LiTH-IDA-R-88-19 ISSN-0281-4250

INSTITUTIONEN FÖR DATAVETENSKAP
UNIVERSITETET OCH TEKNISKA HÖGSKOLAN
LINKÖPING

From a Technical to a Humane Environment: A Software System Supporting Cooperative Work

by

Lin Padgham
Ralph Rönnquist

RESEARCH REPORT
RKLLAB, August 1988

Postadress:
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

Mailing address:
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden

The Department of Computer and Information Science
Linköping University

PhD theses:

* Available at: University of Microfilms Intl., 300 N. Zeeb Road, Ann Arbor, MI 48106, USA.

(*Linköping Studies in Science and Technology. Dissertations.*)

*No 77	**Östen Oskarsson:** Mechanisms of Modifiability in large Software Systems, 1982.
No 94	**Hans Lunell:** Code Generator Writing Systems, 1983.
No 97	**Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983.
*No 109	**Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation, 1984.
*No 111	**Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984.
*No 155	**Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons. 1987.
No 165	**James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning. 1987.
No 170	**Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems. 1987.

Licentiate of engineering Theses:

(*Linköping Studies in Science and Technology. Theses.*)

No 48	**Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching. 1985.
No 52	**Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems. 1985.
No 60	**Johan Fagerström:** Simulation and Evaluation of an Architecture based on Asynchronous Processes. 1985.
No 71	**Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System. 1987.
No 72	**Tony Larsson:** On the Specification and Verification of VLSI Systems. 1986.
No 73	**Ola Strömfors:** A Structure Editor for Documents and Programs. 1986.
No 74	**Christos Levcopoulos:** New Results about the Approximation Behaviour of the Greedy Triangulation. 1986.
No 104	**Shamsul I. Chowdhury:** Statistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology. 1987.
No 108	**Rober Bilos:** Incremental Scanning and Token-based Editing. 1987
No 111	**Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments. 1987.
No 113	**Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge. 1987.
No 118	**Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs. 1987
No 126	**Dan Strömberg:** Transfer and Distribution of Application Programs. 1987
No 127	**Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems. 1987
No 139	**Christer Bäckström:** Reasoning about Interdependent Actions. 1988.
No 140	**Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing. 1988.

LiTH-IDA-R-88-19 ISSN-0281-4250

From a Technical to a Humane Environment: A Software System Supporting Cooperative Work

by

Lin Padgham

Ralph Rönnquist

Abstract: This paper presents LINCKS, an environment which supports co-operative work. LINCKS is presently in the stage of beginning implementation at Linköping University.

Postadress:
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

Mailing address:
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden

1. Introduction

Co-operative work is seen as desirable and/or necessary for a number of different reasons. In many cases the size of a project combined with the requirements on when it must be finished, dictate that it be done by a number of people working together. In other cases the breadth of knowledge required means that a number of people with different backgrounds must be involved.

There is also the aspect that discussion of ideas with others intimately involved with those ideas, leads to greater creativity and exploration of those ideas, than merely thinking about them in isolation. The discussion and working together has a catalytic effect that produces higher quality work than would have been produced by any one of the participants alone.

Unfortunately co-operative work also results in some problems and difficulties. These sometimes outweigh the advantages, and prevent co-operative work from happening.

One of the difficulties of co-operative work is that the parameters 'who', 'what', 'where', 'when', 'why', 'how', must be co-ordinated. These parameters must be resolved for any given piece of work, but as more people are involved, the complexity of deciding and administrating increases sharply.

Co-operative work can be compared with respect to the degree of sameness for the question word parameters. For instance, co-operative work where the participants are at different places has a very different feel, and involves different problems, than when all participants are in aproximately the same place (*where* parameter). Similarly, a co-operative project where all people are working at the same time, raises different issues than a project where the work is sequenced (*when* parameter).

Computer systems which support co-operative work can also be viewed in terms of these parameters. There are then two basic questions whose answers give a good sense of both the flexibility of the computer system and its power. First is the question of how much the question word parameters are allowed to vary (particularly with respect to a same-different scale). This provides a reasonable measure of flexibility. Second is the question of how much the system knows about the parameters, i.e. have the ability to represent and reason about information concerning them. This indicates the power of the system.

The challenge is to develop tools which help to solve the inherent difficulties of co-operative work, without constraining the solutions,thus avoiding a perhaps helpful but narrow system.

