Fundamentals of RE

Chapter 2

Domain Understanding & Requirements Elicitation
Chap. 1: RE products and processes

Chap. 2: Elicitation techniques

- alternative options
- consolidated requirements
- agreed requirements
- documented requirements
A great deal of knowledge acquisition is involved: as introduced in Chapter 1 ... 

- Studying the system-as-is
  - Business organization (knowledge about organization): structure, dependencies, strategic objectives, policies, workflows, operational procedures, ...
  - Application domain (knowledge about the domain in which the problem world is rooted): concepts, objectives, tasks, constraints, regulations, ...
  - Analysis of problems with system-as-is: the actors and resources involved, task and workflows, problem raised in this context, symptoms, causes, consequences
A great deal of knowledge acquisition is involved: as introduced in Chapter 1 ...

- Analyzing **technology opportunities, new market conditions**
- Identifying the **system stakeholders**
- Identifying improvement objectives; organizational & technical constraints on system-to-be; alternative options for satisfying objectives, for assigning responsibilities; scenarios of hypothetical software-environment interaction; requirements on software, assumptions on environment
Domain analysis & requirements elicitation: outline

- Identifying stakeholders & interacting with them
- Domain understanding and requirements elicitation combines different techniques:
  - Artifact-driven elicitation techniques: rely on specific types of artifact to support the elicitation process
    - Background study
    - Data collection, questionnaires
    - Repertory grids, card sorts for concept acquisition
    - Scenarios, storyboards for problem world exploration
    - Prototypes, mock-ups for early feedback
    - Knowledge reuse: domain-independent, domain-specific
  - Stakeholder-driven elicitation techniques: rely on specific types of interaction with stakeholders
    - Interviews
    - Observation and ethnographic studies
    - Group sessions
Domain analysis & requirements elicitation: outline

- Identifying stakeholders & interacting with them
- Artifact-driven elicitation techniques
  - Background study
  - Data collection, questionnaires
  - Repertory grids, card sorts for concept acquisition
  - Scenarios, storyboards for problem world exploration
  - Prototypes, mock-ups for early feedback
  - Knowledge reuse: domain-independent, domain-specific
- Stakeholder-driven elicitation techniques
  - Interviews
  - Observation and ethnographic studies
  - Group sessions
Stakeholder analysis: **prerequisite step**

- Stakeholder cooperation is essential for successful RE
  - Elicitation = cooperative learning

- **Representative sample of stakeholder** must be selected to ensure adequate, comprehensive coverage of the problem world
  - dynamic selection as new knowledge is acquired

- **Selection based on** following criteria...
  - relevant position in the organization
  - role in making decisions, reaching agreement
  - type of contributed knowledge, level of domain expertise
  - exposure to perceived problems
  - personal interests, potential conflicts
  - influence in system acceptance
Knowledge acquisition from stakeholders is difficult (1)

- **Distributed sources, conflicting viewpoints**
  - Many sources from multiple stakeholders should be considered
  - Such sources are often spread out
  - Conflicts may arise based on various reasons: competition among departments, outdated documents, different priorities,…

- **Difficult access to key people & data**
  - Key people are generally very busy
  - They may not be convinced that it is worth spending time on the elicitation process

- **Different background, terminology, culture**
Knowledge acquisition from stakeholders is difficult (2)

- **Tacit** knowledge, hidden needs
  - Stakeholders assume that we know the details or connections among particular elements
  - People involved in routine tasks may have a hard time explaining things from a distance
  - Often do not know what they want and hard to explain
  - ...

- Irrelevant details

- Internal politics, competition, resistance to change, ...

