INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 05 MELBOURNE, AUGUST 15 – 18, 2005

AN OVERVIEW OF THE VRS VIRTUAL PLATFORM

R.I. Whitfield, A.H.B. Duffy, Z. Wu, J. Meehan

Abstract

This paper provides an overview of the development of the virtual platform within the European Commission funded VRShips-ROPAX (VRS) project. This project is a major collaboration of approximately 40 industrial, regulatory, consultancy and academic partners with the objective of producing two novel platforms. A physical platform will be designed and produced representing a scale model of a novel ROPAX vessel with the following criteria: 2000 passengers; 400 cabins; 2000 nautical mile range, and a service speed of 38 knots. The aim of the virtual platform is to demonstrate that vessels may be designed to meet these criteria, which was not previously possible using individual tools and conventional design approaches. To achieve this objective requires the integration of design and simulation tools representing concept, embodiment, detail, production, and operation life-phases into the virtual platform, to enable distributed design activity to be undertaken. The main objectives for the development of the virtual platform are described, followed by the discussion of the techniques chosen to address the objectives, and finally a description of a use-case for the platform. Whilst the focus of the VRS virtual platform was to facilitate the design of ROPAX vessels, the components within the platform are entirely generic and may be applied to the distributed design of any type of vessel, or other complex made-to-order products.

Keywords: Distributed design, integration, data and activity co-ordination, and neutral databases.

1 Introduction

Design and simulation of complex products is increasingly becoming more reliant upon mathematical models in order to consider an increasing number of physical phenomena to provide a more comprehensive analysis of the design problem. Information Technology (IT) has commonly been used to provide solutions to complex design problems such as performing stress analyses using Finite Element Analysis, or calculating aerodynamic characteristics using Computational Fluid Dynamics. However these analysis techniques generally provide point solutions to particular design issues and only go a little way towards leveraging the full potential of IT advances. The technology within this context whilst having the effect of reducing lead times through providing more reliable and efficient analyses, has not been fully explored to the extent that it may: be used to promote the collaboration both within and between organisations through the integration of these point solutions; ensure that the design problem being investigated is consistent across all of the analyses being considered; and ensure that changes are correctly propagated and that design activity is undertaken for the correct reasons.

From a social and psychological viewpoint, a solution to the multi-partner collaborative design problem is realised by allowing the designers to continue using the design tools, techniques, rules and knowledge that they are familiar with, and to provide an over-arching

platform that integrates the tools' and the designers' capabilities, and monitors, manages and co-ordinates their actions. The management within this platform is a pre-requisite for establishing an efficient and effective integrated environment between designers and systems. Whilst this management does not necessarily dictate how the designers perform their activities, it does provide a mechanism for designers to co-ordinate their actions, providing information and guidance to aid the decision-making process, and take action on the basis and in response to others.

In order not to deviate significantly from current design practice there also is a requirement that the users of the platform are not restricted in any way and should be able to use existing legacy and new design systems with a minimum or no modification to either the design and simulation software or the platform itself. Ideally the designers should regard the virtual platform as an extension of their normal working environment and as a means of problem solving that was previously either extremely time consuming, or not possible at all. Above all, the platform should provide a natural designing experience for the designer, and a seamless environment for the integrated systems.

The VRS virtual platform aims to provide support to the users of the platform in order to provide them with a means of combining the tools, knowledge and expertise of the partners involved for the first time within an environment that enables the design and evaluation of novel vessels through the combination of these through-life point solutions. Section 2 describes the structure of the VRS project with respect to the work packages, the associated components that they deliver, and the objectives of these components. Section 3 describes the development of each of these components with respect to the technologies adopted and the relationships between these components. Section 4 presents a simplified example of how the platform may be used, and Section 5 concludes the paper.

2 VRS project structure

The VRS project consists of ten work packages that are responsible for delivering various aspects of the virtual and physical platforms as well as for co-ordinating the project. In order to understand the challenges associated with the development of the VRS virtual platform it is necessary to provide an overview of the structure of the project and the objectives of the work packages.

2.1 Project overview

The approach adopted within the design of the virtual platform was to carry out an iterative process of development, test, implementation and evaluation within and across the work packages leading towards the production of a complete virtual platform. "Wrapping" was required for existing, and new design and simulation tools (WP6) in order to interface with the integration framework (WP1) and cater for platform (hardware and software) independence. A common model containing consistent ship product data was developed to capture the main knowledge, information and geometry of the virtual ship (WP2). The interaction between the tools, their local models, and the common model was managed through version control, consistency/constraint based management and conflict resolution techniques (WP4). The application of life-phases process knowledge/approaches upon the virtual model was investigated through the development of a process control tool (WP5), which provides a means to control and evaluate the "behaviour" of the integrated platform as well as providing support for the life-phases of a ship. The interaction with the platform was realised through a virtual environment (WP3), which concentrates upon techniques and

approaches to develop real time, virtual interaction. The performance data generated from the simulation engine tools was analysed to provide a means to determine and optimise the overall performance and uncertainty of the virtual ship (WP8). Since the performance-modelling tool is not fundamental to the operation of the platform, further discussion of the development of the tool is omitted. All development within the project was generic in nature (other than the ship product data within the common model, and the knowledge within the simulation engine) so that the results can be applied to any industrial domain or discipline. The virtual platform therefore enables extensive simulations, real time virtual interactions, performance analysis and life-phase support to be undertaken irrespective of the application domain or ship type.

