preliminary findings from the DECIDE project

P. Hale, J.Scanlan, C Bru
Faculty of Computing, Engineering and Mathematical Sciences, University of the West of England

M Dunkley
Airbus UK, Filton, Bristol

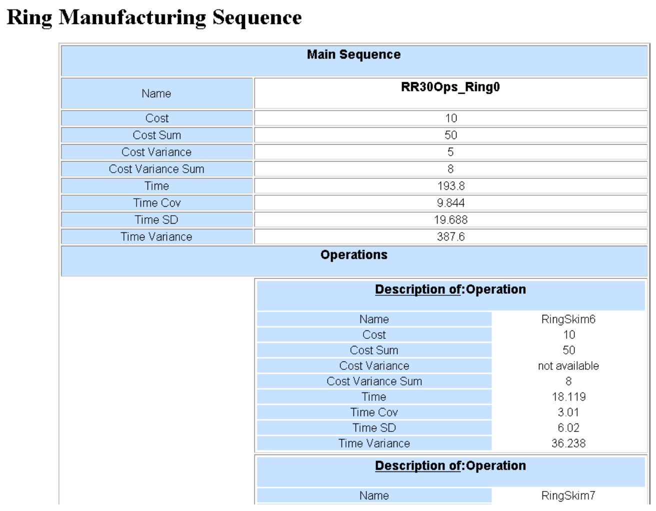
ABSTRACT: This paper outlines the tools and technologies used, or to be used in the DECIDE (Decision Evaluation and Costing In Developmental Engineering) decision support system. The tools and technologies investigated are manufacturing costing, taxonomies and ontologies, structured languages, and user driven programming, information sharing, and simulation. These are all required in order to make a manufacturing decision support system effective, and they must be used in a coherent way. The work of specialists in these areas is described and evaluated. This can help give an understanding to the non-specialist about how an overall system can be constructed from these parts. The University of the West of England Implied Cost Evaluation System (ICES) group have produced manufacturing decision support systems. This paper investigates the research underway into migration of these systems to a collaborative web environment. The research aim is to make it possible to create a generic decision support system, and also to investigate ways for users to create programs within the environment of a knowledge based system.

1 INTRODUCTION

The success of the Internet and Web technologies has made it possible for decision support tools to act over a distributed network, and make use of existing knowledge repositories. A comprehensive decision support tool needs to perform a wide variety of tasks. This has blurred the distinction between different categories of software. This means that individuals need a large breadth of knowledge and experience. It is hoped that this paper might act as a starting point for those needing to acquire this breadth of knowledge in order to design a Web based decision support system, as well as giving an illustration of our approach.

2 AIMS OF THE RESEARCH

Our intention is to provide a system allowing a user to define a part to be manufactured. The system will then assist in generating a process plan. On starting the system the user would be presented tree style representation of default parts, and parts previously defined by other users, based on an inheritance hierarchy of parts. This will be taken from an engineering ontology. The user can then choose a part from this tree or add a new part to the tree. Each node will represent a part and on accessing this node the user will be presented with a part definition screen. This would allow the user to mix and match part definition details from other parts and add the new part definition details as required. The system will then assist in producing a process plan and product data structure, and associate it with this part. Figure 1 shows a possible representation of a manufacturing sequence for an aerospace engine part. This is represented in RDF (Resource Description Framework) which is explained later.



Ring Manufacturing Sequence

Main Sequence	
Name	**RR30Ops_Ring0**
Cost	10
Cost Sum	50
Cost Variance	5
Cost Variance Sum	8
Time	193.8
Time Cov	9.844
Time SD	19.688
Time Variance	387.6

Operations

Description of:Operation	
Name	RingSkim6
Cost	10
Cost Sum	50
Cost Variance	not available
Cost Variance Sum	8
Time	18.119
Time Cov	3.01
Time SD	6.02
Time Variance	36.238

Description of:Operation	
Name	RingSkim7

Figure 1. Representation of Engine Ring Manufacturing Sequence

This example can be viewed on our web site at -

http://www.cems.uwe.ac.uk/~phale/XMLDemonstrators/rdfring.xml

The user could optionally run a web simulation to verify this process plan and product data structure. This will alter these details to reflect resource and time constraints. The aims of the ICES team are outlined in (Scanlan et al. 2000). Within the ICES team (Nour 2001) has created this simulation tool as part of his software relating to project management. (Bru et al. 2002) is working on visualisation of cost information, this is to be made available over the Web and Marsh has created a prototype expert system shell (Marsh et al. 2000). This system (COINER) has much of the functionality required of a Web system, but is usable on a single PC only, and uses a proprietary data format. It is proposed that we create a Web clone of the COINER system. This together with the visualisation research will be the basis for the Web based system currently under construction. We are currently looking for ways of representing the information in an open standard data source such as XML (eXtensible Mark-Up Language), and creating Web services and clients to provide the functionality and user interface. The next section explains some of the research undertaken in order to gain the knowledge to make development of such a system possible.

