Reihe: Telekommunikation @ Mediendienste · Band 10

Herausgegeben von Norbert Szyperski, Udo Winand, Dietrich Seibt, Rainer Kuhlen, Rudolf Pospischil und Claudia Löbbecke

Martin Engelien/Detlef Neumann (Hrsg.)

Virtuelle Organisation und Neue Medien 2000

Workshop GeNeMe2000 Gemeinschaften in Neuen Medien

TU Dresden, 5. und 6. Oktober 2000

JOSEF EUL VERLAG
Lohmar · Köln

Reihe: Telekommunikation @ Mediendienste · Band 10

Herausgegeben von Prof. Dr. Dr. h. c. Norbert Szyperski, Köln, Prof. Dr. Udo Winand, Kassel, Prof. Dr. Dietrich Seibt, Köln, Prof. Dr. Rainer Kuhlen, Konstanz, Dr. Rudolf Pospischil, Brüssel, und Prof. Dr. Claudia Löbbecke, Köln

PD Dr.-Ing. habil. Martin Engelien
Dipl.-Inf. Detlef Neumann (Hrsg.)

Virtuelle Organisation und Neue Medien 2000

Workshop GeNeMe2000
Gemeinschaften in Neuen Medien

TU Dresden, 5. und 6. Oktober 2000

JOSEF EUL VERLAG
Lohmar · Köln

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

GeNeMe <2000 Dresden>:
GeNeMe 2000 : Gemeinschaften in neuen Medien ; Dresden, 5. und 6. Oktober 2000, an der Fakultät Informatik an der Technischen Universität Dresden / Technische Universität Dresden, Fakultät Informatik, Institut für Angewandte Informatik, Privat-Dozentur „Angewandte Informatik". Martin Engelien ; Detlef Neumann (Hrsg.). – Lohmar ; Köln : Eul, 2000
(Reihe: Telekommunikation und Mediendienste ; Bd. 10)
ISBN 3-89012-786-X

Josef Eul Verlag GmbH
Brandsberg 6
53797 Lohmar
Tel.: 0 22 05 / 91 08 91
Fax: 0 22 05 / 91 08 92
http://www.eul-verlag.de
info@eul-verlag.de

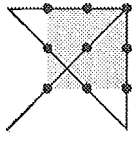
Printed in Germany
Druck: Rosch-Buch, Scheßlitz

Technische Universität Dresden
Fakultät Informatik • Institut für Angewandte Informatik
Privat-Dozentur „Angewandte Informatik“

PD Dr.-Ing. habil. Martin Engelien
Dipl.-Inf. Detlef Neumann
(Hrsg.)

an der
Fakultät Informatik der Technischen Universität Dresden

gefördert von der Klaus Tschira Stiftung,
gemeinnützige Gesellschaft mit beschränkter Haftung,
unter Mitwirkung der Gesellschaft für Informatik e.V., Regionalgruppe Dresden



am 5. und 6. Oktober 2000
in Dresden

http://www-emw.inf.tu-dresden.de/geneme
Kontakt: Detlef Neumann (dn3@inf.tu-dresden.de)

D.2. Ein allgemein gültiges Abrechnungssystem für Online- und Offline-Dienste – „Report on work in progress“

Dr. Wolfram Amme
Heike Hotzel
Prof. Dr. Wilhelm Rossak
René Stolle
Institut für Informatik, Universität Jena

1. Einführung

1.1 eVerlage.de - Das Gesamtprojekt

Das in diesem Paper vorzustellende Projekt ist ein Teilprojekt der Sonderfördermaßnahme 9: *„Erprobung elektronischer Angebotsformen, Abrechnungsmodelle und Zahlungsverfahren auf einer Testplattform für GLOBAL INFO“*. Hierbei soll das System eVerlage zum kommerziellen Vertrieb von Inhalten (textuell, multimedial) geschaffen werden /1/. Am Gesamtprojekt sind beteiligt:

