# SOFTWARE-KAPSELUNG

Eine Technik für die Einbindung
bestehender Software-Bausteine
in neue Applikationen

von
**Harry M. Sneed**

Juni 1998


Software-Engineering Service GmbH
Rosenheimer Landstr. 37
D-85521 Ottobrunn/München

# 1. Zweck der Software-Kapselung

Software-Kapselung ist die Alternative zur Software-Konvertierung und Neuentwicklung. Sie bietet die Möglichkeit die Software in ihrer Urumgebung wiederzuverwenden und zwar ohne eine aufwendige Transformation. Eine gewisse Anpassung läßt sich nicht vermeiden, aber sie bleibt gering im Verhältnis zu der großen Umwälzung, die mit einer Konversion verbunden ist. Die Illusion alte Software unverändert in einem neuen Zusammenhang wiederzuverwenden bleibt eben eine Illusion. Um unverändert wiederverwendet zu werden, müßte die Software von Anfang an dazu konzipiert sein.

Ein häufiger Anlaß für Kapselung ergibt sich dann, wenn ein Anwendungssystem mit dynamischen Verbindungen zu anderen Nachbarsystemen verteilt wird. Eine dynamische Verbindung ist gegeben, wenn ein Programm in einer Anwendung ein Programm in einer anderen Anwendung aufruft oder wenn ein Programm in einer Anwendung direkt auf die Daten der anderen Anwendung zugreift. Bei einer Vorverlagerung des aufrufenden Systems wird aus dem Aufruf ein entfernter Prozeduraufruf (RPC) bzw. eine entfernte Methodeninvokation (RMI). Aus dem direkten Datenzugriff wird ein Aufruf zu einer entfernten Datenzugriffsschicht. In allen Fällen handelt es sich um eine Datenfernübertragung zwischen Systemen in unterschiedlichen Umgebungen.

Zu diesem Zweck wurde die Common Object Request Broker Architecture -CORBA- von der Object Management Group -OMG- spezifiziert. Diese Spezifikation einer verteilten Systemarchitektur mit einer eigenen umfassenden Schnittstellensprache -IDL- wurde bereits schon mehrfach erweitert bzw. ergänzt und hat inzwischen eine normative Auswirkung auf die Industrie. Sie ist zwar keine zwingende Voraussetzung für eine Software-Kapselung, es gebe auch andere Möglichkeiten, entfernte Programme miteinander zu verbinden, z.B. DCOM von Microsoft oder MQ-Series von IBM, dennoch wer nach einer allgemein gültigen Lösung sucht, ist gut beraten sich nach der CORBA-Norm zu richten. Sie wird inzwischen von einer ganzen Reihe verfügbarer Produkte unterstützt.

# 2. Stand der Kapselungstechnologie

Der Begriff „Wrapper“ wurde schon im Jahre 1988 auf einer OO-Tagung in den USA von dem IBM Mitarbeiter Thomas Dietrich geprägt im Zusammenhang mit der Einbindung bestehender Software in neue OO-Systeme. Seitdem wird zu diesem Thema immer mehr und immer häufiger geschrieben, vor allem in den einschlägigen OO-Fachzeitschriften. Es könnte leicht der Eindruck entstehen als wäre „Wrapping“ die Lösung für das Legacy Problem schlechthin. Eine ausführliche Diskussion den Kapselungsmöglichkeiten ist bei Mowbray und Zahavi in ihrem Buch „The Essential CORBA“ zu finden. Dort heißt es „An object wrapper provides access to a legacy system through an encapsulation layer. The encapsulation exposes only those attributes and operation definitions desired by the software architect“. Die Autoren nennen sieben Anwendungsarten für einen Wrapper:

- Layering bzw. Schichtenbildung
- Datenmigration
- System Reengineering

- Middleware-Entwicklung
- Kapselung
- Implementierung einer Objektarchitektur
- Objektauftragsvermittlung

**Schichtenbildung** ist gegeben wenn Schnittstellen aufeinander gelagert werden, z.B. wenn über eine C-API eine CORBA-IDL Schnittstelle gelegt wird, um eine andere Sicht auf die darunterliegende Schnittstelle zu erhalten. Die Details der internen Schnittstelle werden von der oberen verborgen.

**Datenmigration** kann über eine Datenzugriffsschicht implementiert werden, welche die Struktur der vorhandenen Daten hinter der Zugriffschnittstelle verbirgt. Das neue System bekommt die Daten in einer völlig anderen Form als die in der sie gespeichert sind, z.B. als relationale Sichten.

**System Reengineering** kann ebenfalls über eine Modulzugriffsschicht implementiert werden, welche die Struktur der vorhandenen Programme hinter der Zugriffsschnittstelle verbirgt. Das neue System benutzt die Systemfunktionen in einer anderen Weise als die ursprüngliche, z.B. als Methoden für ein Geschäftsobjekt.

**Middleware** umfaßt alle Dienstprogramme die weder zur Basissoftware noch zur Anwendungssoftware gehören. Oftmals dient die Middleware dazu Anwendungsprogramme miteinander zu verbinden und gewisse gewöhnliche Dienstfunktionen für sie auszuführen. Auch diese Programme kann man hinter einer eigenen Schnittstelle verbergen. Dazu wird ein Wrapper benötigt.

**Kapselung** ist die allgemeinste Form des Objekt Wrappings. Sie trennt die Implementierung von der Schnittstelle. Das beste Beispiel hierfür liefert die CORBA Schnittstellenspezifikation, die unabhängig von den aufgerufenen Prozeduren ist. Sie stellt die reinste Form des Wrappings dar.

**Die Implementierung einer Objektarchitektur** setzt Wrapping voraus. Das „Einwickeln“ der Komponenten einer Architektur ermöglicht erst den Grad der Flexibilität, der erforderlich ist um die Mehrfachverwendbarkeit und Anpassungsfähigkeit eines Software-Systems zu gewährleisten. Durch Wrapping werden die internen Komponenten vor den externen Veränderungen geschützt.

**Objektauftragsvermittlung** bietet eine Vielzahl einzelner Dienstleistungen an, darunter auch den Zugriff auf entfernte Objekte oder Funktionen. Wichtig dabei ist, daß der Auftraggeber nicht merkt bzw. nicht wissen braucht, wo und wie seine Aufträge erledigt werden. Alles passiert hinter einem geschlossenen Vorhang. Dieser Vorhang bildet sozusagen die Verpackung der angebotenen Dienstleistung, z.B. die Datenkonversion.

