An Ontology-based Validation Approach to Resolve Conflicts in Manufacturing Design Process

Kazunari Hashimoto, Motoyuki Takaai, Yohei Yamane, Seiji Suzuki, Masao Watanabe, and Hiroshi Umemoto

Research and Technology Group, Fuji Xerox Co., Ltd.
6-1 Minatomirai, Nishi-ku Yokohama-shi Kanagawa 220-8668 Japan
{Kazunari.Hashimoto,Motoyuki.Takaai,Yohei.Yamane,
Seiji.Suzuki,Masao.Watanabe,Hiroshi.Umemoto}@fujixerox.co.jp

Abstract. In design process of manufacturing, designers analyze the factor of phenomena and apply solutions to phenomena (such as resizing, changing electrical circuits, and modification of mass). There may be often conflicts based on chain relations among solutions which designers cannot recognize. We focus on validating these solutions. In this paper, we propose an ontology for representing chain relations in design process. To validate designers' solutions, we propose an algorithm to detect conflicts between a given solution and those for other design cases. In our case study, we introduce our approach to product development and demonstrate the efficacy and practicability of our proposed ontology.

Keywords: Validating, Design Process, Design Knowledge, Fault Tree Analysis, Failure Mode and Effect Analysis, SPARQL

1 Introduction

The phases of the manufacturing design process are as follows: *Phase 1* detecting phenomena, *Phase 2* factor analysis of phenomena, *Phase 3* application of solutions based on those factors, and *Phase 4* implementation of solutions. In this paper, phenomena means troubles which occurs in product design, and solution means designers's approach improving their troubles. Designers need to share design knowledge to enhance the efficiency of the design process. To prevent problems arising in design process, Failure Mode and Effects Analysis (FMEA) and Fault Tree Analysis (FTA) are often employed. There are previous studies based on FMEA and/or FTA for managing the design knowledge with semantic web technology which employ several different methods; i.e., capturing and modeling functional knowledge [1], modeling the process of FMEA [2], and constructing an ontology-based FTA system for applying industrial process [3] [4] [5]. Most of these studies are focused on relations between phenomena, and generally do not examine the relationships between phenomena and solutions, or relations between solutions.

Designers' works based on the design process for a product (hereinafter, referred to as design cases), can be used to determine the conflict between solutions of design cases through communications such as documents and meetings. Conflicts mean potential contradictions between solutions such as resizing, changing electrical circuits, and modification of mass and, as mentioned above, designers often cannot recognize conflicts due to chain relations between solutions and components for a particular product. This can mean that the effect of solution is halved, or a solution produces the opposite of its intended effect. Therefore, it is crucial that designers confirm that the applied solution for a given design case does not conflict with the solutions of other design cases.

In this paper, we propose an ontology for representing chain relations in design process. We then propose an algorithm for detecting conflicts between possible solutions by inferring the trace of instance in stage. Finally, we introduce an approach to product development and demonstrate the efficacy and practicability of our proposed ontology.

2 Proposed Approach

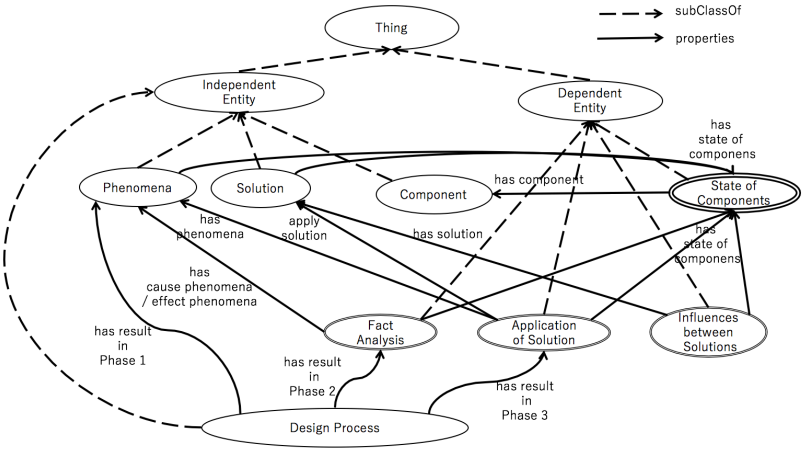

Fig. 1. The simplified version of proposed ontology

2.1 The Architecture of Ontology

There are multiple dependencies on the state of components among phenomena, solution, and components. The state of components indicates the structural relation between components and the functional relationship between components such as power transmission or transmission of movement. Phenomena and solution are represented by these component states. Factor analysis and application of solution are both represented by the state of components.

