An ontology framework for developing platform-independent knowledge-based engineering systems in the aerospace industry

I.O. Sanya & E.M. Shehab

Taylor & Francis
Taylor & Francis Group

An ontology framework for developing platform-independent knowledge-based engineering systems in the aerospace industry

I.O. Sanya and E.M. Shehab*

Product and Service Innovation Centre, Manufacturing and Materials Department, Cranfield University, Bedfordshire, UK

This paper presents the development of a novel knowledge-based engineering (KBE) framework for implementing platform-independent knowledge-enabled product design systems within the aerospace industry. The aim of the KBE framework is to strengthen the structure, reuse and portability of knowledge consumed within KBE systems in view of supporting the cost-effective and long-term preservation of knowledge within such systems. The proposed KBE framework uses an ontology-based approach for semantic knowledge management and adopts a model-driven architecture style from the software engineering discipline. Its phases are mainly (1) Capture knowledge required for KBE system; (2) Ontology model construct of KBE system; (3) Platform-independent model (PIM) technology selection and implementation and (4) Integration of PIM KBE knowledge with computer-aided design system. A rigorous methodology is employed which is comprised of five qualitative phases namely, requirement analysis for the KBE framework, identifying software and ontological engineering elements, integration of both elements, proof of concept prototype demonstrator and finally experts validation. A case study investigating four primitive three-dimensional geometry shapes is used to quantify the applicability of the KBE framework in the aerospace industry. Additionally, experts within the aerospace and software engineering sector validated the strengths/benefits and limitations of the KBE framework. The major benefits of the developed approach are in the reduction of man-hours required for developing KBE systems within the aerospace industry and the maintainability and abstraction of the knowledge required for developing KBE systems. This approach strengthens knowledge reuse and eliminates platform-specific approaches to developing KBE systems ensuring the preservation of KBE knowledge for the long term.

Keywords: knowledge-based engineering; ontology; platform independent; model driven

1. Introduction

Within the aerospace industry, the use of Knowledge-based engineering (KBE) methods and technologies has played a major role in automating routine and mundane design activities in view of supporting the cost-effective and timely development of a product. Developing these knowledge enabled KBE systems usually requires considerable effort in capturing, formalising and codifying knowledge. It has been established within the literature that 80% of design engineers activities is related to repetitive, routine and mundane tasks, while the remaining 20% on innovative tasks (Skarka 2007). However, applying KBE systems could significantly reduce repetitive tasks (80% automation) and allow product design engineers to gear their focus towards innovative design activities.

However, the design and implementation of multidisciplinary KBE systems in the aerospace sector are usually platform specific and domain dependent. Due to the nature of specificity within KBE systems, knowledge reuse becomes an issue because the 'design' and 'implementation' of KBE systems are often tied down to a specific technological platform. In software engineering, a platform-independent knowledge model is a model of a business or software system that is independent of the specific technological platform used to implement it. The notion of a platform-independent model (PIM) is often used in the context of a model-driven architecture (MDA) approach. The idea is to use a transformation language to transform a platform-independent model into a platform-specific model (PSM). In software engineering, the advantages of platform-independent models are separation of business/functionality logic from implementation logic. However, this approach is often not employed in the design and development of KBE systems. For example, the design and development of a fan blade product component KBE system could be implemented in KF (Knowledge Fusion: automation language for KBE system) (PLM world 2014) or in NX (Open KBE automation technology) (Siemens NX 2014) in the aerospace industry. Though these technologies possess powerful KBE functionalities that exploit advanced engineering knowledge, they are still considered as PSMs because its business and implementation

---

logic is specific to a technological platform. These types of KBE systems are usually implemented in a specific programming language (i.e. Java, C#, C++) which means its key design parameters, design rules, design constraints and mathematical expressions are buried in code. Therefore, if there were to be a platform change, this will mean a complete re-write of the KBE software system. This ad hoc approach to developing KBE systems significantly limits abstraction and reusability of engineering knowledge. Additionally, the maintenance of KBE systems developed in a platform-dependent style often demands a considerable amount of man-hours due to the code-intensive and procedural programming approach often employed. The future generation of KBE systems will adopt a platform-independent approach in order to ensure the key knowledge required for a KBE system is created independently of the KBE implementation as illustrated in Figure 1.

Using a model transformation language will enable the transformation of PIMs into PSMs. This approach will significantly introduce a higher level of modularity and reusability in the design and development of KBE systems. Thus, reusing design and manufacturing knowledge from one KBE system to another becomes feasible. This research project is in collaboration between an aerospace company and Cranfield University. The focus of this study is to investigate approaches for developing KBE systems that are platform independent, well structured and offers high level of knowledge reuse. For this purpose, ontology-based approaches for modelling the design parameters and design rules for KBE systems are employed. Additionally, declarative rules coupled with inference engines capable of backward and forward chaining are exploited and preferred than the procedural rule base often employed in developing KBE systems.
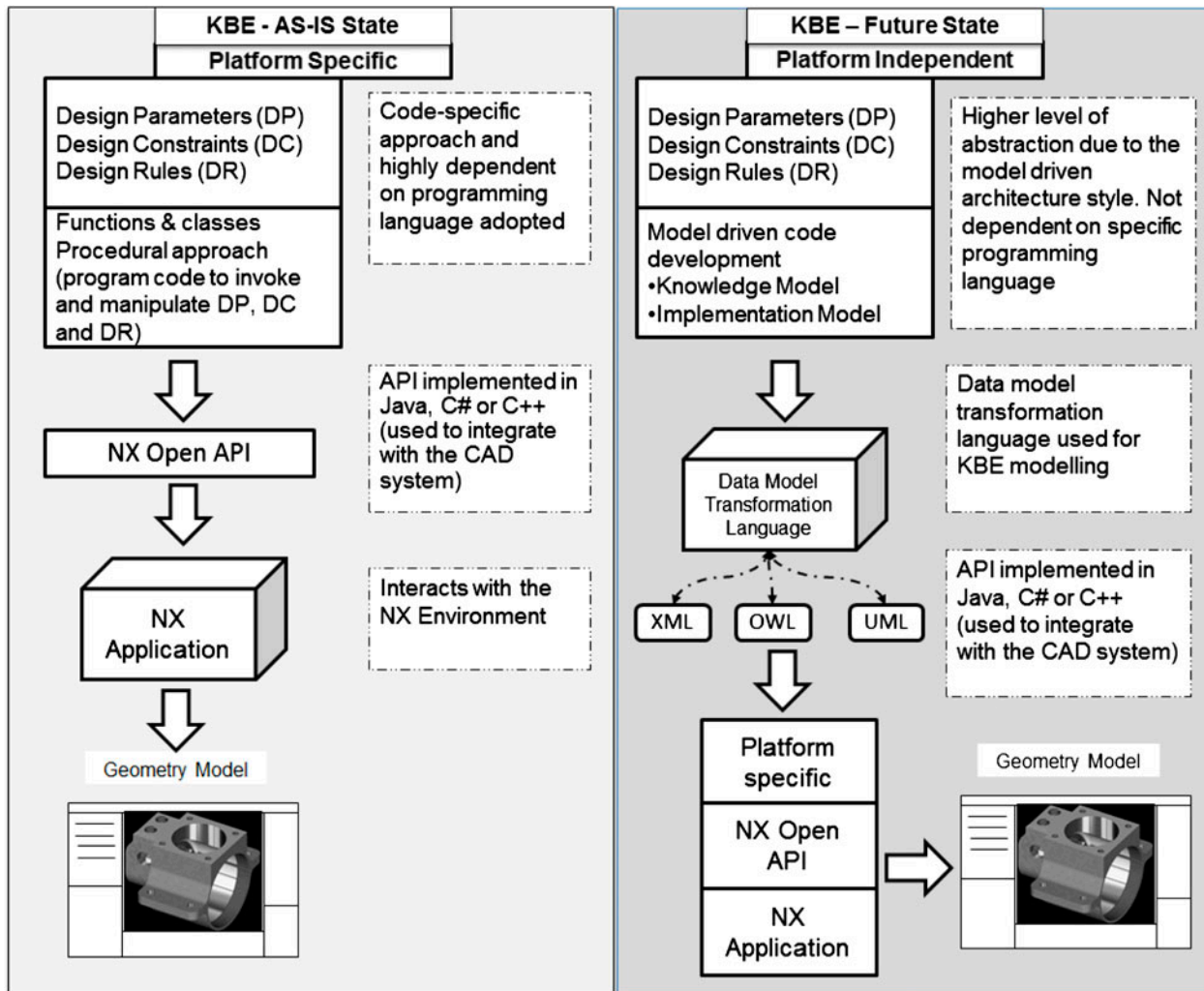

Figure 1. PSM in comparison to platform-independent KBE model in the aerospace industry.

Furthermore, particular attention is given to exploiting application programming interfaces (APIs) as a means of a communication and integration medium between KBE software components.

2. Literature review and research scope

KBE systems are the 'use of advanced software technologies to capture and reuse product and process knowledge in an integrated way' (Stokes 2001). Skarka (2007) identified that knowledge representation within KBE goes past or beyond representing geometrical data and other factors should be considered.

There are two common modelling frameworks for structuring and representing engineering knowledge developed in EU projects to support knowledge-based systems. These are namely Common Knowledge Acquisition and Design Support (CommonKADS 2009) and Methodology and tools oriented to Knowledge-Based Engineering Applications (MOKA) methodology (Stokes 2001). With the CommonKADS methodology, knowledge-based systems are developed in a 'structured, controllable and repeatable way' (Schreiber 2000). However, CommonKADS lacks specialisation to engineering design, therefore making it a generic-purposed framework. Prior to the MOKA framework, there was no common approach for collecting, structuring and formalising engineering knowledge associated with product design. This made updating and reusing modules within KBE systems impossible. MOKA was developed to fill this need. However, MOKA only focuses on the 'knowledge capture' aspect of the knowledge life cycle and often needs to be extended to cover other aspects of the product lifecycle engineering process (Barreiro et al. 2010).

Furthermore, Ma and Liu (2007) have identified that many applications of KBE systems are usually domain specific and ad hoc which signifies specific solutions designed for solving specific design challenges. Further discussions in the literature have highlighted the shortcomings regarding the maintenance and excessive cost of managing several KBE applications and integrating with other IT applications. There is a need to standardise the internal knowledge representation of KBE applications and interfaces with product development solutions (i.e. PLM, CAx). In order to fill in this gap, there has been recent advancement in the development of KBE systems using ontology-based approaches. In recent years, the engineering community (researchers and industry experts) has developed increased interest in the development and maintenance of engineering ontologies to support engineering activities. Ontologies (Davies, Fensel, and van Harmelen 2008; Gruber 1993; Mizoguchi 2003) are a formal specification of a domain which seeks to classify entities and relations that ties them together. Kitamura and Mizoguchi (2007) identified that one of the most promising uses of ontologies is that it enhances knowledge systematisation. This systematisation is mainly about the structure and reuse of knowledge using modelling-based techniques. Recently, some research work has been reported to use ontologies to create semantic-based applications within the aerospace sector (Dadzie et al. 2008), government sector (Horrocks 2008) telecommunication sector (Davies, Fensel, and van Harmelen 2008), medical and pharmaceutical sector (Hawker 2010) manufacturing sector (Kitamura and Mizoguchi 2007; Lin et al. 2010; Mizoguchi 2004; Sanya, Shehab, and Roy 2010) automotive sector (Liang 2012) and recently the PSS (product service system) sector (Doultsinou et al. 2009; Shen, Wang, and Sun 2012). Despite these advances, there has been lack of KBE frameworks for developing ontology-based multidisciplinary KBE systems within the aerospace industry. Shehab et al. (2013) identifies the importance of ensuring enhanced cross- product information collaboration and access within large organisations such as the aerospace industry. Tsai, Sun, and Huang (2006) presents a collaborative web-based XML information sharing system for collaborative product development. However, there was lack of semantics with the information exchange due to the XML-based nature. Kyoung-Yun, Manley, and Hyungjeong (2006) present a collaborative assembly design framework supported by an ontology. The Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL) ontology-based modelling techniques were employed for representing assembly knowledge. The main advantage of this approach is in the collaborative development of products amongst design and other collaborators. Limitations of this approach are in the security and maintenance of such knowledge models. Additionally, there was no framework/methodology presented in maintaining and reusing the developed assembly ontologies.

More recently, Saa et al. (2012) present an ontology-based modelling approach for developing cost-optimised design of complex railway applications. An intelligent Computer Aided Design (CAD) tool named SIA (Automatic Identification System) was presented. Production rules using the OWL and SWRL ontology-based modelling techniques were employed to govern the railway design process. The ontology-based CAD tool is currently being used by railway engineers in ADIF (Administrator of Railway Infrastructure) to develop real-life railway portals. However, the integration medium amongst the related KBE software components was not reported in detail. Additionally, there was no mention of a framework that provides methodological support for KBE experts to develop KBE systems capable of such functions.