2.2 Virtual platform objectives

The functionality and objectives of the main components within the VRS virtual platform are described as follows:

- **Integration**. The aim of the integration work-package is to deliver a strategy and architecture to guide the integration activities of the common model, virtual interaction, inference engine, process modelling and control, simulation engine and performance analysis and reliability components. The main objective of the integration framework is therefore to deliver a flexible protocol and communication mechanism that enables these disparate systems to integrate and co-ordinate their functionality.
- **Common model**. The common model is a database that provides a consistent representation of the data defining the ship systems (ship product model) and external environment (sea state, routes, port facilities), and holds the basic (and common) geometry and information required by each of the integrated simulation engine tools irrespective of the tools' native data formats. The aim when defining the data specifications and schemas for the common model is to consider the functional requirements of the life-phase process models as well as the requirements of the integrated tools in order to ensure that the common model supports the design requirements of the user, as well as facilitating data transfer within simulation and real-time rendering programs developed within the virtual interaction component for example. Cover for the whole lifecycle of the vessel should be provided, from initial design to disposal; hence the data contained within it should be applicable across life-phases.
- **Virtual interaction**. The common model allows distributed manipulation of the ship product model as well as enabling the virtual environment by allowing the development of the product model using the tools within the simulation engine. The virtual environment is the interface that the users utilise in order to interact with the virtual platform. The virtual environment should provide functionality to enable: multiple users; configuration and use of design and simulation tools; access to the common model; visualisation of common model contents; querying of data consistency status; enactment of processes, and use of the performance modelling tool.
- **Inference engine**. The virtual environment enables users to communicate and share product data and information through the ability to remotely access, query and modify the common model. The design or simulation tools being integrated commonly have their own local model, represented either as local files or databases. Changing the data within one tool's local model may have multiple implications or effects on other tools' models. The main objective of the inference engine is to maintain the consistency between these various models through the management of change propagation and

conflict resolution between multiple users. The inference engine must manage: dynamic modification to common and local model data; the variation in information requirements, a mapping of the dependencies and relationships between data within the common model; and consistency management and conflict resolution.

- **Process modelling and control**. The process control tool is a planning and enactment environment for the co-ordination of activities within life-phase process models. This process control tool is used to define an initial sequence of activities, to determine an optimum process schedule, to manage the enactment of the tools within the virtual platform and to manage the processes under real-time conditions. Since the main objective of the process control tool is to demonstrate how distributed activities within a virtual platform can be managed and co-ordinated, there is also a requirement that the process control tool manages the resources that are capable of performing the activities, as well as co-ordinating when and why they should be undertaken.
- **Simulation engine**. The simulation engine represents the integrated design and simulation tools within the virtual platform. These tools enable the design of the hull, general arrangement, propulsion plant, subsystems, and simulation of the operating environment, operations, supply chain and production. The simulation engine is capable of allowing a through life assessment, ranging from concept development to performance trials and operational scenarios. The tools are "wrapped" in order to enable communication with the rest of the virtual platform.

3 Development of the VRS virtual platform

This section describes the components that combine to make the virtual platform. A prototype virtual platform was developed in order to determine the most appropriate technologies, to establish the mode of operation of the platform with respect to usability, and determine any shortcomings of the technologies used. Each subsection describes the results from the prototype development where applicable along with the lessons learned from the prototype in order to develop the next stage of the virtual platform. The relationships between the components of the virtual platform can be seen within Figure 1.

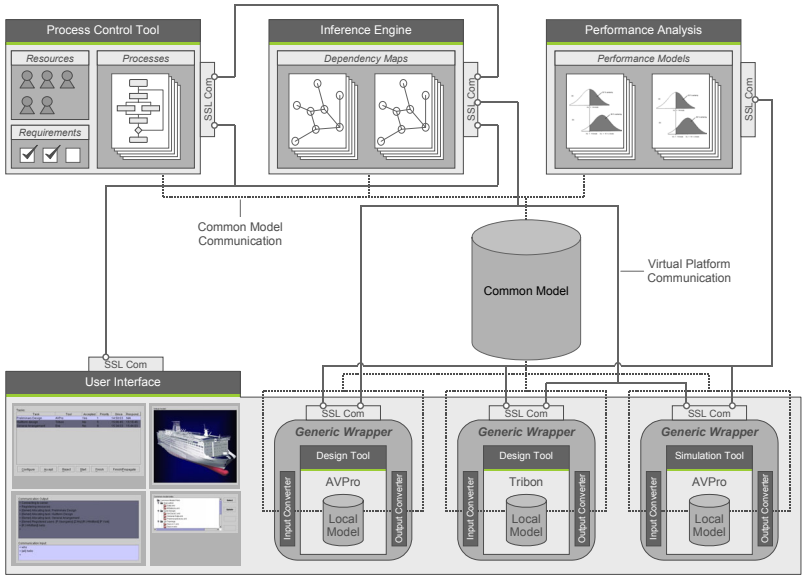

Figure 1. Relationships between VRS virtual platform components.

