Explanation in Episodic and Continuous Decision Support Systems

Joachim Baumeister^{1,2} **and Albrecht Striffler**^{2}
^{1} University of Würzburg, Institute of Computer Science, Am Hubland, 97076 Würzburg, Germany
^{2} denkbares GmbH, Friedrich-Bergius-Ring 15, 97076 Würzburg, Germany
{firstname.lastname}@denkbares.com

Abstract

Advanced decision support systems demand for their episodic and collaborative use in order to solve complex problems. Further, continuous knowledge representations help to build large knowledge spaces. Decision support systems enhanced in these ways, however, require new approaches to explain the derived decisions. In this paper, we propose an explanation approach that is based on the standardized PROV ontology. We discuss its applicability by giving practical examples.

1 Motivation

A new type of decision support systems is emerging in the practical use in industry. *Episodic and continuous decision support systems* are an advanced interpretation of knowledge-based systems. In comparison to classical decision support systems they emphasize the episodic use of the system for finding (complex) decisions and the use of a continuous knowledge representation for representing large knowledge spaces.

- *Episodic decision making:* A (complex) decision is not made during a single session, but the actual decision process is partitioned over time into different *episodes*. Each episode typically covers a different aspect of the decision process and often more than one user is participating in the episodes. In consequence, we face a (collaborative) decision process; here, a complex decision is taken by the aggregation of a collection of sub-decisions. The sub-decisions cover different aspects of the decision and they are derived in different episodes and by possibly different users.
- *Continuous knowledge representation:* In traditional knowledge systems a single knowledge representation is used to build the knowledge base. More complex and larger systems do benefit from the use of hybrid approaches, integrating different representations into one knowledge base. Here, for a single decision/fact different knowledge representations can be *continuously* interweaved. Systems then have to deal with multiple representations during the reasoning process.

For (complex) decision support systems it is necessary to provide helpful explanation for the derived decisions. In the literature [Roth-Berghofer and Richter, 2008] an explanation scenario is described as depicted in Figure 1.

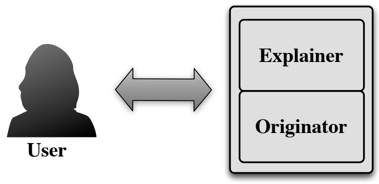


Figure 1: A general explanation scenario.

The user demands for an explanation and interacts with an explanation component. The component consists of the *Explainer* and the *Originator*; in our application scenario the originator is the decision support system, whereas the explainer is the component of the system to generate the explanations to the user.

Transparent explanations improve the general acceptance of users, but can be also used for tutorial and legal purposes by showing the reasons for a particularly derived decision. The commercial application of an episodic and continuous decision support system showed the demand for a new approach in explanation. The following requirements were expressed by the regular users of the system:

- The explanation has to show the temporal development (i.e., the episodes) of a particular decision process.
- All participating users and their competencies have to be integrated in the explanation.
- The explanation has to handle the use of different knowledge representations that were applied for decision making.

This paper presents an extensible explanation approach that meets the requirements stated above. Its main idea is the interpretation of the explanation data as provenance of the entered data and derived decisions but also the decision process itself. As the formal model to represent provenance data the PROV ontology is applied to implement this approach [W3C, 2013a].

The rest of the paper is organized as follows: Section 2 introduces the characteristics of episodic and continuous decision support systems. Section 3 sketches the main ideas of the provenance ontology PROV-O and describes its application within the explanation of decision support systems. Explanation queries can be interpreted as queries to the ontology. In Section 4 typical explanation queries are implemented as SPARQL queries to demonstrate the principal applicability. The paper concludes with a summary of the presented work in Section 5. Also an outlook to future work is given.

2 Episodic and Continuous Decision Support Systems

Before we put more detail on the explanation concept we introduce the concepts of episodic and continuous decision making.

Episodic Decision Making

A complex decision is made not in a "one-shot" session, but multiple users have to contribute their expertise to find a reasonable overall decision. Typically the decision process can be partitioned into a number of aspects that are covered separately. The aggregation of different aspects contributes to an overall decision. Typically, the outcomes of the aspects are represented as (sub-)decisions.

