


# Vendor independent control database for virtual preparation and formal verification

# Vendor independent Control Database for Virtual Preparation and Formal Verification

Petter Falkman, Jonathan Hedvall, Anders Holmblad, Bengt Lennartson

Control and Automation Laboratory Department of Signals and Systems
Chalmers University of Technology S-412 96 Göteborg, Sweden
petter.falkman@chalmers.se

***Abstract*— It is very advantageous to use virtual techniques for testing and developing new hardware and software systems within a manufacturing system. It is, however, of greatest importance that the virtual model can be trusted so that the results of the development and tests can be directly transferred to a real system without any manual last minute changes. In order to trust the result, formal verification techniques can be applied and by doing that guaranteeing a correct system behavior. Today, there is a gap between how systems are modeled in simulation softwares and formal verification softwares and it is therefore hard to perform formal verification. In order to limit the risk of introducing errors it is also important that the specifications created in the simulation softwares are not manually converted into formal languages. The present paper presents a method for sharing information between the different virtual development tools and formal verification tools. A database, storing necessary control information for verification and controller synthesis, is presented.**

**Keywords:** Information exchange, Virtual production, Production preparation, Formal verification, Discrete event systems.

## I. Introduction

The rivalry on the consumer market is turning more intense than ever. Many companies compete by applying advanced technology and good competence. Manufacturers are constantly looking for ways to save seconds in the production preparation process. The ramp up time is often long, and it can take time before a factory is producing at its maximum capacity. Within the production planning area more and more development and testing is performed using different virtual tools and the trend is to continue to increase the amount of tasks that are included in the virtual development and testing. The main reasons is that it is cost effective since no real machine or manufactory system is needed. It is also safe since there are no real components that can be damaged. Novel concepts can be tested prior to manufacturing. The aim is to be able to develop and test an entire manufacturing station including all machines, robots and components including control software before it is built. This is to decrease ramp-up time when all the equipment has been delivered and installed and decrease cycle-time. New simulation tools such as Siemens Process Simulate [1] and Dassault systems Delmia Automation [2] aims at providing such a system. However, todays simulation tools can only show the presence of errors, not the absence. In order to prove that the production plant does not contain any errors, normal simulation is not enough.

In [16] it is described how it is possible to perform formal verification of a sequence of operations in a manufacturing cell, once you have acquired the required information. It is also said that this required information should be reused from various stages of the development of the production cell. Reusing the information that is already there is shortening the lead times of the verification process by avoiding double work.

In the present paper a software tool called Supremica [4], [5], [6], [7], that implements the supervisory control theory [8] and is based on discrete event systems [9], will be used for the verification and controller synthesis of discrete event behavior of the manufacturing systems. Supremica uses advanced algorithms in order to be able to handle large industry applications. Supremica can detect problems in programs that may not be visible by simulation. Supremica can in this way eliminate costly program lockdowns on the production floor long before the programs leave the development stage by analyzing all possible states a system can be in. A modern manufacturing system normally involve millions of states.

In [16] a method is proposed to standardize the information exchange between the different softwares. In [11], it is shown how control information can be extracted from Process Simulate in a vendor independent format, which is of great importance. The extracted control information can then be optimized and manipulated by different software tools such as the verification tool Supremica [7], [10], [6] before the PLC-code and the station logic is generated.

## II. Framework

The main purpose of the suggested method in [16] is to reuse control information from different development tools by sharing a standardized representation of the control information through a database and to use this control information for verification and controller synthesis. The information needed to describe the control information can be sorted into seven categories:

- Relations of Operations, ROP
- Execution of Operations, EOP
- Cycle Start Condition, CSC

- Interlocks, IL
- Coordinated Operations, COP
- Mechanical Components and Functions, MCF
- Function Blocks, FB

An XML schema, see Section III, has been developed for every category. XML files are then used for communication between the different software tools.

