MANTECH Publications

Journal of Data Mining and Knowledge Engineering
Volume 7, Issue 1, January-June 2022

Model of Hybrid Information for Product Development Management

Aryan Gupta[1], Mehul Sharma[2]
Student[1], Professor[2]
Department of Computer Science Engineering
Shri Siddhi Vinayak Institute of Technology, Bareilly
Corresponding Author's Email: - *ag213532@gmail.com[1]*

Abstract

As an underlying paradigm for product development management, a hybrid knowledge model is proposed. It is capable of supporting hybrid features such as ontologies and rules. Effective product development cooperation is dependent on the sharing and reasoning of product information as well as engineering knowledge. Much research has taken product information and engineering expertise into account. However, much past research has concentrated on either developing ontology of product information or developing rule-based engineering knowledge systems. This research demonstrates how a hybrid F-logic-based knowledge model might enable such desirable characteristics.

Keywords: *Product development, Ontology, Rule, and F-logic*

INTRODUCTION

The collaborative product development environment includes product information models, domain-specific engineering knowledge, and different. The product information model includes semantics about product-related data properties such as product structures, work breakdown structure, organizational roles, and so on, whereas engineering knowledge is about the rule-base, including domain-specific knowledge in addition to product engineering such as product configuration, process planning, cost targeting, and so on.

The lack of formal and clear semantics in the product information model, which might allow semantic interoperability, is a

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved

crucial challenge for efficient cooperation in a diverse setting. A wide range of fundamental and practical research efforts on product data interchange and semantic integration has been done throughout the years with the goal of semantic interoperability across partners. Many studies have found that the ontology-based method is best suited for integrating heterogeneous engineering semantics because product semantics developed in ontology languages – such as the Web Ontology Language (OWL) can be stated in a well-defined and clear way.

Another significant challenge for knowledge-intensive product engineering is collecting and utilizing domain-specific information and expertise in decision-making procedures. To address this difficult challenge, rule-based systems combining expert knowledge in the form of If-Then rules have been proposed.

Normally, ontologies and rules are expressed in different Ontologies and rules are typically stated in various knowledge representation formalisms, making interaction challenging. This is also likely to result in information loss during exchanges. As a result, integrating ontology- and rule-based techniques is a critical subject in knowledge-based product development management.

Using the F-logic approach, this work provides a hybrid method for integrating ontology-based and rule-based techniques without sacrificing information.

PRELIMINARY

The F-logic technique expands and subsumes predicate calculus while combining the benefits of logic programming with the object-oriented programming paradigm. F-logic is also known to come in two flavours: first-order and logic programming. F-logic, in particular, combines the logic programming and deductive database paradigms with the object-oriented programming paradigm.

The majority of F-logic uses are for logic-based intelligent information systems and F-logic has been used to express ontologies because to its direct support for object-oriented ideas, frame-based syntax, and substantial support for meta-programming. We present a quick review of the primary characteristics of F-logic here.

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved

Objects

Objects are fundamental syntactic components in F-logic since it incorporates object-oriented programming. Objects can be retrieved via the object identity (OID); for example, John, man, and son(John) are all possibilities.

Methods

Data-F-atoms with a host object, a method object, and a result object are used to specify methods. Any item is permitted to appear in any data-F-atom place. Methods can be single-valued or multi-valued, as indicated by the symbol "; for example, "John[son tom]" communicates that tom (result) is the son (method) of John (host). Similarly, "John[son Tom, Hans]" says that Tom and Hans are sons of John and that John may have further sons.

Class Hierarchy

An Isa-F-atom of the type o:c indicates that object o belongs to class c. A subclass-F-atom of the form sc::c specifies that the class sc is a subclass of class c; for example, Tom: man specifies that Tom is an instance of the class man, and man::person specifies that the class man is a subclass of the class person.

Signature

Signature-F-atoms define a class's schema. They specify the methods in the classes, as well as the types of arguments used by those methods and the methods' ranges. The arrow "*=>" is used instead of "→" to differentiate signature-F-atoms from data-F-atoms; for example, person[son *=> man]. To declare the signature of an inheritable method in F-logic, we use "*=>." When a method is not inheritable, the symbol "=>" is used.

