

Architekturstudie zur OSUN-Plattform

Marc Sihling, Alexander Vilbig, Ingolf Krüger

Institut für Informatik
Technische Universität München
D-80290 München
<http://www4.informatik.tu-muenchen.de/>

27. März 2000

Kurzfassung

Siemens ICN steht vor der Herausforderung, die Architektur ihrer bestehenden Telekommunikationsanlagen im Hinblick auf IP-Technologien zu öffnen. Dabei sind folgende Punkte wesentlich:

Mehrwertdienste durch Netzkonvergenz: Die Konvergenz von IP- und leitungsvermittelten Netzen schafft das Potential für neue Mehrwertdienste (*Internet Supplementary Services, ISS*). Beispiele für solche Dienste sind „Click-to-Dial“ (CtD), „Click-to-Conference“, und „Unified Messaging“.

Verkürzte Entwicklungszyklen: Die hohe Dynamik im IP-Bereich erfordert kurze Entwicklungszyklen für Mehrwertdienste.

Interoperabilität: Externe Dienstbringer sollen über definierte Einstiegspunkte auf Basisdienste der Vermittlungsanlagen zugreifen können. Diese Öffnung ermöglicht die flexible Integration von Telekommunikationsdiensten in bestehende und neue IT-Anwendungen.

Qualität: In Anlehnung an die Zuverlässigkeit, Sicherheit und Performanz der traditionellen Telekommunikationsdienste gelten ähnlich hohe Zuverlässigkeitsanforderungen für die neuen Mehrwertdienste.

Siemens ICN begegnet dieser Herausforderung durch die Definition und Entwicklung der OSUN-Plattform zur Realisierung der ISS. Einer der bereits genannten Mehrwertdienste, CtD, liegt als Pilotimplementierung vor. Diese stellt auch eine Referenzimplementierung eines Ausschnitts der OSUN-Plattform dar.

Der Aufbau einer tragfähigen, flexiblen und leistungsfähigen Software-Architektur ist für das Erreichen der mit OSUN verfolgten Ziele von entscheidender strategischer Bedeutung. Die gewählte Software-Architektur definiert die funktionalen Komponenten, ihre Schnittstellen und Interaktionen, die logische und technische Struktur des Gesamtsystems, sowie die Freiheitsgrade für die zukünftige Weiterentwicklung. Die geeignete Wahl einer Software-Architektur für OSUN ist damit ein Schlüsselfaktor für den Markterfolg der ISS.

Der Gegenstand der vorliegenden Studie ist die Begutachtung der Software-Architektur der CtD-Pilotimplementierung, mit besonderem Augenmerk auf

- die **Erweiterbarkeit** der Architektur um neue Dienste,
- die Unterstützung gegebener **Standards** wie SIP/PINT und Parlay,
- die **Verständlichkeit** der Darstellung, bzw. Dokumentation der Architektur.

Ausgangspunkt der Begutachtung war hauptsächlich die von Siemens ICN bereitgestellte Dokumentation der CtD-Architektur in Form eines Klassendiagramms mit zugehöriger textueller Beschreibung. Zur Veranschaulichung der Systemabläufe lagen Sequenzdiagramme vor. Im Rahmen mehrerer gemeinsamer Workshops von Siemens ICN und der TU München wurde ein gemeinsames Verständnis der Architektur erarbeitet. Zusätzlich war der Quelltext der CtD-Pilotimplementierung verfügbar.

Im Rahmen der Begutachtung wurde zunächst eine Bewertung der Architektur der Pilotimplementierung im Hinblick auf den genannten Kriterienkatalog vorgenommen. Weiterhin wurde die vorhandene Architektur in logische Schichten strukturiert. Dabei wurde insbesondere zwischen der *Anwendungsschicht*, der *Serviceschicht* und der *Vermittlungsschicht* unterschieden. Die Vermittlungsschicht stellt eine Schnittstelle zur eingesetzten Vermittlungstechnologie bereit. Die Serviceschicht beherbergt Basisdienste, wie etwa zum Verbindungsmanagement, zur Autorisierung und zur Abrechnung. Die Applikationsschicht nutzt diese Basisdienste zur Bereitstellung der eigentlichen Mehrwertdienste. Die Schichtung ist dahingehend logischer Natur, daß sie nicht notwendigerweise Prozessgrenzen für die Implementierung vorgibt. Anhand der Schichtenarchitektur wurde eine Reihe von Architektur-Varianten entwickelt, die das Spannungsfeld der oben genannten Evaluierungskriterien abdecken. Entsprechend der weiteren strategischen Ausrichtung der OSUN-Plattform empfiehlt sich die Auswahl einer der Varianten zur Umsetzung der OSUN-Plattform. Schließlich wurden Migrationsstrategien erarbeitet, die von der aktuellen Architektur zu den zusammen mit Siemens ICN als geeignet identifizierten Architekturvarianten führen.

Die Begutachtung liefert folgende Ergebnisse:

- Die vorliegende Pilotimplementierung ist ein wertvoller Beitrag zum Verständnis der Architektur der OSUN-Plattform. Sie kann auch für eine erste Performanz- bzw. Realisierbarkeitsabschätzung herangezogen werden.
- Die Beschreibung der Architektur der Pilotimplementierung läßt sich durch geeignete logische Strukturierung und entsprechende, über Klassendiagramme hinausgehende, Dokumentation zusätzlich aufwerten. Bereits einfache Strukturdiagramme, wie sie in dieser Studie eingesetzt werden, führen zu einer transparenteren Darstellung des Abstraktionsgrades der einzelnen Systemkomponenten.
- Zur Darstellung der logischen Struktur ist die Wahl einer Schichtenarchitektur besonders geeignet. Diese unterstützt insbesondere die klare Kapselung
 - der Vermittlungsdienste gegenüber der Vermittlungshardware einerseits, und
 - der Mehrwert-Dienste bzw. Applikationen gegenüber den Vermittlungsdiensten andererseits.

Dies trägt entscheidend zur Verbesserung der Erweiterbarkeit und Verständlichkeit der Architektur bei. Zudem unterstützt die Schichtenarchitektur den definierten Zugriff auf die angebotenen Dienste über Standardprotokolle und Middleware-Technologien wie SIP/PINT respektive CORBA.

- Die vorliegende Pilotimplementierung setzt die vorgeschlagene Schichtung bereits in Teilen um; insbesondere ist die Schnittstelle zur Vermittlungshardware bereits vorgezeichnet.
- Abhängig von der strategischen Ausrichtung der OSUN-Plattform läßt sich die Schichtenarchitektur im Hinblick auf ihre Leistungsfähigkeit, den Entwicklungsaufwand, sowie die Integration bestehender und sich entwickelnder Standards geeignet zuschneiden.
- Die vorgeschlagenen Migrationsstrategien erlauben einen „sanften“ Übergang von der bestehenden hin zu noch flexibleren, leistungsfähigeren und verständlicheren Architekturen.

Inhaltsverzeichnis

1	Einleitung	4
2	Technisches Umfeld	6
2.1	OSUN	6
2.2	SIP/PINT	7
2.3	Parlay	8
2.4	CORBA	9
3	Evaluierung von Software-Architekturen	10
3.1	Übersicht	11
3.2	Kriterienkatalog	12
4	Evaluierung der OSUN-Architektur	13
4.1	Übersicht	13
4.2	Bewertung	14
5	Architektur-Varianten	15
5.1	Basisarchitektur	15
5.2	Variante 1	17
5.3	Variante 2	18
5.4	Variante 3	19
5.5	Variante 4	20
5.6	Zusammenfassung und Bewertung	21
6	Migrationsstrategie	21
6.1	Phase 1: Übergang von Basisarchitektur zu Variante 1	22
6.2	Phase 2: Übergang von Variante 1 zu Variante 3	23
6.3	Phase 3: Übergang von Variante 3 zu Variante 2	24
6.4	Zusammenfassung	24
7	Zusammenfassung	25

1 Einleitung

In den letzten Jahren hat die Verbreitung und Nutzung von Datennetzen, insbesondere die des weltweiten Internet, einen sprunghaften Anstieg erfahren. Getrieben von technischer Innovation sowie wachsenden Anforderungen der Konsumenten und der Wirtschaft, vor allem im Hinblick auf deren fortschreitende Globalisierung, konnten sich in kürzester Zeit neuartige Dienste und Anwendungen wie das World Wide Web oder die Kommunikation über E-Mail auf dem Markt etablieren. Die hierfür entwickelten Technologien wie HTML [W3C00], Java [Sun00] oder CGI [CGI00] dienen als Basis für immer neue, fortschrittlichere Anwendungen, die traditionelle Geschäftsfelder und Arbeitsabläufe dramatisch verändern oder sogar gänzlich neue Märkte erschließen. Die wachsende wirtschaftliche Bedeutung [Sti00] sowie die kommende Integration von Internet-basierten Diensten und mobiler Kommunikation wird voraussichtlich zu einer Fortsetzung des starken Wachstums in diesem Bereich führen [Kur00].

Demgegenüber wächst der Markt für traditionelle Telekommunikationsdienste wie Telefonie oder Fax auf Basis leitungsvermittelter Netze (PSTN) deutlich langsamer [Maa99], während der anhaltende Preiskampf auf dem liberalisierten Telekommunikationsmarkt die Gewinne der Anbieter zunehmend schmälert [Sal99]. Der Bedarf an Sprachverbindungen zwischen zwei oder mehr Teilnehmern (Konferenz-Schaltung) über Telefone im Festnetz scheint in den westlichen Industrieländern weitgehend gesättigt und wird allenfalls indirekt durch das starke Wachstum der Mobilkommunikation positiv beeinflusst. Andererseits führte die jahrzehntelange Entwicklung und Erfahrung in diesem Bereich zu einer ausgesprochen hohen Qualität, Verfügbarkeit und Zuverlässigkeit der angebotenen Dienste. Darüber hinaus werden auch die notwendigen administrativen Aufgaben wie Teilnehmerverwaltung und Abrechnung von den großen Telekommunikationsanbietern wie AT&T, British Telecom oder MCI zuverlässig erfüllt.

Zukünftige, innovative Angebote verlangen zunehmend die möglichst nahtlose und transparente Integration von Daten-, Video- und Sprachdiensten über verschiedene Netzwerktechnologien. Hierdurch wird es beispielsweise dem Kunden ermöglicht, weiterführende persönliche Beratung und Unterstützung durch ein automatisch vermitteltes Telefonat mit einem Experten zu erhalten, dessen Nummer er vorher beispielsweise aus einer entsprechenden WWW-Seite oder einem elektronischen Telefonbuch ausgewählt hat („Click-to-Dial“). Ein weiteres Beispiel ist die Initiierung und Verwaltung von Telefonkonferenzen über den Computer des Anwenders („Click-to-Conference“), möglicherweise ergänzt durch gemeinsame, rechnergestützte Bearbeitung von Dokumenten durch die Konferenzteilnehmer („Shared Whiteboard“). Wünschenswert aus Kundensicht ist beispielsweise auch eine vereinheitlichte Behandlung von Nachrichten über Sprach-, Fax- und Datenkanäle, weitgehend unabhängig von Ort und technischer Ausstattung des Zugangs („Unified Messaging“).

Die oben aufgeführten Anwendungsszenarien für integrierte multimediale Dienste beschreiben nur einen kleinen Ausschnitt der denkbaren Möglichkeiten. Tatsächlich sind Art, Umfang und Bedeutung der zukünftigen Anwendungen für diese *Converged Services* gegenwärtig noch nicht abzuschätzen. Es handelt sich also um einen neuen, vielversprechenden und äußerst dynamischen Markt, der in den nächsten Jahren erschlossen wird [Ken99]. Deshalb sind die beteiligten Parteien, von unabhängigen Software Entwicklern über Telekommunikationsanbieter bis zu den Herstellern von Vermittlungsanlagen und Netzwerkgeräten, angehalten, rechtzeitig tragfähige Geschäftsstrategien und technische Lösungen zu entwickeln [Del00].

Vor diesem Hintergrund kommt der möglichst offenen, schnellen und einfachen Anwendungsentwicklung besondere Bedeutung zu. Nur wenn es gelingt, den Zugang zur zugrundeliegenden Kommunikations-Infrastruktur einer breiten Basis von Software-Herstellern zu ermöglichen, kann die Dynamik der Internet-Entwicklung auf diesen Bereich übertragen werden. Darüber hinaus ist die Nutzung der Infrastruktur und ihrer Kommunikationsdienste weitgehend unabhängig von der verwendeten Vermittlungstechnik und -hardware zu gestalten.

Hierbei verfolgen die Hersteller von Geräten für paketvermittelte Datennetze die Strategie, grundsätzlich jede Form der Kommunikation, insbesondere auch Sprache und Video, einheitlich über diese Netze abzuwickeln. Jedoch weisen die momentan verfügbaren Lösungen in diesem Bereich, etwa „Voice-over-IP“ [Com98], noch deutliche Mängel bei Qualität und Zuverlässigkeit auf [Maa99, Gri00]. Außerdem werden die eingesetzten Protokolle und Standards wie H.323 [Dat00] der International Telecommunication Union (ITU) oft als zu komplex

empfundene und führen so zu langen Entwicklungszeiten und teilweise inkompatiblen Anwendungen [Kar99]. Demgegenüber versuchen die etablierten Telekommunikationsanbieter ihre Anlagen mit dementsprechenden Schnittstellen auszustatten, um so externen Software Herstellern den Zugang zu hochwertigen Sprach- und Videodiensten zu ermöglichen. Daneben verfolgen sie das Ziel, auch selbst als Anbieter von innovativen, multimedialen Anwendungen aufzutreten, um sich so von der Konkurrenz zu differenzieren und am stark wachsenden Markt zu partizipieren [Kok99]. Diese Ziele führen unmittelbar zu neuen, erhöhten Anforderungen an die Hersteller von Hard- und Software für leitungsvermittelte Netze.

