Kapitel 13

Zusammenfassung und Ausblick

13.1 Zusammenfassung

Ziel dieser Arbeit war, die Entwicklung verteilter Softwaresysteme zu vereinfachen. Ansatzpunkt hierfür ist, wichtige Aspekte der Programmierung verteilter Systeme, die heutzutage getrennt behandelt werden müssen, durch ein geeignetes Konzept zusammenzufassen und durch eine geeignete Abstraktion gemeinsam zu beschreiben. Die betrachteten Aspekte sind Verteilung, Nebenläufigkeit und Persistenz.

Zunächst wurde herausgestellt, dass diese Aspekte heutzutage mit weitgehend orthogonalen Mechanismen behandelt werden. Dafür wurden für jeden dieser Aspekte die wichtigsten, heute zur Verfügung stehenden Mechanismen klassifiziert und beschrieben. Eine Fokussierung auf die Sprache Java und die um sie entstandenen Techniken und Implementierungen spiegelt deren Bedeutung in der Praxis sowie in Wissenschaft und Forschung wider. Wegen ihrer besonderen Eignung für verteilte Systeme durch ihre Eigenschaften Plattformunabhängigkeit, dynamische Ladbarkeit und ihre für solche Umgebungen geeigneten Sicherheitsmechanismen, basieren heutzutage nahezu alle wichtigen neuen Impulse auf diesem Gebiet auf dieser Sprache oder sind eng mit ihr verknüpft.

Für den Aspekt der Verteilung wurden in Kapitel 2 drei zu unterscheidende Kommunikationsprinzipien, die unstrukturierte Datenkommunikation, die Kommunikation über Stellvertreter und die unmittelbare Objektkommunikation herausgearbeitet und an konkreten Mechanismen untersucht.

Der Aspekt der Nebenläufigkeit (Kapitel 3) konnte in die drei Bereiche Nebenläufigkeit durch Hardware oder Prozesse, nebenläufige Kontrollflüsse und objekt- bzw. komponentenbasierte Nebenläufigkeit unterteilt und diese Unterteilung anhand von konkreten Mechanismen untermauert werden.

Für die Persistenz wurden in Kapitel 4 die unstrukturierte Persistenz, die datenstrukturbasierte Persistenz und die objektorientierte Persistenz identifiziert und ebenfalls durch konkrete Mechanismen vorgestellt.

Wichtiges Ergebnis dieser Betrachtung ist, dass in typischen verteilten Systemen alle drei Aspekte berücksichtigt werden müssen. Dabei kann für jeden der drei Aspekte ein konkreter Mechanismus ausgewählt werden, unabhängig von der Entscheidung, welche Technik für die jeweils anderen Aspekte gewählt

wird. Daraus wird eine weitreichend orthogonale Behandlung dieser Aspekte mit heute zur Verfügung stehender Techniken abgeleitet, die den Programmierer zwingt, alle drei Aspekte zu beherrschen und zu behandeln und so zu einer enormen Komplexität bei der Entwicklung verteilter Systeme führt.

Im zweiten Teil der Arbeit wurde hinterfragt, ob diese Orthogonalität zwingend ist, oder ob sich diese Aspekte nicht durch eine geeignete Abstraktion zusammenfassen und dadurch vereinfachen lassen. Die Beobachtung auf anderen Gebieten der Softwareentwicklung legen die Vermutung nahe, dass bei einer erfolgreichen Zusammenfassung eine beträchtliche Vereinfachung zu erreichen ist. Als Beispiel wurde der Paradigmenwechsel von der prozeduralen zur objektorientierten Programmierung angeführt, der sich durch die Zusammenfassung der Aspekte Datenhaltung, Funktionsbeschreibung und Struktur, die in der prozeduralen Programmierung orthogonal behandelt werden müssen, zum Konzept der Klasse charakterisieren lässt (siehe Abschnitt 1.2). Die Vorteile dieser Zusammenfassung waren ausreichend, so dass man behaupten kann, der Paradigmenwechsel habe sich, trotz der Bindung an Altsysteme, weitgehend vollständig vollzogen.

Basierend auf Erkenntnissen anderer Forschungsvorhaben, insbesondere der Entwicklung von Verteilungsmechanismen mit unmittelbarer Objektkommunikation (konkretisiert am Beispiel Emerald) und der komponentenbasierten Nebenläufigkeit (entwickelt als konzeptionelle Erweiterung der Sprache Eiffel) sowie aus der Diskussion aus der Entwicklung des RMI (der in Java standardmäßig eingesetzten entfernten Kommunikation durch Stellvertreter), dargestellt in Kapitel 5, wurde ein eigener Ansatz entwickelt. Grundlegender Gedanke dabei ist eine neu eingeführte Unterscheidung zwischen lokalen und entfernten Referenzen und eine Unterteilung von Objekten zu in sich geschlossenen Gruppen, die von jeweils von einer Ausführungseinheit verwaltet werden. Diese Ausführungseinheit, virtueller Prozessor genannt, sorgt für die Konsistenz der lokalen ebenso wie der entfernten Referenzen auf Objekte der verwalteten Gruppe. Diese Konsistenz kann auch unter Nebenläufigkeit, Migration und Persistenz gewährleistet werden. Dieses Konzept wurde in Kapitel 6 entwickelt und seine Realisierbarkeit wurde in der Umsetzung in einer auf Java basierenden verteilten Programmiersprache namens Dejay gezeigt, die in Kapitel 7 vorgestellt wurde. Einige Aspekte der Implementierung wurden in Kapitel 9 dargelegt. Eine Abgrenzung zu naheliegenden aktuellen Forschungsarbeiten fand in Kapitel 10 Ausdruck.

Im dritten Teil schließlich wurde versucht, den Nachweis zu führen, dass ein geeignetes Konzept gefunden wurde. Dafür wurden in Kapitel 11 konkrete Beispiele für jeden der Aspekte isoliert vorgestellt, bei denen jeweils das Konzept des virtuellen Prozessors zur Implementation erfolgreich verwendet wurde. Anschließend wurde in Kapitel 12 ein größeres Beispiel aus dem Bereich der industriellen Praxis vorgestellt, bei dem das erarbeitete Konzept der virtuellen Prozessoren für alle drei Aspekte gleichzeitig herangezogen werden konnte.

Zusätzlich zu den Vorteilen der Zusammenfassung der drei Aspekte Verteilung, Nebenläufigkeit und Persistenz zu einem Konzept konnten einige weitere Vorteile herausgearbeitet werden. Bei einer getrennten Behandlung dieser Aspekte ist eine Migration nur kaum möglich und praktische Umsetzungen wie durch Voyager weisen deutliche Nachteile auf, die in Abschnitt 2.2.3 aufgezeigt