The IASTED International Conference on Software Engineering (SE 2008)
February 13, 2008, Innsbruck, Austria

UCOT: Semiautomatic Generation of Conceptual Models from Use Case Descriptions

Tommi Kärkkäinen, Miika Nurminen

Panu Suominen, Tuomo Pieniluoma, Ilari Liukko

University of Jyväskylä

Department of Mathematical Information Technology (MIT)

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

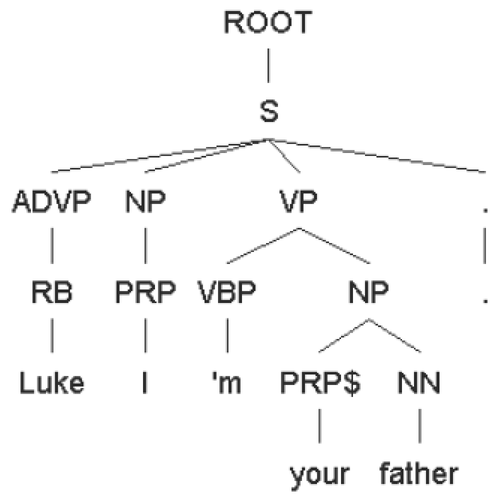
Faculty of Information Technology

Starting points

- 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)
  - (Extensive use of) UML might bring focus on layout instead of actual content (validation)
- 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

NLP for Model Generation

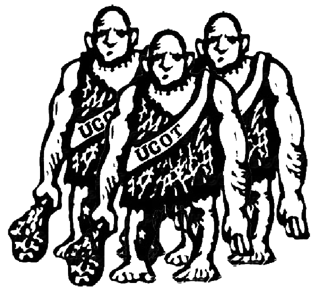
- There exists many approaches for natural language processing
  - Currently statistical parsers are widely used (and also utilized in this work)
- The classical approach for identifying classes from natural language phrases is the noun analysis, introduced by Abbott. Objects correspond to nouns and methods to verbs.
- Abbot’s heuristic can be utilized as a simple form of automated conceptual modeling
- Creating a conceptual model can help locating recurring patterns of domain entities, helping to find common attributes among different systems or within a single system. Such knowledge can be augmented to create a domain ontology.



Related work

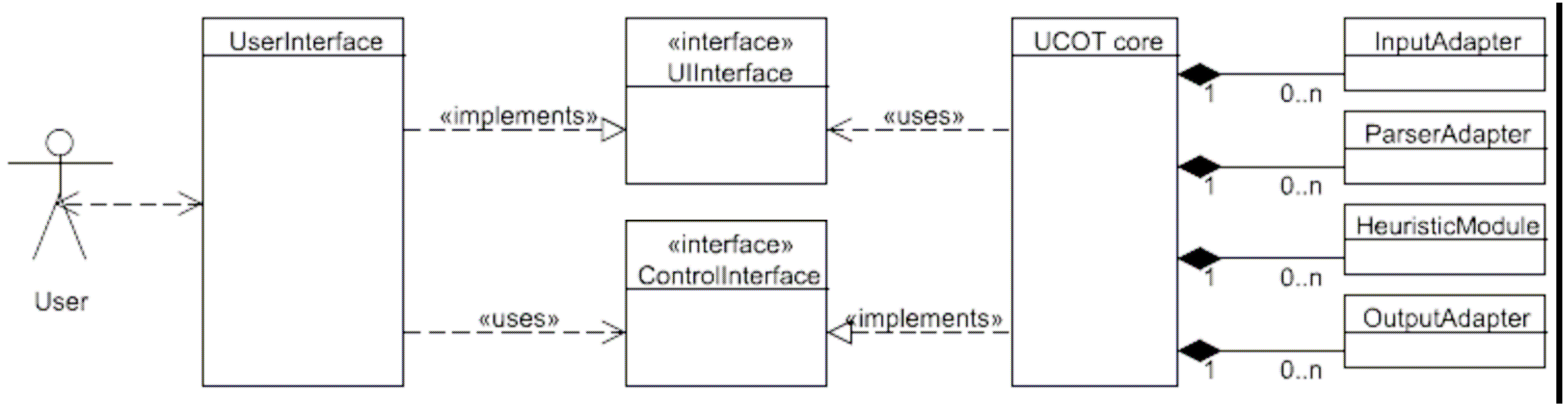
- (Abbott, *1983)* Abbot's heuristic
- (Cockburn, *2000)* use case writing conventions & patterns
- (Klein & Manning, *2002)* Stanford parser (PCFG+dependency model)
- (Anda & Sjøberg, 2005), domain class derivation technique
- (Li, 2000), use case normalization
- (Song *et al*, 2004) taxonomic class modeling methodology
- (Svetinovic, *2006)* OOA critique for conceptual modeling
- (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