Furthermore, the literature revealed the use of a design realisation system using knowledge models (Gunendran and Young 2009) for capturing manufacturing best practice knowledge. The E2KS commercial of the shelf (COTS) ontology

knowledge-based tool was used to demonstrate a KBE prototype. One of the key strengths in this work was in the modelling and formalisation techniques adopted (i.e. Unified Modelling Language (UML)-2 and Object Constraint Language). However, a framework/set of guidelines was not provided on how KBE experts can develop platform-independent KBE systems using the E2KS knowledge base. Furthermore, Chungoora and Young (2010) present a heavy weight ontological foundation for formalising and increasing the expressiveness of knowledge models within KBE systems using Common Logic. The approach was validated using the XKS-high fleet commercial application. However, the authors suggest a limitation of the approach is the lack of APIs as a communication medium between domain ontologies and software applications, thus ensuring direct interaction between the ontology knowledge base and CAD system through a user interface. Additionally, there was no mention on how to develop platform-independent KBE systems using ontologies. Recently, model-driven approaches to support manufacturing system interoperability have been proposed in the literature (Chungoora et al. 2013; Fortineau, Paviot, and Lamouri 2013; Giovannini et al. 2012). These approaches offer advanced capabilities in the design and development of KBE systems. However, they are mainly focused on modelling manufacturing engineering knowledge; design engineering knowledge is often neglected.

Furthermore, Verhagen et al. (2012) suggest in a recent review of KBE research challenges that in order to advance KBE research and development, it is essential to move away from case-based approaches (mostly currently reported in the literature) to methodology guided approaches. Existing KBE methodologies are subjected to improvement and alternatives are required to be developed (Mendikoa et al. 2006; Ruschitzka, Suchodolski, and Wróbel 2010; Sandberg 2003; Verhagen et al. 2012). Additionally, transparency of KBE applications is required in order to move away from black-box KBE applications to more user-friendly, adaptable, structured and reusable knowledge bases (Fan and Bermell-Garcia 2008; Sunnersjö et al. 2006). In a recent critical review (Agyapong-Kodua et al. 2013) of semantic modelling approaches and technologies, it was identified that KBE frameworks should clarify pre-development and post-development phases for KBE development which is often not well specified in KBE systems that uses ontologies as the basis. The literature review has identified that most attempts to link semantic-based techniques to KBE systems often lack a common framework/set of activities to adopt. Furthermore, the notion of a platform-independent knowledge model using ontologies as the basis in KBE systems is often not clear. The scope of this research is to investigate semantic-based approaches and develop a framework that encompasses activities for KBE experts to employ in order to develop structured, reusable and platform-independent KBE systems. This will aid and support the elimination of platform-dependent approaches.

3. Research methodology

This section defines a five-phased structured research methodology approach employed in the development of a model-driven, ontology-based, platform-independent KBE system as presented in Figure 2. The project scope and boundaries were first defined through a number of initial interviews with the main stakeholders. The methodology adopted is mainly qualitative and an applied research approach is employed to ensure KBE experts within the aerospace industry were involved in the development of the KBE framework. In the first phase, requirements elicitation activities were conducted in order to elicit functional and non-functional requirements for the KBE framework. Semi-structured interviews were utilised as the main approach for knowledge capture. Four requirements capture sessions were conducted which lasted over an hour each. A workshop was then used to validate the captured requirements. Experts involved in the requirement capture and validation sessions includes KBE design specialists, KBE product designers, integrated design and manufacture engineers and technical KBE architects. Due to the exploratory nature of the study, the questions were written as open-ended questions to encourage meaningful responses from the expert's own knowledge. The following are examples of the questions addressed in these sessions:

- What are the practical challenges faced in developing and managing KBE systems in the aerospace industry?
- What are the functional requirements of the KBE framework?
- What is the quality attributes of the KBE framework?
- What is the 'business value' for using a platform-independent KBE framework?

The second, third and fourth phase of the research methodology identified techniques and approaches from the software and ontological engineering elements and integrates them both in order to align the captured requirements with specific capable techniques and methods from the software and ontological engineering disciplines. Finally, phase five of the research methodology comprises a proof of concept KBE demonstrator using primitive three-dimensional geometry shapes. Four types of shapes were investigated including a panel shape, sleeve shape, L-beam geometry flange and T-beam geometry flange. Additionally, experts' opinion from the software engineering and aerospace KBE domain were
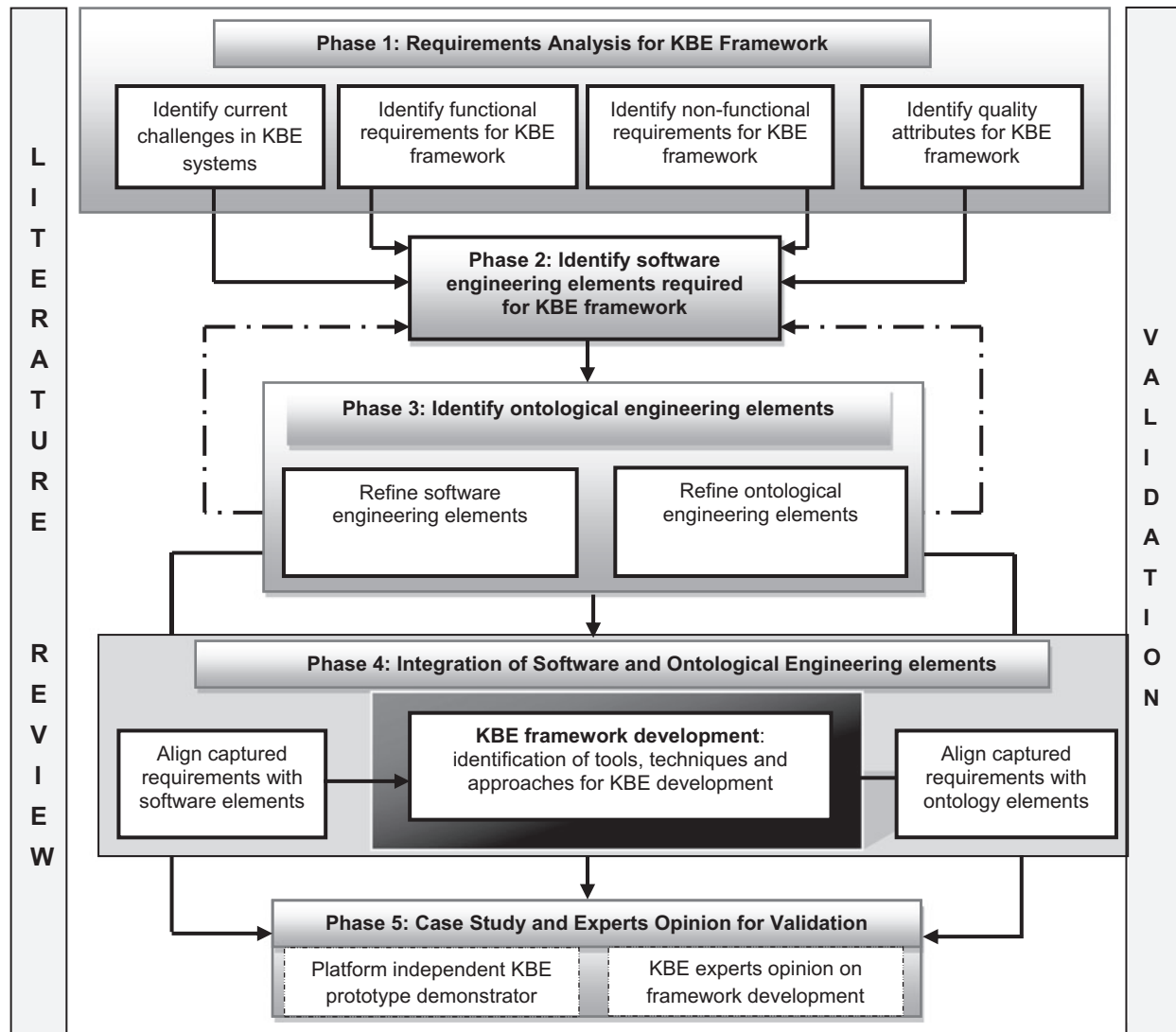
Figure 2. Adopted research methodology for model-driven, platform-independent KBE system development.

captured regarding the platform-independent KBE framework. Thus, key strengths/opportunities and limitations/threats of the KBE Framework were identified.

4. Requirements analysis for KBE framework development

Functional requirements in software engineering describe the function/behaviour of a system and its related components. Functional requirements are often technical and may describe mathematical expressions, technical processes and other functionalities that define the output of the system. It was identified in the requirement elicitation sessions that the development of KBE systems within the aerospace sector is usually geared towards fulfilling functional requirements. Little is often done to fulfil the non-functional requirements of a KBE system. In systems engineering, non-functional requirements specifies criteria used to assess the operation of a system. Non-functional requirements are often referred to as quality attributes of a system and are usually linked with the architecture of a system. Execution and evolution qualities of a system such as security, maintainability, extensibility, scalability, adaptability, portability (*platform independent*) and usability are essential elements of a system. However, these are often not fully addressed and considered in the development of KBE systems in the aerospace industry. The captured requirements for the KBE framework development are assessed against ontology representation languages, general modelling languages, rule-based languages and inference engines and ontology APIs to assess aerospace KBE requirements evaluation criteria.

Table 1 illustrates the evaluation of aerospace KBE design requirements against formal ontology representation languages. It was identified that OWL and OCML provides rich expressiveness for representing domain knowledge and allows for concept definition, taxonomy development, instance creation, single and multiple inheritance, complex relationship definitions, independent and importable knowledge models. RDF(S) and XML were also considered for modelling the KBE domain-knowledge. However, these languages lack formal semantics and the ability to assert and derive facts from a knowledge base which is crucial for modelling a KBE domain. The ability to create instances was also missing from the OIL, DAML and XML ontology representation languages which limit the expressiveness of a knowledge base. Thus, the selection of an ontology representation language for the KBE framework is based on the fulfilment of the evaluated requirement criteria. OWL and OCML were selected as suitable ontology-based languages for formalising a KBE domain.

Table 2 illustrates the evaluation of aerospace KBE design requirements against general modelling languages. The purpose of adopting a generic-purposed modelling language is to crystallise the problem domain before imposing technological elements to the KBE domain. It was identified that the MDA and domain-driven design (DDD) are key elements that should be considered when attempting to develop platform-independent KBE systems. Key difference between the MDA and DDD is that MDA suggests using UML for capturing the problem domain while DDD does not propose a specific modelling technique. However, the intent of both modelling approaches is the same. Data modelling techniques is crucial for describing data required for a KBE system. Therefore, the use of UML, OOP and EXPRESS proved useful in fulfilling this requirement. Process knowledge is also another vital knowledge source required for KBE system development. BPMN and IDEF3 were identified as suitable process modelling techniques for eliciting process knowledge. UML was partly considered for KBE process modelling due to the activity diagram provided by the UML suite; however, in regards to process modelling, BPMN was preferred over UML activity diagram as the strength of UML lies in data modelling and BPMN is an industrial-standard technique for modelling business processes. The UML notation was also selected as it supports the agile software development approach. The combination of MDA, DDD, OOP, UML and BPMN were selected as suitable generic-purposed modelling techniques for formalising the KBE

Table 1. Evaluation of aerospace KBE requirements criteria against ontology representation language.

Requirements criteria	OWL	RDF(S)	XML	OIL	DAML	OCML	SHOE
Model domain knowledge	+	+	+/−	+	+	+	N/R
Structured knowledge base	+	+	+	+	+	+	+
Rich expressiveness and formalisation of domain knowledge	+	−	−	+/−	+/−	+	N/R
Taxonomy	+	+	+	−	+	+	+
Encourages modularity in ontology design	+	+	−	+/−	+/−	+	N/R
Instance creation	+	+	−	N/R	N/R	+	+
Importable and independent knowledge models	+	+/−	+	+/−	+/−	+	+
High-level of knowledge abstraction	+	+/−	−	+/−	+/−	+	N/R

Notes: +, element supports the requirement criteria; −, non-supported requirement; +/−, 'partly supports' a requirement criteria; ?, 'situation where information is not available'; N/R, not required'; OWL, Ontology Web Language; SWRL, Semantic Web Rule Language; OCML, Operational Conceptual Modelling Language; RDF, Resource Description Framework; XML, Extensible Markup Language; OIL, Ontology Inference Layer; DAML, DARPA Agent Markup Language; SHOE, Simple HTML Ontology Extensions; CLIF, Common Logic Interchange Format; ECLIF, Extension Common Logic Interchange Format; Flogic, Frame Logic; BPMN, Business Process Modelling Notation; UML, Unified Modelling Language; OOP, Object Oriented Paradigm; MDA, Model Driven Architecture; DDD, Domain Driven Design; API, Application Programming Interface; GUI, Graphical User Interface; JESS, Java Execution System Shell.

Table 2. Evaluation of aerospace KBE requirements criteria against general modelling languages.

Requirements criteria	MDA	DDD	OOP	UML	EXPRESS	IDEF0	IDEF3	BPMN
Modelling problem-domain	+	+	+	+	+	+	+	+
Expressive notation for representing domain-knowledge	+	+	+	+	+	+/−	+/−	+/−
Data modelling	+	+	+	+	+	+/−	−	−
Modelling of design process	N/R	N/R	−	+/−	−	−	+	+
Supports agile software development	+	+	+	+	+	−	−	+

problem domain before technology implementation. However, the focus of the case study was on modelling data, algorithms and geometry constraints for a KBE system. Therefore, the modelling of KBE process knowledge is not illustrated in the case study described in this paper.

