The IASTED International Conference on Software Engineering (SE 2009)
February 17, 2009, Innsbruck, Austria

# Applying Semiautomatic Generation of Conceptual Models to Decision Support Systems Domain

Miika Nurminen, Panu Suominen,
Sami Äyrämö, Tommi Kärkkäinen

University of Jyväskylä

Department of Mathematical Information Technology (MIT)

Research Group on Computational Sciences, Software Engineering and Education (COSSE)

Faculty of Information Technology

TUOTANTO 2010

VTT

Tekes

# Outline

1. Basic Concepts & Related Work
2. DSS Specification (Use Cases)
3. Generating a Conceptual Model for DSSs (UCOT)
4. Conclusion

# Basic Concepts

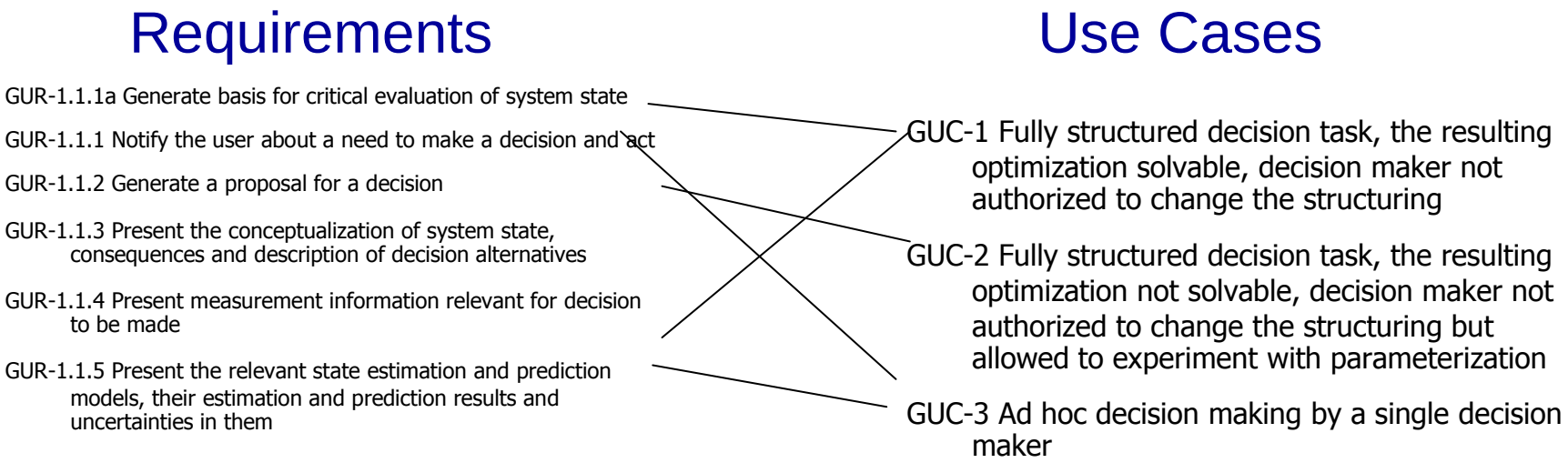
- In requirements analysis domain understanding and shared ontology between stakeholders is needed
  - A domain/analysis model understood and accepted to abstract the shared view is required
  - Use cases provide a process-like view of the requirements with both contextual and structural information for problem solving
- Object orientation in analysis may require unnecessary qualifications from relevant stakeholders (deciders)
- NLP (and other CS ”stuff”, e.g. text mining) can and should be utilized in tools to support automatic analysis
  - UCOT: Prototype/proof-of-concept for semiautomatic discovery of domain concept model from use cases (details: Kärkkäinen *et al*. 2008)
- In the paper, Decision Support Systems are used as an example domain
  - We present a decision support system specification in the form of business use cases and a stereotyped conceptual model based on the specification
  - The conceptual model is generated semiautomatically using UCOT

# Related Work

- OOA/OOD
  - (Abbott*, 1983)* Abbot's heuristic
  - (Cockburn*, 2000)* use case writing conventions & patterns
  - (Liu *et al, 2004)* *UCDA,* class model generation from use cases
  - (Pérez-González *et al, 2005)* *GOOAL, OOA laboratory*
  - ProcMiner (Nurminen *et al*, 2007) use case management
  - UCOT (Kärkkäinen *et al*, 2008) semiautomatic conceptual model generation
- Decision Support Systems
  - (Turban *et al*, 2004) reference model for decision support systems
  - (Arnott, 2006) cognitive biases and decision support systems
  - (Jokinen *et al*, to appear) Generic User Requirements for decision support systems

