Decision Rationales as Models for Explanations

Kenneth Baclawski
Northeastern University
Boston, Massachusetts USA
Ken@Baclawski.org

Abstract

A decision rationale describes the reasons for a decision in an engineering or software development process, so it is a kind of explanation. Conversely, explanations are commonly for decisions that have been made. In this article, we develop a reference ontology for decision rationales, which captures the common features of explanations for decisions in a domain-independent manner. The intention is to tie together the many techniques for explainability in different domains so that the techniques can be shared and possibly even interoperate with one another.

1 Introduction

A decision rationale is an artifact that describes the reasons for a decision. In practice, organizations commonly do not record the knowledge generated during a decision making process. As a result, it can be an expensive and painful process to revisit a past decision when it becomes apparent that the decision is no longer appropriate (Spacey 2016). This problem has been recognized for software development processes, and there are now a number of software tools that assist

developers in capturing and managing decision rationales (See: Section 5)

It should be apparent that decision rationales are a form of explanation; namely, the answer to why a decision was made. Explanations have recently become an important issue. As stated in the Communiqué of the Ontology Summit (2019), “With the increasing amount of software devoted to industrial automation and process control, it is becoming more important than ever for systems to be able to explain their behavior. In some domains, such as financial services, explainability is mandated by law. In spite of this, explanation today is largely handled in an unsystematic manner, if it is handled at all.”

While not all explanations are in response to a decision, such explanations are a significant share of all explanations. Accordingly, a framework for decision rationales would contribute to a common framework for explanations in general. In this article we develop a reference ontology for decision rationales. A reference ontology is an intermediate ontology that is more specific than foundational ontologies (also known as “upper ontologies”) but more general than domain ontologies. A reference ontology deals with a specific issue but is otherwise domain-independent. The advantage of a reference ontology is that it can link together techniques from different domains for purposes such as data integration, software reuse and interoperability. In particular, research and tools for decision rationales for engineering processes could be used for making other systems more explainable.

The reference ontology that we develop originated from the work of Sriram (2002) as well as (Duggar and Baclawski, 2007). The requirements for this ontology were taken from wide range of sources, especially from the Ontology Summit 2019 Baclawski et al (2019), and the fields of Explainable Artificial Intelligence Srihari (2020), commonsense knowledge and reasoning Berg-Cross (2020), medical explanations Baker, Al Manir, Brenas, Zinszer, and Shaban-Nejad (2020), and financial explanations (Bennett 2020).

To illustrate a decision making process, we will use a running example of a specific decision making process for dealing with a problem in industry known as No-Trouble-Found (NTF) or No-Faults-Found (Accenture Communications 2016). The NTF problem is that components used in application areas, such as automobiles, electric utilities, and manufacturing, have mechanisms for indicating component failure. The failure is typically advertised with an alarm. When an alarm is raised, the component may be replaced at little or no cost under the terms of a warranty or service contract. The component that raised the alarm is returned to the supplier and tested in their laboratory. Remarkably, as much as 25% to 70% of the time, the returned component operates correctly when tested. To deal with the problem, the manufacturer will need to test the returned components to determine whether they function correctly. This will generally involve a series of tests that are used to make the decision about whether a component is actually faulty as shown in Figure 1. The figure shows a three-step decision making loop, but an actual decision making process for NTF could have many more steps. While the running example we are using is relatively specific, it is similar to many other decision making processes in which there are three alternatives: accept, reject or get more information.

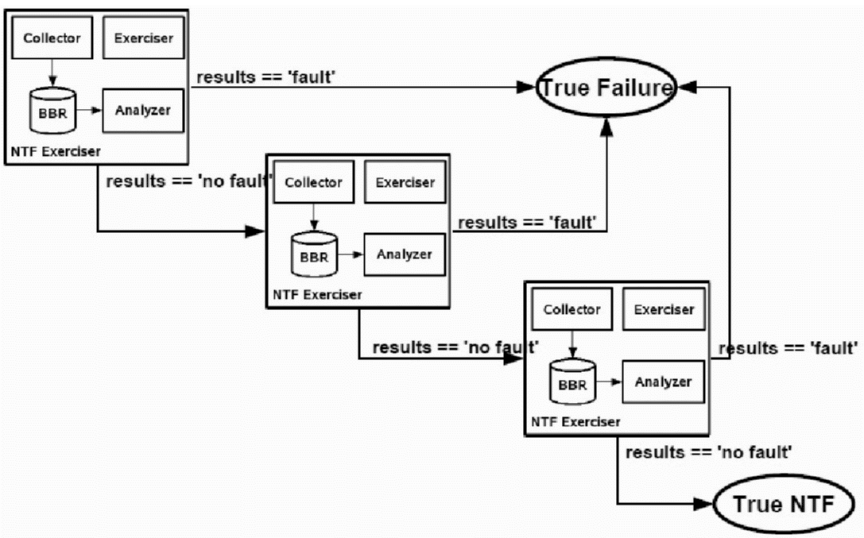

Figure 1: Sequential Hypothesis Flowchart for Electronic Systems and Components Evaluated as "Suspect NTFs" from Baclawski et al (2018)