Die Siemens AG als führender Hersteller von Vermittlungssystemen für den Bereich PSTN begegnet dieser Herausforderung mit der Konzeption einer integrierten Plattform für die Anwendungsentwicklung, der sog. *Open Service Unit* (OSUN) [Lag00]. Die wesentlichen technischen und strategischen Ziele sind hierbei

- Entwicklung einer Grundmenge von konvergierten Diensten auf Basis bewährter Vermittlungsfunktionalität,
- Bereitstellung dieser Dienste für die modulare Anwendungsentwicklung durch Dritte über offene, einfach zu nutzende Schnittstellen,
- weitgehende Unabhängigkeit hinsichtlich herstellerspezifischer Hard- und Software, Protokolle und Vermittlungstechnik,
- sowie der umfassenden Berücksichtigung existierender Standards und Lösungen im Anwendungsumfeld.

Für die Erfüllung dieser Ziele ist eine tragfähige, flexible und leistungsfähige Software-Architektur der OSUN von entscheidender strategischer Bedeutung. Sie definiert die funktionalen Komponenten, ihre Schnittstellen und Interaktionen, die logische und technische Struktur des Gesamtsystems, sowie die Freiheitsgrade für die zukünftige Weiterentwicklung. Die geeignete Wahl einer Software-Architektur für OSUN ist somit ein Schlüsselfaktor für den Markterfolg der avisierten *Converged Services*.

Im Rahmen dieser Studie soll die gegenwärtige Software-Architektur, wie sie in der Pilotimplementierung der CtD-Applikation realisiert ist, hinsichtlich ausgewählter, als wesentlich erachteter Kriterien evaluiert werden. Die Ergebnisse dieser Evaluierung kontrastieren Stärken und Schwächen der Architektur aus externer Sicht und liefern somit kurzfristig einen Beitrag zur Qualitätssicherung des Projektes. Zudem können wertvolle Erkenntnisse und Anhaltspunkte für die mittel- und langfristige Konzeption der Architektur gewonnen werden. Diese weiterführenden Aspekte werden in Vorschlägen für Architekturvarianten und möglichen Migrationsstrategien dokumentiert.

Inhaltlich ist die vorliegende Studie wie folgt gegliedert:

- Kapitel 2 erläutert das technische Umfeld mit seinen wichtigsten Konzepten und schafft so die Grundlage für die weitere Diskussion.
- Kapitel 3 faßt grundlegende wissenschaftliche Erkenntnisse zur Evaluierung von Software-Architekturen zusammen und leitet das konkrete, für diese Studie gewählte Vorgehen sowie die angewandten Kriterien ab.
- Kapitel 4 beinhaltet die eigentliche Evaluierung der gegenwärtigen OSUN-Architektur hinsichtlich der vorher bestimmten Kriterien. Es erfolgt eine ausführliche Bewertung ihrer Stärken und Schwächen.
- Kapitel 5 leitet aus der Bewertung Vorschläge für mögliche Alternativen und Architekturvarianten ab. Hierbei werden Vor- und Nachteile der aufgeführten Lösungen diskutiert.
- Kapitel 6 behandelt Migrationsstrategien, die sich aus der Distanz zwischen gegenwärtiger Architektur und favorisierten Vorschlägen des vorherigen Kapitels ergeben. Hierfür wird insbesondere der nötige Aufwand abgeschätzt und das Entwicklungsrisiko berücksichtigt.

- Kapitel 7 faßt die wichtigsten Ergebnisse dieser Studie zusammen.

Abschließend läßt sich feststellen, daß die fortschreitende Integration von Informations- und Kommunikationstechnologie, gerade auch hinsichtlich einer Kombination mit mobilen Endgeräten, zur Schaffung eines dynamischen und lukrativen Marktes führt. Die Autoren der Studie hoffen, mit der vorliegenden Arbeit einen Beitrag zum Erfolg der Siemens AG in diesem interessanten Bereich leisten zu können.

2 Technisches Umfeld

In den folgenden Abschnitten gehen wir kurz auf das technische Umfeld der OSUN-Plattform ein. Neben der OSUN-Spezifikation erwähnen wir die Standards SIP/PINT, Parlay und CORBA.

2.1 OSUN

Die Spezifikation der „Open Service Unit“-Plattform entstand im Jahr 1999 im Bereich ICN WN CS der Siemens AG. Diese Arbeit folgte insbesondere der Erkenntnis, dass die Integration der diversen Kommunikationsnetze neue, innovative Dienste ermöglicht und somit gerade in der enorm wachsenden Anwendungslandschaft des Internets die Qualität der klassischen Telekommunikation einbringt.

Die Vision dieser Bemühungen umschließt einfache Dienste, wie beispielsweise „Click-to-Dial“, aber ebenso auch Bausteine komplexer Anwendungen, wie z.B. whiteboardunterstützte Videokonferenzen. Zu diesem Zweck müssen die Konzepte der zugrundeliegenden verbindungsorientierten Netzwerke geeignet auf die Anwendungsebene transferiert werden (etwa „Verbindung“ oder „Teilnehmer“).

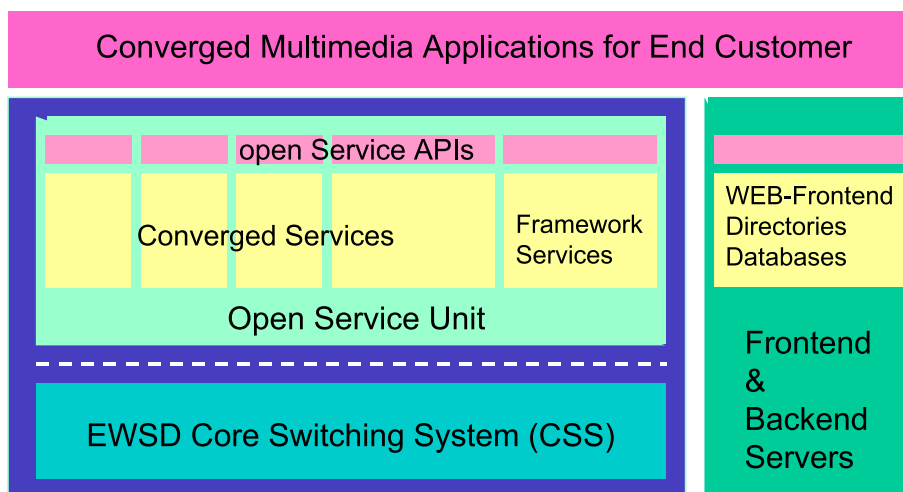


Abbildung 1: Übersicht der Open Service Unit

Abbildung 1 (aus [Lag00]) zeigt, wie sich die OSUN-Plattform in die Gesamtkonzeption eingliedert. Zentral ist die Anknüpfung an die Telekommunikations-Komponente EWSD, die als Dienstbringer das Zusammenspiel mit dem verbindungsorientierten Netzwerk organisiert. Aufbauend auf dieser Funktionalität wird eine Reihe von „converged services“ definiert, deren Dienste über entsprechende, offene Schnittstellen Anwendungen angeboten werden. Dienste eines Frameworks bieten übergreifende Funktionalität an — beispielsweise Mechanismen zur Abrechnung.

Querschnittsdienste, wie beispielsweise Telefonbücher oder auch allgemeine Informationsserver, sind als separate Säule in der Architektur vorgesehen (siehe Abbildung 1, rechts).

Dabei sind explizit Drittanbieter vorgesehen, um einerseits solche modernen Anwendungen zu erstellen und zu vertreiben aber auch um andererseits selbst neue Dienste in das bestehende Framework zu integrieren.

Weiterführende Informationen zur OSUN-Plattform und den damit verfolgten Zielen finden sich bei [Lag00] und [ZM00].

2.2 SIP/PINT

Mit der Verbreitung und dem wachsenden Erfolg des Internet liegt die Idee nahe [Lu 98], die Dienste dieses paketorientierten Netzwerkes mit denen des klassischen, verbindungsorientierten Netzwerkes der Telekommunikation zu neuen Anwendungen zu verknüpfen.

Aus einer Kooperation mehrerer Firmen, darunter auch Siemens, entstand Ende 1999 ein Vorschlag für ein geeignetes Service-Protokoll. Die Bezeichnung PINT steht für „PSTN Internet Interworking Service“ und deutet diese Integration von Internet und Telekommunikation an. Der Ablauf ist dabei stets der gleiche — wie schematisch in Abbildung 2 dargestellt.

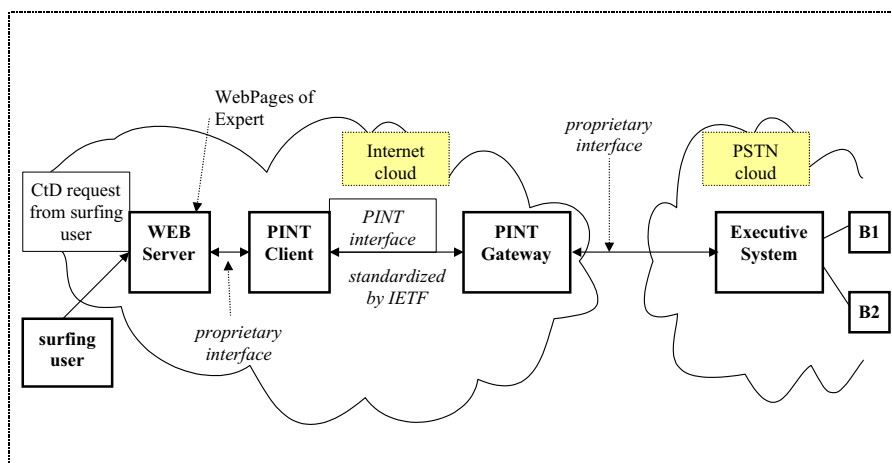


Abbildung 2: Schematische Darstellung eines PINT-Szenarios

Ein Anwender klickt beispielsweise im WWW auf einen Link, der vom jeweiligen Webserver in einen PINT Servicewunsch umgewandelt wird. Dieser kann nun an ein geeignetes PINT Gateway versandt werden — der Webserver agiert hierbei als PINT-Client. Das Gateway sorgt für die korrekte Weiterleitung in das Telekommunikationsnetz und auf diese Weise wird beispielsweise eine Verbindung geschaltet.

PINT sieht eine Reihe von einfachen Diensten vor, die als Vorstufe praktische Relevanz und Machbarkeit der angestrebten converged services demonstrieren sollen. Dazu gehört „Request to call“, also die Möglichkeit aus einem IP-Netz heraus eine Telefonverbindung aufzubauen. Ähnlich geartet ist „Request to fax“, mit dem ein Datenstrom an ein spezifisches Faxgerät gesendet werden kann. Schließlich ist „Request to hear content“ vorgesehen, um eine Telefonverbindung aufzubauen und eine vordefinierte Sprachmitteilung abzuhören.

Um eine Integration des Internets mit den Netzwerken der klassischen Telekommunikation zu erreichen, bedarf es eines verbindungsorientierten Protokolls auf Seiten des klassisch paketorientierten IP-Netzes. Aus diesem Grund stützt sich die Spezifikation von PINT auf dem noch recht jungen SIP-Standard (*Session Initiation Protocol*) ab. Auf diese Weise erreicht man ein durchgängiges Verbindungskonzept in beiden Welten.

SIP entstand im März 1999 als Vorschlag für einen Internet-Standard, der es erlaubt, Verbindungen — sogenannte „Sessions“ — zwischen zwei oder auch mehreren Teilnehmern innerhalb des Internets zu erzeugen, zu verändern und letztlich zu beenden. SIP ist hardware-unabhängig und erweiterbar, und somit ein idealer Partner für PINT. Weitere Besonderheiten des Protokolls bestehen in der Fähigkeit, zu bestehenden Verbindungen weitere Teilnehmer einladen zu können, sowie in der Unterstützung der Mobilität einzelner Benutzer. Nicht zuletzt ist SIP durch seine Einfachheit und Klarheit leicht verständlich und anwendbar.

Durch die Verbindung mit PINT ergibt sich allerdings das Problem, dass nun SIP nicht mehr eine Verbindung zwischen den eigentlichen Kommunikationspartnern aufbaut, sondern lediglich zwischen einem Teilnehmer

und dem nächstgelegenen PINT-Server. Es stellt somit nur die logische Verbindung auf Seiten des IP-Netzes dar. So kann eine Verbindung zwar zustande kommen — der eigentliche Kommunikationspartner hat jedoch eventuell noch gar nicht abgehoben. Somit erklärt sich, dass PINT eine geeignete Erweiterung für SIP spezifiziert. So können beispielsweise auch Fehlermeldungen des PINT-Protokolls geeignet im Rahmen von SIP repräsentiert und gehandhabt werden.