# From Generic Requirements to Use Cases

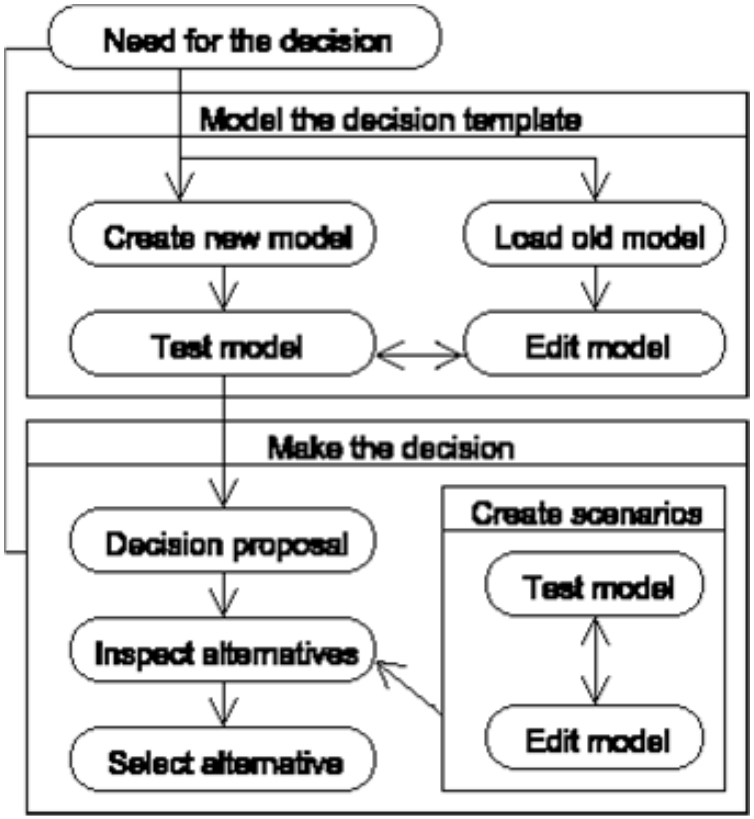
*Source: Operational decision making in process industry - Multidisciplinary approach. VTT Research Notes 2442.*




- Generic User *Requirements* for DSSs (Jokinen *et al.*) were used as a starting point for a new, generic decision support systems specification
- Use cases were rewritten iteratively and generalized to be independent from a particular computational method. Arnott's cognitive biases for decision making were accounted for in the use cases
    - Initial system architecture was designed related to Turban's DSS reference model
    - User roles and information systems noted in use cases were clarified and explicated
    - Main concepts from the use cases were manually classified to stereotypes

