Objektorientierte Programmierung **in traditioneller Entwicklungsumgebung**

In Multiprozessorsystemen mit echtzeitkritischen Anwendungen dominieren noch immer prozedurale Programmierumgebungen. Viele Entwickler scheuen sich vor einer Überarbeitung der Software. Ein objektorientierter Ansatz lässt sich aber durchaus mit vertretbarem Aufwand realisieren. Dann sind auch die Vorteile der Objektorientierung, wie die Erstellung modularer und wartungsfreundlicher Software, nutzbar.

In vielen Softwareprojekten wie Geschäftssoftware, Telefonvermittlungen und Steuerungssystemen kommen prozedurale Sprachen zum Einsatz. Mit Einführung eines object request broker (ORB) können objektorientierte Technologien auch hier erfolgreich eingesetzt werden. Der "object handler" zeigt wie.

"Object Handler"

Die Schwierigkeit objektorientierter Methoden in parallelläufigen Systemen liegt in der Definition und Zuordnung von Ereignisketten zu bestimmten Systemprozessen. In Systemen ohne Parallelverarbeitung werden die aktiven Eigenschaften von Objekten (Methoden) als einfache Prozeduren implementiert. In parallelläufigen Systemen kann dies ebenso erfolgen, falls Rufer und Gerufener im selben Prozess laufen. Die Kommunikation zwischen Prozessen erfolgt per Nachrichtenaustausch. Das ORB-System hat die Aufgabe, die Implementierung der Methoden und der Methodenaufrufe unabhängig vom aktuellen Prozessentwurf zu machen.

Ein objektorientierter Ansatz wird durch starke Datenkapselung erreicht. Jedes Programmmodul mit seinen Daten und Prozeduren lässt sich als Objektklasse verstehen. Instanzdaten können in einer Datenbank gehalten werden.

Der "object handler" enthält die Prozessdefinitionen. Abhängig von den gewünschten Eigenschaften können verschiedene Prozessinkarnationen existieren, die sich in den Prozesseigenschaften (wie Priorität) unterscheiden. Die Zuordnung von Objekten oder Ereignisketten zu bestimmten Prozessen erfolgt tabellengesteuert. Dies macht den Entwurf und die Implementierung unabhängig vom aktuellen Prozessentwurf.

Für den Methodenaufruf stellt der "object handler" eine Schnittstelle zur Verfügung. Abhängig vom in Tabellen abgelegten Prozessentwurf hängt sie eine Anfrage an die eigene Meldungskette, oder die Anfrage wird an die entsprechende object handler instance geschickt. Ein Methodenaufruf über den "object handler" ist also strikt asynchron. Für synchronisierte Ereignisketten ver-gibt der "object handler" eindeutige Nummern, mit denen die Objekte die Ereig-nisketten identifizieren können. Jede Ereigniskette wird ausserdem vom ob-ject handler zeitüberwacht. Überschreitet eine Ereigniskette die Zeit, werden alle beteiligten Objekte benachrichtigt. Späte Änderungen im Prozessentwurf bedürfen keiner Implementierung der Objekte.

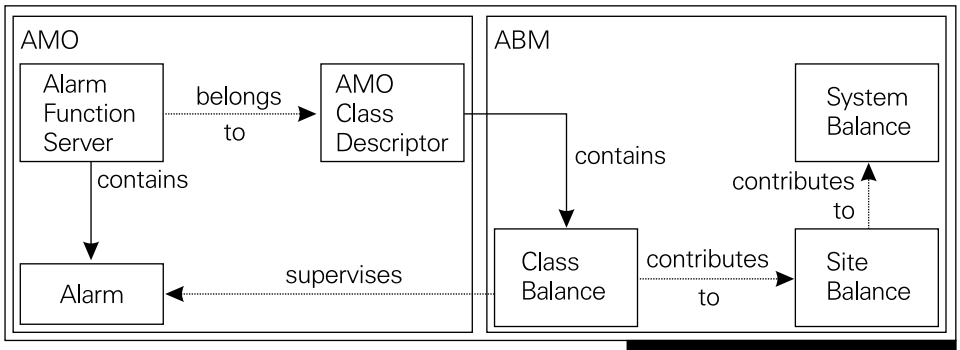
Praktische Erfahrung in der Telefonvermittlung