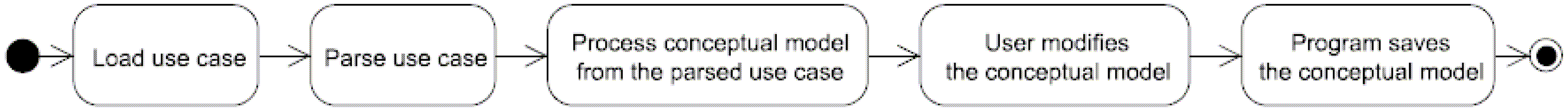
- UCOT (from Use Cases to Original enTities) is a reseach prototype which is designed to automatically analyze use cases and create a conceptual model based on the analysis.
- Use cases are represented as structured text or XML (ProcML) -based descriptions, containing a title and sequences of steps. Additionally use case steps can refer to other use cases.
- 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.
- The system is able to produce different kinds of output formats.
  - The ("pseudo-UML") model view is generated using visualization tool Graphviz
  - GXL (Graph eXchange Language) output
- Realized in real-customer fixed-time capstone project using agile practices
- Implemented with Java

UCOT architecture



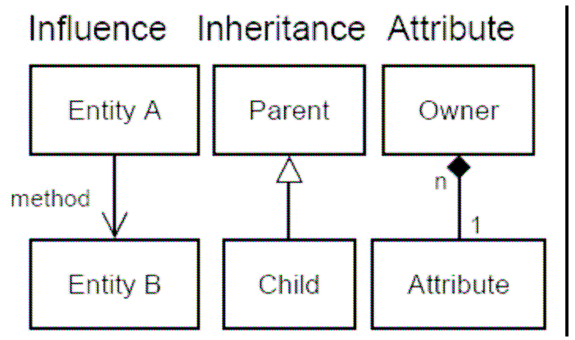
- UCOT system is structured according to the pipes and filters
- The system was designed to make it possible to add or replace components later.
- Key component of the system is Core, whose responsibility is to control other modules, loading them on startup, and direct the data flow.

UCOT architecture and processing flow

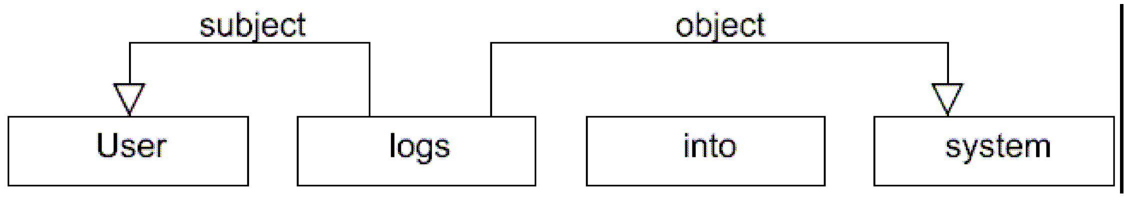

Load use case
Parse use case
Process conceptual model from the parsed use case
User modifies the conceptual model
Program saves the conceptual model

- UCOT components in processing flow
  1. Load use case (InputAdapter)
  2. Parse in an internal data structure (ParserAdapter)
  3. Apply heuristic rules (HeuristicModule)
  4. Modify conceptual model (UI)
  5. Save or export (OutputAdapter)
- Parser and heuristic (as well as other components) are independent of each other – heuristic uses only parts of speech and sentence element data
  - Increases modularity, but introduces some limitations to heuristic processing (especially potential metadata in original input format is not preserved “through” parser)

