Open Real-Time Interfaces for Monitoring Applications within NC-Control Systems

Manfred Weck, Andreas Kahmen
Laboratory for Machine Tools and Production Engineering (WZL)
RWTH Aachen University of Technology
A.Kahmen@wzl.rwth-aachen.de

Abstract
Process monitoring is the key to an increasing degree of automation and consequently to an increasing productivity in manufacturing. The realization of process monitoring functionality demands an extension of the control system. The prerequisite for these extensions are open interfaces in the NC-kernel. Nowadays controls with open NC-kernel interfaces are available on the market. However these interfaces are vendor specific solution that do not allow the reuse of monitoring software in different controls. To overcome these limitations a vendor neutral open real-time interface for the integration of process monitoring functionality into the NC-kernel is presented in this paper. Additionally two implementations of the real-time interface for different target systems are described.

1 Introduction
Machine tools with rising functionality are demanded by the market these days. Innovative concepts and rationalization measures are supposed to lower the costs for all participants in the production process. Thus new control functionality is necessary to gain more flexibility and quality in production. Furthermore users wish to be independent of single suppliers. These demands cannot be fulfilled solely by control vendors. It should rather be the task of technology suppliers to offer products that increase the functionality of controls. For that reason, open control systems are getting more and more popular lately [1,2].

In terms of open controls, different levels of openness have to be distinguished. They can be subdivided into “outer openness” and “inner openness” categories. Controls of the category outer openness aim at a unification and standardization of external control interfaces, like a harmonization of the NC-programming, an extendible Man Machine Interface (MMI) or a standardized interface for drives. Controls of inner openness allow access to control internal data and methods as well as the integration of additional software into the control. Nowadays many controls offer open interfaces in the MMI. This openness is mostly achieved with a PC-based system architecture of the MMI. A few control vendors even took a further step. They provide libraries in the NC-kernel of their control that enable third parties to integrate software at certain points of the control internal data flow. Data access or even the manipulation of data is thereby possible.

Up to now all open control interfaces are vendor specific. Hence porting third party software modules is impossible.

This paper will present a solution for the mentioned limitations. For this purpose the “Application Interface”, which is a new interface for integrating third party software modules into various controls, is introduced. The Application Interface focuses on integrating applications in the field of process monitoring, but it can also be applied to less time critical applications.

2 Architectures of process monitoring systems
Nowadays process monitoring applications are state of the art in industrial production. They are used to ensure safe processes at any time. Mass production is the main field of process monitoring applications. The need for monitoring functionality depends on the degree of automation. Monitoring of tool breakage and tool wear for drilling, milling, turning and threading are the most prevalent applications. Nowadays external sensors are the main source for the acquisition of input data for monitoring applications. However, control internal signals, such as the torque values of digital drives are used more and more.

Today the PLC is the most important link between the control of a machine tool and external components of process monitoring applications. For the data exchange between time-critical applications and the control, fieldbusses, like CAN, Profibus or Interbus, are the preferred communication media. However, for less time-critical applications, that mostly communicate with the control via the MMI, usually ethernet is chosen. Digital and analogue I/Os are still widely used for hardware interfaces. Furthermore, vendor specific solutions in terms of communication media as well as protocols that are developed for specific demands are still applied in certain monitoring applications.
In figure 1 the architecture of a sensor based application for tool monitoring is illustrated. The current cutting forces, which are measured with an external sensor, are preprocessed in an external analyzing unit. Here the cutting forces are also compared to their upper limits. If they exceed these limits, an error message is sent to the PLC where a suitable reaction is triggered. Due to the specific software interfaces that are used for the communication between the PLC and the external components, porting the monitoring application to different controls or even machines with different PLC programs is impossible without changing the interfaces. However, the adaptation of interfaces always implies additional costs as well as the risk of failures.

Due to the specific software interfaces that are used for the communication between the PLC and the external components, porting the monitoring application to different controls or even machines with different PLC programs is impossible without changing the interfaces. However, the adaptation of interfaces always implies additional costs as well as the risk of failures.

