

# Application of ADAMS/SDK as part of a comprehensive machine tool simulation system

B. Denkena, S. Rehling, K. Tracht, Institute of Production Engineering and Machine Tools, University of Hannover

Keywords: ADAMS/SDK, machine tool simulation

## Abstract

As part of a simulation system for the simulation of manufacturing processes on metal cutting machine tools, the dynamical behaviour of the machine tool structure during the manufacturing process has to be simulated.

The simulation of the mechatronical system 'machine tool' demands the consideration of all system components, which interact during the manufacturing process. For the simulation, these system components can be partitioned into three system classes: The system class 'machine dynamic', which simulates the dynamic behaviour of the machine tool structure; the system class 'control and drives', which simulates the numerical control, the programmable logic controller and the drives; and the system class 'cutting process', which simulates the forces which result from the interaction between the cutting tool and the workpiece.

The simulation system is designed as an autonomous, modular application with one simulation module for each system class. As simulation tool for the simulation model 'machine dynamic', ADAMS is used. The implementation of ADAMS in third party applications is well known by the example of CAD Systems (e.g. Unigraphics), which implement ADAMS/SDK to provide dynamic simulation functionality. ADAMS/SDK allows the development of autonomous applications, which are independent of the common ADAMS environment. The main advantage is, that the ADAMS/SDK library provides full control of the simulation at runtime. Furthermore, it offers the possibility to couple the ADAMS solver with external calculation modules via fast in-process interfaces.

This paper describes the application specific requirements on ADAMS/SDK, which are made by the simulation system. Therefore a brief overview of the functionality of the system is given firstly. Further on the experiences with the ADAMS/SDK application-programming interface (API) is described, which were made during the development phase. In this context the possibilities to apply externally calculated forces on the model and the possibilities to query simulation results during runtime are discussed. Positive as well as negative aspects of the API are highlighted.

## 1. Introduction

Nowadays, in metal cutting manufacturing the exact machining result cannot be determined without experiments on the machine. This results in expensive try-out in mass production and subsequent machining in batch production.

With the target of reducing costs and throughput time high efforts are made to develop methods and systems for the prediction of the real machining result.

The development of a system for computer-based simulation of manufacturing processes on metal cutting machine tools (lathes, mills) is the goal of a group of scientists at the Institute of Production Engineering and Machine Tools, University of Hannover. The main application of this system called CutS (shortcut for "Cutting Simulation") is to predict the resulting geometry of the machined workpiece by a comprehensive simulation of the whole manufacturing process. Based on a comparison of the simulation results to the CAD-data of the workpiece, the manufacturing inaccuracies can be determined and the NC-Code of the workpiece can be optimised.

The simulation of the mechatronical system 'machine tool' demands the consideration of all system components, which interact during the manufacturing process. For the simulation, these system components can be partitioned into three system classes: The system class 'machine dynamic', which simulates the dynamic behaviour of the machine tool structure; the system class 'control and drives', which simulates the numerical control, the programmable logic controller and the drives; and the system class 'cutting process', which simulates the forces which result from the interaction between the cutting tool and the workpiece.

This paper introduces the integration of ADAMS/SDK as a part of the simulation system CutS, bound into a module for the dynamical simulation of the machine tool structure, which represents the system class 'machine dynamic'. To give an overview about the integration and functionality of this module within the scope of CutS, the structure and functionality of the whole system CutS is explained firstly.

## 2. The Simulation System CutS

### 2.1. System Overview

CutS is designed as a modular simulation system, in which each of the different system components named above is implemented as a stand-alone simulation module. The core of the system is the so-called "CutS Kernel", which is responsible for the communication between the modules and the control of the simulation. The modules and the kernel communicate via a well-defined interface that implements the Microsoft (Distributed) Component Object Model (COM/DCOM) technology. The use of the COM/DCOM technology enables the system to run the simulation and the postprocessing on different computers in a network to distribute the processor load. The architecture of CutS is shown in figure 1.

