

# Course "Softwaretechnik"

## Book Chapter 4

# Requirements Elicitation (Anforderungserhebung)

Lutz Prechelt, Steve Easterbrook

Freie Universität Berlin, Institut für Informatik  
<http://www.inf.fu-berlin.de/inst/ag-se/>

- Requirements and Requirements Engineering
- Kinds of requirements
- Requirements and modeling
- Hard and soft systems
- Requirements Elicitation
  - identify problem & opportunity
- Elicitation techniques
  - Conventional
  - Representation-based
  - Social
  - Knowledge elicitation

# Where are we?: Taxonomie "Die Welt der Softwaretechnik"

## Welt der Problemstellungen:

- **Produkt (Komplexitätsprob.)**
  - **Anforderungen (Problemraum)**
  - Entwurf (Lösungsraum)
- Prozess (psycho-soziale P.)
  - Kognitive Beschränkungen
  - Mängel der Urteilskraft
  - Kommunikation, Koordination
  - Gruppendynamik
  - Verborgene Ziele
  - Fehler

## Welt der Lösungsansätze:

- Technische Ansätze ("hart")
  - Abstraktion
  - Wiederverwendung
  - Automatisierung
- **Methodische Ansätze ("weich")**
  - **Anforderungsermittlung**
  - Entwurf
  - Qualitätssicherung
  - Projektmanagement

# Where are we?: Anforderungsermittlung

- Einsicht: Man darf sich nicht auf intuitiven Eindruck darüber verlassen, was gebaut werden sollte
  - sondern sollte die Anforderungen systematisch ermitteln
- Prinzipien:
  - **Erhebung** der Anforderungen bei allen Gruppen von Beteiligten
  - **Beschreibung** in einer Form, die die Beteiligten verstehen
  - **Validierung** anhand der verschriftlichten Form
  - **Spezifikation**: Übertragung in zur Weiterverarbeitung günstige Form
  - **Trennung von Belangen**: Anford. möglichst wenig koppeln
  - **Analyse auf Vollständigkeit**: Lücken aufdecken und schließen
  - **Analyse auf Konsistenz**: Widersprüche aufdecken und lösen
  - **Mediation**: Widersprüche, die auf Interessengegensätzen beruhen, einer Lösung zuführen (Kompromiss oder Win-Win)
  - **Verwaltung**: Übermäßige Anforderungsänderungen eindämmen, Anforderungsdokument immer aktuell halten

- What is a Requirement?
  - Something that someone needs in order to solve a problem or achieve an objective:
  - *"A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.  
The set of all requirements forms the basis for subsequent development of the system or system component". [IEEE Std]*
  - Note 1: Often, the "formally imposed document" does not exist, but there is still *somebody* wishing to be satisfied.
    - Informal requirements
  - Note 2: Often, what is written down in the "formally imposed document" will not really *satisfy* the system user
    - Invalid/incorrect requirements
  - Note 3: "System" can be a computer system (**system req's**) or a socio-technical system (**user requirements**)

# Types of Requirements

- Functional requirements:
  - What the system does: the interactions between the system and its environment; independent from implementation
- Nonfunctional requirements:
  - Observable aspects of the system that are not directly related to functional behavior
  - e.g. performance or reliability aspects, etc.
- Safety/security requirements ("shall not" properties)
  - A kind of nonfunctional requirement: Behavior the system must never exhibit
  - e.g. "must be impossible to apply reverse thrust in mid-flight"
- Constraints ("Pseudo requirements"):
  - Imposed by the client or environment in which the system operates
  - Often concern the technology to be used (language, operating system, middleware etc.)

nonfunctional requirements

# Definitions: Requirements Engineering (RE)

- Requirements Elicitation is part of Requirements Engineering
- Requirements Engineering (RE):

"[...] **Requirements Engineering** is the **branch of systems engineering** concerned with **real-world goals** for, services provided by, and constraints on software systems. Requirements Engineering is also concerned with the relationship of these factors to **precise specifications** of system behaviour and to their **evolution** over time and across system families..."  
[Zave94]

"[...] RE is concerned with **identifying the purpose** of a software system, and **the contexts** in which it will be used." [RE'01 CfP]

## 1. Understand the problem

- Requirements Elicitation
- understand the context and the goals in the user's terms

