MODELING, IDENTIFICATION AND CONTROL, 1997, VOL. 18, NO. 2, 89–100
doi:10.4173/mic.1997.2.1

Multi-Purpose Process Simulators*

OMMUND ØGÅRD[1] and TOR IVAR EIKAAS[2]†

Keywords: *Dynamic Simulation, Operator Training, Simulation Architecture*

The first part of this paper describes a multi-purpose dynamic simulator for the Heidrun oil production plant. The simulator integrates a commercial dynamic simulation model with the actual process control system to enable dynamic simulation, control system verification and operator training within the same framework and based on the same models and configuration data. The control system configuration can be imported directly into the simulator. This simplifies simulator maintenance and ensure consistency between the simulator and the real plant. The entire system runs on standard Unix work stations. The simulator has so far been used for initial operator training, control system verification and controller tuning.

The last part of the paper describes how the simulator architecture can be generalized and some aspects of how new developments in product modelling will influence the use of dynamic simulation in the future.

Introduction

The usage of dynamic simulation is rising in today's industry. The Norwegian offshore oil production industry uses dynamic simulation for different purposes. Typical examples are: design and verification of control structures, controller tuning, verification of the process design, testing of operational procedures, operator training and for verification of process modifications during plant operation.

There has so far been a clear distinction between simulators used for engineering purposes and simulators used for operator training. Engineering simulators are based on detailed first principles models and accurate multi-component thermodynamic calculations. Engineering simulators are usually maintained and used during the plant's whole life cycle. The OTISS simulator from the British company SAST has so far dominated this market segment. Engineering simulators are moderately priced and run on standard work stations.

The simulators are traditionally constructed by the simulator vendors, but currently also engineering companies have started to configure process models, see (Kvamsdal and Sivertsen 1995). The simulator based process verification are done by engineering companies during plant design. Operator training is traditionally carried out in special training centres and simulation studies of plant modifications are often accomplished within the oil companies.

Training simulators, in contrast, are tailored to match the training requirements and

Received 16 December 1996.
[1]Email: Ommund.Ogard@ecy.sintef.no
[2]Email: Tor.I.Eikaas@ecy.sintef.no
*An early version of this article was presented at the SIMS '96 Applied Modelling and Simulation Conference, Trondheim, Norway, June 11–13 1996.
†SINTEF Electronics and Cybernetics, Automatic Control, N-7034 Trondheim, Norway.

to mimic the operators' interaction with the control system. The models are traditionally different from those used in engineering simulators, emphasising real time performance and robustness. Special HW might be required to incorporate the control system and the price is high. The use of different models in engineering and training simulators increase the maintenance costs because two models must be updated when the plant undergoes modifications.

The first part of this paper focuses on the HOPE (Heidrun OPerational Experience) simulator. The second part describes how the HOPE simulator architecture can be generalised into a plug and play architecture. Finally, we point out some future trends and discusses how emerging IT standards may impact the use of dynamic process simulation.

The HOPE simulator

Project objectives

The main objectives for the HOPE simulator were as follows:

1. To develop a multi purpose dynamic simulator for the Heidrun offshore oil production plant that enabled simulation studies, operator training and control system verification.
2. The simulator should use the same configuration files as the control system and the Engineering simulator, in this case an AIM-1000 distributed process control system from Simrad Norge and an OTISS simulation model from SAST.
3. The development costs should be lower than for a traditional training simulator.
4. The functionality should be the same as in other training simulators.
5. Both instructor based and self training should be supported.
6. The simulator should run on standard work stations located in a normal office environment both on board and in the Heidrun operation centre.

The training environment did not need to imitate the control room completely. More emphasis was put on making the simulator flexible and facilitate multiple usage. One rationale for this was the fact that only experienced operators were recruited to the Heidrun platform. The focus of the training is therefore moved towards accustoming the operators to the process dynamics and the operational procedures. A copy of the control room environment was therefore not that important, as long as the Human Computer Interface (HCI) was identical to the one used in the control room.

The HOPE simulator includes models of the main process equipment on the platform, i.e. the systems for wellhead operation, oil separation, gas production and produced water treatment.

