Laboratory for Computer Science
Computer Systems Research Division

Request for Comments No. 193
August 11, 1980

ENGINEERING RELIABLE COMPUTER SYSTEMS

Selected Bibliography

by Liba Svobodova

This document contains a list of selected papers that deal with different aspects of the design and implementation of reliable systems, with emphasis on distributed information processing systems. Admittedly, this bibliography is incomplete. First, I may not be aware of some important papers. Second, I want to keep this list selective.

The bibliography is divided into several sections, reflecting different topics. Some papers on the list deal with more than one topic, and may be found under more than one category. The bibliography will hopefully grow with time. Also, it is planned that annotations will be added.

A. Reliable Computer Systems: Terminology, Models, Surveys

AVIZ 77 — Avizienis, A., et al., "Fault Tolerant Computing: Progress Problems, and Prospects," Proc. of IFIP Congress 1977, Toronto, Canada, August, 1977.

This paper concentrates mainly on hardware failures, but includes also a brief discussion of the problem of software design errors (work of the group from the University of Newcastle upon Tyne). It techniques for reliability evaluation. Good survey.

AVIZ 78 — Avizienis, A., "Fault-Tolerance: The Survival Attribute of Digital Systems," Proc. of the IEEE, Vol. 66, No. 10, October, 1978, pp. 1109-1125.

The paper presents the concept of fault-tolerance and the methods for achieving fault-tolerance for both physical and man-made (design) faults. It includes a brief history of fault-tolerant system designs and interesting reliability statistics for the CRAY-1 computer.

BEAU 77 — Beaudry, D.M., "Performance-Related Measures for Computing Systems," Stanford University, Center for Reliable Computing, 1977.

New reliability measures that take into consideration possibly varying levels of performance (computational capacity) due to failures of system components are defined. Performance-related measures are derived for a gracefully degrading system and a multiprocessor system with repair and used to compare two-processor configurations of these systems with corresponding systems where the computational capacity remains fixed while the system is operational. Though the systems examined were drastically simplified, the approach demonstrated in this paper, that is, relating performance and reliability is an important step in computer system evaluation.

MERL 77 Merlin, P.M., et al., "Consistent State Restoration in Distributed Systems," Technical Report No. 113, University of Newcastle Upon Tyne, Newcastle Upon Tyne, England, October, 1977.

The notion of a backward eror recovery is formalized using Occurence Graphs, a cause-effect model based on Petri Nets. The paper defines backward error recovery in the context of this model and presents a protocol for so called weak recovery in a distributed system. The paper touches on the important problem of dependencies that might be hidden at the level of abstraction modeled by an Occurrence Graph and suggests an extention to the model that facilitates inclusion of information about lower level dependencies in the form of constraints. Overall, use of Occurrence Graphs for error recovery is an interesting approach; however, to demonstrate its real potential, it is necessary to apply it to specific recovery problems.

RAND 78 Randell, B., et al., "Reliability Issues in Computing System Design," Computing Surveys, Vol. 10, No. 2, June 1978, pp. 123-165.

A good overview. It defines reliability oriented terms such that they are implementation independent and applicable at any level of abstraction. The appendix contains analysis of several fault-tolerant systems, described in the terms developed in the paper.

WULF 75 Wulf, W.A., "Reliable Hardware/Software Architecture," IEEE Trans. on Software Engineering, SE-1, 2, June, 1975, pp. 233-240.

This paper argues that software correctness is not a sufficient condition for system reliability: it is necessary to take into account possible hardware failures. It advocates run-time type checking and implementation of data abstractions that allows for testing of errors in the internal data structures.

B. Reliable Computer Systems: Specific Designs

AVIZ 77	Avizienis, A., et al., "The STAR (Self-Testing and Repairing) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design," IEEE Transactions on Computers, C-20, 11, November, 1971, pp. 1312-1321.
BART 78	Bartlett, J.F., "A 'Non-Stop' Operating System," IEEE Hawaii International Conference of System Sciences, January, 1978, pp. 103-117.
BASK 72	Baskin, H.B., et al., "PRIME - A Modular Architecture for Terminal-Oriented Systems," Proc. AFIPS SJCC 1972, pp. 431-437.
KATZ 78	Katzman, J.A., "A Fault-Tolerant Computing System," IEEE Hawaii International Conference of System Sciences, January, 1978, pp. 85-102.
ORNS 76	Ornstein, S.M., et al., "Pluribus - A Reliable Multiprocessor," Proc. of AFIPS NCC, 1975, pp. 551-559.
ROBI 78	Robinson, J.G. et al., "Software Fault-Tolerance in the Pluribus," Proc. of AFIPS NCC, 1978.
WENS 76	Wensley, J.H. et al., "The Design, Analysis, and Verification of the SIFT Fault Tolerant Software," Proc. of International Converence on Software Engineering, San Francisco, California, October, 1976, pp. 458-468.

