Tvorba informačných systémov

2. prednáška – čo všetko sa týka špecifikácie požiadaviek

Informačný systém (softvér)

Dve kategórie IS:

–„krabicový“ softvér – predáva sa na trhu všeobecnej verejnosti (generické produkty), napr. Office, operačné systémy, počítačové hry,...

–softvér „na mieru“ – vytvorený pre konkrétne špecifické potreby zákazníka (produkty na zakázku), napr. zdravotná alebo sociálna poisťovňa, správa smetiarskych vozidiel, …

V prvom prípade si požiadavky stanovuje samotná softvérová firma, v druhom prípade je potrebné vyhovieť konkrétnym potrebám zákazníka, tvorcovia IS majú omnoho menej slobody.

Zabezpečenie a určenie kvality softvéru

Model kvality (ISO/IEC 9126-1)

Externé atribúty kvality – z hľadiska behu softvéru

Interné atribúty kvality – vlastnosti z hľadiska vývojára

Atribúty kvality použitia – vlastnosti hotového produktu

Zabezpečenie a určenie kvality softvéru

Externé atribúty kvality:

- spoľahlivosť: softvér pracuje podľa špecifikácie (správnosť) + správa sa „rozumne“ aj v prípadoch nepokrytých špecifikáciou (robustnosť)
- flexibilnosť: jednoduchosť zmeny softvéru v prípade zmien požiadaviek/prostredia
- znovupoužiteľnosť: možnosť použiť softvér (jeho časti) v iných kontextoch
- kompatibilnosť (interoperabilita): možnosť použiť SW produkt s inými produktami

- efektívnosť: schopnosť minimalizovať požiadavky na HW (CPU, interná/externá pamäť, komunikačné kanály, ...)
- portabilita: jednoduchosť prenesenia softvéru na inú HW/SW platformu
- jednoduchosť použitia: miera námahy potrebnej na zaškolenie a používanie softvéru; vrátane jeho inštalácie a správy
- ISO/IEC 9126 Software engineering — Product quality
- ISO/IEC 25000: Software engineering: Software product Quality Requirements and Evaluation (SquaRE)
- môžu byť uvedené ako „non-functional requirements“

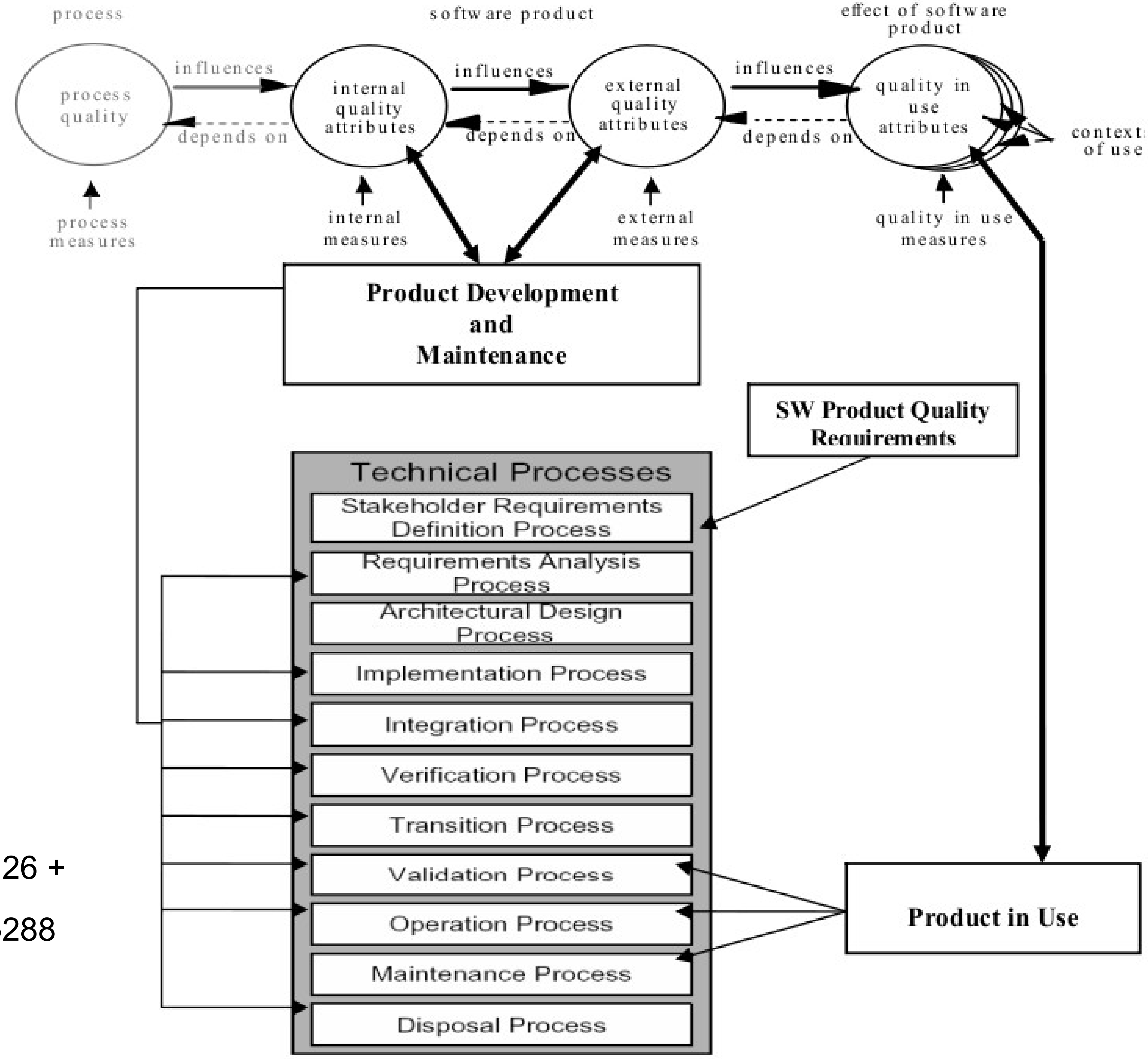
  (požiadavky, ktoré nepopisujú funkcionalitu)

  Nie je možné dosiahnuť všetky naraz!

Interné atribúty kvality:

(neviditeľné pre vonkajšieho pozorovateľa, avšak podstatné pre dosahovanie externých atribútov), napr.