3 BACKGROUND, TOOLS AND TECHNOLOGIES

3.1 *Generative Costing and Decision Support*

(Eaglesham 1998) makes a study of design for manufacturability theory and practice, and of activity based costing methods and value analysis. Much of this analysis is achieved through diagrams, which explain the workflow from design to production. This research is applied in a software tool.

The paper by Eaglesham examines how a decision support system is constructed for composites manufacturing. This is a good area to test a decision support model because composites are often used where metals were used previously. Composites have a different cost/weight relationship, which would invalidate comparison with parametric models based on metallic components. (Brundwick 1995, NASA 2002, Duverlie et al. 1999) explain parametric cost estimation and software for this costing method. Eaglesham examines activity-based costing methods and applies this costing method to his software decision support system. Eaglesham's software makes use of commonly available database and spreadsheet software.

Eaglesham's software can be used at an early stage in a project to generate a process plan using information from a database. This can make it possible to create a generative costing. We define generative costing as costing based on the automated generation of a process plan. This can improve accuracy and conveys additional information, as parametric costing can not be used to generate a process plan. This means the information is useful to design, manufacture, and costing engineers and so can be an important information source for integrated design and manufacture.

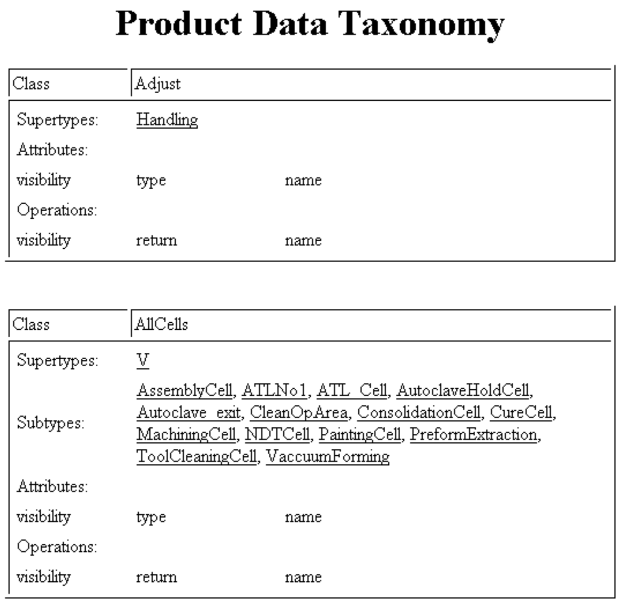
This paper examines how a decision support system could search existing information in order to create a new costing. This is a major undertaking and further research is needed into how such a system can structure and search for what the manufacturing decision software requires. The existing data may be stored in various formats and it is difficult for an automated system to deduce the data's meaning. The use of SQL to illustrate search strategies is a useful step but a sophisticated taxonomy is needed as well in order to make a search strategy work. Eaglesham's paper uses the example of a wing. ICES' aim of allowing costing of generic products would require us to find a way of creating a generic structure into which all things could fit.

The approach used by the ICES team is to apply this style of decision support system to a much wider knowledge domain, to make it possible to use software for analysis of design and manufacture for a large range of products. To achieve this we would need to model the design and manufacture processes. (Roy et al. 2002) modelled the design process and identified Cost Estimating Relationships (CERs). The ICES project concentrates more on manufacturing. Also, ICES research is underway into widening the availability of the software within an organization by taking advantage of advances in software technology for Intranet projects. This can allow us to create a large distributed system, which permits multiple access. (Huang et al. 2001) examines the issues important in developing Web applications for product design and manufacture. Researchers in this field will not usually have the time to investigate the fast changing Web technologies. Huang et al. provide a guide to enable such researchers to develop a prototype Web application.

3.2 *Taxonomies and Ontologies for Decision Support*

An Ontology is a classification structure. A Taxonomy can be just a convenient structure to assist programmers or part of an overall 'thesaurus' which describes and agrees the meaning of things. This 'thesaurus' structure is the ontology and may contain one or more taxonomies. Engineers may have different names for the same thing, e.g. wing skin stiffeners may be referred to as stringers, but rib stiffeners are never called stringers. There is a relationship of stringer to stiffener, which needs to be defined, and this definition depends on the context. An engineering classification scheme or ontology is necessary in order to make communication between engineers precise. Such an ontology can also be used to help non-engineers to understand the terminology. The ontology can also enable communication between the computer systems and users. (Hunter 2002) explains how taxonomies can be the basis of the definitions for an ontology, and that commercial software is becoming available. Hunter gives examples of the Ministry of Defense technology taxonomy, and the Boeing online ontology. The taxonomy "Type-Of" and "Part-Of" relationships can indicate how to construct the taxonomy. (Veryard 2001, McGuinness 2002) provide useful guides on how ontologies can assist in linking distributed data. Unified Modeling Language (UML) software provides a way of visually constructing a taxonomy.