### A. Relations of Operations, ROP

The ROP holds the information of when an operation can be executed. The different ways of executing the operations can be described by a set of relations. There are four different relation types in a ROP: sequential, alternative, parallel and arbitrary order. Sequential relation order is a straight forward sequence that the operations are executed in. The operations are executed exactly the same way every time. An alternative relation is a choice of many operations that can be executed. The way the operations are executed differs from time to time depending on the states of the components in the cell. A parallel relation defines two or more operations executing simultaneously. The arbitrary relation order is similar to the alternative relation order but the choice of operation to execute is, as the name implies, arbitrary. The ROP can be graphically represented, sees Figure 1, in a software tool called Sequence of Charts (SOC) [12].

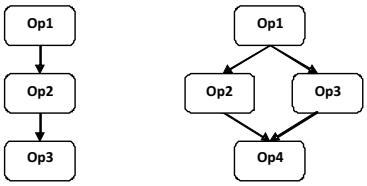


Fig. 1. Two different sequences of operations.

### B. Execution of Operations, EOP

The execution of operations holds information of how an operation is to be executed. On the lowest abstraction level in a cell there are actuators and sensors which can be aggregated into components. The components, in turn, can consist of actuators, sensors and variables. To execute an operation in a safe way, information about booked or free zones in the cell must be available. There must also be information available about what states other machines in the cell are in, see Figure 2. This information is described by a matrix that is be divided into three parts: Internal components (consisting of actuators, sensors and variables), zones, and external components. The initial state of an EOP consists of values that must be fulfilled in order to start the first action. Every row in the matrix describes an action and every action should be executed sequentially [16].

### C. Interlocks, IL

Interlocks have two main purposes. The first is to clearly describe all allowed behavior of the system. The second is, when the system is in manual mode, to survey that no harm

| | internal components | | | | | | ext. comp. | initial state check | |
|---|---|---|---|---|---|---|---|---|---|
| | Y18 | Y16 | Y14 | SG3 | SG6 | zones | | alarm type | alarm delay |
| Initial State | home | locked | close | on | on | - | | 1 | 0 |
| Action 1 | home | unlocked | close | on | on | b(2,4,6,8,10) | | | |
| Action 2 | work pos | unlocked | close | on | on | - | | | |
| Action 3 | work pos | locked | close | on | on | u(6,8) | | | |

Fig. 2. An EOP for the basic operation 43; move Fixture 152 to work position.

can be done to humans or to the cell. This information is described by a matrix, see Figure 3. Every row in the matrix is referred to as a term. Every term describes the different states that must be fulfilled before any change can be done in the cell.

| | Actuator/operation id | | 152Y18 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Action/operation comment | | Go to home position | | | | | | | | |
| | Resource | | FIX152 | | | | | | | | |
| | Internal component states | | | | | Collision zones | | Operations | | ext.comp.st. | |
| | position | fixation | clamp | part sensors | | | | | | resource | |
| | Y18 | Y16 | Y14 | SG3 | SG6 | before | after | not started | not ongoing | compone | Product |
| Term 1 | * | unlocked | * | off | off | - | - | - | - | - | - |
| Term 2 | * | unlocked | closed | on | on | - | - | 63,98 | - | - | S80 |
| Term 3 | * | unlocked | closed | on | on | - | - | 64,99 | - | - | V70 |
| Term 4 | * | unlocked | open | * | * | - | - | - | - | - | - |

Fig. 3. An IL for the actuator 152Y18; go to home position.

### D. Mechanical Components and Functions, MCF

MCF is a list of mechanical components in a cell, used for program generation. The list point out which Function Blocks (FB) that should be instantiated in the PLC-program [16].

### E. Coordinated Operations, COP

The set of Coordinated Operations is a result of the verification process. It has the same structure as the ROP. The operations are executed in a sequence which will eliminate the risk of collisions between machines and possible deadlock states of the cell. To allow different execution orders between the resources in a cell that can handle different product types, the COP is divided into one COP for every resource and product type. The COP is defined so that every operation in the execution sequence has a set of preconditions. Before an operation can be executed, all its preconditions must be fulfilled [16].