- modulárnosť (vhodná architektúra)
- zrozumiteľnosť kódu
- minimum rozhraní
- minimálne rozhrania



ISO/IEC 9126 +

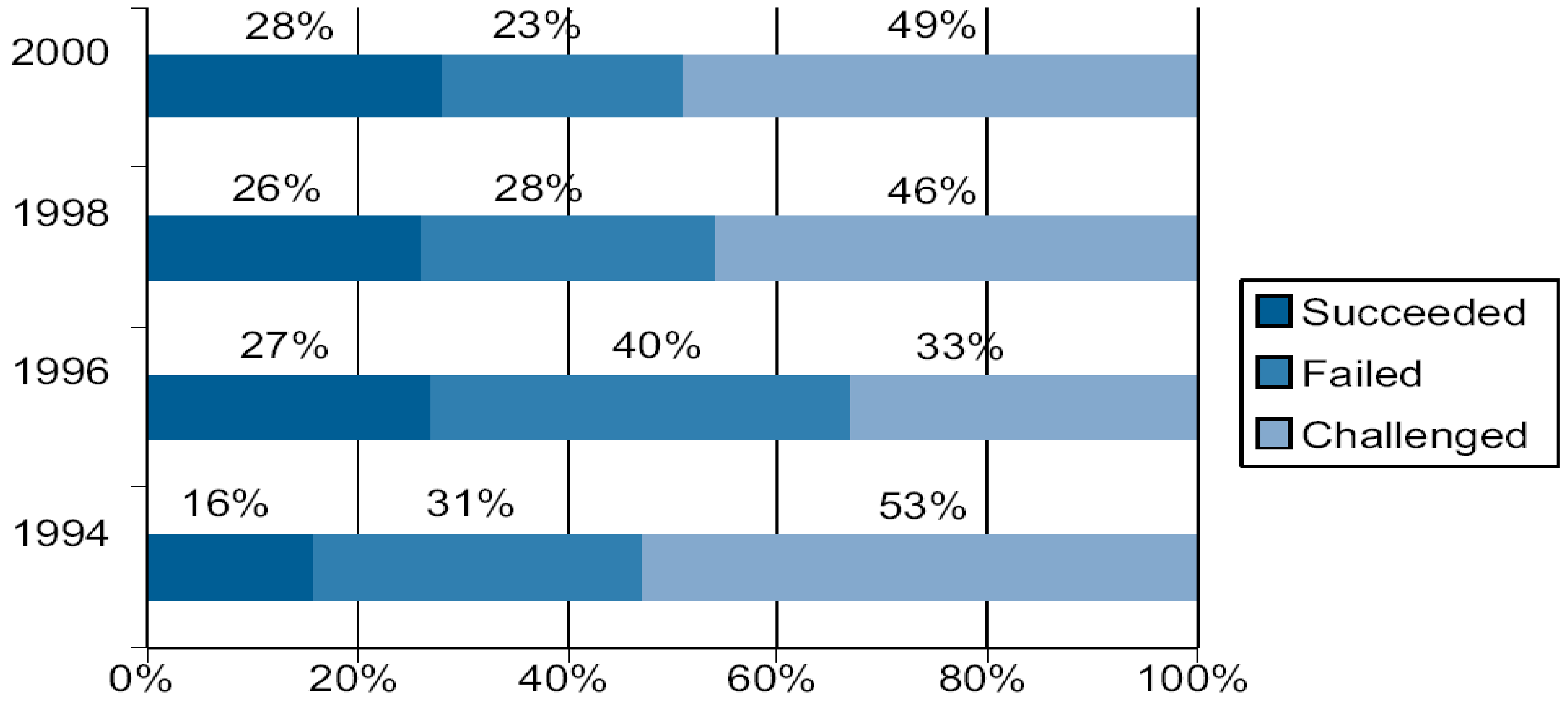
ISO/IEC 15288

Štruktúra nákladov na vývoj softvéru

- závisí od druhu produktu a od procesu vývoja
- pred odovzdaním: cca 60% vývoj, 40% testovanie
  - pri generických produktoch je objem testovania väčší
  - rovnako pri kritických systémoch
- evolúcia: 2-3x viac než náklady do odovzdania

  (platí pri produktoch na zákazku)

Úspešnosť SW projektov



Challenged: prekročenie času/rozpočtu a/alebo nesplnenie požiadaviek

Failed: výsledok projektu nebol uvedený do prevádzky

Zdroj: Standish Chaos Reports

Úspešnosť SW projektov

	1994	1996	1998	2000	2002	2004	2009
Succeeded	16%	27%	26%	28%	34%	29%	32%
Failed	31%	40%	28%	23%	15%	18%	24%
Challenged	53%	33%	46%	49%	51%	53%	44%

National Institute of Standards and Technology (NIST)

* Software defects cost nearly $60 Billion Annually

* 80% of development costs involve identifying and correcting defects

More:

http://www.galorath.com/wp/software-project-failure-costs-billions-better-estimation-planning-can-help.php

Hlavné role v procese vývoja SW

- zadávateľ (zákazník, klient)
- riešiteľ (dodávateľ, vývojár, developer)
- používateľ (end-user)

a ďalšie, napr. prevádzkovateľ …

„stakeholder“

Types of stakeholder

- Software engineers responsible for system development
- System end-users who will use the system after it has been delivered
- Managers of system end-users who are responsible for their work
- External regulators who check that the system meets its legal requirements
- Domain experts who give essential background information about the system application domain