3.1 Integration framework

Rather than focus on the development of the communication mechanism and protocols, the prototype integration framework focussed towards the use of an existing adapter to manage the operation of the design and simulation tools within the simulation engine. The design and simulation tools were distributed across a network with each tool individually managed by an adapter. The tools were mapped directly to activities within the process control tool, hence when a process was due to be undertaken, the process control tool would communicate with the appropriate adapter for the mapped activity to start the tool. The mapping of an activity to a tool limited the tool's application to single activities whereas a number of the tools were capable of performing many design activities. The adapter was only capable of managing the start and completion of the tool, and provided no functionality to access the common model for example, and therefore restricted the overall integrated functionality of the platform. A bespoke wrapper was later developed within WP6, specifically designed to address the integrated management of the design and simulation tools. The prototype integration framework did however enable three key requirements to be identified with respect to the communication of data between the components:

- **Message security.** Secure Socket Layer (SSL) communication should be implemented to utilise public/private key encryption, source authentication and data integrity for the transfer of data between virtual platform components.
- **XML message format.** XML should be used as the underlying language for communication of data between virtual platform components due to its inherent extensibility and support throughout the IT community.
- **Message validation.** XML schemas should be used to automatically check that the messages received by any of the virtual platform components conform to a defined standard.

Existing and well-established integration technologies such as the Common Object Request Brokerage Architecture (CORBA) were considered for the development of the integration framework. Within this context however, CORBA would have been implemented to facilitate the transfer of objects between the tools to be integrated, rather than enabling the insertion of the tools within the integrated platform.

The aim of the integration framework was to create an open architecture that would enable new simulation engine tools to be easily integrated into the platform with a minimum of development to the tool provider (and modification to the platform) when they become available. In order to achieve this, consideration was given for the types of data to be communicated between the virtual platform components, the frequency of communication, and the type of communication (synchronous/asynchronous) when designing the protocol and communication mechanism. A functional protocol was defined based upon the Remote Procedure Call specification of XML (XML-RPC), unifying all of the protocols within the virtual platform into a single "generic" communication protocol - Figure 2. The protocol has elements to indicate the "sender" and "receiver" components in order that the receiver can check that it is a valid functional request from a valid sending component. The "functionName" element within the protocol is used to define the function (that is mapped to some functionality that the receiving component will perform) using the information contained within the "params" element.

An example of the use of the protocol would be a request from the generic wrapper to the inference engine to perform a status check on a piece of data before modification. A mapping would exist between the requested function and functionality encoded within the inference

engine. Any number of parameters may be included within the protocol to enable the enactment of the function by the receiver. Both the sending and receiving components must agree in advance to the structure of the parameters, however the protocol is entirely neutral to the programming languages and the structure of the objects that are used by each component to represent the data.

```
<?xml version="1.0" encoding="UTF-8"?>
<VRSMethod>
    <sender>Sender</sender>
    <receiver>Receiver</receiver>
    <functionName>FunctionName</functionName>
    <params>
        <param>
            <title>Title1</title>
            <value>
                <string>String</string>
            </value>
        </param>
        <param>
            <title>Title2</title>
            <value>
                <struct>
                    <member>
                        <name>Name21</name>
                        <value>
                            <int>42</int>
                        </value>
                    </member>
                    <member>
                        <name>Name22</name>
                        <value>
                            <double>3.14159</double>
                        </value>
                    </member>
                </struct>
            </value>
        </param>
    </params>
</VRSMethod>
```

Figure 2. XML-RPC functional protocol.

Software was developed to enable the secure sending and receiving of XML-based communication between virtual platform components. An Application Programme Interface (API) was designed within the software to enable it to be integrated within each of the necessary components within Figure 1. Due to the dynamic nature of the communication within the virtual platform, it is difficult to predict when each component will be required to process communication, or whether the component will be sending and receiving multiple simultaneous communication. The communication software was therefore developed using a multi-threaded architecture to queue and sequentially process out-going communication, whilst allowing simultaneous prioritised processing of incoming communication.

3.2 Common model

The common model is a repository for the data that is used by the design and simulation tools within WP6 as well as for storage of virtual platform co-ordination and management data from the inference engine (dependency maps), process control tool (process models) and the performance analysis and reliability tool (performance models). Three issues required consideration within the design of the common model: storage (how to store the data), structure (how to format the data), and coverage (what data to store).