Rule-based modelling is critical for developing knowledge enabled design systems. Table 3 evaluates aerospace KBE design requirements with rule-based languages and inference capabilities. Key aerospace design requirements include separation of business logic from implementation logic, declarative programming, backward and forward chain inference reasoning capabilities. SWRL, Flogic, object logic, CLIF and ECLIF all offer these capabilities. However, SWRL is designed to naturally integrate closely with OWL which increases the expressivity of OWL and makes it possible to model rules that integrates directly with concepts and relations defined in the OWL model. Whereas, in other rule languages (e.g. JBoss and Java Expert System Shell (JESS)) the business model is required to be mapped to another language that the rule engine can compile and this increases the maintenance of such production rules. Additionally, SWRL rules can also be used for validation purposes and used to question and reason about a domain in interesting and different ways. However, constraint checking in SWRL is not possible but available in Flogic and object logic rule languages. Nevertheless, SWRL is web-enabled and a commercial declarative rule-based language which could potentially support the collaborative development and distribution of KBE systems on the web. It was also identified that the JESS inference engine works closely with the SWRL rule base.

Table 4 assesses aerospace KBE requirements criteria with ontology APIs. The selection of an ontology API is based on the function and purpose of its application. Regarding KBE design and development, an ontology API capable of exploiting and integrating the ontology knowledge base with a CAD environment was required. The aerospace KBE design requirements for an API include excellent documentation, scalability and extensibility of ontology API and common coding standard. The Jena API and OWL-API was considered as suitable APIs as they are both popular ontology APIs used in the ontology engineering community. However, the Jena API was considered more appropriate due to the amount of accessible documentation of practical demonstrations of using the API to exploit ontologies in interesting ways.

5. Justification of KBE framework components

The proposed components of the KBE framework is justified in relation with a typical KBE lifecycle as illustrated in Figure 3. A meeting was conducted within the aerospace industry with a KBE technical lead/manager, KBE specialist

Table 3. Evaluation of aerospace KBE requirements criteria against rule-based languages and inference engines.

Requirements criteria	Object logic	SWRL	Flogic	CLIF	ECLIF	OCML	JBoss Drools	JESS	Onto-Broker
Reusable design rules and design parameters	+	+	+	+	+	+	+	+	+
Separation of business logic from implementation logic	+	+	+	+	+	+/−	+/−	+	+
Model operational knowledge	+	+	+	+	+	+	+	+/−	+/−
Integrates closely with domain-knowledge	+/−	+	+/−	?	?	+/−	?	+	?
Adaptable production rules	+	+	−	+	+	+	+	+	+
Inference reasoning capabilities	+	+	+	+	+	+	+/−	+	+
Backward chaining	+	+	+	+	+	+	−	+	+
Forward chaining	+	+	+	+	+	+	+	+	+
Constraint checking	+	−	+	+	+	+	?	?	+
Declarative programming	+	+	+	+	+	+	+	+	+
User interface	+	+	?	+	+	+	+	+	+

Table 4. Evaluation of aerospace KBE requirements criteria against ontology APIs.

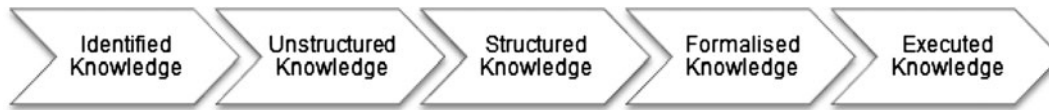| Requirements criteria	Jena API	OWL-API	Alignment API	GATE API
Excellent documentation of API	+	+/−	+/−	+/−
Scalability of API	+	+	?	?
Extensibility of API	+	+	+	+
Semantic model for API	+	+	+	+
Common coding standard	+	+	?	?

Figure 3.   KBE system life cycle in the aerospace industry.

and chief design systems architect to identify the main sources of knowledge required for a KBE system. It was identified that five main sources of knowledge exists for KBE system development. These are namely (1) geometry constraints; (2) engineering process; (3) data types; (4) algorithms/design rules and (5) written word. The KBE lifecycle initiates by eliciting raw unstructured knowledge from three categories of knowledge source: human, document and repositories. The ontology-based KBE framework proposes a standard feature template rather than the conventional MOKA ICARE forms (Stokes 2001). MOKA ICARE forms/templates are generic to KBE applications, thus it often needs to be tailored to a specific product design task. Industrial requirements suggest the need to develop customisable standard templates for various product design disciplines (e.g. design, analysis, manufacturing, inspection) in order to support the capture of product knowledge and aid the transformation of unstructured knowledge into structured knowledge.

The structured knowledge captured within the standard feature template is then transformed into formalised knowledge. In addition, a key aerospace requirement for a KBE system is encouraging modularity in its design and development. The use of an ontology and MDA approach is proposed for the purpose of formalising structured knowledge. One of the most promising uses of an ontology is that it enhances domain knowledge systematisation. The function of an ontology is useful in structuring the characteristics of a product (e.g. product structure, product design parameters, product constraints) and incorporating object orientation and model-driven design encourages modularity in KBE system development.

A critical aspect of KBE system development is the rule base. An essential KBE industrial requirement was the need to create rules that are adaptable and reusable than the procedural approach to rule-based modelling. The declarative approach to developing rules was viewed as an instrumental component in the design and development of future KBE systems in the aerospace industry. Thus, the ontology-based KBE framework needed to address declarative rule-based modelling as an integral element. Once knowledge is fully formalised, it is executed within a CAD environment. A specific aerospace requirement was to distinguish between KBE design knowledge from KBE implementation knowledge and to encourage interaction of various KBE components. Therefore, the use of an API is introduced as a core component in the ontology-based KBE framework. The introduction of ontology APIs and CAD APIs will support the specification and interaction of KBE software components.

6. Framework development

A framework for developing platform-independent KBE systems using an ontology-based approach has been developed in the aerospace industry. The framework consists of four-phase as illustrated in Figure 4, namely (1) Capture KBE system knowledge; (2) Ontology Model Construct of KBE system; (3) Platform-Independent Model (PIM) Technology Selection and Implementation and (4) Integration of PIM KBE knowledge with CAD system. The KBE framework describes a set of activities for KBE experts to employ in order to move from a platform-specific approach to a platform-independent approach. The first phase is to capture the knowledge required for the KBE system. This includes clearly identifying the purpose of the KBE system, defining 'use case' scenarios, including functional and non-functional system functionalities. A knowledge capture template sheet is recommended in order to capture specific knowledge required for the KBE system. The outcome of this phase should be a clear identification of design intent, key design parameters, key design constraints and key design rules required for implementing the KBE system. The second phase focuses on formalising the captured knowledge into an ontology model for the KBE system. It is essential to identify a list of KBE concepts, KBE properties and define concept to concept relationships as well as concept to properties relationships. There is also a need to identify specific instances of the concepts identified. It is recommended to employ the OOP and MDA approach for this phase. Additionally, visualising and graphical representation of the KBE system ontology meta-model is essential in this phase. One of the most important aspects of a KBE system is the rule base. Therefore, there is a need to develop rules that are more maintainable and adaptable. A rule-based approach that provides such features is known as 'declarative rules' rather than the commonly used 'procedural rules'. In declarative programming, the rules are specified by defining what the programme should accomplish rather than describing how to go about
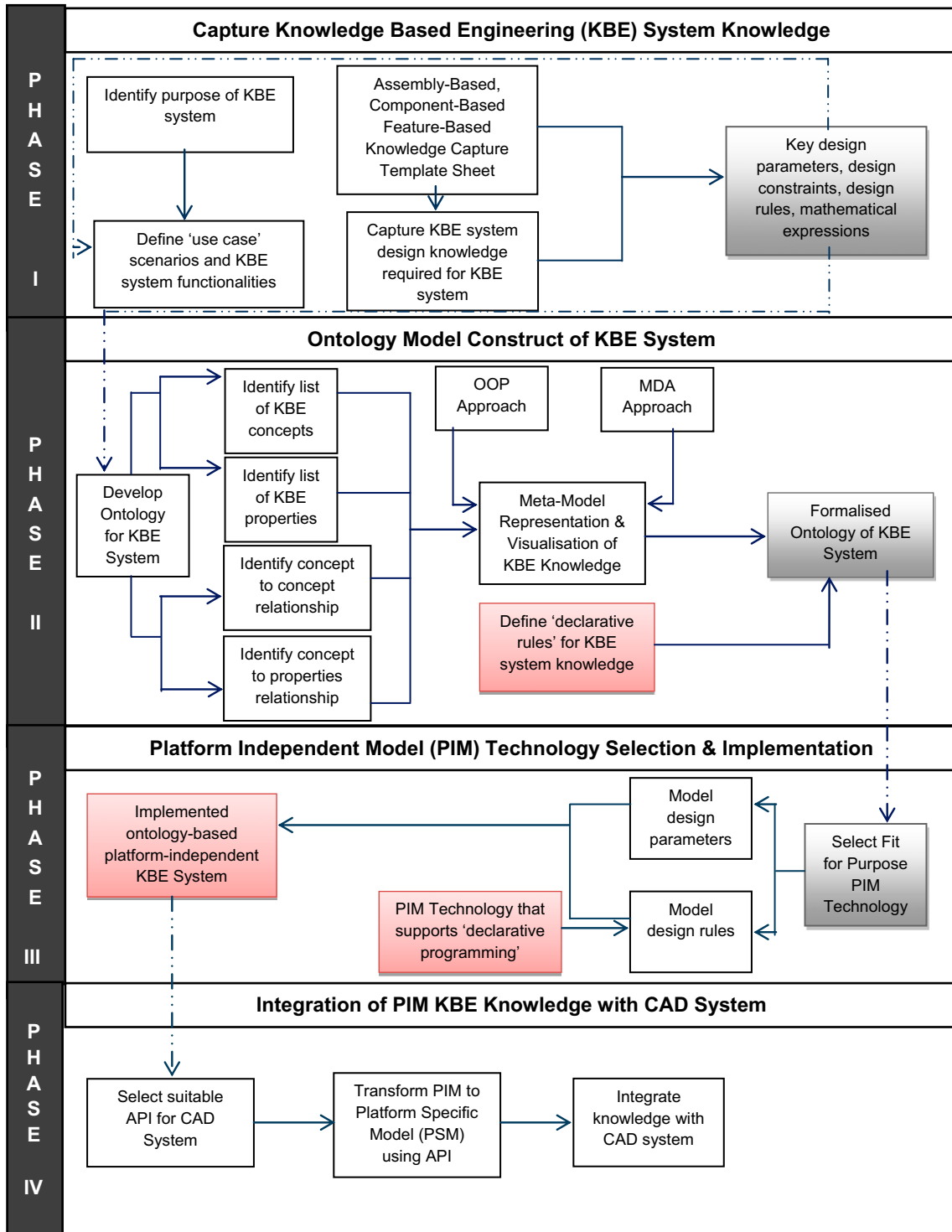
Figure 4.   Framework development for implementing platform-independent, ontology-based KBE systems.

accomplishing it. Therefore, the logic of the computation is expressed without describing its control flow. The declarative rule-based approach to defining KBE rules are more adaptable and maintainable than the procedural approach as this can replace hard to maintain nested (IF … THEN … ELSE) coding commonly adopted in KBE systems. The outcome of phase two is a formalised-ontology, clearly representing the KBE design parameters and design constraints, rule

base (as a set of declarative rules) as ontology concepts and properties using the OOP and MDA approaches. The third phase of the framework focuses on selecting a 'fit for purpose' platform-independent technology and developing a technological architecture that reflects the 'portability' KBE system quality attribute. Though there are not many platform-independent technologies that support KBE systems, there are capable ontology-based technologies that can be used for this purpose. It is essential to select a technology that supports declarative programming in order to model the KBE design rules using this approach. The KBE ontology model developed in the previous phases is implemented using the appropriate ontology-based technology. There might be a need to employ two ontology-based languages in this phase, the first for modelling KBE domain knowledge (i.e. design parameters as a set of concepts and attributes) and the other for modelling key design constraints and rules for the KBE system.

