# A system architecture for computer and robot assisted craniofacial surgery

by A. Pernozzoli[*], C. Burghart[Ψ], J. Brief[*], S. Haßfeld[*], J. Raczkowsky[Ψ], J. Mühling[*], U. Rembold[Ψ], H. Wörn[Ψ]

*We present the concept of a system architecture for the computer aided craniofacial surgery. The architecture is based on CORBA, an industrial standard specification for the development of distributed applications. Our concept includes a fundamental behaviour oriented communication model and some fundamental software safety considerations. We've developed a standard library for the integration of new services and devices into our system architecture. It increases development time noticeably. We tested the performance and usability of our concept on an evaluation set up consisting of a surgery robot system, an infrared navigation system, a force-torque sensor and a visualisation software obtaining excellent results. Future work will consist in the integration of further devices and the extension of our safety concept. An accurate clinical evaluation will take place continuously.*

## 1. Introduction

Computer aided surgical methods and devices are gaining on importance in clinical routine as the quality of the surgical intervention is enhanced to a great extend. Especially craniofacial surgery lends itself to an integration of computer aided methods into the treatment of fractures, tumours, bone repositioning and implantology. The cooperation of the surgeons of the Clinic of Craniofacial Surgery of the University of Heidelberg and the engineers of the Institute of Process Control and Robotics of the University of Karlsruhe targets the development of a computer and robot aided operating theatre. The clinic offers to the engineers an ideal test bed for their computer aided surgical methods and apparatuses. The surgeons evaluate the methods with respect to applicability, ergonomy and safety.

The interaction between different software applications and devices requires a model for the communication and control of the whole system, namely a system architecture.

The architecture of a computer aided surgery environment could be described with an hierarchical model. The surgeon leads the hierarchy, he plans and performs the surgical intervention. The operation planning system supports the surgeon at this stage. The operation plan [Münchenberg-1998] is translated into an instruction graph which contains single procedures for the surgeon and the devices. The plan execution module should also be integrated in the system architecture, since it has to communicate with every device which is part of the operating theatre.

The requirements on a system architecture for the computer aided surgery are as follows:

[*] University of Heidelberg, Clinic of Cranio-Maxillo-Facial-Surgery
[Ψ] University of Karlsruhe, Institute of Process Control and Robotics

- Platform independence
- Asynchronous (non-blocking) communication capabilities
- Distributed software model supporting the development of different software device drivers.
- Real-time capabilities for fast and reliable interaction with the robot control system
- Control of the operational reliability of single devices and the whole system
- Monitoring of system states and user alerting
- Easy to handle programming library

In the following section we will present our approach addressing the described requirements.

# 2. The System Architecture

## 1.1 The Communication model

During a computer assisted operation the hardware devices should exchange information for example about tool positions, trajectories, system states etc. The communication software library should respect the following requirements:

- Typical computer aided surgery environment are based on hardware working under different operating systems, like UNIX, VxWorks and WindowsNT, the communication should be platform independent. Low level communication can be realised via Ethernet, RS-232, CANBUS etc.
- Calls to other services should not block the execution of the calling process (asynchronous communication). Synchronisation points are to be defined when a dependency is given.
- A distributed software model is needed since many devices and software services are to be integrated in the architecture.
- Robot control needs a fast communication between the different services, therefore a real-time environment is needed.

The Common Object Request Broker Architecture (CORBA) defines a framework for developing distributed applications. Many implementations supports different operating systems. Communication between different components can be realised in CORBA by standard function calls, one-directional calls or by the use of an event exchange model.

We chose the Adaptive Communication Environment (ACE) and the Object Request Broker (TAO) developed on the Washington university ([Schmidt-97] and [Schmidt-98]) since it accords to all requirements of our architectural concept.

In particular we make an extensive use of the real-time event service for the exchange of information between the individual processes. The real-time event service offers a priority based event distribution which is needed since critical system states are to be handled immediately.

The information entities are represented by event structures. They consist of a unique event identifier and a data attachment which could be of any desired type. Events are used for requests, responses, state reports or data transfers.