In Section 2, we give some background for decision rationales and compare them with explanations. We then give some of the reasons why it is useful to document decision rationales in Section 3. Generally, one must capture decision rationales immediately or not at all. Consequently, decision rationale management must be an integral part of the software development process. Similarly, explainability should drive the software engineering process from the earliest stages of planning, analysis and design (Clancey 2019). In Section 4 we discuss the process whereby decision rationales are developed. The reference ontology decision rationales is presented in Section 5. We end with a conclusion and acknowledgments.

2 Background

In this section we give some background on decision rationales and compare them with the more general concept of an explanation. The Ontology Summit 2019 covered the notion of explanation so it is worthwhile to review the definition of this concept given there:

> An explanation is the answer to the question “Why?” possibly also including answers to follow-up questions such as “Where do I go from here?” Accordingly, explanations generally occur within the context of a process, which could be a dialog between a person and a system or could be an agent-to-agent communication process between two systems. Explanations also occur in social interactions when clarifying a point, expounding a view, or interpreting behavior. In all such circumstances in common parlance one is giving/offering an explanation (Ontology Summit 2019).

While explanation has a long philosophical history dating back at least to 5000 BCE, formal treatments of rationales are relatively recent. Perhaps the earliest such treatment was the school of philosophy known as scholasticism that dominated teaching in European universities from roughly 1100 to 1700. It focused on how to acquire knowledge and how

to communicate effectively so that it may be acquired by others. It was thought that the best way to achieve this was by replicating the discovery process and by arguing for and against alternatives (O'Boyle 1998). While scholasticism arose in the context of religious instruction, it soon spread to other disciplines.

Another example of scholarly work on decision rationales is in the legal domain. From ancient times, the rationales for legal decisions have been recorded and used in subsequent decisions. This is the basis for what is now referred to as “common law.” There is a substantial scholarly literature on decision rationales in the legal domain. This should not be too surprising since argumentation is so fundamental to legal decisions, and since it is still a requirement that not only the decision itself but also the rationale for the decision should be documented.

In spite of rationales being common in the legal domain, they are relatively uncommon in law codes, and even when laws have explicitly stated rationales, their standing is ambiguous. The Constitution of the United States has a published rationale in the form of a series of articles called the Federalist Papers (Hamilton, Madison, and Jay, 1787). However, the Constitution itself does not explicitly include a rationale, so whether the Federalist Papers could be used by courts for deciding cases is controversial. There is only one amendment, namely the Second Amendment, that explicitly includes a rationale, albeit a very brief one. The interpretation of this rationale and of the amendment as a whole has been highly controversial. Prior to the year 2008, the rationale was taken to be a limitation on the amendment, essentially giving the states the power to organize militias and allowing individuals to bear arms for this purpose. Up to that time, states and the federal government had the authority to regulate ownership of arms for other purposes. However, in 2008, the United States Supreme Court reinterpreted the Second Amendment by ignoring the rationale (Brennan Center 2018; Greenhouse 1998). From the second citation: “Many are startled to learn that the U.S. Supreme Court didn't rule that the Second Amendment guarantees an

individual's right to own a gun until 2008, when District of Columbia v. Heller struck down the capital's law effectively banning handguns in the home. In fact, every other time the court had ruled previously, it had ruled otherwise." This is good case study to show that including or not including a rationale can result in dramatically different interpretations of a law.

3 Purpose of Documenting Decision Rationales

We now give some of the reasons why decision rationales should be documented and reviewed. Put more succinctly (if somewhat inaccurately), we give a rationale for rationales.

An important part of every decision rationale for software development is the list of the alternatives that were considered. Documenting these options can be useful in themselves. According to Sullivan (1999), "...part of the value of typical software product, process or project is in the form of embedded options. These real options provide design decision-makers with valuable flexibility to change products and plans as uncertainties are resolved over time."

