



## An Ontology for Representation of Fixture Design Knowledge

Farhad Ameri<sup>1</sup> and Joshua D. Summers<sup>2</sup>

<sup>1</sup>Clemson University, [fameri@clemson.edu](mailto:fameri@clemson.edu)

<sup>2</sup>Clemson University, [jsummer@clemson.edu](mailto:jsummer@clemson.edu)

### ABSTRACT

As design and manufacturing practices are increasingly becoming global and distributed, collaborative design by dispersed machine agents is garnering more attention as an appropriate framework for product development. One of the major challenges in deployment of agent-based systems in distributed environments is provision of a common language which connects the heterogeneous actors both syntactically and semantically. Focusing on fixture design process, this paper introduces a formal ontology, called FIXON, for representation of domain knowledge. The proposed ontology uses Description Logic (DL) as the knowledge representation formalism, thus enabling automated reasoning and inference by machine agents. To demonstrate how the proposed ontology supports seamless information exchange among machine agents, four application areas, namely, problem formulation, design representation, design retrieval and design validation are discussed.

**Keywords:** fixture design, ontology, description logic.

**DOI:** 10.3722/cadaps.2008.601-611

### 1. INTRODUCTION

Automation of fixture design process has long been recognized by researchers as a requirement for effective integration of computer-aided design (CAD) and computer-aided manufacturing (CAM) systems in modern manufacturing environments [5]. There have been numerous efforts to address automation in fixture design through development of Computer Aided Fixture Design (CAFD) tools and methods. In a broad sense, CAFD methods primarily enable automated calculation of the design attributes of clamps, locators, and supports for varying part designs. Genetic algorithms [16], neural networks [15], and rule-based methods [4] are among the approaches commonly used in CAFD.

In parallel with increased automation of various design activities, engineering design environments have widely become collaborative and distributed. To account for this emerging trend, vendors of engineering design tools, particularly CAD systems, are increasingly supplementing their products with collaborative features. Literature survey revealed that CAFD systems lag behind in terms of capability of operating in collaborative and distributed environments [14]. Insufficient collaboration capabilities of the existing CAFD systems can be attributed to several factors such as difference in the underlying business model (i.e., workflow, versioning etc.), overall system architecture and information modeling formalism. Although explicit integration of existing CAFD systems is a possibility, based on the experience gained from integrating CAD systems, it is not necessarily the most efficient approach for dealing with the requirements of distributed environments. An alternative solution for fixture design in distributed environments is incorporation of the agent technology.

Agents are automated software entities operating through autonomous actions on behalf of their owners, being machines or humans [20]. A multi-agent system (MAS) is a loosely coupled network of agents that interact to solve

problems that are beyond the individual capabilities or knowledge of each agent [7]. In an agent based scenario for fixture design, a CAFD system is decomposed into its constituent functional modules and then, each module is assigned to an autonomous agent. Given the multiplicity of possible solutions for a given fixture design problem as well as the distribution of expertise in the envisioned design environment, fixture design problem lends itself well to agent-based scenarios. Development of a multi-agent system has several dimensions including system architecture, interaction model, and agent communication. One of the major challenges in agent-based system is agent communication. Standard agent communication languages, such as KQML [8] and FIPA-ACL [9], address agent communication problem in the messaging level. However, in the content level, a domain-specific language is required which represents design knowledge in a machine-interpretable fashion. In order to enable intelligent decision making by the machine agents, this language needs to have enough expressive power to formally encode a wide spectrum of knowledge ranging from design specifications and constraints to design rules and axioms. This paper introduces an ontology for conceptualization and representation of domain knowledge in fixture design process. The proposed ontology, called FIXON, is based on Description Logic (DL) [3], a knowledge representation formalism which conceptualizes a domain through concepts and their inter-relations.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of the related works in developing design ontologies. In Section 3, FIXON is introduced through describing its syntax and semantics. In Section 4, some application areas for FIXON are discussed. The paper ends with conclusions.

## 2. ENGINEERING ONTOLOGIES

The concept of ontology was originally used in the philosophical context as the *science or study of being*. But ontology as an artifact was first used in the artificial intelligence (AI) community primarily for the purpose of knowledge sharing. In the context of knowledge sharing, ontology is defined an *explicit specification of a shared conceptualization* [10]. In other words, ontology is a description of the concepts and relationships that can exist for an agent or a community of agents. Since early nineties, ontologies have been subject of research in different AI applications like knowledge engineering, natural language processing and knowledge management. Recently, parallel with rapid growth of information technology and distributed processing of knowledge, ontologies have become more popular due to what they promise: streamlining communication among dispersed agents through providing a shared understanding of a domain of discourse. Through capturing the key concepts and their inter-relations in a given domain, ontologies formally encode semantics in a rich fashion, thus enabling more efficient communication among agents.

