Abstraction-Based Requirements Management

Leah Goldin*
Afeka – Tel Aviv College of Engineering
Department of Software Engineering
218 Bney Efraim Rd., Tel Aviv 69107, Israel
leah@afeka.ac.il

Anthony Finkelstein
University College London
Department of Computer Science
Gower St., London WC1E 6BT, UK
a.finkelstein@cs.ucl.ac.uk

ABSTRACT
One of the most difficult challenges in requirements engineering is understanding the information provided by the stakeholders so as to establish the requirements. Moreover, there is considerable frustration whenever a requirement change is initiated and we are unable to analyse the impact of this change despite (or perhaps because of) having piles of requirements documents. This paper outlines a method for *Abstraction-based Requirements Management* (AbstRM) that is useful for requirements impact analysis and other requirements management activities.

The AbstRM method is exemplified in a worked case study and supported by an integration of AbstFinder and DOORS tools.

Categories and Subject Descriptors
D.2.1 **Software Engineering**: Requirements / Specifications

General Terms
Management, Documentation

Keywords
Requirements Management, Abstractions, Requirements Engineering

1. SCOPE
Systems engineering practice distinguishes between three principal components of requirements engineering (RE) [1] [2], which are active through the whole software life-cycle: requirements elicitation for acquiring the requirements data; requirements analysis to render the requirements data useable through modeling; requirements management [3] to access the requirements data in a controlled manner. Thus, the resulting system quality is heavily depended on good coordination and well integration of those three RE components.

The objective of our work is to use abstractions as a major motive in the RE process, both in requirements knowledge and decision making, while improving the integration of requirements elicitation and requirements management. To achieve this we have developed a method for abstraction-based requirements management, AbstRM, that can be used as a basis for impact analysis and other requirements management activities. This method is exemplified in a worked case study - the Meeting Scheduler System (MSS) [4]. The method is supported by a prototype integration of AbstFinder, an experimental requirements elicitation tool [5] and DOORS, a commercial requirements management tool [6].

In Section 2 we outline the AbstRM method, in Section 3 we show how the products of AbstRM are used to improve the RE process, in Section 4 we give an account of our worked case study and the prototype implementation, and in Section 5 we briefly conclude and summarize our results.

ROA'06, May 21, 2006, Shanghai, China.

2. THE AbstRM METOD
AbstRM method is composed of four key steps including abstractions identification, classification, retrieval, and build of the abstractions network. A summary diagram illustrating the application of the method is shown in Figure 1 along with explanation of each step in the following subsections.

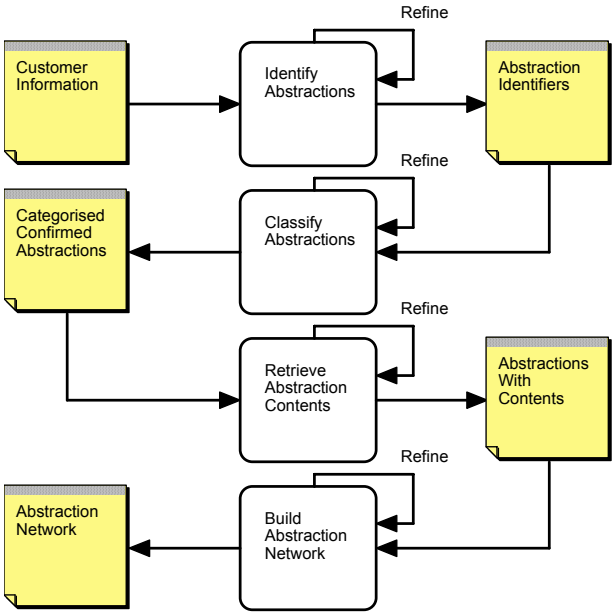

Figure 1: AbstRM method.

In Section 3 we will show how the products of AbstRM are used to improve the RE process, specifically as a mediating representation between the project stakeholders, as a support for impact analysis, and as a basis for requirements documentation.