Possibly the most dramatic example of this was a decision for the Ariane V rocket software that was not reconsidered for the Ariane V rocket. The result was that the rocket crashed on its first launch (Gleick 1996).

It might be worth examining in some more detail what the design decision was that resulted in the Ariane V crash. The Ariane V rocket reused vehicle guidance software from the Ariane IV. These different rockets used different processors and the reuse of the Ariane IV software code failed to operate as expected in the Ariane V. The failure occurred in the inertial reference system, or the Système de Référence Inertielle (SRI). The failure was due to a software exception during execution of a data conversion from a 64-bit floating point in a variable for Horizontal Bias (BH) to a 16-bit signed integer value. The floating point number

which was converted had a value greater than what could be represented by a 16-bit signed integer. The use of a 16-bit signed integer in the Ariane IV was a design decision that was made during the development of the SRI. This design decision was documented and even rigorously proven to be correct for the Ariane IV. Unfortunately, the specifications for the Ariane IV that were used in this proof are not satisfied by the Ariane V. From the Inquiry Report, "The reason for the three remaining variables, including the one denoting horizontal bias, being unprotected was that further reasoning indicated that they were either physically limited or that there was a large margin of safety, a reasoning which in the case of the variable BH turned out to be faulty. It is important to note that the decision to protect certain variables but not others was taken jointly by project partners at several contractual levels." However, the reasoning (i.e., the proof) was not included in the source code so it was not reviewed when the software was reused for the Ariane V. This is an example to show that a formal proof of correctness of software is useless if it is not reconsidered when circumstances change. It also shows the risks associated with software reuse (Lions 1996).

If, as it is hoped, systems begin to be more explainable, the experiences with rationale management could be useful lessons. As the Ariane V disaster illustrates, one such lesson is the issue of decision rationale reusability. The purpose of reusability is to save time and resources and reduce redundancy by taking advantage of assets that have already been created in some form within the software product development process (Lombard Hill Group 2014). Unfortunately, software reuse has not been very successful in general (Schmidt 1999).

4 The Decision Rationale Development Process

The process model for decision rationale development is a basic decision making loop, but it extends it by specifying a data model for the resulting rationale. A use case diagram showing two of the actors and activities during formal documentation and use of decision analysis is shown in

Figure 2, taken from (Duggar and Baclawski, 2007). The two roles/actors in this figure are the developer of the decision analysis documentation and the user of the decision analysis. The user is the agent who is seeking an explanation of the decision. The developer can perform a number of actions on the repository of decision rationales, such as create, modify and reuse/repurpose. Other use cases that are not shown are concerned with activities such as reconsidering decisions and inference/reasoning.

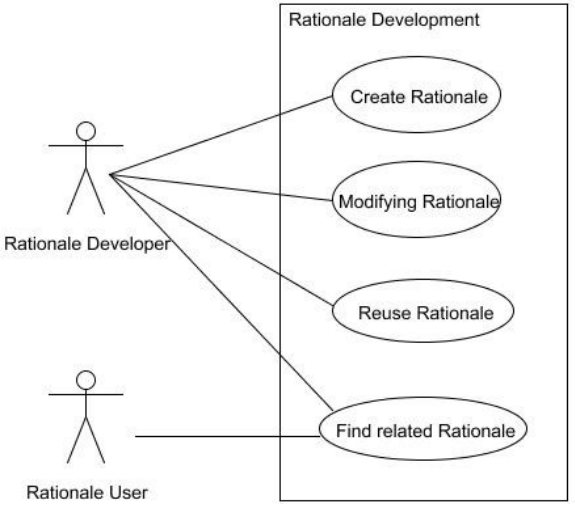

Figure 2: Use Case Diagram for Decision Rationales

The process model for decision rationale development is usually a sub-process of a larger development process. When an issue has been encountered for which a decision is required, a decision making process is performed. Determining and identifying the issue to be resolved may itself be a decision that requires its own decision making process. An example of a process for developing the decision rationale is shown in Figure 3. The process involves a number of steps and iterations as follows:

1. Enumerate all the assumptions that are relevant and can be inferred based on the context or situation.

2. Exhaustive list of all the alternatives that can be chosen for a particular decision have to be documented.
3. Similarly, a list of criteria based on which any alternative would be chosen for an issue/problem is documented.
4. Both step 1 and step 2 are done iteratively till a satisfied list of both alternatives and criteria are available.
5. Relevant arguments for each alternative based on the list of criteria are obtained.
6. Based on the arguments put forward a decision is recommended.