In the present paper we look at the specific problems of co-operative work in a research environment, and describe a system whose intent is to facilitate solutions to those problems. The system, LINCKS, is under development in the Department of Computer and Information Science, at Linköpings University, Sweden. It is currently at the stage of beginning implementation.

2. Research Environment Focus

Tools that have been developed to support co-operative work (e.g. project management applications, revision control systems[TICH82], etc.) tend to have been developed for the commercial industry environment. They do not consider the nature of a research environment, and consequently are often less suitable for such.

In a research environment there is a greater emphasis on creativity, and less on such aspects as security, deadlines, and profit. Most of the difficulties associated with co-operative work are similar for research and industry environments, but the focus of the tools needed, and the nature of the constraints which are acceptable, are different.

A particular difference between a commercial and a research environment, is the importance placed on individuality, and individual work. In a commercial environment, most often all of

a persons work is on a particular group project. There are company rules defining how things should be done and the process things should go through. In a research environment such as a university department, however, people are almost always involved in important individual work as well as some co-operative work.

As a consequence, the tools developed for a research environment must be more general and more flexible than those developed for commercial purposes. The challenge is to obtain this flexibility, without sacrificing powerfulness. If this is achieved, the resulting system will be customizable to a variety of situations, and therefore also adaptable to the needs in a commercial environment.

3. Description of LINCKS

The LINCKS system consists of an object oriented data base and a number of sophisticated tools and services to make this data base flexible, powerful and useful, both for individual and cooperative work. We describe a model which has been developed as the theoretical basis for the LINCKS system. The model concerns the notions of objects and object types, operations and actions, contexts, and communicating agents that operate through contexts.

3.1 Objects

The LINCKS system is based on a least item called *molecule*. A molecule is, intuitively, a smaller piece of text or a figure, and any whole conceptual object, such as document and folder, is contained in a structure of molecules.

The molecules provide denotions and descriptions of objects, and the database is regarded as a mapping from denotions to descriptions. As descriptions are different amounts of information, they are interrelated according to the notion of "more information".

Object typing is derived from similarities in the object descriptions. That is, an object is of a certain type only if its description incorporates the least amount of information required for that type. In terms of "more information", the object description must contain more information than a type norm.

Each type is also associated to a larger amount of information which provides a normal case or default object description for objects of that type. This description is called the type seed. An object is a normal case object for a certain type only if its description incorporates the type seed.

The setting of type norms and seeds is termed *type schema*. Any such schema is overlaid on the database rather than being part of it. As a consequence, users can build different type schemas for different purposes, thus having classification structures tailored for the tasks.

Objects can also be typed prescriptively so as to account for incomplete information in the database. If an object is prescribed to be of a certain type, any changes in the information concerning the type norm also changes the object description.

3.2 Actions

We distinguish between *actions* and *operations* in the way that each action defines a name for a group of operations. The operations are more or less the actual pieces of code, augumented with descriptions relating to their requirements and effect on object descriptions, i.e. independent of type schemas.

Actions are groupings of operations on the basis of similarities in the interpretations of the operations. E.g. a number of operations may be understood as resulting in an insertion, and are therefore grouped into an insert action. At the time of activation, the appropriate operation is derived from the information contained in the object descriptions present, and the requirements of the operations in the action group.

Actions are augmented with descriptions of when and in what way they may be considered useful. These descriptions are analogous to pre and post conditions found in many expert system languages, and can be used in a similar way.

The grouping of operations into actions is termed *action schema*, and an action schema plus a type schema builds a *context*. Typically, each user and each tool in the system define a context.

3.3 Intelligent Services

Intelligent Services is the term used for software which is able to initiate and do intermediate or high level tasks without direct user control. Many activities done by users, have clearly defined goals, but no clearly defined methods. That is, there are a wide number of possible ways that the goals may be accomplished [BRACC/PERN84]. Intelligent services also have the flexibility to find many alternative ways of achieving goals. If one method fails for some reason they can seek other ways of achieving the desired ends.

Most intelligent services need to be adapted to, or built specially for, some particular situation. The specification of actions and operations along with pre and post conditions simplifies the design and implementation of intelligent services. New actions can be composed from existing actions and operations, in goal oriented ways, by general purpose planning services.