Ein Ausschnitt eines typischen PINT-Aufrufs ist im folgenden Beispiel dargestellt (aus [PC99]). Der Benutzer „John Jones“ möchte mit „Mary James“ verbunden werden. Der entsprechende Header ähnelt in seinem Aufbau dem weit verbreiteten Standard HTTP:

```
INVITE sip:marketing@pint.mailorder.com SIP/2.0
Via: SIP/2.0/UDP 169.130.12.5
From: sip:john.jones.3@chinet.net
To: sip:mary.james@mailorder.com
Call-ID: 19971205T234505.56.78@pager.com
Subject: Defective Ironing Board - want refund
Content-type: application/sdp
```

Im Umfeld dieser Studie ist insbesondere die Definition einer möglichen Anbindung von PINT an das World Wide Web von Interesse, die in der PINT-Spezifikation vorliegt. Dabei empfängt ein Webserver Anfragen des Benutzers, übernimmt die Suche nach einem geeigneten PINT-Gateway, und leitet die Anfrage an dieses Gateway weiter. Dem PINT-Gateway gegenüber nimmt der Webserver die Rolle des PINT-Clients ein.

Im Gegensatz zu vielen Internet-Protokollen spielt in der Telekommunikation Sicherheit eine gesteigerte Rolle. Neben der Ausfallsicherheit geht es vor allem darum, dass die angebotenen Dienste nicht von Dritten mißbraucht oder auf Kosten anderer genutzt werden. So gilt es beispielsweise zu verhindern, dass eine Verbindung an einen anderen als den gewünschten Empfänger weitergeleitet wird. Die Spezifikation von PINT beinhaltet eine Reihe von Vorschlägen und Lösungsansätzen zu relevanten Themen, wie z.B. Authentisierung, Authorisierung und Sicherheit [PC99].

2.3 Parlay

Die *Parlay Group* [PG00b] wurde im April 1998 von British Telecom, Ulticom, Microsoft, Nortel Networks und Siemens gegründet, um integrierte, offene Schnittstellen für die schnelle Entwicklung von multimedialen Anwendungen im Telekommunikations- und Netzwerkbereich zu etablieren. Grundsätzlich verfolgt dieser Ansatz somit ähnliche Ziele wie SIP/PINT (siehe 2.2) oder die OSUN-Plattform (siehe 2.1). Jedoch ist Parlay als umfassender Industriestandard zu betrachten, insbesondere nachdem in 1999 mit AT&T, Cegetel, Cisco, Ericsson, IBM und Lucent weitere bedeutende Unternehmen aus dem Anwendungsbereich der Gruppe beigetreten sind.

Das *Parlay API* [PG00a] spezifiziert eine Reihe von objektorientierten Schnittstellen für Dienste, über die mittels COM, CORBA oder Java auf die Funktionalität des darunterliegenden Netzwerks zugegriffen werden kann. Hierbei unterscheidet Parlay zwischen sog. *Framework* und *Service Components*, wie aus der Architektur von Parlay ersichtlich ist (siehe Abbildung 3). Der technologie- und herstellerabhängige Zugriff auf unterschiedliche Netzwerk-Ressourcen ist ausdrücklich nicht Gegenstand der Parlay Spezifikation.

Die Komponenten des Frameworks bieten unterstützende Funktionalität, die von nahezu allen denkbaren Anwendungen benötigt wird, beispielsweise Authentifizierung, Authorisierung oder Protokollierung. Darüber hinaus werden Schnittstellen für die Verwaltung von Diensten (Suche, Registrierung und Inanspruchnahme) sowie das nötige System-Management spezifiziert. Demgegenüber erlauben eine Reihe von definierten Schnittstellen der Service Components wie *Call Control*, *Conferencing Call Control*, *Generic Messaging* oder *User Location* die Entwicklung von komplexen und leistungsfähigen Netzwerk-Anwendungen.

Weiterhin besteht die Möglichkeit, zusätzliche, von Dritten spezifizierte und realisierte Dienste in das Framework zu integrieren. Hierbei erfolgt eine Authentifizierung und Registrierung der Dienste über entsprechen-

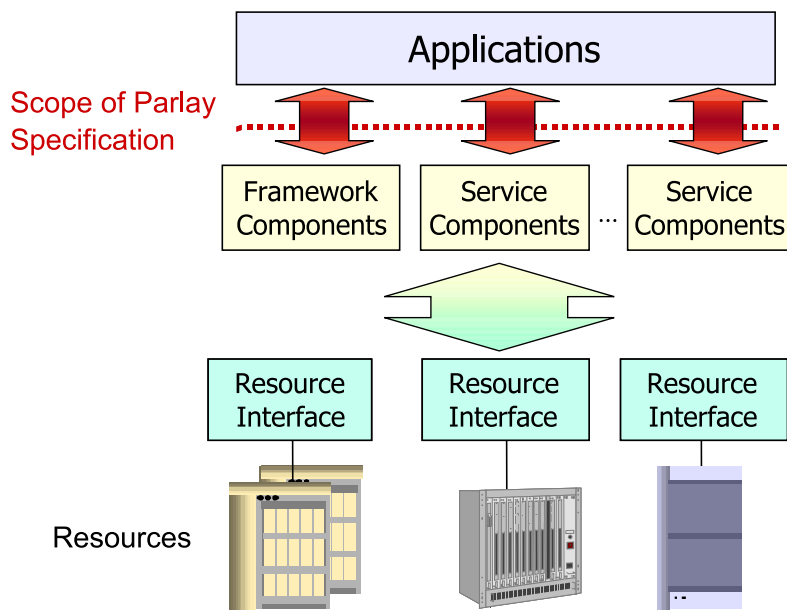


Abbildung 3: Architektur des Parlay Netzwerk-API (aus [PG00a])

de Schnittstellen. Anschließend können Anwendungen die vorhandenen Dienste flexibel über eine Liste von erwünschten Eigenschaften abfragen (*Service Discovery*) und nach einer Authorisierung schließlich benutzen. Die bisher spezifizierten Schnittstellen werden von der Parlay Group kostenlos und lizenzfrei als UML Modelle veröffentlicht. Darüber hinaus sollen verfügbare Abbildungen in die IDL-Dialekte von Microsoft und CORBA die praktische Umsetzung von Parlay erleichtern. Allerdings existiert gegenwärtig keine vollständige Referenzimplementierung des gesamten API.

2.4 CORBA

Die *Common Object Request Broker Architecture* (CORBA) der *Object Management Group* definierte ursprünglich eine technische Lösung zur transparenten Kommunikation verteilter, objektorientierter Anwendungen in einem Netzwerk [OMG00]. Mittlerweile wurde diese Spezifikation zu einer umfassenden Referenz-Architektur, der sog. *Object Management Architecture* (OMA) [OMG97], für die möglichst einfache Entwicklung verteilter, portabler und untereinander kompatibler Anwendungen erweitert (siehe Abbildung 4).

Neben der grundlegenden Kommunikations-Infrastruktur spezifiziert OMA eine Reihe von allgemein verwendbaren Diensten, die sog. *Object Services*, welche in nahezu allen verteilten Anwendungen benötigt werden. Beispiele hierfür sind Namens- oder Verzeichnisdienste aber auch Schnittstellen für Persistenz- und Transaktions-Management. Die sog. *Common Facilities* repräsentieren spezifische Objektmodelle, die für horizontale, benutzerorientierte Bereiche der Anwendungsentwicklung von Bedeutung sind, beispielsweise der Umgang mit zusammengesetzten Dokumenten oder die Online-Hilfe einer Anwendung. Für bedeutsame Anwendungsdomanen wie Finanz- oder Gesundheitswesen werden von der OMG auch ausgewählte Schnittstellen der sog. *Application Objects* standardisiert.

Die Kommunikation innerhalb des Systems erfolgt über einen zentralen Vermittler, den sog. *Object Request Broker* (ORB), der die Kommunikationspartner im Netzwerk lokalisiert und Anfragen geregelt weiterleitet. Die Zielsetzung ist hierbei eine möglichst transparente Kommunikation zwischen verteilten Objekten, die in unterschiedlichen Programmiersprachen realisiert sind. Hierfür werden die Schnittstellen der beteiligten Objekte in der technologie-unabhängigen Sprache *Interface Definition Language* (IDL) beschrieben und über sog. *Language Bindings* in die jeweilige Programmiersprache übersetzt. Die bei der Übersetzung automatisch gene-

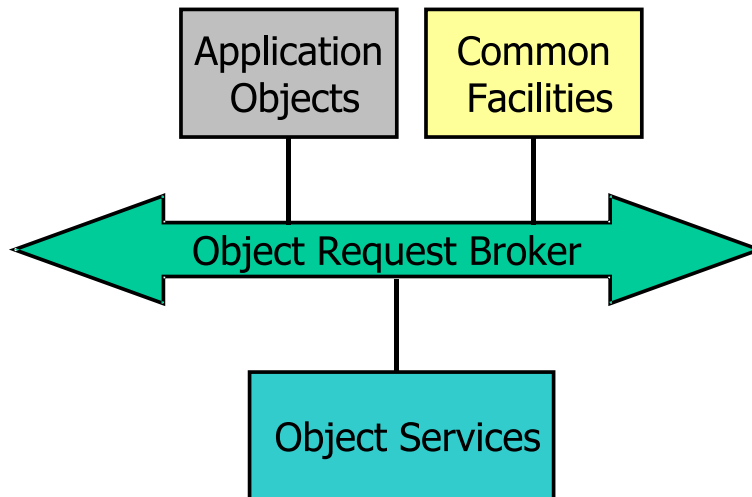


Abbildung 4: Übersicht der Object Management Architecture

rierten Proxy-Objekte dienen als lokaler Kommunikationspartner und leiten Methodenaufrufe über den ORB an das entsprechende, entfernte Objekt weiter.

Je nach Rolle der beteiligten Kommunikationspartner wird zwischen *Clients* und *Servern* unterschieden: Clients nehmen Dienste von eindeutig identifizierten Servern in Anspruch, während Server die Anfragen von mehreren, zunächst anonymen Clients bearbeiten. Hierfür können Server zur Leistungssteigerung mit mehreren Threads arbeiten oder sind sogar selbst mehrfach realisiert.

Mittlerweile existieren zahlreiche freie und kommerzielle Implementierungen des aktuellen CORBA-Standards in der Version 2.3. Sie unterscheiden sich neben Preis, Performanz oder Zuverlässigkeit v.a. in den verfügbaren Language Bindings sowie den realisierten Common Object Services. Jedoch spezifiziert CORBA auch ein Protokoll für die Kommunikation zwischen unterschiedlichen ORBs, so daß die Interoperabilität zwischen Systemen verschiedener Hersteller weitgehend gewährleistet ist.

3 Evaluierung von Software-Architekturen

Software Architektur gewinnt als eigenständige Disziplin des modernen Software-Engineerings zunehmend an Bedeutung. Mit der stetig ansteigenden Komplexität heutiger Systeme wird es immer wichtiger, durch geeignete Abstraktionen verschiedene, konsistente Perspektiven auf ein System anzubieten und somit ein umfassendes System-Verständnis zu ermöglichen. Eine Architektur ist in diesem Sinne das „Minimum an Informationen, das man benötigt, um die Funktionsweise des Systems zu verstehen und erklären zu können“ [Kru99].

Die Erfahrungen der letzten Jahre bestätigen die Bedeutung einer „guten“ Software-Architektur für den Erfolg grösserer Software-Projekte. Trotz dieser inzwischen anerkannten Tatsache wird die Disziplin „Architektur“ im Alltag der Softwareentwicklung noch nicht ausreichend angewandt. Zu den wichtigsten Gründen dafür gehören insbesondere:

- Unklarheit über Architektur-Konzepte und ihre Anwendung.
- Schlechte oder oft sogar fehlende Möglichkeiten zur Repräsentation und Kommunikation einer Architektur.
- Fehlender Einbezug architektur-relevanter Aktivitäten in den firmenspezifischen Entwicklungsprozess.

Ansätze zur Lösung dieser Probleme sollen an dieser Stelle kurz vorgestellt und diskutiert werden. Sie werden im Weiteren hilfreich sein, um eine Evaluierung der bestehenden Software-Architektur vornehmen und geeignete Varianten bewerten zu können.

Schlüssige Definitionen der relevanten Konzepte finden sich bei Mary Shaw und David Garlan [SG96]: Architektur beschäftigt sich ihnenzufolge mit der Organisation eines Systems. Das beinhaltet neben der Auswahl struktureller Elemente („Komponenten“) und ihrer Schnittstellen ebenfalls das Verhalten des Systems als genau spezifiziertes Zusammenspiel seiner Teile. Weiterhin umfaßt der Architekturbegriff bei Bedarf die Komposition solcher Teile zu jeweils größeren Subsystemen. Schließlich ist für das Verständnis einer Architektur der zugrundeliegende „Architekturstil“ bedeutsam. Dieser Stil stellt eine Richtlinie für die Organisation von Komponenten, Schnittstellen und Interaktionen dar.

Ein Hauptproblem liegt immer noch in der adäquaten Repräsentation von Software-Architekturen [BHKS97]. Gängige Architekturbeschreibungssprachen (ADL : *Architecture Description Languages*), wie beispielsweise RAPIDE [LKA⁺95] oder Unicon [SDK⁺95] sind zwar sehr umfassend und erlauben mathematisch-präzise Spezifikationen, sie sind andererseits jedoch wegen des zugrundeliegenden formalen Mechanismus schlecht zu handhaben und zu kommunizieren. In dieser Studie beschränken wir uns dagegen zur Darstellung der Architekturvarianten auf sehr einfache Diagramme, in denen Rechtecke sowohl Komponenten als auch ihre Schnittstellen repräsentieren. Verhaltensspezifikationen werden durch einfache Sequenzdiagramme skizziert, wie sie im Rahmen der UML (siehe [BRS97, RJB98]) definiert sind. Zwar unterstützen die graphischen Beschreibungstechniken der UML derzeit Architekturbeschreibungen nicht adäquat; es gibt jedoch Ansätze, welche die UML-Notationen hinsichtlich der Beschreibung von Software-Architekturen evaluieren und entsprechend erweitern (siehe [EM99]).