F-molecules

It is feasible to combine many F-atoms into a single, useful F-molecule, such as Jonh: man.[son → Tom: man, daughter → Jane: woman].

Rues

The use of rules to generate new information from a given knowledge base is one of F-greatest logic's characteristics. A rule is made up of a head and a body separated by ":-," with the head being an F-molecule and the body being a Boolean combination of F-molecules or negated F-molecules. F-logic, in particular, supports the closed world assumption, which, while not expressly stated, is understood to be untrue. This is also known as non-monotonic.

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved

HYBRID KNOWLEDGE MODEL

This study provides an F-logic-based hybrid knowledge representation and reasoning technique for integrating an ontology-based product information model with a rule-based engineering knowledge model. The product semantics are included in the product information model, whereas engineering-specific knowledge is included throughout the product development process. The majority of product semantics are represented using a frame-based method, but some are specified using rules, and engineering-specific information is generally represented using rules. This integration is made possible by the use of F-logic.

Generic Product Development Process

The generic product development process may be divided into four stages. The process starts with idea development, in which the target market's needs are discovered, different product concepts are produced, and one or more concepts are chosen. The system-level design phase comprises the specification of the product architecture as well as the product's split into subsystems and components.

The detail design phase comprises the comprehensive definition of product attributes such as geometry, materials, and tolerances for all product pieces. Product verification and validation, as well as the building and assessment of several product prototypes, are all Part of the testing and refining process. To summarise, the product development process starts with the determination of client needs and continues through these four stages. The knowledge model makes the product development process more efficient, and hence to make the firm competitive, the relationships between product development stages, product information models, and engineering knowledge bases are depicted in Fig. 1.

We'll go through a project on the conceptual design of automotive air purifiers today. Figure 2 depicts a rudimentary procedure for designing an automobile air cleaner. Target specification decision, fan type selection, and fan design are the three primary responsibilities. Although each task may have several sub-tasks, we do not focus on those sub-tasks and instead concentrate on the three tasks. Almost every time a new air purifier is needed, the goal parameters must be determined in order to meet the needs of the client.

The two most essential goal requirements are noise (dB) and air flow rate (m3/hour),

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved

and they have a significant influence on fan type selection. As shown in Fig. 3, the second duty is to pick the appropriate fan type to fulfil the specified noise and air flow rate. Other specific design activities begin once a project manager decides on a fan type. Each thorough design activity can be completed by geographically dispersed partners, and then design knowledge must be shared across participants.

The application scenario procedure consists of five steps: Step 1 is for knowledge engineers to build knowledge, step 2 is for customer requirements to be set, step 3 is for product structure building by adding a new instance based on the has Direct Comp method, step 4 is for product type selection with reasoning about both product information model and engineering knowledge to realize the instance from the customer requirements, and step 5 is for process planning of the application scenario.