The whole process from steps 1 to 6 could then be iterated till a satisfactory decision is obtained.

This decision rationale development loop is a special case of the general decision making loop developed in (Baclawski et al, 2017). The ontology for the decision making loop is available online at (Baclawski 2016).

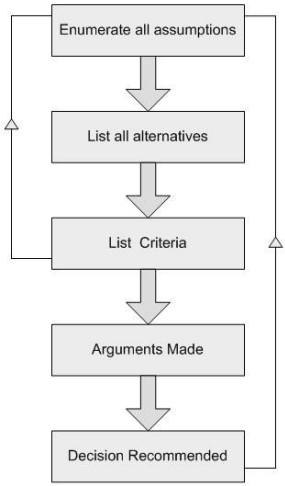

Figure 3: Example of a Decision Rationale Development Loop

For the running example of the NTF decision making process, each step in the process can have three possible outcomes. The component may be found to be actually faulty, the component may be found to be functioning normally, or the component test was unable to establish the condition of the component with sufficient confidence. When the last of these occurs, another test is performed. The rationale for each step in this process has three alternatives. The criterion for each alternative is commonly a range of values for a measurement. The argument for the decision of one step could be as simple as checking whether the measurement is within the range for the corresponding alternative or it could be a more complex statistical or machine learning classification involving both the current measurement and previous measurements.

Some software tools are available for rationale development. Compendium (2020), designVUE (2020) and SEURAT Website (2020) are examples of open source projects that include support for capturing decision rationales. Rationale® (2020) is a commercial product. Gelder (2007) reviews this product. The primary purpose of these tools is to document design decisions during software development. The tools can also be used for documenting more general argumentation, such as in legal cases. These tools do not appear to make use of ontologies.

5 Decision Rationale Reference Ontology

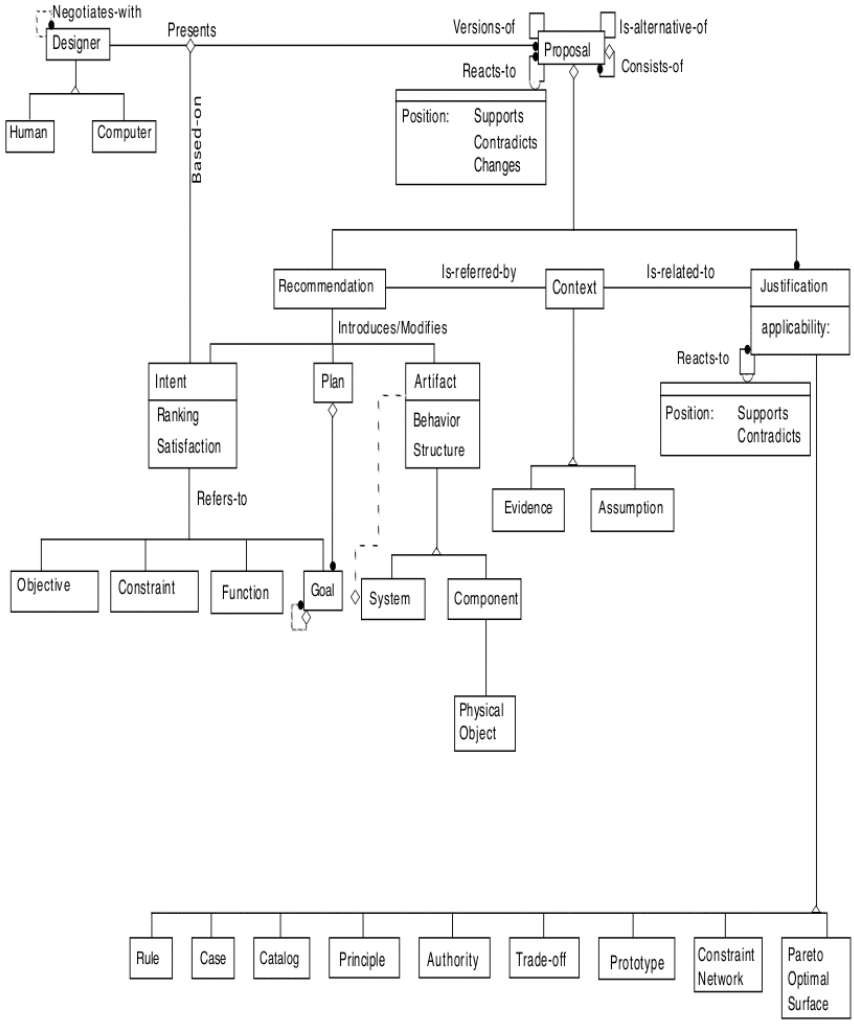

Figure 4: The Design Recommendation and Intent Model

We now formalize the notion of a decision rationale as a reference ontology. The basis for our ontology is the Design Recommendation and

Intent Model (DRIM) that was developed for engineering design decisions but is not limited to that domain (Sriram 2002). The DRIM is shown in Figure 4, using the Object Modeling Technique. We also used some ideas from our own decision rationale ontology in Duggar and Baclawski (2007) which was intended for software engineering using the Eclipse Process Framework.