The basic training simulator functionality in the HOPE simulator is controlled from an instructor HCI that provides functions for:

- Selection and setting of initial conditions
- Selection and setting of scenarios
- Selection and setting of disturbances while the simulator is running
- Taking snapshots of the current state and saving initial conditions
- Replay or re-run from snapshots

Technical solutions

The key point that made the HOPE simulator feasible, was that the AIM-1000 control system was ported from the target real time HW to general work station HW and the Unix operating system. Each Process Control Unit (PCU) and Operator Control Unit (OCU) will therefore execute as an independent Unix process in the simulator. This allows flexible allocation of the PCUs and OCUs on the available computers. It also improves the quality of the configuration testing because the control system topology and thus also the communication paths are the same in the simulator as in the real plant. The data received from the model is fed into the control system SW just above the input cards and the control signal values are fetched just above the output cards. All the functionality and data flow from input to output cards are thus tested.

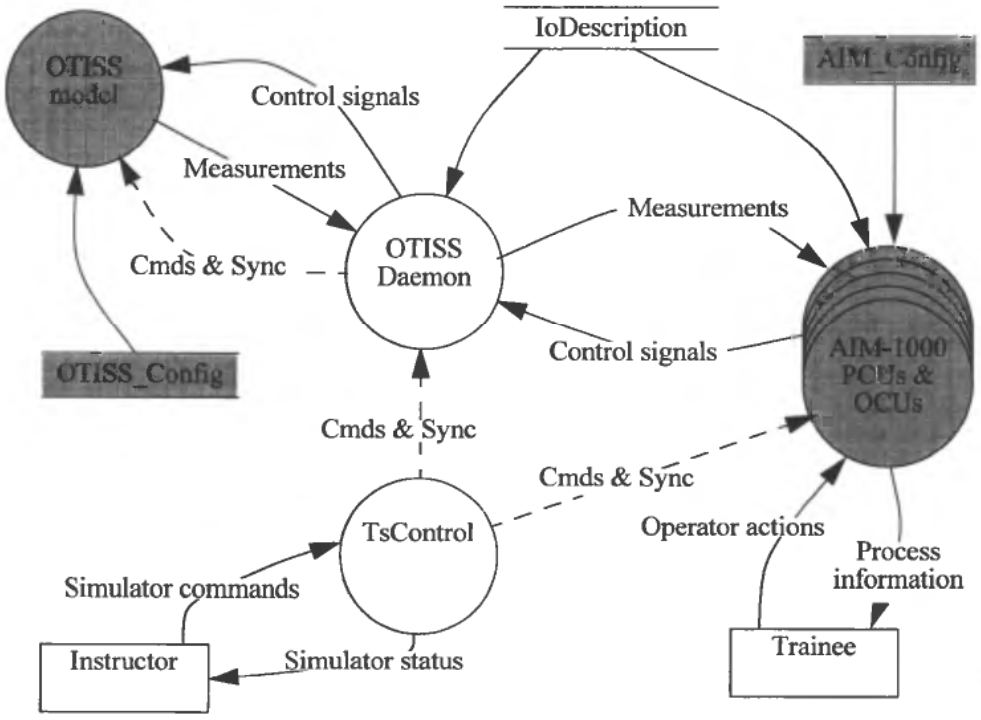

Figure 1. The HOPE simulator architecture.

The system architecture is illustrated in the data flow diagram in Figure 1. The re-used components are shaded. This includes nearly all of the AIM-1000 control system SW, its internal communication protocol and the OTISS simulator. The most extensive SW components in the HOPE simulator are thus re-used. However, some enhancements had to be made in the basic AIM-1000 software to implement snapshots, replay and scenario functions. These modifications, on the other hand, did not interfere with the HOPE simulator's ability to import AIM-1000 configuration files directly.