## III. XML AND SQL

XML is an abbreviation for Extensible Markup Language [13]. XML is spread worldwide and is used as a standard data storage format in many web applications and local applications. The language was born in 1998 when The World Wide Web Consortium (W3C) realized that they were in need of a data storage markup language independent of any platform. Markup language means that data is marked up by tags, see Figure 4. Contrary to Hyper Text Markup Language (HTML), XML is not limited to a predefined set of tags. It is a very flexible language and one of its major advantages is that it is simple to read both for humans and computer programs.


Fig. 4. An example of XML data describing the Physical Resources of a production cell. The data is inserted between two tags with the same name and each element can have an attribute attached to its first tag.

### A. XML schemas

XML schemas are used to define rules for XML files. If there were no definition of the structure of the XML file, it would be very hard to find the relevant data. The XML schema is almost built up the same way as the XML file, but has a different extension (.XSD) that defines it as a schema. The schema sets the rules of the XML file by defining the order of elements and attributes and restricting element values. To make it possible for all users to validate their XML files from any location, the schemas should be stored on a public web server [13].

### B. SQL Server 2005

SQL Server 2005 is a Database Management System (DBMS) for relational databases [14]. In SQL Server Management Studio, the design of the database is done by creating a set of tables, relations, indexes, restrictions etc. Database triggers can be set to execute on a specific user action, such as before or after a row is deleted from a table. In the Management Studio, different queries can be executed on a database which is very helpful in the design, implementation and testing of a new database.

## IV. Database

The database solution was designed in Microsoft SQL Server 2005. The communication with the database can be made with a Java developed user interface and the different restrictions that apply to both the database and the user interface are derived from a set of XML schemas. Schemas for different the different parts, i.e. PR, VR, ROP, IL, and EOP, of the framework is described in [15], [16]. The parsing and generation of the XML data structures is the major part of the predefined code in the database.

### A. Structure

The database contains tables, stored procedures, functions, triggers, restrictions, indexes, keys, user identities, error logs etc. All data to be stored in the database is inserted as elements into the different tables by invoking the stored procedures and functions from a SQL query. This can be done either locally in SQL Server or from a client using an SQL Server driver, for example, Visual Studio. Since it is not reasonable to expect all clients to use Visual Studio and in order to avoid dependability of any other software, a graphical user interface designed in Java using Swing and the Microsoft Java Database Connectivity (JDBC) driver.

### B. Database tables

The tables of the database consist of a predefined set of columns, each using a specific type (e.g. integer, varchar, xml), and an unlimited number of rows. The tables can be divided into seven categories:

- Projects
- Physical resources
- Virtual resources
- ROP
- EOP
- IL
- Other

*1) The Projects category:* The Projects table is the only table in the database without a foreign key, that is, the only table not referencing another table. Instead, the Projects table is referenced directly or indirectly by all other tables. If a row in the Projects table is deleted, so are all rows referencing this row in other tables, with one exception - the PLC_FBs table. In this table, the foreign key value will instead be set to null. The PLC_FBs table is not currently in use, but is likely to hold information about the function blocks of different actuators in the future. The Projects table contains columns for the project identity value, the project name, the project description and the number of missing references in the project.

| **PR type** | **Table(s)** |
|---|---|
| Factory* | Physical_resources |
| Area* | PR_areas |
| Cell* | PR_cells |
| Machine | PR_machines, _def_Machines, _type_Machine, _type_Control_system |
| Equipment | PR_equipment, _def_Equipment, _type_Equipment |
| - with states | PR_states |
| Element | PR_elements, _def_Elements |
| | |
| **VR type** | **Table(s)** |
| Cell* | Virtual_resources |
| Zone | _def_Zones |
| Variable | _def_Variables |
| - with values | _def_Values |

Fig. 6. Resource types and their corresponding tables.

