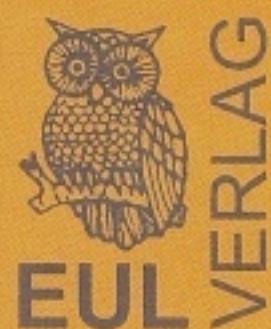
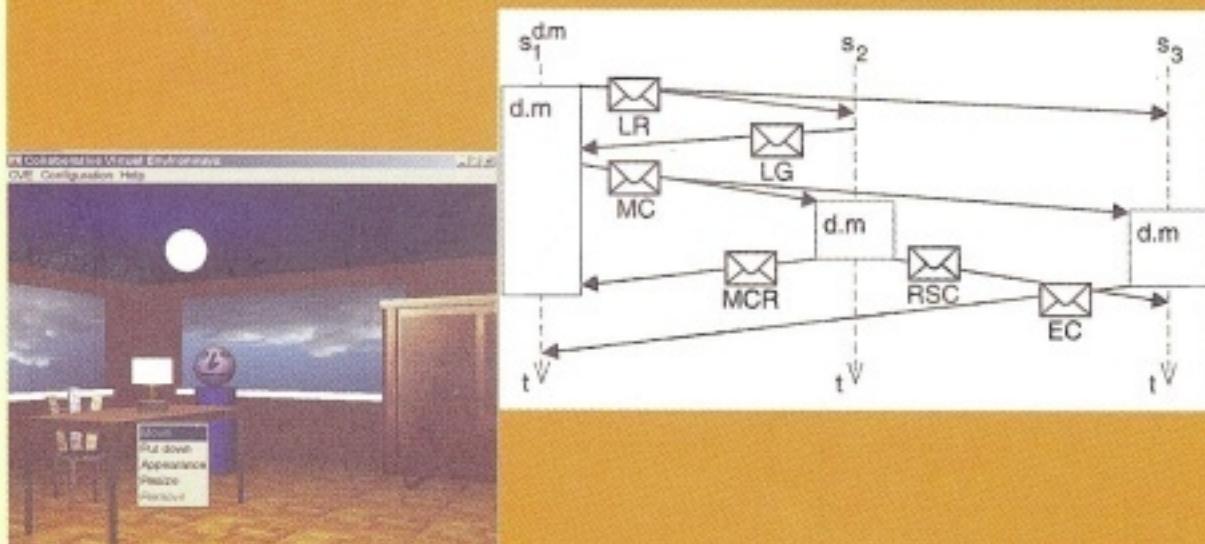


Schriften zu Kooperations- und Mediensystemen · Band 2

Herausgegeben von Volker Wulf, Jörg Haake, Thomas Herrmann, Helmut Krcmar, Johann Schlichter, Gerhard Schwabe und Jürgen Ziegler

Stephan Lukosch

Transparent and Flexible Data Sharing for Synchronous Groupware



Contents

Preface	v
Acknowledgements	vii
Contents	ix
List of Figures	xiii
List of Tables	xv
Nomenclature	xvii
1 Introduction	1
1.1 Problem Statement and Approach	3
1.2 Organization of the Thesis	5
2 System Requirements	7
2.1 DreamTeam	8
2.2 Requirements in the Data Domain	12
2.2.1 Example Application	13
2.2.2 Data Distribution	14
2.2.3 Data Consistency	19
2.2.4 Concurrency Control	20
2.2.5 Data Persistency	23
2.3 Summary	24
3 Related Work	27
3.1 Central State	27
3.2 Replicated State	30
3.3 Flexible State Distribution	35
3.4 Summary	40
4 Approach	43
5 Notification service	47
5.1 Event-based Notification	48
5.1.1 Object-listener	49