![Figure 1: Architecture of an application for tool monitoring based on external sensors](image1)

The advantage of systems using external sensors is that they can be optimized for specific requirements of monitoring tasks. On the other hand, extensive additional components as well as costly adaptations and tuning are necessary. For this reason a trend towards the integration of process monitoring applications into the control can be noticed [3, 4]. Through the integration of monitoring applications, control internal information, such as the current of drives or position values, can be used as data origin for monitoring algorithms. Because of the integration of process monitoring applications, additional hardware for external sensors, preprocessing and analyzing units as well as their electrical connections can be reduced. Figure 2 illustrates the architecture of a control integrated process monitoring system. A data acquisition module filters and amplifies the base signals, which are provided by external sensors or are acquired from the drives. Disturbances in the signals are then eliminated and process data is prepared to be evaluated in an analyzing module. The analyzing module hands over information about disturbances to a reaction module which triggers an appropriate action in the NC-kernel.

![Figure 2: Architecture of a control integrated process monitoring system](image2)

As pointed out in the introduction, controls with an open NC-kernel are already available on the market. This is the prerequisite for the integration of monitoring functionality into the control. However, the interfaces for the integration are still vendor specific. For that reason the requirement for an adaptation of interfaces remains. Both examples have demonstrated that porting process monitoring functionality without adaptation of interfaces is so far impossible. This can only be achieved with the development of vendor neutral open interfaces.

### 3 Requirements on a vendor neutral open real-time interface for process monitoring

Process monitoring functionality requires different levels of real-time performance. For example, monitoring tool wear is less time critical than monitoring tool breakage. The demanded performance of a real-time interface for process monitoring has to meet the requirements of the most critical application. Furthermore it has to be ensured that the interface can be implemented on different hardware platforms (e.g. vendor specific control platforms, DSPs, micro-controllers, etc.). The interface has to be scalable in functionality because not every control platform provides the same features, and finally, the integration of the interface must not affect the stability of the control system.

### 4 Concept of the Application Interface

For a simple integration of process monitoring systems into controls, the monitoring functionality should be split into small modular units such as preprocessing, analyzing or reaction determination units. As illustrated in figure 2, each unit should be encapsulated
in a separate software module. These software modules are called Process Modules. The interfaces of the Process Modules have to follow the specification of the Application Interface. Through this, a seamless integration into the control should be feasible.

During the design of the Application Interface, existing standards or de facto standards were used wherever they were applicable. This approach should also apply to the implementation of the interface in different controls.

Figure 3 illustrates the concept of the Application Interface and the integration of Process Modules. The Application Interface is a uniform interface for data access in the NC-kernel or on external hardware platforms. It comprises a thin communication layer that hides the vendor specific communication mechanisms inside the control. The communication connection between the Application Interface and the control internal data structure is realized individually, depending on communication protocols (e.g. OSACA, CORBA, vendor specific protocols, etc.) and communication media (e.g. ethernet, fieldbus, etc.) that are either already available in the control or that can be adapted easily. Through this, the real-time performance of the interface can also be scaled by choosing an appropriate communication protocol and communication media for a specific control platform.

**Figure 3: Concept of the Application Interface and the integration of process monitoring applications**

To meet the mentioned requirements concerning the real-time performance and the ability to be integrated on various hardware platforms, the Application Interface has been designed as a low level interface using C as programming language. C compilers are provided for almost any hardware platform. Furthermore a C-interface offers efficient methods for data exchange, which fulfill the requirements concerning the real-time performance.

In the following, the concept of the Application Interface, which is composed of the three parts "Communication System", "Reference Architecture" and "Process Module Manager", will be detailed.

**Communication System:**
It is the task of the Communication System to offer mechanisms for the data exchange between different applications and the control. In terms of data transfer the Communication System is the core of the Application Interface. The communication mechanisms have a significant influence on the performance of the interface. That is why the realization of efficient mechanisms for data exchange is essential.

A high flexibility of the Application Interface is achieved by defining a small set of methods that can access and handle different communication objects. This small set of methods covers the basic functionality of reading and writing data values as well as invoking and receiving function calls. Handling different communication objects via one method implies an identification of the objects, which is needed for their distribution in the interface. For this reason all communication objects used by the Process Modules for exchanging data as well as for invoking and receiving function calls have to be registered in the Application Interface during the start up phase of the monitoring system. The descriptions of the objects, which are used for the registration, have to follow the specification in the reference architecture which will be introduced later.

After the registration has been finished, the data exchange between Process Modules and the control system can be handled with simple get and set services for reading and writing data values. For invoking function calls in the control system, a method for requesting a service execution can be used.