A number of other reference ontologies were important inputs to our ontology, including reference ontologies for situation awareness, provenance and decision making. Situation awareness means simply that one knows what is going on around oneself. In operational terms, this means that one knows the information that is relevant to a task or goal. The notion of decision rationale fits well with situation awareness, since a decision rationale is the awareness of the information relevant to the making of a decision. Accordingly, we view a decision rationale as a situation. The ontology for situations and situation awareness was first developed in (Baclawski, Malczewski, Kokar, Letkowski, and Matheus, 2002).

The provenance of an entity represents its origin. This includes descriptions of the other entities and the activities involved in producing and influencing a given entity. All of the various objects involved in a decision rationale are entities for which provenance is important. For example, the decision rationale itself, the problem that is to be solved, the various proposals for solving a problem, and the various arguments in favor of or opposed to each proposal, should all be annotated with the person (or other agent) that created the entity, the time when the entity was created, and so on. The PROV ontology was used for provenance information (PROV Ontology 2013).

Given that a decision rationale is the recording of a decision making process, the process whereby the decision is made should be compatible with the structure of the decision rationale. The ontology for decision making that we use is the Knowledge Intensive Data System (KIDS) (Baclawski et al, 2017). In the KIDS framework, the decision making

process is a loop in which a situation evolves iteratively to achieve the final decision. In the process, subsidiary decisions will be made, each represented by its own situation. The decision rationale ontology is intended to be one kind of situation that the KIDS framework applies to.

While the decision rationale ontology we present here is intended primarily for software development decision making, it is domain-independent and so has other potential application domains. It could be applied to more general engineering decision making; indeed, this was the original domain for the DRIM model from which the decision rationale ontology was derived. Another potential domain is legal decisions. While we are not aware of any ontologies specifically for legal decision rationales, the legal literature does have examples of work on representing both classifications and argumentation rules (Berman and Hafner, 1993; Loui and Norman, 1995).

The decision rationale ontology was developed using Protégé (Protégé 2004; Musen 2015). It imports the PROV ontology so that provenance information can be maintained in a standard manner PROV (2013), and all of the decision rationale ontology classes are subclasses of the prov:Entity class, except for the Collaboration class which is a subclass of prov:Activity. The Rationale class is a subclass of the kids:DirectiveSituation class of the KIDS ontology Baclawski et al (2017) which, in turn, is a subclass of the sto:Situation class of the Situation Theory Ontology (Baclawski, Malczewski, Kokar, Letkowski, and Matheus, 2006). The Decision Rationale Ontology is available online (Baclawski 2020).

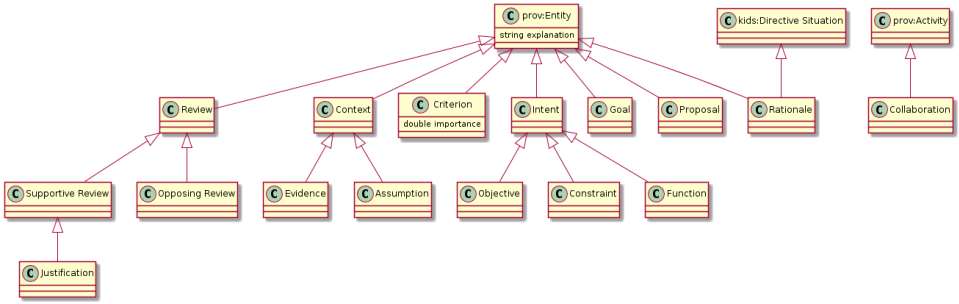

Figure 5: Class Hierarchy of the Decision Rationale Ontology

Figure 5 shows the class hierarchy of the Decision Rationale Ontology. The notation in this figure uses a UML-like notation, but the classes in the hierarchy are not limited to classes in object-oriented software engineering. As noted earlier, all of the classes are subclasses of either prov:Entity or prov:Activity. As a result, all decision rationale artifacts will have all of the many features that the PROV ontology provides, including versioning and provenance information. We added an explanation attribute to the prov:Entity class so that all decision rationale artifacts have a uniform way to explain their role. Potentially, the explanation attribute could contribute to an explainability process as discussed below. The explanation attribute is specified to be a string, but other media could also be used such as diagrams or videos.