A possible way of mapping terms into a taxonomy is to use XML. Figure 2 shows part of a UML design translated into XMI (XML Metadata Interchange) which is used for representing UML diagrams in XML. A stylesheet is used to display the XMI on the Web in a readable form. A user can click on an associated term to view that term.



Product Data Taxonomy

Class	Adjust	
Supertypes:	Handling	
Attributes:		
visibility	type	name
Operations:		
visibility	return	name

Class	AllCells	
Supertypes:	V	
Subtypes:	AssemblyCell, ATLNo1, ATL_Cell, AutoclaveHoldCell, Autoclave_exit, CleanOpArea, ConsolidationCell, CureCell, MachiningCell, NDTCell, PaintingCell, PreformExtraction, ToolCleaningCell, VaccuumForming	
Attributes:		
visibility	type	name
Operations:		
visibility	return	name

Figure 2. Product Data Taxonomy

(Kogut et al. 2002) & (Baclawski et al. 2001) explain how UML can be used as a tool to produce ontologies.

Current research ICES has undertaken is into how to apply the work of Tim Berners-Lee and others in the World Wide Web Consortium (W3C) (W3C 2001) & (Berners-Lee 1999). In order to define a part, material or a process, it is necessary to use Meta-data. This takes the form of Meta–tags to define words as having a particular meaning. XML is used for structuring information. This is a Meta-data language and W3C ensures that the syntax is standardised so different software systems can interpret it. This language can be used to structure web pages so that they can represent and link to programming objects. 'Knowledge sharing within an Internet expert system and between it and other systems can be facilitated using XML' (Dennis 2001). XML can be used as a way to manage workflow within and between organizations (Ferreira & Ferreira 2001). XML provides the basis for creation of taxonomies and ontologies (Bryan 2002). This Meta language is useful as it fits well with both an Object Oriented and a Rule-Based approach to problem solving and software development. Using Meta-tags defined with XML it is possible to create documents that define their own structure. The information is then passed to software, which can take the value and process it, or pass it to other objects within a system to facilitate decision support. XML also helps to provide common definitions to make it easier to use computer middleware such as CORBA (Common Object Request Broker Architecture) and DCOM (Distributed Component Object Model). The XML code is linked to an XSL (eXtensible Stylesheet Language) stylesheet to define appropriate formatting, thus outputting the information in a standardised and comprehensible way. This makes it possible to provide a consistent and understandable user interface.

XML can be used to encode an ontology. (Fensel et al. 1998) describe Ontobroker and the use of XML and RDF within this ontology tool. The use of ontologies is being driven by e-commerce and e-procurement where trading is online (UN/CEFACT 2002) and (ebXML 2002).

XML may not be sufficient on its' own for defining ontologies. The XML syntax defines relationships by their position within the text file. Thus XML syntax always implies a sequence whereas in reality the order of items may be unimportant, also there is no explicit way of representing associations between items, or differentiating between an Inheritance and a Contains relationship. XML schemas and DTDs (Document Type Definitions) can be helpful in defining these relationships, but there is then scope for differences in the way they are defined. RDF has provided a layer of standardised semantics which overlays the basic XML. The RDF text can be embedded within XML. This has allowed us to explicitly represent Engine Ring Manufacture as a sequence of groups of sequential operations. Figure 3 shows the RDF text corresponding to the section of the Engine Ring Manufacturing sequence in Figure 1.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="rdfring.xsl"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
  ns#"
  xmlns:s="http://localhost/schema/">
```

```
    <s:Sequence>
    <rdf:Sequence>
    <rdf:Description about="Sequence">
    <s:Name>RR30Ops_Ring0</s:Name>
    <s:Cost>10</s:Cost>
    <s:CostSum>50</s:CostSum>
    <s:CostVariance>5</s:CostVariance>
    <s:CostVarianceSum>8</s:CostVarianceSum>
    <s:Time>193.8</s:Time>
    <s:TimeCov>9.844</s:TimeCov>
    <s:TimeSD>19.688</s:TimeSD>
    <s:TimeVariance>387.6</s:TimeVariance>
       <s:Sequence>
          <rdf:Sequence>
             <rdf:Description about="Operation">
                   <s:Name>RingSkim6</s:Name>
                   <s:Cost>10</s:Cost>
                   <s:CostSum>50</s:CostSum>
                   <s:CostVariance>not avail-
                   able</s:CostVariance>
                   <s:CostVarianceSum>8</s:CostVarianceS
                   um>
                   <s:Time>18.119</s:Time>
                   <s:TimeCov>3.01</s:TimeCov>
                   <s:TimeSD>6.02</s:TimeSD>
                   <s:TimeVariance>36.238</s:TimeVariance
                   >
                </rdf:Description>
                <rdf:Description about="Operation">
                   <s:Name>RingSkim7</s:Name>