The modular concept of the simulation system and the standardization of the interface allows to replace single modules and to extend the system. Due to this modularity the

system is an ideal research platform to study and analyse different simulation methods and modelling techniques.

## **2.2. Interaction between the simulation modules**

The three simulation modules 'machine dynamic', 'control and drives' and 'cutting process' are working together interdependently and are interconnected to form a closed loop. This is shown in figure 2. The single modules work as follows.

The input of the simulation module 'control and drives' consists of NC-data files of the workpiece which are generated by a CAM application and the simulated axis positions which are fed back from the simulation module machine dynamic. The NC-data is processed by an interpolator, which generates nominal values for the positions and velocities of the machine tool axes. Therefore a virtual NC kernel from Siemens is used which behaves exactly the same as the Sinumeric 840D NCU on the real machine. For the simulation of the machine tool's position control and drives also a realistic simulation model of the control circuit and drives is used. This model is parameterised by the real machine tool initialisation data. The output of the simulation module 'control and drives' are the forces/moments that are generated by the drives.

These forces/moments act on a multibody simulation model of the machine tool structure that is simulated by ADAMS. The model consists of a combination of rigid and flexible bodies. The calculated state of motion between the cutting tool and the workpiece is the input of the simulation module 'cutting process' and acts as the basis for the calculation of the cutting force and the material removal, which then results in the actual shape of the workpiece. The reaction of the cutting force is fed back to the module 'machine dynamic', finally the simulated positions of the machine tool axes are fed back to the simulation module 'control' and drives to close the loop.

## **2.3. Implementation**

Due to the integration of different simulation tools for the different simulation modules and because of higher system performance, the kernel as well as all simulation modules are implemented in C++. The modules include the function libraries and COM interfaces respectively of the concerning simulation tools. Thus it is possible to instantiate fast in-process interfaces and to minimise latencies caused by data transfer between the modules.

## **3. Simulation module machine dynamic**

In the following the characteristics, requirements and the implementation of the simulation module machine dynamic, which includes the ADAMS/SDK function library, are described in detail.

### **3.1. Requirements**

One of the main requirements of the simulation system CutS is an optimised simulation speed. For this reason the simulation module machine dynamic has to be optimised concerning the simulation performance.

Another requirement is to have the possibility to model significant elasticities of the machine tool structure (i.e. the cutting tool). Therefore ADAMS/Flex offers sufficient possibilities.

To realize the coupling and synchronisation to different simulation modules, full time control during the simulation has to be provided to the controlling application. This results from the position control of the machine tool, which has a constant time cycle (e.g. 1 ms). Thus it has to be assured that the solution of the module 'machine dynamic' is well known at these equally spaced discrete points in time. Furthermore there has to be a possibility to consider external calculated forces and parameters during the simulation to be able to process the inputs from another simulation module. ADAMS/SDK meets all these requirements through the solver API and the implementation of user-written subroutines.

### **3.2. Modelling, time variant Parameters**

The modelling work to build the machine tool model is done using ADAMS/View. The geometry of the structure components is imported from a CAD-system. The modal representations of all flexible bodies are calculated by ANSYS™ and then imported via ADAMS/Flex. Figure 3 shows as an example the model of a machine axis that is driven by a linear motor.

All non-linear modelled forces and transfer elements (e.g. friction, spring-damper elements) are defined by external forces and field elements and are named by a CutS internal naming convention. This allows the identification of axes and relating drive forces/moments by other modules. During the simulation, these field elements are calculated by user-written subroutines based on empirically determined characteristic curves. In an analogous manner the external forces are calculated by the corresponding user-written subroutine.

After the modelling process, the model is exported as ADAMS/Solver dataset file and can be transferred to the module 'machine dynamic'.

## **4. ADAMS/SDK Interface**

The module 'machine dynamic' is designed as a stand-alone Win32 application, which implements a rudimentary user interface for setting parameters and output status messages. Therefore the Microsoft Foundation Classes (MFC) are used. A wrapper class named *CAdamsModule* serves as container for the ADAMS/SDK API and manages the interaction between the ADAMS library and the main application (see figure 4).