The fourth phase is to select an API for the CAD system. A suitable API will enable the integration of the platform-independent ontology knowledge base with the KBE CAD system. The input and output of the ontology-knowledge base model is integrated with the KBE CAD system using a suitable API. It is important to note that the last phase might be time-consuming due to the need to understand the API. Platform-independent ontology-based models might require more time to implement than the conventional platform-specific approach. However, the benefit of these PIMs is realised in knowledge abstraction and reusability. This is of course essential for the future of KBE systems and for maintaining and preserving KBE knowledge for the long term. Furthermore, in CAD graphics, the term primitive geometry is used to denote a simplistic representation of geometric objects that a CAD system can handle. The most primitive of CAD drawings are points, straight line segments and more complicated curves. A case study investigating the KBE system development of four primitive geometry shapes using the KBE framework presented is used to illustrate a proof of concept demonstrator. The use of ontologies, model-driven software engineering techniques and object orientation is employed to develop such a demonstrator. Four main primitive feature shapes were considered for this case study, these are 'Panel', 'Sleeve (i.e. cylindrical shape)', 'L-beam flange' and 'T-beam flange.
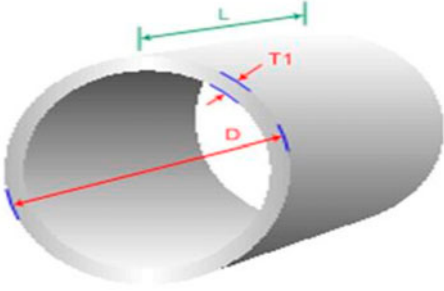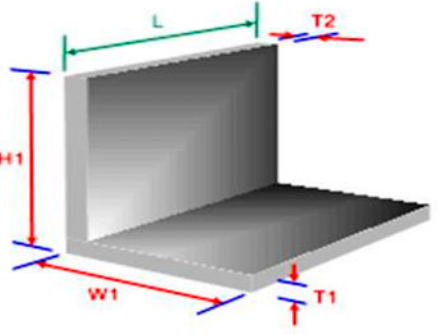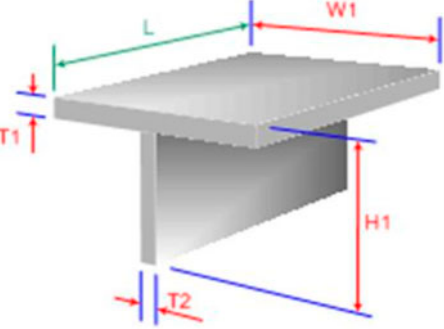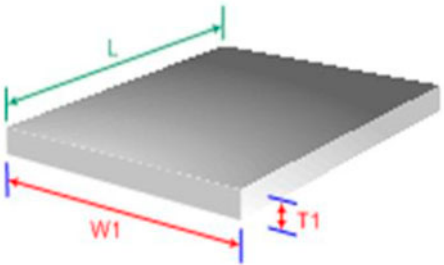
6.1 *KBE system knowledge capture*

In order to capture product knowledge required for a KBE system, a standard feature template was designed with a KBE system specialist with over 22 years of aerospace engineering experience. The knowledge capture template was validated with other KBE specialists in the aerospace organisation. The standard feature template portrays specific information regarding the product component features of a KBE system. Details include the feature name, type of feature (e.g. simple, compound or patterned), category of feature, design drawings, engineering references/drawing number, description of feature, functionality of feature within the context of the product component, key design parameters with examples, design rules and design constraints were identified as important elements in the capture of feature specific knowledge. However, the main elements of feature specific knowledge for a KBE system are the feature's key design parameters, design constraints and key design rules. Therefore, specific emphasis was made to capture this knowledge in order to demonstrate the notion of an ontology-based, platform-independent KBE system. Table 5 presents a summary of the key design parameters and rules for four main feature shapes considered for this study: 'Panel', 'Sleeve', 'L-beam flange' and 'T-beam flange'.

6.2 *An overview of the adopted ontology method*

Several ontology methods exists as reported in the literature, e.g. Uschold and King's (1995), METHONTOLOGY (Fernandez-Lopez and Gomez-Perez 2002), Noy and McGuinness (2001), Pinto and Martins (2004) and Ahmed, Kim, and Wallace (2007). However, for the development of the ontology-based PIM, the Noy and McGuinness (2001) and Ahmed, Kim, and Wallace (2007) ontology methods both offered useful techniques when applied to engineering design. Their methodologies have been adopted and tailored for the purpose of this research study. The Noy and McGuinness (2001) ontology method encouraged the iterative development of engineering design ontologies which proved suitable due to the iterative nature of product design activities. The Ahmed, Kim, and Wallace (2007) ontology method is one of the only ontology method tailored towards engineering design and has been quantified in two aerospace companies. Thus, its suitability to be employed in the context of platform-independent KBE systems. The adapted ontology method steps followed to construct the ontology employed in this research study is as follows: (1) Identification of concepts; (2) Taxonomy and hierarchical classification of product design concepts; (3) Specification of product design instances; (4) Definition of product design geometry properties and (5) Definition of semantic relations between concepts. The next section describes aspects of the ontology method used. However, the focus of this paper is to define an approach for developing platform-independent KBE systems within the aerospace industry.

Table 5. Summary of key design parameters and design rules for primitive geometry shapes.

Feature shapes	Key design Parameters	Key design rules
	• Length (L) • Diameter (D) • Thickness 1 ($T1$)	IF sleeve THEN Periphery = $2 * \pi * D$ Cross section area = $\pi * ((D * T1) - T1^2)$ Raw volume = *finished volume* Developed area = *length* $* (D * \pi)$ Finished volume = *crossSecArea * overall length* Part height = D Part width = D Part thickness = $T1$
	• Length (L) • Thickness 1 ($T1$) • Height ($H1$) • Width ($W1$) • Thickness 2 ($T2$)	ELSE IF L-beam flange THEN Periphery = $2 * (length + W1 + (H1 - T1))$ Cross-section area = $((W1 * T1) + ((H1 - T1) * T2))$ Raw volume = *finished volume* Finished volume = *crossSecArea * overall length* Developed area = *partWidth * length* Part height = $H1$ Part width = $W1 + (H1 - T1)$ Part thickness = $((W1 * T1) + ((H1 - T1) * T2))/W1 + (H1 - T1))$
	• Length (L) • Width 1 ($W1$) • Thickness 1 ($T1$) • Height 1 ($H1$) • Thickness 2 ($T2$)	ELSE IF T-beam flange THEN Periphery = $2 * (length + W1 + (H1 - T1))$ Cross-section area = $((W1 * T1) + ((H1 - T1) * T2))$ Finished volume = *crossSecArea * L* Raw volume = *finished volume* Developed area = *partWidth * length* Part height = $H1$ Part width = $W1 + (2 * (H1 - T1))$ Part thickness = $((W1 * T1) + ((H1 - T1) * T2))/W1 + (H1 - T1))$
	• Length (L) • Diameter (D) • Thickness 1 ($T1$)	ELSE IF Panel THEN Periphery = $2 * (length + W1)$ Cross-section area = $W1 * T1$ Raw volume = *finished volume* Developed area = *length* $* W1$ Finished volume = *crossSecArea * L* Part height = $T1$ Part width = $W1$ Part thickness = $T1$

6.3 *Ontology model construct of KBE system*

A model is used to represent or describe an entity or a system via only defining aspects considered relevant and fit for purpose (ISO 10303-214 1997). Phase 3 of the KBE framework focuses on using OOP constructs and model-driven approach to enforce KBE design decisions and intent and support model-driven applications. The object-oriented and model-driven approaches are common in the software and IT discipline. However, these approaches are not often employed in the design and development of KBE systems. The procedural approach is commonly used for developing these knowledge-intensive KBE systems as identified in the aerospace organisation.

In order to ensure the OOP paradigm is utilised in the ontology development of KBE systems, the three main elements of OOP (i.e. inheritance, encapsulation and polymorphism) should be addressed as presented in Figure 5.
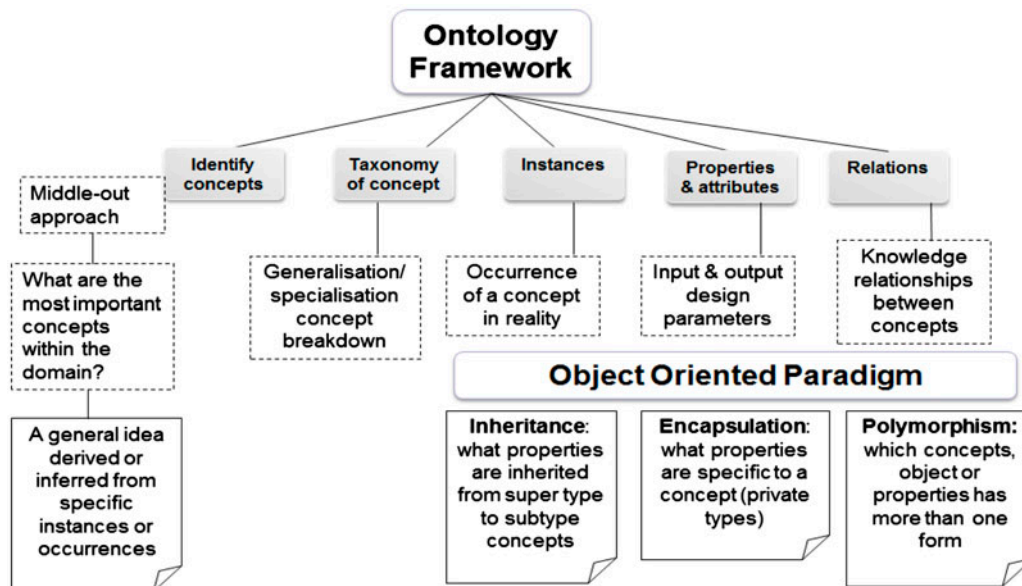
Figure 5. OOP paradigm in ontology development for KBE systems.

Inheritance provides a powerful mechanism for structuring software and elucidates how a child class can be derived from a parent class. Encapsulation embeds the state and values of structured data object within a specific class which prevents unauthorised access to data that has been encapsulated. In the case of a KBE system, there are design rules that should only be specific to a type of product component feature. Employing the use of encapsulation can significantly maintain the security and integrity of these rules in the development of KBE systems. Furthermore, polymorphism in OOP is the ability to develop a function, object or an attribute that has more than one form. A classic example of implementing a polymorphic behaviour in KBE systems is to consider the calculation of an area of a two-dimensional surface or shape. A parent class 'shape' is created and two classes (e.g. rectangle and circle) are derived. However, the area method for both rectangle and circle are calculated differently, thus their functionalities are implemented separately even though they belong to the same method and class. Elements of the OOP are essential to ensure the structure, reuse and maintainability of KBE systems. Additionally, taxonomy (considered as lightweight ontologies) is introduced to support product breakdown structures as Mereology is more critical to component definition than ontology.

Furthermore, the use of declarative rules should be employed for the development of the rule base for KBE systems rather than using the conventional procedural/imperative approach. A declarative approach to developing KBE systems rule base strengthens the reusability and maintainability of KBE systems. A MDA approach is a type of software design developed by the Object Management Group (OMG) that provides a set of guidelines for expressing software models which are constructed as specifications. The most important aspect of MDA is to eliminate technology enthralment and focus primarily on modelling the problem domain. Most KBE applications are often technology specific and platform dependent. Therefore, little focus is emphasised on modelling and formalising the KBE problem domain. Architectural and technological components are important. However, they are subservient to the modelling of a KBE problem domain. The MDA approach supports a model where the names of classes, attributes, functions are derived from the problem domain to establish a DDD. UML is a modelling approach that is used to support a MDA. It also supports the visualisation and representation of domain knowledge. The notion of this phase of the framework development is to clearly understand and crystallise the problem domain without polluting it with technological architectural concerns. The outcome of this phase is a 'pure', 'transparent' and 'crystallise' problem domain model ready to be converted into a complete software system.

Figure 6a presents a UML Meta model representation of part of the product ontology. The focus of the UML Meta model is on the product ontology. The other classes represent ontology concepts developed for other purposes. Many product ontology already exists in the literature (Gimémez et al. 2007; Guo et al. 2003; Panetto, Dassisti, and Tursi 2012). Product design ontologies to support system interoperability within manufacturing enterprises and supply chain (Fortineau, Paviot, and Lamouri 2013; Lu et al. 2013; Patil, Dutta, and Sriram 2005; Tursi et al. 2009) have been proposed in the literature. In addition, product design ontologies for representing the complexity of information for the

product lifecycle management (PLM) are also defined and established in the literature (Fenves et al. 2008; Terzi et al. 2007; Vegetti, Henning, and Leone 2005; Zdravkovic and Trajanovic 2009). Thus, aspects of existing product design ontologies have been adopted for this study. A high-level generic product ontology is defined as one product having many assemblies, one assembly having many components, one component is composed of many features and one feature has one or many shapes. However, the contribution of the product ontology employed in this study is that it has been tailored towards the development of gas turbine engine products as depicted in Figure 6b. Therefore, concepts such as engine design style and engine type are introduced as well as specific instantiation and attributes of each class in relation to a gas turbine engine product. As illustrated in Figure 6b, one aerospace gas turbine product has many engines (e.g. Airbus A350 aeroplane has 2 Trent XWB engines). One engine has many assemblies (e.g. turbines, combustor, fan, compressor, etc.) and one assembly has one design style (e.g. combustor assembly will have a phase 5 rich burn or lean burn design style). One assembly will have many components (e.g. combustor assembly has a fuel injector seal which is associated to a specific type of material) and one component will have many features (e.g. fuel injector seal has a fuel injector seal cooling hole. Lastly, one feature will have many shape types (e.g. fuel injector seal cooling hole has a cylindrical shape type). The properties associated to the identified concept instances are captured in the ontology model. The concept shape has an attribute 'shapeDescription' to differentiate between various types of shape sub-concepts developed in the Meta model construct.

6.4 *Model-driven transformations and specifications*

Figure 7 illustrates the approach taking to develop model-driven, platform-independent KBE systems in relation to the hierarchy of models defined within the MDA approach. The MDA approach stems from the transformation of computation-independent models (CIM) to PIMs and PIMs to PSM. CIM defines high-level requirements that should be addressed by models developed at the PIM level. High-level functional and non-functional requirements have been explained in Section 4. Examples of such requirements includes a declarative, adaptable rule base and reusable product design rules and design parameters. In addition, product designers requires specific access to design resources, functional models, manufacturing, inspection and assembly process knowledge as all these elements influences the way a product is designed. These are described at the CIM level.