                   …

                   </rdf:Description>
                </rdf:Sequence>
             </s:Sequence>
          </rdf:Description>
       </rdf:Sequence>
    </s:Sequence>
  </rdf:RDF>
```

Figure 3. RDF expressed within XML syntax

http://www.cems.uwe.ac.uk/~phale/XMLDemonstrators/rdfring.txt

3.3 *Distributed Computing*

Organizations have experienced a trend from centralized computing to distributed computing. Mainframe computers held all the data and programs. All the main processing was centralized and users could access the computer by means of a ‘dumb’ terminal. The terminal did none of the information processing itself.

In the 1980s and 1990s the Personal Computer (PC) became popular both for work and for personal use. The benefit of this was that it empowered users. As the PC modelled a part of the organizational system, this meant that users who previously did not have the opportunity to influence and understand an organization now did have such a chance. This increase in opportunities has also meant a loss of central control of information. A user can create a document, a model, or a database on a personal computer, which may reside on that computer only, and in a way that is not consistent with the rest of the organization's software. If that employee leaves the company, the information might be lost.

Distributed systems have become the norm and organizations are now trying to regain ownership and control of their information. The organization creates or contracts to a managing authority, which has responsibility for providing and running the computer system. Installation of programs is on a formal basis with this authority's permission. But whenever guidelines are issued centrally, people find ways of circumventing them. Employees of manufacturing companies are no exception. When they are frustrated at the time it takes to have a centralised application developed and approved, they will create a standalone version of this application. This can be achieved through Microsoft Office and its' macro languages. Organizations may be tempted to take back control in order to prevent this happening, but this would be practically difficult. It would also prevent engineers using their greater domain knowledge to create programs based on sound engineering information. This would not empower users instead it would recreate the division between users and software developers, which distributed computing was supposed to bridge. Therefore, alternative strategies must be investigated.

Ways must be found for providing a language 'bridge' between the user's domain and computing domain. This could enable software developers to create the distributed system shell, and engineers to add information using a structured language.

3.4 *Structured languages*

Busy users will not be interested in a system that is difficult to use, but the system must give reasonable results. So it is necessary to make it as easy as possible for a user to enter the information the software needs. Structured languages can be the solution to this problem because they can be understood by a user, and the language is constructed using mathematical rules. So the structured language presents a mathematical representation to the computer and a natural language or diagrammatic representation to the user. (Borthick et al. 2001) explain however, that ambiguity in natural language can make it difficult to translate natural language into SQL.

It is possible to create an extra layer to enable users to specify commands in structured language. This approach of adding extra layers is the way visual programming works. Users provide the information the program needs at the visual interface layer and program code is created automatically. The layers provide the bridge between abstract ideas and computer code. If this approach is taken to its logical conclusion, we could allow the user to specify what the computer should do. Then each layer would communicate this to the layer below until the computer performs the action required. A simple example of this approach is the use of spreadsheets. A user can specify a calculation in mathematical terms using a formula. The spreadsheet then calculates the result of the formula. The user can change the formula if it is incorrect without any need to write code or re-compile. This accounts for the popularity of spreadsheets.

However, spreadsheets do not provide the centralised and structured data-store required for a distributed system. Such systems can be made much more powerful if the information is codified into a relational database structure. Then On-Line Analytical Processing (OLAP) can be used for more sophisticated data collection and analysis. (Lau et al. 2001) explain how OLAP displays a multi-dimensional view of aggregated data, and presents a Rule-Based Analytical Processing (RBOLAP) model which can be used for decision support. The use of RBOLAP techniques is demonstrated using a case study on a mould and die information network.