- Unstable conditions: Personnel turnover, changes in organization, in priorities, ...
Knowledge acquisition from stakeholders is difficult (3)

⇒ Needed:
- Communication skills: for talking to, listening from diverse people
- Trust relationship
- Knowledge reformulation & restructuring (review meetings at appropriate milestones during the elicitation process in order to redirect it if necessary)
Domain analysis & requirements elicitation: outline

- Identifying stakeholders & interacting with them
- Artifact-driven elicitation techniques
  - Background study
  - Data collection, questionnaires
  - Repertory grids, card sorts for concept acquisition
  - Scenarios, storyboards for problem world exploration
  - Prototypes, mock-ups for early feedback
  - Knowledge reuse: domain-independent, domain-specific
- Stakeholder-driven elicitation techniques
  - Interviews
  - Observation and ethnographic studies
  - Group sessions
Artifact-driven: Background study

- Collect, read, synthesize documents about...
  - the organization: organizational charts, business plans, financial reports, meeting minutes, etc
  - the domain: books, surveys, articles, regulations, reports on similar systems in the same domain
  - the system-as-is: documented workflows, procedures, business rules; exchanged documents; defect/complaint reports, change requests, etc.

- Provides basics for getting prepared before meeting stakeholders => prerequisite to other techniques

- Data mining problem in this technique: huge documentation, irrelevant details, outdated info

- Solution: use meta-knowledge to prune the doc space
  - know what you need to know & what you don’t need to know
Artifact-driven: Data collection

- Gather undocumented facts & figures (e.g., relevant facts and figures that are not explicitly available in documents)
  - marketing data, usage statistics, performance figures, costs, ...
  - by designed experiments or selection of representative data sets from available sources (use of statistical sampling techniques)

- May complement background study

- Helpful for eliciting non-functional reqs on performance, usability, cost etc.

- Difficulties:
  - Getting reliable data may take time
  - Data must be correctly interpreted
Artifact-driven: Questionnaires

- Submit a list of questions to selected stakeholders, each with a list of possible answers (+ brief context if needed)
  - Multiple choice question: one answer to be selected from answer list
  - Weighting question: list of statements to be weighted...
    • qualitatively ('high', 'low', ...), or
    • quantitatively (percentages)
    to express perceived importance, preference, risk etc.

- Effective for acquiring subjective info quickly, cheaply, remotely from many people

- Helpful for preparing better focussed interviews (see next)
Questionnaires should be carefully prepared

- Subject to ...
  - multiple biases: recipients, respondents, questions, answers
  - unreliable info: misinterpretation of questions, of answers, inconsistent answers, ....
Artifact-driven: **Card sorts** & repertory grids(1)

- **Goal:** acquire **further info about concepts already elicited** to characterize these concepts for categorize them

- **Card sort:** ask stakeholders to partition a set of cards ...
  - Each card captures a concept textually or graphically
  - Cards grouped into subsets based on stakeholder’s criteria
  - For each subset, ask...
    - ? implicit shared property used for grouping (reason of grouping the cards together ?)
    - ? descriptive, prescriptive ? (to be considered it as a candidate domain property or requirement)
  - Iterate with same cards for new groupings/properties (to have more reasons for grouping the same cards)
Example: meeting scheduling system

- Iteration 1 (first reason): “Meeting”, “Participant” grouped together
  
  reason => “participants shall be invited to the meeting” - prescriptive

- Iteration 2 (another reason): “Meeting”, “Participant” grouped together
  
  reason => “participant constraints for the meeting must be known” - prescriptive
Artifact-driven: Card sorts & repertory grids

- Repertory grid: ask stakeholders to characterize target concept through attributes and value ranges
  => concept-attribute grid - have both (attribute, value range)
  e.g. (Date, Mon-Fri), (Location, Europe) for grid characterizing Meeting concept

- Conceptual laddering: ask stakeholders to classify target concepts along class-subclass links
  e.g. subclasses RegularMeeting, OccasionalMeeting of Meeting

😊 Simple, cheap, easy-to-use techniques for prompt elicitation of missing info

😊 Results may be subjective, irrelevant, inaccurate
Artifact-driven: Scenarios & storyboards

- Goal: acquire or validate info from concrete examples through narratives ...
  - how things are running in the system-as-is
  - how things should be running in the system-to-be

