An Integration Framework for Advanced Knowledge-based Design Supports – A Viewpoint of DRIFT Paradigm –

Yutaka Nomaguchi* and Kikuo Fujita‡
Osaka University

Abstract

This paper discusses possibilities and promises of an advanced knowledge-based design support system based on a framework called DRIFT (Design Rationale Integration Framework of Three layers), which captures reflective design process as a byproduct of inherent design actions. DRIFT is an integration framework of three aspects of design process, i.e., action level, model operation level and argument level. Action level represents a sequence of design operations. Model operation level represents a state transition of design states, each of which records a design snapshot as a set of labels defined over the ontology for conceptually representing an artifact. A truth maintenance system (TMS) is adopted to manage model operation level across the design process. Argument level represents a process of setting problems and alternatives, which is captured by Issue-based Information System (IBIS). The linkages among three levels in principle enable to automatically capture and manage the whole design process through tracking design operations over a design support system. This paper introduces an example of DRIFT-based implementation to show the power of the framework, and then discusses possibilities and promises of an advanced knowledge-based design support system.

Keywords: Design knowledge, design process, design rationale, design support, ontology, reflection-in-action.

1 Introduction

Engineering design is the process for generating a concept of useful artifact or product based on various kinds of scientific and engineering knowledge. In other words, an artifact or product is a shape of knowledge integration, and the design process is for creating concrete knowledge as a tangible entity. As complexity of a product increases under progress of science and technology, advances in welfare, globalization of manufacturing industry, etc., the overall volume of knowledge that is operated and created in the design process exceeds the personal or organizational capability of memory and management. Schön [1982] stated that it is insufficient only to apply the already-systematized knowledge to solve a complex design problem but is important to dynamically, flexibly and adaptively acquire knowledge through reflection-in-action, which is trial-and-error design process that includes framing a problem, suggesting multiple alternatives, evaluating what-if and accepting or rejecting. As the latter knowledge is vital in design, a design support system should capture those reflective aspects of the design process.

We have been developing a framework for an advanced knowledge-based design support system, called DRIFT (Design Rationale Integration Framework of Three layers), which can capture reflective design process as a byproduct of inherent design actions [Nomaguchi et al. 2004, 2006, 2007].

*e-mail: noma@mech.eng.osaka-u.ac.jp
‡e-mail: fujita@mech.eng.osaka-u.ac.jp

A design support system based on DRIFT can capture reflective design process performed over design tools with minimal overhead and with the least interference with natural progression of design activities, and facilitate a designer to do what-if analysis that would be critical to reflection-in-action. This paper introduces an example of a DRIFT-based implementation to show the possibilities and promises of the framework, and then discusses future works toward an advanced knowledge-based design support system that captures reflective aspects of design process more comprehensively.

2 Background

The increasing demand for rationalizing design activities launches a wide branch of engineering design research. Knowledge-based approaches have been initiated in 1980s under the paradigm of expert systems. CATIA [Dassault Inc.,], is a commercial application of expert system paradigm to an integrated CAD system. CATIA is a high-end 3D CAD with a knowledge base that stores design rules and design constraints for semi-automatic geometric modeling. Although there is a controversy about effects and limitations of expert system paradigm, it can be said that it has achieved a constant result such as a knowledge-based CAD.

A current research challenge of knowledge-based approach beyond the expert system paradigm is to handle complicated design knowledge and related information more sophisticatedly and flexibly. It was a remarkable step that the capability of knowledge processing has expanded a computational support to upper streams of design process. Today, ontology-based knowledge processing is getting much attention for enhancing the capability of product data management systems, etc. The concept of ontology is to represent knowledge over shared meta-data and to manipulate its contextual structure under the standpoint of computer engineering and apart from design engineering. For instance, Kitamura has been developed a meta-data schema for systematically representing functionality of a product based on Semantic Web technology for the management of the information content of engineering design documents [Kitamura et al. 2006]. Ontology is originally a term of philosophy that means theory of existence. From artificial intelligence (AI) point of view, an ontology is defined as “explicit specification of conceptualization” [Gruber 1993]. As this definition is widely accepted in the fields of AI and informatics, this research also uses the term in this sense.