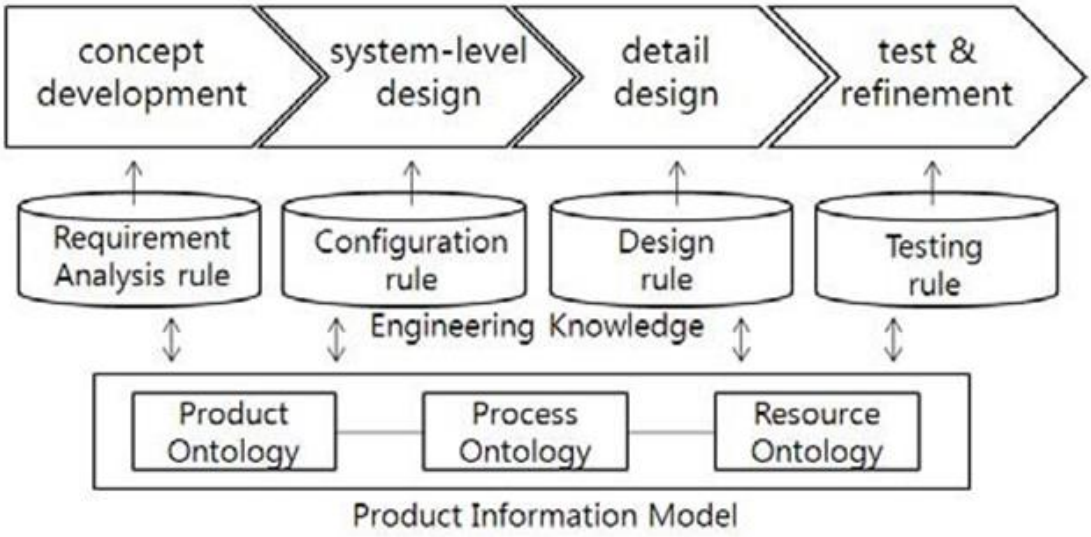


Fig. 1 The interaction between stages and models



Fig. 2 Generic design process of a car air purifier

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved

F-logic has a Hybrid Knowledge Model

Because of its support for object-oriented ideas and frame-based syntax, F-logic has been considered a suitable option for an ontology language. A typical ontology consists of three major components: a taxonomy of classes, concept definitions, and instance definitions. Class taxonomies are expressed directly in F-logic by the subclass-F-atom; for example, composite Part::part. Concept definitions are expressed by signature-F-atoms, such as composite Part [has direct Comp {2:*} *=> part], and instance definitions can be stated as facts using Isa-F-atoms or data-f-atoms, such as Part001:part or CompositePart002[has direct Comp → Part004, Part006, Part007]. Furthermore, derived classes may be defined using rules. For example, if the notions of Part are already specified, we may define a new concept, small Part, by utilizing the statements:

?X:small part :- ?X:part, ?X[size → Small], where ?X is a variable.

Hybrid Knowledge Model for Car Air Purifiers

Table I addresses the different product information models necessary for the conceptual design of automobile air purifiers, and methods may also be developed using rules, for example, the transitive method. A component is defined as follows:

?X[hasComponent → ?Z] :-
?X[hasComponent → ?Y],
?Y[hasComponent → ?Z].

Major choices relevant to the conceptual design of automobile air purifiers, such as goal specification determination, fan type selection, and fan design, are used to develop domain-specific technical knowledge. Table II contains the rule formulations for each job.

Table I: Product Information Model for Car Air Purifiers

Index	Formulae
I1	*productPart::part.*
I2	*primitivePart::part*
I3	*assemblyPart::part[hasDirectComp{2:*} *=> part]*
I4	*carAirPurifier::productPart[hasComponent{1:1}*=>capFanAssy, hasComponent{1:1} *=> capCaseAssy, isInstalled{1:1}*=> position].*
I5	*capFanAssy:assemblyPart[hasDirectComp{1:1}*=> capFan, hasDirectComp{1:1}*=> capMotor].*
I6	*capCaseAssy::assemblyPart.*

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved

I7	capFan::assemblyPart[hasDirectComp{1:1}*=> blade, hasFanType{1:1} *=> fanType].
I8	capMotor::assemblyPart.
I9	blade::primitivePart.
I10	capAxialFan::capFan[hasFanType{1:1} *=> axial].
I11	capCrossflowFan::capFan[hasFanType{1:1} *=> crossflow].
I12	capSiroccoFan::capFan[hasFanType{1:1} *=> sirocco].
I13	capDualFan::carAirPurifier[hasComponent {2:2}*=> capFan].
I14	capSingleFan::carAirPurifier[hascomponent{1:1}*=> capFan]
I15	capSingleSirocco::capSingleFan[hasComponent{1:1}*=> capSiroccoFan
I16	capSingleSiroccoIn::capSingleSirocco[isInstalled{1:1}*=>inside].
I17	capSingleSiroccoOut::capSingleSirocco[isInstalled{1:1} *=>outside].
I18	axial::fanType.
I19	crossFlow::fanType.
I20	sirocco::fanType.
I21	inside::position.
I22	outside::position.

