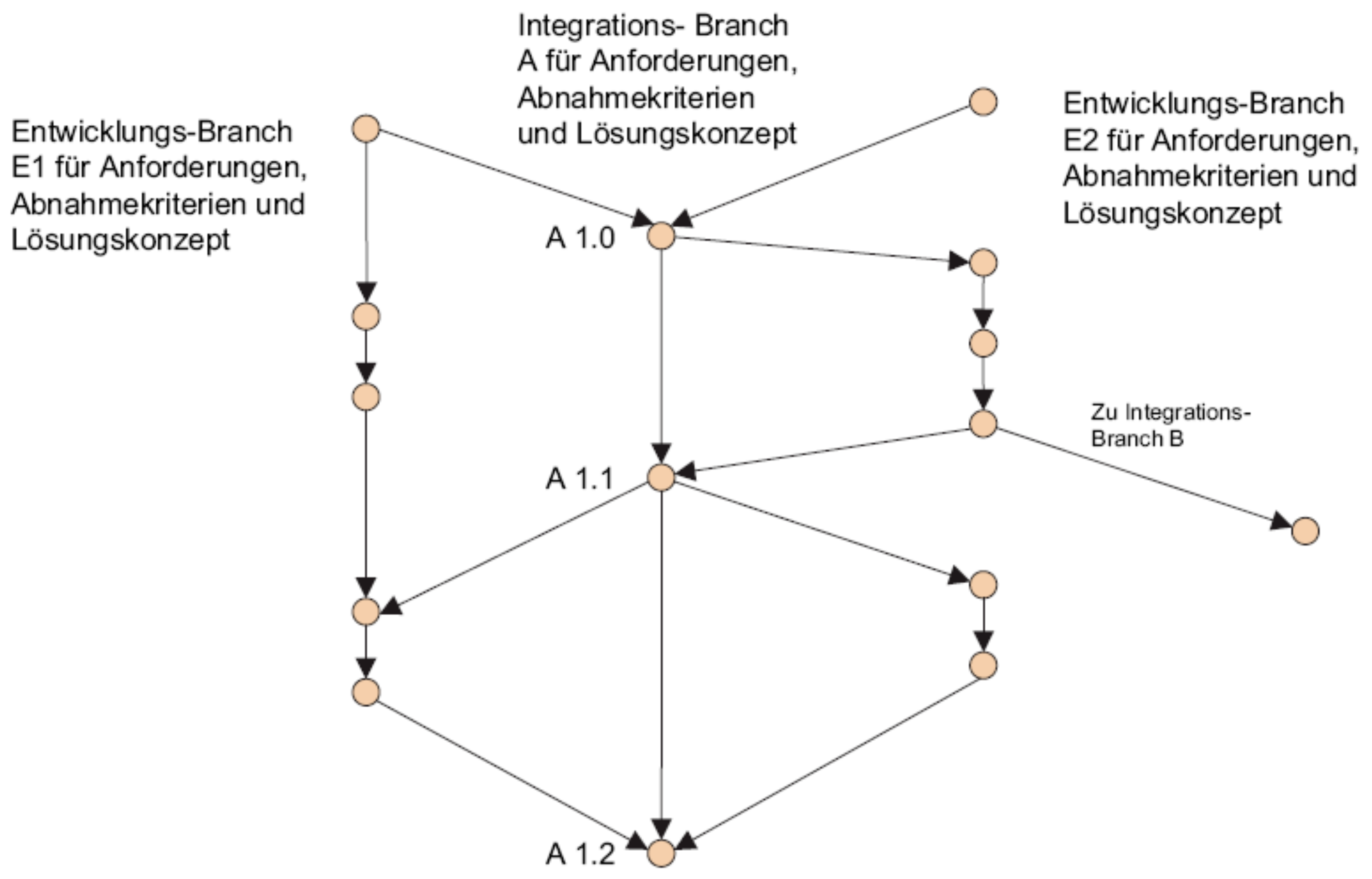
Configuration Management is the foundation of a software project. Without it, no matter how talented the staff, how large the budget, how robust the development and test processes, or how technically superior the development tools, project discipline will collapse and success will be left to chance. Do Configuration Management right, or forget about improving your development process.

From: “Little book of Configuration Management”, Software Programmers Network, 1998

2.4 Configuration Management

CM permits to version documents and source code. With CM it is possible to define a configuration or version and restore it any time, for example to search for bugs or revert changes.

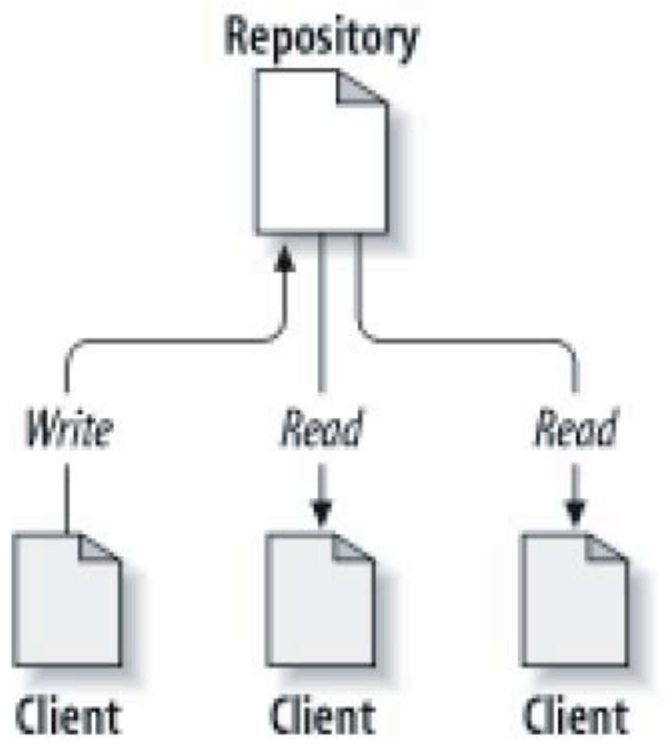
A single configuration is marked by a "tag", e.g., "A1.1".

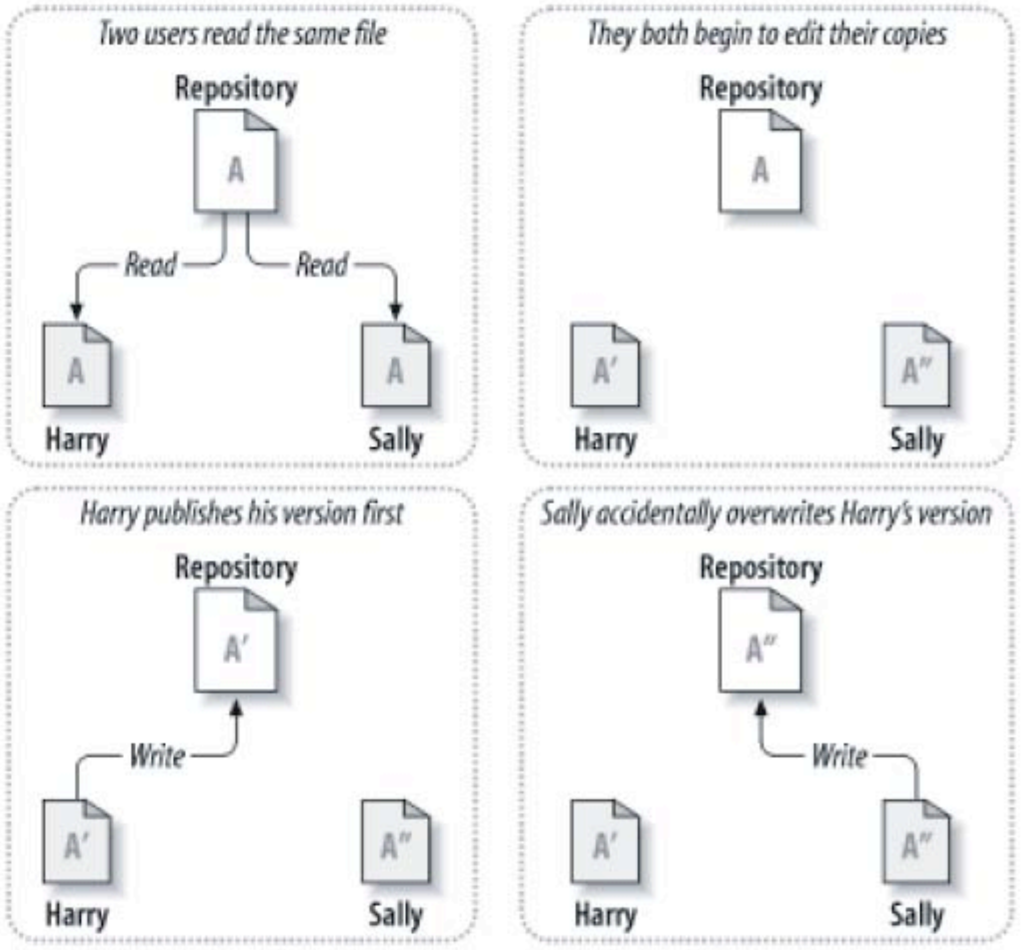

Integrations- Branch A für Anforderungen, Abnahmekriterien und Lösungskonzept
Entwicklungs-Branch E1 für Anforderungen, Abnahmekriterien und Lösungskonzept
Entwicklungs-Branch E2 für Anforderungen, Abnahmekriterien und Lösungskonzept
A 1.0
Zu Integrations-Branch B
A 1.1
A 1.2

2.4 Configuration Management

A centralized CM system keeps all documents in a repository on a server.