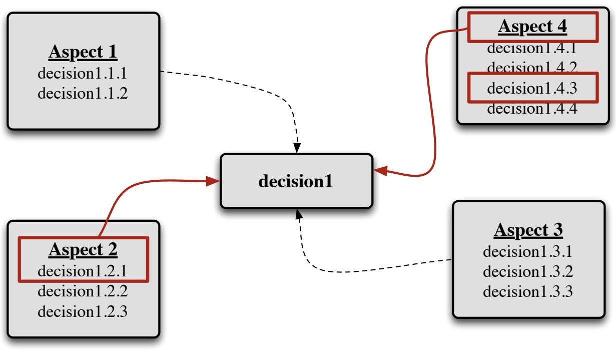


Figure 2: Four aspects are contributing to a final decision `decision1`. Aspects 2 and 4 are responsible for the actual derivation of the decision.

In Figure 2 the aspects 1–4 for the single decision `decision1` are shown. Each aspect itself is represented by a number of questions that need to be answered, so that a decision can be derived. In the example, aspect 2 and aspect 4 actually provide the derived decisions `decision1.2.1` and `decision1.4.3` that support the derivation of `decision1`.

Often not all aspects are covered by a single person but every aspect is handled by a different person or expert group. In medicine, for instance, there exists specialists for the different organs of the human. For a complex evaluation of the patient's physical state, more than one specialist may be consulted. In the technical domain, we see a similar setting: In complex machinery there also exist specialists for the different components of the machine.

Figure 3 shows an exemplary decision process, where the different Users 1–3 are collaborating in a decision process over the time.

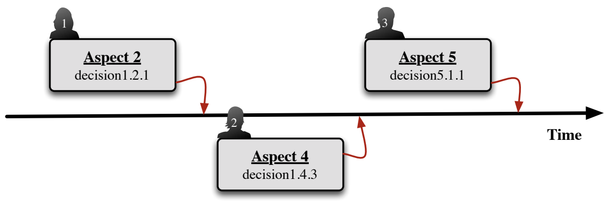


Figure 3: Example decision episodes over time, where different aspects are covered.

The first two decisions `decision1.2.1` and `decision1.4.3` are taken from the previous example shown in Figure 2, whereas User 3 contributes `decision5.1.1` which not relevant for the final derivation of `decision1`.

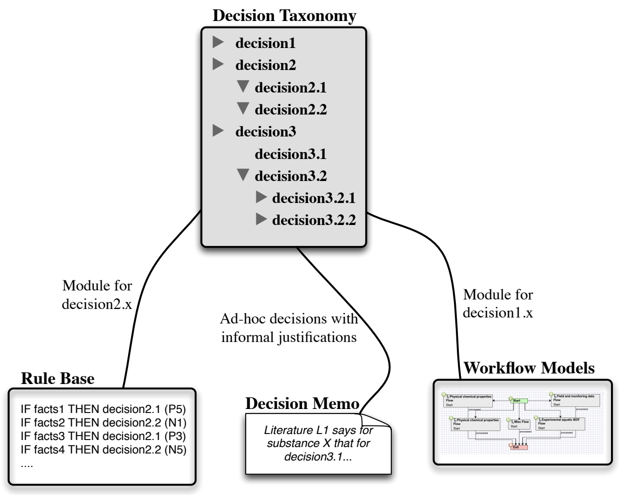
Use of Continuous Knowledge Representations

In complex domains it often is not possible/reasonable to build the entire knowledge base by a single knowledge representation. More precisely, some parts of the knowledge are preferably not formalized by explicit knowledge representations—such as rules or models. Typical reasons for a hybrid approach are as follows:

- *Uncertain domain knowledge:* Parts of the domain are not well-understood in a technical sense. Here, decisions in practice are often based more on past experiences, evidence, and intuition than on strict domain laws and rules.
- *Bloated domain knowledge:* For some parts of the domain, the explicit representation of the knowledge would be too time-consuming and too complex. For instance, much background knowledge needs to be included, that is required for proper decision making. Here, the expected cost-benefit ratio [Lidwell *et al.*, 2003, p. 56] is low, e.g., because many parts will be rarely used in real-world decisions.
- *Restless domain knowledge:* Especially in technical domains, some parts of the domain knowledge are frequently changing due to technological changes. The explicit representation of these parts would require frequent and costly maintenance. Here, also the cost-benefit of the maintenance vs. the utility of the knowledge needs to be evaluated.

Consequently, mixing different knowledge representations with less formal elements seems to be promising. In the past, the knowledge formalization continuum [Baumeister *et al.*, 2011a] was introduced as a mental model to represent different representations in a single systems.