Mowbay und Zahavi gehen weiter und beschreiben mehrere Möglichkeiten einen Wrapper zu implementieren:

- Wrapping über entfernte Prozeduraufrufe (RPCs)
- Wrapping durch Dateiübergaben
- Wrapping mit Sockets bzw. Andockungsanschlüssen (APIs)
- Wrapping über eine Anwendungsprogrammschnittstelle

- Wrapping mit Skriptprozeduren
- Wrapping mittels Makrotechnik
- Wrapping durch gemeinsame Header-Dateien

Diese Techniken können einzeln oder in Kombination miteinander eingesetzt werden. Welche Technik angewendet wird hängt zum einen von den Anforderungen und zum anderen von der Umgebung ab. Sie haben alle ihre Vor- und Nachteile. Wichtig ist, daß die Entwickler der Wrapping Software mit ihnen vertraut sind.

In seinem Buch „Migrating to Object Technology“ geht auch Ian Graham auf das Thema „Object Wrapping“ ein. Er beschreibt einen Wrapper als eine Software-Schicht, die eine Interaktion zwischen konventionellen und objektorientierten Programmen zuläßt. In seinem SOMA Modell spielt das Wrapping, als Bindeglied zwischen den Systemkomponenten, eine besondere Rolle. Allerdings räumt Graham ein, daß „implementing wrappers is not as easy as it may sound“. Die Wrapper-Software muß den Eigenarten der gekapselten Programme oder Datenbanken gerecht werden und dies kann sich im Detail als äußerst schwierig, wenn nicht gar unmöglich, erweisen.

Eine kurze aber aufschlußreiche Behandlung des Themas „Object Wrapping“ veröffentlichte Sanjiv Gossain in der Zeitschrift Object Expert. Gossain beschreibt fünf Alternativen wie auf vorhandene Hostapplikationen zugegriffen werden kann:

- entfernte Prozeduraufrufe,
- Applikationsprogrammschnittstellen,
- CICS Transaktionen,
- Dateiübergaben und
- Datentabellen

Gossain unterscheidet weiter in einzelne Wrapperklassen ohne semantischen Inhalt und in mehrfachen Wrapperklassen mit semantischen Inhalt. Für die Verbindung zu den Legacy Systemen empfiehlt er das Brückenmuster von Gamma. Auch Gossain weist auf die Schwierigkeiten, die bei der Implementierung eines Wrappers auftreten können hin, vor allem in Puncto Performance. Die Datenübertragung zwischen dem Vorrechner und dem Host bleibt seiner Auffassung nach ein potentieller Engpaß.

In anderer Literatur werden vier Arten von Wrapper erwähnt:

- Datenbank-Wrapper
- Service-Wrapper
- Applikations-Wrapper
- Funktions-Wrapper

Ein **Datenbank-Wrapper** ist eine Zugriffsschicht zu bestehenden Datenbeständen. Er macht es Clientapplikationen möglich, auf alte Daten zuzugreifen - sie zu schreiben, lesen, ändern und löschen.

Ein **Service-Wrapper** kapselt Systemdienste wie Drucken, Datenfernübertragung, Email und Speicherverwaltung. Damit können Applikationen diese Dienste in Anspruch nehmen, ohne auf alle interne Eigenschaften eingehen zu müssen.

Ein **Applikations-Wrapper** ist eine Softwareschicht über die Batchläufe, Online-Transaktionen und Programme eines bestehenden Anwendungssystems ausgeführt werden können ohne sie zu verändern. Der Anwender bzw. Client kann die alten Jobs, Transaktionen und Programme als Objekte in die eigene Anwendung einbinden. Der Wrapper stellt die Verbindung her.

Ein **Funktions-Wrapper** ist ein Schnittstellenumsetzer für den Aufruf einzelner Funktionen in bestehenden Programmen. Die alten Programme werden als Objekte, deren Funktionen als Methoden benutzt. In der Regel müssen die Programme zu diesem Zweck angepaßt werden.

Der Wrapper befindet sich in der Regel dort, wo die Altsoftware residiert. Entweder ist er statisch gelinkt oder er bindet die aufgerufenen Module zur Laufzeit. Zwischen dem Wrapper und der verteilten Neusoftware werden Nachrichten hin und her geschickt. Um die Nachrichten zu vermitteln, wird ein RPC Mechanismus oder ein Object Request Broker (ORB) benötigt. In den letzten Jahren haben sich ORBs als das bevorzugte Instrument zur Erledigung dieser Aufgabe durchgesetzt. Insofern ist Wrapping immer mehr in Verbindung mit einem ORB zu sehen. Der ORB verbindet die verteilten Clientapplikationen mit den zentralen Server auf dem der Wrapper mit der Altsoftware läuft.

## 3. Stufen der Software-Kapselung

Eine Kapselungsstufe entspricht einer Granularitätsebene in der Hierarchie eines Software-Systems. Je höher die Stufe, um so größer ist das eingewickelte Software-System und um so umfangreicher die Funktionalität. Im Prinzip können auch ganze Systeme gekapselt werden.

Innerhalb eines betrieblichen Informationssystems gibt es Batchprozesse und Dialogprozesse. Batchprozesse beinhalten einen oder mehrere Prozeßschritte. Hinter jedem Prozeßschritt steckt ein Dienstprogramm oder ein Anwenderprogramm. Ein Batchprozeß ist also eine Kette einzelner Programme, die im Stapelmodus ausgeführt werden. Dialogprozesse bestehen wiederum aus einer Kette von Transaktionsprogrammen. Eine Transaktion ist die Verarbeitung einer Nachricht vom Endanwender im Onlinebetrieb. Das Transaktionsprogramm erhält die Nachricht und führt eine oder mehrere Funktionen aus, bis es zum Schluß die Kontrolle an den Bildschirm bzw. den Bildschirmbediener zurückgibt.

Programme, ob Batch oder Online, bestehen aus einem Hauptprogramm und einer Menge Unterprogrammen, wobei diese Menge allzuoft eine Leermenge ist. Die getrennt kompilierbaren Unterprogramme werden als Module bezeichnet. Sie erhalten ihre Eingaben über eine Parameterschnittstelle und geben ihre Ausgaben über die gleiche Schnittstelle wieder zurück.