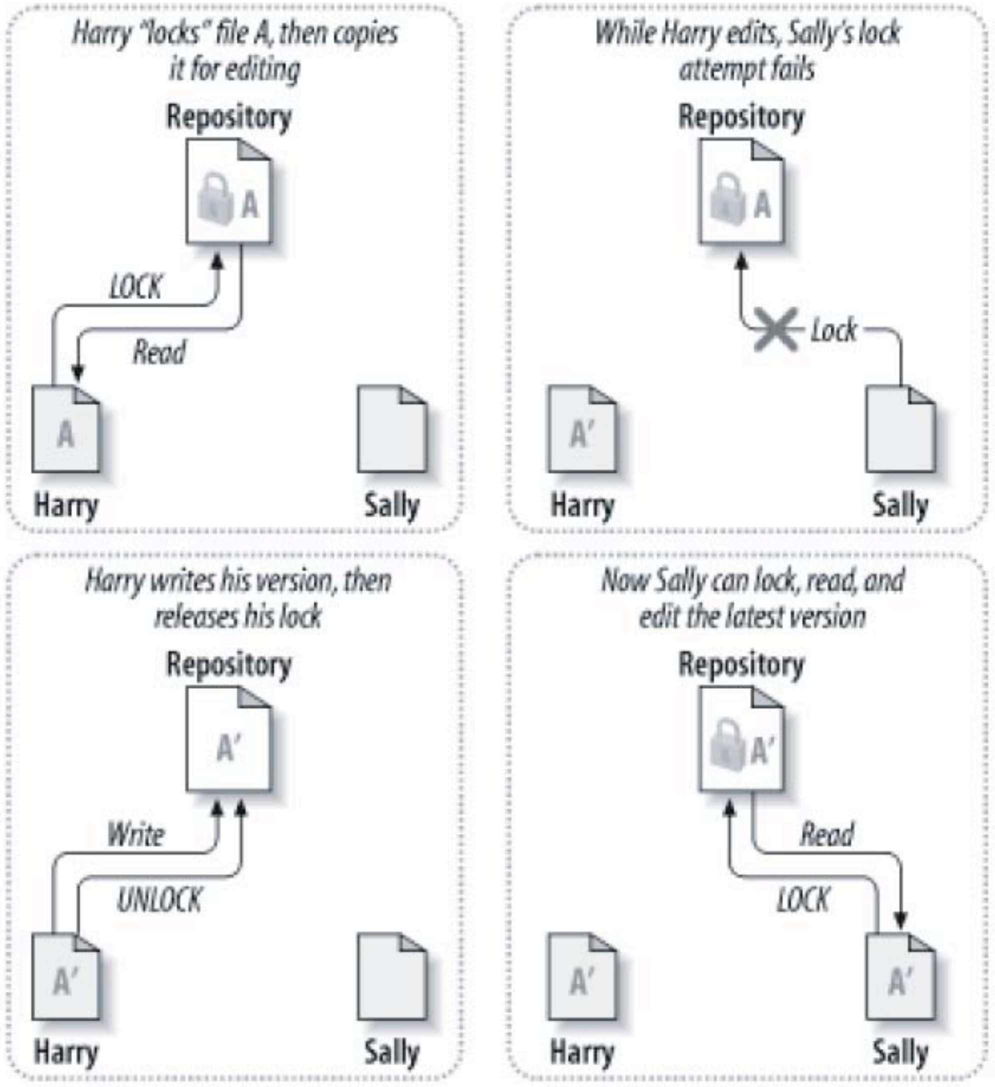
Consider the following scenario, which will lead to a problem:




Two users read the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A′
A″
Harry
Sally
Harry publishes his version first
Repository
A′
Write
A′
A″
Harry
Sally
Sally accidentally overwrites Harry's version
Repository
A″
Write
A′
A″
Harry
Sally

2.4 Configuration Management

Solution A: Lock-Modify-Unlock approach:

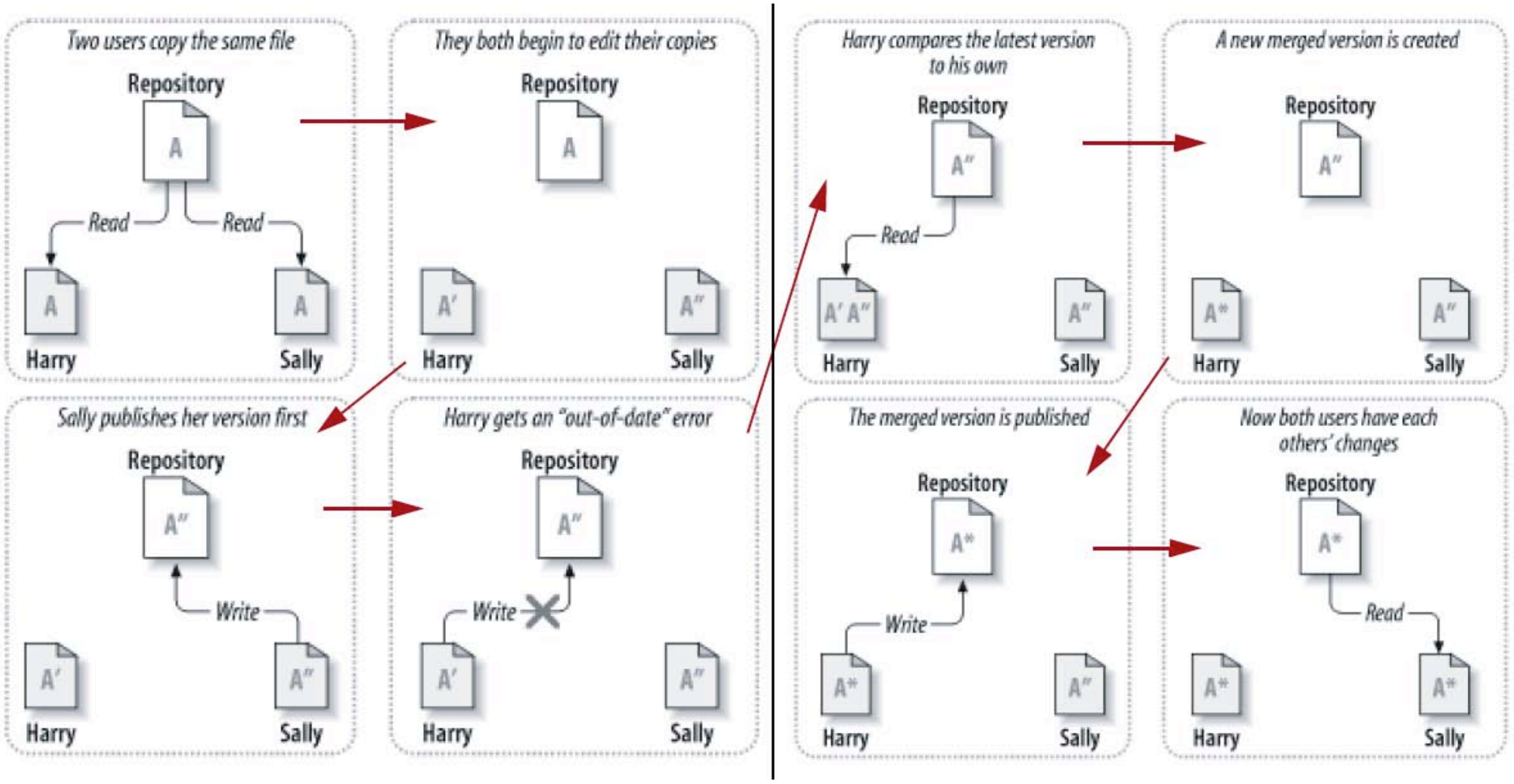

Harry "locks" file A, then copies it for editing
Repository
A
LOCK
Read
A
Harry
Sally
While Harry edits, Sally's lock attempt fails
Repository
A
Lock
A'
Harry
Sally
Harry writes his version, then releases his lock
Repository
A'
Write
UNLOCK
A'
Harry
Sally
Now Sally can lock, read, and edit the latest version
Repository
A'
Read
LOCK
A'
A'
Harry
Sally