UCOT Data model



- The phrase model contains words and their relations in the sentence, independently from the input language and implementation of the parser (currently Stanford)
- Elements of the conceptual metamodel
  - The inheritance relation can be interpreted either as oo-like inheritance or instance-of relationship. The influence relation is used for other types of relationships.
  - Attribute relation is used for aggregation/composition
  - An entity may also have a role (i.e. a stereotype) which can be written under the name of the entity.
- 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



“Bootstrap” example

- Analyzing the use cases that were used to specify the system itself

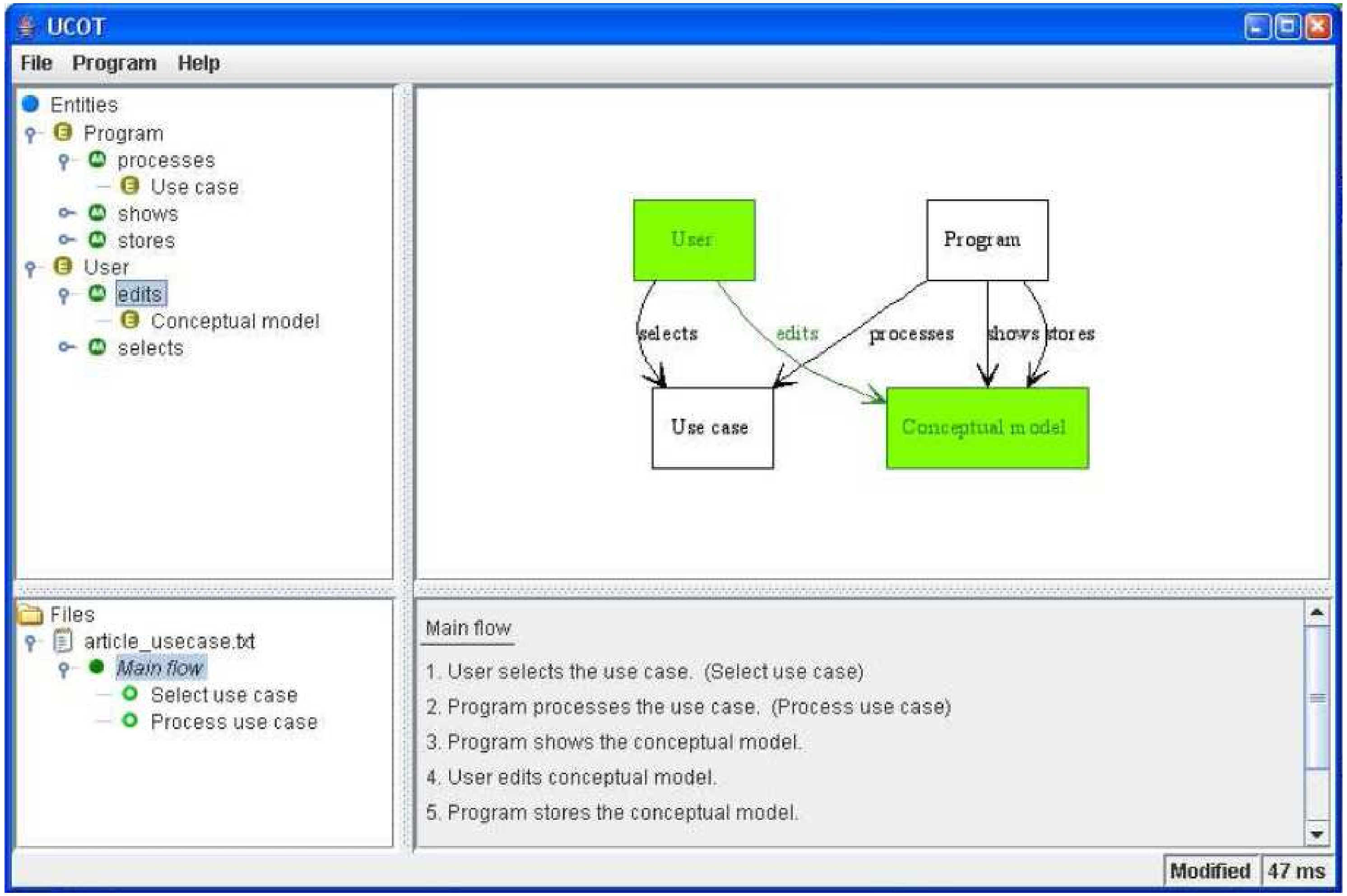
Create conceptual model

```
[name] Main flow [id] 1
[steps]
        User selects the use case. (2)
        Program processes the use case. (3)
        Program shows the conceptual model.
        User edits the conceptual model.
        Program stores the conceptual model.
[end]

[name] Select use case [id] 2
[steps]
        User selects the source of the use cases.
        Program presents the list of the use cases contained in the source.
        User selects use case from the list of use cases.
[end]

[name] Process use case [id] 3
[steps]
        Program passes the use case to the parser.
        Parser returns the parsed use case.
        Program passes the parsed use case to the heuristic.
        Heuristic returns the conceptual model.
[end]
```