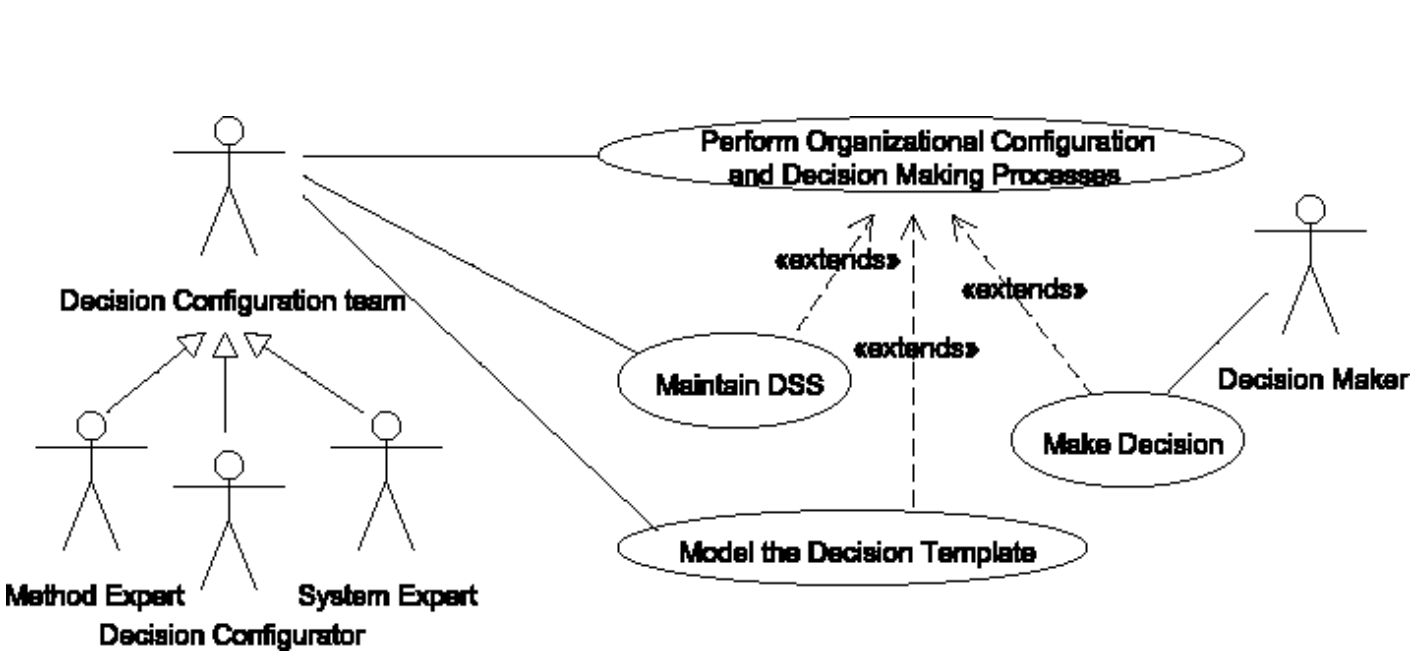
# Use Cases Overview

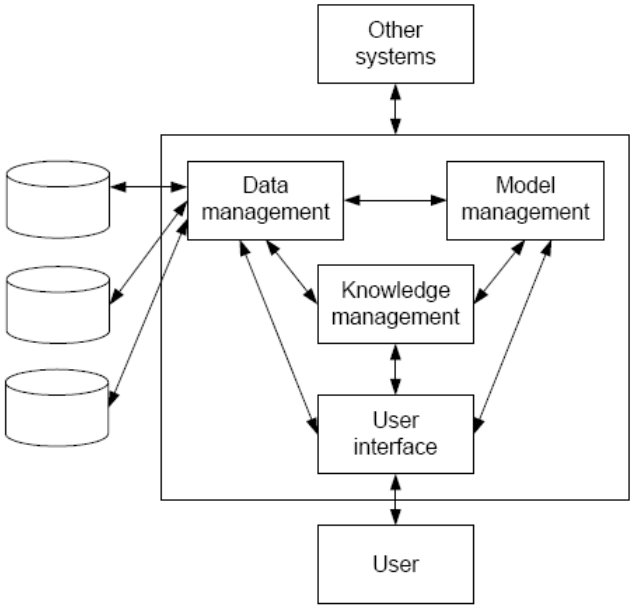
- Generic requirements and revised use cases were encoded in ProcML and transferred to ProcMiner process management system for further processing
- Four use cases were identified:
  1. Perform Organizational Configuration and Decision Making Processes
  2. Model the Decision Template
  3. Make Decision
  4. Maintain DSS
- Use Cases 2-4 are modeled as extensions to use case 1




```
system" postcondition="Decision done and implemented, deviation from
proposed decision and its motivation documented. NOTE! Doing nothing
is a decision." type="main" state="normal" primary="false">
Fully structured decision task, the resulting optimization solvable,
decision maker not authorized to change the structuring
<desc>At any stage the user may retrieve stored sessions about the
same decision task in the order of similarity (judged firstly by same
proposed decision, and secondly by similarity of data). Other actors:
shift foreman or process engineer.</desc>
- <abstraction level="0">
 - <seq>
  - <step id="s1">
      Actor receives a decision proposal and a link to
      documentation about the concepts and continues.
      <document ref="gur1.1.2" />
      <document ref="gur1.1.3" />
    </step>
  - <step id="s2">
      Actor views the data used in generating the decision and
      accepts it and continues.
      <document ref="gur1.1.4" />
    </step>
  - <step id="s3">
      Actor views the robustness of the decision and continues.
      <document ref="gur1.1.8" />
      <document ref="gur1.1.9" />
    </step>
  - <step id="s4">
      Actor asks for the state estimated and the consequences of
      proposed decisions and continues
      <document ref="gur1.1.5" />
    </step>
  - <step id="s5">
      Actor asks for the structured objective(s) and their level of
      satisfaction potential trade-offs.
      <document ref="gur1.1.6" />
      <document ref="gur1.1.7a" />
      <document ref="gur1.1.7b" />
    </step>
  </seq>
</abstraction>
```

# DSS Model




Components of a DSS (Turban 1998 p. 79).