Im Umfeld der Arbeiten zur UML entstand ebenfalls der „Rational Unified Process“ [Kru99]. Ähnlich zu anderen modernen Vorgehensmodellen stellt dieses Prozessmodell die Architektur des Systems in das Zentrum der Entwicklung. Unterschiedliche Beteiligte haben hierbei unterschiedliche Sichten auf die Architektur - sie ist nicht länger ein „flaches“ Gebilde, sondern kann aus diversen Blickpunkten unterschiedliche Aspekte dokumentieren.

Erst mit der Einführung und dem richtigen Zusammenspiel dieser drei Elemente — Konzepte, Beschreibungstechniken und Prozessunterstützung führt Software-Architektur zum versprochenen Erfolg.

3.1 Übersicht

Durch die Tatsache, dass die Architektur eines Systems entscheidend den Erfolg der Entwicklung beeinflusst, empfiehlt es sich bereits frühzeitig das Design einer Begutachtung durch Dritte zu unterziehen. Für das Vorgehen bei einer solchen Evaluierung existieren jedoch keine allgemein anerkannten Vorgaben — dies hängt zu sehr von der aktuellen Situation des jeweiligen Unternehmens ab. Eine Reihe von Erfahrungswerten können jedoch die Eckpunkte der Evaluierung vorgeben und somit einen erfolgreichen Pfad abstecken. Im folgenden stellen wir eine Reihe von „best practices“ vor, die in weiten Teilen auf [ABC⁺97] zurückgehen.

Nach einer Abwägung der Kosten einer Architekturevaluierung gilt es demnach insbesondere, sich über die eigentliche Ziele klar zu werden. Eine beispielhafte Auswahl an Möglichkeiten ist die folgende:

Finanziell: Die Erfahrung zeigt, dass eine frühzeitige Evaluierung der Architektur spätere Kosten vermeiden kann. Insbesondere steigen die Kosten für das Korrigieren von fehlerhaften Designentscheidungen an, je später im Projekt sie behoben werden. So ist es zum Beispiel extrem teuer, ein bereits im Einsatz befindliches System anhalten und verbessern zu müssen.

Verbessertes Verständnis: Einer der grössten Vorteile einer Architektur-Evaluierung liegt in der Tatsache, dass Projektbeteiligte „ihre“ Ergebnisse kommunizieren können müssen. Das erfordert eine intensive Auseinandersetzung mit der Architektur und den jeweiligen Designentscheidungen. Durch die Diskussion im Verlauf der Evaluierung wird zudem aus vielleicht ungewohnten Blickpunkten auf den Entwurf gesehen und noch kaum berücksichtigte Aspekte werden beleuchtet. Nicht zuletzt verbessert sich die Dokumentation des Systems.

Problemerkennung: Meistens werden während der Evaluierung Probleme mit der bestehenden Architektur erkannt und entsprechende Vorschläge zur Behebung diskutiert.

Verdeutlichung der Anforderungen: Da die Anforderungen an das System die Grundlage für die Bewertung seiner Architektur darstellen, ergibt sich eine intensive Auseinandersetzung mit den Anforderungen und ihren jeweiligen Prioritäten und Motivationen. Insbesondere werden Konflikte aufgedeckt und die entsprechenden Trade-offs diskutiert.

Unternehmenskultur: Je öfter die Mitarbeiter eines Unternehmens an einer Evaluierung teilhaben, umso gewohnter werden die jeweiligen Abläufe. Auf diese Weise gehen die jeweiligen Abläufe in die Unternehmenskultur ein und lassen sich weiter optimieren und reglementieren.

3.2 Kriterienkatalog

Um die Bewertung der bestehenden Architektur und der vorgeschlagenen Varianten vorzubereiten, wird hier zunächst ein Kriterienkatalog aufgestellt. Dieser reflektiert strategische, technische sowie fachliche Anforderungen an die Architektur und beurteilt, inwieweit die jeweiligen Charakteristika ausschlaggebend für den Erfolg des Projektes sind. Er stellt somit die Grundlage für jegliche Bewertung dar und ist zudem stark auf die spezielle Situation im Projekt zugeschnitten.

Im weiteren präsentieren wir einen Kriterienkatalog, der hauptsächlich aus den Ergebnissen der zwei Workshops mit Siemens ICN [Wor00a, Wor00b] sowie unseren Erfahrungen bei vergleichbaren Studien zur Evaluierung von Software-Architekturen (vgl. [BRSV98, BHB⁺97, BRS99]) stammt:

Erweiterbarkeit: Spätere Erweiterungen oder Veränderungen eines Software-Systems sind sehr oft mit hohem Aufwand verbunden. Gerade deshalb ist es ausgesprochen wichtig, frühzeitig zu wissen, mit welchen Varianten oder Erweiterung in der Zukunft zu rechnen sein wird und die Architektur darauf entsprechend vorzubereiten. Konkret bedeutet dies, dass eine Reihe von Erweiterungspunkten vorgesehen sind, die aufzeigen wie und in welcher Form der Ausbau des Systems stattfinden kann.

Verständlichkeit: Wie bereits erwähnt besteht eine bedeutende Rolle der Architektur darin, die Organisation des Systems im allgemeinen und das Zusammenspiel der einzelnen Komponenten im besondern zu vermitteln. Wird dieses Ziel verfehlt, steigt der Aufwand für die Realisierung ebenso wie der für die Wartung. Ein üblicher Ansatz liegt im Prinzip von „Teile und Herrsche“. Es gilt also ein besonderes Augenmerk der Frage, ob die Aufteilung des Systems in einzelne Komponenten so geartet ist, dass die Menge der Abhängigkeiten untereinander minimiert wird. Der Erfolg einer solchen Architektur liegt in „schmalen“ Schnittstellen und damit in einer besseren Strukturierung und einem gesteigerten Verständnis.

Integration von Standards: Ausschlaggebend für die praktische Tauglichkeit und somit den Markterfolg eines Software-Systems ist in den meisten Fällen die Frage, wie einfach sich die Nutzung (dies umfasst im Fall der OSUN-Plattform insbesondere die Erstellung portabler Anwendungen durch Dritte) gestaltet. Dies führt fast unmittelbar zur Definition der Schnittstellen. Proprietäre Spezifikationen erfordern maßgeschneiderte Lösungen sowohl von Siemens selbst als auch von Drittanbietern, deren Produkte auf OSUN aufsetzen sollen. In der gegebenen Situation (siehe Kapitel 1) und gerade im Bereich der Telekommunikation erscheint die Verwendung von standardisierten Schnittstellen essentiell.

Skalierbarkeit, Durchsatz: Aus den Gesprächen mit Siemens ICN hat sich ergeben, dass die Leistungsfähigkeit des Systems, also insbesondere der Durchsatz der abzuarbeitenden Anfragen entscheidend für die Praxistauglichkeit des Systems ist. Sinnvolle Aussagen über die Leistungsfähigkeit des Architekturkonzepts erfordern detaillierte Untersuchungen und oftmals Messungen an geeigneten technischen Prototypen. Aus diesem Grund werden Performanzaspekte in dieser Studie nicht berücksichtigt.

4 Evaluierung der OSUN-Architektur

In diesem Kapitel soll die bestehende Architektur der Pilotimplementierung anhand des vorgestellten Kriterienkatalogs beleuchtet und bewertet werden. Grundlage ist hierbei ein UML-Klassendiagramm, das uns von Siemens ICN zur Verfügung gestellt und erläutert wurde. In verkleinerter Form ist es auf der linken Seite von Abbildung 5 dargestellt.

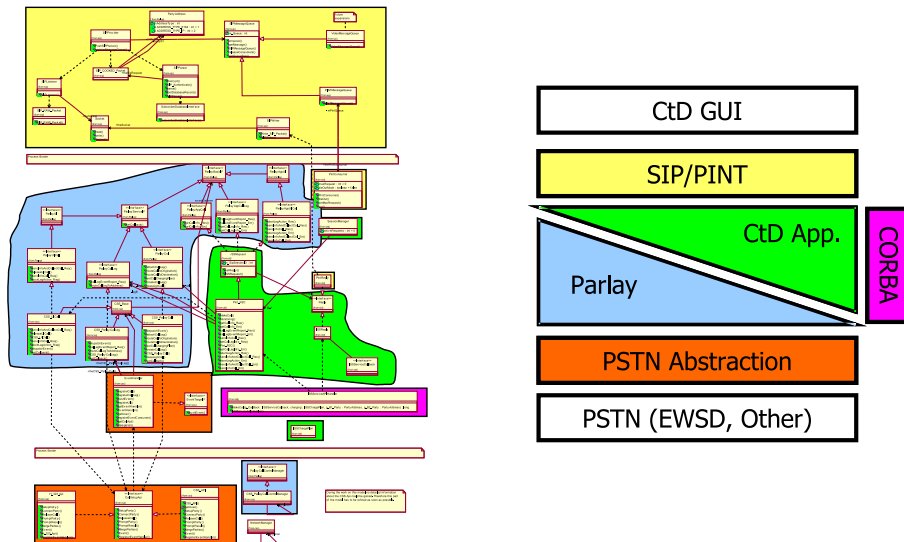


Abbildung 5: Strukturierung der existierenden Architektur

4.1 Übersicht

Ein erstes Augenmerk richtet sich stets auf die Form der Darstellung und die Angemessenheit der verwendeten Mittel. Nicht zuletzt aufgrund der guten Werkzeugunterstützung sind die hier eingesetzten Klassendiagramme der UML eine in Teilen geeignete Beschreibungstechnik für die Darstellung der Komponenten und Schnittstellen einer Architektur. Oft jedoch wird der Designer zu einer detaillierteren Darstellung verleitet, als sie notwendig wäre. Auch in der vorliegenden Graphik sind Details aufgeführt, die auf der Ebene der Software-Architektur einen zu hohen Detaillierungsgrad aufweisen. Dazu gehören beispielsweise konkrete Lösungsansätze, wie die Messagequeue in der SIP-Komponente, Prozeßgrenzen aber auch Schnittstellenfeinheiten wie die „legs“ des Parlay-Frameworks. Im ungünstigsten Fall werden Klassendiagramme so komplex, dass ein Verständnis der eigentlichen Architektur erheblich erschwert wird.

Insgesamt lassen sich die einzelnen Komponenten der OSUN-Architektur gut erkennen — zur Verdeutlichung sind sie in Abbildung 5 farblich unterlegt und auf der rechten Seite vereinfacht dargestellt. Es ergibt sich somit eine etwas detailliertere Schichten-Architektur als ursprünglich durch die zwei Prozeßgrenzen angedeutet :

- Die *Hardware Abstraction* stellt eine gemeinsame Schnittstelle für unterschiedliche Varianten der zugrundeliegenden *Hardware* dar. Angedacht ist neben der EWSD-Plattform auch die Unterstützung eines leitungsorientierten Protokolls in einem IP-Netzwerk, wie es durch den H323-Standard spezifiziert ist.
- Das Zusammenspiel einer durchstichartigen *Parlay*-Implementierung mit der *Click-to-Dial*-Applikation stellt die eigentliche Realisierung dieses „converged services“ dar.
- Die *SIP/PINT*-Schicht übernimmt die Umsetzung der Serviceanfragen von Seiten der Anwendung (symbolisiert durch eine *Click-to-Dial GUI*) auf die Servicerealisierung.

Die besondere Charakteristik einer Schichtenarchitektur und der Grund für ihre weite Verbreitung liegt in der starken Kapselung der einzelnen Komponenten. Das bedeutet einerseits, dass klare Schnittstellendefinitionen den Zugang zu einer Schicht genau spezifizieren, und andererseits, dass eine Schicht nur auf ihre direkt benachbarten Schichten zugreifen darf. Auf diese Weise ist sichergestellt, dass die Abhängigkeiten zwischen den Komponenten der Architektur „lokal“ begrenzt bleiben. Je geringer die Abhängigkeiten zwischen den Komponenten, desto leichter lassen sich später einmal einzelne Schichten austauschen oder erweitern. Wie bei Standardarchitekturen im Umfeld betrieblicher Informationssysteme sind auch im Telekommunikationsbereich Komponenten mit Querschnittsaufgaben *außerhalb* der eigentlichen Schichtung vorstellbar; dies betrifft etwa Komponenten zur Fehlerbehandlung oder zur Authentisierung.

