Author(s): Nurminen, Miika; Suominen, Panu; Äyrämö, Sami; Kärkkäinen, Tommi

Title: Applying Semiautomatic Generation of Conceptual Models to Decision Support Systems Domain

Year: 2009

Version:

Please cite the original version:

Nurminen, M., Suominen, P., Äyrämö, S., & Kärkkäinen, T. (2009). Applying Semiautomatic Generation of Conceptual Models to Decision Support Systems Domain. In R. Breu (Ed.), Proceedings of the IASTED International Conference on Software Engineering (SE 2009) (pp. 7). ACTA Press. http://www.actapress.com/Abstract.aspx?paperId=34625

APPLYING SEMIAUTOMATIC GENERATION OF CONCEPTUAL MODELS TO DECISION SUPPORT SYSTEMS DOMAIN

Miika Nurminen, Panu Suominen, Sami Äyrämö, Tommi Kärkkäinen
Department of Mathematical Information Technology
University of Jyväskylä, Finland
minurmin@mit.jyu.fi, panu.suominen@iki.fi, {samiayr,tka}@mit.jyu.fi

ABSTRACT

This paper presents a decision support system specification in the form of business use cases and a stereotyped conceptual model based on the specification. The use cases are based on generic user requirements and address cognitive biases. The specification can be used to set fixed and common terms among the project participants.

Semiautomatic generation of the conceptual model is demonstrated with mixed results. While there are some shortcomings in the parsing and the result is dependent on the phrasing conventions used in the use cases, the conceptual model highlights the most essential entities in the domain and provides a base for further development phases.

KEY WORDS

Use case, conceptual model, decision support system, systems analysis, domain engineering.

1 Introduction

The task of decision making can be divided into three steps [1]: (1) the identification and listing of all the alternatives (intelligence); (2) the determination of all the consequences resulting from each of the alternatives (design); and (3) the comparison of the accuracy and efficiency of each of these sets of consequences (choice). The division combines the organizational (descriptive, *what* decisions could and should be made) and technical (normative, *how* the decision should be made) views on decision making. OODA Loop [2] is another decision making model, meant for organizations that undergo continuous interaction with their environment. The OODA loop consists of four overlapping and interacting processes (observe, orient, decide, and act) that are in continuous operation during the interaction.

Use cases are a popular method for requirements elicitation. To get a complete picture of the requirements, they are considered with different stakeholders of the SuD (System-Under-Development). A stakeholder is someone with an interest in the future system, e.g. user, administrator, maintainer, etc. Use cases focus on describing the use of SuD as a part of workflows related to relevant stakeholders. Use cases provide an important context for the distinct functional requirements, how they are connected, the situations they are relevant in, and the related trigger conditions. The level of detail of a use case varies and can be adjusted on a per-project basis. Use cases do not describe non-functional requirements of SuD (e.g. performance, security, user interfaces). Also, use cases are not well suited for all software development, such as reactive systems [3].

This paper builds on a requirements specification of a generic (hypothetical) decision support system (DSS) [4] and elaborates the work reported in [5] and [6]. The requirements provide a checklist for the development of any system supporting operative decision making based on statistical decision theory (SDT). However, since organization-wide decision making is more than just an implementation of one decision support technique [1], a DSS should be based on layered architecture, separating task selection and actual decision making support. If possible, predictable deviations from rational decisions (cognitive biases) [7] should be compensated by the system.

We have specified the DSS with business use case - like descriptions [8] along with a conceptual model generated semiautomatically from the use cases. The specification can be used to set fixed and common terms among the DSS stakeholders. The specification can be used to analyze the possible structure and layers of the SuD related to DSS reference models. Further, the stereotypes and concepts discovered from use cases form a base for a domain-specific ontology that can be applied for information integration and reasoning about decision support systems.

The contents of this paper are as follows: Section 2 provides an introduction to related research. DSS specification is presented in Section 3. Section 4 evaluates the specification and concludes the paper.

2 Preliminaries

This section provides a short introduction to related research and desirable goals from system specification, information systems, and human decision making perspectives.

