CREWS Report 99-15

Automatically Structuring Requirement Scenarios

Andreas Becks, Jörg Köller

Lehrstuhl für Informatik V, RWTH Aachen, Ahornstraße 55, 52056 Aachen, Germany
phone: +49 (0)241 21 516
{becks, koeller}@informatik.rwth-aachen.de

Automatically Structuring Textual Requirement Scenarios

Andreas Becks, Jörg Köller

Lehrstuhl für Informatik V, RWTH Aachen, Ahornstraße 55, 52056 Aachen, Germany
{becks, koeller}@informatik.rwth-aachen.de

Abstract. Scenarios are valuable for supporting communication among system developers in the initial phases of requirements engineering. But the problem of how to fruitfully deal with large informal or semi-formal scenario collections consisting of weakly structured texts is still a key research issue. In this paper we report on the application of a novel approach for automatically structuring textual document collections to scenario management. We discuss how structuring scenarios can help to support the analysis and maintenance of scenario collections. To evaluate this approach we present a case study within the CAPE-OPEN project which is concerned with a collaborative effort of standardizing simulator software for chemical engineering.

1 Introduction

Scenarios, in software and systems engineering, describe processes and interactions related to a system under examination. They contain situations which reasonably might occur. It is commonly accepted that developing scenarios is a valuable approach for supporting communication in the initial phases of system development and, furthermore, that scenarios are a key element of change management [20]. The Use Case driven scenario method by Ivar Jacobson [9] has increased the interest in scenario management especially in object-oriented software design. A *Use Case* is a semi-formal textual description of an action that is performed by an external actor on a specific part of a system. The actor can be a user or another module or object of the system. The interaction between system and actor and the subsequent actions are described. This description may include, for example, messages passed to the system or in- or output from or to the actor. Hence, the Use Case is attempting to capture the logical interactions rather than the physical appearance of the system. A *Use Case Model* finally consists of a collection of interrelated Use Cases. There is a UML standard notation for Use Cases [17].

There are still a lot of important research questions regarding the complex and interdisciplinary scenario approach [11]. Among them, the question of how to deal with large collections of weakly structured informal requirement scenarios – such as Use Cases – is crucial. Typically, scenarios evolve over a long period of a system's life cycle and, thus, the problem of maintaining them in a repository arises. Problems coming along in this context originate from the following facets of scenario management:

- *Collaborative aspect*: In many projects the production of scenarios is a cooperative and distributed venture. In particular in world-wide projects, such as standardization efforts, the cooperative aspect has a very special weight (for a detailed experience report regarding simulator standardization in the chemical industries cf. [10]). In addition, the growing importance of the World Wide Web as a global market place accelerates the world-wide distributed production of software. Teams of developers working in different places of the earth meet in virtual workrooms and share their ideas and knowledge. There are several problems coming along with this collaborative approach. Important in the Use Case context is that many of the usage situations described by different authors are more or less similar and can be found in several and rather

different contexts. Sometimes these scenarios contain contradictory statements. Thus, it is very import to synchronize the knowledge elicited by distributed groups and to detect those implicit and hidden relationships in order to recognize (partial) redundancy and avoid inconsistency.

- *Analysis aspect*: In practice the number of scenarios worked out often exceeds a manually manageable size. Thus, it often turns out that it is very difficult to elicit semantic relationships between single scenarios beyond an *a priori* defined structure. To better understand the interplay among system components and to effectively discover the user needs implied in the stories the Use Cases tell it is necessary to gain an intuitive overview about the inherent semantic structure of the Use Case collection. Such a structure would offer much benefit for the interrelation and condensing process of scenario management.
- *Maintenance aspect*: During a system's lifecycle the number of interesting scenarios typically grows over a long period of time. Complex situation descriptions, possibly considered from different viewpoints, are added to the initial collection of scenarios. To interrelate and integrate these cases with the existing ones an intelligent repository would be of high value which automatically correlates semantically similar scenarios.

In this work we face the problems sketched above. The goal is to provide a means for assisting the analysis, correlation and maintenance of complex, collaboratively produced informal scenarios. The idea is to produce a semantically structured overview about large scenario collections. More precisely, we propose to automatically compute a semantic map of textual and informal requirement documents which exposes the semantic structure of the collection and thus helps to understand relationships among single texts. As documents are added to the repository, the map arranges them close to semantically relating scenarios. We will apply this approach to Use Cases generated in the CAPE-OPEN project (cf. section 4).

