Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2013) 000–000

Procedia
Computer Science

www.elsevier.com/locate/procedia

17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2014

Implementing Virtual Engineering Objects (VEO) with the Set of Knowledge Experience Structure (SOEKS)

Syed Imran Shafiq[a]*, Cesar Sanin[a], Edward Szczerbicki[b] & Carlos Toro[c]

[a]*The University of Newcastle, University Drive, Callaghan, 2308, NSW, Australia.*
[b]*Gdansk University of Technology, Gdansk, Poland.*
[c]*Vicomtech-IK4, San Sebastian, Spain.*

Abstract

This paper illustrates the idea of Virtual Engineering Object (VEO) powered by Set of Knowledge Experience Structure (SOEKS). A VEO is the knowledge representation of an engineering object, having embodiment of all its associated knowledge and experience within it. While, the SOEKS is a flexible and standard knowledge representation structure to acquire and store experiential knowledge. The article also presents a case study to demonstrate implementation of VEO in manufacturing scenario. The decision making in manufacturing will benefits from this approach, as it include capturing of experience of an engineering artefact, storage and indexing of knowledge and reuse of knowledge according to needs.

Keywords: Desional DNA; Set of Knowledge Experience Structure (SOEKS); Virtual Engineering Object(VEO).

1. Introduction

A well-established feature of the manufacturing and design is that a significant percentage of the products life cycle is spent on routine tasks up to 80%. It is noted, 'around 20% of the designer's time is spent searching for and absorbing information', and '40% of all design information requirements are currently met by personal stores, even though more suitable information may be available from other sources'[1]. Product related services such as maintenance, retrofitting or user trainings currently account for well over 18% of the total turnover of the German discrete part manufacturing industry with even higher numbers in other countries[2].

* Corresponding author. Tel.: +61 405408834.
E-mail address: SyedImran.Shafiq@uon.edu.au.

This implies that design information and knowledge is not represented in a shared and easily accessible knowledge base.

There are a number of strong arguments for adopting computer integrated knowledge based manufacturing system to meet the above discussed features of successful manufacturing and design. One of the hallmarks of the knowledge base systems is to automate repetitive, non-creative design tasks. Not only does automation permit significant time and cost savings, it also frees up time for creativity, which allows exploration of a larger part of the design envelope. Clearly, in such cases knowledge reuse guided framework may save considerable time and effort

Decisional DNA (DDNA), as a domain-independent, flexible, and standard knowledge repository, can not only capture and store experiential knowledge in an explicit and formal way but can also be easily applied to various domains to support decision making and standard knowledge sharing and communication among these systems [3-5].

In this article, we present an approach that integrates DDNA with VEO to capture and reuse manufacturing experiences. In addition, we demonstrate this approach in a set of experiments in order to test the usability and suitability of DDNA in supporting decision making and knowledge reuse for VEO.

This paper is organized as follows: section two describes the Set of Experience Knowledge structure and Decisional DNA; section three presents the structure of a VEO; section four demonstrates implementation of the Decisional DNA applied to VEO and the experiment results of our approach. Finally, section five discusses concluding remarks and the scope of future work.

2. The Set of Experience Knowledge Structure (SOEKS) and Decisional DNA

The Set of Experience Knowledge Structure (SOEKS or shortly SOE) is a domain-independent, flexible and standard knowledge representation structure. It has been developed to acquire and store formal decision events in an explicit way. It is a model based upon available and existing knowledge, which must adapt to the decision event it is built from (i.e. it is a dynamic structure that depends on the information provided by a formal decision event); besides, it can be represented in XML or OWL as an ontology in order to make it transportable and shareable[6, 7].

SOEKS is composed of variables, functions, constraints and rules as shown in figure 1. Variables normally implicate representing knowledge using an attribute-value language (i.e. by a vector of variables and values), and they are the centre root of the structure and the starting point for the SOEKS. Functions represent relationships between a set of input variables and a dependent variable; moreover, functions can be applied for reasoning optimal states. Constraints are another way of associations among the variables.