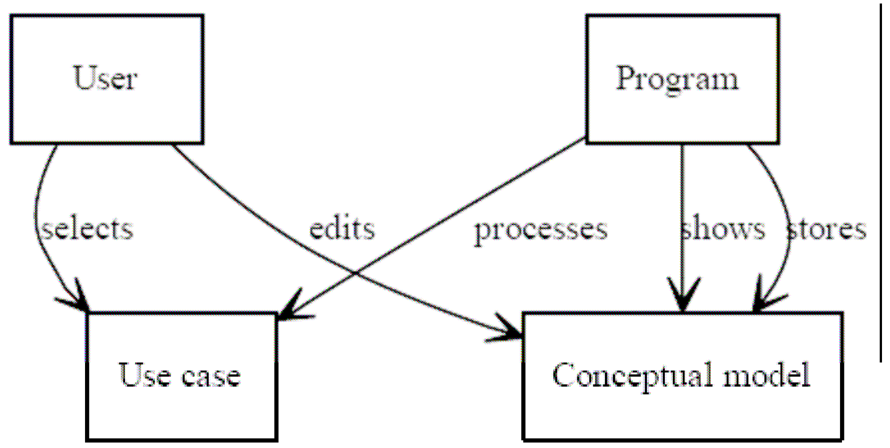
UCOT User Interface



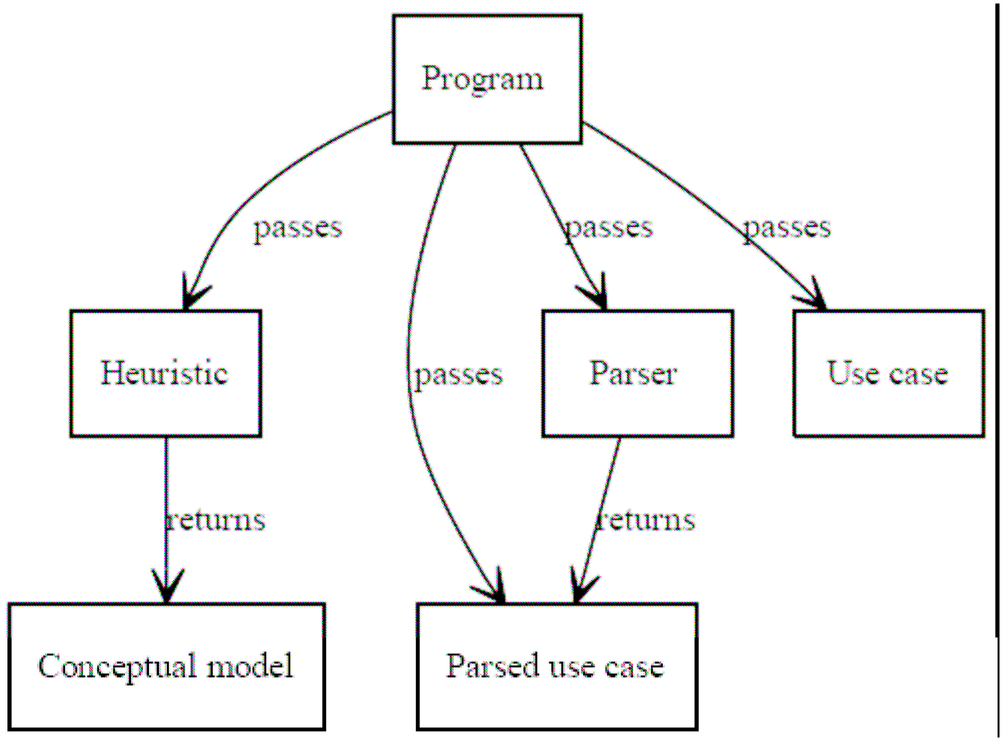
Main flow

1. User selects the use case. (Select use case)
2. Program processes the use case. (Process use case)
3. Program shows the conceptual model.
4. User edits conceptual model.
5. Program stores the conceptual model.

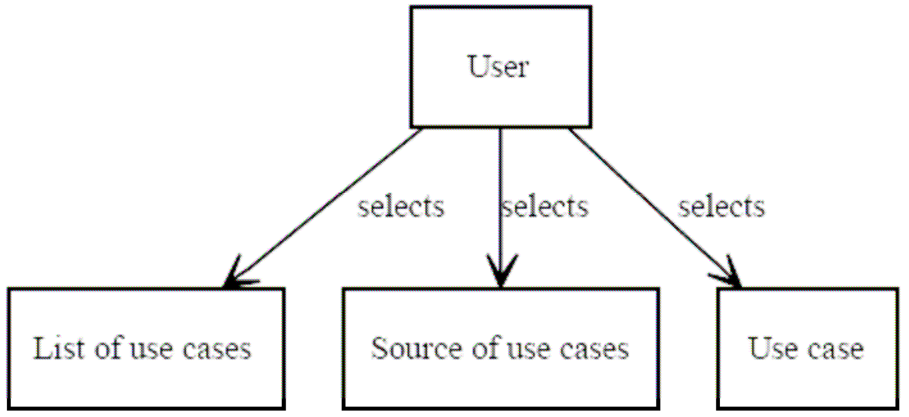
Results (automatically generated)

