Design of Distributed Real Time Systems in Process Control Applications

Oliver Hammerschmidt Holger Vogelsang
University of Karlsruhe
Institute for Microcomputers and Automation
Haid-und-Neu-Str. 7
D-76131 Karlsruhe, Germany
Tel.: +49-721-608-3898
Fax: +49-721-661732
email: {hammer | vogelsang}@ira.uka.de

ABSTRACT

In order to handle the complexity of software for automation systems of larger scale in manufacturing nowadays procedural-oriented (e.g. SCR, RTSA) and object-oriented methods (OOD, OOA, OMT) are used. Within the latter alternative we developed an object- and service-oriented approach to cope with problems of complexity and to ease and accelerate the software design process.
In this paper we present our service-based concept, give a possible definition of basic services and discuss experiences made in an application example of a production cell.

KEYWORDS

Computer-aided system engineering, services, process control, rapid prototyping, distributed and parallel processing, real time systems

INTRODUCTION

The amount of hardware and especially software for systems in automation and control is still increasing. The importance of appropriate design methods for distributed real-time systems is however still not completely understood and has to be improved [Whi93,Oli93,Lav91]. The software used in automation applications represents a mixture of data acquisition, control, data storage and man-machine functions for visualization and user interaction. They are hard to handle due to real time restrictions and intensive interaction with the environment.
Object-oriented techniques using objects or agents of application-specific types and libraries lead to a service-based concept which allows the definition of frameworks with precise guidelines to architect system platforms as building blocks for a baseline infrastructure [STV95]. These frameworks include generalized harmonized guidelines to architect application services, precise guidelines to use existing basic services, and guidelines to architect general or dedicated tools [Vos96, SV95].
In the following section we want to introduce the service concept and discuss its consequences.

SERVICES

One goal of this paper is the proposal for an approach of designing and developing of both software and hardware together, based on a functional and temporal specification. A suitable solution to ensure time requirements is the approach of building system-independent distributed operational units for a given task. These units are called services, which interact by the exchange of orders and results. With a given communication platform it is possible to place the services on different hardware systems independent of the specification [STV95a].

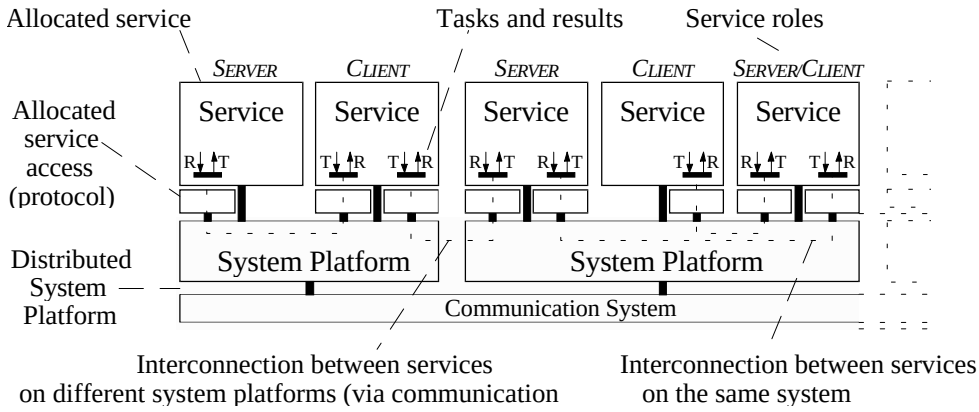

Fig. 1: *Physical Structure of a Service-Oriented System*

Services provide re-usable standardized building blocks which can be activated, controlled and interconnected in order to construct the application based on a service-oriented design scheme. Three basic types of services can be distinguished:

- The system services are needed for communication and process scheduling.
- The basic services are providing a layer for man-machine-interaction, for measurement and control and a database system for persistent data.
- The third type of service is application dependent, which means, that these services are built for a special solution, whereas the other ones have only been configured for a given task.

