Introduction

This paper describes a new architecture for component-based healthcare information systems and discusses first experience gained during its implementation.

Current healthcare information systems are usually an ad-hoc federation of autonomous, monolithic systems held together by communication-servers (figure 1, Departmental View). The drawbacks of such an architecture, like lack of overall information integrity, introduction of artificial boundaries, duplicated functionality etc. are well known. These issues have lead to a common understanding that an architecture for new healthcare information systems should employ some kind of domain specific middleware which provides common services on which applications can be built with highly reduced effort to the needs of the individual organization (see [2], [10 - 14]).

This paradigm change shifts the view from ‘stovepipe’ departmental systems to an enterprise wide system of distributed components with open interfaces (figure 1, Enterprise View).

![Diagram](https://via.placeholder.com/150)

**Figure 1:** The changing paradigm of Health Informatics (after [9])

The European Committee for Standardization (CEN) has already defined a reference architecture for systems built according to this new paradigm in it’s HISA prestandard ([1], [3]). However, building systems from common services does not lead automatically to real interoperability between different systems and components from different vendors. In order to achieve interoperability the components must agree on common interoperability protocols from all levels of the seven-layer interoperability protocol model as shown in figure 2.
A large step towards this goal is architecting the federation of components (versus an ad-hoc federation) and having the individual components share the same architectural vision. This can be achieved by employing the business component approach recently described by Herzum and Sims [5], [6].

To assess the applicability of this approach for healthcare information systems we are currently working on the MINCOS project to develop a component-based integration framework for healthcare applications (MINCOS stands for Minerva Business Component System, where Minvera is the name for our hospital information system). An other major aspect of MINCOS is to leverage standard component application servers supporting Enterprise-JavaBeans [8] to provide a runtime environment for configurable business components conforming to the CORBAmed API [9].

Methods

When building component-based information systems one must focus more on the overall architecture of the system than when building traditional systems. Without a comprehensive architectural model it is impossible to build a coherent system of interworking software components.

In a well designed system it should be possible to trace the business concepts from the enterprise and domain models down to the components implementing them. To make sure that this is possible the MINCOS application architecture employs the concept of the business component. A business component is a well-defined composition of software artifacts which implements an autonomous business concept or business process from the domain model on all system tiers. It can be deployed in a component-based information system as an autonomous, reusable element and has published contracts with other business components. Figure 3 shows its unifying concept through all logical distribution tiers. The business component consists of distributed components on the logical user-, workspace-, enterprise- and resource-tiers, marked with U, W, E and R in the figure, which plug into their respective sockets in a distributed component execution environment (CEE). The logical tiers split a business component along functional boundaries and provide possibilities for a physical distribution of the business component (A common distribution pattern is to put the distributed components of the user-workspace domain on the users workstation, while the enterprise-resource domain is executing on the datacenter servers. But other distributions, even within the same system, are possible).
The availability of a comprehensive execution environment for all tiers is one of the most important aspects of the business component approach. It essentially frees the functional developer from exercising unnecessary creativity for implementing infrastructure code. Instead he can concentrate on the business requirements from the functional and semantics layers of the seven-layer model.

In MINCOS we use a commercial component application server for Enterprise-JavaBeans as a basis for the execution environment on the datacenter servers. An EJB server provides many of the technical core functionalities required for a CEE and allows to build components with maximum portability between server platforms. However, it should be noted that an EJB server does not offer everything needed for a CEE (see ‘Results’).

Usually MINCOS users run the user-workspace domain distributed components on their local workstation. This requires the availability of an execution environment on the workstation for these components. Since there is no established standard for system independent execution environments on workstations we designed the MINCOS Client Component Container (MCCC). It provides an application frame and can dynamically load configurable visual and non-visual distributed components (JavaBeans in our case) from the datacenter servers [4]. Another important service of the MCCC is an interface which makes it possible to exchange context information (like the selected patient) between business components within the MCCC and even with traditional applications on the workstation. This interface is modeled after the new CCOW standard being developed within the HL7 organization [7]. There we are currently actively involved in standardizing a CORBA mapping for CCOW, which will make CCOW more platform independent.

As stated before, for full interoperability there must be consensus over all protocols from the seven-layer model. To achieve consensus on the functional and semantics layers, MINCOS employs the HISA reference architecture and CORBAmed functional interfaces together with the basic concepts of the HL7 information model wherever possible. Extracting the basic concepts is essential for a generic domain-specific architecture since it is clear, for example, that in all healthcare organizations exists the business concept of a patient (or subject of care), but the exact attributes of a patient are likely to be different. MINCOS recognizes this, by providing in this case a ‘person’ business component which gets configured during deployment to handle ‘patient’ entities with the required attributes and persistence mechanisms.

Figure 4 presents the architecture of the resulting ‘person’ business component. The user-tier distributed components (UDC), which handle the interaction with the user, are property pages implemented by JavaBeans which plug into the User Model distributed component from the MCCC. They communicate with ‘person’ workspace-tier distributed components (WDC, non-visual JavaBeans) which implement the user’s model of a ‘person’. The View Manager is responsible for the live-cycle management of the components on these tiers. The enterprise-tier ‘person’ distributed components (EDC) are a collection of Enterprise JavaBeans which export their interfaces over a CORBAmed PIDS façade to make them available to other business components, regardless of their implementation language. These Enterprise JavaBeans use other Enterprise JavaBeans on the resource tier (RDC) to
persist the data corresponding to the ‘person’ entity. The resource-tier shields the enterprise tier from varying persistence mechanisms. This makes it relatively easy to switch, for example, from direct RDBMS access to HL7 messages.

Configuration interfaces are provided on all layers to adapt the business component to the actual healthcare information system to which it is deployed. However, it should be noted that especially distributed components on the upper layers can auto-configure to some degree by requesting meta-data from the lower components. For example, the ‘person’ distributed component on the workspace layer gets configured to handle ‘Patients’ and where to find its enterprise tier. It then asks the enterprise tier about available PIDS-traits and gets meta-information about the corresponding business-data-types from a data-dictionary.

Results and Conclusions
The business component approach allows building a healthcare information system as a federation of business components adhering to a common architectural paradigm. The use of Enterprise JavaBeans helps to free the developers from many issues, like life-cycle management, distribution transparency, high-availability, security etc. CORBA interfaces provide for cross-platform interoperability. Still, an ideal component execution
environment should be language and platform independent, which Enterprise JavaBeans are not. Also Enterprise JavaBeans do not yet provide sufficient capabilities to specify all issues of a business component. Products based on the CORBA Component Model [10] will be an major step into this direction when they become available. More problematic is, that current technologies leave too many technical things to the functional developer. Building a separation layer which shields the functional developer from the technical issues is a huge task. Consequently the separation layer in MINCOS is very thin. It is hoped that vendors will eventually build execution environments and complete business component factories on top of Enterprise JavaBeans and CORBA Component Model servers. Of course, all this still leaves open the question regarding the functionality of a client-side component execution environment. Here MINCOS makes first steps with the definition of the Mincos Client Component Container.

Developing component-based systems also puts more emphasis on the development process itself. Without a well-defined process it is unlikely to produce high-quality components which conform to a common architecture and don’t break the whole federation when deployed. With low quality components the complete system can essentially become unmanageable.

However, the MINCOS system shows that it is possible to overcome many of these issues with currently available technologies. The payback are systems built from components which can be adapted better to an organization’s needs, instead of adapting an organization to some prefabricated system. This opens an interesting path in-between complete in-house development and buying off-the-shelf systems, which will eventually allow healthcare organizations to architect an information system suitable to their needs, instead of plumbing together off-the-shelf systems.

References