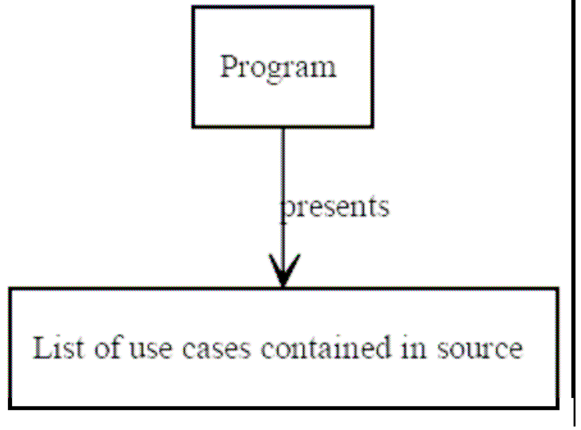


Main Flow



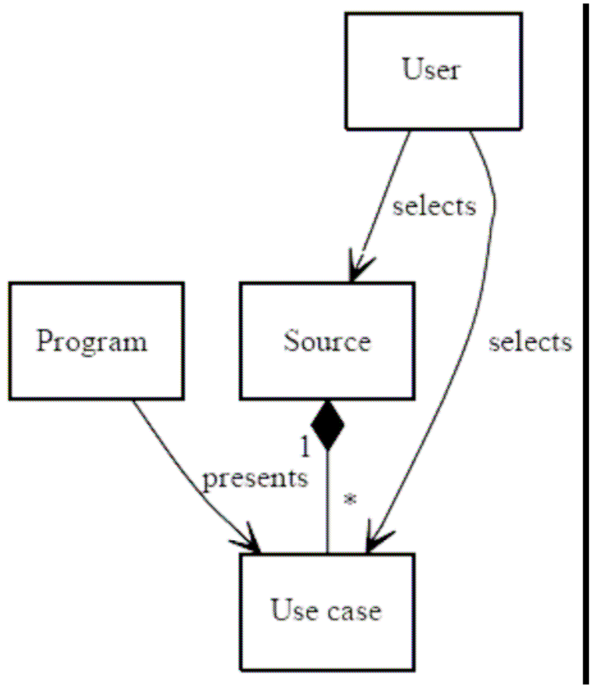
Process Use case



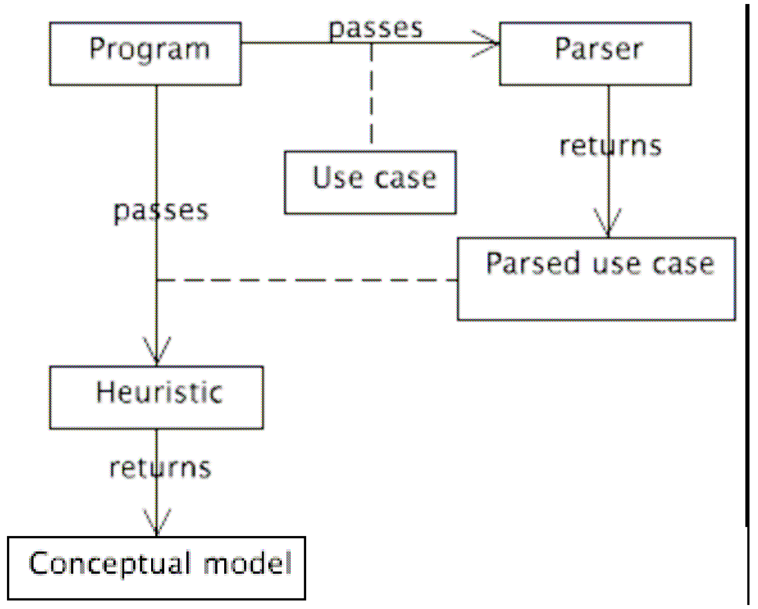


Select Use case

Results (fixed)

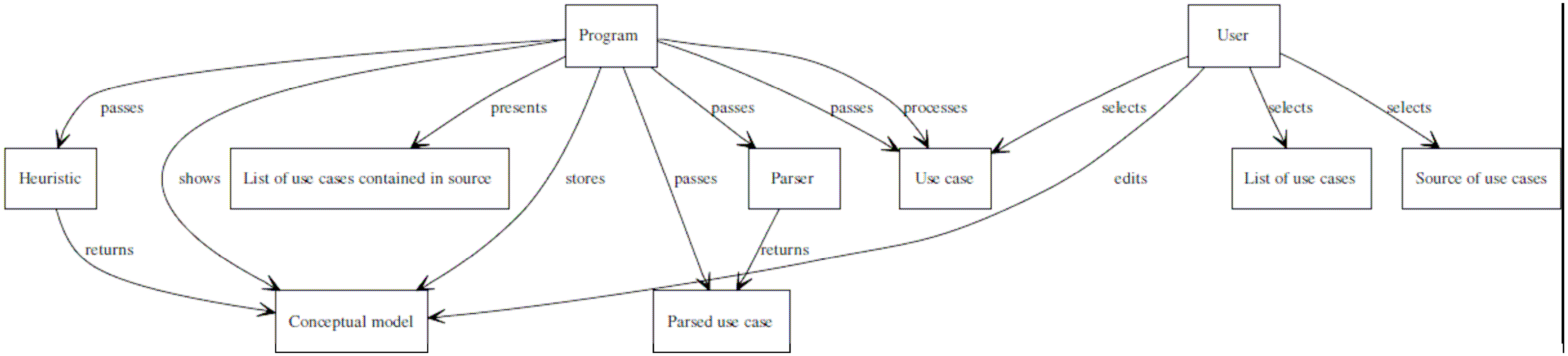


Select Use case (fixed with UCOT)