- Actors:
  - System Expert
  - Decision Configurator
  - Method Expert
  - Decision Maker
  - DSS Configuration Team
- Information systems (vs. Turban's reference model):
  - Method Library (Knowledge Management)
  - Decision History Database
  - Decision Template Database (Model Management)
  - Organizational Data Sources (Data Management)

# Conceptual Stereotypes in Use Cases

- Stereotypes provide 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.
- There is no need to prolong the use case by repeating the user action and system response in connection with the same concepts:
  - For a shared information transfer (*Actor creates X, System stores X*), the step should be described from user's perspective (*Actor creates X,* tag X as persistent data).
- The table contains definitions of the stereotypes in DSS Specification.
  - 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 |
| DecisionModel-Element | General entity related to the decision making model |

# Detailed Example

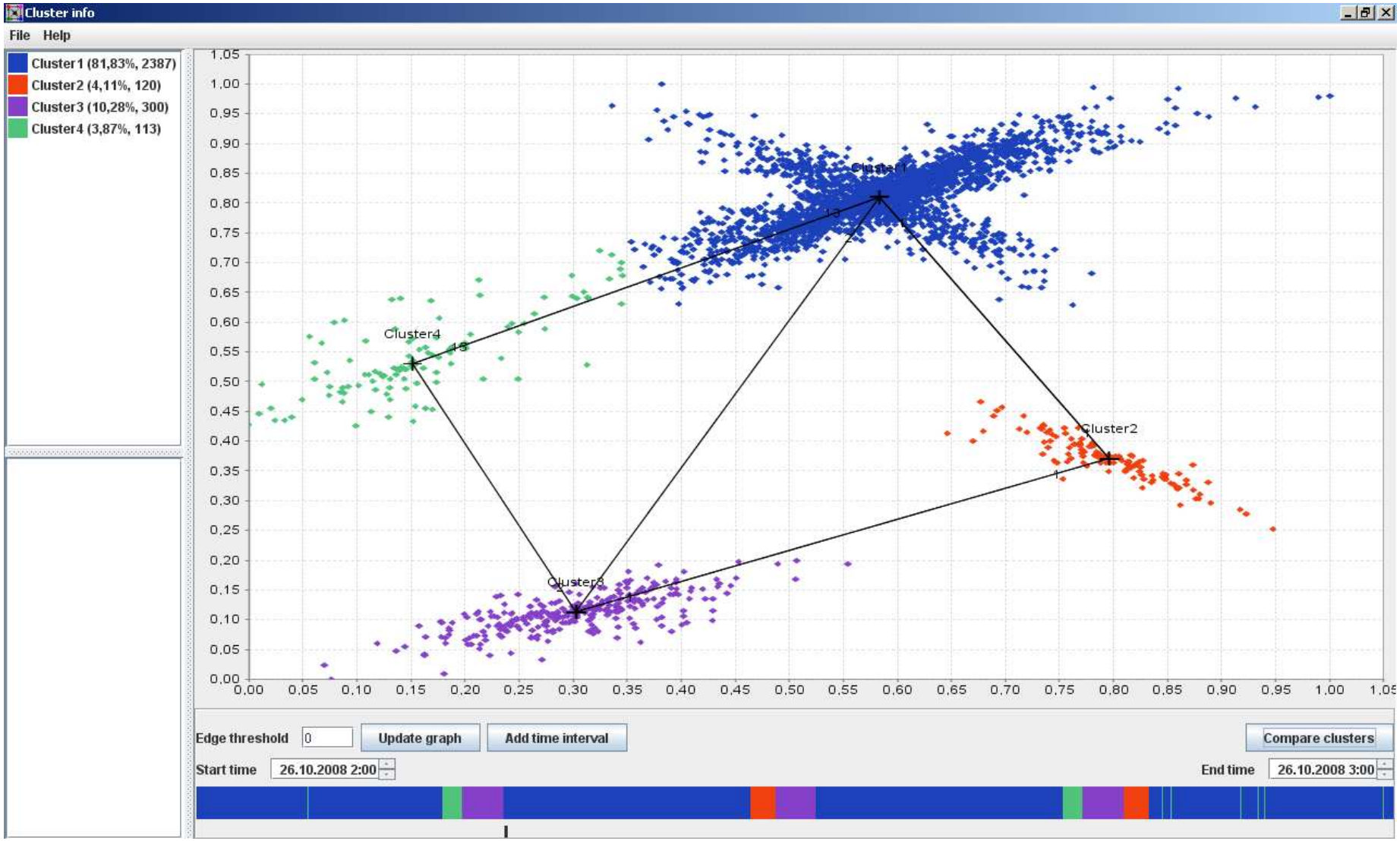
**Use Case 2: Model the Decision Template**

