Schriftenreihe

Österreichische
Computer
Gesellschaft

G. Chroust, A. Benczúr (ed.)

CONnectivity
C J
G S 94
Z
T

Workflow Management

Challenges, Paradigms and Products

R.Oldenbourg Wien München

Workflow Management -
Challenges, Paradigms and Products

CON'94

SCHRIFTENREIHE DER
ÖSTERREICHISCHEN COMPUTER GESELLSCHAFT
BAND 76

Gedruckt mit Förderung des Bundesministeriums
für Wissenschaft und Forschung in Wien

G. Chroust, A. Benczúr (ed.)

Workflow Management -
Challenges, Paradigms and Products

CON'94

R.Oldenbourg Wien München 1994

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Workflow Management : challenges, paradigms and products / CON '94. G. Chroust ; A. Benczúr (ed.). - Wien ; München: Oldenbourg, 1994
(Schriftenreihe der Österreichischen Computer Gesellschaft ; Bd. 76)
ISBN 3-486-23159-6 (Oldenbourg, München) brosch.
ISBN 3-7029-0397-6 (Oldenbourg, Wien) brosch.
ISBN 3-85403-076-2 (Österr. Computer Ges.) brosch.
NE: Chroust, Gerhard [Hrsg.]; CON <9, 1994, Linz>; Österreichische Computer Gesellschaft: Schriftenreihe der Österreichischen ...

Druck: Druckerei Riegelnik
1080 Wien, Piaristengasse 19

ISBN 3-486-23159-6 Oldenbourg München
ISBN 3-7029-0397-6 Oldenbourg Wien
ISBN 3-85403-076-2 Österreichische Computer Gesellschaft

Welcome to the University of Linz

It is a pleasure for me to send my best regards to the participants of the "Ninth Austrian - Informatics conference CON 94".

One of the most promising themes which came up with computer supported cooperative work was the new definition of workflows and their implementation using modern computer technology.

This workshop again shows the high competence of the members of our technical faculty of the University of Linz processing most interesting practical themes. With other words this conference can be seen as a part of the knowledge transfer from science to firms.

Though the program ist very intensive, you should take your time to visit some of the beautiful sights in Upper Austria. Reading the program I found out that you already go to Steyr, one of the oldest and most charming towns of Austria. Hagenberg, where the conference takes place, is a typical nice small village near Linz.

It is a honour for me to welcome guests from abroad and I hope the conference will be worth coming to Austria.

With best wishes

O.Univ.-Prof.Dr. Johannes Hengstschläger
Rektor

PREFACE

Very early in their history humans realized that only cooperative work ensured their survival, be it by hunting a bear, be it by irrigating arid deserts, be it by building bureaucracies. Cooperation and sharing of work is one of the foundations of civilization.

Cooperation needs communication and this again must be based on connecting people to one another. Recognizing that connecting people is one of our keys to the future, the series of joint Hungarian-Austrian conferences have been put under the common motto of 'Connectivity': In 1993 the main emphasis was on the technology of networking, this year's conference will be oriented towards connecting office workers via work flow.

Work Flow Management, *the computer assisted management of business processes through the execution of software whose order of execution is controlled by a computerized representation of the business process* is today considered one of the key technologies for providing efficiency and effectiveness in the office.

The basic idea - defining formally the flow of work in the administration and then use a computer environment to effectively support this flow - is based on a paradigm applied in many other industries (software engineering, manufacturing etc.). Application of this technology in the real world has to consider many facets: technological (formal description methods, interpretation technology and data base support, actual commercial products), organisational (business reengineering) and sociological (acceptance and interfacing).

We are proud to present in this conference a program containing 30 papers, originating from 5 countries, with a majority from Austria and Hungary. They cover all of above issues:

general introduction
- Workflow Concepts
- Limits of Modelling
- Workflow - Past and Present

technology
- Workflow Products
- Modelling Workflow
- Database-support for Workflow

organisation
- Modelling Business Processes
- Applied Workflow Management

sociological
- Cooperation in Europe
- Interfacing with the User

The conference, again, is augmented by two full-day tutorials, directly related to the conference theme:

Active Database Systems by K.R. Dittrich, Switzerland and
Workflow by C. Ellis, USA

We also take the pleasure to thank all those who have contributed to the realization of the conference:

- the submittors of papers and the speakers
- the members of the Programme Committee, especially Prof. J. Demetrovics,
- our tutorial chairperson, G.Kappel,
- the Secretary Generals of the two organising societies, Ms. M. Toth, Mr. W. Grafendorfer,
- the members of the two secretariats, especially Mr. W. Hawlik, Ms. I. Sudra, Ms. I. Jensen and Ms. G. Aranyas,
- the sponsors of this conference,
- the many helping hands like Ms. G. Kotsis, Mr. P. Grünbacher and
- the Kepler University of Linz for logistic support.

We wish the *Ninth Austrian-Hungarian Informatics Conference, CON'94: Workflow Management* a climate of cooperation and active interchange of ideas. We are sure that it will continue the good tradition of past conferences and show an avenue into the future.

September 1994

Gerhard Chroust
Andras Benczur

SPONSORS

Austrian Federal Ministery of Science and Research
Kepler University Linz
Magistrate of the Municipality of Linz
Verein Museum Arbeitswelt, Steyr
Forschungs- und Ausbildungszentrum für Arbeit und Technik, Steyr

PROGRAMME COMMITTEE

Matyas Arato (H)
Andras Benczur (H) *Vice-chair*
Gerhard Chroust (A) *Chair*
Janos Demetrovics (H) *Vice-chair*
Laszlo Hannak (H)
Lutz Heinrich (A)
Gerti Kappel (A) *Tutorial Chair*
Gabor Klimko (H)
Laszlo Langer (H)
Roland Lutz (A)
Hermann Maurer (A)
Roland Mittermeir (A)
Miklos Papp (H)
Gerald A. Pitschek (A)
Tibor Remzsö (H)
A Min Tjoa (A)

ORGANISING COMMITTEE

G. Aranyos (H)
Walter Grafendorfer (A) *Chair*
Irene Jansen (A)
Angela Kohl (A)
Irene Sudra (A)
Maria Toth (H) *Vice-chair*

Contents

Workflow Concepts

Computer Supported Cooperative Work: State and Perspectives

Roland Traunmüller

The term Computer Supported Cooperative Work (CSCW) was brought forth one decade ago. Since then the seedling has sprouted, blossomed and ripened. Hence he first part deals with the emerging of CSCW, its driving forces and the various aspects of CSCW (sections 1-3). Then a taxonomy is presented and some pioneer systems are sketched (sections 4-5). Subsequently the state of the art of CSCW is discussed and illustrated with reference to commercial systems (sections 6-9). Finally three particular perspectives are considered: group enabling of systems with CSCW mechanisms; models of interactions and the need for interdisciplinary approaches to design (sections 10-12).

1. The Emerging of CSCW

1.1 Coining the Term CSCW

The term "Computer Support for Cooperative Work" was coined by Irene Greif and Paul Cashman in 1984, as a prelude to the first CSCW conference held in Austin, Texas, two years later. CSCW was thought of as arising from a particular kind of problem - the need that most people had to cooperate in groups when doing their work - and so this gave rise to the need for particular kinds of software systems, to which the term "groupware" came to be applied.

1.2 Distinguishing Feature of CSCW

Of course systems have always been designed to serve many people. The distinctive feature being identified was that of people cooperating on the same, or related, task by interacting with each other through the machine. Rather than using timesharing precisely for the purpose of sustaining the illusion that users had their own virtual machine entirely to themselves,

systems were now conceived directly to support users in their inter-relations. In the same direction goes the trend towards style usage. Again, it is not claimed that no previous systems ever did this, rather than explicit recognition of the needs of the users of such systems would now enter into the design philosophy.

1.3 CSCW as a New Paradigm

The term "Computer Supported Cooperative Work" was coined in 1984 by Greif and Cashman in preparing the first conference about this topic. It was realised that CSCW arises from a particular kind of problems namely from the needs that some task can only by achieved by some people cooperating as a group. Cooperation means working together as a group, understanding the subtasks of other members, and sharing the data. So the first systems were called groupware in order to show that this kind of work was only accomplished by interacting with each others through the computer. This was quite the opposite to timesharing systems were everybody had the illusion to have his own virtual machine. More, such systems were conceived directly for the use of interaction between users.

1.4 CSCW as New Application Systems

In addition CSCW has brought about systems which have created new application scenarios:

- electronic mail for storing and forwarding messages
- shared workspaces for providing common views on a particular subject
- group authoring enabling cooperative writing with additions, revisions, comments, and annotations
- group decision support systems for argumentation, negotiation, and decision making
- conferencing with wihiteboardsystems, bulletin boards, and videoconferences

1.5 CSCW as an Enabling Technology

Business Process Re-Engineering (BPR) is the motto of new attitude toward the organisation. BPR is an integrated perspective and programme based on organisational change: core processes have to be found and redefined with respect to effectiveness and efficiency. BPR is a hot topic and renowned conferences such as the TC8 '94 Conference [11] are devoted to it.

1.6 CSCW on The Verge of Maturity

Now CSCW has reached the commercial enterprises. As an example the recent edition of an acknowledged computing journal with world-wide circulation [9] reviews almost hundred commercial products for CSCW. In a similar way the quantity of literature on CSCW has

increased over the past years. Hence in this contribution literature is cited only paradigmatically: [1,2,3,5,6] represent main recent CSCW-conferences; [4,7] overview basic concepts of design; [9,10] present and evaluate commercial products; and [8,9,11] discuss applications in office and administration. It should be stressed that this selection is a subjective choice and mainly intended for providing a basis for further investigation.

2. The Driving Forces Behind the Interest in CSCW

2.1 Problems with Existing Systems

First of all, it is the normal and plain case that for achieving a task people have got to work together. Additional driving forces for the spreading interest in CSCW are connected to deficiencies of existing systems:

1. dissatisfaction of the users with existing systems
2. increasing expectations of the users
3. "easy" things have already been done
4. concentration on highly formal, bureaucratic procedures.

Increasingly more powerful, high functional, complex, and user-friendly systems continue to be produced for a wide range of domains. The hardware platforms, digital communication infrastructure, and interface techniques continue to grow in power by the day, while their costs decline virtually. Yet users seem increasingly dissatisfied. Below the surface of the spectacular development there is a steady growth of discontent: Systems which were designed to help and support people in their work, can impede people's working practices or even completely fail.

Some of the frustrations may be linked to exaggerated ambitions in the past. Most of the "easy" things have already been done, and the work settings which are most amenable to a closely-defined procedural approach are already into several generations of systems - those include such applications as payroll, accounting, order processing, and other prevalent office procedures.

All in all, data processing focused on a wide range of applications that are constrained - on administrative, legal and organisational reasons - by highly formal, bureaucratic procedures and so predisposed to "programmed" solutions.

Some discontent may be linked to increased expectations. Whereas once the majority of end-users were either sceptical of computer-based systems, terrified by them, or over-awed by

them they are now much more sophisticated in their approach with the result that they aremuch more willing to be critical. As people come under increasing pressure to deliver in a competitive and insecure environment, they expect to be able to turn to the technology to shoulder part of that burden.

2.2 Support for Cooperation in a Wide Sense

In this context CSCW is gaining prominence, as holding out the prospect of a different approach to systems design which might address at least some part of these problems. So the challenge is to move out from this base into the effective integration of separate systems, and into the support of the "higher-level" organisational processes involving decision-making, negotiation and collaboration - areas characterised by flexibility and rapid change rather than constancy.

A closer inspection of cooperative work situations in business enterprises reveals that coordination is not "the only game". Coordinated work represents only one kind of the three major forms of cooperative work. There is also collaboration and group decisions.

There are further features to be considered in cooperative work: Cooperative ensembles may vary over time. They are marked by an ample local distribution and lack of central control. Likewise work might be as well synchronous as asynchronous.

2.3 The Quest for Productivity

A further problem is the inquiry for productivity. Awkward questions are being asked about the productivity gains which are being delivered. It is difficult to show convincingly that computer-based systems developed in the last couple of decades have been cost-effective. Further, the user will pose increasingly critical questions about whether they are getting the right kind of systems.

In this context CSCW may be seen as part of the recent quest for productivity. Using CSCW and organisational re-engineering might be regarded as two sides of the same coin.

3. Various Aspects of CSCW

3.1 CSCW as a Distinctive Domain

There are some of the 'classic' examples of CSCW systems. Despite this diversity, however, the predominant view among computer scientists (and among social scientists too) is that cooperative work is a distinctive domain compared to work 'in general', and that "groupware" for CSCW (even on an expanded definition) is a specialised class of system within work support systems in general.

However we prefer another perspective which sees CSCW not as a specialised kind of system, but rather as symptomatic of a shift in the way that the system design in general is being conceived. When system design concerns itself with systems which are to be implemented and used in a particular domain, it inevitably acquires a double aspect:

- On the one hand, it has an internal dynamic of its own theories, concepts, terms and techniques.
- On the other hand, it has to engage and seek to understand the world of the particular domain in which it is to operate and which it is designed to support.

3.2 CSCW as Teamwork

One way to understand CSCW is as computer support for "teamwork". Some notes of caution are required. Work which is apparently done on an individual basis nevertheless still has an aspect of social organisation to it.

CSCW also has to deal with intensive collaboration within small, stable, and well-structured groups whose members all know each other as well as with large, unstable setting. Many cooperative ensembles are large, or are themselves embedded within larger ensembles. Often ensembles are transient formations which emerge to handle a particular situation and then dissolve.

The term "teamwork" should not evoke a cosy setting of harmony, concord and mutual support, with everyone pulling together towards a common goal. Organisations are arenas in which four kinds of "game" can and are be played: success of the organisation itself; ones own career, self-advancement and material reward; internal politics of the organisation; and that concerned with the personal connections.

3.3 Supporting Various Forms of Cooperative Work

Although the examples given above are manifold and divergent there is a common denominator namely the emphasis on cooperation within a group. On closer inspection of the examples three major forms of cooperative work can be distinguished:

1. Coordination unifies different activities for the accomplishment of a common goal. Each activity is in an intrinsic relation to preceding and succeeding ones so making synchronisation a major issue.
2. Collaboration is the case of persons working together without external coordination as it is the case in co-editing and shared drawing. It is necessary to have a common information space and to point at a collective goal.
3. Group decisions need cooperation for the accomplishment of a collective decision. Although diverse opinions and interests may prevail a minimum of mutual trust is required.

3.4 CSCW: Enabling Technology and Organisational Issue

In resuming four decades of information technology and organisations Bjoern-Andersen states that this is a paradigmatic shift in the relation between information technology and organisations. Now, in the Nineties, organisational issues have become the driving force. So he cites organisational re-engineering, group support and inter organisational systems. The paradigmatic shift is mirrored in terminology as well: The supportive role of information technology is perfectly described with the term of "enabling technologies". In that way CSCW may be regarded as an enabling technology for the organisational issue of BPR.

Although different BPR approaches may vary in this terms, they all include assessment and re-engineering as two essential stages. Assessment investigates different business activities, identifies the core processes and selects candidates for change. Re-Engineering gives a vision of future processes and seeks enabling factors. Such enablers may be technical or organisational in character. Subsequently a prototype of the process is specified in detail.

3.5 CSCW for "Multi-site / Multi-time / Multi-culture Workgroups"

Office Systems of the future will have to operate under circumstances that today may not be seen as common work environment: "multi-site / multi-time / multi-culture workgroups". With regard to this requirement present Office Systems show big deficiencies that should be improved by adding specific CSCW functions. Encouraging results from several CSCW pilot projects indicate better ways of interaction under such non-trivial work circumstances. Some projects should be mentioned as examples:

- One example are "intelligent" information sharing systems such as "Information Lens", "Object Lens" and "OVAL". They are intended to have people who can manage their own email.
- "TeamWorkStation" pioneers a multi-culture setting for learning and discussion. The pilot version integrates two workplaces and also connects them with four video cameras directed at each workplace and each person at work. So an ideal setting for distant learning is achieved.
- Several projects show the value of desktop conferencing and computer conferencing in scenarios marked by dislocated and asynchronous cooperation. They will be a helpful supplement for multi-time and multi-site offices.

Already these few examples may demonstrate the fact that CSCW functions are a necessary prerequisite for setting the stage for the "multi-site / muliti-time / multi-culture workgroups"

4. Taxonomy of CSCW

4.1 Perspectives for Discussing CSCW Systems

CSCW may be categorised according to perspectives. One basic distinction - namely application scenarios - has already been covered in the first section. Another undisputed categorisation is place and time. In addition, categorisations with respect to various distinctions can be found in the literature:

- The degree of communality is another factor may vary. Communality may be a common goal or use of common resources.
- Further, there might be distinct ways of support such as overcoming obstacles, decreasing uncertainty, or providing active guidance.
- The restrictivity of handling will be another choice ranging from rigidity to flexibility.
- Another form of taxonomy refers to media and structure such distinguishing to text documents, hypertext, multimedia, and hypermedia.

4.2 Categorising Place and Time: The Technological Square of CSCW

A schema regarding different categories of place and time results in a 3x3 technological square as given in the subsequent schema.

	Same Time	*Different Time*	
		Predictable	*Unpredictable*
Same Place	Meeting Facilitation	(Work shifts)	(Team Rooms)
Different Place (Predictable)	Whiteboarding, Desktop-, Video-Conferencing	Electronic-, Voice-Mail	Collaborative Writing
Different Place (Unpredictable)	(Broadcast Seminars)	Computer Conferences	Workflow Management

5. Pioneer Systems in CSCW

5.1 Electronic Mail

To some people electronic mail is seen as the clearest example of a groupware application that has made a significant impact in the work place. Even within a particular department, where people could communicate face-to-face as well, email is used as an alternative because it offers additional possibilities. The ability to send messages electronically to people via a local area network or a wide area network has resulted into a new sense of connectivity. There is a rapid dissemination as well as a widespread employment in business settings aligned with high acceptance across the board.

5.2 The Coordinator System

"The Coordinator System" was one of the first commercially used system, and is also one of the most talked about CSCW applications. This is due to its commercial success as well as to its well-developed theory (Speech-Act Theory). With the assumption of "language as action" it has exerted considerable influence in the research community. The actual system (Winograd, 1986) can be described as a combination of electronic mail with a semi-automatic project management system.

The basic idea is that human communication is based as language for action. Hence people do not simply send mail but they make requests, make promises and offer or decline commitments. So the system has to keep track of all commitments made by the participants. In principle the system also allows free-form responses, but this is a withdrawal from its underlying principal.

5.3 Groupware Systems

"Lotus Notes" is a commercial product which has generated an enormous amount of interest since it was announced first. It can be described as a platform supporting communication and information sharing in an enterprise.

"The Conversation Builder" developed at the Univ. of Illinois, Champlain (Kaplan) is an exmple for a very advanced system that can be labelled general generator system for designated applications.

5.4 Meeting Rooms and Conference Facilities

"Xerox PARC Colab" was a project at Xerox PARC (Stefik et al., 1987) involved building a computerised meeting environment to support small face-to-face meetings. A special room was constructed containing several workstations connected to a local area network. A number of software tools were developed to allow users to jointly work on documents and to share the same views of these documents. This was done according to the motto WYSIWIS - what you see is what I see.

"MMConf" is a system sustaining computer conferences. The discussants have to take the "floor" in order to achieve an active role.

5.5 Intelligent Information Sharing System

The MIT projects "Information Lens", "Object Lens" and "OVAL" belong to a set of much applauded pioneers. The Information Lens system (Malone et al., 1987) is intended to have people managed their email and can be baptised as an "intelligent" information sharing system. The filtering is aimed to screen people from junk mail and the filter only allows in messages of interest, even when they are not directly addressed to specific users. It provides capabilities for organising mail based on various aspects of incoming messages.

Subsequently the Information Lens system has been extended beyond message handling to the Object Lens (Lai, Malone, Yu, 1988) and is now evolving into OVAL (Objects, Views, Agents, and Links) a large set of cooperative work applications (Malone, Lai, Fry, 1992)

"Khronika" is a similar system developed at Xerox EuroPARC in Cambridge, UK and also works at the basis of interest profiles. A "demon", as an intelligent agent, dispatches the collected information to the individual user within a working group.

5.6 Collaborative Writing (Group Authoring)

"ForComment" is a commercial product which is a group authoring tool. The text is imported into the system and then comments and annotations can be added. Despite its basic simplicity the system proved itself commercially successful. Some annoyances were reported due to the fact that texts must be imported from separate individual word-processing systems.

"GROVE" was developed at the MCC for brainstorming and collective editing of reports. "DistEdit" is another group authoring tool for larger groups working in a master/slave status.

5.7 Group Decision Support

"gIBIS" is an application specific hypertext system developed at the MCC in Austin, Texas. Originally it was used for the software development process in structuring and documenting all relevant decision steps.

"Sibyl", developed at the MIT, supports decisions with a particular representation language. Goals, issues, and arguments are collected, structured and documented.

"Ventana GroupSystem" is a spin-off of a group decision support system developed by Nunamaker at the University of Arizona. They support various functions such as brainstorming, ranking choices and facilitate a mechanism to vote on them while still preserving anonymity.

5.8 Multimedia Systems

"TeamWorkStation" (H. Ishii) integrates two workplaces and connects them also with four video cameras directed at each workplace and each person at work. So ideal setting for distant learning is achieved, and that rendered itself helpful even for different task such as acquiring the skill of Japanese hand printing.

"Slate" developed from BBN and "Sepia" developed at GMD are multimedia environments for cooperating in drawing or authoring.

6. Workflow Management Systems

6.1 Workflow Automation

A Workflow Management System is a system which provides procedural automation of a business process by management of the sequence of work activities and the invocation of appropriate human and/or information technology resources associated with the various activity steps (Definition according to Workflow Management Coalition).

Marketing always has need for catchy terms, therefore Workflow Management Systems are advertised as revolution in the office world. In more humble terms Workflow Management Systems would mean a shift in paradigms. Instead of regarding isolated departments, and their systems, as distinct (ordering, accounting, etc.) the entire business activity is looked upon. So in workflow automation the existing LAN infrastructure is used to specify and coordinate business processes that move documents within workgroups. In this regard such systems are often compared with assembly lines for manufacturing.

6.2 Workflow Management and Groupware

There are slight differences in viewing the distinction between Workflow Management and Groupware:

1. Workflow Management and Groupware may be seen as two parts of CSCW: one is characterised by structure and routine whereas the other is project-oriented and information-centred.
2. Others prefer a management view of Workflow Management, and then superpose it over Groupware. Workflow Management gives a process-centric direction to the whole whereas Groupware "per se" is seen as information-centric. In this interpretation Workflow Management is interpreted as process-oriented management activity extending "information management" into the direction of "process management".

The second approach puts Workflow Management Systems in the forefront of Business Process Re-Engineering. Another reason might be a difference in cultural background. In the United States, where the second view is preferred, the appreciation and attitude towards administration is different to that held in Europe.

6.3 Commercial Systems

There is a fast growing number of systems on the market: ActionWorkflow Manager, Archive Lite, Connect, IBM IMagePLus/2, Keyfile, LinkWorks, Lotus Notes, Microsoft Electronic

Forms Designer, WordPerfect in Forms, etc. The capability of most systems comprises such features as: systems installation, business analysis and modelling, workflow creation, and performing different sub tasks such as routing, exception handling, tracking, and reporting.

The systems are distinguished by distinct suitability to different tasks. Low-end-systems allow only very restricted definition of processes whereas their high-end counterparts provide a sophisticated functionality in various directions; e.g. routing, scalability, analysis, modelling and reporting capabilities. Notably support for analysis and modelling of organisations and processes will render high-end systems very attractive.

6.4 Workflow Management Coalition

In the middle of 1993 leading enterprises of the domain founded the Workflow Management Coalition. The goal of this organisation is the definition of vendor independent interfaces for the cooperation of different systems. Such interfaces should enable the specification and implementation of such features as:

- APIs (Application Programmers Interfaces) for interoperability between systems of different vendors;
- exchange of process models at run time;
- integration if functions for individual data processing;
- connection of analysing and reporting instruments.

7. Conferencing Systems

7.1 Email Systems

Email is the foundation of workgroup computing. No wonder then that these systems have achieved a high degree of maturity: CC:Mail from Lotus, Microsoft Mail, and BeyondMail get high score. BeyondMail is esteemed high for its perfected rule-based information handling and filtering.

7.2 Bulletin Board Systems

First Conferencing Systems or Bulletin Boards Systems (BBS) appeared soon after the emerging of Email in the late Seventies. They were designed for dial in use so rendering them the valuable support of remote users. They are often part of an office package, e.g. CC:Mail and WordPerfect Office.

Simple Email based BBS are only one side of the spectrum. The other side are intricate and elegant windows based systems including administrative functions, conferencing and application integration. Most famous in this class is Lotus Notes which has become an archetype system.

7.3 Information Filtering

The filtering is aimed to screen people from junk mail and filter in messages of interest, even when they are not directly addressed to specific users. It provides capabilities for organising mail based on various aspects of incoming messages. Working with semi-structured message templates characteristics can be established that would match corresponding interest profiles. Such categories might comprise organisation, task, urgency, etc. A potential receiver may pose adequate rules for prioritising incoming messages before reading and sort them into folders after reading them. In practice it was often complicated to state rules a priori. So the system has evolved throughout use.

8. Meeting Support Systems

8.1 Forms of Support for Conferences and Meetings

First it seems necessary, to make some remarks on forms of support needed for conferences and meetings and their technical terms:

1. An early approach was given with Conferencing (or Bulletin Board) Systems. They aim at a continuous group conversation and mean unstructured information exchange within a group without any request on synchronicity.
2. Another early form of support centres on the scheduling issue. Although scheduling might be considered only as a preliminary stage or a problem of pure formal character, much attention has been paid to Group Scheduling Systems and Calendaring Systems.
3. Meeting Support Systems are directed at the central problem of the meeting and attempt to support the process itself. They sustain various sub tasks: synchronous communication, setting of agendas, structuring of problems, evaluation of solutions, and facilitating of the discussion. Hence the particular venue of the meeting and its technical infrastructure are under consideration as well.
4. Whiteboard Software aims at viewing the same documents and discussing them. This is performed simultaneously way. Such systems allow changing groups with attendees joining and leaving at their own discretion.
5. Desktop Videoconferencing is still in its infancy. If the capability of video is added, a higher level of technical communication infrastructure is needed.

8.2 Meeting Rooms

The diversity of meeting packages is high with GroupSystem V from Ventana Corp. at the high end and VisionQuest for DOS at the low end.

GroupSystem V developed by Nunamaker has already merited its inclusion as a pilot project (cf. section 5). It has one of the finest selection of group process tools and can be adjusted to various types of meetings. The range of its "idea" processing tools include idea generation, idea consolidation, alternative evaluation, voting, and reporting. Other support tools are concerned with meeting creation, agenda setting, and process facilitation. VisionQuest on the other hand is a low end product with an excellent usability/price ratio.

8.3 Shared Calendaring and Meeting Scheduling Systems

A lot of systems attempt to act as an intelligent substitute for paper diaries. In general it would seem difficult to replace paper diaries. The reason is that physical diaries are unsurpassed in several key terms: portability, flexibility, and usage. Main advantages of paper diaries are the following: physical size, ubiquitous employment, ease of use, speed of handling, and diversity of annotations (clippings, post-it notes, inserts, different colours).

There are now a lot of commercial products available that aim at replacing the paper diary. The major point for their use is not the individual need but the necessity to have scheduling capability in work group computing. Recent products are CaLANdar from Microsystems and Time an Place/2 from IBM.

9. Whiteboard Software and Videoconferencing

9.1 Whiteboard Software

Whiteboard software documents can be transferred and subsequently discussed, commented on and altered by other participants. In that way adequate information sharing with close collaboration is set up. The demands on communication techniques are modest because of the restriction to have only still images. All in all, whiteboard software seems to be both an opportune and a trade-off of the needs of CSCW given the conditions of communication lines which are available.

It is no wonder that the field is prospering. Commercial products vary in suitability: On the high end of the scale is Person to Person for Windows whereas Intel Proshare, a system restricted to a two person communication, is on the low end.

9.2 Videoconferences

Most systems exchange still images allowing only static document sharing and via copying a still image of the counterpart, whereas more sophisticated systems aim at application sharing. Some systems work on normal phone lines but with possible major consequences on quality. An example for such limited system is InVision.

If there are moving pictures being transmitted in videoconferences an appropriate transmission is required. This means service based on digital exchange, e.g. ISDN, as a minimum requirement. An example for a high level product is Telemedia Personal Videosystems form AT&T. Despite bright prospects for the future current usage of Desktop Videoconferencing is limited. Also the look and feel of the systems in use is not very convincing.

10. Group Enabling of Systems with CSCW Mechanisms

10.1 Shared Objects

Sharing objects is a feature essential for CSCW. In principle a fluid transition between individual and a cooperative work situation should be enabled. So, depending on the situation, different forms of sharing objects should be supported:

1. Enabling a real exchange of objects so that different user may run their 'single user' applications concurrently.
2. Sustaining the sharing of objects by means of a particular data management system.
3. Making provisions for the sharing of different views on a common object.

10.2 Particular Ways of Integration

A particular form of integration has been developed for the application domain of office work. "Office suites" are intended to support the fluent integration of cooperative and individual activities by virtue of their being an integrated software package. Such packages with Lotus Notes and Microsoft Works as precursors provide integrated facilities for word processing, email, retrieval, spreadsheets, etc.

10.3 Domain Directories

In most CSCW applications a whole set of domain directories have to be supported:

1. User directories aim to provide the connection of system functions to particular user and their organisational position. Usually they include the following information: user names, access codes, group memberships, privileges etc.
2. Object directories ensure the distributed retrieval of information. For such a retrieval a minim of information is necessary: object names and relations to other objects, creator of the object, time of creation, and changes to the object. In addition a connection to domain specific classification schemes is valuable.

10.4 General Services Sustaining Coordination

General services sustaining coordination have to be provided. They include:

1. Sustaining a common sharing policy
2. Support for a common access policy
3. Aiding the floor control by means of turn taking protocols
4. Building in features for the support of unanticipated use.

10.5 Providing Informal Interaction

Dynamic and complex settings require a minimum of informal interaction. Mechanisms supporting the informal styles of interpersonal interaction are best provided in a conference setting. Besides such conference systems informal channels are realised in very few systems, with TeamWorkStations being a notable exception.

10.6 Indicating Context and Conceptual Framework of Information

1. In general office procedure, as well as in cooperative decision making, it is important to preserver the conceptual framework of information.
2. Often it is necessary to tag certain information with its originator. In that way responsibilities can be accounted and biases can be controverted.
3. In a similar way the perseverance of context of information created will be essential.
4. In addition it is necessary to stress the fact that no information will exist per se. Aim and purpose are intrinsic qualities of information contextual persistance is essential.

10.7 Adjusting Plasticity to CSCW-Mechanisms

1. Adaptability to preferences: Mechanisms should be adaptable to personal preferences as well as to the wishes of a particular cooperating ensemble.
2. Multidimensional Aspects: The various dimensions of articulation work (such as what, where, how, when) are to be managed simultaneously.
3. Semantic Conformity: The semantic level of notational primitives should correspond to the context.
4. Generic Sets of functions: In order to facilitate cooperative management, generic sets of interaction functions have to be built in between the layer of the basic operation system and the application level.

11. Models and Theories for Interaction

11.1 The Role of Models and Theories

A lot of assumptions have been stated, various models have been developed, and some theories have been formulated. Partly they are aimed at understanding the nature of the interaction process, and partly they are intended to control the interaction processes.

11.2 Planning Cooperation Based on Coordination Theory

Coordination theory according to Malone is a bundle of principles governing the planning of activities with respect to their mutual dependencies, For example:

- Activities are allotted to persons and groups.
- Resources are allocated.
- Data and context-information are distributed.
- Activities are triggered and synchronised.

11.3 Speech Act Theory for Establishing a Meta-Dialogue

Speech act theory is the foundation of the language/action perspective. Speech act theory was formulated by Austin and Searle and the used by Flores and Winograd, in "The Coordinator", to build one of the first groupware systems. The different categories of speech are analysed: assertive, directive, comissive, declarative, expressive. As an example the system will analyse the sentence "May I have the file on XY" and recognise that this is not a question to be answered with yes or no but a request to be fulfilled in sending the file. In that way "The Coordinator" uses a meta-dialogue in order to control the interaction.

11.4 Supporting Dialogue by Artefacts

Artefacts play a major part in every day life. Their meaning is usually learnt gradually during basic education. Switching to an unfamiliar environment needs the creation, learning, and comprehension of additional artefacts. The pictograms in an airport may provide a good example for this necessity. In an similar way interaction via computers needs a lot of artefacts to ensure comprehension.

11.5 Structuring Tasks by Means of Articulation Work

Articulation theory, according to Strauss, points at the manifold mechanisms necessary in performing distributed work: divide, coordinate, allocate, schedule, connect etc. In order to reduce complexity important mechanisms of interactions, covering stereotype situations, have been generated. Schmidt and Rodden cite the following ones:

- organisational structures defining roles, obligations and entitlements in formal and informal ways
- plans and schedules covering particular situations such as group calendars, and meeting schedules
- standard operating procedures such as processes defined in work flow management systems
- conceptual schemes such as classification schemes, taxonomies and thesauri.

11.6 Modelling Intelligent Agents by Enactment Theory

Enactment Theory is a very general theory on human agents and can be used in modelling "intelligent agents" in CSCW systems. Mahling proposes enactment chains concerning the following items: task initialisation, planning and scheduling of tasks, task execution, and evaluation of task. In that way a lot of characteristics are attributed to intelligent agents. They comprise a spread of different points such as: wishes and goals, roles and capabilities, activity knowledge, priorities, preferences and expectations.

12. Interdisciplinary Design of CSCW

12.1 Disciplines Contributing to CSCW

There are a lot of disciplines who bring in their specific contribution:.

- At first, CSCW is heavily based on the development of information technology. Thus informatics with its various sub branches is a dominant constituent.

- As associate disciplines, operational research and systems sciences help to break down problem areas as well as domain structures and processes.
- Management sciences and its associated disciplines, e.g. organisational theory, come further in and help to incorporate an understanding of the social organisation of activities. These perspectives may be concerned with matters such as structures and hierarchies, division of labour, organisation of work, rules, and work practice. Also included is work practice with professional and organisational roles - may they be formal or informal.
- Methods from ethnography enable us to characteristically improve work place analysis in non-trivial work settings, e.g. airports and fire brigades .
- Further there is an urging need to take into account the cognitive perspectives in the relationship between users and the systems. It is necessary to extend from the more easily appreciated requirement for the physical and perceptual usability of systems through to the cognitive ergonomics and to human computer interaction (also called HCI, CHI, or MMI).
- Organisational psychology is concerned with different matters including such examples as intellectual composition of groups, motivation to work, interpersonal factors like interpersonal skills and self-presentation.

12.2 On the Quest for An Interdisciplinary Design

A caveat has to be stated: Adding one or the other theory will not suffice to understand and shape CSCW. It is an urgent necessity to bring together different scientific disciplines to a close cooperation in a common goal. To make such cooperation operational it is necessary to combine different areas. So techniques, approaches, and methods from a broad diversity of research fields come in. There are:

1. Enabling technologies from core computer science: networking, databases, user-interfaces, multimedia, knowledge processing,
2. Basic design principles and information systems design methods have to be combined with architectural and technological concepts for CSCW.
3. Organisational and sociological approaches are the third area. They include ethnographic approaches, work analysis and redesign as well as cognitive ergonomics.

Combining concept and approaches of the different disciplines has been a permanent task in informatics. Already, in the first approaches to information systems analysis and design, technology oriented approaches and organisational considerations had to be met. Yet it needed a long struggle until today's methodological landscape of design methods was formed. Hence bringing together the concepts and views of such differing disciplines, as listed above, will be a major challenge.

Acknowledgement: The author is very indebted to Steve Guest, Loughborough University of Technology, for contributing improvements in style and content.

[1] CSCW90: Proceedings of the Conference on Computer-Supported Cooperative Work, Oct 7-10, 1990, Los Angeles, CA, 1990.
[2] CSCW92: Proceedings of the Conference on Computer-Supported Cooperative Work, Oct 31-Nov 4, 1992, Toronto, 1992.
[3] CSCW94: Proceedings of the Conference on Computer-Supported Cooperative Work, Oct 26-28, 1994, Chapel Hill, ACM, New York, 1994.
[4] COST14: SCHMIDT, K. (ed). Developing CSCW Systems: Design Concepts, COST14-Report, Riso-Report, Roskilde 1993.
[5] ECSCW91: Proceedings of the 2nd European Conference on Computer Supported Cooperative Work, Sept 24-27, 1991, Amsterdam, Kluwer, Dordrecht, 1991.
[6] ECSCW93: Proceedings of the 3rd European Conference on Computer Supported Cooperative Work, Sept 10-13, 1993, Milano, Kluwer, Dordrecht, 1993.
[7] SCHAERDING93: SHAPIRO, D., TAUBER, M., TRAUNMUELLER (ed.), Proceedings of the IFIP 13.2 Workshop "Design of CSCW and Groupware Systems", June, 1-3, 1993, Schärding, North-Holland, Amsterdam, in press.
[8] SHAPIRO, D., TRAUNMÜLLER, R., CSCW and Public Administration, in: BONIN, H. (ed.), Proceedings of the IFIP WG 8.5 Workshop "Systems Engineering in Public Administration", Lüneburg, March 1993, North-Holland, Amsterdam 1993.
[9] SPECIAL ISSUE: "The Changing Office", PC Magazine, Vol.13, No.11, June 1994.
[10] SPECIAL SECTION: "The New Document", Vol.19, No.8, Byte, August 1994.
[11] TC8AUS: GLASSON, B., HAWRYSZKIEWYCZ, I., UNDERWOOD, A., WEBER, R. (ed.), Proceedings of the IFIP TC8 Conference on Business Process Re-Engineering, Bond University, May 1994, North-Holland, Amsterdam, in press.

Standardization of Communication between Organizational Decision Units

Miklós Biró, László Kovács
Computer and Automation Institute of the Hungarian Academy of Sciences
Informatics Research Laboratory
H-1111 Budapest XI. Lágymányosi u. 11. Hungary
E–mail: laszlo.kovacs@sztaki.hu

Abstract

A computational Reference Model of Distributed Group Decision Support Systems (RM-DGDSS) is described. This Reference Model is a supplement of the standardized Reference Model of OSI (Open Systems Interconnection) and can be a candidate for standardization in ODP (Open Distributed Processing). The RM-DGDSS defines a distributed architecture environment for supporting a group of decision makers connected via computer network. Entities of the Reference Model and their relationship are presented as well as the relation between the RM-DGDSS and Reference Model of ODP (RM-ODP). Research and development towards the standardization of specialized protocols for decision support (PDS) are outlined.

Key Words: *distributed group decision, GDSS, groupware, collaboration, communication protocol, CSCW, ODP, reference model*

1. Introduction

Organizational decisions usually involve intensive communication between the organizational entities such as different stakeholders, departments, managers. This work[1] is the very first attempt in the direction of the research, development, and standardization of specialized computer network protocols for application in the area of Distributed Group Decision Support Systems (DGDSS). We are convinced that the DGDSS area is mature enough and is at the level where international cooperation is necessary to develop standardized protocols providing communication services for entities that support the human actors of decision processes.

Enabling (computer and communication) technologies, the proliferation of computer network services and distributed systems make it viable to raise the question of computer network support of the cooperative group decision process. [Kovács 93]

In the group decision process a set of human decision makers work together to achieve common judgement on several issues. Generally, this kind of human activity is carried out in decision room environments. Within a decision room environment, decision conferences are organized among human users, mainly managers. Decision conferences are usually synchronous, face-to-face meetings. If several similarly equipped remote decision rooms are connected via real-time video and/or audio links, a media space is created. Practically, within media space environment the same synchronous type of face-to-face meetings can be carried out. In the individual meeting rooms or multimedia

[1] The work is supported by OTKA grants #2571 and #742.

technology created media space, meetings are supported by computer technology besides other forms of analog and/or digital communication technologies (like projected shared video screen, telephone, fax etc.). Within these types of environments network connected workstations provide computational support for the decision making process. This computational support usually covers different phases of the face-to-face meeting (from brainstorming, through formal voting procedures to collaborative minutes writing). Several research prototypes and commercially available systems are attainable for these purposes.

Other distributed approaches of GDSS are based completely on computer network services. One of them is called "decision network" [Finlay et. al. 91]. The decision network approach is based on the idea of computer supported asynchronous meeting of the decision making process. The "meeting" is distributed not only in space, but in time. Participants and/or the decision making process completely rely on the computer network services, without additional communication modes like on-line video/audio links. Usually the decision making process is longer than several days.

In both cases a computer network is necessary to provide the required communication services for human participants and/or their software actors. In this paper a new computational Reference Model of Distributed Group Decision Support Systems (RM-DGDSS) is presented within the framework of the Reference Model of Open Distributed Processing (RM-ODP). The relationship between the RM-ODP and RM-DGDSS is going to be discussed as one of the main topics of this contribution.

The RM-DGDSS defines a distributed architecture for supporting a group of decision makers connected via computer network. As usual, reference model defines a vision, a strategic outline for the standardization of this kind of distributed processing. The main purpose of RM-DGDSS is the description and standardization of basic ideas, entities, algorithms etc. of distributed group decision process. It provides a common language and basic understanding for researchers and developers of this area. Without this common knowledge background, the various ideas and techniques can be easily misinterpreted by different humans.

The structure of the RM-DGDSS is defined in terms of object oriented systems descriptions. Basic entities and their logical relationships are described. Communication requirements are formalized using the technique originally developed for computer communication protocols. A protocol defines the way in which the communication takes place between entities of the Reference Model. The unambiguous definition of protocol gives the rules and regulations that govern the interactions between these entities. If entities that must work together to realize the distributed system follow these rules (protocols) then the interworking can take place without problems. Otherwise the smooth interworking is not guarantied. The specialized protocol definition for DGDSS is a new approach. It has the following advantages:

- The application of the internationally standardized protocols for decision support (DSP) promotes the smooth interworking of multiplatform implementations of decision support systems. It gives the possibility to apply different software products of different vendors that conform to the standard.
- It is a well-know and well proved technique of computer communication and distributed systems disciplines.
- It can enlarge the application areas of DGDSS as it opens a new territory for international cooperation. International decision meetings can be supported via software tools conform to the protocol standard developed.
- It encourages the research and development of formal techniques and models of DGDSS.
- It increases the confidence in the results of group decisions and in the area of DGDSS as well.

2. Reference Model of Open Distributed Systems

The Reference Model of Open Distributed Systems (RM-ODP) provides a co-ordinating framework for the standardization of ODP. It identifies and standardizes the basic elements, the concepts, the algorithms of architectures

of distributed processing developments. The Reference Model of ODP is based on two fundamental requirements that motivate the current developments. The first is the demand for distribution of information systems and the information itself. Distribution is a natural reflection of the real world situation (e. g. distributed enterprise and its operations). Distribution can be used to promote the parallel processing, the timely access of information or to reduce the negative consequences of failures. The second basic requirement is the openness. This concept provides the way of dealing with the heterogeneity of equipments, operating systems, programming languages, da-tabases, applications, etc. used in distributed systems.

The RM-ODP defines the technical bases for standards on ODP architectures and interfaces so coherent distributed systems can be constructed from heterogeneous and reusable components (objects). [RM-ODP 92] According to the organization idea and the terminology of the Reference Model of ODP, our main aim is the development of a "Specific Reference Model" for Distributed Group Decision Support Systems (RM-DGDSS) which covers the individual, particular concepts of group decision support. The basic concepts and common functions of RM-ODP are going to be used and additional conceptual details, specific functions of DGDSS are going to be defined.

The complexity of distributed systems is dealt with two ways of abstractions: the service and the viewpoint concepts. These essential concepts are presented here, in brief.

2.1 The Service Concept

The origin of the idea of service can be found in the traditional black-box concept. In computer-communication systems, the complex and numerous communication functions are decomposed into subsequent layers. This decomposition method subdivides the functions into individual subsets of different layers. Functions of a (N) layer are considered as a communication service provided by the (N) layer. This service (a set of functions) is used by the upper (N+i, i=1,2,...) layer functions without the particular knowledge of the inner details of the (possible complex) lower-layer operations. An (N) layer can be considered as a black box for the upper (N+i, i=1,2,...) layers, while its functions are decomposed into lower layer functions. The (N) layer and all layers below form the (N) service provider. (Figure 1) An (N) service can be described by a set of service primitives (elementary communication functions). These service primitives are accessible through the service boundary (service access points) between the adjacent layers.

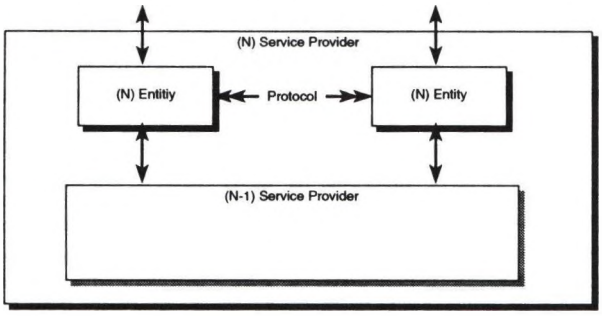


Figure 1. Decomposition of (N) Layer functions

In Figure 1 the (N) layer provides the (N) Service. The (N) Service is decomposed into (N-1) Service and the added communication functions provided by the pair of cooperating (N) Entities. The cooperation is controlled by the (N) Protocol (see below).

The layering concept, as a special kind of decomposition method was successfully utilized in the ISO Reference Model of Open Systems Interconnection for computer communication net-works [MacKinnon 90].

2.1.1 Protocols

Communication protocols are the rules and regulations that control the interworking between remote communication entities. A service defined is provided by a protocol specified. Practically, protocols are complex distributed algorithms, that govern the message passing between remote entities. The formats and semantics of messages, their valid sequences, the messages initiated local actions are described in the protocol specification. Within the framework of ISO OSI, a complete stack of communication protocols was specified.

The ultimate aim of this contribution is the initiation of an international cooperation for specification of a protocol (PDS) specialized for group decision support. This approach is beyond the ISO OSI protocol stack, which contains peer-to-peer protocols for communication purposes. PDS is inherently a multiparty protocol (an algorithm constituting the ability of interworking with more than two communication partners (entities)). This protocol development can only be a part of our bigger and visionary project, namely the development of the Reference Model of DGDSS.

2.2 The Viewpoint Concept

In addition to the service concept of ISO OSI, the RM-ODP introduced an new concept, namely the viewpoints. A distributed system can be considered from different viewpoints that represent different abstractions of the original system. This leads to different representations of the same distributed system. These representations emphasize different concerns, concerns that are relevant or irrelevant from a particular viewpoint. During the work on RM-ODP, five different viewpoints were identified: enterprise, information, computational, engineering and technology. In contrast to the layers of the service concept, these viewpoints are independent of each other. In a particular viewpoint a complete (formal) description of a distributed system can be given. [RM-ODP 92]

- The enterprise viewpoint defines how and where the distributed information system is placed within an enterprise. The users, the objectives of the system, the business requirements are identified and expressed in terms of objects representing user roles, business and management policies and the environment.
- Information viewpoint considers the distributed system in terms of information structures and flows, rules and constraints that govern the manipulation of information. Within this viewpoint the manual and automatic manipulation of information are not differentiated.
- Computational viewpoint describes the distributed systems in terms of programming functions and data types. This kind of description is independent of the computer systems and networks on which the distributed system is based upon. The requirements of distribution transparencies are identified.
- Engineering viewpoint considers the systems from the viewpoint of experts of operating and communication systems. The implementation details are described here.
- Technology viewpoint describes the system in terms of hardware and software artefacts (program products, input-output devices, operating systems versions, communication access points etc.).

These viewpoints have different individual description languages defined for writing specifications of ODP functions. The different languages are interrelated to each other in order to be able to verify the mutual consistency of a global ODP specification. A distributed system is well defined if all the five descriptions from these viewpoints are given.

3. Reference Model for Distributed Group Decision Processes

Reference Model of DGDSS (RM- DGDSS) is an abstract computational model. It defines the basic concepts, the entities, the objects, the operations and the terms used in Distributed Group Decision Making Processes. This general

model does not have any particular implementation issues. The main purpose of creating this reference model is the promotion of the basic understanding of the problem of Distributed Group Decision Making Processes and the initiation of international cooperation on the development of protocol standards that support this type of decision making process. Entities of Reference Model and their relationships are going to be presented in this section.

Reference Model of DGDSS describes a multiparty, multicriteria, distributed single issue decision making process. The RM-DGDSS can be easily extended to define a multi-issues reference model. This extension is left for the reader.

RM-DGDSS is to be developed according to the viewpoints of RM-ODP. This paper doesn't intend to deal with the engineering and the technology viewpoints of RM-DGDSS, it concentrates on the information and the computational viewpoints only. Analysis of the Distributed Group Decision process from the enterprise point of view can frequently be found in the corresponding literature, although the idea of enterprise viewpoint (as it is defined in RM-ODP) has not been mentioned and discussed previously. [Jacob et. al. 92, Kraemer et. al. 88] In our discussion on the development of RM-DGDSS, both informal, and (semi)formal description methods are used.

3.1 Information and Computational Viewpoints

In this section an integrated abstraction is given which corresponds to the information and the computational viewpoints of RM-ODP. Further clarification and separation of the different aspects and these viewpoints are needed.

The following is a list of the fundamental entity types involved in the distributed decision making process described by RM-DGDSS:

- human decision makers (partners, users, participants),
- decision issue,
- subjective and objective criteria including goals, requirements and limitations
- alternatives

Group decision making process is a cooperative work of a finite set of human decision makers. In theory, a subset of the criteria could be considered as representing a decision maker and the group ranking of the alternatives could be determined in similar way to the individual rankings. This means that group ranking could be considered as a special multicriteria decision making problem, where each member of the group has his own set of criteria [Biró et al. 91] [Marchant 92]. A different approach is necessary however, since a social consensus has to be reached in this case, in contrast to lifeless criteria which will never protest [Biró et al. 1992a]. Another argument for differentiating decision makers from criteria is the fact that decision makers are directly associated to the distributed nature of the problem.

The RM-DGDSS assumes that decision makers are located in different places. A computer network is presumed that connects these different locations of participants. All participants can use the services of this computer network including the specialized network service for the decision making process described in this paper. Participants have unique names within the environment. This name is a composite of the natural name and the network address of a participant. RM-DGDSS requires the uniqueness of names of participants. In the case of asynchronous decision process in which e.g. electronic mail service is used as the basic communication facility between participants and their software actors, electronic addresses can be applied as unique names of partners.

Decision process can take place in longer periods of time. In RM-DGDSS, time is considered as a continuous dimension of the decision making process. There is continuous transition from the synchronous decision making process to the asynchronous process, depending on the time duration of the process. In synchronous decision making process participants are on-line. Almost instantaneous events and operations are required. In asynchronous environments longer periods, schedules (like several hours, days, weeks) are assumed for a particular decision process. Different time schedules require different techniques and support tools, but RM-DGDSS doesn't differentiate between these two kinds

of decision processes. Thereafter, RM-DGDSS does not deal with implementation techniques, this remains to the implementation phase of the support tool development.

During decision making, decision makers can enter and leave the process. This means that the set of participants changes dynamically in time and of course, in space, since different participants can move to different space coordinates. This fact is referred to by the activity states of participants. A participant can be active with the meaning that he/she enters the decision group and works together with other active partners. If a participant leaves the decision making process then he/she becomes a passive partner of the process.

Moving is described by the change of location of decision makers. Reference Model considers the decision making process as completely location transparent. Participants can change their location with no restriction at any time. The basic attributes of the decision makers are the following:

```
types
    activityState = (active, passive)
    description = ...
    location = ...
    type = ... (* e. g. boolean, enumerable, integer, float *)

object DecisionMaker is
    Attributes: uniqueName, location, activityState, description, ...
    Operations:
        + EnterDecision(uniqueName,location,description)
        LeaveDecision
        GetUniqueName():uniqueName
        GetUniqueName(description):uniqueName
        GetUniqueName(location,description):uniqueName
        GetLocation():location
        GetDescription():description
        GetState():activityState
        ChangeState(activityState)
        ChangeLocation(location)
        ChangeDescription(description)
```

Decision making process is controlled by a (human) mediator. Mediator initiates, prepares the different phases of the decision making process. He/she defines the issue(s) for decision and organizes the human group of decision makers for the decision making process. His/her responsibility is to distribute the results among the participants after the decision making. The role of the mediator can be performed by a participant as well, but their roles are completely different. During the process, the mediator has a principal responsibility for the convergence of decision making process. His/her is responsible for the complete process and its deliverable, namely the decision.

```
object Mediator is
    Attributes: uniqueName, location, ...
    Operations:
        + NewMediator(uniqueName,location)
        DeleteMediator
        GetName():uniqueName
        GetLocation():location
        ChangeLocation(location)
```

object Decision **is**
Attributes: uniqueName, mediator, issue, alternatives, critera, description, deadline, ...
Operations:
+ NewDecision(uniqueName, issue, deadline,
uniqueName (*Mediator*),
set of uniqueName (*DecisionMaker*),
set of uniqueName (*Alternative*),
set of uniqueName (*Criterion*))
DeleteDecision
GetName():uniqueName
GetIssue():issue
GetMediator():mediator
GetDeadline():deadline
GetDescription():description
ChangeMediator(mediator)
ChangeDescription(description)
ChangeDeadline(deadline)

The alternatives are the basic subjects of decision process. Alternatives are mutually exclusive entities (activities, objects, projects, or modes of behavior) among which a choice is possible within an actual decision. [Zeleny 82] During the preparation phase of the process (see below) decision makers (and mediator) can suggest new alternatives and can propose the suppression of previous alternatives. At the end of the preparation phase a consistent set of alternatives is established. An alternative has a unique name. The mediator provides for the uniqueness of names of alternatives. An alternative can be characterized by a value with respect to each criterion. The type of this value can be selected from the following typeset: {boolean, enumerable, integer, float}.

object Altenative **is**
Attributes: uniqueName, type, description, ...
Operations:
+ NewAlternative(uniqueName,type,description)
DeleteAlternative()
GetName():uniqueName
GetDescription():description
GetType():type
GetValue(uniqueName(*Criterion*)):value

The criteria are measures, rules and standards that guide decision making. The specialized basic subtypes of criteria are attributes, objectives, and goals. Attributes refer to descriptors of objective reality. They can be identified and measured in relative independence from the decision maker's needs or desires. Objectives represent direction of improvement or preference along with attributes (maximize, minimize). Goals refer to required levels of achievement of attributes or objectives. They are a priori determined specific values. [Zeleny 82] Criterion has a unique name. An informal definition is attached which is incrementally specified by the decision makers and/or the mediator. Criteria are weighed according to their relative importance. Weights are the normed numerical values assigned to criteria by decision makers. Weights are negotiable parameters during the preparation phase of decision making process.

object Criterion **is**
Attributes: uniqueName, type, value(*min*), value(*max*), description,weight, ...
Operations:
+ NewCriterion(uniqueName,type,value(*min*), value(*max*),description,weight)

DeleteCriterion()
GetName():uniqueName
GetDescription():description
GetMaxValue():value
GetMinValue():value
GetType():type
GetWeight():weight
GetWeight(uniqueName(*DecisionMaker*))
ChangeWeight(weight)
ChangeWeight(uniqueName(*DecisionMaker*), weight)
ChangeDescription(description)
GetValue(uniqueName(*Alternative*)):value

Competences are assigned to the decision makers measured by an independent partner (mediator) with respect to every criterion. Competences are normed numerical values as well.

object Competence **is**
Attributes: uniqueName, value, ...
Operations:
+ NewCompetence(uniqueName,
uniqueName(*Mediator*),
uniqueName(*DecisionMaker*),
uniqueName(*Criterion*))
DeleteCompetence()
GetName():uniqueName
GetValue():value
ChangeValue(value)

Access rights are restrictions on decision makers for accessing various entities (objects, criteria, etc.) during the different phases of decision making process.

object AccessRight **is**
Attributes: uniqueName, right, ...
Operations:
+ NewAccessRight(uniqueName,
uniqueName(*Mediator*),
uniqueName(*DecisionMaker*),
uniqueName(*Criterion*),
uniqueName(*Alternative*))
DeleteAccessRight
GetName():uniqueName
GetRight():right
ChangeRight(right)

The RM-DGDSS can be represented in a four dimensional space where the dimensions correspond to the Alternatives, DecisionMakers, Criteria, and Time. Different three dimensional mappings can be generated according to the given purpose. These mappings can, of course, be transformed into one another. E. g. The World Bank requires the criteria

and their weights to be determined before any tender evaluations. Therefore it is sensible to consider a mapping which freezes the number of criteria and the criteria themselves. Such a representation is introduced and discussed in the last section. Another approach, the Analytic Hierarchy Process represents the entities in a hierarchic structure descending from an overall goal to criteria, subcriteria, and alternatives in successive levels. [Saaty 90] This approach can be readily generalized to a group decision process as already described above.

4. Protocol for Decision Support (PDS)

A set of protocols is to be specified or simply used for the purpose of supporting the Distributed Group Decision Process. In the definition of a protocol, the objects of RM-DGDSS are used.

There exists a number of protocols which are potentially useful for distributed group decision support. These are for example SMTP, X.400, X.500, telnet, ftp, nntp, etc.. Numerous software systems are built on these protocols and they are becoming more and more popular. However, Distributed Group Decision Support has specific characteristics which are not directly assisted by any of them. The characteristic which refers to the meeting aspects of the group decision processes has no straightforward support . The objects defined in the previous section have no interpretation within the other existing protocols. These facts were at the origin of our motivation for developing the special protocol (PDS) for DGDSS.

