PRACTICAL III

The University would like to develop a News App that covers stuff of general interest to staff and students.

This can be:

University news
Local news
Events

Think up some requirements. Give them a type.

GATHERING REQUIREMENTS

Talk to people

Observation

Examining existing products

Daydream

TALKING TO PEOPLE

Interviews
"Conversation" between one interviewer and one interviewee

Focus groups
One interviewer leads a discussion with multiple interviewees

Questionnaires
Paper-based rather than interviewer led

WHO TO TALK TO

Users
Known users
– people who will use the new system
Potential users
– people who use similar systems or might like to use the new system
Other stakeholders
Anyone else affected by the system
– including other employees, customers, suppliers, competitors(?)

DIRECT OBSERVATION

Watching users carry out the task in the natural setting

Tells us about the task, the user and the context
Requires a lot of time and cooperation

Generates a lot of data
Does being observed alter the way things are done?

INDIRECT OBSERVATION

Tracking users' activity without directly watching them

e.g. User diaries or automatic logging

Particularly useful when direct observation is not possible

Data quality in diaries is only as good as the reliability of the user

Data logging requires no user actions

Ethical issues of logging behaviour

STUDYING DOCUMENTATION

States how users should behave
Doesn't require stakeholder's time
Likely to explain legislative and regulatory issues involved in task

Some parts of task might not be covered by documents
Practice might differ from theory

EXAMINING EXISTING PRODUCTS

Suggests requirements any new product is likely to have

May identify problems with existing systems that new one can improve on

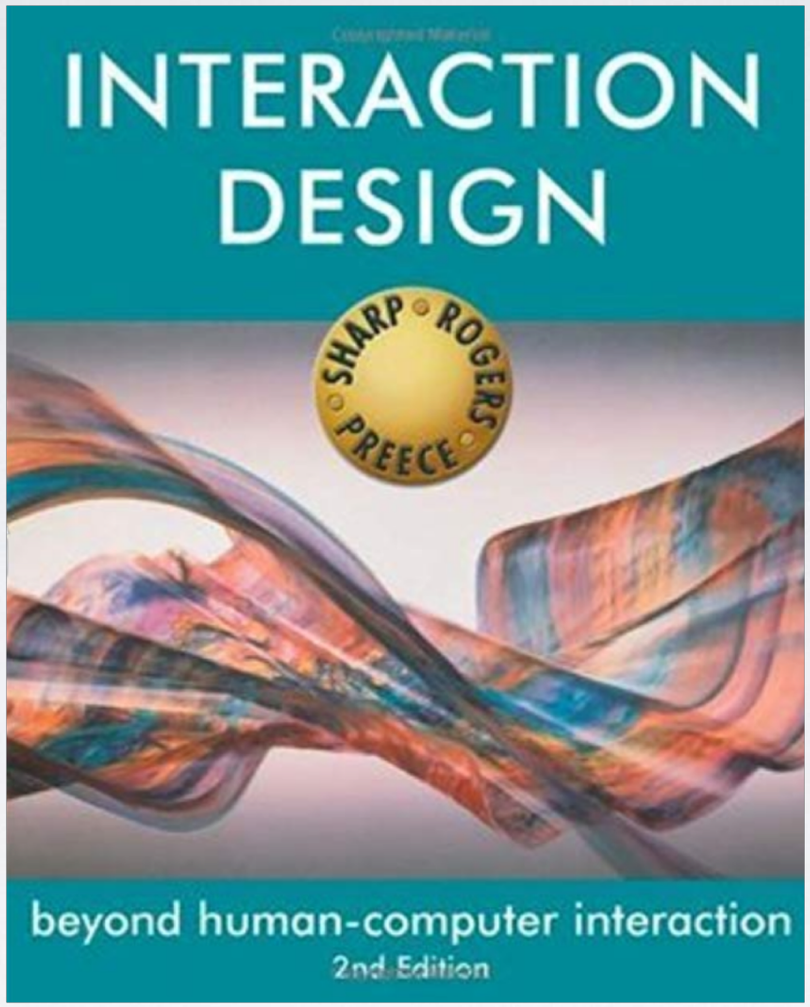
Possible to get stuck in conventional way of thinking
How relevant is this for a ground breaking innovation?

GATHERING REQUIREMENTS

Consider the proposed University News App again.
What activities would you engage in to elicit requirements?
Who would you talk to?
What observations would you engage in?
What existing products would you examine?

Existing news resources

Reading



Interaction Design – Preece, Rogers and Sharp
Chapters 7.6-7.7 and 10.1-10.5

GATHERING REQUIREMENTS

To provide students with an understanding of

- What Requirements Engineering (RE) is all about
- How RE relates to systems and software engineering
- Why successful RE is important in practice
- What constitutes good quality requirements (documents)
- Which RE techniques, processes and methods are available
- And with a working knowledge of
- How requirements can be elicited and documented
- How requirements can be modelled and validated
- How requirements changes can be managed
- How requirements reuse may be facilitated

Clearly, different skills are needed to move from:

what stakeholders want to a working software application!

What is a Requirement?

- New Shorter Oxford English Dictionary:
- "Something called for or demanded, a condition which must be complied with."
- IEEE Standard:
- "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."

Examples of Requirements

Requirements are things to discover before starting to build your product - examples:

•Things a product must do

•– The product shall produce an amended road de-icing schedule when a change to a truck status means that previously scheduled work cannot be carried out as planned.

•Qualities a product must have (fast, usable, secure, ...)

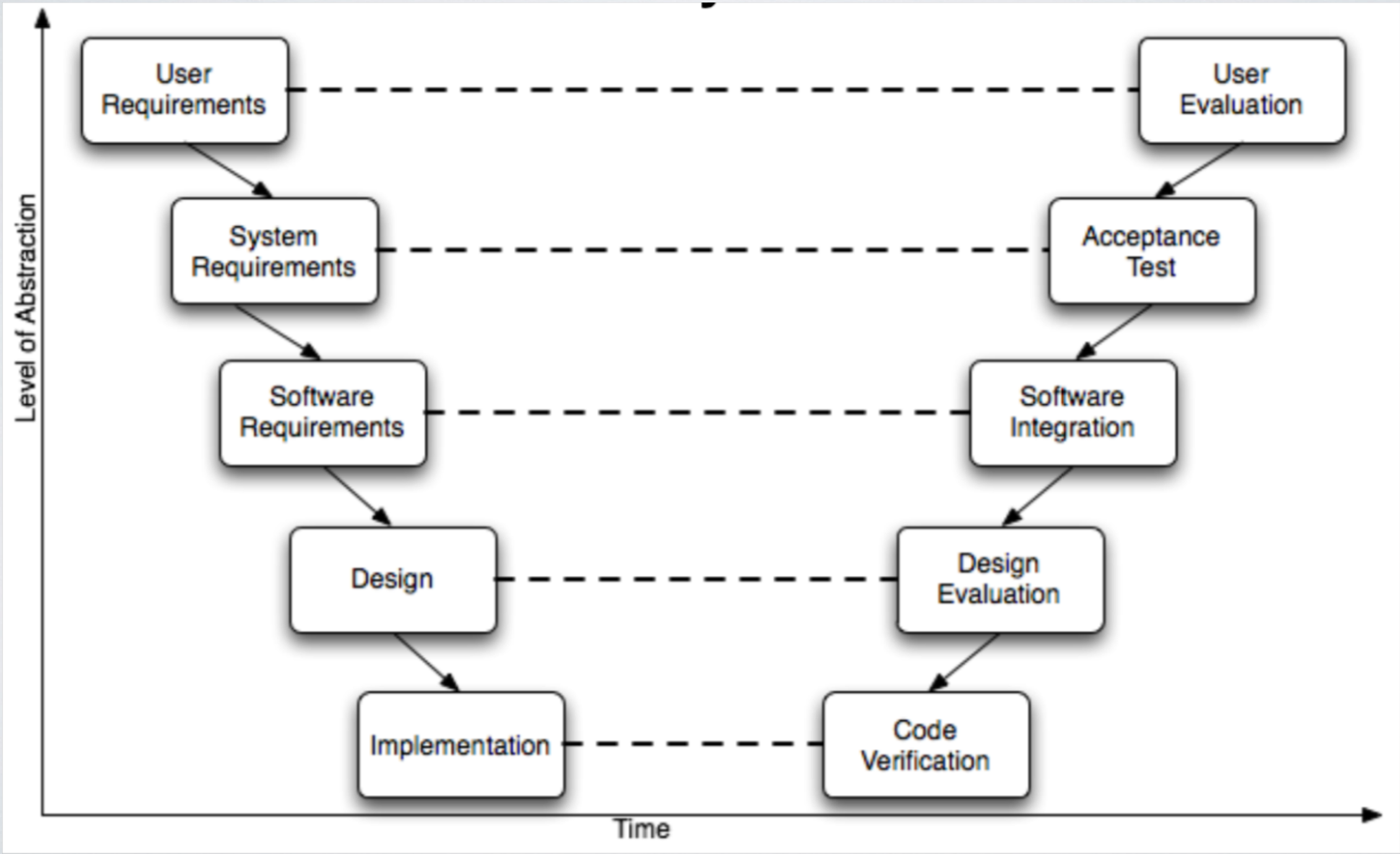
– The product shall use company colours, standard company •logos and standard company typefaces.

•Constraints a product must obey (standards, laws, ...)

•– The product will run on the Department's existing Linux machines.

•– The website must meet MoD safety standards

Requirements in the Engineering Lifecycle



What is Requirements Engineering?

"Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families."

- Real-world goals focus on the `why’ as well as the `what’
- Preciseness paves the way for analysis and validation
- Requirements change and are often re-used in later projects

Parnas, modified by McDermid, modified by Vickers:

“A requirements specification should provide individuals with everything they need to know to satisfy the relevant stakeholders... but nothing more.”