| 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 |

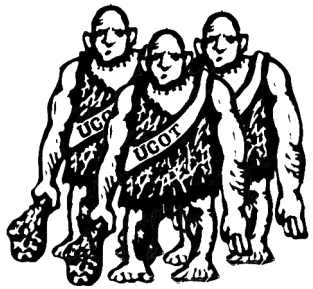
Steps 5-8 can occur many times in any order.

- As an example of applying the use cases to a specific computational method, prototype-based (e.g. k-spatmed) data clustering is demonstrated as a decision support technique
- The specific problem addressed is controlling industrial manufacturing process
- Different product line states are represented as clusters, decisions are reflected as probabilistic transitions between the clusters

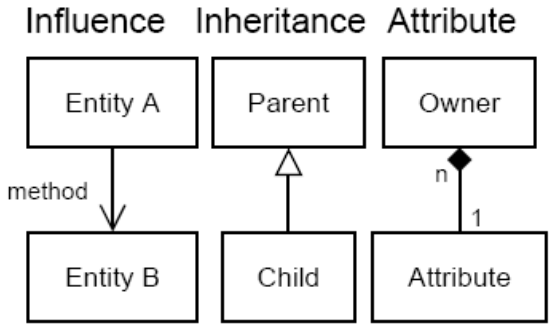
# Data clustering as a decision support technique