For receiving function calls from the control system throughout normal operation of the monitoring system, the function pointers of the corresponding methods in the Process Modules have to be registered in the Application Interface during the start up phase.

The communication via the Application Interface follows the client-server-principle. This means that data or executable functions are offered by server objects to clients anywhere in the control system.

**Reference Architecture:**
The interoperability of Process Modules requires a clear specification of the server communication objects offered by the Process Modules and the Application Interface in addition to the access functions described in the previous paragraphs. This specification has to include non-ambiguous object names and the description of the objects’ attributes. The Reference...
Architecture has to be understood as an open specification that can be extended whenever new Process Modules are developed or existing Process Modules are upgraded with new functionality. It is important, to make sure that the specification is always consistent and compatible to former versions.

In the Reference Architecture, the communication objects are subdivided into objects for data exchange and objects for handling of function calls. The specification of objects for data exchange has to provide the description of the objects’ attributes. These attributes are the object name, a verbal description of the object’s purpose, the identification of the object's internal use, the description of its data type, the object's access rights, the time layer that contains information about the period in which the data can change, the scaling which includes the object's measurement unit and the range specifying the limits of the object's values.

For the communication objects that handle function calls, the logical correlation between the states of the respective Process Module and the functions that are allowed to be called in certain states has to be part of the specification, too. To establish such a correlation, the functions that can be called in a Process Module as well as the states of a module (e.g. READY, ACTIVE, etc.) are organized in so called function groups. Each function group represents an image of a certain operation mode of the Process Module (e.g. the start up mode or the normal operation mode). Therefore in the specification of the communication objects for handling function calls, each function group has to be described with all states and corresponding methods. The methods can also be called with passing arguments as well as return arguments. These arguments have to be specified following the description of the communication objects for data exchange.

The structure of the presented Reference Architecture is closely related to the reference architecture developed in the European Project OSACA (Open System Architecture for Controls within Automation Systems) [5]. Therefore the basic elements of the OSACA reference architecture, like the specification of data types and the formal description templates, have been adopted.

Process Module Manager:
The Process Module Manager is a part of the Application Interface that offers functionality for the coordination of the process monitoring system. Its functionality comprises the execution of the system start up, including configuration and parameterization, as well as the system coordination during normal operation.

By being part of the Application Interface, the Process Module Manager is also part of the control platform. Not every control platform offers all features that are needed for an implementation of the full functionality that a Process Module Manager could provide. That is why conformance classes have been introduced that define the scale of functionality offered by a Process Module Manager for a certain control platform. For the definition of conformance classes, the functionality is subdivided into different levels. At the moment three different levels are defined:

- an elementary level which comprises the basic mechanisms to establish communication connections,
- a second level that offers mechanisms for a dynamic configuration of the monitoring system in addition to the functionality of the elementary level and
- a third level that additionally offers functionality for a full parameterization of the system and for monitoring the output of the Process Modules.

Further conformance classes can be defined at a later time.

For a successful start up of the monitoring system, a certain order has to be kept for the execution of methods that build up communication connections. In the first start up phase, the server communication objects for the data exchange and the objects for the reception of function calls in the Process Modules have to be registered. When all Process Modules have finished the first phase, the Process Module Manager can initiate the registration of client objects. After all Process Modules have returned from the second registration phase, the start up of the monitoring system has been finished and the Process Modules can be switched to normal operation mode.

The mechanisms for building up communication connections are part of the functionality defined in the elementary conformance class. A Process Module Manager based on conformance class two, additionally offers mechanisms for a dynamic configuration of the monitoring system and its communication connections. For a dynamic configuration, the set up of the monitoring system has to be specified in a configuration file that is parsed during start up. This specification of the process monitoring system has to include information about the Process Modules that have to be started and the client communication objects that have to be registered.

A Process Module Manager implemented based on conformance class three provides mechanisms for a parameterization of the monitoring system in addition...
to the functionality defined in the conformance classes one and two. The parameter values can contain limits for data generated in Process Modules, priority values for the Process Modules as well as indications for strategies that are used to shut down the process in case of a disturbance.