At the PIM level, a model that defines the software architectural framework solution from a technology independent view is formalised as depicted in Figures 6a and 6b. The core of the PIM is in the construction of product design ontologies. The concepts identified in the UML metamodel are developed to form part of the product ontology prior to rule-based modelling. UML was used as a notation for designing ontologies to be later formalised using a declarative rule-based approach. The concepts identified in the high-level requirements are further exploited and developed. The UML model representation captures essential segments of product design information. These include the design and development of concepts relating to product, assembly, component feature, material, functional models, shapes, design and analysis process, manufacturing methods and process, design and manufacturing resources, etc. At the PSM level, the model developed at the PIM level is deployed in the chosen technology platform. Examples of PSMs relevant to the purpose of this research study are the PLM environment suites; Teamcenter, NX and standard draw 3D (StdDraw3D). The methodology described in this research study adopts a combination of an ontology-based and model-driven approach towards the design and deployment of product design systems at the PSM level which is essential for the future development of KBE systems (Chungoora et al. 2013) in the aerospace industry. For product design users, PSM-level systems provides a decision-support mechanism that enables product engineers to interact directly with product knowledge bases as presented in the remainder of this paper.

6.5 *PIM technology selection and implementation*

After developing a crystallised model of the KBE problem domain, the next phase is to implement the model using a platform-independent technology. The KBE framework is mainly focused on modelling data types/structures, algorithms and geometry constraints in a platform-independent style. Selecting the most suitable platform-independent technology to implement the required knowledge for the KBE system is essential. The data types/structures are modelled in UML for representation purposes. However, the computational development of the design parameters for the KBE system is modelled in OWL which is a family of knowledge representation languages for developing ontologies (W3C standard). The algorithms and geometry constraints which are mainly the design rules for the KBE system is modelled in a platform-independent technology that supports declarative programming in order to separate 'design' (i.e. business logic) and 'implementation' knowledge. The SWRL is employed as a PIM technology for modelling the design rules required for the KBE system. SWRL is designed to integrate closely with OWL, hence the justification of the technology
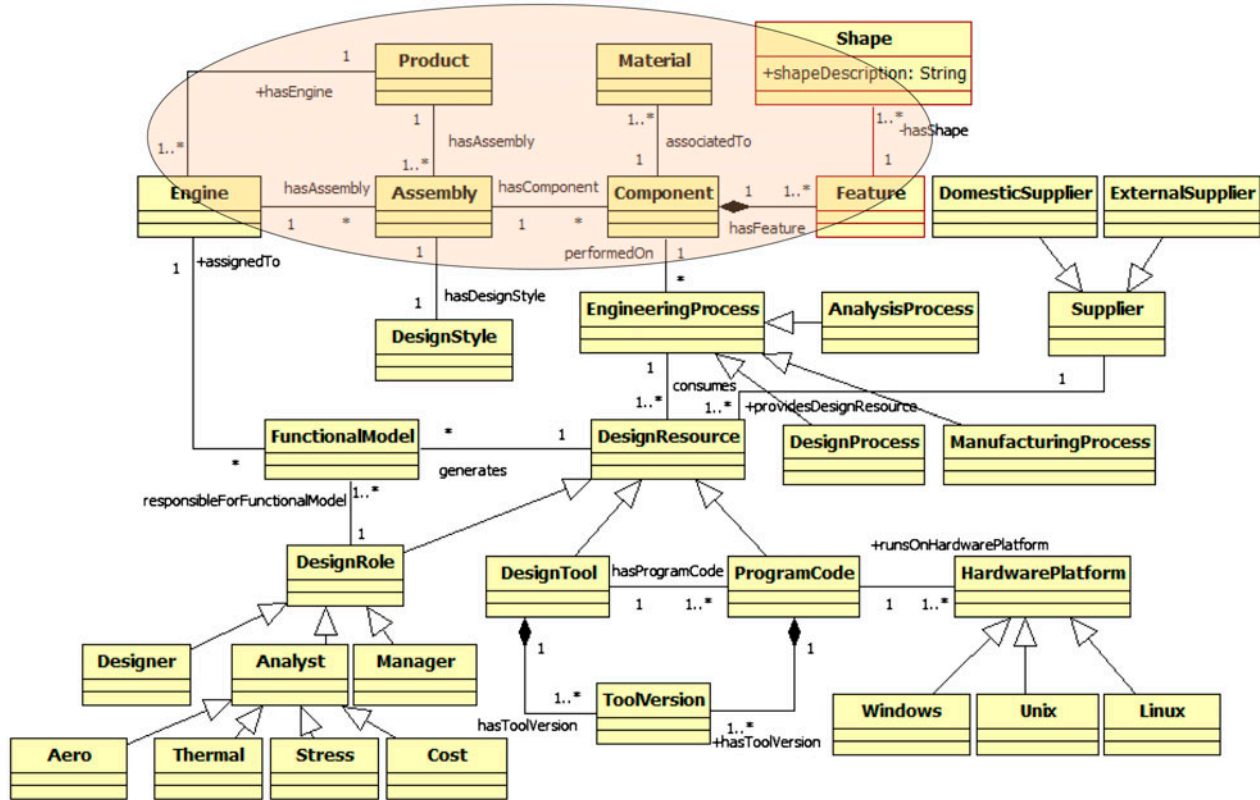
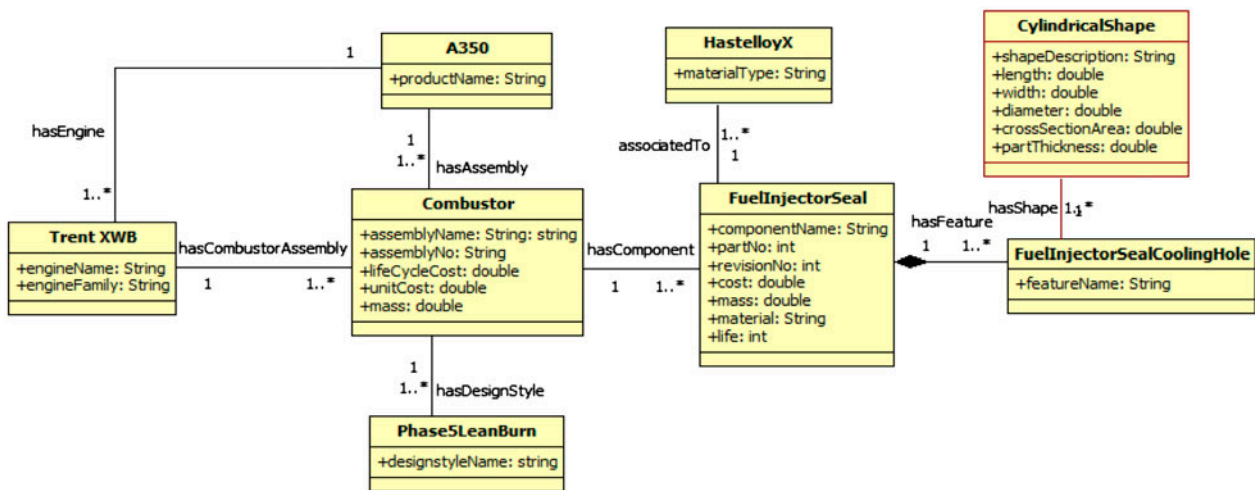Figure 6a. Meta model UML representation of part of the product ontology.

Figure 6b. Meta model UML representation of specific instances of the product ontology.

selection. SWRL increases the expressivity of OWL and makes it possible to model operational knowledge (i.e. rule-based knowledge). The SWRL approach to rule-based modelling can be used to compute outputs for production applications and for validation purposes. The KBE framework proposes the implementation of OWL and SWRL as the basis for developing PIMs for KBE systems as the two technologies supports rule-driven applications. SWRL is primarily a declarative rule-based language that is not bound to a specific execution algorithm. Instead, it uses a formal model-theoretic semantics which determines a complete and sound execution algorithm (MacLarty et al. 2009). One of
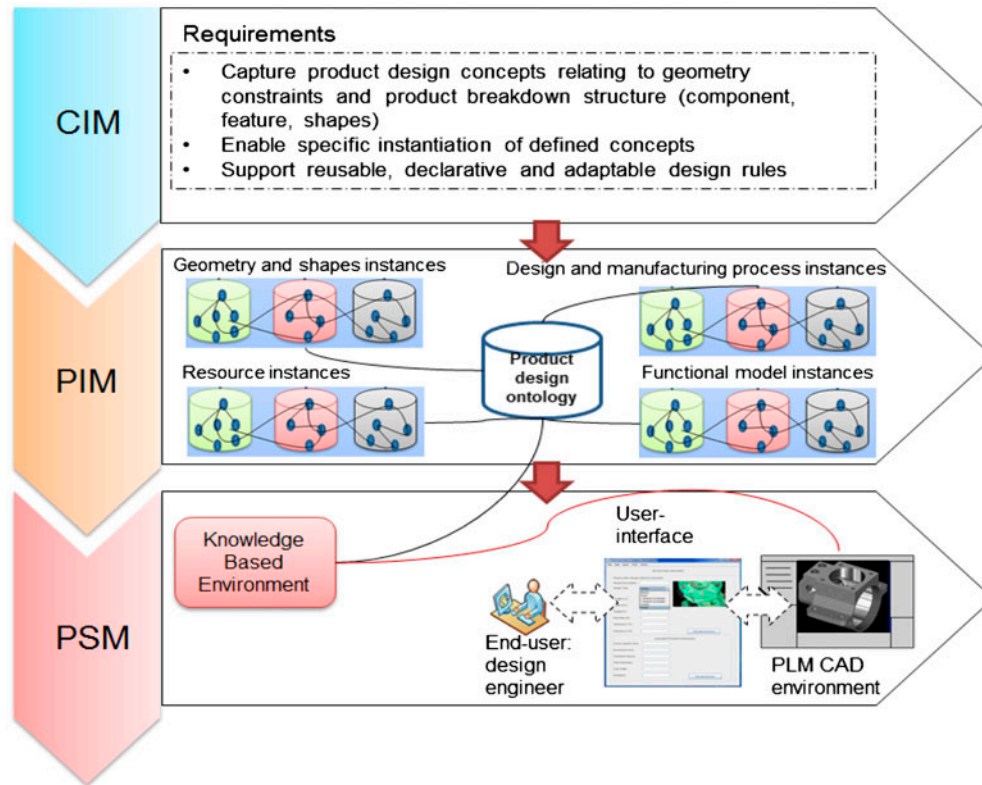
Figure 7. KBE framework development in relation to MDA hierarchy of models.

the main advantages of SWRL is the ability to truly separate business knowledge from technical IT knowledge as priorities can be assigned to rules which in turn determines the order and sequence in which they are executed. Another advantage is that SWRL rules are reusable in different and interesting ways due to its declarative modelling approach. Furthermore, extending the SWRL rule base with domain specific built-ins is achievable, thus, introducing extensibility and maintainability in the development of design rules for KBE systems. The performance and scalability of SWRL rules for real-world business applications has been verified in practical demonstration (MacLarty et al. 2009; Saa et al. 2012).

In conjunction with SWRL technology, the JESS is employed. JESS is a business rules engine used in a software system to execute business rules in a runtime production environment. JESS supports the new type of programming paradigm (procedural to declarative rule-based definitions) and as a result of employing a rules engine like JESS, the system as a whole becomes more adaptable with business rules (developed in SWRL) that can be changed dynamically. JESS supports both backward and forward chaining inference mechanisms which are methods that can be described as working backward or forward from the goal(s). These are often commonly used in artificial intelligence applications, theorem provers and other complex systems.

Once the KBE system design parameters and design algorithms are modelled using a platform-independent technology, the next phase is to use an API to integrate the KBE system knowledge with the CAD system. An API specifies how software components should communicate and interact with each other. In most cases, an API consists of software libraries that define specifications for object, classes, data structures, attributes, etc. Two APIs are advised to be used in this phase, the first is the ontology API and the second CAD API. Few APIs exists for ontology development. Common examples of these APIs are the Jena and OWL API. Criteria's to selecting an API should consider the following elements below:

- Sufficient and excellent documentation of API (*with practical illustrations*)
- Scalability of API
- Extensibility of API
- A semantic model that describes what the API does
  - Naming convention of classes, functions, objects, variables
- An API that follows a common coding standard

After considering the aforementioned elements, the Jena API was selected as the ontology API of choice due to the fact that it had rigorous documentation and it was well maintained than other identified ontology APIs. By employing the Jena Framework, the KBE system knowledge modelled in OWL and SWRL could be consumed, reused and exploited by other types of applications.

6.6 *Architecture of platform-independent KBE system*

Figure 8 illustrates the architecture of the ontology-based, platform-independent KBE system. As portrayed in the architecture diagram, the key design parameters and design geometry rules is constructed in a platform-independent meta-model. The OWL-based representation schema is employed to model the design parameters of the KBE system while the SWRL ontology-based language is used to model the design rules of the KBE system. Additionally, the JESS inference engine is employed to process the developed rules. Through the ontology API, the knowledge contained in the ontology can be exploited and manipulated for use in any programming language and on any software platform and environment, thus emphasising the platform-independent nature of the KBE architecture. Furthermore, the standard draw 3D API is used to integrate the ontology knowledge with the CAD API classes. In addition, a java-based graphical user interface is also developed for product designers to enter their design intent to enable the automated creation of an instance of the geometry model.