- **4 Technische Arbeitsgruppen** (OFFIS Oldenburg, FAST e.V. München, HTWK Leipzig, FIZ Karlsruhe). Diese haben die Aufgabe, das Projekt zu realisieren. Wie in Abschnitt 3 dargelegt, ist das eVerlage-System sehr modular aufgebaut, was eine effektive Zusammenarbeit dieser räumlich verteilten Arbeitsgruppen ermöglicht.
- Derzeit **10 Verlage** (DIN, Hanser, Harry Deutsch, Oldenbourg, Spektrum, Teubner, Thieme, Vieweg, VdF und VDE). Verlage wurden in das Projekt involviert, um einerseits nicht an deren Vorstellungen „vorbeizuentwickeln“, und andererseits mit realen Inhalten arbeiten zu können.
- **3 Bibliotheken** (Universitätsbibliothek Bielefeld, SUB Göttingen, ThULB Jena). Zur möglichst schnellen Schaffung einer breiten Benutzerbasis für das eVerlage-System wurden Bibliotheken mit in das Projekt eingebunden. Deren Aufgabe ist es, eVerlage zu einem Bestandteil ihres (elektronischen) Angebots zu machen, und somit eine transparente Nutzung für deren Benutzer zu ermöglichen. Das eVerlage-System hat zu diesem Zweck entsprechende Schnittstellen, wie in Abschnitt 3 dargelegt wird.

1.2 Das Teilprojekt CAJ

Kleinstbeträge via Internet abzurechnen (Micropayment) ist derzeit eine noch nicht gelöste Aufgabe. Das ist durchaus auch ein Problem für das eVerlage-System, da hier hauptsächlich Kleinstbeträge anfallen. Es gibt verschiedene Lösungsansätze (elektronische Münze, elektronischer Scheck, Chip-/Geldkarten, ...) von verschiedenen Herstellern (Cybercash, Digicash, Mondex, ...). Wirklich durchgesetzt hat sich aber bislang noch kein Produkt oder Verfahren. Viele aktuelle Internetangebote, bei denen Kleinstbeträge abgerechnet werden müssen, arbeiten daher mit Benutzerkonten. Der Benutzer transferiert einen größeren Betrag (Macropayment im Internet stellt kein Problem dar) und kann fortan über sein Guthaben verfügen. Dies schafft natürlich eine Hemmschwelle, zumindest bei den Benutzern, die von dem entsprechenden Internetangebot noch nicht überzeugt sind und es erst einmal nur testen wollen. Das eVerlage-Projekt unterstützt dieses Konzept, es sollen ebenfalls Micropaymentverfahren (derzeit Paybox, Geldkarte ist in Arbeit) unterstützt werden /1/.

Die Bibliotheken wurden nicht zuletzt auf Grund dieses Problems mit in das eVerlage-Projekt eingebunden. Es soll erreicht werden, daß das Abrechnungsproblem bei der Nutzung von eVerlage.de wenigstens für die Nutzer der Bibliotheken komfortabel gelöst ist, so daß diese rege Gebrauch davon machen können.

Mit dem chipkartengesteuerten Abrechnungssystem Jena (CAJ) streben wir an, den softwaretechnischen Grundstein für ein allgemeines, offenes, adaptierbares und nahezu beliebig erweiterbares Abrechnungssystem zu schaffen. Dieses soll prinzipiell von beliebigen Organisationen (Firmen, Universitäten, Behörden, usw.) eingesetzt werden können, bei denen größere Benutzergruppen (autorisiert) auf (kostenpflichtige) Ressourcen zugreifen müssen. Das CAJ soll dazu sowohl an vorhandene Strukturen anpaßbar sein, als auch eigene Strukturen zur Verfügung stellen. Es soll die Abwicklung verschiedener, ggf. organisationsübergreifender Dienste, mit einer Identifikation (Chipkarte oder Benutzername/Paßwort) ermöglichen. Auch nach Einführung des CAJ soll die entsprechende Institution stetig die Möglichkeit haben, dieses System um neue Funktionen zu erweitern bzw. vorhandene Anwendungen zu modifizieren und zu ergänzen.

Diese hohen Ziele des CAJ sollen durch eine extrem offene und intelligente Architektur erreicht werden. Ergebnisse aus anderen Projekten mit ähnlichen Zielen der Verallgemeinerung und Wiederverwendbarkeit von Software werden zum Entwurf des CAJ herangezogen. So wird bei der rein softwaretechnischen Realisierung des CAJ ähnlich zum bekannten IBM-San Francisco-Project™ /2/ vorgegangen. Zunächst soll die Kernfunktionalität vieler/aller denkbaren CAJ-Dienste identifiziert und im sogenannten CAJ-Kernel (CS) implementiert werden. Die eigentliche Funktionalität

konkreter Dienste wird dann durch die Implementierung von CAJ-Applikationen realisiert.