They are restrictions of the feasible solutions, limitations of possibilities in a decision event, and factors that restrict the performance of a system. Finally, rules are relationships between a consequence and a condition linked by the statements IF-THEN-ELSE. They are conditional relationships that control the universe of variables [8].of Knowledge Experience Structure (SOEKS) and DDNA Additionally, SOEKS is designed similarly to DNA at some important features.

First, the combination of the four components of the SOE gives uniqueness, just as the combination of four nucleotides of DNA does. Secondly, the elements of SOEKS are connected with each other in order to imitate a gene, and each SOE can be classified, and acts like a gene in DNA [8]. As the gene produces phenotypes, the SOE brings values of decisions according to the combined elements. Then a decisional chromosome storing decisional “strategies” for a category is formed by a group of SOE of the same category. Finally, a diverse group of SOE chromosomes comprise what is called the Decisional DNA [2].

In short, as a domain-independent, flexible and standard knowledge representation structure, SOEKS and Decisional DNA provide an ideal approach which can not only be very easily applied to various domains, but also enable standard knowledge communication and sharing among these systems [5].

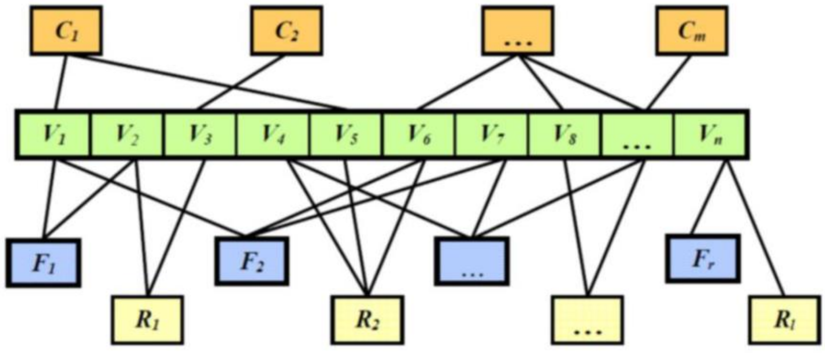

Figure1: The structure of set of experience [9],[6]

3. Virtual Engineering Object (VEO)

The concept of Virtual Engineering Objects (VEO) is powered by SOEKS and DDNA; it is designed to have all the knowledge of the engineering artefact along with the associated experience embedded in it. A VEO is a living representation of an artefact which can behave like an expert of that area and can help the practitioners in effective decision making based on the past experience [5].

A VEO can encapsulate knowledge and experience of every feature related with an engineering object. This can be achieved by gathering information from five different aspects of an object viz. Characteristics, Functionality, Requirements, Connections, Present State and Experience as illustrated in figure 2.

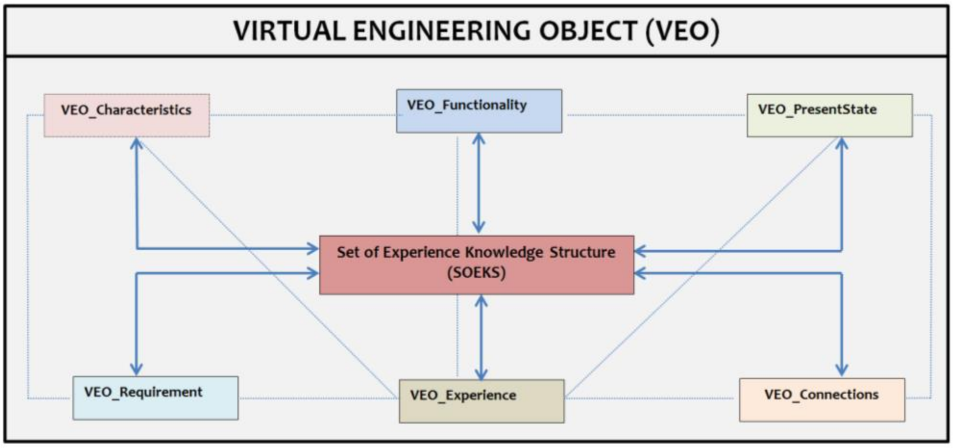

Figure 2. Proposed VEO Structure [5, 10]

The main features of a VEO (shown in figure 2) are as follows:

- *Characteristics* describe the set of physical features and expected benefits offered by the artefact represented by the VEO. Not only the information like its geometry, dimensions, appearance, weight etc. are captured in this module but also the possible advantages like adaptability and the ease of using it can also be achieved from this. Knowledge stored Characteristics assist in better decisions making like, which VEO is best suited for a given physical condition and also when more than one VEO of similar kind are available it helps to decide which is the best in the given situation.