Diese strenge „Trennung der Zuständigkeiten“ ist in der vorliegenden Architektur noch nicht mit ihrer ganzen Konsequenz durchgeführt. So finden sich noch Bestandteile der Hardware-Schnittstelle CSS in Parlay-Komponenten (z.B. `CSS_Parlay_Call`), obwohl durch die `CallSetupAPI` eigentlich von dieser Schnittstelle abstrahiert sein sollte. Ein anderes Beispiel ist die Klasse `PintReply`, die ihren Ursprung offensichtlich in der SIP/PINT-Schicht hat, jedoch ein `Reply`-Interface der darunterliegenden Schicht realisieren muß. Auch die Lösung, über CORBA einen „Quereinstieg“ (siehe Abbildung 5) in die Serviceschicht zu erlauben, erscheint unkonventionell. Diese besondere Schnittstelle ist somit sehr eng verwoben mit der aktuellen Realisierung — ein späterer Umstieg z.B. auf ein anderes Serviceframework würde die erneute Implementierung dieser Schnittstelle erforderlich machen. Oft wird

4.2 Bewertung

Ausgehend von dem in Abschnitt 3.2 angegebenen Kriterienkatalog ergibt sich eine Bewertung der vorliegenden Architektur:

Verständlichkeit: Das vorliegende Klassendiagramm ist zwar zu detailreich, insgesamt jedoch prinzipiell geeignet für diesen Verwendungszweck — insbesondere, da sich der Umfang der Architektur des Beispieldienstes „CtD“ in Grenzen hält. Ein Klassendiagramm kann allerdings nur *ein* Fundament einer Architekturbeschreibung darstellen. Für ein Verständnis des richtigen Zusammenspiels und der entsprechenden Zusammenhänge ist es unerlässlich, weitere Diagramme zu erstellen. Der Vorteil liegt darin, bei der Darstellung eines ausgewählten Aspekts von hierbei unrelevanten, oft technischen Details zu abstrahieren und somit die Komplexität zu reduzieren. Mit passenden Beschreibungstechniken könnte so das Zusammenspiel zwischen Click-to-Dial-Applikation und Parlay-Framework skizziert werden. Andere könnten darstellen, wie sich die Abarbeitung einer Serviceanfrage durch alle Schichten zieht.

Erweiterbarkeit: Durch die Wahl einer Schichtenarchitektur ist es auch hier möglich, einzelne Schichten auszutauschen oder mehrere alternativ anzubieten. Weitere Varianten der Erweiterbarkeit sind jedoch noch nicht klar abzuschätzen. Sicher ist lediglich, dass zu einem späteren Zeitpunkt die Realisierung mit weiteren Diensten angereichert werden soll (z.B. Click-to-Conference). Inwieweit dies zusätzlichen Anpassungsaufwand erfordert, läßt sich nicht ohne weiteres beurteilen. Sollte später einmal die SIP/PINT-Schnittstelle durch ein anderes Protokoll ersetzt werden, wären jedoch durch die starke Verstrickung beider Schichten an dieser Stelle weitreichende Modifikationen erforderlich. Ein in dieser Hinsicht geeigneterer Ansatz wird in Kapitel 5 vorgestellt.

Integration von Standards: Die vorliegende Architektur baut geschickt auf aktuellen Standards der Telekommunikationsbranche sowie des Internet Standardisierungskomitees auf. Dadurch wird insbesondere die praktische Relevanz der OSUN Plattform gewährleistet. Für eine weitergehende Unterstützung etwa von Parlay fehlen jedoch geeignete (abstrakte) Schnittstellen in der Architektur. Dies erschwert beispielsweise flexibles Reagieren auf zukünftige Entwicklungen im Bereich von Standards zur Kopplung von Internet und PSTN-Technologien.

Im Großen und Ganzen läßt sich der bewertete Ansatz als geeignet und durchführbar bezeichnen. Die eingeschlagene Richtung sollte jedoch noch konsequenter verfolgt werden. Vorschläge hierfür werden wir im folgenden diskutieren.

5 Architektur-Varianten

In diesem Kapitel werden auf Grundlage des thematischen und technischen Umfelds (siehe Kapitel 1 und 2), sowie der Ergebnisse der Evaluierung der gegenwärtigen Software-Architektur aus Kapitel 4 eine Anzahl von Vorschlägen für mögliche Alternativen bzw. Varianten erarbeitet. Hierbei sollen die Stärken des gegebenen Ansatzes deutlicher herausgearbeitet werden und Potential zur weiteren Aufwertung der Architektur hervorgehoben werden, um die in dieser Studie berücksichtigten Hauptanforderungen (siehe Kapitel 3) möglichst umfassend zu erfüllen.

Ausgehend von einer grundlegenden Basisarchitektur, die sich weitgehend an den bestehenden Verhältnissen orientiert, werden im Folgenden vier Varianten näher betrachtet. Diese unterscheiden sich hauptsächlich durch den Grad der Integration bestehender Standards, ihrer Komplexität bzw. dem benötigten Implementierungsaufwand, sowie der geschätzten Leistungsfähigkeit im Hinblick auf zukünftige Anforderungen. Eine abschließende Bewertung der vorgestellten Varianten bereitet die Diskussion über mögliche Migrationsstrategien in Kapitel 6 vor.

5.1 Basisarchitektur

Der gegenwärtige Stand der OSUN-Architektur (siehe Kapitel 4), empirische Erkenntnisse sowie die verfügbare Dokumentation [Lag00] legen die Wahl einer Schichtenarchitektur als grundlegende, logische Architektur nahe. Hierbei können folgende Aspekte als allgemeine Vorteile einer solchen Architektur angeführt werden:

- Die Definition der Schichten erlaubt eine klare Festlegung der Aufgaben, welche ihr zugeordnete Komponenten zu erfüllen haben.
- Die möglichen Interaktionen zwischen Schichten bzw. ihnen zugeordneten Komponenten sind eindeutig festgelegt und von weitgehend lokaler Natur. Jede Komponente einer gegebenen Schicht kann ausschließlich mit Komponenten der gleichen, einer direkt übergeordneten oder direkt untergeordneten Schicht interagieren.
- Neue, später hinzugefügte Komponenten können verhältnismäßig einfach den bestehenden Schichten zugeordnet werden, falls deren Definition problemgerecht vorgenommen wurde.
- Im praktischen Einsatz haben sich Schichtenarchitekturen sowohl im Telekommunikationsbereich, etwa im Fall der OSI-Referenzarchitektur der ISO [ISO00], als auch bei betrieblichen Informationssystem bewährt.

Demgegenüber muß allgemein eine Reihe von Nachteilen bzw. möglichen Risiken berücksichtigt werden:

- Durch die starre Festlegung der Kommunikationsgrenzen muß insbesondere bei großer Anzahl der definierten Schichten mit einem erhöhten Kommunikations- und Verwaltungsaufwand gerechnet werden.
- Die Repräsentation von gleichen oder sehr ähnlichen Konzepten in verschiedenen Schichten kann zu Redundanz und Problemen der Konsistenzsicherung führen, falls die spezifizierten Aufgaben der Schichten nicht deutlich genug differenziert sind.
- Eine nicht problemgerechte Definition der Schichten kann die Integration von existierenden Komponenten oder Lösungen behindern.

Im Hinblick auf die in dieser Studie vorrangig betrachteten Anforderungen (siehe Kapitel 3) überwiegen jedoch die genannten Vorteile bei weitem. Die in Abbildung 6 vorgestellte Basisarchitektur weist eine übersichtliche, logische Strukturierung auf, erleichtert die flexible Einbindung von neuen Diensten und Schnittstellen, und ermöglicht die einfache Integration von bestehenden Standards und Lösungen. Daneben bietet sie genügend Freiraum für eine variable Geschäftsstrategie der Siemens AG, wie später noch ersichtlich wird.

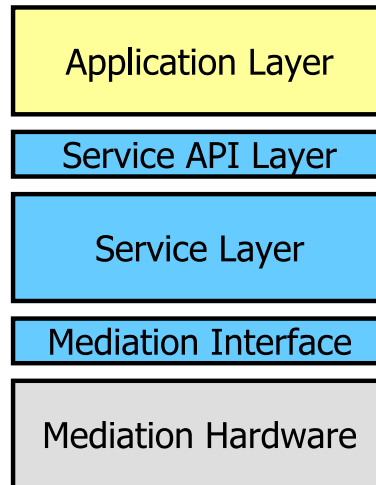


Abbildung 6: Grundlegende Schichtenarchitektur

Zur Erläuterung der vorgeschlagenen Lösung ist eine präzise Abgrenzung der verwendeten Begriffe dienlich. Als **System** wird im Folgenden die Gesamtheit aus Hard- und Software bezeichnet, die in Kombination dem Anwender einen zusätzlichen Nutzen durch Integration von paket- und leitungsvermittelten Netzwerken anbietet. Die **Anwendung** (engl. *application*) ist derjenige Teil des Systems, welcher dem Anwender unmittelbar zugänglich ist. Sie beinhaltet insbesondere die Benutzerschnittstelle, sowie fachliche Logik, um verschiedene Basisdienste gemäß der geforderten Funktionalität zu integrieren. Ein **Dienst** (engl. *service*) ist ein Bestandteil des Systems, der grundlegende Funktionalität kapselt und den Anwendungen über Programmierschnittstellen (engl. *application programming interface*, API) zur Verfügung stellt¹.

Tatsächlich ist die Abgrenzung zwischen Anwendung und Dienst abhängig von der Interpretation des Begriffs „grundlegende Funktionalität“ : die zunächst sehr spezifische Funktionalität einer gegebenen Anwendung kann sich im Verlauf der Systemnutzung als so grundlegend herausstellen, daß sie später als eigener Dienst anderen Anwendungen zugänglich gemacht wird. Diese Migration schafft die Grundlage für eine äußerst flexible Geschäftsstrategie hinsichtlich der Implementierung und Einführung von neuen Diensten oder Anwendungen. Mit der gewählten Begriffsdefinition kann nunmehr die logische Struktur der Basisarchitektur aus Abbildung 6 näher erläutert werden:

Mediation Hardware repräsentiert die zugrundeliegende Vermittlungshardware, die paket- oder leitungsorientiert realisiert sein kann.

Mediation Interface bezeichnet die Schnittstelle zur Vermittlungshardware, über die deren Funktionalität, etwa das sog. *Call Setup*, möglichst technologie- und herstellerunabhängig angeboten wird. Der Leistungsumfang und die Komplexität der zugrundeliegenden Hardware bestimmt den nötigen Implementierungsaufwand.

Service Layer beinhaltet die angebotenen Basisdienste, deren Funktionalität von bestimmten multimedialen Anwendungen benötigt wird. Hierbei werden auch Dienste berücksichtigt, die von allen Anwendungen

¹Die gewählte Definition des Begriffs „Dienst“ ist enger gefaßt, als in der Telekommunikation üblich, um die Beziehungen der Systembestandteile genauer charakterisieren zu können.

genutzt werden (z.B. Authorisierung und Abrechnung). Diese Schicht realisiert die nötige Erweiterbarkeit des Systems hinsichtlich neuer Dienste wie den sog. *Internet Supplementary Services* [ZM00].

Service API Layer repräsentiert die möglichen Programmierschnittstellen, über die Anwendungen die von der Service Layer angebotenen Dienste in Anspruch nehmen können. Die Art des Zugriffs — Methodenaufruf oder Protokollausführung — bestimmt den nötigen Implementierungsaufwand. Diese Schicht realisiert die erforderliche Erweiterbarkeit des Systems hinsichtlich neuer Schnittstellen-Technologien.

Application Layer umfaßt die eigentlichen konvergierten Anwendungen, die dem Anwender eine maßgeschneiderte Lösung durch Kombination aus Daten- und PSTN-basierter Funktionalität anbieten.

Die gewählte Strukturierung spiegelt die Verantwortlichkeiten, Strategien und zeitlichen Verhältnisse wieder. Die blau hinterlegten Schichten in Abbildung 6 werden zunächst ausschließlich von der Siemens AG implementiert. Sie kapseln wertvolles Wissen über die verwendete Vermittlungshardware und die benötigten Basisdienste. Zudem unterliegen sie strengen Anforderungen hinsichtlich Zuverlässigkeit, Performanz, u.ä. Daher ist für diese Komponenten ein längerfristiger Entwicklungszyklus zu erwarten. Hingegen können die Anwendungen der gelb hinterlegten Application Layer von Telekommunikationsanbietern, externen Software-Herstellern oder auch der Siemens AG implementiert werden. Aufgrund der noch ungeklärten, sich rasch verändernden Marktanforderungen ist mit einer starken Fluktuation der realisierten Anwendungen zu rechnen. Dies führt zu einem deutlich verkürzten Entwicklungszyklus mit u. U. weniger stringenten nicht-funktionalen Anforderungen. Sobald sich jedoch bestimmte Anwendungen am Markt etablieren konnten, besteht die Möglichkeit, einen großen Teil ihrer Funktionalität als eigene Dienste zu integrieren.

Die folgenden Abschnitte beschreiben vier mögliche Varianten, die sich aus der vorgestellten Basisarchitektur ableiten lassen. Die vorgestellten Varianten beinhalten konkrete Ausprägungen der abstrakten Schichtenarchitektur und führen so zu einem besseren Verständnis der Grundkonzeption. Hierbei werden auch Implementierungs- und Wiederverwendungsaspekte in einem angemessenen Detaillierungsgrad berücksichtigt.

5.2 Variante 1

Die Architektur-Variante 1 repräsentiert eine direkte Umsetzung der vorgeschlagenen Basisarchitektur ohne explizite Berücksichtigung von existierenden Lösungen wie Parlay. Auf der linken Seite von Abbildung 7 ist die grundlegende logische Architektur für den besseren Vergleich erneut dargestellt, während die rechte Seite der Abbildung die Zuordnung der funktionalen Komponenten des Systems veranschaulicht.