## 2. Formally describe the problem

- Requirements Specification
- describe what the SW must do to reach the goals

## 3. Attain agreement on the problem

- Requirements Validation
- find gaps, mistakes, and inconsistencies in the requirements
- includes conflict resolution, negotiation

## 4. Maintain the agreement

- Requirements Management
- negotiate and decide on changes of the specification

# Conflict is natural and ubiquitous

- Even the most cooperative stakeholders ("Beteiligte") will inevitably have conflicts
- Conflict resolution is a core activity of RE





# Requirements Validation

- Even without conflict, requirements validation is a critical step in the development process
  - after requirements engineering or requirements analysis
  - perhaps again at delivery (during client acceptance test)
- Requirements validation criteria:
  - Correctness:
    - The requirements accurately represent the client's view.
  - Completeness:
    - All possible scenarios in which the system can be used are described, including exceptional behavior by the user or the system
  - Consistency:
    - No functional or nonfunctional requirements contradict one another
  - Feasibility/Realism:
    - Requirements can realistically be implemented and delivered
  - Traceability:
    - It will be possible to trace each system function to a corresponding (set of) functional requirement(s)

- Problem with requirements validation:  
Requirements change during and after elicitation
- Tool support is needed for managing requirements:
  - Store requirements in a shared repository
  - Provide multi-user access
  - Automatically create a system specification document from the repository
  - Allow change management
  - Provide traceability throughout the project lifecycle
- e.g. IBM Rational DOORS
- e.g. an appropriate Wiki tool (for smaller projects)



- **Domain Properties** are properties in the problem domain that are true whether or not we ever build the proposed system
- **Requirements** are properties in the problem domain that we wish to be made true by delivering the proposed system
- **A specification** is a description of the behaviors of the program in the solution domain that the program must have in order to meet the requirements
  - The system specification (system requirements), not to be confused with a statement of the requirements themselves, the requirements specification (user requirements)

# Validation vs. Verification

- Verification checks the equivalence of different formal representations
- Validation checks if a system fulfills the actual expectations in the real world
- **Verification** criteria:
  - Does the **Program** running on a particular **Computer** satisfy the **Specification**?
  - Does the **Specification**, in the context of the given **Domain properties**, satisfy the *stated* **Requirements**?
- **Validation** also checks:
  - Did we understand all the important **Requirements**?
  - Did we understand all the relevant **Domain properties**?

# Validation example

- Requirement R:
  - "Reverse thrust shall only be enabled when the aircraft is moving on the runway"
- Domain Properties D:
  - Wheel pulses are on if and only if wheels are turning
  - Wheels are turning if and only if aircraft is moving on runway
- Specification S:
  - Reverse thrust is enabled if and only if wheel pulses are on
- $S + D$  imply R
  - But what if the domain model D is wrong?

(Do you recognize the example?)