Possible problems:

- Lock is forgotten about
- Unnecessary serialization; Harry wants to change the beginning of the file, Sally the end. Doesn't work.
- False sense of security; if files A and B depend on each other a lock of a single file doesn't help

2.4 Configuration Management

Solution B: Copy-Modify-Merge approach (used by Subversion):



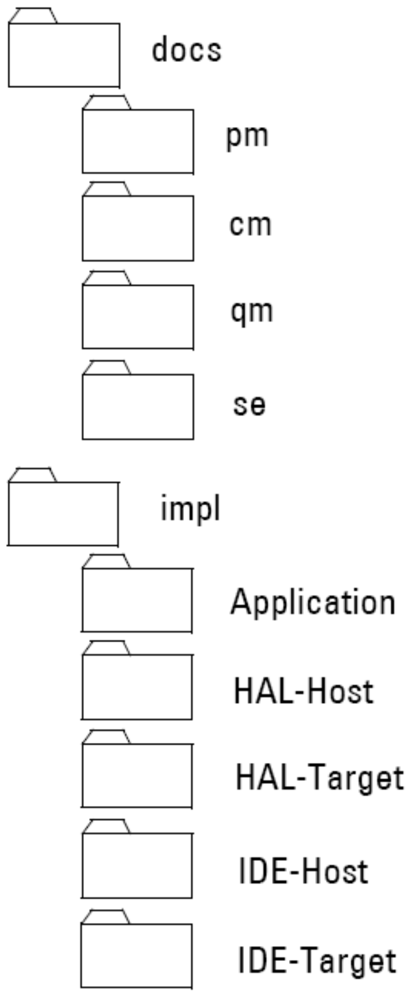
Two users copy the same file
Repository
A
Read
Read
A
Harry
A
Sally
They both begin to edit their copies
Repository
A
A′
Harry
A″
Sally
Sally publishes her version first
Repository
A″
Write
A′
Harry
A″
Sally
Harry gets an "out-of-date" error
Repository
A″
Write
A′
Harry
A″
Sally
Harry compares the latest version to his own
Repository
A″
Read
A′ A″
Harry
A″
Sally
A new merged version is created
Repository
A″
A*
Harry
A″
Sally
The merged version is published
Repository
A*
Write
A*
Harry
A″
Sally
Now both users have each others' changes
Repository
A*
Read
A*
Harry
A*
Sally

Problem occurs if data cannot be merges (e.g., binary data). Therefore Subversion also offers a lock function.

2.4 Configuration Management

Structure of development products

One of the most important aspects in configuration management is the definition of a folder structure. We are using the following scheme:



docs enthält alle Dokumente, die im Zusammenhang mit dem Projekt stehen

pm enthält alle Dokumente zum Projektmanagement.

cm enthält alle Dokumente zum Konfigurations- und Änderungsmanagement.

qa enthält alle Dokumente zum Qualitätsmanagement, z. B. Testpläne.

se enthält alle Dokumente im Zusammenhang mit der Systementwicklung, z. B. Anforderungsdokumente und Entwurfsdokumente.

impl enthält alle Produkte, die mit der Software zusammenhängen (Quellcode, ausführbare Dateien, IDE-Einstellungen).

Application enthält den Quellcode, der Zielssystem-unabhängig ist.

HAL-Host enthält den Code für die simulierte Hardware

HAL-Target enthält den hardwarenahen Code für die echte Hardware.

IDE-Host enthält die Hilfsdateien für die Host-Entwicklungsumgebung (z.B. Visual C++).

IDE-Target enthält die Hilfsdateien für die Ziel-Entwicklungsumgebung (Co-deWarrior).

2.4 Problem and Change Management

Once a project is in progress there will be problem reports (e.g., a document is incomplete, software doesn't work properly) and change requests (e.g., system should behave differently).

We call such reports "issues". They need to be registered so they are not forgotten.

In an analysis step it needs to be decided if these reports require new tasks to resolve them. For this, a change procedure is defined (who does the analysis, who decides what to do: change control board).

We manage all these issues with the tool Track+ (http://www.trackplus.com).

Issues get a priority (urgency).

Issues get a severity (important and urgent issue first, unimportant and non-urgent tasks never).

Issues have a status (e.g., "opened", "working on it", "closed").

Issues always have a manager and a responsible.

Changes in the source code have to be related to their issue number by adding the issue number to the Subversion commit message (e.g., #4711, #0815).

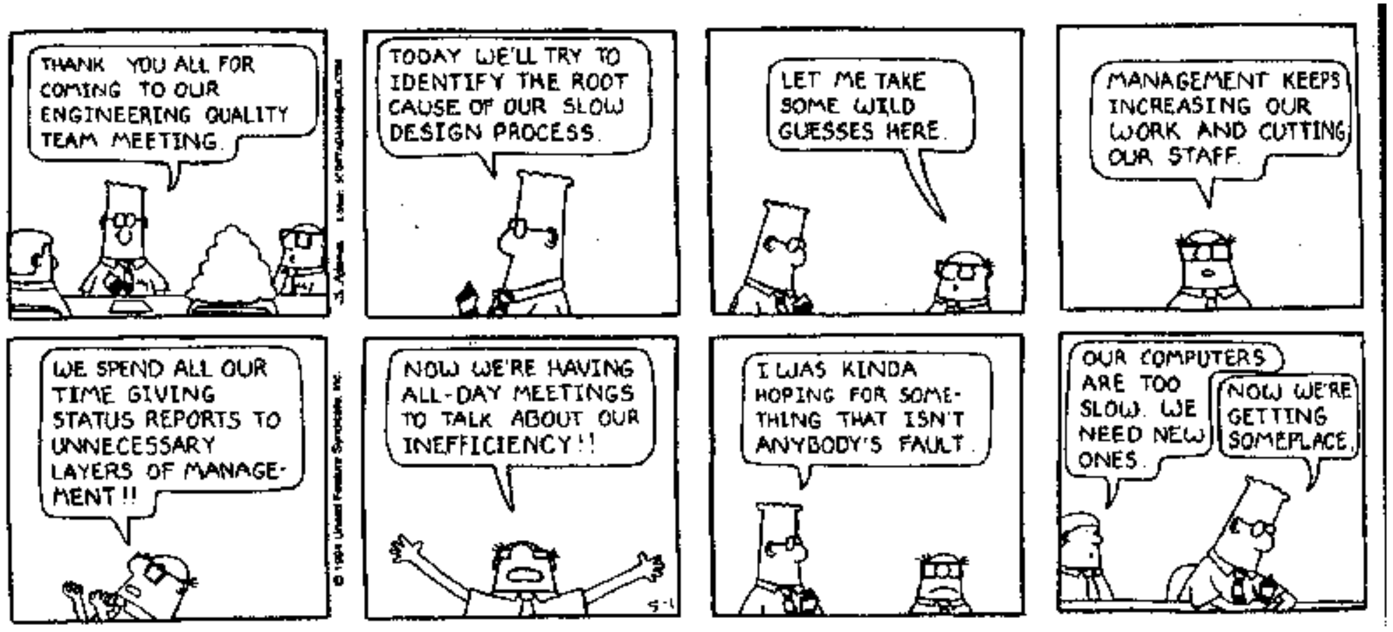
2.5 Quality Management

Quality management has a number of aspects. We will cover only some of them:

- Process quality; the project follows the processes described in its project manual
- Product quality; all documents are complete by form and content; all requirements are complete and correct; the system works according to customer expectation.

Quality is verified through assessments of processes and products.

- Process quality: in our project assessments by instructor. In real life assessments by external auditors, e.g., according to a reference model like CMMI or SPICE.
- Product quality: verification by inspections and test protocols.