Hence:

Scope and view are not restricted to software alone

Requirements (“what”) is not design (“how”)

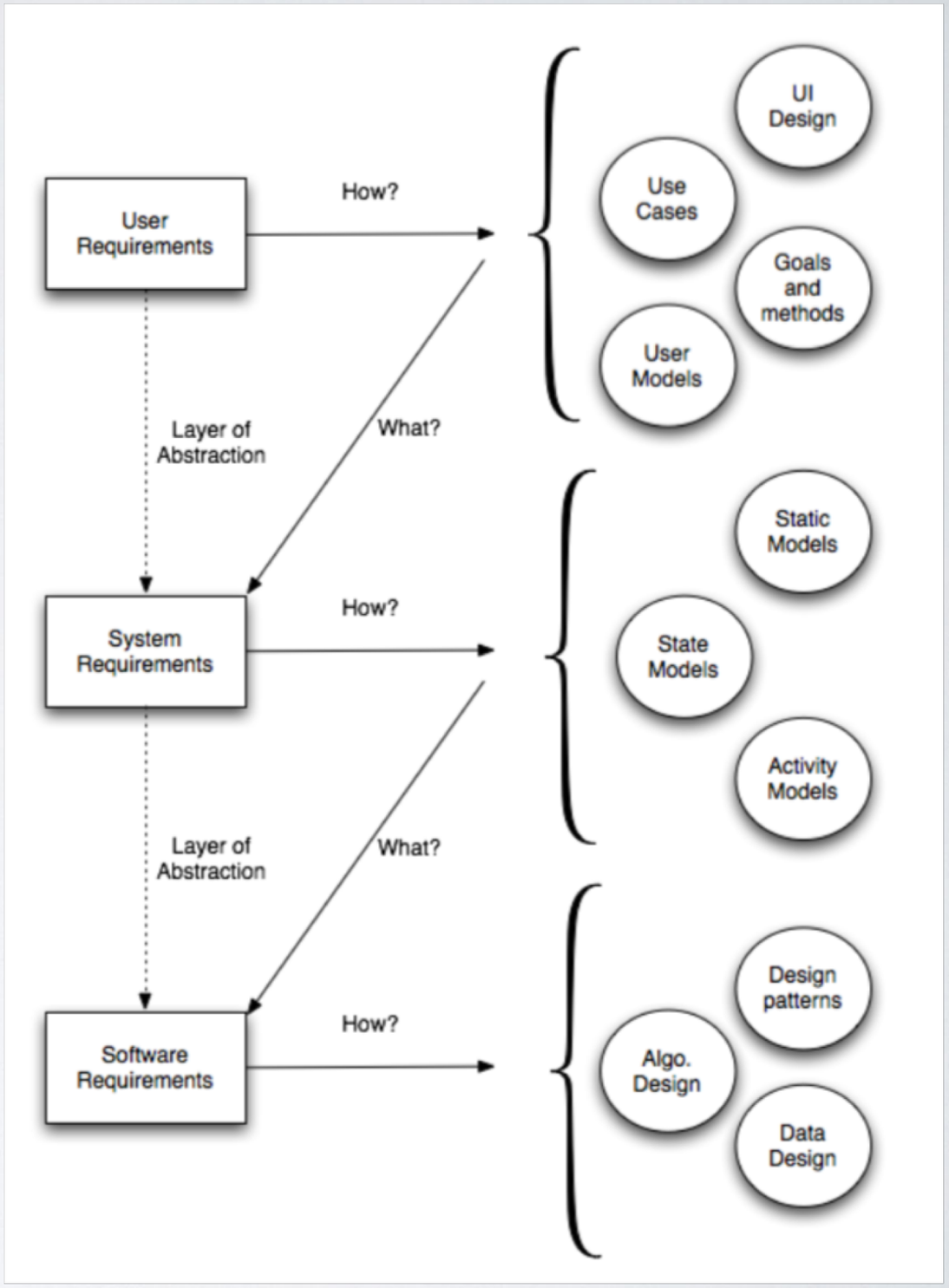
Design decisions belong with designers – May not always be different people!

RE must separate out the requirements from the design decisions

For large systems, it is hard to specify ‘what’ without specifying ‘how’.

The ‘how’ of one abstraction layer forms the ‘what’ for the next layer.

The lower you go in the layers of abstraction, the harder it is to trace back to make sure you are delivering what you originally wanted to deliver.



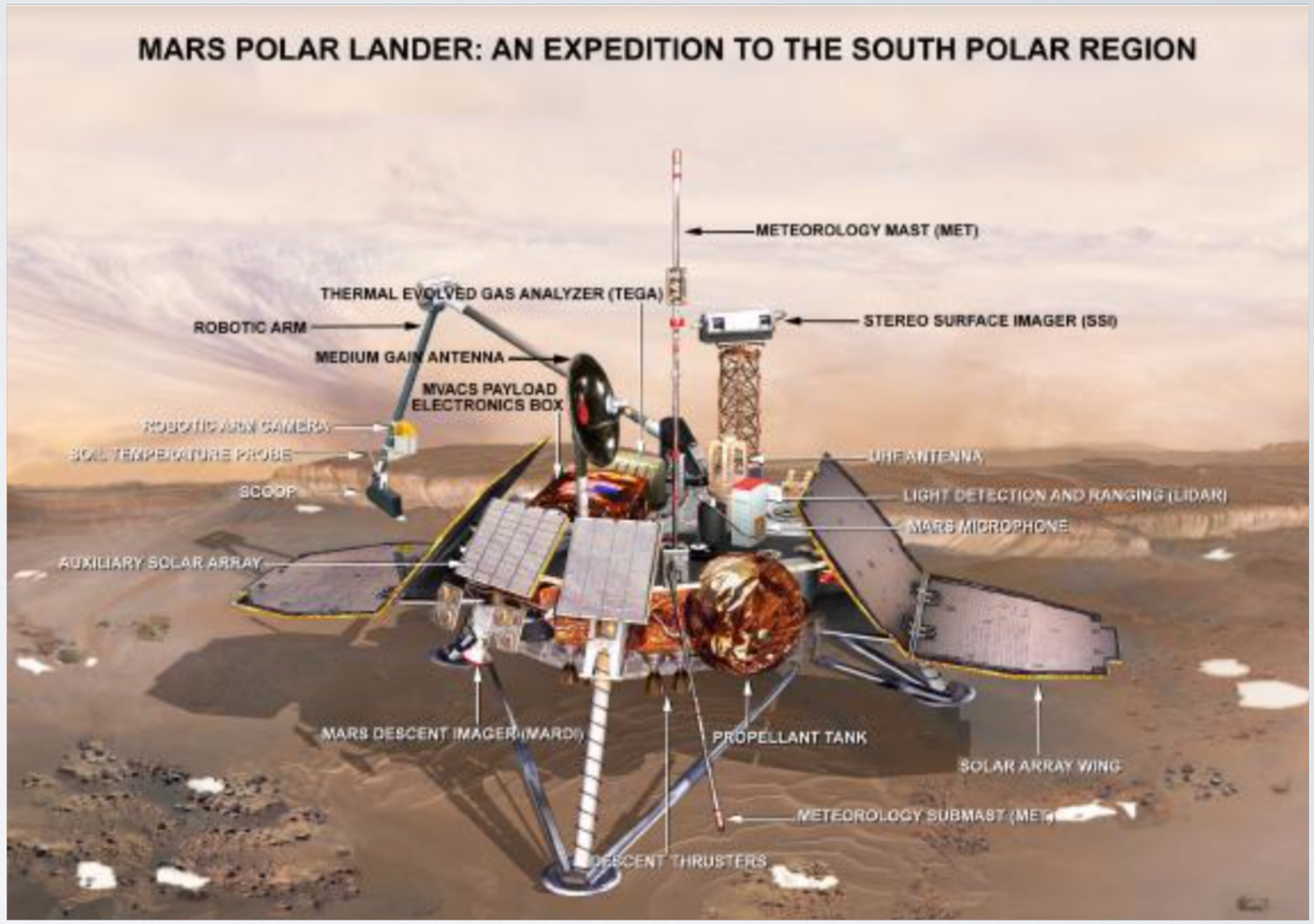
Example of Layered Abstraction

Requirements for a University:
Properties of the University
Location, size, campus, VC,
Departments, ...
•Properties of the Departments
•Names, Heads, staff & students,
•buildings, ...
•Properties of Departmental buildings
•– Location on campus, number of offices, ...
•Properties of Offices
•– Size, Internet connections, occupants,...
•Properties of Occupants
•– Names, titles, positions, ...

Mars Polar Lander

Mission:
Land near South Pole
Dig for water ice
Launched:

3rd January 1999
Arrived at & lost on Mars: 3rd December 1999



[JPL 1999]

What happened? Investigation hampered by lack of data: spacecraft not designed to transmit control information during descent
Most likely cause: noise on leg sensors caused early engine shutdown

Premature Engine Shutdown Scenario

- Cause of error:
  - Magnetic sensor on each leg senses touchdown
  - Legs unfold at 1500m above surface
    - Transient signals on touchdown sensors during unfolding
    - Software accepts touchdown signals if they persist for 2 timeframes
    - Transient signals likely to be this long on at least one leg
- Factors leading to the error:
  - System requirements ignored the transient signals
  - Engineers at code inspection did not understand the effect
  - Testing did not reveal error
    - Unit tests, based on software requirements, did not include transients
    - Integration test was conducted with improperly wired sensors
- Result of error: Engines shut down before spacecraft has landed
  - When engine shutdown software is enabled, flags indicated that a touchdown had already occurred
  - Estimated impact velocity 22m/s; normal touchdown velocity 2.4m/s

Easterbrook 2002

Questions Relevant to RE

How do we ...

•Gather requirements information?

– Elicitation

•Analyse this information?

– Modelling, prototyping & validation

•Get agreement about the requirements?

– Negotiation, conflict resolution

•Communicate the requirements?

– Natural & formal languages, documentation

•Keep the requirements up to date?

– Change management & traceability

This yields a roadmap for the RQE module ...

Nuseibeh and Easterbrook 2000

- Importance of Requirements Engineering
- Activities and processes
- Terminology

- **Software is complex**
  - It is invisible and abstract
  - It is highly modifiable since no fabrication step is involved
- Our society increasingly relies on all types of software
  - **Information Systems**
    - Software supports organisational work, e.g., payroll, customer records, accounting, ...
    - Software includes databases, standard applications as well as Internet applications
  - **Embedded Systems**
    - Software controls complex hardware systems, e.g., aircrafts, cars, industrial plants, cash machines, lifts, ...
- In a decade, 60% of the value of a car will be in its software

Today's Software is Often Money- Critical

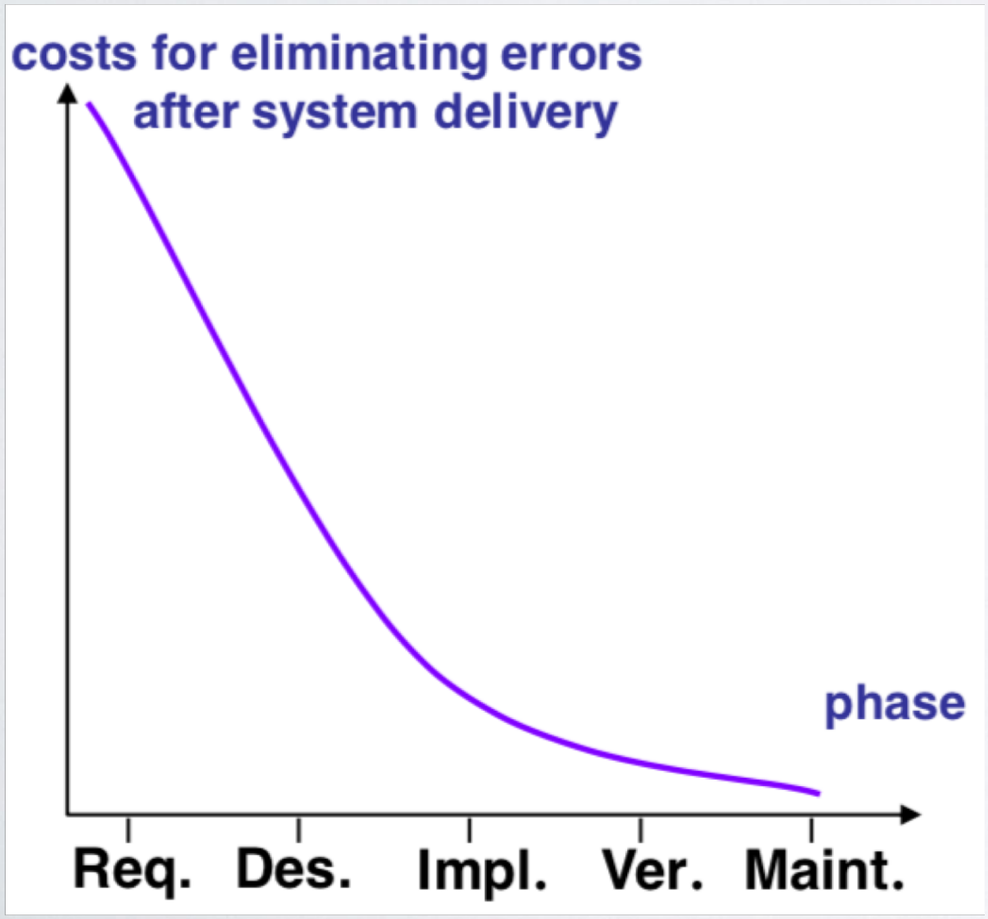
- Re-working defective software is expensive
- Motorola spends 60-80% of software budget on testing
- Extensive wastage on failed projects
- A 1997 U.S. study shows that $145 billion was wasted over 6 years on software that was never delivered

Case Study: The U.K. Passport Office Computer System:

- £240 million project
- Lost 10-year effort
- Wasted £12 million
- Destroyed 500 holidays
- Caused by a lack of validation and (thus) by many changing requirements

Today's Software is Often Safety-Critical

Certification costs are half of software costs (e.g., Boeing 777)
Defects are cheaper to remove the earlier they are found
Requirements defects are more likely to be safety-related



Voyager/Galileo project statistics:

- Coding faults:

6% (of overall faults!)