*2) Physical and Virtual Resources category:* The next two categories are Physical resources (PR) and Virtual resources (VR). They hold information about all the physical resources, variables and zones used in a production cell. At present, only one cell per project is possible, but the structure of the information allows us to define the physical path to this cell, i.e. the area and the factory surrounding it. This facilitates future changes of the information structure to allow multiple cells in each area and multiple areas in each factory. The PR

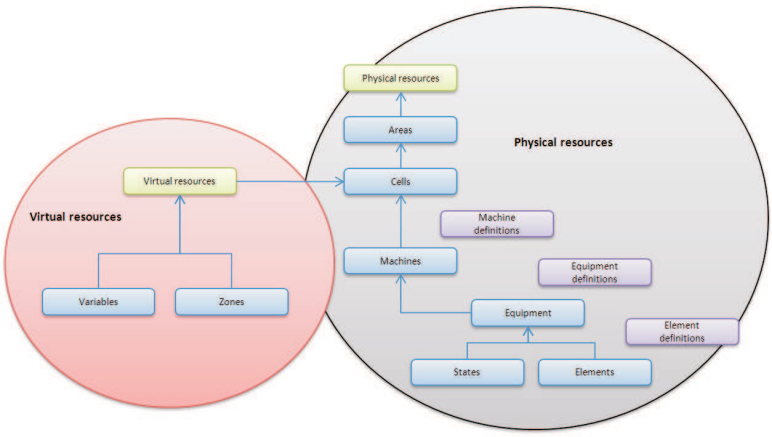


Fig. 5. The relations between the tables containing the Physical and the Virtual Resources.

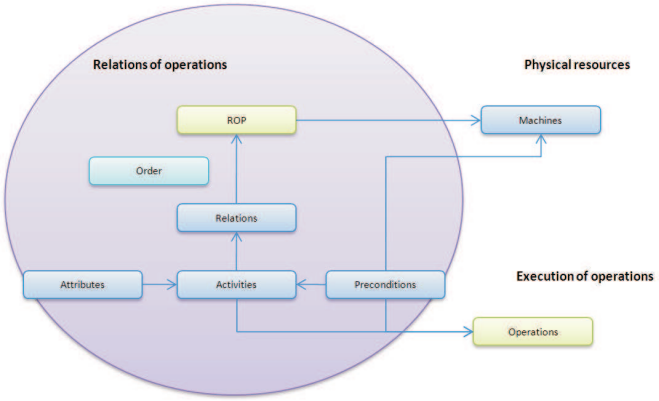


Fig. 7. The relations between the tables containing the Relations of Operations.

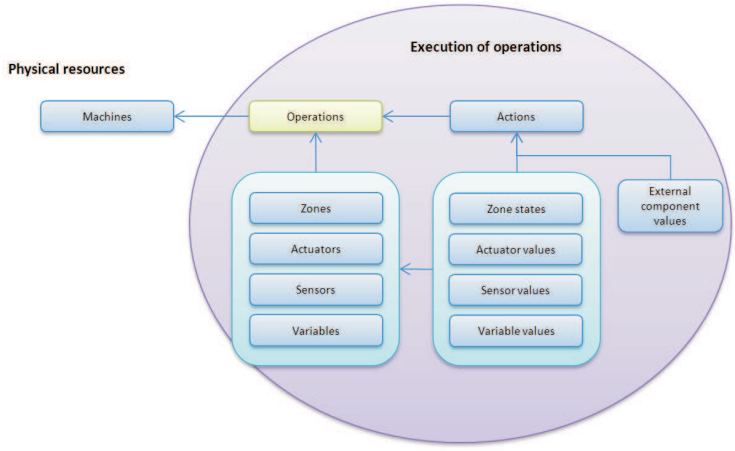


Fig. 8. The relations between the tables containing the execution of the operations.

and VR in a cell can be defined as any of the following types with corresponding database table(s), see Figure 6.

The Physical_resources table and the Virtual_resources table each contain a project reference. The relations between the tables in the database are shown in Figure 5.