Building blocks of each service, so-called service activities (object or agent would be equivalent), represent a virtual instance which is configured within a service. Service activity access points and outputs can be attached to the service activities. The function of a service activity is executed by processing the relevant access information according to the service prescription. The result is available at a defined output point.
Another result of this approach is that changing time constraints during the life-cycle of the system can often easily be solved by changing the distribution of the services with or without adding new hardware components. The part of the automation system with rigorous real-time restrictions is mainly measurement, control and regulation. For these functionalities the presented design approach is able to find the appropriate hardware for the given specification based on integer optimization algorithms. The next sections give an overview over different application-independent services which we find necessary and sufficient to construct.

System services

Services must be interfaceable with each other via communication media. Their logical functionality must allow the allocation of prescriptions, service accesses to the configured services via service access protocols, the interconnection between the allocated services, the control of the services and the execution of the services which includes the intercommunication between the services and the synchronization of the service execution. Information processing units which meet these general requirements are called *system platforms* or system service. They are the backbones of every service-oriented system. System platforms are interconnected via communication systems for a distributed approach in system design. The goal is to hide the physical structure of the total system. The system service provides the basic functionality to build client-server applications, e.g. using distributed objects.

Man machine services

For man-machine-interactions a highly configurable, easy to use service is necessary to prevent the developer from writing special solutions for every task [BSV195, BSV295, BSV395]. These service offers an application programming interface independent from the underlying graphics hardware and screen resolution. The main task of any man machine service (MMS) is to inform a human user of a system's state and to enable modifications of this state. Due to the fact that man can perceive and handle information fastest in a visual way, this is the best channel to inform about complex system states. The optical channel is also a good choice to support human interaction. This is achieved in feeding back the user's actions. A MMS has to provide two basic services:

- the visualization of structured information
- the supported modification of structured information.

Based upon these services any communication between user and machine can be realized. This section presents the concept of a man machine service, usable as a server in a heterogeneous network of computers.

The basic idea of the service is the introduction of *symbols* as state-pictures of structured objects of an application, e.g. process variables of a control unit. The user can define symbols very flexible with an interactive tool or by using the programming application interface of the service. The defined symbols are stored in a configuration database for usage within the application. Symbols can be defined hierarchically, i.e. a symbol can contain other symbols or base symbols. Changing a value of a data type connected to a symbol leads to a different graphical representation. Changing the graphical representation (e.g. the user moves a symbol interactively) leads to a different data type value. The relations between data type values and the resulting images can be defined. This relation is either continuous, where a linear or logarithmic functions is provided, or discrete. All symbols are arranged and positioned in *planes*. Each plane defines a unit of measurement respectively a scale. When assigning a plane to one or multiple windows, the user is able to scale it.
This very simple approach is mighty enough to build high-level graphical user interface: Normal symbols are used to visualize a big amount of user defined data types. The *presentation object* is introduced to offer the developer the facility to group symbols together and to create images of complex data type with a special semantic. There are different types of presentation objects predefined:

- A *picture* is a set of symbols as an image of a set of objects of the application. There are no restrictions concerning the object types. The picture is the basis for all other presentation objects.
- A *menu* is an image for a variable of an enumeration type, each button shows a selectable value. Nearly any kind of symbol can be used as a button.
- A *mask* is an image for an object or a structure of an application: Modifiable components of the object can be changed by the manipulation of the corresponding symbols (sliders, buttons, textfields, ...
- A *table* is an image of an array of objects or structures.

Presentation objects can be build automatically by the service if the type of the corresponding object is known at runtime. Because every presentation object is derived from the same common class, they share the same (small) set of operations. The presentation objects themselves are used to build higher level objects like text editors, hierarchical graphs and help systems. These can be interpreted as pictures with special semantics and a predefined behavior.

Presentations objects itself are useful for the manipulation and visualization of objects. But to allow interactively modifiable and dynamically changeable user interfaces, a powerful mechanism for event recognition and execution is created: A *binding* is an operation, defined on a presentation object. The execution of the operation is triggered by one or more events on this object. The main ideas behind are:

- Several internal operations of the man machine service can be bound to events, so that typical interactions can be created by the interactive GUI-editor without writing any line of code.
- Presentation objects can be bound together to create hierarchical menus, masks and tables.
- User defined operations can be bound to events to create callback functions. An application is able to catch an event using this technique. A very common use of bindings is the connection of the measurement, actuation and control services (MAC) to the MMS. The MAC is controlled or triggered by user actions without the need of an intervention of an application service.