Hauptprogramme und deren Module setzten sich aus einzelnen Prozeduren zusammen. Eine Prozedur ist ein Stück zusammenhängender Code mit einem Eingang und einem gemeinsamen Rückkehrpunkt. Sie wird programmintern aufgerufen. Im Gegensatz zu Modulen sind Prozeduren nicht getrennt compilierbar und in den meisten bisherigen Sprachen haben sie keinen eigenen Datenbereich. Was genau eine Prozedur ist, wird von der jeweiligen Programmiersprache bestimmt. In Assembler sind es CSECT´s oder Codeblöcke. In PL/I sind es die internen Prozeduren. In COBOL können sie sowohl Sections, als auch Paragraphen sein.

Für jede Stufe der Software-Granularität bietet sich eine Kapselung an. Demzufolge gibt es fünf Stufen der Software-Kapselung:

- Prozeßkapselung,
- Transaktionskapselung,
- Programmkapselung,
- Modulkapselung und
- Prozedurkapselung.

Auf der Prozeßstufe wird ein Batchjob vom Client aus gestartet. Der Batchjob verarbeitet eine oder mehrere Eingabedateien, schreibt eine oder mehrere Stammdateien bzw. Datenbanken fort und hinterläßt eine oder mehrere Ausgabedateien. Angestoßen wird sie über eine Kommandoprozedur bzw. eine Job Control Prozedur, die aus der Ferne angestoßen wird. Die Eingabedateien können von einer entfernten Stelle bezogen und die Ausgabedateien an eine entfernte Stelle versandt werden.

Auf der Transaktionsstufe wird ein Dialogschritt von der Ferne ausgelöst. Die auslösende Nachricht kommt aber nicht vom Host Terminalsystem, sondern von einem entfernten Clientarbeitsplatz. Dennoch werden die gleichen Funktionen ausgeführt wie im normalen Onlinebetrieb. Am Ende der Verarbeitung wird die Ausgabenachricht oder Nachrichten an den Auftraggeber zurückgeschickt.

Auf der Programmstufe wird ein Hauptprogramm von einem entfernten Clientprogramm aus angestoßen. Das Programm verarbeitet seine Eingaben, in der Regel in einer Schleife bis alle Eingaben abgearbeitet sind, produziert seine Ausgaben solange es noch Eingaben gibt und gibt die Kontrolle am Ende zurück. Die Eingaben können von einem fremden System stammen, z.B. von dem Clientprogramm, und die Ausgaben können an ein fremdes System wieder versandt werden, z.B. an das Clientprogramm.

Auf der Modulstufe wird ein Unterprogramm aufgerufen. Der einzige Unterschied hier ist, daß der Aufruf aus der Ferne kommt. Die Parameter sind nicht im gleichen Adreßraum wie das Modul selbst, sondern stammen von einem Clientprogramm in einer anderen Umgebung. Sie müssen in der Hostumgebung von der Wrapper-Software bereitgestellt bzw. weitergeleitet werden.

Auf der Prozedurstufe wird nur ein bestimmter Abschnitt eines Programmes angesteuert. Der Rest bleibt nicht betroffen. Es wird nämlich nur an gewissen Stellen im Programm eingestiegen, die sekundären Eingänge, und am Ende der jeweiligen Prozedur wieder ausgestiegen. Zu diesem Zweck müssen die Daten der Prozedur von außen als Parameter bereitgestellt werden. Prozeduren entsprechen Methoden in objektorientierten Programmen und lassen sich leicht von verteilten Objekten per Remote Method Invocation aufrufen.

Auf allen Kapselungsstufen wird eine Steuerungssoftware benötigt, welche die Verbindung zwischen dem Auftraggeber, dem Client, und dem gekapselten Objekt, dem Server, herstellt. Dies wird als Wrapper bezeichnet. Es ist die Aufgabe des Wrappers die Eingangsnachrichten zu erhalten und verwalten, sie in die Dateien, Masken oder Schnittstellen der Zielsoftware umzusetzen, die Zielsoftware anzustoßen, die Ergebnisse der Zielsoftware zu sammeln, sie in Ausgangsnachrichten umzusetzen und diese an den Auftraggeber in der Ferne zurückzuleiten.

# 4. Methodik der Software-Kapselung

## 4.1 Jobkapselung

Um einen Batchprozeß von außen zu starten muß eine Kopie der ursprünglichen JCL-Prozedur abgezogen werden. In der JCL-Prozedur sind Datenbestände dem Job als Ressourcen zugewiesen. Darunter sind neben den Datenbanken, z.B. IMS oder DB-2 und Stammdateien, z.B. ISAM oder VSAM, auch Bewegungsdateien, die in der Regel sequentiell organisiert sind.

Mit einem Editor ist die gewünschte JCL-Prozedur dem Benutzer anzuzeigen. Der Benutzer soll dann jene Data Set Allokationen markieren, die er vom entfernten Server erhalten und die er an den entfernten Server, nach Abschluß des Jobs, übertragen möchte. Anhand dieser Markierungen wird eine zweite JCL-Prozedur generiert bzw. aus der ersten abgeleitet. Diese zweite Prozedur startet zunächst einen File Transfer um die Eingabedateien vom entfernten Server in die Zielbibliothek auf dem Host herüber zu ziehen. Anschließend läuft der Prozeß wie gewöhnlich ab. Nach dem erfolgreichen Abschluß des Jobs, der anhand des Condition-Codes festgestellt wird, wird das File Transfer Utility nochmals angestoßen, diesmal um die Ausgabedateien von der Hostbibliothek an den Server zu übertragen. Danach wird der Prozeß beendet und der Job abgemeldet.

Die Prozeßergebnisse bzw. die Ausgabedateien befinden sich jetzt auf dem Server und können dort vom verteilten System weiterverarbeitet werden.

## 4.2 Transaktionskapselung

Für die Kapselung einer Online-Transaktion wird eine Kopie des Online Programmes, z.B. IMS-DC oder CICS, erstellt. Diese Kopie wird automatisch von einem Tool auf Terminalinteraktionen hin analysiert. In CICS Programmen sind es die EXEC SEND und RECEIVE Makroanweisungen. In IMS-DC Programmen sind es die GET UNIQUE (GU) und INSERT (ISRT) Operationen auf den IO-PCB (Program Control Block).