These issues are however related since the selection of the storage mechanism depends upon the structure of the data to be stored (binary, XML, object-oriented). In addition, the structure of the data is influenced by the requirements of the tools to be integrated as well as the requirements of the users of the platform, and therefore influences and is influenced by the coverage. The current practices and standards used in formatting and structuring engineering data were investigated, including the Initial Graphical Exchange Specification (IGES) and the Standard for the Exchange of Product Model Data (STEP – ISO 10303), and the amount of

data to be modelled (complete product information models for ships have been estimated by Catley to be of the order of between two and ten Gigabytes of data [1]). Considerable effort has been thrust towards the development of ISO 10303 STEP Application Protocols (APs) for Ship Arrangements (AP215) [2], Ship Moulded Forms (AP216) [3], Ship Structures (AP218) [4], Ship Mechanical Systems (AP226 which has since been withdrawn), and Piping (AP227) [5], – Grau [6]. A number of occasions have arisen however when attempts have been made to adhere to these APs with the result that "flavours" of the standard have been required in order to utilise the structured, well-defined and standardised data within the legacy applications [7]. The focus when defining the coverage of the common model was therefore directed towards "the minimum amount of information that would enable the integrated tools to share an accurate representation of the product". ISO 10303 Part 203 is used to define the 3D design of mechanical parts and was used within the common model as the basis upon which to define the geometry of any aspect of the ship product model. The selection of Part 203 also influenced the structure of the data to be contained within the common model.

Conversion of data to and from the common to native formats was also a significant issue that needed to be addressed, which would influence the structure of the data within the common model as well as how easily and successfully design and simulation tools could be integrated within the virtual platform. Having chosen to base the geometrical data within the common model on Part 203, the focus was then to complete the definition of the structure of the data, through the selection of an appropriate language that would facilitate conversion. EXPRESS (ISO 10303 Part 11 and 12) was developed as a language for the definition of STEP data. Major shortcomings of the EXPRESS reference language from a conversion viewpoint were that it was difficult to decompose EXPRESS-based models into more manageable chunks, and it was difficult for a human to interpret what the data within an EXPRESS file represented. Whilst human readability is not an issue during the process of converting data, it is certainly useful if the conversion algorithm developer can understand the concepts that are being converted whilst producing the algorithms. Attempts to undertake the conversion between formats have in the past faced difficulties due to the complexity and formatting of the data, and have in certain cases required a degree of human interaction [8]. Research had been successfully conducted to produce a binding between STEP and XML (ISO 10303 Part 28) and has been applied within the shipbuilding industry [8] with the aim of facilitating the conversion process between the neutral STEP format and the native tool format. It was therefore concluded that an XML mapping of STEP data provided the most appropriate language upon which to base the storage of all data within the VRS platform due to its extensible nature, support for conversion, and the increasing support within the IT industry. Choosing to manage the data within the common model using XML, facilitated the selection of a storage mechanism.

The XML:DB initiative provides standardisation for the development of specifications for the querying, manipulation and management of data stored within XML databases. This initiative enabled the database to be de-coupled from the rest of the virtual platform, such that the database may be exchanged with alternative XML databases without impacting any of the software that communicates with the database. Documents are then managed within the database in hierarchical collections, similar to the directories within a file system.

3.3 Virtual interaction

The virtual environment represents the "window" to the VRS virtual platform. It is intended to provide functionality to enable communication between, and use by multiple users, configuration and use of design and simulation tools, access to the common model, visualisation of common model contents, querying of data consistency status, enactment of

processes and use of the performance modelling tool. The focus during the development of the prototype virtual environment was towards the production of an open architecture that would enable the configuration and remote use of design and simulation tools, and the visualisation of common model contents. A number of the design and simulation tools to be integrated into the platform had the ability to visually display the data in their local models whilst the user was operating the tool. Rather than developing and producing a new visualisation of the common model data within the virtual environment, the prototype environment attempted to enable the user interfaces of the remotely distributed tools to be exported to the virtual environment via the Internet using Virtual Network Computing (VNC) software. VNC enables the viewing and interaction of the desktop of a remote machine within a window of the local machine. The use of this approach meant that design and simulation tools could be "hosted" on remote machines, providing the facility for users to log onto the machine through the virtual environment and interact with the tool whenever required. New tools could be registered and integrated within the platform, and the functionality and visualisation aspects of the tools, utilised irrespective of where either the tool or user were geographically located, and without the need to generate any code to enable this integration - Figure 3. The display of one of these remote machines could in principal be exported to a number of other machines, facilitating the collaborative work of multiple users on the same design problem using the same design or simulation tools.

The network bandwidth required to support this seamless visualisation of remote tools was however demonstrated to be in excess of the bandwidth that was available. Various optimisation algorithms were used within the VNC software in order to reduce the amount of data transferred by sending regions of the display that were changing and by limiting the number of colours that are repainted on the client display. Where this may have been adequate for design situations, where the display does not generally rapidly change, within a simulation situation, it was apparent that the network bandwidth could not support the refresh rates that were required for a smooth transition between frames.

