Incremental Development of Production Units Using HW/SW-Encapsulation and Simulation Supported By a Complete Toolset

Uwe Schmidtmann, Gerd von Coelln, Gerhard Kreutz, Bodo Wenker, Rainer Koers
Institut der Informatik fuer Automatisierungstechnik und Robotik (I^2AR)
FH OOW Emden
{Schmidtmann,Kreutz,Koers,Wenker}@i2ar.de

Abstract

Manufacturing systems are developed and configured using different design tools, e.g. CAD for mechanical construction, CAE for electronic engineering and IDE for software development. These tools work on different views of automation units, which are the basic blocks of manufacturing systems. With this, tool dependent data is generated that relates to automation units involved in the manufacturing process. These tool dependent data sets can be seen as data objects, which are disjunctive in terms of information content.

This paper presents a framework, which allows the integration of different design tools. The framework is based on concept of holistic mechatronic objects (HMO), where automation units and related tool dependent data sets are seen as mechatronic objects and related data objects. The framework allows 3D simulation of manufacturing systems on high performance PC clusters using the VRML export of CAD tools. The virtual reality modelling environment interacts with a simulation of the control environment. Since 3D-information is provided and available via the framework, software for automation units can be implemented and tested in parallel to the hardware development, reducing development time.

1. Introduction

Concurrent manufacturing means handling of complexity and scalability (cf. EU [8, 9], the National Academy of Science in the USA [5], the ARC Advisory Group [1, 2] etc.). Complexity can be mastered by formal methods and tools used in a strict top-down process based on UML and other specification methods. An example for this approach is the new architecture for a model driven development of manufacturing systems proposed by Thramboulidis [19, 20]. On the other side, scalability has to deal with reconfiguration of production units. Producing several customized products on the same process line requires adaptable manufacturing processes and efficient system reconfiguration. Therefore modular equipment is needed with plug-and-play hardware and software components. A construction kit composed of intelligent building blocks together with an integrated tool-chain allows to fullfil the requirements of the future manufacturing (cf. OOONEIDA initiative http://www.oooneida.info).

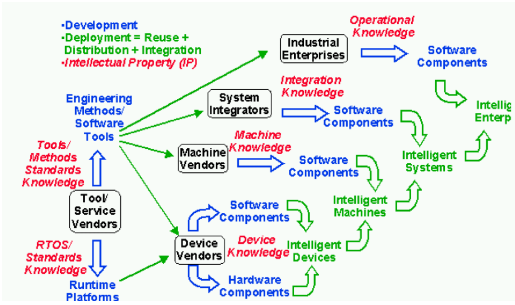


Figure 1. OOONEIDA: Encapsulation of Hard- and Software Know How

Starting in bottom-up direction, the main focus concentrates on providing scalability and flexibility. A production line is made upon building blocks like conveyors, feeders, cutters, etc. where each part is controlled by PCs which are connected with actors and sensors of the unit. Traditionally the mechanical part of such a building block is designed by a mechanical engineer using a CAD tool. Electrical engineers use a design tool for wiring the actors, sensors and PCs. After assembling, a software engineer starts software developing using another IDE tool. All the tools are application specific and not integrated in an overall tool-chain. This means that e.g. the name of a sensor given in the CAD tool is not available in the IDE tool used by the software developer. This is a tremendous waste and a resource of errors. Also, it is difficult to set up integrated verification and simulation tools on such an infrastructure.

The term mechatronic object (s. Amerongen [21], Hewit [12]) was introduced to refer to a synergistic combination of mechanical and electrical engineering and information technology". Bonfè and Fantuzzi [6, 7] have located the control algorithm of a mechatronic object in a reusable function block, on which a common object-

0-7803-9402-X/05/$20.00 © 2005 IEEE

oriented framework for top-down modelling process could be based upon (see e.g. [19, 20]). The mechanical information of a mechatronic object contained in CAD files is still unused. E.g., the name of the driving axis of a conveyor belt could be used in an animation of 3D simulation of the conveyor since the transform node with this name in the VRML export could be connected with a script. The name of a field-bus unit controlling the motor of a driving axis (CAE) should be exported to the IDE such that the software engineer can use it for a transparent naming scheme. In order to provide the information of a building block, hosted at different engineering tools in a transparent manner, a concept is needed that relies on a model-driven framework for content repurposing and synchronizing engineering data. This problem is well known from multimedia applications and business applications. We propose to introduce the concept of a holistic mechatronic object which represents the MO based on the original synchronized data provided by the tools.