The Maritime Information Technology Standard (MiTS) protocol was used for all inter-process communication in the SW developed in the HOPE project. MiTS defines an application program interface that allows flexible network transparent inter-process communication. MiTS is based on TCP/IP and is implemented on different Unix

dialects, Windows and several real time operating systems. The HOPE simulator, as well as the AIM-1000 system, are thus designed to utilize several cooperating computers connected in a standard local area network. MiTS is described more in detail in Rødseth and Haaland (1993).

The following SW components in Figure 1 was developed in the HOPE project:

1. TsControl, the instructor HCI, which also synchronises the simulator to real time and coordinates the data exchange between AIM-1000 and OTISS.
2. The command protocol from TsControl to the OTISS Daemon and AIM-1000, and the data exchange protocol between the OTISS Daemon and AIM-1000.
3. The OTISS Daemon which implements the MiTS interface to SAST's proprietary internal protocol.
4. The IoDescription cross connection file defining the data transfer between OTISS and AIM-1000 by mapping OTISS and AIM-1000 tag names.
5. Some utility programs for setting up the IoDescription.

The HOPE simulator runs on four standard HP9000/735 computers. The OTISS simulator and the OTISS Daemon run on one computer. TsControl and the 19 PCUs and 3 OCUs are distributed on the other three. These three computers are equipped with special AIM-1000 keyboards and are used as operator stations during training. This configuration runs in real time at 1 Hz, which is the same as in the actual plant. The number of connection points and default value settings in the IoDescription file are 1150 and 1400, respectively.

```
AO 20FV____0163 20FY____0163 20FICA0163_CADAS 0.0 1.000000 0.000
DO 21XV____0031 21XY____0031 21XV0031R_CADAS 0 -1 1
AI 20LT____0151 20LT____0151 20LT151DE_OUT 0.0 1.000000 0.000
DI 21XV____0031 21ZSH___0031 21XV0031_HI 0 1.0 0.0
AI 37LT____0151 37LT____0151 NOMODEL 12.345 1.000000 0.000
```

Figure 2. IoDescription file example.

Figure 2 shows a section from an IoDescription file containing the descriptions of the analogue input and output (AI, AO) and digital input and output (DI, DO) signals. The first entry is the signal type, the second the AIM tag number, the third is the corresponding name of the AIM I/O terminal, the fourth is the OTISS tag number and the three last are default value, scaling and bias. The IoDescription file thus allows measurements and control signals to be re-scaled before they are exchanged. The last line in Figure 2 sets a default value on the AIM input terminal on the tag number 37LT_ _0151 to 12.34500.

The selected technical solution allows new control system configurations to be imported directly from the process control system's data base to the HOPE simulator. Only minor modifications have to be carried out in the model and in the IoDescription file to reflect a change in the control system configuration. If for example a new transmitter tag has been added to the control system, a corresponding transmitter module has to be configured into the model, and an extra line has to be inserted into the IoDescription file. This new entry in the IoDescription file links the measurement from the transmitter in the model to the input of the new transmitter module in the control system.

The OTISS model may also run completely disconnected from the HOPE simulator using its own internally modelled control system. This is achieved by a set of SW

switches that allow the user to select whether control signals are to be fetched from the OTISS-Daemon or from the internal controllers. OTISS is used as a stand-alone process simulator if the internal controllers are enabled. Consequently, only one model has to be maintained for the Heidrun plant.

The development project

The HOPE development project divided into three main phases. First, software development then simulator configuration and a third simulator verification phase. We used standard structured technics as described in Yourdon (1989) for system analysis and design. The programming was done in ANSI C. We used the POSIX API to access the operating system, OSF/Motif to build the HCI and MiTS for all inter-process communication. About 4.5 man year were spent in this phase. All the software was developed in this phase, but the test configuration did only include a small number of AIM-1000 and OTISS tags.

The goal for the configuration phase therefore was to establish the full simulator. The main activity in this phase was to set up and test all the cross connections between the full OTISS model and the AIM-1000 configuration, and to define static default values for AIM-1000 modules which did not have a counterpart in the OTISS model. This task involved extensive search for, and pairing of, tag numbers in the OTISS and AIM-1000 configurations. The script language Perl was used to automate as much as possible of these tasks. Circa one man year was spent on this task. The result was a simulator that ran satisfactorily around different steady state values. As much automation as possible in this phase was crucial to avoid faults, ensure consistency, speed up the configuration work and enable fast updates when the plant or control system undergoes changes.