- *Functionality* describes the basic working of the VEO and principle on which it accomplish its operation. Operational Knowledge related with an object like time consumed, its working boundary limits and the outcome of the process that is performed are stored in Functionality. This module of the VEO assists in storing, selecting and reusing the practical details.
- *Requirements* describe the set of necessities of the VEO for its precise working. Information like what and how much power source, how much space and the extent of user expertise necessary for operating a VEO can be stored.
- *Connections* describe how the VEO is related with other VEOs. Many engineering object works in conjunction with other object, these VEO's may be a "part" or may be a "need" of each other. This module of VEO structure is essential for the scaling up and establishing the interconnection of VEO's in manufacturing scenario.
- The *present state* of the VEO highlights parameters of the VEO at the current moment. It is like whether the VEO is ready for a particular operation? If yes, then it also gives an idea about the background of the VEO like its reliability and precision. If it is busy it can predict the expected time when the VEO will be free for the next operation.
- The *Experience* of the VEO deals with the knowledge and information which is dynamic in nature and keeps on changing with the each and new decision, operation or event. Rather it stores every formal decision taken related with the working of the VEO. This segment of the VEO keeps on updating with every activity that is done on the VEO

4. Structure and Implementation of a VEO

For implementation, we integrated the concepts of Decisional DNA and the VEO. As discussed in section 2 SOEKS comprises of Variables, Functions, Constraints and Rules. Moreover in section 3 we also discussed that a VEO structure include elements like Characteristics, Functionality, Requirements, Connections, Present State and Experience. SOEKS files for each element of the VEO in the system are created individually. The goal behind this was to provide a more scalable setting, similar to the one that would be found in describing a diverse range of engineering objects. Weights are assigned to the attributes of the variables of an artefact, and then the six sets of SOEKS are generated. These individual SOEKS are combined under an umbrella (VEO), representing experience and knowledge.

4.1 Test Case Study

The purpose of the present experiment is to apply our proposed architecture to gather, store and reuse information related to the Drilling Machines in a manufacturing setup. Table 1 illustrates the detailed VEO structure for drilling machines. Effort is made to capture and store all the relevant information of the VEO adhering to the format of the SOEKS.

In the Characteristics section of drilling machine VEO physical parameters like area, volume, maximum capacity, manufacturer details, service details are stored. Furthermore rules are laid to extract knowledge about the VEO like Ease of Operation and adaptability.

In Functionality, variables related with the functioning of a drilling machine like cutting speed, feed, depth of cut, drilling diameter, drilling depth etc. are defined along with their constraints and limits. In addition to this knowledge, the outcome of drilling operation like quality of surface finish and precision can also be determined in the form of rules.

How much Space is required? What and how much power source is required? What kind of expertise of the operator is necessary? All these information for each and every operation can be stored in the Requirements section of the VEO.

While working on drilling machine the drilling tool e.g. Twist Drill and the work holding device e.g. vice is considered as separate VEO's. And Information of these and their relation with main drilling machine VEO are stored in the connections.

In the present state not only whether the VEO is free or idle is determined but also knowledge about VEO like its overall reliability and precision till date can be extracted from this segment.

And lastly in the Experience all the variable information related with each operation performed and the formal decisions taken on the drilling machine are stored.