### **4.1. User-written-subroutine DLL**

For the implementation of the described functionality different user-written subroutines are necessary. The ADAMS/SDK only accepts autonomous dynamic link libraries (DLLs) as container for user-written subroutines rather than function pointers to the corresponding routines. These DLLs are linked to the ADAMS/SDK library through the

API function call *InitUserSubs()*, so an autonomous DLL had to be programmed for the simulation module machine dynamic.

The user-written subroutine DLL which is used by the simulation module machine dynamic implements the subroutines *GFOSUB()*, *FIESUB()* and *VARSUB()* to manipulate the variable model parameters during the simulation. By the use of *GFOSUB()* all forces which are produced by the drives and the cutting process, as well as all non-linear modelled friction forces are calculated. *FIESUB()* sets the values for the non-linear modelled stiffness and damping parameters of the force transfer elements. The corresponding characteristic diagrams base on empirically determined values of the real machine. *VARSUB()* is used to update the changing mass and inertia properties of the workpiece, which significantly loses mass during the manufacturing process.

The actual values of the axes and tool positions are determined after each simulation time step using the helper function *SYSARY()* from the utility library and are transferred to the other simulation modules. Thus a closed implementation of the ADAMS/Solver with full simulation control has been established.

## 4.2. Performance

Due to the optimised integration of the ADAMS/SDK into the CutS environment, which accesses the solver database only when it is necessary, the simulation speed (including visualization and animation of the geometry) has been increased. A comparison between the simulation of a machine model in ADAMS/View and the same model using CutS was made. To simulate the drives in ADAMS/View as well as in CutS sinusoidal dummy forces were acting on each model. The simulation parameters were the same in both applications. The comparison showed that the CutS simulation was solved eleven times faster than the same simulation in ADAMS/View.

## 5. Problems and suggestions

The ADAMS/SDK has the limitation, that only autonomous DLLs are possible sources for user-written subroutines. In the present case, the frame application as well as the ADAMS/SDK library have to access the variables that affect the user-written subroutines during the simulation. This is only possible by applying a little programmatic trick. Inside the user-written subroutine DLL global variables are declared which serve as a data source for all values affecting the simulation. The DLL is then loaded by the frame application just before *InitUserSubs()* (the API function witch connects the solver to the user-written subroutine DLL) is called. This makes the relating variables visible in the scope of the frame application. Afterwards *InitUserSubs()* tries to load the DLL a second time, which is ignored by the system, since the DLL is already present in the process space of the frame application. This is shown in figure 4. Thus both the frame application and the ADAMS/SDK library access the same variables of the user-written subroutine DLL. Here the suggestion is to implement an interface that takes function pointers to the user-written subroutines. This would reduce the programming effort.

A second point worth mentioning is that there isn't any user-written subroutine that is called at the end of the simulation. In the present case, the simulation ends after each

time step of e.g. 1 ms and is then started again for the next time step. After each time step the simulation result has to be read out of the solver database, which is done via the helper utility *SYSARY()*. *SYSARY()* allows to access the values of various attributes (e.g. position and orientation) of solver objects at runtime. The problem is that *SYSARY()* can be called out of a user-written subroutine only. Otherwise it has the wrong calling context. A work around is to implement the calls to *SYSARY()* into one of the three used subroutines, but these are called after each iteration step, so there are probably too many unnecessary database accesses. A subroutine which is called after the end of the simulation would help.

## **6. Conclusion**

A comprehensive simulation system which includes different interdependent simulation tools has to manage interaction and communication between the connected modules. This communication causes a serious bottleneck and slows down the whole system. With the use of the ADAMS/SDK library fast inter-process data interfaces could be established and the lag time which results from data transfers between single modules has been effectively minimized.

ADAMS/SDK offers a fast calculating and easy accessible simulation library for the application CutS. The economic efficiency of a stand-alone implementation also has to be considered. CutS is a system for which a machine model has to be modelled only once and is then reused as calculation basis for the manufacturing simulation of multiple varying workpieces. This application causes the stand-alone implementation to be economically efficient. The performance gain only could not legitimate such a model.