Presentation objects with a predefined behavior on events are implemented using bindings. Objects of a higher level, which are using presentation objects such as pictures, are supplying their components with task-specific binding functions to have control over the event responding.

Persistent object and database service

Every of the above mentioned services needs to store its configuration or internal states on a permanent memory even if the host of the service does not contain any harddisc or similar storage. The solution for this problem was the use of a real-time database service, extended by a layer for persistent objects and persistent relations between such kind of objects. The goal of this layer is a "natural" embedding into a given object-oriented programming language (here C++). The intention is to hide most of the additional functionality of the database from the programmer by applying a clear object-oriented design. Persistent objects are usable like any other non-persistent object [VB196, VB296, VBM96, Ste92, SKW92].

Measurement and control services

In the area of automation and control the definition of the above mentioned basic building blocks of a service is surveyable owing to the restricted application field [PR94,EP95]. General measurement and actuation agents, configured along the users needs, can be combined with control transfer functions and specifications to achieve a complete complex automation system. The required functionality for a service supporting control and automation applications will now be discussed in detail.

The basic control services allow to interconnect technical plants to be automated and computer-based systems for automation by measurement (data acquisition), actuation, and control of physical variables (MAC-services).

They comprise system platforms with allocated MAC service prescriptions (soft- or hardwired) extended by converters, signal conditioning devices, actuators, sensors. The actual configuration of an MAC device depends on the functions the services should perform, e.g. a MAC device based on a micro controller can serve only for distributed measurements of analogue physical variables whereas a MAC service based on a more powerful processor can serve for measurement, actuation and control at the same time.

The functionality of the MAC services knows six functions which serve:

- to configure analogue in, digital in, analogue out, digital out, control, and surveillance agents
- to define access points with related access protocols for created MAC agents
- to initialize created MAC agents
- to interconnect access points of MAC agents with access points of other MAC and non-MAC agents
- to control agents by beginning, suspending, resuming, and ending their operation
- to place explicitly orders to agents to be carried out and to commands get results like get measured or put control values

Control agents are composed by associated measurement and actuation agents which are interconnected by building blocks with standard controllers or arbitrary transfer functions. The operations of control services are defined as a composite agent.

All agent functions can be used as programmable interfaces within an implementation environment. The explicit programming of the configuration, including access points and corresponding protocols of the interconnection and initialization of the MAC agents is tedious. Therefore tools which allow man/machine dialogue oriented configuration, test and simulation of MAC agents by a comfortable GUI (graphical user interface) do accompany all MAC services.

The use of services as a more powerful alternative to software libraries does not avoid difficulties with real-time constraints and performance limitations of a chosen hardware platform. Therefore the above mentioned configuration tools support schedulability analysis. Often the required computer performance can only be achieved by a multi-processor or distributed architecture, especially i.e. in the area of robotics. This extends a simple schedulability analysis to an allocation problem.

Since our approach as presented above offers the possibility to predict the systems performance, distribution and allocation in consideration of real-time restrictions can be done before run-time [Ram90]. Therefore execution and communication times are known by a special database. In order to do the task allocation in respect of a positive schedulability analysis, an optimum criteria is needed. We choose costs as optimum, leading to the extension of the database by cost information.

Based on the above mentioned data-sets, the task allocation is to be executed. An integer optimization algorithm as known from the field of operation research has to allocate the tasks to processors within the distributed architecture. An often used heuristics is to map hierarchy and structure of the specification into the hardware architecture similar to analogue computers, but since the allocation problem is completely given by specification and database, there is no need to use heuristics. The simplest integer algorithm is total enumeration, but the problem is np-hard, a more sophisticated algorithm is needed. Since a large sub-set of all existing allocations does not fulfill the real-time constraints given in the specification, the branch-and-bound algorithm is the ideal approach for this problem [BG93]. In each ramification the allocation of one task is decided. If the maximal load of a processor is already reached before all tasks are allocated, it is no more necessary to follow this branch and its allocations. So time for the allocation process can be drastically reduced.