Nachdem die Terminaloperationen erkannt sind, werden sie in Kommentar gesetzt und an ihrer Stelle wird ein vordefinierter Aufruf zu der entsprechenden Wrapper-Prozedur z.B. XTPINP oder XTPOUT eingebaut. Diese LINK Anweisung bewirkt, daß die Steuerung nicht an den TP-Monitor, sondern an den Wrapper übergeben wird um den IO-Auftrag zu erfüllen.

Bei CICS Programmen müssen außerdem noch die Handle Aid und Fehlerbedingungen abgefangen werden, weil die alten Mainframe Terminals mit Funktionstasten arbeiten und diese Tasten auch den Programmablauf steuern. Deshalb müssen sie in der Schnittstelle eingebaut sein, d.h.

- die CICS Response Code,
- die CICS Funktion und
- die CICS Funktionstaste

müssen vom Wrapper belegt werden. Dafür benötigt der Wrapper wiederum eine Information vom Endanwender. Dies gehört zum Inhalt der Eingangsnachricht.

In dem gekapselten CICS Programm wird also nach jedem simulierten RECEIVE die simulierte Funktionstaste abgefragt und entsprechend im Programm verzweigt.

Im Falle von IMS wird die Standard Parameterliste mit den Angaben

- Funktion,
- Status,
- PCB,
- IO-Area und
- Format

benutzt. Hier gibt es keine besondere Bearbeitung.

Ein weiteres Problem ist die Maskengestaltung. In CICS und IMS kann zu jedem Feld in einer Maske die Feldlänge und das Feldattribut z.B. Farbe, Unterstrich, Helligkeit, usw. vom Programm gesetzt werden. Diese Angaben dienen der Datenanzeige. Im Falle einer Kapselung haben sie keine Bedeutung für die Anzeige an dem Client-Arbeitsplatz, da dieser nur die nackten Daten braucht. Sie sollten daher bei einer Maskenausgabe vom Wrapper entfernt werden. Dadurch wird auch der zu übertragende Datenstrom wesentlich verkürzt. Andererseits werden diese Angaben in der Eingabemaske vom Programm abgefragt. Deshalb müssen sie bei einer Maskeneingabe vom Wrapper mit Standardwerten belegt werden. Welche Werte hierfür geeignet sind hängt vom Terminalbedienungssystem

- CICS Basic Map Service oder
- IMS Message Format Service

ab.

### 4.3 Programmkapselung

Die Kapselung eines Batchprogrammes zielt darauf ab, einzelne ausgewählte Eingaben vom Clientprogramm aus zu empfangen und ausgewählte Ausgaben an das Clientprogramm zu senden. Die Eingaben eines Batchprogrammes sind Dateien, die Ausgaben sind Sätze.

Zuerst muß der Benutzer bestimmen welche Ein- und Ausgaben umzuleiten sind. Zu diesem Zweck wird der Source-Code des Zielprogrammes angezeigt, damit der Benutzer die gewünschten Eingabedateien und Ausgabesätze markieren kann. Anschließend wird das Programm so geändert, daß die READ und WRITE Anweisungen durch CALL Aufrufe ersetzt werden.

Bei READ Anweisungen sind die Parameter

- der zu lesende Satz und
- ein Return-Code.

Nach der Anweisung wird der Return-Code auf Ende geprüft und die erforderlichen Endeverarbeitungen angestoßen.

Bei WRITE Anweisungen ist der einzige Parameter

- der zu schreibende Satz.

Aufgerufen wird eine Prozedur im Wrapper die entweder den nächsten Satz aus der Eingabequeue dem Programm bereitstellt, oder den Satz vom Programm abholt und an das Clientprogramm weiterleitet.

### 4.4 Modulkapselung

Module sind selbständig kompilierbare Unterprogramme. In der Regel beinhalten sie ausgegliederte Funktionen und werden zur Ausführung ihrer Funktionen von einem Hauptprogramm - Batch oder Online - aufgerufen. Insofern haben sie schon eine Parameterschnittstelle bzw. eine Linkage Section. Allerdings sind ihre Parameter oft verstreute Adressen.

Um einen entfernten Aufruf zu erleichtern werden die Parameter durch ein Tool in einer einzigen Datenstruktur zusammengefaßt. Über den Wrapper werden die Eingangsparameter aus der Nachricht vom Client zugewiesen und das Modul aufgerufen. Beim Ausgang aus dem Modul werden die Ausgangsparameter wieder zusammengefaßt und über den Wrapper an den entfernten Client weitergeleitet.

Ansonsten laufen Module im gekapselten Modus genauso wie im normalen Betrieb. Diese Kapselungstechnik ist deshalb einfach und mit dem höchsten Grad an Wiederverwendbarkeit zu realisieren.

### 4.5 Prozedurkapselung

Die Prozedurkapselung hingegen ist die schwierigste Art der Kapselung. Dafür ist sie die feinste Stufe der Granularität und die mit der höchsten Wahrscheinlichkeit in das neue Objektmodell hineinzupassen. Das versteht sich, weil einzelne Codeabschnitte dem Methodengedanken am nächsten kommen.

Für die Methodenauswahl wird hier der prozedurale Teil des Zielprogrammes angezeigt. Der Anwender markiert die Code-Abschnitte, die er als Methoden benötigt, wobei er jeder Methode einen eigenen Namen gibt. In COBOL können hierfür SECTIONS oder PARAGRAPHEN, in PL/1 Prozeduren und in Assembler CSECTS oder Code-Blöcke verwendet werden. Die ausgewählten Abschnitte dürfen jedoch nur einen einzigen Eingang und nur einen Ausgang besitzen. Somit sind GO TO Verzweigungen verboten, d.h. das Programm muß strukturiert sein.

Nach der Auswahl der Methoden wird der Source-Code von einem Tool verändert. Zu Beginn eines jeden Abschnittes wird eine ENTRY Anweisung mit Parameterliste eingefügt. Die Parameter sind die Datenstrukturen, die von diesem Abschnitt angesprochen werden, sowohl Eingaben als auch Ausgaben. Am Ende des Abschnittes wird eine RETURN Anweisung eingefügt.