Protocol for Decision Support (PDS) is a multiparty communication protocol, as it is illustrated in Figure 2. This figure shows the parties of the decision making process connected with PDS protocols. The architecture is centralized to the mediator. Nevertheless, other communication means can weaken this centralization by permitting direct cross communication links among the participants. The figure contains organizational units the objects of which are meant to be linked to the same physical control domain.

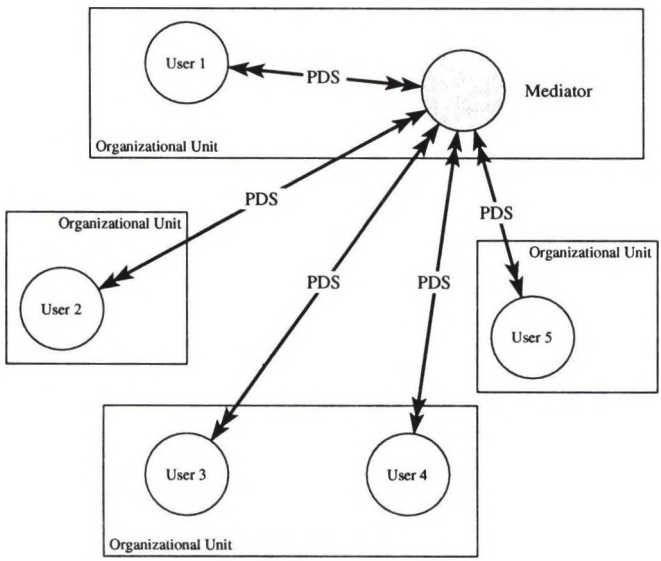


Figure 2 Architecture of Distributed Group Decision Support System

Here only the service definition of PDS is going to be discussed. The protocol specification and its (semi)formal description is out of the scope of this paper. Furthermore, the actual protocol specification[2] of PDS requires international standardization efforts which we intend to initiate and promote.

PDS has three phases, the preparation, the evaluation, and the reporting phases. During the preparation phase mediator creates a *Decision* object (see above) by specifying the name of the *Decision* (uniqueName), the issue (*issue*) of the decision making process. An informal description of the decision (*description*) and the deadline (*deadline)* are attached. The initial sets of alternatives, criteria, and decision makers have to be specified as well. These sets can be modified during this preparation phase. The modifications consist of additions, deletions of objects of alternatives, criteria, and decision makers. The definitions (*description*) of the decision and the other objects are established by means of negotiations which can make use of other available protocol based communication software tools. Weights of criteria with respect to the individual decision makers are either determined by the mediator as a possible result of preceding negotiations or may be defined individually by each decision maker. Competences and access rights have to be assigned by the mediator of course. An essential part of the decision making process is the evaluation procedure (algorithm) which can be negotiated as well during the preparation phase. At the end of this phase the global space of objects of the decision process is ready for use.

The second phase of the operation of PDS is the evaluation of the decision. This phase is initiated by the mediator. The objects of the typical communication in this phase consist of individual value judgements and their confirmations. This procedure is fully under the control of the PDS protocol. The value judgements are captured by the subsystem which is responsible for performing the evaluation algorithms. The result(s) of the evaluation is(are) available for the mediator immediately. The mediator has the possibility to communicate intermediate evaluations to the participants as a built-in service of the PDS protocol.

In the third, reporting phase all available communication services and/or tools can be used to present the result(s) of the decision making process to the participants and/or other interested parties.

This informal definition of the PDS service has to be followed by the formal definition of the service and the informal and formal specification of PDS protocol which is beyond the scope of this contribution. The most suitable form of the follow-up specification and standardization work is the cooperation between interested parties.

5. Example: A Representation of DGDSS

The objective of this section is the presentation of a prototype DGDSS. There is a one-to-one correspondence between the entities of the RM-DGDSS and the entities of the specific representation although this is not a general requirement. The representation is displayed in a three dimensional space where the dimensions correspond to the Alternatives, DecisionMakers and Criteria. There is in fact fourth dimension, the Time which has been left our for the sake of simplicity of this presentation. Time can be displayed as a series of figures of the type below.

The alternatives are represented by separate windows for each particular decision maker. The windows contain scrollbars and textfields containing numerical values corresponding to the various criteria. A scrollbar is used to set the valuation of the alternative with respect to the given criterion by the decision maker. The textfields attached indicate the limiting values, and the current value of the valuation. The weight of a criterion is entered into another textfield under the scrollbar.

The values set by the scrollbars are automatically transferred to the server generally operated by the mediator via the PDS protocol. The values are aggregated by appropriate evaluation algorithms (e.g. multiple attribute utility

[2] The unambiguous specification of the structures, formats of protocol messages and their valid sequences.

decomposition techniques, outranking methods, voting methods, etc.) selected previously in the preparation phase of the decision making process. The results are seamlessly retransferred to the participants and presented to them in a most appropriate form depending on the given problem.

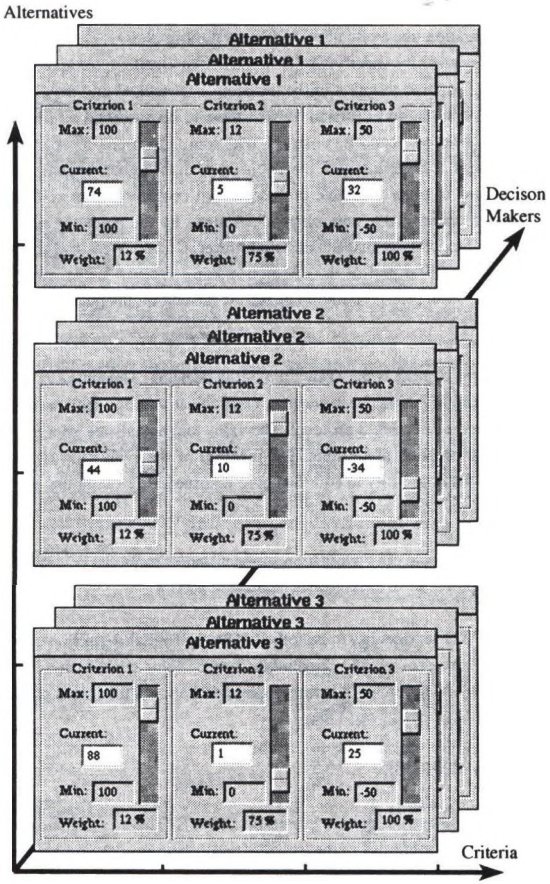


Figure 3 A Representation of the RM-DGDSS Model

6. Conclusion

In this contribution a computational Reference Model of Distributed Group Decision Support Systems (RM-DGDSS) was defined. This Reference Model is a supplement of the standardized Reference Model of OSI (Open Systems Interconnection). The reference model is proposed for standardization within ODP (Open Distributed Processing). This is facilitated by the fact that RM-DGDSS adheres to the structure (viewpoints) of RM-ODP. The RM-DGDSS defines a distributed architecture environment for supporting a group of decision makers connected via computer network. Once the international standardization is completed, users – working with different model representations compliant to the RM-DGDSS in a possibly multivendor environment – can seamlessly cooperate.

References

[Biró et. al. 91] Miklós Biró, Péter Csáki, Mátyás Vermes: "*WINGDSS Group Decision Support System under MS-Windows*", In: Proceedings of the Second Conference on Artificial Intelligence (ed. by I.Fekete and P.Koch). John von Neumann Society for Computer Sciences, Budapest, Hungary, 1991, pp. 263-274.

[Biró et. al. 92a] Miklós Biró, Ferenc Jamrik, Gábor Janek, Előd Knuth,, Tibor Remzső,"*Factors Influencing the Design of a Prototype Decision Support Application for a Distributed Object Management Project*", In: Experience with the Management of Software Projects 1992 (eds. P. Elzer, V. Haase). Annual Review in Automatic Programming Volume 16, Part II Pergamon Press, 1992,pp. 129-132.

[Biró et. al. 92b] Miklós Biró, Ede Bodroghy, Attila Bor, Elő d Knuth, László Kovács, "*The Design of DINE: A DIstributed NEgotiation Support Shell*", In: Decision Support systems: Experiences and Expectations, Proceedings of the IFIP TC8/WG8.3 Working Conference Fontainebleau, 1992, North-Holland, 1992, pp. 103-114 (IFIP Transactions A-9)

[Biró et. al 94] Miklós Biró, László Kovács, "*Protocols for Cooperative Multiple Criteria Decision Processes*", Decision Support in Organisational Transformation, Proc. of the IFIP WG8.3 Working Conference, San Sebastian, Spain, 1994

[Finlay et al. 91] P. Finlay, C. Marples, "*A review of Group Decision Support Systems*", OR Insight Vol. 4. No. 4. 1991

[Jacob et. al. 92] Varghese S. Jabob, Hasan Pirkul, "*A framework for supporting distributed decision-making*", Decision Support Systems 8, North Holland, 1992 pp. 17-28

[Klein et. al. 90] Michel Klein, Leif B. Methlie, "*Expert Systems A Decision Support Approach*", Addison-Wesley, 1990

[Kovács 93] László Kovács, "*Multimedia Groupware Systems*", Rapports de Recherche, Ecole Normale Supérieure de Cachan, LIFAC, 93-3, In: Proceedings of the CON'93, 8th Austrian-Hungarian Conference in Informatics, (eds. D. Sima, G.Haring) R. Oldenbourg Wien, München 1993, pp. 27-41

[Kovács 94] László Kovács, "*The GroupSPACE Concept*", Proc. of the 14th IEEE International Conference on Distributed Computing Systems, Poznan, Poland, 1994

[Kraemer et. al. 88] Kenneth L. Kraemer, John Leslie King, "*Computer-Based Systems for Cooperative Work and Group Decision Making*", ACM Computing Surveys, Vol. 20, No. 2. June 1988, pp.115-146

[Liang et. al. 90] Luping Liang, Samuel T. Chanson, Gerald W. Neufeld, "*Process Groups and Group Communications: Classification and Requirements*", IEEE Computer, February 1990, pp. 56-66

[MacKinnon 90] Dennis MacKinnon, William McCrum, Donald Sheppard: "*An Introduction to Open Systems Interconnection*", W.H.Freeman and Company, New York, 1990

[RM-OSI 80] *Information Processing Systems – Open Systems Interconnection – Basic Reference Model*, ISO 7498.

[RM-ODP 92] *Basic Reference Model of Open Distributed Processing (RM-ODP)*, ISO 10746

[Marchant 92] Thierry Marchant, "*PROMETHEE and GAIA in a multi-decision maker environment*", Technical report, Free University of Brussels, 1992.

[Saaty 90] Thomas L. Saaty, "*How to Make a Decision: The Analytic Hierarchy Process*", European Journal of Operational Research 48 (1990) 9-26.

[Zeleny 82] Milan Zeleny: "*Multiple Criteria Decision Making*", McGraw-Hill, 1982

Workflow Products

"Workflow" by CSE

Gerd JANITSCHEK
CSE Systems Vienna

Product Presentation

"LinkWorks": Workflow with an Object-Oriented Syste

Pieter VAN KAMPEN
DEC

Product Presentation

A Document Management and Archiving System (DOCTAR)

L. Langer

Paper not received in time.

Workflow and Information Structuring from the Perspective of Lotus Notes

A. Kolesar

Paper not received in time.

Usability of Groupware Products for Supporting Publishing Workflows

Franz Burger, Siegfried Reich
FAW - Research Institute for Applied Knowledge Processing
Softwarepark Hagenberg, Hauptstraße 99, A-4232 Hagenberg – Austria
email: {franz, sre}@faw.uni-linz.ac.at

Abstract: *This paper describes different workflow management systems from a publishing point of view. The intention is to derive the special requirements which arise when dealing with compound structured documents and to investigate to what degree these requirements are fulfilled by currently available groupware products. The requirements are identified and an investigation of three typical representatives of workflow tools is given.*

Keywords: *CSCW, Groupware, Workflow, Electronic Publishing, Compound Structured Documents, SGML*

1. Introduction

New evolving technologies in the publishing sector like advanced communication services, development of international standards for the representation of publishing information, new delivery platforms, and others open up new opportunities for the design and development of new information products [4]. According to the idea of concurrent engineering [16] and the prospects of these new technologies not only new products, but also new development *processes* have to be designed complementary in order to exploit the benefits from recent research.

However, even though desktop software products emerged rapidly, there is still a lack of integrated document management systems. *Workflow Management* has become a pervasive term associated with a number of software products, among them 'simple' mail tools, document management tools, as well as 'real' workflow management systems. Workflow Management is seen as a key element

for supporting the process development of publishing needs by integrating existing desktop software solutions.

The expected benefits of workflow management can be summarised as follows [6, 14, 18]:

- increased availability of information, in terms of actuality and access time, i.e. faster and - by use of networks - location independent access to information.
- advanced control of the publishing process, i.e. at any time control over all jobs/tasks or their state, respectively. Management decisions can be made more quickly and are based on the latest information available.
- advanced security and integrity of the handled documents by use of information technology.
- documentation: re-engineering of the existing process results in a (re-)usable documentation.
- closer coordination with customers.
- the possibility to manipulate large quantities of process-data or task-data as it is necessary for large publication projects, like professional reference works.
- the quality of the production process is ensured. With respect to ISO-9000 [8] this can be a crucial success factor.
- productivity increases: this overall benefit is a consequence of all the other benefits. It should result in
- cost reduction.

The objective of this paper is to investigate to what degree currently available workflow products can cope with the requirements being set by electronic publishing. The outline of the paper is as follows: section two deals with requirements of electronic publishing and in this way defines the criteria for the following investigation. In section three we investigate three example products, namely Lotus Notes in the current version 3.0, Visual WorkFlo from FileNet and Action Technologies' Action Workflow. A summary in tabular form as well as a conclusion describing the applicability of these tools are subject of section four.

2. Requirements

We have identified a number of properties of Electronic Publishing that affect the choice of a suitable workflow management system [cf. 14]. These prerequisites are described in detail. It has to be mentioned, that this section deals with *potential* requirements, i.e. the question held in mind is 'How should an ideal workflow management system for the handling of compound structured documents look like?'

2.1. Basics

2.1.1. Heterogeneous Hardware and Operating Systems

Nowadays in almost every organisation various hardware and software platforms do exist. In order to integrate these different systems and of course to reuse existing applications, the system* has to support a client/server architecture running on different hardware and operating system platforms used in the office area, like Macintosh, MS-Windows, OS/2 PM or UNIX with X-Windows. Furthermore, different network topologies like e.g. Ethernet or Tokenring and protocols like e.g. Novell XPC or TCP/IP have to be supported.

2.1.2. User Friendliness

User Friendliness is a matter of course, nevertheless it is of great importance and therefore has to be mentioned. Features like e.g. graphical user interfaces are state-of-the-art and not specific for workflow applications. Other characteristics like the possibility of specifying tasks or users and their roles in a graphical mode, the representation of a publication's status with a graphical tree structure, or the easy changing of the workflow by simple mouse manipulation are workflow specific and of course have to be supported.

2.2. The Underlying Workflow Model

Support of workflow modelling should be the core of every workflow management system. A suitable workflow model builds the basis. Criteria are

- splitting up of tasks into sub tasks
- querying the status of tasks
- support of a role concept
- definition of rules (or constraints) for the automatic handling of tasks
- possibility to change definitions at run-time
- simulation of specified dependencies in order to allow a first evaluation

2.3. Re-Engineering the Business Process

New technologies as well as competitive challenges of the business require changes in the business process. Tool support identifying miscoordination, helping to clarify accountability of tasks and

---

* When from now on simply 'system' is referred to, a workflow management system for handling structured documents is meant.

satisfaction of customer needs, and for reducing time to market by focusing on the whole business process is needed.

2.4. Database Functionality

Every publisher has to deal with a great amount of data. Therefore the database somehow can be described as the core of a publishing application. Besides general needed functionality like manipulation and query mechanisms, functionality providing security, and transaction mechanisms for the handling of distributed structured documents is needed. Especially the following aspects are of great importance:

- Distribution: As publishing is an inherently distributed activity, the underlying database paradigm should be that of a distributed database, too.

- Concurrency: Working on structured documents, as for example SGML [9] documents, results in especially long and nested transactions. The database system therefore should be able to handle such long transactions, i.e. it must not only allow nested transactions but also handle them efficiently.

- Authorisation control: The emphasis with respect to publishing is on guaranteeing the specified user rights on documents. For the specification of access rights a role-based concept would be appreciated.

2.5. Compound Structured Documents

A workflow management system for electronic publishing should be able to deal with *compound structured documents*. In order to understand the correct meaning of this term it is suitable to walk through it word by word: *compound* means that a document consists of several parts of information which may contain multimedia data, i.e. documents containing several different media types like text, images, audio, video, etc. *Structured* means that a document is built according to a specified structure specifying semantically distinguishable elements. A whole document as well as its elements can be handled solely. *Document* at last signals the conceptual representation of publishing information for handling and manipulation.

Therefore, a system and its underlying database should be able to manage documents of that kind. Several standards exist which support the handling of structured documents. The most important of them is SGML (Standard Generalized Markup Language)[9]. A number of other standards are

based on SGML or are 'able' to deal with SGML-documents*. The system- and application-independent representation of publishing information encoded in SGML allows to move towards open publishing systems. That is why a system should support at least the handling of SGML documents. The term *support* signifies that simply managing file pointers is not sufficient: parts of documents as well as attributes of them have to be accessed, e.g. they have to be queried and updated.

2.6. Communication Facilities

As already mentioned above, publishing is an inherently distributed activity. Hence communication is the thing which makes a publishing application really running. Functionality like mailing, sending and receiving of (parts) of documents or document sharing is based on communication facilities. With regard to integration of existing systems as well as openness to future development it is important to be independent of any physical implementations or necessities, i.e. that for example TCP/IP as protocol to be run by choice over an Ethernet or Tokenring network could be an appropriate solution. Support of advanced network technology like ISDN is another necessity arising from the special needs of publishers to transmit for example high volume image data.

2.7. Integration

Integration means reuse of existing documents (import / export facilities) as well as the integration of existing applications, be it standard applications or custom built ones, into the workflow management system. Moreover the problem is that existing applications do not 'know' of the workflow management system and that future applications only will cope with workflow to a certain degree and in a non-standardised way.

2.8. Development Environment

As every organisation is distinct and also within one organisation none two workflows will be the same, the system has to be really adaptable to an organisation's situation and environment. For this reason, an efficient development environment consisting of tools for analysis, specification and prototyping is needed.

---

* Standards to be named are HyTime (Hypermedia/Time-based Structuring Language)[10] for Hypermedia documents and ODA (Office Document Architecture) which is designed for office applications.

3. Representative Tools

This section investigates to what degree three representative workflow management systems can fulfil the requirements having been defined above. The tools under investigation are Lotus Notes 3.0, FileNet's Visual WorkFlo and Action Technologie's ActionWorkflow. Lotus Notes has been chosen because of its wide spread user basis. Visual WorkFlo from FileNet represents a group of tools which have evolved from image and document handling systems to workflow support. Last but not least ActionWorkflow has been taken as a tool with a sophisticated approach for dealing with workflow management.

3.1. Lotus Notes

3.1.1. Basics

Lotus Notes [11, 15] is the groupware tool with a the probably widest spread user basis. Notes consists of two primary programs: the Notes server and the Notes workstation. The Notes server running on OS/2 or Windows PCs provides services to Notes workstation users and other Notes servers, including storage of shared databases and mail routing. The Notes workstation which can be a Macintosh, a PC running Windows or Presentation Manager, or a UNIX machine, communicates with Notes servers. Its availability on heterogeneous hardware and operating system platforms, as well as the support of a variety of communication links is a strong argument for Notes.

The basic units of information in Notes are databases, the documents they contain, and the fields within documents. A database generally contains information in a single area of interest, such as a publication or a specific section. A database either can be used by an individual, or shared among a few people, or used by everyone in the organisation.

In addition to setting up the basics of a Notes workstation, the user can customise details of his workspace to suit his working style and preferences for everyday use. Examples are name and selection of colours, international settings such as alphanumeric sorting in views and units of measurement for margins and tabs.

3.1.2. The Underlying Workflow Model

Lotus Notes in its current version 3.0 provides only fragments of a workflow model and allows thus the simple routing of documents to specified users. A set of predefined functions can be used to build more complex macros.

However, there is no support for modelling tasks and subtasks and workflow related control and tracking functions.

3.1.3. Re-Engineering the business process

Lotus Notes does not support the business re-engineering process.

3.1.4. Database Functionality

Lotus Notes provides different kinds of databases stored locally or spread over a network of single servers. A local database resides on a Notes workstation. Local databases are usually personal databases, such as daily diaries or prototypes of new databases that are not ready to be shared.

A shared database resides on one or more Notes servers, accessible by many users. Databases can be copied to additional servers for easier access to many users using a replication mechanism. With replication, changes to each database replica are distributed to all the others periodically.

Notes manages the concurrent access of several users to one document by use of a simple transaction control mechanism based on locking. Regarding the general limitations, a Notes server is able to cope with about 100 users at a time.

Notes protects information in a variety of ways. Users are granted or denied access to Notes servers through the certificates stored in their user ids. Each Notes database contains an access control list detailing who may open the database, and what operations are allowed on that information. Through the use of security mechanisms such as server access and database access, database managers can define who may use a database and to what extent.

3.1.5. Compound Structured Documents

Notes provides no support of compound structured documents.

3.1.6. Communication Facilities

Notes servers and workstations can be on a single local area network (LAN), on a number of LANs, or on a wide-area network (WAN). Notes servers and workstations on different LANs can communicate through many media, including network bridges/routers, modem and telephone lines, or satellite. Notes servers and workstations are both simply nodes on the network.

3.1.7. Integration

The integration of Notes into other applications can be done by using an application programming interface (API). For sharing information with other applications Notes provides/makes use of the following mechanisms:

- the operating system's Clipboard
- the Notes *File - Import* command
- object linking or embedding using DDE or OLE on PCs, Subscribe on the Macintosh
- Notes file attachments

3.1.8. Development Environment

Lotus Notes comes with a complete development environment which allows the graphical and interactive design of elements of the active (or selected) database: forms, fields, views, macros, and icons. Modifying the design of a database requires at least access to the Designer, except when designing private views and forms.

3.2. Visual Workflo

3.2.1. Basics

Visual WorkFlo [6, 20] from FileNet Corporation is a toolset for developing and managing workflow applications. It consists of the three components Visual WorkFlo/SDK (System Development Kit), Visual WorkFlo/Performer and Visual WorkFlo/Conductor. The whole system is designed for the Windows world. FileNet's development spectrum covers application libraries, which can be invoked via C-function calls or linked to an application as DLL, an English-like programming language including compiler and debugger, as well as tools supporting the graphical building of applications from a library of defined job functions.

3.2.2. The Underlying Workflow Model

Visual WorkFlo provides no basic model for the specification of workflows as is the case for Action Workflow (see below). However, due to its flexibility in terms of programming possibilities Visual WorkFlo allows the modelling of tasks and subtasks, the querying of a task's status, it allows the specification of roles and supports automatic routing.

3.2.3. Re-Engineering the business process

Though providing a design tool WorkFlo does not directly support re-engineering.

3.2.4. Database Functionality

Visual WorkFlo is primarily a Windows client application. Database access is possible in two ways. These are called the On-line and the Authoring Repository, respectively. The On-line Repository checks workflow objects out of the library as the object becomes activated. While the object is active, the object itself and its status information is stored in the On-line Repository. The Authoring Repository is a library of class definitions, WorkOrders, WorkPerformers, etc. providing means for handling multiple versions of workflow objects. Access rights can be specified as attributes of objects.

3.2.5. Compound Structured Documents

FileNet does not support structured documents.

3.2.6. Integration

Visual WorkFlo provides an API which allows the integration of custom applications such as e.g. Visual Basic, a C program or also a terminal emulation script. Object-oriented technology with features like encapsulation, abstraction and inheritance eases the integration of existing as well as new applications.

3.2.7. Development Environment

Visual WorkFlo/SDK allows to graphically build applications from a library of defined job functions (the Authoring Repository). Visual WorkFlo/Conductor for Windows is a systems administration package for managing and modifying the workflow processes and creating

management reports on the status of work. The Conductor has a simulation capability in order to determine bottlenecks, performance issues and the like.

3.3. Action Workflow

3.3.1. Basics

The whole Action workflow system [1, 13] consists of Analyst, WorkflowManager and ApplicationBuilder. The Action WorkflowManager requires OS/2 or Microsoft Windows NT. There exist two different versions: one is based on Lotus Notes, the other is tightly integrated with SQL servers, especially the Microsoft Sybase SQL server*. The former provides functionality like mailing, directory services, replication etc., the latter has been designed for applications dealing with high volume data.

3.3.2. The Underlying Workflow Model

The Action system provides an advanced workflow model (*see Figure 1*). A workflow consists of atomic loops of actions in which a performer completes an action to the satisfaction of his customer. The customer can be external, e.g. a client, or he can be the successor in the workflow loop. Every business process can be mapped onto this model. Splitting up of tasks is done by adding loops to the initial workflow model.

Action's workflow model applies the concept of roles, i.e. tasks can rather be assigned to roles of users than to users themselves. A user can query the status of tasks, he can be given an overview of his task's position in the whole workflow, etc.

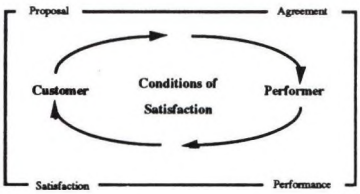


Fig. 1: ActionWorkflow Model [13]

* Other SQL servers should be possible too.

3.3.3. Re-Engineering the business process

The Action Analyst can be used to draw, document and print business processes. The Analyst exposes process inefficiencies using the Action specific workflow model.

3.3.4. Database Functionality

Choosing the SQL server as basis for the WorkflowManager, the database functionality is that provided by the underlying database system, i.e. transaction, distribution and security mechanisms are supported. With Lotus Notes as basis the database functionality is that provided by Notes (see above).

3.3.5. Compound Structured Documents

Compound structured documents are not supported by the Action workflow system.

3.3.6. Communication Facilities

When taking Lotus Notes as basis for the WorkflowManager, replication, electronic mail, security and directory services are supported.

3.3.7. Integration

An API is provided which allows the integration of custom applications as well as standard applications. Action workflow requires applications to be 'workflow-enabled' in order to allow their integration. The term workflow enabled comprises different levels from slight or no modifications up to 'real' workflow-aware applications which e.g. are able to keep track of fulfilment cycle times.

3.3.8. Development Environment

Apart from the WorkflowManager a set of tools is provided, namely the Analyst, the ApplicationBuilder, and an API. The Analyst can be used to re-engineer business processes. The goal is to find activities which support customer satisfaction and to find where miscoordination is occurring. The ApplicationBuilder is a tool for defining and prototyping workflow applications. The API is used for integrating existing or custom made applications, respectively.

4. Summary and Conclusion

Table 1 gives an overview of key characteristics of the investigated workflow systems.

Generally speaking each of the investigated tools has its strengths and weaknesses. Lotus Notes for instance supports a great number of platforms, both with regard to hardware and software. It is also rather strong in terms of distribution. However, Lotus Notes seems to be applicable only to 'simple' workflow applications, on the one hand because of its limited development facilities and primitives and on the other hand because of the lack of a basic workflow model, which both are necessary for building complex applications. Nevertheless, it seems quite usable performing the functionality of the information sharing component of a workflow application.

Visual WorkFlo is designed for client server document imaging workflow applications. It is very strong with respect to modelling and programming possibilities. The language as well as the class library seem to be very useful for developing workflow applications. However, Visual WorkFlo is more a programmer's library, which has to be integrated in workflow applications, than a tool which can be used straight away.

ActionWorkflow is probably the most sophisticated and advanced workflow system. This simply is demonstrated by the fact that it *uses* Lotus Notes for information sharing. The strength of Action Workflow is its (maybe a little bit philosophical) approach for modelling workflows expressed by the Action Workflow Loop. Dyson [20, p.3] sees the main benefit of workflow tools in (1) to make it easy to do develop workflow applications and (2) to provide some methodology and validation tools to help manage the process. ActionWorkflow is clear leader in (2), (1) is well supported by all three candidates. Nevertheless, besides this general conclusion it has to be stated that none of the tools is able to cope with structured documents which really is a prerequisite for publishers.

	Action Workflow (with SQL server)	**Lotus Notes 3.0**	**Visual Workflow**
Basics			
Platforms	MS-Windows; OS/2 or Windows NT	MS-Windows, OS/2, Macintosh, UNIX	MS-Windows, OS/2
Re-Engineering	++	-	-
Database			
Distribution	++	++	+
Concurrency	++	+	+
Authorisation	+	++	++
Compound Structured Documents			
Text Files	++	++	++
Structured Documents (SGML)	-	-	-
Communication			
topologies	Ethernet, Tokenring	Ethernet, Tokenring, Serial communication	?
protocols	TCP/IP	TCP/IP, AppleTalk, Novell XPC	?
Integration			
API	++	+	++
Development Environment			
Workflow Modelling	++	+	++
Script, Macros	-	+	++
Data Modelling	++	+	++
Programming	+	+	++

Legend: **- no support** **+ basic support** **++ support** **? not applicable**

Table 1: Key Characteristics of investigated Workflow Tools

The efficient handling of compound structured documents as is required in the area of electronic publishing needs a very close integration of tasks and structured documents. This means for example that at the time of definition of tasks already parts of documents have to assigned to the task and therefore the performing user. As the investigation has shown, none of the tested systems does offer this functionality.

5. Acknowledgements

This work has been sponsored by the Austrian Federal Research Foundation (FFF) under grant no. 2/310.

6. References

[1] Action Technologies, Inc. What is ActionWorkflow? A Primer. Alameda, California 1993.
[2] BORCHERS, D. Trends und Perspektiven auf dem Groupware-Markt. In: c´t - magazin für computertechnik. Heft 7 1993, S.100 - 106. Heinz Heise GmbH & Co KG. Hannover 1993.
[3] BROWN, H. Standards for Structured Documents; In: The Computer Journal, Vol. 32, No. 6, 1989.
[4] Commission of the European Communities: New Opportunities for Publishers in the Information Services Market. Executive Summary. Directorate General XIII, Information Technologies and Industrie, and Telecommunications. Report EUR 14925 EN. Brussels January 1993.
[5] Communication and Acccess to Information for Persons with Special Needs(CAPS): ODA State of the Art Report. WP4. Acitivity 4.1.1 InfoVisie. Leuven, Belgium, February 1993.
[6] FileNet: Visual WorkFlo: Orchestrating the Business Process. Filenet Corporation, Costa Mesa, California 1993.
[7] GOLDFARB, Charles F. The SGML Handbook. Clarendon Press. Oxford-New York 1990.
[8] GRIESE, J. Der Beitrag von ISO 9000 zur Software-Qualitätssicherung. In: Wirtschaftsinformatik, 35 (1993), S. 575 - 585. 1993.
[9] Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML), ISO 8879, International Organization for Standardization 1986.
[10] Information Technology - Hypermedia/Time-based Structuring Language (HyTime), ISO/IEC 10744, International Organization for Standardization 1992.
[11] Lotus Notes 3.0: Administrator's Guide. Lotus Development Corporation, Cambridge 1993.
[12] MARSHAK, Ronni T. Requirements for Workflow Products. In: Proceedings of Groupware 92. Edited by David D. Coleman. Morgan Kaufman Publishers. San Mateo, California 92.
[13] MEDINA-MORA, R., WINOGRAD, T., FLORES, R., FLORES, F. The Action Workflow Approach to Workflow Management Technology. In: CSCW 92 Proceedings, Toronto, Canada. ACM 1992.
[14] MÖHR, W. et al. Specification of the Common Publishing Tools. EUROPUBLISHING Project(R2042), Technical Report ID R2042/GMD/IPS/DS/R/011/b1, RACE-Programme June 1993.
[15] NASTANSKY, L. Workgroup Computing: computergestützte Teamarbeit (CSCW) in der Praxis; neue Entwicklungen und Trends / hrsg. von Ludwig Nastansky. S + W Steuer- und Wirtschaftsverlag. Hamburg 1993.
[16] PALLOT, M. Enabled interactions using metadata within a CE environment. Proc. 2nd International Conference on Integrated Logistics & Concurrent Engineering, Montpellier, 1994.
[17] Patricia Seybold's Workgroup Computing Report. Vol. 16, No. 5. Boston 1993.
[18] Proceedings of Dokument '93, 10. - 11. März 1993. Austrian Center Vienna. Arbeitsgemeinschaft für Datenverarbeitung (ADV). Wien 1993.
[19] RELEASE 1.0: A Monthly Report by Esther Dyson. August 1992. EDventure Holdings. New York 1992.
[20] RELEASE 1.0: A Monthly Report by Esther Dyson. September 1992. EDventure Holdings. New York 1992.
[21] Seybold Publications Division: Quark's QPS. Seybold Report on Publishing Systems. Vol. 23, No. 2, September 1993.
[22] Seybold Publications Division: Managing the Publishing Process. What You Need to Build a System. The Seybold Report on Desktop Publishing. Vol. 6, No. 11, July 1992.

IBM FlowMark workflow manager

Concept and Overview

Roland Lutz[1]

Abstract
The business processes used in today's enterprises are becoming increasingly complex. It is a challenging prospect to plan, coordinate and manage the the activities and resources required processes, and to track the workflow.

IBM FlowMark for OS/2 is a client/server product that contains a set of workflow management functions which help you to document the business process in your enterprise precisely and consistently, to control the business processes and to improve them progressively.

1.0 Concept

Design and implementation of the FlowMark workflow manager aim at the needs of the market for a workflow manager that allows to control the flow of work in a distributed environment, independently of the contents of single activities.

By using a "plug - socket" technique, activities can be connected with programs or subsystems in a way that enables, at run time, an information exchange between control level and execution level. In addition, activities can be assigned to people, roles and departments.

Thus, the FlowMark workflow manager supports that:

- the right person performs at
- the right time
- the right activity using
- the right application software.

Technologically, the FlowMark workflow manager uses the client/server architecture, object-oriented design, and knowledge-based systems, but also allows to incorporate traditional application software.

Basically, in the FlowMark workflow manager the workflow is defined in models. These models are the unique basis for the documentation, animation and execution of business processes.

People and programs are used in an enterprise in more than one place; for that reason, in the FlowMark workflow manager they are only registered. By flexible assignment of people and programs to activities and by using workflow logic, the flow of work can be "experienced" in an enterprise at the workstation.

---

[1] Vienna Software Development Laboratory, Lassallestr 1, A-1020 Vienna, Austria

2.0 Overview

The FlowMark workflow manager clearly distinguishes between process definition and process execution.

3.0 Process Definition

In process definition, the process flow is graphically modelled, the associated "logistics" (people, programs, data structures) are registered and assigned to activities.

3.1 Process modelling

For process modelling, three basic constructs are used:

- Activity - Connector - Condition

An activity identifies one step in a process, a group of steps (block), or a reference to another process (subprocess). Each activity can have data containers whose values are used to control the workflow, or can be passed on to subsequent activities.

There are two kinds of connectors: control connectors and data connectors. Control connectors determine the possible sequence of activities. Data connectors determine the flow of data if data is to be moved from the output container of one activity to the input container of a target activity. An activity can have more than one incoming and outgoing connectors. Modelling of parallel activities is also possible.

Conditions can be assigned to connectors as well as to activities.

Conditions of connectors influence the path of the workflow. Depending on the result of a condition (for example, amount_of_credit > 50000), the path is open or closed to further processing.

Conditions of activities act as entry and exit control. Entry conditions are used for synchronization, for example can be specified whether to wait for a second signature. The exit condition stops the navigation after an activity, until, for example, a status specified in the condition is reached.

3.2 Animation

By using knowledge-based rules (Prolog2), the behavior of process models can be tested, optimized, and educated. If desired, the model can be tested repeatedly (animation) at build time. Errors in the workflow or in people assignments can thus be recognized by the modeler without the run-time environment and at a time when the programs to be attached are not yet available.

3.3 Graphic, language, documentation

To define the logic, a graphics editor or the supplied language (FlowMark definition language) can be used.

The graphics editor provides symbols for the modeling constructs and converts them directly into executable objects. The drawn model is executable and is always THE source from which to derive:

- Documentation (BookMaster or online Hypertext)
- Facts and rules for animation
- The executable models themselves

Graphically supported functions for re-grouping of activities are available to ease the tasks of process re-engineering.

3.4 Import / Export

By using the definition language, processes can be imported and exported. In language form, they can be easily stored in libraries and archived. The graphic form is, if not available, generated as a proposal.

3.5 Registering programs and people

The input and output data structures of programs can either be collected through FlowMark dialogs or be imported by using the format of the FlowMark definition language. The same is true for people.

Programs are registered by logical names, and call the desired target programs from OS/2 (for example, CICS host transactions from CICS/OS2).

People can be authorized and assigned to organizations or roles.

Assignment of programs and people to activities is done through dialogs or graphically (drag and drop). When assigning programs to activities, they are connected to the activities with the "standard connector." At run time, this enables the information exchange between data of the application program and the container of the activity. (Container APIs / Application Program Interface.)

4.0 Process Execution

FlowMark definitions and assignments are direct sources for the run-time component. Each tested model is provided specifically for execution and is from then on available as a basis for relevant business processes.

4.1 Navigation, work distribution, work lists

The navigator identifies the next activity or activities depending on connectors and conditions by using container data.

By people assignment, the work is distributed to the people involved, and arranged in work lists. Each entry in the work list represents an activity in a current business process that is ready to be performed. When starting, the program assigned during modeling (for example, a CICS or IMS transaction, Image or Office components, an OS/2 or Windows program) is called automatically.

4.2 Work control

Definite business processes can be started, interrupted and restarted, and stopped. Work can be distributed over several work lists and transferred to authorized people. Activities can be started and stopped manually or automatically. If desired, the workflow can be recorded.

Technique

The FlowMark workflow manager is a client/server solution under OS/2. Clients for AIX and Windows are planned. The communication between server and clients is by TelePath, a common distribution layer that can support APPC and other protocols. The FlowMark workflow manager uses object-oriented design in programming and data management. Programming language is C++, the object-oriented database is ObjectStore. Prolog/2 is used for animation. The "standard connector," that is, the container APIs, are available for C, C++, and REXX. Further APIs for controlling running processes, (for example, to start an action externally) are available. The FlowMark definition language is documented, and can, therefore, be used by customers.

5.0 Usage areas

The FlowMark workflow manager can be purposefully used in an enterprise, independently of area of business and size, for everybody who i.e. needs or wants to:

- Document business processes (ISO9000)
- Know and improve business processes
- Control the enterprise by business events
- Standardize business events
- Automate business events
- Standardize application software
- Integrate application building blocks
- Ease the load on application development
- Let the expert departments model processes
- Seek access to a new technology

Cooperation in Europe

Experiences in participating in European Information Technology Projects

Benczur A., Dömölki B.

Paper not received in time.

Cooperation — A European View

A. Gotwald

Paper not received in time.

Modelling Business Processes

Strategic Information System Planning as a Prerequisite for Business Restructuring

Josef Lindermayr [1]
Vamed Engineering, Vienna

Table of Contents

---

[1]VAMED ENGINEERING Krankenhaus Planungs-, Errichtungs- und Betriebsführungs GmbH, Postfach 58, A-1104 Wien, Austria

1. INTRODUCTORY REMARKS TO ORGANIZATIONAL CHANGE

Organizational changes are possibly as old as the formation of organizations itselves. The famous "panta rhei" of the Greek philosopher Heraklit is as valid for social and business organizations of any kind as it is for everyday life.

In business organizations, where frequently many people have to cooperate, the question of organizational change arises, whenever

- the agreement on a set of longterm goals to be achieved, the allocation and distribution of resources and the arrangements of appropriate global measures required to reach these goals (equalling the determination of global business strategy), and / or

- the division of tasks to be fullfilled, the competence and responsibilities of the different business units and subsequently of the members of the organization, distributed preferably according to their traits (equalling the determination of business structure), and / or

- the main methods and procedures applied and processes carried out in the daily business acitivities in order to reach the agreed objectives (equalling the determination of business processes),

are for some reasons **subject to change** caused by some events or developments within or outside the company.

The triggered organizational change can range from minor adjustments of the above-mentioned areas to complete business restructuring or reorganization programs, whereby the former shape of the company will be completely changed.

In the last decades the main focus has been laid on structural changes of business organizations:

Basically strategic considerations (Chandler, 1962: Structure follows Strategy), yet not neglecting other empirical results of the analyses of the causes of organisational change (Rumelt, 1974: Strategy follows Structure; Structure follows Fashion), led to the rethinking of the existing traditional organizational structures (for reference see [1]).

New organizational structures were introduced frequently and subsequently, beginning at the late Nineteensixties:

o divisional organizational structure,

o matrix organisation,

o formation of profit centers and of strategic business units, and

o introduction of cross-section functions (for example: Logistics, Personnel, etc.).

Primarily new structural arrangements have been the main focus of these organizational endeavours, neglecting the vital importance of the optimal formation of complete business processes. Business process changes induced by the new organzational structures have been dealt with more subsequently after the introduction of the new organizational structure, leaving it more or less at the responsibility of the new department head(s).

This departmentalization induced through the new business structure(s) led to fragmented or so-called "broken" business processes, where each of the departments involved is in charge for only a small part of a process and whereby the overall responsibility for a complete business-process got lost in most of the cases.

First in the last few years the vital unit-overlapping business processes have been put in the middle-point of business restructuring considerations, leading to new business restructuring paradigm like

o business process reengineering,

o lean management, lean production, etc.

o fractal organizations, etc. (for reference see [1] and [3])

The vital questions involved in these process-oriented approaches are basically

o which are our business processes,

o which present business processes can be eliminated comletely, for example through outsourcing, highly integrated information systems, etc.,

o how is it possible to optimize the remaining vital business processes, basically from the costumer's point of view.

Optimization in this context refers to all interrelated aspects of any business process:

o optimization of the flow of materials and inputs, work in progress and products,

o optimization of the transformation process of materials and inputs into products and services,

o optimization of the corresponding flow of data and of information supply (access to generalized and specialized information).

In a business restructuring program, all these aspects have to be considered simultaneously in order to optimize any given business process.

Computer-supported business restructuring enables especially the optimization of the flow of data and of information supply through the implementation of decentralized and highly integrated information systems.

Information system integration in this context comprises

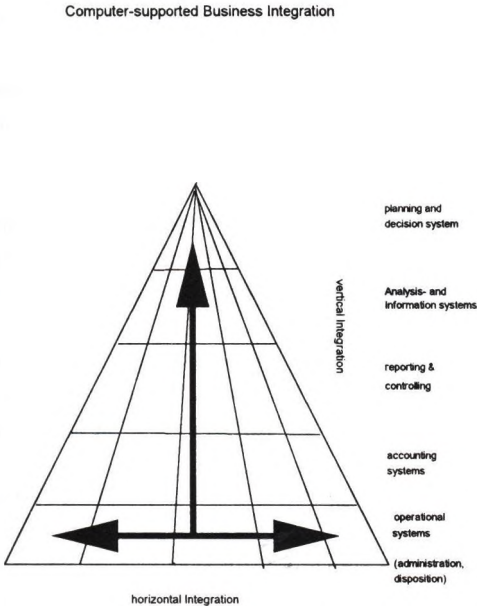
- **data integration** by providing highly integrated (relational) databases, where data entered once into the system are available for all other legal users connected to this system. Logical data integration is a prerequisite for the ad-hoc generation of flexible management information for vital decision making within the system without major additional efforts.

- **workflow integration** by providing a smoothly flow of data and information without any interruptions and unnecessary manipulations through the system; no data captured already has to be reentered again. Workflow integration, both inside and outside of an organization, is the prerequisite for lean administrative business processes, as unnecessary steps and manual activities are avoided. Moreover, besides the economical use of resources, highly integrated administrative business processes are much faster, more safer and reliable.

Work Flow Management Systems are especially aiming at supporting this process integration in the more administrative functions of a company.

The following **Figure 1** shows a schematic view of the integration directions to be achieved by computer-supported business restructuring.

Figure 1: Computer-supported Business Integration



Decentralized and highly integrated information systems enabling computer-supported business restructuring are made available mainly through the following new information technologies and developments:

o the movement for standardization and open systems (ISO/OSI - X.400, X.500; POSIX, XPG/4; SQL2, EPHOS, etc.)

o high-speed networks for multimedia communication (LAN; MAN; WAN)

o client-/server architecture (with the PC/workstation + GUI as front-end computer) and their further development to distributed cooperative systems (ODP)

o relational databases with standardized query languages (SQL2; PC-based Tools, for instance ODBC-interface).

o multimedia technologies - integration of data, text, picture, audio, video

o object-oriented techniques (OOA; OOD; OOP; OODBMS, etc.)

o appropriate application software (for example highly integrated and flexible standard business application software covering almost all vital areas of a company; Work Flow Management Systems, etc.).

The above mentioned information technologies allow the flexible and scalable implementation of **decentralized and highly integrated information systems** supporting the information and data processing needs of almost every area of a business organization.

Computer-supported business restructuring consequently enables not only to make the corporate information system more cost effective but **to make the organization as a whole more effective** (= to do the right things) **and more efficient** (= to do the things right).

The new information technologies offer an opportunity to make a real change, but only if the appropriate technology is applied correctly within the framework of a company-wide strategic restructuring program, as it should be provided by a Strategic Information System Plan.

2. WHY STRATEGIC INFORMATION SYSTEM PLANNING AS PREREQUISITE FOR BUSINESS RESTRUCTURING

As for any reorganization, which results in major changes of the structure and/or processes of any company, the question of an appropriate **management of the planned organizational change** arises in order to implement successfully the intended changes.

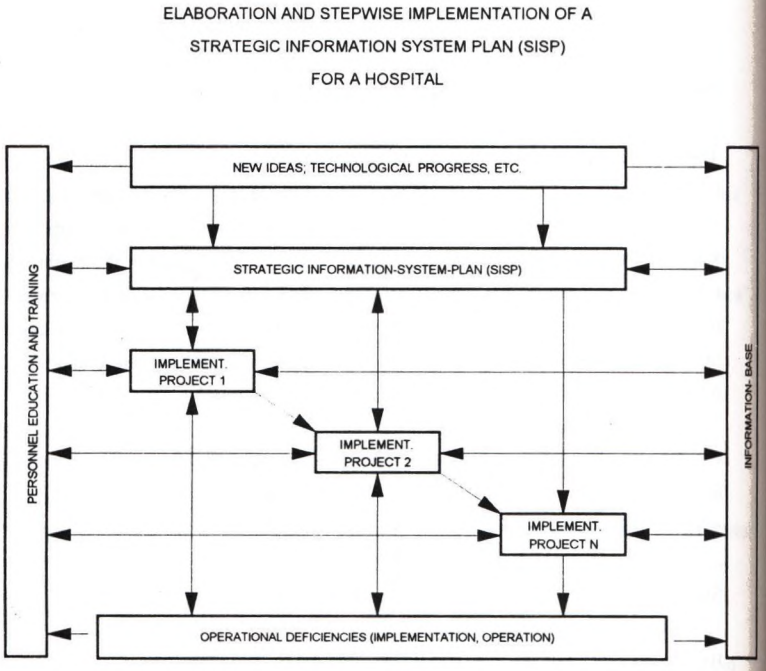
There is clear empirical evidence, that a substantial improvement of business performances through organizational change can hardly be achieved through a sequence of uncoordinated small organizational projects. Besides lacking a strategic view of the reorganization direction, such efforts are highly influenced by the daily business activities of the department(s) involved and lead in general to unsatisfactory results. The implementation of decentralized and highly integrated information systems - not only from the technical but especially from the semantic point of view - cannot be guaranteed by this incremental departmental changes, commonly denoted as "muddling through" or "piecemeal engineering".

On the other hand for many practical reasons, it is not possible, to conceive a comprehensive business restructuring program and to implement it as a whole in a short period of time at once when the decision to go ahead has been made.

Strategic Information System Planning tries to avoid this dilemma:

By preparing a comprehensive Strategic Information System Plan (SISP) the strategic direction for the computer-supported business restructuring program should be officially laid down as a guideline for the subsequent implementation in manageable smaller projects, as depicted in the following **Figure 2**.

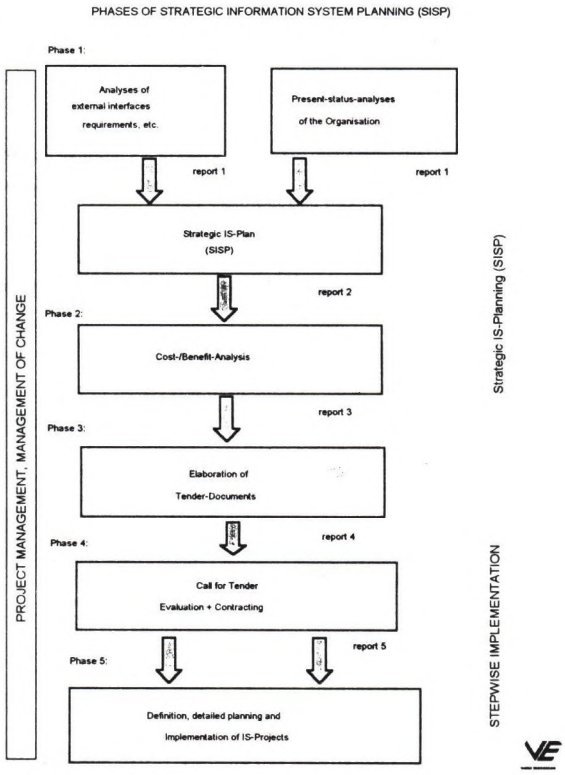
Figure 2: Elaboration and stepwise Implementation of a Strategic Information System Plan



The resulting limited number of (evolutionary) organizational implementation projects can finally lead to a revolutionary result: a completely restructured business organization offering considerable improvements in business performance.

The following **Figure 3** shows the typical sequence of steps of how to proceed in preparing and stepwise implementing a Strategic Information System Plan.

Figure 3: Phases of Strategic Information System Planning



o Preparation of the Strategic Information System Plan (SISP)

A comprehensive, holistic view is very important during the preparation of the SISP, taking into account:

- all present/future business areas and their respective business processes, both inside and outside of the organization,

- consideration of economical, technical, juridicial, social, etc. aspects,

- a time horizon of at least 5 to 10 years covering the lifespan of vital components of an integrated information system (for instance: high-speed communication networks, integrated application software, etc.),

- integrated, participative system design with the main focus on data and process (workflow) integration.

Important strategic decisions, that have serious long-term effects on the organization, have to be unanimously agreed upon by all participating parties, including among others

- set of goals for longterm information strategy,

- decisions upon technological direction (concerning network, hardware, software) aiming for decentralized and integrated information systems,

- determinations of standards, terminology, catalogues, etc. to be followed by all business units involved,

- decisions on resources allocated annually for the business restructuring program.

In preparing a Strategic Information System Plan (SISP), these vital decisions about decentralzied and highly integrated information systems have to be matched with the global strategies of the organization as a whole - making the SISP to a Strategic Reorganisation Plan and Business Restructuring Program, respectively.

In this sense, information system strategy flows directly from the organizations business strategy focussing on the business processes involved in order to serve the customer's heeds.

o Phased Implementation Plan

As **Figure 1** demonstrates, large-scale failures are unlikely in a true phased approach, because the general process of a phased rollout invites the evaluation of the current step before taking the next one.

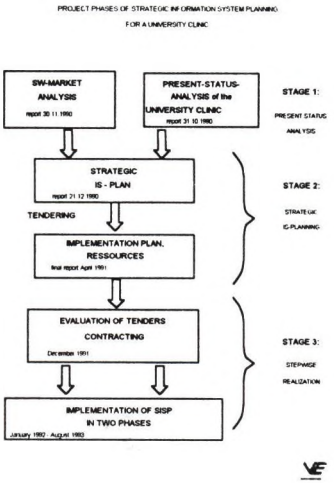
Timing of implementation projects is critical, as a slightly tighter time frame keeps the organization more focused.

3. A PRACTICAL EXAMPLE: STRATEGIC INFORMATION SYSTEM PLANNING FOR A UNIVERSITY CLINIC

The considerations of chapter 2 can be fostered by the practical example of the elaboration and implementation of a Strategic Information System Plan for a major University Clinic.

The individual project phases are shown in the following Figure 4.

Figure 4: Project Phases of a SISP for a University Clinic



The term "Strategic Information System Planning" in this project was really a euphemistic denotation for a complete computer-supported reorganization program for the whole University Clinic.

Major changes have been made in the following ares of the clinic among others:

- patient administration

- keeping of patient record

- patient accounting

- procurement of materials and other inputs

- financial and cost accounting.

Figures 5 to 8 summarize synoptically the main processes in these areas before and after the implementation of SISP.

Figure 5: Schematic View of Patient Administration before SISP-Implementation

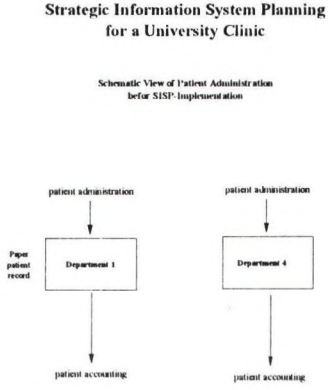


Figure 6: Schematic View of Patient Administration after SISP-Implementation

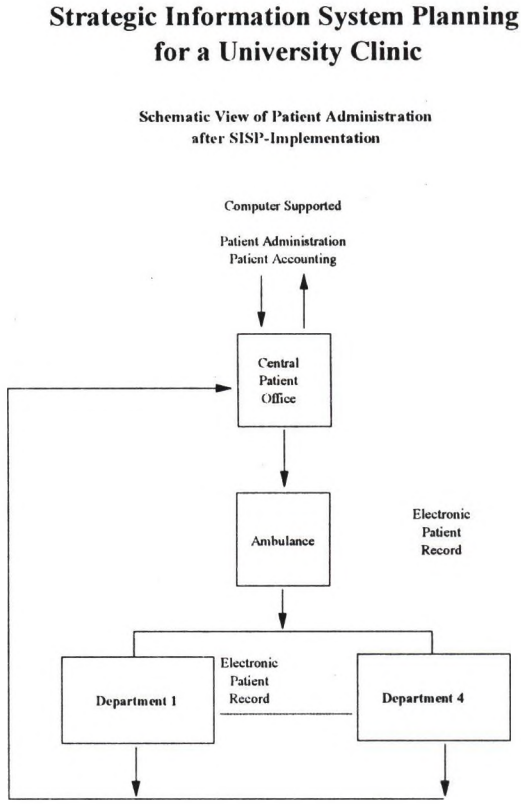


Figure 7: Schematic View of Procurement before SISP-Implementation

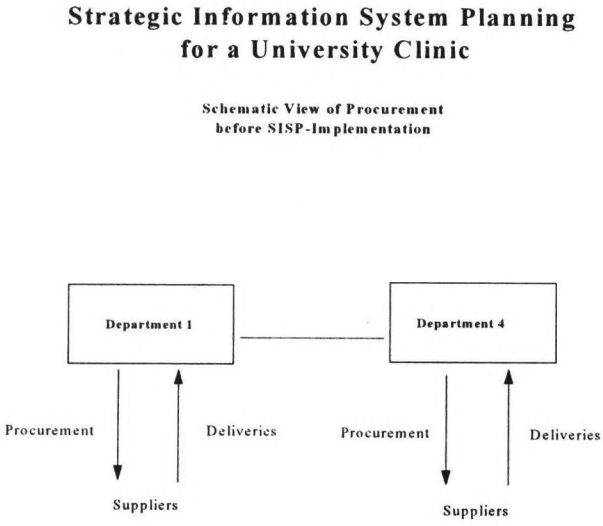
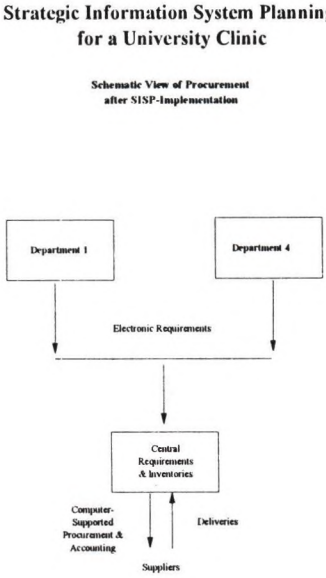


Figure 8: Schematic View of Procurement after SISP-Implementation



The implementation of a decentralized and yet highly integrated information system supporting highly integrated processes as well has been made available basically by means of

o a clinic-wide structured high-speed LAN

o appropriate highly integrated application software, based on an integrated patient database.

Figures 9 to 10 present a schematic overview about the whole system.