- Function faults:

71% (requirements problems)

- Interface faults:

23% (poor communication)

Early Modelling and Analysis is Important

RE is a technical activity

•Modelling and analysis techniques

– Semi-formal techniques are widely used today

• E.g., Unified Modelling Language (UML) and Object-Oriented Analysis (OOA)

– Formal techniques not yet widely adopted in practice • E.g., Software Cost Reduction (SCR)

•Systems analysis

– As used in the information systems world

•Systems theory and practice

– Relevant in the whole-system context

“The cost of good requirements gathering and systems analysis is minor compared to the cost of poor requirements.” [Robertson1999]

Early Modelling and Analysis is Not Enough

There is a need to
– Communicaterequirementstoeveryone
– Seek agreement from all stakeholders
– Understand the context of the system
– Understand the context of the development process –
Keep up-to-date as the requirements evolve
RE involves many non-technical disciplines...

Requirements Need a Process

- The development of requirements
- – Involves two or more individuals ... – ... co-operating to reach agreement – Consumes resources
- A process is needed to do this efficiently
- – Requirements engineering The RE process:
  - – Who are the participants?
  - – In which activities do they engage?
  - – What process is followed to coordinate the activities?

Participants/Actors in RE

• Stakeholders:

•– People or organisations who have a vested interest in the product, who will be affected by the system and who directly or indirectly influence the system requirements

•– Examples of direct stakeholders • Customers & clients

• Operators & users

•– Examples of indirect stakeholders • Developers

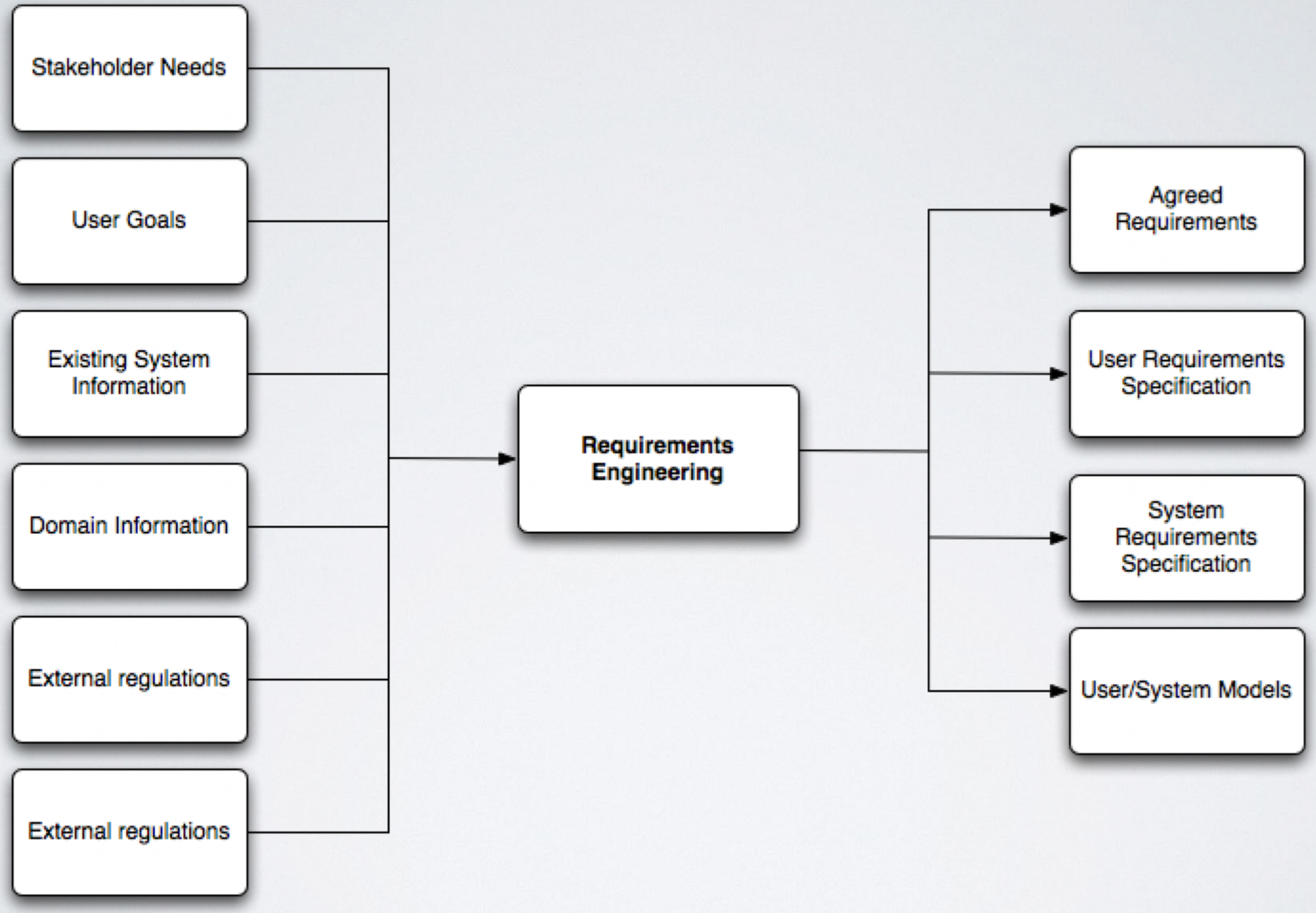
•• Regulatory authorities, professional bodies

•• Domain experts, technical experts

• Other actors (who are not stakeholders):

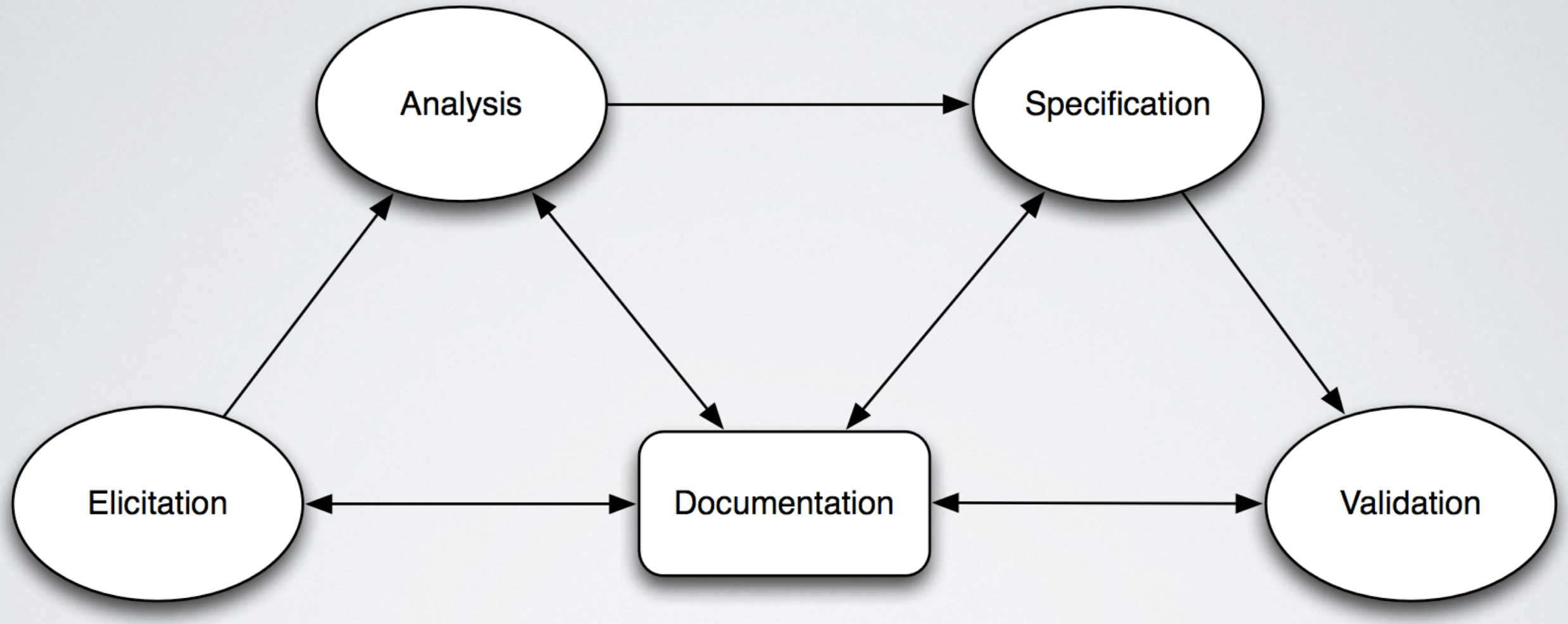
– Projectmanager,Requirementsengineer/analyst,Software engineer