Alle nicht referenzierten Datenstrukturen werden auskommentiert. Die referenzierten Daten werden in COBOL in die Linkage Section verlagert, in PL/1 als Parameter und in Assembler als Dummy Sections deklariert. Dadurch kommen alle Daten, die eine Methode verarbeiten, von außen und werden als Referenzen übergeben - CALL BY REFERENCE. Tatsächlich befinden sie sich im Adressraum des Wrappers. Der Wrapper baut sie aus den Nachrichten vom Client auf.

# 5. Konstruktion der Kapselungsschale

Die Kapselungsschale, bzw. der Wrapper, ist eine Verbindungsschicht zwischen dem Kommunikationssystem und dem Zielobjekt, bzw. dem Job, Programm, Modul oder Codeabschnitt. Sie übermittelt die Nachrichten zwischen dem Vorrechner und dem Hostrechner, setzt die Daten vom einen Format ins Andere um, ruft das gekapselte Programm auf und fängt die IO-Operationen ab. Demzufolge besteht das Wrapperprogramm selbst aus mehreren Prozeduren - eine Hauptprozedur und mehrere Unterprozeduren. Die Hauptprozedur wird vom Kommunikationssystem aus gestartet, die Unterprozeduren werden vom Zielprogramm angesprungen. Es existiert in dem Wrapper eine Schnittstelle nach außen zum Kommunikationssystem und eine Schnittstelle nach innen zum gekapselten Objekt. Insgesamt setzt sich das Wrapper-Programm aus folgenden Komponenten zusammen:

- eine externe öffentliche Schnittstelle
- eine interne private Schnittstelle
- eine Nachrichtenverwaltungskomponente
- einer Datenumsetzungskomponente und
- einer I/O Simulationskomponente

## 5.1 Die Schnittstelle nach Außen

Die externe, öffentliche Schnittstelle ist von allgemeiner Natur. Sie enthält einen fest formatierten Kopfteil und einen variablen Rumpfteil.

Der Kopfteil enthält Angaben wie:

- die Benutzerkennung vom Auftraggeber
- das Arbeitsplatzkennzeichen des Auftraggebers
- das Transaktionskennzeichen
- Datum und Uhrzeit
- den Funktionscode
- die Funktionstaste
- den Fehlercode
- die Methodenbezeichnung
- die Anzahl der Eingabefelder
- den Nachrichtentyp und die Nachrichtenlänge

Der Rumpfteil ist eine Zeichenfolge variabler Länge mit Standard ASCII Zeichen. Binäre, Bit, gepackte und andere besondere Datendarstellungsformen sind wenn möglich zu vermeiden. Auch numerische Werte sollten als Ziffer im Zeichenformat übermittelt werden. Dies erleichtert die Umsetzung. Einzelne Felder sind durch ein Trennungszeichen abzugrenzen, das sonst nicht vorkommt z.B. einen ‘\’.

Der Kopfteil ist für die Kapselungssteuerung gedacht. Im Kopfteil identifiziert das Transaktionskennzeichen, welcher Job, bzw. welche Transaktion oder welches Programm auszuführen ist. Die Methodenbezeichnung enthält den betreffenden

Modul- oder Prozedurnamen. Die Funktionstaste ist für Dialogtransaktionen erforderlich. Der Funktionscode dient der Funktionsauswahl. Andere feste Angaben die dem Wrapper dienen gehören ebenfalls in den Kopfteil.

Der Rumpfteil enthält die eigentlichen Ein- und Ausgabedaten. Er ist für das gekapselte Objekt gedacht.

Als allgemeine Darstellungsform empfiehlt es sich die externe Schnittstelle in der CORBA-IDL Sprache zu beschreiben, auch dann, wenn sie in einer anderen Sprache wie C oder COBOL realisiert wird. Diese Form eignet sich immer als Spezifikation.

## 5.2 Die Schnittstelle nach Innen

Die interne, private Schnittstelle ist von spezifischer Natur. Als solche hängt sie völlig vom gekapselten Objekt ab. Im Falle einer Jobkapselung wird sie eine Kommandoprozedur sein. Im Falle einer Transaktionskapselung wird sie eine Terminalnachricht bzw. eine Bildschirmmaske sein. Im Falle einer Programmkapselung wird sie ein Datensatz bzw. ein Datenstrom sein. Im Falle einer Modul- oder Prozedurkapselung wird sie eine Parameterliste sein.

Das Format der internen Schnittstelle hängt von den Datentypen in der Zielprogrammiersprache oder der Maskensprache ab. So haben Assembler und PL/I andere Datenstrukturen als COBOL oder NATURAL. CICS-BMS Masken haben auch eine andere Gestalt als IMS-MFS Masken. Daraus folgt, daß die interne Schnittstelle fast immer maßgeschneidert oder, was anzustreben wäre, aus dem Quelltext des Zielprogrammes heraus generiert wird.

## 5.3 Die Nachrichtenverwaltungskomponente

Die Verwaltung der Ein- und Ausgangsnachrichten wird wichtig, wenn das gekapselte Objekt eine sequentielle Datei oder einen kontinuierlichen Datenstrom von der Clientapplikation verarbeitet. Der Rumpfteil der externen Schnittstelle ist in diesem Falle nicht eine einzige Zeichenfolge, sondern eine Reihe von Teilfolgen die laufend eine nach der anderen vom Auftraggeber eintreffen. Der Wrapper muß sie in einen Warteschlangenpuffer ablegen, so daß sie hinten angereiht werden, während die vordersten Nachrichten, bzw. Sätze dem gekapselten Programm zugeführt werden. So bald ein Satz gelesen wird, rücken die anderen Sätze in der Warteschlange nach.

Was für die Eingangsnachrichten gilt, gilt ebenso für die Ausgangsnachrichten, nur umgekehrt. Die Ausgabesätze, die aus dem gekapselten Programm herauskommen, müssen solange zwischengepuffert werden, bis sie an den Auftraggeber über die Datenfernübertragung gesendet werden können. In der Regel wird das Programm Ausgaben schneller erzeugen, als sie das Kommuikationsprogramm übermitteln kann. Deshalb muß es möglich sein die Ausgabepuffer in eine Zwischendatei auslagern zu können.

Die beiden Verwaltungsroutinen - die eine für die Ausgabenachrichten, die andere für die Eingangsnachrichten, laufen im asynchronen Modus nebeneinander und dürfen sich gegenseitig nicht behindern. Ihre Zeitanteile müssen so gut wie möglich balanciert werden.