As a pragmatic reasoning approach, we propose to connect the different representations by a common taxonomy of decisions. That way, the different knowledge elements share the same decision space and thus are able to derive the same set of decisions within one process.



```
IF facts1 THEN decision2.1 (P5)
IF facts2 THEN decision2.2 (N1)
IF facts3 THEN decision2.1 (P3)
IF facts4 THEN decision2.2 (N5)
....
```

Literature L1 says for substance X that for decision3.1...

Figure 4: Connecting different knowledge elements by the use of a common decision taxonomy.

In Figure 4, rules, workflow models, and informal decision memos are connected by a taxonomy of decisions.

The combined knowledge base is able to derive the same set of decisions albeit the representations used in reasoning are differing. In literature, formal approaches such as RIF [W3C, 2013b] apply a comparable connection, i.e., decisions are formalized as concepts/instances and rules are defined to derive the existence of the concept/instance.

In a more elaborated approach, the knowledge elements are able to derive decisions in a weighted manner. Here, we propose a score-based approach. Scores have been regularly used as a weighting scheme in knowledge engineering [Puppe, 1998; Miller *et al.*, 1982]. By using scores each knowledge element is not only able to derive a solution categorically, but can attach a score weight to a decision. Every decision provides an account that stores the scoring weights given to the decision during the reasoning process. If a knowledge element "fires", then the corresponding score weight is added to the account of the particular decision. All scoring weights of a single decision are aggregated to a final weight. If the final weight exceeds a predefined threshold, then the decision element is established.

Example: We define a universal set of score weights S = {N1, N2, N3, 0, P1, P2, P3}, where P1,...,P3 are positive score weights and N1,...,N3 are negative score weights. The sum of two equal categories results in the next higher category (e.g. P2 + P2 = P3). A negative and the corresponding positive score weight nullify each other (e.g., N2 + P2 = 0). A decision is established (confirmed), if the aggregation of the collected scoring weights exceeds the score weight P3. In Figure 5, we see that a rule fired a score weight P1 to `decision1`. The decision `decision2` and `decision5` are established, since the aggregation of their score weights exceeds the threshold P3. The decision `decision4` is not established, since a negative weight N3 nullified the positive weight P3.

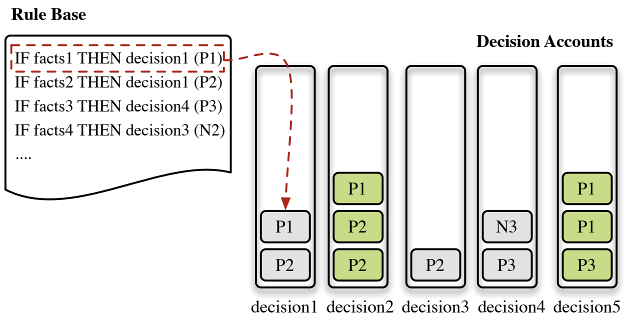

Rule Base
IF facts1 THEN decision1 (P1)
IF facts2 THEN decision1 (P2)
IF facts3 THEN decision4 (P3)
IF facts4 THEN decision3 (N2)
....
Decision Accounts
P1
P2
P1
P2
P2
P2
N3
P3
P1
P1
P3
decision1
decision2
decision3
decision4
decision5

Figure 5: Score accounts of five decisions, and a rule firing a new score weight to the account of `decision1`.

The same decision accounts are also filled by other used knowledge types. For instance, the score weight of an entered decision memo contributes to the account of `decision1`. Further, a traversed workflow model can derive a score weight for `decision2`, which is also added to the decision account.

In this section we described the internal representation of a continuous knowledge representation by the common use of a decision taxonomy. In the following section we introduce an approach to provide explanation capabilities for such a kind of systems.

3 Provenance and Explanation in Decision Support Systems

As motivated above, the process of making a complex decision often involves a number of people contributing to the decision process. Furthermore, the process itself is taking place over a longer period of time. For these reasons, it is very important that the derived decisions are understandable and transparent for all users. These requirements imply the versioning and documentation of decisions and data entries. Changes need to be traceable, as for instance described by [Noy and Musen, 2002; Franconi *et al.*, 2010]. Further also a means of representing the decision process itself is required to be used in an explanation component. Such a component needs to answer (at least) the following questions:

- At which time a particular data was entered and who entered that data?
- Which knowledge elements are responsible for a particular decision?
- What is the history of a particular data and decision?
- Which persons contributed to the process of a particular decision?

We propose the application of the PROV ontology to knowledge elements and the entities of the decision process. The PROV ontology explicitly represents the provenance of entities, i.e., in our case *decisions*, *entered data*, etc. are interpreted as PROV entities. We first give a brief overview of PROV-O and then show its application to decision processes.