Another stream of knowledge-based approaches in the design engineering field is a design-for-X (DFX) methodology. A purpose of a design methodology is to have prescriptions that advocate how design should be done in particular circumstances [Dixon 1987]. Many researchers of engineering design have proposed various DFX methodologies. Among various DFX methodologies, QFD (Quality Function Deployment) is a typical and comprehensive one [Clausing 1994]. It is effective for exploring and defining design requirements by a sequential procedure, for instance, across customer’s requirements, functional realization, manufacturing modules, production process, etc. By means of its reflective refinement, a designer

gets overall image of a product. While such an instruction is easy to understand for designers, it consists of some general and abstract concepts by excluding its case-dependent aspects. Therefore, it is suggested that QFD promotes more efficient and effective information sharing and discussion among designers in practical use [Ohfuji 1990]. Although a DFX methodology is not a theory established by a scientific law such as physics, it is often used in actual design cases as a rational prescription of design contexts.

While paradigms of ontology-based knowledge processing and DFX methodologies have been developed separately, each takes a complementary role for an advanced knowledge-based design support system. Both techniques deal with different aspects of design contexts as discussed in the above. An approach based on a DFX methodology can support a designer by means of prescription of superior design. An information system based on ontology can manipulate large-scaled and multi-disciplinary knowledge. Therefore, the integration of both techniques is a probable framework toward context-rich computer-supported design environments and it is expected to be a base for implementing a design support system that has high intensiveness of various aspects of design knowledge.

3 DRIFT Paradigm

While a DFX methodology gives a prescription of rational design and an ontology gives a framework to handle various aspects of knowledge, design process is inevitably iterative and reflective process due to the open-ended nature of design problems. In order to solve such a complex design problem, Schön [1982] stated that it is insufficient only to apply the already-systematized knowledge but is important to dynamically, flexibly and adaptively acquire knowledge through reflection-in-action, which is trial-and-error design process that includes framing a problem, suggesting multiple alternatives, evaluating what-if and accepting or rejecting. As the latter knowledge is vital in design, a design support system should capture those reflective aspects of the design process.

DRIFT is a software module that dynamically captures such reflective design process as a by-product of design. DRIFT facilitates a designer to compare multiple alternatives concurrently during design process, and to review a rejected design alternative. Figure 1 shows the outline of the situation that a designer is involved under a DRIFT system. DRIFT consists of two subsystems; a subsystem that captures a state transition of design information and a subsystem that captures an argumentation structure. The former subsystem adopts a simple mechanism based on a truth maintenance system (TMS) [Doyle 1979] to capture all design states and to track each of them anytime without redundancy and incorrectness. The latter subsystem adopts an IBIS-based model [Conklin et al. 1988] to represent argumentation structure.

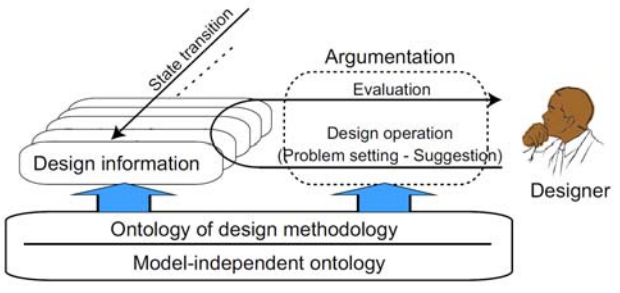

Figure 1. Outline of reflective design process supported by DRIFT framework

Under the mechanism of DRIFT, each design operation is defined as a pattern of problem setting and alternative solution in order to capture both design state transition and an argumentation structure through an inherent design action. For example, a design operation that details a customer need of a product is defined as follows; a problem set by the operation is 'what are sublevel customer needs of it?' and its alternative solution is a set of its sublevel customer needs. Since the system records all alternatives suggested as solutions of the problem, a designer can compare them and review one at any time. An ontology is the meta-data structure that enables capturing design state transition and argumentation structures almost automatically through designer's inherent design actions. While the system includes a set of fundamental ontologies for implementing basic functionality, another set of subsidiary ontologies for capturing and managing practical and complicated design operations is needed. Thus it is necessary and essential that a set of specific ontologies must be formulated for supporting a particular type of design operations in order to apply a design methodology under the corresponding context.