The remainder of this paper is organized as follows: After discussing some related work the notion of semantic structuring and document maps will be introduced, followed by a brief description of techniques applied for generating document maps. Section 4 evaluates the application of document maps to software engineering in the context of a real-world usage experience.

2 Related Work

Due to the fact that Use Cases are textual documents it seems likely that the application of techniques from information retrieval can help to overcome the problems discussed in the last section. Clearly, query based retrieval techniques [13] are only of limited use for the complex analysis of scenario collections since key word searches only poorly support the process of understanding a collection's structure. A more promising approach is browsing through a collection of informal documents which requires some elaborated preprocessing. In [2] a method for automatically constructing hypertexts has been introduced. Its benefit for analyzing and maintaining scenario collections has to be doubted since the engineers can easily get lost in the complex linking structure of hypertext. An intuitive overview about Use Case correlation remains missing. Other approaches from information retrieval are concerned with visualizing the similarity of documents within a collection [1, 6]. Though interesting in this context, the expressiveness and adaptability of these approaches is too limited to offer a more intuitive insight in the collection's semantic structure.

Besides approaches from information retrieval some work concerning structuring and retrieving scenarios and software components, respectively, was conducted in the fields of requirements and software engineering. Regarding the search in repositories techniques for both specification-based and text description-based retrieval and browsing in software libraries have been developed [7, 8, 15]. Due to the highly specialized focus of these approaches they cannot be applied for searching in informal scenario

repositories. In [16] a hypertext model for structuring informal requirement representations (e.g. multimedia documents) is proposed. This model aims at providing a tool for keeping the whole requirements engineering process traceable during a long-termed process of decision and change management. However, quite clearly, the initial phase of projects where informal requirement documents are typically designed in a brainstorming fashion is not supported by a (manually) traceability structuring.

To conclude, the problem of automatically structuring large informal scenario collections to support their analysis and maintenance has not yet been addressed. In [3] we have proposed a conceptual schema which combines techniques from information retrieval with neural network learning and powerful visualization techniques adopted from data mining. Furthermore, this framework allows the integration of domain knowledge to improve the quality of the computed semantic structures. Originally developed for knowledge management in a technical environment, in this paper we report on the application of our new approach for intuitively visualizing the semantic structure of scenario collections.

3 Semantic Maps of Textual Requirement Scenarios

The Concept of Document Maps

In [4] we have proposed the concept of document maps as an interface for specialized branches of digital libraries. This concept will be adopted for software engineering purposes. The main idea of a document map of Use Cases is that it visualizes the semantic structure of the Use Case collection: During a process of automatic semantic structuring a grouping of similar documents is worked out which points out relationships of the collections' documents. The map, then, serves as a basis for an interactive and intuitive interface for exploring the Use Cases.

The Use Case map can be seen as a 'landscape of documents' which expressively presents the structure of the document collection (Figure 1). The single Use Cases are represented as points in the map and similar documents are grouped as neighbored points. Furthermore, the shading of the map implies distance information: The degree of brightness corresponds to the degree of distance where dark areas mean high distances between documents. To use an intuitive metaphor think of the map as a landscape of 'mountains' and 'valleys' where 'valleys of similar Use Cases' are separated by dark borders or 'mountains': the higher the mountain the darker the border and, thus, the greater the dissimilarity of documents or groups of documents, respectively.

For interacting with the map the requirement engineer can mark an area and zoom into the specified field. The documents within the fields are described by their titles. Clicking on a point in the map yields the corresponding document. This allows the user to interactively explore the scenarios and to learn about the inherent structure of the Use Case collection, i.e. to identify relationships between single Use Cases and groups of scenarios. In particular, the following facets of scenario management are supported:

- *Collaborative aspect*: The map helps to identify similarities among different scenarios. By inspecting close relationships which are unexpected by the engineers (partial) redundancy and inconsistency can be detected much easier than by a manual analysis of the Use Case collection. For example, a specific situation may be described by more than one author by mistake. The double description may contain contradictory statements. By relating the corresponding scenarios the map aids the synchronization of knowledge elicited by collaborative working groups.
- *Analysis aspect*: Similarly, the map supports the process of analysing the collection. If, for example, different scenarios contain similar or identical sub-cases, the respective documents will

tend to be located near each other. Thus, the sub-case can be sourced out. Vice versa, highly similar situations can be combined if desired.

- *Maintenance aspect*: The 'Use Case landscape' can serve as an intuitive retrieval interface for a repository of scenarios. Semantically similar cases can be found close together and, thus, the user can browse across groups of related scenarios, exploring the collection and searching for documents.