2. Die Architektur das CAJ

Das CAJ ist ein Online-System. Jede Aktion wird vom CAJ Server (CS) genehmigt oder verweigert. Die Chipkarte dient lediglich der Identifikation (via Chipkartenidentifikationsnummer (ID) o.ä.), welche ebenfalls durch Eingabe von Benutzername und Kennwort erfolgen kann (ähnlich zur Kreditkarte mit Kreditkartennummer, Name und Verfallsdatum der Karte). Der allgemeine Aufbau ist in Abbildung 1 dargestellt.

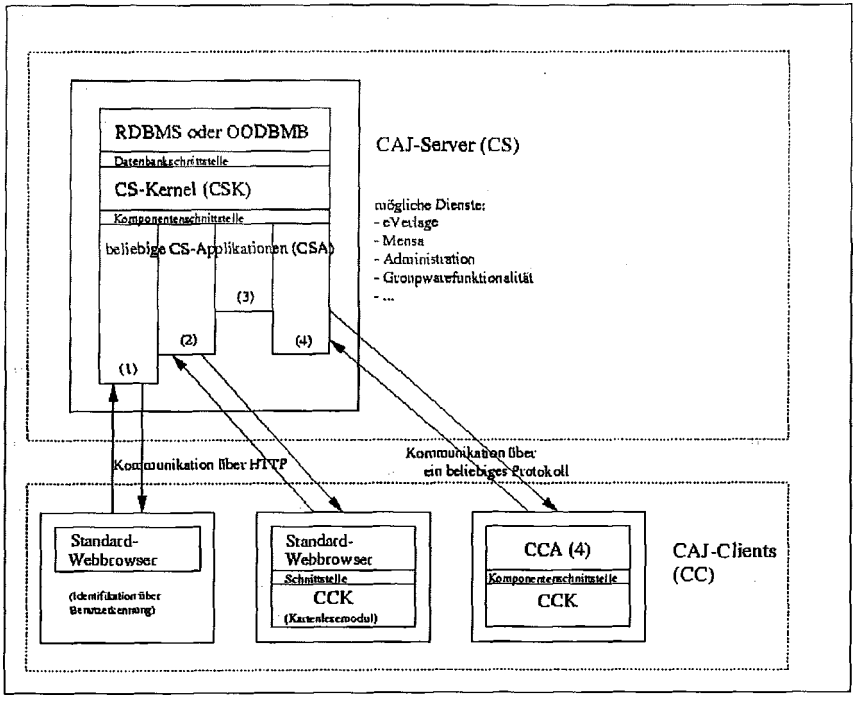


Abbildung 1: Der allgemeine Aufbau des CAJ

Der CS besteht zunächst einmal aus dem CAJ-Server-Kernel (CSK). Dieser verwaltet die Businessdaten, die Daten, welche (prinzipiell) von allen CAJ-Anwendungen benötigt werden. Das sind in erster Linie Benutzer-, Chipkarten- und Transaktionsdaten.

Auf diese Daten kann ausschließlich über die sogenannten CSK-Interfacefunktionen zugegriffen werden, hier einige Beispiele:

- bool checkValid (c_id) – Überprüft die Gültigkeit der Chipkarte mit der ID c_id.
- int getUID (c_id) – Ermittelt den Benutzer zur Chipkarte mit der ID c_id.
- int getAmmount (u_id) – Liefert den aktuellen Kontostand des Benutzers u_id.
- getUserData (u_id) – Stellt Nutzerdaten zur Verfügung.
- bool startPayment (..) – Realisiert eine Zahlung.
- getUserList () – Liefert eine Liste aller Benutzer im System.

Möglich ist dabei die Schaffung verschiedener Interfaces für verschiedene Sicherheitslevel.