Table1: Structure of a VEO

<table>
<tr><th></th><th>VARIABLES</th><th>FUNCTIONS</th><th>CONSTRAINTS</th><th>RULES</th></tr>
<tr><td rowspan="3">CHARACTERISTICS</td><td rowspan="3">veoCODE
veoNAME
veoLENGTH
VeoWIDTH
veoHEIGHT
veoWEIGHT
veoCOLOR
veoMaxWPWeight
veoCAPACITY
veoMATERIAL
veoOPERATOR
veoMANUFACTURER
veoPURCHASE
veoServiceDate</td><td>veoAREA=veoLength*veoWidth
veoVOL=veoLength*veoWidth*veoHeight</td><td>veoLength>0
veoWidth>0
veoHeight>0
veoWeight>0</td><td>IF veoOperator=Skilled THEN
0<veoEaseofOperation<=3
IF veoOperator=Semi-skilled THEN
3<veoEaseOfOperation<=6
IF veoOperator=Un-skilled THEN
7<veoEaseOfOperation<=10</td></tr>
<tr><td rowspan="2"></td><td>veoWPLength<veoMaxWPLength
veoWPWidth<veoMaxWPWidth
veoWPHeight<veoMAXWPHeight</td><td>IF veoNoOfOp<5 THEN 0<Ease of Adaptability<=5
ELSE <Ease of Adaptability <=10</td></tr>
<tr><td></td><td>IF Date-veoServiceDate> 6 months THEN "Service Due"
ELSE "Next service due on … days"</td></tr>
<tr><td rowspan="3">FUNCTIONALITY</td><td rowspan="3">veoCODE
veoNAME
VeoFUNCTION
veoPRINCIPLE
veoSpindleSpeed
veoCuttingSpeed
veoFEED
veoDepthOfCut
veoMaxHoleDia
veoMaxHoleDepth
veoCoolantUsed</td><td>veoMachining Time =
(60*veoSpindleSpeed)/

(π*veoDepthOfHole)</td><td>50≤ veoSpindleSpeed≤ 3000
10≤ veoCuttingSpeed≤50
.2 ≤ veoFeed≤32
veoDiaOfHole<veoMaxHoleDia
veoDepthOfHole<veoMaxHoleDepth</td><td>IF 1.6 < veoSurfaceFinish <= 3.6, THEN Smooth
IF 3.6 < veoSurfaceFinish <= 4.6, THEN Moderate
IF 4.6 < veoSurfaceFinish <= 6.3, THEN Rough</td></tr>
<tr><td rowspan="2"></td><td rowspan="2"></td><td>IF veoDiaofHole-0.05 ≤ veoPrecision ≤ DiaofHole +0.05 ,THEN "Good Precision" ELSE "Not Good Precision"</td></tr>
<tr><td>IF veoMachiningTime>60, THEN "Use Coolant"</td></tr>
<tr><td>EQUIREMENTS</td><td>veoCODE
veoNAME
veoPowerSource
veoSpaceRequired
veoOperatorRequired</td><td></td><td>veoSpaceRequired ≥ veoVolume
220v ≤ veoPowerSource ≤ 250v</td><td></td></tr>
</table>

CONNECTIONS	veoCODE veoNAME veoNEED/PART veoVEO[1]			
PRESENT STATE	veoCODE veoNAME veoTotalOpTime veoSTATUS veoReliability veoProductivity veoPrecision veoBreakdownDetails	veoTotalOpTime = $\sum_{veoCode=1}^{n}(\text{veoMachiningTime})$		IF veoSTATUS =Busy, THEN wait for veoMachiningTime ELSE "Ready"
		veoReliabilty =f(precision,breakdown)		
EXPERIENCE	veoCODE veoNAME veoOperationNo veoWorkpieceMaterial veoDiameterOfHole veoDepthOfHole veoWorkpieceDim veoMachiningTime veoDrilledPrecision veoFEED veoCutttingSpeed veoDepthOFCut veoOperator veoSurafceFinish veoToolUsed veoWorkHoldingDevice veoWPLength veoWPWidth veoWPHeight			

This VEO structure is implemented using JAVA programing language, reason for that is that Structure of the Decisional DNA is constructed in JAVA[4]. Every Variable (see table 1) is stored as the SOEKS variable, to provide an illustration of such a structure, what follows is the VEO Name stored as a SOEKS variable:

```
<variable>
<var_name>VEO_Name</var_name>
<var_type>CATEGORICAL</var_type>
<var_cvalue>DM1 </var_cvalue>
<var_evalue>DM1 </var_evalue>
<unit></unit>
<internal>false</internal>
<weight>0.0</weight>
```

```
<l_range >0.0</l_range>
<u_range >0.0</u_range>
<categories>
<category></category>
</categories>
<priority>0.0</priority>
</variable>
```

Figure 3 shows six JAVA classes are developed having SOEKS Variables, SOEKS Functions, SOEKS Constraints and SOEKS Rules. SOE for each class are stored individually. In a separate class these SOE's are combined to form knowledge and experience repository of the VEO. From this knowledge base manufacturing information can be extracted for future decision making .