Table II: Domain Specific Engineering Knowledge For Car Air Purifiers

Index	Formulae
R1	?X [noiseLevel → small] :- ?X:capFan[requiredNose → ?Y], ?Y > 0, ?Y < 30.
R2	?X [noiseLevel → medium] :- ?X:capFan[requiredNose → ?Y], ?Y > 30, ?Y < 40.
R3	?X [airflowLevel → weak] :- ?X:capFan[requiredAirflow → ?Y], ?Y > 10, ?Y < 30.
R4	?X [airflowLevel → medium] :- ?X:capFan[requiredAirflow → ?Y], ?Y > 30, ?Y < 50.
R5	?X [airflowLevel → strong] :- ?X:capFan[requiredAirflow → ?Y], ?Y > 50, ?Y < 100.
R6	?X[fanType → axial] :- ?X:capFan[noiseLevel → small, airflowLevel → weak] .
R7	?X[fanType → crossflow] :- ?X:capFan[noiseLevel → small, airflowLevel → medium] .
R8	?X[fanType → sirocco] :- ?X:capFan[noiseLevel → medium, airflowLevel → strong] .
R9	?X[installed → outside] :- ?X:carAirPurifier[hasCapFan → {?Y:capFan[noiseLevel → high]}].
R10	?X[installed → inside] :- ?X:carAirPurifier[hasCapFan → ?Y:capFan], not Y[noiseLevel → high].
R11	?X[hasMaterial → steel] :- ?Y:carAirPurifier[hasComponent → ?X:blade, isInstalled → inside].
R12	?X[hasMaterial → plastic] :- ?Y:carAirPurifier[hasComponent → ?X:blade, isInstalled → outside].

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved

Product Structure Building and Type Design

automobile air purifier. 1) Noise requirements of 35 decibels; 2) airflow requirements of 70 cubic metres per hour.

We are required to define cap Fan Assy and cap Case Assy after the index I4 in Table I. To meet the definition of a car air purifier, FanAssy01:capFanAssy and CaseAssy01:capCaseAssy are directed to be added and linked with Cap01 through the method has Direct Comp. As a result, in order for FanAssy01 to be a cap Fan Assy instance, it is required to contain one cap Fan and one cap Motor as direct components, as indicated in index I5. Thus, CapFan01 and CapMotor01 are directed to be built, and they are then associated with FanAssy01 using the has Direct Comp function. Similarly, Blade01 is linked to CapFan01 via the has Direct comp function. Figure 3 depicts the whole product structure up to this point.

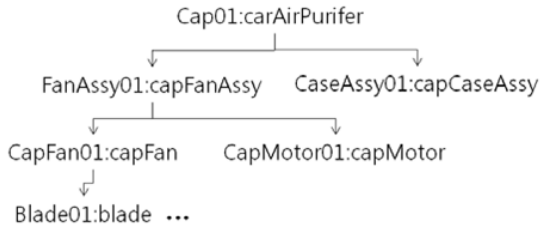


Fig. 3 Product structure of car air purifier

Next, we must choose a fan type that will meet the criteria for noise and airflow. The fan type, "noise Level→ medium," and "air Flow Level → strong" might well be deduced from index R2, and the kind of Cap Fan 01 can be deduced from index R8 in Table II. Based on these findings, Cap Fan 01 is re-conceptualized as cap Sirocco Fan since it possesses Sirocco: Fan Type, which matches the specification of cap Sirocco Fan as given in index 111. Furthermore, has Direct Comp::has Component and the transitivity postulate of has Component lead to the conclusion that Cap01 has the components CapFan01 and CapMotor01. Cap01 is re-conceptualized as cap Single Fan as a result of this. Cap01 is now being re-conceptualized as cap Single Sirocco, as Cap01 only has one fan, and its type is Sirocco. Cap01's installation location is determined by its noise level; if it is loud, it should be positioned outside. We know the Cap01 is mounted inside the automobile since its noise level is not demonstrated to be excessive, i.e., denial under the closed world assumption. Cap01 is re-noncertified into Cap Single Sirocco In, the most particular concept to which Cap01 belongs, using re-notion reasoning. Figure 4 and Table III provide the summary results.