- Storyboard: tells a story by a sequence of snapshots
  - Snapshot = sentence, sketch, slide, picture, etc.
  - Possibly more structured with annotations:
    - WHO are the players, WHAT happens to them, WHY this happens, WHAT IF this does / does not happen, etc
  - Passive mode (for validation): stakeholders are told the story
  - Active mode (for joint exploration): stakeholders contribute to the story
Scenarios

- Illustrate **typical sequences of interaction among system components** to meet an implicit objective

- A **structured form of storyboard covering who, what and how dimensions**

- Widely used for...
  - explanation of system-**as-is** through concrete example of real-life interaction sequence
  - exploration of system-**to-be** + elicitation of further info ... e.g. WHY this interaction sequence? WHY among these components?
  - specification of acceptance test cases

- Represented by text or diagram
Scenario example: meeting scheduling in the system-to-be involve following interactions between a meeting initiator, the software scheduler and meeting participants

1. The initiator asks the scheduler for planning a meeting within some date range. The request includes a list of desired participants.

2. The scheduler checks that the initiator is entitled to do so and that the request is valid. It confirms to the initiator that the requested meeting is initiated.

3. The scheduler asks all participants in the submitted list to send their date and location constraints back within the prescribed date range.

4. When a participant returns her constraints, the scheduler validates them (e.g., with respect to the prescribed date range). It confirms to the participant that the constraints have been safely received.

5. Once all valid constraints are received, the scheduler determines a meeting date and location that fit them.

6. The scheduler notifies the scheduled meeting date and location to the initiator and to all invited participants.
Types of scenario

- **Positive scenario** = one behavior the system should cover (example)

- **Negative scenario** = one behavior the system should exclude (counter-example), e.g.
  1. A participant returns a list of constraints covering all dates within the given date range
  2. The scheduler forwards this message to all participants asking them for alternative constraints within extended date range

  Not to be disclosed to others

- **Normal scenario**: everything proceeds as expected

- **Abnormal scenario** = a desired interaction sequence in exception situation (still positive)

  e.g. meeting initiator not authorized participant constraints not valid
Scenarios: pros & cons (1)

😊 Concrete examples/counter-examples

😊 Narrative style (appealing to stakeholders)

😊 Usage extends beyond the elicitation phase: As animation sequences when we validate requirements, As counter examples when verify behavioral requirements, As test cases when we define acceptance test from the requirements

😊 Inherently partial (cf. test coverage problem)- they do not cover all possible system behaviors under all possible circumstances
Scenarios: pros & cons (2)

⚠️ **Combinatorial explosion** (cf. program traces)- because comprehensive set of scenarios requires us to enumerate multiple combinations of individual component behaviors.

⚠️ **Potential overspecification:** unnecessary sequencing of some interactions. Premature software-environment boundary because of the allocation of responsibilities among interacting components might be premature.

⚠️ **May contain irrelevant details,** incompatible granularities from different stakeholders.
Scenarios: pros & cons (3)

Keep requirements implicit: means capture interaction sequence, but not the reasons why such sequences should or should not take place [that is; the requirement underlying them].

We need explicit requirements to support negotiation, analysis, implementation and evolution.

cf. confidentiality req in negative scenario example

Concrete scenarios naturally jump in anyway...

Invaluable as initial elicitation vehicles
Artifact-driven: Prototypes & mock-ups

- **Goal**: check req adequacy from direct user feedback, by showing reduced sketch of software-to-be in action

- **Prototype** = quick implementation of some aspects of the system-to-be.

  - **Functional prototype**: focus on specific functional reqs e.g. initiating meeting, gathering participant constraints
  
  - **User interface prototype**: focus on usability by showing input-output forms, dialog patterns between SW and users e.g. static/dynamic interaction to get participant constraints

- The focus in general is on requirements that are unclear, hard to formulate or hard to understand

- **Quick implementation**: by use of very high-level programming language, executable spec language, generic services, ...
Requirements prototyping

- Mock-up: proto is thrown away (product = adequate reqs)
- Evolutionary proto: transformed towards efficient code
Prototypes & mock-ups: pros & cons

😊 Concrete flavor of what the software will look like
   => clarify reqs, elicit hidden ones, improve adequacy, understand implications, ...