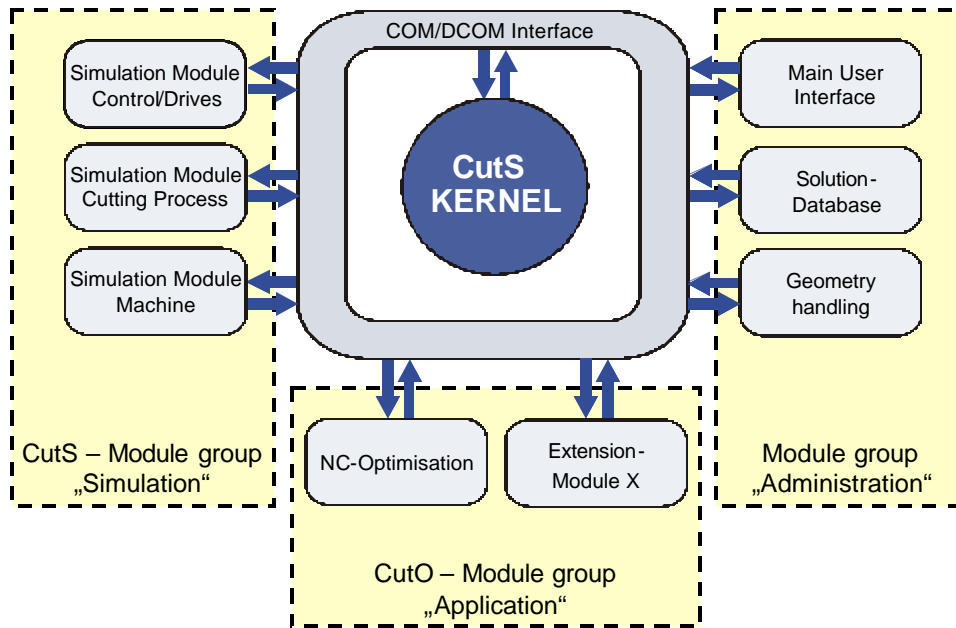


Figure 1: Structure of the simulation system CutS

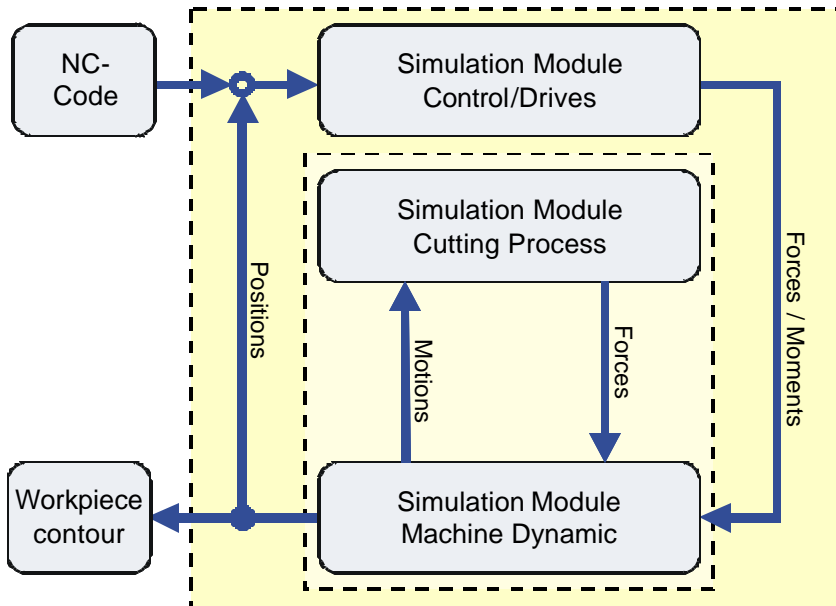


Figure 2: Interconnected simulation modules form a closed loop