3.1 The PROV Ontology in a Nutshell

The PROV ontology [W3C, 2013a] distinguishes three levels of terms defined in the ontology:

1. *Starting Point Terms* to be used to express the basic knowledge about provenance of data.
2. *Expanded Terms* for more expressive definitions of relationships in provenance.
3. *Qualified Terms* integrating the Qualification Pattern [Dodds and Davis, 2012] into the PROV ontology for a very expressive representation of the provenance of data.

In this section we select a helpful subset of "starting points" and "expanded terms" and describe the concepts and relations, that are useful to represent the provenance of decision support systems. In Figure 6 these concepts and relations are depicted.

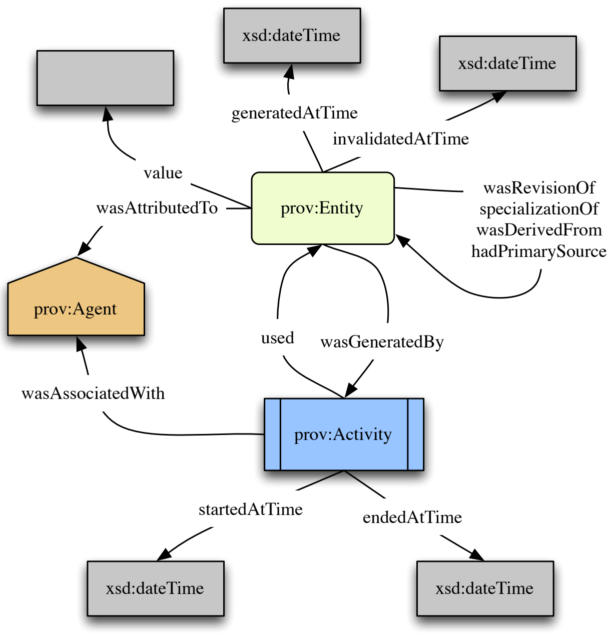


Figure 6: Selected elements of the PROV ontology.

For concepts defined in the PROV ontology the prefix `prov` is used. The three classes `prov:Agent`, `prov:Activity`, and `prov:Entity` are central for describing provenance information. An `prov:Agent` is executing an `prov:Activity` and produces an `prov:Entity`. Consequently, an `prov:Entity` can be attributed to an `prov:Agent` and the `prov:Entity` was generated by a specific `prov:Activity`. An `prov:Activity` is also associated with an `prov:Agent`. In some processes an `prov:Activity` uses an `prov:Entity` for the creation of another `prov:Entity`. An `prov:Activity` has a start and an end time; this is related to the generation time of an `prov:Entity`. When the `prov:Entity` is superseded by a revision (`prov:wasRevisionOf`), then the prov:Entity is invalidated at a specified time. The following properties are also of interest: The property `prov:wasDerivedFrom` states that an instance of `prov:Entity` was transformed into another instance. In decision support systems, the primary source of a specific `prov:Entity` is also of interest (see property `prov:hadPrimarySource`).

In its basic setting we see that the PROV ontology is a suitable starting point for general explanation capabilities in decision support systems. In the following we describe the specific extensions for such systems together with application scenarios.

3.2 The PROV Ontology for Decision Support Systems

When integrating the PROV ontology into decision support systems, we consider a *process-centered provenance*. Here, actions and steps are represented that are used for producing a particular decision.

For the application of the PROV ontology in decision support systems we introduce a number of new concepts sub-classing the known core concepts of PROV. In Figure 7 the most important subclasses are depicted; the prefix `dss` is used for classes introduced for decision support.

At the top of the figure the subclasses of `prov:Entity` are shown: A simple entity can be a decision (`dss:Decision`) or entered data (`dss:FormValue` and `dss:DecisionMemo`). Every decision is associated with a `dss:DecisionAccount` instance, which stores the score weights using `dss:ScoreWeight` instances. The decision account itself is a complex entity, i.e., a `prov:Collection`.

The extension of the concept `prov:Activity` knows two sub-classes: 1) for entering data in memos (`dss:MemoEntry`) and for answering question in forms (`dss:FormEntry`); 2) for the actual derivation of a decision. The latter activity is central for the explanation of different decisions derived during a process.

Two different `prov:Agent` sub-classes are introduced: `dss:TeamMember` to represent users participating in the collaborative decision process and `dss:DomainSpecialist` for building the explicit knowledge base and for giving expert decisions. Further `dss:DSS` represents the actual decision support system.