2.1 On Decision Support Systems

Decisions are a way of addressing a problem, containing a procedure or chain of reasoning as to how the problem

should be solved. If not, decision degenerates to merely guessing. However, the level to which the procedure can be automated varies. Categories of structured, semistructured, and unstructured problems can be defined [9]. For structured problems there exists a known procedure (e.g. standard processes, operations research) to find an adequate solution. Semistructured problems have some parts that are procedurally solvable and others that are not. Unstructured problems have no effective method for solving them. For example, planning for research is an unstructured problem while locating a warehouse is a structured one.

To help solve these problems decision support systems can be employed. The DSS is designed to support the user in making certain decisions. Usually this is achieved through modeling a subset of the real environment and analysing possible outcomes of decision candidates. Sometimes just simple calculations are enough. DSS covers a broad range of applications from simple spreadsheets to sophisticated AI systems – all having in common the goal to ease solving the problem they are designed to help with.

Turban [10] describes three essential subsystems of DSS: data management, model management, and user interface (UI). Hence, DSS is constructed from data, ways to manipulate them, and an interface for the user to interact with the system. Additionally there might be a knowledge management system that enables classification and heuristic evaluation of results, or automatic problem solving. The model is compared with DSS specification in Section 3.2.

2.2 Digitalization and Impact on Information Systems

The amount, degree, and form of communication used in organizations should be taken into account when designing decision support systems. With the current trend of digitalization and the convergence of networks, the amount of available information is higher than ever, generating new problems and adding to the impact of existing ones, such as information overload. The ease of information distribution can impair organizational communication with irrelevant information [11]. Thus, defining and gathering necessary information is a crucial step in realizing decision support.

Despite the increased digitalization of documents, all organizational information will never be available for automated processing. Based on analysis from three industrial and academic organizations [11], it seems that digital documents account for 40%-55% of total organizational communication, leaving out analog representations (e.g. paper) and other communication (e.g. face-to-face, phone). Since some of the analog documents are produced digitally (e.g. printing documents), the actual amount of digital communication might be higher, but still a notable part of communication takes place outside the information systems.

Even if both digital documents and other communication forms are considered, tacit knowledge can not be directly accessed by a DSS, even though it may have a pivotal role in decision making compared to official documentation. In principle, this can be alleviated by expressing tacit knowledge explicitly to become part of the organizational information resource, but in practice this can be difficult and time-consuming. Therefore, one should note that any information system can have direct access only to a fraction of the total knowledge present in an organization.

2.3 Cognitive Biases and Decision Making

Arnott [7] claims that although influences of DSS on decision performance are often disappointing, focusing on decision-making and tailored support can lead to successful systems. An analyst should have knowledge about human decision processes. The decisions can vary from the most rational choice. Predictable deviations from rationality are called cognitive biases. Arnott classifies 37 biases into 6 categories presented here briefly.

Memory biases (hindsight, imaginability, recall, search, similarity, testimony) are due to the fact that people remember familiar events more easily than others. Such a human judgment can yield an incorrect estimation of possibilities. Thus, DSS should provide accurate, impartial information from the past, avoiding overloading users' short term memory.

Confidence biases (completeness, control, confirmation, desire, overconfidence, redundancy, selectivity, success, test) arise from user's overconfidence in one's skills. People tend to choose the first complete-appearing solution disregarding alternatives, concentrating on confirming evidence. DSS should show alternatives and present the uncertainty of information. DSS should keep a record of the decisions made, enabling users to evaluate their actions.

People tend not to adjust enough to a change of environment. This ignorance of potentially significant new data is categorized under **adjustment biases** (anchoring and adjustment, conservatism, reference, regression). DSS using up-to-date models based on recent data facilitates adjustment to the task at hand by the decision maker.

The way information is represented matters. Scale differences between graphs, or overweighting of individual items can lead to wrong conclusions. Problems arising from **presentation biases** (framing, linear, mode, order, scale) can be avoided by using consistent user interfaces.

Situation biases (attenuation, complexity, escalation, habit, inconsistency, rule) include the human tendency to follow a previous course of action, only because it was used before. People also simplify the situation by discounting the level of uncertainty. Appropriate structuring of the decision tasks, as well as history databases are needed.