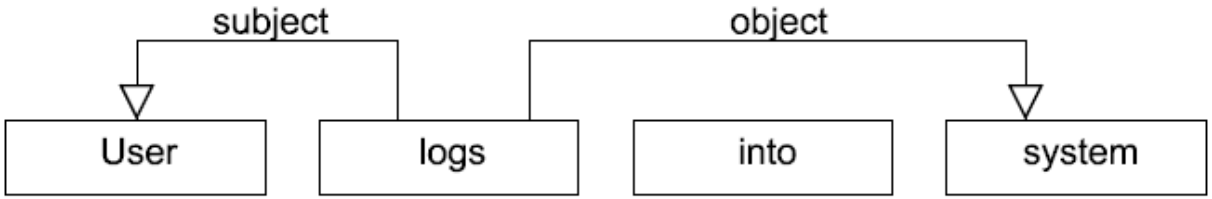


# UCOT - NLP for Model Generation

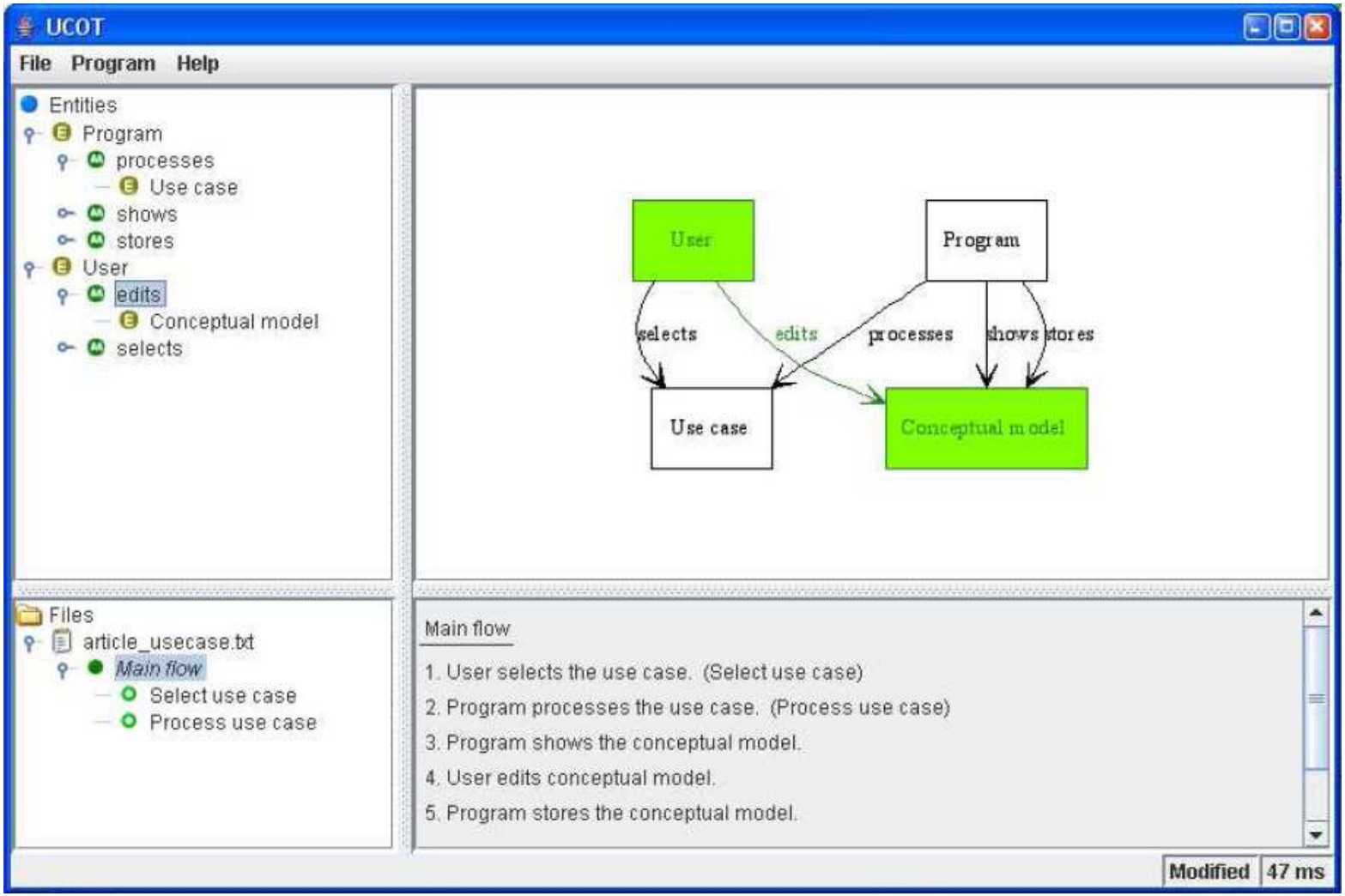



- UCOT (from Use Cases to Original enTities) is a reseach prototype which is designed to automatically analyze use cases and create a conceptual model based on the analysis. (details: Kärkkäinen *et al*. 2008)
- Stanford grammatical parser (extracting both parts of speech and sentence elements) and Abbott's heuristic are used to process the use cases.
- User can modify the conceptual model by combining entities, refining entities and relations, as well as adding roles for the entities.
  - An entity may also have a role (i.e. a stereotype) that can be used to group entities to application domains, architectural components, or predefined, recurring object types (database, document, role, process etc) to ease the transformation from domain analysis to system design.
- Only the simple rules related to Abbot's heuristic (nouns to entities, and verbs to relations between entities) were implemented to preserve the input language independence