The final simulator verification phase was required to make the training simulator fully operable, i.e. to allow the trainee to start up the platform from a shut down state and execute normal operational procedures. This phase included a detailed walk through of the control structures and logics required to start up and shut down the process.

This phase also served as a useful verification of both the control system and the model. Although the control system configuration proved to be of good quality, the needs for some modifications were reported. Shortcomings in the OTISS model were mainly caused by mismatches in the granularity between the model and the control system.

Results and experiences

The simulator has so far been used successfully for control system verification and initial operator training. A copy of the simulator is also placed in the control room on the Heidrun platform to allow on the job training.

The HOPE simulator has also been used to fine tune the control loops by a semi automatic method. Inputs and outputs resulting from system perturbation are logged and used off-line to calculate controller parameters. The method is described in Schei (1994).

The most serious problems during the start up period of the HOPE simulator originate from missing details in the OTISS model. These details are not needed for the operational studies the model originally was designed for, but are important to provide the control system with the necessary inputs and to create realistic operator training. However, it has generally not been difficult to extend the model to overcome

such problems, but the total time spent on this caused some delay in the start-up of the initial training.

One lesson learned is therefore to ensure that the specification of the granularity in the model matches both training requirements and the interfaces to the control system. In addition, means must be provided for adding simple interface modules to tailoring the model to the requirements of the control system. Such a flexibility is needed because the configuration of the simulator will start before the control system configuration is thoroughly tested. The simulator development project must thus be prepared to deal with changes in the control system configuration. To ensure consistency between the control system on the plant and that in the simulator all modifications must be done in the model.

The HOPE simulator concept has been taken over by the AIM-1000 system vendor, Simrad Norge, and is reused in the Norne project which is one of Statoil's next major field developments. Operator training for Norne will take place in the same location as for Heidrun and the same HW will be used for both simulators. The Norne simulator has added one major new feature to the HOPE concept, named Phantom PCU, which is an AIM-1000 PCU connected to the simulator via MiTS. This PCU is completely independent of the rest of the AIM-1000 configuration and is used to generate signals required by the control system, but not available in the model. The configurable PLC functionality available in the AIM-1000 PCUs yields a flexible tool for modelling missing logics in the OTISS model.

Experience from the Norne project shows that current engineering databases do not contain all the information required to configure a simulator. Important data such as compressor and valve characteristics are typically missing. However, the current standard on the engineering databases used to store the control system configuration is fairly complete and allows direct import of configurations from the database to the simulator.

Experience from use of the HOPE concept also indicates that the engineering model should be configured by an integrated team including engineering company staff that knows the process well. The fact that the model also will be used in the training simulator should be accounted for in the requirements for the engineering model. Likewise, the team doing the integration between the engineering model and the control system should contain staff that also work directly with control system configuration.

Multi-purpose simulators

SINTEF is currently working to extend the architecture of the HOPE simulator to make it less dependant of the chosen model and control system vendors, and to allow a more flexible and dynamic connection to computer based training (CBT) and other 3rd party tools (analysis, visualization, controller tuning, optimization, etc.). The goal is thus to be able to easily connect one or more simulators to other relatively self contained units. This work is currently in an early phase where we concentrate on user requirements and possible technical solutions.

The main requirements to the SIMPLY[1] architecture are:

1. Tag based user defined data exchange.
2. Parallel execution of modules and synchronized exchange of data.

[1]SIMulator plug and PLaY.