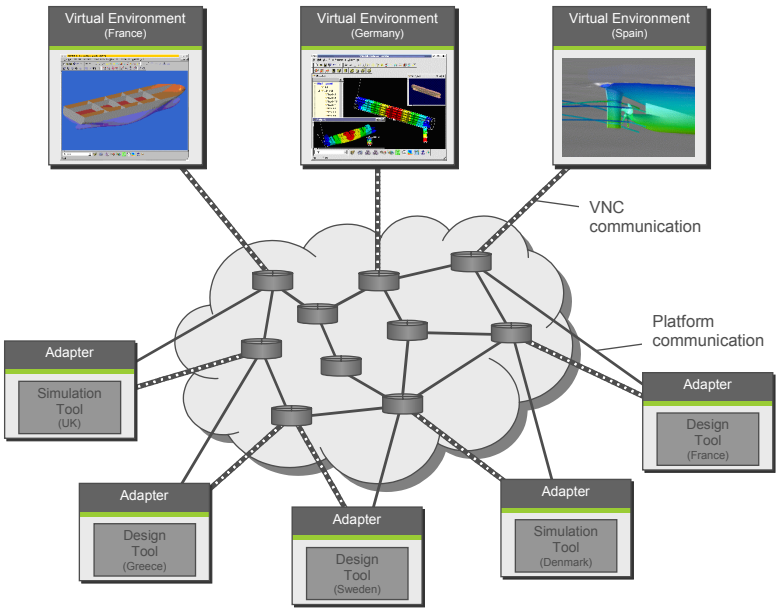

Figure 3. Remote export of visualisation.

Whilst there is currently VNC software available to support the secure communication of the encoded visualisation data, the security issues relating to the fact that the user has access to

the entire desktop of the remote machine were considered to be unsatisfactory. The approach also raised issues relating to licensing of software – not only that used within the platform, but of all other tools available on the desktop of the remote machine, since tools would be available to all partners regardless of whether the user or partner has bought a license for it.

The virtual environment was therefore developed to enable tools that each individual user has available to them locally, to be used within the virtual platform, and for these tools to be available to the users that have registered the tool and not throughout the platform. This change in operation and use of the design and simulation tools, had a significant impact on the rest of the platform: the virtual environment was developed to support the use of the tool to perform particular activities rather than as a means for distributed visualisation and collaboration, which impacted the process control tool with respect to the management of the users and the activities that they can perform, and not the tools that they use to perform them with.

The virtual environment was developed to allow the user to log onto the platform, through interaction with the process control tool. Support was provided to integrate design and simulation tools into the virtual platform through the use of the generic wrapper configuration element. Textual communication between users of the platform was provided, as well as interaction with the inference engine and process control tool to determine the consistency status of data within the dependency maps and start processes for example.

3.4 Inference engine

The aim of the inference engine is to manage and maintain the consistency of the data within the common model through analysis of the relationships between the data. Data that resides within the local models of the design and simulation tools have inherent dependency relationships with the models of other tools and with the common model. Modifying the data within one local model may impact the models of other tools. The inference engine manages these relationships and ensures consistency between the tools by tracking the data usage of each of the tools and activities that are configured for integration into the virtual platform. Since the virtual platform enables multiple users to simultaneously conduct design and simulation activities, the inference engine also controls data access.

The inference engine automatically creates data dependency maps representing the relationships between data within the common and local models. The inference engine does not store any information relating to the values of the data items – it is only concerned with the state of the data. Data items within the inference engine are modelled in a hierarchical level representing the hierarchical structure of the data within the common and local models and may therefore also be used to represent different levels of abstraction. This hierarchical structure does not however contain any relationships across hierarchies to represent that a change in one piece of data (the hull-form) may affect another piece of data (the sea-keeping of the vessel). Relationships across hierarchies may be established either manually via the virtual environment or the inference engine, or automatically during tool integration. When a design or simulation tool is configured within the virtual platform, the tool's configuration contains information relating to the local and common data that will be used as input and output to the tool. The inference engine uses this configuration information to create a relationship between the input and output data that is used as input. In this way, a network of relationships representing a data dependency map may be created automatically.

The inference engine has two modes of operation: an active mode through automatic interaction with the generic wrapper, and a passive mode through manual interaction via the

virtual environment. The active mode is used whenever the user chooses to start an activity that has been scheduled and allocated to them via the process control tool. Functionality is provided within the active mode: for checking the status of the required data before the activity is started to determine if it is already locked for use by another user; locking the data if it is not already locked; automatically notifying other users of the change of the lock status of data that they may wish to use, and the management of potential conflicts that may arise as a result of multiple users modifying separate but related pieces of data.

The inference engine also provides passive functionality, acting as a server and interacting with the virtual environment to enable the users to query the locked or working status of data, facilitate co-operation between users operating on related pieces of data to avoid conflicts, attach notification triggers to data to inform the user when the state of the data changes, as well as modification of the data dependency network.