A more detailed report on the application of Use Case maps can be found in the section 4. Now, we briefly present how Use Case landscapes are generated.

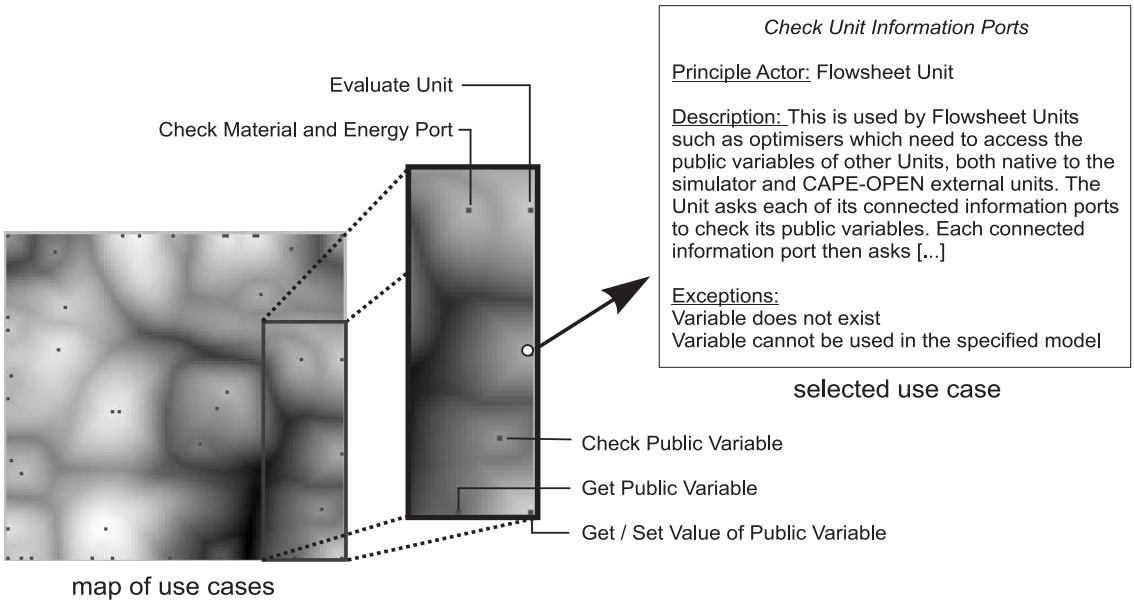

Figure 1: A map of Use Cases and possible interaction. The Use Cases are represented as points in the map, described by their titles. By clicking on a dot the user receives the corresponding Use Case.

The Modular Scheme of Semantic Structuring

The approach of semantic structuring and document maps has been originally developed for knowledge management and retrieval of technical and scientific documents [3, 4]. To reliably asses similarities across documents it is crucial to look at the different document types: Use Cases present a process by describing associated actions chronologically. Typically, actions and objects are clearly pointed out by using unequivocal terms and key words. These characteristics allow the application of statistical information retrieval models for assessing the similarity of the documents. In contrast, a comparison of management documents or medical document abstracts is more sophisticated. In the latter case there are fine granular differences in the meaning of key terms and deep relationships between concepts, so that a high quality similarity assessment should be performed using knowledge based techniques. Due to this varying 'intensity of knowledge' which is necessary to asses similarities across documents we have proposed a framework which enables the integration of both symbolic knowledge representation techniques and statistical document retrieval methods with powerful visualization approaches. In this work we apply this framework to automatically compute the inherent structure of Use Case collections. Due to the scope of this paper we restrict the technical consideration to a brief sketch. A detailed

description of the framework and the applied techniques can be found in [3]. Figure 2 depicts the process of generating a Use Case map which consists of two major steps: analyzing the documents' contents (step 1) and visualizing the collection's structure (step 2). These two steps have to be connected by an appropriate interface.