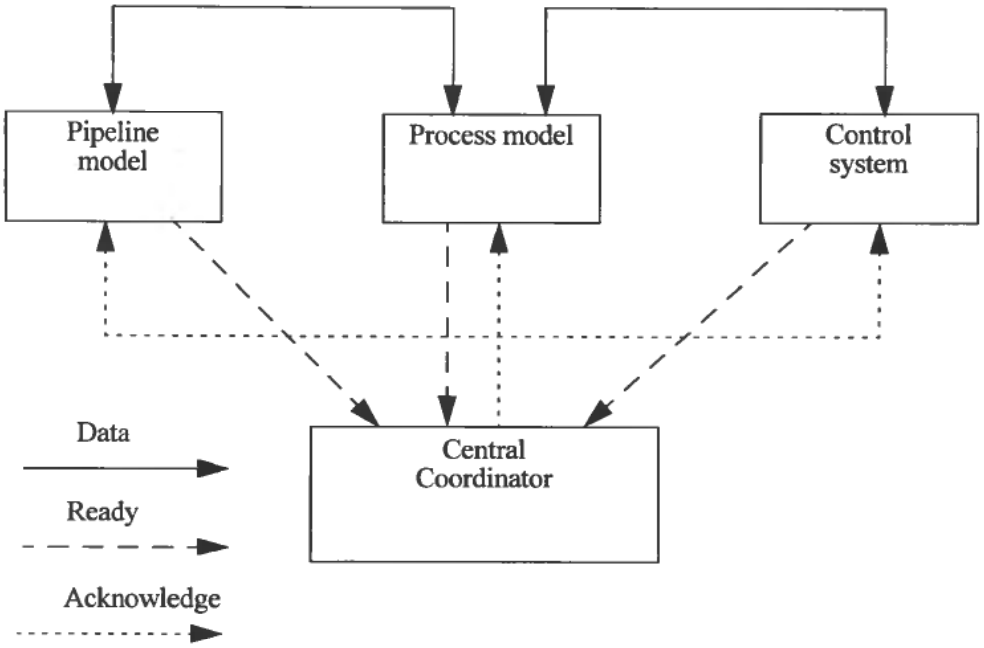

Figure 3. Data exchange synchronization.

3. Self-organizing connection setup. No centralized tag database.
4. Multiple fixed exchange frequencies.
5. A common and extendable command protocol.
6. A logical network with a common API to allow flexible interconnection of modules and transparency of the physical network.

A tag in the SIMPLY architecture is the least uniquely named chunk of data that can be exchanged between to SIMPLY modules. The content of a tag might thus range from an atomic value to a complex data record. A module will typically subscribe on tags from other modules and provide a set of tags other modules can subscribe on. Each module thus has to have knowledge about the tags it receives from other modules.

Parallel execution of SIMPLY modules requires synchronization of the data exchange at fixed intervals in simulated time, especially when the modules are executed without time constraints on different computers. The simplest and most reliable solution is probably to let a central coordination module handle this. Each module then sends a ready signal to the central coordinator when their computations have reached the next data exchange time. The module then enters a wait state. The coordinator broadcasts an acknowledge signal when it has received the ready signal from all modules. This causes the computation towards the next exchange point of time to start in parallel in the modules. This data and control flows are illustrated in Figure 3.

Self-organizing system setup is obtained by letting each module at connection time broadcast a request for the tags it needs. The other modules which issue any of these tags then answer and the subscriptions are established. The addition of a new module also triggers the other modules to send a new request for the tags they eventually are

missing and the new module establishes a subscription if it provides any of the requested tags. Each module sends a ready to start message to the central coordinator when subscriptions are established on all required tags. Finally, the central coordinator broadcasts an enable compute message when all connected modules are satisfied. This connection procedure thus halts when all SIMPLY modules have found suppliers of all their required tags.

Different exchange frequencies make both the data exchange and the self configuration more complex because frequency information has to be handled in the set up procedure and tags with different exchange rates have to be sent in different network messages. Different exchange and in particular non-multiple exchange frequencies might also corrupt or make the numerics inside the modules more intricate. Therefore, if multiple exchange frequencies are used, the longer exchange intervals have to be multiple of the shortest exchange interval.

The command protocol in the SIMPLY architecture must include at least the most common commands found in training simulators. These are: Run, Pause, Load initial value, Store current state as initial value, Load scenario, Activate disturbance, Run in replay mode. Commands generally have to refer to the same point in simulated time in all affected modules. This can be achieved by letting the central coordinator work as a command buffer. A command is thus first sent from one module to the central coordinator where it is buffered until all modules are synchronized in time and then broadcasted to all modules. It is thus up to each module to determine which commands it shall react on and how it shall interpret the command arguments.