The development of ontologies in engineering design has become increasingly important to address the communication gaps in the extended enterprise. Engineering ontologies are developed using entity-relationship models, object-oriented data models, frame-based models, relational databases, semantic networks, and logics-based approaches to name a few. Among the logic-based approaches, Description Logic (DL) is the most widely implemented formalism for knowledge representation in engineering ontologies because it offers several advantages over other ontology representation languages. First, DL is supported by the Semantic Web (SW). Therefore, the tools and technologies (such as browsers and search engines) that are being developed for SW can be readily used for web-based exchange of the information represented in DL-based ontologies. Second, DL is a mathematically rigorous representation that enables reasoning to be performed on the ontology including concept consistency, concept equivalence, concept subsumption, and least common / most specific subsumer identification.

Use of DL-based ontologies for representation of design knowledge has been addressed by several researchers. Kopena and Regli [11] developed an ontology based on DL for formal representation of design artifacts in design repositories. Their work was motivated by the need for intelligent search and retrieval capabilities in the increasingly complex and heterogeneous design repositories. Udoyen and Rosen [18] used DL to represent FEA models for automated retrieval. In their ontology, FEA models are represented through their distinguishing characteristics such as major components, structure, material, and load. Using the classification service provided by DL, they presented a classification-based search approach in order to improve the precision of search results. Nanda et. al [13] developed a DL-based ontology for representation of design artifacts to support product family modeling. They demonstrated how classification capabilities of DL can be used to categorize products based on their functionalities. In the domain of fixture design, Pehlivan and Summers [14] proposed a DL-based ontology for fixture representation. Their work provides a comprehensive conceptualization of the domain of interest. However, their ontology did not capitalize on the axiomatic power of DL. This paper presents a work which builds upon [14] and extends it to cover axiomatic representation of fixture design knowledge in order to enable deduction and reasoning by machine agents.

### 3. KNOWLEDGE REPRESENTATION IN FIXTURE DESIGN

Several approaches for ontology development are documented in the literature. In this work, the method proposed by Uschold and Gruninger [19] is followed. Based on their method, ontology development is composed of five major steps, namely, determination of purpose and scope, ontology capture, ontology coding, evaluation, and documentation. It should be noted that ontology development is a recursive process and the aforementioned steps might be repeated frequently as the ontology evolves over time.

#### 3.1 Identifying Purpose and Scope

FIXON is aimed to provide a common vocabulary for communication of machine agents during fixture design process. Furthermore, through formal representation of the semantics of fixture design knowledge, FIXON enables active involvement of machine agents in problem formulation, fixture synthesis and fixture analysis. The scope of FIXON, in its current state, is limited to representation of machining fixtures designed for prismatic parts.

#### 3.2 Ontology Capture

Ontology capture involves identification of the key concepts and relationships in the domain of interest and production of precise and unambiguous text description of the identified concepts and relationships and, finally, gaining consensus on them. To build a reliable conceptualization in the domain of fixture design, a sound understanding of fixture design process is necessary. This understanding was acquired through studying fixture design textbooks, investigating different CAFD systems, as well as practicing fixture design for several industrial sponsored projects. Fig. 1 shows the major phases of fixture design process together with their corresponding inputs and outputs.

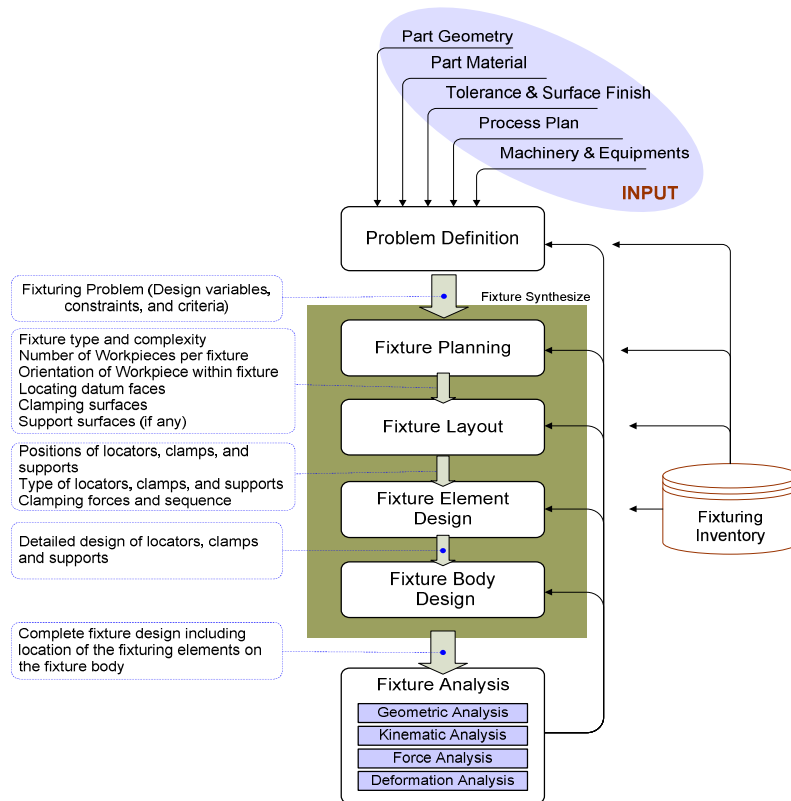


Fig. 1: Major phases of fixture design process together with corresponding inputs and outputs.