Figure 9: Schematic Overview on Structured System Architecture

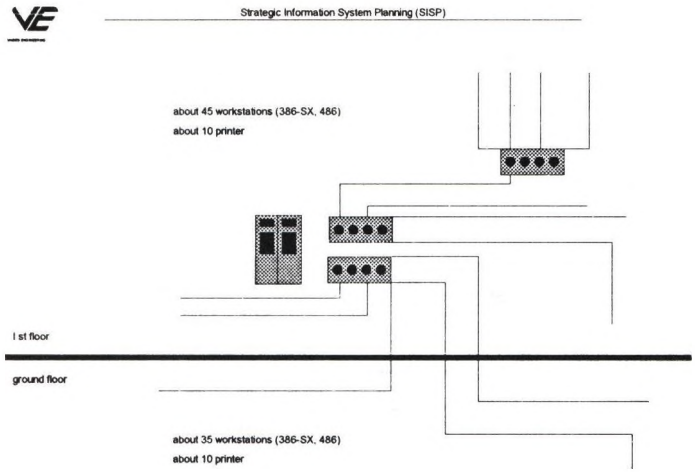


Figure 10: Schematic Overview on Application Software

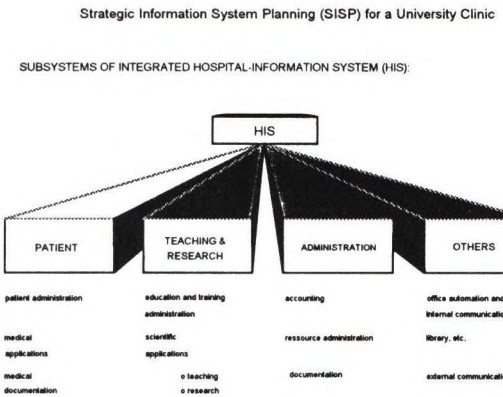
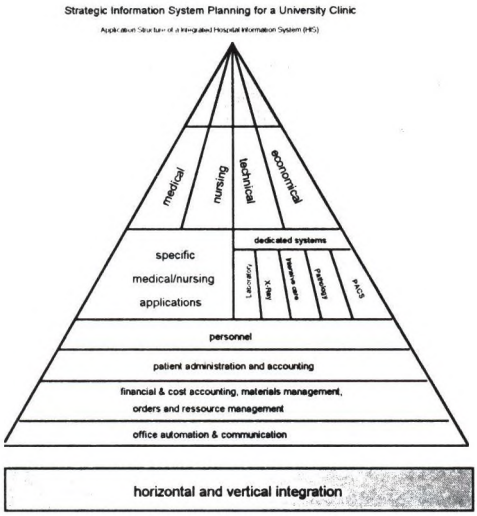


Figure 11: Schematic Overview on Application Software



The elaboration of a Strategic Information System Plan made it possible to implement a decentralized and yet highly integrated information system within the clinic, supporting the vital clinic processes starting from patient admission to patient accounting, medical research work, etc.

This approach prevented the implementation of fragmented departmental systems, which was - due to the highly decentralized structure of a University clinic - already foreseeable. These isolated systems would have basically supported the existing processes in the departments, yet not led to a comprehensive process restructuring.

4. CONCLUSION

Business strategies have traditionally focused on products, customers, markets, and costs, leading to information strategies - which must flow directly from the organizations business strategy - which resulted in fragmented, inflexible information islands.

As the building blocks of corporate strategy are nowadays not products and markets but business processes, any information strategy has to focus on the vital business processes as well.

Available new information technologies, especially high-speed networks, open systems and standardization, client-server architecture, relational databases, etc., allow the implementation of decentralized yet highly integrated information systems, supporting business process restructuring which leads to substantial improvements of company performance.

As an first step for launching a succesful restructuring program, however, a longterm Strategic Information System Plan (eqalling more or less a Strategic Reorganisation Plan) has to be prepared in order to manage the planned organizational change in an appropriate manner.

Setting the strategic direction of the change and deciding on vital information systems components, a stepwise implementation in smaller manageable projects will be made possible without sacrifying the intended organizational change while reducing overall risks involved in such a major corporate reorganization.

Based on practical experiences the following guidlines can be stipulated for any major business restructuring program:

o think big during the planning phase (=SISP) - start smaller in implementation,

o integrated, participative system design aiming at decentralized yet integrated information systems,

o look at business processes from an customer's point of view before business restructuring,

o select application software supporting the new integrated and lean business processes before hardware solutions,

o apply standards as much as possible enabling the implementation of open and scalable systems.

Bibliography

[1] Michael Hammer and James Champy: Reengineering The Corporation - A Manifesto for Business Revolution, Nicholas Brealey Publishing, London, 1993

[2] Werner Kirsch, Werner-Michael Esser, Eduard Gabele: Das Management des geplanten Wandels von Organisationen, C. E. Poeschel Verlag, Stuttgart, 1979

[3] Hans-Jürgen Warnecke: Revolution der Unternehmenskultur - Das Fraktale Unternehmen, Springer Verlag, 1993

Appendix

EPHOS	European Procurement Handbook Version 2 (1994)
GUI	Graphical User Interface
ISO / OSI	International Standardization Organization / Open Systems Interconnection
LAN	Local Area Network
MAN	Metropolitan Area Network
ODBC	Microsoft Open Database Connectivity
OOA	Object Oriented Analysis
OOD	Object Oriented Design
OODBMS	Object Oriented Database Management Systems
OOP	Object Oriented Programming
ODP	Open Distributed Processing - ISO Reference Model
POSIX	Portable Operating System for Computer Systems
SISP	Strategic Information System Plan(ning)
SQL 2	Standard Query Language, Version 2 (1993)
WAN	Wide Area Network
XPG/4	X/Open Portability Guide, Version 4 (1993)

BUSINESS MODEL REENGINEERING WITH THE AVALON SOLUTION

Janos Ladonyi

Being in business we constantly need to consider the means and tools, by which we can stay or become competitive. In today's always changing economical and business environment, totally new approaches are needed to survive.

As it comes clearly from a Gartner Group report [1], business process reengineering gives the only chance to achieve this goal. This is not an easy road to follow, and success rate is low. But no other choice is given for those, who do not want to stay behind.

There are four aspects of business process reengineering, and only dealing with all of them in a holistic way, may promise the innovative change.

The four aspects are:
- Business Model,
- Organisational Model,
- Information Technology Model,
- Manufacturing Software Model.

In other words it means: Process, People, Technology and Application.

1. Business Model

The first and most important task is how the business process works today, and then to search and determine how we want to improve or change it for competitive advantage.

There are three common challenges for manufacturing companies:
- globalization: as the whole world market shrinks, companies will work in networked systems, and the competitive weapon is *real-time responsiveness*,
- product and process innovation forces companies to develop new products and new processes *faster* than before,
- environmental awareness and safety regulations become more and more key challenge to respond appropriately.

Here we see, that *time* is a key factor and is one of the most valuable attribute to deal with. For achieving this, tomorrow's manufacturing business environment will move from sequential/functional one to a process oriented/parallel and defuctionalized one.

2. Organisational Model

Today's manufacturing organisation is a composite of three organisational models. The primary model is hierarchical, the secondary is matrix and the third is the team.

The hierarchical organisation comes from the military and is functional oriented. Its strength is in the structure and that it is predictable. Predictability is also its weakness, and it is too rigid. Directions come from the top and typically the wrong people are to make decisions.

The matrix organisation comes from marketing and is function and product oriented. It is multifaceted and cross-fertilising, but also confusing and unpredictable. Typically no one is really empowered to make decisions.

The team model comes from manufacturing and typically the correct people are enabled to make the decisions. Teams are process oriented, efficient and empowering, but hard to control and implement.

The present organisational structures are merely mechanical and in the future they need to be rather humanised. The future organisational model will be inverted: team will be dominant and hierarchy will be used for direction and empowerment. In this scenario the role of the team will be to define opportunities and problems, then to solve the problems. The role of hierarchy will be to select, empower and direct the teams.

3. Information Technology Model

The fact is that information technology and the application models have not been developed so fast as the manufacturing business models have. This means, that applications are static compared to the faster changing dynamic business world. How to eliminate that gap between the two? Only the right information technology can help. We need dynamically engineered applications!
Now at the first time of IT history, all the elements - like hardware, operating systems, networking, databases, development tools, user interfaces and applications - have reached the quality, integration and performance, that makes it possible to create the really integrated and on-line networked systems. Now it becomes possible to implement and use real client/server architecture, which means distributed application logic and distributed database. Later this will be rather a publish/subscribe architecture, where everybody can be a server or a client connected to a common information bus.

The real benefit and result of this can be seen on the application level, where finally each user - no matter if he is sitting in the financial department, in the quality assurance, in shop floor, in procurement or marketing, or he is a CEO or the president -, will have his or her *personal view* of the company, having on-line the information and control, he/she needs on that very place.
How to measure the quality of the vendor's information technology and how to differentiate the offerings? Evaluating the five layers of IT, like hardware, operating system, network, database, development and user interface, we should say, that decision on the first three: hardware, operating system and networking is not strategic. In database, development and user interface layers the confusion today is much more significant and therefore the decision is more strategic.

Database analysis indicate, that the right decision is to have applications with native and multi RDBMS (Relational Database Management System) support. The right development tools are 4GL (Fourth Generation Language) and CASE (Computer Aided System Engineering), trending towards object oriented technology, and the user interface trends to be graphical and window based.

4. Manufacturing Software Model

The focus of CIM (Computer Integrated Manufacturing) will shift to integration over the next years. This will permit manufacturing companies to dynamically engineer the applications to changing business models, while at the same time create alternative personal views throughout the enterprise.

This is what the dynamically growing AVALON Inc. provides for the global market and IQSOFT, a flexible system house and marketing company brings to Hungary. The solution is based on the CIIM (Computer Interactive Integrated Manufacturing) software and comprises all the elements of total technology transfer connecting CIIM and the customer. CIIM, the most flexible client/server software on the market and the customers, each with unique needs and ways of doing business. AVALON Software based in Tucson, Arizona, is a leading provider of manufacturing and distribution solutions for enterprise resource planning (ERP). The company's flagship product, Avalon CIIM, is a suite of fully integrated, client/server software applications which automate and integrate manufacturing, distribution and financials. It is the only manufacturing solution developed natively for the Oracle and Sybase relational database management systems, uses computer aided software engineering (CASE) tools, and supports a wide range of harrware and operating platforms.

Avalon,s technology transfer encompasses the expertise of manufacturing, project management, business analysis, system engineering, software programming and education.The core activity of the technology transfer is the Reengineering Workshop, that identifies the ways to improve business processes and add value to the AVALON CIIM system. Four tightly integrated programs help that process: AVALON

Implementation Methodology (AIM), AVALON University, Customer Support and Certified AVALON Implementor Program (CAIP).

This is the way how the AVALON solution fulfils every aspects of business process reengineering for the future: through the technology transfer process, it will be ensured, that the right business and organisational models will be created, the CIIM solution provides customers with most powerful technology basis and state of the art functionality, but first of all an always easily, fast and reasonably adaptable solution to the constantly changing business environment.

[1] GARTNER GROUP, Presentation, IAUG Forum, Tucson AZ, U.S.A. 1994.

Enterprise Modeling Using OOA Techniques

Michael Bauer, Claudia Kohl, Heinrich C. Mayr

Institut für Informatik, Universität Klagenfurt
e-mail: {bauer,kohl,mayr}@ifi.uni-klu.ac.at

and

Johann Wassermann

Wirtschaftskammer Kärnten

Abstract

We show, by applying the Object Oriented Modeling Technique OMT to a real world example, some benefits of an object oriented approach to conceptual enterprise modeling (design). Two design strategies are investigated: The first views a business process to be the 'sum' of all those activities objects perform to achieve the goal of that process. The process, therefore, is not specified explicitly. The second strategy takes a workflow oriented view and treats a business process itself as a (for some time) persistent object that gets into different states in reaction to the activities of the (actor) objects involved. It is shown that, with some restrictions, OMT may be used for both strategies, and that following the second may help to reduce complexity because it allows for rather natural design results.

1. Introduction

An essential point of information engineering within an organization is the development of a comprehensive *information system architecture* [Ei94, Kr90, Sc92a, Tu91]. One of the basic building blocks of such an architecture is the so-called *enterprise datamodel* the aim of which is to integrate the various (sub)datamodels of the different enterprise divisions. The use of the notion 'datamodel' reflects the actual state of the art: Attention mainly is payed to the static aspects of

an enterprise; dynamic properties (business processes and their causal and temporal interdependencies, i.e. the functional and dynamic model) are covered at most in part. So it is quite obvious, that the standard meta model actually used for enterprise modeling is the well known entity relationship model (ERM) [Ch76] or one of its numerous variants.

On the other hand, *Object Oriented Analysis (OOA)* more and more becomes the modern way of conceptual design in the information system life cycle [KKM93], introducing the integrated treatment of static and dynamic aspects of a given Universe of Discourse. So it is a rather selfsuggesting idea, to use OOA concepts for enterprise modeling in general and for business process modeling in detail.

Again there is a number of more or less similar OOA methods on the market, e.g., [CY91, Ru91, Ja92, WWW90]. However, independently of the specific choice of one of them, two basically different approaches for object oriented business process modeling can be distinguished:

- Understand a business process as the 'sum' of all those activities objects perform to achieve the goal of that process, see e.g. [FS93b].

- Understand a business process to be itself a (for some time) persistent object that gets into different states in reaction to the activities of the (agent) objects involved.

At first glance the first approach seems to be the more natural one. However, the second approach allows for more concise and transparent models due to the fact, that it corresponds to a workflow oriented view where the flow objects themselves are of central interest.

This paper addresses the question to what extent a well known OOA method is appropriate for both approaches to business process modeling. In detail, first results of a project are presented, that is to investigate the suitability of two specific method representatives in a real world environment of some complexity - the Chamber of Economics of Carinthia. The methods in question are the OOA-part of the Object Modeling Technique (OMT, [Ru91]) and the Semantic Object Modeling approach (SOM, [FS93b]). Within this paper we restrict ourselves to the former.

Section 2 of the paper gives a short synopsis of how the Austrian Chambers of Economics are organized.Within section 3 we then introduce the 'miniworld' of one of the Chambers typical business processes, i.e. Bill Appraisal, which will serve us for an example throughout the paper.

Section 4 points out some important aspects of object oriented business process modeling and gives a very short survey on OMT.

Section 5 then outlines parts of a conventional object oriented conceptual design using OMT, whereas within section 6, we present some design results where the Bill Appraisal process itself was modelled to be an object on its own. The paper closes with a short outlook on further research.

2. The Austrian Chambers of Economics Organization

The Austrian Chambers of Economics form a legal and politically independent organization, that is financed by its members, i.e., the entirety of all Austrian enterprises and business makers. Their mission is ".. to act in their member's common behalfs .." [HKG94] with, in particular, tasks like

- the representation of their member's interests in front of the parliament, the government and of the other administrative authorities,
- the appraisal of bills and other important decrees, federal as well as local ones,
- the representation through experts at public or semi-public institutions, such as funds, commissions, committees, consultants (e.g. consultant for foreign affairs, price commissions, agricultural fund, social insurance companies, assessorts at labour court, conciliation boards, arbitration courts for social insurance affairs and the cartel court).

In order to comply with these duties, the Chamber Organization is structured into two 'dimensions': into ***regional units*** (Chamber of Economics in each federal country and the Austrian Chamber of Economics, Vienna) and into ***professional divisions*** ('sections' and 'professional groups'). The chambers, burdened with the major part of the administrational issues, are organized in departments, some of them distributed over regional district bureaus. The professional divisions are governed by elected honorary officials and managed by employed secretaries. For an overview see figure 2.1.

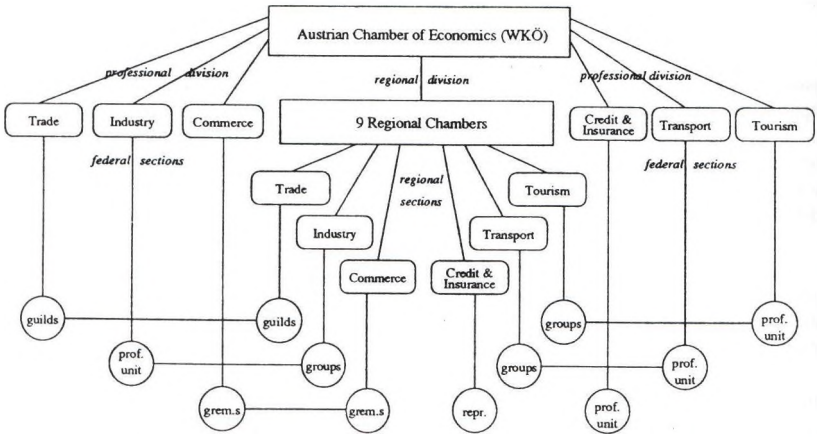


Figure 2.1: Organization of the Austrian Chambers of Economics

With respect to the professional divisions, both the Austrian Chamber of Economics (WKÖ) and the regional Chambers (WKs) consist of six socalled ***sections***, namely ***Trade, Industry, Commerce, Finance-, Credit- and Insurance Affairs, Transport and Tourism***. WKÖ comprises about 130 professional units, in general corresponding to professional groups on the regional level. Groups within the trade section are called

regional and federal guilds, respectively; regarding the commercial sections, they are called ***regional and federal gremiums***; within the regional sections of finance-, credit- and insurance affairs, they are called ***professional representatives***.

Professional groups and ***units*** are legal entities on its own. They have to pursue the interests of their particular branches, such as constructing engineers, glaziers, carpenters, electrical engineers, chemical industry, food trading, banking, forwarding agents, restaurants, etc.

The sections have to act for and to balance the interests of the various branches they comprise. On the chamber level, again, the matters common to all members have to be coordinated and divergent professional interests of subunits have to be balanced and harmonized. The WKÖ, finally, coordinates the matters concerning the entire Austrian economy.

3. The Business Process *Bill Appraisal*

Besides of regional bills and decrees no less than about 1000 federal bills p.a. have to be appraised by the Chamber organization. This leads to a substantial effort furnished by about 600-700 persons. In detail, the opinion on a given federal bill is formed on a federal basis: A large number of experts (belonging to or related with organizational units that are affected by the respective bill) throughout the organization is asked for their opinions. These are collected and brought together at the different levels of the organization. Obviously, this necessitates a procedure that provides for one and only one statement of the whole organization and does so within a given time schedule. This procedure is a democratic one and pertains to the basic concepts of the chamber system. It therefore is called ***balance of interests***.

For federal bills the main steps of that process actually are the following (see fig. 3.1):

Departmental distribution within WKÖ:
WKÖ receives the draft of a federal bill from the responsible federal ministry. It transmits that draft to its competent department (such as the department for social policy), to the federal sections and to professional units.

Federal distribution:
The competent WKÖ department then transmits the bill text together with additional information (identification key, distribution list, context of the bill, background information etc.) to the corresponding departments of all local chambers. The federal section and the professional units forward the bill to their corresponding sections and professional groups in the local chambers.

Regional distribution:
Within each WK the bill is distributed to the competent department, then to the competent section(s), subsequently to the competent professional group(s) and finally to the district departments and to the members.

Obviously, distribution must not always be effectuated throughout all these levels (and in fact is not) but may be interrupted at any level. On the other hand, at any of these distribution levels additional information may be enclosed.

Formation of opinion:
Any unit of the organization is free to consult (external or member) experts in the course of its opinion forming process. The formation of the common opinion starts at the final level of distribution. In the case of district department and member level, the opinions are collected either in written or in oral form and are put

together into an appraisal by the manager of the respective professional group. This appraisal, which must always be written down, is forwarded by the manager of the professional group to the superior section.

On the level of the sections and of the departments the process of opinion forming is repeated on the basis of appraisals they received (consolidation of opinions). It is important that any appraisal is signed by the manager of the respective unit. The professional group and the section of the local chamber likewise forward their (consolidated) appraisal to the respective federal section and the professional unit.

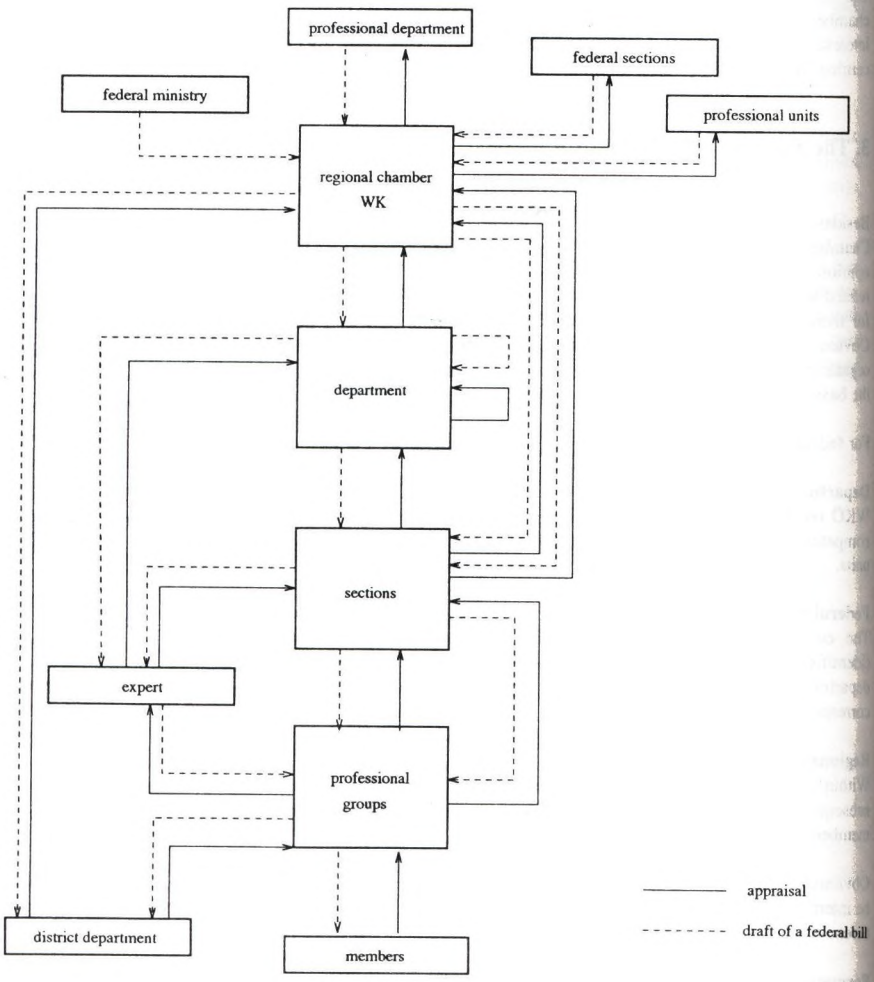


Figure 3.1.: Document flow for Bill Appraisal

A crucial point of this process is the deadline by which the appraisal has to be transmitted to the sending ministry. Therefore, in each distribution step, the distributing unit again sets a deadline to the receiving ones. It may occur that an organizational unit gets into difficulties to meet the given deadline and therefore requests the superior organizational unit for an prolongation. Either the latter is in a position to grant such an prolongation itself or it must forward the request up to the next level. If no prolongation is granted and/or in case of substantial lack of time no appraisal will be obtained from the requesting unit.

It should be clear by now, that computerizing this process on the basis of a workflow system might allow for substantial savings in pass through times and administrational efforts. It is not at least for that reason that we choose it to be investigated within our study.

4. Object oriented business process modeling

There are two approaches to view an organization when using object oriented modeling concepts: A *structure-oriented* and a *behavior-oriented* approach. The first one corresponds to the classical system theoretic approach [KKM93] (that has been adopted by most of the methods of object oriented analysis), i.e., to view a certain universe of discourse (UoD) as a system of interrelated agents that cooperate to reach the common system goals by interchanging flow entities (e.g. messages). As a consequence, business processes are not modeled explicitly as objects on their own but are derived from sequences of object activities and message passing actions.

In contrast to that, the behavior-oriented approach considers business processes on their own [MHH93, FS94] by viewing the dynamical aspects of an organization as a system of business processes. Within this context, a business process my be seen as a bundle of goal directed *actions* that are performed by *actors*, i.e., the organizational units. The actions have causal and temporal interdependencies and are performed sequentially and/or concurrently.

Commonly, a business process is expected to have a defined result, e.g. some document(s), events etc.. In the course of an action job orders and/or messages are exchanged between the actors involved [FS93b]. This, usually, is done using documents of any appropriate kind. The documents which are affected by a given business process may be seen as interrelated [Sc92b].

Organiszational units are the active elements within an organization. When performing an action it does so in order to accomplish one or more *tasks*. Depending on its nature the accomplishment of a task might be fully or partly automated using technical support like machines, information systems etc. [Sc91a].
Similarily, we may distinguish business processes corresponding to the degree as to which their execution can be determined (e.g., in the sense of proceduralization). A purchase process, e.g., might be fully specified without free hand to its actors whereas a trading process might leave unpredictable decisions to its actors and therefore is not completely determined.

In general, a business process may not be completely determined, whenever units are involved to it which are authorized to decide on and/or to "invent" the concrete action to perform. A method for modeling business processes, therefore, has to allow for that kind of indeterminism.
Even if not made explicit, as is done for example in [SST93], OOA methods do so because of their (common) fundamental abstraction concepts that may be applied inversely during a modeling process (generalization/specialisation, aggregation/decomposition).

What we want to discuss within this paper is to which extent such methods may be used to model organizations and their behavior from both viewpoints mentioned before. This is done by using, as an example, one of the most popular OOA methods, namely the Object Modeling Technique (OMT, [Ru91]) and applying it to the restricted miniworld of Bill Appraisal within the Austrian chamber of economics organization.

OMT offers concepts to describe a system from three different but related viewpoints, each capturing important aspects of that system.

The *object model* captures the static aspects of the system by defining the objects, their structure and their interrelationships. It's graphical representation is done using a so-called *object diagram* which contains, similarly to entity relationship diagrams, representation concepts for classes, associations (relationships), attributes, multiplicities, generalizations and aggregations.

The *dynamic model* covers the behavioral properties of the system by means of finite state automata with input which are graphically represented using so-called *state diagrams*. Finally, the functional model defines, on a non-procedural level, the activities of the active elements (the objects), i.e., it describes, what the system does. Data flow diagrams [Yo89] are used for their representation.

Despite of some deficiencies (see, e.g., [KKM93]) OMT allows for a comprehensive conceptual design of important UoD aspects thus providing a basis for the next step in oo system development, i.e., object oriented design (OOD).

5. Conventional Approach using OMT

Within this section we sketch the conceptual model of the Bill Appraisal miniworld using OMT. Of course, only a simplified version fits into this papers limited space. In particular, we disclaimed, from the static model, the introduction of any object attributes. The dynamic model is restricted to the state automata of the most important objects that are involved in Bill Appraisal, e.g., organisational unit, manager and typist.

The complete model is documented in [Ba94]. The pictures have been produced by using OMTool, a software product of the General Electrics Company.

Figure 5.1 shows the main structural object, namely *organisational unit*, with its relationships to bills to appraise (*delegation*), returned opinions and involved persons. Organisational units may be sub- or superunits of others (hierarchy). As a consequence the *delegation* and *return* relationships each express that forwarding bills and returning opinions always involve a super-and a subunit. There is no way in OMT to express that, e.g., a *delegation* instance only might relate units that are related by a *structure* instance, too.
More generally, relationships between relationships are not covered so that additional consistency constraints have to be introduced. For reasons of simplicity such constraints are omitted within this paper.

A conceptual schema of the behavior of managers with respect to Bill Appraisal is given in figure 5.2. At first glance, this state diagram looks confusing, a problem that could be avoided by consequently using abstractions like generalization and aggregation and/or dynamic user interfaces. For the purpose of the paper, however, we had to put it onto one single page.

Note that alternatives of the course of action of Bill Appraisal come up as different state transitions of managers. To a smaller extent this is also true for organisational unit and secretaries. Trivially, it should be possible to interrupt a manager for emergency calls and exception handling when being engaged in Bill Appraisal activities.

That, again, has been omitted for simplicity just like all the work manager do when not involved in Bill Appraisal; after all, managers are supposed to never being 'idle' ....

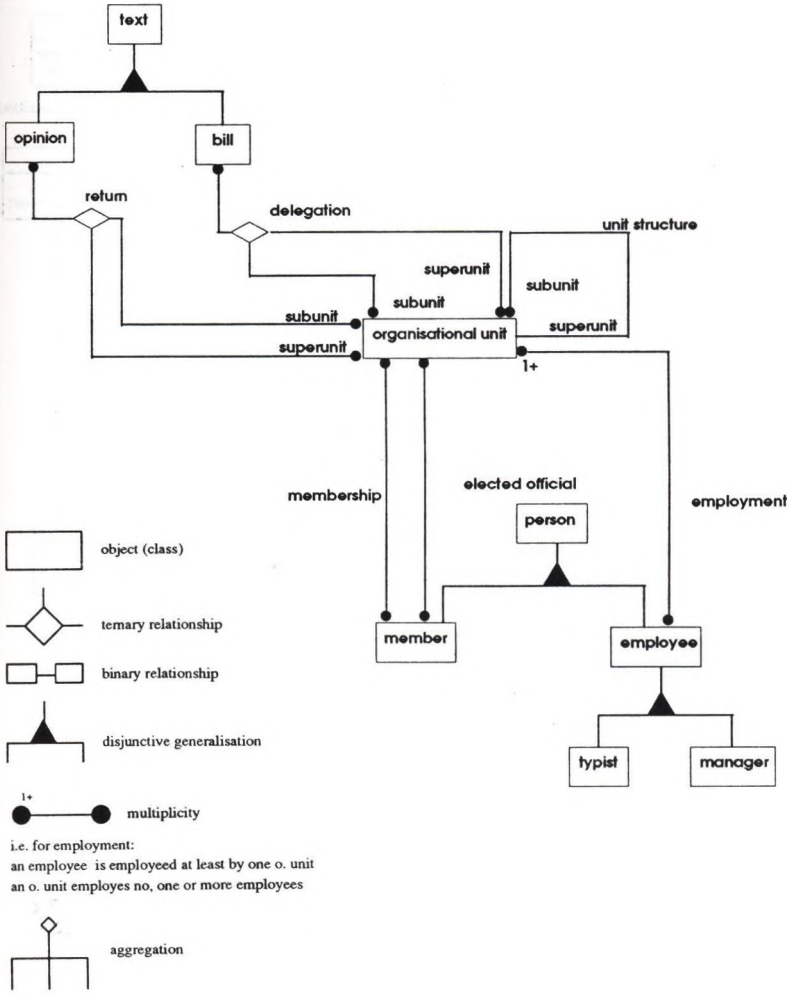
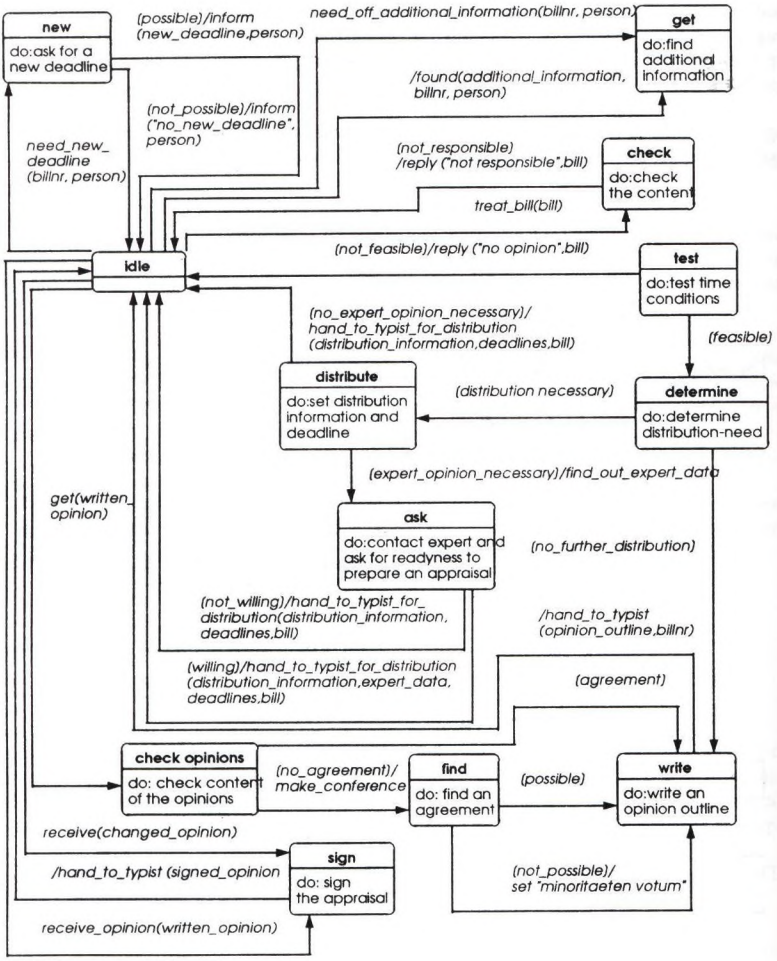


Figure 5.1: Conventional approach, static model



Notation:

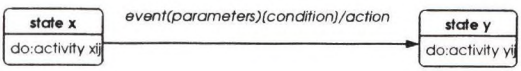


Figure 5.2: Conventional approach, state diagram of object manager

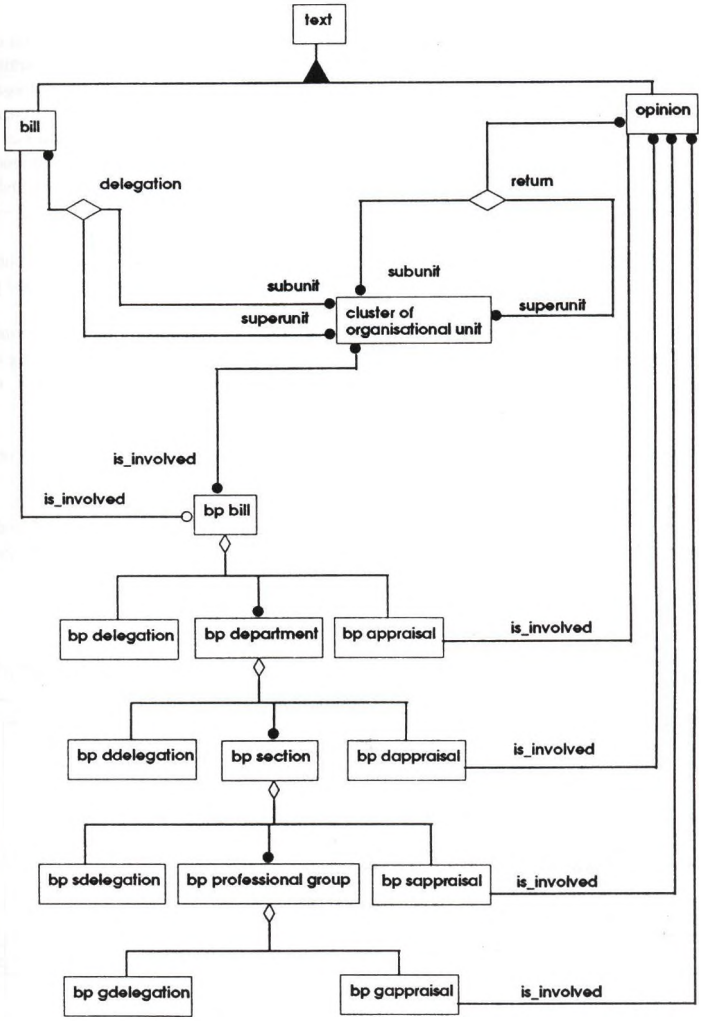


Figure 6.1: The object bp bill

6. Business process oriented approach using OMT

We now proceed to view business processes as objects on their own. This leads to an extension of the static model (figure 5.1) by an object (class) *bp bill* as shown in Figure 6.1. Since Bill Appraisal may be modularized into subprocesses following the organizational levels of the Chamber Organization a decomposition hierarchy has been introduced in our static model. This leads to very simple dynamic models of the 'module processes', see figures 6.2 -6.4 for the state diagrams of the top level processes bp bill, bp delegation and bp appraisal. The diagrams for the lower level process may be derived analoguously. Note that the Chambers business rules for Bill Appraisal now are associated with the process itself and not spread over the behavior of the actors involved.

Figure 6.5. illustrates this fact: the managers state diagram now does contain any 'control structure element' with respect to Bill Appraisal: He makes decisions and 'tells' them to the process (by message passing).

Again, the presented schemata are incomplete. E.g., the static model should specify that an opinion is formed by one and only one appraisal bp. As has been mentioned already this OMT has no modeling concepts to cope with restrictions concerning relationships between relationships, so that additional consistency constraints would have to be introduced.

The state diagrams do not contain names for events if these may be trivially derived from the corresponding state.

On the basis of the foregoing comments, the introduction to Bill Appraisal given in sections 2 and 3, and the legends of figures 5.1 and 5.2 the schemata should be understandable without detailed explanations.

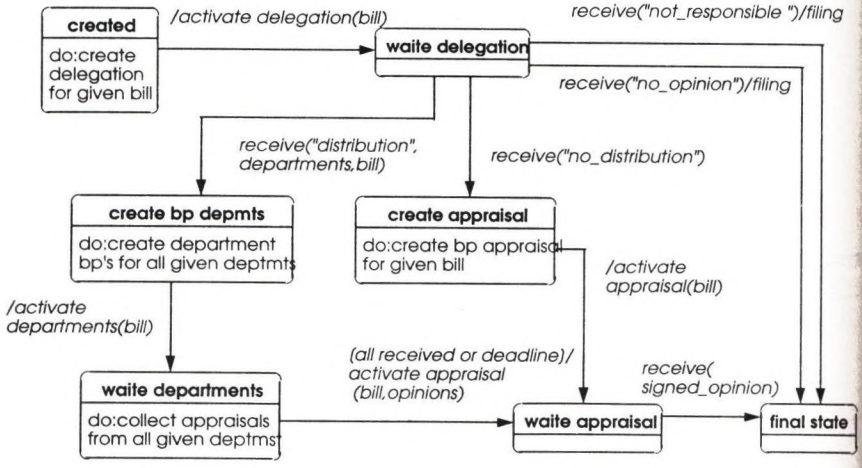


Figure 6.2: State diagram of bp bill

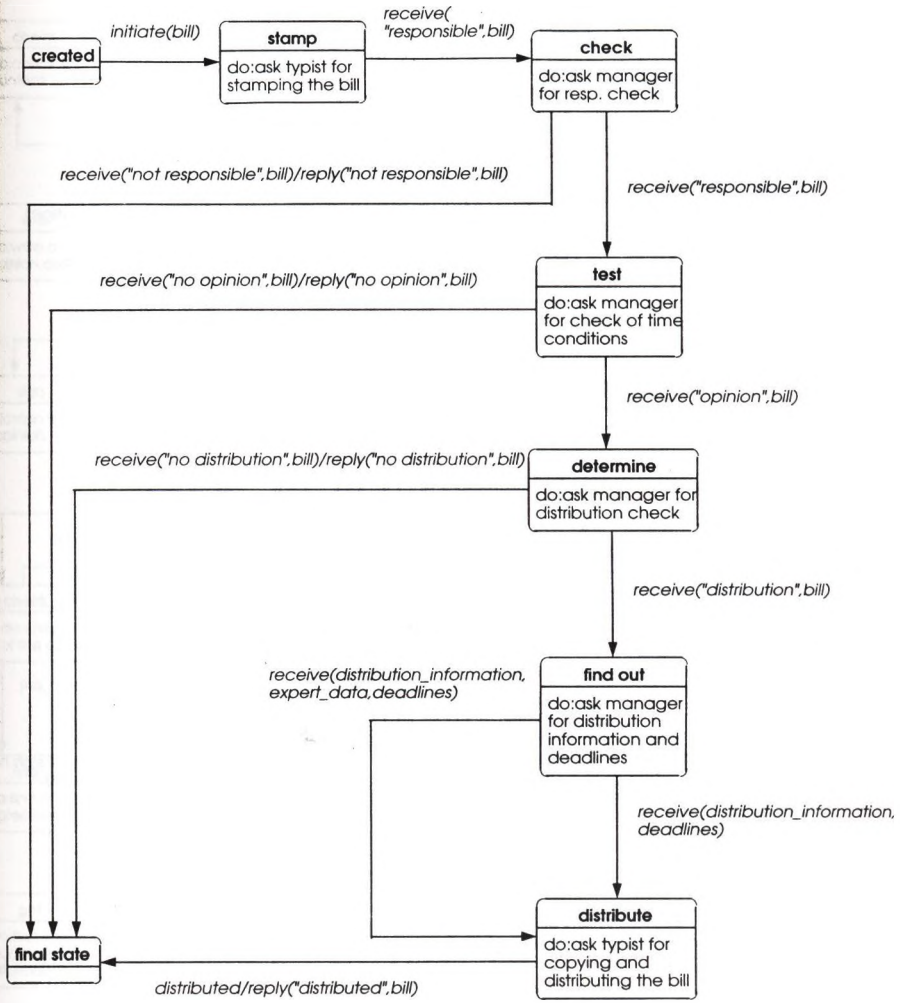


Figure 6.3: State diagram of bp delegation

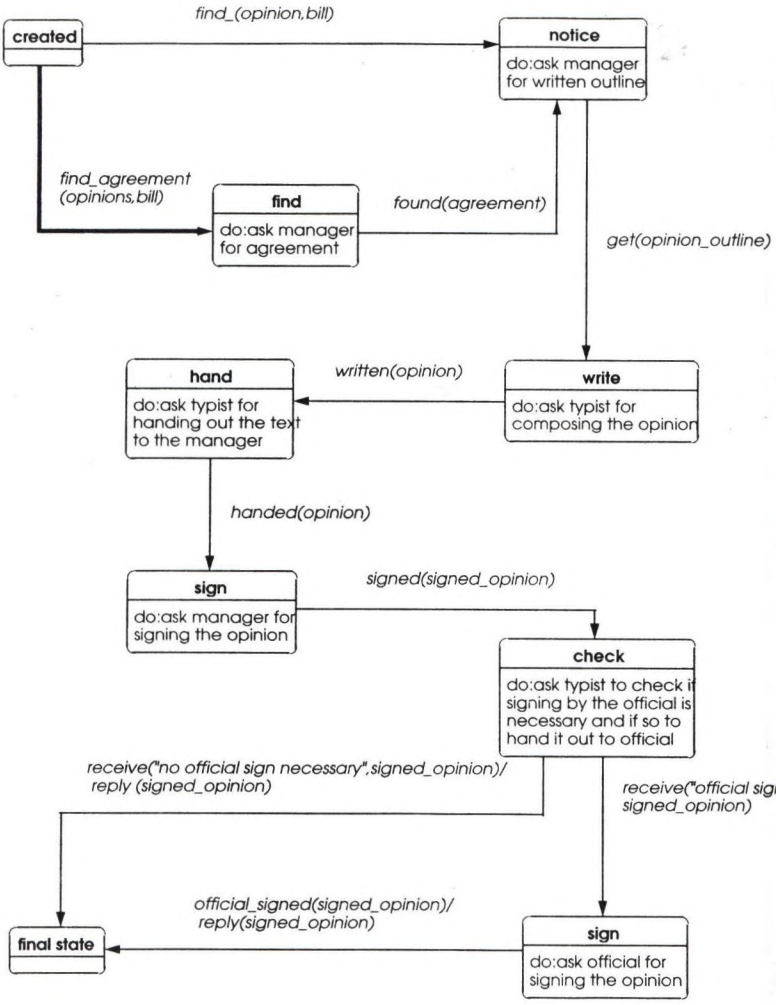


Figure 6.4: State diagram of bp appraisal

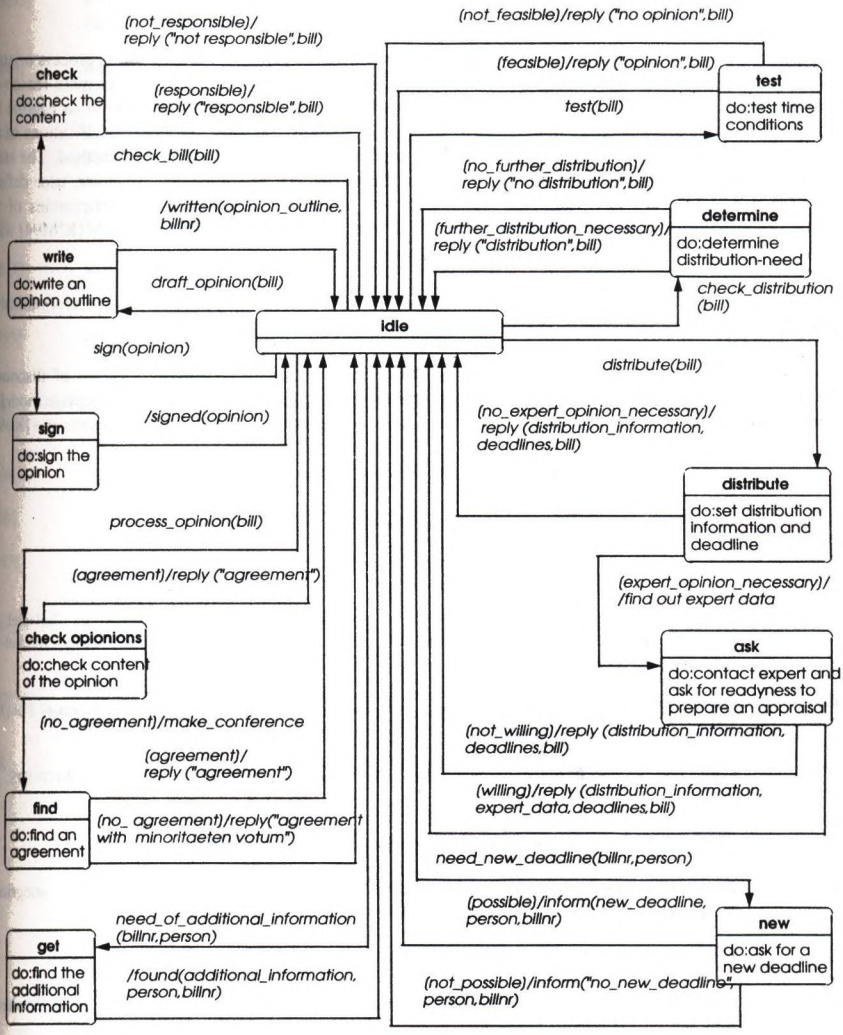


Figure 6.5: State diagram of manager, bp oriented approach

7. Summary

The example discussed within this paper suggests that enterprise modeling using a business process oriented approach leads to comprehensive but less complex models than other approaches do. The main reason for that seems to be the fact that modeling business processes as objects on their own allows to capture the control logic of a business process more naturally than by mixing it into the behavior specifications of the actors involved. Trivially, such an approach needs to rely on an object oriented design method. The study shows, that OMT, a popular OOA method, may be a candidate for that purpose. However, user defined consistency constraints are necessary for really complete designs because of some deficiencies of the modeling concepts offered by OMT. Our experiences with a couple of other OOA methods [K^{4}M94] show that this critique my be generalized to all methods we investigated.

A (intended) side effect of our study is the fact that its results enter into a business process reorganization project of the Austrian Chambers of Economics.

Actually we are investigating the practicability of SOM [FS93b] within the given universe of discourse. Further research is dedicated to the integration of organizational and operational issues of enterprise modeling on a more fundamental level incorporating results of modern organization theory. For first results see [Ko94].

Literature

[Ba94] Bauer, M.: Konzeptueller Geschäftsprozessentwurf mittels OMT und SOM am Beispiel der Gesetzesbegutachtung in den österreichischen Wirtschaftskammern. Diploma thesis, Institut für Informatik, Universität Klagenfurt, 1994.

[BCN92] Batini, C; Ceri, S.; Navathe, S.B.: Conceptual Database Design. The Benjamin/Cummings Publ.Co., 1992.

[Ch76] Chen, P.P.-S.: The Entity-Relationship Model - Towards a Unified View of Data. ACM TODS, 1/1, 1976.

[CY91] Coad, P.; Yourdon, E.: Object Oriented Analysis. Prentice Hall, 1991.

[Ei94] Eicker, S.: Die Informationssystem-Architektur als Teilschema eines zentralen, integrierten Metadatenpools. GI Fa 5.2 Rundbrief, 1/94.

[FS93a] Ferstl, O.K.; Sinz, E.J.: Grundlagen der Wirtschaftsinformatik, Oldenbourg-Verlag, 1993.

[FS93b] Ferstl, O.K.; Sinz, E.J.: Der Modellierungsansatz des semantischen Objektmodells SOM. Bamberger Beiträge zur Wirtschaftsinformatik, Nr.18, 1993.

[FS94] Ferstl, O.K.; Sinz, E.J.: Tool-based Business Process Modeling Using the SOM Approach. Bamberger Beiträge zur Wirtschaftsinformatik, Nr.19, 1994.

[Gr83] Grochla, E.: Unternehmensorganisation. Westdeutscher Verlag, Opladen, 1983.

[Ja92] Jacobson, I. et.al.: Object-Oriented Software Engineering. Addison Wesley, 1992.

[HKG94] Handelskammergesetz. Österreichisches Bundesgesetz vom 24.7.46, 10. Novelle vom 1.1.94.

[KKM93] Kaschek, R.; Kohl, C.; Mayr, H.C.: Grenzen objekt-orientierter Analysementhoden am Beispiel einer Fallstudie mit OMT. In [MW93], pp. 135-154.

[KKM94] Kaschek, R.; Khouri, B.; Kohl, C.; Kop, Ch.; Mayr, H.C.: A Hierarchical Classification System for Comparing OOA Methods and Tools. Poster Contribution to CRIS'94, Maastricht.

[Ko94] Kohl, C.: Die Anwendbarkeit von OOA-Konzepten in der Unternehmensmodellierung. Proc. EMISA/MOBIS Workshop "Modellierung von Geschäftsvorfällen und -prozessen", Münster, 1994. (to appear)

[Kr90] Krcmar, H.: Bedeutung und Ziele von Informationssystem-Architekturen. Wirtschaftsinformatik 5/90.

[MHH93] Müller-Luschnat, G.; Hesse, W.; Heydenreich, N.: Objektorientierte Analyse und Geschäftsvorfallsmodellierung. In [MW93], pp. 74-92.

[MW93] Mayr, H.C.; Wagner, R. (eds.): Objektorientierte Methoden für Informationssysteme. Springer Verlag, 1993.

[Ru91] Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.: Object Oriented Modeling and Design. Prentice Hall, 1991.

[Sc92a] Scheer, A.W.: Architecture of Integrated Information Systems. Springer, 1992.

[Sc92b] Schmidt, S.: Modellierung von Büro-Informationssystemen. Vdf-Verlag, Zürich, 1992.

[SST93] Schewe, B.; Schewe, K.-D.; Thalheim, B.: Verfeinerungsschritte im Rahmen einer objektorientierten Entwurfsmethodik. Datenbank-Rundbrief der Ges. für Informatik, Nr. 12, Nov. 93, pp. 7-14.

[Tu91] Tulowitzki, U.: Anwendungssystemarchitekturen im strategischen Informationsmanagement. Wirtschaftsinformatik, 2/91, pp. 94-99.

[WWW90] Wirfs-Brock, R.; Wilkerson, B.; Wiener, L.: Designing Objected-Oriented Software. Prentice Hall, 1990.

[Yo89] Yourdon, E.: Modern System Analysis. Prentice-Hall International Editions, 1989.

Limits of Modelling

Functional and Behavioral Aspects of Process Modeling in Workflow Management Systems

Stefan Jablonski[1]
Digital Equipment GmbH, AIT Karlsruhe
Vincenz-Priessnitz-Strasse 1, D-76131 Karlsruhe
Tel.: [49] (721) 6902-25 Fax: [49] (721) 696816
e-mail: jablonski@kampus.enet.dec.com

Abstract. *Process modeling is a multi-facet task. Functional, behavioral, organizational, and informational aspects have to be tackled. This article presents a comprehensive framework for process modeling. The model proposed allows to describe and execute either well structured process chains and highly flexible negotiation types of processes. Both process categories are enacted by a Workflow Management System.*

1 Introduction

The design of Cooperative Information Systems is a complex task that needs methodological support. It combines methods and techniques from organization theory, behavioral theory, decision-theory, and database theory. These techniques are necessary in order to obtain a complete and formal conceptual model for an Information System. Such a model is necessary in order to facilitate

- human understanding of a system
- process optimization
- automated execution support.

Modeling information systems mostly has focused on the data related features of Information Systems. Data are analyzed in order to obtain a conceptual database scheme; sometimes also dataflow is captured [25, 27]. Modeling system dynamics, i.e. the specification of processes and

[1] *New address:* University of Erlangen-Nuemberg, Department of Computer Science VI (Database Systems), Martensstrasse 3, D-91058 Erlangen; Tel.: [49] (9131) 85-7893, Fax: [49] (9131) 32090, e-mail: jablonski@informatik.uni-erlangen.de.

their control flow, has been neglected fairly or is too much theory driven and therefore not practical for real world problem modeling (e.g. Petri-Net approaches). In this paper we introduce an approach for process modeling that is tailored to model business processes in office environments. Since we are specifically looking into the area of *Cooperative Information Systems*, processes are supposed to support collaborative work settings. Cooperation can take place either as well-specified process according to strictly defined rules or as loosely defined activity that performs according to vague guidelines. Both process categories must be coped with simultaneously by Cooperative Information Systems because both process categories exist concurrently in office environments.

Examples for strict forms of cooperations are travel claim reimbursement processes: people involved have to act precisely according to exactly defined rules [15]. In contrast, the process of finding date, place, and agenda for a meeting is vague, although the types of elements used in such a process (e.g. suggestions, agreement, rejection, commitment) are known beforehand. Such kind of processes can be characterized as conversational team work [18].

Our goal in modeling processes and specifically in handling control flow between processes is to capture solely the true constraints on ordering. What happens very often in conventional approaches is that only a limited sets of constructs for control flow specification is supported (usually serial, parallel, and alternative execution). These constructs cannot be tailored to real problem requirements. Thus, language constructs have to be used that are too restrictive. This overspecification often leads to bad system performance or inappropriate system behavior what is one of the principal causes of change requests in application systems. An example explains this situation: When an application has to be modeled which needs five activities to be executed serially but in any order, 5! (i.e. 120) different ways of modeling the serial control flows are possible. One or a few alternative serial courses of execution will be modeled; anything else is not expressible with the constructs introduced before. It is most likely that over time some change request will occur to support other execution sequences as well. To add these new routes of execution is cumbersome and is not acceptable in huge systems. Therefore we have to support control mechanisms which allow to model the true constraints between processes. The probability that these specifications are resistant to change requests as experienced above is very high.

This paper introduces an hybrid approach to process modeling which covers either well structured and loosely structured process types. It is part of a comprehensive project where the link between *Business Process (Re-)Engineering* and *Workflow Management (WFM)* is investigated. WFM systems (WFMSs) are considered to provide the execution infrastructure for business process reengineering; they identify a new platform for process execution in distributed environments.

This paper deals with artefacts which are relevant on the level of business process modeling. It is not intended to talk about pure logic, although many artefacts introduced later look very logic oriented (cf. Section 3); nonetheless, profound logic is needed for the final implementation of our approach (e.g. [19]). Therefore the criteria our approach must be measured against are not logic driven (e.g. completeness) but modeling issues like ease of use, reusability, tailorability, coherence.

We have used the term *process* extensively up to this point without having defined it clearly. Because process is a widely spread term everybody interprets it differently. We adopt for the discussions in this paper the definition from [12]. There, processes are defined as *sets of partially ordered steps intended to reach a goal.* Process are composed of *process elements.* The most fine-grained, atomic process element is a *process step.* According to [7] processes are characterized by the following aspects/perspectives:

Perspective/Aspect	Characterization
functional	What processes are performed?
behavioral	When are processes performed?
organizational	Who performs processes?
informational	What information elements are produced or consumed by processes?

Sections 2, 3, 4, and 5 are dealing with these aspects although we concentrate on the functional and behavioral aspects since the organizational aspect and the informational aspect are discussed elsewhere broadly [5] [13]. In order to give an idea of how these aspects act in combination, we discuss the example depicted in Figure 1: The processes (workflows) that have to be executed are 'process travel claim', 'submit travel claim', 'approve travel claim', and 'reimburse client'. The latter three are a refinement of the overall travel claim reimbursement process. Having nested the three workflows within the workflow 'process travel claim' means that they are implementing this compound workflow. The ordering between workflows is specified by arrows. Person who are supposed to execute the workflows are attached (e.g. Client, Manager). Data elements being passed between workflows are travel claim data (TC). Elementary workflows are implemented by applications; for instance, the elementary workflows 'submit travel claim' and 'approve travel claim' are all implemented by specific functions of the program 'TravelClaimSystem'.

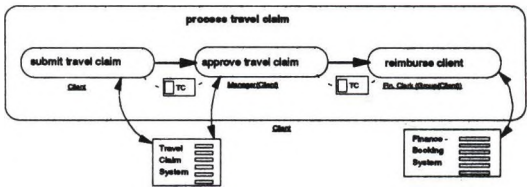


Figure 1: Process travel claim

In this paper, we introduce a process model for Workflow Management Systems, i.e. a workflow model. It is characterized by the *modular treatment* of the above mentioned aspects in one comprehensive model. Meyer [22] states that modularity (of software) is the prerequisite for the so-called external quality factors correctness, robustness, extendibility, reusability, compatibility, efficiency, portability, verifiability, integrity, and ease of use. It's obvious that for our purpose of process modeling in WFMS not all these factors are of the same relevance, however, factors like correctness, extendibility, reusability, verifiability, and ease of use are mandatory for each workflow model.

2 The Functional Perspective

2.1 Workflows

Workflows are the implementation of either processes, process elements and process steps. Thus, workflows are recursive structures, i.e. a workflow may consist of further workflows. Figure 1

shows the workflow 'process travel claim'; it consists of three so-called *subworkflows*, 'submit travel claim', 'approve travel claim', and 'reimburse client'. Although not shown in the figure, each of the subworkflows could also be composed of further subworkflows. Probably, the workflow 'reimburse client' is composed of subworkflows 'return approved travel claim' and 'transfer money'. An outermost workflow, i.e. a workflow that does not have any superworkflows, is called *top-level workflow*. Workflows containing other workflows are called *composite workflows*. Workflows not composed of other workflows are called *elementary*. Applications are referenced by elementary workflows. They are implementing the functionality of an elementary workflow. Applications are programs, transactional steps of TP-Monitors, command procedures, or any other executable piece of code. Two main groups of applications can be distinguished: *legacy applications* and *non-legacy applications*. Legacy applications are characterized by the fact that they cannot be modified. They are inherited from past system states and have to be treated as they are. Non-legacy applications are easier to cope with. They can be tailored to the specific needs of WFM if they have to be incorporated into a workflow. From a modeling point of view, applications can be regarded as a very specialized form of workflow.

