Implicit Analogy-Based Cost Estimation Using Textual Use Case Similarities

M. Auer, C. Becker, A. Rauber, S. Biffl

Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstr. 11, A-1040 Vienna, Austria
e-mail: martin.auer@tuwien.ac.at
tel: ++43 676 ----------
fax: ++43 1 58801 18899

Abstract

Software cost estimation is a ubiquitous task in software project portfolio management. Analogy-based estimation approaches are particularly suitable to reuse past project experience to create estimate proposals.

However, to find project data of past, similar projects, one usually must rely on similarity measures based on explicit project metrics, like a team's project experience, the number of interfaces, etc. This requires the consistent collection of such project metrics over time, which can be tedious, error-prone and expensive.

This paper proposes an alternative approach: to use text artifacts arising in the regular course of a project to determine the similarity of project entities. Use cases' text descriptions are used as entities; a variation of self-organizing maps is used to determine their similarities.

The paper points out possible applications and limitations of the approach, analyzes several real-world use case sets, describes problems and suggests rules for writing suitable use cases.

Neural Networks, Software Engineering, Data Mining, Data Visualization, Information Engineering

1 Introduction

Most relevant software portfolio decisions—like resource allocation, bidding, or start scheduling—rely on supporting procedures like cost estimation or risk assessment. Usually, a key for successful decision support is seen in software and process measurement, i. e., the definition, collection, storage and analysis of past project features, preferably in quantitative metrics.

Yet in many industrial environments such explicit measurement programs are never implemented, or fail. The reasons, outlined in more detail in the following section, are manifold, but they can be summed up: project team members often perceive such programs as economically inefficient, at least in the short term.

However, in certain circumstances *explicit* measurement might be unnecessary; instead, *implicit* metrics can be used, merely derived from artifacts that are created anyway during the development process, yielding valuable information without or with neglectible extra cost.

One area we deem suitable for such an approach is analogy-based cost estimation. It tries to identify past projects that are similar to the project to estimate, and to create an estimate proposal by using the historical cost data. Traditionally, analogies are defined as distances of explicitly collected project features or metrics.

This paper proposes to use the textual description of requirement documents—more precisely, structured chunks of documents, use cases—to determine implicitly the similarity of various requirements, allowing for a fast search of similar past project requirements without explicit project metrics. We try to evaluate this approach on several industrial use case sets, and find out, whether those sets can be used without substantial modification in writing them.

Section 2 describes related work on cost estimation, use cases and digital libraries, with section 3 introducing self-organizing maps, used to organize the use cases by semantic similarity. Section 4 presents the use cases sets used. Section 5 gives the results. Section 6 discusses the lessons learned and outlines the benefits of the approach. Finally, section 7 gives an outlook on further research in this area.

2 Related Work on Cost Estimation, Use Cases and Digital Libraries

Cost estimation [11] is a vital task in software project portfolio management. A variety of different methods have been proposed to support cost estimation: algorithmic models like COCOMO 2 [4], neural networks [3], expert judgement [9], or analogy-based approaches [22].

The fundamental principle of those methods is that similar projects are likely to have similar cost characteristics—this is expressed most directly by the analogy-based approach, where estimates are derived from historical project information. One or several similar past projects are selected according to a simple distance function δ on the l project features or metrics $d_1, \ldots, d_l$ (scaled to a unit interval with s_i, and weighted according to a metric's influence with w_i):

$$\delta(p, p') = \sqrt{\sum_{i=1}^{l} w_i s_i^2 (d_i - d_i')^2} \quad (1)$$

The actual costs of the selected projects with the smallest distance are then used to estimate the new project; visualization methods like multidimensional scaling [2] can be used to further support estimators.

While this method is straightforward and highly transparent (considered an important feature for the real-world application of a cost estimation method [7]), it has a main drawback: it requires the consistent collection of software project metrics over a long period of time, as only company-specific data can reasonably be expected to yield high-quality estimates [10].

However, defining and collecting such software metrics explicitly is notoriously difficult. Maxwell [14] describes several common pitfalls in software measurement. Jeffery et al. [10] describe data collection as an "expensive and time-consuming process for individual organizations". Auer et al. [1] point out, that the necessary automation of data collection can be difficult to achieve in highly heterogeneous software development environments. Ruhe et al. [19] declare that as data collection has to be carefully planned, many organizations simply "do not have enough resources for the required measurement methods".

