15th Triennial World Congress, Barcelona, Spain

AN XML-BASED DATA EXCHANGE ARCHITECTURE FOR CACE

T. Varsamidis, R. G. Edwards

School of Informatics
University of Wales, Bangor, UK

Abstract: Control systems design tools that have not been specifically constructed to work together may not easily exchange data. The purpose of a data exchange mechanism is to standardise and automate this process for all tools supporting the underlying technology. This paper proposes a control systems design data exchange architecture based on the recently developed XML standard, which allows tools to improve their data exchange capabilities and at the same time benefit from a wide range of data services and data communication possibilities offered by XML. *Copyright © 2002 IFAC*

Keywords: computer architectures, computer-aided control systems design, data models, information integration, object modelling techniques.

1. INTRODUCTION

During the 1990s a number of methods for providing standardised means for information exchange were developed in a variety of research-based and commercial-based areas such as Software Engineering (Long, and Morris, 1993), the automotive industry (Wandmacher, 1997), and the petrochemical industry (POSC, 1995). Similar approaches have been presented in the area of Computer-Aided Control Engineering (Grübel 1994; Varsamidis, *et al*., 1994a). Proposals for data exchange standards in all these areas ranged from simply describing a common data model to be used by all software tools, to defining a complete set of data-related specifications including the data model, the data exchange mechanism, the communication protocol and the support services mechanisms. It is now acknowledged that in the case of many software tools collaborating towards a common design goal, as is the case of Computer Aided Control Systems Design, the definition of a coherent data exchange approach is key towards the successful set up of tools that can exchange information (Bass, and Kazman, 1999).

2. THE UNIFIED INFORMATION MODEL CACE ARCHITECTURE

The Unified Information Model (UIM) architecture defined by Varsamidis, *et al*., (1994b) satisfies the requirements for a data exchange mechanism for CACE through the use of the EXPRESS ISO standard modelling language (ISO, 1994) as the medium for describing and propagating CACSD information. The Unified Information Model is a data model which gives a hierarchical description of all data structures encountered in the lifecycle of a control systems design project, ranging from the low-level data structures for control system description to the (abstract) high level of a project-based approach for a control systems design (fig. 1). The model does not attempt to redefine the structures of information that may be applicable to each stage of the design lifecycle. Instead, it allocates ‘space’ for existing and proven data structures in a wider view of the whole set of information used. In other words, the UIM is an information model that defines the links and relationships between specific levels of information in the design lifecycle (for example, it maps experimental results to certain versions of control system models which are subsequently linked to specific sets of model parameter values). The model itself is a ‘container box’ for existing data structures that can still be freely used as-is by CACSD tools; only once inside the UIM they become part of a more generic description of control systems design information. The approach is generic enough so as to allow for different types of data structures to be handled as separate or complementary types of information as, for example, in the case of a control system model which may come in either graphical or algebraic form. The ability of the UIM model to handle such diverse forms of data representations is based on the requirement that the architecture does not aim at directing the flow of information but rather assist it.

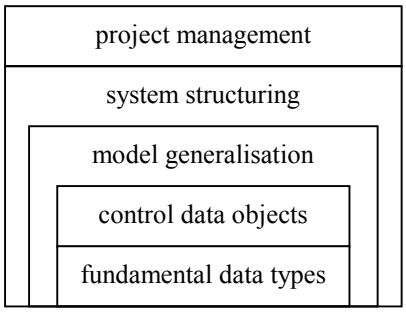

Fig. 1. Levels of abstraction of control systems design information.

The UIM was originally implemented in the modelling language EXPRESS, which in turn makes use of the neutral, ASCII-based STEP data file format for the description of control information. Exchange of CACSD information through this medium presupposes that the tools involved in the exchange are aware of the UIM EXPRESS model, though they may only be making partial use of it. Because of the standardised neutral format that EXPRESS offers, it is guaranteed that the software tools involved in the exchange will be able to parse the information. Additionally, the global picture of all information that the UIM offers ensures that this combination of the information model and the data exchange mechanism can support any type of tool operation in the design lifecycle.

For creating a system that automates data exchange, it is necessary for the system to not only understand all information to be handled, but to also provide concrete mechanisms for directing the exchange process. This means that not only does the software need to know how to pass the data around when prompted to do so, but that it can also initiate the relevant software operations with as little involvement of the user as possible. In other words, in the ideal control systems data exchange, one would want to click on button at the sender tool and have the data appear in the receiving tool. However, such a high level of automation requires that the communicating tools have to incorporate the appropriate implementation mechanism for this functionality; if the do so, it is advantageous to use a standardised system for the purpose.