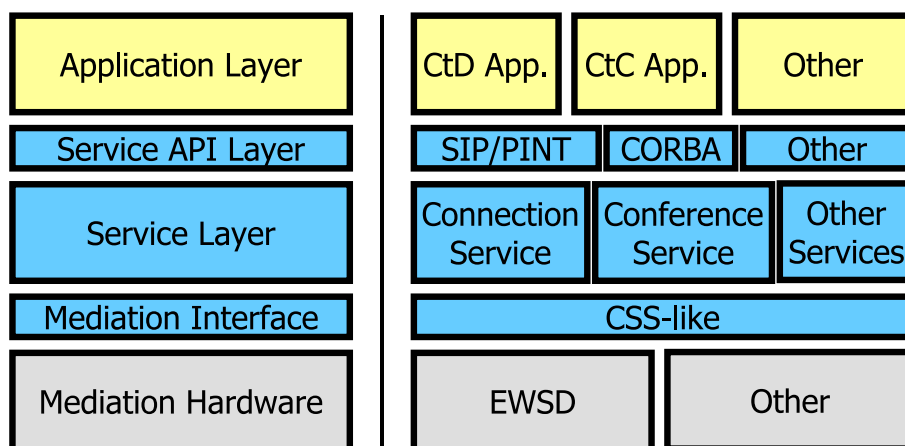


Abbildung 7: Architektur-Variante 1

In der untersten Schicht können die EWSD der Siemens AG oder andere Vermittlungssysteme eingesetzt werden. Die Schnittstelle zur Hardware (*Mediation Interface*) bietet vergleichbare Abstraktionen und Funktionalität.

lität wie die Primitive des sog. *Core Switching System* (CSS). Dieses beinhaltet Möglichkeiten, um einen Ruf zu initiieren (*Call Setup*), zu beenden (*Call Release*), Ereignisse (*Events*) zu generieren u.ä., und entspricht somit weitgehend dem Interface `CallSetupAPI` der gegenwärtigen Architektur (siehe Kapitel 4). Hierbei kann auch paketorientierte Vermittlungshardware berücksichtigt werden, etwa über eine geeignete Abstraktion von H.323.

Der Service Layer werden unmittelbar die vollständig neu entwickelten Basisdienste zur Realisierung der ISS-Funktionalität zugeordnet. Hierfür wird beispielsweise ein generischer Verbindungsdienst (engl. *Connection Service*) benötigt, der voraussichtlich von nahezu allen Anwendungen oder anderen Diensten in Anspruch genommen wird. Abhängig von den konkreten Anforderungen und dem vorgesehenen Zeitplan, bietet dieser Dienst die Möglichkeit, Sprach- und Videoverbindungen zwischen zwei Teilnehmern einzurichten, zu verwalten und zu beenden. Hierbei müssen verschiedene Signalisierungsvarianten, Adressformen u.ä. berücksichtigt werden.

Weiterhin denkbar ist ein Konferenzdienst (engl. *Conference Service*), der es Anwendungen erlaubt, Sprach- oder Videokonferenzen zwischen zwei oder mehr Teilnehmern einzurichten, zu verwalten und zu beenden. Hierfür kann ggf. die Funktionalität des Connection Service genutzt werden. Daneben sind auch weitere, eher technisch-orientierte Dienste wie Authentifizierung, Authorisierung oder Abrechnung in der Service Layer lokalisiert.

Der Zugriff auf die Dienste der Service Layer erfolgt ausschließlich über Komponenten der Service API Layer. Im Fall eines IP-basierten Zugangs über SIP/PINT (siehe Abschnitt 2.2) ist hierfür eine durchaus aufwendige Implementierung des zugehörigen Protokollautomaten erforderlich. Nach den Vorgaben des Protokolls muß die Schnittstellen-Komponente geeignete Methoden des zuständigen Dienstes aufrufen, Callback-Nachrichten entgegen nehmen, diese auswerten und in geeigneten Paketen an die Anwendung zurück senden. Im Fall einer objektorientierten, entfernt zugänglichen Schnittstelle über CORBA (siehe Abschnitt 2.4) ist im wesentlichen nur die Anbindung der über IDL generierten Skeleton-Klassen an die Methoden der Dienste zu leisten. Offensichtlich sind in der Service API Layer zukünftig noch weitere Schnittstellen, etwa H.323, vorstellbar.

In der obersten Schicht sind die eigentlichen Anwendungen lokalisiert. Hierbei bestimmen Marktanforderungen und verfügbare Dienste der Service Layer den Implementierungsaufwand. Für eine Click-to-Dial Anwendung (siehe Kapitel 1) ist neben einer einfachen grafischen Benutzerschnittstelle nur noch wenig zusätzliche Funktionalität erforderlich, da sie sich unmittelbar auf den Connection Service abstützen kann. Eine vergleichbare Aussage trifft auch auf eine einfache Click-to-Conference Anwendung zu, während eine Konferenzanwendung mit gemeinsamer, verteilter Bearbeitung von Dokumenten entsprechende Funktionalität des Datennetzes integrieren muß.

Abschließend kann festgestellt werden, daß Architektur-Variante 1 eine geringe Anzahl von Schichten sowie Komponenten aufweist und auf eine explizite Integration von Parlay verzichtet. Diese Tatsachen erleichtern Dokumentation und Verständnis der Architektur, führen zu einer besseren Performanz, und verringern voraussichtlich den nötigen Implementierungsaufwand. Allerdings wird bewußt auf Wiederverwendung existierender Parlay-Lösungen verzichtet, so daß eine sorgfältige Migrationsstrategie für eine mögliche, zukünftige Integration von Parlay erarbeitet werden muß.

5.3 Variante 2

Die Architektur-Variante 2, wie in Abbildung 8 dargestellt, ersetzt die proprietäre Service Layer der Basisarchitektur durch das als Industriestandard festgelegte, sog. *Parlay Service Framework* (siehe Kapitel 2.3). Entsprechend sind die avisierten ISS-Dienste, etwa Connection oder Conference Service, als vollwertige Parlay-Dienste in das System integriert (in Abbildung 8 dunkelblau hervorgehoben). Hierbei kann u. U. auf die Implementierung eigener Connection und Conference Services verzichtet werden, falls Parlay bereits vergleichbare Dienste definiert. Darüber hinaus können die spezifizierten Framework-Dienste von Parlay für Authentifizierung, Authorisierung, Abrechnung, u.ä. durchgängig wiederverwendet werden.

Allerdings ist hierfür eine nahezu vollständige Implementierung von Parlay erforderlich. Diese Tatsache führt

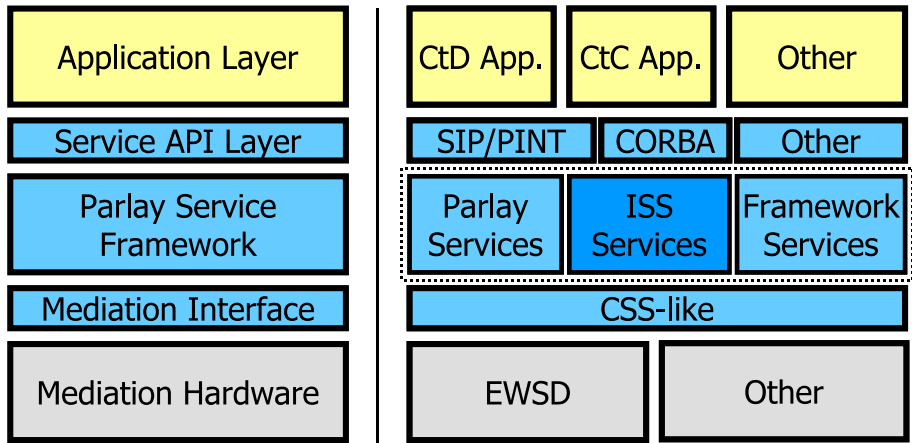


Abbildung 8: Architektur-Variante 2

zu einem stark erhöhten Aufwand zur Realisierung der vorgestellten Lösung. Weiterhin ist zu klären, inwieweit andere Parlay-Dienste über die Service API Layer für Anwendungen zur Verfügung gestellt werden. Während eine entsprechende Abbildung für CORBA-IDL bereits von Parlay spezifiziert wird, ist hierfür im Fall von SIP/PINT eine proprietäre Erweiterung erforderlich. Letztlich muß untersucht werden, ob die von Parlay angebotene Funktionalität einen angemessenen Abstraktionsgrad aufweist, der eine möglichst einfache Anwendungsentwicklung durch Dritte erlaubt.

5.4 Variante 3

Architektur-Variante 3 ist eine Kombination der beiden vorherigen Varianten (siehe Abbildung 9). Hierbei benutzt die Service Layer das Parlay Service Framework, um den Anwendungen die benötigte Funktionalität anzubieten. Das Parlay Service Framework wiederum setzt direkt auf dem Mediation Interface auf.

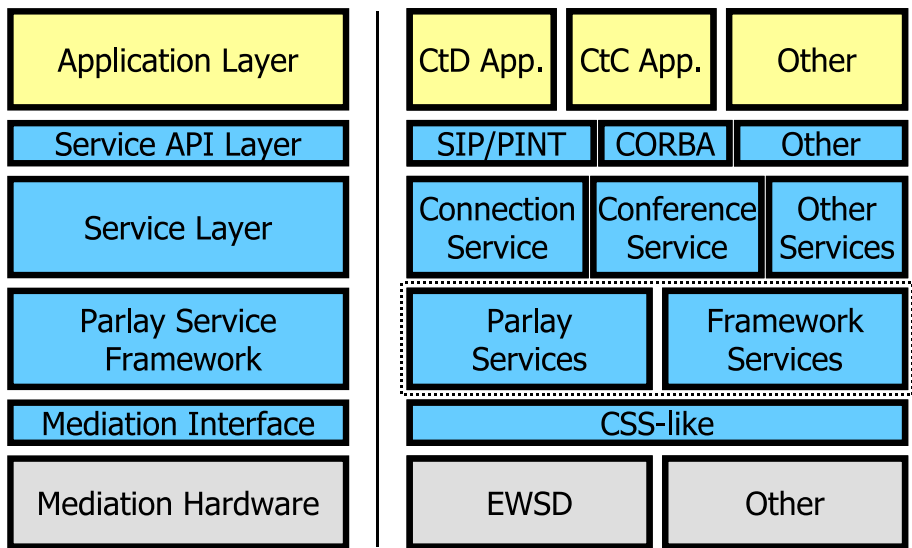


Abbildung 9: Architektur-Variante 3

Im Unterschied zu Variante 2 ist in diesem Fall Parlay nicht von außen zugänglich und muß somit nicht vollständig implementiert werden. Es genügt, zunächst nur die wirklich von den Diensten der Service-Layer benötigte Funktionalität zu realisieren. Später kann diese Implementierung Zug um Zug erweitert werden. Dies

erlaubt eine wohldefinierte Migrationsstrategie.

Allerdings weist Variante 3 eine hohe Anzahl von Schichten und Komponenten auf, so daß mit Risiken hinsichtlich Verständlichkeit und Performanz zu rechnen ist. Darüber hinaus ist zu klären, inwieweit sich die Konzepte und Abstraktionen der Service Layer und des Parlay Service Frameworks hinreichend differenzieren lassen. Nur so kann die klare logische Struktur der Architektur erhalten und Probleme hinsichtlich Redundanz bzw. Konsistenz vermieden werden.

5.5 Variante 4

Architektur-Variante 4 sieht eine zusätzliche Schicht, das sog. *Service Framework*, zwischen Service Layer und Mediation Interface vor (siehe Abbildung 10). Innerhalb dieser Schicht wird ein gemeinsames Objektmodell des Anwendungsbereichs mit Entitäten „Party“, „Session“ oder „Conference“ sowie ihren logischen Beziehungen realisiert. Zusätzlich sorgen Komponenten des Frameworks für Aufbau und Verwaltung des Objektmodells. Hierfür wird auf Funktionalität des Hardware Interfaces zurückgegriffen.

Durch das gemeinsame Objektmodell manifestiert sich ein Verständnis des Anwendungsbereichs auf einem abgestimmten Abstraktionsgrad, der durch das Parlay-Objektmodell u. U. nicht erreicht werden kann. Außerdem ermöglicht die vorgeschlagene Lösung komplexe und weitreichende Interaktionen zwischen den Diensten der Service Layer. Beispielsweise ergibt sich so die Möglichkeit der nahtlosen Migration von Sprach- oder Videoverbindungen zwischen verschiedenen Konferenzen oder die Erweiterung einer einfachen Verbindung zwischen zwei Teilnehmern zur Konferenz. Entsprechend leistungsfähiger können die eigentlichen Anwendungen realisiert werden.

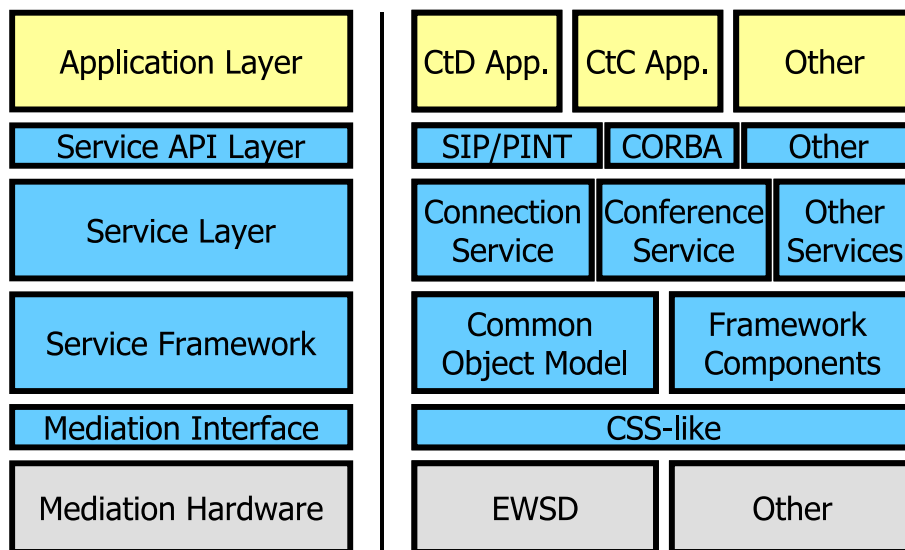


Abbildung 10: Architektur-Variante 4

Den genannten Vorteilen stehen offensichtlich entsprechende Nachteile gegenüber: Die hohe Anzahl an Schichten und Komponenten verursacht möglicherweise die bereits bekannten Probleme hinsichtlich Verständnis und Performanz des Systems. Weiterhin ist es erfahrungsgemäß schwierig, auf Anhieb wirklich passende Abstraktionen des Anwendungsbereichs zu definieren. Spätere Änderungen führen zu einem stark erhöhten Implementierungsaufwand und zu Problemen bei der Evolution des Systems. Letztlich ist zu klären, inwieweit Parlay in die vorgestellte Architektur-Variante integriert werden kann.