### 5.4 Die Datenumsetzungskomponente

Die Prozeduren der Umsetzungskomponente erfüllen die Funktion die Eingabedaten aus der externen Schnittstelle in die interne Schnittstelle umzusetzen und die Ausgabedaten aus der internen Schnittstelle in die externe Schnittstelle zu verwandeln.

Bei der Umsetzung der Eingaben wird die ASCII Zeichenfolge zerlegt um die einzelnen alphanumerischen Werte herauszuholen. Danach werden sie in das interne Format konvertiert, z.B. binär oder gepackt. Da nicht alle internen Parameterdaten aus der externen Schnittstelle ableitbar sind, wird die Umsetzungsroutine die Lücke ausfüllen müssen, sowie bei den Attributbytes in den Masken.

Bei der Umsetzung der Ausgaben werden die Daten in der internen Schnittstelle von ihrem internen Datenformat in das externe Zeichenformat übersetzt und in einer Zeichenfolge zusammengefaßt. Zwischen den Werten werden Trennungszeichen eingefügt, damit der Empfänger sie unterscheiden kann. Null und Leerwerte können hier verdichtet werden, um die Länge der Nachricht zu kürzen.

### 5.5 Die IO-Simulationskomponente

Die Funktion der Simulationskomponente ist die Ein- und Ausgabeoperationen des simulierten Bildschirmsteuerungs- oder Dateiverwaltungssystems durch eigene Ein/Ausgabe Operationen zu ersetzen.

Für jede zu simulierende Eingabe/Ausgabe Operation ist eine Simulationsprozedur - ein Stub - erforderlich. Die Eingabesimulation nimmt den nächsten Satz, bzw. die nächste Maske, aus der Eingabeschlange und kopiert ihn in den Eingabebereich des Zielprogrammes. Manchmal genügt es lediglich einen Zeiger zu setzen, der auf den nächsten Satz hinweist. Er wird immer um die Satzlänge erhöht.

Die Ausgabesimulation kopiert die Daten aus dem Ausgabebereich des gekapselten Objektes in die Ausgabewarteschlange. Hier genügt es nicht einen Zeiger zu setzen. Die Daten müssen physisch bewegt werden um nicht von der nächsten Ausgabeoperation überschrieben zu werden, d.h. der Wrapper muß die Ausgaben sichern und im Ausgabepuffer so lange aufbewahren, bis er sie über das Kommunikationsnetz an den Auftraggeber absenden kann.

## 6. SoftWrap - ein Werkzeug für die Aufbereitung der zu kapselnden Programme

Bei der Kapselung vorhandener Anwendungsprogramme ist zwischen zwei Vorgängen zu unterscheiden. Im einen Vorgang wird die Anwendersoftware verändert um den gewünschten Zugriff überhaupt zuzulassen. Im anderen Vorgang wird die Wrapper-Software erstellt um den Zugriff durchzuführen. Für die beiden Vorgänge werden völlig unterschiedliche Werkzeuge benötigt.

Ein Werkzeug für die Anpassung der Zielprogramme ist das Tool SoftWrap vom Autor. SoftWrap bietet vier der fünf Kapselungsstufen an:

- Transaktionskapselung
- Programmkapselung
- Modulkapselung und
- Prozedurkapselung

und zwar für die drei Programmiersprachen

- Assembler
- PL/I und
- COBOL

Bei der Transaktionskapselung kapselt SoftWrap sowohl IMS-DLI, als auch CICS-BMS Operationen und leitet sie an die betreffende Wrapper-Prozedur um.

Bei der Programmkapselung kapselt SoftWrap alle sequentiellen Dateioperationen - Eingabe und Ausgabe - die vom Anwender bestimmt werden.

Bei der Modulkapselung optimiert SoftWrap die Parameterschnittstellen, damit sie von Außen leichter zu behandeln sind.

Bei der Prozedurkapselung baut SoftWrap Eingangsmasken mit Parameterlisten in den vom Anwender ausgewählten Codeabschnitten ein, so daß sie von Außen aufrufbar werden.

Das wesentliche was SoftWrap leistet ist eine Kopie des ursprünglichen Programmes vollautomatisch aus dem Original abzuleiten. Auf diese Weise ist es möglich nach jeder Programmänderung eine neue Kopie für die Kapselung ohne zusätzlichen Aufwand zu generieren. Die Alternative wären tiefgreifende manuelle Änderungen zum Code, die eine neue Fehlerquelle eröffnen. Es steht deshalb außer Frage, eine computergestützte Lösung wie SoftWrap sei fast unentbehrlich.

## 7. ObjectWrap - ein Framework für die Kapselung von Legacy Funktionen auf dem Host

Es wurde schon darauf hingewiesen, daß der Wrapper selbst keine generelle Lösung sein kann. Zu groß sind die Unterschiede zwischen den Benutzerumgebungen und den Benutzeranforderungen. Dennoch lassen sich einzelne Komponenten der Wrapper-Software sehr wohl standardisieren, so daß ein Wrapper Framework möglich ist. Dies ist der Gedanke hinter dem Produkt „ObjectWrap“.

In ObjectWrap werden die Komponenten für

- die Nachrichtenverwaltung und
- die IO-Simulation

sowie die Wrappersteuerung als schlüsselfertige C Prozeduren bereitgestellt. Außerdem ist die externe Schnittstelle fest definiert und als CORBA-IDL Schnittstelle spezifiziert.

Offen bleibt die interne Schnittstelle und die Datenumsetzungskomponente. Die interne Schnittstelle wird aus dem gekapselten Source-Code generiert. Zunächst ist sie in der Sprache des Zielprogrammes z.B. in Assembler, PL/I oder COBOL. Anschließend wird sie per Tool in eine C Schnittstelle umgewandelt. Mit Assembler können hier einige Probleme auftreten, mit PL/I und COBOL ist es relativ problemlos.

Übrig bleibt die Datenumsetzungskomponente. Es obliegt dem Anwender diese C Prozeduren zu schreiben. Sie sind von den anderen Komponenten weitgehend entkoppelt, ihre Schnittstellen - die externe und die interne - sind in Header-Dateien bereits kodiert, so daß es nur noch nötig ist den Umsetzungscode zu schreiben. Für einfache Parameterlisten, die der Modul- und Prozedurkapselung dienen, können sogar diese Umsetzungsprozeduren aus den beiden Schnittstellen generiert werden.