We propose an ontology that represents these multiple dependencies in the design process. Fig.1 illustrates the simplified version of the proposed ontology. *Independent entity* class represents an independent concept with *phenomena* class, *solution* class, and *component* class as sub classes. *Dependent entity* class represents the dependence between *Independent entity* classes with *design process* class, *state of components* class, *fact analysis* class, *application of solution* class, and *influences between solutions* class as sub classes. *State of components* class represents the dependence between *component* classes. *Fact analysis* class, *application of Solution* class, and *influences between solutions* class represent dependencies among *phenomena* class, *solution* class and *state of components* class. *Design process* class is represented by links to *phenomena* class, *fact analysis* class, and *application of solution* class. *Influences between solutions* class can represent the chain relation between solutions and validate solutions by the following algorithm.

2.2 Validating Solutions

We assume that designers validate the solutions which they apply to phenomena. Let DC be design case, and DK be design knowledge. Let $Comp$ be instances of *component* class, and Sol be instances of *solution* class. The instances of $Comp_c^{DC}$ and Sol_c^{DC} represent components and a solution in the design case c. The instances of $Comp_c^{DK}$ and Sol_c^{DK} represent components and a solution in design knowledge , corresponding to $Comp_c^{DC}$ and Sol_c^{DC} . Via instances of *state of components*, $Comp_c^{DC}$ is linked to Sol_c^{DC} and $Comp_c^{DK}$ is linked to Sol_c^{DK}. The application of solution means linking Sol_c^{DC} to Sol_c^{DK} and linking $Comp_c^{DC}$ to $Comp_c^{DK}$.

Designers must validate that the solution for the current case does not conflict with the solutions for existing cases. Therefore, we propose the following algorithm for detecting conflicts between solutions:

For case cA and case cB, an algorithm for detecting the conflict between Sol_{cA}^{DP} and Sol_{cB}^{DP} is as follows:

*Step*1 Inferring the solution trace $SolTrace_{cA-cB}^{DK}$ from Sol_{cA}^{DK} to Sol_{cB}^{DK}. $SolTrace_{cA-cB}^{DK}$ is composed of instances of *solution* class and *influences between solutions* class. It is sufficient that at least one instance of *influences between solutions* class represents a contradiction relation. Therefore a conflict exists between Sol_{cA}^{DK} and Sol_{cB}^{DK}.

*Step*2 Inferring the component trace $CompTrace_{cA-cB}^{DK}$ from $Comp_{cA}^{DK}$ to $Comp_{cB}^{DK}$. $CompTrace_{cA-cB}^{DK}$ is composed of instances of *component* class and *state of components* class, linked to $SolTrace_{cA-cB}^{DK}$.

*Step*3 Inferring the component trace $CompTrace_{cA-cB}^{DC}$ from $Comp_{cA}^{DC}$ to $Comp_{cB}^{DC}$. It is sufficient that each instance of $CompTrace_{cA-cB}^{DK}$ is linked to an instance of $CompTrace_{cA-cB}^{DC}$ for *component* class.

The proposed algorithm enables us to detect conflicts by inferring the contradiction trace from chain relations, which the proposed ontology represents in stages. The inference process in each step is implemented using Jena API [6] and SPQRQL query.

3 Case Study

The proposed method was introduced to one design case of our developing product. In *Phase 1*, the designer found phenomena in which the component " Cable 50 " extends out of the component Sensor 20. In *Phase 2*, the designer analyzed factors which indicated that the capacity of Sensor 20 was too small. In *Phase 3*, the designer applied the solution of decreasing the thickness of Sensor 20, thereby increasing its capacity. The designer then validated the conflict between the solution of the current case and the solutions of existing cases. Fig.2 illustrates a use-case for detecting the conflict between solutions using our proposed method. Results showed that our proposed method could accurately detect the conflict between (S10) *decreasing the thickness of Sensor 20* and (S20) *increasing the mass of Frame 10*.