In the conceptualization phase, special attention was given to CAFD information models because, collectively, they provide a comprehensive set of concepts and relations in fixture design domain. Some of the key concepts, appearing

in most of CAFD systems, include surface, tolerance, material, clamp, locator, workpiece, process, position, vertex, machine etc.

Tab. 1 shows some of the core concepts identified in the domain of fixture development together with their textual descriptions.

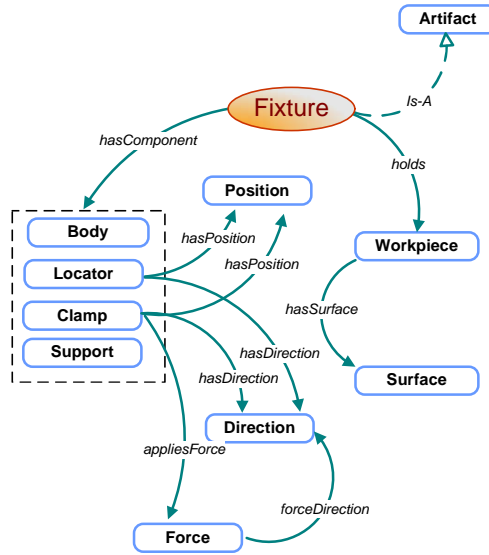


Fig. 2: Concept diagram for the core concepts in FIXON.

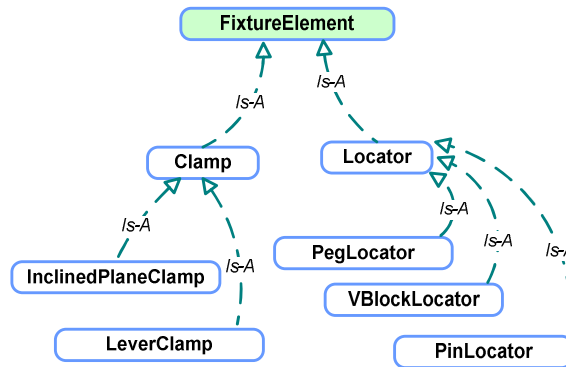


Fig. 3: A partial taxonomy of fixturing elements.

The reader is referred to Pehlivan and Summers [1] for a complete description of the concepts captured in the domain of fixture design. Once the concepts are identified, the next step in the conceptualization phase includes identification of relationships between the identified concepts. Relationships between some of the core classes of fixture ontology are demonstrated in Fig. 2. Of particular interest are “is-a” relationships that connect the concepts in a taxonomical fashion. Some of the major taxonomies in FIXON are surface taxonomy, material taxonomy, clamp taxonomy, and locator taxonomy. Fig. 3 shows the top most level of clamp and locator taxonomy.

### 3.3 Ontology Coding

Ontology coding entails explicit representation of the conceptualization captured in previous stage in an ontology language. In this work OWL DL is selected as the ontology language. OWL DL is a sub-language of OWL which is recommended by World Wide Web Consortium (W3C) as the ontology language of the Semantic Web.

Concept	Definition
Fixture	A device for locating, holding and supporting a workpiece during a manufacturing process.
Locator	A fixed component of a fixture that is used to establish and maintain the position of a part in the fixture by constraining the movement of the part.
Clamp	A force-actuating mechanism of a fixture used for holding a part for securely in the fixture against all other external forces.
Support	A fixed or adjustable element of fixture placed below the workpiece to prevent or constraint deformation.
Body	The major structural element of a fixture that maintains the spatial relationship between the fixturing elements.
Active Surface	A surface of the workpiece which is subjected to the action of cutting tools.
Datum Surface	Datum surfaces are reference surfaces where the dimensions are to be maintained and measured.
Clamping Surface	A surface which is subject to clamping force
Clamping Force	A force exerted by a clamp
Machining Force	A force exerted by the cutting tool during the machining operation.

Tab. 1: Definition of some of the core concepts in FIXON.