Špecifikácia požiadaviek

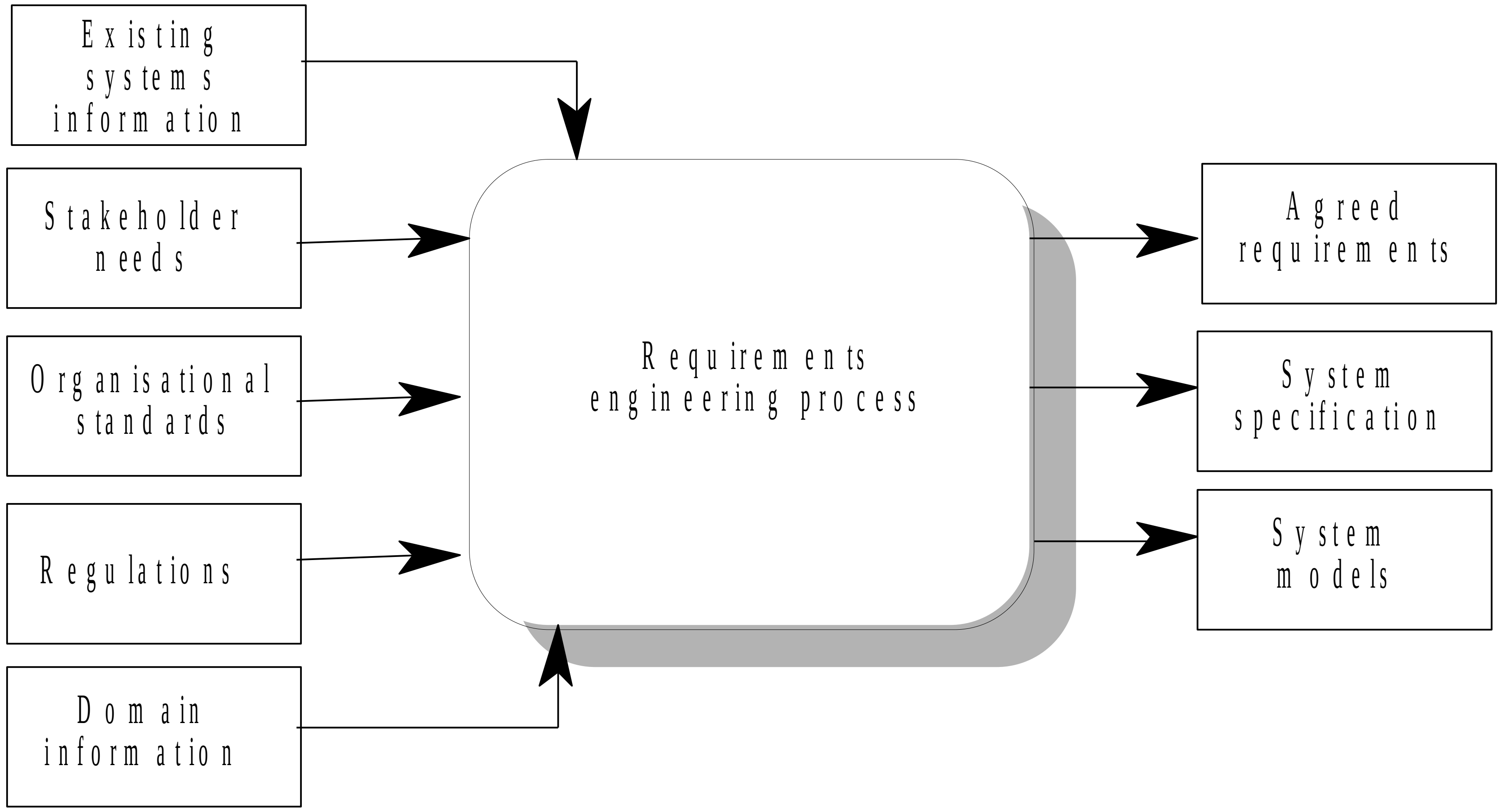
- cieľ:
  - vytvorenie uceleného katalógu požiadaviek na produkt (t.j. čo zadávateľ od produktu požaduje)
- výsledok (špecifikácia požiadaviek) obsahuje:
  - popis funkcií produktu
  - popis údajov, s ktorými produkt pracuje
  - ostatné vlastnosti produktu a obmedzenia naň kladené (tzv. non-functional requirements)
- špecifikácia je záväzná pre zadávateľa aj riešiteľa

- vlastnosti:
  - špecifikácia požiadaviek musí byť zrozumiteľná pre zadávateľa aj riešiteľa
  - musí byť jednoznačná, úplná a vnútorne konzistentná
- prostriedky:
  - neformálne: prirodzený jazyk, diagramy, ...
  - semiformálne: čiastočne formalizované jazyky (napr. UML)
  - formálne: jazyky s matematickým základom (napr. Z)

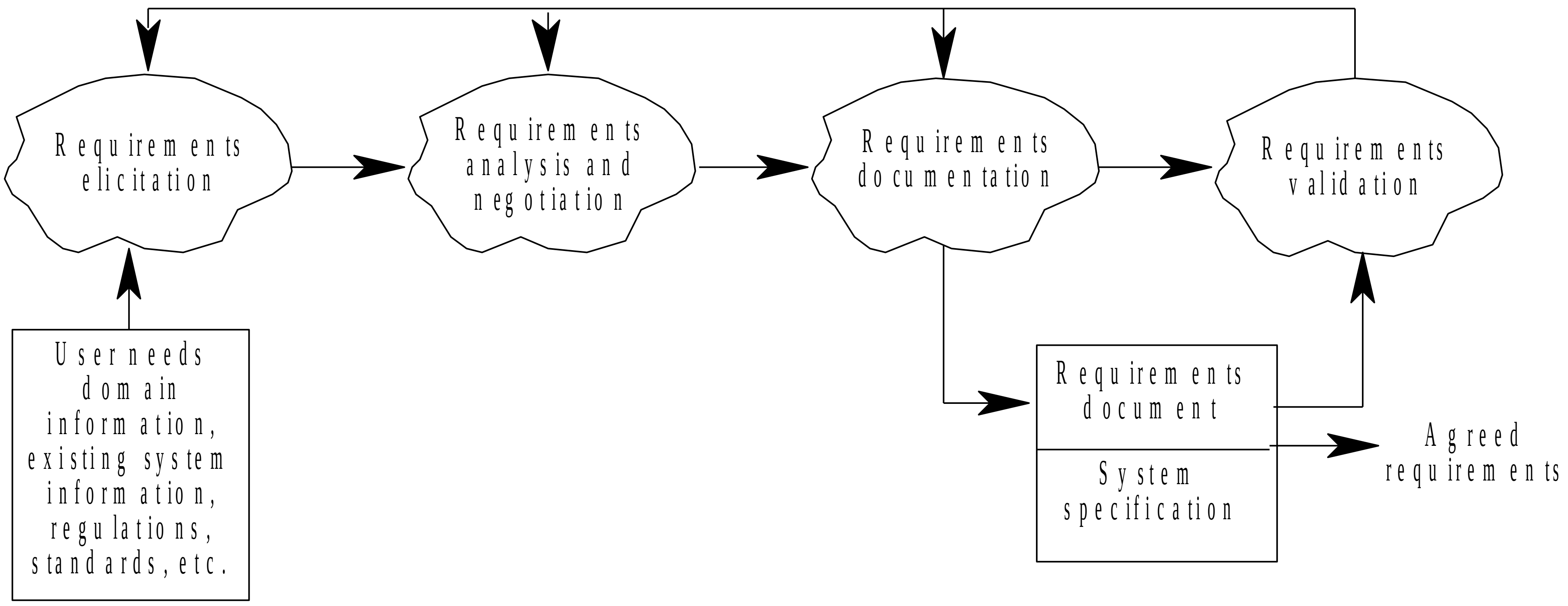
čím formálnejšie prostriedky, tým je špecifikácia presnejšia, ale pre laika menej zrozumiteľná; plne formálne jazyky sa používajú len v určitých prípadoch (napr. kritické systémy)

- problém:
  - zadávateľ si nevie jasne predstaviť, ako bude výsledný systém vyzerať
  - riešiteľ nie je expert v oblasti, pre ktorú systém implementuje
  - riešenie: čo najviac kvalitnej komunikácie!