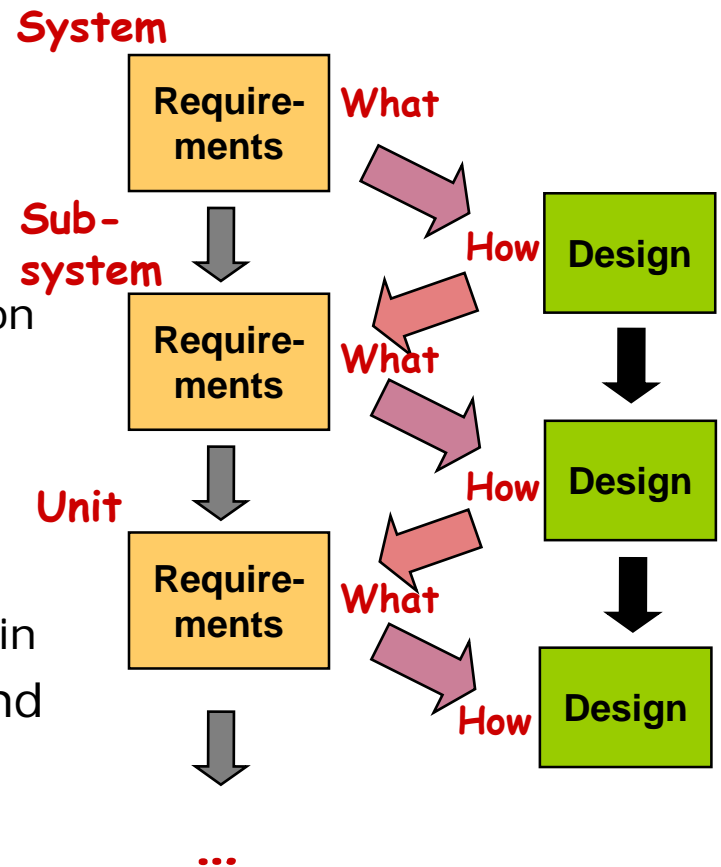


# Another validation example

- Requirement R:
  - "The database shall only be accessible by authorized personnel"
- Domain Properties D:
  - Authorized personnel have passwords
  - Non-authorized personnel do not have passwords
- Specification S:
  - Access to the database shall only be granted after the user types an authorized password
- $S + D$  imply R
  - But what if the domain assumptions are wrong?

# What vs. How

- Traditionally, Requirements should specify 'what' without specifying 'how'
  - But this is not always easy to distinguish:
    - What does a car do vs. a bike?
      - (Don't mention the motor: 'how!')
      - The 'how' at one level of abstraction forms the 'what' for the next level
- A suitable distinction
  - 'What' refers to a system's purpose
    - it is **external** to the system
    - it is a property of the application domain
  - 'How' refers to a system's structure and behavior
    - it is **internal** to the system
    - it is a property of the solution domain



# What is a **System**?

## Definition of a System:

- Some part of reality that can be observed to interact with its environment
  - Separated from its environment by a **boundary**
    - Boundary may be difficult to decide: **"soft" system**
  - A system receives **inputs** from the environment and sends **outputs** to the environment
  - Many systems have a **control mechanism**
  - Most systems have interesting **emergent properties**
- Examples:
  - cars, cities, houseplants, rocks, spacecraft, buildings, weather, ...
  - operating systems, DBMS, The Sims, the Internet
- Non-examples (there aren't many!):
  - numbers, truth values, letters



# Most systems are "soft"

- The software we will eventually write is not "the system" with respect to requirements engineering
  - The software is the system only in the solution domain
  - but not in the problem domain
- Rather, other things are also part of the system in the problem domain:
  - the people using the software,
  - the ways in which they use it,
  - many other environmental factors
- This larger system we need to understand during requirements elicitation
  - Rule of thumb: If people are involved in any way, never confuse the software with the system
  - Remember "Auswirkungen d. Informatik"?:  
underground train with taped-down "GO" button?
    - Very simple software, but a surprising system



- Starting point: Some notion that there is a "problem" that needs solving
  - e.g. dissatisfaction with the current state of affairs
  - e.g. a new business opportunity
  - e.g. a potential saving of cost, time, resource usage, etc.
- The requirements engineer must:
  - become enough of an expert in the problem domain to
  - identify the problem and opportunity and
  - elicit enough knowledge to analyze requirements for
  - validity, consistency, and completeness

## W6H

The journalist's technique:

What?

Where?

Who?

Why?

When?

How?

(Which?)

# Identifying the problem and opportunity

- Which problem needs to be solved?
  - identify problem **Boundaries**
- Where is the problem?
  - understand the **Context/Problem Domain**
- Whose problem is it?
  - identify **Stakeholders (Betroffene, Beteiligte)**
- Why does it need solving?
  - identify the stakeholders' **Goals**
- How might a software system help?
  - collect some **Scenarios**
- When and how does it need solving?
  - identify **Development Constraints**
- What might prevent us solving it?
  - identify **Feasibility** and **Risk**

Very  
useful slide

## W6H

The journalist's  
technique:

What?

Where?

Who?

Why?

When?

How?

(Which?)

# Difficulties of Elicitation (1)

- Limited observability
  - The problem owners might be too busy solving it in its current form
  - Presence of an observer may change the problem
- Bias
  - People may not be free to tell you what you need to know
    - Political climate & organizational factors
  - People may not want to tell you what you need to know
    - The outcome will affect them, so they may try to influence you (hidden agendas)
- There will be conflicts between different sources
  - People have conflicting goals
  - People have different understandings
- Thin spread of domain knowledge
  - It might be distributed across many sources
  - Is rarely available in explicit form

# Difficulties of Elicitation (2): Tacit knowledge

- Tacit knowledge  
(The "say-do" problem)  
Three stage model of learning:
  - 1) cognitive – verbal rehearsal of tasks
  - 2) associative – with repetition, verbal mediation disappears
  - 3) autonomous – no conscious awareness of performance.

