

# A Computational Model of Argumentative Design Rationale

Yoshikiyo Kato\* and Koichi Hori†

## Abstract

Although the importance of recording design rationale has been recognized, it has not been widely accepted in practice. Major reasons hindering the acceptance of such practice are: 1) the cost associated with capturing design rationale, 2) failure to perceive the direct benefits of describing design rationale, and 3) the difficulty of utilizing captured design rationale. This study aims to overcome the capture bottleneck of design rationale by employing computable design rationale representation. In this paper, we propose a computational model of design rationale based on a formal argumentation framework, and report an experiment of describing a design rationale in NHL knowledge representation language and applying the argument generation functionality of the legal reasoning system New HELIC-II.

## 1 Introduction

Today's society relies heavily on various large complex systems. In order to design, construct, and operate such systems efficiently and reliably, those who engage in such activities should share their knowledge about the systems. An important aspect that should be incorporated when one is engaged in activities related to large complex systems is *design rationale*.

Design rationale is the theoretical basis or designer's intention behind the design decisions of a system. Not only is it useful for system designers to understand the designs used by others, but it is also important for those who engage in the phases that come after design (i.e., construction, test, and operation) to gain a better understanding of the system they are dealing with.

Although the importance of recording design rationale has been recognized, it has not been widely accepted in practice. Major factors hindering the acceptance of such a practice are: 1) the high cost of recording design rationales, 2) failure to perceive the benefits of recording design rationale, and 3) the difficulty of utilizing recorded design rationale. With regard to the first factor, a tremendous cost is reportedly associated

---

\*Institute of Space Technology and Aeronautics, Japan Aerospace Exploration Agency, 2-1-1 Sengen, Tsukuba, Ibaraki 305-8505 JAPAN, email: kato.yoshikiyo@jaxa.jp

†Research Center for Advanced Science and Technology, The University of Tokyo, email: hori@ai.rcast.u-tokyo.ac.jp

with formally capturing design rationale. [5, 13, 4] This difficulty belongs to the same class of problems as the bottleneck of knowledge acquisition for expert systems, or the capture bottleneck in knowledge management. [10]

The purpose of this study is to overcome the capture bottleneck of design rationale by providing to the user a perceivable benefit for describing design rationale. In particular, we propose a computational model of design rationale based on a formal argumentation framework, and provide reasoning services based on the proposed model. This paper presents the proposed model and reports a preliminary experiment in which a simple design rationale based on the proposed model was written in NHL knowledge representation language [11], to which the argument generation capability of the New HELIC-II legal reasoning system was applied.

## 2 Argumentation Framework

This section presents an overview of the argumentation framework employed in this study. We review the argumentation framework presented in [12], which is also employed by the legal reasoning system New HELIC-II, discussed later in this paper.

A rule takes the following form:

$$\mathbf{n} : p_0 \leftarrow p_1 \wedge \dots \wedge p_n \quad (1)$$

where  $\mathbf{n}$  is the rule name, and each  $p_i$  is a literal. Literals are either in the form of  $q$  or  $\neg q$ , where  $q$  is an atom, and  $\neg$  is a classical negation.

Rule name  $\mathbf{n}$  is a label for  $\mathbf{r}(X_1, \dots, X_n)$ ;  $\mathbf{r}$  is a functional symbol; and  $X_1, \dots, X_n$  indicate variables in a rule. By assigning terms  $t_1, \dots, t_n$  to variables  $X_1, \dots, X_n$ , names for all the rule instances can be obtained.

A preference relation can be defined between rules.  $\mathbf{n}_2 \succ \mathbf{n}_1$  indicates that rule  $\mathbf{n}_2$  is preferred to rule  $\mathbf{n}_1$ . Relation  $\succ$  is transitive and antisymmetric (a strict partial order).

**Definition 1** We say that  $A$  is an argument for  $p$ , if  $A$  is a set of rules such that  $A$  is consistent and  $p \in E(A)$ , where  $E(A)$  denotes the extension of  $A$ , or the set of statements derivable from  $A$ .

**Definition 2** We say that an argument  $B$  for  $q$  is a counterargument to an argument  $A$  iff  $A$  includes a sub-argument  $A_1$  for  $\bar{q}$ , where  $\bar{q}$  denotes the complement of  $q$ .