In a concrete scenario instances of the classes are created storing the provenance information of the decision making process. We demonstrate the concrete use by an example, where instances of the example use the prefix `ex`.

A taxonomy of decisions is typically formalized by narrower/broader relations of the SKOS [W3C, 2009] ontology.

```
ex:decision1 rdf:type dss:Decision;
   skos:narrower ex:decision1.1;
   skos:narrower ex:decision1.2;
   skos:narrower ex:decision1.3.
ex:decision2 rdf:type dss:Decision;
   skos:narrower ex:decision2.1;
   skos:narrower ex:decision2.1.
```

In the example, the decisions `ex:decision1` and `ex:decision2` are defined together with more specific sub-decisions. A particular decision can be inferred in different ways: 1) The use of explicit inference knowledge used by a decision support system. 2) The creation of a decision memo by a user. We examine both alternatives in the following.

Explicit Inference of Decisions

The explicit reference and the ontological representation, respectively, is exemplified by the activity of a firing rule. The instances are graphically depicted in Figure 7-(4).

The user `ex:teamMemberMM` enters a `ex:formValue1` entity during the activity `ex:formEntry1`. The entity `ex:formValue1` is responsible for deriving a specific score weight `ex:scoreWeight1`. The derivation of the score weight is represented by an instance of `dss:DecisionDerivation`, i.e., `ex:decisionDerivation1`, which itself uses a rule included in the rule base `ex:ruleBase1` The activity `ex:decisionDerivation1` added the score weight to the defined score account `ex:decisionAccount1`, which itself is linked to the corresponding decision `ex:decision1`.

Inference by Decision Memos

The reasoning and representation of decisions taken by decision memos is very similar to the approach described above for rules. Actually, only the instances of `dss:DecisionDerivation` and `dss:Data` are connected differently to the score weight of the decision. In Figure 7-(4) a dotted box is depicted at the right. The box