Mit dem CSK allein wird noch keine wirkliche Funktionalität erreicht. Dazu müssen CAJ-Applikationen implementiert werden. Auch hier unterscheiden wir die Server- und Clientseite. Wenden wir uns zunächst den CAJ-Server-Applikationen (CSA) zu. Diese können über eine CORBA-Schnittstelle auf die CSK-Interfacefunktionen zugreifen. Neben der Anwendungslogik müssen die CSA einige Funktionen implementieren, die vom CSK aufgerufen werden, beispielsweise:

- bool addUser (u_id) – Damit teilt der CSK den CSA mit, daß es im System einen neuen Benutzer mit der ID u_id gibt.
- bool delUser (u_id) – Der Benutzer u_id soll aus dem System entfernt werden.
- bool goOffline () – Die CSA wird gebeten, sich zu beenden.
- bool resMessage (msg_id) – Die CSA erhält eine allgemeine Nachricht.
- bool verifyData () – Es wird angewiesen, die Konsistenz der Applikationsdaten mit den Businessdaten zu überprüfen.

Diese Funktionen nennen wir CSAK-Interfacefunktionen, in Abgrenzung zu den CSAC-Interfacefunktionen, welche der entsprechende CAJ-Client aufrufen kann.

Über die CSK- und CSAK-Interfacefunktionen ist also eine bidirektionale Client/Server-Kommunikation möglich. Die Booleschen Rückgabewerte der CSAK-Interfacefunktionen geben dem CSK ein Feedback über den Erfolg der geforderten Aktion. So wird ein Benutzer erst dann aus dem System gelöscht (delUser()), wenn alle CSA entsprechende lokale Vorkehrungen dazu getroffen haben.

Abbildung 2 zeigt den Aufbau des CS im Überblick. Kommunikation ist ausschließlich entlang der gezeichneten Pfeile möglich. Die CSA können nicht direkt miteinander kommunizieren, sondern ausschließlich über den CSK.

CORBA kommt zum Einsatz, weil es die notwendige Funktionalität liefert und zudem plattform- und programmiersprachenunabhängig ist. Die Alternativen (DCOM™, EJB™) sind entweder plattform- oder sprachabhängig. Der Mico-ORB soll eingesetzt werden, weil dieser sehr standardkonform implementiert ist. Er ist einer von sehr

wenigen ORBs, die ein entsprechendes Zertifizierungsverfahren der Opengroup bestanden haben /3/4/5/. Via IIOP können aber auch beliebige andere Produkte mit dem Mico-ORB und damit mit dem CSK kommunizieren.