These parameters are needed as a basis for mechanisms that ensure safety operations of the process monitoring system. One of these mechanisms analyzes the data generated in the Process Modules. If a trend towards the limit of a certain data value can be observed, the machine tool operator is notified. If this trend moves too fast for a notification of the operator, the Process Module Manager initiates a controlled shut down of the process. The strategy of the shutdown has to be included in one of the parameterization values. It is important to choose an adequate strategy in order to avoid endangering the operator or damaging the machine tool or the workpiece (e.g. in case of a tool break while drilling, a different strategy has to be chosen than if a break occurs while finishing a free form surface).

The priority values for Process Modules are needed to prevent deadlocks if more than one Process Module attempt to set the same data value for the NC-kernel (e.g. if the feedrate has to be reduced because of a spindle overload detected by one module, but at the same time another module attempts to rise the feedrate to avoid chattering).

Similarly to the system configuration the parameter values are specified in a parameterization file that is parsed during the start up of the monitoring system.

5 Implementations of the Application Interface

This section presents the implementations of the Application Interface in a Siemens control with open NC-kernel (Sinumerik 840D) and in an OSACA conform control, the WZL-NC developed at WZL.

In figure 4 the Application Interface for the Sinumerik 840D is illustrated. As stated before, the Process Module Manager is part of the Sinumerik 840D control platform. It organizes the start up of the process monitoring system according to the start up of the Sinumerik control system. The Application Interface builds up the communication connections between the Process Modules and the control internal data. In this example the implementation of the Process Module Manager follows the elementary conformance class. Therefore the communication connections are coded in the Process Modules which restricts the flexibility of a dynamic connection management.

The implementation of the Process Modules follows the specification of the Application Interface. Consequently the Process Modules can be ported on any control platform that offers an Application Interface.

For an OSACA control, the Process Module Manager can be provided as Architecture Object (AO). It is again part of the control software, because for OSACA controls, the control functionality is encapsulated in discrete AOs. The encapsulation of the Process Module Manager’s functionality offers the option of porting it to different OSACA conform controls.

The WZL-NC offers all features for an implementation of the full functionality that a Process Module Manager can provide (conformance class three). That is why a dynamic connection management and an individual parameterization has been realized for this control. Furthermore security measures for the monitoring of data generated in the Process Modules, the prevention of deadlocks and the execution of controlled shutdowns have been implemented to
ensure a safe operation of the process monitoring system. By linking the Process Modules with the Application Interface, the Process Modules are upgraded to be complete OSACA AOs. Through this they can be operated on any OSACA conform control platform. Figure 5 illustrates the implementation and the use of an Application Interface in an OSACA control.

## 6 Portability of Process Modules

Figure 6 illustrates the portability of Process Modules that are implemented following the specification of the Application Interface. Process Modules that are linked with an Application Interface for OSACA control platforms can be ported between different OSACA platforms without any change. In figure 6 the Data Acquisition Module can be integrated either on the OSACA control or on the OSACA platform in the MMC of the Siemens control. For other system modifications, such as porting a Process Module from the NC-kernel of a Sinumerik 840D to an OSACA control, an Application Interface for the control platform that the Process Module is supposed to be integrated on has to be linked to the Process Module. Nevertheless in both cases the implementation of the Process Modules does not have to be changed.

![Figure 6: Portability of Process Modules following the specification of the Application Interface](image)

## 7 Conclusion

These days, machine tools with rising productivity are demanded by the market. To reach this aim, the degree of automation has to be increased. For this reason, process monitoring functionality is essential.

The use of external sensors is the state of the art in acquiring information for process monitoring. Because of the availability of controls with open NC-kernel, which is the prerequisite for the integration of process monitoring functionality into the control, a trend towards control integrated solutions can be noticed. However, due to the vendor specific open interfaces, the software modules cannot be ported to different controls. Therefore vendor neutral open interfaces for the integration of third party software into the NC-kernel have to be developed. In this paper the Application Interface, a real-time interface for the integration of third party software modules, is introduced. It focuses on application in the field of process monitoring, but it can also be applied to less time critical applications. The Application Interface is composed of three parts: a Communication System that handles the data exchange, a Reference Architecture for the specification of communication objects and a Process Module Manager for the configuration and parameterization of the monitoring system.

Implementations of the Application Interface for a control with vendor specific open NC-kernel interfaces and for an OSACA conform control have been described. Finally, the portability of monitoring modules has been explained by means of exchanging Process Modules between different control platforms.

## 8 References