**Definition 3** An argument  $A$  is defeated iff a sub-argument  $A_i \subset A$  is directly defeated.

**Definition 4** An argument  $A_i$  for  $q$  is directly defeated iff there exists an argument  $B$  for  $\bar{q}$  such that:

1.  $\exists \mathbf{r}_2 : \bar{q} \leftarrow w_1 \wedge w_2 \wedge \dots \wedge w_n$  such that  $\mathbf{r}_2 \in B$  and  $\mathbf{r}_2 \succ \mathbf{r}_1$ , where  $\mathbf{r}_1 : q \leftarrow s$  and  $\mathbf{r}_1 \in A_1$
2. All sub-arguments  $B_i$  of  $B$  ( $B_i \subset B$ ) are justifying arguments.

**Definition 5** An argument  $A$  for  $p$  is plausible iff  $A$  has no defeating counterargument  $B$ .

**Definition 6** An argument  $A$  for  $p$  is justifying iff  $A$  has no plausible counter argument  $B$ .

### 3 Computational Model of Design Rationale

#### 3.1 System Design Evaluation

Design is an iterative process of synthesis, analysis, and evaluation [2]. Most critical decision-making in a design process happens during or after the evaluation phase of each iteration, before moving on to the next iteration. In this respect, it is crucial to consider the evaluative aspect of a design process when one tries to capture design rationale.

For modeling the evaluative aspect of a design process, we draw upon the concept of system design evaluation in systems engineering. [3] Figure 1 depicts the design process model identified in this study. In this model, *requirements* for the system to be designed and the *system configurations* (or *design options*) corresponding to those requirements are assumed to be given. Analyzing requirements allows the derivation of *design criteria* against which design options are to be evaluated. Design options are not usually directly comparable to design criteria. Instead, *Technical Performance Measures (TPM)* or *Design Dependent Parameters* of each design option, have to be obtained through analysis, simulation, testing, or any other necessary means. Once the relevant design criteria and the TPMs are collected, the design options are evaluated. Based on the results of evaluation, decisions are made regarding what must be done in the next iteration.

#### 3.2 Argumentative Design Rationale Framework

Argumentative Design Rationale Framework (ADRF) represents the design process model described above, in terms of the argumentation framework presented in Section 2.

An argumentative design rationale framework  $\mathcal{F}$  is a tuple,

$$\mathcal{F} = \langle O, E, M, V, DC, J, C \rangle \quad (2)$$

where  $O$ ,  $M$ ,  $DC$ , and  $C$  are sets of literals, while  $E$ ,  $V$ , and  $J$  are sets of rules. Each element  $o_i \in O$  represents a design option.  $M$  consists of literals representing TPMs  $\{m_j\}$  such that  $m_j(o_i, X)$  gives the value  $X$  of the TPM  $m_j$  for the design option  $o_i$ . Elements of  $DC = \{dc_j\}$  are literals representing design criteria such that  $dc_j(o_i, X)$  gives the value  $X$  of the design criteria  $dc_j$  for the design option  $o_i$ .  $C$  is a set of literals  $\{pro(X), \neg pro(X)\}$  representing claims either favoring or rejecting design options.

$E$  is a set of rules representing an *evidence* for a design option taking a particular value in a certain TPM. An evidence rule takes the form of

$$e_i : m_j(o_k, x_l) \quad (3)$$

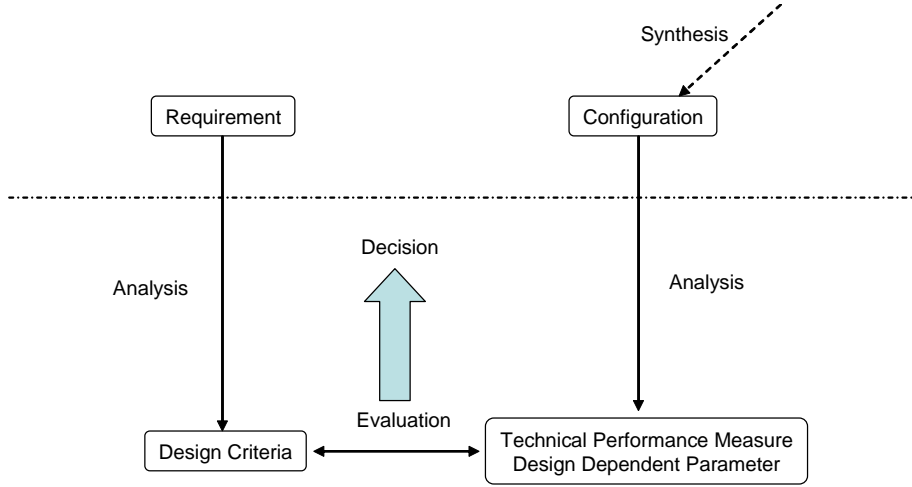


Figure 1: A design process model based on system design evaluation.

meaning there is evidence  $e_i$  that design option  $o_k$  has a value of  $x_l$  for the TPM  $m_j$ .