APPLICATION EXAMPLE: PRODUCTION CELL

The explain design concept was successfully used in a control application of a production cell [VH96]. This cell is subject of a specification methods survey at the FZI Karlsruhe [LL95]. It is equipped with two conveyor belts, a traveling crane, a two-armed robot, an elevating rotary table and a press. Metal plates are put on the feed belt which conveys them to the table. The latter brings the blanks in the right position to be picked up by the robot. The robot handles the plates between table, press and deposit belt. To increase the utilization of the press, the robot is operating with two arms, one for loading, the other one for unloading the press where the blanks are forged. The specification of the control system, including safety properties (no collisions), timing requirements, liveness, efficiency (maximal throughput), and a graphical visualization with different views could be realized and fully tested on the plant only by using the three predefined basic services for control, visualization and data storage.

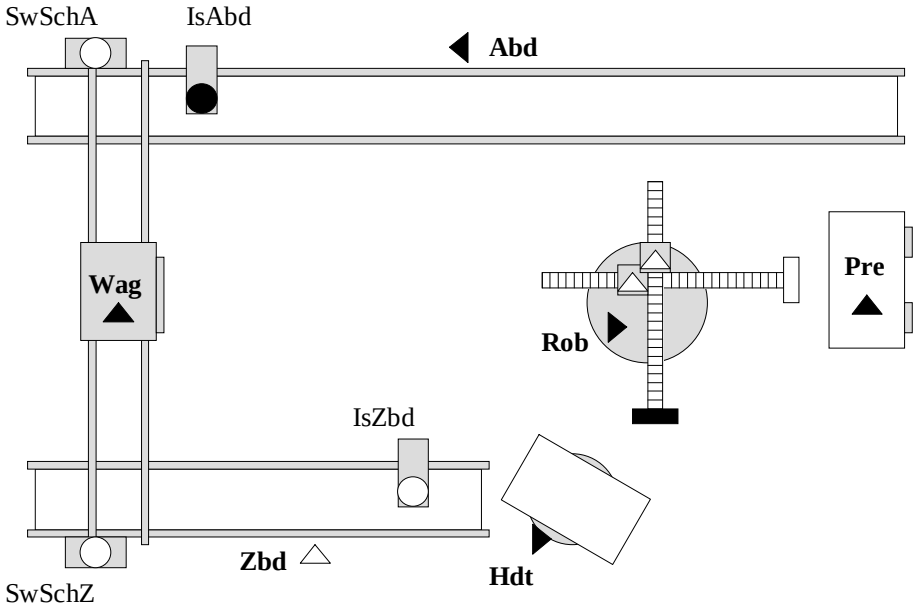

Fig. 2: *Model of the production cell*

SUMMARY

The presented service-oriented concept offers the chance to construct very complex distributed control systems within a short time frame by a hardware-independent specification which contains only the functional information plus real-time restrictions. This specification can be elaborated by the use of comfortable graphical editors without knowledge of any implementation in procedural language. The underlying hardware architecture can be replaced without reprogramming, the re-allocation of tasks is done automatically, i.e. a prototype within a control application can be tested on a PC-cluster, whereas the final implementation will be "lean" on a set of microcontroller modules. It is easy to expand an existing system with or without adding new hardware components in case of specification changes due to the open system architecture. Since the automation and visualization of the production cell was realized within one week, the described approach would be an ideal platform for solutions in "Rapid Prototyping".

ACKNOWLEDGEMENT

This paper is based on research done at the Institute for Microcomputers and Automation, Prof. Schweizer and Prof. Brinkschulte, with partial support of the German Research Foundation (Graduiertenkolleg "Controllability of Complex Systems", DFG Vo 287/5-2). The case study "production cell" was realized in co-operation with the Computer Science Research Center FZI-Karlsruhe.