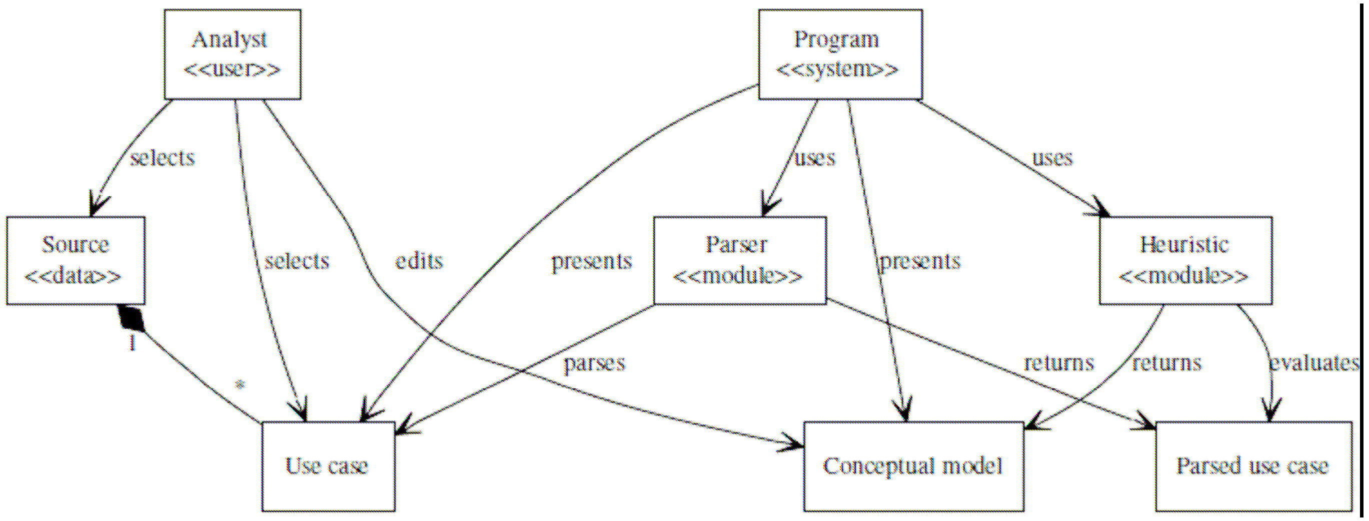


Process Use case (idealized)