As a foundation of DRIFT, an ontology should be formulated for defining two concepts; taxonomy of design to manage complicated design information through design process, and patterns of problem setting to provide design operations. Both of them are embedded in DFX methodologies. A methodology has taxonomy of design information by necessity, and has a pattern of design operation. This is why this research stated that an ontology extracted from a DFX methodology is valuable for an advanced knowledge-based design support system. For example, QFD is a series of two-dimensional tables between system decompositions in different aspect for mapping and arranging their consistency and soundness toward product integrity. This viewpoint implies the following concepts for product representation and design operations. A product is represented in different views. It has a top node and is hierarchically and recursively decomposed into sub-nodes in each view. Associated nodes are linked each other across difference views. Each node or linkage has attributes characterized it. Each attribute has a value.

QFD operations are syntactically composed of definition of the top node, its decomposition into subsidiary nodes, assigning level of importance to each subsidiary node, assigning level of contribution of a node in a view to another node in another associated view, evaluating mapping of levels of importance across a series of views, and overall procedure and its outcomes are reflectively refined for accommodating their mutual compromise. Contextual structure of these all can be a base of building ontology for supporting a specific design activity under the knowledge-based design support system.

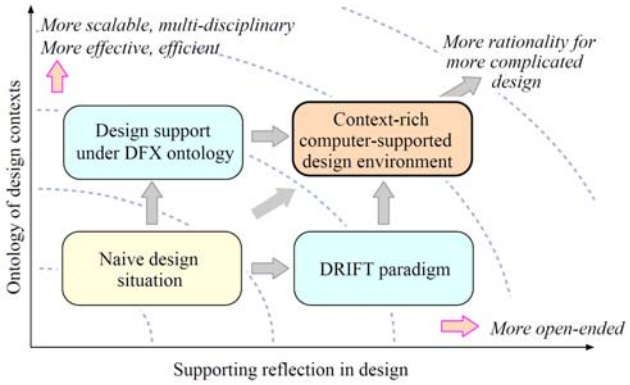

Figure 2. More rational design through an advanced knowledge-based design support system based on DFX ontology and DRIFT

Figure 2 briefs the above mentioned roles of a DFX ontology and DRIFT toward an advanced knowledge-based design support system. A DFX ontology enhances scalability and multi-disciplinary of design contexts. DRIFT paradigm enhances capability of supporting reflection in design in order to handle more open-ended design problems. By integrating the two axes, a more context-rich computer-supported design environment that supports more rationality for more complicated design could be implemented.

4 Implementation of DRIFT

4.1 Implementation Architecture

In order to demonstrate the validity and promise of DRIFT, a prototype design support system under a QFD-based cost-worth analysis method [Fujita et al. 2001] was developed in Java programming language (jdk 1.4.1) on Windows XP. Figure 3 shows the architecture of the system.

Under the integrated system, a designer carries out design by using tools of design methodologies; value graph, function-structure mapping, QFD two-dimensional tables and cost-worth graph. The system sends design operations to TMS when a designer input design information on the tools. Design process is automatically recorded in three levels; action level, model operation level and argumentation level along designer's actions and operations over the embedded tools. A designer can edit description of an argumentation structure. A recorded design process is stored in database in XML format.

The following subsections briefs this implementation. The detail of DRIFT and the prototype system is explained in our articles [Nomaguchi et al. 2004, 2006, 2007].

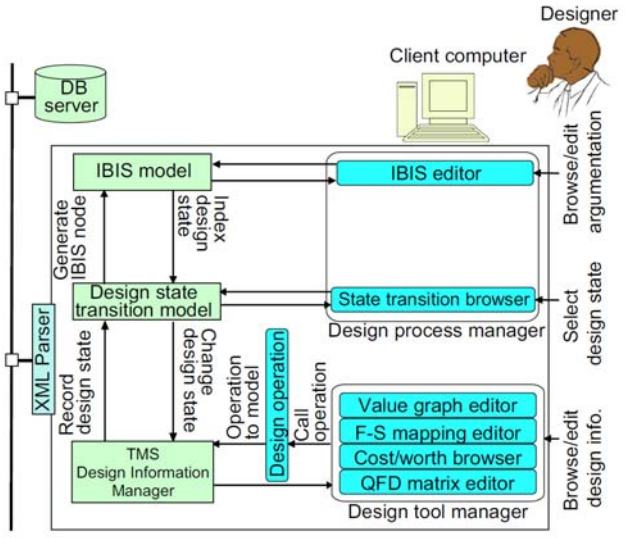

Figure 3. Architecture of DRIFT-based system

4.2 Ontology Definition

Before implementation of a DRIFT-based system, an ontology corresponding to a set of DFX methodologies is configured as a base of a knowledge-based design support system. An ontology consists of two layers; *model-independent layer*, which consists of concepts independent from any specific design methodology, and *model-dependent layer*, which consists of concepts to represent specific prescriptive design methodologies.