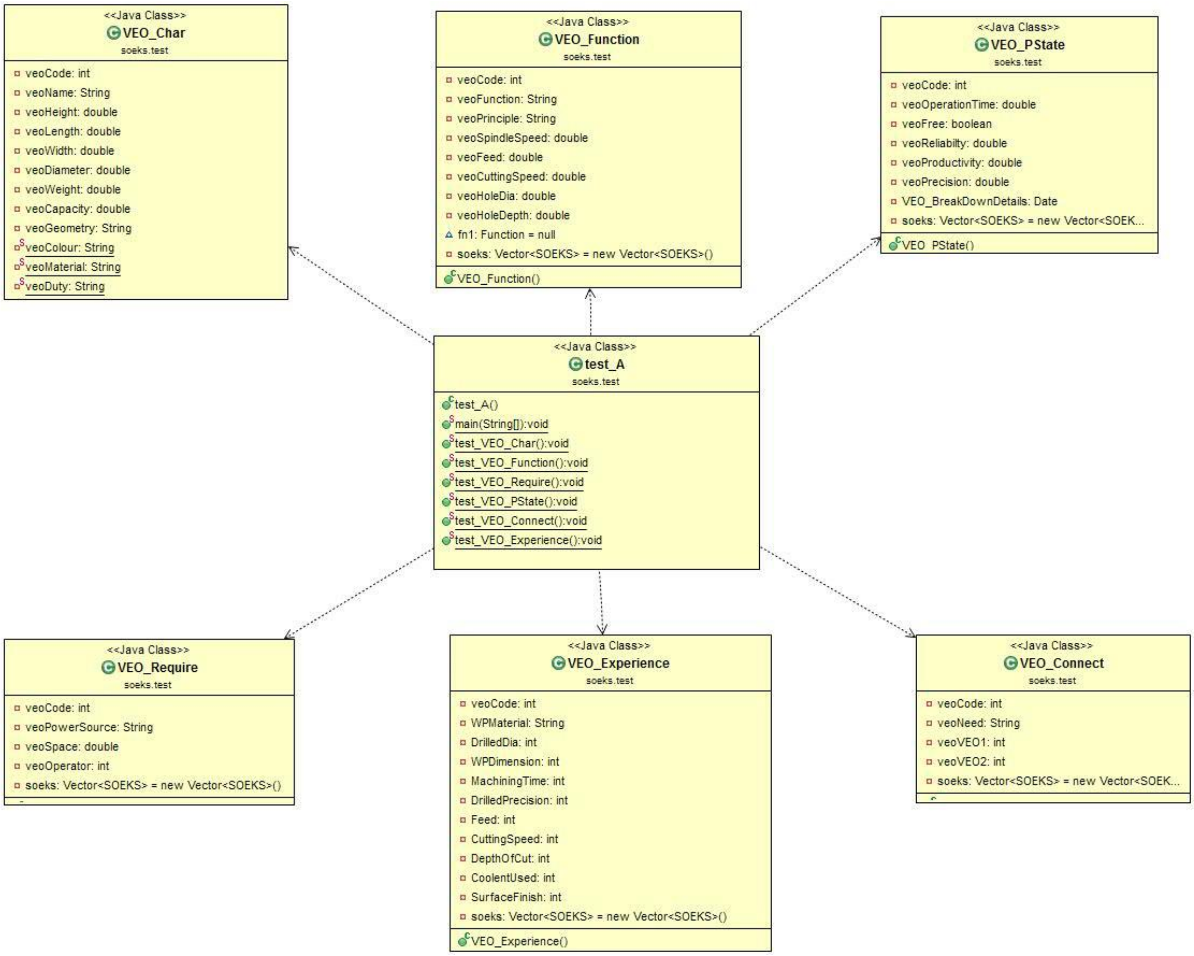

Figure3 JAVA class diagram

4.2 Decisional DNA based VEO Experimental Results.

We are able to capture and store information every operation that is performed on the VEO and then update the knowledge base of the VEO. The gathered information is effectively and efficiently converted into Decisional DNA structure. Furthermore we are able to query the VEO and based on the experience it can predict and suggest options available according to our need.