2.1 Step 1: Identifying Abstractions

The first step in our method is *abstraction identification.* When people try to understand a requirements document, they usually abstract the contents. In this case, abstracting means ignoring enough details to capture the main ideas or concepts in the requirements. Abstraction identification is the cognitive process [7], [13] of identifying the primary conceptual elements - *abstractions* - in a requirements document, such that it helps humans to understand that document. An abstraction is not the same as a requirement, the relationship between requirements and abstractions is generally many to many. Abstractions can however serve as a prompt for requirements, and can be used to negotiate a shared understanding of the domain with the customer.

Abstraction identification is performed on any and all customer information, such as requests for proposal, interviews with customers and users, informal technical notes, etc. Information related to particular abstractions is commonly spread among several documents and may be inconsistent because of different sources or originating viewpoints. This information is usually difficult to find and is not available as a whole for decision making or analysis. Heretofore, REngineers have identified abstractions manually; they scan the requirements information, trying to note important subjects and objects of sentences, and determine the abstractions from them. However, this process is labour intensive and error prone.

AbstFinder [5] is a tool for finding abstractions in natural language text, using signal processing algorithms, thus it is language independent. AbstFinder treats a sentence as a stream of arbitrary characters with the strings of words appearing anywhere; this enables finding commonalities at any place in any order.

The input to AbstFinder is the customer information (CI). Ideally, it should contain what the customer believes is a complete description of the system to be built. This description should consist primarily of natural language text, but it can contain pictures if the text of the picture can be readily retrieved.

The output of AbstFinder is a list of *abstraction identifiers*. This takes the form of a list of significant words or short phrases relating to candidate abstractions. This list can then be refined by the REngineer in conjunction with the stakeholders to produce a list of confirmed abstractions. Though much of the intellectual effort entailed by abstraction identification remains, the REngineer can have reasonable confidence that no relevant information has been overlooked in the suggested list of abstractions. The list of abstraction identifiers as generated by AbstFinder for the MSS requirements is shown in Appendix A.

2.2 Step 2: Classifying Abstractions

The output of abstraction identification is an unstructured list of abstractions. In order to make these abstractions useable the next step in our method is the classification of the abstractions with respect to a framework of meta-concepts. We have selected the meta-concepts built into KAOS a goal-oriented requirements specification scheme [8]. Our selection reflects our views about the appropriate approach to requirements analysis however, it is not necessary to use KAOS and other sets of concepts may be used without significantly altering our method. The principal meta-concepts of KAOS are: actions; entities; agents; goals and constraints. Table 1 below shows the confirmed abstractions from the MSS requirements categorised according to this scheme.

In fact, classifying the abstractions according to meta-concepts creates a generic view of the domain subjects and concepts, e.g., actions, entities, goals, constrains, etc. This in addition enables the use of the abstractions as input to a variety of requirements analysis formal methods, e.g., the abstractions can be used as an initial list of objects/classes in OOA [9].

Table 1: MSS classified abstractions

Actions	Entities	Agents	Goals	Constraints
(re)plan	request	participant	extensible	concurrent
organise	date	initiator	flexible	priority
conflict	exclusion	representative	useable	constrain
resolution	(date)			
communicate	preference	attendee	privacy	
	(date)			
interact	(date) range	person		
negotiate	location			
determine	room			
handle	time			
explicit				
(make)	information			
aware				
	equipment			

2.3 Step 3: Retrieving Abstraction Contents

The next step for the REngineer is to marry each categorised confirmed abstraction with its contents. The abstraction contents is all the sentences, collected from different places in the requirements information, that deal with the subject of the abstraction. Generally this is taken to be all the sentences in which the abstraction identifier appears. In our prototype we are able to retrieve these sentences and to gather them into a document for each abstraction. Some manual inspection and editing work is required in order to ensure that problems with ellipsis and other forms of reference are properly handled. Additional work will be required to identify and retrieve synonyms. Further refinement of the abstraction contents may be helpful at this stage though in most cases the content will remain ambiguous, incomplete, and inconsistent. An example of *equipment* abstraction with associated content taken from the MSS is shown below.

Abst-Id: EQUIPment

(1) The **initiator** also asks active **participants** to provide any special equipment requirements on the meeting **location** (e.g., overhead-projector, workstation, network connection, telephones) [from s14]

(2) It [meeting **room**] should meet the equipment requirements [from s27]

The "Allocated Requirements" are the actual requirements of the *equipment* abstraction from the MSS requirements document. The requirements are sometimes added with context information for clarification reason, e.g., in requirements 2) "It" refers to the [*meeting room]*. Within each requirement the words in bold reflect the cross reference links to other abstractions which will be explained in step 4 in section 2.4..