Sutton (2001) and Huber (2001) illustrate how codifying knowledge into a knowledge based system for decision support is likely to be very difficult. Most people 'just do' a task and therefore never write down instructions for others. This highlights the difficulty of getting information into a knowledge base when it may be either only in individuals' minds, or completely unstructured.

3.5 *Web Simulation*

The development of visual user interfaces was a major step forward. The use of pictorial metaphors such as folders to represent a collection of files has greatly aided human computer interaction. This technique can be used more dynamically in simulations. Simulations represent the real world problem and provide constant feedback to the user. In this sense, all software should be regarded as a simulation, software models some aspect of the real world. Pictorial metaphors are static, while a person's mental model is made up of mental images connected together by a set of actions. People remember by running a mental model like a simulation. So a dynamic simulation is more memorable and conveys more information than a static picture.

Simulation systems should be designed to link with other software so that data can be fed to and from these other systems. A simulation can then be a sub-system within a larger knowledge system. The web environment is suitable for simulations which may be part of the overall decision support system, because Internet protocols can be used for distributed applications, and because Java and the .NET languages both support the creation and/or use of web components and web services. This could make possible the design and creation of web simulation by assembly of appropriate software components. (Stevens et al. 2001) examines UML design of simulation system components. (Kuljis and Paul 2001) explains how XML and VRML (Virtual Reality Modeling Language) can be used for simulation. Kuljis and Paul give a summary of the aims of the AARIA project for manufacturing simulation over the Internet. They argue the need for web-based simulations to be focussed on solving real-world problems in order to be successful. (Miller et al. 2001) also explains the technology behind web-based simulations, and argues the need for demonstration of web-based simulations for major projects.

4 CURRENT RESEARCH

4.1 *Architecture*

Figure 4 shows the architecture of the DECIDE decision support system. Although much of the software required has been prototyped, it has not yet been fully assembled into such an architecture. This will involve adaptation and integration of components from myself, Marsh, Bru, and Nour.

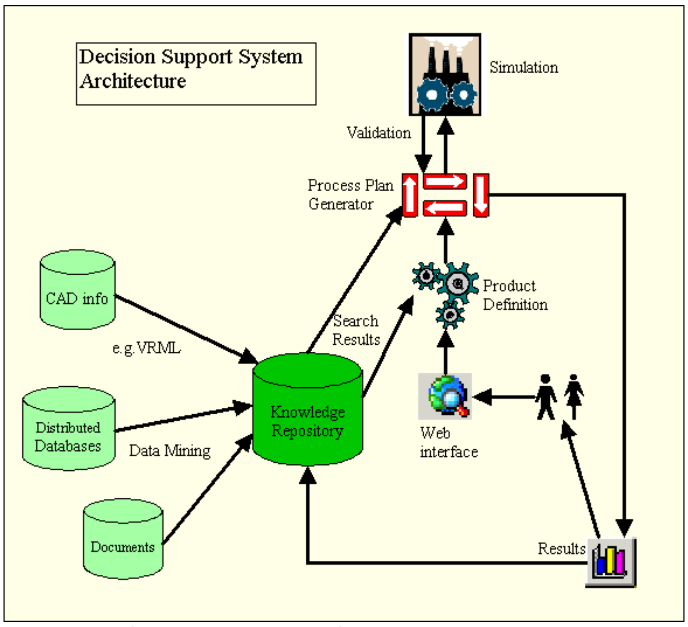


Figure 4. Software architecture for DECIDE system

The research behind this system is now explained in terms of the thinking behind this system and the development of an early prototype.

4.2 *Knowledge Repository*

At present we have a centralised database of product, part, process, and tooling information. XML, and RDF code imposes an object oriented structure on the information accessed in the database. There is not yet sufficient information in our database for data mining techniques to be necessary. Our UML documents have been translated into XMI. Further research is underway into using stylesheet transformations as a way of translating from one text form to another e.g. VRML to RDF. (Kogut & Holmes 2002) have researched DAML (DARPA Agent Markup Language) annotation of web documents. DAML is an extra layer of semantics that can overlay RDF to enable it to represent document structure in a more refined way. This sort of research may make it possible to extract some information from HTML documents for the knowledge repository.

4.3 *User Driven Programming*

A computer should interpret and understand the information provided by the user. It is not very practical for the users to specify changes to the software provider every time the manufacturing process to be modelled changes. It would be much more useful to allow users to drive the program by specifying what they want and allowing the program to respond dynamically, as is achieved with COINER. So it is necessary for our project to provide a Web version of this functionality.

(Lee et al. 2000) present a distributed visual reasoning system for intelligent information retrieval on the Web. The user can design a query by linking active icons, and then inputting the required parameters. The user can then see the structure of the query and obtain results from the information database.

Figure 5 is a modification of the GraphLayout demonstration that comes with Java jdk1.1.4 release and was created by (Nguyen 2001). This type of user interface could allow a user to create a program using structures and visual queries and is similar to the COINER visual interface.

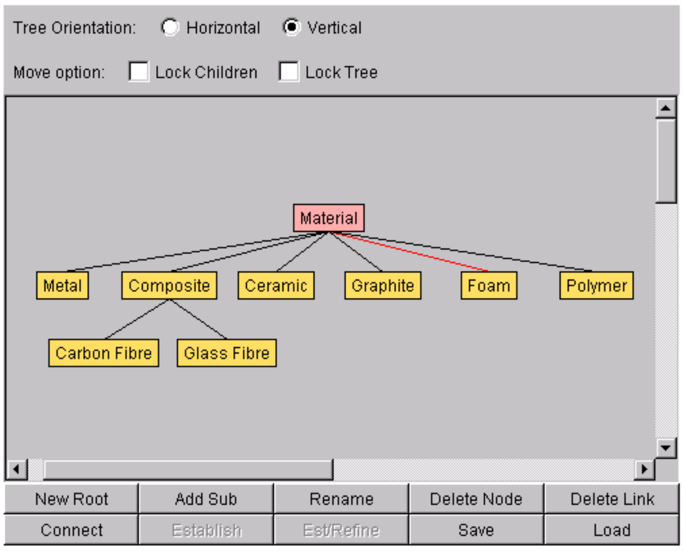


Figure 5. User interface for user driven program

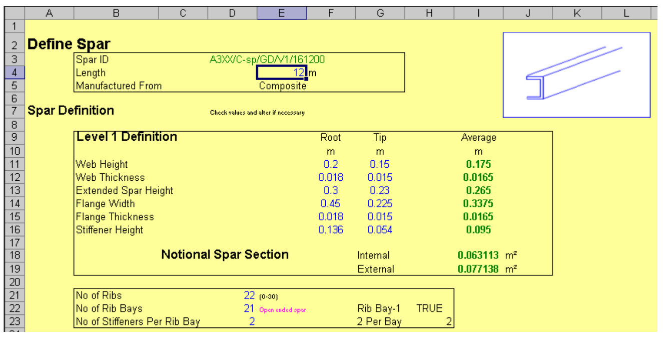