On the basis of three existing tools a link between the CAD, CAE tools of EPLAN (LOGOCAD/TRIAGA and ePlan5, http://www.eplan.de) and the IDE tool of SENGATEC (*ems-drd*, http://www.sengatec.de) is established.

Our concept relies on a message based and distributed control platform. Since the devices encapsulate the I/O operations, applications have to deal with sending and receiving messages, only. Therefore, if real drivers are replaced by virtual drivers, the connection between the real world (VRML) and a simulation of the underlying I/Os is established. Connecting 3D simulations based on VRML exports with the I/O-Hardware simulation establishes a powerful environment which enables software development in parallel to hardware construction. The *rt-dvm*, which was created by the company SENGATEC and the University of Applied Sciences in Emden, meets all demands and will be used for a first implementation.

This paper is organized as follows. Section 2 gives an overview of the real-time distributed virtual machine (*rt-dvm*) and the IDE tool-set *ems-drd*. Section 3 presents a framework for integrating three tools LOGOCAD/TRIAGA (CAD 3D), ePlan5 (CAE) and *ems-drd* (IDE) such that a mechatronic object can be seen as a building block. Section 4 describes the model of 3D simulation on a PC cluster based on the VRML export of LOGOCAD.

2. IDE ems-drd and the real-time distributed virtual machine rt-dvm

Following the idea of the OOONEIDA-initiative, a development system was created [17], which implements two major concepts of OOONEIDA. On one hand a distributed, real-time capable, virtual machine was created (real-time distributed virtual machine, *rt-dvm*), to transparently program the control logic within a distributed system. On the other hand a set of development tools was implemented (easy modular system for distributed real-time development, *ems-drd*), to provide an easy access to the functions of *rt-dvm* during software development.

rt-dvm platform implementation

The *rt-dvm* operates within the real-time kernel of the underlying operating system (RT-OS), which serves as an abstraction layer. Changes within the control applications do not become necessary when changing to another hardware platform or to another RT-OS.

The *rt-dvm* itself is based on Linux in combination with RT-Linux from FSMLabs Inc. (http://www.rtlinux.org). A port to RTAI (http://www.aero.polimi.it/~rtai) does exist as a beta version. The approach to use a micro kernel within Linux is considered to be useful, because it offers all internet technology for non-realtime Linux and uses real-time scheduling only for time-critical applications. Another advantage is that the source code of the *rt-dvm*, Linux and RT-Linux is fully available. Even after years of usage, the user can modify his own system.

The *rt-dvm* consists of a message routing unit, called core-dvm together with driver and control objects. System tasks start and monitor the failsafe operation of the DVM together with agents that run within normal Linux user space. The Linux agents can be addressed over Ethernet, for instance by a development workstation. They are used for application development, supervision, performance measurement etc. of the entire system. Driver and control objects can communicate in real time on one node, or, using a real time Ethernet solution, over several nodes. (see Fig. 2.)

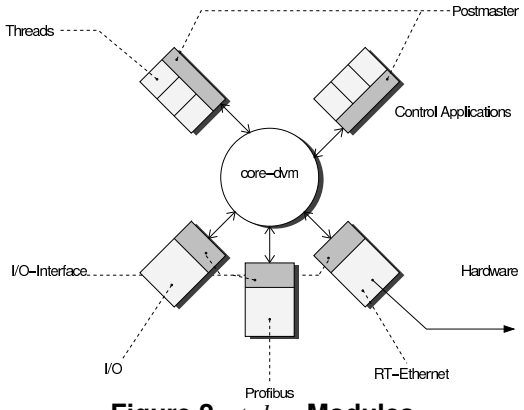


Figure 2. *rt-dvm* **Modules**



Figure 3. *rt-dvm* **message format**

For the communication a homogeneous message format (see Fig. 3) was defined. The *rt-dvm* wide addressing of an entity occurs via the pair *node_id* and *port* with

```
<Port> ::= < device>|<task >
```

The port specifies the appropriate message queue for the needed operation.