3.4 History Mechanisms

There are three different types of history information which are available in LINCKS.

Object history
: Information about how a given object looked at a particular point in time. (e.g. show me report X as it was when the last printout was made?)

Text history
: Information about the evolution of a certain instance of an object. This is often identical to the above, but at times it is not. (e.g. A working paper may be incorporated into book A as chapter Y. A history of the evolution of that piece of text which is chapter Y of book A, shows the development of the working paper, as well as any development since it was incorporated into book A).

Command history
: Information about the commands that have been performed previously and their results.

The history mechanisms incorporates a notion of parallelism through partially ordered time. Two people may develop the same paper separately for a time, later merging their efforts back into a single version of the paper. The two parallel development paths are stored as such in the history.

The history information allows for the recreation of objects as they existed at any past point in time. It also allows for examination of the effects of actions, and use of this information by intelligent services.

3.5 Portables

An integral part of the LINCKS system is the portable laptop computer. It can be connected to the main system, loading various portions of the data base, to allow work to be done remotely. On return from a remote working session, the new work is integrated with the main data base. It is, of course, also possible to use the portable as a connected workstation.

Portables, due to size and processing limitations, do not supply the complete functionality available on the main system. However, they do 'know about' the main system, so that tasks which cannot be done are queued to be executed later by the main system. Thus, the user has a sense of having done those tasks on the portable (in that he has given the commands and need no longer think about them), even though the execution actually occurs at a later point in time.

4. Cooperation Support in LINCKS

4.1 Support for Group PLUS Individual Views

The support for different views comes largely from the notion of contexts, where each context contains information about the *types* of things recognised, and the *actions* recognised. The system is then able to understand and map between these contexts. As a result, it is not necessary for a user to go into any special mode, or observe any special naming conventions, when working on group projects.

The group view becomes, by default, the combination of all the individual views. An individual can choose to communicate with others from his own viewpoint, in which case the system can translate it into terms meaningful to the one communicated with, or he can use the superset of concepts, and communicate directly in the others terms.

This particular feature is extremely important if the user is to be doing a combination of individual and group work.

4.2 Support for Co-Authoring

Support for co-authoring is largely in terms of support for parallel work and intelligent merging of two versions of the 'same' object. Authoring, generally, is supported by the data base being made up of many small chunks which can be arranged as parts of many different larger objects. This is also supportive of some forms of co-authoring. If for example, some instructions on using a communication package between two machines are updated, then all documents containing those instructions are automatically updated at the same time. This is very valuable in maintaining consistency of documents which are written in many different pieces, by different people and for different purposes.

In a computer system, whenever more than one person is working on the same object, be it a document or a program, there is a need to in some way ensure that they do not overwrite one another's changes. One common way to do this is to strictly control either the 'what' or the 'when' aspects of the situation, i.e. the pieces which each person may work on are strictly controlled, or alternatively, there is some mechanism to ensure that the timing of the work is sequential. In LINCKS however, in accordance with our aim of maximum flexibility,

we attempt to provide support for co-authoring without strict controls.

LINCKS does not place any locks on objects that a user is working with, but the system does know if two users are working on the same object. The users involved are notified so that they can make a judgement about whether such parallel work is likely to take more effort than is worthwhile to later resolve.

The representation of objects as a structure of relatively small pieces facilitates the merging of two versions of an object. It makes it easier to check whether changes to an object (such as a document) occur in unrelated areas, and allow for automatic merging. Because the LINCKS system always keeps all old versions, it is quite simple for users to disqualify a particular automatic merge and return to the two parallel versions in order to merge them in some other way.

4.3 Support for Admimistration

To make co-operative work viable, some administrative overhead is needed. LINCKS provides a number of services to minimise the users time, effort, and involvement in administrative tasks.

Some different types of administrative tasks and services to support them are discussed. These include the various aspects of arranging meetings, meeting deadlines, and planning of tasks.

4.4 Support for graceful evolution

Group projects tend to evolve by what can be described as 'programming through progressive enrichment'[SAND79]. This enrichment can have a disruptive effect on the overall development, particularly if a number of people are working together. LINCKS limits such disruptive effects by using the history mechanism to ensure that if a new method or version of something does not apply to some data which predated that version, then the older version is used.

Some examples are given of where this can prove useful.