➔ Experts are not aware of what they know and cannot introspect reliably
- Representational Problems
  - Experts don't have the language to describe their knowledge
    - Spoken language lacks necessary precision
    - Knowledge Engineer and Expert must work together to create a suitable language and representation formalism
  - Different knowledge representations are good for different things
- Brittleness
  - Knowledge is created, not extracted: incomplete, overly simplified

# Difficulties of Elicitation (3): Distortions

## Sender-related:

- Social pressure
  - Response to verbal and non-verbal cues from an interviewer
- Group think
  - Response to reactions of other experts
- Impression management
  - Response to imagined reactions of managers, clients, etc.
- Wishful thinking
  - Response to hopes
- Availability
  - Some data are easier to recall than others
- Underestimation of uncertainty
  - Tendency to underestimate by a factor of 2 or 3

## Receiver-related:

- Misinterpretation
  - due to lack of knowledge
- Misrepresentation
  - Expert cannot accurately fit a response into the requested response mode
- Anchoring
  - Contradictory data is ignored once an initial solution is available

## Sender- and receiver-related:

- Inconsistency
  - Statements made earlier are forgotten

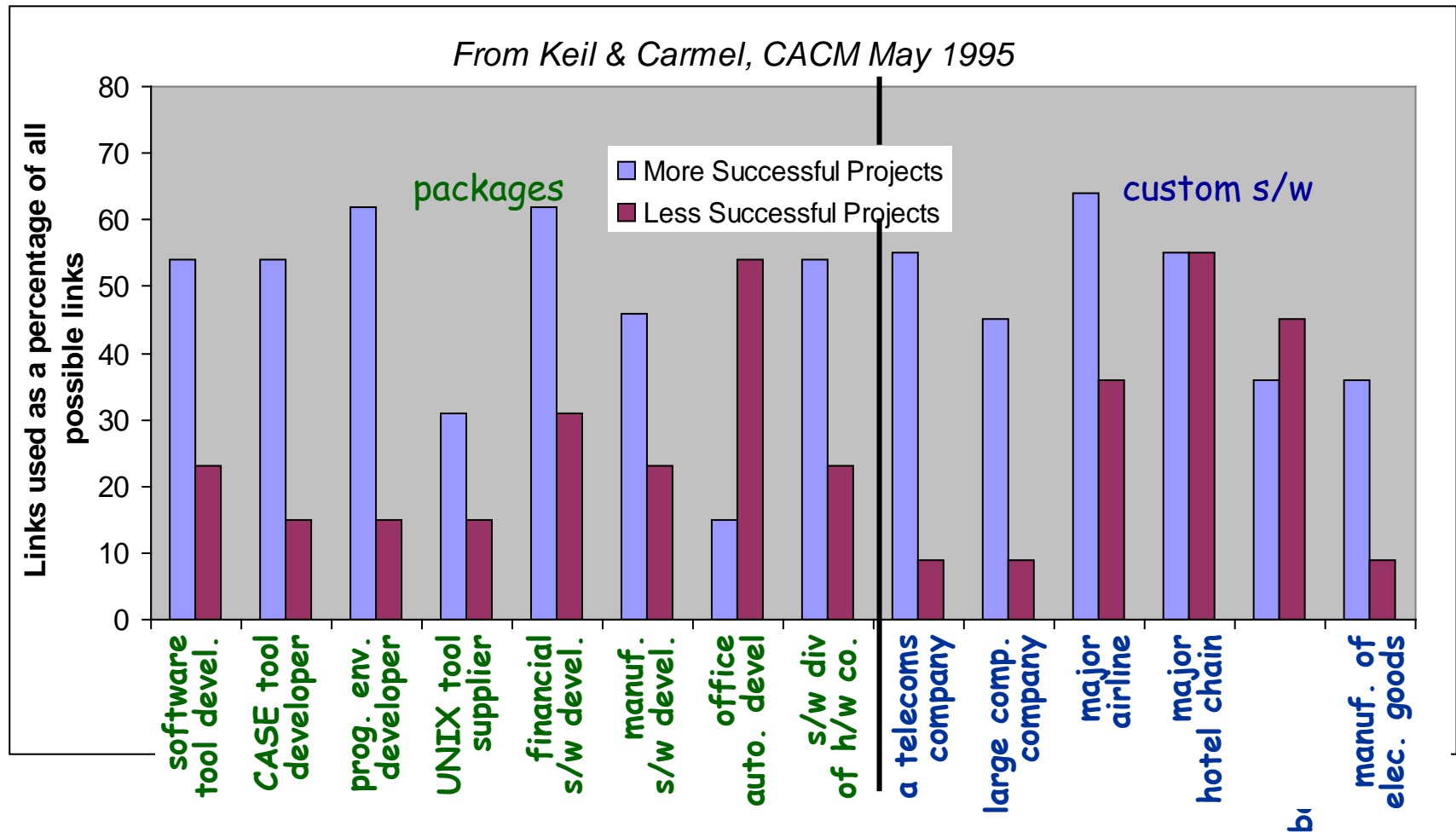
# Difficulties of Elicitation (4)