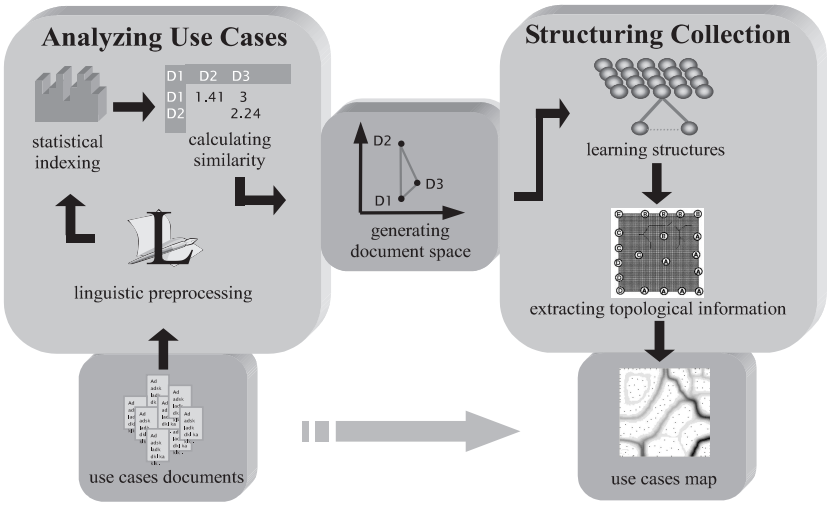

Figure 2: The process of calculating a Use Case map

Step 1: The input to the process pipeline is a collection of Use Cases. In the Use Case application the first component is realized using the vector space model of information retrieval [18]: First, the texts are preprocessed linguistically, i.e. so-called stop words (conjunctions, articles and fillers) are removed and the remaining terms are reduced to their stem. Each Use Case can then be indexed by counting the relative frequency (weight) of the resulting terms. Formally, the documents are described by (very high dimensional) term vectors consisting of real-valued components which describe the weight of an indexing term. Based on this vector representation a measure of (dis-) similarity can be applied, such as the well-known cosine measure or the Euclidian distance.

Interface: The next step includes the analysis and visualization of the document collection's structure. Unfortunately, most visualization techniques are not able to handle extremely high-dimensional input spaces as delivered by term vector indexing. Furthermore, to allow the incorporation of symbolic document indexing and comparison techniques a common interface to visualization methods was defined. The desired interface is a multi-dimensional space with a moderate number of dimensions which reflects the documents' (dis-) similarity in its topology. Such a space can be generated using the technique of Multi-Dimensional Scaling [14] where objects are mapped into *m*-space – where *m* is user-defined and of moderate size – minimizing the relative error of the distances in *m*-space regarding the 'true' (given) distances of the objects.

Step 2: The final step of the structuring process is performed by a self-organizing feature map [12] which "learns" the structure of the Use Case collection. This single-layered neural network maps the multi-dimensional document vectors, representing the Use Cases, to a 2-dimensional grid of units, preserving most of the topological information. The mapping is realized by an unsupervised learning algorithm which orders the network's weight vectors according to the distance of the corresponding document vectors. After the learning process the topological information encoded in the network has to be extracted. This is realized by applying a visualization technique proposed in [19] which assigns dark shades of grey to units of the grid representing large distances between documents and bright colors to units representing high similarity, respectively. The result of this process is the desired map of Use Cases.

4 The Application of Document Maps in Software Engineering – A Case Study

Now the application of the document map concept for software engineering projects will be discussed. We use the Use Cases generated during the CAPE-OPEN project as a real-world example to present the benefits of this approach. CAPE-OPEN is an EU funded project with participants from the chemical industry (BASF, Bayer, BP, DuPont, ELF, ICI), software vendors (AspenTech, HyproTech, QuantiSci) and universities (Imperial College London, INPT Toulouse, RWTH Aachen) under co-ordination of the French process licensing company IFP. It aims at defining a new standard for high-performance process simulation software. Process simulators are highly sophisticated pieces of software designed for creating mathematical models of manufacturing facilities for processing and/or transforming materials (chemical, oil, food). These tools have become vitally important for chemical engineering companies for several reasons: The market is rapidly growing while innovation cycles are shrinking. This pressure is strengthened by a growing sensitivity for environmental issues.

The process simulators that are currently in use are closed monolithic applications. They are quite inflexible when it comes to integrating new components. Another drawback of this situation is that it is almost impossible to combine modules from different vendors into one single simulator. In practice, such a combination is of high interest due to the limitations of individual products. Therefore, the CAPE-OPEN standard aims at creating a framework for open component based simulation software. To identify the components in such an open simulator the Use Case approach was applied.

Before going into detail, we will give a brief introduction to the subsystems of a process simulator. This will render a basis for the discussion of the Use Case clustering. We will discuss the CAPE-OPEN Use Case map on a general level before selecting a specific area and performing a more detailed analysis.