A concept of the latter layer is defined as a subclass of model-independent layer concepts. The combination of the above two layers makes an ontology more expansible. When a new design methodology is integrated to the design support system, arrangement of new concepts is required only in the model-dependent layer.

4.3 Model-independent Concepts

The following five concepts are defined as model-independent concepts. Figure 4 shows the definition in UML (Unified Modeling Language) format.

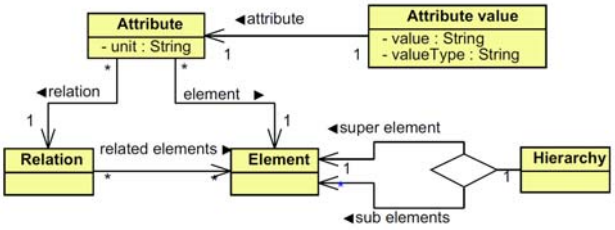

Figure 4. Model-independent concepts

Element is the concept that constructs a product. This is further categorized into three concepts; customer need, function and entity.

Hierarchy is the concept that represents a hierarchical relationship between elements. A hierarchy node has an association to an element, which is a super level node of the hierarchy, and elements, which are sub level nodes of the hierarchy.

Relation is the concept that represents a relation between elements.

Attribute is the concept that represents a character of an element or a relation. An attribute node has an association to an element or a relation.

Attribute value is the concept that represents a value of an attribute. An attribute value node has a value and a value type as property.

4.4 Model-dependent Concepts

A concept for representing specific methodology is defined as a subclass of a model-independent concept. There is a possibility that other concepts are defined in addition to concepts explained in this subsection when the other methodology is integrated.

Value graph describes development of a top customer need 'good product' into sublevel customer needs. Function-structure mapping describes development of functions and entities of a product, and relationships between them. In order to integrate them in DRIFT, three concepts; customer need, function and entity, are defined as sublevel concepts of element.

QFD two-dimensional table describes correlation factors between different aspects such as ones between customer needs and functions, ones between functions and entities, etc. These correlation factors are used to deploy weights of customer needs to weights of entities by simple matrix calculation. The following additional four concepts are introduced to integrate QFD in the framework.

C-F relation and *F-E relation* are both subclass concept of relation. They are used to represent the existence of correlation between a customer need and a function, and one between a function and an entity.

Weight is a subclass concept of attribute. It is used to represent a weight of a customer need, a function and an entity.

Relation factor is a subclass concept of attribute. It is used to represent the correlation factor of a C-F relation or an F-E relation.

Cost-worth graph describes a balance between relative worth and relative cost of an entity. The following three additional concepts are defined to integrate cost-worth graph in the framework.

Relative worth is a subclass of attribute. This is calculated by regularization of weights of entities which are calculated by QFD two-dimensional tables.

Cost is a subclass concept of attribute. It is used to represent cost of an entity.

Relative cost is a subclass concept of attribute. It is used to represent relative cost of an entity. Its value is calculated by regularization of cost of entities.

4.5 Patterns of Problem Setting

A design process under the implemented methodologies is, for example, executed by the following steps; making QFD two-dimensional tables of a product, estimating cost of entities, evaluating a balance of cost and worth, and go back to a former step if necessary. These steps are not proceeded straight forward but in iterative way. Therefore, a designer is required to reflectively consider elements, their relations, weights of customer needs, and cost of entities represented in QFD two dimensional tables. Ten patterns of problem setting are extracted for capturing a whole design process in this methodology.