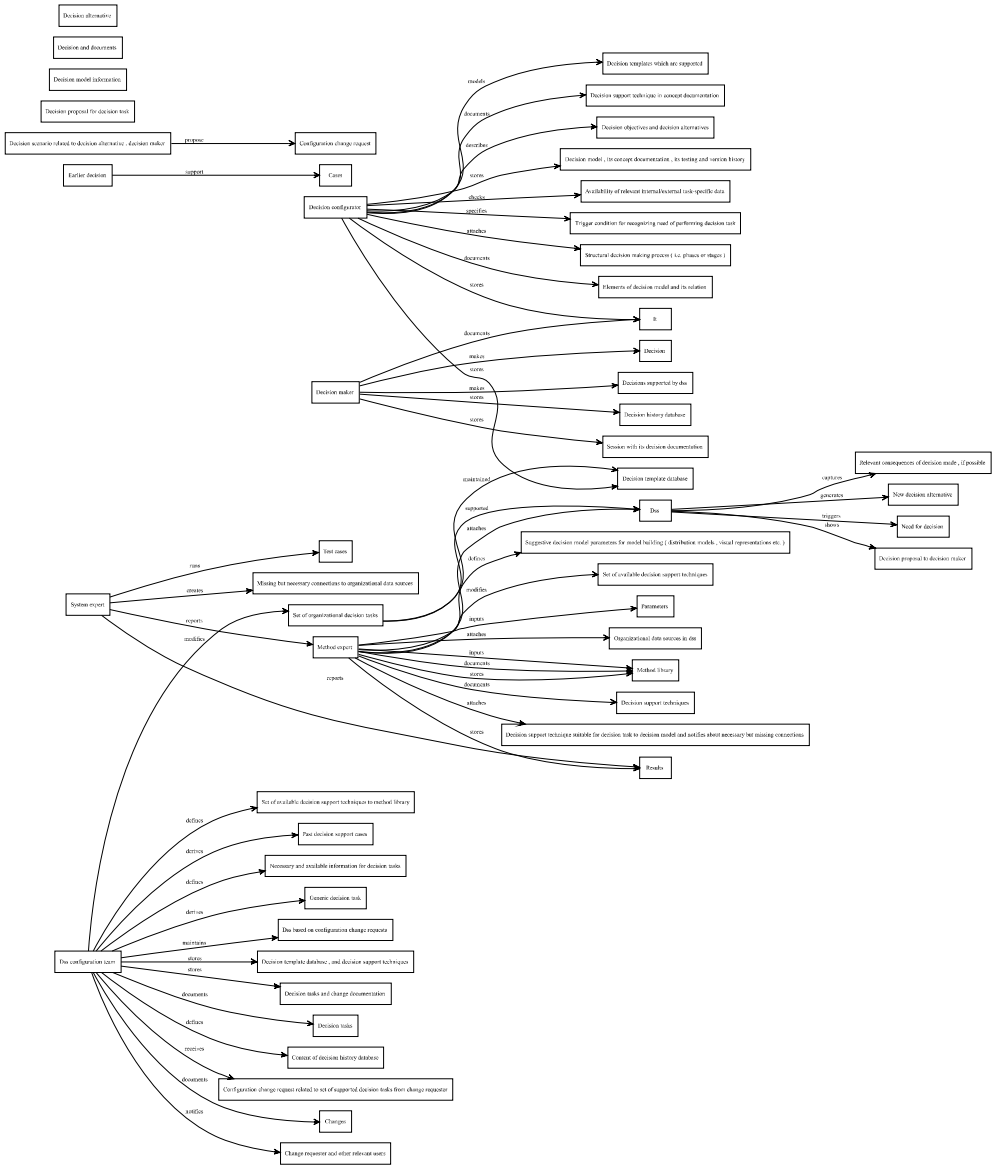



# UCOT User Interface

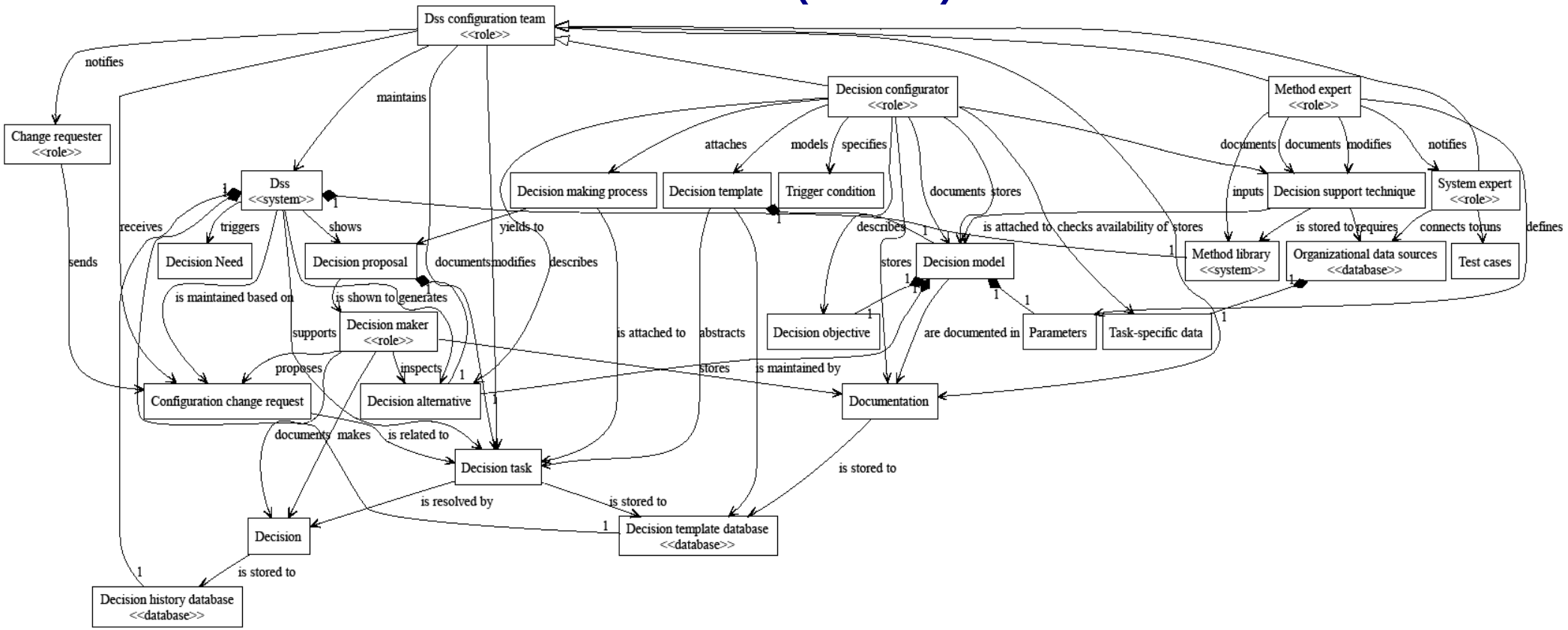


# Results (automatically generated)

- An unmodified, automatically generated entity model is not useful as such because of the limitations in heuristic and parsing
- The relations are greatly dependent on phrasing conventions in use cases
- E.g. sentences should start with a subject: *"actor asks"*, *"actor views"*, but some of the the sentences lack the subject, complicating the parsing: *"Apply the guided tour"*, *"Ask the user"*…
- Sometimes the entities are not separated correctly: e.g. *"Dss based on configuration change requests"* interpreted as one entity.
- Manual modifications clarify the model, but it is crucial to use consistent writing conventions up front.



# Results (fixed)




- ”Word order” was modified manually to simplify relations (”Decision Maker stores Documentation to Decision History Database” -> ”Decision Maker stores Documentation”, ”Documentation is stored to Decision History Database”)
- Synonymous concepts were unified and entities consisting of entire clauses were splitted
- The modified model is not ”final” (e.g. elaborated entity standardization, additional stereotypes and relations), but as such helps to see central concepts of the application domain and considerations for system architecture

# Evaluation

- Conceptual Model highlights the essential concepts in the application domain (e.g. entities with high connectivity)
  - Roles: *DSS Configuration Team, Decision Configurator. Method Expert*
  - Inf. systems: *Decision Template Database, Decision History Database*
  - *DSS, Documentation, Decision Task, Decision Support Technique, DecisionModel*
- Conceptual Model provides base for further development phases without commiting to a specific method (OOA/D, DSL/Domain engineering etc)