C. Error Resistant Software

ANDE 76 Anderson, T., et al., "Recovery Blocks in Action: A System Supporting High Reliability," Proc. of International Conference on Software Engineering, San Francisco, California, October, 1976.

ANDE 78 Anderson, T., et al., "A Model of Recoverability in Multi-level Systems," IEEE Trans. on Software Engineering, Vol. SE-4, No. 6, November, 1978.

Multilevel schemes can be interpretive, where each level is completely interpreted by the next lower level, or extended, where several levels are supported by the same interpreter, but each level provides some new abstract objects to the next higher level. Recovery techniques are discussed for both types of multilevel systems. For the extended interpreter case, two different recovery schemes are described: disjoint and inclusive. In the disjoint scheme, recovery of a calling program does not necessitate recovery of the underlying extentions, whereas in the inclusive scheme all extentions are recovered by the interpreter. The paper does not discuss explicitly the problem of system crashes: some form of inclusive scheme would have to be used, but additional precautions, not considered by the authors, would have to be taken in the implementation of recoverable extended objects.

ANDE 79 Anderson, T., and Lee, P.A., "The Provision of Recoverable Interface," The Ninth Annual International Symposium on Fault-Tolerant Computing, Madison, Wisconsin, June 1979, pp. 87-94

This paper restates many of the ideas presented in ANDE 78. It concentrates on the problem of recovery in multilevel systems where the levels are implemented as extentions of some basic interpreter. The concept of disjoint and inclusive recovery is demonstrated in an example of a simple file system.

BANA 77 Banatre, J-P., et al., "Reliable Resource Allocation Between Unreliable Processes," Technical Report No. 99, University of Newcastle Upon Tyne, Newcastle Upon Tyne, England, April, 1977.

FORS 77 Forsdick, H.C., "Responding to Errors in a Computer System," M.I.T., Laboratory for Computer Science, Computer Systems Research Division, Request for Comments No. 144, June, 1977.

GOOD 75 Goodenough, J.B., "Exception Handling: Issues and a Proposed Notation," CACM, Vol. 18, No. 12, December 1975, pp. 683-696.

LEE 78 Lee, P.A., "A Reconsideration of the Recovery Block Scheme," University of Newcastle Upon Tyne, England, Technical Report No. 119, January, 1978.

LEVI 77 Levin, R., "Program Structures for Exceptional Condition Handling," Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, June, 1977.

Levin's exception handling mechanism can handle, in addition to the exceptions arising directly from an execution of a specific program (flow class conditions) exceptions pertaining to shared data structures (structure class conditions). A set of criteria for evaluation of exception handling mechanisms is defined and a number of existing and proposed mechanisms are discussed in this light. The new mechanism is quite complex, and it is questionable whether such power and flexibility is indeed needed. Overall, the report is a good exposition of the problems involved and the possible approaches to their solution.

LISK 77 Liskov, B., et al., "Structured Exception Handling," M.I.T. Laboratory for Computer Science, Computer Structures Group, CSG Memo 155, December, 1977 (revised).

The paper argues that the termination model for exception handling is more compatible with the structured design methodologies than the resumption model, and that is sufficient for the foreseeable needs. Syntax and semantics of the CLU exception handling mechanism are described and demonstrated by examples. The paper is well written, with a clear statement of the goal, and well reasoned arguments to justify the design decisions.

MELL 77 Melliar-Smith, P.M., Randell, B., "Software Reliability: The Role of Programmed Exception Handling," Proc. of ACM Conference on Language Design for Reliable Software, (Operating Systems Review, Vol. 11, No. 2, April 1977).

PARN 76 Parnas, D.L., et al., "Response to Undesired Events in Software Systems," Proc. of International Conference on Software Engineering, San Francisco, California, October, 1976, pp. 437-446.

RAND 75 Randell, B., "System Structure for Software Fault Tolerance," IEEE Trans. on Software Engineering, SE-1, 2, June, 1975, pp. 220-232.