Because workflows are representing either processes, process elements, and process steps, only one concept is needed in order to cope with different kinds of processes, simple and complex ones, in the same model. Workflows determine how people involved in a problem solving process cooperate and collaborate.

Like procedures in programming languages, workflows are black boxes; only the signature determined by the type of a workflow and the types of its in- and out-parameters are externalized. This feature supports reusability of workflows which is an absolute requirement for modern software development. The objective is to populate public class libraries of workflows. Our approach to process design is maximizing synthesis of existing workflow specifications [26].

A workflow type is defined as follows (non-terminal symbols are put in parentheses ('<', '>'); terminal symbols are without parentheses, they are written with capital initials)[2]:

```
Workflow Type <workflow-type>
In: <parameter-list> Out: <parameter-list>
Subtype Of <workflow-type>
<workflow-body>
```

The symbols are self-explanatory. `<workflow-body>` represents the kernel of a workflow, it will be refined stepwise in the following sections. Since we are using an object oriented approach, inheritance is supported (`Subtype Of`).

For a composite workflow processing starts at the top-level workflow by executing the first underlying layer of subworkflows. These subworkflows are processed according to the precedence structure defined (cf. Section 3). Recursively processing continues until the bottommost level is reached; on this level elementary workflows are found exclusively. Elementary workflows are processed by executing the applications referenced in their bodies.

Two main classes of workflows can be distinguished:

Prescriptive workflows. Eligible (sub-)workflows (and also the precedence structure between workflow instances) are known a priori.

Descriptive workflows. Instances of participating (sub-)workflows are not known beforehand but are determined during processing. However, their types are also known a priori.

---

[2]In this paper, we do not intend to introduce a formal script language. However, we use a kind of script language merely for the purpose to show the semantics of modeling constructs.

Conventional office procedures (e.g. process travel claim) are examples for prescriptive workflows: each step of execution is very well known in advance. Design tasks are examples for descriptive workflows; although workflow types needed to describe a certain problem are known, the concrete way of processing them, i.e. the corresponding workflow instances, is not known beforehand.

2.2 Prescriptive Workflows

Prescriptive workflows are specified by declaring the set of subworkflow and application instances. This is done in subsection `<subworkflows>` and `<applications>` of `<workflow-body>`:

```
<subworkflows> ::=
  {<workflow-type>: <workflow-instance> {, <workflow-instance>}*;}*
  {<application-type>: <application-instance> {, <application-
 instance>}*;}*
```

`<activity-type>` refers to a type of an activity; `<activity-instance>` describes an instances of a specified activity type. In elementary activities, applications `<applications>` are specified.

The above shown construct is the basis for an interesting observation which very often leads to confusion in WFMSs: let's assume that a new workflow type `Act_new` is to be specified. In the definition of this type the already specified types `Act_old1` and `Act_old2` are referenced because `Act_new` references subworkflows of these two types. The corresponding two lines of the specification look as shown below:

```
Workflow Type Act_new
...
/* specification of subworkflows */
  Act_old1: Act_old1_inst1, Act_old1_inst2;
  Act_old2: Act_old2_inst;
```

Note that `Act_new` is the name of a type to be specified. `Act_old1` and `Act_old2` are names of already known types. `Act_old1_inst1`, `Act_old1_inst2` and `Act_old2_inst` are instances of the latter two types, in programming languages they would be called variables. They will be instantiated during the execution of instances of type `Act_new`.

2.3 Descriptive Workflows

Some sort of processes are characterized by knowing *what* to do, but not knowing exactly *how* to do it. Descriptive workflows are supposed to cope with this problem. An example clarifies the need for descriptive workflows: Negotiations can be modeled by the workflow types 'suggesting', 'asking', 'answering', 'agreeing', 'disagreeing' [17]. For a brainstorming session, the types 'articulate idea', 'articulate assumption' are essential. Although all types might be sufficient to set up negotiations and brainstorming sessions, respectively, it is not known whether and how many instances will be needed eventually in order to model a concrete scenario. In well going negotiations there is no need for workflows of type 'disagreeing'; in cumbersome negotiations, unfortunately, the workflow type 'agreeing' is not used at all (cf. Section 3.3). Descriptive workflows are therefore characterized by not knowing the workflow instances needed eventually, but being able to perceive the types of instances needed principally (cf. example 'negotiation' above).

What is required for descriptive workflows is *intensional modeling* [2]. In contrast, prescriptive workflows are modeled extensionally: all extensions, i.e. instances, are defined a priori. Intensional modeling provides an intensional framework for workflow execution: workflow types that might

be needed for solving a specific problem are declared together with a statement about the goal and purpose of a workflow. Besides, mandatory control flow restrictions are also specified (cf. Section 3.1.3)[3]. Figure 8 demonstrates pragmatically and practically how descriptive workflows are specified. In [26] the logical foundation for descriptive workflows are given.

For the specification of subworkflows in descriptive workflows the following language construct is used:

```
<subworkflows> ::=
  <workflow-type> (<description>){, <workflow-type> (<description>)}*
```

For each subworkflow a description of its need and purpose is attached. This description supports the process of finding the right workflow instances to perform a certain function. In [16] and [26] formal approaches to specify intensional artefacts like goals (described via mandatory features) are shown.

3 The Behavioral Perspective

The interdependencies and interrelationships between workflows, i.e. the behavioral aspect, is characterized through the *control flow* (synonymously: *flow control*). Flow control determines when workflows are performed. For the description of flow control a workflow body is extended by a section `<flow-control>`. Flow control is specified in terms of execution rules determining the sequence to perform workflows.

Flow control determines when resources executing workflows are cooperating and collaborating. There is a wide range for cooperation and collaboration respectively. Prescriptive and descriptive control types can be distinguished. Tightly fixed forms of control prescriptively define the way workflows are executed, i.e. how cooperation and collaboration takes place; loosely described forms of control merely establish an ordering framework for workflows. This framework opens many degrees of freedom for workflow processing. The actual execution of workflows must comply with this framework.

3.1 Control Flow Types

In this subsection, we introduce meaning, graphic and textual representation of flow control statements. In Section 3.2 the semantics of flow control statements is discussed.

Recall that flow control always implies an ordering relation on subworkflows. Watching them from the standpoint of a superworkflow, flow control determines the sequence of subworkflow execution.

3.1.1 Prescriptive Flow Control

There are three basic forms of prescriptive flow control, serial execution, alternative execution, and parallel execution. They will be explained in this subsection.

Serial Execution

Serial Execution is a very restrictive form of ordering subworkflows. Exclusively one course of processing is possible, namely strictly sequential execution.

---

[3]The intensional framework for descriptive workflow modeling is defined by two 'rules': the instantiation 'rule' says that instances can be derived from declared workflow types; the control flow 'rule' says that instances have to be put in the precedence structure defined by the control flow constructs.

Because the 'bubbles and arcs' notation is very common and popular for the description of workflows we are presenting sample workflows in terms of this notation. Figure 2a shows an example of a workflow A witch enforces strictly sequential processing of its subworkflows B, C; the execution sequence [BC] is mandatory. We use the notation '[...]' to denote the history of execution; this notation is similar to the notation introduced in [3] used for describing execution histories for transactions in database systems.

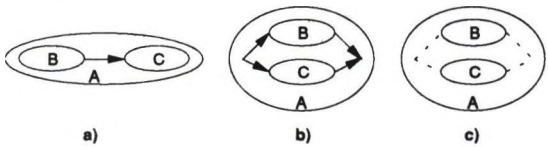


Figure 2: Types of Prescriptive Workflows

Below, the script version of the example in Figure 2a is presented.

```
Workflow Type A
...
-> (B, C)
...
```

For convenience, the following language transformation is valid (B, C, D are workflows), because sequencing is either distributive and transitive:

```
-> (B, -> (C, D)) == -> (-> (B, C), D) == -> (B, C, D)
```

From a language point of view the above transformation is not critical, although from a semantics point of view it might bear critical consequences. Specifying `(B, -> (C, D))` instead of `(-> (B, C), D)` might have an impact on error treatment. Having C and D more tightly coupled than B to the pair C D might mean, that C D should be treated as an atomic unit. If either C or D cannot be executed, the effects of the second workflow should be discarded as well. Nevertheless, specifying flow control and specifying error behavior is orthogonal; the latter is detailed in [14].

Alternative Execution

The order of processing is less strictly determined for workflows in which *alternative execution branches* can be specified; alternating courses are mutually exclusive; they have to be joined eventually. Conditions attached to alternative branches must be logically disjoint. Altogether they should cover the whole range of possibilities. Looping, i.e. iterative processing of (sequences of) workflows, is enabled through the introduction of alternative execution. Of course, a construct like the *goto* construct known from programming languages is necessary as well. However, we do not intend to introduce this construct at the user interface but use it internally only.

Figure 2b depicts an example for workflows with alternative courses of execution. Either [B] or [C] are executed depending on whether condition *cond()* is evaluated to true or to false. *cond()* is a problem specific condition.

```
Workflow Type A
...
α (cond(), B; ~cond(), C)
...
```

For the conventional forms of loops the following self-explanatory constructs are introduced:

```
while_do (cond(); B)
repeat_until (cond(); B)
for_do (cond(); B)
```

In the latter case, *cond()* indicates the starting value of a loop variable, the increment and the final value of the loop variable.

Parallel Execution

As a third form of prescriptive execution control *parallel execution* is introduced. In this case workflows can be activated concurrently.

Figure 2c shows the following example: B and C as the only subworkflows of composite workflow A have to be executed in parallel [B||C].

```
Workflow Type A
...
|| (B, C)
...
```

For convenience, the following transformation is valid (B, C, D are workflows):

```
|| (B, || (C, D)) == || (|| (B, C), D) == || (B, C, D)
```

The latter transformation motivates to offer a very interesting language construct which is a special case of parallel execution. Let's assume that a workflow B has to be executed multiple times, however not necessarily sequentially but concurrently. The former way of execution can be achieved with a loop; for the latter way of execution a new construct must be introduced. An appropriate notation would look like:

```
|| (B, B, B, ...).
```

B has to be executed as often as a problem specific condition prescribes; but this hardly can be expressed by using the dot notation. A language construct for this form of execution is called *repetition*. A more appropriate notation for repetition is

```
ρ(B).
```

It depends on the implementation of flow control whether an upper and/or lower bound for repetition has to be specified. It is also implementation dependent when instances of workflow B can be generated. Restrictive types of implementation would only allow to generate instances at the very beginning of the execution of a repetition; dynamic implementations would allow to generate new instances while other are already in execution. ρ() generally denotes a condition function, controlling how often B is going to be instantiated. Besides other things, ρ() can be related to

- the number of instances that can be created,
- the time frame which delimits the creation of instances.

An example justifies the meaning of the replication construct: For an assembly different parts have to be ordered. It is hard to determine a priori how many different parts are needed because either each assembly is different and the stocks vary as well. For each part to be ordered the *order_part(.)* workflow must be called. If this would be done within a loop, all parts are ordered sequentially which unnecessarily extends latencies. This is a waste of time and also might bother the personnel since they block each other from ordering parts. It is obvious how easily this situation can be modeled with the repetition construct in a natural way and how execution time can be saved.

3.1.2 Descriptive Flow Control

For the specification of descriptive flow control two different types of conditions are introduced, *temporal* and *existence conditions*. As elaborated in [19] and approved in [1] these two types of dependencies are adequate and necessary to describe almost all usual protocols for the synchronization of computing agents. In our case these agents are materialized as workflows.

In contrast to prescriptive flow control where each specification results in a unique and concrete template for processing, descriptive flow control merely characterizes equivalence classes of processing. Temporal conditions denote the timely relationship between workflows; existence conditions describe mandatory existential dependencies.

Temporal Condition 'Deadline'

A first type of temporal flow control has the form

```
B < C.
```

It is called *deadline* since the execution of workflow B is limited by the occurrence of workflow C timewise. The above statement expresses that B has to be executed before C, if it is processed at all. It is forbidden that B and C are executed in parallel. Therefore, the following rules describe permissible executions:

- If C hasn't started yet, B can be executed.
- If B hasn't started yet, C can be executed.
- If B is finished, C can be executed.
- B need not to be executed at all.
- C need not to be executed at all.

Thus, the following sequences of execution are valid: [], [B], [C], [BC]. The empty sequence [] is allowed since the deadline rule does not demand to execute participating workflows at all.

Each specification of descriptive control therefore produces a class of equivalent and valid processing scenarios. In Figure 3 either the graphical notation for deadline and also equivalent specifications which fulfill the deadline semantics are shown. These specifications form a so-called *set of valid execution sequences (SVES)* of the deadline condition. Each valid specification only shows a piece of the semantics expressed by the deadline condition. All specifications together show the complete semantics of this temporal condition. Figure 3 demonstrates that if only sequences, alternatives, and parallelism are available the declaration of workflows would drastically get cumbersome. This observation also applies to the forthcoming discussions on delay and existence conditions.

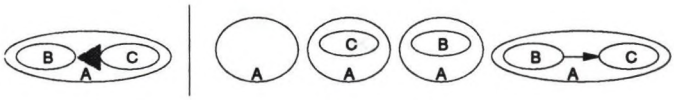


Figure 3: Condition Type 'Deadline'

Following, the script form of the example in Figure 3 is shown:

```
Workflow Type A
  ...
  < (B, C)
  ...
```

A sample use of the deadline construct can be found in the business field when relationships to actual and potential customers have to be maintained. Customers are posting requests (workflow B) while a vendor has to respond with some offer (workflow C). If a request has been received, an offer should be made ([BC]). If for a very long period of time a customer does not order anything, it is common practice to send him a 'blind' offer ([C]). Sometimes it is not possible to respond to an inquiring order ([B]) because no offer can be made. If a request comes in after an offer has already been made, a new instance of workflow A has to be created to cope with this event. It is

worth being mentioned again that all alternative courses of processing are enabled by just one construct, the deadline construct. This alleviates modeling drastically.

Recall that the modeling construct *deadline* is used to express dependencies between workflows. It must not be mixed up with the *temporal constraint* bearing the same name mostly. This latter construct denotes one specific point in time which delimits the occurrence of events in general, execution of workflows in our special case. It has to be regarded as an artefact which belongs to execution policies (cf. Section 4). A workflow which is not executed within such a temporal deadline is regarded as having failed. This is totally different to our language construct which does not indicate whether workflow execution failed: here, some workflows are just not executed because of specific relationships to other workflows. Those workflows are called to be *disabled*. Being disabled is different from having failed (cf. Section 3.2).

Temporal Condition 'Delay'

Delay is a second form of temporal flow control:

```
C > B.
```

Workflow C must be delayed until workflow B has finished or will never be executed. This means that

- B can be processed any time
- C can be processed only if
  - either B was executed already and has finished
  - or B will never be executed.
- B need not to be executed at all.
- C need not to be executed at all.

In order to know whether workflow B will still be processed a specific variable has to be inquired (see Section 3.2 for details). The following sequences of execution are valid: [], [C], [B], [BC].

In Figure 4 either the notation for delays and the SVES for the delay condition are shown for the following sample specification:

```
Workflow Type A
...
> (C, B)
...
```

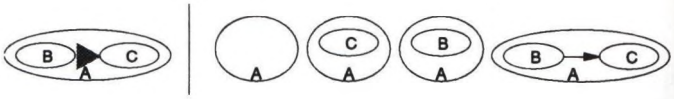


Figure 4: Condition Type 'Delay'

Although the SVESs for deadline and delay conditions look equal, their semantics, i.e. the reasons for coming into existence, are significantly different (cf. Section 3.2). In case 'delay' B can be executed as long as an indicating condition holds. This condition is dependent on application oriented matters but not directly on workflow C. In case 'deadline' the situation is quite different: the occurrence of C determines and delimits abruptly the chance of B to be executed.

As an application of the delay condition the start procedure of an aeroplane is discussed (workflow A). Workflow C stands for taxing and take off; workflow B models the boarding procedure. If boarding has terminated, take off can take place ([BC]). If boarding cannot take place at all (for example because it is a non-intended intermediate stop), take off can occur

without having to wait for the end of boarding [C]; in this case boarding was disabled before. In case the flight has to be cancelled because of bad whether conditions, take off cannot be done, although boarding might already have finished ([C]). If the whole flight is cancelled before boarding started yet, neither B nor C is executed ([]).

Existence Condition

Existence conditions are formulated as follows:

```
B => C.
```

The execution of C is enforced when B was executed. Thus if C is disabled and cannot be executed any more, B must be disabled, too. If B will not be executed, the execution of C is optional. [BC], [CB], [C||B], [C], and [] are permissible sequences of execution. Figure 5 depicts the graphic notation for existence conditions as well as the SVES.

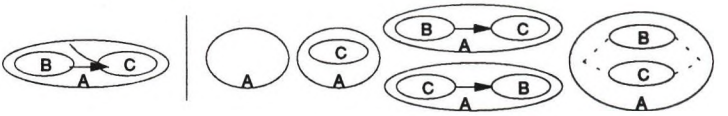


Figure 5: Condition Type 'Existence'

The script version of the example is shown below:

```
Workflow Type A
...
=> (B, C)
...
```

Example. Workflow A represents a scenario dealing with the demonstration of a prototype. Workflow C comprises everything that has to be done in order to set up the demonstration. Workflow B stands for all matters necessary to announce a demo. Principally it does not matter whether an announcement takes place firstly or a demo is set up firstly ([BC], [CB], [B||C]). But if it becomes obvious that the demo cannot be set up, no announcement should be made ([]). Even if no announcement is made a demo might be prepared ([C]). The example shows how dependent the outcome of an existence condition is on the ability to execute a workflow C. Only the knowledge that C can be executed allows to initiate execution of workflow B. This knowledge has to be reflected in the system.

Generalization

Each the deadline, the delay, and the existence primitive can be generalized. That means, relationships between multiple workflows can be specified. Then, B represents a set of workflows $B_1, ..., B_b$ and C represents a set of workflows $C_1, ..., C_c$. The three primitives result in

```
<< (B1 ... Bb, C1 ... Cc)
>> (C1 ... Cc, B1 ... Bb)
=> (B1 ... Bb, C1 ... Cc).
```

How these complex constructs are derived from the basic forms is shown in the following:

Deadline:

```
<<(B1 ... Bb, C1 ... Cc) <=>
<<(B1 ... Bb, C1) ∧ ... ∧ <<(B1 ... Bb, Cc) <=>
<<(B1, C1) ∧ ... ∧ <<(Bb, C1) ∧ ... ∧ <<(B1, Cc) ∧ ... ∧ <<(Bb, Cc)
```

A B_i can execute before an arbitrary C_j starts execution.

Delay:

$>>(C_1 \ldots C_c, B_1 \ldots B_b) <=>$
$>>(C_1, B_1 \ldots B_b) \wedge \ldots \wedge >>(C_c, B_1 \ldots B_b) <=>$
$>>(C_1, B_1) \wedge \ldots \wedge >>(C_1, B_b) \wedge \ldots \wedge >>(C_c, B_1) \wedge \ldots \wedge >>(C_c, B_b)$

All C_j are delayed by B_i. Only if all B_i have executed or have promised not to execute at all, a C_j can be performed.

Existence:

$=>(B_1 \ldots B_b, C_1 \ldots C_c) <=>$
$=>(B_1 \ldots B_b, C_1) \wedge \ldots \wedge =>(B_1 \ldots B_b, C_c) <=>$
$=>(B_1, C_1) \wedge \ldots \wedge =>(B_b, C_1) \wedge \ldots \wedge =>(B_1, C_c) \wedge \ldots \wedge =>(B_b, C_c)$

A B_i can only execute if all C_j can be executed.

The six primitives introduced above can also be nested by replacing a workflow placeholder with another control construct. In the example

```
--> (B; C),
```

B is substituted with a deadline dependency between D and E, and C is substituted with parallel execution of F and G:

```
--> (<<(D; E); ||(F; G))
```

The execution semantics is as follows: after the deadline construct is performed, F and G can execute in parallel.

In the example

```
>> (B; C)
```

B and C as the placeholders of the delay primitive are substituted with `--> (D, E)` and `=>(F; G)`, respectively:

```
>> (-->(D; E); =>(F; G))
```

Before D (and afterwards E) can execute, the existence relationship between F and G must be performed, i.e. either G, F and G, or none of the two workflows have been performed.

The set of prescriptive and descriptive control flow constructs introduced so far make up a decent basis for the definition of additional problem-specific control flow constructs (*macros*; cf. Section 3.1.4).

3.1.3 Control Specification for Descriptive Workflows

For descriptive workflows instead of workflow instances, workflow types are referenced in control constructs. Instances of these types can be generated in order to implement the functionality of a workflow. When control is defined between types $t_1, \ldots, t_n$ instances of these types have to obey the control order specified on the type level. See Section 3.3 for an example.

Control specification on the type level together with descriptive control types allow to specify unstructured forms of interactions (e.g. negotiation). [18] conveys the global idea of how this can be accomplished.

3.1.4 Macros

After having provided a basic set of primitives for control flow specification, most kinds of problem specific control types can be formulated. As an example the so-called *skip* macro is introduced. `skip(cond(), B)` means that workflow B has to be skipped when condition `cond()` holds; otherwise it has to be executed. The definition of the 'skip' macro is

```
skip (cond(), B) ::=      α (~cond(), B; cond(), Δ);
```

whereby Δ represents the *empty workflow*. Any other macro rule can be specified either based on the fundamental constructs introduced in this section or on formerly defined problem specific constructs.

3.2 Semantics

After having introduced different kinds of control constructs we are able to clarify the notion of a workflow. Given that B and C are workflows, then

- -> (B, C)
- α (cond(), B; ~cond(), C)
- || (B, C)
- < (B, C)
- > (B, C)
- => (B, C)

are workflows, too.

In order to describe the semantics of the execution model we use state transition diagrams. The following states have to be introduced (A is a workflow the states listed below are associated with):

executed

Workflow A is executed. Note that execution happens atomically, has no timely extension and cannot fail. (This is only needed for the following simplified discussion of the model.)

disabled

Workflow A is not permitted to be executed.

blocked

Workflow A is currently not executable.

enabled

Workflow A is ready to be executed.

We assume that each workflow is in one of the four states *enabled, disabled, blocked, executed* (cf. Figure 6). By issuing the operation *enable()* a state transition from *blocked* to *enabled* happens. *disable()* sets the state of a workflow from either *enabled* or *blocked* to *disabled*. *execute()* transforms the state of a workflow from *enabled* to *executed*. *blocked()* sets a workflow from state *enabled* back to state *blocked*. The above mentioned operations are the only permissible ones and only applicable in the described situations. Setting the state of a workflow to *executed* causes an instantiation of the workflow. *disabled* and *executed* are final states.

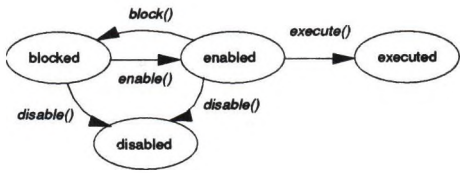


Figure 6: State Transition Diagram for Workflows (basic version)

Notice that the above introduced states are sufficient for showing the basic semantics of the model; for explaining implementation issues - the timely extension of workflow execution has to

be taken into consideration - we have to refine the state *executed* as well as the operation *execute()*. The latter operation must be split up into operations *start()* and *finish()* at least; it is recommended to add operations like *pause()* and *resume()* also in order to achieve a more handy system. The introduction of these new operations also requires the introduction of new states: *started, finished, paused*. Figure 7 depicts the extended state transition diagram for workflows. When the state *finished* is introduced *executed* is no longer a final state but *finished* replaces it.

Problem specifically more operations can be added. Also, error recovery has to be reflected in a more complete version of a state transition diagram for workflows.

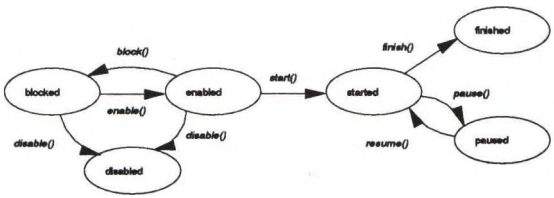
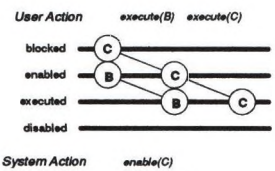


Figure 7: State Transition Diagram for Workflows (extended version)

To illustrate the semantics of the control constructs introduced in Section 3.1 we are going to show the eligible state transitions which can occur during processing the control constructs. Each semantics diagram consists of three parts: In the middle part workflows and their states / state transitions are depicted. At the top of a diagram user actions are shown while at the bottom part system workflows are described. If no user action is involved, system actions occur instantaneously. Otherwise, they are triggered by the user action(s) initiated in the same time slot. Time proceeds from left to right. In order to keep the diagrams simple and readable, alternative execution sequences are either depicted in separate diagrams or are indicated by alternative dotted state transitions in the same diagram.

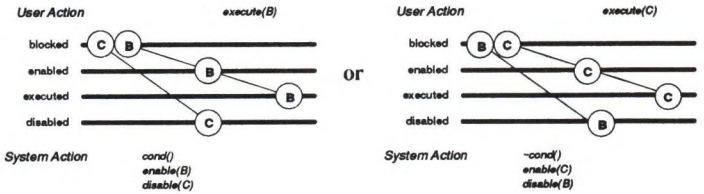
In the following, we assume a compound workflow A that consists of subworkflows B and C. B and C are going to be executed. In order to simplify the description we use the simplified state diagrams for workflows (cf. Figure 6). When all subworkflows of a workflow are either disabled or executed, i.e. they are in a final state, processing of a workflow terminates.

Serial Execution: ***-> (B, C)***



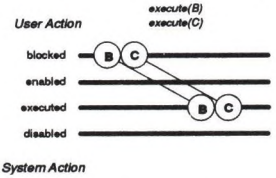
Initially workflow B is enabled. After having started B, C becomes enabled automatically. After C is processed, the workflow terminates.

Alternative Execution: α ***(cond(), B; ~cond(), C)***



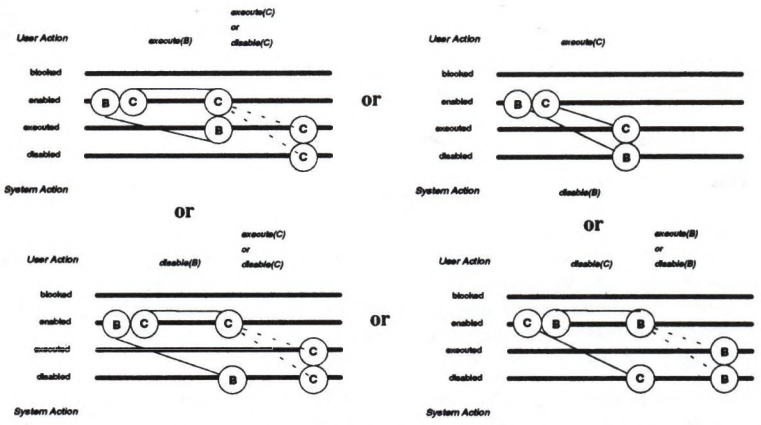
Firstly, B and C are blocked. Depending on condition cond() either B or C is enabled by the system automatically and will be executed eventually. The other workflow will be disabled automatically.

Parallel Execution: $\|$ ***(B, C)***



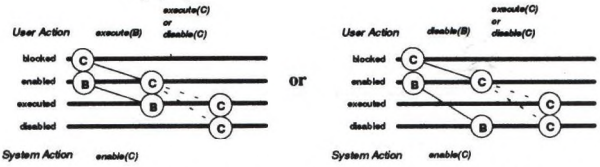
Either workflow B and workflow C must execute; however, they can execute independently.

Deadline: $<$ ***(B, C)***



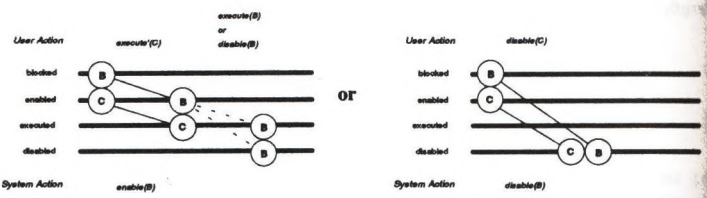
Both workflows B and C can be skipped by disabling them. When workflow C is executed at first, workflow B cannot be executed any more; it will be disabled automatically by the system.

Delay: ***> (C, B)***



The only but from a pragmatic point of view absolutely significant difference to the deadline construct is, that the delayed workflow cannot be executed without knowing that the delaying workflow has either been disabled or executed. Therefore, first B must be disabled or must have been executed before C can possibly execute.

Existence: ***=> (B, C)***



In case of existence conditions, a new feature of our model has to be introduced. The existence condition says that C has to execute if B is executed. That means: B can only be executed if either C was executed already or will be executed eventually. The existence condition does not imply any time order on the execution of both workflows. In order to capture this semantics our modeling means have to be extended. We introduce *execute'(.)* in order to express, that either a workflow has executed or is *promised* to execute eventually. Therefore, the case 'C executes before B' is justified in the SVES for existence (Figure 5). In this scenario, C has promised to execute eventually after B has finished.

3.3 Example

This subsection is to demonstrate a comprehensive example that consists of either prescriptive and descriptive elements. The overall purpose of the example is to set up a meeting (Figure 8); the whole process is initiated by a manager. First a preparing step has to be performed: a secretary is collecting dates about vacant meeting rooms. After that the potential participants of the meeting have to negotiate about date and location. After the participants have agreed upon date and location this room will be reserved by a secretary and the participants will be invited finally. Meeting data (MD) are exchanged between the workflows. They contain information about meeting place, date, and also about the potential participants.

This example shows that either prescriptive workflows - searching for a room and finally reserving it - and pretty loosely structured, descriptive workflows are needed simultaneously in the same scenario. The specification of the step 'negotiate' demonstrates how workflow type declarations are used in order to build the framework for a negotiation. For instance, if somebody is making a commitment, vague suggestion must not follow any more; if a question is asked, an answer must follow. Notice that the 'bubbles' used in the node 'negotiate' are representing workflow types and

not instances. An potentially arbitrary number of instances can be generated in order to solve the problem of agreeing on date and place for the meeting.

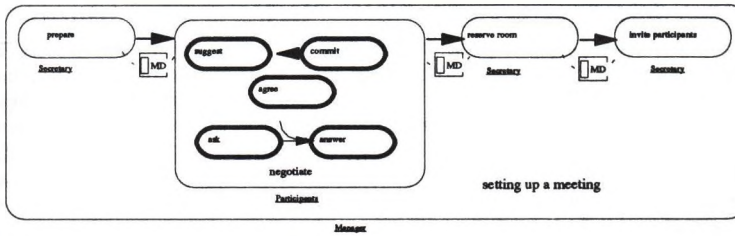


Figure 8: Example: Setting up a Meeting

4 The Organizational Perspective

4.1 Resources, Roles, Policies

So far, we have elaborated how a workflow is structured and how workflow execution is controlled. We will investigate organizational issues in this section. Particularly, we tackle the problem of *who* has to execute a workflow. In this paper only some fundamental issues are discussed; [5] and [6] detail organizational issues in the realm of process engineering in a very comprehensive manner.

Three major concepts are introduced which facilitates the assignments of actors to work, i.e. they determine who has to execute a workflow. The three concepts are *resources*, *roles*, and *policies*.

We presuppose an organizational database which captures significant and essential information. For instance, basic organizational entities like *employees, departments, business units* are described. Besides, information about the *interrelationships and interdependencies between organizational entities* is sustained. The *reports-to relationship* determining the manager-subordinate relation and the *belongs-to relationship* determining the department employees are associated with are two of the most fundamental ones.

All active elements (people, programs, robots, etc.) captured in organizational databases are called *resources*. They can execute workflows. Assigning resource(s) to a workflow denotes that the workflow has to be performed by this/these resource(s). Although feasible in principal, this solution is pretty inflexible. When people (resources) change their status, their assignments to workflows have to be revisited whether they are still valid or not. For example, they might change the group they are working for. To check assignments for validity is a very time-consuming and cumbersome task and also might cause severe integrity violations.

To overcome the drawbacks stemming from this static type of assignment, *roles* are introduced. A role is described by a certain set of *capabilities*. For example the role 'group manager' is characterized by the capability 'group authority'. An actual group manager who does have this capability is able to play the role 'group manager'. Roles are attached to workflows expressing that workflows can be executed by resources who are able to play the roles attached. In a specific scenario a workflow has to be executed by a role 'secretary'. All resources currently able to play

this role are permitted to perform the workflow. Role assignment to workflows is very often resistant to changes in an organization. For example, in case a person qualified as secretary leaves the company, workflow descriptions referring to the 'secretary' role do not have to be changed, because other persons are also able to play the role.

Sometimes even more tailorability and flexibility than provided by the role concept is required. Often assignments of roles to workflows depends on the context of actual execution. The context of a workflow is defined by all sorts of information elements that exist in the realm of a workflow (e.g. time, location, predecessor workflow). For instance, the workflow 'travel claim approval' has to be performed by the manager of the person filing the claim in case the amount of reimbursement does not exceed a certain amount; if this limit is passed, the travel claim has to be approved by the manager's manager, i.e. the procedure has to be executed by the manager's manager. Such assignments are called *policies*.

In order to sustain system dynamics, it is very important that workflow related issues and organisation related issues are specified independently. However, both pieces have to be interrelated eventually in order to specify who is eligible to execute a particular workflow.

4.2 Notification

In order to inform resources about work to do a *notification mechanism* has to be provided for. Notifications contain indications about *what* to do, *why* to do it, and *how* to do it, in order to make the execution context clear to the resource [5]. Notifications are organized in so-called *work-to-do lists*. Each resource is associated with one or multiple work-to-do lists. (S)he might maintain one work-to-do list for each role (s)he is able to play. Whenever an entry appears in a work-to-do list the associated action must be performed. Mostly this means to execute an application or to authorize, i.e. to initialize, a composite workflow.

When roles are assigned to workflows, normally more than one person is notified about work to do. On the other side not all work items are supposed to be executed by any person that knows about them, i.e. by anybody who has a corresponding entry in her/his work-to-do list. Thus, the notification mechanism must take care that these work items are *synchronized*, i.e. that the correct number of resources execute a particular piece of work (this might be 1 to n resources).

5 The Informational Perspective

The informational aspect of processes deals with data production and data consumption by workflows; thereby data flow in workflow networks is established. We will not detail the area of data management in this paper but refer to [13] for a comprehensive discussion.

In the area of data management for workflows two classes of data are distinguished: *control data* and *production data*. Production data comprise all data that are essential for an application area. For example, documents like travel claim forms or data about organizational context are production data. Control data are the minimal set of data which has to be exchanged between workflows in order to indicate against what data set a workflow should execute. In most cases control data are pointers to production data (e.g. to a document) like primary keys of relational tables (which point to a complex data record). In WFMSs only control data have to be dealt with; they identify associated production data which are management by a specialized data management system [13].

One of the most interesting effects of data management in the realm of workflow management is its impact on control. To illustrate this, we consider two workflows A1 and A2 which belong to independent (top level) workflows. Assuming both workflows have to access the same data element d exclusively, another interdependence between the two workflows is created. The exclusive data access enforces serialized execution of A1 and A2.

6 Related Work

In this section we cite some related work which covers a broad spectrum of different approaches. Of course, the comparison of these approaches to the approach introduced in this paper is sometimes pretty 'uneven', since not all approaches emphasize the aspects fundamental for our work. We do not intend to compare our approach to approaches that only tackle process modeling, but we want to compare to approaches which deal with process modeling and process execution as well.

The DISDES approach is presented in [23]. The model for an Organizational and Information System consists of Workflows, Processes, Positions, Persons, Users, Organizational Groups, among other things. These elements are aiming at the functional and the organizational aspects of process modeling. Although the principle approach is very promising, a bottleneck will be its limitation to a predefined set of objects available to reconstruct a problem space. In contrast, our approach allows to introduce and define arbitrary user defined objects (e.g. for expressing organizational issues or for defining behavior). For instance, the 'reports to' relationship seems to be the only one to relate objects of Organizational Groups in DISDES. If a particular application area requires another relationship between organizational objects, DISDES cannot support this.

Almost the same observation made for DISDES applies to ActMan [15]. ActMan nicely copes with the integration of existing applications into workflows; also the informational aspect of process modeling is considered. But both are tackled in the same static way as in DISDES. Besides no organizational aspects are considered. Also the behavioral aspect is very limited to prescriptive courses of processing.

The AMIGO workflow model is described as a model for Group Communication processes [8]. It represents a simple but powerful model for workflow management. AMIGO is lacking a structured way for workflow definition; nesting and reuse of workflows is not an issue. All control aspects have to be modeled by condition/action pairs which makes the definition of behavior pretty cumbersome even in simple cases (e.g. when serial execution has to be expressed). Also the logical correctness of the specification might be difficult to proof. From an organizational point of view, only roles are known. They have to be defined separately and independently for each workflow which might violate security [5]. The interoperability of workflows designed independently is also not possible since roles might be used differently in those workflows.

Another interesting approach to process modeling, CIMOSA, can be found in [20]. There the relationship between Domain Processes, Business Processes, and Enterprise Workflow is shown nicely. For modeling Enterprise Workflows - they are equal to workflows introduced in this paper - they are using procedural rules which describe the control flow between workflows. So called functional entities represent resources, who can executed pieces of work. CIMOSA does not allow to specify descriptive types of control flow. Specification of policies is only possible in a very limited manner.

A number of other approaches to WFM can be found in literature [10] [21]. However, many approaches which are called WFMS merely deal with the behavioral aspect of processes [4] [9] [24]. Besides, many approaches to extended transaction management are also put into the category of WFM [1] [11] [19]. We prefer to sustain a distinction between these approaches and WFMSs. We see very close relationships among these fields, but want to clearly separate them. We also agree, that there is mutual leverage among the fields as can be seen in [14].

7 Conclusion and Outlook

We detailed the functional and the behavioral aspects of a general approach to process modeling in the realm of WFM. Our experience with analyzing various problem domains and the feedback we are getting from several domain experts indicate that our approach is promising since it doesn't presume the customer's problem in a certain way but allows the user to express 'his/her language for problem solving'. This is accomplished through the avoidance of a preconceived user view on a problem domain and by the provision of a meta model which facilitates the definition of arbitrary problem specific object types.

We have implemented a prototype of a Workflow Management System that supports the process model introduced in this paper. We are now going to test the usability and the comprehensiveness of our process model in joint project with potential users of our WFMS.

References

[1] Attie, P.; Singh, M.; Rusinkiewicz, M.; Sheth, A.: Specifying and Enforcing Intertask Dependencies. *MCC Technical Report Number: Carnot-245-92*, 1992

[2] Beech, D.: Intensional Concepts in an Object Database Model. *Proc. OOPSLA*, 1988, pp. 164 - 175

[3] Bernstein, P.A.; Hadzilacos, V.; Goodman, N.: *Concurrency Control and Recovery in Database Systems*. Reading, MA, Addison-Wesley, 1987

[4] Breitbart, Y.; Deacon, A.; Schek, H.-J.; Sheth, A.; Weikum, G.: Merging Application-centric and Data-centric Approaches to Support Transaction-oriented Multi-system Workflows. *SIGMOD RECORD*, Vol. 22 (1993), No. 3

[5] Bussler, C.: Capability Based Modeling. *Proc. International Conference on Enterprise Integration Modeling Technology (ICEIMT)*, 1992, Hilton Head, SC, USA

[6] Bussler, C.: Jablonski, S.: An Approach to Integrate Workflow Modeling and Organization Modeling in an Enterprise. *Proc. Third IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE)*, 1994, Morgantown, WV

[7] Curtis, B.; Kellner, M.I.; Over, J.: Process Management. *Communications of the ACM*, Vol. 35 (1992), No. 9, pp. 75 90

[8] Danielsen, T.; Pankoke-Babatz, U.: The AMIGO Workflow Model, in: Speth, R. (Ed.): *Research into Networks and Distributed Applications*, Elsevier Science Publishers B.V. (North-Holland), Amsterdam 1988, pp. 227 - 241

[9] Georgakopoulos, D.; Hornick, M.F.; Manola, F.; Brodie, M.L.; Heiler, S.; Nayeri, F.; Hurwitz, B.: An Extended Transaction Environment for Workflows in Distributed Object Computing. *IEEE Data Engineering*, Vol. 16 (1993), No. 2

[10] Ghoneimy, A.; Hsu, M.; Kleissner, C.: An Execution Model for a workflow Management System. *Proc. Workshop on High Performance Transaction Processing Systems*, Asilomar, CA, September 1991

[11] Guenthoer, R.: Extended Transaction Processing Based on Dependency Rules. *Proc. RIDE-IMS 93*, Vienna

[12] Humphrey, W.S.; Feiler, P.H.: Software Process Development and Enactment: Concepts and Definitions. *Tech. Rep. SEI-92-TR-4. Pittsburgh: Software Engineering Institute*, Carnegie Mellon University, 1992

[13] Jablonski, S.: Data Flow Management in Distributed CIM Systems. *Proc. 3rd International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, March 1992, Lyon, France, pp. 65 - 78

[14] Jablonski, S.: Transaction Support for Workflow Management. *Proc.International Workshop on High Performance Transaction Systems,* Asilomar, CA, 1993

[15] Jablonski, S.; Reinwald, B.; Ruf, T.; Wedekind, H.: Event-Oriented Management of Function and Data in Distributed Systems. *Proc. 2nd International Working Conference on Dynamic Modeling of Information Systems*, July 1991, Washington, D.C., USA, pp. 107 - 127

[16] Käfer, W.: A Framework for Version-based Cooperation Control. *Proc. 2nd Symposium on Database Systems for Advanced Applications (DASFAA)*, Tokyo, Japan, 1991

[17] Kirsche, T.: A Database Query Language for the Processing of Preliminary Data in and between Cooperating Groups. in: Stucky, W.; Oberweis, A. (Eds.): *Proceedings BTW*, Springer-Verlag, 1993, pp. 196 - 205

[18] Kirsche, T.; Reinwald, B.; Wedekind, H.: Processing Dynamic Interactions in Cooperative Databases. *Proc. 27th Annual Hawaii International Conference on Systems Sciences HICSS-27*, 1994

[19] Klein, J.: Advanced Rule Driven Transaction Management. *Proc. IEEE COMPCON*, 1991

[20] Kosanke, K.: CIMOSA - A European Development for Enterprise Integration, Part 1: An Overview. *Proc. International Conference on Enterprise Integration Modeling Technology (ICEIMT)*, 1992, Hilton Head, SC, USA

[21] McCarthy, D.; Sarin, S.K.: Workflow and Transactions in InConcert. *IEEE Data Engineering,* Vol. 16 (1993), No. 2

[22] Meyer, B.: *Object-oriented Software Construction,* Prentice Hall, Englewood Cliffs, NJ 1988

[23] Reim, F.: Organizational Integration of the Information System Design Process. *Proc. 4th Conference on Advanced Information Systems Engineering*, 1992, Manchester, UK

[24] Sheth, A.; Rusinkiewicz, M.: On Transactional Workflows. *IEEE Data Engineering,* Vol.16 (1993), No.2

[25] Shlaer, S.; Mellor, S.J.: *Object Oriented Systems Analysis*, Modeling the World in Data, Prentice-Hall, Englewood Cliffs, N.J., USA, 1988

[26] Tsichritzis, D.C.; Nierstrasz, O.: Application Development Using Objects. Bullinger, H.J.; in: Protonotarius, E.N.; Bouwhuis, D. (Eds.): *Information Technology for Organizational Systems*, North-Holland, Amsterdam, 1988, pp. 15 - 23

[27] Yourdon, E.: *Modern Structured Design*, Prentice-Hall, Englewood Cliffs, N.J., USA, 1989

BUSINESS MODELS AND THEIR DESCRIPTION

Gernot Starke[1]

Abstract: *This paper presents a language- and tool-independent framework for business process modeling. It systematically evaluates potential payoffs and benefits of such models. A detailed and versatile process-driven cost model to determine the actual costs of business processes is introduced. This language-independent cost model contains a set of parameters which can be easily tailored to specific application domains. Finally a sample application of the cost model is presented.*

1 Introduction

Precisely defined and concise models of business processes are useful tools in the optimization of such processes. Driven by the need for improved quality and enhanced productivity at lesser cost, process optimization has become one of the buzzwords in nearly every industry.

In the first part of this paper the foundations of business process modelling are introduced, followed by an overview of modelling languages, paradigms and tools. The potential benefits of business process models are discussed. In the next section a language-independent and process-driven cost-estimation model is presented, which can be easily adapted to specific application domains. Such a cost-model represents a benefit of business process modelling which is often neglected in the process literature. Finally a typical application of business process modelling is shown, where a complex business process has to be re-engineered.

---

[1] Schumann Unternehmensberatung AG, Aachener Straße 1053-1055, D-50858 Köln, Germany, Phone (+49) (0)221 - 48 909 473, E-Mail: starke@nlk.schumann-ag.de

2 Business Process Modelling

The scientific discussion on modelling business processes, alias workflows, or development processes has increased significantly during the last few years. There is a steadily growing number of modelling languages, techniques, methods and systems, but the terminological foundation is still disputed. Furthermore there is only limited consensus on the goals and potential benefits of process modelling [12].

Due to those problems, different approaches to process modelling cannot be adequately compared to each other. Process model users are blinded by false hopes and expectations [10]. On the other hand potential benefits are not properly recognized. In a broader sense, different people talking about process models simply do not understand each other very well.

In the following section we will introduce a language- and tool-independent framework, which aims at conquering the aforementioned problems. At first we will present a set of basic process constructs, which constitute the *ingredients* of process models. To facilitate description of different abstraction levels, we then discuss the relation between *generic* and *instantiated* process models. For a more detailed introduction to process modelling see [2] and [13].

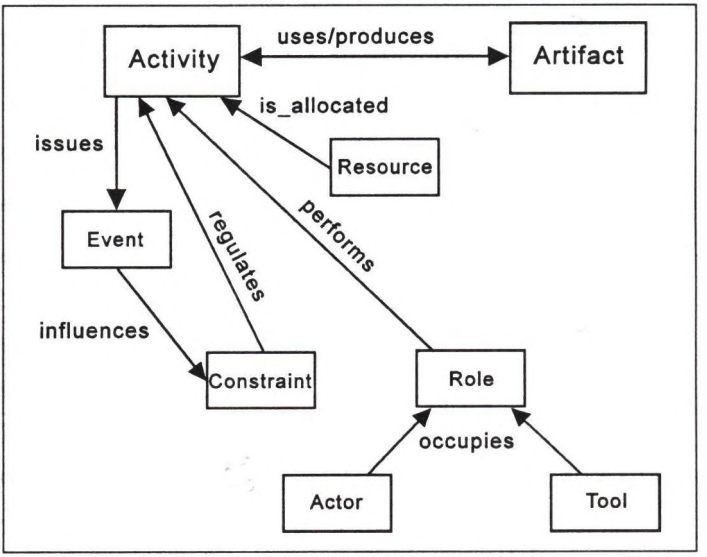


Figure 1: Basic Process Modelling Concepts

2.1 Ingredients of Process Models

Figure 1, taken from [13], gives a graphical overview of the basic process modelling ingredients and their relations. The following table gives brief explanations and presents some examples.

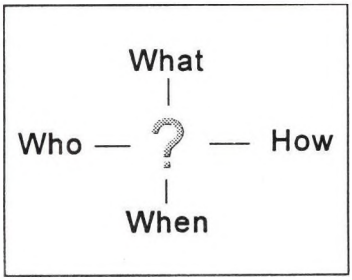
Construct	Remark	Examples
Activity	Comprises everything which is *done* or *performed* within the process.	Sending a letter, visiting a client, filling a credit-application form
Actor	Appears only in instance-models	Mr. Clinton
Artifact	Every product or document produced or consumed during process execution	product specification, invoice credit-application
Constraint	Every condition influencing process execution	activity must be started before May 1st.
Event	Differentiated between external and internal events. Internal events are issued by activities, artifacts, roles or resources. They denote any change-of-state	activity-finished, activity-started, letter-written, letter-sent, printer-out-of-paper
Resource	Entity passively utilized during process execution	person-time, computing-time, money
Role	Contains ability and responsibility for execution or support of activities	project-manager, financial controller
Tool	Possibly active entity	compiler, text-processor

Table 1 Basic Process Model Ingredients

Most commonly, business process models present answers the key process questions:

„*What* should be done, *how* and *when* to do it, by *whom* shall it be done?"

The basic terminology from figure 1 is related to those questions in the following manner:



- „What?" What *artifacts* are used and produced throughout the process?
- „How?" How are the *activities* performed and which *resources* are used?
- „When?" *Events* and *constraints* control and regulate execution of activities.
- „Who?" *Roles*, *actors* and *tools* perform the activities.

Concerning the execution of process models, *constraints* and *events* are just one possibility of incorporating explicit control. Alternative solutions include events/triggers, rule-systems or semaphores. From a formal viewpoint these systems to formulate control behave similar, but we found the constraint/event system to be most understandable, also for process model users. Constraints and events are being applied in systems modelling languages, like structured analysis [15]. The concepts can be easily mapped to arbitrary modelling languages.

2.2 Generic versus Instantiated Models

Models of business processes differ in their level of abstraction.

- *Generic* models are templates, describing a *type* or *class* of similar processes. Generic models describe the overall structure of business processes. They do not contain project specific information, like names or resource allocations. An example is Evaluate-Credit-Application in a bank.

- *Instance* models represent single business processes. Instance models describe specific business processes in detail. They are derived or instantiated from generic models and contain project specific information. Instance models are used to support process execution. An example is Evaluate-Smith-Credit-Application, which is derived from the generic evaluate-credit-application mentioned above.

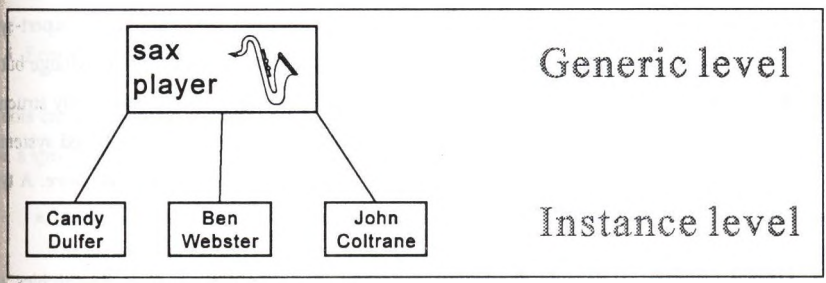


Figure 2: Generic model versus instance model

Both kinds of models are abstractions of real processes. A generic model usually contains *type variables*, e.g. roles, which are bound to existing persons within *instance models*. Generic models can be regarded as analogous to *types* in conventional programming languages or *classes* in object-oriented languages. Generic models are used for a-priori analysis of processes. Instance

models on the other hand allow for a-posteriori analysis, which can provide feedback for process optimization. Such feedback will often result in modification or adaptation of the corresponding generic model.

2.3 Process Modelling Languages and Paradigms

As already mentioned above, several paradigms exist to model business processes. The most commonly used are briefly described below.

- **Imperative** approach: based on conventional programming languages like Ada. Overview and understandability of models is limited and expression of control and data-typing is strong. No language constructs are available to describe agents, tools and resources. Imperative lack the ability to visualize models in a graphical manner. A typical representative is Arcadia [14].

- **Functional** approach: based on the λ-calculus, similar to languages like Lisp. Functional languages focus on activities. Documents and products are treated as arguments or parameters to functions. Control is expressed via recursion, data type modelling is usually weak. The overview and understandability of functional models is limited, due to the lack of graphical and hierarchical representation mechanisms. This disadvantage is similar to the imperative languages. A typical representative is the language CoShell, embedded into the ProcessWeaver system [3].

- **Rule-based** approach: based on *if-condition-then-action* rules, often applied in expert-system construction. Rule systems are well suited to express application-specific knowledge but lack the possibility to describe data or explicit control. As rules cannot be hierarchically structured, rule-based models tend to become bulky and difficult to comprehend. Rule-based systems are sometimes combined with other modelling approaches to serve as *knowledge store*. A typical representative is Marvel [9].

- **Object-oriented** approach: modelling processes by constructing class hierarchies using inheritance and polymorphism is a current research topic in systems engineering. Overview and understandability of such models is high and re-use of models is very easy. There exists only limited experience in the application of object-oriented modelling techniques to business processes. Typical representatives are Objectory [6] and Coad/Yourdon OOA/OOD [1].

- **Graph- or net-based** approach: based on Petri-nets or similar graph structures, they provide graphical overview and good understandability of models. Graph-based approaches are most widespread in commercial workflow-management systems, as they provide excellent support for model execution. They usually concentrate on activity modelling, therefore the data aspects are less elaborated. Typical representatives are ProcessWeaver [3] and Staffware [11].

- **Structured** approach: based on structured system analysis techniques like structured analysis [15]. These approaches provice good overview and understandability due to graphical notation. Hierarchisation of system models allows description of arbitrary abstraction levels, facilitating communication about models. The integration of functional and data aspects is very tight, resulting in concise and complete models. As structured methods have been successfully used in system analysis for several years, modelling experience and knowledge exists. This increases the acceptance of structured methods. A typical representative is the language Pro-SA [13]. Structured languages can also be called net-based, as they consist of different kind of flow-diagrams, which are specialized forms of nets or graphs.

Several alternative approaches are mentioned in literature, which have only achieved limited interest: decision-based systems [7], system-dynamics [4], entity-process-models [5] and so forth. Reference [13] contains a detailed technical discussion of over 30 different languages and tools.

2.4 Process Support Tools

Tools can be roughly divided into four different categories, which are not necessarily disjunct, i.e. a specific tool may fall into more than one category.

- **Modelling** tools allow business processes to be modelled, preferably by using interactive graphical editors. This category is partly covered by CASE-tools and diagram-editors.

- **Simulation** tools allow models to be simulated or animated. Process behaviour can be studied and possible weaknesses can be detected before the real processes are performed.

- **Analysis** tools allow detailed analysis of business processes. Some tools detect deadlocks, race-conditions or similar problems. Analysis tools can also outline critical pathes and resources.
- **Execution support** tools aid the business process user within execution of the real process by providing tool support. The common term for this tool category is *workflow management tool*.

Instead of presenting a comprehensive list of commercially available tools, which would be out of the scope of this paper, a proposition for a list of tool requirements is given, which can be regarded as minimal qualifying factors for practically applicable process tools. This list should include multi-platform (hardware and software) availability, heterogenous LAN and WAN communication, client-server support, interfaces to arbitray applications, graphical user interface, sophisticated security concept, support of several different commercial database systems

3 Potential Benefits of Business Process Models

Developing and maintaining business process models is a difficult, time-consuming and cost-intensive task. Business process engineering as a new technical discipline will only be accepted in industrial practice if substantial benefits or payoffs can be reached, which will not be achieved by other means. Potential benefits have to be systematically analyzed and evaluated beforehand to avoid unfullfillable expectations. In [10] the disappointment of top-management executives concerning business process reengineering is described. If the goals and potential benefits of business re-engineering projects would have been systematically analyzed before, such *false hopes* would probably never have been raised.

The reason for this procedure is a historical one: especially in computer science several *buzzwords* have come up like shining stars in research and practice during the last few years, which could by no means fulfill the overly high expectations of users. Artificial intelligence is only one example for this phenomenon.

The following paragraph gives an overview of the most important potential benefits of business process modelling. A detailed discussion is presented in reference [13].

- **Quality**: the ultimate goal is to improve the quality of products or services in order to satisfy user and customer needs. An example is the fulfillment of the ISO-9000 quality standards, which is considerably harder without applying defined process models.

- **Productivity**: either more output produced or less time needed, a key factor for every market-oriented enterprise.

- **Process execution support**: process models allow supportive tools to be incorporated into process execution.

- **Understandability**: a clearly defined process model enables every process participant to understand what is done, leading to improved motivation and acceptance.

- **Concentration on technical issues**: process models free process participants from the need to discuss process concerns. They can concentrate on producing high-quality solutions instead of wasting time negotiating a common process.

- **Explicit process knowledge**: domain-specific know-how is conserved within process models. Processes become repeatable. This aspect becomes vital in case key personnel is transferred or companies are restructured.

- **Integration**: process models have integrative effects. Different sub-models (like project management or quality control) can be incorporated. Process models help to avoid *media breaks*, which are otherwise difficult to handle.

- **Communication** and **cooperation**: by providing a common terminological basis, process models facilitate communication and cooperation between process participants.

- **Cost estimation**: process models provide means to estimate financial effects of process changes and are therefore a managerial instrument to control process optimization.

The aspect mentioned at the end of this list, cost estimation, is detailed in the following section. The reason to focus on cost estimation is that this discipline, although of high importance in industry, has long been neglected by the research community. Experiences from industrial process model users show that the financial consequences and benefits of those models have to

be demonstrated and proven in advance. With the cost model presented in the next section the achievement of this crucial goal becomes realistic.

4 Business Process Cost Estimation

One of the prime goals of every competitive, market- and customer-oriented enterprise is the estimation of the *cost* of any business process conducted within their realm. Cost estimation and determination is therefore an important managerial instrument, the results being equally interesting for process model users and process modellers.