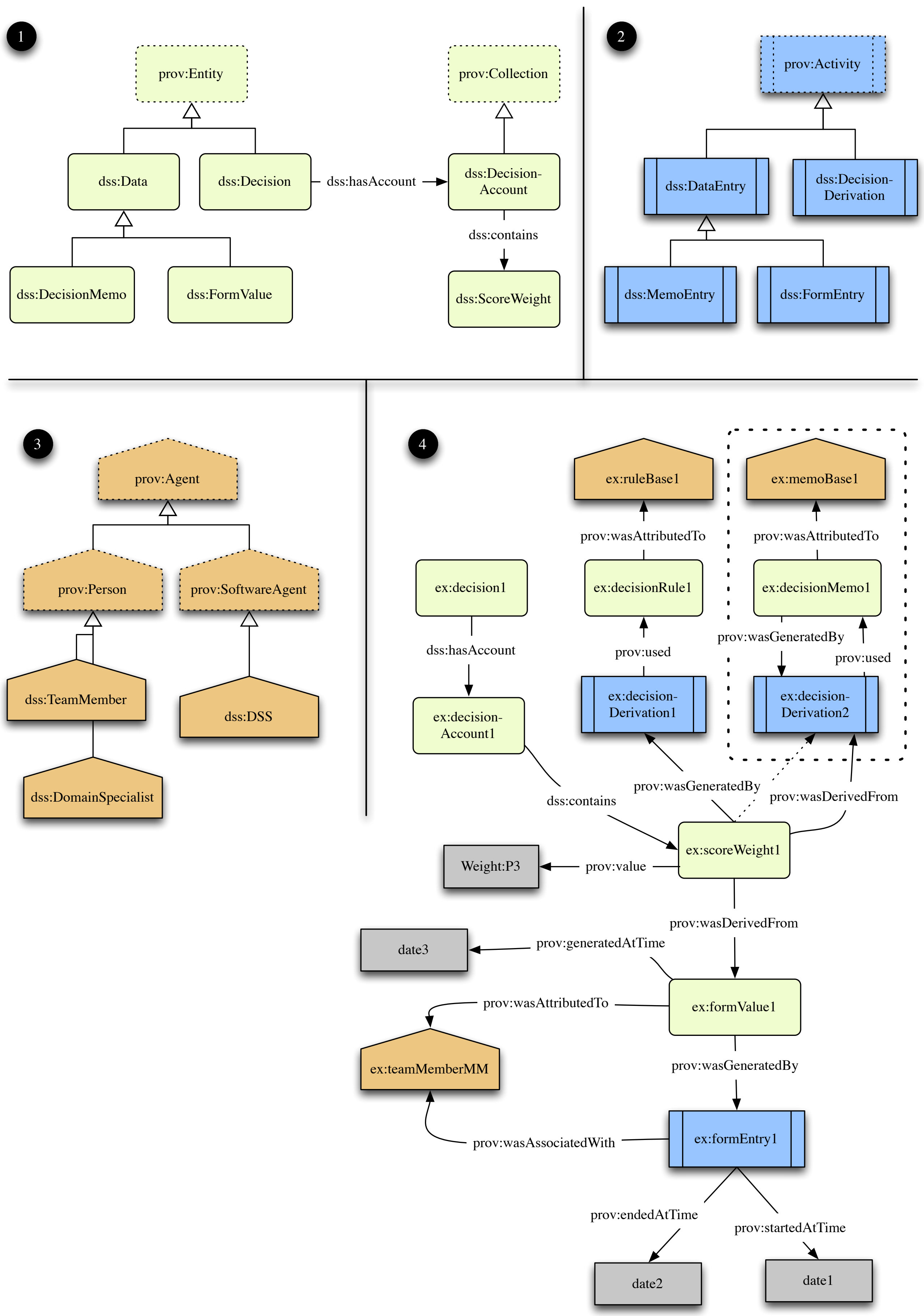


Figure 7: (1) Subclasses and relations `prov:Entity`; (2) Specific subclasses of `prov:Activity` and (3) specific subclasses of `prov:Agent`; (4) Instances and relations created when a decision rules fires for a concrete decision `ex:decision1`. The dotted boxes show the alternative use case, where a decision memo is responsible for deriving the decision.

shows the alternative instances when entering a decision memo.

The user `ex:teamMemberMM` enters the decision memo `ex:decisionMemo1` during the activity `ex:memoEntry1`. In the decision memo the user also enters a score weight for `ex:decision1`. Consequently, the decision memo derives the `ex:scoreWeight1` that was generated by the instance `ex:decisionDerivation2`. This instance is connected with the actual `ex:decisionMemo1` stored in the data base for memos (`ex:memoBase1`).

4 Querying for Explanation

By using an ontology representation of the decision process, the justification of a particular decision can simply be queried. Moreover, ad-hoc explanations can easily be constructed by new queries. When representing the ontology as RDF triples, the standard query language SPARQL [W3C, 2013c] can be used. Of course, an intuitive visualization of the query results needs to be defined, but this is typically up to the application front-end of the decision support system.

In the following, we exemplify the explanation capabilities of the presented PROV extension by defining the SPARQL queries for the questions posed at the beginning of Section 3. It is worth noticing that the definition of SPARQL queries is up to the administrators of the system. For end-users the results of these queries should be presented in a user-friedly manner.

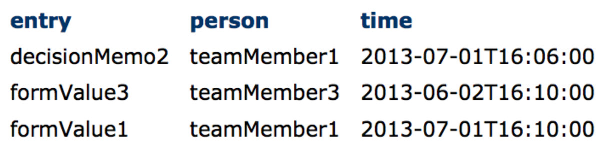
For demonstration purposes we implemented an extended version of the example depicted in Figure 7 in the knowledge modelling environment KnowWE [Baumeister *et al.*, 2011b; 2012].

At which time was a particular data entered and who entered the data?

The following SPARQL query lists all entries and the persons involved in creating a corresponding entry. Additionally the generation time of the entry is shown.

```
SELECT ?entry ?person ?time
WHERE {
  ?entry
     prov:wasGeneratedBy ?activity.
  ?activity
     prov:wasAssociatedWith ?person.
  ?entry
     prov:generatedAtTime ?time.
}
```

Figure 8 shows the results of the SPARQL query above: The three entities `decisionMemo2`, `formValue1`, and `formValue3` are listed with their creators and creation date.



entry	person	time
decisionMemo2	teamMember1	2013-07-01T16:06:00
formValue3	teamMember3	2013-06-02T16:10:00
formValue1	teamMember1	2013-07-01T16:10:00

Figure 8: Entities and persons involved in the creation of the entities.

The query can be further constrained to a specific data entry. Then this query is similar to the last query of this section.

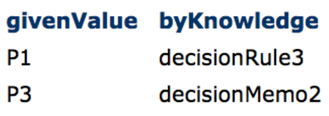
Which knowledge elements are responsible for a particular decision?