The first part of the paper explains the concept of recovery blocks in sequential programs. The second part discusses the error recovery problems amongst interacting processes. Finally, the paper discusses the recovery problems in multilevel systems. This last part is rather murky; it tries to distinguish between conceptual system levels and levels of interpreters -- a confusing problem, but worth investigating.

SHRI 78 Shrivastava, S.K., Banatre, J.-P., "Reliable Resource Allocation Between Unreliable Processes," IEEE Trans. on Software Engineering, Vol. SE-4, No. 3 May 1978, pp. 230-241.

Concurrent processes can interact in two different ways: by cooperating on the same task, and by competing for shared resources. The paper studies the problem of recoverability of competing processes. Shared resources are controlled by recoverable monitors. However, recoverable monitors must be used through a new program structure called a port. A port contains forward procedures that specify what is to be done with the shared resource under normal condition and backward procedures that will undo the effects of the forward procedures. The proposed scheme can handle algorithmic errors and errors in the input and output (including operations on secondary storage); it is not designed to deal with processor failures, although such an extention might be possible.

VERH 76 Verhofstad, J.S.M., "Recovery for Mult-Level Data Structures," Technical Report 96, University of Newcastle Upon Tyne, Newcastle Upon Tyne, England, December, 1976.

VERH 77 Verhofstad, J.S.M., "On Multi-Level Recovery: An Approach Using Partially Recoverable Interfaces," Technical Report 100, University of Newcastle Upon Tyne, Newcastle Upon Tyne, England, May, 1977.

YAU 76 Yau, S.S., *et al.*, "An Approach to Error-Resistant Software Design," *Proc. of International Conference on Software Engineering*, San Francisco, California, October, 1976, pp. 429-436.

The paper discusses an approach for building error-resistant application software. The error detection and recovery takes place on three different levels: the module level (similar to the recovery block method developed at the University of Newcastle upon Tyne), the program level (flow of control and data between modules) and the system level (communication with other processes, use of global shared data). The error detection and recovery code is protected from the rest of the program; the paper suggests use of protection mechanisms similar to the rings of protection in the Multics System. The paper contains some good ideas, but ignores a great number of problems and lacks clarity.

D. Coping With Design Faults (By Eliminating or Masking)

CHEN 78	Chen, L., Avizienis, A., "N-Version Programming: A Fault-Tolerance Approach to Reliability of Software Operation," The Eighth Annual International Conference on Fault-Tolerant Computing, Toulouse, France, June 1978, pp. 3-9.
FABR 73	Fabry, R.S., "Dynamic Verification of Operating System Decisions," CACM, Vol 16, No. 11, November, 1973.
NEUM 76	Neumann, P.G., et al., "Software Development and Proofs of Multi-Level Security," Proc. of International Conference on Software Engineering, San Francisco, October, 1976, pp. 421-428.
RAND 75	Randell, B., "System Structure for Software Fault tolerance," IEEE Transactions on Software Engineering, SE-1, 2, June, 1975, pp. 220-232. (See also Part C.)
SCHR 77	Schroeder, M.D., et al., "The Multics Kernel Design Project," Proc. of the Sixth Symposium on Operating Systems Principles, November, 1977, pp. 43-56.

E. Software Testing

GOOD 77	Goodenough, J.B., "Survey of Program Testing Issues," Softech Inc., Massachusetts, 1977.
SCHI 78	Shick, J.G., et al., "An Analysis of Computing Software Reliability Models," IEEE Trans. on Software Engineering, SE-4, 2, March 1978.
ANDR 79	Andrews, D., "Using Executable Assertions for Testing and Fault Tolerance," The Ninth Annual International Symposium on Fault-Tolerant Computing, Madison, Wisconsin, June 1979, pp. 102-105

F. Recovery Strategies For Data Base Systems

BJOR 73 Bjork, L., "Recovery Scenario for a DB/DC System," Proc. of the ACM National Conference, 1973, pp. 142-146.

DAVI 73 Davies, C.T., "Recovery Semantics for a DB/DC System," Proc. of ACM National Conference, 1973, pp. 136-141.

DAVI 77 Davies, C.T., "Data Processing Spheres of Control," IBM Systems Journal, Vol. 17, No. 2, 1978, pp. 179-198.

GRAY 77 Gray, J.N., "Notes on Data Base Operating Systems," Advanced Course on Operating Systems, Technical University, Munich, Germany, 1977.