It would be preferable to assess project or project artifact similarities implicitly, without relying on explicit metrics. Several methods can be applied [5]; one possibility is to measure textual similarities between text documents created in the development process. Users benefit because similar documents are uncovered in an intuitive way, with no need for additional metric data collection. This approach is evaluated within the framework of the *SOMLib* project [1] [16, 17]. The *SOMLib* project is based on unsupervised neural network technology for the task of document archive organization. In particular, the approach relies on the *Self-Organizing Map* (*SOM*) [12, 13], and variants thereof, such as the GH-SOM [18], providing a map-based representation of a document archive. In such a representation, documents dealing with similar topics are located next to each other. The obvious benefit for the user is that navigation in the document archive is similar to the well-known task of navigating in a geographical map. Previous applications of *SOM* were, for instance, knowledge discovery in Web navigation patterns [23].

Certainly, it is not suitable to compare whole project requirement documents to each other—yet standardized, smaller text fragments could be used, if the way they are created is sufficiently standardized. A potential candidate are *use cases* [6], which are gaining importance in eliciting and documenting user requirements. The use case methodology is applied in a variety of areas, ranging from e-commerce applications [21] to embedded systems [15]. Other potential text artifacts suitable for this approach include user stories or scenarios; however, this paper analyzes industrial use case sets.

3 Text-Based Analogies

Self-organizing maps (*SOM*) are a way to map high-dimensional feature sets to lower-dimensional representations. This section describes the *SOM* approach in detail.

Indexing

In order to utilize the *SOM* for organizing documents by their topic, a vector-based description of the content of the documents is created. Instead of manually or semi-automatically extracted content descriptors, a rather simple word frequency based description is sufficient to provide the necessary information. For this word frequency based representation a vector structure is created consisting of all words appearing in the document collection. This list of words is usually cleaned from so-called stop words, i. e., words that do not contribute to content representation and topic discrimination between documents. Simple statistics allow the removal of most stop words in a very convenient language- and subject-independent way (words appearing in too many documents can be removed without the risk of loosing content information; words appearing in less than a minimum number of documents can be omitted, as well).

The documents are described within the resulting feature space of commonly between 2,000 and 15,000 dimensions, i. e., distinct terms they are made up of. Binary indexing may be used to describe the content of a document by simply

[1]The *SOMLib* project homepage is available at `http://www.ifs.tuwien.ac.at/~andi/somlib`

stating whether a word appears in the document, or more sophisticated schemes can be used, such as $tf \times idf$, i. e., term frequency times inverse document frequency [20]. This weighting scheme assigns higher values to terms that appear frequently within a document, i. e., have a high term frequency, yet rarely within the complete collection, i. e., have a low document frequency. Usually, the document vectors are normalized to unit length to make up for length differences of the various documents.

The Self-Organizing Map

The self-organizing map [12] provides cluster analysis by producing a mapping of high-dimensional input data $x \in \mathbf{R}^n$, onto a usually 2-dimensional output space while preserving the topological relationships between the input data items as faithfully as possible. This model consists of a set of units, which are arranged in some topology, where the most common choice is a two-dimensional grid. Each of the units i is assigned a weight vector m_i of the same dimension as the input data, $m_i \in \mathbf{R}^n$. These weight vectors are initialized with random values.

During each learning step, the unit c with the highest activity level, i.e. the *winner c* with respect to a randomly selected input pattern x, is adapted in a way that it will exhibit an even higher activity level at future presentations of that specific input pattern. Commonly, the activity level of a unit is based on the Euclidean distance between the input pattern and that unit's weight vector. The unit showing the lowest Euclidean distance between it's weight vector and the presented input vector is selected as the winner.

Adaptation takes place at each learning iteration and is performed as a gradual reduction of the difference between the respective components of the input vector and the weight vector. The amount of adaptation is guided by a learning rate α that is gradually decreasing in the course of time. This decreasing nature of adaptation strength ensures large adaptation steps in the beginning of the learning process where the weight vectors have to be tuned from their random initialization towards the actual requirements of the input space. The ever smaller adaptation steps towards the end of the learning process enable a fine-tuned input space representation.