OWL has an XML-based syntax and hence, it has enough portability, flexibility, and extensibility for web-scale applications. Also, OWL uses Description logic (DL) as the knowledge representation language. Elementary descriptions in DL are atomic concepts and atomic roles. Complex descriptions can be built from elementary ones through the use of constructors such as intersection ( $\cap$ ), union ( $\cup$ ), negation ( $\neg$ ) and quantification [13][19]. DL is equipped with a formal, logic-based semantics. Therefore, it allows one to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base. Typical reasoning tasks in DL include checking subsumption, equivalence, satisfiability, and disjointness.

Since the semantics of the fixture ontology are mainly in the form of concept definitions and their interrelations, the expressivity offered through DL is sufficient for knowledge representation. Also, the automatic reasoning capabilities offered by DL support active involvement of machine agents in decision making process. In this section some of the formal definitions of the core concepts of FIXON are provided. In OWL's terminology, a *defined* concept is represented by a set of necessary and sufficient logical constraints (*axioms*).

### 3.3.1 Fixturing Components

Definition (1): Fixture is an artifact that holds, supports, and locates a workpiece and has at least one clamp and six locators.

$$\text{Fixture} \equiv \text{Artifact} \cap (\exists \text{ holds. Workpiece}) \cap (\exists \text{ supports. Workpiece}) \cap (\exists \text{ locates. Workpiece}) \\ \cap (\exists \text{ hasComponent. Clamp}) \cap (>= 6 \text{ hasComponent. Locator})$$

Definition (1) is more specific and detailed than the common-sense definition provided in Tab. 1. A more detailed ontological definition reduces the possibility of misinterpretation. At the same time, extremely detailed definitions lower the generality of ontology and increase the computational complexities of ontology-based reasonings. The last constraint in definition (1) is incorporated based on 3-2-1 principle, a principle for secure location of prismatic parts through restricting all 6 degrees of freedom.

Definition (2): Clamp is a fixture component with a fixed position that constrains the movement of workpiece through applying force in exactly one known direction.

$$\text{Clamp} \equiv \text{FixtureElement} \cap (\exists \text{ constrains. Workpiece}) \cap (\exists \text{ applies. (Force} \cap (=1 \text{ hasDirection. Direction} \cap (\exists \text{ hasPosition. Position}))$$

Definition (3): Locator is a fixture component with a fixed position that constrains the movement of the workpiece.

$$\text{Locator} \equiv \text{FixtureElement} \cap \exists \text{constrains.Workpiece} \cap \exists \text{hasPosition.Position}$$





As can be seen in definition (3), the only difference between clamp and locator in the present conceptualization is that locator does not apply force to constrain the workpiece. More complex definition for fixturing components can be obtained through supplementing the basic concepts (such as clamp and locator) with extra constraints.

Tab. 2 demonstrates the FIXON representations of some of the standard fixturing components available in commercial fixturing systems.

3.3.2 Surfaces

Surface is a core concept of the fixture ontology. Since in this work parts are assumed to have prismatic shape, their bounding surfaces are always flat. In FIXON, a *FlatSurface* is formally defined as a surface with exactly one normal vector and at least three edges.

$$\text{FlatSurface} \equiv \text{Surface} \cap =1.\text{hasNormalVector} \cap >=3.\text{hasEdge}$$

Fixturing Component	Name	FIXON Description
	Slotted-heel clamp strap	$\equiv \text{Clamp} \cap \exists \text{hasType.CamClamp}$ $\cap \exists \text{hasComponent. (Strap} \cap \exists \text{hasFeature.Slot)}$ $\cap \exists \text{hasMaterial.1018Steel}$
	Diamond locating pin	$\equiv \text{Locator} \cap \exists \text{hasFeature.Shoulder} \cap (\exists =1 \text{ locatingAxis})$ $\cap \exists \text{usedWith.Roundpin} \cap \exists \text{hasMaterial.1144Steel}$
	Hook Clamp	$\equiv \text{Clamp} \cap \exists \text{hasComponent (PlainArm} \cup \text{TappedArm)} \cap \exists \text{has workCondition. (Force} \cap \text{hasLevel.high)}$ $\cap \text{hasworkCondition (Space} \cap \text{hasLevel.tight)}$
	Serrated Adjustable Edge Clamp	$\equiv \text{Clamp} \cap \exists \text{hasFeature.Slot}$ $\cap \exists \text{hasComponent.MountingScrew} \text{ hasWorkCondition. (Workpiece}$ $\cap (\exists \text{hasAspectRatio.high}) \cap \exists \text{hasGeometry. Round)}$

Tab. 2: FIXON definition of different modular fixturing components.

Fig. 4 (a) shows a surface of a prismatic part represented through its normal vector (v) and four edges (eij). As specified in Tab. 1, there are two basic types of surfaces in fixturing: *ActiveSurface* and *NonActiveSurface*. An active surface as shown in

Fig. 4 (b) is subjected to the action of cutting tools. In FIXON, a boolean property (i.e., *isActive*) is used to specify whether a surface is active or not. Likewise, *ClampingSurface* and *LocatingSurface* are characterized through *isClamping* and *isLocating* boolean properties respectively. A locating surface cannot be an active surface. This constraint is formally addressed in FIXON through representing *LocatingSurface* and *ActiveSurface* as two disjoint classes.