All values in the message header are interpreted in BIG-ENDIAN format. Several control tasks can be

grouped into user modules. A specific predefined task within the module takes responsabilty for starting, stopping and time supervision of the control tasks. This postmaster also organizes queues for incoming messages addressed to the contained user tasks and a timeout system for receiving purposes.

Devices under *rt-dvm*

Each device has a uniform interface which consists of three message boxes (see Fig. 4): Over the raw Port (RAW) data is received that suits exactly the underlying hardware. Without further processing these messages are executed. The blocked-message port (BLK) can reassemble fragmented messages or split up several messages that arrive within one buffer. The control-message port (CTL) receives control commands for the device. These can activate/deactivate interrupts, start a periodical notification or put the device into a certain state.

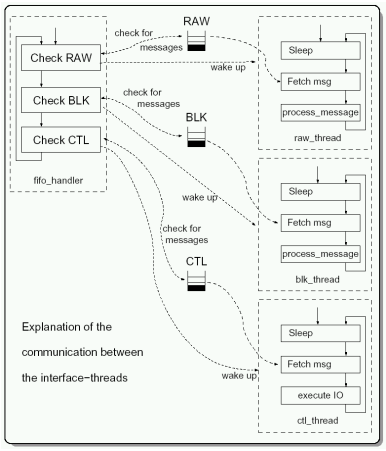
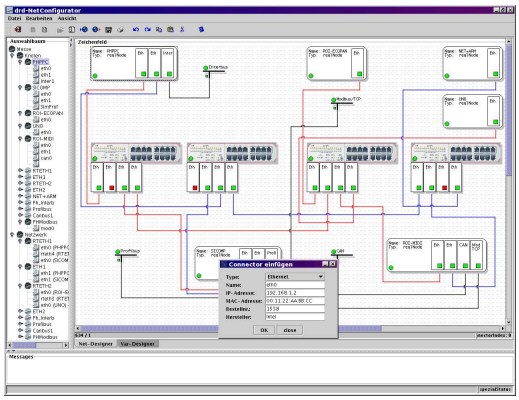


Figure 4. RT-DVM devices

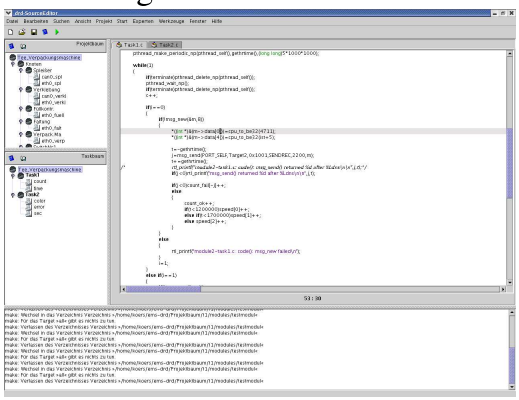
The device specific-format of a *rt-dvm* message is defined through an XML file, to allow an easy access for configuration and programming tools.

Development Tools for *rt-dvm*

Several tools have been developed, that simplifies the usage for an unexperienced user. The development programms are grouped into several sets – in the following paragraph only the most central tools are described. The base set consists of the planning and programming instruments *drd-NetConfig* and *drd-SourceConfig*. The process of making hardware accessible under Linux and the monitoring issues must be done manually. With the *drd-NetConfig* (see Fig. 5) a project can be planned and all fieldbuses, IO's and communication paths are registered and checked for consistency. Functional parts can be combined into *virtual nodes* and all configuration data is stored into the database. Virtual nodes can be exported in an XML File. Functional parts are stored in a database, allowing the reuse in following projects. The *drd-NetConfig* produces a directory structure according to the specification of the *rt-dvm* and also generates the necessary configuration files. Drivers for (fieldbus-)cards can be loaded into an online target system or be written into the off-line compact flash image.