The central class in the ontology is the Rationale class. This class reifies the notion of a decision rationale. In addition to being a subclass of prov:Entity, the Rationale class is a subclass of kids:DirectiveSituation which links the Rationale with the ontology of the decision making process that produces the decision rationale. During such a process, a decision may depend on other decisions, and this is represented by the dependsOn object property. In the running example of an NTF decision making process, each step of the process depends on the previous step. To understand how the Rationale class represents a decision rationale we need to examine the object properties shown in Figure 6. The Goal class represents the problem that the decision process is solving. For the NTF

problem the goal is to determine whether or not a returned component is actually faulty. Various alternative Proposals are suggested, one of which is selected as the recommendation. A Proposal can have sub-proposals specified by the consistsOf object property. In the NTF example, the three alternatives are the proposals. In this example, a proposal could have a more complex structure if a component test is more elaborate. Indeed, a component test could itself be a decision making process. The proposals need to satisfy various Criteria. The Criterion class serves to specify how important a particular requirement is, where an importance level of 1 means that the criterion is mandatory, while lower levels represent criteria that are desirable but not essential. The actual requirement is specified by the Intent class, which has subclasses Objective, Constraint and Function that specify different kinds of requirements. An Objective is a characteristic that is to be optimized. A Constraint is a mandatory restriction such as a maximum allowed value. A Function requirement is a performance characteristic of activities or behavior of the solution to the problem. For the NTF example, the Intent could be a Constraint if the test is a simple measurement or the Intent could be a Function if the test is a more complex test of the behavior of the component.

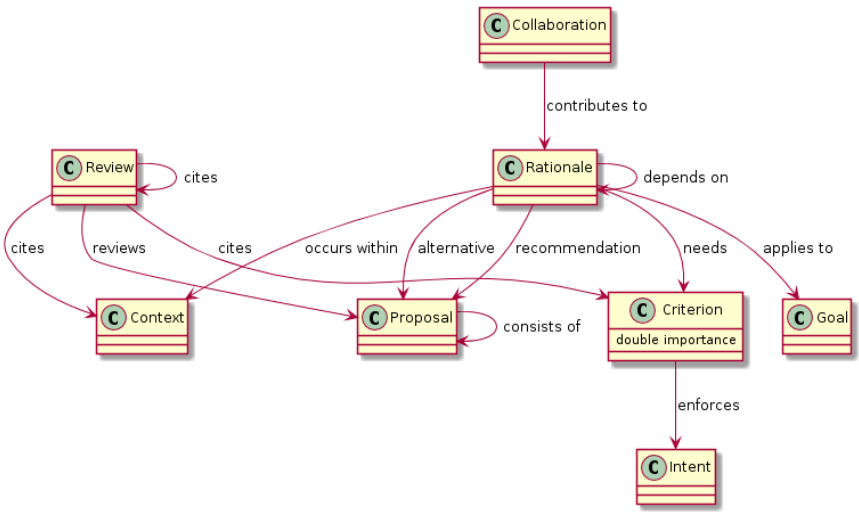

Figure 6: Object Properties of the Decision Rationale Ontology

Since the decision is a selection among alternatives, one needs some way to distinguish them. The is done by means of Reviews. Each Review gives an argument either in favor of a Proposal or against a Proposal. The subclasses SupportiveReview and OpposingReview distinguish these two cases. The explanation for the final decision that selects the recommendation is the Justification. Since a Justification is supportive of the recommended Proposal, Justification is a subclass SupportiveReview. Reviews can cite other Reviews and Criteria in their explanation. A Review can also cite Context information. Context represents background information that may be relevant to the decision. There are two subclasses of Context. The Evidence subclass represents observations and experiments that are relevant to the decision process and believed to be facts. The Assumption subclass represents conjectural information that may or may not be the case but which is relevant to the decision process. Context information may include references to published research papers or books.

Reviews can be the result of a collaborative process involving several individuals. Such a process could be cooperative or antagonistic. If the latter, then the collaboration is likely to represent a negotiation process. At first it appears that Collaboration is unconnected with any Reviews or individuals. In fact, there are connections, but they are represented using object properties of the PROV ontology, and so do not appear in Figure 6.