*3) The ROP category:* The ROP information is stored in 6 tables; ROPs, ROP_relations, ROP_order, ROP_activities, ROP_predecessors, ROP_attributes. The ROPs table is the main

table. It contains the ROP name, the ROP type, a comment, the machine name and a Project reference. The type indicates if the information has been validated with Supremica or not and can be set to either ROP or COP. The relations between the tables are shown in Figure 7. The ROP tables are referencing two tables outside its category; the Machines table and the Operations table. The referencing tables: Activities and Precondition are not connected to these tables immediately, instead they store the machine name and the operation name locally and tries to link the tables by matching the names. Since the names must be unique within each project, a unique match can be found and used as a reference. The Order table stores information about the sequential order of the activities and sub relations within each relation.

*4) The EOP category:* The EOP information, or operation information, is stored in 13 tables: Operations, OP_actions, OP_actuators, OP_sensors, OP_variables, OP_zones, OP_ext_components, OP_actuator_values, OP_sensor_values, OP_variable_values, OP_zone_states, OP_ext_comp_values, _type_EOP. The Operations table is the main table and it contains the operation name, the operation type, the duration of the operation, a comment, the machine name, the alarm type of the operation, and a project reference. The operation type can be set to any of the values (i.e. restrictions) in the _type_EOP table. The types are currently restricted to ”alternative” or ”basic”. The relations between the tables are shown in Figure 8.

*5) The IL category:* The IL information is stored in 17 tables: ILs, IL_terms, IL_actuators, IL_sensors, IL_variables, IL_ext_components, IL_zones, IL_operations, IL_actuator_values, IL_sensor_values, IL_variable_values, IL_ext_comp_values, IL_before_zones, IL_after_zones, IL_not_ong_operations, IL_not_st_operations, IL_products_in_terms. The ILs table is the main table and it contains the IL name, either an operation name or an actuator name, a comment, and a project reference. The relations of the tables are shown in Figure 9.

### C. Inserting data into database

To process the data that should be stored in the database, stored procedures are used. The stored procedures are executable scripts that can handle data in different ways. For instance, a stored procedure can be used to handle the data stored in an XML file. The procedures that are used for parsing XML data into the database are:

- InsertPRXML
- InsertVRXML
- InsertROPXML
- InsertEOPXML
- InsertILXML
- InsertProjectXML

The procedure *InsertPRXML* is described in more detail in Section IV-C.1. The rest of the insert procedures work in a similar way and is not described in detail in the present paper.

*1) The InsertPRXML procedure:* The physical recourses XML file describes all mechanical components in the cell. To extract this information from the XML file, it must be scanned through level by level. Every level in the XML hierarchy has specific information that will be stored in specific database tables with certain relations.

The XML data holds information about the cell and includes information about the factory and the area that the cell is located in as well as the machines the cell consist of and their equipment and configuration. Each machine can be built up with a number of equipments consisting of sensors and actuators. The actuator can also be built up with other sub actuators and sub sensors. Theoretically, an infinite number of levels can be set up with equipments and sub equipments in this way. The levels are explained in Figure 10. Too solve this infinity problem when extracting the information from an XML file, a recursive method was implemented.

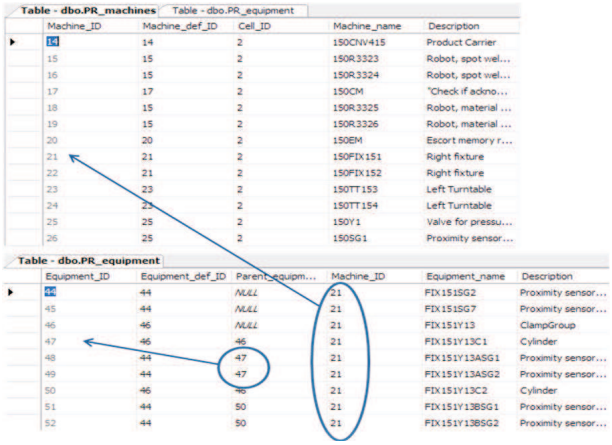


Fig. 10. The relations between equipment, sub equipment and machines in the database.