A Conceptual View on Process Simulators

Simulators differ widely in architecture and implementation but all have common functionality imposed by the underlying modelling tasks which they address. This functionality can be summarised in terms of four key 'conceptual' component types:

- *Simulator executive:* This component is the simulator's core as it controls the set-up and execution of a simulation. It is responsible for installing other components, registering them in a repository, managing interactions with users, accessing and storing data, and, finally, for reporting and analyzing simulation calculations. Furthermore, it is responsible for a consistent flowsheet set-up, error checking and preparatory work on solving it (graph analysis).
- *Unit Operation Modules:* These components represent the behavior of physical process steps (e.g. a mixer or a reactor). They are linked to the simulation flowsheet which represents an abstraction of the plant structure. They compute the quality of a material stream of their outlet if the according information is given at the inlet. The simulation models are assembled from predefined libraries of unit operation modules into a flowsheet which represents the overall plant and is handled by the simulator executive.
- *Physical properties packages:* An important functionality of a process simulator is its ability to calculate thermodynamic and physical properties (e.g. density or viscosity) of materials. Thermodynamic packages are complex and highly optimized pieces of software. Since they provide the basic calculations for all unit operations, the overall performance and quality of a simulator strongly depends on its thermodynamic package. It is estimated that up to 75% of the simulation time is spent for these calculations [5].

- *Numerical solvers:* This includes both the specialised mathematical methods used to evaluate the equations that describe a *unit operation* (unit solving) and the methods used to evaluate the overall flowsheet (flowsheet solving).

This coarse structure of a process simulator was the starting point for the generation of the Use Cases which, of course, should yield a finer subdivision of a simulators functionality.

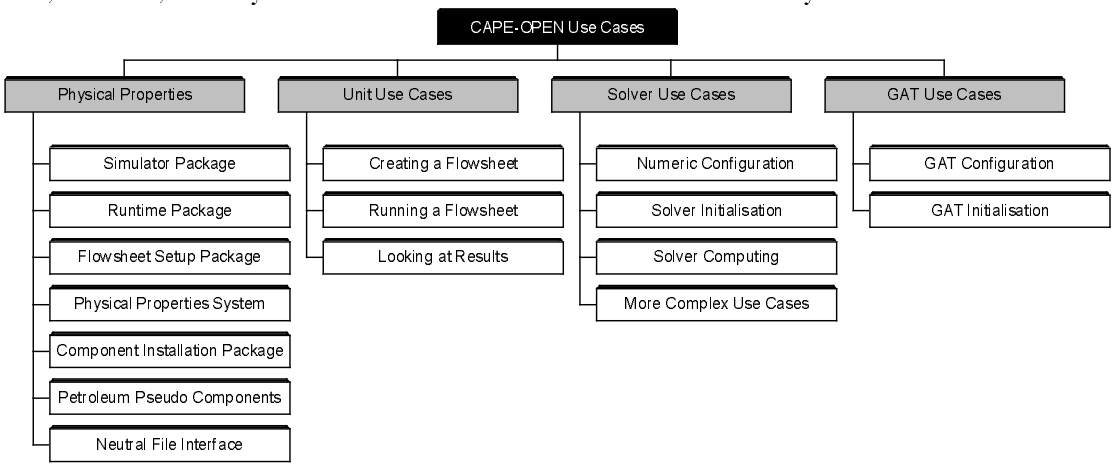

Figure 3: Hand-crafted Use Case Hierarchy for CAPE-OPEN

Use Cases in CAPE OPEN

Use Cases are used in CAPE-OPEN to represent the requirements for the three most important subsystems of a process simulator namely Unit Operations, Physical Properties and Numerical solvers. Because it is one goal of the new standard to reduce the size of the executive no Use Cases were created for it. Nevertheless, an important feature of a simulator executive had to be taken care of: graph analysis. A graph analysis tool (GAT) checks how a flowsheet can be solved by determining how to break up cycles in it.

Up to now more than 160 Use Cases have been developed in CAPE-OPEN. The process of developing these Use Cases was distributed and carried out by people coming from different organizations. Hence, there was the permanent danger of producing redundancies and inconsistencies. To overcome this problem, all Use Cases were agreed upon by the consortium and finally put into the hierarchy. In the next section we will compare this hand crafted structure with the contents of the automatically generated semantic maps.

A Semantic Map of Use Cases

Looking at the 'Use Case landscape' (figure 4) the user can identify four major areas, each of which is subdivided into smaller areas containing sub-groups of documents. Figure 5 presents the map of Use Cases where the document representatives are marked by an icon. Each icon type identifies one of the sub-groups in the Use Case hierarchy and every icon represents one Use Case belonging to the corresponding sub-group. The four areas are separated by 'deep' dark ditches reflecting the *a priori* subdivision.