5.1.2 Call-listener	49
5.2 Mapping-based Notification	51
6 Concurrency Control	53
6.1 Object-based Concurrency Control	53
6.2 Method-based Concurrency Control	54
6.2.1 Intra-object Concurrency Control	55
6.2.2 Inter-object Concurrency Control	55
6.2.3 Example	56
6.3 Extensibility and Adaptability	58
6.3.1 Locks	58
6.3.2 Concurrency Control Schemes	59
7 Flat Distributed Actions	61
7.1 Object Registration	61
7.1.1 Substitute Generation	62
7.1.2 Initialization and Configuration	63
7.1.3 Object Creation and Distribution	65
7.2 Object Duplication	70
7.3 Object Dereistration	73
7.4 Flat Method Calls	75
7.4.1 Flat Modifying Method Call	76
7.4.2 Reading Method Call	81
8 Nested Distributed Actions	85
8.1 Follow-up Modifying Method Call	88
8.1.1 Initialization Phase	88
8.1.2 Pre-concurrency Control Phase	89
8.1.3 Call-distribution Phase	91
8.1.4 Call-execution Phase	92
8.1.5 Result-distribution Phase	92
8.1.6 Post-concurrency Control Phase	93
8.1.7 Post-execution Phase	93
8.1.8 Example	94
8.2 Follow-up Reading Method Call	95
8.2.1 Initiating Modifying Distributed Action	96
8.2.2 Initiating Reading Method Call	98
8.3 Follow-up Object Registration	99
8.3.1 Initialization Phase	101
8.3.2 Call-distribution Phase	102
8.3.3 Call-execution Phase	102
8.3.4 Concurrency Control Phase	103
8.3.5 Post-execution Phase	103
8.4 Follow-up Object Duplication	104
8.4.1 Initialization Phase	105
8.4.2 Pre-concurrency Control Phase	106

8.4.3	Call-distribution Phase	106
8.4.4	Call-execution Phase	106
8.4.5	Post-concurrency Control Phase	106
8.4.6	Post-execution Phase	106
8.4.7	Example	107
8.5	Follow-up Object Dereistration	108
9	Data Persistency	111
9.1	Shared Data Objects	112
9.1.1	Storage	112
9.1.2	Loading	114
9.2	Session State	116
9.2.1	Storage	116
9.2.2	Loading	117
9.2.3	Distribution	118
10	Latecomer Support	121
10.1	Direct State Transfer	122
10.1.1	Connection Phase	125
10.1.2	Initial Phase	125
10.1.3	Final Phase	128
10.2	Replay	133
10.2.1	Connection Phase	134
10.2.2	Initial Phase	135
10.2.3	Final Phase	136
11	Data Distribution	139
11.1	Static Distribution Schemes	139
11.2	Adaptive Distribution Schemes	140
11.2.1	User-oriented Adaptive Distribution	143
11.2.2	Network-Oriented Adaptive Distribution	145
11.2.3	State Transmission Protocol	147
12	System Implementation	151
12.1	Communication Layer	153
12.2	Data Layer	155
12.3	Service Layer	157
12.4	Application Layer	161
13	Application Development and Reuse	165
13.1	Diagram Editor	167
13.1.1	Step One	168
13.1.2	Step Two	169
13.1.3	Step Three	173
13.2	Sketch Editor	173
13.3	File Viewer	174

13.4 Spreadsheet	175
13.5 Discussion	176
14 Conclusions	179
14.1 Summary	179
14.2 Comparison to Related Work	182
14.3 Future Work	183
Bibliography	185
Index	199

JOSEF EUL VERLAG

Schriften zu Kooperations- und Mediensystemen

Synchronous groupware brings together users, who are geographically distributed and connected via a network. It encompasses a wide range of applications like collaborative whiteboards, text editors or virtual environments.

Developing synchronous groupware is a difficult and time-consuming task, an application has to establish network connections between the collaborating users, handle parallel input from many users, has to offer group functions for collaboration-awareness, and has to manage shared data. The latter belongs to the main obstacles during the development of synchronous groupware.

This thesis describes *DreamObjects*. *DreamObjects* is a platform that focuses on shared data management. It offers flexible and extensible solutions for the recurring problems that are related to data sharing and achieves a maximum of transparency for the developer, as it hides all necessary mechanisms for managing the shared state.

DreamObjects supports a variety of distribution schemes. The data of an application can be distributed to one developer-selected site, all participating sites, or a dynamically changing set of the participating sites. Similarly, a developer can choose between different mechanisms for concurrency control. If the offered solutions do not meet the requirements of the application, developers can even integrate their own solutions in the platform.

A user, who joins a session, can choose, whether he wants to join with a replay of the session or with a direct state transfer. Both mechanisms are fault-tolerant, decentralized, and do not block current participants of the session in their work. Finally, *DreamObjects* offers a decentralized persistency service to support the transition between asynchronous and synchronous work.



Stephan Lukosch, born 1973 in Herne, Germany, studied computer science from 1993 to 1998 at the University of Dortmund. Since 1998, he is working as a researcher at the institute for cooperative systems at the University of Hagen. In June 2003, he received a Dr. rer. nat. in computer science from the University of Hagen. His research interests include development and runtime support for synchronous groupware applications. During his work at the University of Hagen, he developed a platform that simplifies the development of synchronous groupware applications by relieving developers from data sharing issues.