2.4 Step 4: Building an Abstraction Network

Once the abstractions and their contents have been gathered, the next step is to organise this information as a network in which the nodes are the abstractions and their associated content, and the arcs represent key relations between these abstractions. We distinguish two basic types of links between abstractions: cross reference and sub-abstractions relationship.

A cross reference relationship occurs where one abstraction is mentioned in the content of another abstraction. A sub-abstraction occurs where an abstraction represents a high level concept that includes lower level concepts, and their requirements are actually a sub-set of the main abstraction requirements. For instance, *date* abstraction includes *preference dates, exclusion dates* and *date range* sub-abstractions. The cross-reference links can be identified relatively simply by text search though, as always, some editing may be required. The sub-abstraction links can be identified already at the classification step 2, where candidates for sub-abstractions are usually recognised under the same meta-concept. The cross-reference links identification can be done automatically via scripts, the sub-abstraction links identification requires the active involvement of the REngineer.

An example of an abstraction network showing that part of network relating to the entities and agents of the MSS is shown in Figure 2.

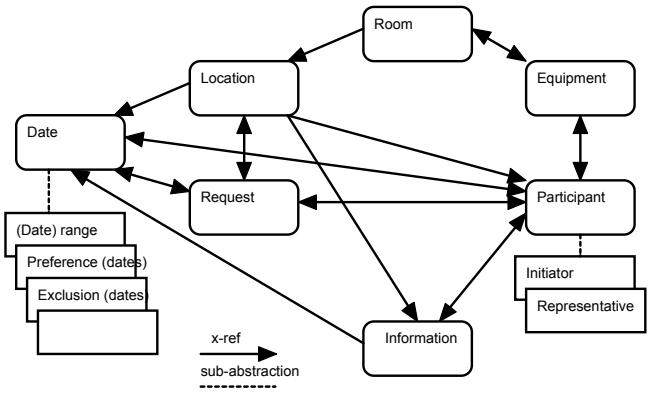

Figure 2: MSS abstraction network

3. AbstRM USAGE BENEFITS

The principal products of AbstRM method can be used to improve the RE process and achieve a tighter integration between requirements elicitation and requirements management. It involves the use of abstraction networks as a mediating representation between project stakeholders to improve elicitation, and as a support for impact analysis to improve requirements management as well. The use of abstractions as a basis for RE documentation including executive summary is also considered as a RM benefit.

3.1 Mediating Representation for Elicitation

The abstraction network can be used as a mediating representation between project stakeholders. The contents of the abstractions are in the customer's original language, with minimal interpretation. The links can be used for navigation and obtaining more information about a relevant subject. Because the contents of each abstraction may be derived from different sources it is easier to identify conflicts and missing information, and to discuss them with the customer. Negotiation with the customer, in his or her own language, is necessary in order to resolve inconsistencies and to add more information for the purpose of obtaining useful requirements. Furthermore, the abstraction network gives the REngineer the ability to identify mutual influences among abstractions and to identify requirements based on that influence.

In order to show the use of the abstraction network as a mediating representation for requirements elicitation we can look at the example in Figure 2 which shows the entities and agents of the MSS.

Initial inspection of the content of the entity and agent abstractions in Table 1 suggested the relationships between the entities. The *date* abstraction, which contains the larger number of requirements, has 3 sub-abstractions: *(date) range, exclusion (dates), preference (dates)*. The *location* abstraction has synonyms: *place, address, and space*, which entailed adding those sentences relating to those terms to the contents of the *location* abstraction. These synonyms were identified while refining related abstractions such as *request*. The *participant* abstraction has 2 sub-abstractions: *initiator* and *representative*. It also has 2 synonyms: *attendee* and *person*.