Taking a closer look we can identify the physical properties Use Cases in the north-western area of the map. The numerical solver Use Cases are located in the center and the GAT Use Cases can be found in

the south-western sector. Finally, the unit operation Use Cases are on the eastern edge of the map. Apart from this general view there are some areas, where Use Cases of different type are mixed. This expresses an interrelation between the sub-groups of the manually generated hierarchy thereby yielding additional information. We can derive from this that the four main groups are not standalone but are somehow connected. This is an advantage of the semantic map over the pure hierarchy and we will now explain where these connections originate from.

First, we look at the middle of the northern edge of the map. There we have a mixture of properties package and unit operation Use Cases. As mentioned above, the routines offered by a physical properties package form the basis of the calculations performed in a unit. The mixture of both Use Case types reflects this fact. In particular, these Use Cases describe on the one hand that a unit calls a calculation routine (e.g. calculate pressure, temperature,…) within a properties package. The according properties package's Use Cases describe a function call from a unit.

Now we consider the north-eastern corner of the map. Here, unit Use Cases and numerical solver Use Cases are mixed up. Unit operations do not only perform pure thermodynamical calculations carried out by a properties package but sometimes have solve difficult equation systems. This is the connection between both Use Case types. The unit calls the numerical solver to create an equation system, chooses some initial values and starts the solving process. On the other side the solver accepts these calls and returns the values of it's calculations. This is all reflected by the conglomerate of different Use Cases in the north-eastern corner.

Having explained what additional information about the interrelation of different Use Case types can be found in the semantic map we will now investigate how the sub-groups in the hand crafted Use Case hierarchy are reflected in the semantic map. Therefore, we will take a closer look at unit Use Cases in the eastern area. Recall that we have three sub-groups of unit Use Cases: The *Creating a Flowsheet* group tackles how a unit is created, deleted and initialized and how it is combined with other units to model a process. We can see the creation / deletion Use Cases in Figure 4 (*Save, Restore, Create, Delete Unit*). The initialization is handled in *Set Unit Specific Data.* The combination with other units is done via so called *ports* and via the flowsheet. The corresponding Use Cases are *Delete Unit From Flowsheet, Delete Existing Port* and other Use Cases dealing with ports and flowsheets which are not marked in Figure 4. The *Looking at Results* sub-group can be found in the *Creating a Flowsheet* group. Because we have only one Use Case in this sub-group we cannot derive any relevant semantic information from its position.

Looking at the *Running a Flowsheet* sub-group we can make an interesting observation: There are two strictly separated clusters of Use Cases in the north-eastern and the south-eastern corner of the map. The northern Use Cases are mixed with the solver Use Cases for reasons we have explained above. These unit Use Cases are mainly concerned with creating and initializing equation systems that are needed *internally* in the unit. The southern unit Use Cases tackle how data from the results of these internal calculations is communicated to other units. This information is needed to solve the overall flowsheet, i.e. running the simulation on flowsheet level. This also explains that these unit Use Cases are located nearby the GAT Use Cases which form the basis of the overall flowsheet solving. Hence, we have found an additional subdivision of a manually generated sub-group which we could call *Internal Unit Solving* and *Flowsheet Solving*.

To sum up, it turns out that the automatically derived Use Case landscape reflects the expert structure – a good validation of the approach. Furthermore, it provides additional information about the sub-groups themselves and their interrelations. One the one hand there are areas where different Use Case types are mixed which we can interpret as Use Case interrelations. We have verified these connections and explained them. On the other hand we have discovered that a sub-group could have been split up into two new groups because the corresponding Use Cases were located in two separated clusters. The semantic justification for this has been shown.

The calculation of the map required 14 seconds for preprocessing the documents and creating the semantic documents space and 1 minute 25 seconds for training and visualizing the neural network on a

Pentium II 450 MHz computer with 128 MB RAM. Hence, it would have been possible to use this clustering algorithm in the process of setting up a hierarchy. It could have formed a basis and an aid for the people who have created the structure shown in figure 3.