NetConfig



SourceConfig

Figure 5. Multi-Developer environment

When hardware configuration is finished, *drd-SourceConfig* helps the user to develop his application. Compiling and downloading of binaries are functions of this tool, as well as debugging an application on the target system. Here, all configuration data from the NetConfig is available, for instance the symbolic names of the nodes, available fieldbusses, IOs and IP addresses.

3. Holistic mechatronic objects and the process information bus

Reducing the time needed for reconfigurations leads to shortening the engineering process. A platform which enables visualization and simulation of mechatronic objects at an early stage will improve the developing process. In the traditional way of plant simulation (cf. Ong/Nee[15] and Mayr [13]), the mechatronic object (MO) will be reconstructed with specific tools instead of using the 3D CAD information of the engineering tool.

Investigation on available CAD tools showed that VRML files are exported with a reduced granularity. Dealing with less complex objects should be beneficial for a cluster based simulation, but during VRML export sometimes also important information about kinematics is

lost. This information can be extracted, using additonal tools and manpower.

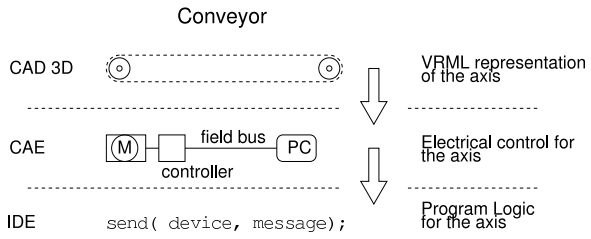


Figure 6. Obtaining information out of a tool-chain

The mechatronic object ”conveyor” depends on several tools, CAD 3D, CAE and an IDE. Normally, all the tools are disjunct (cf. 6). Only, with some effort and going into details of the individual tools additional information can be obtained, e.g. from corresponding listings. To shorten the design cycles it is necessary to provide a unified metamodel for mechatronic objects, which synchronizes knowledge and common concepts from different tools into one integrated view (see 7). This approach could be similar to the model of Obrenovic, Starcevic and Selic [18].

In order to obtain such models with common metatransformation rules our project starts with a concrete tool-chain based on the CAD tools of our partner EPLAN and our IDE platform *ems-drd* (cf. 8) to get a better imagination of what will be feasible.

EPLAN has started a similar project which aims to connect the own CAD tools, in a very sophisticated way, by a so called process information bus. The framework of EPLAN allows the creation of abstract objects based on the distributed CAD tools. At present, an interface to our IDE is developed such that the process information bus can be used in each direction. I.e. on the way up, the lower level of the metamodel is created to see what could be reached.

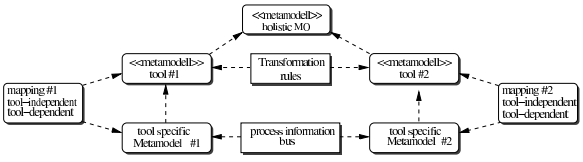


Figure 7. Metamodel for holistic mechatronic objects

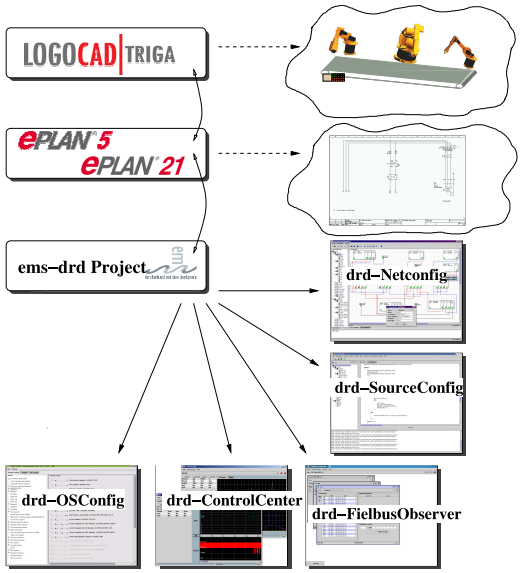


Figure 8. Tool-chain with mechatronic objects

4. Simulation

A simulation based on 3D models, supported by simulation tools is the logical consequence and an additional goal for our project. With this, a few changes have to be made to the *rt-dvm* to make it suitable for a simulation.