As an extension to standard competitive learning, units in a time-varying and gradually decreasing neighborhood around the winner are adapted too. The neighborhood of units around the winner may be described implicitly by means of a (Gaussian) neighborhood-kernel h_{ci} taking into account the distance—in terms of the output space—between unit i under consideration and unit c, the winner of the current learning iteration. This neighborhood-kernel assigns scalars in the range of [0, 1] that are used to determine the amount of adaptation ensuring that nearby units are adapted more strongly than units farther away from the winner. A Gaussian may be used to define the neighborhood-kernel, where $||r_c - r_i||$ denotes the distance between units c and i within the output space, with r_i representing the two-dimensional vector pointing to the location of unit i within the grid:

$$h_{ci}(t) = e^{-\frac{||r_c - r_i||^2}{2 \cdot \delta(t)^2}} \tag{2}$$

It is common practice that in the beginning of the learning process the neighborhood-kernel is selected large enough to cover a wide area of the output space. The spatial width of the neighborhood-kernel is reduced gradually during the learning process such that towards the end of the learning process just the winner itself is adapted. Such a reduction is done by means of the time-varying parameter δ. This strategy enables the formation of large clusters in the beginning and fine-grained input discrimination towards the end of the learning process.

With α representing the time-varying learning rate, h_{ci} representing the time-varying neighborhood-kernel, x representing the currently presented input pattern, and m_i denoting the weight vector assigned to unit i, one obtains the following learning rule:

$$m_i(t+1) = m_i(t) + \alpha(t) h_{ci}(t)[x(t) - m_i(t)] \tag{3}$$

Pragmatically speaking, during the learning steps of the self-organizing map a set of units around the winner is tuned towards the currently presented input pattern enabling a spatial arrangement of the input patterns such that alike inputs are mapped onto regions close to each other in the grid of output units. Thus, the training process of the self-organizing map results in a topological ordering of the input patterns.

An extension to *SOM*, especially for large, growing sets of documents, is the growing hierarchical *SOM*. Additional parameters determine the behavior of the grow process. Please refer to [18] for more details.

4 Use Case Set Description

Unsurprisingly, it is non-trivial to get access to real-world instances of use cases in industrial environments. While several companies and institutions do actively apply use cases in describing user requirements (indeed, more than we initially expected), they are reluctant to give access to those use case sets, often even under terms of non-disclosure agreements. Similar experiences were indicated to us by several academics working in the field of use case applications.

In addition, some basic criteria had to be met by the use case sets, further narrowing the number of settings for analysis:

Name	**Domain**	**Number of use cases**	**Avg. use case size, in characters**	**Std. dev. use case size**
TICKET	Administration	66	1396	460
AUTO	Automotive industry	20	1054	365
COLLAB	Collaboration, document management	26	657	244
MOBILE	Mobile communication	448	1814	642

Table 1. Properties of use case sets

- *Size.* A set of use cases should contain sufficient use cases to make analogy-based estimation feasible. Several sets containing less than 20 use cases were dismissed.
- *Relevance.* A set should consist of real-world use cases, used in industrial environments, and not be artificially constructed. It should involve large institutions with a software engineering track record, not just small software producers.
- *Structure.* The use cases should more or less conform to structured use case templates and not be totally informal text fragments describing some ill-defined system aspects.
- *Readiness.* The use cases should—except for conversion to txt-files, and replacement of special characters—be ready to use with the neural network techniques applied; it should not be necessary to prepare or substantially change the use cases—this would not be possible in a real-world environments and reduce the efficiency gains of implicit measurement of existing artifacts to nothing.

We finally obtained access to four suitable sets of use cases; they are described in the following. Company names and details had to be omitted due to the terms of confidentiality agreements.