3.5 Process control tool

The prototype process control tool was developed to manage and enact processes through the communication with the adapters developed within the integration work package in order to start a design tool whenever a hull-form design activity requires enactment for example. Processes within the prototype process control tool consisted of activities that were directly mapped to tools that were capable of performing the activity. Due to tool management limitations of the adapter, it was not possible to provide any additional information to the tool regarding the rationale for undertaking the activity. In addition, each tool that was integrated within the virtual platform was mapped to an activity within a process of the process controller. From a process perspective; the tool could only be used to perform a single activity, whereas in reality a number of the tools were capable of performing a number of different activities. The prototype process control tool also had no formalised modelling of the capability of the resources that were logged onto a platform.

A resource model was created within the process control tool to enable the management of user information and enable them to log onto the virtual platform using the virtual environment in order to be allocated design activities. Resources were modelled within the process control tool as having capability (the measured ability to perform an activity), and commitments (information related to which activities they are currently undertaking, and have undertaken in the past). Information is also modelled with respect to the resource's IP address, as well as other contact details. Within the context of the virtual platform, the process control tool regards a resource as being an autonomous agent capable of performing an activity, and as such manages either human or computational resources. Whenever a tool is integrated, the user is expected to define the activity that they will perform with the tool. This mapping is provided as part of the configuration information of the tool within the virtual environment and not in the process control tool. Processes can therefore be managed and co-ordinated incorporating activities at any level of abstraction. Once the configuration is complete, the virtual environment communicates with the process control tool to update the resource's details with information regarding this additional capability.

The process control tool can manage and enact simultaneous processes consisting of any number of interconnected activities. The activities (and the process control tool) were developed using object-oriented design procedures and are therefore not limited to the activity types defined below:

- **Start Activity**. The process control tool uses the start activity to determine the starting point for the process as well as the activities that follow. Each process has exactly one start activity that is included within each process by default and cannot be removed.

- **Design Activity**. The design activity is used to define the nature of any activity that would be allocated to a resource. Additional information is provided to the resource regarding a description of the activity, as well as an optional list of requirements that the resource will be expected to check to determine whether they have been satisfied once the design activity has been completed.
- **Process Activities**. The process control activities can be embedded within processes to change the state (start, stop, pause, continue) of any of the other processes.
- **AND Activity**. Each of the activities defined above can only have one connection either leading into or out of the activity. The AND activity enables multiple activities to be conducted in parallel by dividing the flow within the process, or waiting for multiple activities to be completed by joining the flow within the process.
- **OR Activity**. The OR activity is used in conjunction with a conditional activity, to indicate that the process flow will continue when any of the preceding activities are completed.
- **XOR Activity**. The process control tool can manage conditions that return a logical (true/false) result to indicate whether the condition has been satisfied such as the requirement for example which has it's status set by a resource when it is associated with a design activity. The XOR activity checks the status of the condition that has been used to define it, and directs the process flow on the basis of the outcome. Conditional connections are used to connect the XOR conditional activity to other activities, to indicate the process flow that would be undertaken for each outcome.

Two approaches are available to determine the most appropriate resource to perform an activity: single activity and multi-process scheduling. Single activity scheduling considers the request for a resource on an activity-by-activity basis. When a user configures a tool to be integrated within the virtual platform, the virtual environment communicates with the process control tool to register the user's new capability. When an activity is due to be performed using single activity scheduling, the process control tool firstly generates a list of the resources that are capable of performing the activity. This list is then filtered to determine which of these user are currently online, as well as which are the most efficient at performing the activity. Single activity scheduling selects the most appropriate resource for each individual activity, without considering the process as a whole and hence cannot guarantee that the process lead times will be optimum.

Multi-process scheduling uses an optimisation algorithm to simultaneously consider all of the activities within all of the active processes that require resources. Using multi-process scheduling, the process control tool will automatically generate a schedule whenever it attempts to start an activity that has not previously had a resource scheduled for it. Using this approach the scheduling becomes dynamic, reacting to the changing process demands, as well as simultaneously considering the most appropriate resources in order to minimise the lead-times of all of the active processes. Whenever an activity is completed, the associated scheduled resource is removed, in order that if the activity were to be repeated due to iteration for example, the scheduling procedure would be repeated and therefore not use the same previously scheduled resource. A shortcoming of this approach is that the scheduling algorithm does not consider the availability of resources during working hours, which is compounded by the fact that the resources may be distributed across various time-zones, as well as the possible variation in the schedule and potential un-availability of a scheduled resource some time the future. These issues could however be addressed by continually

assessing the deviation from the schedule and re-scheduling when the deviation exceeds pre-defined limits [9].

3.6 Simulation engine

The simulation engine represents all of the design and simulation tools that are integrated into the virtual platform. These tools represent the functionality that is required to design the ROPAX vessel from concept to detail, and simulate the performance of the ROPAX vessel with respect to the environment, operations, supply chain and production. In order to enable this design, the associated design and simulation tools require integration within the virtual platform. This integration was initially provided by the adapters as described within Section 3.1. Additional management functionality was however required within this wrapping in order that the tool usage could be co-ordinated. This functionality was provided within WP6 in the form of a generic wrapper, consisting of two separate modules, which would be used to integrate any of the simulation engine tools within the platform.

