1. Einleitung und Überblick 1

2. Programmiersprachen für verteilte Anwendungen 6
2.1. Kommunikationskonstrukte in einigen nichtsequentiellen Sprachen 8
2.1.1. Parlog 8
2.1.2. Linda 9
2.1.3. Orca 10
2.1.4. SR (Synchronizing Resources) 11
2.1.5. Emerald 12
2.1.6. SINA 13
2.1.7. Concurrent C++ 14
2.1.8. Occam 15
2.2. Objektorientierung und Verteilung — Probleme 16
2.2.1. Paradigmen objektorientierter Programmierung 16
2.2.2. Sprachkonzepte für Nichtsequentialität 18
2.2.3. Aus Wiederverwendung resultierende Sprachanforderungen 23
2.2.4. Schlußfolgerungen 27

3. Der Objektraum-Ansatz — Kommunikation mit Klassen 28
3.1. Linda im Detail 28
3.2. Objektorientierung und lose gekoppelte Kommunikation 29
3.3. Der Objektraum-Ansatz 31
3.4. Objektmodell 32
3.5. Die Objektraum-Sprache 34
3.6. Assoziative Adressierung 36
3.7. Objektraum versus Linda — Einordnung 39
3.8. Verwandte Arbeiten 41

4. Prozeßinteraktionen in verteilten Programmen 44
4.1. Datenfluß durch Filter 45
4.2. Klienten und Server 51
4.3. Replizierte Prozesse 54

5. Ein Ansatz für nichtsequentielles Programmieren in C++ 59
5.1. Prinzipielle Möglichkeiten 59
5.2. Nichtsequentialität auf Basis des Objektraum-Ansatzes 60
5.3. Ein Beispiel 63
5.4. Entwurf einer Klasse aktiver Objekte 66
5.5. Schlußfolgerungen 71

6. Prototypimplementationen verteilter Anwendungen 72
6.1. Generierung verteilter Implementationen aus LOTOS Spezifikationen 73
6.2. Verteilte Kommunikation über einen Kommunikationsbaum 75
6.2.1. Der Algorithmus 75
6.2.2. Implementation des Algorithmus im Objektraum 78
6.2.3. Optimierung der Implementation 81
6.3. Kommunikation über globale "Ereignisobjekte" 82
6.3.1. Der Algorithmus 82
6.3.2. "Ereignisobjekte" 83
6.3.3. Deterministische Synchronisation 84
6.3.4. Prozesse mit *Interleaving*-Semantik 86
6.3.5. Nichtdeterministische Synchronisation 88
6.4. Analyse der Algorithmen, Erweiterungen 91
6.4.1. Berücksichtigung von Datenstrukturen in $LOTOS_0$ 91
6.4.2. Vergleich der Algorithmen 92

7. Anwendungen — Kommunikation über den Objektraum 94
7.1. Ein paralleles, verteiltes `make` 94
7.1.1. Koordination paralleler Aktionen über den Objektraum 95
7.1.2. Der grundlegende Ansatz 97
7.1.3. Umgang mit Fehlerzuständen, Terminierung 101
7.2. Ein verteilter Zeitdienst 104
7.2.1. Definition eines Kommunikationsprotokolls 104
7.2.2. Erweiterung des Kommunikationsprotokolls 106

7.3. Ein verteiltes Firmentelefonbuch 108
7.3.1. Objekte als Datenkapseln 108
7.3.2. Neue Objektraum-Operationen 110

8. Die Prototypimplementation des Objektraum-Ansatzes 111
8.1. Ein typisches Szenario: Klienten benutzen den Objektraum 111
8.2. Die Schnittstelle zu Klientenprogrammen — `libos++` 113
8.2.1. Vererbung von Funktionalität 113
8.2.2. Verteilter Typdienst 114
8.2.3. Verteilte Prozeßerzeugung 117
8.2.4. Der ORS-Präprozessor 119
8.3. Kommunikation zwischen Klienten und Objektraum 121
8.3.1. Das Protokoll auf den UNIX *message queues* 121
8.4. Objektraum-Managerprozesse 124
8.4.1. Speicherung von Objekten 124
8.4.2. Bearbeitung von Operationsanforderungen 124
8.4.3. Implementation von Paßprädikaten 127
8.4.4. Prozeßstruktur eines Objektraummanagers 128
8.5. Beschleunigung von Objektraum-Operationen 130
8.5.1. Effizienz der Prototypimplementation 130
8.5.2. Eine effizientere Implementation des Objektraums 132
8.5.3. Andere Implementationskonzepte 138

9. Zusammenfassung, Ausblick 140

10. Bibliographie 145