The *time* abstraction is a good example of how AbstRM method and the abstraction network revealed a very typical confusion of terms and helped to identify the way in which this could be corrected, with customer consent of course. The abstraction *time* can be used in 2 different senses: *time* as date, and *time* as performance, it is only when attempting to collect the abstraction contents that this is clarified. When using *time* in the sense of date-time, the relevant content can be added to the *date* abstraction. When using *time* in the sense of performance-time, the relevant content can be added to another *performance* abstraction (which is not at the MSS case study). Thus, *time* does not seem to be an issue by itself in the MSS and some parts of the validation process can be eliminated. This phenomenon of overloading a term is very frequently encountered in RE, particularly where there are many people involved in the process. Generally such overloading is only identified late in the project life-cycle with obvious consequences. The use of the abstraction network provides a simple and direct way to identify the problem at an early stage.

3.2 Requirements Impact Analysis

Inevitably requirements change, and these change impact all of the artefacts produced during the development process. Thus whenever a change is proposed, it is necessary to identify the potential effects of that change. This process is known as impact analysis. Good impact analysis assists the project stakeholders in making better decisions about changes and their consequences.

The core nature of the abstraction network is its conceptual organisation. Thus, whenever any requirement change is suggested related to any of the abstractions, tracing the attached links should give significant clues to the impact of that change.

Again let us consider an example drawn from the MSS. Suppose that one of the MSS clients, who has been using a "version 1.0" MSS for a while, wishes to add the requirement: "The MSS shall be able to accommodate teleconferencing". Since this requirement concerns a new (rather ambiguous) telecommunication technique, it seems that the *equipment* abstraction is the most appropriate place for it. Figure 3 shows the impact of this change by reference to the abstraction network, starting from the *equipment* abstraction and showing the flowdown by following the cross-reference links between the abstractions.

The numerals refer to Table 2 which shows the requirements that would have added to each abstraction as a result of the proposed change. The associative navigation between the concepts allows a more systematic impact analysis process. A good example is the knock-on effect of the change to the *equipment* abstraction on the *request* abstraction, i.e., teleconference can be acquired during the request, rather than a result of not being able to meet at some location. This effect on request was not intuitive at all, and couldn't be reached without the systematic and iterative conceptual approach of the abstraction network.

Table 2: New requirements due to "teleconferencing" change

No.	New requirements
I.	"No need for meeting room where meeting via teleconferencing" added to the *room* abstraction after following the *equipment-room* link.
II.	"Need special help for deaf participants, such as simultaneous caption" added to the *participant* abstraction after following the *equipment-participant* link. Also to add to the *equipment* abstraction.
III.	"Need teleconferencing equipment at each participant location" added to the *location* abstraction after following the *room-location* link. This new requirement sharpens the distinction already observed between the *participant location* and the *meeting location* which are both contained in the same *location* abstraction.
IV.	"Need teleconferencing equipment at each participant location" added to the *participant* abstraction after following the *location-participant* link.
V.	"Need to request a meeting via teleconferencing specifically" added to the *request* abstraction after following the *participant-request* link. This has to be validated by the customer or other clients, thus teleconferencing might be specifically requested, or could be a default strategy in the event of failure to organise a meeting.
VI.	"The information about the meeting will specify if it is by teleconferencing" added to the *information* abstraction after following the *participant-information* link.

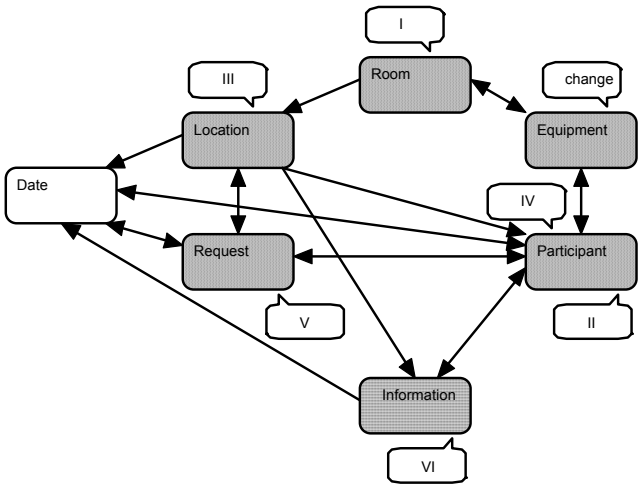

Figure 3: Change impact of the "teleconferencing" on MSS

3.3 Requirements Executive Summary (Index Page)

Most complex requirements documents include executive summaries. These summaries have to include all the key issues and should be complete while excluding unnecessary details. The confirmed abstraction identifiers can serve as a valuable checklist for producing such summaries for management. Obviously this cannot be done automatically however the list of abstractions considerably simplifies the task.