3.3.3 Forces

Major forces that are influential in fixturing problem include static clamping forces applied by the clamps to secure the part in its location and dynamic machining forces applied by the tools during the operations [14]. Fig. 5(a) demonstrates clamping and machining forces in fixturing. In FIXON, clamping and machining forces are represented by *ClampingForce* and *MachiningForce* classes respectively. The OWL representation of *ClampingForce* is depicted in Fig. 5 (b). As can be seen in Fig. 5 (c), *hasForceSource* property is a functional property. Functional properties in OWL are those properties that can appear only once in any instance of the classes which are within the domain of the property. *hasDirection* and *hasValue* are two properties which further describe the *Force* class in FIXON.

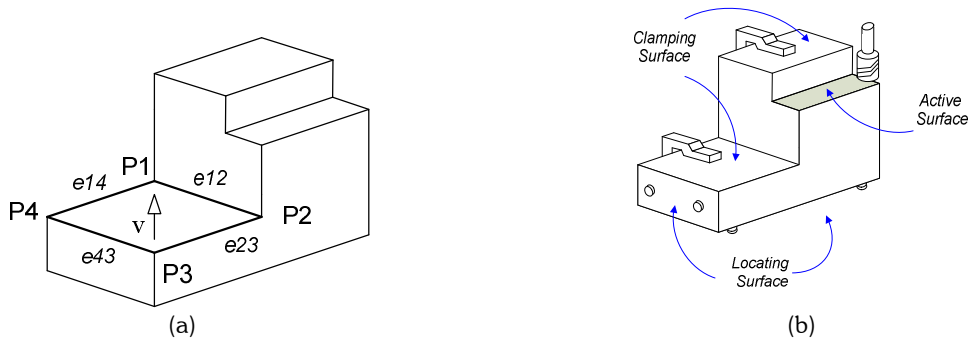


Fig. 4: (a) A surface of a prismatic part represented through its normal vector and four edges (b) different types of surface in FIXON.

(a)

```

<owl:Class rdf:ID="ClampingForce">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Force"/>
        <owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:ID="hasForceSource"/>
          </owl:onProperty>
          <owl:someValuesFrom>
            <owl:Class rdf:ID="Clamp"/>
          </owl:someValuesFrom>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>

```

(b)

```

<owl:ObjectProperty rdf:about="#hasForceSource">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Force"/>
</owl:ObjectProperty>

```

(c)

Fig. 5: (a) Clamping and machining forces in fixturing (b) formal definition of clamping force in FIXON represented in OWL format (c) *hasForceSource* is a functional property in FIXON.

### 3.3.4 Workpiece

Geometry of the workpiece in FIXON is represented through its elemental surfaces described by *Surface* class. Apart from the geometry, material of the workpiece is the other required piece of information in fixture synthesizing process. FIXON imports the *Workpiece* class from Manufacturing Service Description Language (MSDL), an ontology for representation of manufacturing capabilities [1]. The *Workpiece* class in FIXON inherits all the properties of the *Workpiece* class in MSDL including material property and is further augmented by *hasSurface* property.. Class reuse is a recommended practice in ontology development since it enables easier integration of different ontologies.

### 3.3.5 Process Plan

Information modeling for process planning within the context of specific CAPP applications is widely addressed in the literature [12]. However, no ontological approach for neutral representations of process plans is currently available. This work does not intend to introduce a comprehensive ontology for process planning as it requires a separate research endeavor. Nevertheless, since process plan is among the most prominent inputs of fixture design process, FIXON includes a partial process plan representation. Process plan representation in FIXON includes sequence of machining operations, their process type and their corresponding forces. Each operation in FIXON is represented by *Operation* class. Fig. 6 shows the screenshot of *Operation* class as implemented in Protégé. Each *Operation* class is attached to a *Process* class (defined in MSDL) through *hasProcess* property. This property is a functional property which means every *Operation* has only one *Process*.

### 3.3.6 Tolerance

Information about tolerance and surface requirements of a part is a necessary input to fixture synthesize phase. To represent tolerance and surface finish requirements, the required classes are again imported from MSDL. *Tolerance* and *SurfaceFinish* classes are used in MSDL for representing the manufacturing capabilities of suppliers. However, their conceptualization sufficiently meets the needs of FIXON. In the next section, some of the applications of FIXON in fixture design process are discussed.