The basic idea of the stored procedure is to split up the XML data into smaller XML variables and process each of these variables separately. A variable can, for instance, represent a machine and every machine is processed one at a time. When all information has been extracted at the machine level, all equipment is parsed into XML variables. The variables are sent to the stored procedure EquipmentParser. To preserve the relations between the equipment and the machines a machineID and the parent equipmentID are also sent to the EquipmentParser procedure. The procedure extracts all information from the equipment, and if the equipment consists of sub equipment, it parses all sub equipments into XML variables and calls itself again. This procedure is repeated until all sub equipment has been parsed. Figure 11 shows how this is done.

### D. Extracting data

There are four ways to build XML from a relational database in Transact-SQL; the RAW, the AUTO, the EXPLICIT and the PATH mode. The RAW mode is probably the fastest and most intuitive way of building XML. The

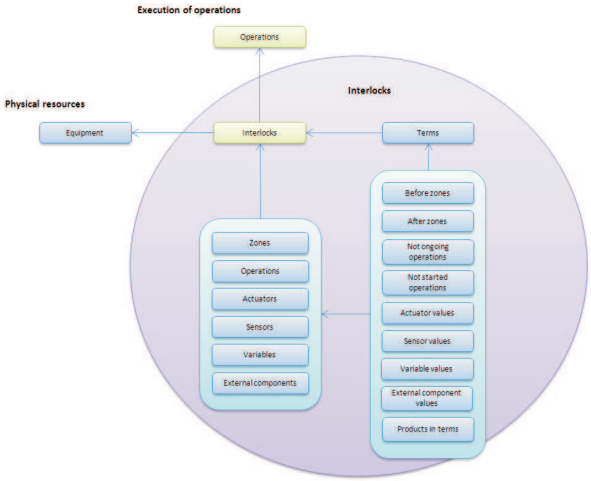


Fig. 9. The relations between the tables containing the interlocks information.

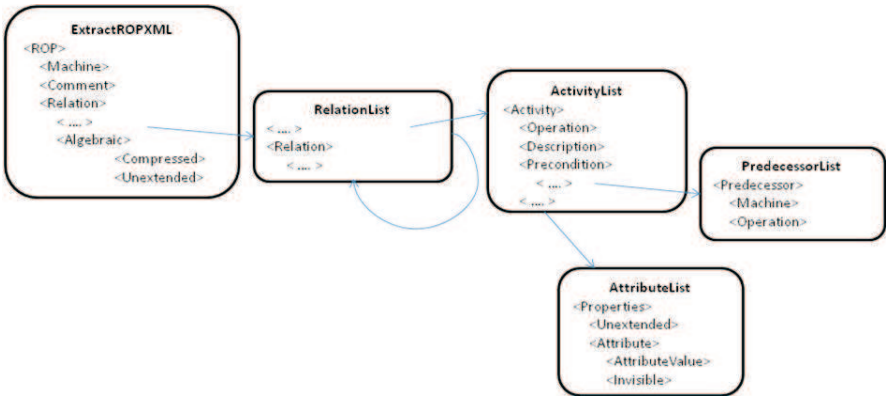


Fig. 12. The ExtractROPXML function and its four sub functions; RelationList, ActivityList, PredecessorList and AttributeList. The RelationList function is the only recursive function.

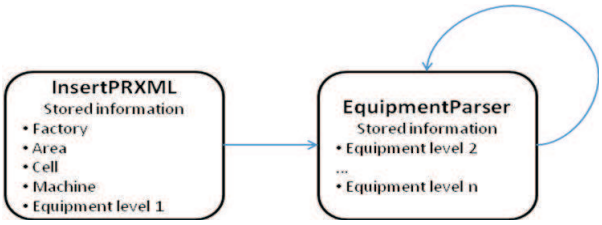


Fig. 11. The stored procedures that manage the physical resources data and parses the XML into different tables in the database.