There are in principle two possible solutions to the last requirement. Either to develop a specialized software layer on top of a standard network protocol such as TCP or UDP, or to use one of the emerging object oriented integration mechanisms such as CORBA or OLE. The underlying network mechanisms should in any case be hidden from the application programmer by a well defined API. The main components in this API is illustrated in Figure 4. SIMPLY module consists of one NetNode object and a set of Tag objects.

The NetNode object is responsible for receiving and sending net-messages and the corresponding mapping of net-messages to commands and data. It also has to take care of the synchronization with the central coordinator and to trigger the application side when new data or a command needs to be processed. The Tag objects are the application programmers interface between the application data and the NetNode.

This type of generic simulator architecture will allow vendor independent integration of models and control systems, and it will provide a flexible test bed for evaluation and verification studies of both control system and operational procedures. Easy access and flexible integration will also open up for new applications of computer based training. The need for specialised and expensive training centres will diminish and there will be more room for self-training and simulator based on the job training. The simulator will be available in the operational environment and can be used both for scheduled training programs and for more ad hoc preparation for critical operations.

Standardisation efforts on product models

In order to get acceptance for a modular and flexible simulator architecture like SIMPLY, it must take into consideration the ongoing standardisation efforts on product models and information exchange.

The STEP community represents a major effort in establishing standards for product

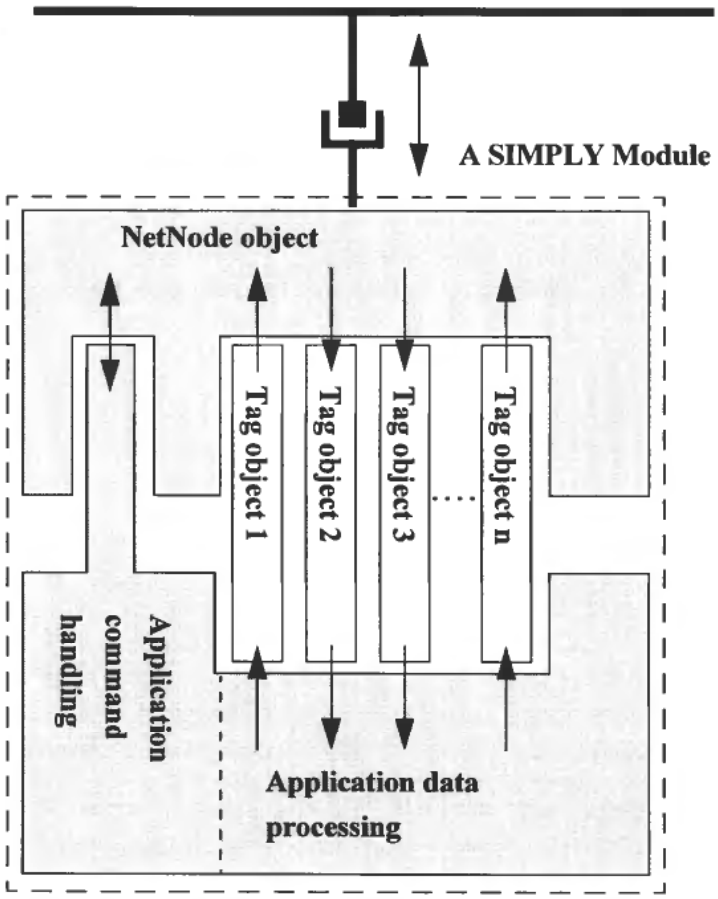

Figure 4. The SIMPLY module SW.

model data. These standards have traditionally focused mostly on mechanical design, especially of ships, cars, planes, etc. However, there is now a strong activity on establishing standards for formal description of process plants and exchange of process engineering data.