Dadurch deckt ObjectWrap mindestens 75 % der Wrapping Funktionalität ab und läßt sich dennoch von Fall zu Fall anpassen. Es ist ein klassisches Beispiel für die Anwendung der Framework-Technologie.

## 8. PMS - ein Produkt für die Kapselung bestehender Prozesse

Kapselung als Migrationstechnik ist nicht unbedingt mit der Objekttechnologie verbunden. Es geht auch ohne. Der Beweis dafür liefert das Produkt PMS - Process Management System.

Mit PMS ist es möglich von einem verteilten Prozeß aus auf Programme, Transaktionen, Module und Prozeduren auf dem Hostrechner zuzugreifen. Der Anwender braucht lediglich die Schnittstellen des Zielprogrammes im Clientprogramm aufzubauen und über einen entfernten Prozeduraufruf an PMS zu übergeben. PMS lädt das Zielprogramm und startet es an der gewünschten Einstiegsstelle. Auch sekundäre Eingangsstellen können über PMS angesteuert werden, so daß eine Kapselung einzelner Prozeduren durchaus möglich ist.

Die Datenübertragung erfolgt über einen globalen Datenpuffer mit zwei Speicherbereichen

- ein langlebiger Speicherbereich für jene Daten, die für die Dauer der Sitzung gelten (Open Dialog bis Close Dialog)
- ein kurzlebiger Speicherbereich für jene Daten, die nur für die Dauer einer Transaktion gelten (Open Transaction bis Close Transaction)

Es ist die Aufgabe von PMS den Speicher für diese Bereiche zuzuweisen und sie mit den Eingangsparametern des Clientprogrammes zu belegen. Im globalen Speicherbereich werden die IMS und CICS Kontrollblöcke aufgebaut. Auf diese Weise befindet sich das auszuführende Zielprogramm in seiner gewohnten Umgebung und läuft ab, als ob es direkt unter IMS oder CICS laufen würde.

In dieser Hinsicht erfüllt PMS alle wesentlichen Funktionen eines Wrappers. Es empfängt Aufträge als Nachrichten von den Clientapplikationen, es kapselt die Kundendaten in den eigenen Speicherbereichen, es steuert die gewünschten Legacy Funktionen an und übermittelt ihre Ergebnisse zurück an den Auftraggeber.

# 9. Weitere Wrapper Produkte

## 9.1 Softbench von HP

Softbench ist eine objektorientierte Entwicklungsumgebung von Hewlett-Packard für C++ Programme. Es unterstützt primär die Entwicklung der Clientapplikationen, aber es kann auch den Zugriff auf vorhandene Legacy-Produkte ermöglichen. Die neuen OO-Anwendungen werden mit existierenden Legacy-Anwendungen dadurch integriert, daß eine Schnittstelle oder Wrapper um die Altsoftware herum gebaut wird. Dies ermöglicht es Legacy Programme bzw. Module auf einem Hostrechner aufzurufen. HP empfiehlt folgende dreistufige Vorgehensweise für eine Ist-Aufnahme:

- Als erstes sollte der Anwender eine Bestandsaufnahme der existierenden Software durchführen. Dies entspricht einer Ist-Aufnahme, um festzustellen, welche Funktionen und Daten in die neuen objektorientierten Systeme aufgenommen werden sollten. Dafür benötigt der Anwender Reverse Engineering Tools.

- Als zweites sollte der Anwender eine Reihe neuer verteilter Applikationen mit GUI und lokaler Verarbeitungslogik entwickeln. Dabei soll er darauf achten, daß die alten Programme und Datenbestände nach Möglichkeit eingebunden werden. Die alten Funktionen werden über einen objektorientierten Wrapper aufgerufen bzw. es werden die alten Daten über eine ähnliche Schnittstelle wiedergewonnen und fortgeschrieben. Es gilt hier eine möglichst hohe Wiederverwendungsrate zu erreichen.

- Als drittes sollte der Anwender die Applikationen im Unternehmensnetz verteilen. Es ist eine 3-Schichten Architektur anzustreben mit Vermittlungsrechner zwischen dem Zentralrechner und den Endgeräten.

HP bietet einen Object-Wrapper namens OO-DCE an, der DCE Funktionen als C++ Objekte anbietet. Ein weiteres Produkt ist Odapter, der eine objektorientierte Sicht auf SQL-relationale Datenbanken bietet. Aus einer OSQL-Sprache werden C++ oder Smalltalk-Klassen generiert, die zur Laufzeit Objekte aus Oracle-7 und ALLBASE Datenbanken kreieren. Da es sich hier um eine umfangreiche Unternehmung mit zahlreichen Risiken handelt, bietet HP einen Migrationsdienst durch ihr Object-Oriented Solutions Center.

## 9.2 ObjectBroker von BEA

ObjectBroker wurde ursprünglich von DEC entwickelt, um die CORBA Norm zu erfüllen. Inzwischen hat die Firma BEA das Produkt übernommen und mit dem TP-Monitor TUXEDO integriert. Er wurde vor allem im Hinblick auf die Kapselung alter Anwendungen konzipiert, weshalb er hier als Wrapping Produkt eingestuft wird. Der ObjectBroker bietet drei Mechanismen um alte Programme und Daten zu kapseln, ohne in den Code eingreifen zu müssen.

- eine Skriptsprache für die Erstellung benutzerspezifischen Wrapperprozeduren,
- einen CORBA konformen Servertreiber um alte Programme auszuführen,
- eingebaute Clientstubs, um die Verbindung zum Server herzustellen.

Mit der Skriptsprache werden Applikationsprogrammschnittstellen (APIs) geschaffen, die alte Programme als Methodensammlungen behandeln lassen. Die Skriptsprache ähnelt einem Kommandozeilenprogramm wie UNIX Shellskript und VMS JCL Prozeduren. Es invokiert eine gekapselte Funktion mittels eines Betriebssystem EXEC Kommandos und fängt die Ausgaben der Funktion ab.

ObjectBroker generiert ein Serverskelett, das die dahinter liegende Legacy-Funktionen über eine API aufruft. Das Skelett wird wiederum von einem Clientauftrag angestossen. Somit dient es als Nachrichtenvermittler zwischen neuen Clientprogrammen und alten Serverprogrammen.