4.5 Support for geographical flexibility

The inclusion of portables within the system increases geographical flexibility. This is an advantage in both individual and co-operative work. It is however particularly pertinent to co-operative work in those situations where a person travels a lot. Such a person may have difficulty engaging in any co-operative projects, because so much effort would be required to integrate work that he did while away with work that had happened in the meantime. LINCKS facilitates this integration both by its interface with the portable, and by its intelligent communication services. The interface between the portable and the main system incorporates tools which automatically integrate the portables work contents into the main data base. The communication services are aware of the person's absence, and on his return, notify him of things related to his work which have happened while he was away. This minimizes the need either for the person on his return to find out from the group things that have happened, or for the group to be constantly aware of his absence, and send him mail.

5. Conclusion

In designing the LINCKS system we have paid much attention to ways of providing services, without adding constraints. We have tried to realise the notion of the computer system as a co-operating partner, rather than either a dumb tool or a controller. With respect to the question word parameters 'who', 'what', 'where', 'why', 'when' and 'how', we have tried to enable the system to have knowledge about them, to be able to reason about them, but not to constrain their values in any way, i.e. to allow each parameter to take on as many different values as possible in different situations. With respect to co-operative work this results in the possibility for many different flavors of co-operation.

The flexible typing model facilitates many different versions of *how* things are organised. The inclusion of portables gives great flexibility regarding *where*. The version structure combined with the networked molecular data base and the services for merging objects, provide flexibility in terms of *who, what* and *when*. Flexibility of *why* is supported both by the general orientation of the system, and by the arrangement of the data base, which allows the structuring and restructuring of objects regardless of the intentions with which they were originally created. The specification of operations as constrained only by execution limitations rather than by presumed meaning, also frees users from the need to have the same view of usefulness as the original writer of the operations.

Working co-operatively has some inherent problems, whose solutions necessitate some constraints. However, we consider it important to not add unnecessary rigidity by specifying how the problems must be solved. Rather, we attempt to provide tools and services which support solutions of the problems in a wide variety of ways. It is also important, particularly in a research setting, to ensure that co-operative work is compatible with individual work, without limiting it.

6. *Bibliography*

1. Aiello, L., Nardi, D. and Panti, M. *Structural Office Modelling: A first step towards the office expert system.* Proceedings of 2nd ACM Conference on Office Information Systems, June 25-27 1984, Toronto, Canada, p. 161-169

2. Barbi, E., Calvo, F., Perale, C., Sirovich, F., Turini, F. *A Conceptual approach to Document Retrieval.* 1984 ACM

3. Biggerstaff, Ted J., Endres, D. Mack, Forman, Ira R. *Table: Object Oriented Editing of Complex Structures.* 1984 IEEE 334-345

4. Bracchi, Giampio Pernici, Barbara *The Design Requirements of Office Systems.* ACM Transactions on Office Information Systems, Vol 2, No 2, April 1984, p. 151-170.

5. Croft, Bruce W. Lefkowitz, Lawrence S. *Task Support in an Office System.* ACM Transactions on Office Information Systems, 1984 Vol. 2, Issue 3, p. 197-212.

6. Goldberg, A., Kay, A. *SMALLTALK-80: The Language and Its Implementation,* Addison-Wesley, 1983.

7. Kimura, Gary D. Shaw, Alan C. *The Structure of Abstract Document Types.* Second ACM Conference on Office Information Systems, June 25-27 1984, Toronto, Canada, p. 161-169

8. Lin, Wente K., Ries, Daniel R., Blaustein, Barbara T. Chilenskas, R. Mark *Office Procedures as a Distributed Database Application.* Data Base, Winter 1984, 5-10

9. Lochovsky, F.H. A Knowledge Based Approach to Supporting Office Work. Database Engineering, IEEE Computer Society, Vol. 16, No. 3, Sept. 1983, p. 43-51

10. Sandewall, E. *Self-description and reproduction in distributed programming systems.* Informatics Laboratory, Linköping University, report number LITH-MAT-R-79-22, August 1979.

11. Tichy, W.F. *Design, Implementation and Evaluation of a Revision Control System,* Proceedings of the 6th International Conference on Software Engineering, 1982

12. Ullman, J.D. *Principles of Database Systems,* Computer Science Press Inc., USA, 1982.

13. Zdonik, Stanley B. *Object Management System Concepts.* SIGOA Newsletter, Second ACM-SIGOA Conference on Office Information Systems, vol. 5, no. 1-2, p.13-19.