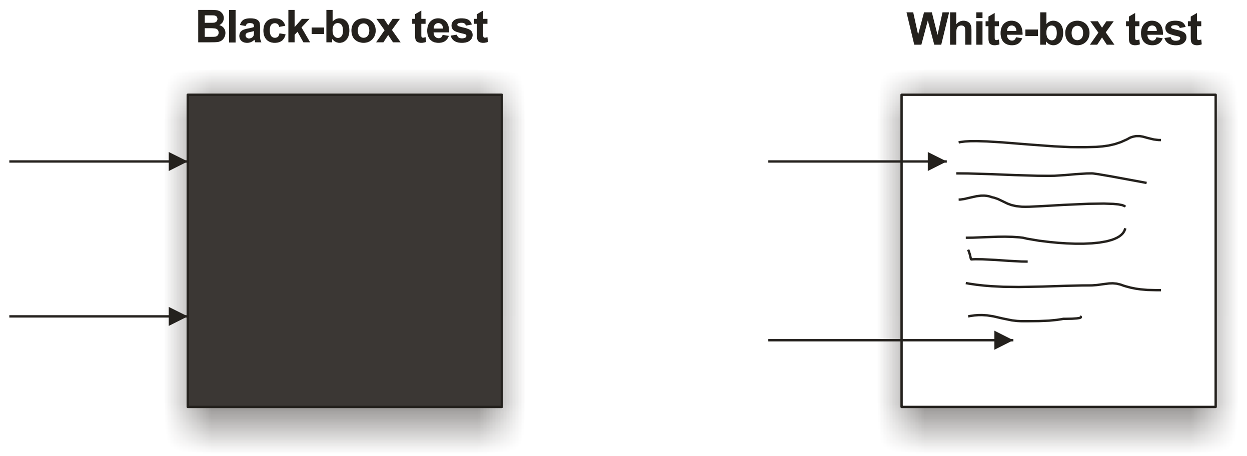

THANK YOU ALL FOR COMING TO OUR ENGINEERING QUALITY TEAM MEETING.
TODAY WE'LL TRY TO IDENTIFY THE ROOT CAUSE OF OUR SLOW DESIGN PROCESS.
LET ME TAKE SOME WILD GUESSES HERE.
MANAGEMENT KEEPS INCREASING OUR WORK AND CUTTING OUR STAFF.
WE SPEND ALL OUR TIME GIVING STATUS REPORTS TO UNNECESSARY LAYERS OF MANAGEMENT!!
NOW WE'RE HAVING ALL-DAY MEETINGS TO TALK ABOUT OUR INEFFICIENCY!!
I WAS KINDA HOPING FOR SOMETHING THAT ISN'T ANYBODY'S FAULT.
OUR COMPUTERS ARE TOO SLOW. WE NEED NEW ONES.
NOW WE'RE GETTING SOMEPLACE.

2.5 Quality Management

System functions should be as far as possible assessable through automated tests (e.g., HIL, hardware in the loop).

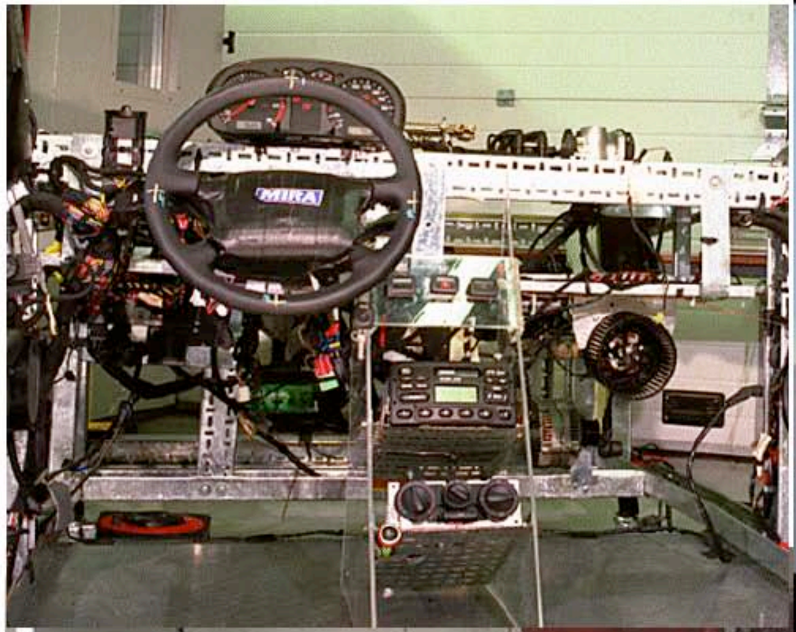
Malfunctions need to be registered with problem and change management during and after the integration step (hardware plus software, or software component A with software component B).

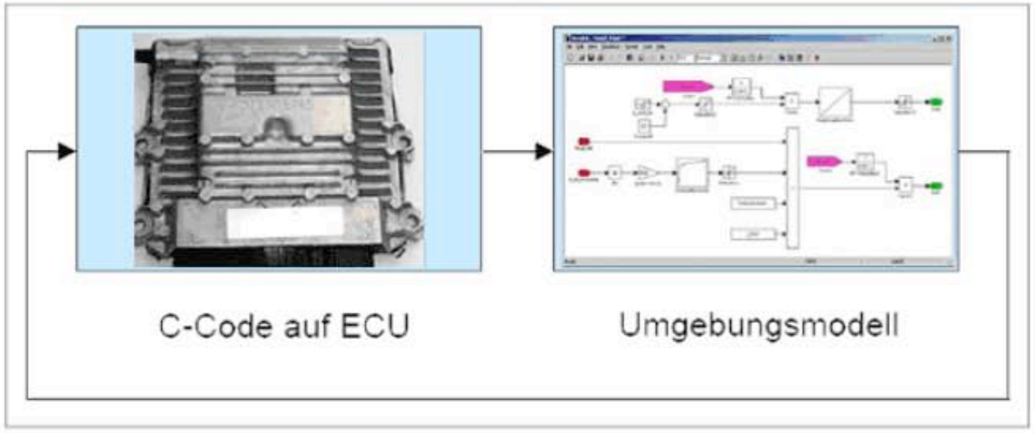
We distinguish between black-box tests and white-box tests.



Example for HIL (see next slide):

2.5 Quality Management





2.6 System Development

System development consists of

- Requirements engineering; the acquisition, structuring, documentation, validation, and management of requirements
- Design; the description of a solution meeting the requirements
- Implementation; the realization of the solution components described in the design
- Integration; the integration of the different solution components into a complete solution
- Test; the verification (against the requirements and design) and validation (against customer expectation) of the solution. This part can also be attributed to QM.

In this lecture, we will focus on requirements engineering, design, and implementation.

Model Driven Engineering

Higher productivity in the development of software intensive systems can be gained