Statistical biases (base rate, chance, conjunction, correlation, disjunction, sample, subset) result from misinterpretation of random variable -based data.

Compensating for the erroneous behavior of the user should be considered when designing DSS, because these biases might alter the decision significantly. System can help the user overcome these shortcomings with carefully considered user interface, representation of statistical data, or just informing the user of common potential mistakes in

the current situation. If the problem can be structured, the program can follow the actions of the user.

3 DSS Specification

In this section, a use case specification and an entity model of a generic DSS is presented. The requirements in [4] were the starting point, but use cases were generalized to be independent of a computational method (e.g. SDT), organizational and technical context was clarified, and cognitive biases [7] were accounted for. The use cases establish a connection between descriptive and normative views on decision making. Consistent writing style and terminology was enforced by iterative writing and multiple reviews.

3.1 Roles

The following roles are applied in the use cases. Depending on the organization and the position of a decision maker, multiple roles could be performed by a single person.

System Expert is responsible for the DSS maintenance and configuration, data connections, software/ method extensions and updates. This role presumes extensive technical skills in information technology, software engineering, and to some extent, in knowledge discovery, data mining/analysis, and statistics. The tasks of the System Expert might be partly or fully outsourced.

Decision Configurator is responsible for the availability of necessary data sources. He/she analyzes the information sources/flows in the organization and responds to the data requests from the Method Expert and/or the Decision Maker. The role presumes extensive skills in information technology, data engineering, knowledge management, and possibly to some extent, software engineering.

Method Expert (Analyst) is responsible for applying the computational methods of DSS to the datasets. He/she has expertise in selection, usage and configuration of the methods, as well as deep understanding in optimization, data mining/analysis, statistics, and related methods. The Method Expert must be able to communicate about technical issues with the System Expert/Decision Configurator, and the meaning of the representations with the Decision Maker. The tasks of Method Expert might be outsourced.

Decision Maker is an experienced domain specialist who makes decisions and is usually responsible for the outcomes. The Decision Maker is not expected to have detailed technical understanding of decision support methods or models, but based on domain experience he/she can evaluate the impact of different decision alternatives provided by the Decision Configurator, or the DSS.

DSS Configuration Team combines the appropriate level of management and experts representing the aforementioned roles. The team maintains the DSS by analyzing the need for supporting new decision tasks, decision making principles, methods, models, and pre-configuring decision templates that guide predefined decision tasks.

3.2 Information systems

DSS contains or interfaces with the following libraries, databases, and other systems. The actual configuration and location of the systems depend on the organization. For example, a workflow system with executable process engine might provide information about decision tasks to DSS.

Method Library (Knowledge Management Subsystem [10]) contains implementations of the decision support techniques. New methods can be added by the System Expert. Utilization of the methods requires the tuning of parameters (distribution parameters for state estimation model, prototypes for clustering etc.) or retrieving them from the Decision History Database, and the testing (validity, sensitivity etc.) of parameters.

Decision History Database contains data about previous decisions and analysis steps that are supported by the system. This includes the relevant information about decision making cases (timestamp, problem description, consequences etc.), data sources, operational tasks and method selections, input parameters, and obtained models (alternatives). This enables repetition of the previous decision cases with new data and parameters. The results of the actions are gathered. Non-direct consequences are reported to the database by Decision Configurator or Decision Maker. Direct outcomes are collected automatically if possible.

Decision Template Database (Model Management Subsystem [10]) consists of predefined decision making tasks that can be used to guide the decision support process. Each template defines the method selections, appropriate parameter settings, perhaps pre-adjusted models, visual representations, and informative descriptions. The templates are defined by the DSS configuration team.

Organizational Data Sources (Data Management Subsystem[1] [10]) are information systems that are used in the day-to-day operation of the enterprise. These provide the input data for the DSS. The System Expert is responsible for providing connections to data sources.