To make QFD two-dimensional tables of a product, a designer should enumerate elements of each view. Value graph and function-structure mapping help this process. In these tools, a designer firstly sets a top element of each view and then he/she details it to sublevel elements. When three views are considered in QFD, the following three problems should be considered; '*what is a top customer need?*', '*what is a top function?*' and '*what is a top entity?*'

When a designer uses value graph and function-structure mapping, he/she enumerates elements by detailing an abstract element to concrete sublevel elements. Detailed sublevel elements would be often detailed to more concrete sublevel elements. When three views are considered in QFD, the following three problems should be considered; '*what are sublevel customer needs?*', '*what are sublevel functions?*' and '*what are sublevel entities?*'

A designer sets weights of customer needs by considering what kind of customers would be a target of a product. For this operation, the problem '*how much is a customer need important?*' can be considered. The weight is usually marked by three grades, such as 1, 3, and 9. When customers who care value-adds of cellular phone are target, for example, the following mark can be an alternative solution; 'good quality of call: 3, fascinating appearance: 3, many value-adds: 9, high reliability: 1.'

By QFD two-dimensional tables, a designer considers relationships between elements and relation degrees. Since we consider three views, there are the following two problem settings; '*which functions are related to a customer need?*' and '*which entities are related to a function?*'

A designer estimates cost of entity. For this operation, the problem '*how much is cost of entity?*' can be considered.

5 Demonstration on Example Implementation

This section briefly illustrates an application of the implemented prototype to cellular phone design for demonstrating its capabilities and promises. In this example, it is assumed that a product is aimed to Japanese market and that it is equipped with several value-adds such as camera, music player, electric money, bar-code scanner, etc.

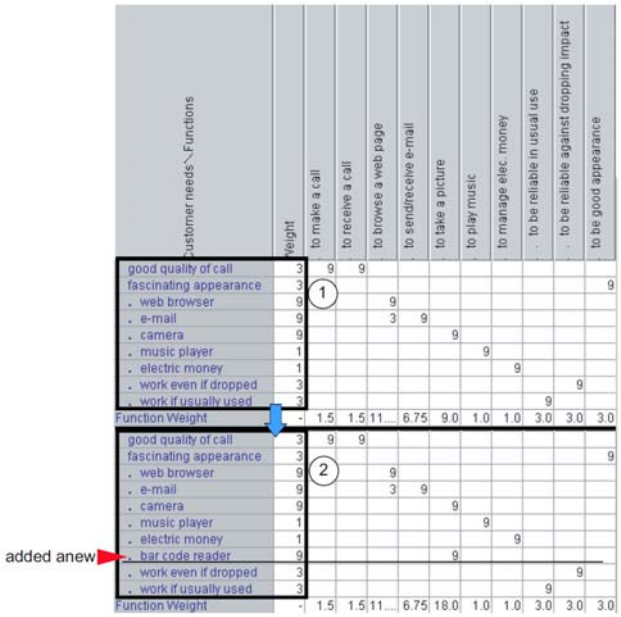

Figure 5. Alternatives of detailing customer needs

First, a designer enumerates customer needs, functions and entities of a cellular phone by using value graph and function-structure mapping. Then, he/she uses QFD matrix to set weights of customer needs as shown in Part 1 of Figure 5. The weights are deployed to weights of entities by automatic calculation of QFD matrix. A designer also sets cost of each entity, and evaluates balance of relative cost and relative worth of respective entities over the cost-worth graph, which is shown in Part 1 of Figure 6. This graph reveals that cost of camera lens would be higher than its worth. In order to dissolve this unbalance of cost and worth, it is recommended to a designer to choose either from among two options, (A) reducing relative cost or (B) increasing relative worth as shown in Part 1 of Figure 6. Part 2 of Figure 5 shows an alternative of detailing customer needs in order to increase relative worth of a camera lens. A new customer need, 'barcode reader,' is added. A relative worth of a camera lens is increased by this alternative as shown in Part 2 of Figure 6.

The design process of the above illustrative exercise is automatically captured as a byproduct of a sequence of design operations over the system. Figure 7 shows a part of the captured design process in argumentation level. An issue node (a node with a question mark) and a position nodes\ (a node with an exclamation mark) are automatically generated. A crossed node is a rejected position. An argument node (a node with a words balloon mark) is added manually by a designer to explain the branch of position nodes. Figure 7 shows that two positions are suggested for an issue of detailing a customer need for many value-adds, and that a position that adopts bar-code reader is active now. Since all operations and all design states are recorded under the TMS mechanism, a designer can review discarded positions at any time for evaluating alternatives if he/she wants to review any of them again, and he/she can go back to any former design state.