- Because of the limitations in the parsing, manual corrections must be made to the model, especially to relations
- Maintenance becomes an issue if the use cases are modified after editing the entity model – use cases and conceptual models are not synchronized automatically
- Diagram representations do not scale to large models, partial views (e.g. multifaceted search functionality) should be added
- 2-way linking between use cases, requirements, and entities is needed
  - Modifications to one model should be automatically reflected in other models

# Conclusion & Further Research

- The use cases presented in the paper provide a generic model and common terminology for decision support systems specification indepently of the computational method (e.g. statistical decision theory, data clustering) used
- Stereotypes and semiautomatically generated entity model clarify requirements analysis and domain understanding
- The quality of the original requirements and use cases (writing conventions, consistency) affect substantially to usefulness of the generated model
- Overall, with realistic-size models automatic conceptual model generation proved not to be as useful as originally hoped
  - UCOT user interface does not scale well to large models
  - Compare the effort needed to fix automatically generated model vs. creating the model manually
- Effective usage of the model needs better software support (linkage, traceability, maintenance) – automatically generated or not
  - Current requirements management software packages (e.g. Borland Requisite pro) provide some of the needed functionality, but in general, do not link the requirements artefacts to conceptual models
  - UML tools have a way to express conceptual models (Class Diagram), but the model cannot be naturally used with requirements-level modeling elements – linking classes to elements in Use Case diagram is not enough!