Inputs and Outputs of the RE Process



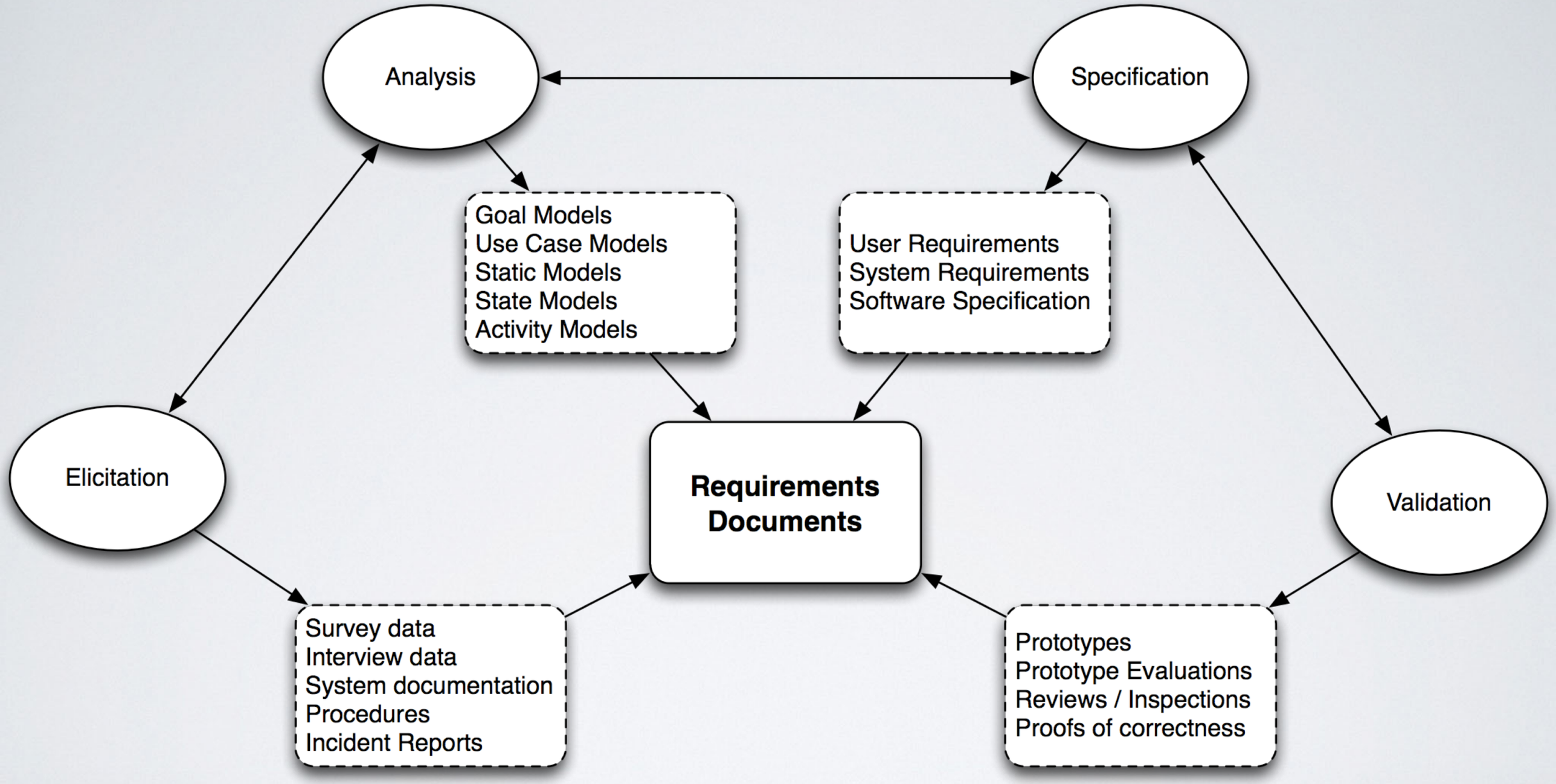
Kotonya and Sommerville 1998

An RE Process Model

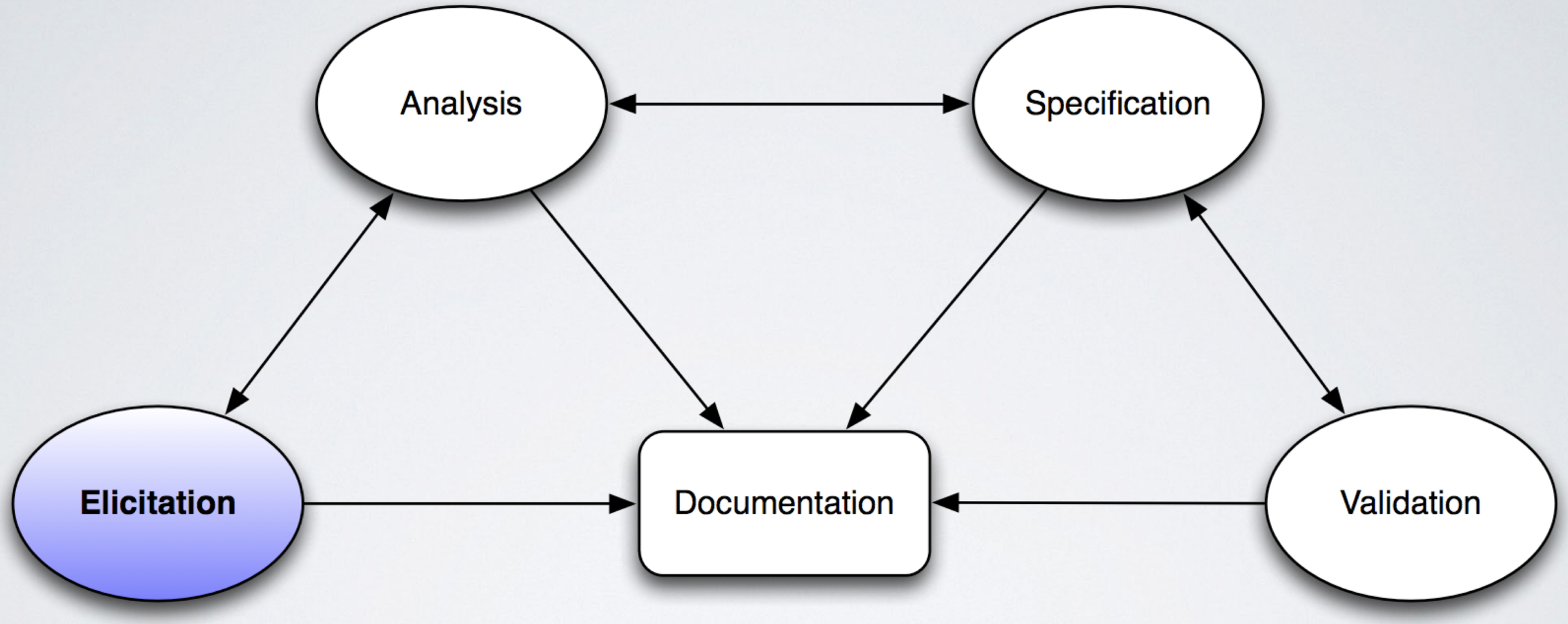


Sommerville 2005 & Van Vliet, 2008

A more detailed version



Elicitation



Requirements Elicitation

What it is:

– Collecting information to identify problems and opportunities – Finding out information about the software to built, including

• The application domain

• The environment in which it will be used Why it is hard:

– Thin spread of domain knowledge, tacit knowledge – Limited observability

– Training (lack of?) in techniques for elicitation

– Poor tool support for elicitation process

What techniques are used:

– Interviews

– Rapid Application Development (RAD) workshops – Scenario-basedmethods

Context of Requirements Elicitation

Start at a problem that needs solving, e.g.,
– A new business opportunity
– Dissatisfaction with the current state of affairs
Collect sufficient information to
- Identify the problem boundaries (What problem?)
- Understand the problem domain (Where is the problem?)
- Identify the stakeholders (Whose problem?)
- Identify the stakeholders' goals (Why is problem interesting?)
- Visualise some scenarios (How might software system help?)
- Identify constraints (When does problem need solving?)
- Identify feasibility & risk (What might prevent us solving it?)

- Become an expert in the problem domain

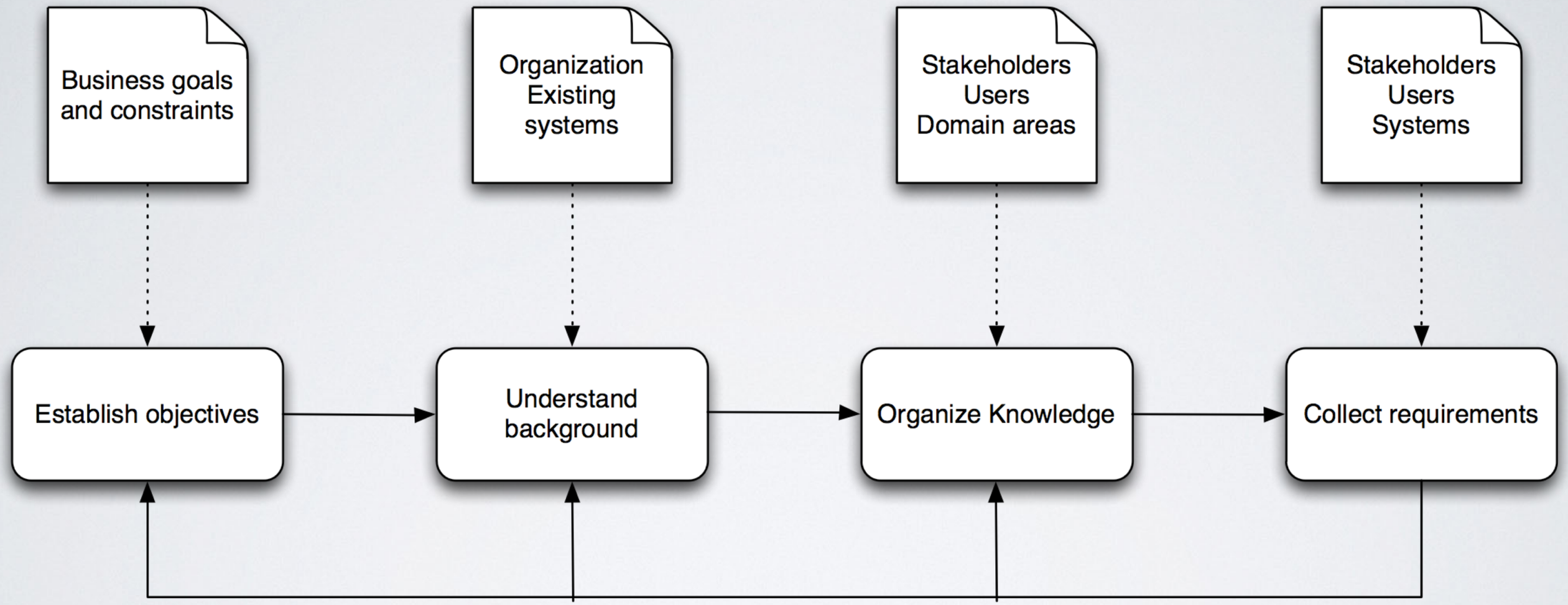
Stakeholders & Requirements Analyst

Stakeholders:
– Source of knowledge about the work
– Visionaries of what the work should be
– Evaluators of requirements analyst's ideas for the product

Requirements analyst:
– Translator of stakeholders' input into product specification
– Visionary of a new or better product
  - Observe and learn from stakeholders
  - Interpret stakeholders' work
  - Invent new or better ways to do the work
  - Record the results in requirements documents, analysis models, ...

Techniques are needed for eliciting requirements

An Idealised Elicitation Process



IceBreaker: A Case Study

Overview:

[Case study taken from Robertson and Robertson 1999]

IceBreaker uses various data to predict when ice will form on roads

It then schedules and dispatches trucks to treat the roads with de-icing material before the roads become dangerous

Background Information

Background (provided by the customer):

"Roads freeze in winter and icy conditions cause road accidents that kill people. We need to be able to predict when a road is likely to freeze so that our depot can schedule a de-icing truck in time to prevent the road from freezing. We expect a new system will provide more accurate predictions of icy conditions by using thermal maps of the district and the road temperatures from weather stations installed in the roads, in addition to the weather forecasts. This will lead to more timely de-icing treatment than at present which will reduce road accidents. We also want to eliminate indiscriminate treatment of roads which wastes de-icing compounds and causes environmental damage."