Das Konzept wurde in der Systemalarmierung des Telefonvermittlungssystems EWSD (Elektronisches Wählsystem Digital) realisiert. Für die Implementierung wurde die PASCAL-ähnliche prozedurale Programmiersprache CHILL verwendet. Die Instanzdaten sind in einer objektrelationalen Datenbank gespeichert.

Moderne Telekommunikationssysteme werden über die objektorientierte Q3-Schnittstelle bedient. Für die Alarmierung ist das Hilfsobjekt CurrentAlarmSummaryControl (CASC) sowie das Attribut CurrentProblemList (CPL) definiert. Dieses Attribut kann in allen Objekten verwendet werden, die in einen Fehlerzustand fallen können, der durch das Bedienpersonal zu beheben ist. Die Objekte werden als alarming managed objects (AMO) verallgemeinert. Es ist die Aufgabe eines CASCs, die Alarme einer AMO-Gruppe zusammenzufassen.

System Alarming entschied, eine proprietäre CASC-Version mit leicht vom Standard abweichenden Eigenschaften zu implementieren. Dieses Objekt wird Alarmbilanzmonitor (ABM) genannt. Die Erstimplementierung erfolgte im traditionellen Entwurf. Die Software wurde entsprechend ihrer Prozesse und Ereignisketten strukturiert. Zur Datenhaltung wurde ein fachliches Datenmodell entworfen, das unverändert in der objektrelationalen Datenbank abgelegt werden konnte. Es stellte sich heraus, dass jede kleine Änderung im Datenmodell eine Vielzahl von Codeänderungen nach sich zog. So entstand der Wunsch, für die nächste Version eine stärker objektorientierte Programmierung einzusetzen.

In der zweiten Version wurde deshalb der "object handler" implementiert. Jede Software, die auf einem Datenobjekt arbeitet, wurde in einem Codemodul zusammengefasst. Nach einer Stabilisierungsphase konnten die Objektmodule ohne Rücksicht auf Systemprozesse oder Plattformkommunikation realisiert werden.



Internal representation of AMO and ABM (first version)

Für die dritte Version wurde eine Überarbeitung des Modells vorgenommen. Dies geschah aus mehreren Gründen: Im Modell waren die Daten zur Realisierung des ABM nicht genau von den Daten der AMOs getrennt. Dies war verbunden mit ständigen strukturellen Problemen. Das Modell setzte auf ein Leistungsmerkmal der Datenbank zur Datenverteilung im Mehrprozessersystem auf und war in lastkritischen Situationen nicht tragfähig. Das Modell gab zudem eine dreistufige Bilanzierung der Alarme vor, die periodisch berechnet wurde und zu hohen Prozessorlasten führte.

Das neue Modell war strukturell bereinigt, beschränkte sich auf die Grundfunktionen des Datenbanksystems und beinhaltete nur eine zweistufige Bilanzierung. Da keine Software zur Prozesskommunikation angefasst werden musste und ein Großteil der Software innerhalb der Objekte wiederverwendbar war, schafften es zwei Personen allein, die Analyse und Implementierung der Überarbeitung in wenigen Wochen durchzuführen. Der Änderungsaufwand beschränkte sich auf die Anpassung der Datenbankdefinitionen, der Prozessinkarnationen im "object handler", der Steuertabellen und die Änderung der Objektmodule, die durch die Überarbeitung betroffen waren.

Autoren:
Michael Wagner
Friedrich Eisenhauer
Siemens ICN WN CC EB C11

Ansprechpartner:
Friedrich Eisenhauer
Siemens ICN WN CC EB C11
Tel.: 089/722-35063
Fax: 089/722-26226
E-mail:
friedrich-h.eisenhauer@icn.siemens.de