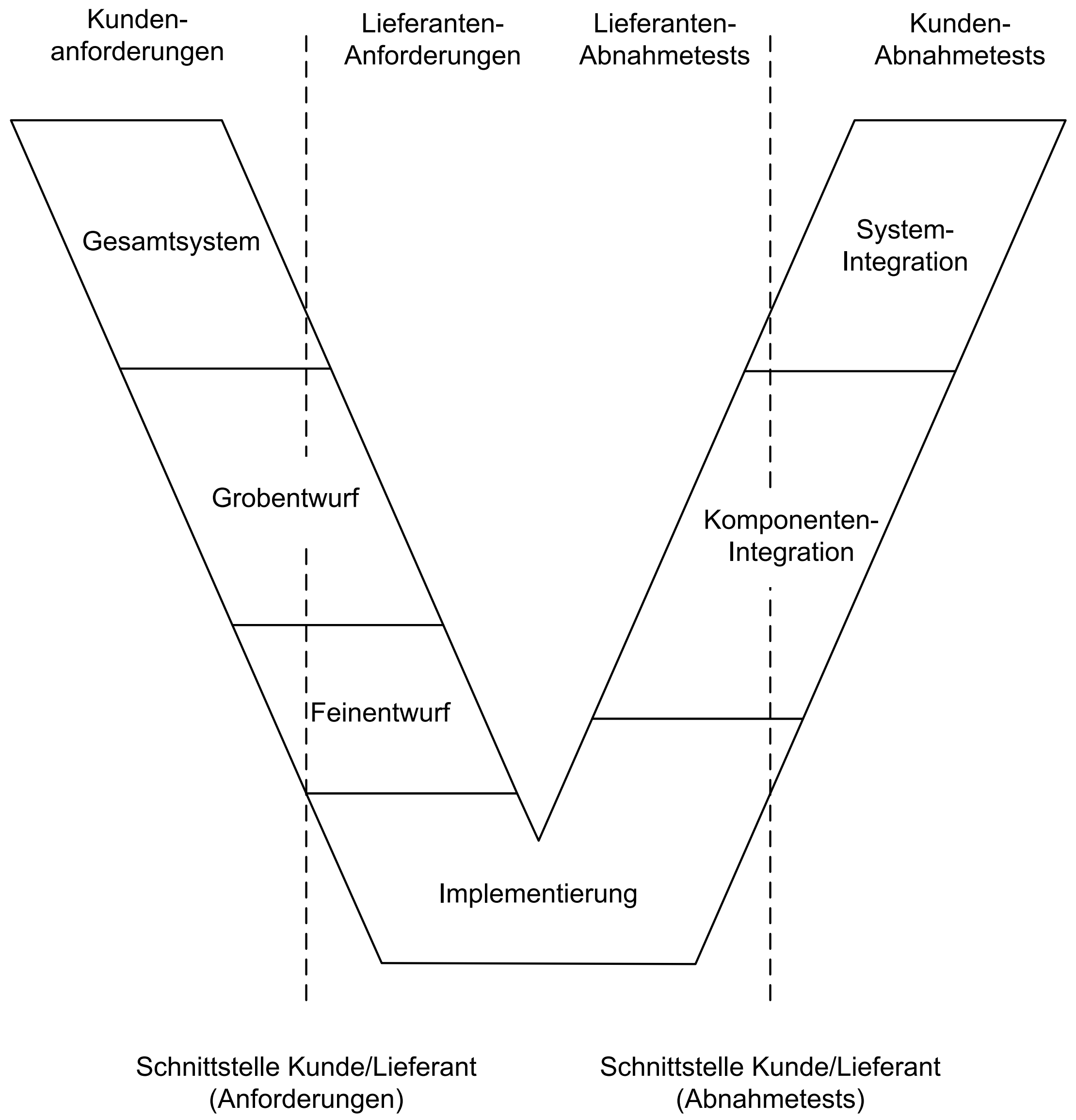
- Reuse of existing solutions or solution components (e.g., product lines)
- Higher level of abstraction in the description of the solution

Model driven engineering (MDE) utilizes both approaches. Solutions are described by models, for example in UML or domain specific description forms. Code can be generated from these models, ranging from skeletons to complete products.

Sample of MDE tools for real-time systems:

- Matlab/Simulink
- ASCET from ETAS

2.6 System Development

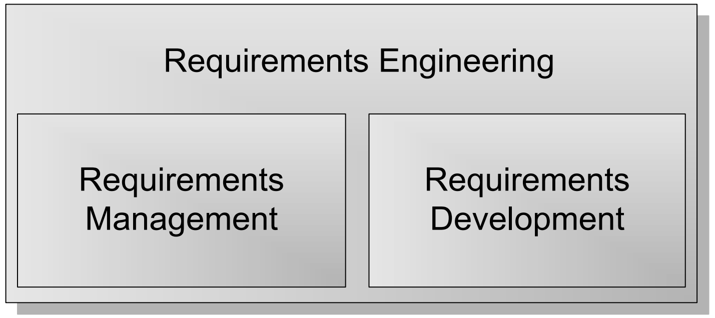


2.7 Requirements Engineering for Real-Time Systems

Requirement engineering is concerned with determining the goals, functions, and constraints of systems and the representation of these aspects, e.g., in models.

The goal of the requirement engineering process is to create complete, correct, and understandable (for both, the customer and developer) requirement specifications.

According to the CMMI, requirements engineering can be structured into requirements management and requirements development.



Requirements management means:

- Managing all changes to requirements
- Maintaining the relationship between requirements, the project plans, and the work products
- Identifying inconsistencies among requirements, the project plans, and the work products
- Taking corrective action

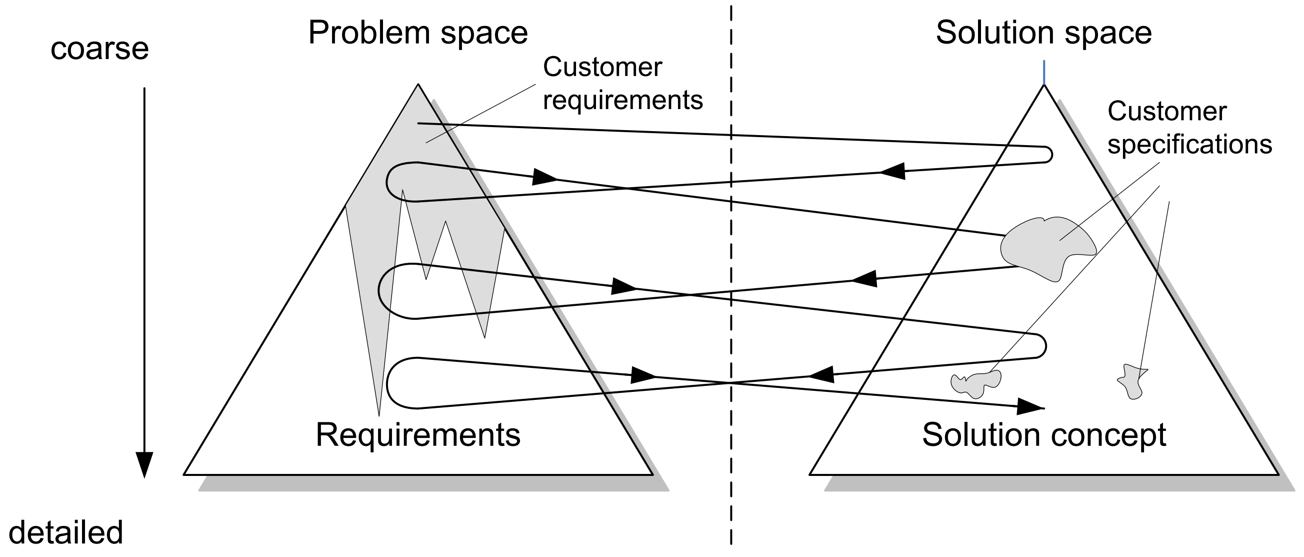
2.7 Requirements Engineering

Requirements development means:

- Collecting stakeholder needs, expectations, constraints, and interfaces
- Translating these artifacts into requirement specifications

In real-time systems, the requirement engineering process is characterized by:

- A lot of requirement changes during the development process
- Different levels of detail
- Separation between problem space and solution space due to company IP protection
- High complexity
- Reuse of existing solutions
- Adaptation to different customer processes



Requirements, Solution Concept, Solution

There are basically two schools in requirement engineering:

- Requirements as a specification of the solution (e.g., IEEE STD 830)
- Requirements as a pure description of the problem to be solved, without any consideration of a solution (the European RE purist's view)

Both approaches have their difficulties when it comes to praxis:

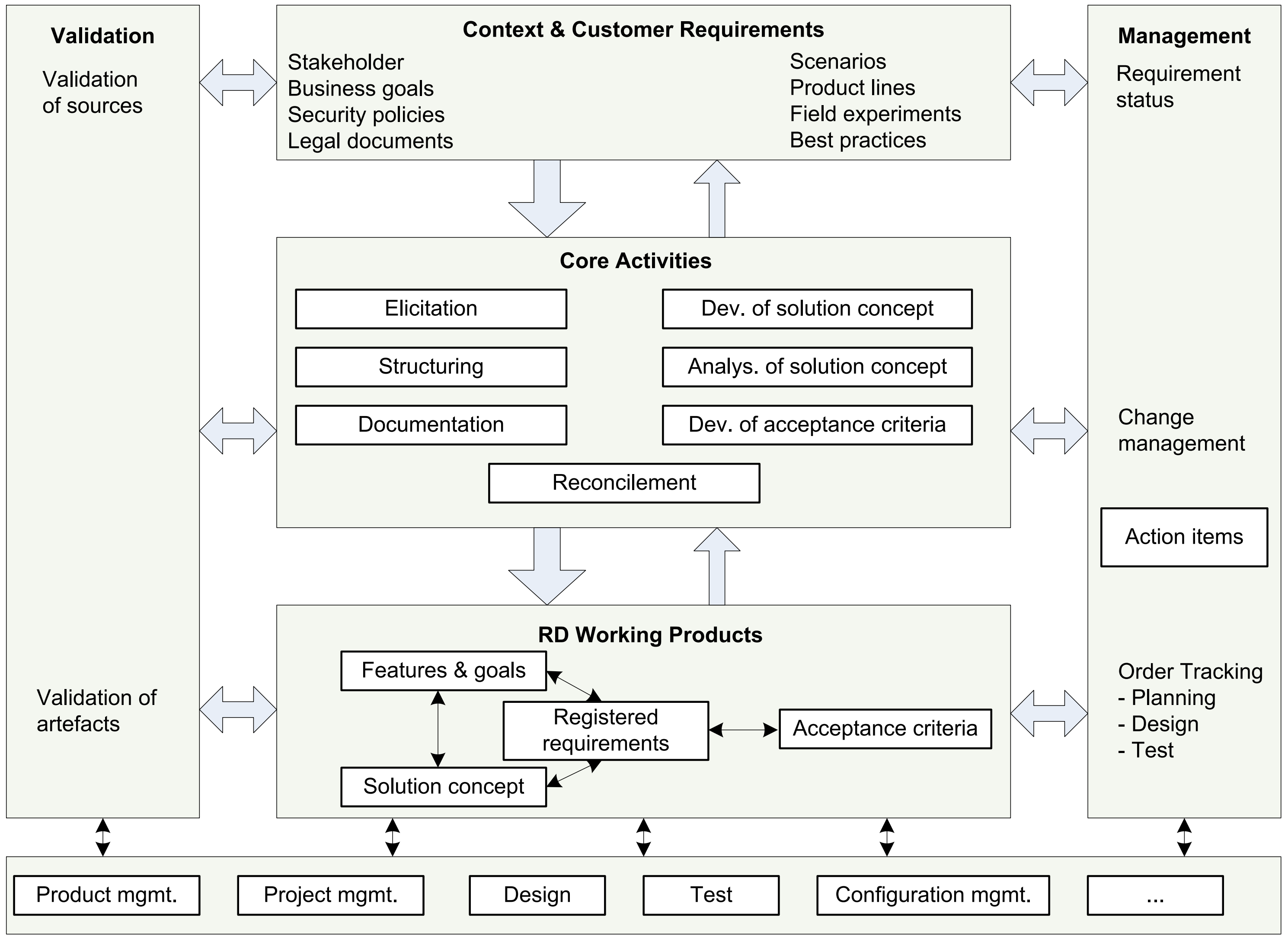
- Looking at a solution too early possibly limits the solution space and might distract from the essential requirements
- Not considering the solution space can prevent finding important requirements that can only be found by looking at a solution model.
  Oftentimes there is already a solution for a similar problem (in most cases there is no green field development)
- Oftentimes there are interdependencies between requirements and solution

We therefore propagate the inclusion of a solution concept into the RE process.

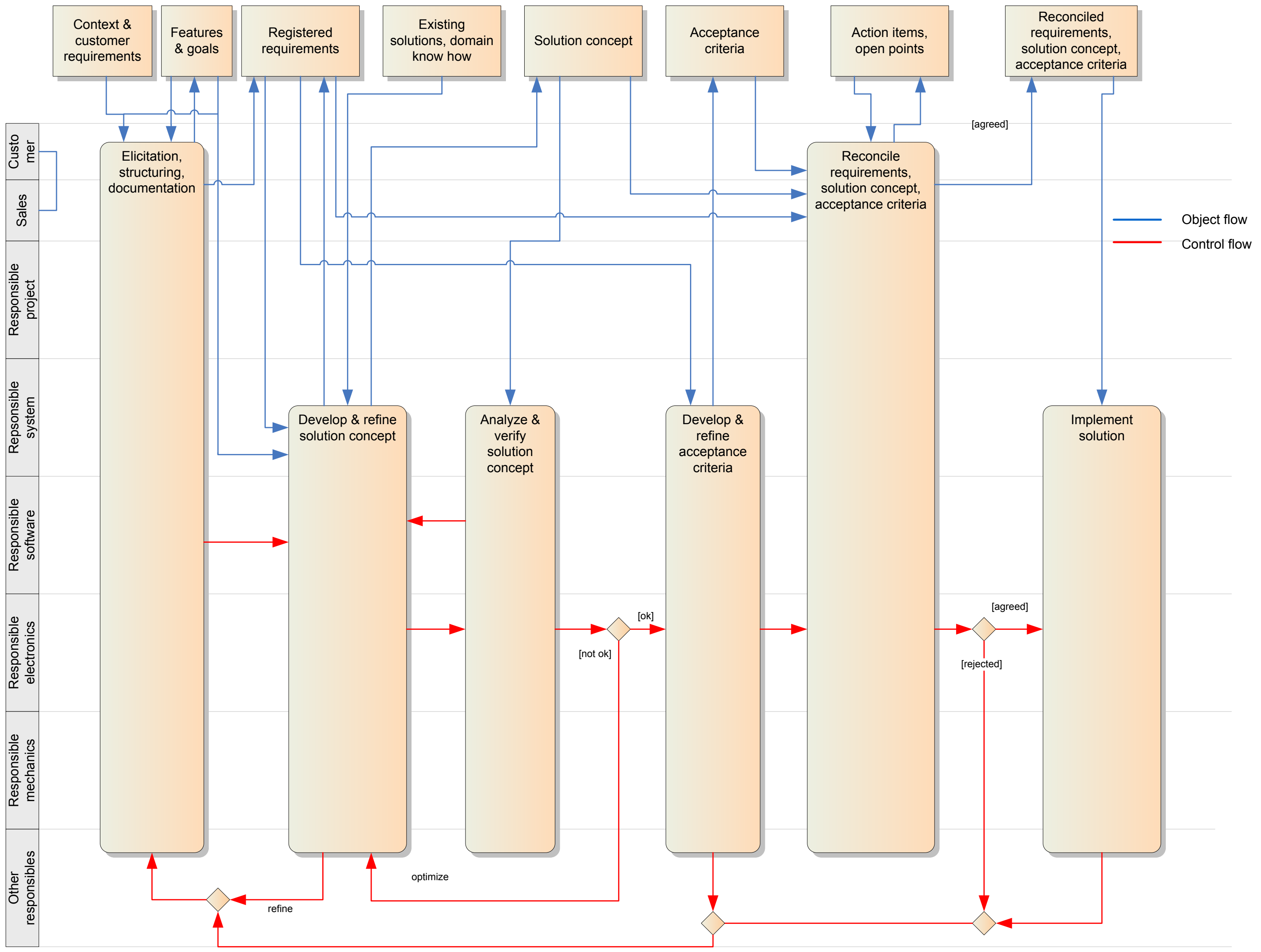
The solution concept must not be mixed up with the solution itself! It serves purely for the elicitation and illustration of the requirements and as a model for a possible solution. It must be as far as possible free from implementation details.