An executive summary for the MSS requirements is given in Figure 4. This summary can be also used as an index page into the underlying abstraction network.

The purpose of the Meeing scheduler System is to support the *organisation* of meetings. The *determination* of a meeting is a result of an *initiator request* and the ability of the *participants* or their *representatives,* to *attend* a meeting in an well *equipped room* at some *location* on a certain *date* for a period of *time*, according to their *constraints* and *preference* and *exclusion* dates.

The MSS shall support the *(re)planning* of a meeting while *handling priorities*, changes, *constraints*, and maintain confidentiality wrt participant's *privacy*.

The MSS shall support the organisation of a meeting including *communication* and *interaction* between participants, *negotiation* and *conflict resolution*, and keep participants *aware* about the *information* on and for the meeting.

The MSS shall be *flexible* and *usable* for different *concurrent users,* and should be *extendable.*

Figure 4: Executive summary for the MSS

3.4 Requirements Documentation and Traceability

Requirements documentation, i.e., Software Requirements Specification (SRS), is usually based on standards such as MIL-STD [10], IEEE [11], ESA [12], etc. Actually, all the standards require traceability of requirements in order to assure validation of requirements.

An important problem in writing the SRS is in deciding, i.e., deciding what subjects to write under what section of the document template. Having the abstractions categorised into the meta-concepts of objects, actions, agents, goals, constraints, etc. shown in Table 1, enables a very easy match between the SRS sections to the relevant subjects of the requirements. Thus, the abstractions can serve as sub-sections of the requirements document template. In Figure 5, a proposal for SRS section 3 according to MIL-STD-498 for the MSS is demonstrated.

srs3.1 functional requirements

srs3.1.1.plan/ replan srs3.1.2.communicate (interaction, negotiation, make aware) srs3.1.3.negotiation srs3.1.4.organise (determine, conflict resolution,...) srs3.1.5.handle explicit...

srs3.3 external interface requirements

srs3.3.1.request srs3.3.2.date srs3.3.3.location srs3.3.4.Information srs3.3.5.equipment srs3.3.6.room srs3.3.7.priorities....

srs3.12 personnel related requirements

srs3.12.1.participant srs3.12.2.initiator
srs3.12.3.representative

Figure 5: Executive summary for the MSS

The subjects taken from the abstractions list may be used in designing the outline of the formal spec documents, mainly to enable a shared terminology and avoid overloading of terms.

4. IMPLEMENTATION

A prototype implementation of the AbstRM method was achieved through the integration of AbstFinder with DOORS and complemented by a technical account of the application to the MSS case study.

Once AbstFinder was applied on the MSS requirements, each abstraction identifier was marked-up as a formal DOORS module, and were then populated with its content using the objects marked-up from the MSS document, i.e., all the sentences/requirements containing the abstraction identifier or its synonyms. All cross-reference links were generated and placed in a DOORS link module via DXL script language of DOORS. Since the sub-abstractions were contained in their parent abstraction module, the sub-abstraction links were generated automatically by DOORS.

Control of AbstFinder and the additional functionality required for the AbstRM method was integrated into the icon bar of DOORS. We also provided a related abstractions script which allows the user to navigate the abstraction network, for the sake of impact analysis.

To support the requirements management implications of AbstRM our prototype included an Update SRD script in which changes to the abstraction network were propagated to the relevant section of an ESA-PSS-05 SRD, the Software Requirements Document.

5. CONCLUSION

This paper has outlined a method for Abstraction-based Requirements Management. A proof of concept is presented in the form of a worked case study of the MSS and supported by an integration of AbstFinder and DOORS. Clearly our tool is only a rough prototype and further work will be required to make this industrially useable.