EXPRESS data files (known as STEP files) can describe information but cannot make the data's presence known to other software, which is why the UIM approach couples EXPRESS with the message passing mechanism of CORBA (OMG, 1995). CORBA is a standardised architecture for the communication of software; it enables software processes (for example, the "Send Data" option found in a tool) to send messages to other software processes for inquiring about their capabilities, or for updating their status or remotely invoking them (as the case would be with a possible "Receive Data from External Tool" function). CORBA forms the communication link and software tools can then connect to each other; STEP file information is streamed onto the communication channel and with the knowledge of the UIM information arrangements

Fig. 2. Using the Unified Information Model, EXPRESS, STEP and CORBA to exchange data between a CACSD tool and a database.

the tools selectively send or receive what is of interest to them. This final addition of CORBA to the UIM architecture completely automates the data exchange process and allows for an open approach to software communication, where a variety of tools can offer or receive not only data, but also other services to or form other CACSD tools. The outline UIM-based data exchange architecture for CACSD can be seen in fig.2.

3. APPLYING A UIM-BASED DATA EXCHANGE THROUGH XML

The recent advent of web-oriented computing has lead to the emergence of new requirements for the communication of information. This is due to the vast potential range of receivers for each piece of information and the fact that these receivers may not all be interested in the same aspects of certain information, or may have a different interpretation of it. The situation can be seen as an extended version of the CACSD tools communication problem described above, i.e. tools may need the same information for different purposes, or may represent the same information in different ways.

A key technology in the attempt of making information spread through the web in a standardised format is that of the eXtended Markup Language, XML (W3C, 2000). XML is an ASCII-based tagged data description format. Any piece of data that is to be treated as an element of a certain data type is tagged with the appropriate set of tag identifiers. For example, some amount of money could be described as XML data in the form of

```
<pocketMoney>
  <dollarsInNotes>55</dollarsInNotes>
  <dollarsinCoins>1.75</dollarsInCoins>
</pocketMoney>
```

Software receiving this data set may then read the tags and decide whether the data is of use to it or not, and if it is, how to handle it, e.g. for the example given here, sum up the amount and present it to the user in some other currency. For each of the tages to be understood, it must have already been defined in an appropriate XML schema, which is a document enlisting the set of tags to be used for all relevant data, along with the rules of constructing the

information. It is easy to see that, given the proper set of tags defined, one may represent all target-specific information in a way that is software independent. This concept very well matches the aims of CACSD tool data exchange using current technology in the same way that similar approaches have been outlined from very early on in CACSD (Rimvall, 1986) with the technology of the day.

However, the scope and potential of XML is much wider than simply describing data. A number of XML-related technologies are now emerging, and we are currently shifting the original UIM implementation to accommodate these new paradigms. Our aim is to use these technologies to build a software system for CACSD that satisfies the data exchange requirements for

a. tool independent description of information
b. coverage of all control systems design lifecycle information
c. layered definition of control systems structures, models and submodels
d. automated control of the data exchange
e. well defined ways of transforming the information to match the specific CACSD tool data representation

With XML covering the requirement for (a), the following sections present a set of complementary recent standards and recommendations that address the remaining of the requirements while taking advantage of the latest web-enabled technologies.

3.1 Coverage of all control systems design lifecycle information

The control systems design process involves a number of steps that differ in the type of information they require and produce. For example, a simulation stage may result in a set of matrices representing graph data, while running a set of experiments may require information on the version of the models used, a graphical representation of their structure and the value of certain parameters.

In order to be able to produce XML documents describing such a variety of control systems design information, it is therefore necessary to make use of a rich set of appropriate data tags. A complete set of such definitions for the meaning of control engineering information can be given in the form of what is known as an XML schema (W3C, 2001b). The CACSD XML schema then acts as the information context for any related XML data. Any tool wanting to send or receive data can do so by accessing the CACSD schema and format or interpret the data accordingly. This is a standard way for software tools for understanding information, as witnessed in the case of web browsers which can display previously unknown types of information read in XML documents. The potential of the mechanism is such that standards bodies such as OASIS (2001) are currently shaping recommendations for schemas for a large number of application areas such as. Therefore, the CACSD XML schema approach offers the opportunity for the definition of standardised CACSD data structures for use with the new medium of XML, which is flexible, extensible, widely used and well supported.

3.2 Layered definition of control systems structures

As fig. 1 shows, CACSD information is not ‘flat’. It may be classified from fine-grain (as in the case of fundamental control data types) or have a high degree of abstraction, e.g. the information related to a CACSD project. Furthermore, specific CACSD data structures in this hierarchy may also be aggregations of other data structures: a control systems model may have a mathematical representation while at the same time it consists of other simpler control system models connected to each other.