- Personal and interpersonal factors



# Importance of links with customer(s)

- Successful projects tend to have more customer links





- Traditional Approaches

- Introspection
- Existing Documents/Data
- Interviews
  - Open-ended
  - Structured
- Surveys/Questionnaires
- Group elicitation
  - Focus Groups
  - Brainstorming
  - JAD/RAD workshops
- Prototyping

- Representation-based approaches

- Goal-based
- Scenario-Based
- Use Cases

- Contextual (social) appr.

- Ethnographic techniques
  - Participant Observation
  - Ethnomethodology
- Discourse Analysis
  - Conversation Analysis
  - Speech Act Analysis
- Participatory Design
- Sociotechnical Methods
  - Soft Systems Analysis

- Cognitive approaches

- Task analysis
- Protocol analysis
- Knowl. Acquisition Technqs
  - Card Sorting
  - Laddering
  - Repertory Grids
  - Proximity Scaling

- **Traditional Approaches**

- **Introspection**
- **Existing Documents/Data**
- **Interviews**
  - Open-ended
  - Structured
- **Surveys/ Questionnaires**
- **Group elicitation**
  - Focus Groups
  - Brainstorming
  - JAD/RAD workshops
- Prototyping
- **Representation-based approaches**
  - ...

- **Contextual (social) appr.**

- Ethnographic techniques
  - Participant Observation
  - Ethnomethodology
- Discourse Analysis
  - Conversation Analysis
  - Speech Act Analysis
- Participatory Design
- Sociotechnical Methods
  - Soft Systems Analysis

- **Cognitive approaches**

- Task analysis
- Protocol analysis
- Knowl. Acquisition Technqs
  - Card Sorting
  - Laddering
  - Repertory Grids
  - Proximity Scaling

# Method: Introspection

- Just sit down and think what the requirements may be
  - Very popular with software engineers
  - But then often in the form:  
Just sit down and think up some requirements
- Advantages
  - Simple, quick, cheap, no misunderstandings
- Disadvantages
  - Often not applicable ("I have no idea")
  - **Can be extremely misleading**
  - (The mantra of usability people is: "Users are not like us!")

- Identify Collections of existing Hard Data
  - Facts and figures, financial information,...
  - Reports used for decision making,...
  - Survey results, marketing data,...
- Advantages
  - Can be quick and cheap
  - Sometimes offers very detailed information
- Disadvantages
  - **Most often not applicable**
  - Data may be biased
  - Data may be outdated

- Types:
  - Structured – agenda of fairly open questions
  - Open-ended – no pre-set agenda
- Advantages
  - Rich collection of information
- Disadvantages
  - **Interviewing is a difficult skill to master**
  - Large amount of qualitative data can be hard to analyze
  - Hard to compare different respondents
- Watch for
  - Tacit knowledge (and post-hoc rationalizations)
  - Removal from context
  - Influence from interviewer's attitude