Gray's lecture notes emphasize the need to integrate solutions to two complex problems: synchronization of concurrent transactions on a data base and recovery from system failures. In particular, recovery from synchronization errors (deadlocks) and recovery from failures in the underlying hardware and software can be handled by a unified mechanism. The problem of synchronization and recovery is extended to distributed data bases, through a two-phase commit protocol. These lecture notes are still more or less in a draft form, often sketchy, the reasons for some of the details of the described protocols is left to the imagination of the readers. In spite of these problems, the notes present a more complete view of reliable transaction management than any other publication.

GRAY 79 Gray, J.H., et. al., "The Recovery Manager of a Data Management System," IBM Research Division, Technical Report No. RJ2623, August 1979.

This report finally presents the recovery of RSS, the storage subsystem of System R, as it is actually implemented. The recovery manager uses a combination of shadow files and undo/redo log. The shadow files represent a consistent state of RSS; they serve as a system checkpoint. Recovery of individual transaction from a system crash consists of undoing all actions preceding the system checkpoint for those transactions that have not committed prior to the crash and redoing all actions following

the system checkpoint for those transactions that have committed. New recoverable object types can be easily incorporated into this scheme.
A critical evaluation of the recovery manager is included. The recovery manager as implemented performs very successfully. Although providing such comprehensive recoverability is quite expensive, it accounts only for about 10 percent of the total system cost. The main drawback of the system R recovery scheme is the use of shadow files, which are very expensive for large shared files. Also, save points for individual transactions are not as powerful as they ought to be, due to the underlying operating system.

LIND 79 Lindsay, B.G. et al., "Notes on Distributed Databases," IBM Research Laboratory Report RJ2571, San Jose, California, July 14, 1979.

The notes cover several topics, in varying depth. Chapter 1 surveys the problem of replicated data. Chapter 2 develops quite a complex mechanism and protocols for access authorization in database systems. The remainder is devoted to transaction management. Chapter 4 describes a sophisticated transaction recovery scheme based on a transaction log. The same material was covered in GRAY 77, however, these notes present it with much better clarity. The last chapter presents a robust transaction management scheme for distributed systems that supports quite general patterns of transactions. Highly recommended reading.

LORI 77 Lorie, R.A., "Physical Integrity in a Large Segmented Data Base," ACM Transactions on Data Base Systems, Vol. 2, No. 1, March, 1977, pp. 91-104.

STER 73 Stern, J., "Organization and Operation of the Multics Backup System," Project MAC Internal Document, Multics Checkout Bulletin (MCB) 1077, March 23, 1973.

VERH 78 Verhofstad, J.S.M., "Recovery Techniques for Database Systems," Computing Surveys, Vol. 10, No. 2, June 1978, pp. 167-195.

Verhofstad defines recovery as the restoration of the database after a failure to a state that is acceptable to the users. Several different "states" that may be targets of a recovery in different situations are defined. These states reflect the completeness and the consistency of the information in the database. Recovery techniques are summarized according to the type of database state that can be restored with each technique. Individual techniques are illustrated by examples and related to the other techniques; the complementary aspects of different techniques are emphasized. I found the classification used in the paper quite useful, in particular where it is applied to explain recovery mechanisms used in different systems.

G. Atomic Updates In Distributed Systems

GRAY 77	Gray, J.N., "Notes on Data Base Operating Systems," Advanced Course on Operating Systems, Technical University, Munich, Germany, 1977. (See also Part F.)
HAMM 79	Hammer, M.M., Shipman, D.,"Reliability Mechanisms for SDD-1: A System for Distributed Databases," Computer Corporation of America and Massachusetts Institute of Technology, July 31, 1979.
LAMP 79	Lampson, B. et al., "Crash Recovery in a Distributed Data Storage System," XEROX Palo Alto Research Center, California, April 1979 (to be published in CACM). The first version of this paper was written and began circulating in 1976; this version is often cited as the source of the two-phase commit protocol. The new version has a remarkably different emphasis. It presents implementation of a "stable" system as a lattice of abstractions built from realistically unreliable physical components. Atomic transactions that involve several physical nodes are easy to implement, using the two-phse commit protocol, if all nodes are stable systems. Highly recommended.
LIND 79	Lindsay, B.G., et al., "Notes on Distributed Databases," ITM Research Laboratory Report RJ2571, San Jose, California, July 14, 1979. (See also Part F.)
MONT 78	Montgomery, W.A., "Robust Concurrency Control for a Distributed Information System," Massachusetts Institute of Technology, Laboratory for Computer Science TR-207, January, 1979.
REED 78	Reed, David P., "Naming and Synchronization in a Decentralized Computer System," Massachusetts Institute of Technology, Laboratory for Computer Science TR-205, October, 1978.