The following SPARQL query inspects the connected nodes of the decision instance `ex:decision1` in order to check for derived values. The `FILTER NOT EXISTS` extension guarantees that only valid entities are shown.

```
SELECT ?givenValue ?byKnowledge
WHERE {
  ex:decision1
     dss:hasAccount ?account.
  ?account
     dss:contains ?weights.
  ?weights
     prov:value ?givenValue.
  ?weights
     prov:wasGeneratedBy ?activity.
  ?activity
     prov:used ?byKnowledge.

  FILTER NOT EXISTS {
    ?weights
    prov:invalidatedAtTime
    ?invalidated. }
}
```

Figure 9 shows the results of the SPARQL query above: Derived values are shown together with the knowledge elements—decision memos and rules–that are responsible for the existence of the values.



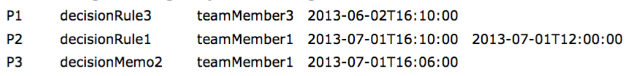
givenValue	byKnowledge
P1	decisionRule3
P3	decisionMemo2

Figure 9: Valid values derived together with the acting knowledge element.

What is the history of a particular data and decision (including involved persons)?

For a given decision `ex:decision1` the following SPARQL query identifies all values that were given to that decision. For each value, also the used knowledge element and the acting person is retrieved. Also the validity of the value is printed; values with empty invalidated column are currently valid.

Figure 10 shows the results of the query: We see that `ex:decision1` retrieved three values, whereas value P2 is already invalidated.



value	usingKnowledge	byPerson	generated	invalidated
P1	decisionRule3	teamMember3	2013-06-02T16:10:00	
P2	decisionRule1	teamMember1	2013-07-01T16:10:00	2013-07-01T12:00:00
P3	decisionMemo2	teamMember1	2013-07-01T16:06:00	

Figure 10: History of values for the given decision `ex:decision1`.

```
SELECT ?value ?usingKnowledge ?byPerson
       ?generated ?invalidated
WHERE {
  ex:decision1
     dss:hasAccount ?account.
  ?account
     dss:contains ?data.
  ?data
     prov:value ?value.
  ?data
     prov:wasDerivedFrom ?entity.
  ?data
     prov:wasGeneratedBy ?activity.
  ?activity
     prov:used ?usingKnowledge.
  ?entity
     prov:wasAttributedTo ?byPerson.
  ?entity
     prov:generatedAtTime
        ?generated.

  OPTIONAL {
     ?data prov:invalidatedAtTime
     ?invalidated. }
}
```

5 Conclusions

We conclude the paper with a brief discussion and an outlook to the future work.

5.1 Discussion

Advanced decision support systems allow for the distributed and episodic handling of complex decision problems. They handle large knowledge spaces by mixing different knowledge representations with informal decision justifications. When implemented in a distributed setting, the transparent justification of derived decisions is of prime importance. In this paper we introduced an explanation approach of continuous knowledge representations that is based on the PROV ontology. We described how an ontology representation of the decision process and the derived decisions can be used to generate transparent explanations.

In the literature the related ontology models can be found: [Evangelou *et al.*, 2005] describe an ontology to support collaborative decision-making. They propose the model KAD (Knowledge-Argument-Decision) to facilitate exchange between decision makers and their argumentation. The KAD ontology model defines the three main classes `discussionParticipant`, `coreEntity`, and `coreProcess`, where their semantics is related or can be aligned to the starting point terms of the PROV ontology. Here, more focus is set on supporting the argumentation and discussion between decision makers. [Kornyshova and Deneckère, 2010] also propose an ontology for decision making. The *decision making ontology* (DMO) tries to support IS engineers in their decision making during an information systems project. The proposed ontology is evaluated by instantiating it to a requirements engineering process. The ontology is very elaborated and could be connected with the PROV ontology. For our purposes (continuous knowledge representation and episodic use) the extensions described in Figure 7 (1) need to be also made.

5.2 Future Work

At the current state, explanations are based on SPARQL queries. Albeit a very general approach, the construction of such queries can be cumbersome for standard users. For this reason we aim to define a simplified language to define explanation queries quickly in an intuitive manner. In Section 4 we demonstrated the access to typical explanation queries by SPARQL expressions. Although the shown results include all relevant information needed for an explanation, the presentation is likely to be not very intuitive. Therefore we are planning to investigate ontology visualization approaches [Fluit *et al.*, 2002; Katifori *et al.*, 2007] to render the results of the explanation query in a more user-friendly manner.