On the dynamic modelling level, a number of high-level modelling languages like Omola, Ascend and Model.La have been developed during the last years. Some low-level model interface descriptions like DSblock exist for complex dynamic systems allowing the exchange of models or model components between different modelling and dynamic simulation environments.

The POSC/CAESAR project is a collaboration of Petrotechnical Open Software Corporation (POSC) and the CAESAR Offshore Project to develop standards to meet requirements for efficient electronic exchange and sharing of oil and gas facility life cycle information. There are currently seven specifications that POSC supports. The most important in this context are Base Computer Standards (BCS), Data Access and Exchange (DAE) and Inter-Application Communication (IAC).

Process Data eXchange Institute (pdXi) is another important step towards a standard

for process information exchange. The purpose of pdXi is to develop and maintain open approaches to exchange and manage process data between computer applications, databases, and organizations within the process engineering discipline and with other disciplines. An important result from pdXi is a set of data models encapsulating relevant conceptual design information from operating companies, software vendors and engineering contractors. The major components in the pdXi model are *Planning Level, Process Simulation, Physical Properties, Materials* and *Process Equipment Models*. An application has been submitted by the ISO/STEP organisation to provide an application protocol (ISO 10303/AP231) Process Engineering Data: Process Design and Process Specification of Major Equipment) for conceptual process engineering. See Baldwin and de Almeida (1995) for more details.

Within the European ESPRIT program the KACTUS project (Esprit Project 8145) has focused on knowledge modelling and reuse of knowledge from a more generic perspective. Here, the oil production process has been described primarily from a functional and operational point of view with emphasis on establishing formal description of operational procedures and control system requirements and structures. Some of the results from the KACTUS project are presented in Øgård *et al.* (1995).

These emerging standards will in the future enable different types of applications to share data from a common engineering database because the structure and interpretation of the data are documented and made available through the APIs. The ability to connect dynamic simulators directly to an engineering database and product models will simplify the task model building and updating. The plant topology and all physical parameters are made directly available from the engineering database and the simulator model will be updated with values which are consistent with the current stage in the engineering project.

This together with a formalized representation of operational procedures and control structures will cause the use of dynamic simulation and model based analysis to be more firmly integrated into the engineering project. This will apply for both plant and control system design as well as for the design of operational strategies and procedures.

Impacts on simulator usage

The engineering process

A flexible simulator architecture together with close integration between engineering databases and simulation will enable fast and efficient analysis and verification of design decisions in a tight closed loop fashion. The turnaround time between analysis, design decisions and verification will thus be curtailed. A consequence of this might be that the traditional waterfall project model and the document producing organisation of large engineering projects will be turned more towards a spiral development model consisting of successive steps of requirement analysis, design and verification as illustrated in Figure 5. The design process will be more directed towards database completion than towards document production and the verification will rely on simulation based testing of the design decisions stored in the database.

The final output of the engineering process might thus be to give the oil company and the equipment vendors access to a standardized and verified database. This in contrast to the current situation where the engineering projects result in large piles of paper based documentation which has proved to be very hard to maintain and take into efficient use in the production phase.