SHAP 77 Shapiro, R.M., et al., "Reliability and Fault Recovery in Distributed Processing," Oceans '77 Conference Record, Vol. II, Los Angeles, California, October, 1977, pp. 31D-1 --31D-5.

SHAP 78 Shapiro, R.M., et al., "Failure Recovery in a Distributed Data Base System," COMPCON Spring '78, February, 1978.

STON 78 Stonebraker, M., "Concurrency Control and Consistency of Multiple Copies of Data in Distributed Ingres," Proc. of the Third Berkeley Workshop on Distributed Data Management and Computer Networks, Lawrence Berkeley Laboratory, University of California, Berkeley, August 1978, pp. 235-258.

TAKA 79 Takagi, Akihiro, "Concurrent and Reliable Updates of Distributed Databases," M.I.T., Laboratory for Computer Science, Computer Systems Research Division," Massachusetts Institute of Technology, Laboratory for Computer Science, TM-135, November, 1979.

TRAI 79 Traiger, I.L., Gray, J.N., Galtieri, C.A., Lindsay, B.G., "Transactions and Consistency in Distributed Database Systems," IBM Research Division, Technical Report No. RJ2555, June 1979.

Reliability (failure transparency) is only a small part of this report, however, the report presents a good model of transactions that is important for reasoning about reliability of distributed systems. A transaction is an abstraction that represents a logical unit of work and that provides location transparency, replication transparency, concurrency transparency, and failure transparency. The notion of consistency and the protocols that ensure consistency developed in ESWA 76 are extended in this report to distributed systems.

H. Replicated Databases

ALSB 76 Alsberg, P.A., "A Principle for Resilient Sharing of Distributed Resources," Proc. of International Conference on Software Engineering, San Francisco, California, October, 1976, pp. 562-570.

GARC 79 Garcia-Molina, H., "Performance of Update Algorithms for Replicated Data in a Distributed Database," Stanford University, Stanford Computer Science Laboratory Memo CSL TR-172, Department of Computer Science Report No. STAN-CS-79-744, June 1979.

MENA 78 Menasce, D.A., Popek, G.J., Muntz, R.R., "A Locking Protocol for Resource Coordination in Distributed Databases," submitted to ACM Transactions on Database Systems, 1978.

The paper presents a centralized locking scheme for distributed systems that includes recovery protocols to deal with failures of the participating nodes and network partitioning. Three disjoint recovery mechanisms are provided: recovery of a single node, replacement of a failed centralized controller, and merge of network partitions. The locking scheme seems to carry with it a substantial overhead (each node maintains a complete lock table for data items in all accessible nodes), and the proofs presented to demonstrate the infallibility of the protocols are incomplete; in particular, the fact that there is a critical time interval during which no node may make an independent decision about the fate of a lock request (granted or refused) is ignored.

I. Robust Communication Protocols

REED 76	Reed, D.P., "Protocols for the LCS Network," M.I.T., Laboratory for Computer Science, Local Network Note No. 3, November 29, 1976.
REED 77	Reed, D.P., "A Protocols for Addressing Services in the Local Net," M.I.T. Laboratory for Computer Science, Local Network Note No. 5, February 15, 1977.

J. Reliable Communication Networks

LEE 79	Lee, R., "The Architecture of a Dynamically-Reconfigurable Insertion-Ring Network," IBM Research Laboratory Report RJ2485(32434) 3/13/79, San Jose, California.
CLAR 77	Clark, D.D., "A Contention Ring Network," M.I.T. Laboratory for Computer Science Local Network Note No. 11, September 1977.
SALT 79	Saltzer, J., and Pogran, K., "A Star-Shaped Ring Network with High Maintainability," Proceedings of the NBS-Mitre Local Area Communications Network Symposium, Boston, Massachusetts, May 7-9, 1979.

K. Copying With Errors in the Input to a System

HAMM 76	Hammer, M., "Error Detection in Data Base Systems," Proc. of AFIPS NCC, 1976, pp. 795-801.