As the next practical step we are planning to implement and evaluate the proposed ontology and explanation capabilities for a decision support system, that is already in use. The KnowSEC system supports the decision work for chemical safety within a unit of the Federal Environment Agency in Germany (Umweltbundesamt). At the current state, the systems manages more than 42.000 sub-decisions for more than 11.000 chemical substances; many of the decisions were automatically derived. We refer to [Baumeister *et al.*, 2013] for more details.

References

[Baumeister *et al.*, 2011a] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe. Engineering intelligent systems on the knowledge formalization continuum. *International Journal of Applied Mathematics and Computer Science (AMCS)*, 21(1), 2011.

[Baumeister *et al.*, 2011b] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe. KnowWE: A semantic wiki for knowledge engineering. *Applied Intelligence*, 35(3):323–344, 2011.

[Baumeister *et al.*, 2012] Joachim Baumeister, Jochen Reutelshoefer, Volker Belli, Albrecht Striffler, Reinhard Hatko, and Markus Friedrich. KnowWE - a wiki for knowledge base development. In *The 8th Workshop on Knowledge Engineering and Software Engineering (KESE2012)*, http://ceur-ws.org/Vol-949/kese8-05_04.pdf, 2012.

[Baumeister *et al.*, 2013] Joachim Baumeister, Albrecht Striffler, Marc Brandt, and Michael Neumann. Towards continuous knowledge representations in episodic and collaborative decision making. In *The 9th Workshop on Knowledge Engineering and Software Engineering (KESE2013)*, 2013.

[Dodds and Davis, 2012] Leigh Dodds and Ian Davis. *Linked Data Patterns Linked Data Patterns Linked Data Pattern*. http://patterns.dataincubator.org/book, 2012.

[Evangelou *et al.*, 2005] Christina Evangelou, Nikos Karacapilidis, and Omar Abou Khaled. Interweaving knowledge management, argumentation and decision making in a collaborative setting: the kad ontology model. *International Journal of Knowledge and Learning*, 1(1):130 – 145, 2005.

[Fluit *et al.*, 2002] Christiaan Fluit, Marta Sabou, and Frank van Harmelen. Supporting User Tasks through Visualisation of Light-weight Ontologies. In *Handbook on Ontologies in Information Systems*, pages 415–432. Springer, Berlin, 2002.

[Franconi *et al.*, 2010] E Franconi, T Meyer, and I. Varzinczak. Semantic diff as the basis for knowledge base versioning. In *13th International Workshop on Non-Monotonic Reasoning (NMR)*, pages 7–14, 2010.

[Katifori *et al.*, 2007] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methods - a survey. *ACM Comput. Surv.*, 39(4), November 2007.

[Kornyshova and Deneckère, 2010] Elena Kornyshova and Rébecca Deneckère. Decision-making ontology for information system engineering. In Jeffrey Parsons, Motoshi Saeki, Peretz Shoval, Carson C. Woo, and Yair Wand, editors, *ER*, volume 6412 of *Lecture Notes in Computer Science*, pages 104–117. Springer, 2010.

[Lidwell *et al.*, 2003] William Lidwell, Kritina Holden, and Jill Butler. *Universal Principles of Design*. Rockport Publishers, October 2003.

[Miller *et al.*, 1982] Randolph A. Miller, Harry E. Pople, and J. Myers. INTERNIST-1, an Experimental Computer-Based Diagnostic Consultant for General Internal Medicine. *New England Journal of Medicine*, 307:468–476, 1982.

[Noy and Musen, 2002] Natalya F. Noy and Mark A. Musen. PromptDiff: a fixed-point algorithm for comparing ontology versions. In *In 18th National Conference On Artificial Intelligence (AAAI-2002*, pages 744–750, 2002.

[Puppe, 1998] Frank Puppe. Knowledge Reuse among Diagnostic Problem-Solving Methods in the Shell-Kit D3. *International Journal of Human-Computer Studies*, 49:627–649, 1998.

[Roth-Berghofer and Richter, 2008] Thomas R. Roth-Berghofer and Michael M. Richter. On explanation. *Künstliche Intelligenz*, 22(2):5–7, May 2008.

[W3C, 2009] W3C. SKOS Simple Knowledge Organization System reference: http://www.w3.org/tr/skos-reference, August 2009.

[W3C, 2013a] W3C. PROV-O: The PROV Ontology: http://www.w3.org/tr/prov-o/, April 2013.

[W3C, 2013b] W3C. RIF-Core Recommendation: http://www.w3.org/tr/rif-core/, February 2013.

[W3C, 2013c] W3C. SPARQL 1.1 recommendation: http://www.w3.org/tr/sparql11-query/, March 2013.