- *Set 1: TICKET.* TicketLine is a ticket reservation system developed at the Vienna University of Technology, primarily by graduate students, for use in both graduate and undergraduate courses. Three main releases were developed over the years, each with state-of-the-art methods and terminology; release 3 used use cases as primary requirement description approach. The use cases are in German; special characters were removed prior to the clustering. This set contains 66 use cases.
- *Set 2: AUTO.* Here, a control application in the automotive industry is described by use cases. The original specification on which the use case set is based was provided by a very large, international car manufacturer. The use case description was developed at a large German software engineering institute. This set contains 20 use cases (in English).
- *Set 3: COLLAB.* This is a large collaboration and document management framework. Project partners in this joint effort included an international company that is a large producer of white goods, and a very large international company in the market of global telecommunication systems and equipment. This set contains 26 use cases (in English).
- *Set 4: MOBILE.* This large use case set describes the behavior of software functionality in the field of mobile communication. It was created at one of the largest international organizations operating in communication, power infrastructure, and electrical engineering (not the company involved in set 3). This set contains 448 use cases (in English).

The use cases were extracted from the requirement documents and converted (from doc or pdf) to plain-text txt-files. Introductions, use case diagrams etc. were omitted. Table 1 gives an overview of the four sets, along with average use case size, and variance of use case size in characters.

5 Results

This section presents the maps generated by the self-organizing map approach, and gives the parameters used.

Figure 1 represents the map of the TICKET use case set. The sets' identifiers are abbreviations of their content, e. g., "Suc" stands for "Suche" ("search"). Visibly, similar abbreviations are clustered together. It is important to note that these identifiers or abbreviations are not part of the use cases' text, and thus are not analyzed by the *SOM*; they merely serve to identify use cases and to assess the result of the clustering process.

Figures 2 and 3 represent the smaller use case set AUTO and COLLAB, respectively. Here, we modified the identifiers to ensure compliance with the confidentiality agreements.

Table 2 gives an overview of the parameters used:

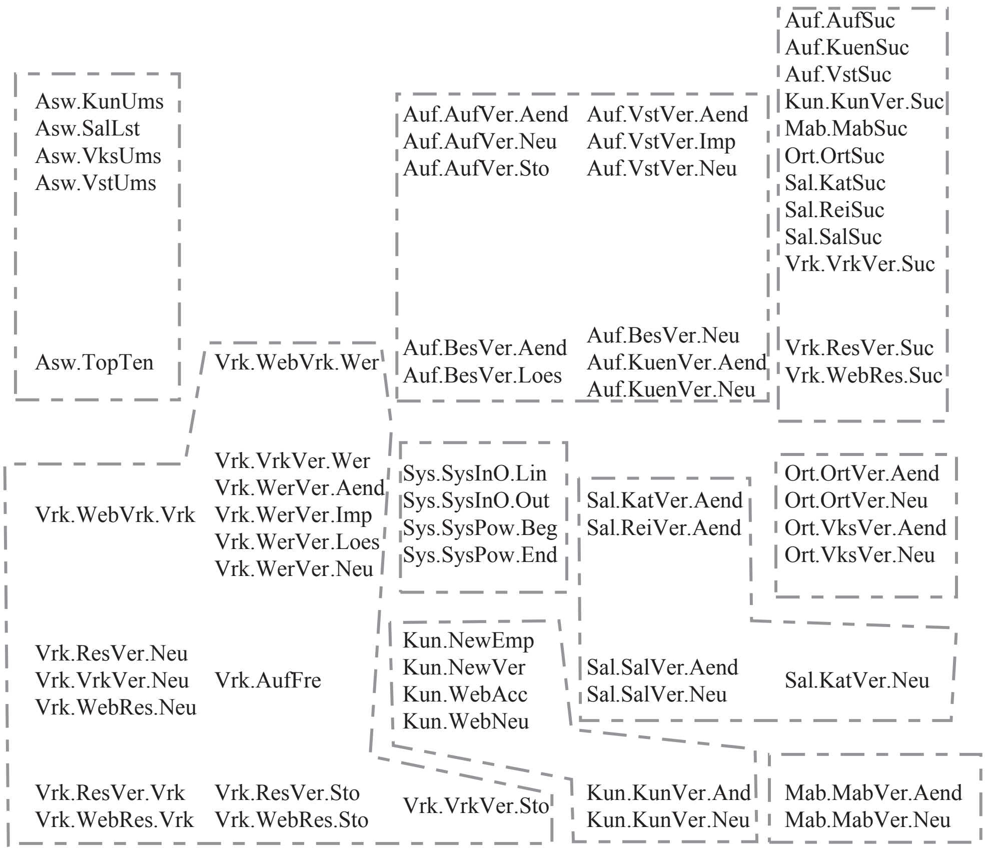

Figure 1. Map of use case set TICKET

	AUTO	COLLAB	TICKET	MOBILE
SOM mode	static	static	static	GHSOM
Dimensions	163	106	313	466
Lower limit	0.05	0.09	0.05	0.03
Upper limit	0.7	0.5	0.7	0.7
Map size	3x3	4x4	5x5	N/A
Iterations	15000	15000	50000	N/A
MQE	6.48	3.03	8.7	N/A