REFERENCES

[BG93] I.M. Bomze, W. Grossmann : "*Optimisation : theory and algorithms*", BI 93, pp. 431-449

[BSV195] U. Brinkschulte, Marios Siormanolakis, Holger Vogelsang, *"Man Machine Service"*, Workshop Proceedings, KEOOA 95, Knowledge Engineering and Object Oriented Automation Workshop, Strasbourg, May 1995

[BSV295] U. Brinkschulte, Marios Siormanolakis, Holger Vogelsang, *"Visualization and Manipulation of Structured Information"*, Conference Proceedings, Visual'96, International Conference on Visual Information Systems, Melbourne, February 1996

[BSV395] U. Brinkschulte, Marios Siormanolakis, Holger Vogelsang, *"Graphical User Interfaces in Heteregeneous Systems"*, Conference Proceedings, EI'96, International Symposium on Electronic Engineering: Science and Technology, San Jose, February 1996

[EP95] C. Ebert, E. Pereira : "*Design methods for real-time software systems*", Journal ATP 4/95, pp. 12-22.

[Lav91] J.Z. Lavi et.al.: "*Formal Establishment of Computer Based Systems Engineering Urged*", IEEE Computer, 24 (3), pp. 105-107, 1991

[LL95] C. Lewerentz, T. Lindner (Eds.): "*Formal Development of Reactive Systems - Case Study Production Cell*", LNCS 891, Springer, 1995

[Oli93] D.W. Oliver: "*A Tailorable Process Model for CBSE*", Draft, GE Research & Development Centre, February 1993

[PR94] C. Pereira, Th. Rathke : "*Objektorientierte Entwicklung von Echtzeitsystemen in der Automatisierungstechnik*", Proc. 39, Int. Wissensch. Kolloquium, Sep. 94, Illmenau

[Ram90] K. Ramamritham: "*Allocation and Scheduling of Complex Periodic Tasks*", 10th IEEE-conference on Distributed Computing Systems, 1990, pp. 108-115

[SKW92] Vivek Singhal, Sheedal V. Kakkad, Paul R. Wilson, *"Texas: An Efficient, Portable Persistent Store"*, Proc. of the Fifth International Workshop on Persistent Object Systems, San Miniato, Italy, September 1992

[Ste92] Al Stevens, *"Persistent Objects in C++"*, Dr. Dobb's Journal, December 1992

[STV95] G. Schweizer, B. Thomé, M. Voss: "*A Systems Theory Based Approach to Systems Engineering of Computer Based Systems and its Consequences*", In: Melhart, B., Rozenblit, J. (Eds.): 1995 Intern. Symposium and Workshop on systems Engineering of Computer Based Systems, 1995

[STV95a] G. Schweizer, B. Thomé, M. Voss: "*A Systems Engineering Approach for Computer Based Systems*", In: Systems Engineering in the Global Market Place. Proceedings of the 5th annual intern. Meeting of NCOSE, 1995

[SV95] G. Schweizer, M. Voss: "*Systems Engineering and Infrastructures for Open Computer Based Systems*", To appear in: Pichler, F. (Ed.): Computer Aided Systems Technology - EUROCAST '95, LNCS, Springer, 1995

[VB196] Holger Vogelsang, Uwe Brinkschulte, *"Persistent Objects In A Relational Database"*, submitted paper to ECOOP'96

[VB296] Holger Vogelsang, Uwe Brinkschulte, *"Relational Databases for Object-Oriented Applications"*, submitted paper to BNCOD-14

[VBM96] Holger Vogelsang, Uwe Brinkschulte, Marios Siormanolakis, *"Archiving System States by Persistent Objects"*, in: Proceedings of the IEEE conference on ECBS'96, Friedrichshafen, Germany

[VH96] M. Voss, O. Hammerschmidt: „*A Case Study in CBS-Development - Production Cell*“, Proceedings of ECBS'96, Friedrichshafen

[Vos96] M. Voss: „*Systems Theories and Architectures for ECBS*“, Proceedings of ECBS'96, Friedrichshafen

[Whi93] S. White et.al.: "*Systems Engineering of Computer-Based Systems*", IEEE Computer, 26 (11), pp. 54-65, 1993