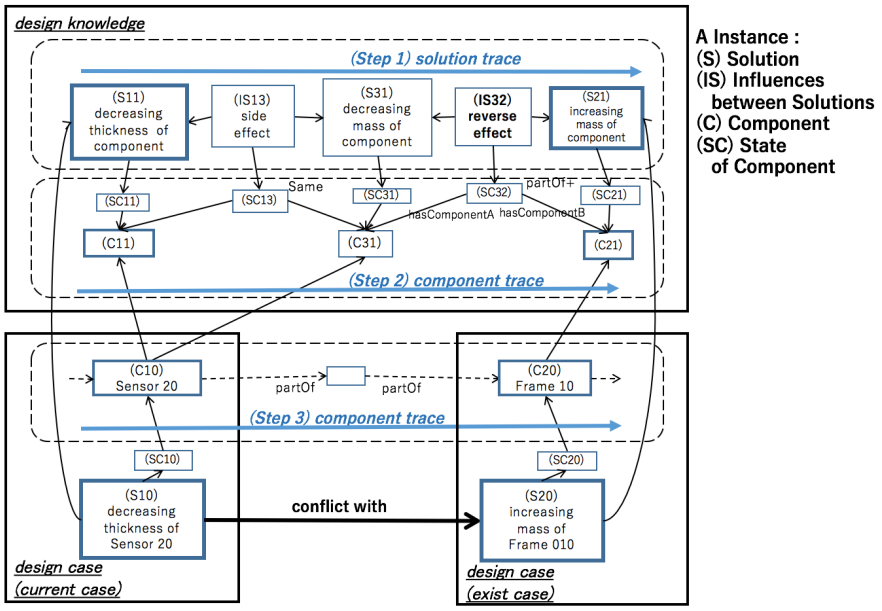

Fig. 2. A use-case of detecting the conflict between solutions using proposed approach

In *Step*1, our method inferred the solution trace of design knowledge as follows: (S11) *decreasing thickness of component* → (IS13) → (S31) → (IS32) → (S21)*Increasing Mass of Component*. It was sufficient that this solution trace included the contradiction relation (IS32)*Reverse Effect*.

In *Step*2, corresponding to the above solution trace, our method inferred the component trace of design knowledge as follows: (C11) → (SC13) → (C31) → (SC32) → (C21).

In *Step*3, corresponding to the above component trace, our method inferred the component trace of the design case as follows: (C10) →... → (C20). Finally, the link (C21) → (C20) was inferred using (C31) → (C10), (C31) → (C32) →

(C21). When the component trace from (C10) to (C20) existed, the conflict between (S10) and (S20) was indicated.

As mentioned above, this case study clearly shows that the proposed method can detect conflicts between solutions and represent the potential chain relation which designers could not identify. This chain relation was defined as follows: (S10) → (S11) → (IS13) → (S31) → (IS32) → (S20). As a result, our method could detect potential conflicts between the solutions of 10 design cases.

4 Conclusion

In this paper, we focused on validating solutions in design cases for product manufacturing. We proposed an ontology for representing chain relations among solutions and components, and proposed an algorithm for detecting conflicts between their solutions. Our method realizes detecting the potential contradiction among designers' solution in the initial design phase, so that they can reduce design troubles in the post-process, and the entire design man-hour is reduced. In our case study, the proposed method was applied to our developing product and could detect potentials conflict between the solutions of 10 design cases.

Designers must continuously enhance their design knowledge, and we believe our our proposed method is a useful tool for the enhancement of that knowledge. When conflicts which designers can grasp are not detected by the proposed algorithm, the area between solutions includes either erroneous instances or missing instances. Designers can then identify the areas which require enhancement in terms of design knowledge.

It is hoped that the proposed method presented in this paper will contribute to improved solution validation in manufacturing design.

References

1. Yoshinobu, Kitamura.: Roles of Ontologies of Engineering Artifacts for Design Knowledge Modeling, International Seminar and Workshop Engineering Design in Integrated Product Development, 59–69, (2006)
2. Zobia, Rehman., Claudiu, V, Kifor.: An Ontology to Support Semantic Management of FMEA Knowledge, International Journal of Computers Communications & Control, 11(4):507–521, (2016)
3. Ioannis, M, Dokas., Cork, Ireland.: Ontology to Support Knowledge Representation and Risk Analysis for the Development of Early Warning System in Solid Waste Management Operations, International Symposium on Environment Software Systems, (2007)
4. Martin, Melik-Merkumians., Alois, Zoitl.: Ontology-based Fault Diagnosis for Industrial Control Applications, IEEE conference on ETFA, (2010)
5. Allan, Venceslau., Raphaela, Lima., Luiz, Affonso, Guedes., Ivanovitch, Silva.: Ontology for Computer-aided Fault Tree Synthesis, IEEE Conference on ETFA (2014)
6. Apache Jena. https://jena.apache.org/documentation/inference/index.html