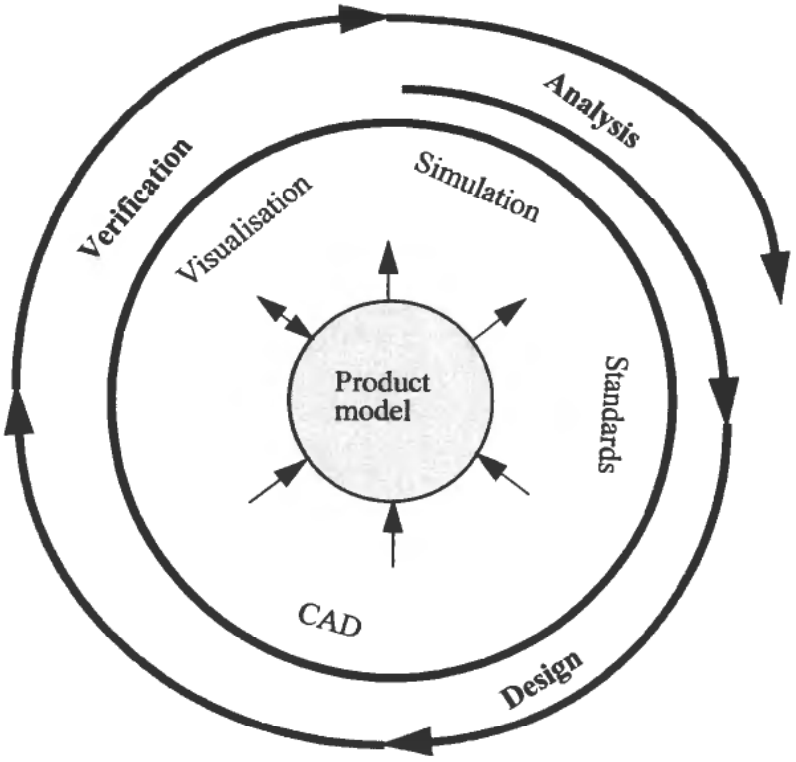

Figure 5. The use of simulation in a spiral project model.

Operator training and training simulators

This type of generic simulator architecture will allow vendor independent integration of models and control systems, and it will provide a flexible test bed for evaluation and verification studies of both control system and operational procedures. Easy access and flexible integration will also open up for new applications of computer based training. The need for the traditional, specialised and expensive training centres will diminish and there will be more room for self-training and simulator based on the job training. The simulator can be made available in the operational environment and used for both scheduled training programs and more ad hoc preparation for critical operations.

A flexible simulator architecture and close integration with engineering databases will allow more comprehensive use of CBT for process operators. The simulator training can be enhanced with visualisation tools connected to the same product model to show the physical location and appearance of, for example, the failed component causing the disturbances in the simulation. Virtual reality capabilities may extend this potential further into maintenance planning and preparation.

An open simulator architecture might also open new market segments for creative software companies to develop CBT tools independently of the traditional model and simulator vendors. There is currently a need for improved self training facilities and especially technics for evaluation of the operators' actions and problem-solving during a training scenario.

A close connection to product models and control system configurations will ease simulator maintenance and ensure consistency between simulator and plant.

Process maintenance and operation

Easy access to a flexible simulator with well defined APIs and connection mechanisms will increase the ability to improve process and control system design, to identify bottlenecks and test and verify optimization strategies. The HOPE simulator has, for instance, been extended with a general API that allows simple access for other applications both to the command protocol and to process data within the simulator. Further extension of the API will allow a flexible and dynamic connection for other 3rd party tools for analysis, controller tuning, optimization, etc.

REFERENCES

BALDWIN, J. and DE ALMEIDA, J. (1995). "Information and data exchanger breakthrough—the new pdXi interface", AIChE 1995 Summer National Meeting, Boston.

FRAY, R. (1995). "Compact simulators for fossil-fuelled power plants". IEEE Spectrum, Vol. 32, No. 2.

KVAMSDAL, H. M. and SIVERTSEN, T. (1996). "Dynamic simulation—requirements to topside flow sheet simulators based on experience from the Njord simulator". In *Proceedings of the 38th SIMS Simulation Conference* (Trondheim, Norway, June 11–13). Edited by Torleif Iversen, Scandinavian Simulation Society.

RØDSETH, Ø. J. and HAALAND, E. (1993). "MITS: An Open Standard for Integrated Ship Control". Proceedings of ICMES 93, Hamburg, September 1993.

SCHEI, T. S. (1994). "Automatic Tuning of PID Controllers Based on Transfer Function Estimation". Automatica, Vol. 30, No. 12, pp. 1983–1989. Elsevier Science Ltd.

YOURDON, E. (1989). "Modern structured analysis". Prentice-Hall, Inc.

ØGÅRD, O., WAHL, P. E., EGGEN, J. H., LANGELAND, O., TELNES, K. and SJONG, D. (1995). "An ontology for process operation". To be published in the forthcoming book "*Artifical Intelligence in the Petroleum Industry: Symbolic and Computational Applications II*", Bertrand. Braunschweig, ed. Editions Technic 1996.