😊 Other uses: user training, ...

👎 Does not cover all aspects
   - missing functionalities
   - ignores important non-functional reqs (performance, cost, ...)

👎 Can be misleading, set expectations too high

👎 ‘Quick-and-dirty’ code, hard to reuse for sw development

👎 Potential inconsistencies between modified code and documented reqs which results to decrease the confidence in the results
Artifact-driven: Knowledge reuse

- **Goal:** speed up elicitation by reuse of knowledge from experience with related systems
  - knowledge about similar organization, domain, problem world: requirements, assumptions, dom props, ...

- Reuse-based parts of the process combine the following steps:
  1. RETRIEVE relevant knowledge from other systems
  2. TRANSPOSE it to the target system
  3. VALIDATE the result, ADAPT it if necessary & INTEGRATE it with the system knowledge already acquired

- Transposition mechanisms:
  - instantiation (memberOf)
  - specialization (subClassOf) + feature inheritance
  - reformulation in vocabulary of target system
Two different types of reuse can be introduced based on whether

- the reused knowledge is domain independent or
- the reused knowledge is domain specific
Reuse of domain-independent knowledge

- Consider the requirement taxonomy
- Reuse of a meta-model
Reuse of **domain-independent** knowledge: requirements taxonomies

- For each leaf node in available req taxonomies:
  
  "Is there any system-specific req instance from this class?"

- More specific taxonomy => more focussed search

![Diagram showing Performance Requirement, Space, Time, Main Storage, Secondary Storage, ResponseTime, Throughput, OffPeakThroughput, PeakThroughput, PeakMeanThroughput, PeakUniformThroughput]

- Reusable catalogue in (Chung et al 2000)

- Response time for ...
  - participant constraints?
  - meeting scheduling?
  - meeting notification?

- Mean number of meetings to be scheduled at peak times?
Reuse of domain-independent knowledge: RD meta-model

- RD meta-model = concepts & relationships in terms of which RD items are captured
- We acquiring knowledge about the organization or about the target system (e.g., RD items) as instantiation of elements of the meta-model according to which the organization or system is modeled
- It help to figure out what questions to ask and when
- Elicitation by meta-model traversal
Reuse of domain-independent knowledge: 
RD meta-model

- If organization is modeled in terms of actors, resources and dependencies, the meta-model contain meta-classes: Actor, Task, Resource and Dependency

- If organization is modeled in terms of goal, objects, agents and operations, the meta-model contain meta-classes: Goal, Object, Agent and Operation
Reuse of domain-specific knowledge(1)

- Abstract domain = concepts, tasks, actors, objectives, reqs, dom props abstracting from a class of domains
- RD items acquired as specializations of abstract items to target system (feature inheritance + system-specific renaming)
Reuse of **domain-specific knowledge(2)**

- **E.x.** **Target system:** *library management*
  **Domain:** *resource management*
  
  **Abstract elements:**

  
  - **Tasks** such as ‘managing the acquisition of resource units’, ‘handling resource requests’, ‘tracking the history of resource usage’ etc.
  
  - **Actors** such as ‘resource user’, ‘resource manager’ etc.
  
  - **Objectives** such as ‘ensuring wide accessibility of resource units to users’, ‘ensuring appropriate resource localization in the repository for easy retrieval’ etc.
  
  - **Requirements** on the usage of resource units such as imposing some precondition on its usage (e.g. payment or reservation) and limiting usage time or simultaneous use of multiple units (e.g. ‘a user may not use more than x resource units at a time’).
  