Combined model



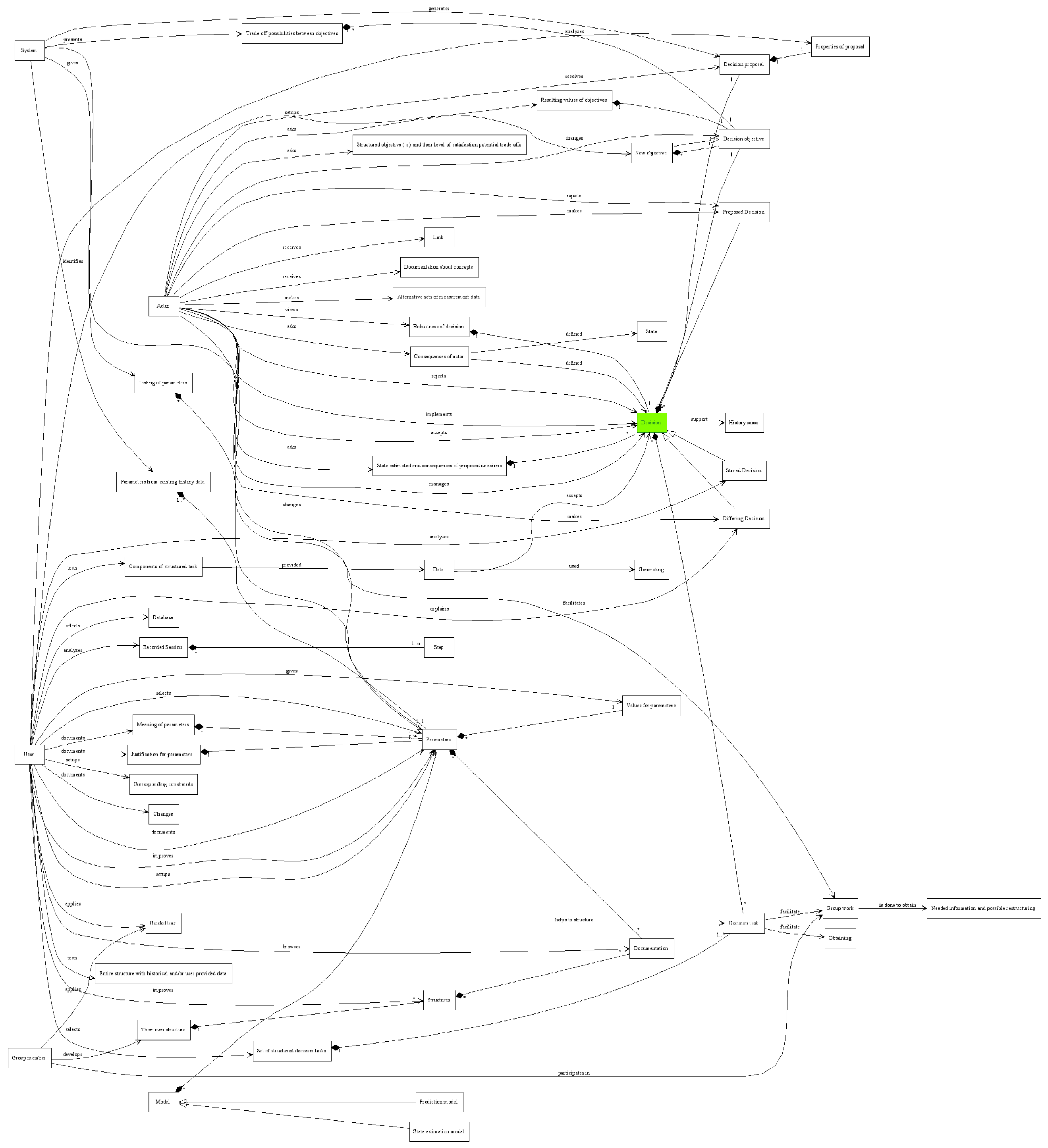
Combined model (initial)



Combined model (fixed)

Evaluation

- Example
  - Generated structure represents the architecture actually implemented (good software architecture?)
- Metamodel & use case processing
  - There should be a way of representing n-ary relations (i.e. relations made of three or more participants)
  - Additional meaning extraction (e.g. relation type classification, additional sentence element processing) should be utilized in heuristic module, but would destroy its language independence
  - Additional semantic annotations should be obtained in the parsing phase to extract more specific entity roles.
- General
  - The quality of output depends essentially on the work done in earlier phases necessary to produce it (writing conventions, terminology, etc)
  - Increasing the number of use cases also increases the need for creating more abstract views of them, although the amount of information in the model can exceed the limits of human cognitive capacity



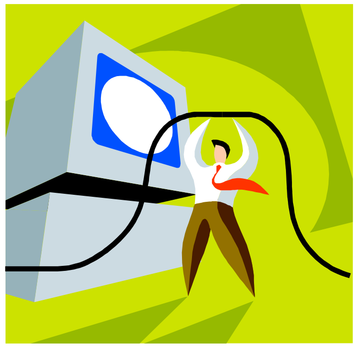
In reality
(work in progress):
Modeling Decision
Support System
based on Statistical
Decision Theory

Conclusion & further research

- UCOT system provides appropriate support to speed up domain understanding by focusing the domain analysis efforts on the most essential domain entities, their relations, and roles as part of the problem to be solved
- Allows role-based separation of relevant concepts
- Advanced heuristics require additional information of the language semantics or input format metadata

- The model should be alternatively be presented in a more behavioral oriented way (cf. sequence diagram)
- Use cases, requirements lists, concept models, and concept definitions ("glossary") should be linked together in a unified structure
- Gathering experience when utilized in larger-scale software production
- More thorough evaluation needed

Thank you!



minurmin@mit.jyu.fi
http://www.mit.jyu.fi/minurmin/