3.3 Conceptual Stereotypes

Informal definitions for concepts (glossary) is needed to establish a joint language between stakeholders [12]. We base the glossary on use cases thus providing both documentation about a concept and its context of use. Attaching a stereotype to each concept creates a classification of them, supporting the transfer from domain analysis into system development. Stereotypes also clarify the structuring of use cases, because joint concepts related to system usage and its realization are tagged [8]. There is no need to prolong the use case by repeating the user action and system response in connection with the same concepts [13]: for a shared information transfer (*Actor creates X* $\rightarrow$ *System stores X*), the step should be described from user's perspective (*Actor creates X*, tag X as persistent data).

[1] If the data is gathered to a data warehouse to be used by DSS, this would be equivalent to Turban's Data Management Subsystem.

Use Case 1: Perform Organizational Configuration and Decision Making Processes

Id	Description	Concepts: Stereotype
1	DSS Configuration Team defines the set of organizational Decision Tasks to be supported by DSS and maintained by Decision Template Database.	DSS Configuration Team: Role Decision Task: UserElement DSS: System Decision Template Database: Database
2	DSS Configuration Team defines Necessary and Available Information for Decision Tasks.	Necessary Information: Data Available Information: ExternalData
3	DSS Configuration Team defines the set of available Decision Support Techniques to Method Library.	Decision Support Technique: DecisionModelElement Method Library: System
4	Method Expert documents the Decision Support Techniques to Method Library.	Method Expert: Role
5	DSS Configuration Team defines the content of Decision History Database. Meta-information and comments can be stored along with a reference to the applied decision support model.	Decision History Database: Database
6	Decision Configurator Team models the Decision Templates (2), which are supported.	Decision Template: DecisionModelElement
7	Decision Maker makes Decisions (3) supported by DSS.	Decision Maker: Role Decision: Action
8	DSS Configuration Team maintains DSS (4) based on Configuration Change Requests.	Configuration Change Request: Document

Use Case 2: Model the Decision Template

Id	Description	Concepts: Stereotype
1	DSS Configuration Team derives a generic Decision Task from the past decision support cases.	DSS Configuration Team: Role Decision Task: UserElement
2	Decision Configurator checks the availability of relevant internal/external task-specific data.	Decision Configurator: Role
3	Method Expert attaches the Decision Support Technique suitable for the Decision Task to the Decision Model and notifies about necessary but missing connections from DSS to Organizational Data Sources. Method Expert might decide to load an existing model to be the base of the model.	Method Expert: Role Decision Support Technique: DecisionModelElement Decision Model: UserElement Organizational Data Source: Database DSS: System
4	System Expert creates the necessary but missing connections to Organizational Data Sources.	System Expert: Role
5	Decision Configurator specifies Trigger Condition for recognizing the need for Decision Task.	Trigger Condition: Action
6	Method Expert defines the suggestive Decision Model Parameters for model building and inputs the parameters into the Method Library.	Decision Model Parameter: DecisionModelElement Method Library: System
7	Decision Configurator describes Decision Objectives and Decision Alternatives.	Decision Objective: DecisionModelElement Decision Alternative: DecisionModelElement
8	Decision Configurator attaches a structural Decision Making Process (i.e. phases or stages) yielding to a Decision Proposal for each Decision Task and stores it in the Decision Template Database.	Decision Making Process: Process Decision Proposal: UserElement Decision Template Database: Database
9	System Expert runs test cases and reports the results to the Method Expert.	
10	Decision Configurator documents the elements of the Decision Model and its relation to Decision Support Technique in Concept Documentation and stores the Decision Model, its Concept Documentation, its testing and version history in the Decision Template Database.	Concept Documentation: Document

Steps 5-8 can occur many times in any order.