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved

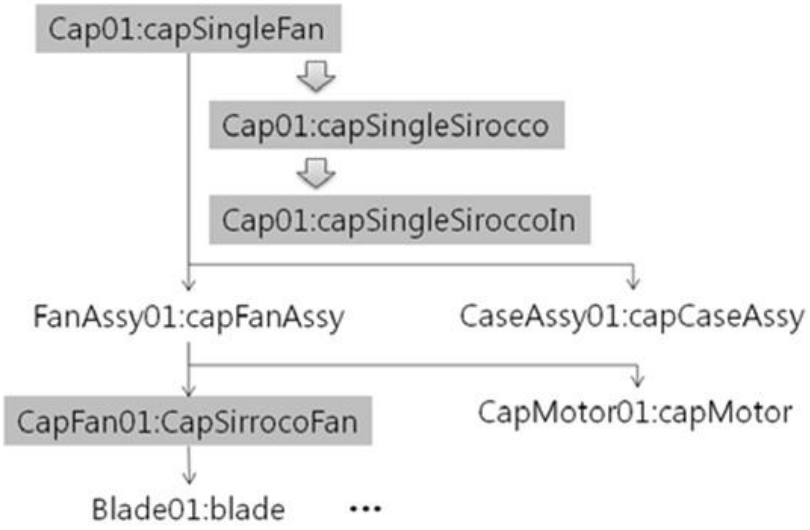


Fig. 4 Product type design of car air purifier

Table III: Hybrid Reasoning For Product Type Design

reasoning	Answer	Justification
Conversion	*?X [noiseLevel → medium], ?X[airflowLevel → strong]*	*R2, R5*
Fan Type Selection	*?X[fanType → sirocco]*	*R8*
Re-Concept	*CapFan01: capSiroccoFan*	*I12*
Re-Concept	*Cap01: capSingleFan*	*I14*
Re-Concept	*Cap01: capSingleSirocco*	*I15*
Installation Place Selection	*?X[installed → inside]*	*R10*
Re-Concept	*Cap01: capSingleSiroccoIn*	*I16*

CONCLUSION

The product information model and the domain-specific rule model are the two basic forms of product knowledge in a collaborative product development environment. The product information model comprises ontological semantics about the product's qualities such as structure, material, and features. In contrast, the domain-specific rule model comprises particular knowledge (represented as specific rules) as well as product engineering activities such as product configuration, material selection, and process planning.

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved

Typically, these two models have been stated in distinct knowledge representation formalisms, making interaction problematic. By utilizing the F-logic methodology, this work provides a hybrid method for integrating the ontology-based and rule-based approaches.

REFERENCES

1. M. Dong, D. Yang, and L. Su, "Ontology-based service product configuration system modeling and development," Expert Systems with Application, vol. 38, pp. 11770-11786, 2011.

2. R. Studer, V. R. Benjamins, and D. Fensel, Knowledge Engineering: Principle and methods," Data& Knowledge Engineering, vol. 25, no. 1-2, pp. 161-197, 1998.

3. S. R. Gorti, A. Gupta, G. J. Kim, R. D. Sriram, and A. Wong, "An object-oriented representation for product and design processes," Computer-Aided Design, vol. 30, no. 7, pp. 489-501, 1998.

4. R. Sudarsan, S. J. Fenves, R. D. Sriram, and F. Wang, "A product information modelling framework for product lifecycle management," Computer-Aided Design, vol. 37, no. 13, pp. 1399-1411, 2005.

5. L. Patil, D. Dutta, and R. D. Sriram, "Ontology-Based Exchange of Product Information Semantics," IEEE Trans. On Automation Science and Engineering, vol. 2, no. 3, pp. 213-225, 2005.

6. M. K. Smith, C. Welty, and D. L. McGuinness, OWL Web ontology language guide, W3C recommendation, http://www.w3.org/TR/2004/ REC-owl-guide-20040210/, 2004.

7. K. Y. Kim, H. Yang, and D. W. Kim, Mereotopological assembly joint information representation for collaborative product design," Robotics and Computer-Integrated Manufacturing, vol. 24, no. 6, pp. 744-754, 2006.

8. J. D. Noh, H. W. Suh, and H. J. Lee, "Hybrid knowledge representation and reasoning with ontology and rules for product engineering," ASME, IDETC/CIE, 2009.

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved

9. S. Staab and R. Studer (eds.), Handbook on Ontologies, International Handbooks on Information Systems, Springer-Verlag, Berlin, Heidelberg, 2009. pp. 45-70.

Page 31-41 © MANTECH PUBLICATIONS 2022. All Rights Reserved