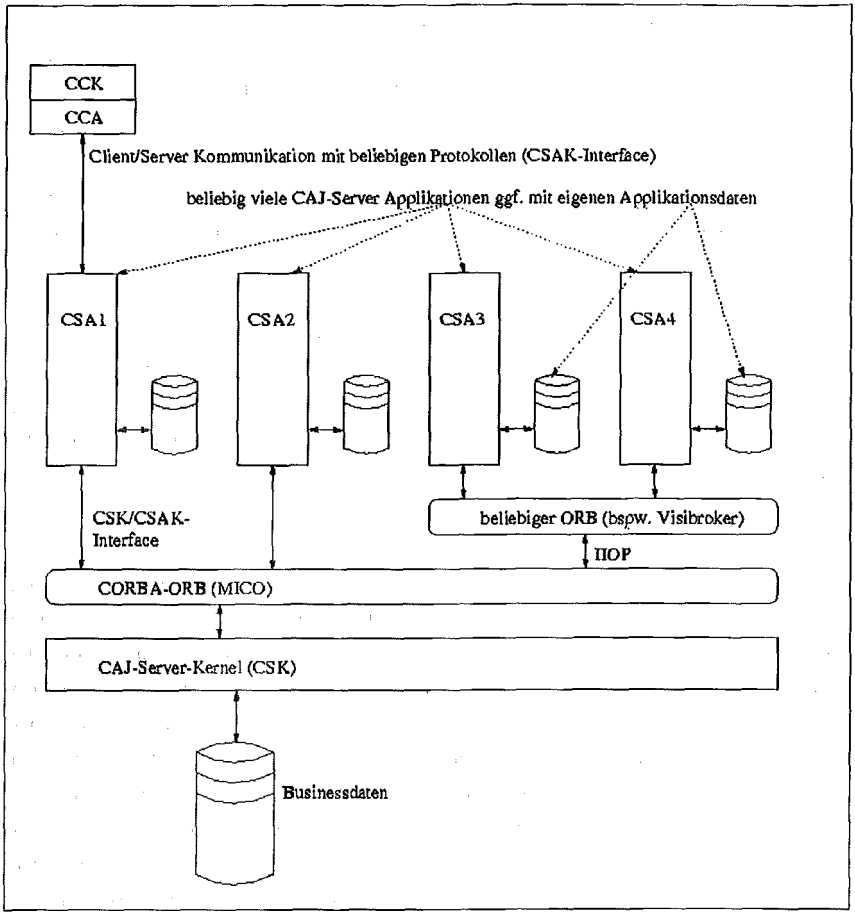


Abbildung 2: Der CAJ Server

Die Daten des CSK (Businessdaten) werden in der Regel nicht ausreichen, um konkrete Dienste zu realisieren. Deshalb haben die CSA die Möglichkeit, eigene, lokale Daten zu verwalten, die Applikationsdaten genannt werden sollen. Diese werden in separaten Datenbanken gehalten, wobei es sich um eigene Datenbankinstanzen handeln kann, oder um Bereiche der Hauptdatenbank, die aber strikt von den Businessdaten getrennt sind,

auf welche die CSA ausschließlich über die CSK-Interfacefunktionen zugreifen können. Diese strikte Trennung von Applikations- und Businessdaten schafft ein hohes Sicherheitsniveau, erfordert aber in bestimmten Situationen auch einen Mehraufwand bei der Implementierung von CSA. So können etwaige referenzielle Integritäten nicht vom DBMS verwaltet, sondern müssen durch die Applikationen überwacht werden.
In vielen Fällen muß neben der CSA ebenfalls eine CAJ-Client-Applikation (CCA) entwickelt werden. Diese kann mit dem CAJ-Server ausschließlich über die bereits angesprochenen CSAC-Interfacefunktionen der entsprechenden CSA kommunizieren, erlangt also keinen direkten Zugriff auf den CSK. In der anderen Richtung stehen der CCA Funktionen des CAJ-Client-Kernels (CCK) zur Verfügung, der zur Zeit lediglich als Deviceinterface zu diversen Chipkartenlesegeräten fungieren soll. Dabei soll der CCK zunächst folgende Funktionen implementieren:

- getCID () – Ermitteln der Kartenidentifikationsnummer.
- keepCard () – Einbehalten der Karte (im Falle eines Diebstahls o.ä.), sofern das Lesegerät diese Möglichkeit bietet.

Neben der Neuimplementierung können ebenfalls fertige Clients (z. B. Webbrowser) als CCA fungieren. In diesem Fall muß die CSA ein entsprechendes Protokoll (z. B. HTTP) unterstützen.

3. Anwendungsbeispiel eVerlage.de

Die Beziehung zwischen dem CAJ und dem eVerlage-System kann als die Beziehung zwischen einer Bank und einer Kreditkartengesellschaft angesehen werden. Ein Kunde dieser Bank erhält von der Bank eine Kreditkarte ausgestellt und kann damit die Leistungen der Kreditkartengesellschaft nutzen, ohne sich vorher bei dieser anmelden zu müssen, er muß sich nicht einmal der Tatsache bewußt werden, daß er ein Kunde der Kreditkartengesellschaft ist. Vergleichbar kann der Nutzer des CAJ die Dienstleistungen des eVerlage-Systems nutzen, ohne sich dort explizit anzumelden. Dies übernimmt das CAJ, sobald der Nutzer ins CAJ eingetragen wird. Das eVerlage-System ist auf derartige Anforderungen bestens vorbereitet. So bietet es via Java RMI Funktionen an, um Benutzer einzutragen (userRegistration()), Benutzerdaten zu modifizieren und anderes mehr. Auch die eigentliche Recherche, Übertragung der Dokumente und Abrechnungsdaten erfolgt via RMI.
Das eVerlage Team hat schon sehr früh eine Trennung des Systems in 3 Komponenten vorgenommen:

- **Der Provideragent** speichert Dokumente und Metadaten und liefert ein Interface, um diese abzurufen. So können Suchanfragen an den PA gerichtet werden (searchRequest()). Dieser ermittelt die Treffer und liefert das Ergebnis in Form eines

Baumes zurück. Um die Darstellung dieser Treffer (ggf. auf einer Webseite) muß sich dann der Useragent (siehe unten) kümmern.