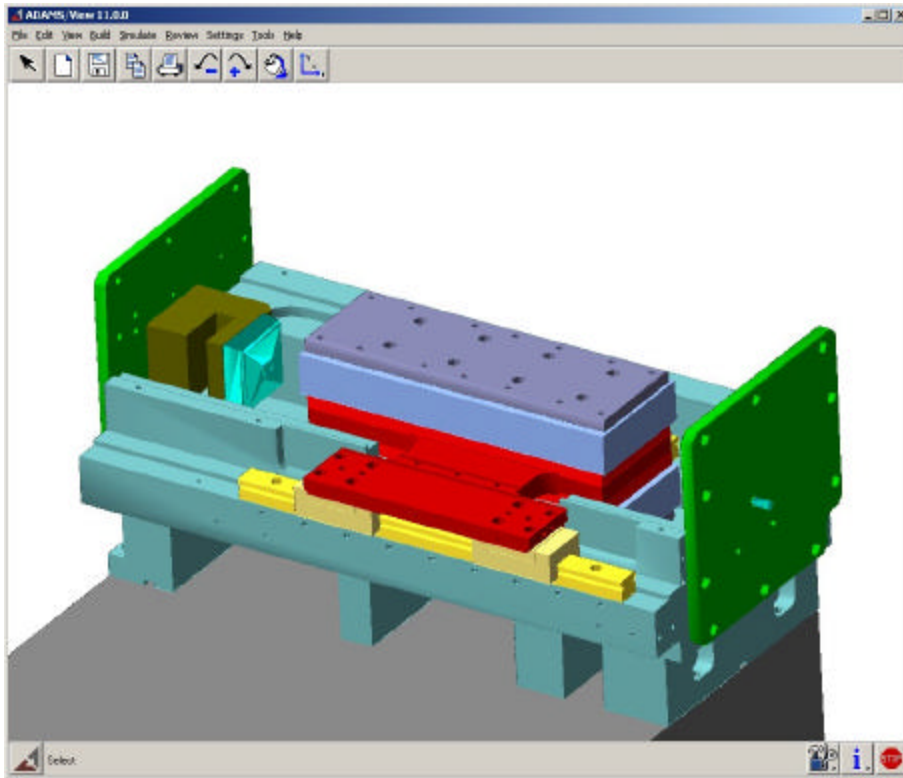


Figure 3: Model of machine axis with linear motor

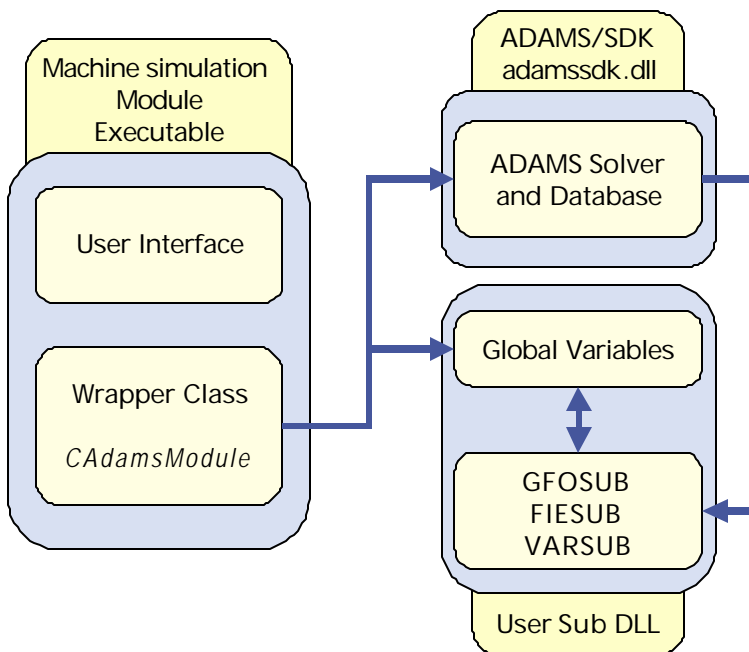


Figure 4: Implementation of the user-written subroutine DLL