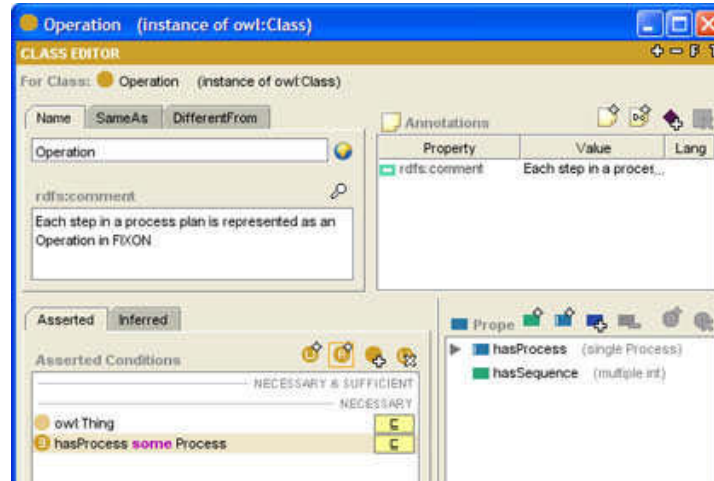


Fig. 6: Screenshot of the *Operation* class created in Protégé.

## 4. APPLICATION AREAS

### 4.1 Problem Formulation

Problem formulation is the first phase of fixture design process which results in framing a fixture design problem in terms of design variables, operating conditions, constraints and criteria [14]. Design variables for this phase are fixturing elements, such as different types of locators and clamps (either available locally or available in remote libraries), assembly and configuration of elements, adjustable parameters, such as clamping forces, and scaling for different sizes of workpieces. The design constraints include form closure type, such as free resting, static and dynamic accessibility, ease of loading and unloading, and deformation. Finally, the primary design criteria include the cost of manufacturing, the level of labor required, productivity, efficiency, and quality [14]. Since a wide spectrum of information with different levels of details are contained in a fixturing problem, it is necessary to represent the problem in a structured and unambiguous manner such that all parties involved in the fixture design project can share a common understanding of the problem. FIXON, with its shared syntax and semantics, provides a common language for representing various fixturing problems posed by dispersed users. Fig. 7(a) shows the source code for a simple fixture design problem formulated in FIXON. In this example, the provided information includes workpiece information and operation information. *FIXONProblem* is the main class that contains the details of fixturing problem.



## 4.2 Fixture Representation

Regardless of the complexity of fixtures in terms of type and number of components, fixtures can always be represented through their clamping and locating elements. More specifically, the minimum requirement for representation of a fixture includes position of clamps and locator, clamping force, and sequence and surfaces associated to each clamp and locator. Fig. 7(b) shows a fixture design (represented by clamps and locators) as well as the FIXON code for description of one of the clamps used in the design. Formal and standard representation of fixture design is an essential prerequisite for efficient search of design repositories. Furthermore, it facilitates design exchange among CAFD systems that submit to FIXON.

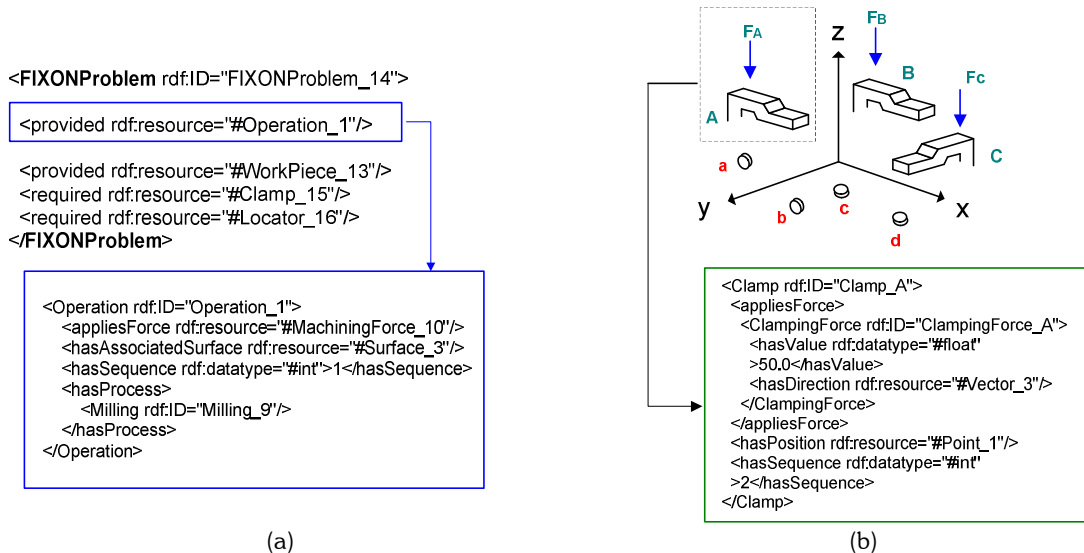


Fig. 7: (a) OWL source code for a FIXON problem (b) FIXON representation of fixture design through the clamping and locating elements.