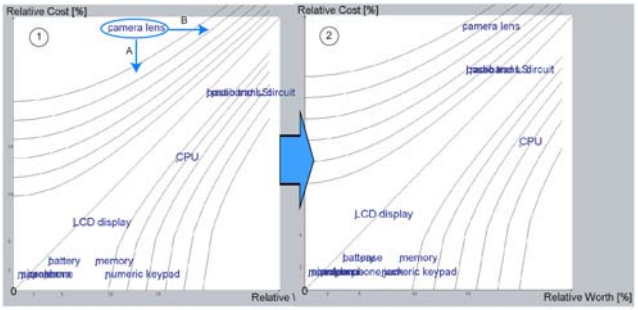

Figure 6. Revision of cost-worth balance

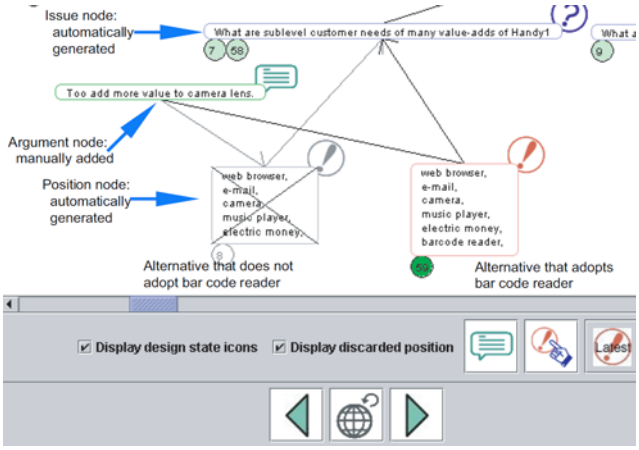

Figure 7. A part of captured design process

6 Discussion and Future Works

6.1 Capability of DRIFT

Through several experiments on the case studies with the example problem of Section 5, it is ascertained that the system can automatically and smoothly record designer's design process that includes operations, alternatives and argumentation. Such functionality facilitates designer's reflective refinement of alternatives over the prototype system through making full use of related methodologies.

Behind the DRIFT, a model operation plays the key role under the template of the design operation. Its content is described based on the ontology of design extracted beforehand. In other words, it is impossible for DRIFT to capture design operations beyond the concept provided beforehand by the introduced ontology. The prototype system introduced in Section 4 can capture reflective design process about designing customer's needs, function, entity and its cost-worth balance, but can not capture design process beyond the implemented DFX methodology. Therefore, it is necessary to expand the ontology and the templates of design operations additionally in order to capture design process that is more complex than such an example case demonstrated in Section 5.

6.2 Expanding Ontology

It can be said that it should be a fundamental research challenge to clarify and integrate ontology of a conceptual artifact model and a design operation toward a more advanced knowledge-based design support system. The standpoint of our research is that a DFX methodology should be something related to ontology. Because a purpose of a DFX methodology is to provide prescriptions that advocate how design should be done in particular circumstances, taxonomy of design and a pattern of design operations are tacitly included in a DFX methodology. Their meaning is what an ontology is.

The example in Section 5 shows that the contextual structure of QFD is appropriate as a basis of building ontology for a DRIFTT system. An ontology of value graph, function-structure mapping and cost-worth graph is defined with small modification. This kind of expansions can be expected to be feasible for other DFX methodologies. For example, FMEA (Failure Mode and Effects Analysis) contains taxonomy and design operation about failure mode analysis. Design for assembly, such as Westinghouse method [Sturges 1992], contains taxonomy and designing assembly sequence and part geometry. Because these methodologies assume description of a system structure like one shown in Subsection 4.3, model-dependent concepts of theses methodologies could be defined as subclass concepts of the model-independent concepts. In addition, overall procedure and its outcomes are reflectively refined for accommodating their mutual compromise. DRIFT's functionality is expected to facilitate designer's reflective refinement of alternatives over these methodologies.