$V$  is a set of rules representing an *evaluation* for a design option taking a particular value in a certain design criteria.

$$v_i : dc_j(o_k, x_l) \leftarrow f_i(M, O) \quad (4)$$

Here,  $f$  is a formula including TPMs for design options  $O$ . The form of an evaluation rule indicates that the value for a design option is dependent on the values of TPMs.

Finally,  $J$  is a set of rules representing *judgments* on design options based on the value of design criteria. Judgment rules take the following form,

$$j_i : c_j(o_k, x_l) \leftarrow g_i(DC, O) \quad (5)$$

where  $g$  is a formula containing design criteria  $DC$  for design options  $O$ .

## 4 Preliminary Experiment

As a preliminary experiment, we have written a design rationale based on ADRF in NHL (New HELIC-II Language) [11] and applied the argument generation capability of the New HELIC-II legal reasoning system. The purpose of the experiment was to confirm that a design rationale, albeit a simple one, could be represented with the proposed framework, and that reasoning upon it is possible.

### 4.1 Design Rationale Example: Star Tracker Design

We take an example from the design of a star tracker as material for describing design rationale. A star tracker is a device to determine the orientation of a satellite in the

celestial sphere by looking at celestial bodies. A parameter that has to be determined in designing a star tracker is the field-of-view (FOV), the angle in the object space over which objects (celestial bodies in the case of star trackers) are recorded on the imaging sensor.

We introduce two design criteria that are affected by FOV: the identifiability of views and the pointing accuracy of the orientation. Identifiability of the view is how well one can identify the position of a view in the celestial sphere. We adopt as the TPM of identifiability the probability that the star tracker can determine the position in the celestial sphere at which the view was taken, given an arbitrary view. We refer to this TPM as “coverage”. By extracting relative positions of the celestial bodies in the image and comparing them to the catalog it has, the star tracker is able to identify the position of the image in the celestial sphere. The celestial bodies and their relative positions are clues for identifying the view; therefore, as the number of celestial bodies in the image increases, the identification task becomes easier.

As for pointing accuracy, attitude determination error is usually used as its TPM. However, since the final performance in error determination depends on many factors other than FOV, it is not easy to relate FOV directly to the attitude determination error. Instead, for pointing error in this example we use the resolution of the view (i.e., the number of image sensor’s pixels per unit area in the view) as the TPM. The wider the FOV, the lower the resolution; the narrower the FOV, the higher the resolution.

In terms of FOV, the trade-off relation between the two design criteria is clear. Increasing the FOV results in better identifiability but sacrifices pointing accuracy. Decreasing the FOV results in better pointing accuracy but compromises identifiability.

## 4.2 Describing Design Rationale in NHL

NHL is a knowledge representation language for the New HELIC-II legal reasoning system [11]. NHL is based on inheritance-based logic programming language LOGIN, [1] and has additional capabilities such as negation-as-failure (NAF), defeasible reasoning (based on the framework reviewed in Section 2), and case-based reasoning. The knowledge base for New HELIC-II consists of three types of knowledge: types, rules, and partial orders.

Figures 2 and 3 present the design rationale of the star tracker design discussed in the previous section, described in NHL. Figure 2 presents the rule definition describing the design rationale for a star tracker design. Rules `analysis_1` through `analysis_4` are evidence rules providing TPM values for each design option. Rules `design_criteria_1` and `design_criteria_2` are evaluation rules giving values for design criteria, `coverage_requirement` and `accuracy_requirement`.