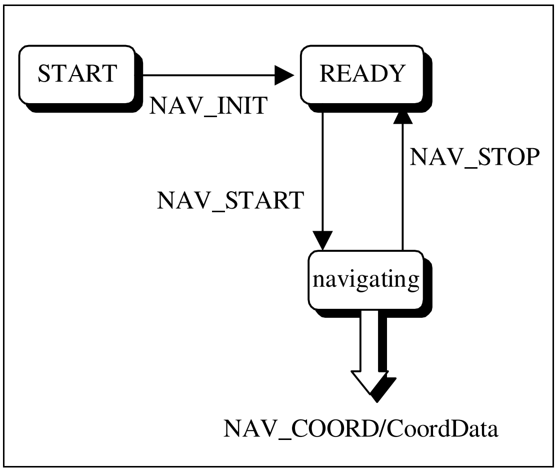


**Figure 1.** State graph example for the navigation system.

The architecture is behaviour oriented which means it reacts on incoming events. **Figure 1** shows a small example. START represents the initial state of the process after it is been started. The navigation system can be initialised by a NAV_INIT-event when it is in START-state. After initialisation the process goes to the READY-state and is ready for navigation. The service goes to the *navigating*-state after it has received a NAV_START event. It starts a thread which reads the navigation coordinates an sends them to the event channel periodically. The NAV_STOP event stops the navigation, the service goes back to the READY-state.

### 1.2 The virtual device concept

Our set up can be used in passive mode (without robot system) and active mode (including the robot system).

The modular conception of the system architecture allows adding or removing devices and services as needed. The surgeon and his assistants are able to plug in the desired or needed hardware as described in the operation plan. Future hardware changes are supported by an hardware independent driver model, a so called **virtual device model**.

The virtual devices should represent the necessary subset of function. The supported functionality must be selected in respect to the device features and the requirements of the computer aided surgery environment. Hardware specific features can be supported through additional interface functions, but, if hardware is changed later on, the use of these features could result in a time expensive redesign of the client applications.

As described earlier our hardware test set up consists of mainly three hardware devices, the robot control system, the navigation hardware and the force torque sensor. We focused on the selection of a subset of features which are probably common to different implementations of the devices like position control, force values or navigation coordinates.

**Figure 2** illustrates the virtual device model. Hardware devices are represented by their drivers. The devices communicates with the system through an event channel. Calls to the software driver are done by sending events through the event channel. Driver responses are also propagated through the event channel.

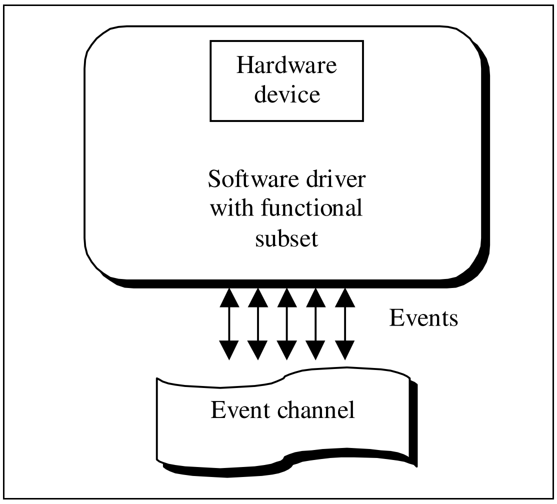


**Figure 2.** Event based virtual device model

### 1.3 Implementation details

A disadvantage of CORBA is the large implementation overhead for the registration of the ORB and the needed services. We've implemented a base class (*DERAS_Base*) which performs the CORBA overhead and defines standard macros for the definition of event callbacks in C++. Basically a developer can implement a service compliant to our system without any CORBA knowledge, just by requesting an event identifier set from the system administrator and implementing the callbacks for these events. Some system critical functions like periodic control procedures, timeout exceptions and service registration are implemented in the base class..

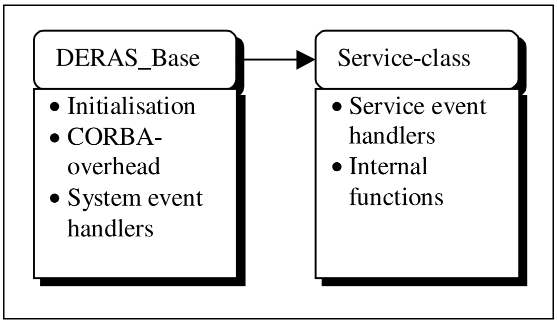


**Figure 3.** *DERAS_Base* Class inheritance example.

Every service must be inherited from the *DERAS_Base* class. It is started by a call to the base *::run()*-method and registers automatically to a real-time event channel. Registered events are filtered [Schmidt-94]. The application doesn't have to handle undesired events which increases the services performance. **Figure 3** shows an example for the definition of a service class.