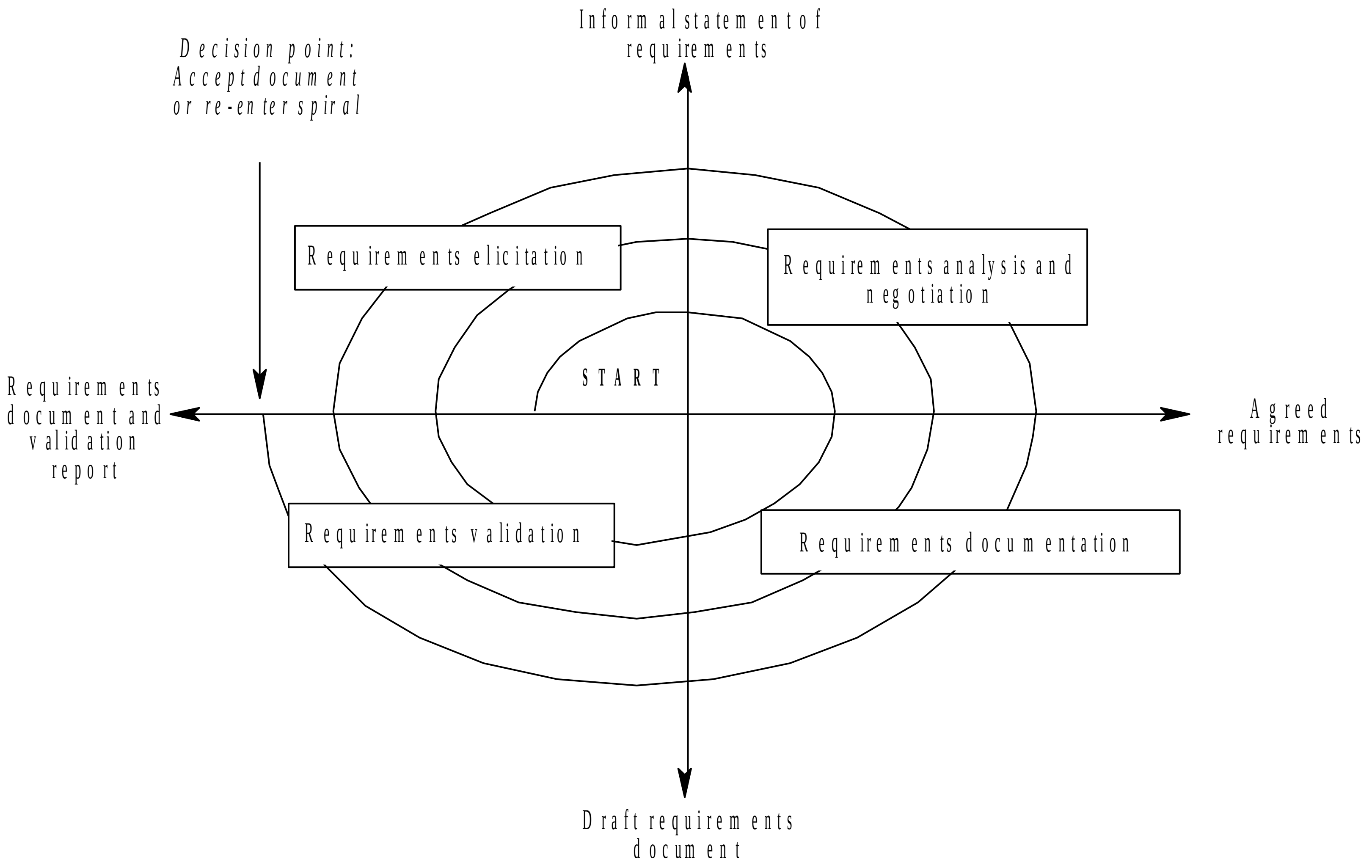
RE process - inputs and outputs



Coarse-grain activity model of RE


Requirements elicitation
Requirements analysis and negotiation
Requirements documentation
Requirements validation
User needs domain information, existing system information, regulations, standards, etc.
Requirements document
System specification
Agreed requirements

Spiral model of the RE process


Decision point: Accept document or re-enter spiral
Informal statement of requirements
Requirements elicitation
Requirements analysis and negotiation
START
Requirements document and validation report
Agreed requirements
Requirements validation
Requirements documentation
Draft requirements document

RE process problems

- Lack of stakeholder involvement
- Business needs not considered
- Lack of requirements management
- Lack of defined responsibilities
- Stakeholder communication problems
- Over-long schedules and poor quality requirements documents

Katalóg požiadaviek

- je formálnym dokumentom, ktorý ako základ komunikácie používajú zákazníci, vývojári a management
- opisuje
  - Služby a funkcie, ktoré má systém poskytovať
  - Obmedzenia (constraints) podmienok, v ktorých softvér bude pracovať
  - Celkové vlastnosti systému, vrátane obmedzení na vyplývajúce vlastnosti
  - Definície ostatných systémov s ktorými bude systém integrovaný.

Katalóg požiadaviek

- ďalej opisuje:
  - Informácie z oblasti príslušnej aplikačnej domény – ako v nej prebiehajú procesy, výpočty, aké sú závislosti
  - Všetky obmedzenia na spôsob vývoja softvéru
  - Hardvér i softvér na ktorom sa má systém prevádzkovať
- V každom prípade by katalóg požiadaviek mal obsahovať úvodnú kapitolu s celkovým prehľadom systému, komerčnými potrebami z ktorých vychádza zákazka a slovníkom pojmov, ktorý vysvetľuje použitú terminológiu.

Requirements document structure

- IEEE/ANSI 830-1993 standard proposes a structure for software requirements documents
- Introduction
  - 1.1 Purpose of requirements document
  - 1.2 Scope of the product
  - 1.3 Definitions, acronyms and abbreviations
  - 1.4 References
  - 1.5 Overview of the remainder of the document