- **Der Centralagent** verwaltet Nutzer-, Lizenz-, Transaktionsdaten, etc. Diese Komponente stellt ebenfalls eine Schnittstelle zur Verfügung, um (begrenzt) auf diese Daten zugreifen zu können.
- **Der Useragent** implementiert letztlich die **Benutzerschnittstelle** zum eVerlage-System. Er kommuniziert **mit dem Provideragent um Recherchen durchzuführen** und Dokumente **zu erhalten. Im Centralagent werden die anfallenden Beträge gebucht, entsprechende Lizenzen gespeichert etc.**

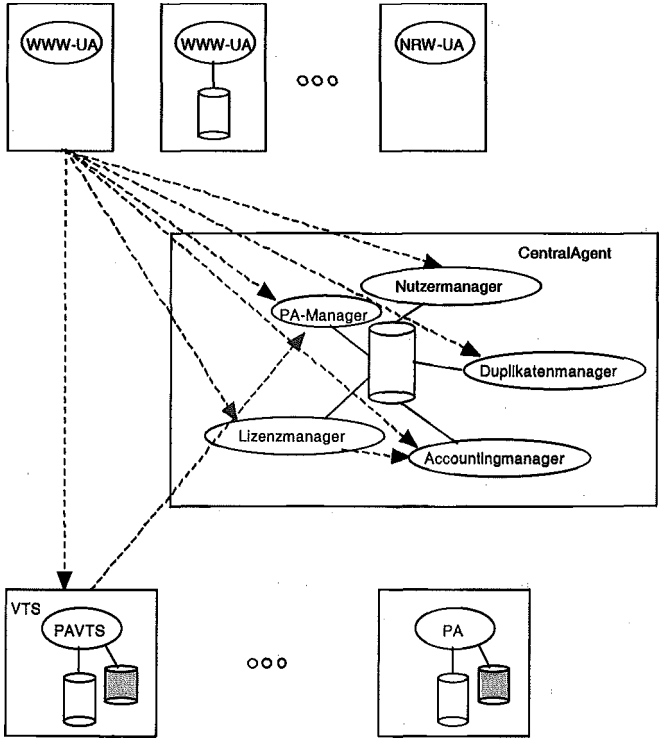


Abbildung 3: Architektur des eVerlage-Systems

Die eVerlage-CSA stellt einen spezialisierten Useragent dar, der eine Brücke zwischen dem CAJ und dem eVerlage-System realisiert. Wie bereits angedeutet, wird jeder CAJ-

Benutzer implizit ein Nutzer von eVerlage.de. Die addUser()-Funktion der CSA muß also umgehend die Benutzerregistrierung (userRegistration()) im Centralagent des eVerlage-Systems vornehmen. Danach kann der CAJ-Nutzer sofort in eVerlage.de recherchieren und, sofern er ein gedecktes Konto im CAJ hat, auch kostenpflichtige Dokumente abrufen. Um die Buchung der (Kleinst-)Beträge kümmert sich das CAJ.
In den Applikationsdaten verwaltet die CSA "eVerlage" die Zugangsdaten zum eVerlage-System. Nutzernamen werden dabei etwa nach folgendem Schema vergeben:

```
CAJ_USER_Nutzerlogin
```

wobei `Nutzerlogin` das Login des Nutzers im CAJ ist. Da das Nutzerlogin im CAJ eindeutig ist, wird die Bezeichnung im eVerlage-System ebenfalls eindeutig sein, da der Präfix CAJ_USER_ ausschließlich für CAJ-Benutzer reserviert ist. Wie mit den Paßwörtern verfahren wird, steht zur Zeit noch nicht fest.
Die CSA-Funktion delUser() sollte den Benutzer entsprechend am eVerlage-System abmelden, wobei derzeit die Frage offen bleibt, was mit den erworbenen Lizenzen des Nutzers geschehen soll.