Step 1: Establishing Objectives

General goals of the business

"The Highways Department is responsible for maintaining all the roads in its county. One of its most critical tasks is to keep the roads free of ice during winter when icy conditions are likely to cause accidents."

Outline description of the problem

The customer statement seen before

Purpose of the system

"To accurately forecast road freezing times and dispatch de-icing trucks."

Advantage: "To reduce road accidents by forecasting icy road conditions."

Measurement: "Accidents attributed to ice shall be no more than 15% of the total number of accidents during winter."

Constraints, e.g., budget, schedule, interoperability, …

System Boundary

What is being developed?

Only the software …

A database for storing weather information

An algorithm for scheduling de-icing trucks

… or also the hardware

Weather stations and de-icing trucks

… or perhaps other services as well

Weather forecasts

Knowing the boundary is important

To identify the right stakeholders

To correctly estimate the development costs and risks

To decide on how to proceed

Step 2: Understanding the Background

The organisation where the system is to be installed

Organisational structure

E.g., do neighbouring Highway Departments interact?

The application domain of the system

De-icing of roads (how, when, how much, how often, …)

Scheduling of de-icing trucks

Existing systems which may be replaced

Weaknesses and strengths

E.g., current procedure for scheduling de-icing

Step 3: Organising Knowledge

Identify stakeholders

Roles in the organisation

Client: Saltworks Systems

Represented by Mr Mack Andrews, Chief Executive

Customer: North Yorkshire County Highways Department

Represented by Jane Torville, Director

Discover the organisation's priorities

“Safety first”

Analyse domain knowledge

Existing system documentation, textbook information, …

(filter out the irrelevant parts)

Acquiring Knowledge of Requirements

Reading documents

User manuals, e.g., manuals of weather stations to be used

Supporting literature, e.g., journal article on *"Cost-Effective Snow and Ice Control in the 21st Century"*

Accessing people's knowledge via

Interviews, questionnaires, …

… see "elicitation techniques" to be discussed later

Observing phenomena

Objects

People's behaviour

Types of Knowledge

Behaviour

Can be observed

Process

Can be discovered from documentation, or

Is explicitly known

Data

Information

Some information is explicitly known

Collecting Requirements

- Traditional elicitation techniques
  - Background reading
  - Interviews
  - Questionnaires/surveys
  - Observing behaviour
  - Reusing requirements
- Collaborative elicitation techniques
  - Brainstorming (Thought-showers?)
  - Rapid Application Development (RAD) workshops
  - Prototyping *(to be discussed in two weeks; also a validation technique)*
- Model-based elicitation techniques
  - Scenario-based techniques, use cases

Background Reading

- Familiarisation with the organisation being investigated
- Sources of information:
  - Company reports, organisation charts, policy manuals, …
  - Job descriptions
  - Documentation of existing system
- Advantages:
  - Getting an understanding of the organisation
  - Helps preparing for more efficient meetings with people who work in the organisation
- Disadvantages:
  - Time-consuming
  - Lots of irrelevant details

Interviews

- Types of interviews:
  - Closed interviews, with a pre-defined set of questions
  - Open interviews, with no pre-defined agenda
- Good for eliciting non-tacit knowledge:
  - General description of work
  - Explicit procedures
  - Difficulties faced
  - Critical incident reporting
- Not useful for:
  - Concealed knowledge
    - Where actual practices deviate from laid-down procedures
  - Tacit knowledge (critical incident technique helps with this)
  - Domain knowledge
    - Terminology may be unfamiliar to interviewer
    - Knowledge that is taken for granted (semi-tacit)

Interviewing Tips

- Start off by setting the interviewee at ease
  - E.g., talk about the English weather
- Ask whether recording the interview is possible
  - To be able to recall all details
- Ask simple questions first
  - E.g., “How long have you worked in your present position?”
- Follow-up interesting leads
  - E.g., “Could we pursue what you just said a little further”
- Ask open-ended questions last
  - E.g., “Is there anything else you would like to add?”

Questionnaires

Choosing adequate sample size and selection
Avoiding ambiguous questions
Not everyone is answering the same question
Avoiding leading questions
People naturally tend to answer what you are asking about
Avoiding loaded questions (*plurium interrogationum*)
Questions that presupposed conditions “Is your brother in the army?”

Avoiding open-ended questions
Very hard to analyse
Prototyping and testing questionnaires before using them!

QUESTIONNAIRES

- Advantages:
  - Quickly collect information from large numbers of people
  - Can collect some qualitative data
  - Can collect attitudes, beliefs, characteristics, …
- Disadvantages:
  - Presupposed categories provide little context
  - Time consuming to examine, clean and analyse
  - Time consuming to construct (requires high precision when compared to other methods)
  - Little room for users to convey their needs

OBSERVING BEHAVIOUR

Techniques:

Direct observation of work practices, or even apprenticeship

Recording work practices (video) and analysing the record (discourse analysis etc.)

Spending an extended time in the environment, until accepted as part of the workplace (ethnography)

Good for:

Understanding behaviour that is difficult to describe

Understanding how people behave, not how they say they do

Difficulties:

Time-consuming

Does not directly address requirements

May not pick up exceptions

BRAINSTORMING

A quick method for generating new ideas

Technique:

Invite participants from a wide range of disciplines and a broad range of experiences

Be imaginative and generate as many ideas as possible

Evaluate ideas after the meeting, not during the meeting

Advantages:

Focus on creativity

Quick

Disadvantages:

Many ideas might turn out to be ‘rubbish’

Many half-formed ideas (need further investigation)

Rapid Application Development Workshops

Aims of Rapid Application Development (RAD):
Eliciting domain knowledge
Reaching consensus on requirements
All this, rapidly

Duration of a RAD:
Within one day for the workshop itself
Contrasts with weeks on other forms of elicitation

RAD method:
Facilitated by trained unbiased facilitator (not a stakeholder)
All significant stakeholders must attend the workshop
Workshop members must have authority to agree conclusions

Advantages:

If successful, ensures rapid completion of the process

May bring out different aspects of domain knowledge

Through interaction of different points of view

Disadvantage:

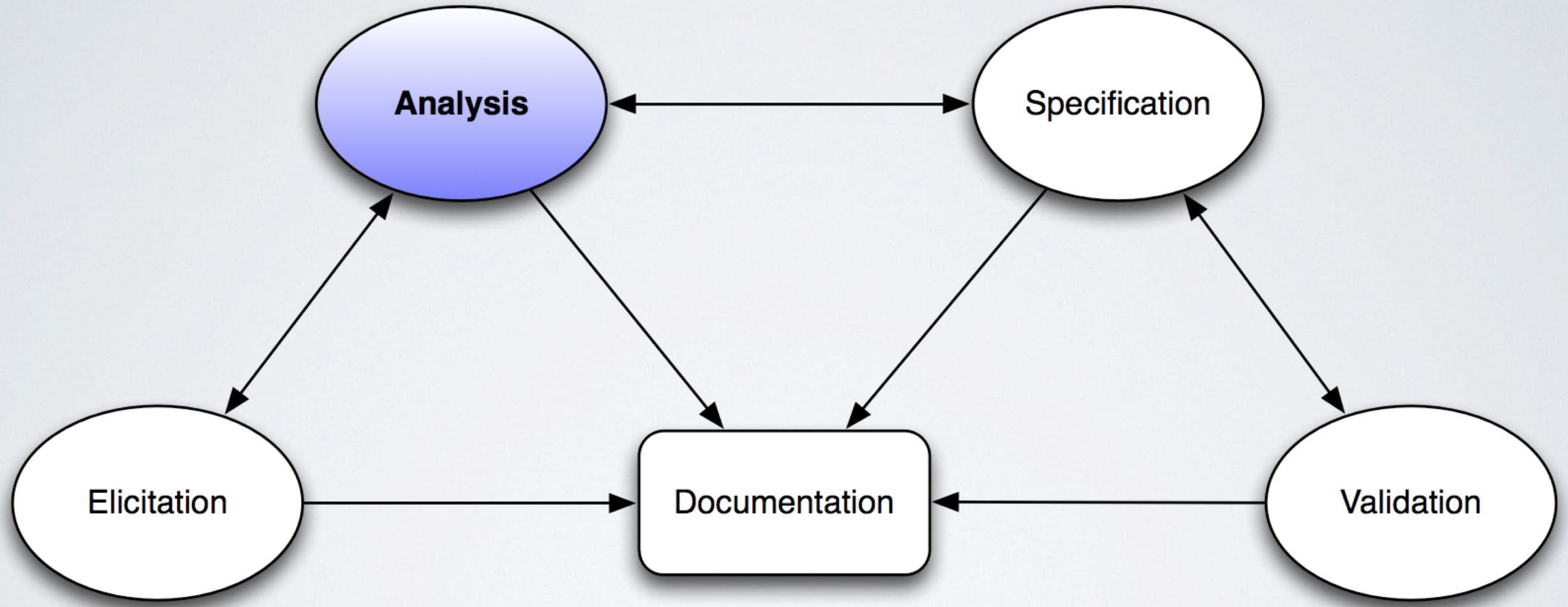
Assumes that all necessary knowledge and expertise is represented

Clash of cultures could result in failure

For more details:

I. Graham. “Requirements Engineering and Rapid Development”. Addison Wesley, 1998.

Requirements Analysis



What it is:

Building models of requirements that are amenable to evaluation of its properties

Identifying conflicts between requirements (stakeholders)

Why it is hard:

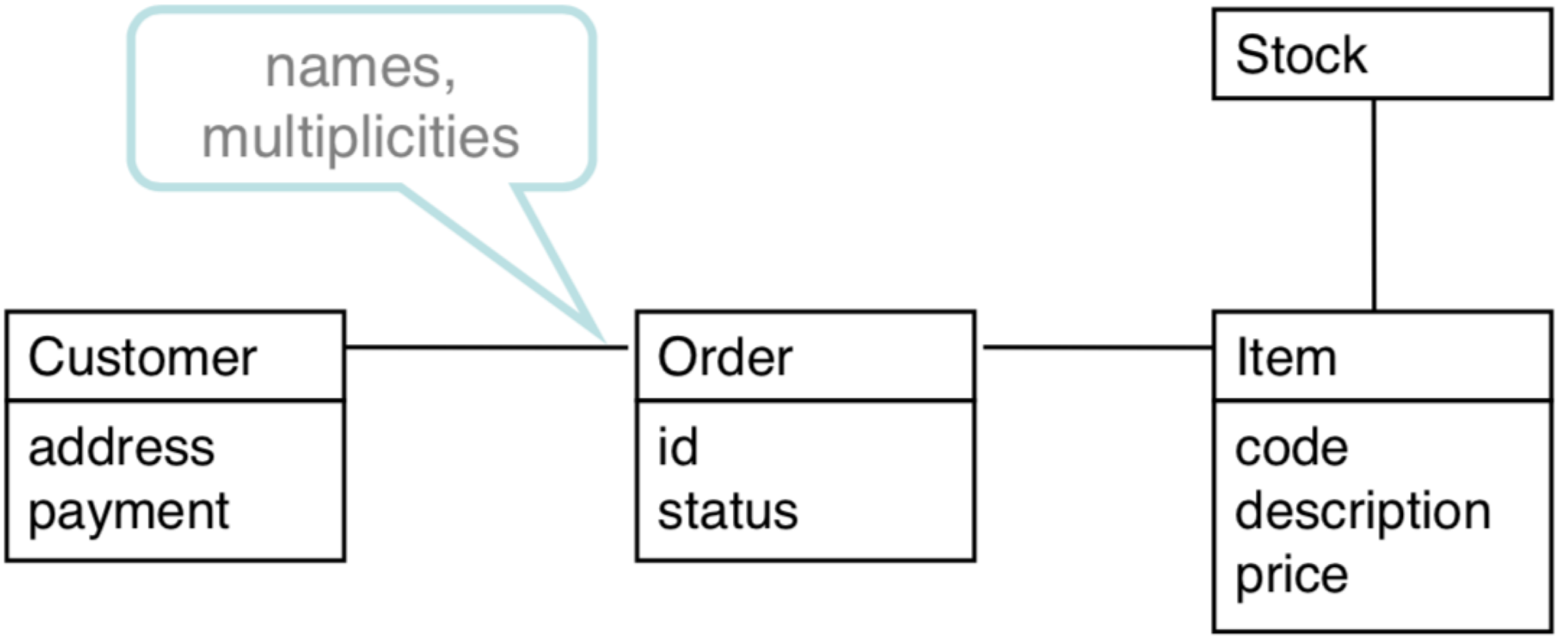
Formally modelling natural language requirements is an art form