Use Case 3: Make Decision

Id	Description	Concepts: Stereotype
1	DSS detects a Trigger Condition for a need for decision and shows Decision Proposal to Decision Maker.	DSS: System Trigger Condition: Action Decision Proposal: UserElement Decision Maker: Role
2	Decision Maker selects Decision Alternative to be inspected.	Decision Alternative: DecisionModelElement
3	DSS shows Decision Model information related to the Decision Alternative.	Decision Model: UserElement
4	Decision Maker inspects and alters the Decision Scenario related to Decision Alternative. Decision Maker can propose a Configuration Change Request.	Decision Scenario: UserElement Configuration Change Request: Document
5	DSS generates and shows new Decision Alternative.	
6	Decision Maker makes Decision, documents it, and stores the Session with its Decision Documentation to Decision History Database. Decision Maker may accept a Decision Alternative, decide not to make a Decision, or cancel the Decision Making Process.	Decision: Action Decision Making Process: Process Decision Session: Data Decision Documentation: Document Decision History Database: Database
7	DSS captures all relevant Consequences of the Decision made, if possible. Consequences can also be documented manually.	Consequence: Document

Steps 2-5 can occur in any order and many times.

Use Case 4: Maintain DSS

Id	Description	Concepts: Stereotype
1	DSS Configuration Team receives Configuration Change Request related to the set of supported Decision Tasks from Change Requester. DSS Configuration Team has regular meetings to assess DSS and change requests.	DSS Configuration Team: Role Change Requester: Role Configuration Change Request: Document Decision Task: UserElement
2	DSS Configuration Team accepts or rejects the Configuration Change Request based on stored Decisions in Decision History Database and available information on documented Consequences of Decisions made.	Decision History Database: Database Decision: Action Consequence:Document
3	Method Expert modifies the set of available Decision Support Techniques and stores the results to Method Library.	Method Expert: Role Decision Support Technique: DecisionModelElement Method Library: System
4	DSS Configuration Team modifies the set of organizational Decision Tasks.	
5	DSS Configuration Team documents the changes in Decision Tasks, stores them and Change Documentation to Decision Template Database, and notifies the Change Requester and other users.	Change Documentation: Document Decision Template Database: Database

Steps 3-5 are performed only if Configuration Change Request was accepted in Step 2.

The following table contains definitions of the stereotypes. DecisionModelElement is specific to DSS domain; other stereotypes are domain-independent.

Stereotype	Description
Action	Functionality needed by SuD
Data	Persistent information used internally by SuD
Database	Database to be managed by SuD
Document	Document to be produced by SuD or a report that SuD must generate to a user
ExternalAction	An external action that SuD must take into account
ExternalData	Relevant data stored by other systems available for SuD
ExternalRole	Human or device that SuD must communicate with
Metadata	Data about data
Process	An ordering of work activities across time and place with a beginning and an end with inputs and outputs [14]
Role	Stakeholder representatives who share the same roles and responsibilities with respect to the project [12]
Selection	A particular choice related to a particular UserElement
System	SuD or other information system related to use case
UserElement	An element representing the interaction interface between a user role and SuD
DecisionModelElement	General entity related to the decision making model

3.4 Use cases

Figure 1 represents the basic structure of the DSS described in detail in tables UC1-UC4. Numbers in parentheses denote links to another use case. UC1 presents the general process of utilizing a DSS in an organization. The process starts with the need for the decision, triggered automatically by the system or manually by the user. New Decision Tasks are modeled in a decision template (UC2). After the model exists the DSS can generate a decision proposal that Decision Makers can use to make decisions (UC3). The Decision Maker is able to change the data, and possibly the parameters in the model to evaluate the alternatives. Finally, the Decision Maker can propose configuration change requests that can be implemented by the Configuration Team in the process of maintaining DSS (UC4).

As an example of applying the use cases, we demonstrate prototype-based (e.g. k-means, k-spatmed) data clustering (i.e. grouping of similar elements in a dataset) [15] as a decision support technique. The specific problem addressed here is controlling manufacturing process in the product line in process industries (see [6]). Different product line states are represented as clusters, decisions are reflected as probabilistic transitions between the clusters.

In use case 2, estimating Parameters (step 6) means the estimation of clusters and prototypes comprising the decision model with chosen data. Uncertainty of the clusters (state estimates) and consequent actions can be evaluated using SDT-methods. Decision objectives/alternatives (step 7) means the attachment of different control parameters to the current and desired state and consequent state alternatives and their probabilities. Decision making process (step 8) can be sequential or parallel, relying on a single decision maker or a group of experts. Proposed decision alternatives are attached to each of the cluster prototypes.