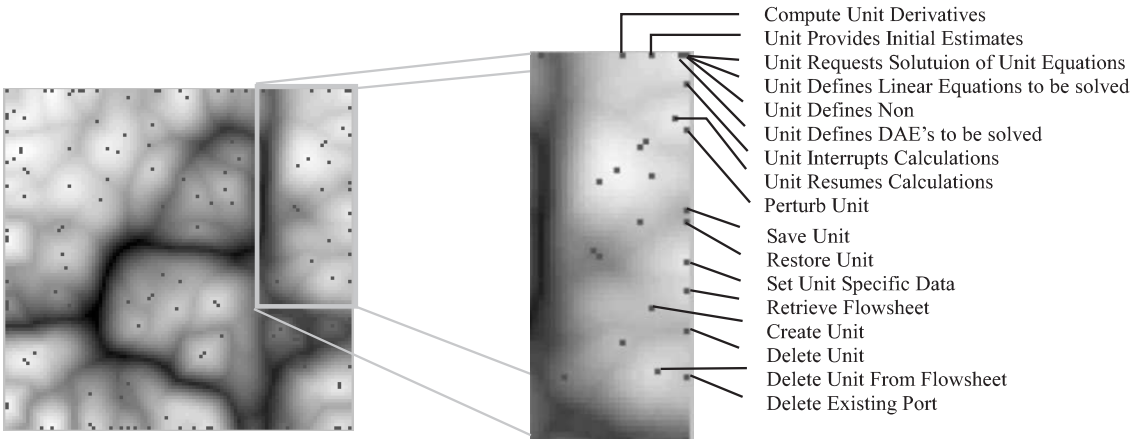

Figure 4: Use Case landscape. The detail shows a part of the 'Unit' Use Case group, describing physical processing unit operations. Whereas the inscribed documents at the top are concerned with calculation matters the sub-group below deals with handling 'unit' software objects.

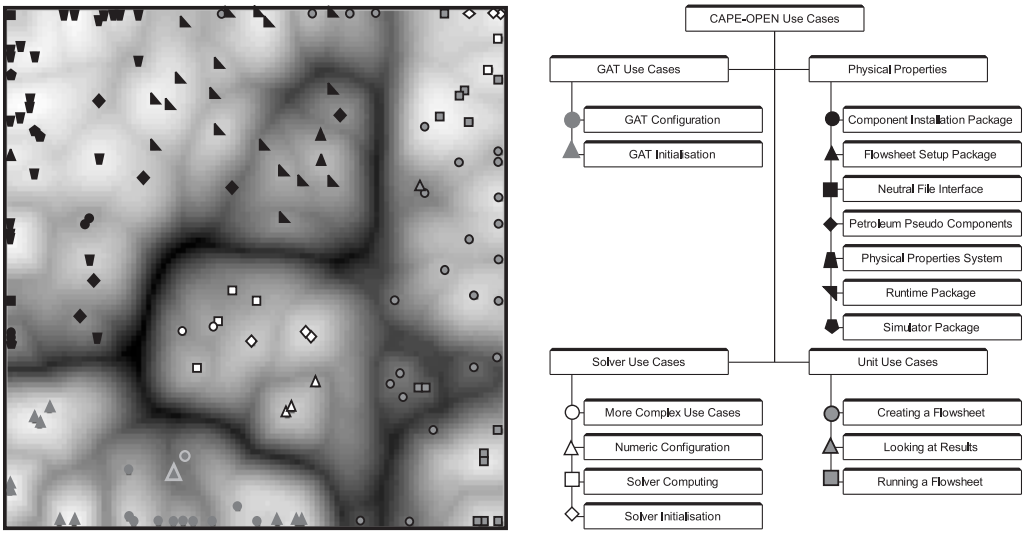

Figure 5: Landscape of Use Cases (left), marked by their membership in a manually created expert structure (right).

5 Conclusion and Outlook

Scenarios are of high value for supporting communication among system developers in the initial phases of requirements engineering. Furthermore, they are seen as key element for change management. A crucial point in this context is handling large collections of weakly structured textual scenarios such as Use Cases. In this paper we have addressed some problems coming along with the scenario approach, namely synchronizing knowledge elicited by collaborative working groups, analyzing and condensing informal scenarios, and maintaining them in a growing repository.

We applied our method of automatically structuring text documents to informal scenario collections and evaluated the approach using Use Cases from the CAPE-OPEN project. It turned out that the automatically derived structure reflects the main groups of the manually generated hierarchy. Furthermore, the system engineers gain additional insight in the scenario interrelations. The plausibility of the computed structure was verified by comparing the semantics of the textual scenarios with the information extractable from the document map. Thus, our approach has proven to be a valuable support for analyzing and condensing scenario collections in real-world problems.

Reconsidering the problems of distributed scenario generation and maintenance the Use Case map can be used as a tool integrated in a centralized repository. There, the map gives scenario developers hints regarding the correlation of newly generated scenarios within the general context. Additionally, it aids the author in avoiding inconsistency and redundancy of his descriptions.