Die im Clientprogramm eingebaute Serverstellvertreter- bzw. Stubroutine ermöglicht es, die Server-Software als EXEC Modul, als statischer Linkmodul oder als dynamischer Linkmodul bzw. DLL zu implementieren. Damit kann die Laufzeit besser beeinflußt werden. Allerdings unterstützt das Produkt nur eine C++ Anbindung und benutzt die OLE Automation um auf Server-Objekte zuzugreifen.

### 9.3 Parts von Digitalk

Parts ist ein objektorientiertes visuelles Tool für die Entwicklung von Wrapper Software für Smalltalk-Anwendungen. Sie läuft unter OS/2 und stellt dort eine Verbindung zu CICS her. Über OS2/CICS werden Nachrichten an das MVS/CICS gesendet, um dort Transaktionen zu starten.

Parts ist in Amerika der weitverbeiteste Wrapper für Klientenapplikationen, die in Smalltalk geschieben sind. Es unterstützt nicht nur Programmaufrufe von Hostprogrammen sondern auch Zugriffe auf Hostdatenbanken. Für die Smalltalk-Welt ist Parts der logische Wrapperansatz mit den meisten Funktionen. Mittlerweile besitzt es auch eine OMG-IDL Schnittstelle.

### 9.4 ObjectStar von der Antares Alliance Group

ObjectStar ist ein objektorientierter Wrapper, der die Verbindung zwischen verteilten Objekten in einem Client/Server Netz mit Hostprogrammen verbindet. Es bedient Frontend Applikationen die unter UNIX und Windows-NT laufen. Es stellt eine Verbindung zu MVS Applikationen auf dem Host her. Dabei werden die Nachrichten über ein ORB - ORBIX oder SOM - vermittelt und umgesetzt. ObjectStar unterstützt auch den Zugriff auf VSAM Dateien sowie IMS und DB-2 Datenbanken. Allerdings muß das System vor Ort an der Zielumgebung und den zu kapselnden Software-Bausteine angepasst werden. Das Produkt wurde bereits mehrfach in den USA eingesetzt um Hostsysteme in eine Client-Server-Welt zu integrieren.

### 9.5 SOMobjects bzw. Component Broker von IBM

SOMobjects ist die derzeitige CORBA-Implementierung von IBM. Es benutzt die OMG-IDL ergänzt durch SOM Macros, um Smalltalk, C++ und OO-COBOL Objekte

miteinander zu verbinden. Von den *Object Services* deckt SOM 3.0 lediglich den Namensdienst (*Naming Service*) ab.

Die Weiterentwicklung von SOM 3.0 erfolgt im Rahmen des Component Brokers (CB). Eine erste Version des Component Borkers hat IBM für den Herbst '97 angekündigt. Der Component Broker besteht aus den zwei Subsystemen

- CB Connector und
- CB Toolkit.

Der CB Connector, auch mit SOM 4.0 bezeichnet, wird - laut Angabe von IBM - eine vollständige CORBA-Implementierung sein, die eine Vielzahl der *Object Services* unterstützt wie z.B. den Transaction und Security Service. Darüber hinaus ist der CB Connector ausgelegt, alte Hostapplikationen, die unter den TP-Monitoren CICS bzw. Encina mit den Datenbanksystemen DB2 oder IMS ablaufen, einzubinden.

Das CB Toolkit unterstützt die Entwicklung von verteilten Clientapplikationen in C++, Smalltalk, und OO-COBOL auf der Basis von Visual Age. Für die Migration der alten COBOL Programme in OO-COBOL bzw. COBOL für MVS, bietet IBM unter VisualAge Konvertierungswerkzeuge, die eine Einbindung mit CB-Connector ermöglichen. In dieser Richtung ist von der IBM in den kommenden Jahren noch mehr zu erwarten, denn die Software-Kapselungstechnologie sthet erst am Anfang.

## 10. Kapselung bestehender Anwendungen in der Praxis

Aus fachlicher Sicht kommt es darauf an die bestehenden Anwendungen zu kennen und zu wissen welche Software-Bausteine und Datenbestände in die neue objektorientierte Architektur überhaupt hineinpassen. Diese werden markiert und für die Kapselung aufbereitet. Die Aufbereitung wird immer eine andere sein, abhängig vom Gegenstand der Kapselung - Prozeß, Transaktion, Programm oder Prozedur.

Auf den höheren semantischen Ebenen sind die Bausteine größer und deshalb schwieriger zu kapseln. Große Bausteine haben in der Regel komplexe Schnittstellen, die nur aufwendig zu bedienen sind, d.h. auf der Seite des Clientprogrammes muß mehr geschehen. Eventuell müssen die Clientklassen un den vorhanden Baustein herumgebaut werden. Vor allem bei Online-Transaktionen ist es schwierig alle Vorzustände herzustellen, damit die Transaktion korrekt verarbeitet wird. Falls die alte Anwendung in einer benutzerspezifischen Umgebung läuft, muß diese Umgebung zuvor für die verteilte Verarbeitung angepaßt werden. Dies allein kann ein eigenes Projekt sein.

Auf den unteren semantischen Ebenen sind die Bausteine kleiner und leichter zu kapseln. Ihre Schnittstellen haben nur einen begrenzten Datenumfang. Somit sind sie leichter aufzubauen. Problematisch hier ist der Zugang zu den Code-Abschnitten, die oft tief im inneren der Programme liegen. Um sie in die neue Anwendung einzubinden, muß der Anwender genau wissen was sie leisten. Dazu muß er sich mit dem alten Programm im Detail auseinander setzen. Wenn die Abschnitte allzu klein sind, wird es sich kaum lohnen sie zu kapseln. Der Aufwand sie über das Netz aufzurufen steht dann in keinem Verhältnis zum Funktionumfang. In diesem Fall empfiehlt es sich sie neu zu schreiben. Dies ist ein allgemeines Problem der Software-Wiederverwendung.

In der Praxis soll der Anwender von Fall zu Fall entscheiden, was und wie gekapselt wird. Ein Patentrezept gibt es nicht. Neben den Performance Kriterien, sind der Aufwand für die Kapselung und der Umfang des Eingriffs in die bestehende Produktionsumgebung zu berücksichtigen. Diese Kostenfaktoren sind gegen das Ersparnis durch die Wiederverwendung der Altsoftware abzuwägen.