6.7 *Modelling of KBE design parameters and design rules*

This phase of the KBE framework models the KBE design parameters for the primitive geometry shapes described in Table 5. The concept 'shape' is defined in the product design model using the protégé 3.5 platform. As specified in the UML model in Figure 6a, the relationship 'hasShape' between concept 'feature' and 'shape' is defined using the object
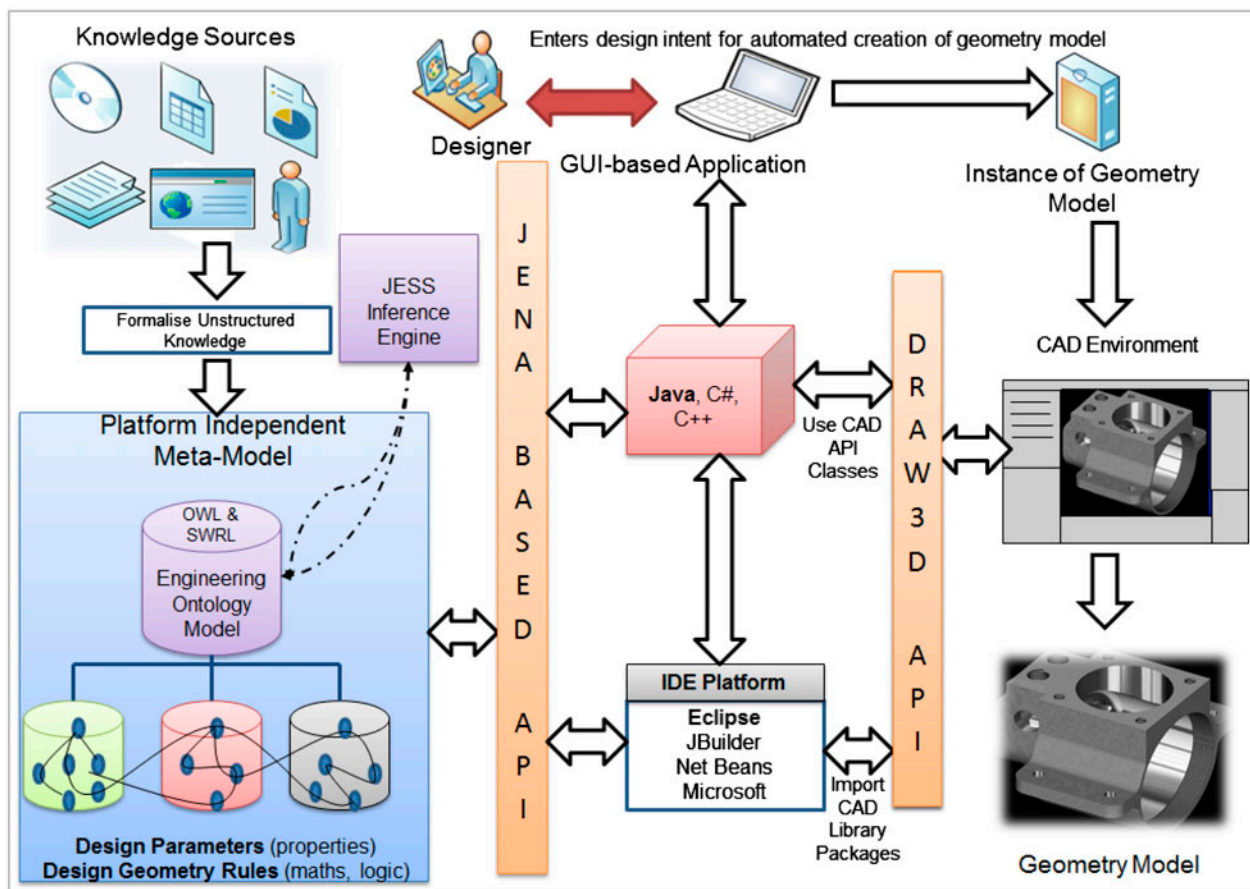

Figure 8.   Ontology-based, platform-independent KBE system architecture.

property. Additionally, sub-classes of concept shape are developed as well as key data properties and data types (e.g. double, integer and string) for the KBE system. These include design parameters such as cross section area, developed area, thickness1, periphery and other properties required for the primitive shapes KBE system. Furthermore, instances of the shapes cylinder-'sleeve', rectangle-'panel', rectangle-'I-beam flange' (*i.e.* 2 *rectangles joint* via *edges*) and rectangle-'T-beam flange' (2 *joint rectangles* via *the middle*) are defined within the geometry ontology model. Furthermore, existencial rules for each of the geometry concepts defined are specified as a predicate using the OWL and SWRL rule-based modelling language. For example, the OWL expression shown below defines an interaction between an inverse relationship and domain and range constraints on a property:

> Individual(a: Cylinder type(owl:Thing)
>
>   value(a:is_shape_of a:FuelInjectorSealCoolingHole))
>
> Individual(a: FuelInjectorSealCoolingHole
>
>   type(owl:Thing))
>
> ObjectProperty(a:hasShape domain(a:feature) range(a:shape))
>
> ObjectProperty(a:is_shape_of inverseOf(a:hasShape))

The expression illustrates that a cylinder is a shape of the fuel injector seal cooling hole. Therefore, the fuel injector seal has a shape cylinder. This infers and asserts that a fuel injector seal cooling hole must be a feature and a cylinder must be a shape.

After modelling the design parameters for the KBE system using the OWL syntax, the next activity conducted was to model the design rules using the SWRL declarative rule base language. The KBE design rules captured in each primitive shape (sleeve, panel, L-beam flange and T-beam flange) is modelled using the SWRL technology. The process of executing the rule base is as follows: OWL > SWRL > JESS > OWL. The OWL KBE design parameters and SWRL rule base is converted into the JESS rule engine knowledge. To illustrate, 24 rules, 14 OWL class declarations, 4 OWL instance declaration (representative of each primitive shape) and 35 OWL data properties has been exported to the rule-engine. After running the JESS rule-engine, the rules are executed (as long as the SWRL syntax is modelled correctly) and inferred back to the OWL model.

An example of a core geometry design rule for a T-shaped rectangle is expressed below. This rule expresses that for all shape that has a shape description of 'panel rectangle', its finished volume is calculated by multiplying the length property of the panel rectangle instance with the calculated cross section area. The previous being the first aspect of a hypothetical proposition also known as the antecedent, the following implication of such a rule expression is defined in the consequent using the → symbol in SWRL which assigns the result of the rule execution to the property of the finished volume of the panel rectangle instance.

> shapeDescription(?x, 'panel rectangle') ∧ length(?x, ?y)
> ∧ crossSectionArea(?x, ?z) ∧
> swrlb:multiply(?panelVolume, ?y, ?z) →
> finishedVolume(?x, ?panelVolume))

Another rule-based expression used to calculate the cross section area of a cylindrical shape with an instance of sleeve is defined in SWRL below. This expression states that for all shape description that has an instance of 'sleeve cylinder', get its diameter and multiply by the value of thickness 1. Then subtract the resulting value with thickness 1 squared and multiply by 3.14 (pi). The consequent and implication of this geometry rule is then assigned to the cross section area for the instance sleeve. Evidently, SWRL supports predefined built-ins using predicates i.e. mathematical expressions, constraints and IF/ELSE logic. It is also possible to develop predefined functions which can be reused by the KBE expert.

> shapeDescription(?x, 'sleeve cylinder')
> ∧ diameter(?x, ?y) ∧ thickness1(?x, ?z) ∧
> swrlb:subtract(?diaThick, ?y, ?z) ∧
> swrlb:multiply(?var1, ?diaThick, ?z) ∧ swrlb:multiply(?sleevearea, 3.14, ?var1) →
> crossSectionArea(?x, ?sleevearea))

Figure 9 illustrates an example of a more complex rule used to express KBE product design part thickness rule for the T-beam flange shape developed in SWRL. The rule is only executed if the shapeDescription is 'T-shaped Flange'. If otherwise, the rule is not executed when imported into the JESS rule engine. The result of the invoked rule is associated to the data property 'part thickness' of the relevant shape which is inferred back in the OWL model.

A disadvantage of SWRL is in its computational complexity as more resources in terms of time and storage are consumed while rules execute. This indicates that the more complicated a rule is the more time and storage is required for rule execution. Though this is a well-known issue of SWRL and developers are currently addressing this. In the context of geometry creation and for SWRL to be fully employed in KBE systems, advanced pre-defined mathematical expressions are required. Mathematical expressions such as standard deviation, calculus and other geometrical math-based formulas are needed for complex rule definitions. These expressions will be required in practical KBE system development for gas turbine engine products.

6.8 *Integration of PIM KBE knowledge with CAD system*

Standard Draw 3D (CAD package) API was selected in view of demonstrating a proof of concept as there was a complete reference manual for the API and it possessed a wide variety of product design features and functionalities. The object oriented java programming language and the Eclipse software environment has been selected as the language and platform of choice to integrate the platform-independent KBE knowledge with the CAD system. The Jena-based ontology API was used to exploit the design parameters and rule-base for the KBE system. Additionally, the standard draw 3D API was imported into the Eclipse environment to enable the integration of KBE knowledge with the standard 3D drawing packages (i.e. java library classes). This phase of the KBE development required code-intensive activities. Using the import statements supported by the java programming language, the KBE ontology knowledge and standard drawing 3D packages were imported and the link between both API's were established syntactically. Prior to this, a java-based GUI (graphical user interface) was developed as illustrated as one of the key elements in the KBE system architecture (Figure 8). The KBE java-based GUI in Figure 10 enables designers to enter their design-intent (i.e. input design parameters) for the geometry model primitive shapes. This enabled direct exploitation of the ontology model with the CAD system through the API as identified as a limitation in previous work done in this area (Chungoora and Young 2010).

Figure 11 illustrates the development of the cylindrical geometry shape using the input design parameters entered by the product designer. Once the button 'calculate dimensions' is clicked, the design rules are invoked and the product ontology is used to automate the creation of engineering geometry based on the design intent and calculated dimension parameters. The instance of the geometry model representing the input and output design parameters is displayed in the CAD design space environment.
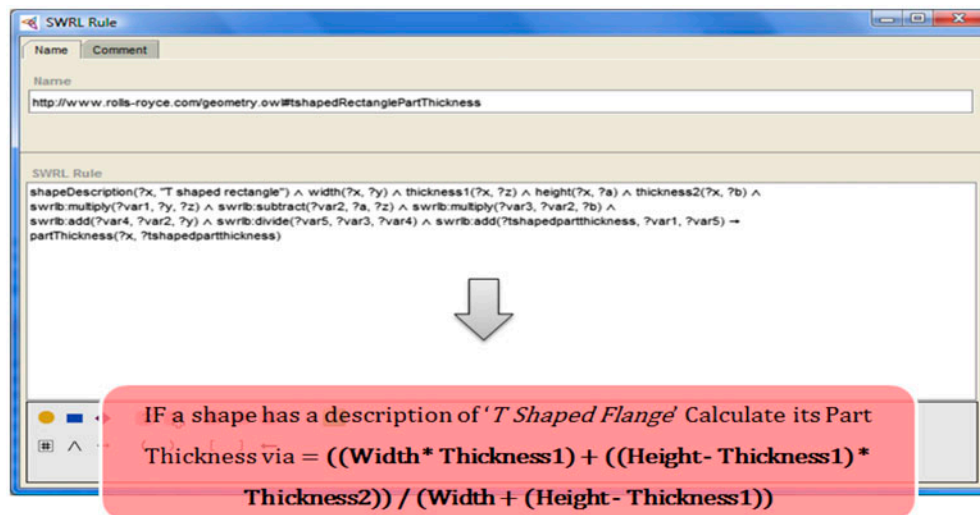