In use case 3, Decision Alternatives (step 2) are based on state change history of the cluster model that documents the influence of the different actions (process control adjustments) with respect to states (clusters). Decision Model Information (step 3) can be a visualization of process data and cluster evolution. For example, taking action A when the process is in cluster (state) 1 leads to the state change from cluster 1 to cluster 2 with 95% probability and to cluster 3 with 5% probability. Decision Maker can also explore the previous decisions and the resulting consequences. When altering Decision scenario (step 4), the Decision Maker could request a change on the number of cluster prototypes, clustering principle (e.g., different distributional assumptions), or other specific parameters.

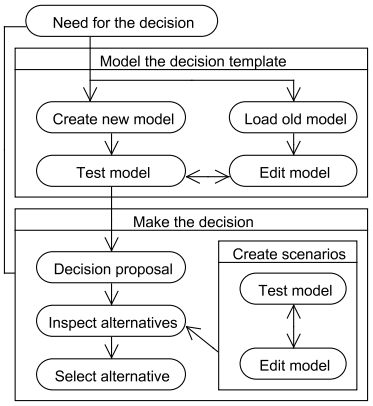

Figure 1. Basic workflow for a decision support system

3.5 Entity Model

Use cases describe the domain in one viewpoint. The information is mostly not properly organized for software development. The development process can be facilitated by extracting a domain model from the use cases. We encoded the use cases in ProcML – an XML format that allows attaching metadata to use case steps [16]. Use cases were analyzed by UCOT (Use Cases to Original enTities) software [5], a prototype of a system which is designed to automatically analyze use cases and create a conceptual model based on the analysis. A grammatical parser[2] and Abbott's heuristic [17] were used in the process.

An unmodified, automatically generated entity model is not useful as such because of the limitations in heuristic and parsing. After initial processing the conceptual model was refined manually by merging and splitting duplicate and ambiguous entities. Attribute and relation information was also adjusted to reflect the application domain. Nonessential entities were omitted for easier understandability. Finally, stereotypes were added to some of the concepts. The final model is illustrated in Figure 2 and shows approximately how entities of the system act together.

[2] Available at http://nlp.stanford.edu/downloads/lex-parser.shtml

Although the use cases were mostly written using strict conventions (e.g. using subject-predicate-object structure), it proved to be exceedingly difficult to stick with simple structures. The use cases were iterated by four different authors and as the domain understanding increased, the complexity of the sentences increased as well. A specific problem was the complex relationship between Decision Support Technique, Decision Model, and Decision Task. They are referred to in many steps and often in an ambiguous way (e.g. "Method Expert attaches the Decision Support Technique suitable for the Decision Task to Decision Model") that is difficult to interpret automatically.

A limitation in UCOT data model is the lack of support for n-ary relations. Since the use cases contained some 3-ary relations (e.g. "Decision Configurator stores Documentation to Decision Template Database"), we had to divide them to elementary relations ("Decision Configurator stores Documentation" and "Documentation is stored to Decision Template Database"). Singular vs. plural forms, and the use of pronouns ("Decision Maker makes Decision and documents it") yielded unnecessary entities that had to be merged. Overall, the system was not effective in processing long sentences and produced entities that contained either multiple concepts (e.g. "Decision Objectives and Decision Alternatives") or both a concept and a relation (e.g. "DSS based on configuration change requests").

Even though it is relatively straightforward to "clean up" the model with UCOT after initial processing, maintenance becomes an issue if the use cases are modified after editing the entity model. Since the relations from the entity model are not explicitly linked back to the use cases, the entity model may have to be recreated from scratch after modifications. In future, some of the changes could be applied to the entity model automatically by recording user actions. ProcML data model could also be extended with full entity linkage: as the use cases are processed, UCOT would tag each word with related entities in XML.

Based on the entity model, it seems that the roles "DSS Configuration Team", "Decision Configurator", and "Method Expert", and databases "Decision Template Database" and "Decision History Database" are highly connected. Other key entities include "DSS", "Documentation", "Decision Task", "Decision Support Technique", and "Decision Model". Many entities starting with word "Decision" are most likely more connected than they need to be. Currently, the model is not as understandable as we had hoped prior to use case specification. However, some hints about the architecture can be discovered. For example, the activities of "Method Expert" and "System Expert" related to "Decision Support Technique" and "Organizational Data Sources" are somewhat isolated from the rest of the system, so they are candidate entities for separate components. On the other hand, because of the high connectivity of "Documentation", it might make sense to construct a common documentation system used by different subsystems.