Requirements document structure

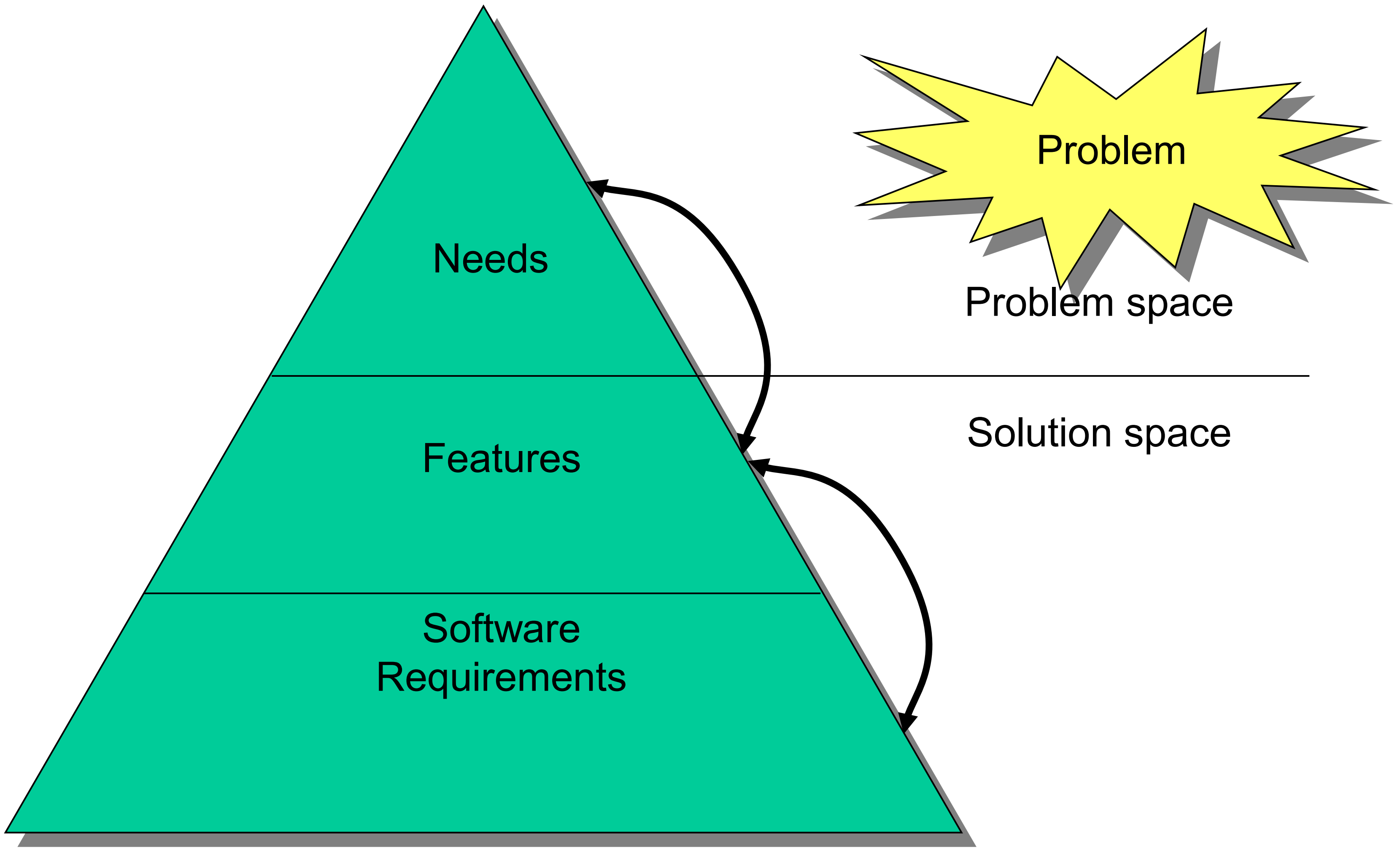
- 2. General description
  - 2.1 Product perspective
  - 2.2 Product functions
  - 2.3 User characteristics
  - 2.4 General constraints
  - 2.5 Assumptions and dependencies
- 3. Specific requirements

  Covering functional, non-functional and interface requirements.
- 4. Appendices
- Index

Users of requirements documents

- System customers
  - specify the requirements and read them to check they meet their needs
- Project managers
  - Use the requirements document to plan a bid for system and to plan the system development process
- System engineers
  - Use the requirements to understand the system being developed
- System test engineers
  - Use the requirements to develop validation tests for the system
- System maintenance engineers
  - Use the requirements to help understand the system

Requirements categories overview



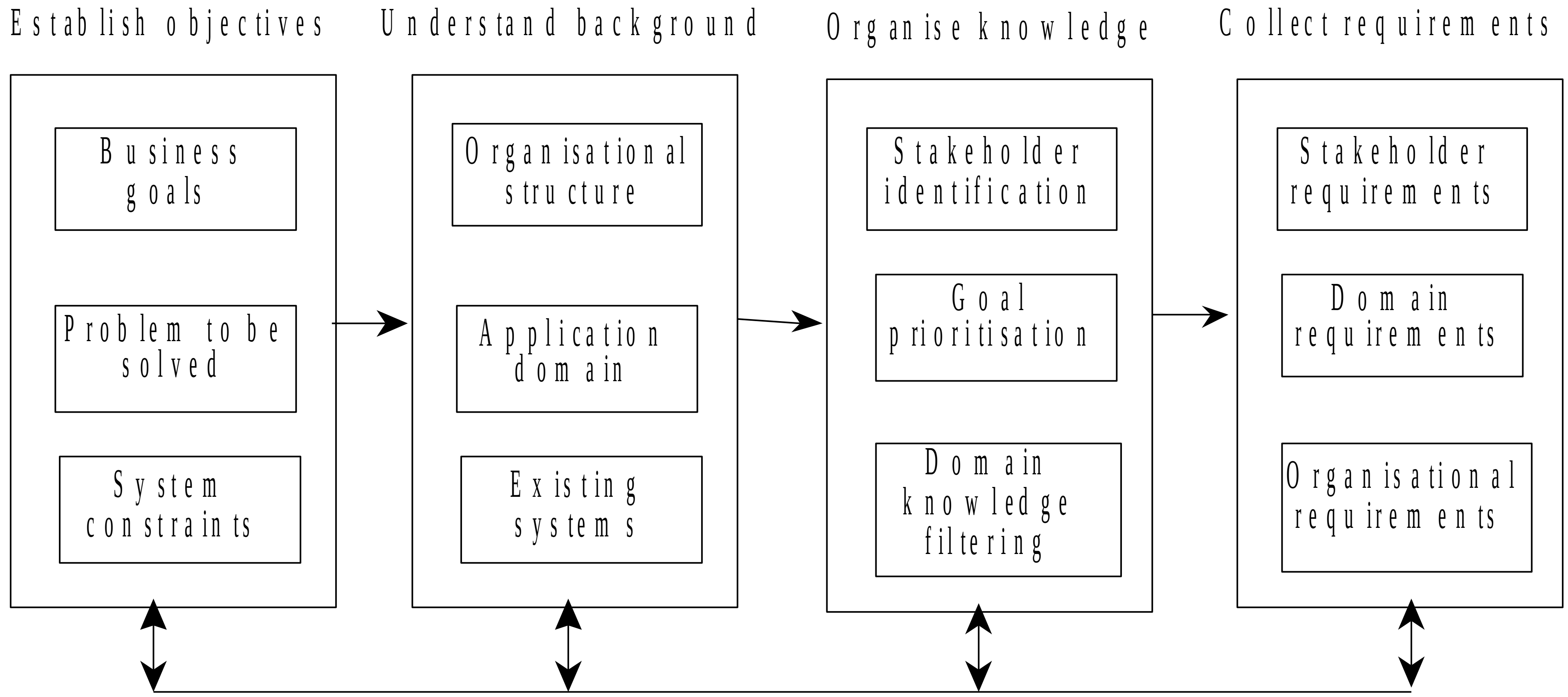
It is important to understand that, (except where problem is motivated by the technology),

the problem is an artifact of the problem domain and is generally technology neutral.

Requirements problems

- The requirements don’t reflect the **real needs** of the customer for the system.
- Requirements are **inconsistent** and/or **incomplete**.
- It is expensive to make **changes** to requirements after they have been agreed.
- There are **misunderstandings** between customers, those developing the system requirements and software engineers developing or maintaining the system.

The requirements elicitation process



Elicitation stages

- Objective setting
  - The organisational objectives should be established including general goals of the business, an outline description of the problem to be solved, why the system is necessary and the constraints on the system.
- Background knowledge acquisition
  - Background information about the system includes information about the organisation where the system is to be installed, the application domain of the system and information about existing systems
- Knowledge organisation
  - The large amount of knowledge which has been collected in the previous stage must be organised and collated.
- Stakeholder requirements collection
  - System stakeholders are consulted to discover their requirements.