Figure 3 defines preference relations between units, which are sets of rules. Five units and two standpoints are defined. Standpoints indicate which design criteria have more importance. Standpoint `weigh_coverage` defines preference relations so that rules concerning `coverage_requirement` are preferred, while in `weigh_accuracy`, preferences are set so that the rules concerning `accuracy_requirement` are preferred.

```

&define_RBR_rule.
{
% Judgements
decision_1:: select(object=X/stt)
  <- satisfy(agent=X, object=coverage_requirement),
    satisfy(agent=X, object=accuracy_requirement).
decision_2:: select(object=X/stt)
  <- satisfy(agent=X, object=coverage_requirement).
decision_3:: select(object=X/stt)
  <- satisfy(agent=X, object=accuracy_requirement).
decision_4:: -select(object=X/stt)
  <- not satisfy(agent=X, object=coverage_requirement).
decision_5:: -select(object=X/stt)
  <- not satisfy(agent=X, object=accuracy_requirement).

% Design Criteria
design_criteria_1:: satisfy(agent=X, object=coverage_requirement)
  <- derive(agent=X/configuration, object=Y/stt_tpm[coverage=>[98..100]]).

design_criteria_2:: satisfy(agent=X, object=accuracy_requirement)
  <- derive(agent=X/configuration, object=Y/stt_tpm[resolution=>[80..9999]]).

% Analysis (Evidences)
analysis_1:: derive(agent=c1, object=stt_tpm[coverage=>80, resolution=>200]).
analysis_2:: derive(agent=c2, object=stt_tpm[coverage=>90, resolution=>100]).
analysis_3:: derive(agent=c3, object=stt_tpm[coverage=>99, resolution=>67]).
analysis_4:: derive(agent=c4, object=stt_tpm[coverage=>98, resolution=>80]).
}

```

Figure 2: Rule definition for design rationale.

```

% Definition of units
&define_unit.
satisfy_both := {decision_1}.
satisfy_coverage := {decision_2}.
satisfy_accuracy := {decision_3}.
not_satisfy_coverage := {decision_4}.
not_satisfy_accuracy := {decision_5}.

% Definition of standpoints
&define_standpoint.
weigh_coverage := {
  not_satisfy_accuracy < satisfy_coverage,
  satisfy_accuracy < not_satisfy_coverage
}.
weigh_accuracy := {
  satisfy_coverage < not_satisfy_accuracy,
  not_satisfy_coverage < satisfy_accuracy
}.

```

Figure 3: Definition of partial order relations for rule preferences.

No	X	Top Rule	No Standpoint	weigh_coverage	weigh_accuracy
A1	c1	decision_3	P	D	J
A2	c2	decision_3	P	D	J
A3	c3	decision_2	P	J	D
A4	c4	decision_1	J	J	J
A5	c4	decision_2	J	J	J
A6	c4	decision_3	J	J	J

Table 1: The arguments with the consequence of `select(object=X)`, and their evaluation for different standpoints, generated by New HELIC-II from the rules defined in Figures 2 and 3. P, D, and J indicate that the argument is either plausible, defeated, or justified.

### 4.3 Experiment: Argument Generation

We applied the argument generation capability of New HELIC-II to the rule set described in the previous section. We had the system generate arguments with the consequence of `select(object=X)`, and the evaluation of each argument (i.e., *plausible*, *defeated*, or *justified*) described in Section 2.

Table 1 presents the results. X indicates which design option the argument supports. Top Rule is the rule that directly derives the consequence `select(object=X)`. The remaining columns reflect the evaluation of the arguments according to the standpoints. The column entitled “No Standpoint” shows the results when no standpoints are applied (i.e., no preferences are imposed between any rules). The columns “weigh\_coverage” and “weight\_accuracy” are the results when standpoints defined as in Fig. 3 are applied.

Before discussing the results, first we look at the characteristics of each design option in terms of TPMs and design criteria. c1 and c2 score *satisfy* for accuracy design criteria, while failing in coverage design criteria. c3 gets *satisfy* for coverage design criteria, although it fails in accuracy design criteria. c4 gets *satisfy* for both design criteria.

These characteristics are apparent in Table 1. In all cases, arguments supporting c4 are always valued *justified*. There is no counterargument for them, as counterarguments could be valid only when the design option does not qualify for either of the design criteria, where either rule `decision_4` or `decision_5` becomes valid. In the case of “weigh\_coverage,” arguments for c1 (A1) and c2 (A2) are defeated because they do not qualify for the coverage design criteria, while the argument for c3 (A3) is justified even though it does not *satisfy* the accuracy design criteria. The opposite results are obtained in the case of “weight\_accuracy.”