Table 2. ***SOM*** **parameters**

- "*SOM* mode" describes, whether a static or growing hierarchical *SOM* was used; the latter is particularly useful for very large document sets and was thus used for the MOBILE use case set.
- "Dimensions" gives the number of words used to index the use cases.
- "Lower/upper limit" are the %-limits for automatic stop-word definition: if a word appears in too few documents ("lower limit"), or in too many ("upper limit"), it is not included in the set of indexed words.
- "Map size" is the grid size of the 2-dimensional map.
- "Iterations" gives the number of iterations used to generate the map.
- "MQE" is the mean quantization error of the cells contained in a map. The quantization error of a cell measures the dissimilarity of all input data mapped onto a particular unit; the comparison metric is Euclidean [18].

In all maps, the identified clusters were marked manually.

6 Discussion

Although there are many different approaches to support people in estimating software project efforts, few of them are actually applied in typical industrial environments. One of the main reasons clearly is that estimation usually needs to rely on explicitly collected measurement data which is costly and time-consuming to collect.

Moreover, many estimation methods lack of transparency and accessibility. Black-box approaches methods, e. g., relying on neural nets, often provide little insight on how they reach a certain estimate, and do little to foster portfolio measurement data understanding.

Analogy-based methods rely on similarities between projects expressed as distances between (often interdependent) attribute sets. Humans, however, are not particularly good at analyzing complex data without the aid of visualization techniques. Thus common tools supporting analogy-based methods, e.g., spreadsheet applications, offer very limited benefit.

Thus, this paper proposed a transparent way of visualizing the similarity of use cases without the need of explicit measurement data collection; an implicit analogy metric is obtained by using textual similarity.

This is achieved using self-organizing maps, which have a strong tradition in the text mining area. They are a very powerful tool for detecting structure in high-dimensional data and organizing it accordingly on a two-dimensional output space. Several of its proven properties seem to be highly useful and supportive in the context of project portfolio cost estimation and decision support, where users should benefit particularly from clustering techniques that uncover similar documents and bring these similarities to the user's attention, allowing for: the co-location of similar, yet not identical topics; the content-based organization to facilitate browsing of documents according to subject hierarchies; the topology-preserving representation of input similarities in terms of distances in the output space; and the ease by which a user gains an idea regarding the structure of the underlying data by analyzing the map [17].

But what is the difference to merely using a standard text indexing and analyzing the nearest neighbors of a given use case? The 2-dimensional representation of the use case set TICKET can illustrate several key benefits of the graphical clustering.

Using merely the most similar text documents does not give an impression of a particular cluster size. On the upper left, or on the lower right of figure 1, for example, relatively small clusters of use cases can be found. On the other hand, on the upper right, a larger number of use cases is clustered together. In the use case set AUTO given in figure 2 most clusters (except the 4-use case cluster at the top of the map) consist of two use cases. This becomes immediately visible on the map, but remains invisible using mere text indexing and a list of the most similar documents. This information influences how many neighboring use cases should reasonably be included in the analysis.

A special case are surely isolated instances of use cases, which differ that much from others that they do not appear in any cluster. For example, denoting the lower left cell in figure 1 with $(0, 0)$, such use cases can be found at $(1, 1)$, $(4, 1)$, or $(1, 3)$ in the TICKET use case set, or on the upper left part of the COLLAB use case set given in figure 3. Analyzing such use cases—in our case, estimating related costs using cost information of similar/near use cases—should be done with particular attention as the most similar use cases are already significantly different and their cost information may thus be misleading. Again, a conventional list of the nearest documents would not yield this important information of a use case being isolated.