With a business process model plus the corresponding cost information at hand, user and modeller become able to evaluate and quantify the benefits of re-engineered, modified or optimized processes. For process modellers or tool vendors such a cost model will therefore serve as the prime argument to introduce workflow management tools and related technology.

Two approaches to determine these costs are most widespread:

- A *posteriori* estimation or determination of all factors influencing the costs. With such instruments, for example bookkeeping, the *prediction* of costs is difficult. They are well-suited to determine the overall price of any product or service *during* or *after* production time. They cannot be satisfactorily applied in optimizing business processes, as they do not provide means for *what-if* analysis.

- A *priori* estimation. Very often this in-advance-estimation is done by simple guessing, where the vast experience of the estimators make their guesses accurate, although they are not repeatable, nor can they be proved or validated in any way.

Both methods are neither suited to evaluate, least to optimize business process. A precisely defined cost estimation model, integrated into the business process model is needed to accurately estimate the overall cost of complex, interwoven and partly iterated office tasks. Such a cost

model will will help to enhance the quality of business process models, as it allows for process optimization. Financial effects of process changes become immediately visible.

Alternative processes can be compared with respect to their financial effects without implementing these processes. Simulation tools can be applied to drive such optimization strategies, often coined *business process re-engineering*.

4.1 Process-Driven Cost Model

The cost-model is composed of a set of cost-parameters associated with the basic process ingredients, which have been introduced in section 2.1. Table 2 shows the parameters of the cost model. Those parameters can be seen as attributes to the corresponding process entities.

As the model itself is completely language- and tool-independent, it is not expressed in a closed formula, allowing it to be easily adapted and adjusted to domain-specific needs.

4.2 Tailoring the Cost Model

The cost parameters should be viewed as dynamic and flexible. In specific application domains, like insurance or banking, the set of parameters presented in Table 2 might be sufficient, whereas in research&development departments several new categories of cost parameters might be needed. The author is well aware that the model presented here can therefore only serve as a *foundation* of a practical and applicable cost model.

To tailor this cost model with an existing process modeling language to suit domain-specific needs, the following steps have to be performed:

1. The language constructs representing the basic process constructs have to be enriched by the cost parameters as *attributes* or *annotations*.
2. The cost parameter values or variables within the generic model have to be determined. In case of an arbitrary activity this could lead to an expression like: „*Every minute of executing this activity costs 5 monetarial units*". Here it will be necessary to estimate certain parameters, but the estimates have to be repeatable. Due to the nature of the basic process constructs, the appropriate parameters or attributes are often measurable.

3. The cost parameter values for the instance model have to be determined. Here it becomes necessary to measure, calculate or estimate numerical values, for example activity durations.

Concept	Cost parameter	Description/Examples
Activity	initialization	cost of initializing the activity.
	duration	activity cost per duration or production unit.
	de-initialization	cost of de-initializing the activity.
Actor	initialization	initial training costs. Relevant only once for every actor.
	enabling	cost of enabling the actor to execute a certain activity, e.g. travel, accommodation.
	execution	cost per duration unit of this actor executing an activity.
Artifact	initialization	cost of initializing this artifact, e.g. opening a new file for a given project.
	storage	cost of storing this artifact. Probably varies with age or size of artifact (large document is more expensive to store than small one).
	handling	cost of handling the document, e.g. transporting it between departments. This parameter is especially important, as it covers the time it takes to transport artifacts.
	de-initialization	cost of de-initializing this artifact.
Event	initialization	cost of initializing an event.
	execution	cost of sending the event to the appropriate activity, e.g. notifying the project manager.
Resource	availability	cost of making a resource available within the process, e.g. writing appropriate contracts.
	initialization	cost of initialization of resource within the process, e.g. inserting paper into a printer.
	consumption	cost of resource per unit, e.g. sheet of paper.
	de-initialization	cost of de-initializing this resource, e.g. taking application-specific paper out of a printer.
Tool	availability	cost of making a tool available within the process, e.g. obtaining a computer
	initialization	cost of initializing the tool, e.g. transporting the slide projector into the conference room.
	execution	cost per duration or production unit of this tool.
	de-initialization	cost of de-initializing this tools, e.g. cleaning it.

Table 2: Cost Model Parameters

The cost estimates obtained with this simple cost model will demonstrate the financial effect of any process modifications immediately. In case a process simulation tool is used, the cost estimates can be automated.

5 Sample Application of the Process-Driven Cost Model

A promising application of the process-driven cost estimation is the introduction of modern costing systems, e.g. the *target-costing* systems currently successfully used in many Japanese corporations.

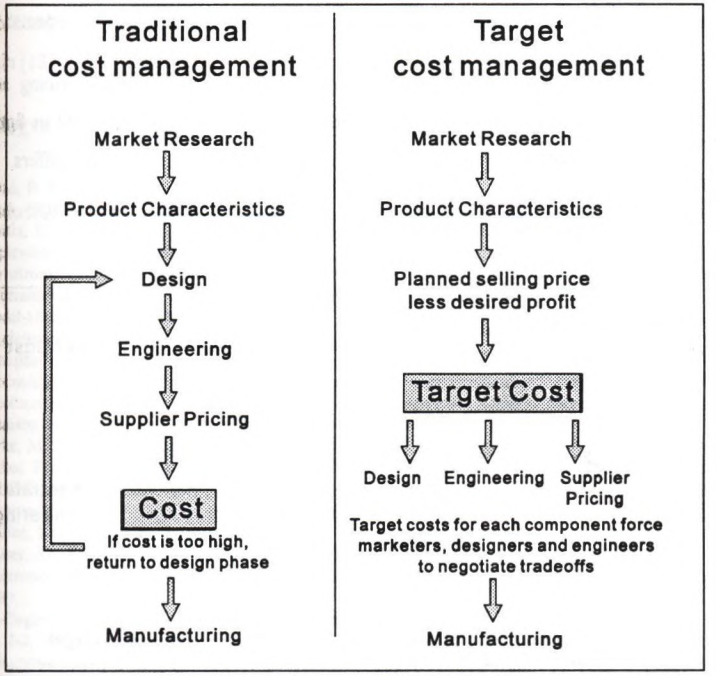


Figure 3: Traditional versus Target cost management (from F. Worthy: Japan's smart secret weapon, in „Fortune", August 1991, p. 49).

Figure 3 shows a comparison between the traditional Western-American cost management system and the Japanese target-costing system.

The transition from the traditional system to the target-costing system is extremely difficult due to the different philosophies underlying both systems. A defined business process model can

facilitate this fundamental process modification by providing understandability, overview and a basis for communication and cooperation. Especially within the negotiation phase in the target costing system this becomes vitally important.

This problem serves as an excellent example of a process *re-engineering* problem. The cost model introduced in the preceding section can support this process modification by providing a structured *what-if* analysis of the two different possibilities, clarifying potential financial consequences of both models. Please note that these crucial consequences can be determined in advance.

The formal process models underlying figure 3 differ in several fundamental aspects:

- The input parameters for the design, engineering and supplier pricing activities are different. The *design* activities of both process models are depicted in figure 4. From that illustration it can be seen that the output of those activities also differs.
- A new activity type has to be introduced in target-costing model: *determine-planned selling-price*

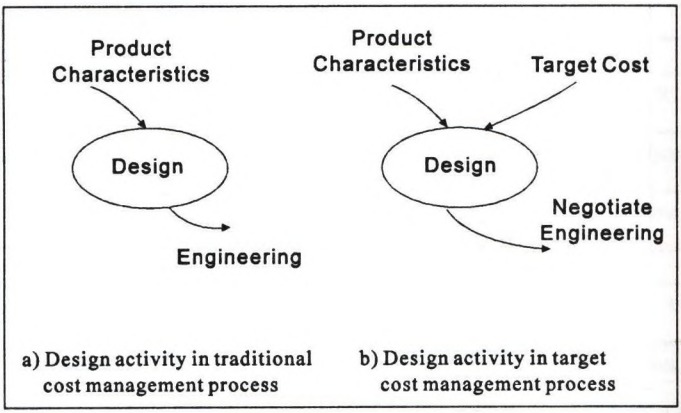


Figure 4: Different design activities

6 Conclusion

Instead of summarizing this short paper the author shall warn process enthusiasts that this technology is still in its infancies. There exist severe discrepancies between user expectations and

corresponding tool functionality. Unsatisfying interfaces between different process models cause expensive and dangerous *media-breaks*. No international process modelling standard exists which could focus tool developers onto a common terminological basis.

Despite these problems, process engineering is still a promising discipline. If it is applied with the required carefulness, its benefits will outweigh its disadvantages.

The greatest and most urgent problem today seems to be the lack of process modelling experience. It is not always obvious whether disagreeable models arise from a bad modelling language or from a lack of modelling experience. From that point it seems natural to concentrate on established modelling languages like *structured analysis* [15], an approach which is also favored in [13].

7 Bibliography

[1] Coad, P. & Yourdon, E. Object-Oriented Analysis. Yourdon Press, Prentice Hall, Englewood Cliffs, New York, 1990.

[2] Curtis, B., M. Kellner & J. Oliver. Process Modelling. Communications of the ACM, Vol. 35, Nr. 9, September 1992, pp. 75-90.

[3] Fernström, C. PROCESS WEAVER: Adding Process Support to UNIX. Proceedings of the Second International Conference on the Software Process, Februar 1993, Berlin, pp. 12-27.

[4] Abdel-Hamid, T.K. & S.E. Madnick. Software Project Dynamics: An Integrated Approach. Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[5] Humphrey, W. & Kellner, M.. Software Process Modelling: Principles of Entity Process Models. In: Proceedings of the 11th International Conference on Software Engineering, May 1988, pp. 331-342

[6] Jacobson, I., M. Christerson, Patrik Jonsson & G. Övergaard. Object-Oriented Software Engineering. Addison Wesley, 1993.

[7] Jarke, M., M. Jeusfeld & T. Rose. Process Management in the DAIDA Environment. In: T. Katayama, Editor. Proceedings of the 6th International Software Process Workshop, Hakodate, Japan, October 1990. IEEE Computer Society Press., pp. 117-120

[8] Jakubczik, G.D & Skubch, N. Business Process Reengineering: Maßstab für den langfristigen IV-Einsatz. Online, No. 4, 1994 (in German).

[9] Kaiser, G. Rule-based Modelling of the Software Development Process. In: Proceedings of the 4th International Software Process Workshop, May 1988. ACM Software Engineering Notes, 14(4), June 1989.

[10] Re-Engineering: Topmanager sind von Resultaten enttäuscht. Computerwoche, No. 24, 17th June 1994, pp. 1-2, IDG Publishing, Munich (in German).

[11] Schumann AG: Staffware. Internal technical report, Schumann Unternehmensberatung AG, Aachener Str. 1053, D-50840 Köln, Germany, 1993.

[12] Starke, Gernot. Why is Process Modelling so Difficult?. In: B.Warboys, Editor. Proceedings of the Third European Workshop on Software Process Technology, Grenoble, Frankreich, Februar 1994. Springer Verlag, LNCS Nr. 772, pp. 163-166

[13] Starke, Gernot. Sprachen zur Software-Prozeßmodellierung. Doctoral thesis at Lehrstuhl für Systemtechnik, J.Kepler Universität Linz, 1994 (in German). Shaker Verlag, Aachen, 1994.

[14] Taylor, R., F. Belz, L. Clarke & L. Osterweil. Foundations for the Arcadia Environment A rchitecture. In P. Henderson, Editor. Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, pp. 1.13. Also published as ACM SIGSOFT Software Engineering Notes, 13(5), November 1988

[15] Yourdon, E. Modern Structured Analysis. Prentice-Hall, 1989.

DYNAMIC VALIDATION OF WORKFLOW PROCESS MODELS BY GRAPHICAL ANIMATION

Ralf Staenglen[1]

Abstract

Creating workflows by first modeling them using graphical, CAD-like tools is becoming more widespread [2]. Unfortunately, such models often have logical errors that cause the resulting workflows to behave differently from how they should. It is difficult and expensive to find these errors during runtime, so we developed a tool that animates a workflow model and so lets the modeler validate its dynamic behavior. Our tool uses a rule-based system implemented using IBM Prolog/2, and is currently a component of FlowMark/2, IBM's workflow management product.

1. Introduction

Often a new process model contains bugs. They occur when definitions in the model conflict with each other or with the intention of the workflow, or when data is missing. Process animation lets you find such errors by imitating the execution of the process being modeled. It lets you move through the process step by step, exploring various paths and imitating the behavior of people and programs in the process. You can animate a workflow model at any time during its development; the process does not have to be complete and the application programs normally attached to activities in the process are not needed.

To understand this article, it will help to know the modeling constructs used by IBM FlowMark/2 [1,5]. For an overview of the underlying mathematical formalism, see Leymann [3,4].

---

[1] IBM Deutschland Entwicklung GmbH, P.O. Box 1380, D-71003 Boeblingen
E-mail: r_staenglen at vnet.ibm.com

1.1 Basic Modeling Constructs

A workflow model reflects the process flow and the staff organization. The animation facility depicts both. The following discussion focuses on the process organization.
A workflow model is a complete representation of a process and contains the following elements.

Activities: An activity represents a fundamental business action. FlowMark/2 distinguishes among the following types of activities:

Program (elementary) activity: A program activity is the smallest unit of work in the workflow. It can be performed automatically by a plug-in program that does not require user interaction or manual interaction by a person assigned to the activity.

Activity block: An activity block combines activities on a horizontal level. This allows hierarchical modeling and loops.

Process activity: A process activity calls another process. Process activities not only provide another way to organize processes hierarchically, but offer a way to reuse existing processes. They let you use the same process several times in parallel.

Each activity has a start and an end condition. The start condition determines when the activity can be started. The exit condition determines when the activity is to be regarded as successfully completed. An exit condition can be any Boolean expression.

Data Containers: Data containers offer the ability to exchange navigation relevant data between activities. Each activity or process can have an output and an input container assigned to it.

Data Connectors: Data connectors connect the output container of an activity with the input container of another or the same activity. They define the flow of

navigation relevant data. Data needed by the application typically do not flow along data connectors, but remain encapsulated in the application data base.

Control Connectors : Control connectors model the potential flow of work within a workflow model. They connect activities to define a sequence of work. Each control connector has a transition condition assigned to it. A transition condition can be any Boolean expression derived from the output data of the activity left by the control connector. This lets you model context dependent flow of control within a workflow.

2. Concepts of process animation

To validate a model using animation you do not need the programs that are attached to program activities. Instead, you imitate such programs by entering their output data manually. A process need not be complete or error free to be animated. This lets you validate models in early stages of development. When talking about validation of a process model, we should distinguish among three different scenarios:

2.1. Finding logical errors in the dynamic behavior of a process

In this case, a stepwise navigation through the process model is needed. Examples of errors are: the data transferred do not meet the conditions of the control flow or the data needed by another program are not correctly transferred or the sequence of tasks makes no sense - for example there is an activity that may never end.

For example, imagine the process behind an automatic banking machine. There is an activity that involves checking if your bank card has been stolen by verifying the PIN number. The process modeler who modeled the condition between the check and the release of money might have modeled: "Give out the money if the PIN number is wrong". The condition is syntactically correct and the process can be executed, but is obviously wrong. A static checker will not detect such an error, but with process animation you see the process propagating along the wrong path.

2.2 Regression testing

You can save a sequence of navigation steps and reuse it later for regression testing.

2.3 Validation of the process

You can use animation to demonstrate the process to the organization that plans to implement it. The animation is like a movie that shows the propagation through the model.

Animation addresses all three scenarios.

3. The Animation facility

Figure 1 shows an example of an animated process model. During animation, the tool shows the same graphical view as in the modeling component *(central window in fig.1)*. The wheels indicate activities, the solid lines describe control connectors and the dashed lines describe data connectors. The bars upon the wheels show the activity status. Connectors also change their colors depending on state (unevaluated, true, false). The window shown in the left lower corner is an imitated program. You are asked for output data (credit amount).

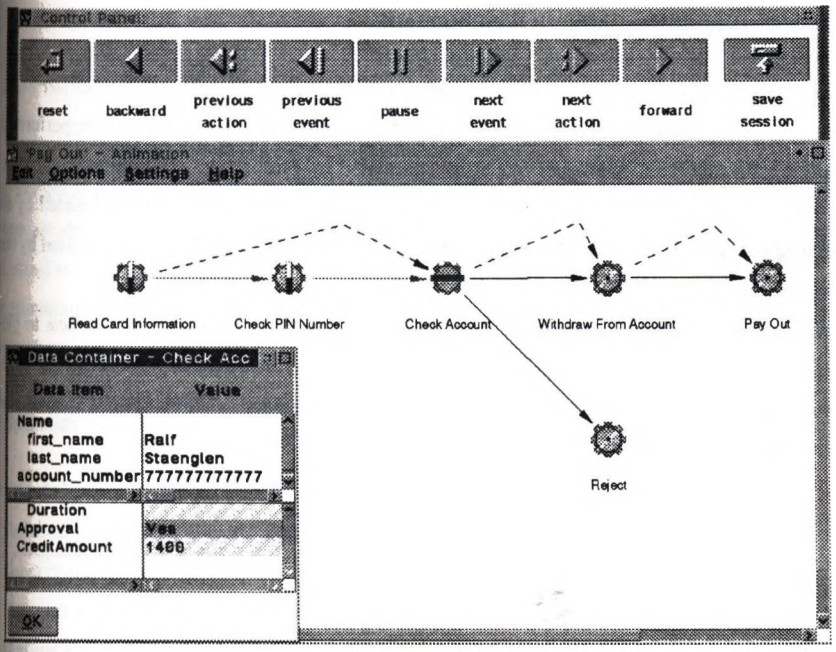


Figure 1: Example for an animated process model. The window in the middle shows the animated process graph. The window on the left lower corner is an imitated program. The upper window shows the control panel.

You can navigate through the model with a VCR-like user interface, which has buttons for forward, next user action (single frame), last user action and backward *(upper window in fig.1)*. Animation sessions can be saved for later use. This is helpful for continuing work at the same position it was stopped or to replay a complete sequence. If information needed to navigate through the model is missing, you are prompted that information, for example: program data or the state of a\n erroneous condition. Status changes, errors and other useful information are written to a log.

4. Implementation

We used Prolog/2 to implement the animation system. This had the following advantages:

- Fast development
- Developers could concentrate on the logic concepts instead of the details of physical implementation such as memory management.

4.1 Architecture of the animation component

The workflow metamodel is stored in the animation rule base as set of descriptive rules *(see Fig3)*, for example 'control connectors enter activity'. A modification in the metamodel can be performed by changing the rule base. A process model is represented by Prolog facts containing the specific model information *(see Fig.2)*. For example, 'control connectors enter activity' is represented by an identifier of the activity and identifiers of the entering control connectors. The facts are used by the descriptive rules mentioned above.

The tool creates an instance of a process model by copying the contents of the Prolog facts to global terms. To each instance, it assigns a set of global terms. Only instances of objects having an instance in reality can be created. Instances of a process are necessary to allow several instances of the same process to be used by the same parent process.

```
Facts
activity_fact("76", 'Check Account', manual, automatic,...)
control_connector_fact("19", conditional, "76" , ...)

Rule
control_connector_enters_activity(CCONN_ID, ENTERING_ACTIVITY_ID) <-
        control_connector_fact(C_CONNECTOR_ID, *, ENTERING_ACTIVITY_ID, *, *).
```

Figure 2 Examples of Prolog facts and rules

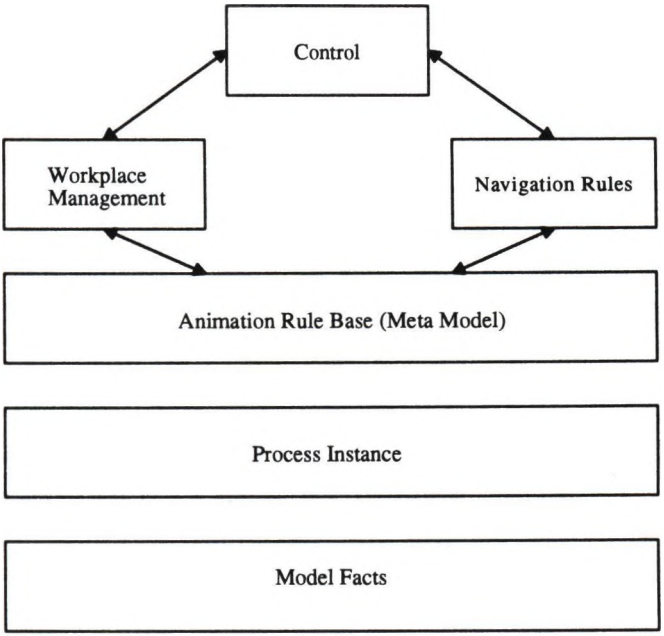


Figure 3 Architecture of the animation component

The navigator processes the graph. The navigator is implemented as an event handler. Navigation events are described by Prolog terms. The navigation events occurring during forward animation are stored so that the user can retrace the process flow stepwise.

An animation control component manages all animation relevant tasks (messages from user input, navigation ,...) and a workplace management component handles the modeler' s workplace.

5. Summary

Process Animation satisfies two needs:

1. The need to check in an early stage of process development if the process will address the customer requirements.
2. The need to detect logical errors in the process dynamics can be performed without using the runtime environment.

Acknowledgement
I appreciate Craig Blaha for reviewing this paper.

References:

[1] IBM FlowMark for OS/2 Modeling Workflow Release 1.0, Vienna 1994

[2] CAFASSO, R. IBM shows its flow cards, Network World, 11(7) 1994 1 and 64.

[3] F. LEYMANN and W. Altenhuber, Managing business processes as an information resource, IBM Systems Journal 33(2) (1994).

[4] F. LEYMANN. A meta model to support the modeling and execution of processes. Proc. 11th European Meeting on Cybernetics and System Research EMCR92 (Vienna, Austria, April 21-24 1992), World Scientific 1992, 287-294.

[5] F. LEYMANN and D. ROLLER. Business Process Management with FlowMark. Proc. compcon spring 94 (San Francisco, February 28 March 4) 1994, IEEE Computer Society Press 1994, 230-234.

Interfacing with the User

Benefits of Groupware in Software Engineering Environments

Paul Grünbacher*

Abstract *Current CASE products and software engineering environments provide little support for workgroups and teams. The development of software, however, is a highly social and creative activity involving frequent interactions between developers. Integrating systems providing support for group activities and interactions ("groupware") and of software engineering environments (SEEs) promises benefits for the overall software life cycle. This article aims to (1) give an overview about the field of CSCW and groupware (2) present groupware applicable in different stages of the software life cycle with focus on approaches to capture the design rationale (3) propose strategies to integrate both groupware tools and software engineering environments.*

Keywords *groupware, CSCW, design rationale, CASE, software engineering environment (SEE), process model*

1 Introduction

Empirical studies prove that typical software engineers spend between 30% and 60% of their working time communicating with members of the development team [2, 11, 18, 28]. The development of software is therefore a highly social activity involving frequent interactions between the members of the development team. Adequate communication support is thus a real need for effective software production. But while the main focus of present CASE products and software engineering environments is the support of software engineering tasks, communication and group processes are still supported insufficiently. CSCW technology [6, 9] aims at providing support for workgroups. The integration of systems providing support for group activities and communication (CSCW technology,

*Institute of Systems Sciences, Dept. of Systems Engineering & Automation, Johannes Kepler University Linz, Austria

groupware) and of software engineering environments (SEEs) promises benefits and is a challenging field of research.

This article is structured as follows: Section 2 gives a survey about the field of CSCW and groupware. Section 3 describes groupware applicable in different stages of the life cycle focusing on cooperative design tools and derives implications and benefits for the software development process. Section 4 identifies ways to integrate groupware tools into software engineering environments.

2 CSCW and Groupware

While traditional software systems support the interaction between a single user and the system, *groupware systems* aim at providing support for workgroups and teams. The notions of CSCW and groupware have rapidly become catchwords in recent years. The term *groupware* was introduced in 1982 by *Johnson-Lenz* [13]. The term CSCW (an acronym for Computer Supported Cooperative Work) was coined in 1986 by *Greif* and *Cashman* [9].

Ellis et al. define groupware as

> computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment [6].

It is interesting to remark that different terms have been established for similar concepts in the research communities of environments and CSCW [32]. *Schefström et al.* remark that the corresponding notions for 'shared environment' and 'common task or goal' in the CSCW community are 'information management' and 'process management' in the environment community. See more on the intersection of CSCW and process modeling in [25].

2.1 The Evolution of Group Support

A short look at the evolution of workgroup support in the past 30 years presents an interesting development [8, 10]. In the 1960s mainframe computers first allowed 'corporate computing'. The introduction of the personal computer established 'personal computing' in the end of the 1970s. The spread of personal computers and LANs gave way to 'group computing'. Groupware technology can thus be viewed as a technological link between corporate computing and personal computing.

In 1989 *Alan Kay* characterized a scenario for software yesterday, today and tomorrow (see table 1). The first users of computer systems were experts in machine rooms. The personal computer and

Table 1: **Scenario for Software of today, yesterday and tomorrow** [17]

Function	*Yesterday*	*Today*	*Tomorrow*
Where	Machine Room	Desktop	Wherever
Who	Expert	Individuals	Collaborative Groups
What	Edit	Layout	Orchestrate
How	Remember and type	See and point	Ask and tell
Style	Data / procedures	Object oriented	Rules

Dimensions	*Same Time*	*Different Times*
Same place	face-to-face interaction	asynchronous interaction
Different places	synchronous distributed interaction	asynchronous distributed interaction

Figure 1: **Dimensions of Groupware [6]**

the concentration on the field of Human-Computer-Interaction allowed individuals to work with the computer desktop. Collaborative groups with geographically distributed members are tomorrow's users of computer systems.

2.2 Dimensions of Groupware

In order to structure the large number[1] and variety of groupware systems the following dimensions are frequently applied (see figure 1) [17, 19]:

Time: Group interactions occur at the same time *(synchronous interaction)* or at different times *(asynchronous interaction)*.

Place: The participants of a groupware session either meet face-to-face (same place) or they have different physical locations (different place).

Task: Groupware systems provide support for different types of group processes and communication needs (see table 2).

[1] In his report "The *unofficial* Yellow Pages of CSCW, Groupware and Prototypes" *Malm* lists 314(!) projects, products and prototypes in the field of CSCW and Groupware (see table 2)

Table 2: **Groupware products, projects and prototypes with interaction modes**

Type (with examples)	*Synch.*	*Distr. Synchr.*	*Asynch.*	*Distr. Asynch.*
Activity Management and Coordination (ASCW, OSCAR, Collaboratory)	X	X	X	X
Appointment schedulers (Schedule+, CaLANdar, Organizer)	X	X	X	X
Argumentation Systems (gIBIS, IBIS)	X	X		
Co-authoring systems (GROVE, GroupWriter, CoEd)	X	X	X	X
Conferencing systems, Desktop Teleconferencing systems (ASCW, Delphi)	X	X		
Group Decision Support Systems (Decision Conference, SIBYL)	X	X		
Information sharing systems (LiveWare, Lotus Notes)	X	X	X	X
eMail, bulletin boards (Lotus Notes, OVAL)	X	X	X	X
Workflow systems (Workflow, Beyond Mail)	X	X		

2.3 Classification of Groupware Systems

Table 2 structures current groupware systems according to the criteria given in section 2.2.

3 Applying Groupware in Software Development

Software development is an important field of application for groupware. Before considering the use of groupware in different stages of the development cycle a look at the degree of interaction between group members is helpful.

An investigation by *Brodbeck et al.* shows that interactions occur especially in early stages of the life cycle (analysis, design, prototyping) where the requirements and the architecture of the future system are determined in meetings [2]. The design of a software system is a highly interactive task and tool support for this important group process is still inadequate. Results from an informal study by *Kedzierski* show that only 21% of a designer's activities were covered by a software development environment during sample project (see table 3). Enhancing software design tools with communication facilities for different interaction types is thus a real need.

This section will therefore concentrate on groupware applicable in design activities (e.g. argumenta-

Table 3: **Division of Designer Time** [15]

Interaction Type	*Percent of Designer Time*
Questioning	27%
Informing	25%
Complaining	13%
Planning and discussion	14%
Trying existing commands	21%

tion systems, group decision support systems).

3.1 Capturing the Design Rationale

There is a growing interest in capturing the design rationale in the field of cooperative design [1, 14, 21, 23, 24]. Approaches to capture and preserve the design rationale aim to find and record the *why* that underlies the *what* [36], the *underlying intent* [16]. In order to achieve these goals discussions and decision making processes with alternatives, goals, arguments, evaluations, ... are recorded.

3.1.1 Design Rationale Representations

In order to carry on computer-supported discussions representations of argumentation structures based on rhetorical models have to be used. Actually a few representations are widely accepted.

IBIS Most representations are heavily influenced by the IBIS (=**I**ssue **B**ased **I**nformation **S**ystem) structure [20] developed by *Kunz* and *Rittel* with its most important derivation gIBIS, which can be regarded as a modern, graphical incarnation of IBIS. A discussion in gIBIS is carried on by raising *issues*, responding to issues using *positions*, and supporting or objecting positions with *arguments* (see figure 2).

DRL DRL=(**D**ecision **R**epresentation **L**anguage) was developed by *Lee* and *Lai* [24] and enhances gIBIS with facilities for the decision process. DRL was also implemented in a tool called SYBIL described in [22]. The main objects in DRL are *decision problem*, *goal*, *alternative*, *claim* and *question*.

Miscellaneous Representations Other representations include the Toulmin's model of argumentation [35], QOC =(Question, Option, and Criteria) [26] and the Potts & Bruns model [31].

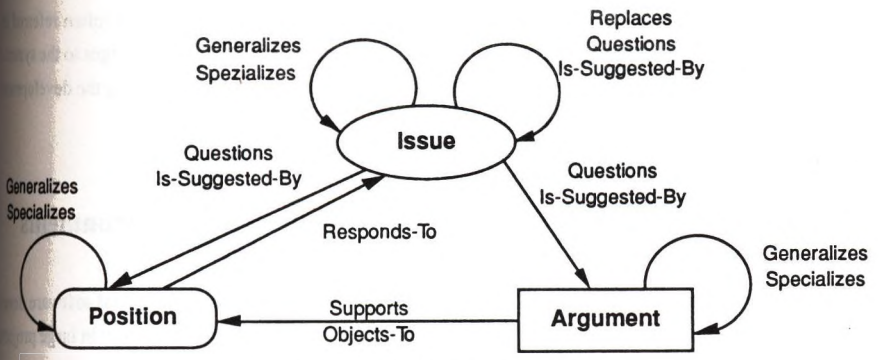


Figure 2: **The *gIBIS* vocabulary [24]**

3.2 Benefits and Implications for Software Engineering Tasks

Issue-based groupware technology supports group decision making, conversational structuring and the management of group memory, promotes critical reflection during design and eases group decision making. It helps the project team to detect design elements that had "fallen through the cracks" [36]. Moreover, recorded deliberations of the design process promise benefits for the overall life cycle, e.g. during maintenance [29]. In a field trial on an issue-based information system *Yakemovic et al.* report the positive effects of capturing discussions of designers using a groupware tool [36].

Design The analysis of design options is relieved. Explicitly stating the issues helps the team to understand the problem. In the case of changing requirements the capture of the design intent and decision process will help to redesign the system.

Implementation Design documents serve as input for the implementation phase. Knowledge about the *why* of a design eases implementation process.

Documentation System documentation as one important part of product documentation describes the product which is being developed from the point of view of the engineers. Recorded discussion processes are an appropriate mean to improve documentation quality and serve as an additional documentation of the product.

Inter-project benefits Design knowledge from a specific project can be reused by other projects to avoid rehashing of design discussions.

Maintenance and redesign of artifacts Maintenance is necessary to correct errors of a system (corrective maintenance) and to adapt the system to changing requirements. A major task during

the maintenance of a system is to capture the design rationale of a system (often referred as design-recovery), to understand the plan of the designers, before making changes to the system. The process of maintenance is eased by applying IBIS based groupware during the development of a system. Experiences are reported in [7, 36].

4 Towards Collaborative Software Engineering Environments

Powerful technology is needed to make groupware application a success in industrial software development practice because the application of groupware (e.g. gIBIS-based technology) in large projects will result in a huge amount of information.

This section describes the idea of utilizing the benefits of state-of-the-art software engineering environments for successful groupware application.

Section 3 presented applications of groupware technology in the software life cycle. The use of single, isolated tools, however, contradicts the idea of software engineering environments (SEEs) [3, 4] which aim to support the whole software life cycle by providing a description of the intended process *(process model)* [33], a standardized tool-interface, administration of deliverables *(repository)* and an interpreter of the process model *(process engine)* [12].

State-of-the-art SEEs typically support workgroups by repositories, which feature sharing of common information, keeping information private until it is stable and correct, controlling access and resolving duplication [30] to ensure the integrity of data stored in the repository. This view of a SEE as a special kind of database application leads to transaction oriented, atomic acting of developers as pointed out in [5] and does not meet the real needs of collaborative software development. It is therefore important to enhance current SEEs with facilities for workgroup support.

4.1 Coordinating tools and human agents

Figure 3 depicts four problem areas in the coordination of tools and human agents.

Tool/Tool Covers the aspect of control integration (see section 4.2). Technologies applied in this field include procedural interfaces, remote procedure calls and broadcast message systems.

Human/Tool A situation where a user starts an activity and uses a software tool (typical situation of using a CASE tool).

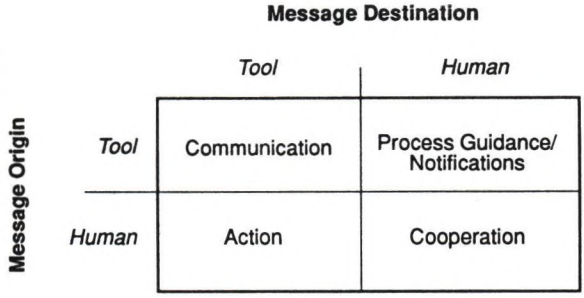


Figure 3: **Message Matrix [32]**

Tool/Human A process models enacted by a process engine notifies and guides the engineer to carry out an action.

Human/Human Cooperation between human agents. A software engineering environment should support cooperation between human agents. Future SEEs should provide a platform for human interaction and communication and therefore incorporate CSCW concepts. *Yourdon* coined the term 'groupware CASE tools' [37].

Two strategies to enhance human to human cooperation are discussed in the following section: (I) Integration of existing groupware into SEEs and (II) customization of SEEs to enhance workgroup support. On the one hand software development will benefit if groupware is integrated in the overall (existing) process. On the other hand groupware system can profit from SEE services (e.g. repository, configuration management, versioning, UI-integration).

4.2 Strategy I — Tool Integration

Strategy (I) aims at integrating an existing groupware system into a software engineering environment. The problem of tool integration is generally discussed in terms of three dimensions (see figure 4).

Data Integration Degree to which data generated by one tool can be utilized by other tools. Approaches to achieve data integration reach from simple files to shared, distributed information bases [3].

Control and Process Integration While control integration provides mechanisms allowing the cooperation of tools, process integration aims at coordinating humans and tools to achieve a common goal. In the field of process modeling some possible implications of groupware integration must

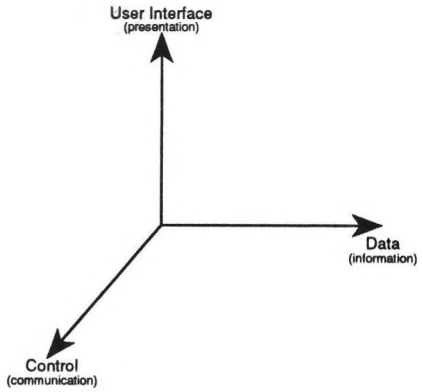


Figure 4: **Dimensions of Tool integration [32]**

be mentioned: Real processes are based upon human interactions like commitment, expertise, issue (open, resolved, dissolved), position, argument, decision, decision problem, alternative, assumption, goal, objective etc. [17, 25, 34]. Thus new classes of deliverables and new types of processes should be provided.

User Interface Integration Different tools should provide a similar user interface with consistent look and feel. User interface management systems (UIMS) and style guides can be applied.

In order to ease the integration process a tool has to provide a database interface, a communication interface to other tools and an interface to an UIMS (User Interface Management System). These requirements are not yet fulfilled for most groupware systems, which are often single, stand-alone products not designed with the goal of integration in mind and thus don't *'swallow their own medicine'*.

4.3 Strategy II — Customizing of SEEs

While strategy (I) aims at integrating an existing groupware system into a SEE strategy (II) tries to utilize some of the facilities provided by software engineering environments to construct groupware. Current SEEs provide facilities to customize tools and services to specific needs [30]. Customization would cover the following aspects:

Process model Definition of new result types and activity types with the associations between them

Tools Customization of embedded tools, e.g.

Table 4: **Comparison of strategies**

	(I) integration	*(II) customization*
+	utilize benefits of SEE	provide same environment (e.g. same UI)
+	uses existing groupware	higher user acceptance
-	different look and feel	environment customization facilities might be insufficient
-	data integration problems	
-	control integration problems	

- Definition of new diagram types (shapes, line paths, customization of drawing rules)
- Text editor

Help system Adaption of the help system to new items introduced

Environments supporting customization features (e.g. MaestroII [27]) make it possible to realize CSCW concepts and groupware functionality.

4.4 Comparison of strategies

Table 4 summarizes the pros and cons of the two strategies.

5 Conclusion and Future Work

A number of interesting questions arise when considering integration aspects of groupware systems and software engineering environments. Software development will benefit if groupware concepts and functionalities are integrated in the overall process. Groupware system can profit from various SEE services.

Further work will concentrate on the following topics:

- Capturing of design process (realization of gIBIS in MAESTROII)
- Embedding of social protocols to address the social dynamics of group activities
- Impacts on process models and process engine

References

[1] Bodker, S. et al. Computer support for cooperative design. In *CSCW '88, Portland, Oregon*, 1988.

[2] Brodbeck, F. C. et al. Tätigkeitsschwerpunkte und Qualifikationsanforderungen in der Software-Entwicklung: Eine empirische Untersuchung. *Software-Technik-Trends*, 13/2, 1993.

[3] Chroust, G. Distributing data in software engineering environments. *Proceedings of the 8th Austrian-Hungarian Conference*, 1992.

[4] Chroust, G. *Modelle der Software-Entwicklung*. Oldenbourg Verlag München Wien, 1992.

[5] Denert, E. Dokumentenorientierte Software-Entwicklung. *Informatik-Spektrum*, 16:159–164, 1993.

[6] Ellis, C., Gibbs, S., and Rein, G. Groupware—some issues and experiences. *Communications of the ACM*, 34(1), 1991.

[7] Flor, N. and Hutchins, E. Analyzing distributed cognition in software teams: A case study of team programming during perfective software maintenance. *Fourth Workshop: Empirical Studies of Programmers*, pages 36–64, 1993.

[8] Gappmaier, M. and Heinrich, L. J. Computerunterstützung kooperativen Arbeitens (CSCW). *WIRTSCHAFTSINFORMATIK*, 6:340–343, 1992.

[9] Greif, I. *Computer-Supported Cooperative Work: A Book of Readings*. Morgan Kaufmann Publishers, Inc, 1986.

[10] Grudin, J. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM*, 17(1):93–104, 1994.

[11] Hämmäinen, H. Form-based approach to distributed cooperative work. *Acta Polytechnica Scandinavica, Mathematics and Computer Science Series, Helsinki*, 58, 1991.

[12] Hardt, S. *Sprachunabhängige dynamische Ausführung von Vorgehensmodellen*. PhD thesis, Johannes Kepler-Universität Linz, Abteilung für Systemtechnik und Automation, 1994.

[13] Johnson-Lentz, P. and Johnson-Lentz, T. Groupware: The process and impacts of design choices. *Computer-Mediated Communications Systems: Status and Evaluation, Academic Press*, 1982.

[14] Kaplan, S. M. et al. Overview: ConversationBuilder project. Technical report, University of Illinois at Urbana-Champaign Department of Computer Science, 1993.

[15] Kedzierski, B. I. Communication and management support in system development environments. In Greif, I., editor, *Computer-Supported Cooperative Work: A Book of Readings*. Morgan Kaufmann Publishers, 1986.

[16] Klein, M. Capturing design rationale in concurrent engineering teams. *IEEE Computer*, pages 39–47, 1993.

[17] Krasner, H., McInroy, J., and Walz, D. B. Groupware research and technology issues with application to software process management. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 21/4, 1991.

[18] Kraut, R. E., Fish, R. S., Root, R. W., and Chalfonte, B. L. Informal communications in organizations—form, function and technology. *People's Reactions to Technology in Factories, Offices, and Aerospace*, pages 145–199, 1990.

[19] Krcmar, H. Computerunterstützung für die Gruppenarbeit — Zum Stand der Computer Supported Cooperative Work Forschung. *WIRTSCHAFTSINFORMATIK*, 4:425–437, 1992.

[20] Kunz, W. and Rittel, H. Issues as elements of information systems. *Center for Planning and Development Research—Univ. of California, Working Paper*, 1970.

[21] Kyng, M. Designing for a dollar a day. *Proceedings der CSCW '88, Portland, Orgeon*, 1988.

[22] Lee, J. Sibyl: A tool for managing group decision rationale. *Proceedings der CSCW 90, Los Angeles*, 1990.

[23] Lee, J. Extending the potts and bruns model for recording design rationale. *CH2982-7/91/0000/0114 01.00 IEEE*, 1991.

[24] Lee, J. and Lai, K. Y. A comparative analysis of design rationale representations. Technical report, MIT, Center for Coordination Science, CCS 121, 1991.

[25] Lonchamp, J. A process-centered framework for asynchronous collaborative work. *Proceedings of the Third European Workshop on Software Process Technology*, pages 261–269, 1994.

[26] MacLean, A., Young, R. M., and P., M. T. Design rationale: The argument behind the artifact. In *Proceedings CHI '89 Austin, TX*, 1989.

[27] MaestroII. *Software Engineering Platform*. Softlab–Munich, 1993.

[28] McCue, G. M. IBM Santa Teresa Laboratory—architectural design for program development. *IBM Systems Journal*, 17:4–25, 1978.

[29] Olson, J. et al. How a group-editor changes the character of a design meeting as well as its outcome. *Proceedings CSCW 92, Toronto*, 1992.

[30] OVUM. *OVUM evaluates—CASE Products*. Ovum evaluates, 1993.

[31] Pott, C. and Bruns, G. Recording the reasons for design decisions. *Proceedings of 10th International Conference on Software Engineering*, pages 418–427, 1988.

[32] Schefström, D. and Broek, v. d. G., editors. *Tool Integration—Environments and Frameworks*. John Wiley & Sons, 1993.

[33] Starke, G. *Sprachen zur Software-Prozessmodellierung*. PhD thesis, Johannes Kepler-Universität Linz, Abteilung für Systemtechnik und Automation, 1994.

[34] Starke, G. Why is process modelling so difficult? In Warboys, B., editor, *Software Process Technology*, pages 163–166. Springer Verlag, 1994.

[35] Toulmin, S. *The Uses of Argument*. Cambridge Univ. Press, England, 1958.

[36] Yakemovic, B. and Conklin, J. Report an a development project use of an issue-based information system. *Proceedings CSCW 90, Los Angeles*, 1990.

[37] Yourdon, E. *Decline & Fall of the American Programmer*. Yourdon Press Computing Series, 1992.

USER INTERFACE – CERTIFICATION AND AUTHENTICATION[1]

by

Gábor Fazekas and János Kormos[2]

ABSTRACT. In this note, we show how the computer aided automatic fingerprint person identification can help the authentication of documents in office management systems. We describe a possible way for automatic fingerprint identification and explain how it can be approved by statistical methods.

Introduction

One of the fundamental problems in office management systems is the authentication of vouchers, certificates and other recorded documents. The question of athoritativeness may be rather critical in such cases where it is strongly connected with the responsibility and authority of a given person. Traditionally, the authentication of a document could be proved for example by the sign or stamp of the resonsible person. However, in the office work flow such solutions can not be reliable enough: it is almost impossible to decide the originality of a sign and/or a stamp after a fax tramsmission. So it seems to be quite obvious to connect the problem of the certification of the authoritativeness of transmitted documents in an automated office management system with the problem of automatic person identification. A very natural solution of the person identification problem might be the use of fingerprints. Using fingerprints for automatic person identification in security systems is quite a new idea and it differs from the classical criminalistic applications mainly in two aspects. First, a person who has to prove his identity during an authentication process or an access control by his fingerprint will always be cooperative because he is interested in the success. This usualy results in good quality of input. On the other hand, the person can simultaneously give his identity code which could significantly reduce the searching and matching time.

The original approach – described in this note – has a syntactic and heuristical character. In our lecture statistical methods will also be applied concerning the fingerprint person identification problems and a statistical justification will be given in connection with our old 'syntactic - type' results.

Fingerprint person identification

Fingerprints have been used as one of the reliable ways of identifying individuals for a long time. The early Egyiptians and Chinese were already known to have used them to identify criminals and to record business transactions. [3] In the last century F. Galton pointed out that the minutiae of fingerprints remain unchanged throughout the life of an individual. Since then fingerprints have been used as one

[1]The research was supported by the Hungarian Scientific Grants OTKA Nr.1655 and Nr.T 014250.

[2]Institute of Mathematics and Informatics, Lajos Kossuth University, H-4010 Debrecen, Pf. 12, Hungary

of the basic means for the identification of criminals in the law enforcement. The desire to process large number of prints in a short time and the need for automatization enchanced the role of computers in fingerprint classification. Recently [2] we have developed a system for automatic person identification based on fingerprint recognition. Our system seems to be well applicable in certain security systems.

From the point of view of the hardware, this system includes the following components: (Fig. 1)

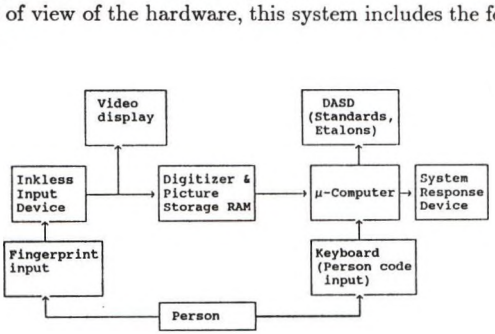


Figure 1

The inkless input device is painless for the users because it can sense and registrate fingerprints directly from fingers. It contains a common TV-camera supplemented by a special optical system. The critical part of this optical system is a rectangular prism glass and, the sensing is based on the total reflection phenomemon as it is illustrated in the following (cross section) figure:

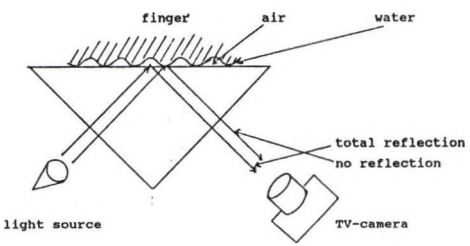


Figure 2

A light ray will be totaly reflected from the upper surface of the prism if the finger doesn't toutches the prism surface. This is the case for the walleys between the ridges and this is not the case for the ridges, because there is always a little amount of water on the finger. So the ridges result dark spot lines on the video screen.

The digitizer converts the video image into a digital image and stores the result in a special picture storage RAM. The result is a 256x256 pixel array where each pixel is represented by a 6-bit gray level value. Figure 3 shows a typical input image

obtained in this way.

The digitized fingerprint will be processed by a microcomputer. One of the most important problems occured during the processing is the encoding of the input image in such a way that the code should contain enough information to decide whether or not the actual fingerprint is exactly the same as a previously encoded and stored one.

Figure 3. A 64-graylevel input image

In order to get efficient codes the encoding method utilizes both global an local features of fingerprints.

Global features are the shape of ridge lines and the presence or absence of singular areas such as loops, whorls, arches and deltas. These properties allow a very natural primary classification of fingerprins.

The most typical local features are the fingerprint minutiae. These are irregularities such as ridge line endings and joinings, dots, short ridge, gaps, e.t.c. ,whose types and locations are unique for every individual. The minutiae and their relative location are so important, that although every fingerprint pattern contains about 100 minutiae, practically as few as 10 is cosidered sufficient to identify an actual pattern. The encoding algorithm is performed in several steps: The first step is the binarization and smoothing. The input image thresholded first into a binary image. The threshold is selected dinamically by the help of gray level histograms to make the number of black pixels equal to the number of white pixels. The Figures 4 and 5 show the result of binarization.

A smoothing process performed on the binarized image removes 'salt and pepper'

noise, fills small isolated holes in the ridges and in the background and bridges small gaps. (Figure. 5)

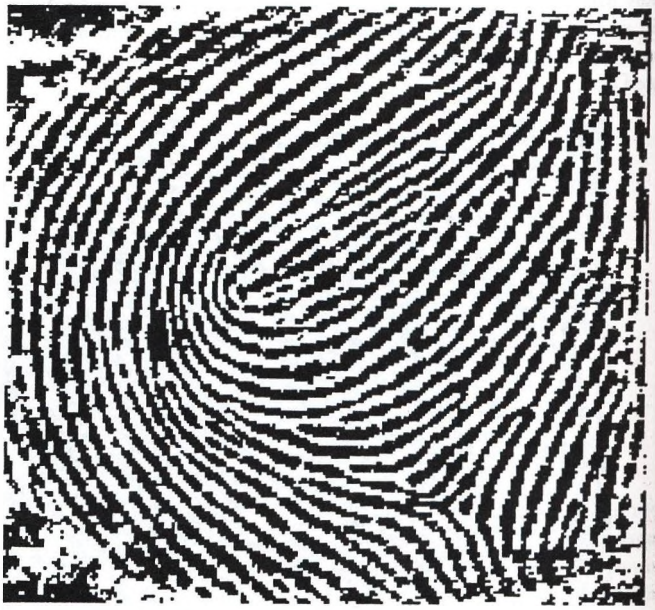


Figure 4. Thresholded image

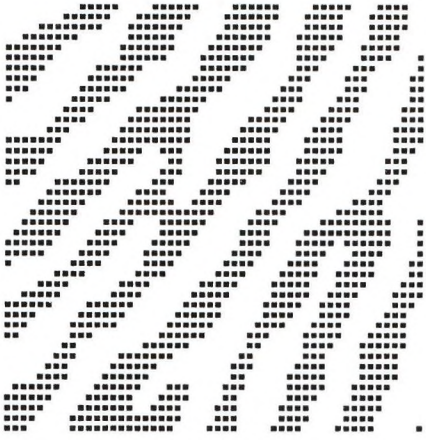


Figure 5. A window from a thresholded image

In the next step a skeleton is extracted from the binary image by the help of

a thinning process. This means that all ridge lines will be replaced by abstract discrete geometrical lines. In the Figure 6, '*' signes represent the points of the skeleton.

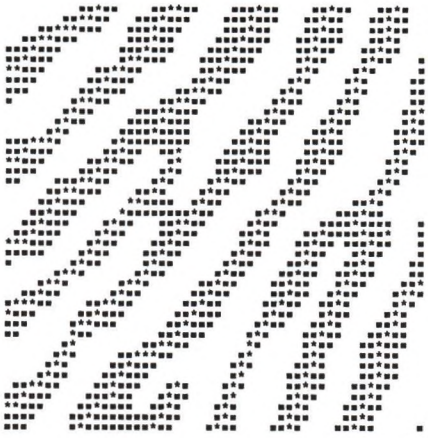


Figure 6. Skeleton

The flow of a ridge line can be locally described by chain coding. (Figure 7) The chain code assigned to each point of the skeleton is one of the following four elementary directions: –,|,/,\ or it is '+' (joining) and '#' (ending). This allows the correction of several errors occured during the sensing.

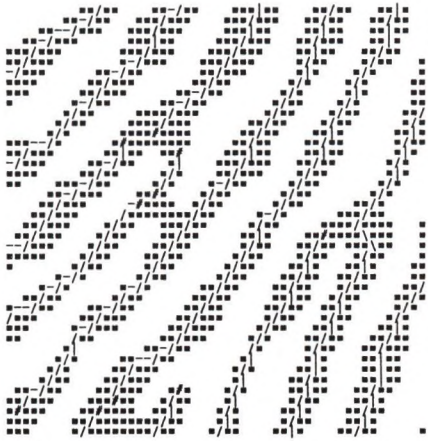


Figure 7. Chain coded skeleton

The chain coding process can be performed very quickly: we can follow point by

point every skeleton line and a point gets its chain code depending on the situation of its adjacent points.

In the next step we compress the chain code matrix into a direction code matrix or sampling matrix. The chain code matrix ,i.e. the fingerprint image is divided into submatrices of the same size corresponding to an array system and the predominant ridge direction is determined and represented again by one of the basic direction patterns: –,|,/,\ . The resulted direction code matrix (Fig. 8.) very good describes the global features of the actual print. One can compare the binarized image of a print with its direction pattern matrix. It is remarkable that singular areas (loops, whorls, deltas) always have a corresponding characteristic minor in the sampling matrix.

The direction code matrix can help the correct positioning of the fingerprint image before matching. Generally, prints taken from the same finger at different instances are likely to differ in orientation and position. But, in our system, the input device doesn't allow any significant change of orientation (rotation) of the finger during sensing. So the matching process has to facilitate only with the effect of translation. The actual translation values (displacements) can be determined by comparing the characteristic minors of the corresponding sampling matrices.

We introduced a Hamming like distance between sampling matrices and, the corresponding minor to a characteristic minor will be searched by minimizing this distance. If the resulted minimal values are too large then the further matching will be refused by the system. In the next step the system tries to identify all the minutiae which were recorded previously in the standard. In most existing fingerprint recognition system, the minutiae locations are recorded relative to an x-y grid that is superimposed on the print. In our system, we use three intrinsic coordinate system each of them corresponding to a characteristic minor of the direction code matrix. In this way, the effect caused by deformation can be reduced. The matching ends by an algorithm that determines the degree of correlation between the minutiae locations and directions in the considered prints.

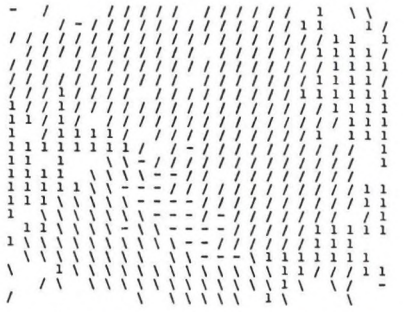


Figure 8. Direction pattern matrix to Figure 1

An experiment based on the above descibed method has been carried out on about 20 different fingerprint samples and repeated many times. The system has never made a first order error, i.e. it hasn't ever declared two prints to be the same if it wasn't the case. It made several times a second order error, i.e. it refused a

print in such cases when it should have been accepted. This was mainly because of the bad quality of input.

References

[1] FAZEKAS, G., On the Coding of Fingerprint Images, in: Proc. of the International Workshop on Algebraic and Combinatorial Coding Theory, Varna, Bulgaria, 18-24 September, 1988.

[2] ISENOR, D.K., ZAKY, S.G., Fingerprint Identification Using Graph matching, in: Patt. Recogn., Vol. 19(1986), No 2.

[3] MOAYER, B., FU, K-S., A Tree System Approach to Fingerprint Pattern Recognition, in: IEEE Patt. Anal. and Mach. Intel., Vol. PAMI-8(1986), No 3.

Changing language - changing routine

National Environments for Computer Applications.....

when two are making the same , it isn't the same .....

Petr Doucek

Key words:

user interface, hardware, human - computer communication, national language support for computer application, data processing in small firms,

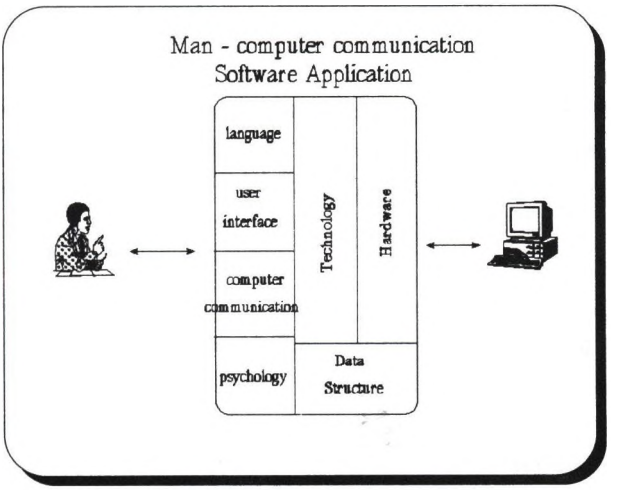
Abstract:

Using network communication and software application there is a need to adapt a software product to local conditions of the country . In practice it means that some system components must be changed. There are three principal standard components that are expected to be modified - hardware, user interface and language. Each of them can be modified by various ways in reality. Primarily these factors are of importance in small and medium firms. Small firms are not able to adapt complex applications to own conditions and that's why there exist troubles for data management and data processing. For solving these problems there are in principle two ways - find a computer application that complies with people and cultural habits or adapt people habits to any computer application.

1. Integration with specific problems

In time, where millions of people are connected by very sophisticated computer networks and software is moving around the world, we meet one specific sort of problem. It can be specified as national support for program products. To solve this particular problem is not very difficult, but the real problem is not only in the language area. To change the user's language means to change also his thinking, the cultural and social aspects of his communication and his technological knowledge. By using any application in different areas we can see three potential groups of problems:

- **technological,**
- **technical (hardware),**
- **social - psychological.**



Pic 1. Application's role in communication

These problems are very important for small and medium firms in all countries where the world standard computer language - English - does not seem to be a standard language for communication between people.

Knowing any language is not the only condition for a good communication with people in foreign countries. There is also a necessity to know and to a certain extent accept another culture, the habit and thinking of this country (these countries) where the language is used as mother tongue.