The AbstRM method is itself still at the proof of concept stage. We believe that it may yield benefits in providing support as a mediating representation between stakeholders, impact analysis and requirements documentation. AbstRM directs the REngineer as to how to structure the requirements knowledge into the standards requirements templates, and enhances traceability between early customer information and system specification. By these means AbstRM may contribute to the integration between requirements elicitation and requirements management .We envisage that our method will be of most use in the software intensive system development where document-intensive processes that characterise standard systems engineering practice are required. The restructuring of the data into abstraction network enables a conceptual partitioning of the requirements data that can be implemented on the web as abstractions sites. The abstractions list itself can be used as an index to create the home page of the specific project requirements intranet.

Our work is clearly at an early stage and further significant work to test the approach at scale is required.

6. REFERENCES

[1] Berry, M. D. and Lawrence, B. Requirements Engineering. *IEEE Software 15, 2* (March 1998), 26–29.

[2] Nuseibeh, B. and Easterbrook, S., Requirements Engineering: A Roadmap. In *The Future of Software Engineering 2000*, ed. A. Finkelstein, ACM, Limerick, Ireland, June 2000.

[3] Paulk, M.C., Curtis, B., Chrissis, M.B., and Weber, C.V. *Key Practices of the Capability Maturity Model,* Technical Report, CMU/SEI-93-TR-25, Software Engineering Institute, Pittsburg, US, 1993

[4] A. van Lamsweerde, R. Darimont, and Ph. Massonet. *The Meeting Scheduler System: Problem Statement*. //ftp.info.ucl.ac.be/pub/publi/92/MeetingScheduler.ps

[5] Goldin L. and Berry M.D. AbstFinder Prototype Abstraction Finder for Natural Language Text for Use in Requirements elicitation. *Automated Software Engineering*, Kluwer Publishers, Netherlands 1997.

[6] DOORS: Requirements Management Tool, Telelogic. http://www.telelogic.com/corp/products/doors/doors/

[7] Kramer, J. Abstraction – is it teachable? Or the devil is in the detail. In the *16th Conference of Software engineering Education and Training*, IEEE Computer Society, Madrid, Spain, 2003.

[8] P. Bertrand, R. Darimont, E. Delor, P. Massonet, A. van Lamsweerde. GRAIL/KAOS: an environment for goal driven requirements engineering. In the proceedings of *ICSE'98 - 20th International Conference on Software Engineering*, IEEE-ACM, Kyoto, April 98. //ftp.info.ucl.ac.be/pub/publi/98/icse984p.ps.gz

[9] Booch, G., *Object Oriented Analysis and Design*. Second Edition, Redwood City, CA, 1994

[10] *Software Development and Documentation.* MIL-STD-498, 1994.

[11] *IEEE Recommended Practice for Software Requirements Specification*, IEEE STD 830-1993, 1993.

[12] European Space Agency Development Standards. *ESA Software Engineering Standards*, ESA-PSS-05, Issue2, February 1991, http://www.estec.esa.nl/wmwww/WME/essde/swstd/swstd000.htm

[13] R. Lecoeuche, Finding Comparatively Important Concepts Between Texts, IEEE International Conference on Automated Software Engineering (ASE), Grenoble, France, 11-15 September, 2000

7. APPENDIX A: MSS list of abstraction identifiers as generated by AbstFinder

participants |
conflict|
room |
managed|
equipment |
processes |
private |
partial |
request|
participants |constraints |
location | date |date location |
request|dependen|
request||time s|
support |replan|flexib|
account|
replan|planning |bound |
period|
activities|
determine|
exclusion |dates |
preference|important |
initiator |represent|
initiator |extend|
information |potential |
accom|modate |
dependen| handling explicit |
address |variations |
project| initiator asks |overhead |
project|organizing |number |
stated |resolution | resolution |
required |organization |interaction|
negotiation required |
handling explicit |priorities |
small | elapsed |determination co|
attend| dates |exclusion |attend |
preference |preference sets |
dates preference |
organization |determine|participat|
range |
allocated|
attendees |person|
range outside |
amount | keep |
range | proposed| proposed |
proposed|preferred |
extend|accommodate | accom|
attendees |amount |places|
communicat| communication|concerned |
proposed |accommodate |
person|places|s allocated o|person |
participant|aware |
keep | communicat| communication|participant|aware |
person|participat|participant|person |

---

* Leah Goldin is currently a research fellow at University College London, UK