Another interesting aspect is that the main entity of a use case (denoted by the first 3 characters of the use case identifier, e. g., customer, show, system, etc.) is not determining the clustering. For example, the text describing the procedure of searching for entities (identified by the suffix "Suc") exhibits more textual similarities, overriding the entities' identification similarity and yielding the cluster on the upper right. The search logic was described in greater detail; it seems plausible that estimating a new search use case should analyze other entities' search use cases.

The benefits of visualizing portfolio data in the proposed way could be categorized as follows:

- *Efficient measurement.* No costly consistent collection of project data according to predefined explicit metrics is necessary; instead, artifacts arising during the regular course of requirements analysis together with implicit analogy metrics are used. Thus applying the proposed method is highly efficient: All that is required to visualize a set of use cases is to extract them from existing documents and find optimal parameter settings. Costs and complexity, and thus also the inhibition level of introducing the method into practice, are substantially reduced.

- *Efficient to learn.* The proposed method is straightforward and transparent; even estimators not acquainted with it can immediately grasp the process and the natural visualizations' implications.

  Moreover, as no definition of metrics, model configuration or difficult-to-reproduce algorithms are involved, users are far more likely to accept and apply this method in the first place.

- *Efficient to use.* In practice, one often has to deal with a yet more or less unknown project structure. This leads to one of the main strengths of the *SOM* [18]: It gives the user a natural visualization and allows him or her to build a mental model of the underlying project structure facilitating convenient exploration of (probably unknown) past project data. Therefore, orientation for the estimator is highly improved, and it is easy to gain a fast overview of a project's properties, its functionality's cluster structures and sizes. This applies for team members new to a project as well as experienced managers and estimators tackling a new project.

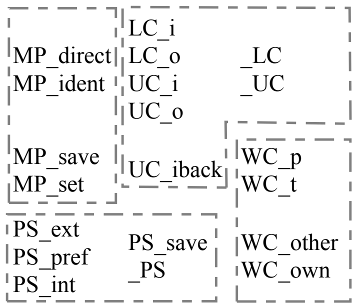

Figure 2. Map of use case set AUTO

> For example, in the MOBILE project consisting of more than 440 use cases, the growing hierarchical self-organizing map provides the user with a roadmap witch separates main topical branches, guiding him or her through the various topical sections. It thus substantially reduces the effort needed to gain an overview of the internal project structure and allows comfortable browsing of requirements hierarchies.

These properties of the proposed visual representation—transparency and simplicity, overview and understanding—also seem to be of particular relevance for industrial practice as expert judgement based on informal analogies between projects is by far the most frequently applied estimation technique [8].

The authors are also aware of several instances in industrial environments where estimation was hindered by its relation to software measurement being perceived differently by various stakeholders—by using self-organizing maps to visualize implicitly collected clusters of requirements, the connection between demands and effort becomes transparent, and estimates are easier to agree upon.

Limitations

A company using this approach to quickly find similar past requirements in order to estimate new ones should implement a requirement database; manual document handling and conversion from doc to txt file formats would nullify efficiency gains by this implicit analogy-based approach.

If use cases are written by different people, their language and wording could influence the clustering; therefore important terms should be defined and managed centrally. (This is helpful to obtain consistent requirements, anyway.)

If use case templates are used, e. g., containing section like “precondition” etc., empty sections should not be removed. Thus, the section name will be correctly identified as stop-word and won’t influence the clustering. Authors’ names, however, should be removed from the use case text, potentially requiring a change in the use case template and database.

To keep the system as flexible and generally applicable as possible, no language- or domain-specific optimizations for the content-based analysis, such as the use of specific stop-word lists, were performed. Language-specific adaption may further improve the resulting clustering, albeit sacrificing the language and domain independence [18].

7 Conclusion and Outlook

This paper presented the application of self-organizing maps to specific requirement artifacts, use cases, in order to implicitly determine analogies between new and historic software requirements for software cost estimation. The main motivation was to avoid costly explicit software measurement.

The approach was tested on several real-world industrial use case sets. The clusters obtained did indeed conform to our expectations: use cases dealing with similar topics or aspects of a projects were visibly clustered together.

In our view, this approach should be particularly helpful within a well-defined environment, for example, a company, which performs a large number of similar projects in comparable circumstances. Using the approach on singular, innovative projects, or comparing projects of different environments are unlikely to yield satisfactory results.

Further research will define the overall estimation process using this approach in greater detail, and describe tool enhancements to support it.