There is probably a way for building a common base for communication between different groups of people - determine a general set of language knowledge that must be disposed by all the people in the world. But this solution does not solve groups information interchange, where people in one group are using the same terms as in another group but with another meaning.

2.1. Technology - bridge over the troubled water

A group of **technological problems** seems to be very extensive. Now in the process of European integration the major problems for computer based common communication is laying in differences between two former European political blocks - Eastern and Western countries.

Differences between people living in the former East block countries and West Europe can be characterised as follows:
Programmers, project managers and other people from *Western countries* taking part in the build up of information systems have tendencies to apply high advanced technology and decision making systems corresponding to the high standards and information technology in their country.

People from *Eastern countries* are not so skilled in advanced technologies, but they are used to work more creatively, and more oriented toward the solution of the same or similar problems with modernist technology.

Even in different Eastern countries there exist the diverse types of system environments - for example in the Czech Republic there is an extensive use of Fox Base or dBase based databases for small and middle-sized firms, for large applications databases in the environment of operation system UNIX. In small firms in German speaking countries integrated software product Open Access is often used (perhaps in Northern America too) but there are difficulties in communication and data interchange in joint ventures between Czech and German firms on the territory of Czech Republic. And WHY ? The major problem seems to be - different habits in information processing, preferences of one technology over the other .

2.2. Hardware - universal ... .really?

Technical problems are closely related to different types of computers or computer equipment. There are countries where the computer God is named IBM and on the other hand in some other countries (territories) IBM abbreviation represents the symbol for technique and technology that is not acceptable at all. Selling software applications, one must keep in mind this specific characteristic of a particular country.

Used hardware is also various in different firms. I am talking not about different suppliers of hardware but about different systems, operation systems and user interfaces. For small firm it makes no sense to buy a large computer network based on VAX machines and UNIX environment exploitation on the other hand information system of a large firm cannot be processed on network of PCs.

How were these technical (hardware) problems solved in the Czech Republic ? There are three periods to be mentioned in this connection . First period was about 1960 - 1980. In this time we could see a very limited number of computers in business area and about all of these computers had terminal batch communication. About in 1980 started a large invasion of PCs. Some project managers and information bosses thought that PCs are a remedy for solving all existing problems. And after connection these PCs into networks (about 1985) a period of liquidation of central computers with terminal communication started. But the spiral of life goes on and the reality is a green tree not only the a grey theory. About at the same time in our republic a new computer wave began - client server architecture, distributed databases and again terminal connection in computer centres.

Now one can see a large diversification in our country. Small firms prefer PC based data processing, medium firms have PC based on heterogeneous data processing by using network communication (PC or PC - mainframe) and large corporations are working with network or networks - some of them are PC (Novell) based, some of them are based on various UNIX versions - with homogenous user interface.

2.3. Human factor and it's limitation for computer applications

The Social-psychological group of problems can be characterised from a different point of views. In my opinion there are four main factors that have influence on human - computer relations:

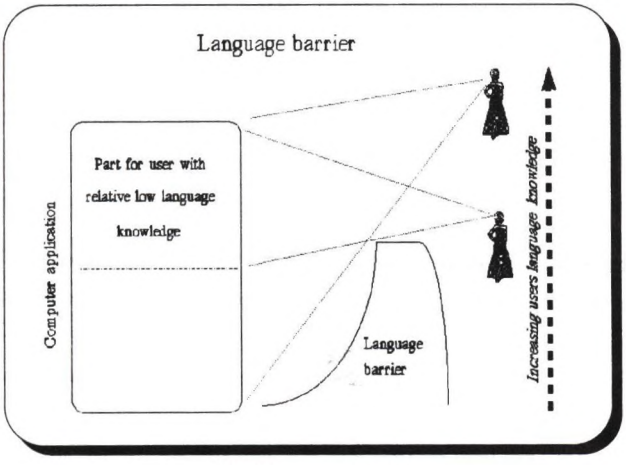
- language barrier,
- users interface,
- computer communication,
- user's psychology.

Each of these components has its important place in the implementation of modern computer technologies in practice. To forget any of them in a process of building some project means

that this project will be never successful finished. There's are about 70 percent of all built projects (according to Siemens statistics).

2.3.1. Language barrier

Language - software applications are prepared for a relative wide group of routine users. But in reality there is a language barrier for the use of this adapted software. My experience is, that any software product that will be put into routine usage in firms **must** be translated into national language. Exception are only those firms that have enough money for buying the best qualified employees. These people ought to have a good command of a foreign language. Demonstration of this reality can be found in all small firms in the Czech Republic. As example can be shown text processing. About three years ago one could find only local Czech text editors in Czech small firms. It was a product of T602 software corporation but this editor was rather different form European (World) editor s standard. The only one reason why this editor was used was that there were not any world standardised editors on the territory of the Czech Republic that had been translated into Czech. According to this conditions (language barrier) the technology knowledge has changed - in this case text editor knowledge - in one specific region.



Pic.2 Language barrier

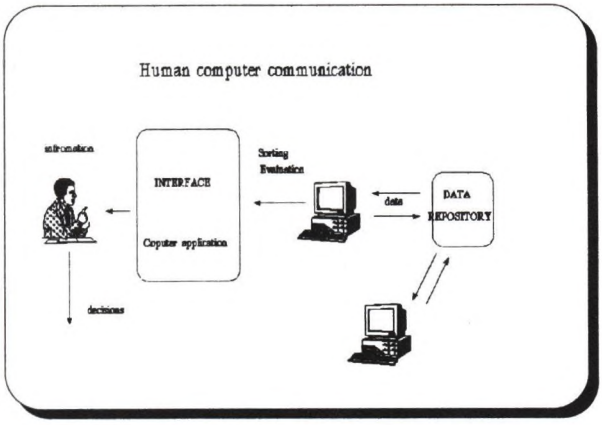
2.3.2. User interface

User interface - is also one of problems that are underestimated. Let us take accounting, for example. Preparing accounting software for British employees and trying to implement this application under the conditions of Central European countries (Czech Republic, Hungary, Poland, Slovak Republic) is a very optimistic intention. Of course there are little different routines but the main difficulty will be the very disputable sequence of screens and also the complete data processing of the whole system. I have seen applications that were translated and modified from one local environment into the other and it was very interesting that results were different. Knowledge of information technology was not the most important thing.

In a user interface there can be also involved screen design. All problems that are connected to correct and right design of screens must be discussed and also solved very carefully, but each person has its different point of view, what are acceptable colours, variable fields, texts, windows and space combination on screen for computer application. These views are based on cultural conditions, on habits, upbringing and the education level of the whole group (nation).

2.3.3. Computer communication

Computer communication does not seem to represent any problem. The user applies keyboard (mouse e.t.a.) and the computer gives him (her) any messages or information. But in comparing more computer applications we can see basically different approaches to message interchange between a user and a computer. Preparing for successful business application requires user friendly communication.



Pic. 3 Interface's role

A user does not like to change his habits. He expects that in all applications (used by him) its messages are located at the same place on the screen. and if there is one software product that does not satisfy his requirement, this application is not used at all. Of course, if it depends on his free will.

2.3.4. Psychology - human's standard complex

Psychology - this effect is specific for Eastern European countries (also for developing countries of Asia, Africa and Southern and Middle America). Several groups of employees - majority of middle age and age before the end of an economic active life - are not skilled in computers technologies. That's why firms have problems with good computer skilled managers. Only in the last period of time - about the last two or three years - a new generation (also computer skilled) of young managers is coming into the firm boards.

For not computer skilled people first step by using computer is to lose fear of a new element in life - of computer. A computer machine is not a human being - **it** cannot be lied out, talked

down, deceived. **It** is correct all the time around - excluding situations where is some hardware damage or wrong program lines sequence. **It** is not ill, has no private problems and should give always the same answer to the same question. In this sense computer is for an employee a new type of partner or colleague.

It is very useful to give some interesting programs to employees at the first time and in this way to show them that a computer is not a horrible thing and it can be used also for entertainment and for getting more information in a real work.

The second stage - after fear leaving - are different courses for employees. During these courses it is necessary to control motivation of people taking part at them and also its quality.

Conclusions

For the process of European integration a very large data and information interchange between former political different blocks - East Block countries and West Block countries seems to be typical. Due to former different approaches to a computer technique and modern information technologies - in these countries there are different habits and routines in data processing and in information management. Discrepancies are in hardware and in technologies used, but the main part of this problems is combined with human factor in computer based information systems. Quality of this human factor depends in a substantial degree on every day routine, on the level of computer knowledge, on the average education level and on other cultural and social aspects. If any firm wants to penetrate new software application markets, it is necessary to know and to keep in mind those specifically conditions and for these changed conditions to modify offered computer applications. Without modification computer applications are often not marketable.

References

1)Andrews, L.: The Art of Using Computers, Boyd and Fraser Publishing Company, 1986

2)Chroust, G.: Modelle der Software-Entwicklung Aufbau und Interpretation von Vorgehensmodelle, Oldenbourg Verlag, Muenchen - Wien, 1992

3)Doucek, P.: Mathematical Modelling of Software Projects, University of Economics Prague, 1992.

4)Doucek, P.:Modelový přístup k odhadům základních ekonomických charakteristik softwarových projektů, disertační práce, VŠE Praha, 1992

5)Doucek, P., Gala, L.: Do boje proti počítačové negramotnosti aneb jsem gramotný ?, E a My, Praha 1990, str.8

6)Smith,A.N., Medley D.B.: Information Resource Management, South-Western Publishing Co. 1987

Applied Workflow Management

Communication Support in the Workflow Management Environment LEU*

Volker Gruhn

Abstract

Business process modeling and analysis are subjects which are discussed for years most of all in the academic community. In addition, business process enaction is an area of commercial interest without being based on well-researched concepts. In order to close this gap, well-founded concepts for process enaction are required. The goal of this article is to identify communication support needed in business process enaction. This is based on a distinction of different types of activities which are to be carried out in business processes. Moreover, we discuss the implementation of the discussed communication support in the workflow management environment LEU.

1 Business Processes and Workflow Management

Most of todays business objectives can only be accomplished in a joint effort by many people: in some projects, work may be divided among ten of them, in others among hundreds and in some large ones even among thousands of people. The management of people alone, their allocation to tasks according to their skills, the communication between them, the integration of their respective results represents a task of formidable complexity. In the following we use the term *business process* to denote a set of logically related activities, which are carried out to reach a defined outcome. This explanation corresponds to the common understanding of business processes as discussed in [16]. The people who participate in a business process are called *human actors* in the following.

An effective support of business processes is one of the most promising approaches to improve productivity in various application domains. Typical examples of business processes are discussed below:

- **Production processes** focus on the production of a certain product, for example, the assembly of a car. The activities involved in such processes, their scheduling, and the expected outcome is usually well-defined.

*LEU is an abbreviation of the German translation for *LION Engineering Environment*.

- **Project management processes** focus on keeping the project infrastructure functioning and on providing circumstances in which the project goal can be reached. Examples of activities in project management processes are meetings, document reviews, workshops and work allocation to team members. Typical documents manipulated are project plans, task lists, meeting agendas etc.

- **Administration processes** occur in any company that sells or purchases services and/or products. Typical activities of administration processes are to handle product requests, to administrate information about customers, to process orders, to manage invoices, etc. Typical documents manipulated are customer records, invoices, orders, etc.

- **Decision processes** occur in management tasks. Managers need support in order to take decisions. In special cases, agreements of people allocated at different hierarchical levels are needed and have to be supported. Typical activities of decision processes are to extract condensed data which are appropriate as decision basis and to record rationales for decisions.

- **Office processes** deal with the flow of documents in the office. Typical documents manipulated in office processes are letters, reports, announcements, etc. Most business processes have interfaces to office processes and, finally, the efficiency of many business processes is determined by the office processes they depend on.

- **Software processes** are a subclass of production processes. A production process whose final outcome is a piece of software is called a software process. Due to the immateriality of that product and to the poor understanding of software development, software processes pose some extra requirements for process management (like easy modification, completion of modeling while processes are already running, very flexible allocation of human actors to activities etc.). Typical examples of activities of software processes are requirements analysis, design, quality assurance, etc. Typical documents manipulated in software processes are modules, design descriptions, bug reports etc.

In principle all these types of business processes can be described by similar means (activities, document types, responsibilities, pre- and postconditions of activities, order of activities etc.). In the literature a wide range of entity types building business processes can be found [24, 5, 2].

A workflow management environment should support comfortable modeling of business processes, analysis of business process models, and encation support for business processes. While process modeling and analysis have been discussed for years in the software process community [22, 19] and in the administration process community [9], process enaction is actually done in some industrial projects [16] without being based on formal concepts.

Our experience with business processes from different application domains [8, 13] shows that it is essential to *carefully* assist and guide human actors in processes. This means, that enaction has to cope with acceptance problems as soon as it becomes too strict. A *careful* approach could be to just provide support without enforcing its use. Therefore, our approach is to provide communication support for different types of activities carried out in business processes.

In the further course of this article we first distinguish different types of activities of business processes in section 2. Section 3 gives a brief introduction to the workflow management environment LEU. Then, section 4 discusses the communication support provided in LEU. Finally, section 5 concludes this article with pointing what the first experiences in using LEU for business process management look like.

2 Different Types of Activities in Business Processes

The very most explanations of the term process (in the software process community, in the industrial engineering community [6] or in the traditional business process community [16]) center around the notion of *activities* [14]. An *activity* is usually understood as a unit of work that cannot be usefully subdivided. This is obviously not a formal definition. It depends on who decides about activity subdivision. Accordingly, the notion of *activity* cannot be formalized.

Nontheless activities build the skeleton of processes. They are the entities where human involvement into business processes can be pinpointed. Activities may require different kinds of human participation [25]. In order to discuss mechanisms needed to support human actors, we distinguish different kinds of activities.

Automatic activities: Automatic activities can be carried out without any human intervention. They can be carried out by devices. Examples are:

- compiling a module as soon as it is edited (in a software process),
- archiving a document as soon as it is released (in any business process dealing with documents),
- producing a back up of a system state at a predefined point of time (in a computer-supported business process).

Automatic activities have to be programmed. It has to be defined which piece of software has to be executed, when the activity is started. It has to be defined how parameters are passed to that activity and where results are stored. Thus, they are described by an input/output behavior and by a body, which specifies what has to happen, when the activity is executed (compare below).

```
archive:  IN f1:  document;
          INOUT path:  file-sys;
BODY:     copy (f1, path);
DESCR:    archive a document in a certain
          directory of a file system.
```

Individual interaction activities: Individual interaction activities require the participation of only one human being. They can be carried out either by a human actor and a device or by a device alone. Examples are:

- editing a module,
- reading a document.

In addition to a description which is similar to the description of an automatic activity (input/output behavior and body), an individual interaction activity contains an informal description. This informal description is a help for the human actor who participates. Moreover, it is defined which human actors are allowed to participate. This can either be done by identifying the potential actor by name or it can be done via a role concept [21]. Besides the general definition of roles, it is moreover necessary to define, which roles/human actors are allowed to access which documents. It may, for example, be useful to define that everybody playing the role *secretary* is allowed to edit documents, but that everybody who modifies a contractual document must have additional legal expertise. Thus, permissions to participate in activities are defined by roles and value-dependent restrictions. The example discussed in the following is an *edit*-activity, in which the tool *WordPerfect* is used. It is defined that human actors playing the role *secretary* are allowed to participate and that moreover only secretaries to whom the corresponding document is assigned are allowed to edit it. The additional constraint specifies that contractual documents can be modified only by secretaries with the qualification of having legal expertise.

```
edit:  INOUT m1:  doc;
       WordPerfect (m1);
DESCR:        edit a document with WordPerfect in
              write mode
ROLE:         secretary
CONSTRAINT:   m1 must be assigned to
              secretary;
              if m1 ∈ {contractual docs} :
                   QUAL: legal expertise;
```

Social interaction activities: Social interaction activities require the interaction of several human beings. They are carried out by two or more human actors.

Examples are:

- discussing a design document in a review team,
- report results in a meeting.

We distinguish these activities from individual interaction activities, because *group activity is* vastly *different from individual activity and has its own needs and requirements* [18]. Section 4 will show that different enaction mechanisms are required for individual and social interaction activities.

Social interaction activities are described by their input/output behavior, by their body, and by a so-called dialogue description. A dialogue description defines how many human

actors playing which roles have to interact with each other to carry out the described activity. The body of the activity given below defines that the input parameters are passed to all potential human actors by means of the basic service *send-ascii*, that then the interaction has to take place, and that the result object should be produced by using the basic service *WP-write*. The dialogue description given below defines that at least 9 team members must be present to have a report meeting, that one reporter must be available (who is also identified to be the responsible interaction manager), and that 2 or 3 management control board (MCB) members must be willing to participate. The interaction manager is responsible for the coordination of the activity (compare section 4).

```
report results in meeting:
IN:  ag:  agenda
OUT: min:  minutes
     send-ascii(ag);
     report results in meeting;
     WP-write(min);
DESCR: one reporter reports results of a
  working group.  All participating actors decide
  about further work for the group.
DIALOGUE_NAME: report results in meeting
PARTICIPANTS:
  ROLE: team member NUMBER: >9
  ROLE: reporter    NUMBER: 1
        (responsible interaction manager)
  ROLE: MCB member  NUMBER: 2 - 3
```

We consider this distinction of activity types useful because they require different kinds of communication support. In the following we first give a brief introduction to the workflow management environment LEU. Then we discuss how the different kinds of communication support are implemented in LEU.

3 The Workflow Management Environment LEU

LEU[1] is a workflow management environment. It is currently used in a large project, which develops business processes for the administration of apartments of seven apartment administration companies. Altogether, several hundred thousands of apartments will be managed within business processes based on LEU. The installation of these processes in the apartment administration companies will start in January 1995.

[1] *LEU* is an abbreviation of the German translation for *LION Engineering Environment*.

Since the goal of this paper is to illustrate the communication support provided for different types of activities, this introduction is rather brief. For a more detailed discussion of LEU we refer to [12].

LEU supports the management of business processes. Business process management in LEU consists of process modeling, process model analysis, and process enaction. The LEU approach to process management is based on two ideas:

- Modeling, analysis, and enaction should be possible in an incremental way. This is illustrated in Figure 1 by the relation between these process management activities.
- Modeling, analysis, and enaction should be based on just one representation of the knowledge about business processes. This is illustrated in Figure 1 by only one process model representation which is accessed within all process management activities.

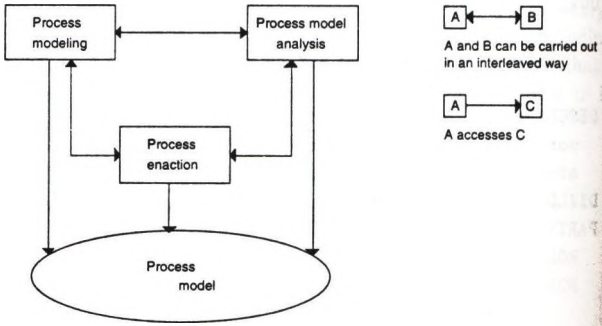


Figure 1: Incremental process management in LEU

Process modeling is based on data modeling, activity modeling, and organization modeling:

1. In LEU **data models** are used to describe the structure of objects (and their relationships), which are manipulated within a business process. Data models in LEU are described by means of extended entity/relationship diagrams [3]. Object types like *letters, documents, contracts* are described as entity types. Object types can either be of a predefined format (postscript, WorfPerfect, etc.) or they can be structured. In case of structured object types attributes have to be defined. A contract, for example, could be defined by a contract number of attribute type *integer*, by the contract parties identified by name (and, therefore, of type *string*), and by a contract text of type *text*. Relationships between object types can be of different cardinalities. They can be optional or mandatory. A scheme of a LEU data model is shown in the bottom part of Figure 2.

2. In LEU **activity models** are used to define activities to be executed in a business process. Moreover, activity models define the order in which activities of a business process have

to be carried out. Activity models in LEU are described by means of FUNSOFT nets. FUNSOFT nets [15, 7] are high level Petri nets [20], whose semantics is defined in terms of Predicate/Transition nets [10]. In FUNSOFT nets, the T-elements (represented as rectangles) are called agencies. Agencies represent activities. Activities in FUNSOFT nets are described as sketched in section 2. S-elements (represented as circles) are called channels. They represent document stores. FUNSOFT nets are hierarchically structured by means of T-element refinement [17]. FUNSOFT nets do not only contain a definition of activities and their parameterization, but also an order of activities. They allow to define that activities have to be carried out sequentially, concurrently or alternatively. A scheme of a FUNSOFT net is sketched in the central part of Figure 2.

3. **Organization models** in LEU are used to define which organizational entities are involved in a business process. Organization modeling is based on a hierarchical role concept. Roles are sets of permissions for the execution of activities. Roles can be attached to organizational entities. Human actors can be assigned to organizational entities. Moreover, they can be attached to roles directly. This latter possibility is useful, when certain human actors have certain permissions independent of the organizational entity they are assigned to. Organization models in LEU are described by means of organizational diagrams. The relationships between organizational entities, roles and human actors are defined in a tabular form. The top part of Figure 2 sketches an organization model.

Once these aspects of processes have been modeled, it is necessary to integrate them. Integration of data models, activity models, and organization models means to define:

- which channels of FUNSOFT nets are typed by which object types identified in data models (indicated by the arrows annotated with *typing* in Figure 2),
- which organizational entities are responsible for which activities (indicated by the arrows annotated with *responsible* in Figure 2).

Process analysis helps to avoid that erroneous or inefficient process models are enacted. Process analysis in LEU is based on process simulation, verification of process model properties, and evaluation of already finished processes. Simulation of processes imitates the input/output behavior of activities. Process simulation helps to get a *feeling* for a process. It gives an idea about potential process states. During process simulation a so-called simulation trace is recorded. Based on this trace bottlenecks, critical path, maximum, average and minimum number of objects per channel can be figured out. This information helps to detect errors early or to gain confidence into a process model. Verification of process model properties proves properties of process models, such as the absence / existence of deadlocks, the existence of useless object types etc. [4]. Finally, the evaluation of already finished processes (which are recorded in so-called execution traces) helps to calibrate process models. Assumptions about time-consumptions of activities and the degree of concurrency of activities can be compared with a *real* process. This can be a help for further improvement of process models. All three branches of process analysis are discussed in detail in [11].

The final goal of business process management is to enact business processes. That means, to ensure that a process behaves as specified in the process model. **Process enaction** means to coordinate human actors and to provide them with required communication support. Since

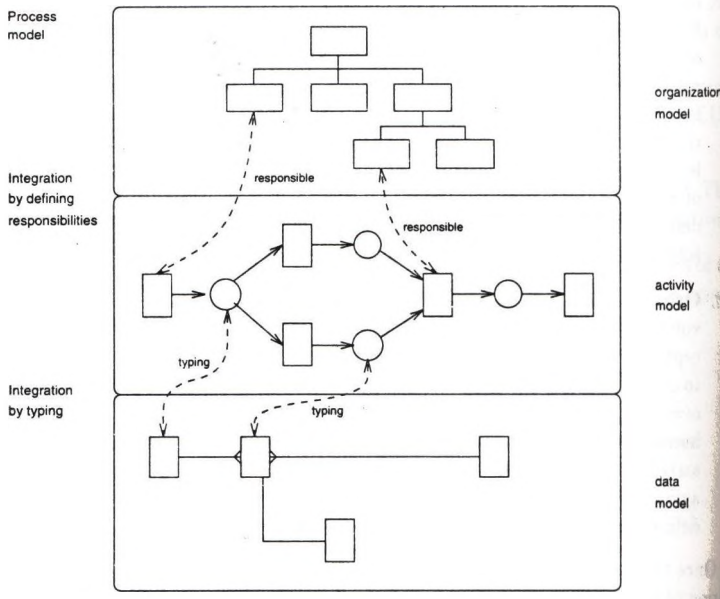


Figure 2: Integration of data, activity, and organization models

the LEU communication support for process enaction is the key subject of this article we do not discuss it here, but devote the following section to this subject.

4 Communication Support in LEU

4.1 Communication Requirements

Different kinds of activities demand different kinds of communication support [1]. The communication between two automatic activities is different from the communication between an automatic activity and an individual interaction activity, which is itself different from communication support needed between human actors participating in two related individual interaction activities or even between human actors participating in one social interaction activity.

Communication provided in a completely automatic process (no interaction activities at all) is called **interoperation**. Interoperation support is needed when information is exchanged between automatic activities. Figure 3 shows that interoperation takes place, for instance, between tools that reside on different devices. This communication does not need any human intervention. Examples of interoperation communication are:

- A set of compiled modules must be linked. Therefore, the compiled modules must be

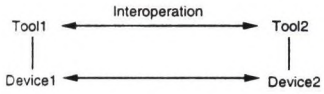


Figure 3: Interoperation communication

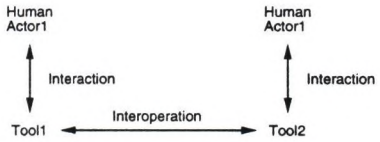


Figure 4: Interoperation and interaction communication

communicated between the compiler and the linker.

- A figure produced by a drawing program must be incorporated into a piece of text. Thus, the figure representation must be communicated between the text processing system and the display system.

The communication infrastructure required to support an interaction process (a process in which only one human actor participates) includes additional interaction support. It requires communication between a human actor on the one hand and a tool, respectively a device, on the other hand. Figure 4 shows where interaction occurs in an interaction process. Each of the tools shown in Figure 4 is used by only one human actor. The exchange of information produced by using the tools occurs through interoperation between these tools.

A process with many participants requires support for communication between human actors. If the communication between different human actors is provided by a chain consisting of an individual interaction activity followed by another individual interaction activity, we call the respective communication interworking [1]. This type of interworking is close to the notion of interworking as it is identified in the ESF project [23]. Interworking support is also needed in each social interaction activity. Interworking support is, for instance, needed in the discussion of a document as well as in the joint preparation of a meal. It is needed between a human actor writing a requirements document and another one who transforms that document into a design document.

Figure 5 illustrates the communication infrastructure in support of interoperation, interaction, and interworking. Figure 5 shows two interworking links. One of the interworking links is a direct link between the two human actors. This link depicts non-formalized communication that is not supported by any kind of communication protocol. It is based on spoken

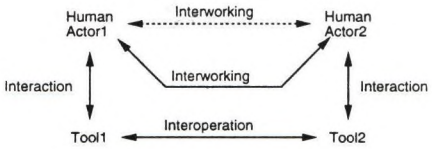


Figure 5: Interoperation, interaction, and interworking communication

language. The other interworking link represents the formalized communication between human actors who participate in related individual interaction activities by means of interaction and interoperation support. This kind of interworking communication is indirect.

4.2 Enaction Support and Components in LEU

Interoperation, interaction, and interworking (direct and indirect) require different enaction support. Enaction support for interoperation requires that automatic activities are started and that data between automatic activities is transported. As soon as all inputs are available and all other activity's preconditions are fulfilled automatic activities can be started.

The encation mechanism for individual interaction activities is more sophisticated. As soon as all input parameters are available and all other preconditions are fulfilled an individual interaction activity has to be offered to all human actors who have the right to participate in that activity. An offer of an activity to somebody could, for example, be implemented by writing that activity and the actual input parameters into the personal agenda of the human actor. As soon as one of the human actors declares that he is ready and willing to participate, the offer has to be removed from the personal agendas of all other actors.

The enaction mechanism for social interaction activities is even more difficult. The responsible interaction manager can propose a date for the social interaction. Moreover, the activity's inputs are made accessible to all potential participants. In the example discussed above the meeting's agenda is sent to all potential participants. Based on the agenda items they can decide whether or not they want to participate. Every potential attendee can agree or disagree to the proposed date within a certain period of time (also fixed by the interaction manager). As soon as this period of time is over, the interaction manager checks whether the interaction activity can take place on the proposed date. It can only be scheduled if a sufficient number of attendees has accepted. What is sufficient in this context is determined by the description of the social interaction activity which encompasses a description of the number and roles of expected participants (compare to dialogue description given above where it is specified that, for example, 2 or 3 management control board members must participate). If the number of process participants who agree is sufficient, then the interaction manager can decide whether he wants to schedule the meeting for the proposed date or whether he prefers to make another attempt to propose a date. Another attempt may be useful (even if *enough* human actors are ready to participate) if the interaction manager wants particular human actors to participate

and if these are unable to participate at the proposed date. Once a date has been fixed, all human actors who agreed to that date get an entry in their personal agendas, which denotes the social interaction activity and the proposed date. When the proposed date has come, all human actors get a reminder. At the end of the social interaction activity, the interaction manager is responsible for feeding the required result back into the process. In the example discussed, it is specified that this has to happen by means of the service *WP-write*.

The different kinds of enaction support discussed above are implemented in different enaction components. In the following we discuss the architecture of enaction components of LEU.

Figure 6 shows the relationship between different enaction components and their embedding into the overall architecture. Boxes represent components (i.e. module hierarchies). Arrows between boxes indicate that there is a *call*-relationship between these components. The annotations of arrows give examples of demanded services. The component *Process engine*, for example, calls the service *start activity* from component *Automatic activity handler*. Between some components we find double-headed arrows (e.g. between *Process engine* and *Agenda controller*). In that case the arrow annotations indicate which service is demanded by which component (e.g. the *Process engine* demands to *fill entries* into personal agendas, and the *Agenda controller* returns entries which have been selected (service *selected entry*)). The main functionality of these components and their interfaces to other components are discussed in the following:

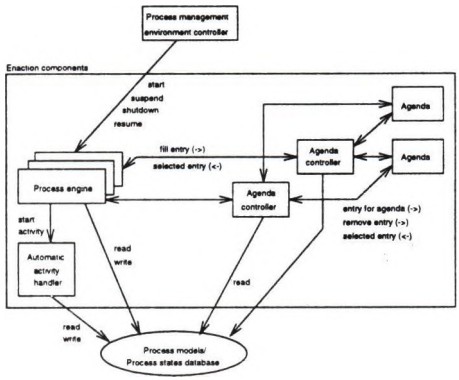


Figure 6: Potential architecture of enaction components

- There is one **Process engine** for each business process running. It can be useful to understand a large project as a set of communicating processes in order to keep them manageable. Therefore, more than one process engine may be necessary. The process engine drives the process forward. Process engines can be started, suspended, shut down, and resumed by the environment control. The process engine for a process P identifies all activities of P for which all inputs are available and which, therefore, could be executed. In

case of automatic activities, it starts them via the *Automatic activity handler* (compon
below). In case of interaction activities, corresponding entries are sent to the proce
engine's *Activity controller*. To identify executable activities, a process engine reads th
database storing process models and process states. To update the process state afte
executing an activity, the process state database is modified.

- The only functionality of the **Automatic activity handler** is to receive execution re
quests from all process engines, to start the requested activities, and to return whethe
or not the execution has been successful.

- As soon as a process engine is started, a related **Agenda controller** is created. An agend
controller administrates all interaction activities of the related process. It is connected t
the personal agendas of all human actors who may participate in the process. The set o
human actors who may participate can be identified on the basis of roles and permission
which are also stored in the underlying database. Agenda controllers manage individua
and social interaction activities. If an individual interaction activity could be executed
the agenda controller sends it to the personal agendas of all human actors who could
participate. If an individual interaction activity is selected by a human actor, the agend
controller returns this information to its process engine which starts the activity at the
workstation of the human actor. When a social interaction activity could be executed, the
agenda controller sends it to all concerned personal agendas. Then, the agenda controller
manages the agreement procedure. When the social interaction activity is finished, the
agenda controller returns a corresponding acknowledgement to the process engine.

- There is one personal **Agenda** for each human actor. As soon as a human actor logs into
LEU, his/her personal agenda is started and automatically connected to the agenda con
trollers of all processes, the human actor has permissions/roles to participate in. Figure
7 shows an example of an personal agenda from an apartment administration process. It
contains some status information about the human actor and his/her role. In the example,
Smith plays the role *rent calculation clerk* and is allowed to participate in two processes
(*rent calculation for new apartments* and *rent calculation after renovation*) in which he
can interact within three individual interaction (*read_app_description, insert_parameters,*
check_expenses) and one social interaction activity (*present_expected_rents*). The last
agenda column shows since when the activities are executable. The additional date for
the social interaction activity indicates for which point of time the *meeting* is scheduled.
The * means, that *Smith* is the responsible interaction manager.

5 Conclusion

We have identified different kinds of human involvement into business processes. We discussed
different enaction mechanisms and we proposed an architecture for corresponding enaction
components. Moreover, we discussed the implementation of these components in LEU. LEU
is currently used to model, analyze, and enact business processes from the area of apartment
administration for several hundred thousands of apartments. Within this project it turned out
that the flexible adaptation of organization and activity models is an essential feature of LEU.

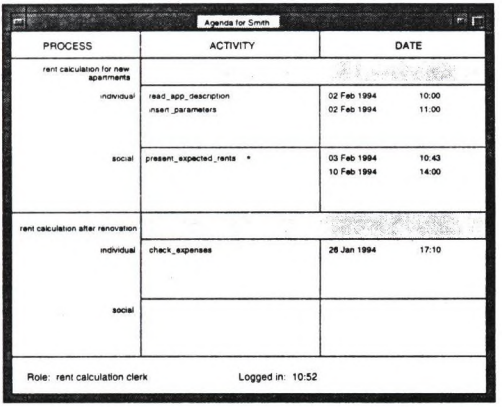


Figure 7: Example of a personal agenda

Particularly the modification of activities to be carried out in processes is very important. The communication support provided for the discussed types of activities is deliberately used by human actors in our enaction experiments done so far. This motivates the assumption that the process enaction facilities implemented in LEU are experienced as a relief at least if a huge number of objects is to be administrated.

References

[1] R. Adomeit, W. Deiters, B. Holtkamp, R. Rockwell, F. Schülke, and H. Weber. *Software Engineering Environments and Software Factories.* 1992.

[2] S. Bandinelli, M. Braga, A. Fugetta, and L. Lavazza. *The Architecture of SPADE-1 Process-Centered SEE.* In B. Warboys, editor, *Software Process Technology - Proceedings of the 3rd European Software Process Modeling Workshop*, pages 15–30, Villard de Lans, France, February 1994. Springer. Appeared as Lecture Notes in Computer Science 772.

[3] R. Barker. *CASE*Method Entity Relationship Modelling.* Addison-Wesley, Wokingham, England, 1990.

[4] A. Broeckers and V. Gruhn. *Computer-Aided Verification of Software Process Model Properties.* In *Proceedings of the Fifth Conference on Advanced information Systems Engineering (CAiSE)*, Paris, France, June 1993.

[5] G. Chroust. *Application Development Project Support (ADPS) an Environment for Industrial Application Development. ACM SIGSOFT Software Engineering Notes*, 14(5), July 1989.

[6] T.H. Davenport and J.E. Short. *The New Industrial Engineering: Information Technology and Business Process Redesign. Sloan Management Review*, 1990.

[7] W. Deiters and V. Gruhn. *Managing Software Processes in MELMAC.* In *Proceedings of the Fourth ACM SIGSOFT Symposium on Software Development Environments*, pages 193–205, Irvine, California, USA, December 1990. Appeared as ACM Software Engineering Notes, 15(6), December 1990.

[8] W. Deiters and V. Gruhn. *Software Process Technology Transfer - A Case Study Based on FUNSOFT Nets and MELMAC.* In *Proceedings of the 8th International Software Process Workshop*, Schloss Dagstuhl, Germany, March 1993.

[9] G. Erdl and H.G. Schönecker. *Geschäftsprozessmanagement - Vorgangssteuerung und integrierte Vorgangsbearbeitung (in German).* FBO - Fachverlag f"ur B"uro und Organsiationstechnik, Baden-Baden, FRG, 1992.

[10] H.J. Genrich. *Predicate/Transition Nets.* In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to other Models of Concurrency*, Berlin, FRG, 1987. Springer. Appeared in Lecture Notes on Computer Science 254.

[11] V. Gruhn. *Validation and Verification of Software Process Models.* PhD thesis, University Dortmund, June 1991. Appeared as Technical Report No. 394/91.

[12] V. Gruhn. *Entwicklung von Informationssystemen in der LION-Entwicklungsumgebung (in German).* In G. Scheschonk and W. Reisig, editors, *Petri-Netze im Einsatz für Entwurf und Entwicklung von Informationssystemen*, pages 31–45, Berlin, FRG, September 1993. Springer.

[13] V. Gruhn. *Software Process Simulation in MELMAC. Systems Analysis - Modelling - Simulation*, 11:121–141, 1993.

[14] V. Gruhn. *Software Process Management and Business Process (Re)Engineering.* In B. Warboys, editor, *Software Process Technology - Proceedings of the 3rd European Software Process Modeling Workshop*, pages 250–254, Villard de Lans, France, February 1994. Springer. Appeared as Lecture Notes in Computer Science 772.

[15] V. Gruhn and R. Jegelka. *An Evaluation of FUNSOFT Nets.* In J.-C. Derniame, editor, *Software Process Technology - Proceedings of the 2nd European Software Process Modeling Workshop*, pages 194–214, Trondheim, Norway, September 1992. Springer. Appeared as Lecture Notes in Computer Science 635.

[16] M. Hammer and J. Champy. *Reengineering the Corporation.* Harper Business, New York, US, 1993.

[17] P. Huber, K. Jensen, and R.M. Shapiro. *Hierarchies in Coloured Petri Nets.* In *Proc. of the 10th Int. Conf. on Application and Theory of Petri Nets*, Bonn, FRG, 1989.

[18] D.A. Norman. *Collaborative Computing: Collaboration First, Computing Second. Communications of the ACM*, 34(12), December 1991.

[19] L. O'Conner, editor. *Proceedings of the 2nd International Conference on the Software Process - Continuous Software Process Improvement*, Berlin, Germany, February 1993.

[20] W. Reisig. *Petrinetze (in German).* Springer, Berlin, FRG, 1986.

[21] J. Sa and B. Warboys. *Integrating a Formal Specification Method with PML: A Case Study.* In J.-C. Derniame, editor. *Software Process Technology - Proceedings of the 2nd European Software Process Modeling Workshop*, pages 106–122, Trondheim, Norway, September 1992. Springer. Appeared as Lecture Notes in Computer Science 635.

[22] W. Schäfer, editor. *Proceedings of the 8th International Software Process Workshop.* Schloss Dagstuhl, Germany, October 1993.

[23] W. Schäfer and H. Weber. *The ESF-Profile.* In *Handbook of Computer Aided Software Engineering*, New York, 1988. Van Nostrand.

[24] C.J. Tully. *Software Process Models and Programs: Observations on their nature and context.* In *Proceedings of the 4th International Software Process Workshop*, Moretonhampstead, Devon. UK, May 1988.

[25] B. Warboys, editor. *Proceedings of the 3rd European Workshop on Software Process Modelling*, Villard de Lans. France, February 1994. Springer. Appeared as Lecture Notes in Computer Science 772.

Applying Workflow Management Concepts in Public Administration

A case study in the Austrian ministry of science and research

Robert Kristöfl[1)], Thomas Grechenig[2)]

[1)]Austrian Ministry of Science and Research
ADV Planning and Organisation
1014 Wien, Bankgasse 1/209
kristoefl@mail.bmwf.gv.at

[2)]Vienna University of Technology
Dep. of Software Engineering
1040 Wien, Resselgasse 3/2/188
grechenig@eimoni.tuwien.ac.at

Abstract. *Is the electronic document for public administration a mere marketing gag of companies selling workflow management systems or is the actual gain even high enough to justify any necessary effort of getting familiar with such systems?*

Workflow does not only promise to exchange any paper document for an electronical one as well as the event-driven handling and passing on of the future electronical document, but calls for a comprehensive reorientation of a whole business.

A redesign of any involved department-specific information system is a necessary consequence. What is still unclear is jurisdiction concerning aspects such as electronical approbation or the security of the authenticity of documents. The effects upon organizational structures and the reactions of the users resulting therein, are hardly predictable or even estimable. Nevertheless, a successfull realization calls for a complete specification of the workflow architecture and a migration concept based therupon.

Keywords: electronic document, workflow architecture, migration concept, electronic approbation.

1 Introduction

In the course of the reform of the federal administration the council of ministers passed a concept with the goal of achieving a coordinated, economic and qualified use of information technique in the public administration of all the ministeries on Jan. 28, 92.

In addition to this, the Austrian Chancellor issued a decree on establishing a coordination commission for information technology on Sept. 4, 92. According to this guidelines this commission was reorganized and constituted anew. For substantial work within the various subjects expert groups were set up.

The expert group "Kanzleiinformationssysteme und elektronische Aktensysteme" was founded for cooperation between the various ministeries. This group is concerned with upgrading the existing office informations system (KIS) towards a full document routing system. The main tasks of this working group have been

-) defining a general catalogue of criteria serving the purpose of software and hardware system evaluation.
-) defining a strategy for inhouse-development.
-) defining a concept for setting up KIS within the organization.
-) checking and guaranteeing (or even changing) the corresponding legal framework.

The whole work was based upon the manual of office organization within the federal administration issued by the expert group "Kanzleiwesen, techn. Kommunikation und Dokumentation" on Dec. 14, 92.

Getting the concept into effect is the task of every single ministry on its own: The existing information technology infrastructure has to be adapted to the general requirements, the special functional range for the workflow management system within a certain ministry has to be defined and the interfaces between the single components have to be specified.

Nowadays available workflow management products have been designed for specific domains with a broad variety of organizational backgrounds. They all have their strengths and weaknesses. None of them fits the requirements of the Austrian federal administration so long. Developing an effective strategy that can combine as well the process of adaption and configuration of industrial workflow management systems as also inhouse software development and maintenance will be the main task for the two years to come.

2 Current Status of automated Information Systems

A few years ago CASE was called the one and only development environment in the area of software engineering; nowadays critical voices from the side of the providers of object-oriented development environments and also workflow management systems are heard.

Workflow systems promise easy and flexible modelling with help of graphical editors, this is how highly complex applications can be divided in simple processes and can be handled in an event-driven way. So it seems that the problems in introducing these new technologies are not caused by technical but rather by organizational difficulties.

This euphoric thoughts cannot be supported to that high extent: On one hand the methods that are supported by CASE and the model for software development based on upon CASE are well approved and on the other the shift from a merely function-driven towards a process-driven approach does not change the complexity of software at all. On the contrary: The institutionwide process-oriented view will augment complexity and almost demands the use of methods for achieving a certain software quality. Figure 1 shows the structure of the currently used functionally decomposed information system with the meta-data of the paper document that are kept parallel.

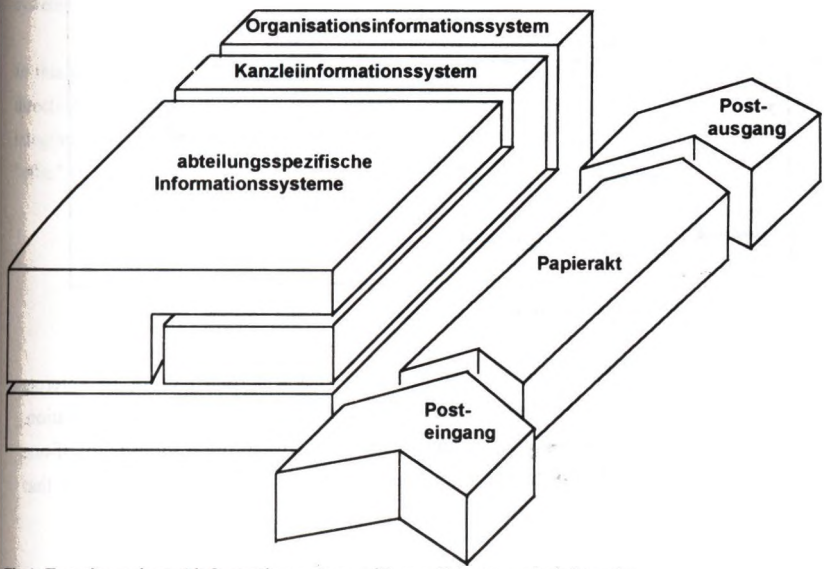


Fig.1: Function-oriented information systems with parallel paper administration

The structures of the information models mostly depend on the demand of information for closed domains of application as seen from a single department. The integration as such is achieved with the parallel written piece of information, especially the paper document. What you try to avoid is a redundant keeping of data when using meta-data of paper documents in common.

Figure 2 shows the connectivity of the information models of organization, office and a certain department-specific use for non-regular dotation.

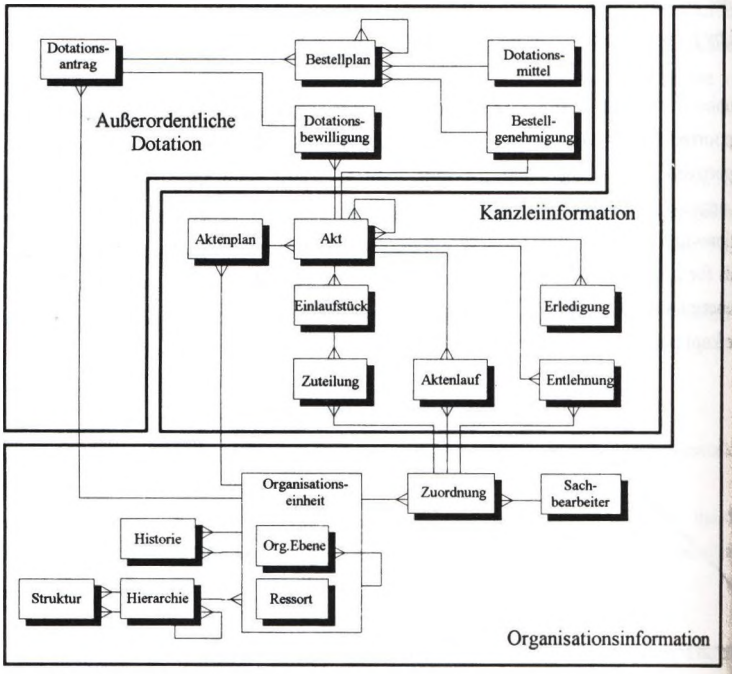


Fig.2: Connectivity of the information models for organization, office and non-regular dotations.

From the office-point-of-view it is of no use to realize that a certain part of an entering document in connection with non-regular dotation contains the application for the dotation. And in the same way there is no need to see the connection between the responsibility of one clerk for a certain entering document and the corresponding clerk of the acceptance of that namely document for non-regular dotation.

For an office a whole process begins with the receipt of an abstract entering document and ends with the mailing of finished outcomes. For a certain department on the other hand a

whole process contains the whole procedure of the application for non-regular dotation up to the single acceptances. In the course of this process the closed process of the office is passed several times in different shapes.

3 Information Flow within the Organization

Although the development of information systems is well supported through methods and although the realizations work well for special ratings and calculations there is a strong demand for integration of the documents themselves.

The reason is that on one hand the necessary effort for the data input is quite high and on the other that the predefined meta-informations are no longer sufficient for some necessary operations on them. A fast access to information - in an analogous way of information systems - is more and more required also for the contents of the documents.

Figure 3 shows the possible variations of dataflow from incoming mail to outgoing mail. This scheme shows the width from non-predefined ad hoc work-steps up to the clearly defined work sequence according to the office regulations. Ad hoc work sequences are at the time easiest covered with user-oriented active office-automation- or groupware-systems. On the other hand, the strongly formalized document flow calls for the use of workflow management systems.

In this connection it is clear that the days of proprietary complete solutions are over. The direction to go on can only be a cooperation of differing systems as parts and components in integration and no longer the use of parallel systems that exclude each other and very often "offer" noncompatible functionality for very similar fields.

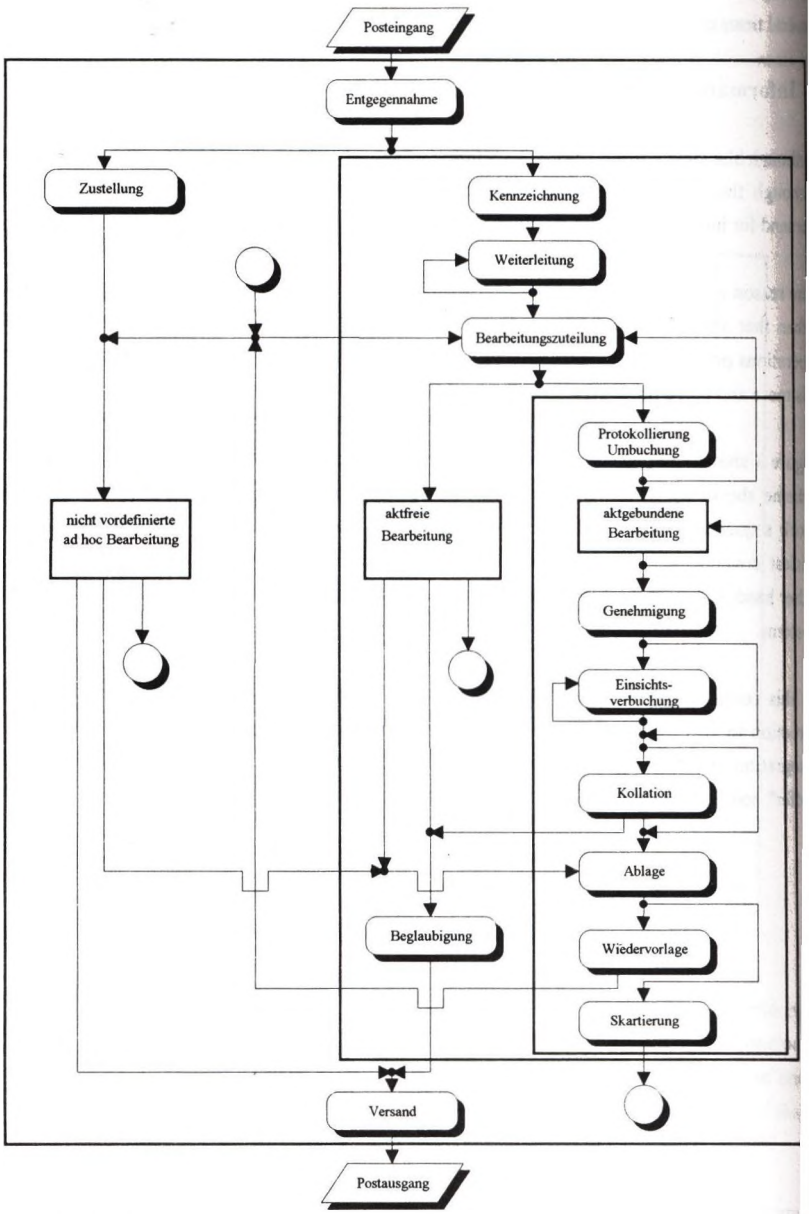


Fig.3: Scheme of document flow

4 A Conceptual Workflow Architecture

The scheme of the document flow which is shown in figure 3 causes a demand for a workflow architecture that is built in several layers as shown in figure 4.

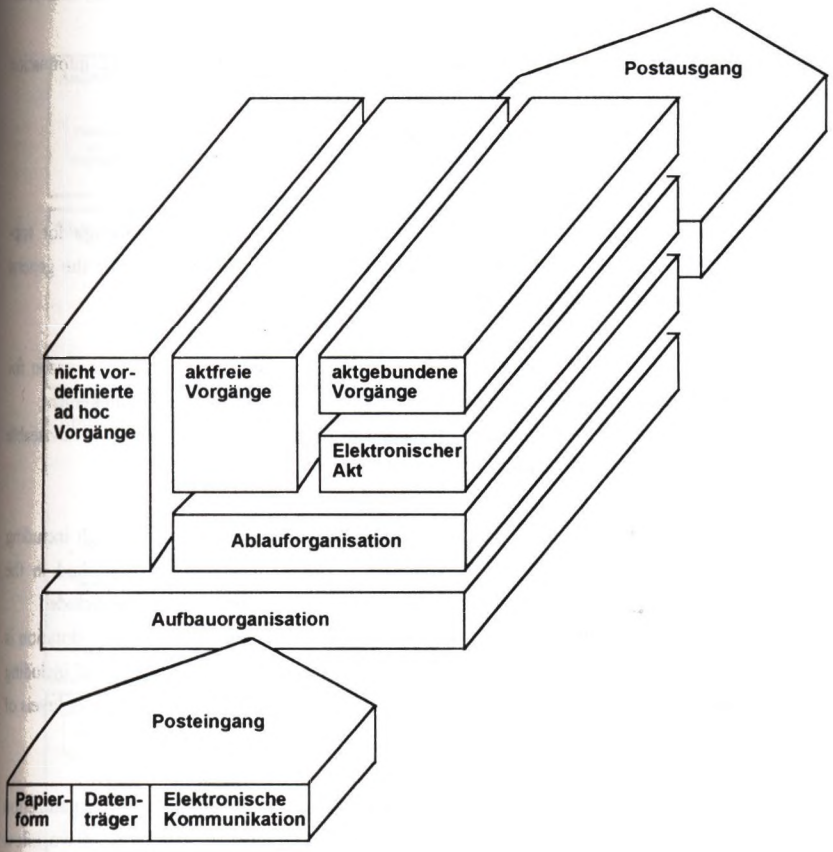


Fig.4: Workflow architecture

The internal definition of the organization as a lowest layer of architecture describes the organizational structure, and their inner structures and hierarchy in its static and dynamic form. With the help of an organigram the single clerks are given their share of a whole task that belongs to a certain department. There are several sender- and receiver-directories set up to enable automation in receiving and delivering the mail income and the mail output.

This definition of the organization belongs to non-predefined ad hoc procedures as well as also to the definition of the workflow in the ministry based therupon. Its main part is the definition of the document path of the marked mail items that turn up with incoming mail and of the documents created during any process, may it be highly or less strictly formalized. Most important and therefore central is the document flow of the electronic document based thereupon.

The majority of the involved procedures of now supported department-specific information systems belong to the strictly formalized procedures of the highest layer.

5 Migration

Electronic approbation nowadays is seen either as a taboo or as the challenge for top-management. Not, for the reason that the legal situation is still unclear, but for the general reservations towards active participation.

Deny of that participation or handing the duty on to one´s secretary is another reason for calling for more efficient and more flexible structures in the hierarchy of lean management.
In order to create a successfull migration strategy and to get it working then, it is adviseable not to put electronic approbation in the first place.

The first step of migration is in making information models object models through including the corresponding document objects. In addition to the structured data described in the entities, texts, tables, graphics and sometimes also sound or animation have to be included.
The corresponding model for the strictly defined workflow in case of non-regular dotation is shown in figure 5. In contrast to figure 2 the meta-data are reduced with the help of including the documents, they could be unwoven in many places and thus could be put on lower levels of architecture.

As a next step in including a workflow-management system the definition of the organization has to be set up. This will cause the first problems then. The internal structures of workflow systems are usually not put open and normally only to be included through API´s. Besides, the quite complex relations of existing information systems are hardly the same in definition and structure. In this field it will be difficult to come closer if you stick to the demand for a common, united and non-redundant definition of the organization.

The next step of migration is towards the organization of workflow. Nowadays systems give sufficient support for document routing. In case the highly or less strictly formalized individual steps of tasks handling consist of functions only, they can be integrated in a provided workflow

processing without any difficulties. If they are composed of functions and flows at the same time, the surrounding workflow processing must go on without any breaks in the individual process, though. And in the same way document routing should not be seen as a closed process, as in the case of ministrywide processes there are several single document routings that differ from each other to finally get the outcome of the whole process. The single document routing must be an integrable part of the complete workflow.

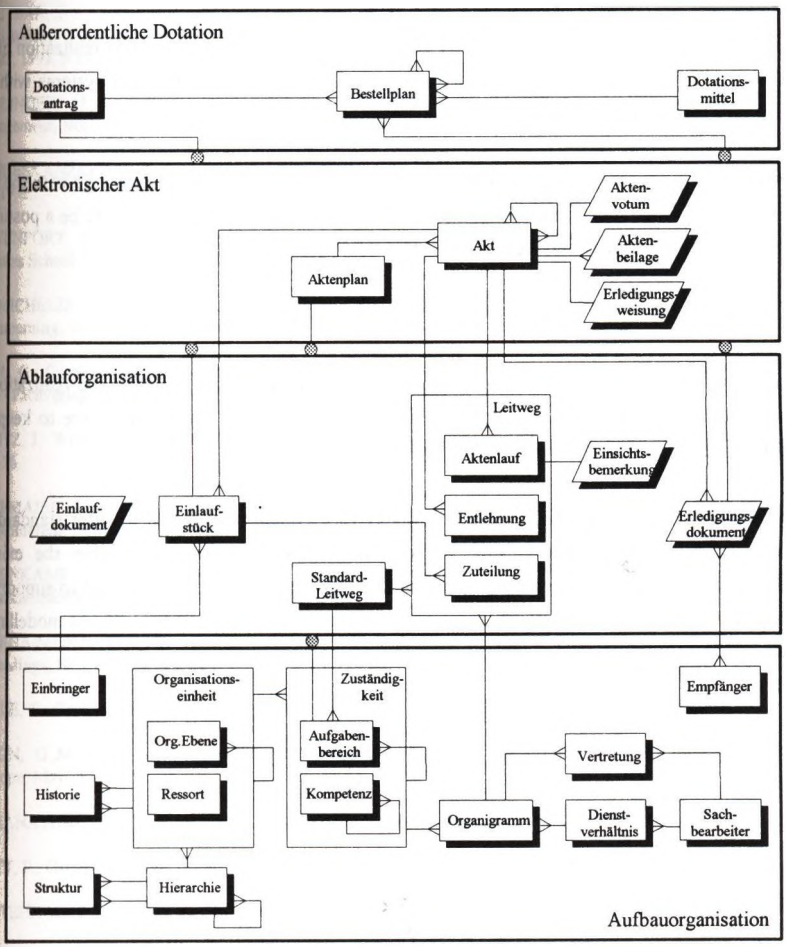


Fig.5: Object model for the strictly defined process for non-regular dotation.