Such an interface can be used to create taxonomies or to send information to a Web decision support program. As well as Inheritance relationships, such diagrams could model a manufacturing process flow. The ontology can be used to limit terms to only those which the organization accepts.

We are researching (Web Service Description Language) and UDDI (Universal Description, Discovery and Integration). This involves creating web services and allowing client programs to use them. The prototype uses XML to structure documents and navigation. Our intention now is to use XML and web services to gather information, structure programs, database(s), and computer communication.

The database structure is reflected back to the users as structured documents. For example, for a parts data structure, there is a parts mapping. The parts mapping is the intermediary between the client, which requests the parts data and the underlying data source which provides it. The parts mapping would ensure the data was properly structured and semantically correct. The semantics could be checked by ensuring an agreed taxonomy is used with values defined by their position within the structure, rather than merely by arbitrary labels.

Sometimes it might be necessary to convert unstructured documents into XML. An example is the conversion of Excel sheets to XML, which can be achieved using automated tools. (Liu et al. 2001) describes an XML enabled wrapper construction system. The wrappers convert HTML documents into XML, thus bringing it within reach of sophisticated query tools, and within a defined taxonomy.

When constructing our prototypes it was thought useful to use the spreadsheet interface familiar to many users. However, installing spreadsheets on client computers does not provide multiple user access. Our intention has been to provide a centralised spreadsheet, which makes use of a structured database to store the information, which it relies upon. A further advantage of this is that the database can be accessed by other applications. Figure 6 is an example page from our implementation of a decision support tool for manufacture of tooling for wing spar manufacture. This was created using ExcelWriter, software (SoftArtisans 2002) for creating or convert-



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2	**Define Spar**											
3		Spar ID		A3XX/C-sp/GD/V1/161200								
4		Length			12	m						
5		Manufactured From			Composite							
6												
7	**Spar Definition**			Check values and alter if necessary								
8												
9		**Level 1 Definition**				Root	Tip		Average			
10						m	m		m			
11		Web Height				0.2	0.15		**0.175**			
12		Web Thickness				0.018	0.015		**0.0165**			
13		Extended Spar Height				0.3	0.23		**0.265**			
14		Flange Width				0.45	0.225		**0.3375**			
15		Flange Thickness				0.018	0.015		**0.0165**			
16		Stiffener Height				0.136	0.054		**0.095**			
17												
18			**Notional Spar Section**				Internal		**0.063113**	m²		
19							External		**0.077138**	m²		
20												
21		No of Ribs		22	(0-30)							
22		No of Rib Bays		21	Open ended spar		Rib Bay-1	TRUE				
23		No of Stiffeners Per Rib Bay		2			2 Per Bay	2				

ing existing spreadsheets, for a distributed web environment.

Figure 6. Distributed Spreadsheet Spar Definition

This distributed spreadsheet communicates with a centralised relational database. This has allowed us to build a manufacturing data source, which can then be re-used for other manufacturing problems, and contribute to a manufacturing taxonomy. We have also created a spreadsheet representation of the Ring Manufacture sequence using ExcelWriter and XML.

To achieve user driven programming it is necessary to produce dynamic programs. A program cannot properly respond to a user just by taking inputs and producing outputs, it must adapt to the user and new requirements. Practical research is needed into how dynamic software systems can be created. (Huhns 2001) explains that current techniques are inadequate, and outlines a technique called Interaction-Oriented Software Development, where agents co-operate to achieve a goal. Huhns concludes that there will soon be a direct association between users and software so that they can create programs, in the same way as web pages are created today. More research is needed into making this a reality.

The APHID system (Thomson et al 2001) makes use of a concept mapping that has relationships between concepts in a course of study. This sort of concept map can also be applied to define a process plan. Another useful idea is that of a hypermedia user interface design pattern. As for software patterns, the intention is to document the user interface design principles. The APHID system partially automates the process of user interface creation.

If we can allow software to understand the structure of information, the job for the user of passing this information to the computer and understanding the result can be made much easier. Then it can become possible to gather and store all the knowledge in a central repository.