- Advantages
  - Can quickly collect info from large numbers of people
  - Can be administered remotely
  - Can collect attitudes, beliefs, characteristics
- Disadvantages
  - Simplistic (presupposed) categories provide very little context
    - No room for users to convey their real needs
- Watch for:
  - Bias in sample selection (especially with self-selection)
  - Too-small sample size
  - Suggestive questions → answers will be biased
  - Ambiguous questions → not everyone answers the same question
  - Questionnaires **MUST** be prototyped and tested

- Types:
  - Joint/Rapid Application Development (JAD/RAD) Workshops
  - Focus Groups
  - Brainstorming
- Advantages
  - More natural interaction between people than formal interview
  - Can gauge group reaction to mock-ups, storyboards, etc.
- Disadvantages
  - May create unnatural groups (uncomfortable for participants)
  - **Danger of Groupthink**
  - May only provide superficial responses where detail is needed
  - Requires a highly trained facilitator
- Watch for
  - Sample bias
  - Dominance and submission



- Traditional Approaches
  - Introspection
  - Existing Documents/Data
  - Interviews
    - Open-ended
    - Structured
  - Surveys/Questionnaires
  - Group elicitation
    - Focus Groups
    - Brainstorming
    - JAD/RAD workshops
  - Prototyping
- **Representation-based approaches**
  - **Goal-based**
  - **Scenario-Based**
  - **Use Cases**

- Contextual (social) appr.
  - Ethnographic techniques
    - Participant Observation
    - Ethnomethodology
  - Discourse Analysis
    - Conversation Analysis
    - Speech Act Analysis
  - Participatory Design
  - Sociotechnical Methods
    - Soft Systems Analysis
- Cognitive approaches
  - Task analysis
  - Protocol analysis
  - Knowl. Acquisition Technqs
    - Card Sorting
    - Laddering
    - Repertory Grids
    - Proximity Scaling



# Goal-based approaches

- Approach

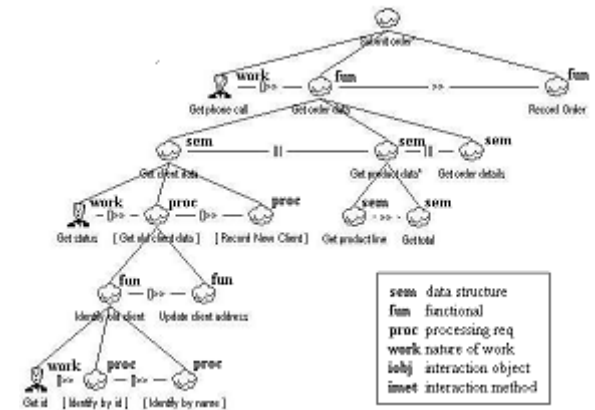
- Focus on why systems are constructed
- Express the 'why' as a set of stakeholder goals
  - The top-level goal is often "save money" or "make money"
  - Use goal refinement to arrive at specific requirements
- Goal analysis: document, organize and classify goals
- Goal hierarchies show refinement and obstacle relationships between goals

- Advantages

- Reasonably intuitive
- **Sound basis for conflict resolution**

- Disadvantages

- Hard to cope with evolution of goals
- **Either very complex goal hierarchy (can lead to analysis paralysis) or lack of detail**



- Scenarios
  - Specific sequence of interaction between actor and system
    - typically between 3 and 7 steps
  - May be:
    - positive (i.e. required behavior)
    - negative (i.e. an undesirable interaction)
  - Often used as a first step when writing use cases
- Advantages
  - Very natural: stakeholders tend to use them spontaneously
  - Short scenarios very good for quickly illustrating specific interactions
- Disadvantages
  - Lack of structure: need use cases or task models to provide higher-level view

- What is a use case?
  - A description of a sequence of actions that a system performs that yields an observable result of value to a particular actor
    - i.e. a description of a set of possible concrete scenarios that have a common purpose
  - Typically written in natural language
- Advantages
  - Easy to write, easy to understand (natural representation)
  - Helps in drawing system boundary
- Disadvantages
  - Use cases do not represent nonfunctional requirements
  - **Use cases do not capture domain knowledge**
  - Sometimes confused with a precise specification