The Decision Rationale Ontology is derived from the DRIM model shown in Figure 4. Most of the classes in the Decision Rationale Ontology have the same (or very close) meaning as the corresponding class in DRIM. The Context, Evidence, Assumption, Objective, Constraint, Function, Goal and Proposal are the same as in DRIM. For more about what these classes mean, see Chapter 8 of (Sriram 2002). The Review class hierarchy was derived from the Justification and Recommendation classes in DRIM. For example, Recommendation has been replaced by the recommendation object property, but the meaning is largely the same. The Decision Rationale Ontology reifies as classes

some characteristics of DRIM that were not classes or were implicit. The negotiates-with relationship is reified as the Collaboration class, which allows the collaboration to be a subclass of prov:Activity. The relationships with Intent were reified so that they could have additional information. The Intent class itself differs only in that Goal is no longer a subclass. This was done to make it easier to integrate the ontology with decision making ontologies such as KIDS. The Designer class of DRIM is represented with the prov:Agent class. The versions-of and is-alternative-to object properties are represented with the prov:wasDerivedFrom, prov:alternateOf, and prov:specializationOf object properties of PROV, although the meanings are somewhat different. The Plan, Artifact and Physical Object classes of DRIM were not included because they deal with the subsequent implementation of the decision, which is important but out of scope to the decision rationale.

There are several steps in formulating any explanation about a system. As noted above, an explanation is the answer to the question “Why?” possibly also including answers to follow-up questions. The goal that is being achieved by the decision rationale should be explained sufficiently so that one can find the decision rationales that are relevant to the question by using search techniques. The explanations associated with each entity in a decision rationale may be used to answer questions about the decision rationale. The justification of a decision rationale explains why the decision proposal was selected (i.e., the answer to a question about why the recommended proposal was chosen). For the running example of the NTF decision making process, the explanation for why the component was either put back in the warehouse or thrown away is in the Justification of the final recommendation of the process. Other reviews explain why alternative proposals were not selected (i.e., the answer to a counter-factual question about why another proposal was not chosen). In the NTF example, one might ask why further testing was not performed. This would be especially important if the component was very expensive. The criteria that constrain the potential proposals explain why other possibilities were not considered (i.e., the answer to

contrastive questions about why another decision was not considered). In the NTF example, one might ask why the customer who returned the component was not contacted to determine more information about why the component was thought to be faulty. The explanation is simply that the goal was only to determine whether the component was faulty, not why it was returned. The dependencies among decision rationales allow for follow-up questions that explore decisions in more depth. In the NTF example, one might inquire about the reason for the goal or why the tests were being performed in the particular order and not some other order. These are concerned with the design of the process rather than the process steps. The design was the result of its own decision making process and rationale. An example of how one can optimize the order of the steps in the NTF decision making process is developed in Section II of (Baclawski et al, 2018).

6 Conclusion

We have shown that decision rationales are an effective basis for explaining some features of a system. Specifically, we have shown how decision rationales can be used to answer all of the main kinds of explanation questions for decisions: direct questions, counter-factual questions, contrastive questions, and followup questions. We also discussed how decision rationales can be developed, and presented a reference ontology for decision rationales. Having explored the concept of the decision rationale, we propose that they could be a significant contributor to explainability.

Acknowledgments

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

References

Baclawski, K. (2016). The KIDS Ontology version 2.0. Retrieved from http://bit.ly/2xZuTNJ

Baclawski, K. (Ed.). (2019). Ontology Summit 2019: Explanations. Retrieved from http://bit.ly/2z0JGY4

Baclawski, K. (2020). Rationale ontology. Retrieved from https://bit.ly/3eg4HQO

Baclawski, K., Bennett, M., Berg-Cross, G., Fritzsche, D., Sharma, R., Singer, J., . . . Whitten, D. (2020). Ontology Summit 2019 Communiqué: Explanation. Applied Ontology. DOI: 10.3233/AO-200226

Baclawski, K., Chan, E., Gawlick, D., Ghoneimy, A., Gross, K., Liu, Z., & Zhang, X. (2017). Framework for ontology-driven decision making. Applied Ontology, 12 (3-4), 245–273. Retrieved from https://bit.ly/2LYPszt

Baclawski, K., Chystiakova, A., Gross, K., Gawlick, D., Ghoneimy, A., & Liu, Z. (2019, April). Use cases for machine-based situation awareness evaluation. In IEEE Conference on Cognitive and Computational Aspects of Situation Management.