2.7 Requirements Engineering

RE Framework



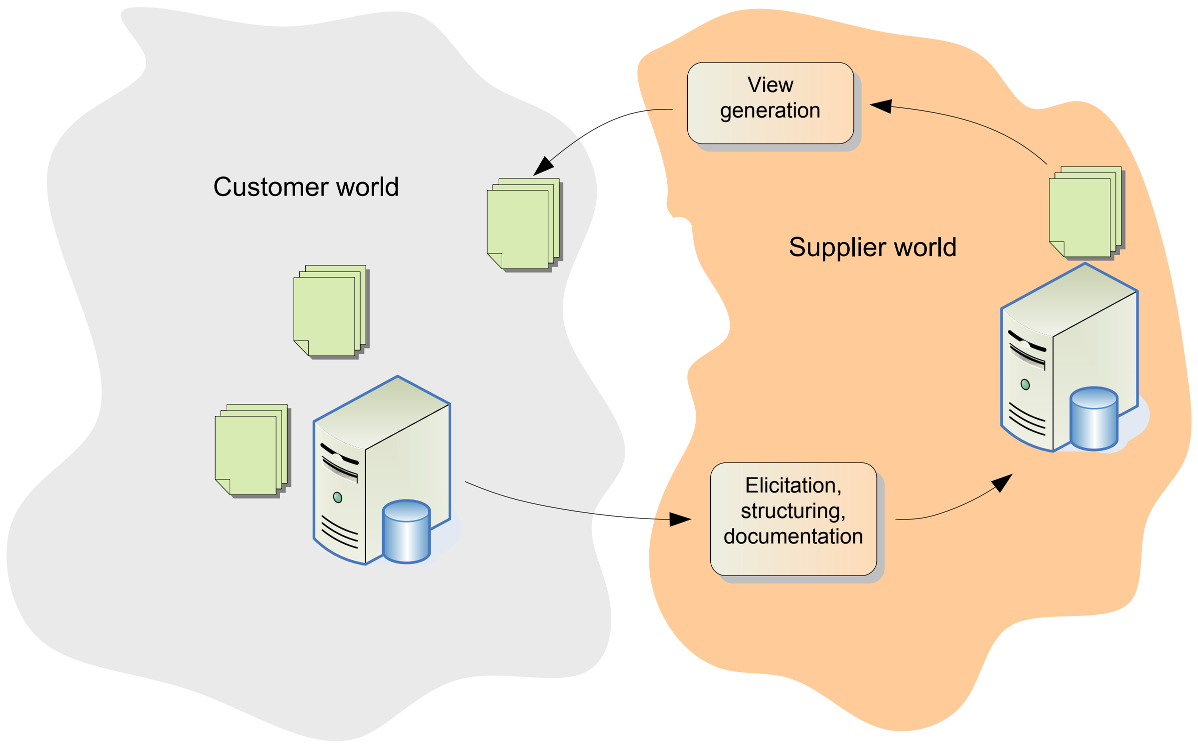
Sample RE Workflow


Context & customer requirements
Features & goals
Registered requirements
Existing solutions, domain know how
Solution concept
Acceptance criteria
Action items, open points
Reconciled requirements, solution concept, acceptance criteria
Customer
Sales
Responsible project
Repsonsible system
Responsible software
Responsible electronics
Responsible mechanics
Other responsibles
Elicitation, structuring, documentation
Develop & refine solution concept
Analyze & verify solution concept
Develop & refine acceptance criteria
Reconcile requirements, solution concept, acceptance criteria
Implement solution
[agreed]
Object flow
Control flow
[ok]
[not ok]
[agreed]
[rejected]
optimize
refine

2.7 Requirements Engineering

RE Customer-Supplier Interface

In the development of RT systems, customers and suppliers have their own development process worlds. Requirements need to be shared across the customer-supplier process boundary.

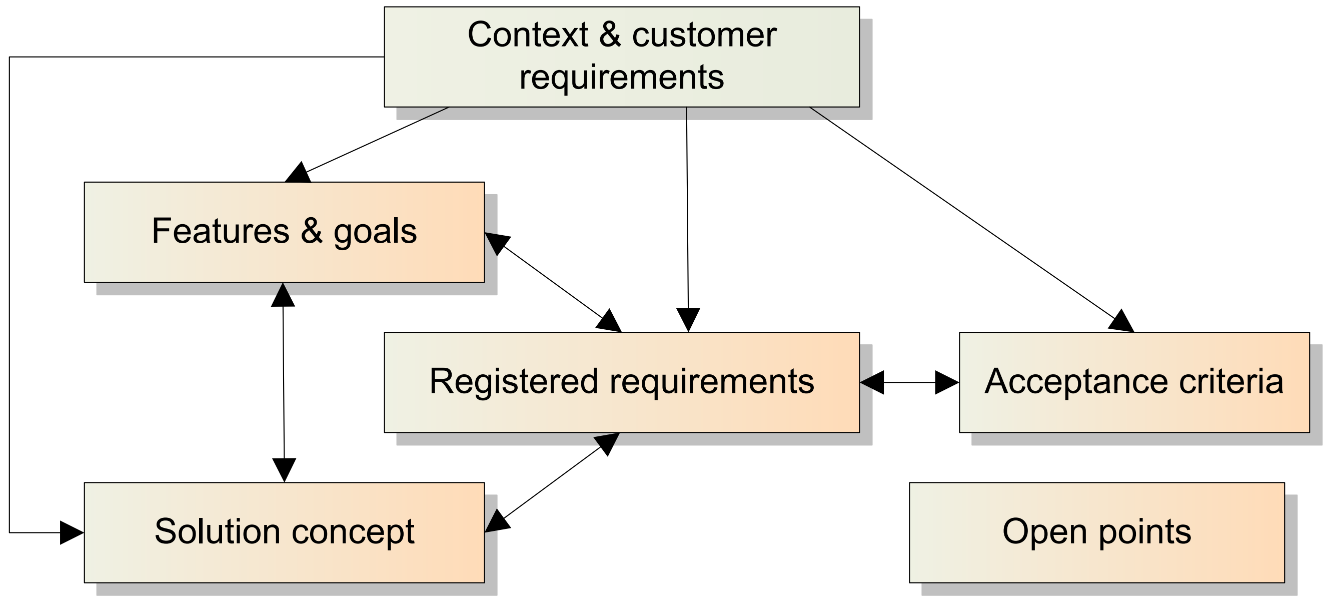


There are tools which permit to share requirements across such boundaries (e.g., DOORS)

RE Artifacts

2.7 Requirements Engineering

The documents created as a result of RD activities we call "artifacts". We can categorize RD artifacts in groups.

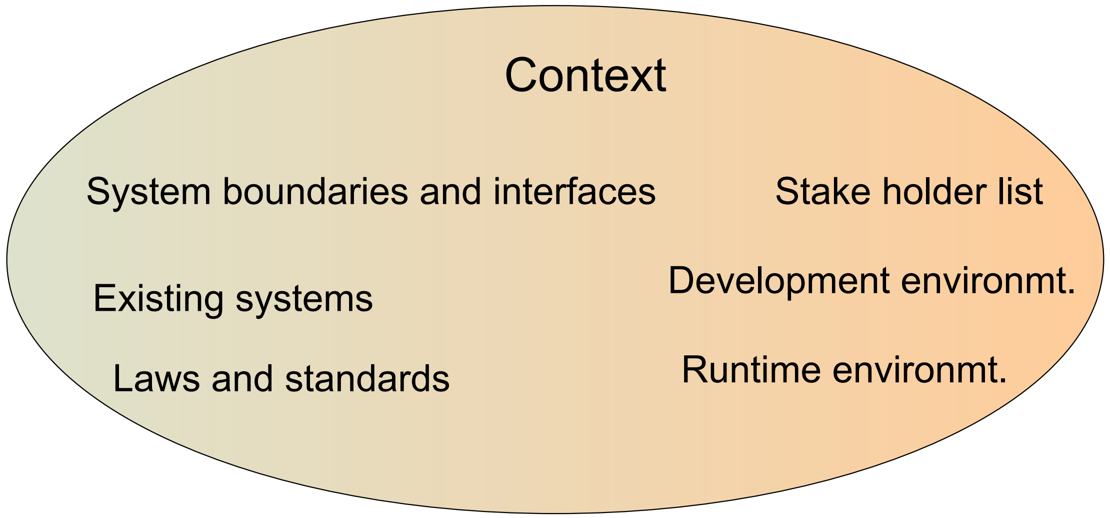


2.7 Requirements Engineering

Context:

The context defines the boundary conditions for the development of a product.

The context is typically pre-defined and not negotiable.



The stakeholder list contains all persons and organizations that have a direct or indirect interest in the development results, e.g., customer representatives, product management, sales.

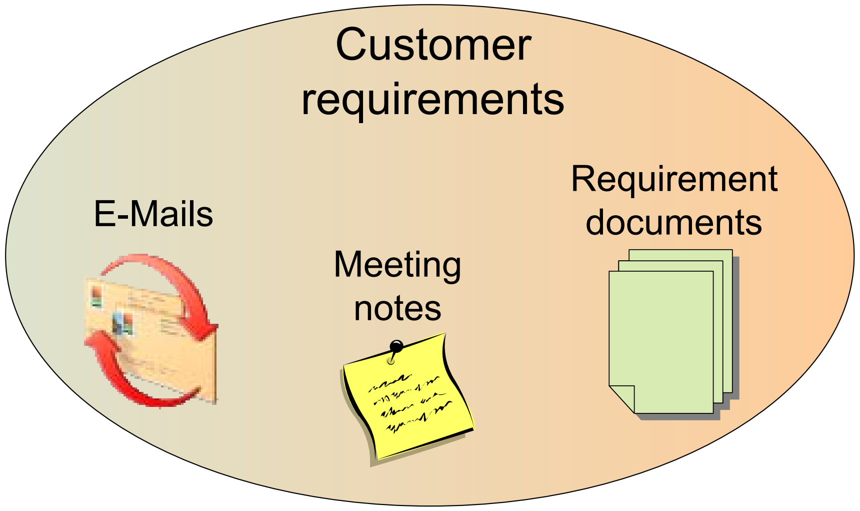
The boundary conditions list existing systems, standards, laws, and regulations, development environment, and runtime environment.