  - **Domain properties** such as ‘a resource may have multiple units’, ‘a resource unit is a unit of one single resource only’, ‘a resource unit is no longer in the repository when it is used’ etc.
Reuse of **domain-specific knowledge**(3)

- **E.x** Specialize abstract elements such as:

  - resource, resource unit, repository, user, manager

  into the notation of:

  - book, book copy, library shelves, patron, library staff
Reuse of **domain-specific knowledge**(4)

**Abstract domain**

**Concrete domain**

- **Resource**
- **Limited Use**
- **User**
- **GetUnit**
- **Book**
- **Limited Loans**
- **Patron**
- **BorrowCopy**

**Specialization**

**Spec inheritance**

- "A user may not use more than X resource units at a time"
- "A patron may not loan more than X book copies at a time"
Reuse of domain-specific knowledge

- **Same abstract domain** may have multiple specializations
  e.g. resource management ← library loan management, 
  videotape management, flight or concert seat allocation, ...

- **Same concrete domain** may specialize multiple abstract domains
  e.g. library management (has three parts):
  loan management → resource management
  book acquisition → e-shopping
  patron registration → group membership management
Reuse of domain-specific knowledge (6)

- To increase the adequacy of reused knowledge, the abstract domain should be made more structured and more accurate;

  e.g. resource management is structured in terms of multiple specializations of the resource concept:
  
  returnable vs. consumable resource
  (distinguish between these two concepts)

  sharable vs. non-sharable resource
  (distinguish between these two concepts)

  => “A book copy can be borrowed by one patron at a time”
  (domain prop for non-sharable, returnable resource)
Knowledge reuse: pros & cons

😊 Expert analysts naturally reuse from past experience

😊 Significant guidance and reduction of elicitation efforts

😊 Inheritance of structure & quality of abstract domain spec

😊 Effective for completing RD with overlooked aspects

😊 Effective only if abstract domain sufficiently “close”, accurate

😊 Defining abstract domains for significant reusability is hard—specially when domain is complex

😊 Validation & integration efforts

😊 Near-matches may require tricky adaptations
Domain analysis & requirements elicitation: outline

- Identifying stakeholders & interacting with them
- Artifact-driven elicitation techniques
  - Background study
  - Data collection, questionnaires
  - Repertory grids, card sorts for concept acquisition
  - Scenarios, storyboards for problem world exploration
  - Prototypes, mock-ups for early feedback
  - Knowledge reuse: domain-independent, domain-specific
- Stakeholder-driven elicitation techniques
  - Interviews
  - Observation and ethnographic studies
  - Group sessions
stakeholders-driven: Interviews

- Primary technique for knowledge elicitation
  1. Select stakeholder specifically for info to be acquired (domain expert, manager, salesperson, end-user, consultant, ...)
  2. Organize meeting with interviewee, ask questions, record answers
  3. Write report from interview transcripts
  4. Submit report to interviewee for validation & refinement

- Single interview may involve multiple stakeholders
  - saves times
  - weaker contact; individuals less involved, speak less freely

- Interview effectiveness:
  \( \frac{\text{utility} \times \text{coverage of acquired info}}{\text{acquisition time}} \)
Types of interview

- **Structured interview:** *predetermined set of questions*
  - specific to purpose of interview
  - some open-ended, others with pre-determined answer set
  => more focused discussion, no rambling among topics

- **Unstructured interview:** *no predetermined set of questions*
  - free discussion about system-as-is, perceived problems, proposed solutions
  => exploration of possibly overlooked issues

=> **Effective interviews should mix both modes ...**
  - start with structured parts
  - shift to unstructured parts as felt necessary
Interviews: strengths & difficulties

😊 May reveal info not acquired through other techniques
- how things are running really, personal complaints, suggestions for improvement, ...