5. Conclusion

We presented an approach that allows a VEO to capture and reuse its own experiences by applying Decisional DNA. The SOEKS and DDNA-based VEO seems to be a suitable and comprehensive tool for knowledge discovery We described the architecture of our approach and implemented and tested our concepts in the form of a case study. The illustrative empirical experiment presented in this article also showed that this structure can be used as an effective and precise prediction tool The Decisional DNA, as a novel knowledge representation structure, not only can be applied to VEO but indeed improves the manufacturing performance when it is using experience. It additionally enables users to make their knowledge shareable, transportable, and easily understandable.

6. Future Work

To continue with this idea, further research and refinements are required, and our efforts are currently directed towards:

- Formulation of general VEO structure.
- Refinement of variables that can represent VEO in a more general way.
- Further development of the VEO rule base database.
- Further development of user management based on gathered experience.

References

[1] D. Baxter, J. Gao, K. Case, J. Harding, R. Young, S. Cochrane, *et al.*, "An engineering design knowledge reuse methodology using process modelling," *International Journal of Research in Engineering Design,* vol. 18, pp. 37-48, 2007.

[2] J. X. G. R. Sharma, "A knowledge-based manufacturing and cost evaluation system for product design/re-design," *Int J Adv Manuf Technol,* vol. 33, pp. 856-865, 2007.

[3] C. Sanín and E. Szczerbicki, "Experience-Based Knowledge Representaion: SOEKS," *Cybernetics and Systems,* vol. 40, pp. 99-122, 2009/02/12 2009.

[4] C. Sanin and E. Szczerbicki, "Towards the Construction of Decisional DNA: A set of Experience Knowledge Structure JAVA Class within an Ontology System," *Cybernetics and Systems,* vol. 38, pp. 859-878, 2007/10/31 2007.

[5] S. I. Shafiq, C. Sanin, and E. Szczerbicki, "Set of Experience Knowledge Structure (SOEKS) and Decisional DNA (DDNA): Past, Present and Future," *Cybernetics and Systems,* vol. 45, pp. 200-215.

[6] C. Sanin and E. Szczerbicki, "Extending Set of Experience Knowledge Structure into a Transportable Language extensible Markup Language," *Cybernetics and Systems,* vol. 37, pp. 97-117, 2006/03/01 2006.

[7] C. Sanín, L. Mancilla-Amaya, Z. Haoxi, and E. Szczerbicki, "Decisional DNA: The Concept and its Implementation Platforms," *Cybernetics and Systems,* vol. 43, pp. 67-80, 2012/02/01 2012.

[8] C. Sanín, L. Mancilla-Amaya, E. Szczerbicki, and P. CayfordHowell, "Application of a Multi-domain Knowledge Structure: The Decisional DNA," in *Intelligent Systems for Knowledge Management*. vol. 252, N. Nguyen and E. Szczerbicki, Eds., ed: Springer Berlin Heidelberg, 2009, pp. 65-86.

[9] C. Sanin and E. Szczerbicki, "Set of Experience: A Knowledge Structure for Formal Decision Events," *Foundations of Control and Management Sciences,* vol. 3, pp. 95-113, 2005.

[10] C. S. Syed Imran Shafiq, Edward Szczerbicki, Carlos Toro, "Decisional DNA based framework for representing Virtual Engineering Objects," in *Intelligent Information and Database Systems, 6th Asian Conference, ACIIDS 2014, Bangkok, Thailand, April 7-9, 2014,*

Proceedings, Part I vol. 8397, B. A. N.-T. Nguyen, B. Trawinski, K. Somboonviwat Ed., ed: Springer, Lecture Notes in Artificial Intelligence, 2014.