In this domain there will be a demand for interfaces for the CASE dictionary and the workflow systems, if automatic generation of the specified process models should be possible. To approach a more flexible use, a dynamic interpretation of the workflow processing out of the data dictionary during the runtime would be a much more effective solution. In this way a generation or programming of the workflow processing would no longer be necessary.

For the internal view within the ministry there is a last step of migration: The realization of the individual processes. In most cases a redesign of the existing information systems with the necessary reductions and additions will be inevitable.

An interesting aspect for the future is the external electronic document exchange with the help of EDI. Standardization from the side of the suppliers of workflow systems will be a postulate, to make communication between various workflow systems possible.

6 Conclusion

The above explanations are meant as arguments to accept the challenge of the realization of electronical documents with the help of workflow management. Yet, you'd have to keep the effort to get acquainted with the technology in mind.

And, above all, there is a need for the suppliers to be open-minded and to offer standardized interfaces to a higher extent, such as to make a more flexible use within the existing information technology systems possible. And in addition to this, there is a need to support the design through the use of methods for the organization modelling and the process modelling as well.

References

ABI, R.: Workflow Automation: The new Competitive Edge - Tools Techniques and Software, Unitech International Corporation, 1992.

BAECKER, R.: New Paradigms for Computing in the Nineties, Department of Computer Science, Univ. of Toronto, Canada, Graphics Interface 1991, p.224-229.

BLOOR, R.: Workflow and Document Management, DBMS, Vol.7, No.10, 1994, p.12-14.

BREITBAR, T.Y.: Merging Application-centric and Data-centric Approaches to Support Transaction-oriented Multi-system Workflows, Department of Computer Science, ETH Zürich, Switzerland, Sigmond Record, Vol.22, 1993, p.23-30.

BULLINGER, H.,J.: Informations- und Kommunikationssysteme für schlanke Unternehmungen, Office Management, Vol.31, 1993, p.6-19.

BUNDESKANZLERAMT: Handbuch für die Kanzlei- und Büroorganisation in der Bundesverwaltung, BKA, Dez.1992.

DAVENPORT, T.,H.: Process Innovation - Reengineering Work through Information Technology, Havard Business School Press, Boston, 1993.

DE MICHELIS, G.: Routines and Conversations, Dept. of Computer Science, Milan Univ., Italy, Structured Programming, Vol.14, 1993, p.110-118.

ERDL, G.: Geschäftspprozessmanagement- Vorgangssteuerungssysteme und integrierte Vorgangsbearbeitung, FBO - Fachverlag, Baden-Baden, 1992.

GANTZ, J.: Workflow Software: Working magic or wreaking havoc ?, Networking Managment, Vol.3, 1993, p.75-76.

GAPPMAIER, M.: Kooperative Vorgangsbearbeitung, Eine Untersuchung der Arbeitsorganisation, Fachkonferenz Dokument '93, Wien, 1993.

HASENKAMP, U.: Konzepte und Einsatzmöglichkeiten von Workflow-Management-Systemen, Wirtschaftinformatik 93, Heidelberg, 1993, p.405-422.

MEDINA-MORA, R.: The Action Workflow Approach to Workflow Management Technology, CSCW Proceedings, Nov.1992, p.281-288.

NOLLE, T.: Groupware: The Next Generation, Business Communication Review, Aug.1993, p.54-58.

OLSON, G.,M.: Defining a metaphor for Groupware, Software Engineering Institute Pittsburgh, IEEE Software, May 1992, p.93-95.

RAMANATHAN, J.: The Workflow Process Paradigm, Oracle Magazine, Vol.VIII, No.3, 1994, p.120.

WATT, P.: Groupware, One for all, all for one, Database Programming and Design, Jan.1994, p.25-32.

WHITEHEAD, R.: Software for Groupies, IBM System User, Feb.1993, p.47-48.

Distributed Check Processing based on a Client-Server Solution

A. Kapusy, L. Langer

IQSOFT Intelligent Software Co.
Teleki Blanka u. 15-17.
H-1142 Budapest
Hungary

Summary

This paper describes a practical example of processing paper-based documents through recognition that is an increasing demand nowadays. The application was installed at the Hungarian Post Clearing (Accounting) Center and is capable of recognizing, processing and archiving internal postal documents.

The system has a client-server structure, hardware platform independent, open system, of which the main processing unit is an IBM RS/6000 minicomputer containing ORACLE relational database, while the workstations are IBM PC-s with MS-Windows to run client programs. The system can easily be configured with both a small number of workstations and hundreds of them working in clusters.

The system is able to process any kind of paper-based documents, e.g. shares (securities), bonds, cheques, vouchers, questionnaires and other document forms, using numerous parameters in program modules.

1. Introduction

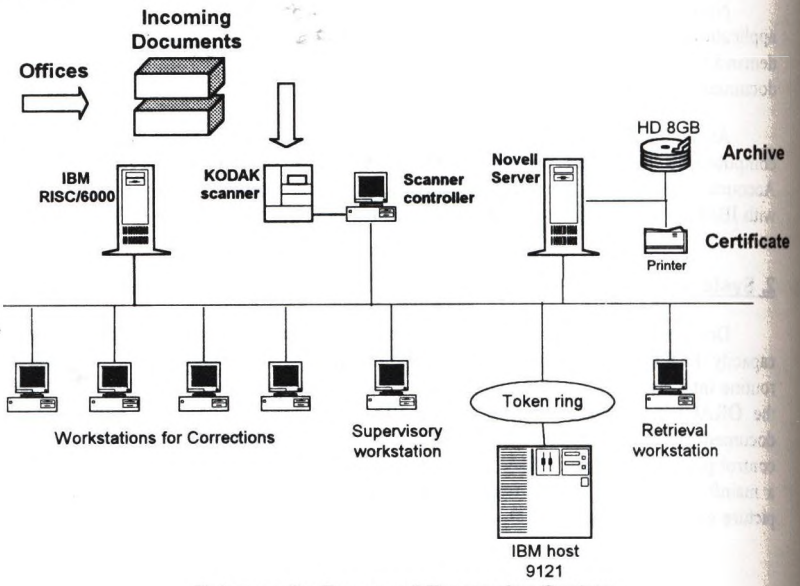
Nowadays computer systems used to process paper documents find their application in increasing number of fields. In addition to archiving there is a growing demand to recognize and process information both printed and handwritten from these documents.

At the Hungarian Post one of the most important developing trend is the computerized document processing. One example of this modernization is the Daily Account (Clearing) system (called NER) that was developed by IQSOFT in cooperation with IBM as main contractor and it was installed in the first quarter of 1994.

2. System Overview

Documents coming in from branch post offices are the input to the system. A high capacity KODAK 900 scanner reads in these papers sheets first, then a recognition routine interprets the numbers indicated on them. After checking the data will be fed to the ORACLE database residing in an IBM RS/6000 minicomputer. Any erronous documents or unrecognized numbers will be corrected at IBM PC based workstations. A control program written in MAGIC reads the corrected documents and transfers them to a mainframe system for further processing. Data from documents and their scanned picture will be stored in an archive database.

The schematic diagram of the document processing system is shown in the figure below.



Schema of a Document Processing System

2.1. Overview of postal processes

The Daily Account (Clearing) System (NER) is capable of accounting the daily transactions of the Hungarian post offices. Each post office in Hungary generates a report from their (cash) transactions every day and send these documents to the Central Accounting Office of Hungarian Post located in Budapest.

The NER system recognizes, validates and processes the incoming documents, then a check processing system reads its output data to make cross-controls.

2.2. Main considerations in system development

In program design we attached great importance to the following aspects

- easy to use
- easy to learn
- ergonomical design
- reliability.

These aspects were important due to the high fluctuation in manpower and to the exhausting work that requires much concentration.

2.3. Dimensions of the system

We had to pay great attention to size the system correctly because of the huge amount of documents and of the limited processing time.

Nearly 3200 post offices send 20-25,000 documents as a daily average to the Central Accounting Office. There are eight types of incoming documents. One document has typically eight handwritten and two printed numbers with an average of six digits each. Six hours are available to process the documents including corrections at 12 workstations.

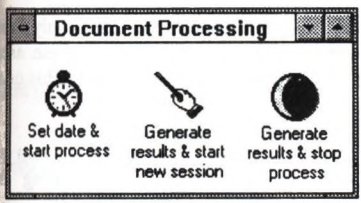
The documents are approximately of A5 size. Their scanned and compressed image is reperesented typically by 5 kByte. The daily volume of images to archive is about 120 MByte including their corresponding data.

3. The structure of the system

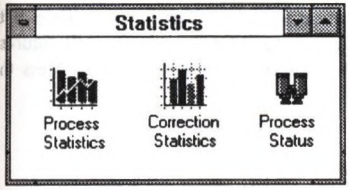
3.1. Supervisory workstation

The functions of the NER system is controlled from the supervisory workstation. Early each day, the supervisor enters the date of documents to be processed, starts the character recognition routine and the document checking program, starts the correction sections and generates results at the end of sections and, finally, stops the system in the evening.

In princip, supervisory workstation can be any of the PC-based workstations but only one can exist in the system. Because some of the tasks will run on the RISC computer having AIX operating system that is rather complicated for a non-professional user, we implemented Windows icons that communicate through "remote shell" commands with the UNIX system. Here are some examples of these icons.



The "Document Processing" window contains icons to normal system processes like starting a day, starting a session, generating result and stop the system.



The NER system generates statistics automatically at the end of sessions and end of each day, but some of them can also be initiated from a supervisory window.

The supervisor has some administrative tasks like to add or remove corrective worker, to give access rights for them, etc.

3.2. Document Scanning

The NER system uses a KODAK ImageScan 990 scanner. The throughput of the scanner is 120 pieces of A4 size documents per minute that is 12-15,000 pieces postal documents per hour.

The KODAK scanner has red drop-out and on-board data compression to reduce file size to an average 5 kByte per document. It contains an automatic document feeder, an exit hopper and a built-in document printer, it detects jams and skews and it can record the documents parallelly onto microfilm.

The scanner controller program runs on a PC under MS-Windows. The program reads the images from the scanner in batches and forms packages for further processing. The program detects skewed and back-sided documents, and in such a case the operator has the choise of ignoring or rescanning the faulty documents. The scanned images are continuously displayed in a small but resizable window, which enables the monitoring of the image quality.

3.3. Character recognition

The character recognition program reads the image files from the disk of the database server and recognizes both printed numbers (OCR) eg. location code and handwritten numbers (HNR). The program creates descriptor files with the recognized data as numeric characters, unrecognized ones are represented by asterisks.

The recognition is optimized in such a way, that the program examines only those areas in the image where numbers are assumed. A parameter file defines these areas together with the character length and the type of the number (OCR or HNR). This gives an easy way to customize the routine.

Due to the very efficient algorithm and optimization, the recognition program is a very fast process, and it runs on the RS/6000 computer parallel to scanning and correcting processes.

The speed of the recognition is in proportion to the average performance of the system. The recognition time of a document containing 8 OCR and 40 HNR figures is less than 0.5 second. The percentage of unrecognition or misrecognition is less than 0.1% for OCR and 3% for HNR.

3.4. Correction

After scanning and recognition the data will be stored in the ORACLE database. The program validates the recognized numbers, controls the cross-relations, if given, and selects the documents to be corrected.

The corrective workstations are IBM PCs and the corrector programs runs under MS-Windows. The program reads the erronous documents and their data from the database or collects all documents sent from the same post office if there is any cross relation error among them (certain types of documents have cross-relations).

Entering the corrective program

A corrector can enter the corrective program by giving his/her name and the password. The program check the access rights and prohibits unauthorized access. On each document corrected, the corrector's name will also be indicated by storing it also in the database.

Structure of the corrective screen

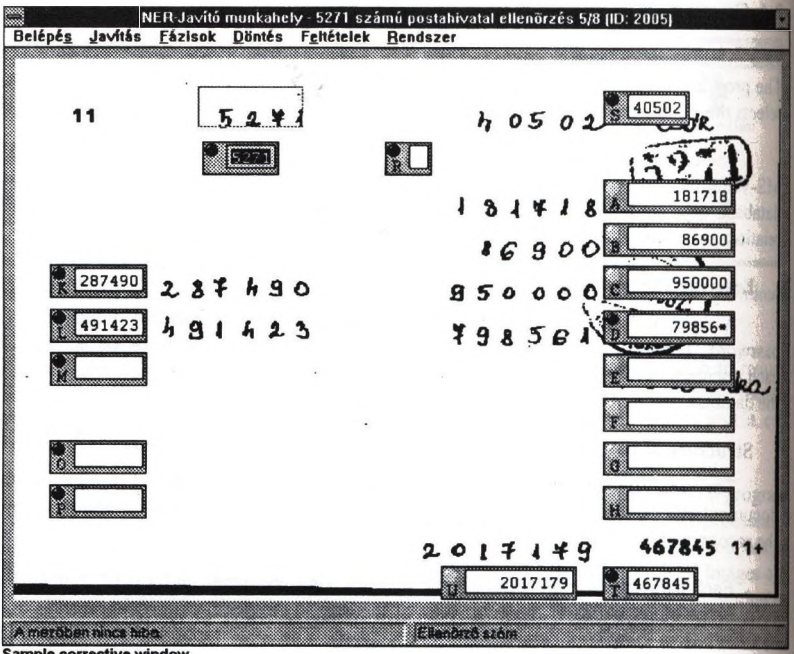
The window of the corrective program contains the scanned image of the document, menu line, status line and the program generated so-called data windows with the recognized numbers

Each data window has the data, an identification character and a small bullet. The color of the bullet indicates the error type of the data. The identification character is used as a hot key to jump to a data window immediately. The error types and their colors can be eg. as follows:

no error	—	black
recognition error	—	red
date error	—	blue
logical error	—	pink
checksum error	—	yellow
forced checking	—	white
relation error	—	magenta.

The color of the data window helps the corrector to find the data to be corrected and the reason that caused the error.

The figure belowshows an example of the correction screen (with Hungarian texts).



Sample corrective window

Identification keys are not the only way to move on the screen. Some special keys (eg. plus, minus, asterisk -- all located on the numeric keypad) help to jump into windows containing erroneous data. Of course the mouse can also be used but it is much less effective.

The user can define the parameters of data windows like position, identification code or their sequence either interactively or by editing the parameter file. Interactive modifications are subjects to special authorization.

There are other special keys for different special action, like to hide and unhide one or more windows if they cover some interesting area of the image.

The corrective program has special functions. One can rotate the image (usually it is done automatically), create new documents to replace missing ones, search for specific documents or print an image with or without data.

Status line

The last line within the correction window is divided into two parts. The first part consists of an error message belonging to the actual data window while the second part contains the identification of that window.

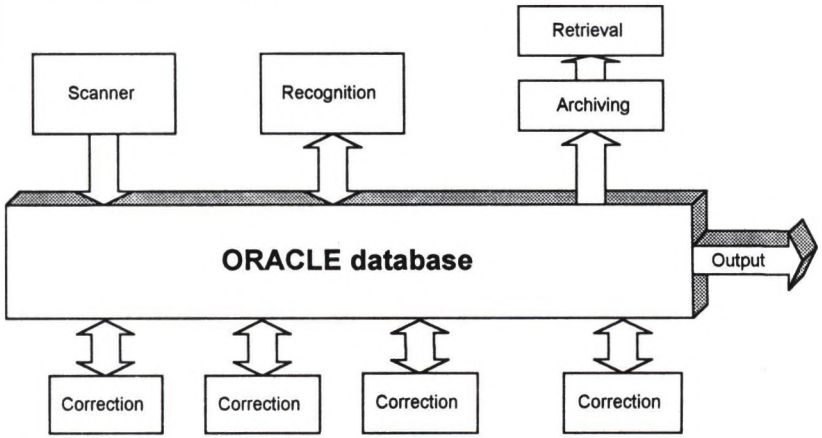
3.5. Archiving

The archived data are stored in an archive database server. From this database all data can be retrieved with their image. Search condition can be defined to any field with different logical operations(greater, less, before, after, etc.).

The 8 GB disk capacity of the archive server can store about 10-12 weeks of data. To guarantee the 5-year archiving time defined by Hungarian law, the data can be copied onto magneto-optical disks at regular intervals.

4. Document-Flow

The following picture shows the document-flow in the NER system.



Document-flow

Each document has some status information about its phase in the working process. So the system always knows the location and status of the document, the results of different process stages, who corrected it and what the next step is. Information is registered in the database and a report can be generated if necessary.

Information mentioned above together with the security features built into ORACLE ensures the NER system to be highly reliable and robust.

5. Fields of applications

Due to its highly configurable and parameterized programs used, the system can be used in such applications where the task is to scan, recognize, correct and archive documents. Such fields include processing of shares, checks, forms and questionnaires.

The distributed and client-server architecture of the NER system guarantees its efficient application in both a small application with a few correction workstations to

process a few thousands of documents per day and in a large, clustered system processing about million documents a day.

Modelling Workflow

Qualitative and Quantitative Analysis of Business Workflows using Generalized Stochastic Petri Nets *

A. Ferscha

Institut für Angewandte Informatik und Informationssysteme
Universität Wien
Lenaugasse 2/8, A-1080 Vienna, AUSTRIA
Email: ferscha@ani.univie.ac.at

Abstract

The Petri net formalism, due to its graphical modeling support and profound mathematical background, has been successfully used in the analysis of systems that obey concurrency, synchronization, mutual exclusion and nondeterminism. Computerized tools supporting the model development process and automating the analysis are available today. In this work we propose the tool supported analysis of business workflows represented by Petri net models with stochastic transition timing. Qualitative analysis by applying well developed techniques allows to answer questions related to behavioral properties of the workflow system. Performance characteristics obtained from quantitative analysis can be used for workflow performance tuning, business process reorganization and throughput optimization.

1 Introduction

Analysis and modeling of information systems in business organizations has traditionally focused on data objects, data flows and the transformation of data. With this approach however, only the organization's data and those parts of business processes that interact with the data is addressed. More recent advances in information system modeling also respects interaction and coordination of processes not necessarily generating or transforming data. As a consequence,

*This work was supported by a grant from the *Oesterreichische Nationalbank*, Austria, Project No. 5096.

the information technology in business enterprises used today is – beyond transaction processing - also for communication and coordination. A successful integration of these systems into the enterprise often requires *modeling* of the organizational process.

Detailed business process modeling is required for example for the purpose of redesigning an enterprise's business process to make the organization more efficient (*Business Process Reengineering*). Another example would be for the purpose of establishing advanced coordination technology in the organization, i.e. provide automated support for the management of dependencies among the interacting agents (humans, machines) responsible for executing a business process by using e.g. distributed, networked environments. Finally, process-driven software development environments, i.e. computerized tools to integrate software related management into the software development process highly depend on reliable business organization models.

With todays growing needs for involving automation in those areas, business process modeling becomes a vital issue, not only in analyzing, redesigning, performing and controlling work, but also in allocating responsibilities among humans and machines. What distinguishes business process modeling from other modeling disciplines is that many of the phenomena modeled must be performed by humans ("human-executable business processes"), not (only) by machines [Denn 94]. Rather than focusing solely on the human-machine interface, the flow of information into/from the machine and the transformation of data therein, business process modeling aims at a description of the active and passive agents and on their interacting behaviors, regardless whether a computer is involved in the transaction or not.

2 Workflow Modeling and Analysis

2.1 Terminology

In order to be able to argue upon business process phenomena we first restrict ourselves to a terminology abstract enough to support modeling in a case, environment and purpose independent way, and furthermore to be able to employ formal and potentially automated analysis frameworks. We define the following terminology (abstracted from [Holt 85]):

Process Element An abstract, atomic piece of work conducted as a contribution in the fulfilment of the organizations goals will be referred to by the term *process element*. No internal substructures of a process element will be considered at the abstraction level of its representation in a workflow *model*, thus it can be viewed as indivisible when being performed. *Work*, in this framework, will be seen as a product created or modified by the enactment of a process element. A process element may require a *resource* or a *set of resources* for their execution, the execution itself may or may not consume time.

Process A *business process*, or equivalently a *process*, is a set of partially ordered process elements. The time-interval of the execution of the constituent process elements is determined by the *structure* of the process, i.e. the causal (precedence) relation among process elements, together with the availability of resources necessary to execute them. The resource requirement of a process is the union of process element resource requirements.

Agent An *agent* is an actor (human or machine) which performs one or more process elements or processes. Agents represent the set of resources required to execute processes. Their availability over time (during the execution of a business process) is determined by the precedence order and execution time of assigned process elements.

Role A *role* is a coherent set of process elements to be assigned to an agent as a unit of functional responsibility. We say that an agent participates in the execution of a process in a certain role, or in other words, the process elements assigned to one agent defines the role of his involvement in the execution of the process. A single agent can participate in the execution of more than one business process. In this case his involvement is described by the set of roles he performs.

Work The product of the execution of a process element is called *work* in its most abstract sense. Since *work*, as initially assigned, or as created as the outcome of the execution of a process element represents the input to subsequent process elements, we talk about *workflow systems*, and *workflow models* in their representation as models.

A workflow model therefore is an abstract description of an actual or proposed set of business processes composed by a set of selected process elements. The model appropriately has to describe the input and output relation of work for every process element, and above that, the assignment of process elements to the responsibility of agents. The latter necessitates the preservation of causal (flow) relationships defined in processes, but also the work scheduling and management strategy applied by an agent to execute the assigned process elements. This is essential especially for agents acting in several roles (involved in more than one process).

An important aspect of a real system of business processes is its *temporal behavior*. For a model to be adequate in this respect it, must reflect delays and durations of process element executions. As a consequence, the modeling formalism has to provide sufficient expressive power to characterize the time dynamics of the system.

In the next section we shall map the abstracted framework of business process systems into the domain of Generalized Stochastic Petri Nets (GSPNs) [Ajmo 87], and show how those systems are modeled and analyzed using GSPNs.

2.2 Petri Net Workflow Modeling

Petri nets (PNs) [Mura 89] have been widely and successfully used as a good modeling tool for the qualitative and quantitative analysis of asynchronous concurrent systems with synchronization, nondeterminism, conflicts (choices) and sequencing. Fields of applications in science and engineering, range from performance and reliability modeling, communication protocols, parallel and distributed software systems, formal languages, flexible manufacturing systems and automation control to VLSI, programmable logic, compiler technology, operating systems and, recently, also to man machine interaction, *computer supported cooperative work* and *office information flow modeling* [Holt 83], [Flei 89], [Elli 93], [Proc 94].

Through the rest of this work we will be concerned with the analysis of work*flows*, and shall for our arguments only use the standard definition of generalized stochastic Petri nets as in

[Ajmo 87]. Clearly, for other important issues related to workflow-modeling like state dependencies, data driven triggering, comprehensiveness of models, etc. further concepts are needed: (*i*) *data types* for state descriptions, (*ii*) data dependent enabling semantics, data dependent timing, (*iii*) explicit representation of control activities, (*iv*) modularity and hierarchy. With another work, using *colored timed Petri nets*, we shall demonstrate how (*i*) through (*iv*) can be successfully handled. We recall: a Petri net (PN) is usually denoted by a tuple $(P, T, F, W, \mu^{(0)})$ where P is the set $(p_1, p_2, \ldots p_{|P|})$ of *places*, graphically presented as circles, T is $(t_1, t_2, \ldots t_{|T|})$, the set of *transitions*, drawn as bars, and $F \subseteq (P \times T) \cup (T \times P)$ defines an input- output relation to and from transitions, graphically represented by a set of directed *arcs*. As a graph, PN is bipartite among P and T with (a set of) input arcs $I \in (P \times T)$ pointing from P to T and output arcs $O \in (T \times P)$ pointing from T to P. Similarly we look at input/output-relations as functions $I \subset (P \times T)$ and $O \subset (T \times P)$, s.t. the input places to $t \in T$ is denoted by $I(t)$, and the set of outputplaces of $t \in T$ by $O(t)$; similar for places. $W : F \mapsto I\!N^+$ assigns weights $w((p,t))$ to arcs $(p,t) \in F$ to denote their multiplicity. $\mu^{(0)}$ is a *marking* (vector) generated by the marking function $\mathcal{M} : P \mapsto I\!N^0$, expressing for every p the number of *tokens* $\mu^{(0)}(p)$ initially assigned to it. The dynamic behavior of a PN is described in terms of two rules:

(*i*) (*enabling rule*) A transition $t \in T$ is *enabled* in some marking μ iff each of its input places holds a "sufficient" amount of tokens, i.e. iff $\forall p \in I(t)$, $\mu(p) \geq w((p,t))$ in μ. $E(\mu)$ is the set of all transitions enabled in μ. Every $t \in E(\mu)$ may or may not fire.

(*ii*) (*firing rule*) When a transition $t \in T$ fires in μ, it creates μ' by removing a certain amount of tokens from its input places and depositing a certain amount of tokens in its output places: $\forall p \in I(t) \cup O(t)$ $\mu'(p) = \mu(p) - w((p,t)) + w((t,p))$. The firing of t in the marking $\mu^{(i)}$ (reached after i other firings) is denoted by $\mu^{(i)} \xrightarrow{t} \mu^{(i+1)}$.

Inhibitory arcs invert the enabling rule in a way that $\forall p \in I(t)$, $\mu(p) = 0$ if (p,t) is an inhibitory arc. *Priority levels* can be assigned to transitions such that only transitions enabled at the highest priority level are considered for firing. (PNs with either inhibitory arcs or priorities raise the expressive power of PNs to that of Turing machines.)

A marking $\mu^{(j)}$ is said to be *reachable* from marking $\mu^{(i)}$ if there exists a sequence of transitions $\sigma = (t_k, t_l, \ldots)$ such that $\mu^{(i)} \xrightarrow{t_k} \mu^{(i+1)} \xrightarrow{t_l} \mu^{(i+2)} \ldots \xrightarrow{t_m} \mu^{(j)}$, or $\mu^{(i)} \xrightarrow{\sigma} \mu^{(j)}$ for short. We refer to the set of states reachable from $\mu^{(0)}$ by *any* σ as the *reachability set* $RS(\mu^{(0)} = (s_1, s_2, \ldots s_{|RS|}$ of the Petri net.

In the standard PN definition, state changes as induced by transition firings are instantaneous. When interested in *quantitative* (performance) aspects of business workflow systems, we must consider state changes to take time. Several time extensions of the basic Petri net model have been proposed in the literature, allowing the integration of both qualitative and quantitative analysis in a single modeling framework . In Generalized Stochastic Petri Nets [Chio 93] state change delays can be expressed by assigning *enabling delays* to transitions $\tau : T \mapsto I\!R$. We distinguish three important cases of firing delays $\tau(t_i)$, or for short τ_i, namely the case where $\tau_i = 0$ (*immediate transitions*), i.e. the state change caused by the transition firing takes zero time, and the case where τ_i is an instance of an exponentially distributed random variable

$\tau_i \sim exp(\lambda)$ (*stochastic timed transitions*). If the set of transitions contains only stochastic timed transitions where the firing delay random variable is exponentially distributed, we are within the class of *Stochastic Petri Nets* (SPNs), whereas if it contains both stochastic timed and immediate transitions, we are within the class of *Generalized Stochastic Petri Nets* (GSPNs). (The graphical representation of an immediate transition is a *bar*, *filled boxes* stand for timed transitions.) The enabling and firing rule for GSPNs are derived as follows:

(*i*) (*GSPN enabling rule*) A transition $t \in T$ is *enabled* in some marking μ *at time* $\mathcal{T}$, iff $\forall p \in I(t)$, $\mu(p) \geq w((p,t))$ in μ, similar to PNs. $E_{\mathcal{T}}(\mu)$ is the set of all transitions enabled in μ *at time* $\mathcal{T}$.

(*ii*) (*GSPN immediate transition selection rule*) Let $E_{\mathcal{T}}(\mu)$ contain *only immediate* transitions. $t_i, t_j \in T$ are said to be in *structural conflict* (t_i *SC* t_j), iff $I(t_i) \cap I(t_j) \neq \emptyset$, i.e. firing one transition will disable the other one. $t_i, t_j \in T$ are in *effective conflict* at time $\mathcal{T}$ (t_i $EC_{\mathcal{T}}$ t_j), iff at time $\mathcal{T}$ the actual marking is μ s.t. $t_i, t_j \in E_{\mathcal{T}}(\mu)$, and $\mu \xrightarrow{t_i} \mu'$ might cause that $t_j \notin E_{\mathcal{T}}(\mu')$. Clearly, due to timing, SC is necessary but not sufficient for $EC_{\mathcal{T}}$. Nevertheless, we see that if t_i, t_j are both immediate, and $t_i, t_j \in E_{\mathcal{T}}(\mu)$, then ($t_i$ *SC* t_j) $\Rightarrow$ (t_i $EC_{\mathcal{T}}$ t_j). We can assign to every immediate transition a probability $\mathcal{P}(t_i)$ with $0 \leq \mathcal{P}(t_i) \leq 1$, such that if ($t_i$ $EC_{\mathcal{T}}$ t_j), then t_i is selected for firing with propability:

$$\mathcal{P}[t_i \text{ selected for firing at } \mathcal{T}] = \frac{\mathcal{P}(t_i)}{\sum_{j|(t_i\ EC_{\mathcal{T}}\ t_j)} \mathcal{P}(t_j)}.$$

An immediate transition selected for firing *must* fire.

(*iii*) (*GSPN immediate transition priority rule*) Let $E_{\mathcal{T}}(\mu)$ contain immediate *and* timed transitions. Then immediate transitions are always selected prior to timed transitions for timing; (*ii*) must however be respected.

(*iv*) (*GSPN timed transition race policy*) Let $E_{\mathcal{T}}(\mu)$ contain *only* timed transitions, and let $RET(t_i)$, $t_i \in E_{\mathcal{T}}(\mu)$ be the remaining enabling time of t_i. $RET(t_i)$ is computed from X_i, the total enabling time, minus the time expired since the enabling instant of t_i. The transition t that *must* fire next is the one with $t|min_i RET(t_i)$.

(*v*) (*GSPN firing rule*) When a transition $t \in T$ fires in μ at T, it creates μ' similar to the PN firing rule.

Both the set of reachable states in an SPN and in a GSPN are isomorphic to a continuos time discrete state Markov chain (CTMC), allowing Markovian analysis as a quantitative evaluation. This feature will be used for the quantitative analysis of workflow models.

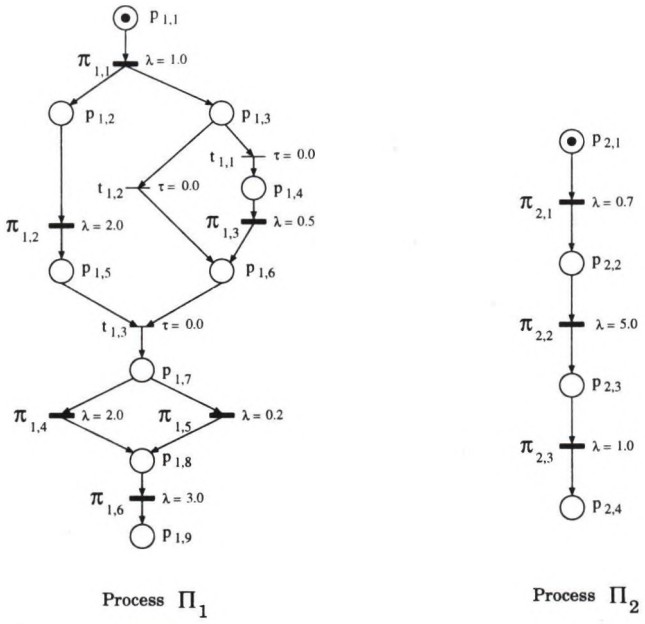


Figure 1: GSPN Models of two Processes

2.3 GSPN Representations of Workflow Models

A natural correspondence among the GSPN enabling and firing rule and the dynamic behavior of workflow systems can be exploited. We describe a process element π as a single transition $t \in T$, and the precedence relation among process elements $\pi_i \in \Pi$ that constitute a process Π by a GSPN $= (P, T, F, W, \mu^{(0)}, \tau, \mathcal{P})$. The GSPN flow relation F models the flow of work among process elements $\pi_i \in \Pi$ integrated into the set of transitions $T = (t_1, t_2, \ldots \pi_1, \pi_2, \ldots)$. We distinguish transitions $pi_i \in T$ which stand for process elements, and transitions $t_i \in T$ that do not have a counterpart in the workflow system. Transitions $pi_i \in T$ have assigned enabling delays $\tau(pi_i) \sim exp(\lambda_i)$ characterized by the parameter λ_i of the exponenetial distribution, whereas transitions $t_i \in T$ do not consume time for their enabling ($\tau(t_i) = 0.0$), but can have assigned a probability $\mathcal{P}(t_i)$ to model random selection of tokens for firing (*random switches*). Obviously the latter are used to model the structure of work flows rather than their time dynamics.

Figure 1 shows GSPN models of two (business) processes and explains the analogy. Process Π_2 is constituted by three process elements $(\pi_{2,1}, \pi_{2,2}, \pi_{2,3})$ obeying a sequential execution precedence. The process elements $(\pi_{1,1}, \pi_{1,2}, \ldots, \pi_{1,3})$ build Π_2 with a more complex precedence

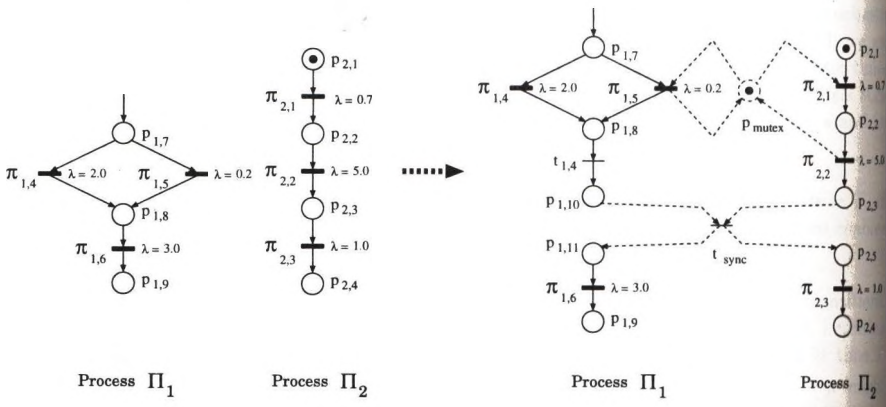


Figure 2: Modeling Process Interaction

relation: The execution of the process element $\pi_{1,1}$ spawns two pieces of work ($\pi_{1,2}$, $\pi_{1,3}$) that can be processed independently later on. As an example let $\pi_{1,1}$ represent the registration of a loan application, that invokes two subsequent processes. Future solvency has to be examined ($\pi_{1,2}$), and, for certain credit types, an instalment plan has to established ($\pi_{1,3}$). $\pi_{1,3}$ is executed with a certain probability, which is modeled using the random switch among $t_{1,2}$ and $t_{1,1}$ (by $\mathcal{P}(t_{1,2})$ and $\mathcal{P}(t_{1,1})$). As soon as both process elements $\pi_{1,2}$ and $\pi_{1,3}$ have been executed, further handling of the application is enabled. $\pi_{1,4}$ and $\pi_{1,5}$ as alternative processes model the different timings of the loan allowance, e.g. related to the requested amount. After allowance has been given, a credit account is opened $\pi_{1,6}$. Note that $\pi_{1,2}$ and $\pi_{1,3}$ in practice could be assigned to a single agent who would then execute one after each other, thus ignoring potential parallelism as a source for optimizations.

Figure 1 describes two isolated business processes. The interweaving of processes can be described by GSPN constructs, depending on the nature of process interactions. Two possibilities are envisioned in Figure 2: The marked place p_{mutex} models *mutual exclusion* among the process elements $\pi_{1,5} \in \Pi_1$ and the process elements $\pi_{2,1}, \pi_{2,2} \in \Pi_1$. An example would be the sharing of a resource among process elements (write access to a customer record). *Synchronization*, as another possible process dependency, is modeled by the transition t_{sync} in Figure 2, by which the processes elements $\pi_{1,6} \in \Pi_1$ and $\pi_{2,3} \in \Pi_2$ are forced to start at the same instant of time. With this construct it is possible to "block" one flow of work until another flow has reached a certain stage, etc. Basically every Petri net model construct can be used to describe the interdependencies among processes.

So far, with the GSPN models for processes and process interactions, nothing has been said about the assignment to agents for the proper execution of the processes. In our modeling approach, the mapping of processes to agents is implicit in the GSPN, whereas interactions

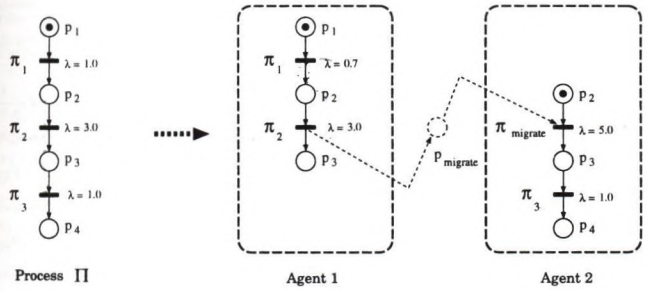


Figure 3: A Process Assigned to two Agents

among agents are again made explicit. Consider for example the assignment of the process Π (Figure 3) in such a way to two agents, that agent 1 executes $\pi_1, \pi_2 \in \Pi$ and agent 2 executes $\pi_3 \in \Pi$. Then we model the work migrating from agent 1 to agent 2 by a migration place $p_{migrate}$ and the time overhead for work migration as an explicit processes $\pi_{migrate}$ with the agent for whom additional work arises. In Figure 3 we have modeled migration overhead to arise for agent 2 only.

2.4 Quantitative Analysis of Workflow Models

When interested in *quantitative* aspects of business workflow systems, the whole markovian analysis framework can be used within the GSPN formalism to derive performance metrics. If the CTMC described by the set of reachable states is ergodic (irreducible, non-null recurrent and aperiodic) then a steady-state probability vector $\vec{p}Q = \vec{0}$ can be found, Q being the infinitesimal generator obtained from the reachability set. In other cases a transient probability vector at time τ can be derived from $\vec{p}(\tau) = \vec{p}(0)\ exp(Q\tau)$, where $\vec{p}(0)$ is the distribution at time 0. Discrete event simulation is an alternative evaluation method that can be applied in any case.

Workflow Performance Upper Bound Applying Markovian steady state analysis to the process models in Figure 1, we find an upper bound on the attainable execution performance for both processes by extending the GSPN process models with a flow loop-back (e.g. in the case of Π_1 looping tokens arriving in $p_{1,9}$ back to $p_{1,1}$ with an immediate transition) and by solving $\vec{p}^{opt} Q = \vec{0}$. $\vec{p}_i^{opt}$ for a reachable state $s_i \in RS(\mu^{(0)})$ represents the steady state probability of that state s_i. Since the analysis of processes does not consider *any* other execution constraint than the precedences of the contained process elements, $\vec{p}^{opt}$ represents the steady state probability vector under "optimal" conditions (no execution blocking due to shared resources, busy agents, synchronization constraints etc.). On the other hand, by adding process interactions, providing a limited amount of resources for process execution and by assigning processes to agents, further constraints are imposed on the flow of work within the process, thus slowing down process

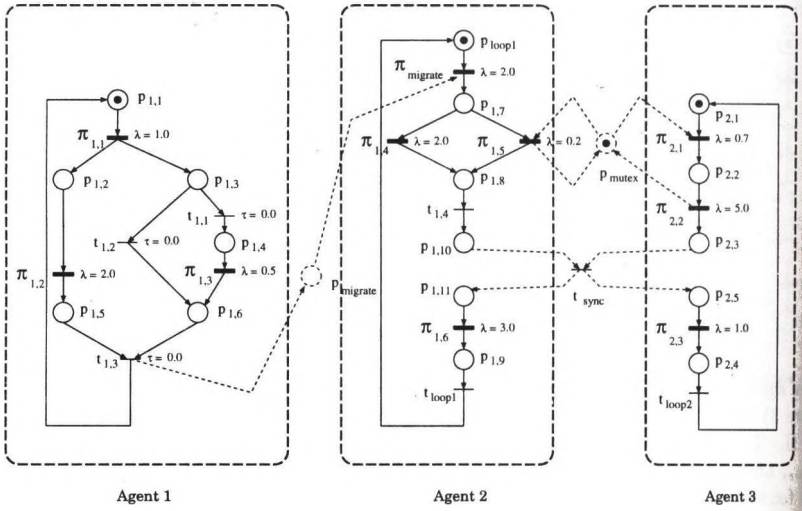


Figure 4: Workflow among Agents

execution. $\vec{p}^{vpt}$ therefore represents an ideal case that may not occur in a real execution of the business process.

As a metric of performance we consider here the frequency of transition firings, or in other words, the number of process executions per unit of time. For Π_1 and Π_2, Table 1 summarizes the "upper bound" on those frequencies for the particular parametrization of the models. Due to the timing of the model, the throughput e.g. in process element $\pi_{1,1}$ cannot be larger than 0.323847 pieces of work per unit of time, if the whole process Π_1 is executed by a single agent. Due to various other conditions, it will be even less.

Actual Workflow Performance In Figure 4 we have spatially grouped Π_1 and Π_2 so as to be executed by three agents. Agent 1 is dedicated to execute $\pi_{1,1}, \pi_{1,2}, \pi_{1,3} \in \Pi_1$, agent 2 is assigned $\pi_{1,4}, \pi_{1,5}, \pi_{1,6} \in \Pi_1$ and agent 3 shall execute Π_2. The interaction among agent 1 and agent 2 is reflected by the work migration dependency expressed by place $P_{migrate}$, agent 2 and agent 3 share a common resource (P_{mutex}), and have to synchronize their progress of work (t_{sync}). It can be seen directly that both P_{mutex} and t_{sync} represent a source of workflow delay with the consequence of performance loss. $\pi_{1,5}$ and $(\pi_{2,1}, \pi_{2,2})$ can only be executed mutually exclusiv to each other, i.e. execution of one hinders the other from possible simultaneous execution. $\pi_{1,6}$ and $\pi_{2,3}$ must start at the same time, causing one agent to idle wait for the other one. Table 1 shows, in the column with throughput data after isolating agent 1, that the throughput in Π_2 $(\pi_{2,1}, \pi_{2,2}, \pi_{2,3})$ drops from 0.380435 to 0.355895 after the assignment of processes to agents.

The interaction among agent 1 and agent 2 represents a unidirectional dependency, where

	Process Element	Parameter λ	Process Executions per Unit Time upper bound	Agent 1 isolated	3 Agents
Π_1	$\pi_{1,1}$	$\lambda = 1.0$	0.323847	0.434782	0.446953 +/- 3.45383
	$\pi_{1,2}$	$\lambda = 2.0$	0.323847	0.434783	0.444511 +/- 3.15835
	$\pi_{1,3}$	$\lambda = 0.5$	0.161924	0.217391	0.232025 +/- 0.89726
	$\pi_{1,4}$	$\lambda = 2.0$	0.294406	0.324995	0.322393 +/- 2.36941
	$\pi_{1,5}$	$\lambda = 0.2$	0.029441	0.030900	0.043962 +/- 1.11638
	$\pi_{1,6}$	$\lambda = 3.0$	0.323847	0.355895	0.363913 +/- 0.88590
	$t_{1,1}$	–	0.161924	0.217391	0.234467 +/- 0.53056
	$t_{1,2}$	–	0.161924	0.217391	0.212486 +/- 3.41533
	$t_{1,3}$	–	0.323847	0.434783	0.444511 +/- 2.67727
	$\pi_{migrate}$	–	–	0.355895	0.366355 +/- 7.01617
	t_{loop1}	–	–	0.355895	0.363913 +/- 0.88590
	t_{sync}	–	–	0.355895	0.363913 +/- 1.24648
	t_{loop2}	–	–	0.355895	0.363913 +/- 0.39892
Π_2	$\pi_{2,1}$	$\lambda = 0.7$	0.380435	0.355895	0.366355 +/- 2.07943
	$\pi_{2,2}$	$\lambda = 5.0$	0.380435	0.355895	0.363913 +/- 0.72823
	$\pi_{2,3}$	$\lambda = 1.0$	0.380435	0.355895	0.363913 +/- 0.39892

Table 1: Process Execution Performance

agent 1 takes a 'producer' role, making the work progress of agent 2 dependent on its own execution performance. Opposed to that, agent 2 cannot influence the work progress of agent 1. Place $P_{migrate}$ models asynchronous work passing interaction, raising a stability issue for the specific process assignement. Clearly, the processing speed of agent 1 has to be less than the one of agent 2, otherwise agent 2, at least in the long run, would be overwhelmed by the work generated by agent 1. In terms of the GSPN model, this means that the place $P_{migrate}$ is *structurally unbounded*, i.e. the number of tokens in $P_{migrate}$ can grow without limit. The system remains stable as long as the token arrival rate to $P_{migrate}$ is less than the token absorption rate from $P_{migrate}$. For the particular timing parameters in a steady state analysis for a model that isolates (decouples) agent 1 from the other agents we find the firing frequency of $t_{1,3}$ as 0.434783 which is equivalent to the token arrival rate, and the firing frequency of $\pi_{migrate}$ as 0.355895 which is equivalent to the token absorption rate. Since $t_{1,3} > \pi_{migrate}$, $\mu(P_{migrate})$ actually grows infinitely and a steady state solution of the GSPN model in Figure 4 does not exist. Either transient analysis or discrete event simulation must be applied as a solution method. The last column in Table 1 reports the process element execution frequencies attainable with the assignment to three agents as obtained from a simulation of the model. Figure 5 shows the token distribution in $P_{migrate}$ as observed during simulation for a simulated time interval of 100 time units. From the average number of tokens in $P_{migrate}$ we can expect on average 4.38217 pieces of work to reside in a buffer between agent 1 and agent 2.

The token distribution in the places of the GSPN workflow model in Figure 4 itself represents a useful performance metric. For instance we obtain the probability of one token being in place $p_{1,10}$ in steady state as $\mathcal{P}[\mu(p_{1,10}) = 1] = 0.54092$. Analogously we have $\mathcal{P}[\mu(p_{1,10}) = 0] = 0.45908$, $\mathcal{P}[\mu(p_{2,3}) = 1] = 0.06450$ and $\mathcal{P}[\mu(p_{2,3}) = 0] = 0.93550$. Thus the forced synchronization of T_{sync} imposes an average blocking overhead in the amount of about 6.5 % of

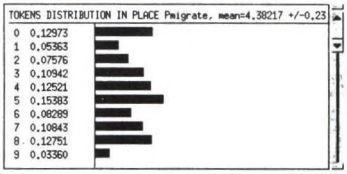


Figure 5: Token Distribution in Place $P_{migrate}$

the overall execution time on agent 3, and a blocking overhead of about 54.1 % on agent 2. A different configuration of processes elements or a different process assignment to agents might have avoided this inefficiency.

2.5 Qualitative Analysis of Workflow Models

The key issue to workflow modeling and analysis in a business system is not only to understand the time and resource requirement to perform each process element (task, activity), or the *waiting time* between process elements, and the *idle time* for agents, but also in time independent behavior. So when interested in *qualitative* properties of the modeled system, the broad body of Petri net structural analysis techniques is available, which allows to answer questions of the following type with the corresponding technique [Mura 89]:

- Will an agent, when executing process elements ever return in its initial state? (*Reachability, Home State, Reversability*)
- What is the amount of process elements to be executed between two given process elements? (*Synchronic Distance*)
- Will a process element, by the structure of the process, ever be executed? (*Reachability, Liveness*)
- Are there closed cycles of work flow that could be rearranged to reduce overall flow throughput based? (*P-invariants, Synchronic Distance*)
- Does a subset of process elements create more work than it can execute? (*Boundedness, Deadlocks, Traps*)
- Once a process element has started execution, can it be interrupted (preempted) by the execution of another process element? (*Persistency*)
- Does the repeated execution of a certain process element prevent another process element from becoming executed? (*Fairness*)
- etc.

3 Conclusions and Future Work

We have defined an abstract framework for modeling business workflow systems in terms of process elements, (business) processes, agents and roles. The exploitation of the expressive modeling power of Petri nets for workflow modeling, and the profound body of Generalized Stochastic Petri net evaluation and analysis techniques for performance (quantitative) and structural (qualitative) system properties was demonstrated. A simple example illustrated our approach.

So far we have focused on the formalization of *flows* of abstract work. Besides this, there remain other concepts that need a thorough representation in a workflow model: *hierarchy* is necessary for representational reasons but is not available in the 'flat' GSPN model, *modularity* is needed to ease a fast creation, manipulation and reuse of models, and explicit *data types* are required for data dependent flow control and data dependent timing. Basically any kind of high-level Petri net [Jens 92] with a coloring mechanism, timing and hierarchies can serve for those purposes. Graphical tools with sophisticated user interfaces for such net classes are available today [Meta 91]. In addition to the extensions related to the representation of work flow systems, analysis methods have to be developed serving purposes related to a business organization's goals. Specifically *scenario management* and *performance optimization* tools are demanded for a fast and automatic (or at least semi-automatic) analysis of workflow system designs.

References

[Ajmo 87] M. Ajmone Marsan, G. Balbo, G. Chiola, and G. Conte. "Generalized Stochastic Petri Nets Revisited: Random Switches and Priorities". In: *Proc. of the International Workshop on Petri Nets and Performance Models, August 24 - 26, 1987, Madison, Wisconsin.*, pp. 44 – 53, IEEE Computer Society Press, 1987.

[Chio 93] G. Chiola, M. Ajmone Marsan, G. Balbo, and G. Conte. "Generalized Stochastic Petri Nets: A Definition at the Net Level and its Implications". *IEEE Transactions on Software Engineering*, Vol. 19, No. 2, Feb. 1993.

[Denn 94] P. Denning. "The Fifteenth Level". In: *Proc. of the 1994 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, Nashville, 1994*, pp. 1 – 4, ACM, 1994.

[Elli 93] C. A. Ellis and G. J. Nutt. "Modeling and Enactment of Workflow Systems". In: M. Ajmone Marsan, Ed., *Proc. of the 14th Int. Conf. on Application and Theory of Petri Nets 1993, Chicago, June 1993*, pp. 1 – 16, Springer Verlag, Berlin, 1993.

[Flei 89] H. Fleischhack and A. Weber. "Rule Based Programming, Predicate Transition Nets and the Modeling of Office Procedures and Flexible Manufacturing Systems". Tech. Rep. Bericht TI 3, Universit"at Oldenbourg, 1989.

[Holt 83] A. Holt, H. Ramsey, and J. Grimes. "System Technology as a Basis for a Programming Environment". *ITT Electr. Commun. 57*, No. 4, 1983.

[Holt 85] A. Holt. "Coordination Technology and Petri Nets". In: G. Rozenberg, Ed., *Advances in Petri Nets 1985*, pp. 278 – 296, Springer Verlag, Berlin, 1985.

[Jens 92] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use (Volume 1). EATCS Monographs on Theoretical Computer Science*, Springer-Verlag, Berlin, 1992.

[Meta 91] Meta Software. "Design/CPN. A Tool Package Supporting the Use of Colored Petri Nets". Tech. Rep., Meta Software Corporation, Cambridge, MA, USA, 1991.

[Mura 89] T. Murata. "Petri Nets: Properties, Analysis and Applications". *Proceedings of the IEEE*, Vol. 77, No. 4, pp. 541–580, Apr. 1989.

[Proc 94] "Proceedings of the Workshop on Computer-Supported Cooperative Work, Petri Nets and related formalisms, held within the 15th Int. Conf. on Application and Theory of Petri Nets, Zaragoza, June 21^{st}, 1994". 1994.

A reference model for Workflow systems: Functions and Requirements

G.A. Pitschek

Paper not received in time.

Trigger Modelling for Workflow Analysis

Stef Joosten
Design Methodology Group
Centre for Telematics and Information Technology
University of Twente

July 5, 1994

1 Introduction

An empirical study into experiences with workflow management [JAD+94] showed that triggers are an important notion in describing a workflow system, relating the activities to one another that are performed by people and/or machines. If activities, roles and triggers are so important, the analysis and design of workflow systems should take these notions as a starting point. This makes workflow modelling different from information systems modelling, which conventionally starts with the modelling of datastructures (e.g. Entity-Relationship modelling) or processes (e.g. Dataflow modelling or Process modelling).

Based on this empirical observations, we describe and formalise a modelling technique called *trigger modelling*. Based on the notions that are being used in practical situations, this technique aims at supporting workflow analists in the analysis and design of workflow systems.

2 Workflow Definitions

The workflow-concepts are defined in terms of the following basic notions:

event something that happens; something that occurs (example: the occurrence of a letter being posted)

actor one that acts (example: the person posting a letter)

object something that is or is capable of being seen, touched, or otherwise sensed (example: the letter)

These notions are used as the conceptual building blocks of the definitions that follow. The internal structure of events, actors and objects is not considered relevant. The meanings of these notions are taken from [Mer63], because we want to use these words in their 'conventional' meaning. All other notions are ultimately defined in terms of events, actors and objects.

Definition 1 (activity) *An* activity *is a set of events that occur under the responsibility of one actor.*

This definition emphasises the responsibility for an activity, rather than performing the activity. The definition allows an activity to be performed by several people, as long as one actor is responsible. For example, sending out a letter may involve secretaries, delivery services, etc., but it is considered an activity when those acts are performed under responsibility of the sender.

The verb *perform* is used with respect to an activity. An activity is *performed* if the events in the activity occur. There must be one or more actors to make these events happen. Wherever the distinction between performance and responsibility is relevant, but not clear from the context, the phrases *responsible actor* and *performing actor* are used. This distinction appears to be important for the analysis of a workflow system.

Actors can either be human or automated. Both types of actors are treated on the same level of abstraction in order to model the interaction properly. This hybrid nature (i.e. both human and automated aspects) is a characteristic of workflow systems.

An event is *carried* by an object. For example, a damage claim form can carry the event of submitting a damage claim. Objects can have any physical form, for example a telephone call, a letter, a note, a form, an electronic message. Objects may have information content as well.

An *event* occurs as a result of performing an activity. In turn, activities are performed as a result of the occurrence of events. For example, the submission of a damage claim (event) can occur as a result of assessing a particular damage (activity). In turn, submission of that claim causes the insurance company to start processing the claim (activity). This behaviour is called *triggering*.

Definition 2 (trigger) *An event e* triggers *an activity a if the occurrence of e causes a to be performed.*

In everyday language, the verb is used in three grammatically different ways. In the sentence: *a triggers b*, *a* can be an event, an activity or an actor, but *b* is always an activity. Each of these three ways can be interpreted as a grammatical variation of definition 2. Consequently, it is not necessary to provide definitions for each of the alternatives. The alternatives are illustrated by means of the following examples.

triggered by:	Sample sentence
event	Arrival of the damage claim form triggers the claim registration procedure.
activity	Submitting the damage claim form triggers the claim registration procedure.
actor	The customer triggers the claim registration procedure.

The word *trigger* is also used as a noun. In that case, a *trigger* is the object that carries a triggering event. For example, the damage claim form can be called a trigger.

Definition 3 (process) *A* process *is a set of activities that share a common purpose.*

Processes are defined to give a name to a set of activities that are related in a way that makes sense in a specific situation. Processes can be divided in subprocesses, which corresponds to the subset relation between sets. The distinction between a process and an activity is motivated by the difference in responsibility. An activity has one actor that bears responsibility for performing. A process may involve different responsible actors.

To illustrate the definitions, we give an ER-diagram that represents the relation between the notions *event*, *object*, *activity*, *process* and *actor*, in the notation according to [EN89] (figure 1). This figure can be skipped safely by readers who are unfamiliar with Entity-Relationship modelling, because the text contains the same information.

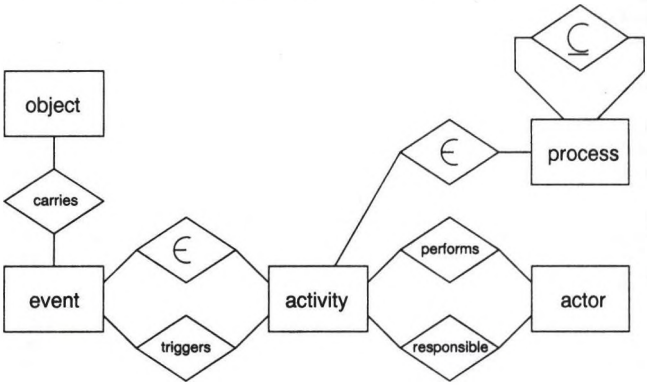


Figure 1: ER Model of Concepts

Definition 4 (workflow) *A* workflow *is a system whose elements are activities, related to one another by a trigger relation, and triggered by external events.*

Examples are: the set of activities in an insurance company caused by a damage claim; the work caused in a hospital when a new patient is admitted and the activities triggered in a bank by a loan request. A *workflow system* contains a workflow, all actors and all structures and the means involved in that workflow. The notion of workflow system does not refer to the technology alone, but includes all related elements.