14. Zisman, M. *Use of Production Systems for Modelling Asynchronous, Concurrent Processes.* In Pattern Directed Inference Systems, Waterman and Hayes-Roth, Eds., Academic Press, New York, 1978, 53-68

Avdelning, institution, fakultet
Division, department, faculty

Department of Computer and
Information Science
Linköping University

ISBN:

ISSN: ISSN-0281-4250

Rapportnr:
Report no: LiTH-IDA-R-88-19

Upplagans storlek:
Number of copies:

Datum:
Date: August 1988

Projekt:
Project:

Titel:
Title: From a Technical to a Humane Environment:
A Software System Supporting Cooperative Work

Författare:
Author Lin Padgham and Ralph Rönnquist

Uppdragsgivare:
Commissioned by:

Dnr.:
Call no:

Rapporttyp:
Kind of report:

- ☐ Examensarbete/Final project
- ☐ Delrapport/Progress report
- ☐ Reserapport/Travel report
- ☐ Slutrapport/Final report
- ☒ Övrig rapport/Other kind of report

Rapportspråk:
Language:

- ☐ Svenska/Swedish
- ☒ Engelska/English
- ☐ _____

Sammanfattning (högst 150 ord):
Abstract (150 words):

This paper presents LINCKS, an enviroment which supports co-operative work. LINCKS is presently in the stage of beginning implementation at Linköping University.

Nyckelord (högst 8):
Keywords (8):

Bibliotekets anteckningar

A Selection of Previous Research Reports.

LiTH-IDA-R-88-12 **Erik Sandewall, Johan Hultman, Erik Tengvald:** Software Architecture and Programming Paradigms for Robotics Applications.

LiTH-IDA-R-88-11 **Mats Wirén:** An Incremental Chart Parser for PATR.

LiTH-IDA-R-88-10 **Mats Wirén:** A Control-Strategy-Independent Parser for PATR. Revised version of paper appearing in *Proc. of First Scandinavian Conf. on Artificial Intelligence*, Tromsö, Norway, March 9 - 11, 1988.

LiTH-IDA-R-88-09 **Peter Fritzson:** Incremental Symbol Processing.

LiTH-IDA-R-88-08 **Erik Sandewall:** Formal Semantics for Reasoning about Change with Ramified Causal Minimization.

LiTH-IDA-R-88-07 **Erik Sandewall et al:** IDA Annual Report 1987.

LiTH-IDA-R-88-06 **Christer Bäckström:** A Representation of Coordinated Actions Characterized by Interval Valued Conditions.

LiTH-IDA-R-88-05 **Christer Bäckström:** Keeping and Forcing: How to Represent Cooperating Actions.

LiTH-IDA-R-88-04 **Wlodek Drabent, Simin Nadjm-Tehrani, Jan Maluszynski:** Algorithmic Debugging with Assertions. Also in *Proc. of META 88 - Workshop on Meta Programming*, Bristol, June 21-24 1988.

LiTH-IDA-R-88-03 **Staffan Bonnier, Jan Maluszynski:** Towards a Clean Amalgamation of Logic Programs with External Procedures. Accepted to the *Joint Fifth Int. Logic Programming Conf. and Fifth Logic Programming Symp.*, Seattle Washington, August 15-19, 1988.

LiTH-IDA-R-88-02 **Ulf Nilsson:** Inferring Restricted AND-Parallelism in Logic Programs using Abstract Interpretation.

LiTH-IDA-R-88-01 **Jacek Leszczylowski, Staffan Bonnier, Jan Maluszynski:** Logic Programming with External Procedures: Introducing S-Unification. A shorter version is to appear in *Information Processing Letters*, North Holland.

LiTH-IDA-R-87-26 **Jonas Löwgren:** Applying a Rapid Prototyping System to Control Panel Dialgues.

LiTH-IDA-R-87-24 **Sven Moen:** Drawing Dynamic Trees.

LiTH-IDA-R-87-22 **Lars Ahrenberg, Arne Jönsson:** An Interactive System for Tagging Dialogues. Also in *Proc. of 14th Conference of the Association for Literary and Linguistic Computing*, Göteborg 1-5 June, 1987.