😊 On-the-fly acquisition of information appearing relevant
- new questions triggered from previous answers

😊 Acquired info might be subjective (hard to assess)

😊 Potential inconsistencies between different interviewees

😊 Effectiveness critically relies on interviewer's attitude, appropriateness of questions
stakeholders-driven: Observation & ethnographic studies

- Focus on task elicitation in the system-as-is
- Understanding a task is often easier by observing people performing it (rather than verbal or textual explanation)
  - cf. tying shoelaces
- Passive observation: no interference with task performers
  - Watch from outside, record (notes, video), edit transcripts, interpret
  - Protocol analysis: task performers concurrently explain it
  - Ethnographic studies: over long periods of time, try to discover emergent properties of social group involved about task performance + attitudes, reactions in specific situation, gestures, ...
- Active observation: you get involved in the task, even become a team member
Observation & ethnographic studies: pros & cons (1)

😊 May reveal ...

- tacit knowledge that would not emerge otherwise
  e.g. ethnographic study of air traffic control => to reveal an implicit model of air traffic that an automated version of the system needed to preserve. On the other word to analyze how controllers handle paper strips representing flight plans
- hidden problems through tricky ways of doing things
- culture-specific aspects to be taken into account

😊 Contextualization of acquired info

😊 Slow & expensive: to be done over long periods of time, at different times, under different workload conditions
Observation & ethnographic studies: pros & cons (2)

Data mining problem, interpretation problem

Potentially inaccurate (people behave differently when observed)

Focus on system-as-is

Some of the interviewing guidelines are relevant
Group sessions

- More perception, judgment, invention from interactions within group of diverse people

- Elicitation takes place in series of group workshops (a few days each) + follow-up actions
  audiovisuals, wall charts to foster discussion, record outcome

- **Structured** group sessions:
  - Each participant has a **clearly defined role**
    (leader, moderator, manager, user, developer, ...)
  - **Contributes to req elaboration** according to his/her role, towards reaching synergies
  - Generally **focused on high-level reqs**
  - **Variants:** focus groups, JAD,(Joint Application Development), QFD (Quality Function Development), ...
Unstructured group sessions (brainstorming):

- Participants have a less clearly defined role
- Two separate stages ...

1. Idea generation to address a problem:
   as many ideas as possible
   from each participant
   without censorship/criticism

2. Idea evaluation:
   by all participants together
   according to agreed criteria (e.g. value, cost, feasibility)
   to prioritize ideas
Group sessions: pros & cons

😊 Less formal interactions than interviews
   => may reveal hidden aspects of the system (as-is or to-be)

😊 Potentially ...
   - wider exploration of issues & ideas
   - more inventive ways of addressing problems

😊 Synergies => agreed conflict resolutions

😊 Group composition is critical ...
   - time consuming for key, busy people
   - heavily relying on leader expertise & skills
   - group dynamics, dominant persons => biases, inadequacies

😊 Risk of ...
   - missing focus & structure => rambling discussions, little concrete outcome, waste of time
   - superficial coverage of more technical issues
Combining techniques

- Elicitation techniques have complementary strengths & limitations
- **Strength-based combinations** are more effective for full, adequate coverage
  - artifact-driven + stakeholder-driven
- **Examples**
  - **Contextual Inquiry**: workplace observation + open-ended interviews + prototyping
  - **RAD**: JAD group sessions + evolutionary prototyping (with code generation tools)
- Techniques from other RE phases support elicitation too
  - Resolution of conflicts, risks, omissions, etc.
Domain analysis & requirements elicitation: summary

- Identifying the right stakeholders, interacting the right way
- Artifact-driven elicitation techniques
  - Background study as a prerequisite
  - Data collection, questionnaires for preparing interviews
  - Repertory grids, card sorts for concept characterization
  - Scenarios, storyboards for concrete exploration
  - Prototypes, mock-ups for early feedback & adequacy check
  - Knowledge reuse brings a lot: domain-independent, domain-specific
- Stakeholder-driven elicitation techniques
  - Interviews are essential - structured, unstructured, cf. guidelines
  - Observation, ethnographic studies for hidden knowledge
  - Group sessions for broader, more inventive acquisition & agreement

Model-driven elicitation provides focus & structure for what needs to be elicited