Interviews

- The requirements engineer or analyst discusses the system with different stakeholders and builds up an understanding of their requirements.
- Types of interview
  - *Closed interviews*. The requirements engineer looks for answers to a pre-defined set of questions
  - *Open interviews* There is no predefined agenda and the requirements engineer discusses, in an open-ended way, what stakeholders want from the system.

Interviewing essentials

- Interviewers must be open-minded and should not approach the interview with pre-conceived notions about what is required
- Stakeholders must be given a starting point for discussion. This can be a question, a requirements proposal or an existing system
- Interviewers must be aware of organisational politics - many real requirements may not be discussed because of their political implications

Scenarios

- Scenarios are stories which explain how a system might be used. They should include
  - a description of the system state before entering the scenario
  - the normal flow of events in the scenario
  - exceptions to the normal flow of events
  - information about concurrent activities
  - a description of the system state at the end of the scenario
- Scenarios are examples of interaction sessions which describe how a user interacts with a system
- Discovering scenarios exposes possible system interactions and reveals system facilities which may be required

Observation and social analysis

- People often find it hard to describe what they do because it is so natural to them. Sometimes, the best way to understand it is to observe them at work
- Ethnography is a technique from the social sciences which has proved to be valuable in understanding actual work processes
- Actual work processes often differ from formal, prescribed processes
- An ethnographer spends some time observing people at work and building up a picture of how work is done

Prototyping

- A prototype is an initial version of a system which may be used for experimentation
- Prototypes are valuable for requirements elicitation because users can experiment with the system and point out its strengths and weaknesses. They have something concrete to criticise
- Rapid development of prototypes is essential so that they are available early in the elicitation process

Prototyping benefits

- The prototype allows users to experiment and discover what they really need to support their work
- Establishes feasibility and usefulness before high development costs are incurred
- Essential for developing the ‘look and feel’ of a user interface
- Can be used for system testing and the development of documentation
- Forces a detailed study of the requirements which reveals inconsistencies and omissions

Types of prototyping

- Throw-away prototyping
  - intended to help elicit and develop the system requirements.
  - The requirements which should be prototyped are those which cause most difficulties to customers and which are the hardest to understand. Requirements which are well-understood need not be implemented by the prototype.
- Evolutionary prototyping
  - intended to deliver a workable system quickly to the customer.
  - Therefore, the requirements which should be supported by the initial versions of this prototype are those which are well-understood and which can deliver useful end-user functionality. It is only after extensive use that poorly understood requirements should be implemented.

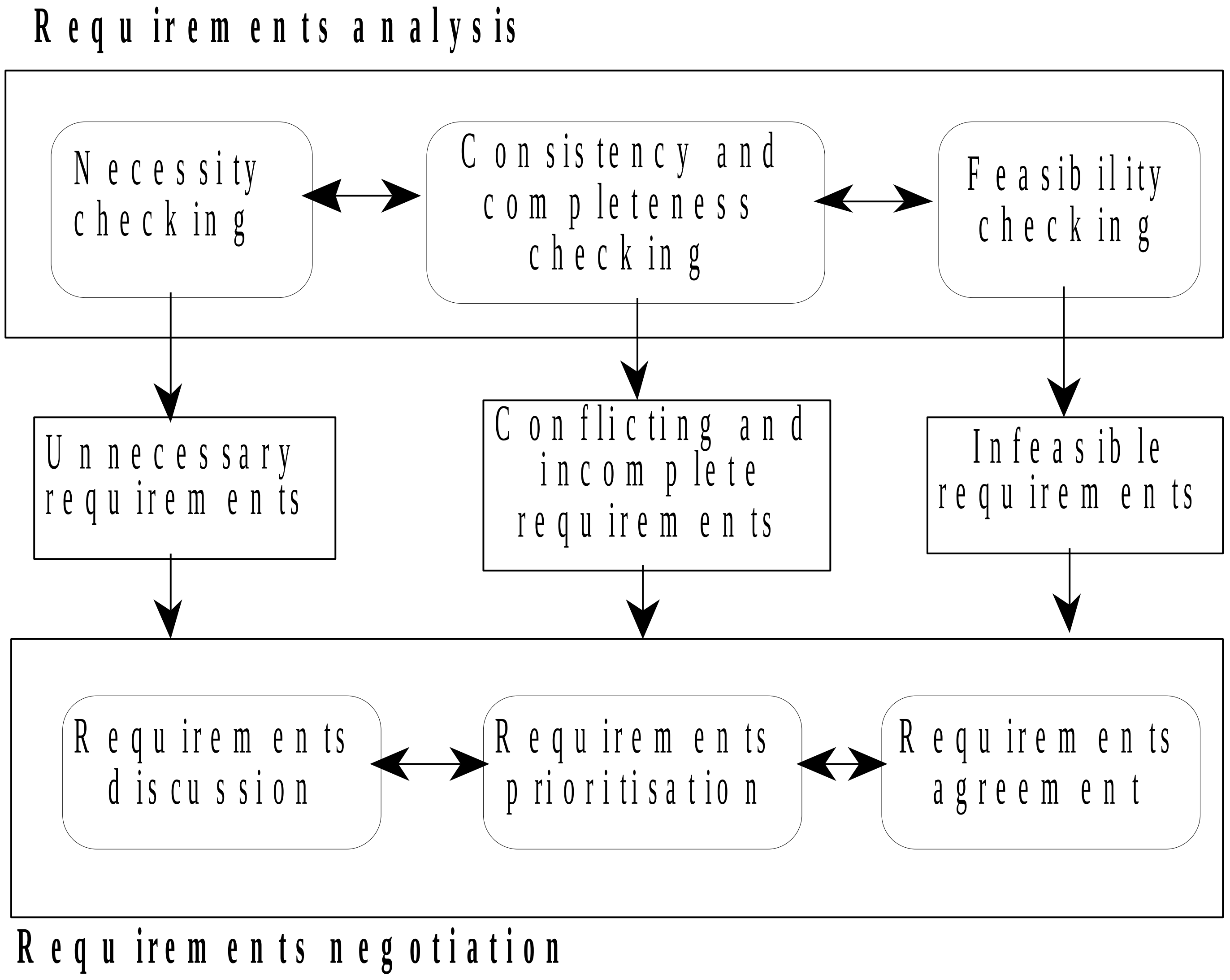
Prototyping costs and problems

- Training costs - prototype development may require the use of special purpose tools
- Development costs - depend on the type of prototype being developed
- Extended development schedules - developing a prototype may extend the schedule although the prototyping time may be recovered because rework is avoided
- Incompleteness - it may not be possible to prototype critical system requirements

Approaches to prototyping

- Paper prototyping
  - a paper mock-up of the system is developed and used for system experiments
- ‘Wizard of Oz’ prototyping
  - a person simulates the responses of the system in response to some user inputs
- Executable prototyping
  - a fourth generation language or other rapid development environment is used to develop an executable prototype

Requirements analysis and negotiation



Requirements analysis

- The goal of analysis is to discover problems, incompleteness and inconsistencies in the elicited requirements. These are then fed back to the stakeholders to resolve them through the negotiation process
- Analysis is interleaved with elicitation as problems are discovered when the requirements are elicited
- A problem checklist may be used to support analysis. Each requirement may be assessed against the checklist

Analysis checklists

- *Premature design*
  - Does the requirement include premature design or implementation information?
- *Combined requirements*
  - Does the description of a requirement describe a single requirement or could it be broken down into several different requirements?
- *Unnecessary requirements*
  - Is the requirement ‘gold plating’? That is, is the requirement a cosmetic addition to the system which is not really necessary.
- *Use of non-standard hardware*
  - Does the requirement mean that non-standard hardware or software must be used? To make this decision, you need to know the computer platform requirements.