Changes to the rt-dvm

All devices of *rt-dvm* are addressed via messages. It is easy to substitute a driver for real hardware (IO, Field-bus etc.) by a virtual driver (shown in Fig. 9) which forwards all IO requests to a simulation system.

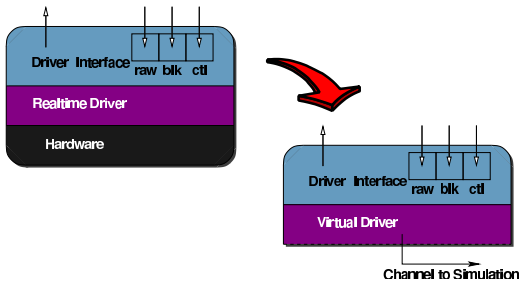


Figure 9. Changing driver objects for real hardware to virtual drivers for simulation purposes

Incoming data has to be forwarded into the non-realtime part of the target OS to be send to the simulation system. According to the specification of the *rt-dvm*, functionalities of the *rt-dvm* may only be used within *control modules* and *driver interfaces*. System specific calls may be used within the *rt-dvm* implementation itself and within real-time drivers. The drivers are not needed for simula-

tion, so porting of the virtual machine itself to the simulation platform should be sufficient to fully abstract from the target system. Nevertheless, simulation configurations are imaginable, where parts are simulated and other parts consist of real hardware.

Distriuted simulation concept

The simulation system will consist of three major modules, which will be implemented using a J2EE technology on a PC cluster. For the visualization and interaction a 3D terminal is needed.

Simulation Agent: The communication partner for real or simulated target nodes is the simulation agent. There will be multiple instances of this agent running on different cluster nodes. A broadcast method will ensure load balancing.

IO Simulator: All information about the underlying hardware will be simulated in this environment (see Fig. 10). So each signal on electrical connections and resulting actions are defined within this software module Therefore it is important that all instances can communicate with each other. The data for the corresponding mechanical operation is transmitted to a Java 3D-Simulator. So collision detection and other interaction is done in Java 3D.

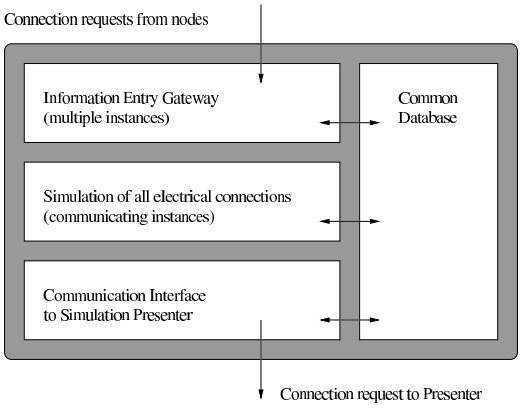


Figure 10. Conception of the IO simulatior

Simulation Presenter: The complete 3D-Model is computed here. *Java3D* will be used on a PC cluster (see Fig.11). At the present, there are two options which could be chosen. First, visualization could run on a Cluster using the technology developed by Guarraciano, Laccetti and Romano [11]. The other option consist of separating the mechatronic objects into separate scene graphs which could be treated separately.

Information about collisions of mechanical objects is handled by the corresponding sensor nodes, which transmit information back to the *IO Simulator*.

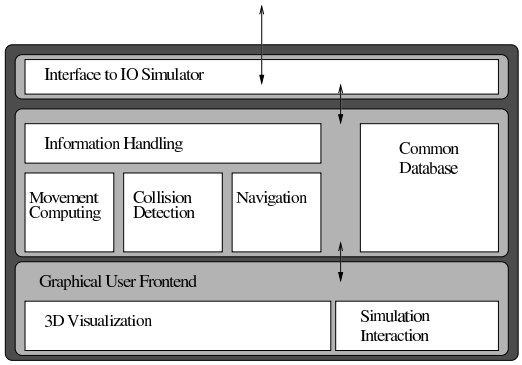


Figure 11. Conception of the IO-simulatior

5. Conclusion

The bottom-up approach of building holistic mechatronic objects together with a simulation and visualization, meets the requirement of the Production 2020. This paper presents an approach for an integrated simulation environment which will speed up the development process. Furthermore, it is expected that merging both approaches bottom-up and top-down, e.g. a MIM model form [20], will lead to improved productivity.