Figure 9. KBE product design rule for part thickness T-beam flange shape.
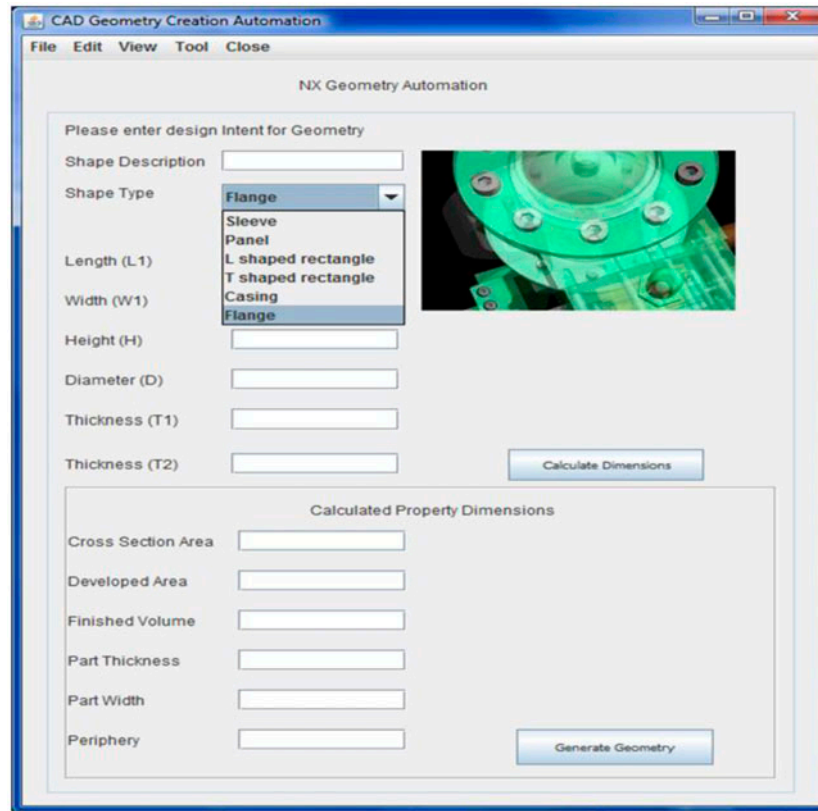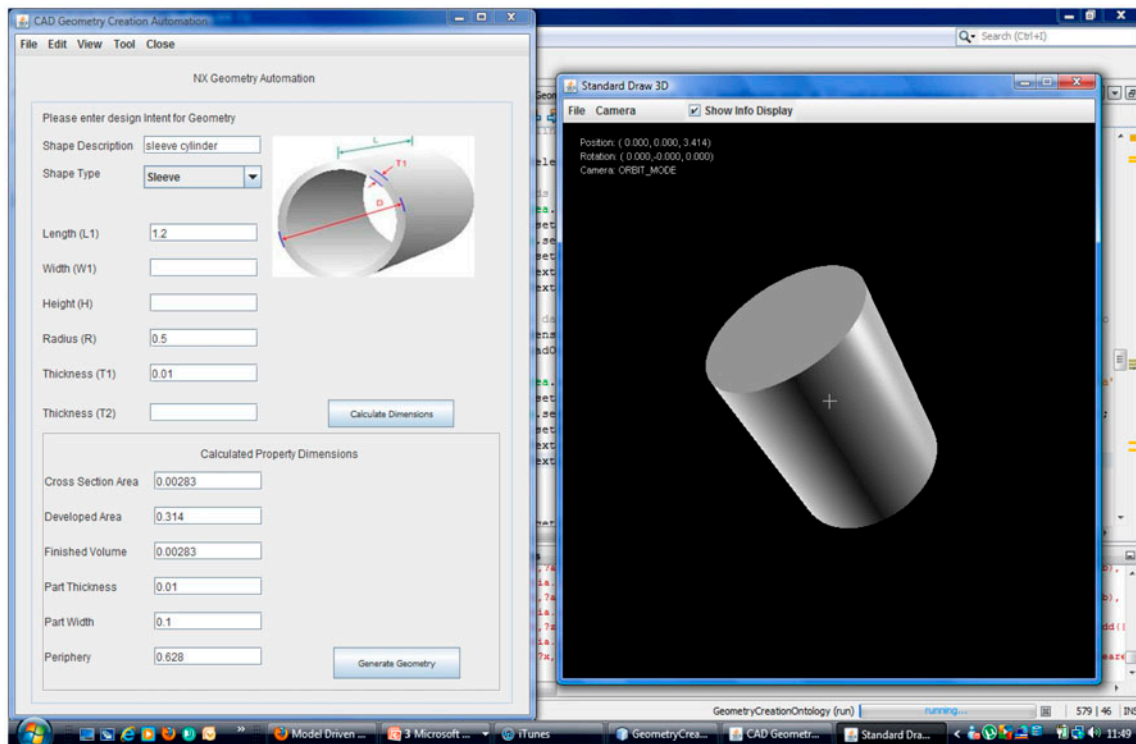
Figure 10.   KBE java-based graphical user interface.

Figure 11.   Automated KBE engineering geometry for sleeve shape.

7. Validation and benefits

The validation completed for the KBE framework is mainly qualitative in nature with quantitative aspects. Throughout this study which has been conducted with an aerospace organisation, every utilised method, technique and approach was validated systematically with experts within the aerospace industry in order to ensure reliable results and recommendations. A workshop was conducted in which the framework was presented to 9 experts from aerospace and software engineering sectors. Participants include KBE technical lead/manager, KBE design specialist, KBE product designer, semantic technologist/architect, technical architect lead, software developers and aerospace product design experts. All experts possessed several years of experience in KBE system design and development. Key performance metrics for the KBE framework was assessed by aerospace experts, namely (1) generalisability of framework; (2) scalability of framework; (3) reusability of rule base; (4) maintainability of rule base; (5) cost-savings; (6) nature of logic and (7) portability quality attribute. Excerpts of the questions addressed in this workshop are as follows:

- Please assess the generalisability of the KBE framework?

Example of scale used to capture this: 8 in this case mean the KBE framework is 80% generic from an expert perspective. This scale has been used to assess the remaining performance metrics. The experts also provided additional comments on each performance metric.

1		2	3	4	5		6	7	8		9	10
Not Generic			Generic with major issues				Generic with minor issues				Fully Generic	

- Please assess the scalability of the KBE framework?
- Will the framework reduce cost of maintaining practical KBE systems?
- How does the framework support the portability quality attribute of a KBE system?
- Does the framework provide practical solutions in developing KBE systems that are platform independent in the aerospace industry?

Table 6 provides statistics for the outcome of experts' answers in the workshop, including how many experts were involved, their individual assessment of each performance metric and average (%)/overall assessment of answers to the KBE evaluation metrics for the model-driven, platform-independent KBE framework.

Experts involved in the workshop suggested that the KBE framework approach towards developing ontology-driven platform-independent KBE systems is 76.4% generic (i.e. generic with minor issues) and could potentially be adopted in developing practical KBE systems in the aerospace industry. KBE specialists emphasised that the KBE approach has illustrated the capture of business logic of a KBE system at the appropriate level of abstraction which is important for

Table 6. Validation of KBE framework using key performance KBE metrics identified by aerospace experts.

No	Experts	Validation performance assessment criteria						
		Generic (%)	Scalable (%)	Reusable (%)	Maintainable (%)	Cost reduction (%)	Logic (%)	Portable (%)
1	KBE technical lead	87	45	70	89	50	90	80
2	KBE design specialist	80	55	82	91	60	80	90
3	Product designer	70	40	72	70	42	70	75
4	Semantic technologist	77	33	70	75	30	79	70
5	Product designer	67	50	78	80	40	70	60
6	Software developer	75	50	80	80	30	70	80
7	Technical architect lead	82	30	70	75	20	72	70
8	Senior design specialist	80	40	75	70	35	80	70
9	Semantic technologist	70	50	75	80	20	80	82
	Average	76.4	43.6	74.6	79	36.3	77	75.2

the management and maintenance of KBE systems. However, there is a need to apply this approach on a larger scale to validate its scalability (i.e. complex geometry shapes) as the overall expert average for the KBE framework scalability performance is 43.6% which suggests the scalability of the KBE framework needs to be quantified within practical KBE systems. How would the technologies used within the KBE framework handle growing amount of complexity? Within the aerospace industry, KBE systems are required to cope with thousands of design rules in view of developing quality aerospace products. Therefore, there is a need to adopt technologies that are scalable and extensible. Additionally, one of the key strengths of this approach is in the definition of declarative rules which are more adaptable and maintainable than the procedural programming approach. Additionally, the separation of business logic and implementation logic is a key element in moving towards more maintainable, reusable KBE systems. This is established in the software engineering discipline but often not employed in KBE systems. The KBE framework places sufficient emphasis on this approach as agreed by experts. The experts agreed with the use of OO and model-driven concepts to help promote reuse of knowledge and map knowledge to KBE systems. The aerospace design experts in the workshop suggest that the tools and techniques of the KBE framework are highly reusable (74.6%) and the approach will ensure KBE knowledge maintainability (79%). However, application of the KBE framework requires experienced personnel with knowledge of the tools to be used. This requires specialist training of the practitioner and /or creation of specialist (simplified) tools to support its use.

It was identified that the KBE approach codifies business logic and tackles one of the core elements of KBE systems (i.e. rule-based modelling). One focus of KBE is rule-based systems, e.g. either for capture of business rules or case management. Ontologies provide the structure to make KBE manageable and this is made evident in the KBE framework case study. Due to the focus on the rule base, the experts identified that the KBE framework is 77% logical. However, to improve this, KBE architects suggest that product design has to integrate with many pre-existing tools and data formats. Therefore, the framework should also consider automation tools for extracting ontologies from existing structured data sources for the KBE system.

Due to the use of ontology APIs, it was identified that the KBE framework offers high-level of reuse and a better management of KBE complexity than conventional approaches. The KBE knowledge can be integrated with other KBE systems, thus, making this approach portable. The experts agreed that the KBE framework offers a high-level of portability (75.2%) which is essential for the future generation of KBE system development. However, it was identified by technical KBE leads in the aerospace industry that platform-independent approaches might not necessary reduce the development efforts (man-hours) required for implementing knowledge enabled KBE systems. In fact, it might increase it due to the need to understand and code the ontology and CAD APIs as agreed by many of the experts in the workshop. An average of 36.3% of experts suggests that platform-independent KBE approaches will reduce cost. This means up to 64% of experts suggest that this will not be the case and development cost will increase if model-driven KBE systems are adopted. However, benefits will be realised in the simpler maintenance of such systems which will positively impact maintenance cost. Finally, it was identified that the KBE framework should also consider the service oriented architecture approach for handling distributed KBE systems. The use of Representational state transfer web services and distributed modelling approach could strengthen the KBE framework.

8. Discussions

The results presented in this work illustrates a platform-independent approach for developing KBE systems that offer higher level of knowledge structure and reuse than the procedural/imperative approach often used in conventional KBE systems. The main advantage of system architectures that are platform independent is the explicit separation of business logic from implementation logic which is essential for developing portable, adaptable and dynamic systems. Additionally, platform-independent approach supports the long-term preservation and reuse of knowledge within KBE systems. Furthermore, it is essential for KBE experts to adopt the MDA and OOP approach which ensures the accurate abstraction and formalisation of the KBE problem domain before polluting it with technological enthrallment. Modelling knowledge models in a platform-independent style will strengthen reuse and eliminate re-build of these models in an ad hoc manner each time a KBE system is implemented. Additionally, platform-independent approaches supports the non-functional/quality attributes of a KBE system. Though KBE system development is often focused on fulfilling functional requirements and exploiting engineering knowledge, non-functional requirements are often not fully considered in practical KBE systems within the aerospace industry. However, in order to ensure the scalability and evolution of KBE systems, non-functional requirements must be explicitly considered within the architecture of KBE systems.

Though the concept of platform-independent systems is commonly employed in the software engineering discipline, it is often not fully embraced in the KBE domain. Attempts to develop KBE architectures that are platform independent are often case based. It is essential to move away from case-based approaches into methodology-/framework-guided

approaches as presented in this study. Methodology-/framework-guided approaches will govern the development of KBE systems that fulfils non-functional requirements. Detailing phases, activities and suggesting approaches, tools, techniques and technologies for developing such systems is essential in contributing towards future generation of KBE system design and development. The idea is not to enforce a strict methodology/framework for developing such systems but to suggest alternatives, innovative approaches and improvement to old methods for developing KBE systems.

Furthermore, there is a need for KBE developers to start utilising and exploiting the full capabilities of APIs as presented in this study (e.g. Jena and standard draw 3D APIs). APIs are one of the core components that enable the exploitation of knowledge within KBE systems. Additionally, there is a need for more developed APIs for KBE systems as this will provide a communication and integration medium between different KBE components, thus supporting platform independent and other types of adaptable KBE systems. However, it is important to highlight that platform-independent approaches might actually be time consuming and require more man-hours to implement due to the need to understand and train KBE experts in utilising APIs. Communicating and integrating KBE components using APIs is a code-intensive activity. However, it is critical for the maintenance and preservation of KBE knowledge for the long term. In addition, graphical user interfaces that bridge the gap between KBE knowledge, CAD system and product designers (end-user) are also of high importance as presented in this study. This will ensure that end-users (i.e. product designers) can directly interact with the ontology knowledge base and CAD system through a developed GUI that connects to all KBE components as presented in the primitive shapes KBE system case study.

9. Conclusions and future work

This paper has reported an ontology-based methodological framework for developing platform-independent KBE systems. The framework is composed of four main phases and uses elements from the software and ontological engineering disciplines such as MDA, OOP, UML, APIs and ontology-based languages (i.e. OWL and SWRL). This paper has demonstrated the use of declarative programming as a means of distinguishing between KBE business logic and implementation logic which is an essential element for KBE systems.

Furthermore, this paper has illustrated the need to address the 'portability' quality attribute/non-functional requirement of a KBE system and ensure this is reflected in the KBE system architecture. The importance of formalising and crystallising the KBE problem domain before assessing technological constraints is addressed using comprehensive standard feature templates, UML-based structures and model-driven approaches.

A critical use of this approach is in the modelling of domain and operational knowledge using ontology-based techniques. This approach is essential in maintaining and evolving the next generation of KBE systems within the aerospace industry. A case study investigating the development of primitive geometry shapes has been conducted to demonstrate the proposed approach.