The model presented is by no means complete or final – it merely provides an alternate view to use cases that should be updated when needed. Being generated from informal descriptions, the model does not necessarily represent the exact entities and relations (cf. ER-diagram in database design) in the system, but helps to find the most essential (e.g. densely connected) entities that should be concentrated on. Still, the model is a base for further development phases and can be utilized in various ways, depending on the development methodology. One could proceed with object-oriented analysis and design, separating classes and objects from the entity model and extending it with more detail. The entity model could also be generalized to a domain (meta)model or ontology to represent requirements that are common for a set of applications.

4 Conclusion

There are a multitude of approaches to support decision making in organizations. We have tried to cover some relevant aspects of decision support systems by suggesting a new generic use case -based specification for a DSS. Use cases were based on generic user requirements [4] and generalized to accommodate different decision support techniques and cognitive biases. Although somewhat abstract in nature, the use cases clarify especially the context (e.g. roles and information systems) needed to establish a DSS. The semiautomatically generated entity model points out essential concepts from the problem domain and can be used as a base for more detailed specifications.

Although the use cases were based on generic user requirements, the explicit link between requirements and use case steps was not preserved. While ProcML supports linking requirements to use cases, as the meaning of particular steps was changed or as use cases were split or joined, tracing the original requirements to use cases was somewhat cumbersome without further software support. A more serious shortcoming is the lack of linkage between the entity model and use cases – the transformation is one-way and manual corrections must be made to the entity model. In future development, the generated conceptual model should be synchronized with manually specified entities and stereotypes marked in use cases.

Combining use cases to a stereotyped entity model seems to be a promising approach for requirements elicitation and conceptual modeling regardless of the methodology (e.g. OOA/D, domain engineering) used in later development phases. Stereotypes provide metadata that can be used for code generation and to simplify the original use case descriptions. Some of the stereotypes (e.g. Role, System, Process) are domain-independent, but a general method to derive domain-specific stereotypes (e.g. DecisionModelElement) is yet to be explicated. Ultimately there could be transparent 2-way linking between requirements, use cases, and entities in a unified model residing in a knowledge base. Depending on the modeling task, different views of the model could be exported to achieve productivity gains in systems development.