Each user's knowledge will be different. Thus, it is necessary to adapt the user interface to accept different information from each type of user in a standardised interface. If the system can be made to know what information it has received, it can also know what information is missing and therefore where to make inferences. The system can also check whether the user can improve the accuracy of the estimate by adding further information. Intelligent agent objects can be used to achieve this. An intelligent agent is an object that can react and adapt autonomously (Grove 2000, Sycara 1998).

(Ciancarini et al. 2001) explain that web documents can be generated on the fly. This can allow the user interface to respond dynamically to choices. Code can be attached to the documents themselves. This code can activate certain behaviour based on the XML content of the document. Code can also be created separately and called on as a service when a document needs it. Java or the Microsoft .NET languages can provide information mapping and calculations. Scripting languages assist with the provision of the user interface, and passing of information. Some of this technology is used in our decision support system. Once a user enters a product definition, this can trigger script code within the web form receiving the information. This code could pass the product definition to a web service. The web service could check its' database of appropriate process plans and infer the closest match. The user can then accept this process plan, or amend it. A good explanation of Microsoft .NET is (Mingins et al. 2001).

This use of the Web as a link to a multitude of applications is termed the Active Web and is described in (Morris et al. 2001). This paper also explains some current limitations on the technology required to make this vision attainable. The use of the web as a 'WWW shell' is compared and contrasted with expert system shells. This is particularly appropriate to our research as we intend to develop a distributed WWW shell version based on the COINER Expert System Shell software previously developed. This Expert Shell provides decision support for manufacturing. Morris et al. illustrate that Web technologies are not ideal for creating an expert system as workarounds are needed to duplicate the functionality. The advantages of using the Web are in its' wide user acceptance and ability to support a distributed application.

5 CONCLUSIONS AND FUTURE WORK

This paper has given an idea of what tools and technologies could be used in a distributed decision support system. It would be possible to update and improve such a system as it would be assembled from many interactive components. The system must be flexible enough to respond to a wide range of user queries; this entails the need for a sophisticated ontology and the use of structured language. This ontology is currently represented using XML, XMI and RDF Web Meta-data, which access information from a relational database. This ontology is crucial to development of the distributed system.

Software has been produced as a proof of concept in order to access this information and generate a cost, and a process plan. This has been developed for decision support and costing of aerospace products, but eventually will be applied to more general design and manufacture problems. Further work will be undertaken on a manufacturing simulation system to be integrated into the decision support system. This system is to be regarded as a prototype for a decision support system for wider applications. It is important to identify those tools and techniques necessary for construction of any decision support system and apply these to our prototype.

A wider goal is to enable users to add information to a structured distributed knowledge based system.

REFERENCES

5.1.1 *References*

Baclawski, K. & Mieczyslaw, K. & Kogut, P. & Hart, L. & Smith, J., & Holmes, W. & Letkowski, J. & Aronson, M. 2001. Extending UML to Support Ontology Engineering for the Semantic Web http://ubot.lockheedmartin.com/ubot/papers/publication/UMLOntology.pdf, accessed on 8th April 2002

Berners-Lee, Tim 1999. The future of the Web, http://www.w3.org/Talks/1999/0414-LCS35-tbl/slide1-1.html, accessed on 8th April 2002

Borthick, A. F. & Bowen P. L. & Donald R. J. & Micauel H. K. T. 2001. The effects of information request ambiguity and construct incongruence on query development. *Decision Support Systems* 32: 3-25

Bru, C. & Scanlan, J. & Hale, P. & Dunkley, M. 2002. Visualisation of Cost Information, *9th ISPE International Conference on Concurrent Engineering: Research and Applications, Cranfield University.*

Brundwick, Bill (ed.) US Dept. of Defense 1995. Parametric Cost Estimating Handbook Joint Government/Industry Initiative

Bryan, M. (ed.) 2002. Context: The key to understanding ontologies. http://www.personal.u-net.com/~sgml/context.htm, *The SGML Centre*, accessed on 8th April 2002

Ciancarini, P. & Rossi, D. & Vitali, F. 2001. Designing a document-centric coordination application over the Internet. *Interacting with Computers* 13: 677-693