A requirement for the CACSD data exchange is that the exchange medium can capture the breadth of these representations. XML is well suited to this task as it can describe data of arbitrary complexity in hierarchical fashion. The following simple example shows one possible way that the structure of a matrix can be declared in an XML schema.

```
<complexType name="matrix">
 <sequence>
  <element name="rows"
        type="xsd:integer"/>
  <element name="columns"
        type="xsd:integer"/>
  <element name="data"
        type="xsd:float"
        maxOccurs="unbounded"/>
 </sequence>
</complexType>
```

Based on the above XML schema, one may define the following XML description of a matrix with its data.

```
<matrix>
  <rows>2</rows>
  <columns>1</columns>
  <data>12.34E-5</data>
  <data>0</data>
</matrix>
```

The XML schema example states that a matrix type contains other elements inside it, some of which may be present more than once (`maxOccurs` for `data` is `"unbounded"`). Further refinement may be applied by controlling these elements. For example, a rule may be set that the number of `data` be equal to `rows * columns` . The existing types can then be used as elements of more complicated ones. Such definitions can be used to build data structures to any desired level of complexity.

The previous implementation of control data structures in the UIM-based data exchange used

EXPRESS for the description of data. Recent reports on the comparison of EXPRESS with XML suggest that the latter can accommodate the structures of the former (Barkmeyer, and Lubell, 2001). This ensures the smooth transition of the architecture into the new data definition paradigm and, in general, promises backward compatibility with systems based on the EXPRESS standard.

3.3 Automated control of the data exchange process

Once the definition of a common data model for the exchange of control systems information is in place, additional XML-related technologies can improve this data transfer. As discussed with the case of CORBA for the UIM data exchange architecture, it is desirable for the system to automate the process of requesting, retrieving and sending data to other tools, so as to minimise user involvement. Instead of having to save the data of a source tool using a specific set of options, then convert it and finally load it to the target tool, it is better to make tools exchange data 'transparently' – that is, become aware of where the data needs to go and then themselves arrange for the exchange configuration details. XML-related mechanisms providing such functionality include the XML Metadata Interchange specification, XMI (OMG, 1999), and the Simple Object Access Protocol, SOAP (W3C, 2001a). With these mechanisms, automated exchange is not restricted to pairs of tools, but rather any compliant tool can 'talk' to any other tool in the design suite. The use of such a specification in the control systems design environment is a realistic approach that improves the quality of services of the tools by offering a number of advantages with respect to the exchange of information. In the case of XMI (Iyengar, 1999):

- is a neutral, platform-independent open interchange format for data in distributed environments
- works with the internet and builds on existing industry standards
- is easy to implement in current software tools
- breaks the wall between incompatible tools, repositories and applications
- it is stream-based, i.e. the data to be handled can be stored in a traditional file system or streamed across the Internet

Although the details of these mechanisms are beyond the scope of this paper, it is important to note that they can be readily used where necessary for enhancing the operation of the CACSD tools through the automation of the exchange process.

3.4 Well-defined ways for transforming information

Given the range of data that is used in the CACSD design lifecycle, it is expected that not all tools exchanging data will be able to understand the data format coming from other tools, even if it refers to the same information. Our research in the case of the UIM-based data exchange mechanism has shown that it is common for control tools to want to follow the path of: requesting data from a source tool; convert it to the appropriate format; perform some operations on it; convert the results back to the source tool format; return the results to the source (Varsamidis, *et al*., 1997). What this means is that tools may be referencing the same data structures, but this does not guarantee that they will also view them in the same way.

Another potential problem is that sometimes a tool may selectively pick some data from a larger dataset and ignore its context information. For example, if the designer extracts a control systems model from a database, changes it, and then sends it back to the database, it may be desired for the database links to the old model to be redirected to the new version of it. But this can only be done if the model that was extracted from the database has not lost its 'identity mark', which may have been treated as context information and thus disregarded by the extraction process. Because of such problems, it is necessary to ensure that data does not lose its original identity, context or relation to other data structures during the communication of tools. In other words, the transformation of information must always be complete and verifiable.

Although it is possible to devise ad-hoc ways for performing such transformation tasks, XML provides the XML Stylesheet Language Transformation recommendation, XSLT (W3C, 1999), whose purpose is to allow only valid ways of moving XML data from some source information pool to a target one. It is obviously the responsibility of the designer to decide what constitutes a valid transformation. With such XSLT-based rules in place, any data exchange is guaranteed to be correct in terms of structure, i.e. with no incomplete, unreferenced or non-valid data descriptions.