In order to effectively and efficiently achieve this expansion, a principle and guideline should be developed for facilitating ontology extraction from DFX methodologies because development of a DFX methodology has been mainly achieved in empirical ways apart from developing computer support tools for design.

Our fundamental idea is that an ontology should be defined in order to support the dynamic aspect of design process that includes reflection-in-action. Schön [1982] stated that a good reflective practitioner has some constants such as the languages that practitioners use to describe reality and conduct experiments, the appreciative systems they bring to problem setting, and the overarching theories by which they make sense of what happens in an artifact. From the viewpoint of promoting reflection-in-action, ontology of an artifact representation should contain these constants. In the case of QFD-based cost-worth analysis method used in the prototype system, for example, the method provides the language to describe an artifact such as customer needs, relative worth, cost, and so on. The method provides the appreciative system of design as a pattern of a design operation. The method provides the overarching theory that relative cost and relative worth of each component of an artifact should be balanced. Of course, this implementation is just an example. A more general and wider set of ontology should be extracted for more flexibly and efficiently integrating various kinds of DFX methodologies and the other design methods. This is an important future work of this research for enhancing the power of DRIFT framework.

6.3 Further Expansions

The same expansion can be considerable either for the other design methods that support downstream of design process, such as design optimization or physical analysis. An optimization method provides mathematical formulation and procedures to find a design solution that minimizes an objective function and satisfies constraints. However, an optimization critically depends on a designer's intention because of several natures of optimization. For example, different design parameters and design variables are emphasized or neglected depending on mathematical formulation such as selection of the objectives and constraints. Therefore, overall procedure of modeling optimization should be reflectively refined.

It is indispensable to evaluate an artifact based on a mathematical model of a physical phenomenon in order to rationalize product

design. Physical phenomena are independent from a designer's intention. However, models of physical phenomena unavoidably depend on a designer's intention. When its focus shifts to designing a large-scaled and complicated system, analysis of an artifact becomes a multi-disciplinary problem. It is necessary to approximate such complicated physical models and selectively integrate them in order to surely analyze it within reasonable time. This modeling process should be also reflectively refined.

While some knowledge-based support systems have been developed to support modeling optimization and modeling physical phenomena [i.e., Witherell et al. 2007, Grosse et al. 2005], they rarely focus on the above reflective nature of modeling process. A DRIFT framework would provide break-through technology either for this direction.

6.4 Challenges beyond DRIFT Paradigm

Capturing design process that is beyond pre-defined ontology is anyhow indispensable. It is necessary to give a designer the freedom flexibly describing knowledge outside the frame of ontology, because some descriptions based on such user's intervention would contain important design rationale. In the prototype system, a designer can manually add the content of an argument node in IBIS. There are a number of research papers that tackles user's-intervention-based capturing. For example, ADD [Garcia et al., 1992] captures design rationale by 'designer's apprentice' mechanism that asks a designer when nonstandard parameter value is inputted. JANUS [Fischer et al., 1996] has a conceptual model of kitchen layout and a knowledge base to critique a construction design. Such a mechanism that supports user's intervention is included in our future works. However, a careful examination whether the promotion of intervention does not become an obstruction of designer's thinking is also indispensable.

This research does not focus on navigation and retrieval of design rationale so far. These are also included in our future works. Because DRIFT captures all actions, their associated model snapshots, and argumentations in design process, captured design rationale would contain a ton of information, a part of which is relatively generic and valuable in a class of design situations, but the other part of which is relatively specific and useless. Therefore, extracting a proper subset of design rationale is another challenge for realizing the power of DRIFT. At least, as DRIFT captures both of an artifact representation and design process representation in well-structured format, they will be valuable context information to retrieve design rationale.

7 Conclusion

This paper discusses possibilities and promises of an advanced knowledge-based design support system based on a DRIFT framework, which is a new framework for capturing reflective design process of practical products. This paper introduces an example of DRIFT-based implementation to show the power of the framework, and then discusses future works and challenges of an advanced knowledge-based design support system. This application demonstrated that the framework can capture dynamic aspect of design process in which a designer frames a problem, suggesting multiple alternatives, evaluating what-if and accepting or rejecting an alternative with the least interference with natural progression of design activities. It will help a designer dynamically, flexibly and adaptively acquire knowledge through reflection-in-action. Because any design stage contains reflective refinement of alternatives, the same expansion can be done for the other DFX methodologies and the other design methods, such as optimization or physical analysis, when the issues stated in Section 6 are resolved.