Of course, the map alone offers important but limited functionality. The process of detecting inconsistencies and redundancies is aided by the map but the finer work of figuring out why unexpected similarities have occurred has still to be done manually. Therefore, suitable inference techniques have to be developed which require a translation of natural language Use Cases into a more formal representation and need to be supported by domain semantics.

Our future work includes the provision and integration of explicit defined domain semantics. This allows, for example, to define semantic relationships between simulator components which are then considered by the automatic structuring process for further enhancement of the derived structures. Furthermore, we intend to incorporate “views” on document collections so that the structuring can be adjusted respecting a focus of interest.

Acknowledgements. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) in its focused doctoral programme on Informatics and Engineering at RWTH Aachen. Furthermore, parts of this work were supported by the Commission of the European Union under BRITE-EURAM project CAPE-OPEN. The authors wish to thank J. Rack for extensive and fruitful discussions on the subject.

6 References

1. Allan, James, Leouski, Anton V., Swan, Russell C. Interactive Cluster Visualization for Information Retrieval. Tech. Rep. IR-116, Center for Intelligent Information Retrieval, University of Massachusetts, 1997
2. Allan, James. Automatic Hypertext Construction, PhD Thesis, Cornell University, 1995
3. Becks, A.; Sklorz, S., Jarke, M.: Document Maps: Semantic Structuring of Technical Document Collections. Crews Report 99-05, RWTH Aachen, 1999. Available at http://SunSITE.Informatik.RWTH-Aachen.DE/CREWS/reports.htm
4. Becks, A.; Sklorz, S.; Tresp, C.: Semantic Structuring and Visual Querying of Document Abstracts in Digital Libraries. In: Lecture Notes in Computer Science 1513: Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries, Crete, Greece, 1998, pp. 443-458
5. Brice A, Johns W R. Open Process Simulation. OO-CAPE report. QuantiSci report IC4381-2, 1995

6. Chalmers, Matthew, Chitson, Paul. Bead: Explorations in Information Visualization. In: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, 1992, pp. 330-337
7. Fischer, Bernd. Specification-Based Browsing of Software Component Libraries. Proc. of ASE-98: The 13th IEEE Conf. on Automated Software Engineering, Honolulu, Hawaii, 1998
8. Girardi, M.R., Ibrahim, B. An approach to improve the effectiveness of software retrieval. Proceedings of the 3rd Irvine Software Symposium, Irvine, CA, 1993
9. Jacobson, Ivar; Christerson, Magnus; Jonsson, Patrik; Övergaard, G.: Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, Reading, 1992
10. Jarke, M.; Becks, A.; Köller, J.; Tresp, C; Braunschweig, B.: Designing Standards for Open Simulation Environments in the Chemical Industries: A Computer-Supported Use-Case Approach. To appear in: Systems Engineering - Sharing the Future: Proceedings of the Ninth Annual International Symposium of the International Council on Systems Engineering, Brighton, England, June ,1999
11. Jarke, M.; Tung Bui, X.; Carroll, J. M. Scenario Management: An Interdisciplinary Approach. Requirements Engineering 3:155–173, 1998
12. Kohonen, T.: Self-Organizing Maps. Springer, Berlin, 2nd Edition (1995)
13. Korfhage, Robert. Information Storage and Retrieval. Wiley & Sons, New York, 1997
14. Kruskal, J.B.,Wish, M.: Multidimensional scaling. SAGE publications, Beverly Hills, 1978
15. Maarek, Y.; Berry, D., Kaiser, G. An Information Retrieval Approach for Automatically Constructing Software Libraries. IEEE Transactions on Software Engineering, 17(8), pp. 800–813, 1991
16. Pohl, K.; Haumer, P. HYDRA: A Hypertext Model for Structuring Informal Requirements Representations. Proc. of the 2nd Int. Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ 95), Jyvskyl, Finnland, 1995
17. Rumbaugh, J.;Jacobsen, I. and Booch,G., *Unified Modeling Language Reference Manual*, Addison Wesley, 1997.
18. Salton, G. (Ed.): The SMART Retrieval System – Experiments in Automatic Document Processing. Prentice Hall, New Jersey, 1971
19. Sklorz, S.: A Method for Data Analysis based on Self Organizing Feature Maps., Proc. of the World Automation Congress (WAC '96), Vol.5 TSI Press Series, 611-616, ISBN 1-889335-02-9, Albuquerque, USA (1996)
20. Weidenhaupt, Klaus; Pohl, Klaus; Jarke, Matthias; Haumer, Peter: Scenarios in System Development: Current Practice. IEEE Software, March/April 1998