The focus when developing the generic wrapper was on facilitating the open architecture and providing support for any tool irrespective of the function that the tool provides, the programming language that it was written in, or the platform that it operates on. The configuration module is a graphical interface that enables the generation of tool integration information, relating to the management of input and output data, data conversion algorithms, and design and simulation tools.

Once the configuration of the tool is complete, the associated activity becomes available for enactment within the interface of the user's virtual environment. Information is also sent to the process control tool to inform it that the user is now capable of performing the configured activity.

When an activity within a process has been scheduled to a resource, the process control tool communicates with the virtual environment of the resource informing it that the activity needs to be performed. The enactment module downloads the data from the common model, converts the input data to the native format, runs the design and/or simulation tools, converts the output data to the neutral format once the tool use is complete, and uploads the data to the common model in accordance with the rules defined within the configuration. The enactment module also manages communication with the inference engine to check the lock status of any of the data that it will be using. A dialog is displayed if the required data is already in use (and therefore locked). Alternatively, the inference engine will lock the data for the user during the enactment of the activity.

4 Use of the VRS virtual platform

When the user starts the virtual environment component, they are presented with a login dialog to control access to the virtual platform. The user is expected to provide information relating to their username and password. The process control tool prohibits further access to the platform if the details do not match those contained within an encrypted database. Once the username and password have been validated, the process control tool registers the details to indicate that the user is online in order that activities may be scheduled and allocated to the user in the future.

The first time the user logs onto the platform, the virtual environment will create a new profile, hence the user is expected to use the configuration module of the generic wrapper in order to define the activities that the user can perform – see Section 3.6. Once the user has

completed the configuration procedure, the associated activities that have been mapped are registered with the process control tool as new capability.

The user can visually interact with the data within the common model by selecting the element to view (such as the general layout of the decks for example). This visual representation of the common model data is available to all of the users of the platform in order that every user can visualise the progress of the design irrespective of their expertise or available tools. This representation does not however allow modification of the data in the same way that the design and simulation tools would. The virtual environment also allows manipulation of the data dependency maps within the inference engine, and starting of processes within the process control tool.

When a process is started, the process control tool will firstly attempt to allocate resources to the activities within the process that require resources using either single activity or multi-process scheduling – see Section 3.5. Once a resource has been identified, the process control tool will communicate with the virtual environment of the resource and allocate the activity to them. When the activity is started, the configuration information is loaded into the generic wrapper enactment module, which will extract the information relating to the input and output data that the associated tool will use. The enactment module will then communicate with the inference engine in order to establish the status of the data. If a different user has already locked the data, the inference engine will inform the user that they may only use a copy of the data, or alternatively defer the activity until later. If a copy of the data is used, any data that is generated from the copy cannot be uploaded to the common model. This limitation is made to prevent inconsistencies arising from multiple users simultaneously accessing and modifying the same piece of data. If the data is locked, the user may interact with the inference engine via the virtual environment to either: request notification when the lock is released; establish which resource has locked the data and communicate with the resource in order to collaboratively work on the data. If the data is not locked when the user starts the activity, the inference engine will automatically lock the data, and allow the user to modify it in accordance with the configuration. The inference engine will also check to see if any other user has currently locked (and is therefore working on) any related data through analysis of its data dependency maps. For example, user A may currently be using the hull-form of the ship to conduct a damage stability analysis, whilst user Z is allocated an activity to modify the hull-form. A relationship would exist within the inference engine between the hull-form and the damage stability performance through the configuration of the damage stability analysis tool by user Z. If user A makes a modification to the hull-form, it could impact the simulation activities of user Z. The inference engine would therefore communicate with both users and provide notification that the activities that they are performing could potentially be in conflict with each other. Given this notification of a potential conflict, it is left up to the associated users to ensure that the actions that they undertake do not result with an actual conflict.

Once it has been established that the data either isn't locked, or is locked and therefore copied, the generic wrapper enactment module will download the data from the common model, store it in the defined locations on the user's local machine, run the conversion algorithms, and the pass the converted input data to the tool which is then started. The design or simulation tool may then be used in a manner according to the requirements of the activity. Given the nature of the VRS design problem, many of the design tools require a significant amount (days, weeks or months) of effort in order to generate the required output. Hence provision is provided within both the virtual environment and the generic wrapper to enable the activity to be stopped, and re-started any number of times, without repeating the process of checking for data locks, and downloading the data. The data that is associated with the activity remains locked until the activity is completed.

Once the user completes the activity, the generic wrapper uploads the generated output data to the common model, communicates with inference engine in order to release the data locks, and informs the inference engine that the output data that the user has created has changed. The inference engine uses this information to manage the consistency of the data within the common model, to ensure that any changes that are made to the model are correctly propagated, and to communicate with the process control tool to undertake appropriate activity. The virtual environment displays a dialog to the user showing the requirements that have been associated with the design activity by the process control tool. The user can use this dialog to select which of the requirements that have been satisfied as a result of performing the activity and may be used by the process control tool to take alternative action. The virtual environment communicates with the process control tool to inform it that the activity is now complete, as well as providing information relating to the state of the requirements. If a requirement is not satisfied, the process control tool may either: re-direct the process to activities that are known to affect the requirement, or ignore the failed requirement and direct the process as planned with the knowledge that the requirement will be satisfied later.