Analysis checklists

- *Conformance with business goals*
  - Is the requirement consistent with the business goals defined in the introduction to the requirements document? Requirements ambiguity
- *Requirements ambiguity*
  - Is the requirement ambiguous i.e. could it be read in different ways by different people? What are the possible interpretations of the requirement?
- *Requirements realism*
  - Is the requirement realistic given the technology which will be used to implement the system?
- *Requirements testability*
  - Is the requirement testable, that is, is it stated in such a way that test engineers can derive a test which can show if the system meets that requirement?

Requirements interactions

- A very important objective of requirements analysis is to discover the interactions between requirements and to highlight requirements conflicts and overlaps
- A requirements interaction matrix shows how requirements interact with each other. Requirements are listed along the rows and columns of the matrix
  - For requirements which conflict, fill in a 1
  - For requirements which overlap, fill in a 1000
  - For requirements which are independent, fill in a 0

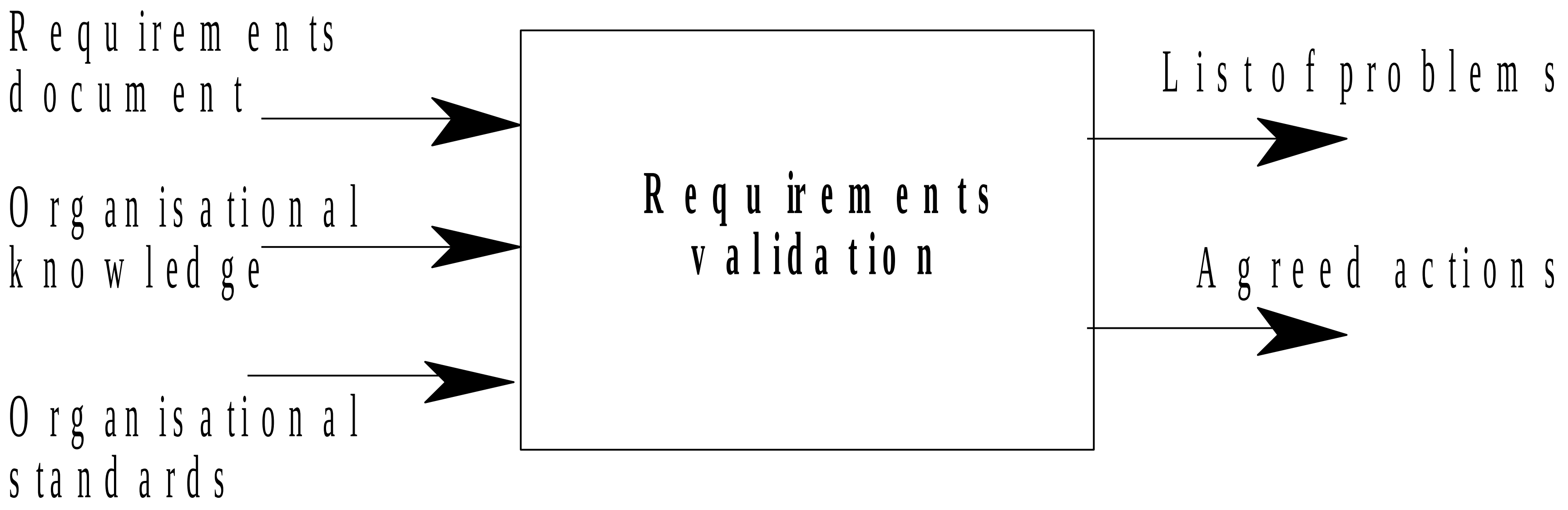
Requirements negotiation

- Disagreements about requirements are inevitable when a system has many stakeholders. Conflicts are not ‘failures’ but reflect different stakeholder needs and priorities
- Requirements negotiation is the process of discussing requirements conflicts and reaching a compromise that all stakeholders can agree to
- In planning a requirements engineering process, it is important to leave enough time for negotiation. Finding an acceptable compromise can be time-consuming

Analysis and validation

- Analysis works with raw requirements as elicited from the system stakeholders
  - “Have we got the right requirements” is the key question to be answered at this stage
- Validation works with a final draft of the requirements document i.e. with negotiated and agreed requirements
  - “Have we got the requirements right” is the key question to be answered at this stage

Validation inputs and outputs



Problem actions

- *Requirements clarification*
  - The requirement may be badly expressed or may have accidentally omitted information which has been collected during requirements elicitation.
- *Missing information*
  - Some information is missing from the requirements document. It is the responsibility of the requirements engineers who are revising the document to discover this information from system stakeholders.
- *Requirements conflict*
  - There is a significant conflict between requirements. The stakeholders involved must negotiate to resolve the conflict.
- *Unrealistic requirement*
  - The requirement does not appear to be implementable with the technology available or given other constraints on the system. Stakeholders must be consulted to decide how to make the requirement more realistic.

Review checklists

- *Understandability*
  - Can readers of the document understand what the requirements mean?
- *Redundancy*
  - Is information unnecessarily repeated in the requirements document?
- *Completeness*
  - Does the checker know of any missing requirements or is there any information missing from individual requirement descriptions?
- *Ambiguity*
  - Are the requirements expressed using terms which are clearly defined? Could readers from different backgrounds make different interpretations of the requirements?

Review checklists

- *Consistency*
  - Do the descriptions of different requirements include contradictions? Are there contradictions between individual requirements and overall system requirements?
- *Organisation*
  - Is the document structured in a sensible way? Are the descriptions of requirements organised so that related requirements are grouped?
- *Conformance to standards*
  - Does the requirements document and individual requirements conform to defined standards? Are departures from the standards, justified?
- *Traceability*
  - Are requirements unambiguously identified, include links to related requirements and to the reasons why these requirements have been included?