Additionally, an essential element of the proposed framework is in the use of APIs and GUIs that enables direct integration and communication between a KBE knowledge base, CAD system, GUI application and end-user (e.g. product designer). The system architecture resulting from this work has been implemented using the Standard Draw 3D package, Jena-based ontology API and Eclipse IDE for developing the graphical user interface. The implemented prototype system captures the benefits and limitations of using KBE architectures that are platform independent. Furthermore, practical KBE and software engineering experts have qualitatively validated the proposed KBE framework. The proposed KBE framework could be tailored to support intelligent ontology-driven industrial applications.

A limitation of this research study is that the proof of concept demonstrator was limited to primitive shapes rather than a real engine product. Thus, there is a need to assess how the knowledge base especially the design geometry rules developed in SWRL will cope with the growing amount of complexity in a real KBE gas turbine product. Due to this, there were limitations in assessing and exploiting the full potential and capability of SWRL. Additionally, from an ontology engineering perspective, the focus of this study was on the geometry ontology. Integrated knowledge types such as design for analysis (i.e. aero, stress, thermals), design for service, design for cost, design for manufacturing and design for inspection was not considered in the modelling of the SWRL rules. For practical model-driven, ontology-based KBE systems to be developed, such systems will need to maintain a knowledge base of thousands of rules from various cross-functional and multidisciplinary aerospace functions due to the complexity of gas turbine products.

Future work involves exploring and assessing the scalability of the knowledge base which is another type of non-functional requirement. How would the knowledge base handle growing amount of knowledge complexity (i.e. multidisciplinary skill sets: aero, stress, thermals, geometry, cost and maintenance) that is commonly required for developing gas turbine products within the aerospace sector. Additionally, a further concern that is of significance is that of quantifying the success factors of model-driven, platform-independent KBE systems in monetary and quantifiable (e.g. lead time reduction, cost saved) terms.

Acknowledgements

References

Agyapong-Kodua, K., N. Lohse, R. Darlington, and S. Ratchev. 2013. "Review of Semantic Modelling Technologies in Support of Virtual Factory Design." *International Journal of Production Research.* 51 (14): 4388–4404.

Ahmed, S., S. Kim, and K. M. Wallace. 2007. "A Methodology for Creating Ontologies for Engineering Design." *Journal of Computing and Information Science in Engineering* 7 (2): 132–140.

Barreiro, J. Martinez, P. Susana, C. Eduardo, and Braulio J. Álvarez. 2010. "Comparative Synthesis Between STEP and MOKA Methodologies and New Proposal for the Scope of Manufacturing and Inspection Processes." Annals of DAAAM Proceedings 2010, 82–95, Zadar, Croatia, October 20–23.

Chungoora, N., and R. I. M. Young. 2010. "The Configuration of Design and Manufacture Knowledge Models from a Heavyweight Ontological Foundation." *International Journal of Production Research.* 49 (15): 4701–4725.

Chungoora, N., R. I. M. Young, G. Gunendran, C. Palmer, Z. Usman, Najam A. Anjum, A. N. Cutting-Decelle, Jennifer A. Harding, and K. Case. 2013. "A Model-driven Ontology Approach for Manufacturing System Interoperability and Knowledge Sharing." *Computers in Industry* 64 (4): 392–401.

CommonKADS. 2009. "Engineering and Managing Knowledge." Accessed May 17, 2010. http://www.commonkads.uva.nl/

Dadzie, A. S., R. Bhagdev, A. Chakravarthy, S. Chapman, J. Iria, and V. Lanfranchi. 2008. "Applying Semantic Web Technologies to Knowledge Sharing in Aerospace Engineering." *Journal of Intelligent Manufacturing* 20 (5): 611–623.

Davies, J., D. Fensel, and F. van Harmelen. 2008. *Towards the Semantic Web: Ontology-driven Knowledge Management*. New York: Wiley.

Doultsinou, A., R. Roy, D. I. Baxter, and J. X. Gao. 2009. "Developing a Service Knowledge Reuse Framework for Engineering Design." *Journal of Engineering Design.* 20 (4): 389–411.

Fan, I. S. P., and P. Bermell-Garcia. 2008. "International Standard Development for Knowledge Based Engineering Services for Product Lifecycle Management." *Concurrent Engineering-Research and Applications* 16: 271–277.

Fenves, S., S. Foufou, C. Bock, and R. D. Sriram. 2008. "CPM2: A Core Model for Product Data." *Journal of Computing and Information Science in Engineering* 8 (1): 14501–14507.

Fernandez, L. M., and A. P. Gomez. 2002. "Overview and Analysis of Methodologies for Building Ontologies." *Journal of the Knowledge Engineering Review* 17 (2): 129–156.

Fortineau, V., T. Paviot, and S. Lamouri. 2013. "Improving the Interoperability of Industrial Information Systems with Description Logic-based Models – The State of the Art." *Computers in Industry* 64 (4): 363–375.

Gimémez, D. M., M. Vegetti, H. P. Leone, and G. P. Henning. 2007. "PRoduct ONTOlogy: Defining Product-related Concepts for Logistics Planning Activities." *Computers in Industry* 59 (2–3): 231–241.

Giovannini, A., A. Aubry, H. Panetto, M. Dassisti, and H. El Haouzi. 2012. "Ontology-based System for Supporting Manufacturing Sustainability." *Annual Reviews in Control* 36 (2): 309–317.

Gruber, T. R. 1993. "A Translation Approach to Portable Ontology Specifications." *Knowledge Acquisition* 5 (2): 199–220.

Gunendran, A. G., and R. I. M. Young. 2009. "Methods for the Capture of Manufacture Best Practice in Product Lifecycle Management." *International Journal of Production Research* 48 (20): 5885–5904.

Guo, M., S. Li, J. Dong, X. Fu, Y. Hu, Q. Yin 2003. "Ontology-based Product Data Integration." In Proceedings of 17th International Conference on Advanced Information Networking and Applications, ISBN 0-7695-1906-7, 530–533.

Hawker, K. 2010. "Deploying Semantic Technology within an Enterprise: A Case Study of the UCB Group Project." Semantic Technology Conference SemTech2010, San Francisco, CA, June 21–25.

Horrocks, I. 2008. "Ontologies and the Semantic Web." *Communications of the ACM* 51 (12): 58–67.

ISO 10303-214. 1997. *Second Committee Draft for Industrial Automation System and Integration-product Data Representation and Exchange – Part 214: Application Protocol: Core Data for Automotive Mechanical Design Process*. Geneva: International Standards Organization.

Kitamura, Y., and R. Mizoguchi. 2007. "Ontology-based Systematisation of Functional Knowledge." *Journal of Engineering Design.* 15 (4): 327–351.

Kyoung-Yun, K., D. G. Manley, and Y. Hyungjeong. 2006. "Ontology-based Assembly Design and Information Sharing for Collaborative Product Development." *Computer-Aided Design* 38: 1233–1250.

Liang, Janus S. 2012. "An Ontology-based Method for the Development of a Troubleshooting Configuration System." *International Journal of Computer Integrated Manufacturing* 25 (2): 189–210.

Lin, L. F., W. Y. Zhang, C. Y. Lou, C. Y. Chu, and M. Cai. 2010. "Developing Manufacturing Ontologies for Knowledge Reuse in Distributed Manufacturing Environment." *International Journal of Production Research* 49 (2): 343–359.

Lu, Y., H. Panetto, Y. Ni, and X. Gu. 2013. "Ontology Alignment for Networked Enterprise Information System Interoperability in Supply Chain Environment." *International Journal of Computer Integrated Manufacturing* 26 (1–2): 140–151.

Ma, Q. C., and X. W. Liu. 2007. "Review of Knowledge Based Engineering with PLM." *Journal of Applied Mechanics and Materials* 10: 127–131.

MacLarty, I., L. Langevine, M. V. Bossche, and P. Ross. 2009. "Using SWRL for Rule-Driven Applications." Accessed February 9. http://www.missioncriticalit.com/pdfs/SWRL-Case-Study-2009.pdf

Mendikoa, I., M. Sorli, J. Barbero, and A. Carrillo. 2006. "Distributed Product Design and Manufacturing Based on KBE." In *Computer Supported Cooperative Work in Design II*, edited by David Hutchison, Takeo Kanade, Josef Kittler, J.-P. Kleinberg, and Friedemann Mattern, 404–413. Berlin: Springer.

Mizoguchi, R. 2003. "Part 1: Introduction to Ontological Engineering." *New Generation Computing* 21 (4): 365.

Mizoguchi, R. 2004. "Tutorial on Ontological Engineering Part 2: Ontology Development, Tools and Languages." *New Generation Computing.* 22 (1): 61–96.

Noy, N., and L. McGuinness. 2001. *Ontology Development 101: A Guide to Creating Your First Ontology.* Stanford, CA: Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics. Technical Report SMI-2001-0880, March.

Panetto, H., M. Dassisti, and A. Tursi. 2012. "ONTO-PDM: Product-driven ONTOlogy for Product Data Management Interoperability within Manufacturing Process Environment." *Advanced Engineering Informatics* 26 (2): 334–348.

Patil, L., D. Dutta, and R. Sriram. 2005. "Ontology-based Exchange of Product Data Semantics." *IEEE Transactions on Automation Science and Engineering* 2 (3): 213–225.

Pinto, H., and J. Martins. 2004. "Ontologies: How Can They Be Built?" *Knowledge and Information Systems* 6 (4): 441–464.

PLM World. 2014. "Knowledge Fusion Tips and Tricks." Accessed February 9. http://www.plmworld.org/e/in/eid=26&req= info&s= 3506&all=1

Ruschitzka, M., A. Suchodolski, and J. Wróbel. 2010. "Ontology-based Approach in Hybrid Engineering Knowledge Representation for Stamping Die Design." In *New World Situation: New Directions in Concurrent Engineering*, edited by J. Pokojski, S. Fukuda, and J. Salwiński, 227–235. London: Springer.

Saa, R., A. Garcia, C. Gomez, J. Carretero, and Felix Garcia-Carballeira. 2012. "An Ontology-driven Decision Support System for High-performance and Cost-optimized Design of Complex Railway Portal Frames." *Expert Systems with Applications* 39 (10): 8784–8792.

Sandberg, M. 2003. "Knowledge Based Engineering – In Product Development." Technical Report, Lulea University of Technology, Department of Applied Physics and Mechanical Engineering, Division of Computer Aided Design, Lulea.

Sanya, I., E. M. Shehab, and R. Roy. 2010. "Semantic Knowledge Based Approach for Cost Modelling." Proceeding of the 8th International Conference on Manufacturing Research (ICMR 2010), 101–107, Durham, September 14–16.

Schreiber, G. 2000. *Knowledge Engineering and Management: The CommonKADS Methodology.* Cambridge, MA: MIT Press.

Shehab, E., C. Fowler, A. Rodriguez Gil, H. Abdalla, M. Darwish, H. Abdulhafed, A. Ahmed, et al. 2013. "Enhancement of Product Information Collaboration and Access in the Aerospace Industry." *International Journal of Production Research.* 51 (11): 3225–3240.

Shen, J., L. Wang, and Y. Sun. 2012. "Configuration of Product Extension Services in Servitisation Using an Ontology-based Approach." *International Journal of Production Research.* 50 (22): 6469–6488.

Siemens NX. 2014. "About NX Software." Accessed February 9. http://www.plm.automation.siemens.com/en_gb/products/nx/about-nx-software.shtml

Skarka, W. 2007. "Application of MOKA Methodology in Generative Model Creation Using CATIA." *Engineering Applications of Artificial Intelligence* 20 (5): 677–690.

Stokes, M. 2001. *Managing Engineering Knowledge: MOKA: Methodology for Knowledge Based Engineering Applications*. London: Professional Engineering.

Sunnersjö, S. M., Mikael Cederfeldt, Fredrik Elgh, and I. Rask. 2006. "A Transparent Design System for Iterative Product Development." *Journal of Computing and Information Science in Engineering* 6: 300–307.

Terzi, S., H. Panetto, G. Morel, and M. Garetti. 2007. "A Holonic Metamodel for Product Traceability in Product Lifecycle Management." *International Journal of Product Lifecycle Management* 2 (3): 253–289.

Tsai, C. Y., T. H. Sun, and J.-X. Huang. 2006. "A Web-based XML Information Sharing System for Collaborative Product Development." *International Journal of Production Research* 44 (15): 2955–2976.

Tursi, A., H. Panetto, G. Morel, and M. Dassisti. 2009. "Ontological Approach for Products-centric Information System Interoperability in Networked Manufacturing Enterprises." *IFAC Annual Reviews in Control* 33 (2): 238–245.

Uschold, M., and M. King. 1995. "Towards a Methodology for Building Ontologies." IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing, 6.1–6.10, Montreal, Canada. .

Vegetti, M., G. P. Henning, H. P. Leone. 2005. "Product Ontology: Definition of an Ontology for the Complex Product Modelling Domain." 4th Mercosur Congress on Process Systems Engineering, Rio de Janeiro, Brazil, August 14–18.

Verhagen, Wim J. C., P. Bermell-Garcia, Reinier E. C. van Dijk, and R. Curran. 2012. "A Critical Review of Knowledge-based Engineering: An Identification of Research Challenges." *Advanced Engineering Informatics.* 26: 5–15.

Zdravkovic, M., and M. Trajanovic. 2009. "Integrated Product Ontologies for Inter-organizational Networks." *Computer Science and Information Systems* 6 (2): 29–46.