EXPLICIT mode is the most flexible mode but is very complex to use. The PATH mode is a later development from Microsoft and it was first implemented in SQL Server 2005. This method has the EXPLICIT modes flexibility, like the possibilities of mixing elements and attributes, but requires much less code to write. To build an XML file, the database functions are used since they can return large XML variables. The functions are programmed in the PATH mode and some of the functions for extracting data can appear rather complex. The reason for choosing the PATH mode is that it allows much flexibility with less code compared to any of the other types.

The function *ExtractROPXML* is described in more detail in Section IV-D.1. The rest of the insert procedures work in a similar way and is not described in detail in the present paper.

*1) The ExtractROPXML function:* The ExtractROPXML function is the most complex function in the database. The purpose of the function is to extract a ROP from the database as XML data. It consists of four sub functions, see Figure 12. When creating the ROP XML it is very important to retain the structure and especially, the order of the relations and the activities. The order is set up in the RelationList function. The ROP can have an unlimited number of relation levels. To handle this, the RelationList function is made recursive and it will generate a new relation branch each time it is called. To generate an instance of the Activity element and its child elements, the ActivityList function is used. ActivityList calls two other sub functions that generate the instances Predecessors and Properties and their sub elements. The structure of the ExtractROPXML is explained in Figure 10.

### E. The Client User Interface

From the user interface, data can be exported and imported to/from the database over a network connection using the TCP/IP protocol. In detail, a predefined SQL query with variable parameters is sent from the interface application, invoking a specific stored procedure or function. When, for example, exporting a ROP from the database the following events will occur. A query will be sent from the interface application requesting the execution of the function ExtractROPXML(@projectID, @ROPNameID). The parameters of the function are always set in the interface before the query can be sent. The function looks up the single row matching the parameter values in the ”ROPs” table in the database. Next, an XML file is generated by the ExtractROPXML function in a predefined structure, fetching information from all tables referencing the ROPs table. The XML file is returned by the database function and received by the interface application as a String object. Figure 13 shows the main window of the interface before a connection is made with the database

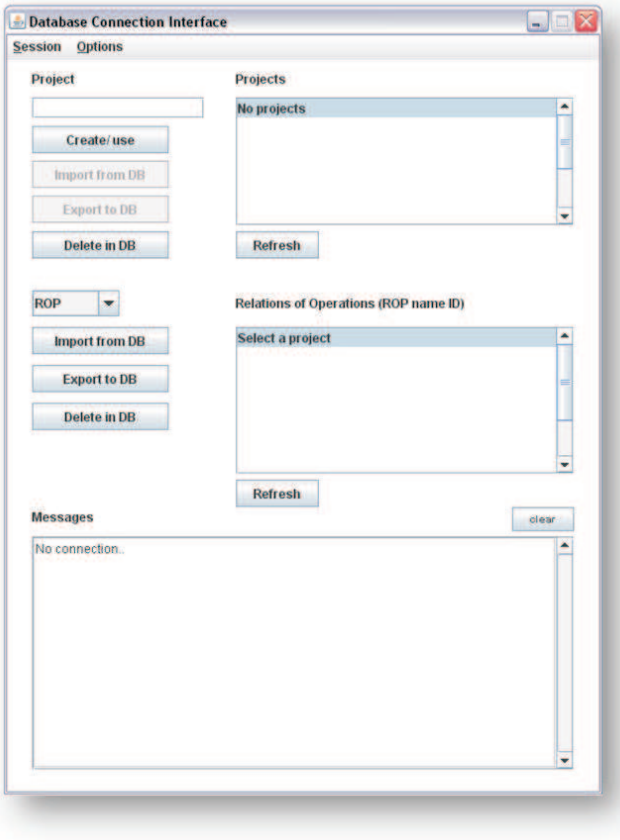


Fig. 13. The client user interface, used to communicate with the database.

## V. Conclusion

The present paper has presented a control database that can be used in a general and flexible way by simulation tools, formal verification tools, as well as for automatic control program generation. A vendor independent method for exchanging information between development tools as well as storing in the database using XML-schemas has been applied.