LiTH-IDA-R-87-21 **Harold W. Lawson, Jr.:** Challenges and Directions in Computers and Education.

LiTH-IDA-R-87-20 **Krzysztof Kuchcinski, Zebo Peng:** Parallelism Extraction from Sequential Programs for VLSI Applications. This paper is to appear in *Microprocessing and Microprogramming, the Euromicro Journal*, 1988.

LiTH-IDA-R-87-19 **Zebo Peng:** Construction of Asynchronous Concurrent Systems from their Behavioral Specifications. Also in *Proc. of the 10th World Computer Congress*, Dublin, Ireland, Sept. 1-5, 1986.

LiTH-IDA-R-87-18 **Henrik Nordin:** Reuse and Maintenance Techniques in Knowledge-Based Systems.

LiTH-IDA-R-87-17 **Tony Larsson:** Specification and Verification of VLSI Systems Actional Behaviour This is a close version of a paper presented at *the 8th international conference on Computer Hardware Description Languages, CHDL*, 87.

LiTH-IDA-R-87-16 **Shamsul I. Chowdhury:** State of the Art in Statistical Expert Systems. Also in *Proc. of the Conference on Expert Systems and their Applications*, Avignon, May 1987.

LiTH-IDA-R-87-15 **Nils Dahlbäck:** Kommunikation med datorer i naturligt språk - vad är det och vem behöver det?

The Department of Computer and Information Science
Linköping University

organizes undergraduate and graduate studies in *Computer Science*, *Telecommunication and Computer Systems*, and *Administrative Data Processing*. Research activities have an emphasis on advanced software technology and computer systems design and are organized in a number of research laboratories:

- *ACTLAB - Laboratory for Complexity of Algorithms*, which is concerned with the design and analysis of efficient sequential and parallel algorithms, and complexity theory, especially in the areas of computational geometry, data structures on bounded domains and graph algorithms.
- *ASLAB - Application Systems Laboratory,* which studies design of advanced support systems for interactive use of computers, including tools for automated construction of applications software.
- *CADLAB - Laboratory for Computer-Aided Design of Electronics*, which concentrates its research activities around tools for integrated development of hardware and software, graphics-based modelling and simulation techniques.
- *LIBLAB - Laboratory for Library and Information Science,* which studies methods for access to documents and the information contained in the documents, concentrating on catalogs and bibliographic representations, and on the human factors of library use.
- *LOGPRO - Laboratory for Logic Programming*, which concentrates its research activities around foundations of logic programming, relations to other programming paradigms and methodology.
- *NLPLAB - Natural Language Processing Laboratory*, which conducts research related to the development and use of natural language interfaces to computer software.
- *PELAB - Programming Environments Laboratory*, which works with design of tools for software development, specific functions in such tools and theoretical aspects of programs under construction.
- *RKLLAB - Laboratory for Representation of Knowledge in Logic*, which covers issues and techniques such as non-monotonic logic, probabilistic logic, modal logic and truth maintenance algorithms and systems.

Research Reports 1988

LiTH-IDA-R-88-19 **Lin Padgham, Ralph Rönnquist:** From a Technical to a Humane Environment: A Software System Supporting Cooperative Work. Also in *Proc. of the GDI Internationa Conference on USER INTERFACES,* Rüschlikon, Switzerland, October 20-21, 1986.

LiTH-IDA-R-88-18 **Nils Dahlbäck:** Mental Models and Text Understanding - a Commented Review.

LiTH-IDA-R-88-17 **Bengt Lennartsson:** Multi-Level Software Architectures - Industrial Experiences and Research Directions. This report is a revised version of a paper in *Advance Working Papers of The Second International Workshop on Computer-Aided Software Engineering.* Cambridge, Massachusetts. July 12-15, 1988.

LiTH-IDA-R-88-16 **Lin Padgham:** A Model and Representation for Type Information and Its Use in Reasoning with Defaults. Also in *Proc. of AAAI'88, American Association for Artificial Intelligence*, 1988.

LiTH-IDA-R-88-15 **Lin Padgham, Ralph Rönnquist:** LINCKS: An Imperative Object [illegible] *Proc. of the 20th Annual Hawaii Int. Conf.*

LiTH-IDA-R-88-13 [illegible] gssystem för autonoma