Checklist questions

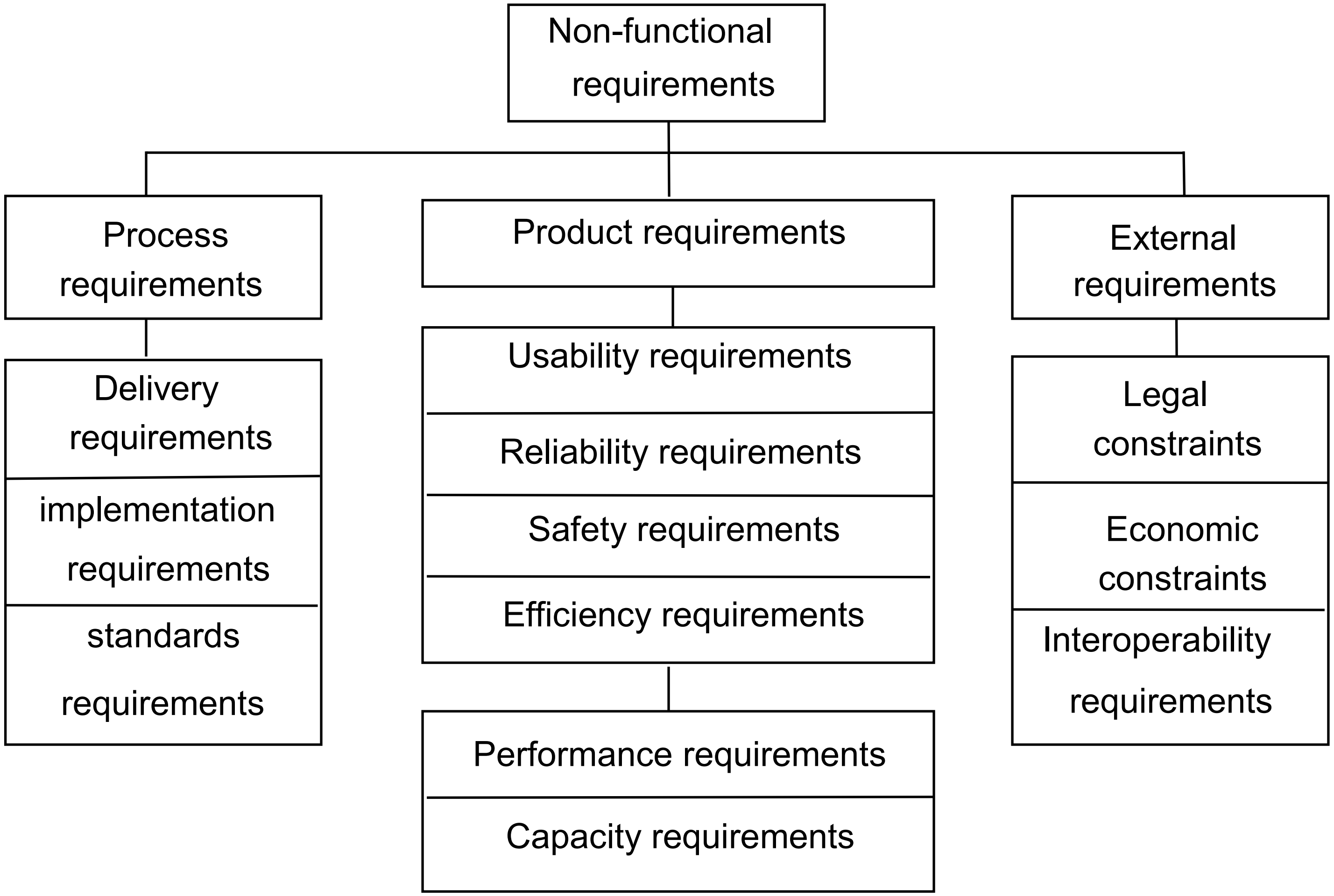
- Is each requirement uniquely identified?
- Are specialised terms defined in the glossary
- Does a requirement stand on its own or do you have to examine other requirements to understand what it means?
- Do individual requirements use the terms consistently
- Is the same service requested in different requirements? Are there any contradictions in these requests?
- If a requirement makes reference to some other facilities, are these described elsewhere in the document?
- Are related requirements grouped together? If not, do they refer to each other?

Types of Non-functional requirements

The 'IEEE-Std 830 - 1993' lists 13 non-functional requirements to be included in a Software Requirements Document.

- Performance requirements
- Interface requirements
- Operational requirements
- Resource requirements
- Verification requirements
- Acceptance requirements
- Documentation requirements
- Security requirements
- Portability requirements
- Quality requirements
- Reliability requirements
- Maintainability requirements
- Safety requirements

Classification of NFRs

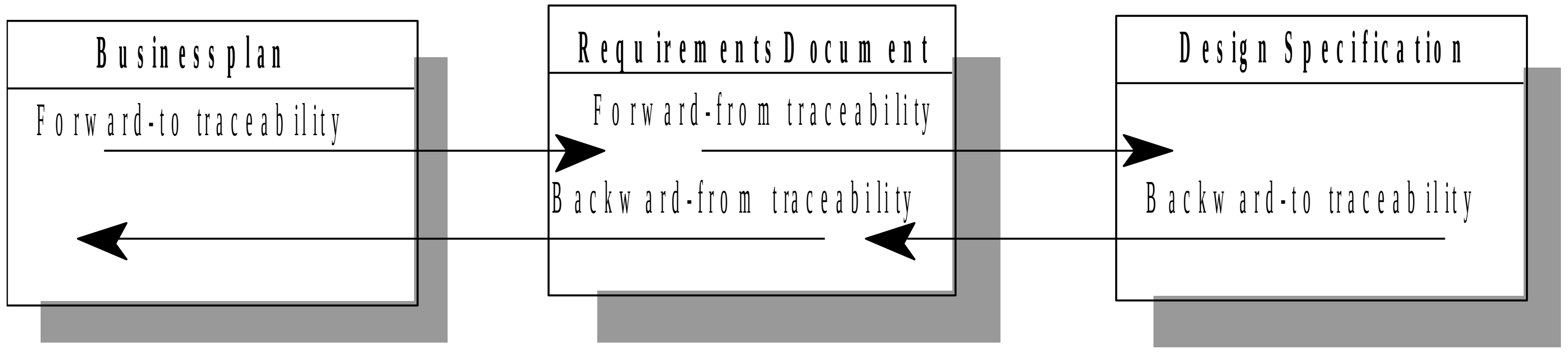


Requirements identification

- It is essential for requirements management that every requirement should have a unique identification
- The most common approach is requirements numbering based on chapter/section in the requirements document
- Example: Symbolic identification
  - Requirements can be identified by giving them a symbolic name which is associated with the requirement itself. For example, EFF-1, EFF-2, EFF-3 may be used for requirements which relate to system efficiency

Traceability

- Traceability information is information which helps you assess the impact of requirements change. It links related requirements and the requirements and other system representations
- Types of traceability information


Business plan
Forward-to traceability
Requirements Document
Forward-from traceability
Backward-from traceability
Design Specification
Backward-to traceability