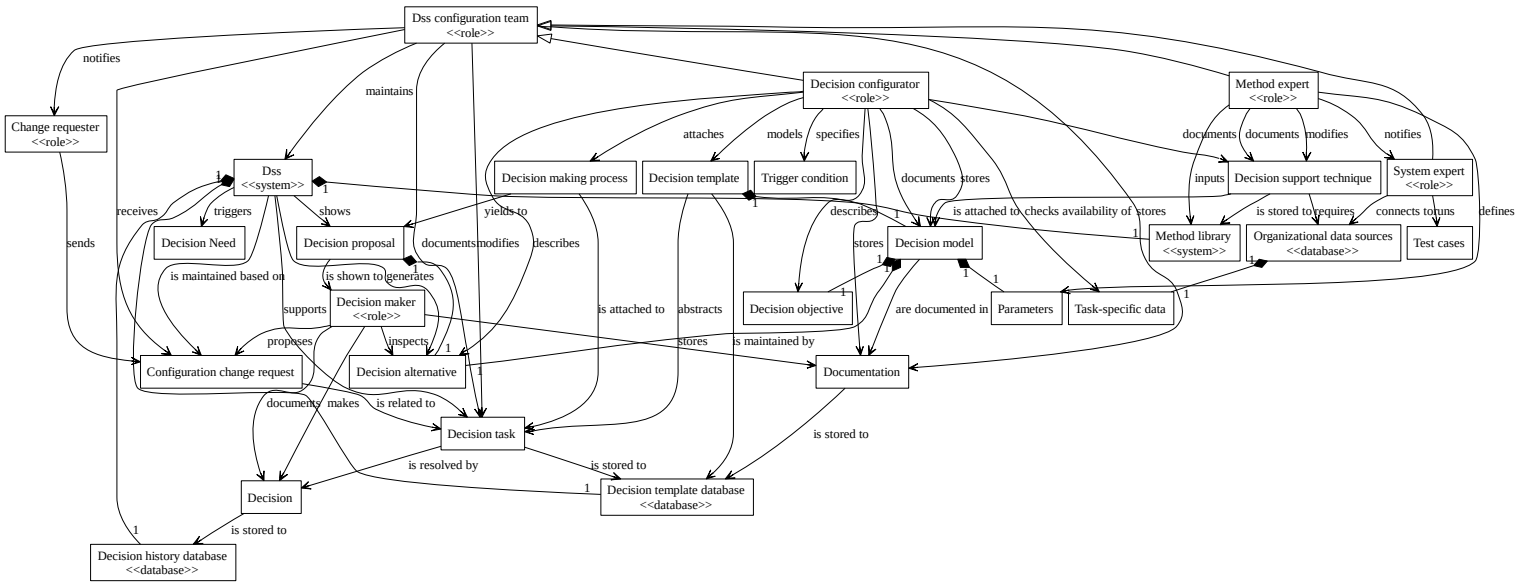

Figure 2. Modified DSS entity model based on use cases

Acknowledgements

This work was supported by National Technology Agency of Finland under project ProductionPro (Dnro 2832/31/06).

References

[1] H. A. Simon. *Administrative Behavior (3rd edition)*. The Free Press, 1976.

[2] J. Boyd. *A Discourse on Winning and Losing*. Defense and the National Interest.

[3] M. Jackson. *Problem frames: analyzing and structuring software development problems*. Addison-Wesley, 2001.

[4] H. Jokinen, J. Grén, K. Hukki, K. Konkarikoski, T. Kärkkäinen, U. Pulkkinen, R. Ritala, P. Suominen, and J.-P. Ylén. Generic user requirements for operational decision support system. *To be submitted*.

[5] T. Kärkkäinen, M. Nurminen, P. Suominen, T. Pieniluoma, and I. Liukko. UCOT: Semiautomatic generation of conceptual models from use case descriptions. In *IASTED Int. Conf. on Software Engineering (SE 2008)*, 171–177. ACTA Press, 2008.

[6] M. Nurminen, P. Suominen, S. Äyrämö, and T. Kärkkäinen. Use cases for operational decision support system. In T. Mätäsniemi, editor, *Operational decision making in process industry*. VTT, 2008.

[7] D. Arnott. Cognitive biases and decision support systems development: a design science approach. *Information Systems Journal*, 16(1):55–78, January 2006.

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

[9] G. Gorry and M. Scott-Morton. A framework for management information systems. *Sloan Management Review*, 13(1):55–71, 1971.

[10] E. Turban, J. E. Aronson, and T.-P. Liang. *Decision Support Systems and Intelligent Systems (7th Edition)*. Prentice-Hall, 2004.

[11] T. Kilpeläinen. *Genre and ontology based business information architecture framework (GOBIAF)*. PhD thesis, University of Jyväskylä, 2007.

[12] K. Bittner. *Use Case Modeling*. Addison-Wesley, 2002.

[13] R. Wirfs-Brock. Designing scenarios: Making the case for a use case framework. *The Smalltalk Report*, 3(3), Nov-Dec 1993.

[14] T. H. Davenport. *Process Innovation – Reengineering Work through Information Technology*. Harvard Business School Press, 1993.

[15] S. Äyrämö. *Knowledge Mining Using Robust Clustering*. PhD thesis, University of Jyväskylä, 2006.

[16] M. Nurminen, A. Honkaranta, and T. Kärkkäinen. ProcMiner: Advancing process analysis and management. In *IEEE 23rd Int. Conf. on Data Engineering Workshop (TDMM)*, 760–769, 2007.

[17] R. J. Abbott. Program design by informal English descriptions. *Commun. ACM*, 26(11):882–894, 1983.