The process control tool also has functionality to enable users of the virtual platform to attach notes to the details of an activity once the activity is completed. This functionality enables a user to provide precise details of the activity that they have performed. For example, it may be necessary to modify the general arrangement if it is established that the evacuation time for the vessel is not appropriate, and in doing so, the user responsible for modifying the general arrangement has repositioned a bulkhead. The user can therefore attach a note to state that the bulkhead has been repositioned, in order that any subsequent activities can consider the new bulkhead position. Alternatively, the user may attach a note to state the reason why a certain requirement was not satisfied. These notes are propagated throughout the processes by the process control tool and may be added to or removed when appropriate by the allocated resources and provides a means of directing design activity towards specific issues.

A number of evaluation scenarios have been created in order to test the virtual platform during the development of both the prototype and the current version. The most recent scenario had users distributed across Europe in France, Greece, Sweden, and the UK, logged onto the platform and co-ordinating their activities to demonstrate the design of a vessel, from a hull-form concept, through to the detail design of the hull including hull-fairing, generation of the general arrangement of the decks within the hull using the hull-form profiles at various sections, and finally generating a simulation of the performance of the vessel with respect to the evacuation of 2000 passengers. The focus of these demonstration scenarios was not on the actual design, but on the operation and performance of the virtual platform in supporting the design.

5 Conclusion

The VRS virtual platform was produced through a collaboration of approximately 40 European partners from the shipbuilding, regulatory, consultancy and academic sectors. The overall aim of the virtual platform was to provide long term benefit to the maritime industry as a whole through the embedding of high technology in the very fabric of designing, building and operating the ships of the future. In particular, the platform aims to integrate and simulate critical technologies of ship systems to ensure safe and reliable transportation of goods and people. This necessitated that due attention was paid and concerted effort expended in the conception and development of an integrated platform to allow for systematic integration, management and testing of the various technologies to ensure overall design optimisation and product performance, while addressing all the life phases of the ship including design,

production, operation and disposal. The paper describes the structure of the VRS platform with respect to the various components, as well as the objectives of each of the components.

The VRS virtual platform has had to tackle a number of difficult IT issues that have been compounded by the complexities associated with the design problem. The selection of two key technologies have however aided the success of the platform: Java for the development of all of the virtual platform components; and XML as the underlying language of the communication between the components, the contents of the common model, and the individual models of the inference engine and process control tool.

Acknowledgements

This research is funded by the European Commission (grant No. G3RD-CT-2001-00506), which is part of the Fifth Framework Programme for Research, Technological Development and Demonstration.

References

1. Catley, D. *Prototype STEP data exchanges in ship initial design and the provision of an applications programmer interface to "Tribon"*. in *International Conference on Computer Applications in Shipbuilding*. 1999. Massachusetts, USA: Massachusetts Institute of Technology.
2. *Industrial automation systems and integration - Product data representation and exchange - Part 215: Application protocol: Ship arrangement*. 2004, International Organization for Standardization. p. 1052.
3. *Industrial automation systems and integration - Product data representation and exchange - Part 216: Application protocol: Ship moulded forms*. 2003, International Organization for Standardization. p. 947.
4. *Industrial automation systems and integration - Product data representation and exchange - Part 218: Application protocol: Ship structures*. 2004, International Organization for Standardization. p. 1837.
5. *Industrial automation systems and integration - Product data representation and exchange - Part 227: Application protocol: Plant spatial configuration*. 2001, International Organization for Standardization. p. 1299.
6. Grau, M., Koch, T. *Applying STEP Technology to Shipbuilding*. in *International Conference on Computer Applications in Shipbuilding*. 1999. Massachusetts, USA: Massachusetts Institute of Technology.
7. Whitfield, R.I., A.H.B. Duffy, J. Meehan, Z. Wu, *Ship product modelling.* Journal of Ship Production, 2003. **19**(4): p. 230-245.
8. Rando, T.C., *XML-based interoperability in the integrated shipbuilding environment (ISE).* Journal of Ship Production, 2001. **17**(2): p. 69-75.
9. Coates, G., A.H.B. Duffy, R.I. Whitfield, W. Hills, *An integrated agent-oriented approach to real-time operational design co-ordination.* Articifial Intelligence for Engineering Design, Analysis and Manufacturing, 2003. **17**: p. 287-311.

Robert Ian Whitfield

CAD Centre, DMEM, University of Strathclyde, 75 Montrose Street, Glasgow G1 1XJ, UK
Phone: +44-141-548 3020 Fax: +44-141-552 7896 Email: ianw@cad.strath.ac.uk