5.6 Zusammenfassung und Bewertung

Wie sich in der Diskussion der vorherigen Abschnitte ergeben hat, können die vorgestellten Architektur-Varianten hinsichtlich der Integration von Parlay, der geschätzten Komplexität bzw. Implementierungsaufwand, sowie ihrer Leistungsfähigkeit unterschieden werden. Da diese Kriterien nicht als orthogonal zu betrachten sind, werden die Verhältnisse in Abbildung 11 in getrennten Dimensionen dargestellt.

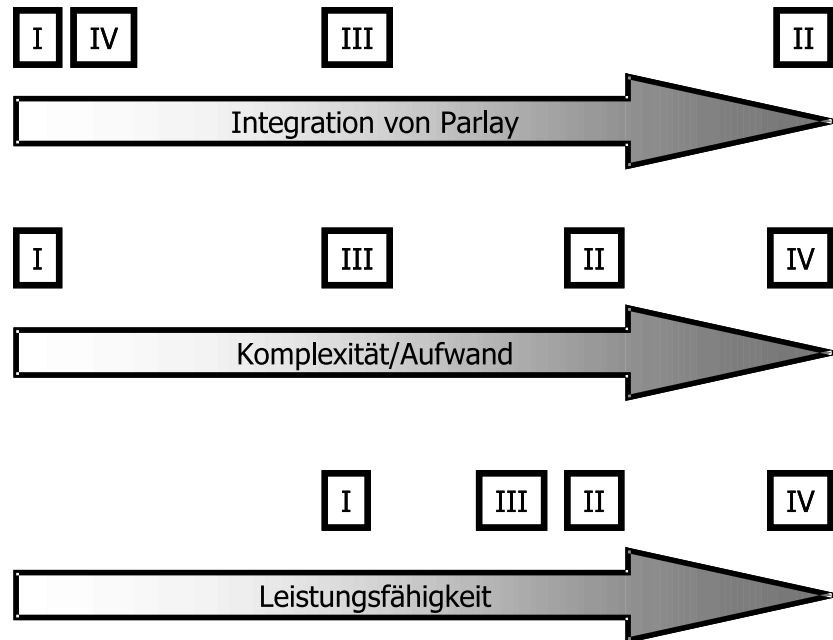


Abbildung 11: Bewertung der Architektur-Varianten

Variante 1 und 4 erfordern zunächst keine Integration von Parlay, während Variante 2 eine vollständige Implementierung von Parlay voraussetzt. Variante 3 erlaubt einen variablen Grad der Integration bzw. nötigen Implementierung.

Dementsprechend kann der Implementierungsaufwand für Variante 2 deutlich höher als für Variante 1 abgeschätzt werden. Variante 3 verhält sich wiederum variabel, je nach Vollständigkeit der Parlay-Integration. Hingegen muß die Komplexität von Variante 4 als besonders hoch eingeschätzt werden, da die Realisierung des Service Frameworks voraussichtlich äußerst aufwendig ist.

Jedoch ermöglicht das gemeinsame Objektmodell des Service Frameworks weitreichende Interaktionen zwischen den Diensten, so daß die potentielle Leistungsfähigkeit von Variante 4 als ausgesprochen hoch eingeschätzt werden muß. Variante 2 und 3 sind demgegenüber etwas weniger leistungsfähig, wobei die genaue Einordnung offensichtlich von der Qualität der Parlay-Integration abhängig ist. Entsprechend der geringen Komplexität sowie der fehlenden Integration von Parlay ist für Variante 1 nur eine mittlere Leistungsfähigkeit zu erwarten.

Die letztendliche Auswahl einer geeigneten Architektur-Variante richtet sich nach Strategie und Zeitplan der Siemens AG. Hierbei ist die vorgeschlagene Basisarchitektur flexibel genug, um eine schrittweise Evolution zwischen den Varianten zu realisieren. Die erforderliche Migrationsstrategie ist Gegenstand des nächsten Kapitels.

6 Migrationsstrategie

Anhand der vier unterschiedlichen Architektur-Varianten und ihrer jeweiligen Eigenschaften wird in diesem Kapitel ein konkretes Vorgehen diskutiert. Ohne genaue Kenntnis einer konkreten Geschäftsstrategie versteht

sich der hier präsentierte Vorschlag lediglich als allgemeiner Leitfaden für die Realisierung eines OSUN-Pilotsystems. Konkret wird ein Migrationspfad vorgestellt, der sich in drei einzelne Phasen untergliedert. Während Phase 1 Ziele einer eher kurzfristigen Planung verfolgt, sollen in Phase 2 mittelfristige und schließlich in Phase 3 langfristige Ziele erreicht werden. Es ergibt sich hierbei eine Migration der zugrundeliegenden Architektur ausgehend von Variante 1 über Variante 3 zu Variante 2 (siehe Kapitel 5.2, 5.4 und 5.3). Die folgenden Abschnitte stellen zuerst die jeweiligen Meilensteine in einer Übersicht dar, besprechen dann konkrete Realisierungsdetails und verweisen schließlich auf ausgesuchte Aspekte einer Implementierung.

6.1 Phase 1: Übergang von Basisarchitektur zu Variante 1

Zielsetzung

Innerhalb der nächsten Zeit erscheint es insbesondere interessant, die Machbarkeit und praktische Relevanz der „Converged Services“ demonstrieren zu können. Die Menge der Anwendungen ist hierbei zunächst beschränkt auf „Click-to-Dial“, die Menge der Zugriffsschnittstellen auf das SIP/PINT-Protokoll. Auf diese Weise entsteht ein erster, technischer Durchstich der OSUN-Plattform, der insbesondere für Leistungstests Verwendung finden kann, um Aussagen über die Skalierbarkeit der Architektur treffen zu können. Die im letzten Kapitel besprochene Architekturvariante 1 erscheint für diese frühe Phase ideal, nicht zuletzt da die bereits existierende Pilotimplementierung mit geringem Aufwand migriert werden kann. Zudem sollte der entstehende Prototyp in dieser Phase um neue Anwendungen (z.B. „Click-to-Conference“), neue Basisdienste und neue Zugriffsschnittstellen (vor allem CORBA) ausgebaut werden. Dabei gewinnt das Entwicklerteam wertvolle Erfahrungen, die mittel- und langfristig wichtig sind. Dies gilt insbesondere, sobald ein geeignetes Framework definiert wird, das Erweiterungen des Systems in den festgelegten Richtungen explizit unterstützen soll.

Realisierung

Durch einige wenige Modifikationen kann die bestehende Pilotimplementierung zu einer Architektur wie in Variante 1 beschrieben migriert werden:

- Entscheidend erscheint die Aufteilung der jetzigen Click-to-Dial Applikation (siehe Abbildung 5) in einen allgemeinen Dienst zum Aufbau von Verbindungen zwischen zwei Partnern und eine neue „CtD-Anwendung“, die dem Benutzer die entsprechende Funktionalität nun auf einfache Weise anbietet. Der Aufwand für diesen Schritt ist minimal aufgrund der Tatsache, dass die Anwendung Click-to-Dial fast unmittelbar auf die Funktionalität der zugrundeliegenden Hardware abbildbar ist. So können die bisher realisierten Parlay-Klassen übernommen werden, da sie prinzipiell bereits einen Connection-Service realisieren. Jedoch sollte der Aufwand investiert werden, eine generische Schnittstelle für die Dienste der zweiten Schicht zu spezifizieren. Durch diese Abstraktion fällt es später leichter, neue Dienste hinzuzufügen und nicht zuletzt vermittelt sie ein Verständnis über die geeignete Grundfunktionalität eines Dienstes.
- Insgesamt muß eine klare Aufteilung der einzelnen Komponenten auf die einzelnen Schichten der Architektur erfolgen. Dies erfordert vor allem die detaillierte Spezifikation und Dokumentation der jeweiligen Schnittstellen. In diesem Zusammenhang ist es wünschenswert, mehrere Komponenten pro Schicht zu realisieren, um eventuelle Fehler in der Spezifikation frühzeitig aufzudecken. Die Abschottung der Schichten mit Hilfe von Schnittstellen erlaubt es insbesondere, die bereits bestehenden Parlay-Klassen im Ergebnis der ersten Phase zu belassen, auch wenn sie außerhalb der Schicht nicht sichtbar sind.
- Aufwendiger ist die Spezifikation einer Schnittstelle zum leitungsvermittelten Netzwerk einzuschätzen. Hierbei könnte man durch eine Anlehnung an bestehende Lösungen, wie beispielsweise CSS oder auch Teilen des H.323-Standards, den Aufwand begrenzen.

- Da geplant ist, das Pilotsystem eventuell auch außerhalb von Siemens zum Einsatz zu bringen, müssen bereits Querschnittsfunktionalitäten wie z.B. Sicherheitsmanagement und Abrechnungsverfahren realisiert sein. Es ist allerdings ausreichend, diese Komponenten recht einfach zu halten, um keine unnötige Komplexität in das System einzubringen.

Implementierungsaspekte

Click-to-Dial ist ein gutes Beispiel für die Fragestellung, ob und in welcher Weise ein zustandsbehaftetes Protokoll in die oberen Schichten der Architektur transferiert werden kann. So könnte in der SIP/PINT-Schnittstelle vermerkt sein, welche Verbindungen eröffnet wurden, damit fortgesetzte Zugriffe auf dieser Information aufsetzen können. Alternativ könnte diese Komponente ohne eigene Zustände implementiert sein, wodurch bei jedem Zugriff der eigentliche Zustand des Protokolls erneut rekonstruiert werden muß. Ein Vorteil hierbei liegt in einem beschleunigten Zugriff auf Protokollzustände. So erlaubt diese Lösung z.B. die Administration von Verbindungen innerhalb der Serviceschicht.

Als problematisch in dieser frühen Phase könnte sich weiterhin der Focus auf eine ausgesuchte Anwendung herausstellen. Es sollte also stets beachtet werden, wie sich das Hinzufügen neuer Dienste und Anwendungen auf die zugrundeliegende Struktur auswirken kann.

6.2 Phase 2: Übergang von Variante 1 zu Variante 3

Zielsetzung

Nach einem erfolgreichen technischen Durchstich erscheint es in der mittelfristigen Planung wichtig, durch einen Ausbau des Systems in die Breite, die Einbettung des Systems in die technische Landschaft und somit seine praktische Relevanz zu demonstrieren. Aus diesem Grund sollte das System um weitere Dienste und entsprechende CORBA-Schnittstellen erweitert werden. Ebenso ist es wünschenswert, weitere Vermittlungstechnologien evtl. auch von anderen Herstellern zu berücksichtigen.

Mit der Einführung zusätzlicher „Converged Services“, über CtD hinaus, ergibt sich nunmehr zwingend die Notwendigkeit zu einem umfassenden technischen Framework, das die Menge der angebotenen Dienste unterstützt und koordiniert. Mittelfristig bietet sich daher eine Migration zur Architekturvariante 3 (siehe Abschnitt 5.4) an. Eine zu diesem späteren Zeitpunkt zur Verfügung stehende Parlay-Implementierung könnte diesen Schritt erheblich vereinfachen.

Realisierung

Ist noch keine zufriedenstellende Implementierung des Parlay-Standards verfügbar, so müssen nun Teile dieses Frameworks selbst realisiert werden. Dafür ist ein Verständnis erforderlich, welcher Bereich von Parlay notwendig ist (naheliegendermaßen erscheinen „Generic Call Control“ und einige ausgewählte Framework Services) und auf welche Teile verzichtet werden kann.

Während die bestehenden Dienste bisher direkt auf die CSS-ähnliche Schnittstelle zugegriffen haben, müssen sie nun mit den Parlay- und Framework-Services kommunizieren. Dadurch wird es erforderlich, die entsprechende Funktionalität neu zu implementieren.

Ein möglicher Wiederverwendungseffekt besteht in der Migration von Diensten, die in Phase 1 bereits implementiert wurden, hin zu Teilimplementierungen der Parlay Framework-Services. Beispiele dafür sind die Komponente zur Abrechnung sowie das Sicherheitsmanagement. Im Lichte dieses Wiederverwendungspotentials ist es sinnvoll, schon bei der Durchführung von Phase 1 die Parlay Framework-Services im Auge zu behalten.

Implementierungsaspekte

Als kritisch könnte sich erweisen, dass im Parlay-Framework Funktionalität spezifiziert ist, die sich mit Teilen der bestehenden Pilotimplementierung überschneidet. Um dadurch entstehenden Problemen auszuweichen

empfiehlt es sich, frühzeitig auf die Dienste von Parlay umzusteigen.

Wie oben bereits angesprochen ergibt sich die Problematik der Konsistenz von Protokollzuständen über die einzelnen Schichten der Architektur hinweg. Aus diesem Grund sollte die Positionierung der Zustandshaltung innerhalb der Schichtenarchitektur sorgfältig geplant bzw. durch Maßnahmen zur Konsistenzerhaltung abgesichert werden.