## 4.3 Design Validation

Automated validation of fixture design is one of the useful features of FIXON enabled by its underlying logic-based formalism. A DL reasoner can compare the design against predefined design constraints and verify the validity of design. For example, a DL reasoner can check whether, in a particular design, workpiece is properly confined based on 3-2-1 principle or not. 3-2-1 principle states that a workpiece is completely constrained when banked against three points in one plane, two points in another plane, and one point in a third plane if the planes are perpendicular to each other. Below is the FIXON representation of 3-2-1 principle:

$$\text{ConstrainedWorkpiece} \equiv \text{WorkPiece} \cap \exists \text{hasSurface. (Surface} \cap (=3 \text{ hasLocator}) \cap (\exists \text{perpeTo. (Surface} \cap =2 \text{ hasLocator}) \cap (\exists \text{perpeTo. (Surface} \cap =2 \text{ hasLocator}))))$$

The right-hand-side of the above equation represents a unanimous class which defines a confined part. It should be noted that the term *ConstrainedWorkPiece* by itself has no meaning and there is no need to hard-wire it as a keyword inside the algorithms which do reasoning on FIXON content. In fact, it is the logical definition of this class which is of significance for the semantic reasoning purpose.

## 4.4 Design Reuse

The design environment envisioned in this work is characterized by distribution. In such environments, there exist several distributed design libraries containing FIXON representation of existing designs. Particularly, given the growing interest in modular fixtures with standards elements, design reuse is becoming a common practice. In this context,

availability of an effective search mechanism for retrieval of existing designs is necessary. Rich representation of design information, at the semantic level, through FIXON can significantly improve the performance of search process. Standard RDF query languages (such as SPARQL [17]), for instance, can be used for querying FIXON files. Fig. 8 shows a SPARQL query which returns all fixtures which use at least one pin locator in their design.

```

PREFIX msdl: <http://www.aid.ces.clemson.edu/FIXON.owl#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf_ns: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?fixture
FROM <http://www.aid.ces.clemson.edu/fixonInstance01.owl#>
WHERE {
?fixture fixon:hasCompomet ?locator.
?locator rdf_ns:type fixon:PinLocator.
}
    
```

Fig. 8: SPARQL query: this query returns all fixtures which use Pin Locator.

Also, sophisticated heuristics can be used for searching FIXON representations. For example, graph matching techniques can be applied for measuring the similarity of existing designs. Furthermore, capitalizing on classification capabilities of FIXON enabled by DL, a flat population of fixtures can be automatically classified into a hierarchical fashion, thus improving fixture reuse practice through forming fixture families.

Fig. 9 shows the subsumption relationship between two fixtures (ABC and XYZ). As can be seen in this figure, XYZ Fixture is subsumed by ABC Fixture based on the components that appear in their definitions. Finding all sub-classes of ABC is equivalent to finding all fixtures that have some type of *toggle clamp* and *pin locator* in their design.

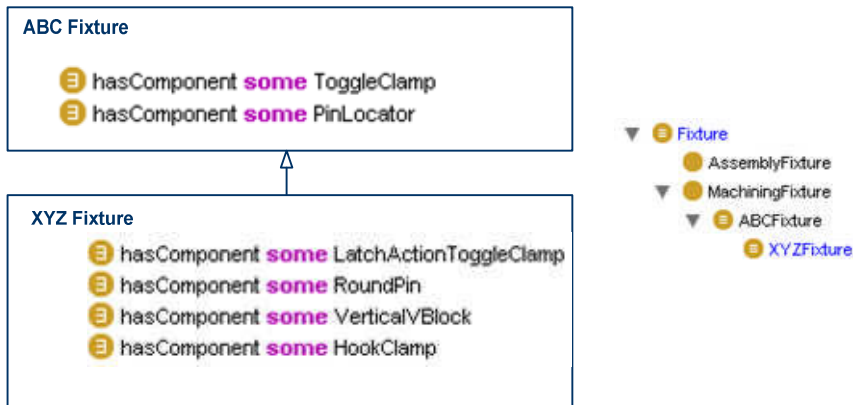


Fig. 9: A subsumption relationship between two subclasses of the Fixture class in FIXON.

**5. SUMMARY AND CONCLUSION**

This paper introduced FIXON as an ontology for representation of fixture design knowledge. FIXON provides a common vocabulary for communication of machine agents during fixture design process. OWL DL is used as the ontology language of FIXON. DL, as a logic-based formalism, provides powerful means for automatic validation of design as well as design search and retrieval in semantic level. Accordingly, FIXON enables active involvement of machine agents in problem formulation, fixture synthesise and fixture analysis. Furthermore, through providing a platform-independent and neutral language for exchange of fixture design knowledge, FIXON supports integration of disparate CAFD systems in distributed environments.