Acknowledgement

We would like to thank Isabel John, Gerald Reif and Matthias Heindl for supporting our search for suitable use case sets.

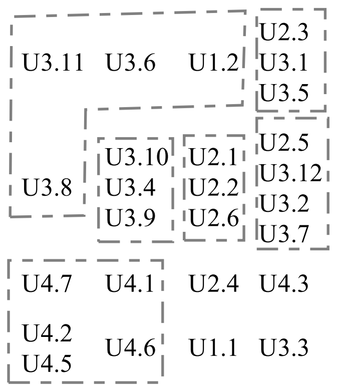

Figure 3. Map of use case set COLLAB

References

[1] M. Auer, B. Graser, and S. Biffl. A survey on the fitness of commercial software metric tools for service in heterogeneous environments: Common pitfalls. In *Proceedings of 9th IEEE International Software Metrics Symposium (METRICS 2003)*, September 2003.

[2] M. Auer, B. Graser, and S. Biffl. Visualizing software project analogies to support cost estimation. In *Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS 2004)*, April 2004.

[3] J. Bode. Decision support with neural networks in the management of research and development: Concepts and application to cost estimation. *Information and Management*, 34(1):33–40, 1998.

[4] B. W. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W. Brown, S. Chulani, and C. Abts. *Software Cost Estimation with Cocomo II*. Prentice Hall, 2000.

[5] C. Cherif Latiri and S. Ben Yahia. Generating implicit association rules from textual data. In *Proceedings of the 2001 ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'01)*, June 2001.

[6] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 2000.

[7] G. Finnie and G. Wittig. A comparison of software effort estimation techniques: using function points with neural networks, case based reasoning and regression models. *Journal of Systems Software*, 39:281–9, 1997.

[8] J. Hihn and H. Habib-Agahi. Cost estimation of software intensive projects: A survey of current practices. In *Proceedings of the 13th International Conference on Software Engineering (ICSE'91)*, pages 276–87, May 1991.

[9] R. Hughes. Expert judgement as an estimating method. *Information and Software Technology*, 38(2):67–75, 1996.

[10] R. Jeffery, M. Ruhe, and I. Wieczorek. A comparative study of two software cost modeling techniques using multi-organizational and company-specific data. *Information and Software Technology*, 42(14):1009–1016, 2000.

[11] C. Jones. *Estimating Software Costs*. McGraw-Hill, 1998.

[12] T. Kohonen. *Self-organizing maps*. Springer-Verlag, Berlin, 1995.

[13] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. Self-organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585, May 2000.

[14] K. D. Maxwell. Collecting data for comparability: Benchmarking software development productivity. *IEEE Software*, pages 22–25, September/October 2001.

[15] E. Nasr, L. McDermid, and G. Bernat. Eliciting and specifying requirements with use cases for embedded systems. In *Proceedings of the 7th International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'02)*, pages 350–357, January 2002.

[16] A. Rauber and D. Merkl. The SOMLib Digital Library System. In S. Abiteboul and A. Vercoustre, editors, *Proceedings of the 3rd European Conference on Research and Advanced Technology for Digital Libraries (ECDL99)*, number LNCS 1696 in Lecture Notes in Computer Science, pages 323–342, Paris, September 1999. Springer.

[17] A. Rauber and D. Merkl. Text mining in the somlib digital library system: The representation of topics and genres. *Applied Intelligence*, 18(3):271–293, 2003.

[18] A. Rauber, D. Merkl, and M. Dittenbach. The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, November 2002.

[19] M. Ruhe, R. Jeffery, and I. Wieczorek. Cost estimation for web applications. In *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*, pages 285–294, May 2003.

[20] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, MA, 1989.

[21] F. Sheldon, K. Jerath, and O. Pilskalns. Case study: B2B e-commerce system specification and implementation employing use-case diagrams, digital signatures and XML. In *Proceedings of the 4th International Symposium on Multimedia Software Engineering (MSE'02)*, pages 106–113, December 2002.

[22] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(12):736–43, November 1997.

[23] A. Zeboulon, Y. Bennani, and K. Benabdeslem. Hybrid connectionist approach for knowledge discovery from web navigation patterns. In *Proceedings of the 2003 ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'03)*, July 2003.