The paper has described how the proposed method can enable reuse of information, decrease manufacturing system preparation time as well as decrease ramp-up time of new systems or the introduction of a new part in an existing system.

It has been shown how control information is structured and how information can be exported and imported from the database. It has also been described how information can be exchanged between development software tools such as Process Simulate and Supremica.

An interface between the user and the database has been presented in order to have easy access to the database.

It has been shown how this is exported and imported into the database using pre-defined sets of import and export structures in order to simplify the use of the database information. The use of pre-defined structures will ease the workload of the database in cases where many users are connected at the same time. It also simplifies the users handling of the information.

## References


[1] Siemens, Webpage, http://www.plm.automation.siemens.com/en_us/ products/tecnomatix/, Date of access: 2009-04-02.

[2] Dassault Systemes, Webpage, http://www.3ds.com/products/delmia/ welcome/, Date of access: 2009-04-02.

[3] J. Richardsson, ”Automated Verification and Generation of Flexible Automation Control, Control and Automation Laboratory, Chalmers University of Technology, 2007, PhD thesis, Göteborg, Sweden, Technical report nr 2643, ISBN/ISSN: 91-72919624.

[4] Chalmers University of Technology, Automation reserarch group, Webpage, http://www.supremica.org, Date of access: 2009-04-02.

[5] K. Åkesson, Methods and tools in supervisory control theory, Control and Automation Laboratory, Chalmers University of Technology, 2002, PhD thesis, Göteborg, Sweden, Technical report nr 431.

[6] K. Åkesson and M. Fabian and H. Flordal and R. Malik, Supremica - An Integrated Environment for Verification, Synthesis and Simulation of Discrete Event Systems, Pros. of 8th Workshop on Discrete Event Systems (WODES’06), 2006, Ann Arbor, MI, USA, pages 384-385.

[7] K. Åkesson and M. Fabian and H. Flordal and A. Vahidi, Supremica - A Tool for Verification and Synthesis of Discrete Event Supervisors, 11th Mediterranean Conference on Control and Automation, 2003, Rhodos, Greece, June.

[8] W.M. Wonham and P. Ramadge, 1987, On the Supremal Controllable Sublanguage of a Given Language, SIAM Journal on Control and Optimization, volume = 25, number = 3, pages = 637-659.

[9] C.G. Cassandras and S. Lafortune, Introduction to Discrete Event Systems, Kluwer Academic Publishers, 1999.

[10] K. Åkesson and H. Flordal and M. Fabian, Exploiting Modularity for Synthesis and Verification of Supervisors, Proc. of the IFAC World Congress on Automatic Control, 2002, Barcelona, Spain, July.

[11] P. Falkman and F. Westman and C. Modig, Verification of Operation Sequences in Process Simulate by Connecting a Formal Verification Tool, Proc. in the 7th IEEE International Conference on Control and Automation (ICCA09), 2009, 9-11 December, Christchurch, New Zealand.

[12] M. Johansson and M. Kjellgren, Automatic Generation of Control Functions for Manufacturing Automation Systems, Control and Automation Laboratory, Chalmers University of Technology, 2007, MSc thesis, Göteborg, Sweden, Technical report nr 2643, ISSN 99-2747920-4; nr EX074/2007.

[13] G. Powell, Beginning XML Databases, Indiana: Wiley Publishing, 2007, Indianapolis.

[14] A. Watt, Microsoft SQL Server 2005 Programming For Dummies, Indiana: Wiley Publishing, 2007, isbn 9780471774228.

[15] O. Ljungkrantz and K. kesson and J. Richardsson and K. Andersson, Implementing a Control System Framework for Automatic Generation of Manufacturing Cell Controllers, Proc. of the 2007 IEEE International Conference on Robotics and Automation, 2007, pages 674-679, 10-14 April.

[16] J. Richardsson, Automated Verification and Generation of Flexible Automation Control, Control and Automation Laboratory, Chalmers University of Technology, 2007, PhD thesis, Göteborg, Sweden, Technical report nr 2643, ISBN/ISSN: 91-72919624.