## 5 Discussion and Related Work

The example presented in this paper is rather simple, and we have to admit that the scalability of the proposed framework is open to question. The difficulties in applying the framework to real-world problems will probably arise from the organization and

maintenance of those models which consist of the argumentation structure; i.e. representation of design options, TPMs, design criteria. For the rules to be universal, it is desirable to have generic representation of those models, i.e. allowing variables or composite structures. However, aiming at the full-blown formal system from the beginning is not feasible, nor is it our primary interest. Instead, our purpose is to demonstrate the feasibility of representing design rationale in formal argumentation framework, and to support design reasoning. With this in mind, our plan is to represent design options as simple propositions, instead of formula with predicate symbols and variables.

There are several semi-formal argumentative design rationale representations proposed and studied in the literature. IBIS was proposed by Kunz and Rittel [8] as an argumentative problem solving model for decision-making in a complex problem setting with multiple stakeholders involved. Various software tools have been developed based on IBIS: gIBIS for capturing design rationale [5, 6] or HERMES for supporting decision-making [7]. QOC was proposed by MacLean et al. [9] as a semiformal notation for design rationale. It was introduced to be used in a style of analysis called *Design Space Analysis*, in which one considers possible alternative designs of an artifact and understands the artifact in terms of its relationship to the alternative designs. The proposed framework differs from these representations in that it assumes a design process model. It is retrofitted to the engineering design, and provides a context for the user, which we believe is an important aspect for the usability of the method.

## 6 Conclusions

In this paper, we have proposed a computational model of design rationale based on a formal argumentation framework. We have implemented a design rationale of a star tracker design with the legal reasoning system New HELIC-II. We also demonstrated that the proposed model is capable of representing a trade-off relation, and that arguments for design decisions can be evaluated according to the standpoints.

We obtained promising results in terms of feasibility of computationally modeling design rationale and the ability to provide reasoning services. However, our initial motivation was to tackle the capture bottleneck problem, and by no means can we expect the engineers to write a rule set like the one we have presented in this paper. Thus, the next step for this study will be to develop a user interface for helping engineers or designers construct models without manually writing the rules.

## Acknowledgments

This study is partly supported by Grant-in-Aid for Young Scientists (B) No.16700158 from the Japanese Ministry of Education, Culture, Sports, Science and Technology.

## References

- [1] H. Ait-Kaci and R. Nasr. LOGIN: A logic programming language with built-in inheritance. *Journal of Logic Programming*, 3:185–215, 1986.

- [2] M. Asimow. *Introduction to Design*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1962.
- [3] B. S. Blanchard and W. J. Fabrycky. *Systems Engineering and Analysis*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 1998.
- [4] S. Buckingham Shum and N. Hammond. Argumentation-based design rationale – what use at what cost. *International Journal of Human-Computer Studies*, 40(4):603–652, 1994.
- [5] E. J. Conklin and K. B. Yakemovic. A process-oriented approach to design rationale. *Human-Computer Interaction*, 6:357–391, 1991.
- [6] J. Conklin and M. L. Begeman. gIBIS: A tool for all reasons. *Journal of the American Society for Information Science*, pages 200–213, May 1989.
- [7] N. Karacapilidis and D. Papadias. Computer supported argumentation and collaborative decision making: the HERMES system. *Information Systems*, 26(4):259–277, 2001.
- [8] W. Kunz and H. W. J. Rittel. Issues as elements of information systems. Technical Report S-78-2, Institut für Grundlagen Der Planung I.A, Universität Stuttgart, 1970.
- [9] A. MacLean, R. M. Young, V. M. E. Bellotti, and T. P. Moran. Questions, Options, and Criteria: Elements of Design Space Analysis. *Human Computer Interaction*, 6:201–250, 1991.
- [10] E. Motta, S. B. Shum, and J. Domingue. Ontology-driven document enrichment: principles, tools and applications. *International Journal of Human Computer Studies*, 52:1071–1109, 2000.
- [11] K. Nitta, M. Shibasaki, T. Sakata, T. Yamaji, H. Ohsaki, S. Tojo, I. Kokubo, and T. Suzuki. Knowledge representation of New HELIC-II. Technical Report TM-1301, Institute for New Generation Computer Technology (ICOT), 1994.
- [12] G. Sartor. A simple computational model for nonmonotonic and adversarial legal reasoning. In *Proceedings of the Fourth International Conference on Artificial Intelligence and Law*, pages 192–201, 1993.
- [13] F. M. Shipman and C. C. Marshall. Formality considered harmful: Experiences, emerging themes, and directions on the use of formal representations in interactive systems. *Computer-Supported Cooperative Work*, 8(4):333–352, 1999.