References

[1] ARC. PLC's: Is There a Future? Technical report, ARC Advisory Group, Enterprise & Manufactoring Strategies for Industry Executives, Three Allied Drive, Dedham, MA 02026 USA, http://ARCweb.com, 7 2001.

[2] ARC. Automation for the Discrete Industrie Worldwide Outlook. Technical report, ARC Advisory Group, Enterprise & Manufactoring Strategies for Industry Executives, Three Allied Drive, Dedham, MA 02026 USA, http://ARCweb.com, 10 2003.

[3] I. M. Author. Some related article I wrote. *Some Fine Journal*, 99(7):1–100, January 1999.

[4] F. Benzi, G. Buja, and M. Felser. Communication architectures for electrical drives. *IEEE Transactions on Industrial Informatics*, pages 47–53, February 2005.

[5] J. Bollinger and et al. *VISIONARY MANUFACTURING CHALLENGES FOR 2020*. National Academy Press, Washington DC, 1998.

[6] M. Bonfè and C. Fantuzzi. Mechatronic objects encapsulation in iec 1131-3 norm. *Proceedings of the 2000 IEEE International Conference on Control Applications*, pages 598–603, September 2000.

[7] M. Bonfè and C. Fantuzzi. Design and verification of mechatronic object-oriented models for industrial control systems. *Proceedings ETFA'03 IEEE International Conference on Emerging Technologies and Factory Automation*, 2:253–260, September 2003.

[8] EU-Commission, editor. *MANUFACTURE — a vision for 2020 Assuring the future of manufacturing in Europe*, Luxembourg, 2004. Office for Official Publications of the European Communities, ISBN 92-894-8322-9.

[9] EU-Commission. Towards an IST Strategy for manufacturing. Technical report, EU-Commission, Draft Internal Report, 2004.

[10] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Elsevier/Morgan Kaufmann, Amsterdam, Boston, Heidelberg, London, San Francisco, 2004.

[11] M. Guarracino, G. Laccetti, and D. Romano. A model-driven approach to content repurposing. *IEEE International Conference on Cluster Computing*, pages 201–208, December 2000.

[12] J. Hewit and K. Bouazza-Marouf. Practical control enhancement via mechatronics design. *IEEE Transactions on Industrial Electronics*, 34(1):16–22, Februar 1996.

[13] H. Mayr. *Virtual Automation Environments, Design, Modelling, Visualisation, Simulation*. Marcel Dekker, N.Y., Basel, 2002.

[14] Z. Obrenovic, D. Starcevic, and B. Selic. Model-driven approach to content repurposing. *IEEE Multimedia*, 11(1):62–71, Februar 2004.

[15] S. Ong and A. Nee, editors. *Virtual and Augmented Reality Applications in Manufacturing*. Springer, London, Berlin, Stuttgart, 2004.

[16] U. Schmidtmann, G. Kreutz, B. Wenker, and R. Koers. Eine echtzeitfhige, virtuelle Maschine auf heterogenen automatisierungsclustern. *GI Eingebette Systeme, PEARL2004 Boppard*, pages 80–86, September 2004.

[17] U. Schmidtmann, G. Kreutz, B. Wenker, and R. Koers. ems-drd – a new open platform for distributed realtime programming. *Proceedings of the Second International Conference on Industrial Informatics INDIN'04*, pages 80–86, June 2004.

[18] B. Selic. A model-driven approach to content repurposing. *IEEE Computer Society*, pages 62–71, March 2004.

[19] K. Thramboulidis. Model integrated mechatronics: An architecture for the model driven development of manufacturing systems. *IEEE International Conference on Cluster Computing*, page 6, 2004.

[20] K. Thramboulidis. Model-integrated mechatronics – toward a new paradigm in development of manufacturing systems. *IEEE Transactions on Industrial Informatics*, pages 54–61, Februar 2005.

[21] J. van Amerongen. The role of control in mechatronics. *Engineering Science and Education Journal*, pages 105–112, June 2000.