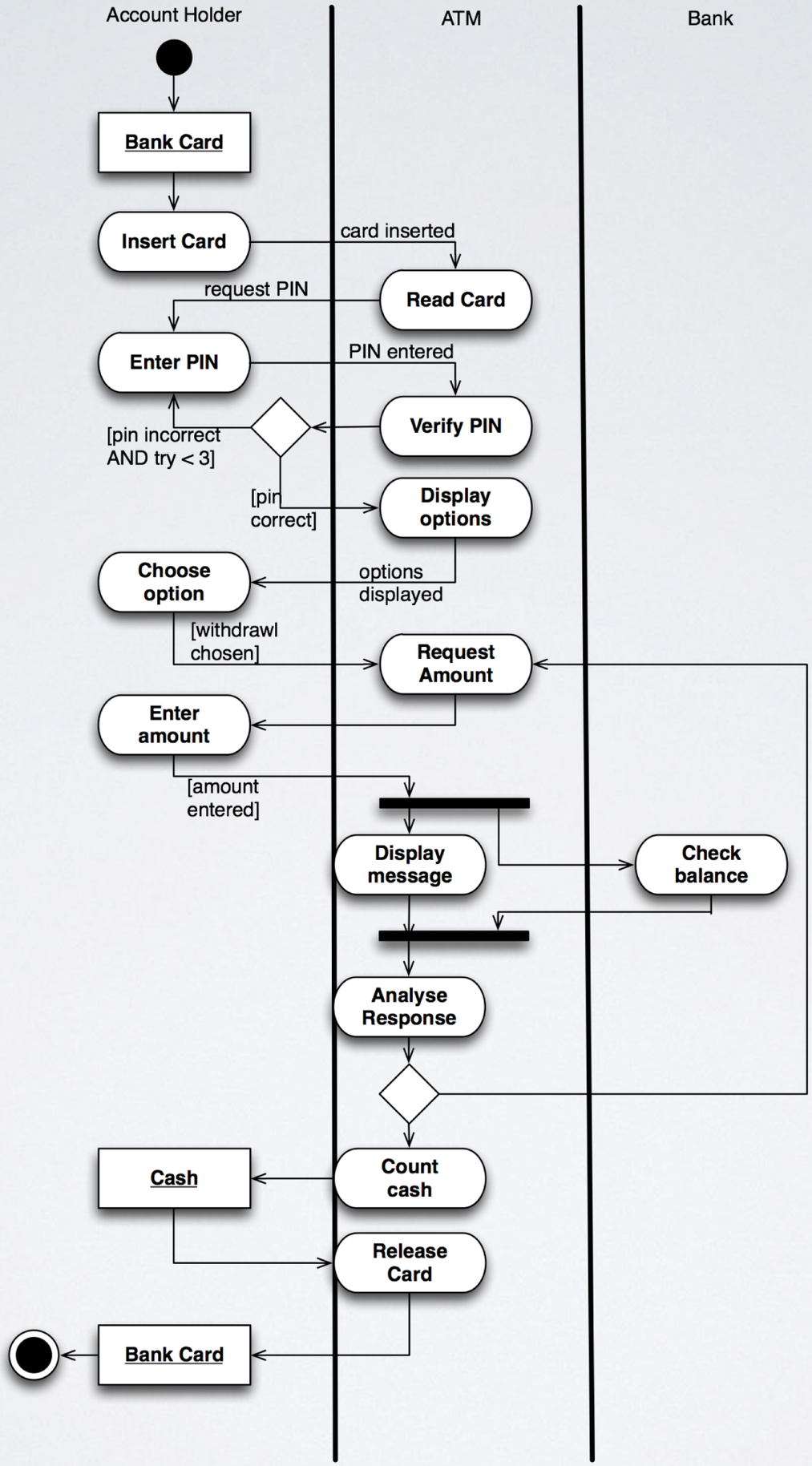
Many requirements conflict, especially when many stakeholders are involved

What techniques are used:

(Diagrammatic) modelling languages, e.g., UML (Unified Modelling Language), SCR (Software Cost Reduction) tables, …

Use Case example: Customer places an Order




Account Holder
ATM
Bank
Bank Card
Insert Card
card inserted
Read Card
request PIN
Enter PIN
PIN entered
Verify PIN
[pin incorrect
AND try < 3]
[pin
correct]
Display
options
Choose
option
options
displayed
[withdrawl
chosen]
Request
Amount
Enter
amount
[amount
entered]
Display
message
Check
balance
Analyse
Response
Cash
Count
cash
Release
Card
Bank Card

Summary

Sequence and activity diagrams give you a view of the interactions between objects.

Can provide an overview of where breakdowns can occur.

Very good for locating missing conditionals that could lead to system failure.
Extremely complex diagrams suggest that you may need to simplify or re-factor.

Assume you have data ...

What do we do with all of this data that we have collected?

We can analyse it using a variety of quantitative and qualitative methods

We can model it in terms of what the users want to do.

– Goal based requirements analysis

– Use cases / scenarios

There have been many attempts to describe what people do with systems:

Hierarchical task analysis (Dekker)
– Model of tasks that agents perform

Cognitive models of action (Neisser, Norman)
– Represents how people act on a goal
– Cognitive modelling

So how does it help?
It unites some of the human ideas with some of the system ideas
It also provides a different view of a system

What are goals?

Prescriptive statements of future intent;
often a single statement.

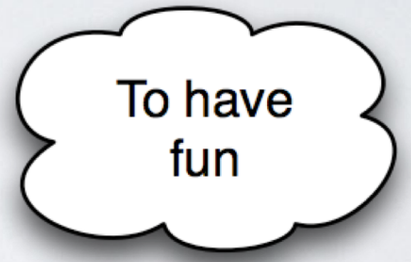
Goals may be defined at different levels of analysis:

– Get money from the ATM
– Prevent aeroplanes from colliding
– Generate enough power to meet demand
– Maximise business revenues and profits
– Have fun

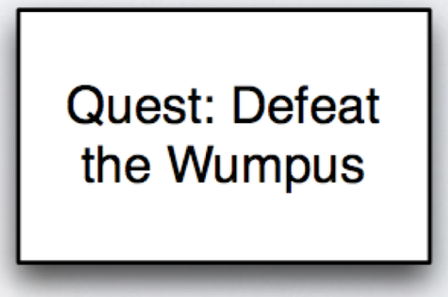
What do we have in systems?

- **Hard Goals**: Goals we can measure, quantify, and describe in its entirety

- **Soft Goals**: Something that we know we need but cannot describe fully (may be comprised of several hard goals)



- **Resource**: Something that can be used to achieve goals by an actor in the system.



- **Plans**: How actors will execute actions to achieve goals in the system.

Hard vs. Soft Goals

The designer knows exactly how to measure whether an actor succeeded in their goal:

If Healer != Heal Tank then

Tank takes too much damage

Tank dies

Everyone dies

Monster does victory dance

How can we define fun?

Ask the player:

Engaged in the game

Immersed in the game

Desire to keep playing

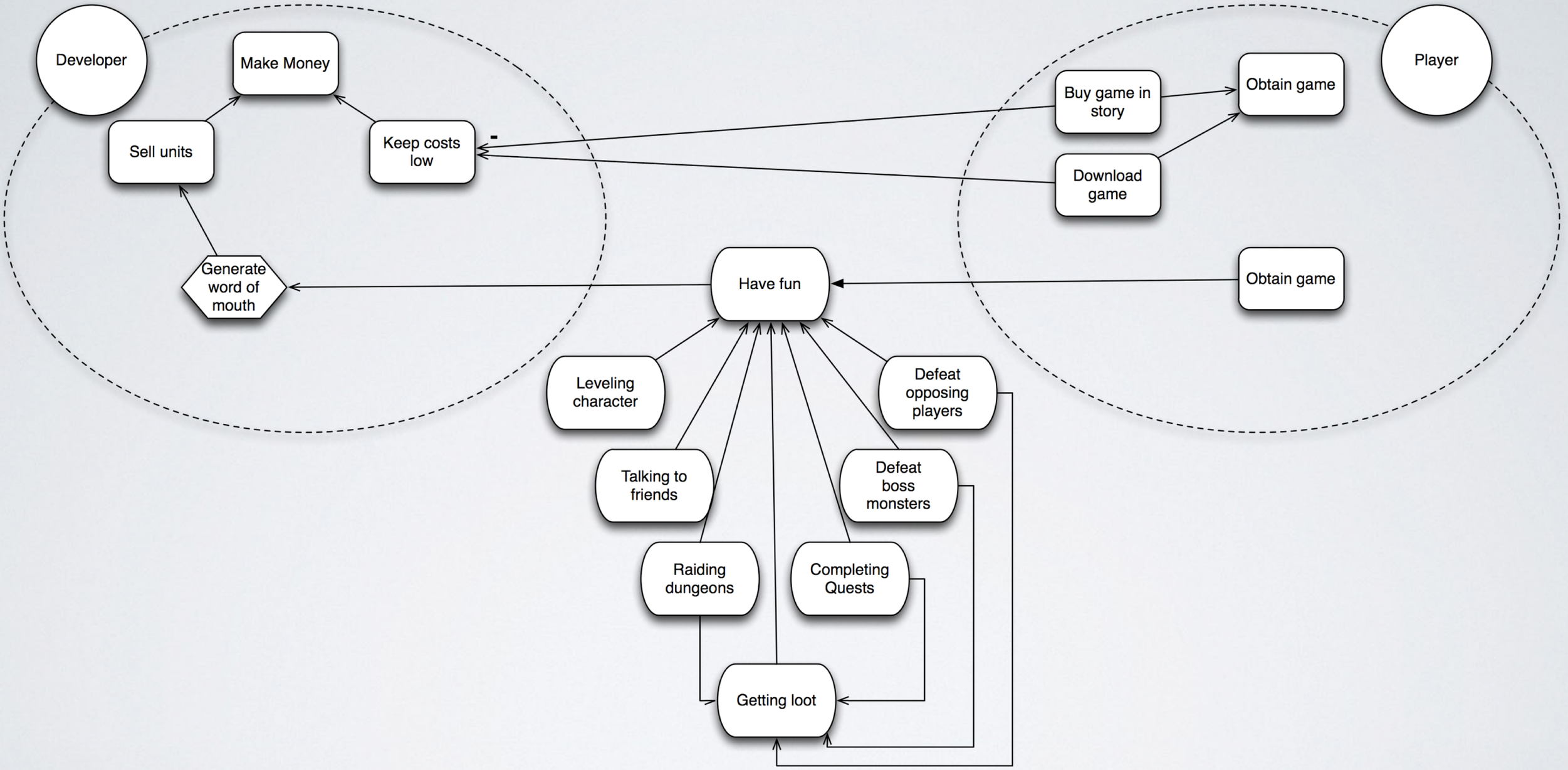
Number of levels achieved

Ask the developer:

Length of time played

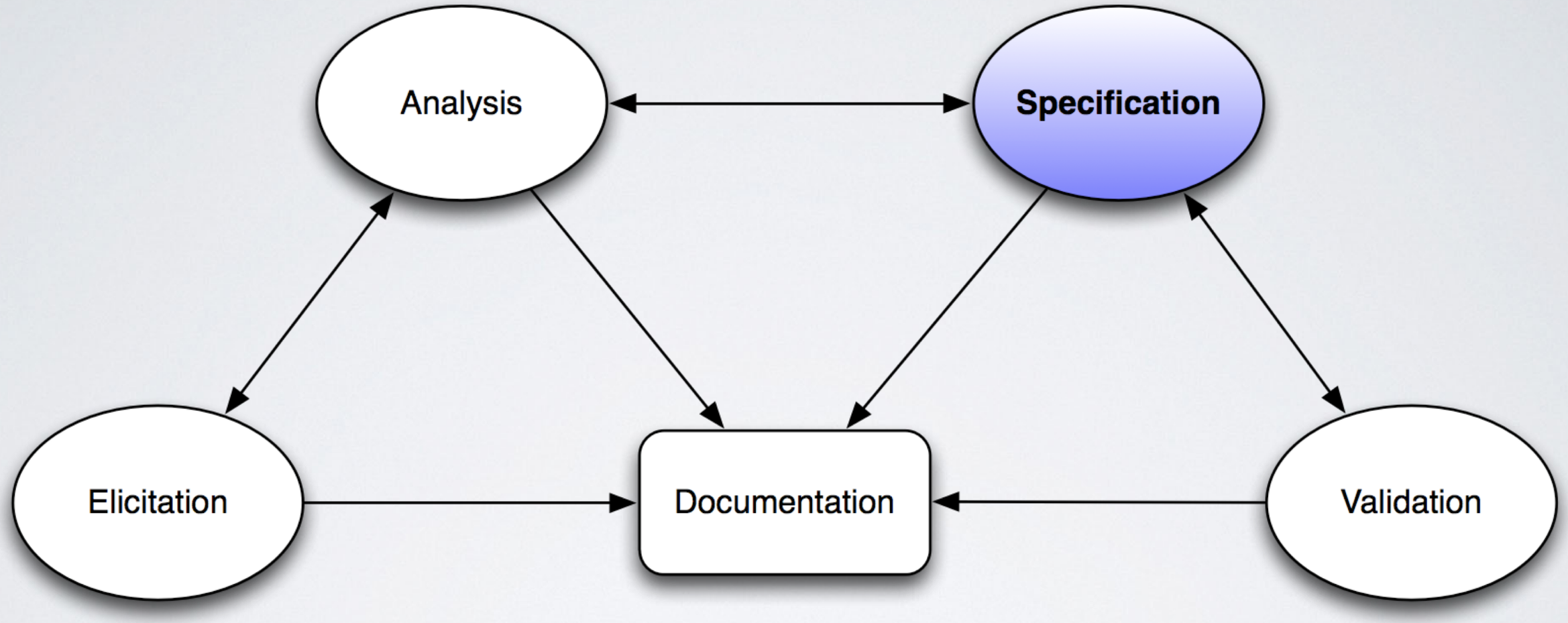
Number of levels achieved

Challenges overcome


Developer
Make Money
Sell units
Keep costs low
-
Generate word of mouth
Player
Buy game in story
Obtain game
Download game
Obtain game
Have fun
Leveling character
Talking to friends
Raiding dungeons
Getting loot
Completing Quests
Defeat boss monsters
Defeat opposing players

Goal modeling provides a view of what stakeholders want their agents to do in the system.

Specification



What it is:

Description of what users need to be able to do with the system

Description of what the system must do for the stakeholders involved.

Description of the qualities that the system user/system functionality must have.

Why is it hard?

Correctness in specification is a hard problem.

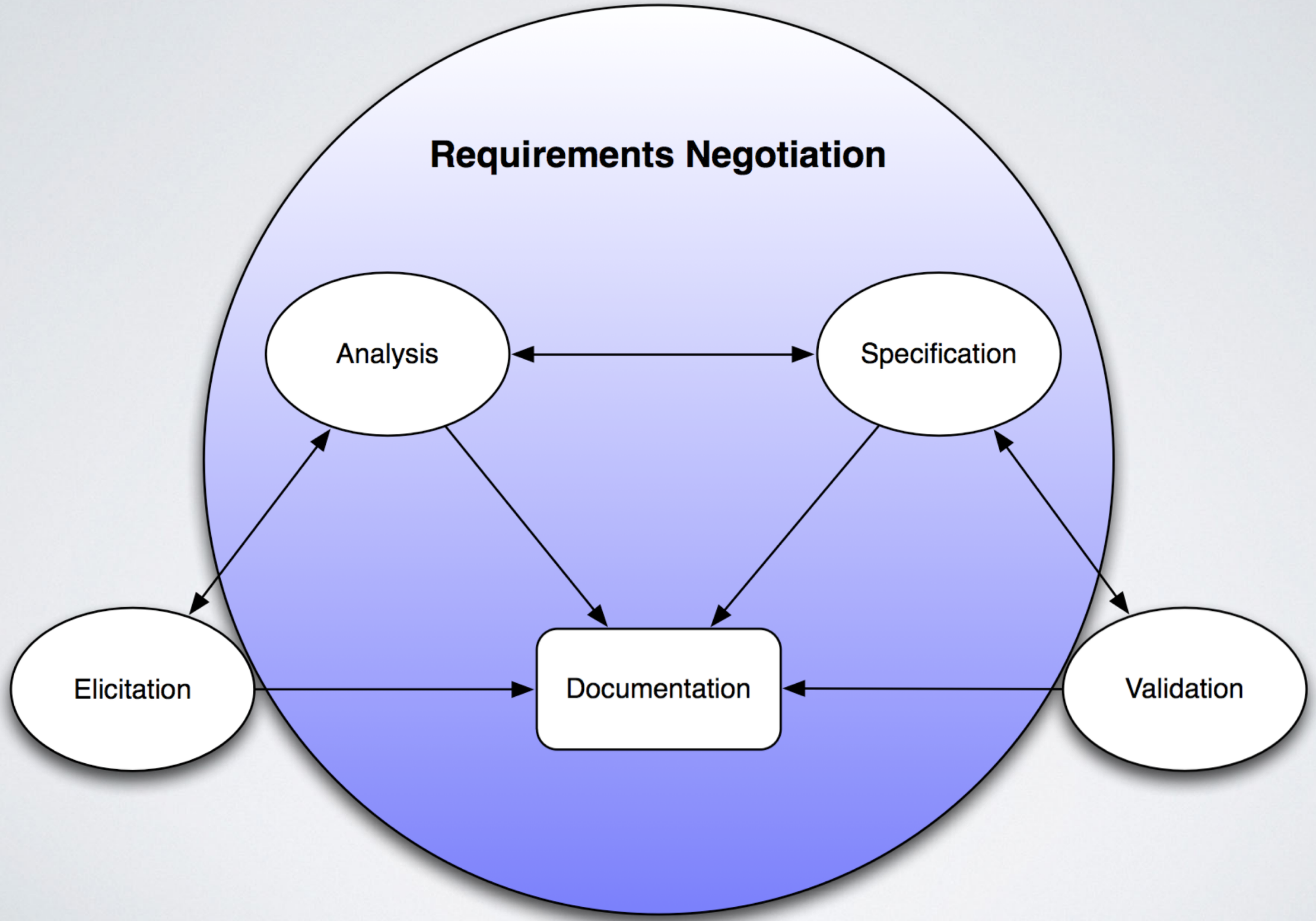
Completeness is impossible to check.

What techniques are used?

Requirements matrices

Template guided requirements

Requirements Negotiation



Requirements Negotiation

What it is:

When stakeholders have conflicting requirements, "getting to yes" can be a challenge

Why is it hard?

Negotiations require deep understanding of the domain and the stakeholders.

Negotiations require identification of real requirements vs. "fake requirements".

Some conflicts may never be resolved.

What techniques are used?

Consensus building

Majority rule

Appeal to Authority

Conflict Resolution

How to resolve detected conflicts?

– In a review meeting: do not resolve conflicts but record them!

Sources of conflict are often non-technical

– Personal preferences, values and beliefs
– Control over resources

Methods for settling conflicts:

- Co-operative methods: Negotiation and education
- Competitive methods: Combat coercion and competition
- Third party methods: Arbitration and appeals to authority

Negotiation to Resolve Conflicts

Requirements analysis **always** throws up problems

Stakeholders usually have conflicts about
– Specific, contradictory requirements
– Priorities between requirements

Negotiation is needed
– To define compromises
– To reach agreement

Procedural approach
– Informal negotiation: simpler, less confrontational
– Formal negotiation: negotiation meeting

Negotiation: Decision Makers & Information

Decision makers:

– Requirements engineer facilitates between stakeholders

Sometimes a professional facilitator is needed

– Only the stakeholders make the business decisions

Stakeholders at negotiation must have authority to change/ prioritise/concede requirements

Information:

– Clarify the facts, and ensure everyone is fully informed

Each stakeholder will initially have only a partial view of the system

– Ensure that the goals lying behind the requirements are clear

These goals need to be met, possibly by different requirements

Negotiation: Discussion

Is it a simple misunderstanding of goals?
– This should have been caught earlier, but...
– ... better late than never (or during implementation)

Is it an overspecification of a goal?
– Explore alternative non-conflicting requirements
– Make sure that the new requirements are not in conflict, too!

Is it a "gold plated" goal?
– Get stakeholder to admit it is not business-critical
– Conduct a cost/benefit analysis

Negotiation: Discussion

Is it a genuine conflict?

Beware: Some requirements cannot be negotiated
Example: safety standards

What if non-negotiable requirements conflict?