Baclawski, K., Malczewski, M., Kokar, M., Letkowski, J., & Matheus, C. (2002, November 4). Formalization of situation awareness. In Eleventh OOPSLA Workshop on Behavioral Semantics (pp. 1–15). Seattle, WA.

Baclawski, K., Malczewski, M., Kokar, M., Letkowski, J., & Matheus, C. (2006). The Situation Theory Ontology. Retrieved from http://bit.ly/1yrikQj

Baker, C., Al-Manir, M., Brenas, J., Zinszer, K., & Shaban-Nejad, A. (2020).

Applied Ontologies for Global Health Surveillance and Pandemic Intelligence. J. Wash. Acad. Sci.

Bennett, M. (2020). Financial Industry Explanations. J. Wash. Acad. Sci.

Berg-Cross, G. (2020). Commonsense and Explanation: Synergy and Challenges in the Era of Deep Learning Systems. J. Wash. Acad. Sci..

Berman, D., & Hafner, C. (1993). Representing teleological structure in case-based legal reasoning: The missing link. In Fourth Int. Conf. On Artificial Intelligence and Law (pp. 50–59).

Big Trouble with “No Trouble Found” Returns: Confronting the High Cost of Customer Returns. (2016). Retrieved from http://bit.ly/2vO5QZD

Clancey, W. (2019, February). Explainable AI Past, Present, and Future: A Scientific Modeling Approach. Retrieved from http://bit.ly/2Scjvo6

The Compendium Website. (n.d.). Retrieved from http://bit.ly/3avsavd

The designVUE Website. (n.d.). Retrieved from http://bit.ly/2yFGpA3

Duggar, V., & Baclawski, K. (2007, November 5). Integration of decision analysis in process life-cycle models. In International Workshop on Living with Uncertainties. Atlanta, Georgia, USA.

Gleick, J. (1996, December 1). A bug and a crash: Sometimes a bug is more than a nuisance. New York Times Magazine.

Greenhouse, L. (27, 2008, June). Justices, ruling 5-4, endorse personal right to own gun. In The new york times. Retrieved from https://nyti.ms/3giVER3

Hamilton, A., Madison, J., & Jay, J. (1787). The Federalist: a Collection of Essays, Written in Favour of the New Constitution, as Agreed upon by the Federal Convention, September 17, 1787, in two volumes (1st ed.). New York: J. and A. McLean.

How the NRA Rewrote the Second Amendment - Brennan Center for Justice. (2018). Retrieved from https://bit.ly/2zwypCq

Lebo, T., Sahoo, S., & McGuinness, D. (Eds.). (2013, April 30).

PROV Ontology (PROV-O). Retrieved from http://bit.ly/2xPcx2k

Lions, J. (1996). ARIANE 5 Flight 501 Failure: Report by the Inquiry Board. Retrieved from https://bit.ly/2ZzUoDb

Loui, R., & Norman, J. (1995). Rationales and argument moves. Artif. Intell. Law , 3 , 159–189. Retrieved from https://doi.org/10.1007/BF00872529

Musen, M. (2015, June). The Protégé project: A look back and a look forward. In AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence (Vol. 1). Retrieved from https://bit.ly/2zwqBQY

O'Boyle, C. (1998). The art of medicine: medical teaching at the University of Paris, 1250-1400. Leiden: Brill.

Protégé website. (2004). Retrieved from http://bit.ly/AASA

The Rationale ® Website. (n.d.). Retrieved from https://bit.ly/2TKLgIa

Schmidt, D. (1999). Why software reuse has failed and how to make it work for you (Tech. Rep.). Nashville, Tennessee: Vanderbilt University. Retrieved from https://bit.ly/2M1vGUc

The SEURAT Website. (n.d.). Retrieved from http://bit.ly/2VPfLg3

Spacey, J. (2016). What is a design rationale? Retrieved from https://bit.ly/3c5vf5V

Srihari, S. (2020). Explainable Artificial Intelligence: An Overview. J. Wash. Acad. Sci.

Sriram, R. (2002). Distributed and Integrated Collaborative Engineering Design. Glenwood, MD 21738: Sarven Publishers. ISBN 0-9725064-0-3

Sullivan, K. (1999). Software design as an investment activity: A real options perspective. Real Options and Business Strategy: Applications to Decision Making, 215–261.

van Gelder, T. (2007). The rationale for Rationale ®. Law, Probability and Risk, 6, 23–42. Retrieved from doi:10.1093/lpr/mgm032

What Is Software Reuse? (2014). Lombard Hill Group. Retrieved from http://www.lombardhill.com