UC_23 - eVerordnung_erstellen_Abschnitt_3	
Beschreibung	Die Informationen für eine eVerordnung werden zusammengestellt und signiert. Die eVerordnung wird auf eGK oder VODD verschoben.
Anwendungsumfeld	Institution des Akteurs Arzt (z. B. Arztpraxis, Krankenhaus)
Vorbedingungen	<ul style="list-style-type: none"><li>• Regelmäßige Nutzung der eGK ist überprüft.</li><li>• technisch nutzbare eGK liegt vor (gemäß [gemFK_CMSeGK_Nutz]).</li><li>• Zugriffsauthentisierung durch Akteure ist erfolgt.</li></ul>
Auslöser	Eine eVerordnung soll erstellt werden.
Eingangsdaten	VSD_eGK / VOD_AM / Signaturdaten
Ergebnisse	Die eVerordnung wurde auf VODD oder eGK gespeichert.
Nachbedingungen	Die eVerordnung steht auf VODD oder eGK bereit zur weiteren Verwendung.
Beteiligte Akteure	- Arzt - Mitarbeiter medizinische Institution]
Geschäftsobjekte	eVerordnung
Standardablauf	1 [Arzt oder Mitarbeiter medizinische Institution]: Informationen für die eVerordnung zusammenstellen 2 [Arzt]: Informationen der eVerordnung anzeigen (zur Signatur) 3 [Arzt]: Informationen der eVerordnung signieren 4 [Arzt oder Mitarbeiter medizinische Institution]: Schlüssel für die

# Note: Beware of natural language!

- Natural language is easy-to-use, natural, and often appropriate for describing requirements
- But it is rarely precise!
- Example:  
"Buffalo once roamed the plains in large numbers"



Buffalo once roamed the plains in large numbers.

- Traditional Approaches
  - Introspection
  - Existing Documents/Data
  - Interviews
    - Open-ended
    - Structured
  - Surveys/Questionnaires
  - Group elicitation
    - Focus Groups
    - Brainstorming
    - JAD/RAD workshops
  - Prototyping
- Representation-based approaches
  - Goal-based
  - Scenario-Based
  - Use Cases

- **Contextual (social) approaches**
  - **Ethnographic techniques**
    - Participant Observation
    - Ethnomethodology
  - Discourse Analysis
    - Conversation Analysis
    - Speech Act Analysis
  - Participatory Design
  - Sociotechnical Methods
    - Soft Systems Analysis
- Cognitive approaches
  - Task analysis
  - Protocol analysis
  - Knowl. Acquisition Techniqs

# The ethnomethodologist's view

## Requirements elicitation is a social activity

- Because it involves people-to-people communication
  - through discussions, observation, etc.
- Because it involves negotiation in bringing about consensus when there is disagreement.
- Because it affects and changes human activity systems

## The domain of application is often a social world

- Need techniques that uncover the order of the social world
  - Social order might not be immediately obvious or describable
  - Social order cannot be assumed to have an *a priori* structure
- **Social order can only be understood through immersion**
  - Social order is constructed by the participants' actions
  - Need to witness the unfolding of social phenomena
  - Cannot just collect data using pre-given categories

- Assumptions
  - Social world is ordered
  - We cannot know the order a-priori
  - To understand it, we need to immerse in its natural setting
- Categories
  - Most conventional approaches assume preexisting categories
    - This may mislead the observer (appropriation)
  - Ethnography attempts to use the subjects' own categories
    - Related to postmodern deconstruction: "there is no grand narrative"
- Measurement
  - There is no scientific objectivity about social phenomena
  - We need to use the subjects' own measurement theory

# Method: Participant observation ("teilnehmende Beobachtung")

- Approach
  - Observer spends time with the subjects, joining in, long enough to become a member of the group ('longitudinal studies')
- Advantages
  - Highly contextualized and relatively reliable
  - Reveals details that other methods cannot
- Disadvantages
  - **Extremely time consuming!**
  - Resulting 'rich picture' is hard to analyze
  - Cannot say much about the results of proposed changes
- Watch for
  - going native!