FIXON is successfully implemented as the message content ontology of AFFIXED [2], a multi-agent based system for fixture design. In AFFIXED, FIPA-ACL provides the required protocols for message exchange whereas; FIXON provides the required building blocks for semantic representation of the message content, being fixturing requirements, fixturing problem, partial design segments, or analysis results. Multi-agent technology together with ontologies, as two

emerging technologies, are envisioned to form a computational framework which represent a potential solution for knowledge management in distributed engineering design [6].

Since FIXON is developed in a lab scale, its current conceptualization is not comprehensive enough to cover needs of real-life industrial applications. For example, FIXON, in its current status, does not cover assembly and inspection fixtures. FIXON, like any other ontology, is a living entity that needs to be evolved in time as it is being used by dispersed users in real applications. As an envisioned area of extension, FIXON will be used for representation of modular fixture information. With ever increasing application of standard and modular fixturing systems in manufacturing industry, formal representation of standard fixturing elements, available in distributed libraries, is becoming an indispensable necessity for reuse of fixture design knowledge.

## 5. REFERENCES

- [1] Ameri, F.; Dutta, D.: An upper ontology for manufacturing service description, in Proceedings of ASME International Design Engineering Technical Conference, Philadelphia, Pennsylvania, October 2006.
- [2] Ameri, F.; Summers, J. D.: AFFIXED: A Multi-Agent System for Modular Fixture Design, Third International Conference on Design Computing and Cognition, Georgia Institute of Technology, Atlanta, GA, June 2008.
- [3] Baader, F.; Nutt, W.: Basic description logics, in The description logic handbook: Theory, implementation, and applications, Cambridge University Press, 2003, 43–95.
- [4] Bausch, J. J.; Youcef-Toumi, K.: Computer planning methods for automated fixture layout synthesis, Paper presented at the Proceedings of Manufacturing International '90, 1990, 225-232.
- [5] Cecil, J.: Computer-aided Fixture Design - A review and future trends, International Journal of Advanced Manufacturing Technology, 18, 2001, 790-793.
- [6] Chira, C.: An agent-based approach to knowledge management in distributed design, Journal of Intelligent Manufacturing, 17, 2006, 737–750.
- [7] Durfee, E.; Lesser, V.: Negotiating task decomposition and allocation using partial global planning, Distributed Artificial Intelligence, vol. 2, 1989, 229–244.
- [8] Finin, T.; Fritzson, R.; McKay, D.; McEntire, R.: Kqml as an agent communication language, in Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94), Gaithersburg, MD, USA, ACM Press, 1994, 456–463.
- [9] FIPA Agent Communication Language, The Foundation for Intelligent Physical Agents, <http://www.fupa.org>.
- [10] Gruber, T. R.: Toward principles for the design of ontologies used for knowledge sharing, International Journal of Human-Computer Studies, 43(5-6), 1995, 907-928.
- [11] Kopena, J. B.; Regli, W. C.: Design repositories on the semantic web with description logic enabled services, In First International Workshop on Semantic Web and Databases, 2003.
- [12] Kulvatunyou, B.; Ivezic, N.; Wysk, R. A.; Jones, A.: Integrated product and process data for business to business collaboration, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 17, 2003, 253–270.
- [13] Nanda, J.; Thevenot, H. J.; Simpson T. W.; Kumara, S. R. T.: Exploring semantic web technologies for product family modeling, in Proceedings of ASME DETC04, October 2004.
- [14] Pehlivan, S.; Summers, J. D.: A review of computer-aided fixture design with respect to information support requirements, International Journal of Production Research, 2006, 1-19.
- [15] Roy, U.; Liao, J.: A neural network model for selecting machining parameters in fixture design, Integrated Computer-Aided Engineering, 3(3), 1996, 149-157.
- [16] Senthil Kumar, A.; Subramaniam, V.; Seow, K. C.: Conceptual design of fixtures using genetic algorithms, International Journal of Advanced Manufacturing Technology, 15(2), 1999, 79-84.
- [17] SPARQL Query Language for RDF, <http://www.w3.org/TR/rdf-sparql-query/>, 2006.
- [18] Udoyen, N.; Rosen, D. W.: Description logic representation of finite element analysis models for automated retrieval, in Proceedings of ASME DETC06, September 2006.
- [19] Uschold, M.; Gruninger, M.: Ontologies: Principles, methods and applications, Knowledge Engineering Review, 11(2), 1996.
- [20] Yuan Y.; Liang, T. P.; Zhang J.: Using agent technology to support supply chain management: Potentials and challenges, in Supply Chain Transformation in the eBusiness Environment: Issues, Solutions and Future. First Annual Symposium on Supply Chain Management, Toronto, Canada, 2003.