Dennis, G. B. 2001, Koala Electronic Publishing. http://www.koalapub.co.uk/, accessed on 8th April 2002

Duverlie, P. & Castelain, J. M. 1999. Cost Estimation During Design Step: Parametric Method versus Case Based Reasoning Method. *Advanced Manufacturing Technology*, 15: 895-906

Eaglesham, M. 1998. A Decision Support System for Advanced Composites Manufacturing Cost Estimation. *Virginia Polytechnic Institute and State University, PhD*

ebXML 2002, ebXML Enabling a Global Electronic Market. http://www.ebxml.org, *OASIS & UN/CEFAC*, accessed on 9th January 2007

Fensel, D. & Angele, J. & Decker, S. & Erdmann, M. & Shnurr, H. & Studer, R. & Witt, A. 1998. On2broker: Lessons Learned from Applying AI to the Webhttp://www.aifb.uni-karlsruhe.de/WBS/Publ/2000/llfaattw_dfeetal_2000.pdf accessed on 9th January 2007

Ferreira, D. & Ferreira, J. 2001. Designing Workflow-Enabled Business-to-Business Infrastructures. *7th International Conference on Concurrent Enterprising:* 81-89

Grove, R. 2000. Internet-based expert systems, *Expert Systems* 17(3)

Huang, G. Q. & Mak, K. L. 2001. Issues in the development and implementation of web applications for product design and manufacture. *Computer Integrated Manufacturing* 14(1): 125-135

Huber, G. P., 2001, Transfer of knowledge in knowledge management systems: unexplored issues and suggested studies. European Journal of Information Systems, Vol 10 pp 80-88.

Huhns, M. N. 2001. Interaction-Oriented Software Development. *International Journal of Software Engineering and Knowledge Engineering* 11(3): 259-279

Hunter, A. 2002. Engineering Ontologies, http://www.cs.ucl.ac.uk/staff/a.hunter/tradepress/eng.html, accessed on 8th April 2002

Kogut, P. & Cranefield, S. & Hart, L. & Dutra, M. & Baclawski, K. & Kokar, M. & Smith, J. 2002. UML for Ontology Developmenthttp://whitepapers.techrepublic.com.com/whitepaper.aspx?docid=92641 accessed on 9th January 2007

Kogut, P. & Holmes, W. 2002. AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. http://ubot.lockheedmartin.com/ubot/papers/publication/AeroDAML2.doc, accessed on 8th April 2002

Generate DAML Annotations from Web Pages

Kuljis, J., Paul R. J., 2001, An appraisal of web-based simulation: whither we wander?. *Simulation Practice and Theory* 9: 37-54

Lau, H. C. W. & Bing, J. & Lee, W. B. & Kau, K. H. 2001. Development of an intelligent data-mining system for a dispersed manufacturing network. *Expert Systems* 18(4)

Lee, C. & Chen, Y. T. 2000. Distributed visual reasoning for intelligent information retrieval on the web. *Interacting with Computers* 12: 445-467

Liu, L. & Calton, P. & Wei, H. 2001. An XML-enabled data extraction toolkit for web sources. *Information Systems* 26: 563-583

Marsh, R. & Hill, T. & Scanlan, J. & Dunkley, M. & Cleevely, P. 2000. Probabilistic Pseudo-generative Cost Modelling Through Virtual Template Propagation. *CEAS Conference on Multidisciplinary Aircraft Design and Optimisation*: 49-55

McGuinness, D. L. 2002. Conceptual Modeling for Distributed Ontology Environments. http://www.ontology.org/main/papers/iccs-dlm.html, accessed on 8th April 2002

Miller, J. & Fishwick, P. A. & Taylor, S. J. E. & Benjamin, P. & Szymanski, B. 2001. Research and commercial opportunities in Web-Based Simulation. *Simulation Practice and Theory* 9: 55-72

Mingins, C. Nicoloudis, N. 2001. *.NET:* A New Component-Oriented Programming Platform. *Journal of Object-Oriented Programming*: October/November 2001

Morris, S. & Neilson, I. & Charlton, C. & Little, J. 2001. Interactivity and collaboration on the WWW - is the 'WWW shell' sufficient?. *Interacting with Computers* 13: 717-730

NASA, US Air Force. Kelley, Cyr (ed.) 2002. NASA/Air Force Cost Model. http://www.jsc.nasa.gov/bu2/NAFCOM.html, accessed on 8th April 2002

Nguyen, Luan 2001. Classifier Editor. http://ai-web.ecst.csuchico.edu/tasks/classification/Nguyen/Chico/project/Editor/Editor.html, accessed on 8th April 2002

Nour, M. 2001. Modelling the Thinking Process of an Aerospace Design Organisation. *CEAS Conference on MultidisciplinaryAircraft Design and Optimization, Maternushaus Koln, Germany*

Roy, R. & Bendall, D. & Taylor, J. & Jones, P. & Madariaga, A. & Crossland, J. & Hamel, J. & Taylor, I. 2002. Development of Airframe Engineering CER's for Aerostructures. http://www.cranfield.ac.uk/sims/cim/research/publications/paper_files/roy/paper1.doc, accessed on 8th April 2002

Scanlan, J. & Hill, T & Marsh, R. & Bru, C. 2000. Cost Modelling for Aircraft Design Optimisation, *Engineering Design 2000 conference, Brunel University.*

Softartisans. 2002. ExcelWriter. http://www.softartisans.com/excelwriter.html, accessed on 8th April 2002

Stevens, P. & Pooley R. 2000. Using UML Software Engineering with Objects and Components. ISBN 0-201-64860-1

Sutton, D. C., 2001, What is knowledge and can it be managed?. European Journal of Information Systems, Vol 10 pp 72-79.

Sycara, K. P. 1998. The many faces of agents. *AI Magazine* 19(2)

Thomson, J. R. & Greer J. & Cooke J. 2001. Automatic generation of instructional hypermedia with APHID. *Interacting with Computers* 13: 631-654