2.7 Requirements Engineering

Customer requirements:

The customer requirements are the central artifacts in requirements engineering. The supplier typically has little control in what way the customers conveys his requests. Thus, any format needs to be acceptable (e-mails, meeting notes, documents, databases, etc.).

The customer requirements collect anything the customer demands.



Customer requirements thus have a similar quality as the context, but they can be more vola-tile.

Customer requirements can contain goals, pure requirements, parts of a solution concept or even solution, and acceptance criteria.

2.7 Requirements Engineering

Features and goals:

Features and goals define the intentions of the stakeholders.

Features form a good basis for the elicitation of requirements.

Goals are helpful to describe the purpose of the system understandable for all parties in-volved.

Goals are typically more stable than requirements, since they describe the purpose of a sys-tem on a higher level of abstraction.

Goals can be defined by the customer, or can be developed jointly between customer and supplier.

Features offer a helpful overview over the relevant properties of a system. They are well suit-ed for communication with sales and customer.

2.7 Requirements Engineering

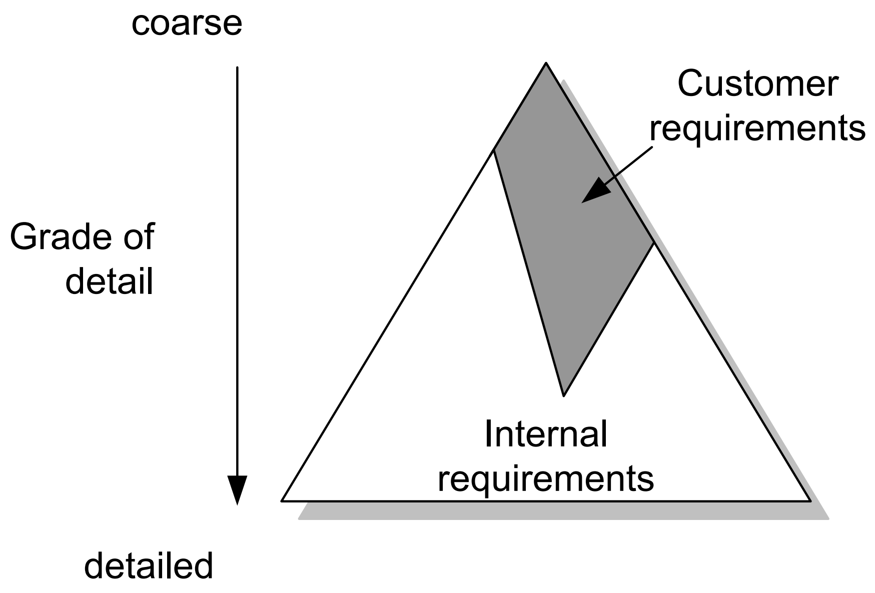
Registered requirements:

The registered requirements collect, structure, and document all customer requests in a form suitable for further development by the supplier.

The registered requirements are the supplier internal representation of all relevant requirements to a system. They can contain customer requirements as well as supplier specific internal requirements.

The registered requirements serve to isolate the customer's development process from the supplier's development process.

The registered requirements can exist on several levels of detail.



2.7 Requirements Engineering

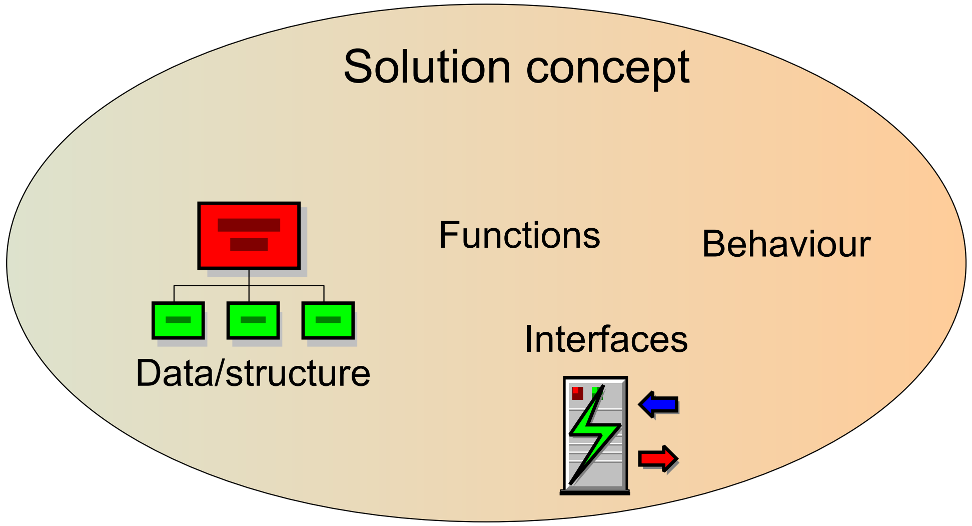
They can be organized such that they consist of a collection of references to existing requirements or solutions and additional change requests towards these solutions. This is particularly helpful where many similar solutions need to be developed (e.g., in case of a product line approach).

The registered requirements can be categorized, for example in customer driven requirements (CMMI: customer requirements) and internal requirements (CMMI: product and product component requirements). In German speaking countries there is a distinction between "Lastenheft" and "Pflichtenheft" requirements.

Solution concept:

The solution concept describes possible solutions at a significant higher abstraction level than the solution itself. Example: paper model of a house, and the house itself.

The solution concept can be structured into a description of data and structure, functions, behavior, and interfaces.



Typical forms of description for the hardware are block diagrams, tables with input and output connections, blue prints, wiring diagrams, VHDL descriptions, and prototypes.

Typical forms of description for the system and software are block diagrams, textual descriptions, state diagrams, activity diagrams, simulation models, and graphical representations.

The solution concept can be expressed as a modification of an existing solution.

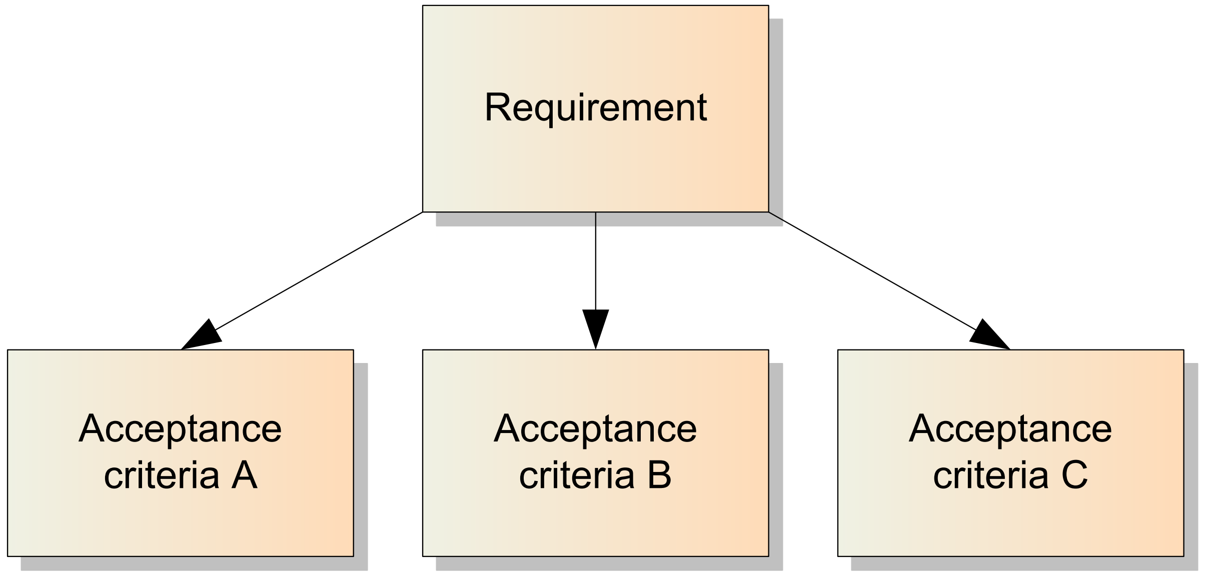
2.7 Requirements Engineering

Acceptance criteria:

Acceptance criteria document what the customer expects from the system before he accepts it as satisfactory.

Acceptance criteria shall help to assess whether the system meets all requirements.

For a single requirement, there can be more than one acceptance criteria.



Open points list:

The Open points list documents at each point in time questions to be cleared. Entries in this list can be linked with other artifacts when they are related to them.

Points to Remember