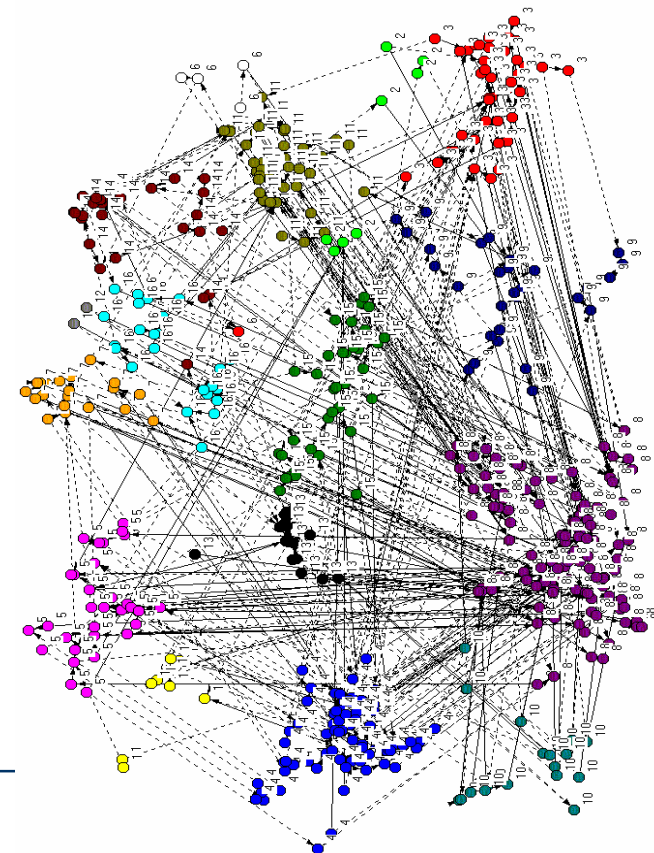
- Traditional Approaches
  - Introspection
  - Existing Documents/Data
  - Interviews
    - Open-ended
    - Structured
  - Surveys/Questionnaires
  - Group elicitation
    - Focus Groups
    - Brainstorming
    - JAD/RAD workshops
  - Prototyping
- Representation-based approaches
  - Goal-based
  - Scenario-Based
  - Use Cases

- Contextual (social) appr.
  - ...
- **Cognitive approaches**
  - Task analysis
  - **Protocol analysis**
  - **Knowledge Acquisition Techniques**
    - **Card Sorting**
    - Laddering
    - Repertory Grids
    - **Proximity Scaling**

# Method: Protocol Analysis

- Based on protocols created from vocalizing behavior
  - either "think-aloud" or retrospective
- Analyze protocols to reveal requirements
- Advantages
  - Direct verbalization of cognitive activities
  - Embedded in the work context
  - Good at revealing interaction problems with existing systems
- Disadvantages
  - **Interpretation requires introspection, hence unreliable**
  - No social dimension

- Given some domain objects, derive a set of dimensions for classifying them:
  - Step 1: pairwise proximity assessment among domain elements
    - captures tacit knowledge of expert
  - Step 2: automated statistical analysis to build multi-dimensional space to classify the objects
- Advantages
  - Helps to elicit mental models where complex multivariate data is concerned
  - **Good for eliciting tacit knowledge**
- Disadvantages
  - Requires an agreed-upon set of objects
  - Only models classification knowledge, **not performance knowledge**



# Method: Card sorting

- For a given set of domain objects, written on cards:
  - Expert sorts the cards into groups...
  - ...then says what the criterion was for sorting, and what the groups were

- Advantages

- Simple, amenable to automation
  - Elicits classification knowledge
- **Good for eliciting tacit knowledge**

- Disadvantages

- Requires an agreed-upon set of objects
- Only models classification knowledge, **not performance knowledge**

Airport classification criteria proposed by CASA



Data from [CASA](#)

- **Requirements** represent the goals to be reached via a software system
- A **specification** describes what the software must do in order to fulfill the requirements
  - assuming certain domain properties are met
- Requirements **elicitation** is the basic step of **Requirements Engineering**
  - others are Req. Specification, Req. **Validation**, and Req. **Management**
- Requirements Elicitation must overcome many recurring problems
- Many different elicitation **techniques** should be combined

- James Robertson, Suzanne Robertson: "Mastering the Requirements Process: Getting Requirements Right", 3rd ed., Addison-Wesley 2012
- Donald Gause, Gerald Weinberg: "Exploring Requirements – Quality before Design", B&T, 1989
  - auf deutsch: "Software Requirements: Anforderungen erkennen, verstehen und erfüllen", (vergriffen)

**Thank you!**