References

CLAUSING, D. 1994. *Total Quality Development. A Step-By-Step Guide to World-Class Concurrent Engineering*, ASME Press.

CONKLIN, J., AND BEGEMAN, M. L. 1988. gIBIS: A Hypertext Tool for Exploratory Policy Discussion, *ACM Transactions on Office Information Systems*, Vol. 6, No. 4, pp. 303-331.

DASSAULT, INC. 2007. CATIA V5R17 [online], Available from: http://www.catia.com/ [Accessed 13th June 2007].

DIXON, J. R. 1987. On Research Methodology towards a Scientific Theory of Engineering Design, *AI EDAM*, Vol. 1, No. 3, pp. 145-157.

DOYLE, J. 1979. A Truth Maintenance System, *Artificial Intelligence*, Vol. 12, No. 3, pp. 231-272.

FISCHER, G., LEMKE, A. C., MCCALL, R. AND MORCH, A. I. 1996. Making Argumentation Serve Design, *Design Rationale – Concepts, Techniques, and Use* (Moran, P., AND Carroll, J. M., Eds.), Lawrence Erlbaum Associates, Mahwah, New Jersey, USA, pp. 267-293.

FUJITA, K. AND NISHIKAWA, T. 2001. Value-Adds Assessment Method for Product Deployment across Life Stages through Quality Function Deployment, *Proc. 13th International Conf. on Engineering Design — ICED '01, Design Methods for Performance and Sustainability*, pp. 405-412

GARCIA, A. C. B. AND HOWARD, H. C. 1992. Acquiring Design Knowledge through Design Decision Justification, *AI EDAM*, Vol. 6, No. 1, pp. 59-71.

GROSSE, I. R., MILTON-BENOIT, J. M. AND WILEDEN, J. C. 2005. Ontologies for Supporting Engineering Analysis Models, *AI EDAM*, Vol. 19, No. 1, pp. 1-18.

GRUBER, T. 1993. A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, Vol. 5, No. 2, pp. 199-220.

KITAMURA, Y., WASHIO, N., KOJI, Y., SASAJIMA, M., TAKAFUJI, S. AND MIZOGUCHI, R. 2006. An Ontology-based Annotation Framework for Representing the Functionality of Engineering Devices, *Proc. DETC'06 ASME 2006 Design Engineering Technical Conf. and Computers and Information in Engineering Conf*, DETC2006-99131.

SCHÖN, D. A. 1982. *The Reflective Practitioner – How Professionals Think in Action*, Basic Books Inc.

NOMAGUCHI, Y., OHNUMA, A. AND FUJITA, K. 2004. Design Rationale Acquisition in Conceptual Design by Hierarchical Integration of Action, Model and Argumentation, *Proc. DETC'04 ASME 2004 Design Engineering Technical Conf. and Computers and Information in Engineering Conf*, DETC2004/CIE-57681.

NOMAGUCHI, Y., TAGUCHI, T. AND FUJITA, K. 2006. Knowledge Model for Managing Product Variety and its Reflective Design Process, *Proc. DETC'06 ASME 2006 Design Engineering Technical Conf. and Computers and Information in Engineering Conf*, DETC2006-99360

NOMAGUCHI, Y. AND FUJITA, K. 2007. Ontology Building for Design Knowledge Management Systems Based on Patterns Embedded in Design-for-X Methodologies, *Proc. of 16th International Conf. on Engineering Design (ICED '07)*.

OHFUJI, T., ONO, M. AND AKAO, Y. 1990. *Quality Function Deployment*, Union of Japanese Scientists and Engineers. (in Japanese)

STURGES, R. H. AND LILANI, M. 1992. Toward an Integrated Design for an Assembly Evaluation and Reasoning System, *Computer Aided Design*, Vol. 24, No. 2, 1992, pp. 67-79.

WITHERELL, P. KRISHNAMURTY, S. AND GROSSE, I. R. 2007, Ontologies for Supporting Engineering Design Optimization, *Journal of Computing and Information Science in Engineering*, Vol. 7. No. 2, pp. 141-150.