Die neu hinzugekommene Zwischenschicht, zusätzliche Dienste, sowie der Einsatz des Frameworks können zu einer gesteigerten Komplexität im Zusammenspiel der einzelnen System-Komponenten führen. Zur Beherrschung dieses Risikos bietet sich die frühzeitige Erstellung einer Reihe geeigneter Prototypen an. Anhand dieser Prototypen läßt sich insbesondere eine Performanzabschätzung vornehmen; ebenso lassen sich kritische Abläufe aufdecken und optimieren.

6.3 Phase 3: Übergang von Variante 3 zu Variante 2

Zielsetzung

Im Workshop [Wor00b] wurde Architekturvariante 2 (vgl. Abschnitt 5.3) als Ziel des Migrationspfads identifiziert. Neben einer klaren Struktur kombiniert dieser Vorschlag auf ideale Art und Weise die relevanten Standards. Siemens erwartet, dass auf lange Sicht eine geeignete Implementierung des Parlay-Frameworks zur Verfügung steht, die im Pilotsystem Verwendung findet und die Migration vereinfacht.

Realisierung

Um die Fähigkeiten des Systems adäquat demonstrieren zu können, ist die Einbindung neuer, attraktiver Anwendungen ausgesprochen wichtig. Dafür käme zum Beispiel „Shared Whiteboard“ oder „Unified Messaging“, oder einer der anderen Vorschläge aus [ZM00] in Betracht. Nur auf diese Weise kann demonstriert werden, wie einfach das System erweitert werden kann und wie dynamisch die Verwendung der dargebotenen „Converged Services“ angelegt werden kann.

Spätestens in dieser Phase müssen die existierenden Dienste vollständig zu Parlay-Diensten migriert werden. Dies erfordert die Implementierung der durch Parlay vorgegebenen Dienstschnittstellen. Dafür können Teile des proprietären Servicemanagements durch die Verwendung entsprechender Parlay-Framework-Services ersetzt werden.

Implementierungsaspekte

Zumindest in Teilen zieht die Migration hin zu Parlay eine Änderung der entsprechenden proprietären Schnittstellen in Parlay-CORBA-Schnittstellen nach sich. Auch die längerfristige, parallele Unterstützung von proprietären und Parlay-Schnittstellen kann erforderlich sein und sollte berücksichtigt werden. Eventuell bietet sich Siemens die Chance, den SIP/PINT-Standard hinsichtlich der Unterstützung von Parlay-Services voranzutreiben. Bei einer Veränderung der Zugriffsschnittstellen muß eine geeignete Strategie zur Anpassung bestehender Anwendungen erarbeitet werden.

6.4 Zusammenfassung

In diesem Kapitel wurde ein evolutionäres Vorgehen vorgestellt und diskutiert, um einerseits frühzeitig einen funktionierenden Prototypen der OSUN-Architektur zu erhalten und andererseits auf lange Sicht eine stabile, skalierbare Architektur zu erzielen. Die einzelnen Migrationsschritte gewährleisten, dass erarbeitete Ergebnisse zu großen Teilen wiederverwendet werden können und auf diese Weise der gesamte Aufwand minimiert wird.

7 Zusammenfassung

Die Bereitstellung einer geeigneten Software-Architektur zur Integration von Diensten aus den Bereichen IP und PSTN ist ein Schlüsselfaktor für den Markterfolg der Internet Supplementary Services. Die gewählte Architektur bestimmt in hohem Maße die Flexibilität und den Entwicklungsaufwand bei der Einführung neuer Mehrwertdienste; damit ist die Software-Architektur sowohl unter Kosten-, als auch unter time-to-market-Gesichtspunkten von strategischer Bedeutung.

In den vorangegangenen Abschnitten wurde die Architektur der vorliegenden CtD-Pilotimplementierung insbesondere im Hinblick auf ihre

- Erweiterbarkeit,
- Verständlichkeit, und die
- Einbindung vorhandener und sich entwickelnder Standards

untersucht. Die Auswahl dieser Kriterien aus dem in Abschnitt 3.2 angegebenen Katalog erfolgte im Rahmen gemeinsamer Workshops von Siemens ICN und TU München auf Basis der bei Siemens ICN konkret vorliegenden Anforderungen.

Die übersichtliche Darstellung einer Software-Architektur hat großen Einfluß auf deren Verständlichkeit und Kommunizierbarkeit im Kontext aller an der Systementwicklung beteiligten Parteien. Dies erleichtert auch die frühzeitige Erkennung von Problemfeldern in der Architektur und hilft dabei, Anforderungen an die Teilkomponenten des Systems klar herauszuarbeiten. Bisher ist die Beschreibung der CtD-Architektur im wesentlichen durch ein Klassendiagramm gegeben. Daher schlagen wir vor, die Dokumentation um Architekturdiagramme, wie sie etwa in Abschnitt 5 gezeigt sind, zu erweitern. Anhand dieser Diagramme läßt sich eine geeignete logische Strukturierung herausarbeiten, was zu einer zusätzlichen Aufwertung der Architekturbeschreibung führt.

Im Hinblick auf die Erweiterbarkeit der OSUN-Plattform um zusätzliche Dienste schlagen wir weiterhin die logische Strukturierung des Systems entlang einer Schichtenarchitektur vor. Ziel ist dabei die Kapselung der Vermittlungsdienste gegenüber der zugrundeliegenden Vermittlungshardware einerseits, sowie die Kapselung der Applikationen gegenüber den Vermittlungsdiensten andererseits. Eine derartige Schichtung unterstützt, über den Gewinn an Erweiterbarkeit und Verständlichkeit hinaus, die Einführung von klar definierten Schnittstellen zu den angebotenen Diensten. Diese Schnittstellen können geeignet auf Standardprotokolle wie etwa SIP/PINT, oder Middleware-Technologien wie etwa CORBA aufsetzen.

In der CtD-Pilotimplementierung ist die vorgeschlagene Schichtung zum Teil bereits umgesetzt. Die Kapselung der Vermittlungshardware ist durch eine entsprechende Klasse vorbereitet. Damit ist die vorliegende Pilotimplementierung nicht nur in ihrer Funktion als Beispielimplementierung ein wertvoller Beitrag zum Verständnis der OSUN-Plattform.

Die konkrete Ausprägung der Schichtenarchitektur ist unter anderem von der weiteren strategischen Ausrichtung der OSUN-Plattform abhängig. Einflußfaktoren sind hierbei besonders die Beachtung von Standards aus dem IP- und PSTN-Bereich (SIP/PINT, Parlay, und H.323 sind Beispiele dafür), sowie die Aufteilung der Dienstentwicklung zwischen Siemens ICN und Drittanbietern. Um die Auswirkungen dieser Einflußfaktoren aufzuzeigen, wurden in dieser Studie vier Architektur-Varianten diskutiert. Diese unterscheiden sich besonders bezüglich ihrer Leistungsfähigkeit, des notwendigen Entwicklungsaufwands für die Architektur und neu bereitzustellende Dienste, sowie der Integration bestehender und sich entwickelnder Standards. Die Architektur-Varianten bieten die Möglichkeit, die Ziel-Architektur bezüglich der erwähnten Einflußfaktoren, sowie der

Architektur-Evaluierungskriterien auszuwählen.

Gemeinsam mit Siemens ICN wurde ein Migrationspfad von Variante 1 über Variante 3 hin zu Variante 2 als zielführend identifiziert. Die dazu erforderlichen Schritte wurden ebenfalls im Rahmen dieser Studie aufbereitet.

Zusammenfassend läßt sich feststellen, daß bei Umsetzung obiger Empfehlungen die Darstellung der Architektur zusätzlich an Struktur gewinnt, ihre Verständlichkeit erhöht wird, und die Erweiterbarkeit um neue Mehrwertdienste deutlich verbessert wird.

Literatur

- [ABC⁺97] Gregory Abowd, Len Bass, Paul Clements, Rick Kazman, Linda Northrop, and Amy Zaremski. Recommended Best Industrial Practice for Software Architecture Evaluation. Technical Report CMU/SEI-96-TR-025, Software Engineering Insitute of Carnegie Mellon University, 1997.
- [BHB⁺97] Manfred Broy, Franz Huber, Klaus Bergner, Andreas Rausch, and Marc Sihling. Nemetschek: Evaluation of the O.P.E.N. architecture, report for Nemetschek AG, November 1997.
- [BHKS97] Manfred Broy, Christoph Hofmann, Ingolf Krüger, and Monika Schmidt. A Graphical Description Technique for Communication in Software Architectures. Technical Report TUM-I9705, Technische Universität München, 1997.
- [BRS97] Klaus Bergner, Andreas Rausch, and Marc Sihling. Using UML for modeling a distributed Java application. Technical Report TUM-I9735, Technische Universität München, Institut für Informatik, 1997.
- [BRS99] Klaus Bergner, Andreas Rausch, and Marc Sihling. Architektur und Entwurf der Electronic Sourcing Workbench, Bericht für die Healy Hudson GmbH, October 1999.
- [BRSV98] Klaus Bergner, Andreas Rausch, Marc Sihling, and Alexander Vilbig. Component-oriented GUI-Architecture: The Redesign of a Medical Patient Browser, August 1998.
- [CGI00] CGI Resource Index. The CGI Resource Index. <http://www.cgi-resources.com/>, 2000.
- [Com98] Compaq Computer. Internet Telephony. <http://www.digital.com/info/LIW06W/>, 1998.
- [Dat00] DataBeam Corp. A Primer on the H.323 Series Standard. <http://www.databeam.com/h323/h323primer.html>, 2000.
- [Del00] John Delaney. Next Generation IP Services To Unify Voice and Data, Enhance E-Commerce and Change the Telecommunications Industry Forever. <http://www.ovum.com/news/NGSpr.htm>, 2000.
- [EM99] Alexander Egyed and Nenad Medvidovic. Extending Architectural Representation in UML with View Integration. In Robert France and Bernhard Rumpe, editors, *UML 99 - The Unified Modeling Language, Beyond the Standard*, volume 1723 of *Lecture Notes in Computer Science*. Springer Verlag, 1999.
- [Gri00] Charles Gritton. IP Voice Quality: Is There a Price to Pay? *Telecommunications Online*, February 2000.

- [ISO00] International Standardization Organization. International Organization for Standardization Homepage. <http://www.iso.ch>, 2000.
- [Kar99] Asim Karim. H.323 and Associated Protocols. <ftp://ftp.netlab.ohio-state.edu/pub/jain/courses/cis788-99/h323/index.h%tml>, 1999.
- [Ken99] Michael Kennedy. The New Public Network. *Telecommunications Online*, August 1999.
- [Kok99] Mathias Kokot. Opening the Carriers' Internet Door. *Telecommunications Online*, December 1999.
- [Kru99] Philippe Kruchten. *The Rational Unified Process - an Introduction*. Addison Wesley, 1999.
- [Kur00] Jürgen Kuri. Grabenkämpfe. *Magazin für Computertechnik*, 4:158–160, February 2000.
- [Lag00] G. Laghi. Business Opportunity Specification 262 - Commercial Platform for SURPASS. BO262v04+.doc, 2000.
- [LKA⁺95] D. Luckham, J. Kenney, L. Augustin, J. Vera, D. Bryan, and W. Mann. Specification and Analysis of System Architecture Using Rapide. *IEEE Transactions on Software Engineering*, 21(4):336–355, 1995.
- [Lu 98] H. Lu et al. Toward the PSTN/Internet Inter-Networking–Pre-PINT Implementations, RFC2458, November 1998.
- [Maa99] Kamel Maamria. The Net Telephony Dilemma. *Telecommunications Online*, July 1999.
- [OMG97] Object Management Group. A Discussion of the Object Management Architecture. <http://www.omg.org/library/oma/oma-all.pdf>, 1997.
- [OMG00] Object Management Group. Object Management Group Home Page. <http://www.omg.org>, 2000.
- [PC99] Scott Petrack and Lawrence Conroy. The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services, October 1999.
- [PG00a] Parlay Group. Parlay API Business Benefits White Paper. White_paperv2.01.doc, 2000.
- [PG00b] Parlay Group. Parlay Website. <http://www.parlay.org>, 2000.
- [RJB98] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison Wesley Publishing Company, December 1998.
- [Sal99] Peggy Salz. The Gold Rush is Over. *Telecommunications Online*, October 1999.
- [SDK⁺95] M. Shaw, R. DeLine, D. Klein, T. Ross, D. Young, and G. Zelesnik. Abstractions for Software Architectures and Tools to Support Them. *IEEE Transactions on Software Engineering*, 21(4):356–372, 1995.
- [SG96] Mary Shaw and David Garlan. *Software Architecture - Perspectives on an Emerging Discipline*. Prentice Hall, April 1996.
- [Sti00] Wolfgang Stieler. Gipfeltreffen. *Magazin für Computertechnik*, 4:88, February 2000.
- [Sun00] Sun Microsystems. The Source for Java Technology. <http://java.sun.com/>, 2000.
- [W3C00] The World Wide Web Consortium W3C. HTML Home Page. <http://www.w3.org/MarkUp/>, 2000.

[Wor00a] Workshop zur Vorstellung der OSUN Architektur, 3. Februar 2000.

[Wor00b] Workshop zur Evaluierung der OSUN Architektur, 23. Februar 2000.

[ZM00] Zygan-Maus. Business Opportunity Specification 233 - Internet Supplementary Services Network Concept. IUS_BO233.doc, 2000.