The object oriented approach of our implementation leads to a fast and easy to handle development of services as we already experienced.

## 3. System safety

Interaction between humans and robots in critical environments requires a well devised safety concept. Thus an analysis of the systems safety should consider the complete set up and all its components as a unit. The operation of each device as well as the communication between different devices should be reliable. At this stage our focus is the software safety concept, further developments will include hardware safety.

Watchdog processes controls the operational reliability of the system reports errors to the user and perform an automatic recover if possible. Redundant control-procedures ensure the safety of the system. We've also developed a system registry for scheduling and control purposes.

Every service in our environment must register itself in a common registry database. The registry keeps information about the services and checks them periodically. Failures are immediately reported through system events. The registry reports failures to the user (the surgeon or his assistant) and is able to restart registered services if needed.

Beside the reliability of the device driver services also the consistency of data must be ensured. As CORBA is based on standard network-protocols, some low level error checking is already performed. Critical services performs additional consistency checks by the generation of checksums. We plan to include data checking as a base functionality of the *DERAS_Base* class.

## 4. Clinical tests

First tests using an infrared navigation system (NDI-Polaris), a surgical robot (MAQUET CASPAR) and a force-torque sensor have already been performed

[Burghart-1998]. We also have developed and tested an SGI-Inventor based visualisation and navigation software which was integrated in our environment. We tested the performance and usability of our concept the evaluation set up obtaining excellent results. The event throughput (300-1000 events/sec) is more than sufficient for our purposes. The set up is much easier to handle, since the different devices are controlled automatically.


**Figure 4**. Test set up with the surgical robot and the infrared navigation system

## 5. Conclusion and future goals

A consistent system architecture is fundamental for the application of more complex computer aided surgery environments in the clinical practice. Our ORB-based approach is open to any extension since it is based on a standardised library. A connection to a patient database is also possible since many clinical administration systems (SAP R/3) supports the CORBA standard.

Future work includes the integration of endeffectors, further sensors and human-machine-interfaces into the presented architecture. A continuous clinical evaluation in the animal laboratory and in the operating theatre will lead to a constant improvement of the safety concept.

## 6. Acknowledgement

This research was performed at the Clinic of Craniofacial Surgery headed by Prof. Dr. Dr. J. Muehling at the University of Heidelberg, and the Institute of Process Control and Robotics, headed by Prof. Dr.-Ing. U. Rembold and Prof. Dr.-Ing. H. Woern, at the University of Karlsruhe. The work is being funded by the German Research Foundation, as it is part of the special research program "Information Technology in Medicine - Computer and Sensor Supported Surgery".

## References

[Burghart-1998] Burghart, C.; Hassfeld, S.; Raczkowsky, J.; Rembold, U. & Wörn, H., 'A Surgical robotic System for Maxillofacial Surgery', Proceedings of Computer Assisted Radiology and Surgery (CAR'98), Tokyo 1998

[Münchenberg-1998] Münchenberg, J.; Brief J.; Hassfeld, S.; Raczkowsky, J.; Rembold, U. & Wörn, H., 'Expert Supported Operation Planning in the Maxillofacial Surgery', Proceedings of Computer Assisted Radiology and Surgery (CAR'98), Tokyo 1998

[Schmidt-94] Schmidt, D.C., 'High-Performance Event Filtering for Dynamic Multi-point Applications', Proceedings 1st Workshop on High Performance Protocol Architectures (HIPPARCH, Sophia Antipolis, France, 1994, p 1--8.

[Schmidt-97] Schmidt, D.C.; Bector, R.; Levine, D.; Mungee, S. & Parulkar, G., 'TAO: a Middleware Framework for Real-time ORB Endsystems', Proceedings of the Workshop on Middleware for Real-Time Systems and Services, San Francisco, California 1997.

[Schmidt-98] Schmidt, D.C., 'An Architectural Overview of the ACE Framework: A Case-study of Successful Cross-platform Systems Software Reuse', *USENIX login magazine, Tools special issue*, 1998.