As with some of the previous mechanisms discussed, the purpose of XSLT may not directly relate to the design process for control systems; however, it is a key tool for ensuring the validity of the information to be exchanged. This technology can then enhance the quality of the design process and relieve the user from the burden of having to verify the correctness of the exchange process. In this respect, XSLT is yet another way of improving data management in a control systems design environment.

4. CONCLUDING REMARKS

We have presented the concept of data description with XML and how this can be used to enhance any task involving data in the computer-aided design of control systems. Along with XML, a number of supporting technologies have been introduced, to demonstrate the strength of the standard and the potential benefits of using these technologies in CACSD.

Our implementation of the XML-based data exchange architecture extends the scope of the Unified Information Model approach and aims to updating the system to the latest technology. Implementation has only started very recently and is work in progress. However, the concepts involved in the design do offer a solid ground for pursuing the development of mechanisms that can have significant impact in the use of control systems design tools.

XML is a recently developed, platform-independent technology that has been adopted very quickly by software tools in a diverse range of application areas. This is due to its expressive power for modelling information and the support services it offers for data, including exchange, storage, distribution, and transformation. These services closely match the need for software quality and ease of use in the control systems design process. We are proposing an architecture for incorporating all these standard mechanisms in a system that can support all compliant control system design tools throughout the CACSD design lifecycle. The resulting mechanism does not restrict the functionality of any of the tools while at the same time it enhances their abilities with the provision of functions that aim directly at improving the quality of the design process for the control engineer.

REFERENCES

Barkmeyer, E.J, and J. Lubell (2001). XML Representation of EXPRESS Models and Data. *XML Technologies and Software Engineering (XSE2001)*. Toronto, Canada.

Bass, L., and R. Kazman (1999). *Architecture-Based Development. Technical Report CMU/SEI-99-TR-007, ESC-TR-99-007.* Carnegie Mellon Univerisity, PA, USA.

Grübel, G. (1994). The ANDECS-CACE Framework R_SYST for Integrated Analysis and Design of Controlled Systems. *Proc. of the IEEE/IFAC Joint Symposium on Computer-Aided Control Systems Design, CACSD '94.* Tucson, AZ, USA.

Iyengar, S. (1999). Overview of XMI. *XMI Technology Briefing Presentations.* http://cgi.omg.org/news/pr99/completeXMI.ppt, Object Management Group.

ISO (1994). *Industrial Automation Systems and Integration Standard 10303: Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual*. International Standards Organization.

Long, F., and E. Morris (1993). An overview of PCTE: A Basis for a Portable Common Tool Environment. *Technical Report CMU/SEI-93-TR-1, ESC-TR-93-175.* Carnegie Mellon University, PA, USA.

OMG (1995). *The Common Object Request Broker Architecture and Specification, revision 2.0.* Object Management Group, Inc., Framingham, MA., USA.

OMG (1999). *XML Metadata Interchange*. http://www.omg.org/technology/xml/, Object Management Group, Inc., Framingham, MA., USA

POSC (1995). *Software Integration Platform Specification, Epicentre Data Model, Version 2.0*. Petrotechnical Open Software Consortium, Houston, TX, USA.

Rimvall, C.M. (1986). *Man-Machine Interfaces and Implementational Issues in Computer-Aided Control System Design*. Diss. ETH No. 8200, Swiss Federal Institute of Technology.

Varsamidis, T., S. Hope and C.P. Jobling (1994a). Information Management for Control System Designers. *IEE Int. Conf. Control'94*. Coventry, UK.

Varsamidis, T., S. Hope, and C.P. Jobling (1994b). A Unified Information Model for Computer-Aided Control Engineering. *Proc. of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design, CACSD'94*, Tucson, AZ, USA.

Varsamidis, T., C.P. Jobling and S. Hope (1997). Towards a Repository Service for Computer-Aided Control Engineering. *7th IFAC Symposium on Computer-Aided Control Engineering, CACSD'97.* Gent, Belgium.

W3C (1999). *XSL Transformations (XSLT) Version 1.0*. http://www.w3.org/TR/xslt-19991116, World-Wide Web Consortium.

W3C (2000). *Extensible Markup Language (XML) version 1.0 (Second Edition)* . http://www.w3.org/TR/2000/REC-xml-20001006, World-Wide Web Consortium.

W3C (2001a). *Simple Object Access Protocol, version 1.1*. http://www.w3.org/TR/SOAP/, World-Wide Web Consortium.

W3C (2001b). *XML Schema*. http://www.w3.org/XML/Schema, World-Wide Web Consortium.

Wandmacher, R. R. (1997). How SC4 Meets the Business Need. *IEE Meeting on ISO 10303 – Standard for the Exchange of Product Data*. London, UK, March 1997.