Negotiate with the safety authority
May be genuinely impossible to agree

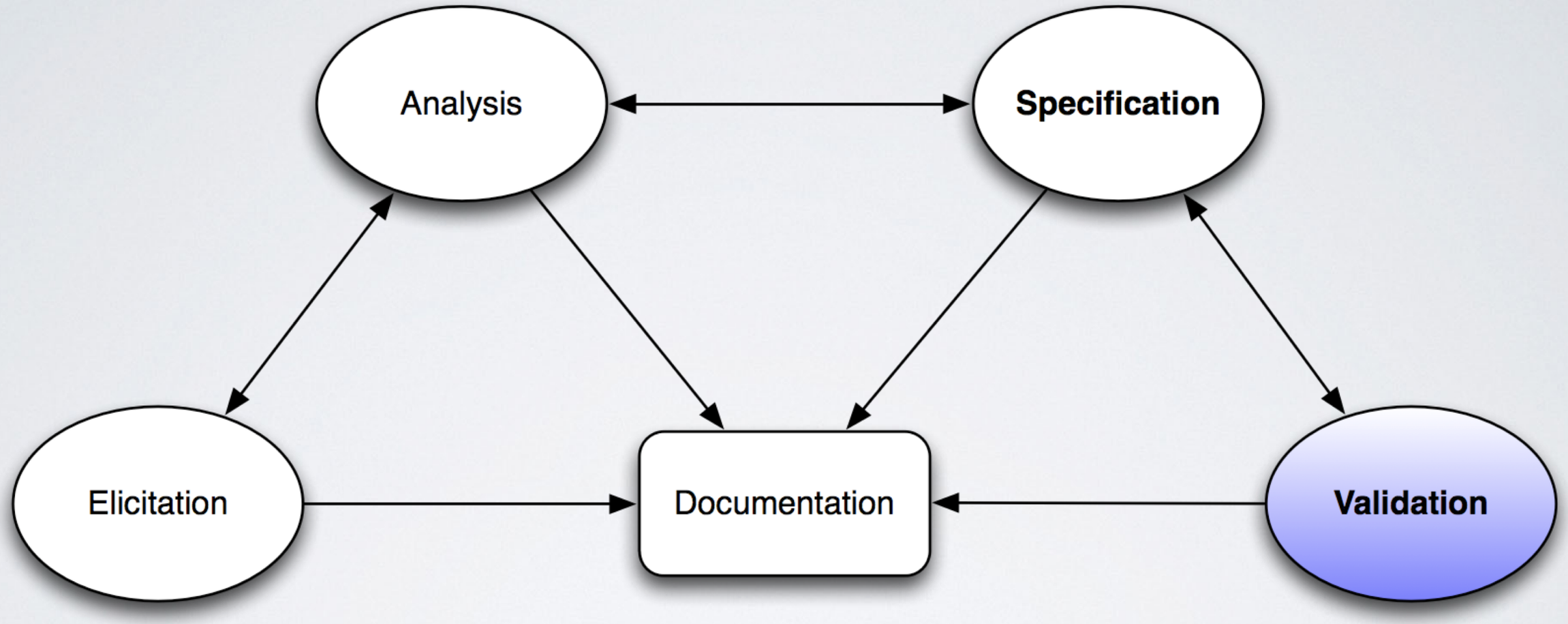
Negotiation: Resolution

Ensure the agreed action is clearly understood by all stakeholders
– Change the conflicting requirement(s)

Based on re-examination of goals, or result of bargaining
– Change the conflicting priorities (possibly to zero!)
Update the requirements
– Include the reasons behind the new version
Avoids having to go round the same loop again, later on!
The situation may change later, allowing re-evaluation

If the stakeholders cannot, or refuse to, compromise
– May have to go up a level of authority
Last resort only: better to get “buy-in” from all stakeholders

Requirements Validation



What it is:

Checking the requirements documents to make sure that the right system is built

Why it is hard:

There is no other formal document to check requirements documents against

Getting the requirements wrong can be very costly

What techniques are used:

Informal techniques: Reviews, inspections, walkthroughs

Semi-formal techniques: Prototyping, animation

Formal techniques: Formal Methods

Approaches to Validation

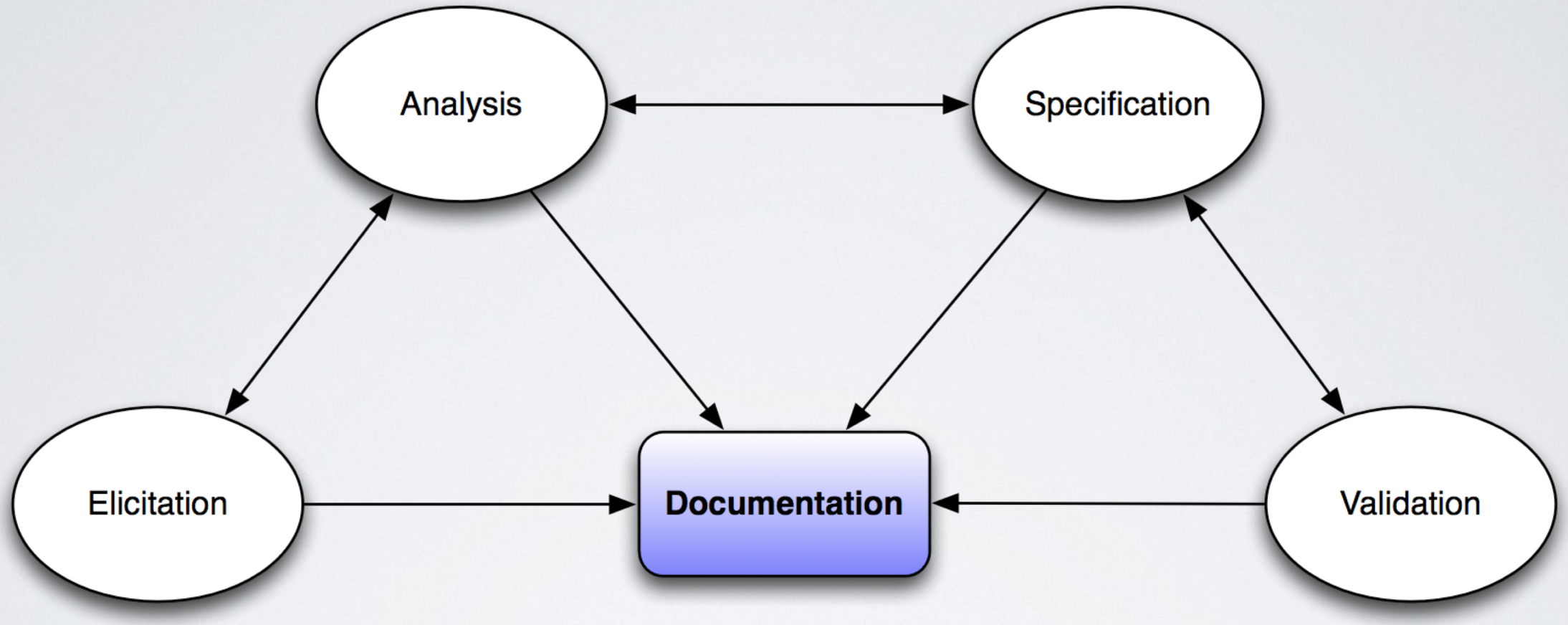
"Soft" (non-mathematical) approaches:

- Reviews & inspections
- Prototyping & testing

"Hard" (mathematically rigorous) approaches:

- Formal Methods
- Model simulation

Requirements Documentation



What it is:

Requirements Documentation is often the one time to record valuable information (frequently as the basis of a legal contract)

Requirements documents are the key for communicating requirements

Why it is hard:

Requirements documents need to be communicated across (contractual & language) boundaries

What techniques are used:

Employing a requirements engineer to facilitate the process and to avoid common errors

Controlled natural language

Automatic generation

Recognise the Diversity of Software Projects

- Requirements for large embedded systems are quite different from those for small information systems
- Complex/Simple, long time-scale, budget
- Requirements for interactive technologies are different than those for non-interactive systems
- More/less knowledge of user goals required; less/more autonomy in the system;
- Since the problems are diverse, so will the solutions be
- Different approaches to requirements elicitation
- Different approaches to requirements analysis
- Formal methods, MDA, UCD
- Different distances between requirements and software

Requirements Document Must Not Include

Project development plans

Staffing, schedules, methods, tools, ...

Lifetime of the requirements document spans until the software is made obsolete

Lifetime of development plans is much shorter

Product assurance plans

Verification, validation, testing, quality assurance, ...

Different audiences and different lifetimes

Designs

Requirements and designs have different audiences

Analysis and design are different areas of expertise

Structure Specific Requirements

3.1 External Interface Requirements

User interfaces, hardware interfaces, software interfaces, communications interfaces

3.2 Functional Requirements

Introduction, inputs, processing, outputs

3.3 Performance Requirements

3.4 Logical Database Requirements

3.5 Design Constraints

Standards compliance, hardware limitations, ...

3.6 Attributes

Availability, security, maintainability, transferability, ...

Specific Requirements

Organise specific requirements section by:

System mode
E.g., operational, training, emergency, ...

User class
E.g., crew, passengers, maintenance workers, ...

Objects
E.g., patients, sensors, nurses, rooms, ...

Features
E.g., local call, call forwarding, conference call, ...

...

Summary

- Documentation is the link through all stages of the requirements process.

- Traditional software development environments have many types of documents which must be managed.

- Often will be dictated by a standard, but every company has its own procedures and variations.

Types of RE Projects

Where to move within the domain model in practice depends on:

Source of requirements

Customer driven
– specific software for a specific customer type

Market driven
– software to be sold in the market

Hybrid
– specific customer first, market software eventually

User driven
– specific software for a group of users

Nature of the product
One-off (‘bespoke’) vs. packaged (‘shrink wrapped’)

Single system vs. product family (‘product line’)

New system vs. upgrade from existing system

Role of Requirements

Role of requirements may vary accordingly:

As a statement of the problem to be solved

As a contract between customer and developers

As a means of communication between designer, customer, end-users

As supporting artefacts for system validation and system evolution

Practical IV

Groups of two

One of you is a stakeholder

One is a developer / software engineer

Stakeholder wants app X developed

Title, summary

List requirements:
functional / non-functional

Summary

Requirements are the basis for ensuring that we are constructing the "right" (desired) system

Requirements documents/specifications are needed to obtain agreement about what a system should do, and to which quality

Requirements engineering is the task of creating agreed requirements documents/specifications

Requirements engineering has many aspects: to elicit, represent, record, analyse, manage, trace, re-use...

Some Recommended Books

•Sommerville, I. Software Engineering, 8th Edition, Pearson Education Limited, 2008 (Academic Textbook)
•Sommerville, I. and Sawyer, P. Requirements Engineering: A Good Practice Guide, 2004. (Practitioner's companion)
•Robertson, S. and Robertson, J. Mastering the Requirements Process. Addison Wesley, 1999. (Practitioner's companion)
•Kotonya, G. and Sommerville, I. Requirements Engineering - Processes and Techniques. John Wiley, 1998. (Academic textbook)
•Cockburn, A. Writing Effective Use Cases. Addison Wesley, 2000.
•Fowler, M. and Scott, K. UML Distilled (2nd ed.). Addison Wesley, 1999.
•Checkland, P. and Scholes, J. Soft Systems Methodology in Action. John Wiley, 1999.
•Jackson, M. Problem Frames and Methods, Addison Wesley, 2000. Kruchten, P. The Rational Unified Process (2nd ed.). Addison Wesley, 2000.
•Van Lamsweerde, A. Requirements Engineering: From System Goals to UML Models to Software Specifications. John Wiley, 2010