4. Anwendungsbeispiel Zugangskontrollsystem

Die Flexibilität des CAJ erlaubt dessen Einsatz auf sehr unterschiedlichen Gebieten. So läßt sich beispielsweise ein Türöffnungssystem realisieren, bei dem es in der Regel nichts abzurechnen gibt. Zu implementieren sind hier die CAJ-Server und Client-Applikation.
Die CSA "Zugangskontrollsystem" beinhaltet in ihren lokalen Applikationsdaten eine Übersicht über alle zu verwaltenden Türen (und ggf. auch der Gebäude) und deren Zuordnung zu den Benutzern, die diese öffnen dürfen. Die Applikation muß Möglichkeiten zur Verfügung stellen, diese Daten zu pflegen, insbesondere muß realisiert werden:

- Übernahme aller Nutzer bei Neuinstallation dieser Applikation. Dazu kann die CSK-Interfacefunktion getUserList() benutzt werden.
- Implementierung von addUser(), Zuordnung dieses Nutzers zu den ihm offenstehenden Türen.
- Implementierung von delUser(), Löschen aller Zuordungen.
- Funktionen von An- und Abmelden von Türen.

Werden sehr viele Türen verwaltet, muß an dieser Stelle mit Templates oder ähnlichem gearbeitet werden, die ebenfalls in der CSA implementiert werden können.
Da das DBMS nicht über die referenzielle Integrität wachen kann, muß die CSA "Zugangskontrollsystem" hier sehr sauber operieren. Auf die Aufrufe addUser() und delUser() muß entsprechend reagiert werden. Auch sollte die Funktion verifyData()

gewissenhaft implementiert werden, so daß Inkonsistenzen (beispielsweise nach einem Systemausfall) schnell aufgedeckt werden können.
Die CCA muß dagegen nur eine sehr simple Funktionalität realisieren, wie die folgende Tabelle zeigt:

CCA	c_id = CCK.getCID() CSA.checkCardAccess (c_id, door_id)
CSA.checkCardAccess	CSK.checkCard (c_id) u_id = CSK.getUID(c_id) checkUserAccess (u_id, door_id)
CCA	Entweder: Öffne Tür Oder: Laß die Tür geschlossen (und behalte die Karte)

Tabelle 1: Funktionalität der CCA

Hier wird ein Autorisierungsvorgang dargestellt, wie er immer durchlaufen wird, wenn eine Tür geöffnet werden soll. Zunächst wird die Kartenidentifikationsnummer ermittelt, wobei auf eine CAJ-Client-Kernel-Interfacefunktion zurückgegriffen wird. Nun wird bereits die CSAC-Interfacefunktion checkCardAccess() aufgerufen. Diese ermittelt den zur Karte gehörigen Benutzer (aus den Businessdaten) und prüft anhand der Applikationsdaten. Das Ergebnis entscheidet, ob die CCA die Tür öffnet oder weiter geschlossen hält.
Das Zugangskontrollsystem bietet ebenfalls eine gute Anwendung für eine allgemeine Nachricht (resMessage()). So wäre es denkbar, eine Nachricht „Feuer-Alarm" zu kreieren, auf die das Zugangskontrollsystem entsprechend reagieren könnte, beispielsweise durch das Öffnen aller Türen in dem betroffenen Gebäude.

5. Aktueller Stand und Ausblick

Hier wurde unsere Arbeit zum gegenwärtigen Zeitpunkt dargestellt. Die vorgestellte Architektur wurde in kleinen Bruchstücken bereits implementiert, um sicherzustellen, daß diese tatsächlich realisierbar ist. Mit den reinen Implementierungsarbeiten soll umgehend begonnen werden, ein Prototyp wird in wenigen Wochen fertiggestellt sein. Obwohl wir die Architektur gründlich geplant und diskutiert haben, ist es dennoch nicht auszuschließen, daß sich im Laufe der Implementierung noch das eine oder andere ändert, den aktuellen Projektstand erfahren sie unter http://caj.informatik.uni-jena.de.

6. Referenzen

/1/ http://www.everlage.de

/2/ http://www.ibm.com/software/ad/sanfrancisco

/3/ http://www.mico.org

/4/ http://www.omg.org

/5/ http://www.opengroup.org