3 Representation

A workflow can be represented in a trigger model. Each activity is represented by a rectangle, containing the name of the activity. An arrow pointing towards an activity means that the activity can be triggered by the events that occur as a result of the activity at the other end. The trigger model is divided into columns, each of which contains the activities associated with a particular role. Figure 2 gives an example of a trigger model. The specific nature of a trigger model becomes clear when figure 2 is elaborated in more detail. Figure 3 is a detailed version of the same workflow. However, the mail sorting process has been made visisble. This process of refinement can continue until the modeller has charted all relevant aspects of the workflow under consideration. An important observation in figure 3

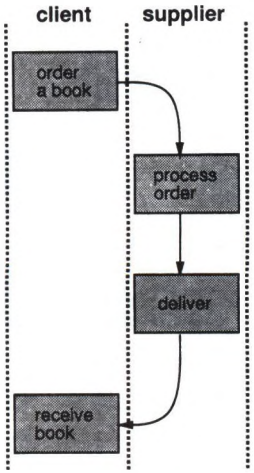


Figure 2: Trigger Model

is that the the mail collector is not triggered by the client. Fact is that the letterbox is emptied at regular times, so the mail collector is triggered by a clock. This shows that a trigger model does not necessarily follow the flow of matter or information. Therefore it is different from dataflow models.

Workflow analysis is set off by determining the workflow to be modelled. The workflow analyst makes a choice as to what is considered part of the workflow. The roles have to be determined, and interviews are conducted to find out which activities are performed in which roles. By pursuing the trigger arrows, all activities that are considered part of the workflow can be traced with limited effort.

When an analyst models a workflow, discussions with people involved must focus on activities, roles and triggers. At a later stage, the analyst must fill in the activities in the trigger model. For that purpose, rectangles can be substituted by:

a circle , if the activity is an action. This means that the activity is considered to be atomic, i.e. not having any internal structure. A single decision is an example of an action.

a triangle , if the activity synchronises the triggers. Usually, it distributes on trigger over several others, or it synchronises several triggers into one.

another network , if the activity is too complicated, it can be analysed in a separate model. The rectangle is then used as a placeholder for that separate model.

This model can be used to generate workflow control. Due to the presence of circles and triangles, it is likely to be less suitable for communication with customers.

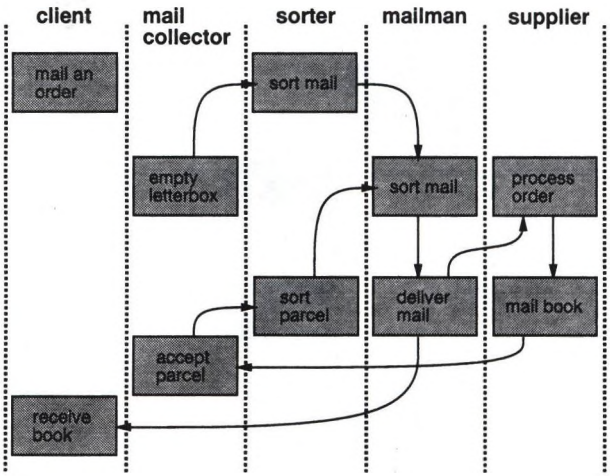


Figure 3: Trigger

4 Meaning

The purpose of modelling a workflow system is first of all to describe the dynamic behaviour of the system in terms of triggers. Triggers are conventionally modelled by means of Petri-nets [Pet77]. Therefore, we will give meaning to trigger models by mapping a trigger model to a Petri-net. Each arrow from the trigger model is mapped to the corresponding Petri-net as described in figure 4. The places and transitions that are drawn in the shaded area are new elements in the Petri-net. The elements outside the shaded area are existing elements.

To explain Petri-nets is beyond the scope of this paper; we assume readers to be familiar with the way tokens are passed in Petri-nets. Those readers unfamiliar with Petri-nets should know that this graphical technique is well established for studying the dynamics of systems. The semantics of Petri-nets has been defined mathematically. They are used to study deadlock behaviour, to simulate and animate the dynamics of systems. In this paper, Petri-nets form the semantics of trigger models by means of the mapping shown in figure 4.

5 Example

We give an example to show how the modelling technique works and the relation to the dynamic behaviour. The model is obtained by an analyst, interviewing people who are responsible for activities that belong to a particular workflow. The analyst makes a list of activities for each role, and determines how every activity is triggered.

Figure 5 describes a complaint procedure in some organisation. A complaint is filed immediately by a representative, who will execute all contacts with the customer personally.

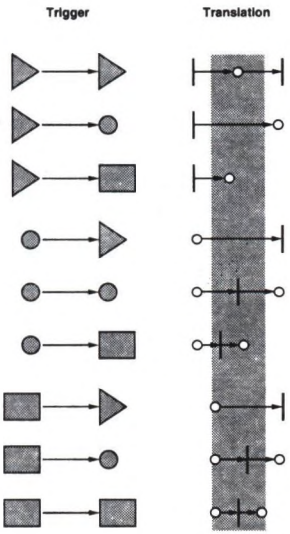


Figure 4: Translation of triggers to Petri-nets

Filing the complaint triggers an inspector to look into the complaint, and the responsible manager to acknowledge the complaint. This manager can reject the complaint at all times, which has to be communicated to the customer by the representative. Once the inspector has finished analysing the complaint, a summary is sent to the responsible manager. At this point, the representative can negotiate a solution with the customer, based on the analysis of the expert. The events of reaching a solution (activity: negotiate solution) and the analysis being produced will (together) trigger the manager to start work on the solution. Once that work is finished, the representative verifies whether the customer is satisfied. A terminated complaint procedure is logged by a librarian for future use.

This initial model contains rectangles only. It can be wise to use this simple model initially, not to confuse a client with four different symbols in the model. However, during the interviews the analyis has acquired knowledge about timing behaviour that is not represented in figure 5. Notice that the verbal description contains such timing information, such as 'terminated', 'once the inspector has finished' and 'at all times'. The model can be refined by introducing synchronisation and distribution objects (triangles) and actions (circles) where appropriate. Activities that remain abstract can safely keep their rectangular shape, such as 'file complaint' and 'analysis' in figure 6. This is a sound way of coping with incomplete information. In a later stage, these rectangles can be filled by 'pasting' another network that contains the details. This allows hierarchical decomposition. The Petri-net in figure 7 can be constructed by a computer, using the mapping shown in figure 4.

In this example, three steps (figures 5, 6 and 7) were used. The first is a limited version of trigger models, in which only rectangles are used. These are useful for communication with customers. The second contains design decisions with respect to timing. It can be

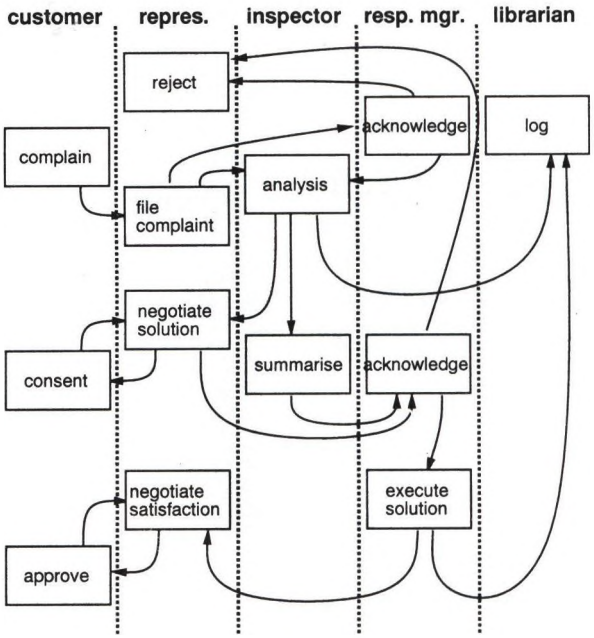


Figure 5: Trigger model of complaint procedure

transformed directly into a Petri-net, in which the remaining rectangles will show up as stubs.

6 Other meanings

Models have many different meanings. To illustrate this point, we give an alternative meaning to trigger models.

A communication model shows the different locations and the communication links between those locations. An initial choice might be to assign a different location to each role.

The communication patterns can be derived by assuming that every event that triggers an activity corresponds to the communication of an object between the respective agents. Each arrow in the trigger model represents a stream of events. The modeller has to find out which objects are communicated to trigger an activity: e.g. a note, a fax, a spoken command, a letter or a form, a phone call. The task of the modeller is to make a list of triggers in the trigger model, and to find out for each trigger:

1. the carrier (telephone, fax, e-mail, p-mail, speech, etc.)

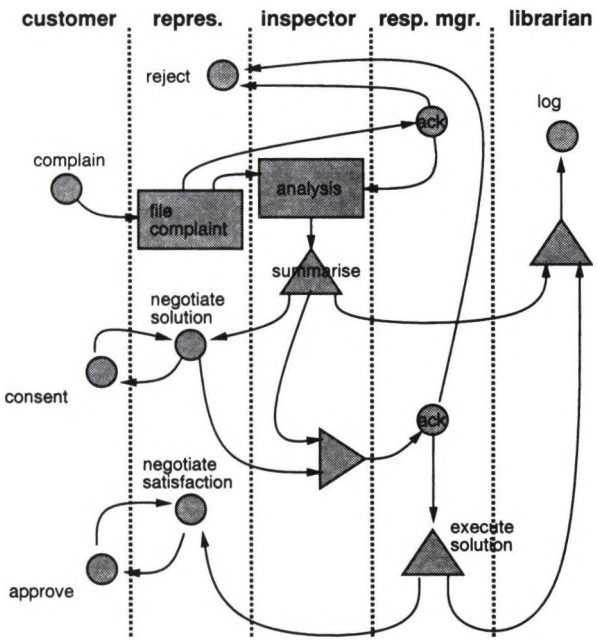


Figure 6: Trigger model of complaint procedure

2. the object type (including datastructure)
3. requirements on the information content of each event

This list is necessary to implement the workflow, whether it is automated or not.

Figure 8 shows a Communication model. It has been derived from the trigger model by naming each arrow in the trigger model, and by taking together all activities that belong to the same role. The communication model is the starting point for the technical design of the workflow system.

7 Validation

Validation of trigger modelling has been done in two different ways. The first way was to construct a prototype of the transformation between trigger models and Petri-nets, to show that the transformation is feasible. The second way of validating the technique was to conduct an experiment in a controlled environment of the use of the technique, in order to show that it is usable. Both ways are reported upon in this section.

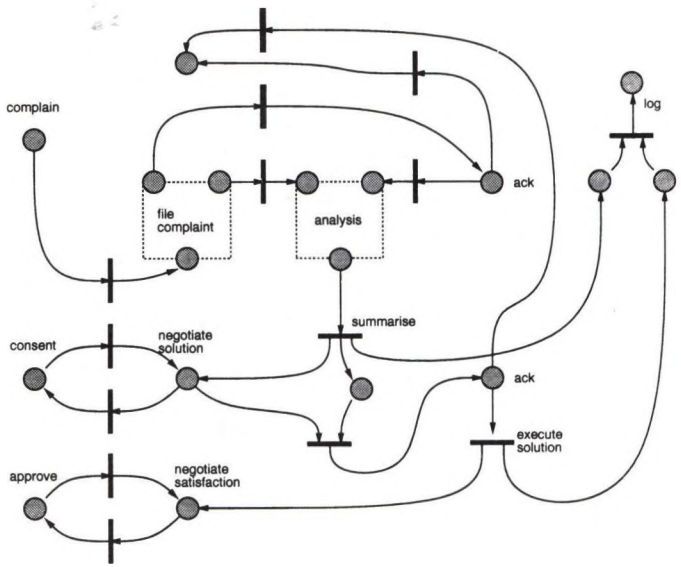


Figure 7: Petri-net model of complaint procedure

7.1 Prototype

A functional programming language [BW88] was used to make a prototype of trigger models in terms of graphs. This was done along the lines of [JB94], which describes how to describe a modelling technique in terms of syntax, semantics and pragmatics. Presentation of the prototype is considered beyond the scope of this paper.

A function is defined that maps a trigger model to a Petri-net. This function represents the semantics of the trigger model. Section 4 is a (pictorial) description of that function. This prototype was used to experiment with the transformation, in the sense that the computer can derive a Petri-net from arbitrary trigger models.

7.2 Experiment

An experiment has been conducted to determine whether trigger modelling is easy to use. We were also interested to observe which "common mistakes" would be made by initial users of the technique.

We have picked first year students in Business Information Technology (freshmen, 18-19 years of age) as initial users. These students were asked to make a trigger model of a given catalogue production process, while working in small groups of 3 to 5. Both the trigger modelling technique and the catalogue production process were new to all of the students.

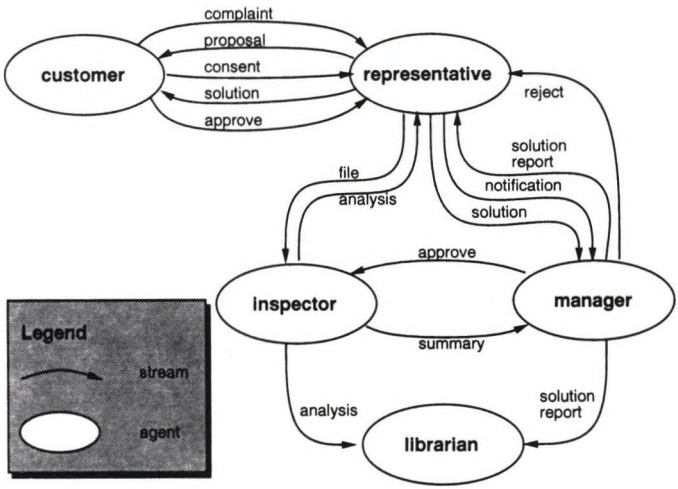


Figure 8: Communication model of complaint procedure

There was 1.5 hours to complete the job. We were interested in answers to the following questions:

1. How many groups managed to produce any result at all?
2. For those groups who did, how many roles, activities and triggers were chosen?
3. Which modelling mistakes are observed?

Results were that 13 out of 14 groups produced a trigger model. Table 1 shows the number of roles, activities and triggers for each group. The most common mistakes in the modelling technique were:

1. department names are being used instead of roles;
2. arrows are used as dataflow rather than as trigger;
3. activities are spread over several columns;
4. arrows are freely forked, or double arrows are used.

These observations provide useful input for a minicourse in trigger modelling.

The overall results of this experiment are quite satisfactory. If freshmen can do this in an hour and a half, then system analysts in practice will most likely have little trouble in starting to use the technique.

	role	activity	trigger
1	6	17	22
2	9	17	17
3	6	16	18
4	9	27	34
5	9	14	15
6	10	18	24
7	5	13	11
8	7	14	12
9	5	7	13
10	8	18	22
11	7	14	14
12	7	20	26
13	5	10	12
average	7.2	15.8	18.5

Table 1: Results of trigger modelling experiment

8 Conclusions

We have presented the trigger modelling technique. It is based on the experience of workflow analysts, using the concepts of workflow (activities, roles and triggers). Because trigger models have semantics in terms of Petri-nets, they can be readily used for workflow automation. These semantics are presented in the article, while a prototype implementation exists. There is empirical evidence that the trigger modelling is easy to learn. From these results, we conclude that trigger modelling is a useful technique for the analysis of workflow systems. It is a formal modelling technique in that it has formal semantics.

Further research needs to be done into other interpretations of trigger models. In this article we suggest to derive a communications model from the trigger model, but clearly this needs further elaboration.

Other work needs to be done in order to obtain a CASE tool for the design of workflow systems. This tool must at least support trigger models, their transformation into Petri-nets and their further transformation into workflow control software.

Other experiments are needed to validate the choice of concepts in trigger modelling.

References

[BW88] Richard Bird and Philip Wadler. *Introduction to Functional Programming*. International Series in Computer Science. Prentice Hall, New York, 1988.

[EN89] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley World Student Series. Benjamin/Cummings, Redwood City, CA 94065, 1989.

[JAD+94] Stef Joosten, Gaston Aussems, Matthijs Duitshof, Richard Huffmeijer, and Erik Mulder. Wa-12: an empirical study about the practice of workflow management. Technical report, University of Twente, dept. of Comp. Sc., P.O. Box 217, 7500 AE ENSCHEDE, the Netherlands, July 1994. Research monograph.

[JB94] S. Joosten and S. Brinkkemper. Meta-modelling in a functional language. submitted to the 13th International Conference on the Entity-Relationship approach, Manchester, UK, 1994.

[Mer63] Merriam-Webster, Inc. *Webster's 7th Collegiate Dictionary*, 1963.

[Pet77] J. Peterson. Petri nets. *ACM Computing Surveys*, 9(3), September 1977.

Database Support for Workflow

A Workflow System Based on Active Databases

Johann Eder, Herbert Groiss, Harald Nekvasil

Institut für Informatik

Universität Klagenfurt

Universitätsstr. 65

A-9022 Klagenfurt, AUSTRIA

e-mail: {eder,herb,nekvasil}@ifi.uni-klu.ac.at

Abstract

We present the architecture of a workflow system based on active databases. Business processes can be defined in an easy-to-use graphical workflow description language. Process descriptions written in this language are compiled to rules and executed in a system based on an active database. Thus the workflow system can take full advantage of the capabilities of a database system such as reliability, recovery, concurrency control, transactions, and authorization. We describe the general architecture of such a system and the implementation of a first prototype and discuss the advantages of this approach for building as well as applying workflow machines.

Keywords: *workflow management, active database, trigger, dynamic modeling*

1 Introduction

The aim of workflow systems is to support business processes. From an abstract perspective a business process consists of a sequence of tasks. The process specifies

- which tasks have to be performed
- in which sequence (probably depending on decisions which are part of the process),
- by whom
- under which constraints (time, quality).

Such business processes can be found in businesses, industries, and administration. The tasks can be performed automatically, by humans or by interaction of humans with information technology (IT). Traditionally, business processes are mainly managed using paper, forms, and other communication media. Traditional IT supports business processes only in a rather limited way. It is restricted to standard processes and is conceived as very inflexible. But current economic changes flashlighted by buzzwords such as lean management, just in time production, and computer integrated manufacturing, require enterprises as well as administrations to be highly reactive to external and internal events, to participate in tightly integrated processes and to be able to flexibly adjust these processes.
Advantages of applying workflow systems to business processes comprise the following:

- *Specification:* The application of workflow systems leads to a better specification of business processes, of regular (standard) processes and even more of special ad-hoc processes. Even if this is not a technical matter, experience shows that the organizational analysis and design needed to employ workflow systems increases the quality of business processes.
- *Documentation:* The application of workflow systems leads directly to an exact documentation of business processes. It should be noted that process documentation is an inherent necessary feature for quality management. This integrated documentation also yields better traceability of processes, built-in status accounting, and improved responsiveness.
- *Turn-around:* A primary goal for employing workflow systems is to reduce turn-around times and therefore to improve reactiveness.
- *Flexibility:* In comparison to traditional software solutions, workflow systems are much easier to adapt. They allow a very dynamic and flexible redesign of business processes to adapt to business needs. Furthermore, standard cases / processes as well as non-standard ones can be dealt within the range of one system.

- *Integration:* Workflow systems can act as 'glue' between different ITs allowing also the integration of legacy systems in new business processes.

The aim of our work is the automation of such business processes, also called workflows. This requires in a first step the storage of the documents handled by the agents in a database, and electronic forwarding of the documents from one agent to the next.

This is the conventional way using tools like text-processors, spreadsheets, databases and electronic mail. When only these tools are used the knowledge about and the responsibility for the process remains with the agents, who process the documents and decide then to which successor these documents have to be delivered.

An automation of this task requires:

a) a model (schema) of the process,

b) an automatic delivery mechanism for documents according to the process information,

c) a mechanism for automatic invocation of programs.

We call the latter two items a workflow machine. This workflow machine is data and event driven and uses the process information to decide about the delivery of a document finished by an agent and the invocation of automatic agents.

In the last years systems for automating business processes have been studied in the area of office automation or office information systems [7], [10], [12] [4], [11],[1],[9],[13]. Only recently the term *workflow* was coined for such types of systems and interest in such systems exploded. More than 40 workflow management systems with quite different capabilities, are on the market today and most of them went to the marked in the past two years. There seem to be commercial as well as technological reasons for this rush. The commercial reasons for stimulating the demand for workflow management systems have been outlined above. The technological reasons are seen in the high availability of fast communication infrastructures, client server solutions, powerful client workstation, and the need to integrate legacy information systems.

The main contribution of our approach is the usage of active databases to implement the workflow machine. Active databases are well suited for applications which are inherently data driven or event driven (for an introduction into the field of active databases refer to [3], [2]). These systems extend

conventional (passive) databases with production rules. They allow the specification of actions which are executed automatically whenever certain events occur and certain conditions are satisfied. The specification of Event, Condition and Actions is done declaratively with so-called ECA-rules.
Each database access from a user or an application program (insert, update, delete, select) is seen as an event, which can trigger the application of a rule. If a rule is triggered, the conditions of the rule are evaluated. If they are satisfied, the actions of the rule are applied. Conditions are descriptions of database states, actions are operations, which can modify the database or start external procedures.
In this paper we use the syntax of SQL3 [8], where the basic structure of a rule is:

```
create trigger name on table
after event
when condition
then action
```

With create trigger a rule is defined, which reacts on changes of the table *table*. The *event*, which triggers the rule is specified next and the *conditions* - a SQL query - follow the keyword when. Actions are database actions formulated in SQL.
Because the description of the processes in terms of triggers is on a very low level, such programs are hard to read and to debug. Therefore, we describe the workflows in an easy-to-use graphical high level language designed specifically for this purpose and translate the specifications of workflows into triggers of an active database system. This has also the advantage of independence of the descriptions from a specific product or trigger language.

What are the advantages of using active databases as base technology for implementing workflow systems?

- All dynamic information like the (dynamic) status of processes, documents, etc. are mapped to the database and maintained within a database system. Thus the capabilities of database systems like safety, authorizations, and most important recovery are immediately available. In particular, in the case of system crashes, the recovery mechanism of the database also recovers the dynamic state of all processes.

- Workflow processes should provide a high degree of concurrent execution to decrease turn-around times. The transaction mechanism permits to increase concurrency in a safe way. The

concurrency control system of the database can directly be used and it is not necessary to reimplement an additional one for the workflow machine.

- If *active* databases are employed, the database is not only the blackboard for the workflow scheduler and the workflow processes, but it *is* rather the workflow machine itself. In particular, the scheduler and the agents no longer have to poll the database whether the preconditions of some process are fulfilled, creating an unnecessary high workload or reducing responsiveness. Previous work has shown that a central scheduler has advantages over sending or polling strategies [5].

In the next section we introduce the language designed for specifying the processes, in section 3 the translation process of a workflow description to the rules of the active database and in section 4 the system architecture of our prototype implementation is described.

2 The workflow description language - WDL

The main design criteria of WDL were: easy to use for an end user to design simple workflows, flexibility in describing a wide variety of business processes, direct compilation to an executable workflow. Note that WDL describes only the communication between tasks, i.e. the data flow and control structure between tasks, but does not specify the internal structure of a task or which modifications a task performs.

The basic modelling concepts are:

- users and roles,
- forms, and
- processes, consisting of tasks and flows.

At first a brief description of these concepts is given:

user: describes an agent, who can perform tasks (some data manipulation)

role: defines a set of users with common properties (e.g. clerk) or as members of an organizational unit

form: data container holding the information that flows between the different agents. Forms are used for representing and manipulation of information

process: describes the structure of a complex, distributed job; i.e. which tasks and flows it is composed of

task: defines an elementary activity (i.e., done by one agent)

flow: defines the transmission of information (a form) between two agents

workflow: aggregation of processes

Many concepts of the modelling language can be expressed in a comprehensive WDL process diagram.

2.1 Description of the graphical notation

A process diagram specifies the structure of one specific process involving the tasks, the data flows between them and users respectively the roles performing these tasks.

Fig. 1 shows such a diagram for the process of an application for a business trip. This process requires interaction of different persons and departments. An applicant who plans to make a trip needs a permission from the head of the department and the dean. After the trip he gets the money from the personal department.

The main elements of the graphic representation are tasks and flows:

A task is an elementary activity done by one person or one computer program. What exactly happens when a task is executed is not in the focus of the description, typically a task changes the contents of some forms.

A task is represented by an rectangle. Inside the rectangle the name of the task and the agent (the user or role performing the task) is written (the name of the agent is enclosed between brackets). If the task is processed automatically the pseudo-user SYSTEM is specified. This allows the definition of arbitrary programs for manipulation of the data and therefore the integration of other application programs into the workflow. Sometimes it is useful to define the user dynamically, i.e. send it to the task as content of a field in a form. In this case we write DYNAMIC into the user field. It is also

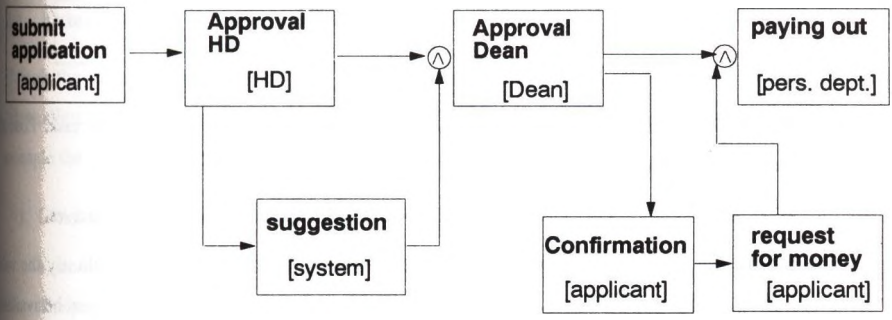
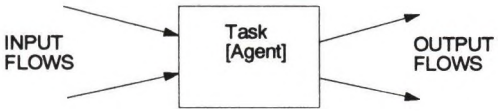
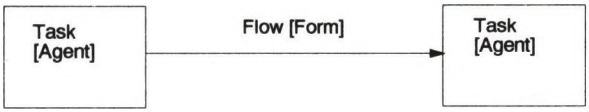


Figure 1 **WDL process diagram for a business trip**

possible to specify a task timeout. That means after the specified duration a timeout is signalled.

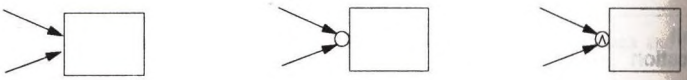


A flow connects two tasks and is denoted by an arrow. Considering one task you can group the flows into input flows and output flows. Input flows are the incoming flows of a task delivering the necessary forms to the task. Output flows specify the outgoing forms after completion of the task. Similar to tasks you can define the delay for a flow. The form is then transmitted after the specified delay.



Usually a task will have more than one input and output flow. Therefore we introduce the following concepts for specifying constraints on the input and output side of a task:

1. Input side:



We can define some preconditions which must be fulfilled before activating the task. Possible preconditions are:

a) no precondition: That means the task is activated if one of the input forms arrives.

b) a boolean expression together with an optional predicate: We define explicitly the valid combinations of the input forms (e.g., f1 AND f2, that means forms f1 and f2 must be available before activating the task) and a predicate for synchronizing the forms (e.g., f1 AND f2 [f1.name = f2.name], that means forms f1 and f2 must be available and reference the same name).

c) a synchronization point: A form can be sent to more than one successor task for parallel manipulation. At the end of such a parallel processing the synchronization point is only passed if each of these parallel tasks are finished.

2. Output side:

After completing the task each output flow transports its forms to the specified successor task (if an optional condition is valid). Consider the following special concepts:

a) disjunction: The actual form is either sent to task A or task B depending on the condition specified by the flows. Using this concept we can model conditional flows.

b) a form is sent to task A and task B for parallel manipulation: This is the counterpart of the synchronization point introduced in the above paragraph. For modelling parallel manipulation a form 'splitting' at the begin and a synchronization point at the end has to be defined.

Note that the concept of dynamic users is very powerful, for example e-mail can be modelled if the user can write the field of a form where the agent of the next task is read from. The process diagram

of e-mail is a task with a flow starting and ending at this task.

2.2 Extensions to the graphical notation

Though the process diagrams are very illustrative in showing what is going on, some additional information have to be expressed for compilation of the description to an executable workflow. For example the types of the fields in the forms have to be specified.

1. General information for the workflow:

   a) associated users and roles:

   The process diagram just shows the participating users and roles. In addition you have to define all the users participating in a workflow and the association of roles to users.

   b) structure of the involved forms:

   Again the process diagram just shows the flow of the forms without defining the structure of the form. A form consists of fields each having a type. We allow atomic types (string, number, boolean and character), the type table (a collection of tuples of atomic types), as well as references to external files. Moreover, the appearance of a form in the user interface has to be defined.

2. Additional information for tasks:

   a) postconditions:

   You can explicitly define some postconditions to enforce a valid state. The task can only be successful completed if the postconditions are fulfilled.

   b) procedure:

   A before-procedure and an after-procedure can be specified for execution before activating or after completing a task respectively.

   The procedures and the post-condition are optional.

   c) selection criterion:

   Specifying a role as task performer requires the definition of a selection criterion. This criterion is used for the assignment of the task to a concrete user during the execution. Possible criteria are: choose user with minimal workload, choose user randomly, etc.

d) access structure: It is possible to specify which fields of a form a task can read or change.

We have not defined an exact syntax, how the preconditions, postconditions and the procedures are specified. This is left unspecified, because it depends on the concrete implementation, i.e. in our case on the data manipulation language of the database management system.
In section 4 we describe a graphical design tool facilitating the specification of workflows.

2.3 Execution model

A WDL process description defines when and under which conditions a form is transported from one task to a successor task. What is done within a task is not specified. After such a form manipulation in a task A is finished, the workflow system executes the following steps:

1. the optional after-procedure of task A is processed.
2. The postconditions of task A are evaluated. If the postconditions are fulfilled, the forms manipulated by this task are marked as processed and the task is finished, in the other case the task gets an error message.
3. Every output flow of task A is checked and if the flow condition is met, the form is sent to the successor task and gets the status pending.
4. The preconditions of every successor task are evaluated. If all preconditions of a task are met the task is ready.
5. If there is a task ready, the next step is the assignment of a user to the task if the specified agent of the task is a role. The selection criterion is evaluated and a concrete user is assigned to the task.
6. Next the (optional) before-procedure is started.
7. The user interface of the user assigned to the successor task gets now a signal that the task can be started.

3 Translating a workflow description into rules

In this section we describe the principles of the implementation of a workflow system based on WDL with active databases. The whole description of a process in WDL is stored in the rules and tables of the database.

The structure and content of the forms as well as the information about users and roles are maintained in database tables. Additional fields are needed for administrative and dynamic information: the holder of the form, the task which currently has access to it, and the status (pending, active, etc.).

The rules are automatically generated from the declarative descriptions of the tasks and flows by the WDL compiler. Therefore, the active database management system is the workflow server and has the functionality described in the process specification.

Mainly, the rules react on changes of the status fields of the forms. For example, when a task is finished it changes the status of the processed forms from active to processed. This event fires a rule which runs the post-procedure and changes the status of the forms again.

In this way a chain of rule applications is initiated, whenever a task is completed. In analogy to the steps described above, the description of a workflow is translated into several groups of rules.

For each flow one rule is generated (called flow-rule), triggering when a task is completed, i.e. after the satisfaction of the postcondition. This corresponds to the third step of the above execution model.

The following rule specifies a flow of a form of type *form_i* from *task_A* to *task_B*, where the form is sent if the condition *flow-condition* is met.

```
create trigger flow_n_step3 on form_i
after update status
when new.status='finished'
and form.type=form_j and form.task = task_A and flow-condition
then update new set task = task_B;
```

The rule fires on changes of the status field in the table *form_i*. The condition is met if the new value of the status is 'finished'. In this case the task field of the form is set to the successor task.

Like in the above example, the rules are built from fixed templates into which the information from the process specification is filled in, e.g. from-task, to-task, form, and flow-condition.

The following types of rules are generated for each task:

post-task rule: This rule triggers when the task is finished and executes the after-procedure (step 1 of the execution model).

postcondition rule: The rule tests the postconditions of the task (step 2).

precondition rule: This rule tests the precondition of a tasks: This is necessary if the task has more than one input flow. On each arrival of a form at the task this rule is triggered and checks whether all forms necessary for the execution of the task are available (step 4).

dispatch rule: The rule exists, whenever the performer of the task is specified as a role together with a selection criterion (step 5). The non-empty user field of the task after the execution triggers the next rule:

pre-task rule: This rule applies the before-procedure. After completion this rule sends a signal to the client program (either the standard client or an application program performing the task).

We want to emphasize that the whole workflow manager simply consists of all the rules resulting from the compilation of WDL workflow specifications. All other necessary features are already provided by the database management system.

4 System Architecture

To evaluate our approach we implemented a prototype workflow manager using the ORACLE database management system version 7, which provides a simple rule system. As hardware platform we use a cluster of SUN workstations with SUNOS 4.1.2 and OpenWindows.
The system consists of four components:

- the server,
- the user interface client,
- the workflow design tool and the WDL compiler,
- the monitoring client.

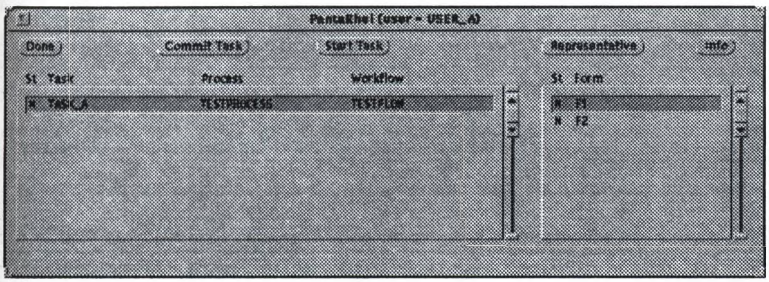


Figure 2 **user client interface**

The server is the active database management system with the rules, forms, tasks and users specified in the database. No additional code is necessary, because the communication of the other components is done exclusively via database accesses.

4.1 user client

This component is the interface of the normal user to the workflow system, Fig. 2 shows the appearance on the screen. The typical process of handling is similar to the processing of mail:

The user interface notifies the reception of a task. When the user selects a task, he gets a task description with some general information (sender, corresponding process, description of the task, etc.) and a list of forms. He can now view and edit the received forms. In this step the user can only see the forms and fields which are marked as visible or editable in the task description and can only edit the fields declared as editable. During filling in the forms the user can rollback the modifications of each form. The work with a task is concluded with a commit, which results in communication with the server for running the post-procedure, checking the post-condition, and removing the task from the users active-task list.

The explicit archivation of the forms is not necessary, as the history mechanism of the server keeps the whole history of each form. Every user can view all forms he handled in the past.

Moreover the user can send copies of the forms to other users like ordinary mail. This allows informal communication in addition to predefined workflows.

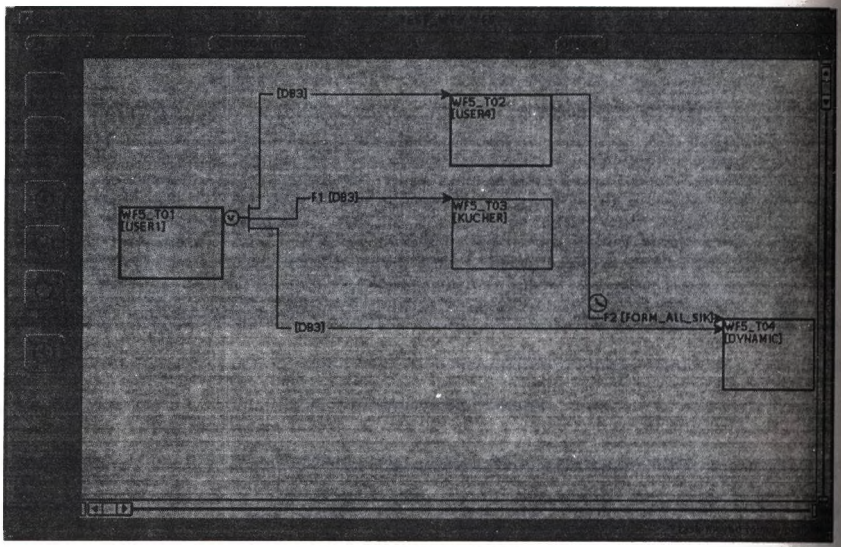


Figure 3 **designer interface**

4.2 workflow designer interface

The purpose of this component is to allow an interactive graphic design of workflow processes and forms. The user interface of the process designer is shown in Fig. 3. The second part of this tool is the compiler which translates the WDL workflow description into the rules of the active database. The maintenance of the processes and workflows is done by the database. The designer can use previously defined forms and tasks. After a newly defined process is stored in the database, it is possible for the users responsible for the initial task to initiate the process.

4.3 monitoring client

An important task in workflow systems is monitoring, e.g. inspecting which forms are pending, how long are the active task lists of the different users, etc.

The structure of this component is very similar to the ordinary user interface. The main difference is, that all forms currently in the system are visible. With different views all forms of a type or all forms belonging to a process can be viewed. The contents of the forms can be edited.

In addition, the monitoring client is used for maintenance tasks like installing users.
The implementation of this component was very simple due to the availability of all needed data in the database. The information about the stati of the tasks, the location of forms or the workload of users can be retrieved with simple SQL-queries.

5 Conclusions

In this paper we proposed a new approach for the development of workflow management systems. We presented a workflow description language for the graphically assisted specification of workflow processes. The goals for the design of the language were to make it as easy as possible such that also (skilled) endusers may use it to define workflows in an ad-hoc manner and on the other hand that it scales up to be able to be used for all processes. To make the language simple we use well known metaphors like business forms and support the specification of workflows with a graphic workflow designer tool. Since the language supports the specification of arbitrary conditions and features higher order constructs such as the specification of a receiver as part of a task, it scales up to represent workflows of any complexity.
The most important contribution of our approach is to show how active databases can be used to facilitate the development of workflow management systems and the application of such systems. Modern database management system are capable of storing and manipulating any kind of data, so it is quite natural to use database systems to maintain all data relevant for business processes. The advantages of our approach can be summarized as follows:

efficiency: This approach showed to be very efficient for the development of a workflow management system since it can use directly all the features of a database management system like transaction management, concurrency control, access authorization and recovery. So the necessary code for a workflow management system can be minimized. This approach is also very efficient for the actual processing of workflows, since the trigger concept of active databases is a very efficient way to schedule tasks, transport data between tasks and launch processes.

reliability: All relevant dynamic information about processes is mapped into the database. So the recovery mechanism of the database management system is used for storing the data as well

as the state information of all workflow processes in a reliable way. In the case of system failures not only the data but also all the information about processes are recovered. A further aspect of this approach is that no user or program can circumvent the workflow manager - be it intensionally or by accident. All changes to data relevant for a workflow are monitored by the active database and, therefore, by the workflow manager.

extendability: The workflow description language allows an easy extension of workflows. Furthermore, since all changes to data are monitored by the workflow manager, arbitrary existing application programs can be used within workflows without changing them. They can be automatically launched from the workflow manager and the changes they perform on data can immediately trigger workflow processes. So the tight integration of workflow manager and database system facilitates the development of workflows as integration platform for existing isolated applications.

traceability: Since all changes to relevant data are managed or monitored by the workflow system, all such changes can be automatically documented. All business processes under control of the workflow manager are documented and can be traced - meeting an important requirement of quality assurance procedures without additional effort.

We have successfully applied this approach in the development of a prototype workflow management system. Active databases have proven to be a powerful technology for implementing such a system. The software engineering problems arising in programming with rules have been avoided through the usage of a higher level language for describing business processes. The usage of a standard commercial database brought the benefits of a stable, system available on different platforms, but had the drawbacks of a limited trigger mechanism: In Oracle it is not possible to use triggers for changing the table which initiated the trigger application. Moreover, triggers reacting on temporal events are not supported.

In the future we plan to extend our system in several directions. We will integrate an extended transaction concept, allowing long running activities accompanied with a compensation mechanism for handling exceptions (e.g. cancellations) requiring the description of inverse tasks and inverse activities. We will work on a characterization of well-formed processes and check processes for well-definedness: e.g. two parallel tasks should not alter the same attribute, or: reading a value

requires earlier writing. Finally we will port the prototype system to other database management systems following the object-oriented or the object-relational paradigm.

Acknowledgements: The authors thank Werner Liegl, Jürgen Modre, and Michael Stark for their efforts implementing the workflow designer interface.

References

[1] C.A.Ellis and M. Bernal. Office-Talk-D: An Experimental Office Information System. In *Proc. First ACM-SIGOA Conference on Office Information Systems*, 1982.

[2] S. Chakravarthy, editor. *Data Engineering Bulletin, Special Issue on Active Databases*, volume 15. December 1992.

[3] Umeshar Dayal. Active Database Management Systems. *Proc. of the Third International Conference on Data and Knowledge Engineering, Jerusalem*, 1988.

[4] J. Eder, G. Kappel, A. M. Tjoa, and R.R. Wagner. A Behaviour Design Methodology for Form Flow Systems. Technical report, Universität Klagenfurt, Insitut für Informatik, 1987.

[5] J. Eder, G. Kappel, and H. Werthner. Evaluation of Scheduling Mechanisms of a Dynamic Data Model by Simulation. In *Prof. of Int. Conference on Measurement and Control*, pages 86–91, 1986.

[6] H. Groiss and J. Eder. Active Databases and CIM (in German). In *Tagungsband des ADV-Kongresses Informations- und Kommunikationstechnologie für das neue Europa*, pages 861–870, 1993.

[7] Heikki Hämmäinen, Eero Eloranta, and Jari Alasuvanto. Distributed Form Management. *ACM Transactions on Information Systems*, 8(1):50–76, January 1990.

[8] ISO and ANSI. Working Draft Database Language SQL (SQL3). Digital Equipment Corp. Maynard, MA, August 1993.

[9] V.Y. Lum, D.M. Choy, and N.C. Shu. OPAS: An Office Procedure Automation System. *IBM Systems Journal*, 21(3), 1982.

[10] Clarence Martens and Frederick H. Lochovsky. OASIS: A Programming Environment for Implementing Distributed Organizational Support Systems. *SIGOIS Bulletin*, 12(2):29–42, 1991.

[11] D. R. McCarthy and S. K. Sarim. Workflow and Transactions in InConcert. *Data Engineering Bulletin*, 16(2), June 1993.

[12] Michel Tueni, Jianzhong Li, and Pascal Fares. AMS: A Knowledge-based Approach to Task Representation, Organization and Coordiantion. *SIGOIS Bulletin*, 9(2):78–87, April 1988.

[13] M.M. Zloof. Office-By-Example: A Business Language that Unifies Data and Word Processing and Electronic Mail. *IBM Systems Journal*, 21(3), 1982.

ON THE ACCESS TO DISTRIBUTED DATA AND IT'S STANDARDIZATION

Libor Gála and Jaroslav Jandoš

Abstract

Paper is concerned with access to distributed data. Both forms of data are evaluated - files and tables (databases). The core of the paper is to show that the models of RFA (Remote File Access) and RDA (Remote Data Access) are basically the same and that standardization of remote access is needed for implementation of other than strictly homogenous systems. Various standards of DAP (Data Access Protocol) are shown.

Key words

Distributed data, DBMS , RFA, RDA, DAP, standardization, ODBC, IDAPI.

1. Introduction

By distributed system we understand a set of loosely coupled machines (so called "nodes") interconnected by a communication network. Loosely coupled machines are computer systems per se (including operating system and software), they don't share common memory and

communicate by messages. Data can be distributed into the nodes of this system either in form of files - which are managed by file system - or in the form of tables (databases), which are managed by (local) DBMS (Data Base Management System).

2. Files in the nodes

Files are generally managed by file system, usually a subsystem of operating system in given node, which provides file operations to its clients. File operations are generally based on client/server model. On distributed system the file system is organized as DFS (Distributed File System) which has the following basic features

- its data (files), client and servers are distributed among machines of distributed system
- it is a distributed implementation of centralized multi-user time sharing model of access to files
- it's distinctive features are the multiplicity and autonomy of clients and servers in the system
- it allows users of physically distributed machines to share data
- files located on given node are managed by (local) file system, according to requests of client process, which is typically located on remote node.
- it should (ideally) look to the clients like a centralized file system

The last feature leads to the requirement of multiplicity and dispersion of servers and files being transparent to clients. Transparency has several dimensions, namely

- network transparency, clients can access remote files using the same set of file operations as applied to local files
- location transparency, i.e. from the name of the file it's physical location is not seen
- location independence (often called "file migration") - it is not necessary to change the name of the file in case of change of file location

Location independence is a stronger property than location transparency.

Various DFS support these transparencies in various extent. While all support network transparency, most of them support location transparency (e.g. Sun NFS provides static location transparency), only some of them (e.g. Andrew) support location independence.

2.1. File operations

File operations provided by DFS can be divided into three basic groups:

a) (own) file operations as Open file, Close file, Read from a file, Write to a file, Delete file etc.

b) directory operations. By directory we understand a special file used for mapping of textual filename to physical filename. It contains also information on access control.

c) transactional operations, enabling execution of transactions on given file.

While most DFS support operations of groups a) and b), only some of them support operations of group c).

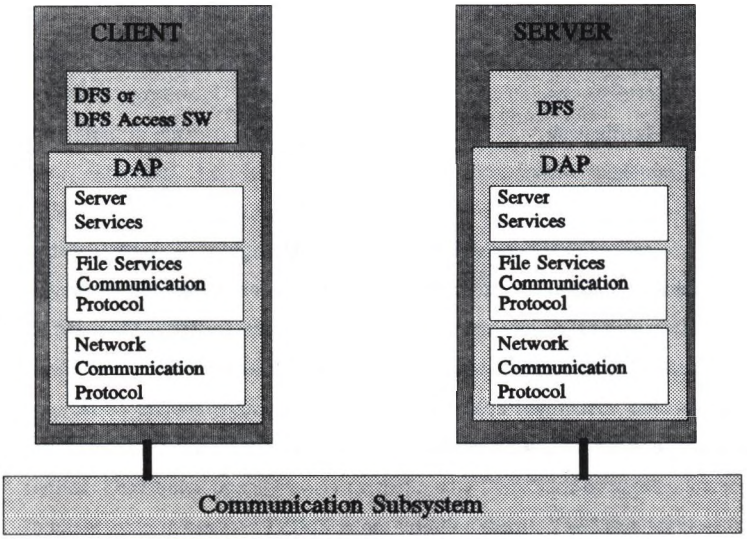
2.2. DFS implementation

DFS can be implemented:

- as part of the server "distributed operating system", e.g. Locus
- or as software layer managing communication between conventional operating system and file system, e.g. NFS.

DFS is implemented in all nodes of system which act as servers. In some DFS the given node can typically act either as client or as server (e.g. NFS), in others DFS the dedicated servers are typical (e.g. Andrew).

2.3. Client/server model - RFA



Pic.1: Basic Model of RFA

The access of client to files located on remote node of server is called RFA (Remote File Access), *see pic. 1.*

This client/server model is typically implemented using RPC (Remote Procedure Call).

The basic model underlying RFA is that of a file client (e.g. PC) accessing remote file server, which provides file services (by installed DFS software) by it's local file system (and local files) to a set of (local or) remote clients.

In this model the following elements are specified:

- server services expressed in command syntax of service requests
- (file services) communication protocol , i.e. message formats and communications rules

- network communication protocol, i.e.set of communication protocols defined in the various layers of communication architecture e.g. ISO or TCP/IP.

These elements comprise the DAP (Data Access Protocol).

Note: there are several other issues concerning DFS, namely transparency of replication (which is solved only partially), reliability, scalability and others.

2.4. DFS standardization

As the DFS is part of distributed OS (or is very closely tight to its kernel), file services of DFS are typically implemented in a system with all nodes running the same distributed OS. There is no international de jure standard (known to authors) for DFS. Many DFS follow the SunNFS syntax (function calls), which can be considered as de facto standard in this area. One of these DFS - Andrew File System - is considered to be another de facto standard of DFS and part of OSF DCE (Distributed Computing Environment).

3. Databases in nodes

Although the concept of file distribution - and file servers - is appropriate for certain applications, it has the following drawbacks. All operations on files - other than operation carried out by file server - i.e. selections, summaries etc, are carried out in such a way, that the file (or part of it) is transferred to the client, where these operations are carried out. Heavy network traffic is created and generally powerful client machine is required.

These drawbacks pushes the interest to distribution of databases into system nodes and therefore to the notion of database, rather than file, servers. Research and standardization in the area of database servers is much more active.

3.1. Basic architectures of databases on distributed system.

There are three common architectures of databases on distributed system. These are the following

- *independent databases*, i.e autonomous DBMS in various nodes of distributed system, with no data integration. Distributed transactions, if needed, are performed by application programm in client node.
- *federated databases*, i.e. loose cooperation of independent databases. Partial data integration and cooperation is provided by global rules (e.g. naming conventions, import/export mechanisms) which are obeyed by all DBMSs. There is still no GCS (Global Conceptual Schema) and distributed transactions/operations are carried out by application programm.
- *distributed database*. In this architecture the application program is provided with logical view of one logically centralized database. There is (one) GCS, providing high data integration. At the same time the autonomy of DBMS in various nodes is very limited. Distributed operations are carried out by DBMS (or by application programm in client node in case the DBMS is not installed there). The basic features of (ideal) distributed database are specified by (2). They include several flavours of transparency (independence on) .

Some basic features of these architectures are given in *pic. 2.*

Commercially implemented distributed databases are often based on DBMS of one vendor (e.g. Informix, Oracle, Sybase) - so called strictly homogenous DDBMS (Distributed Database Management System). As more heterogenous DBMS should be included in distributed database for various reasons (for instance corporation mergers), the problem of database standardization (RDA, DAP) is becoming more important.

Architecture	GCS	Integration	Autonomy
Independent databases	No		
Federated databases	No	increasing	decreasing
Distributed databases	Yes		

Pic.2: Some Basic Features of Database Architecture

3.2. Phases of access to distributed tables

In order to satisfy the data integrity rules, the access to distributed tables is based on transactional processing.

From the point of view of transaction structure, which consists of several function calls (SQL statements), the following three steps are usually considered:

I. All calls in given transaction refer to one and only one database. This structure is called "remote unit of work".

II. Every call in given transaction refers to one database. Various calls in given transaction can refer to different databases (in different nodes). This structure is called "distributed unit of work".

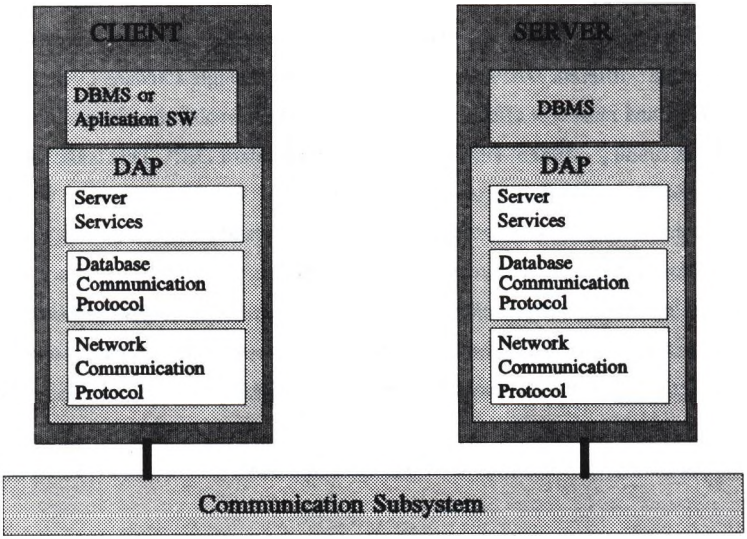
III. Every call in given transaction can refer to several databases. This structure is called "distributed request".

These steps are gradually implemented in all software for access to distributed tables.

3.3. Client/server model - RDA

Access of client to database located on remote node of database server is called RDA

(Remote Database Access). It is valid in all database architectures described above. Basic RDA model is the same as RFA model, *see pic. 3.*



Pic.3: Basic Model of RDA

In RDA model the following elements are specified:

- server services, expressed in database language
- (database) communication protocol (formats information for transmission across the network)
- network communication protocol

All these elements comprise the DAP (Data Access Protocol). The software components used for implementation of above given elements are called "middleware". The syntax of database language (and data formats) used in client application programm is determined by server and usually called as (server) API (Application Program Interface).

3.4. Standardization of RDA

Different DBMSs use different DAP (Data Access Protocol). If application needs access to different DBMS, different DAPs are required. The use of single standard DAP would lead to the simplest and most cost effective solution , as it would enable the given client to access many servers running different DBMS -- but all with standard DAP. This is the idea behind the standardization of DAP.

Currently there are several different de facto and de jure standards of DAP. They all specify approaches for accessing remote relational data (or non relational data that can be accessed by SQL) - using SQL language - managed on various hardware platforms by DBMS supplied by various vendors. All standards use client/server model. Although they all use (a versions of) SQL, there are many differences between all these DAPs.

Currently especially the following DAP standards are used:

IBM DRDA RDA

This DAP, as well as DRDA (Distributed Relational Database Architecture), was developed by IBM. New developments are based on consultations of IBM with DRDA Implementor s Advisory Council, which includes Sybase, Ingress, Oracle and others. DRDA was designed as high performance architecture (large enterprise, transfers of large data quantities). DAP uses, as all DRDA, SNA communication facilities - namely SNA LU 6.2. At the moment it is mostly implemented in IBM environment. DRDA, as well as ISO RDA and other quoted DAP standards, supports multiple levels of SQL standard. SQL servicing in DRDA is the best of all presented DAPs, as it enables to use specific SQL commands for various servers (SQL DBMSs), even commands outside the scope of SQL standards. As well as ISO RDA, this DAP supports remote unit of work and distributed unit of work - with provision for two phase commit, which is valuable for distributed applications.

ISO RDA

This DAP was released as ISO international standard ISO 9579 in March 1993. It uses ISO OSI (Open systems Interconnection) communications. It supports the common subset of SQL standards. On the contrary to DRDA DAP, ISO RDA is considered to be a low cost specification, which is easy for software vendors to implement.

X/Open RDA

This DAP is based mainly on SAG (SQL Access Group) contribution. It was published in May 1993. Key participants include main computer and database vendors. It is based on ISO RDA, however implementation of X/Open RDA is not able to interoperate with implementation of ISO RDA and also none of these DAP is subset or superset of the other. X/Open therefore does not support some of SQL facilities supported by ISO RDA (for instance defined DBL commands, which allow for execution of stored SQL statement on server) and support some other SQL facilities supported by ISO RDA. TCP/IP is the communication environment of X/Open RDA.

ODBC

ODBC (Open DataBase Connectivity) DAP was released by Microsoft in 1992. SQL syntax is based mostly on SAG specification (Phase I and Phase II) and it is superset of SAG CLI (Call Level Interface). Other facilities, especially formats and protocols, are based on ISO RDA.

SAG (PI, PII) does not specify stored procedures, nor two phase commit (so it is not possible to use distributed unit of work). These facilities will be specified in SAG Phase III.

ODBC was initially targeted on Windows clients, but later has been implemented in many client s environments including OS/2, Macintosh etc.

ODBC standard does not specify network connectivity , on the contrary to SAG, which specifies ISO over TCP/IP network protocols. ODBC supports 3 levels of SQL syntax. It provides access to relational databases and some support for nonrelational databases.

IDAPI

IDAPI (Integrated Database Application Programming Interface) represents Borland's recent specification of DAP (relaunch of Borland's former ODAPI as IDAPI), with sponsorship of IBM, Novell and Wordperfect. Specification was released in February 1993. It is based on SAG and X/Open work and represents an extension of SAG CLI namely in the direction of record at a time interface to nonrelational databases - dBase and Paradox. IDAPI is oriented towards OS/2, DOS and Windows client environment.

4. Conclusion

This paper is an attempt to show

a) that the basic models of RFA and RDA are the same, with differences shown above

b) that implementation of systems with distributed data is simplified, and in many cases enabled at all, by standardization.

In case of distributed files the de facto standard is NFS syntax, with AFS being the typical modern solution.

In case of distributed databases, the implementation of systems with heterogenous relational SQL databases is made possible by RDA or DAP standardization. In this context, two facts should be kept in mind:

- every given DAP (DRDA, ISO RDA, X/Open, ODBC) is oriented towards specific applications and environment. It remains to be seen, if and how quickly the unique standard DAP is accepted, which covers all environments and platforms
- standard server API (as part of DAP standard) is basically a common subset of API s of various database engines (DBMS). It therefore limits users to the functions supported within the standard server API (e.g. ISO RDA) specification. Very powerfully functions, which are specific for given DBMS, cant be used. Especially these functions made distinction between DBMS of various vendors (Oracle, Sybase etc).

Each DBMS could provide the "native API" which enables use of all its functions, as well as "standard API" which enables easy integration of DBMS into heterogenous system. Powerful applications will be implemented with native API, cooperative ones in heterogenous environment with standard API.

5. References

2. [1] Colouris G., Dollimore J.: Distributed Systems, Concepts and Design, Addison Wesley, 1988
3.1. [2] Date C.J.: An Introduction to Database Systems, Vol.I, Fifth Edition, Addison Wesley, 1990
3.4. [3] Finkelstein R.: ODBC, Beware the Hype, Network Computing, January 1994
3.4. [4] Hoven J.: Plug n Play Data: Standards First, Database Programming and Design, January 1994
3.1. [5] Jandoš J.: Distributed Computer Systems (in Czech), Senzo, Praha 1993 (ISBN 80 901245 18)
2.2. [6] Levy E., Silberschatz A.: Distributed File Systems: Concepts and Examples, ACM Computer Surveys, December 1990
3.3. [7] Lamersdorf W.: Remote Database Access, in IAO Forum T15, Springer 1990
3.1. [8] Özsu M., Valduriez P.: Principles of Distributed Database Systems, Prentice Hall, 1991
3.4. [9] Robertson B.: Database Networking Middleware: Where the Action Is, Network Computing, January 1994
3.4. [10] Zimowski M.: DRDA, ISO RDA, X/Open : A Comparison, Database Programming and Design, June 1994

Workflow — Past and Present

A Part of Office Automation History

Győző KOVÁCS
Vice President
John von Neumann Computer Society
1054 Budapest V. Báthori u. 16

Since using milestones helps dealing with history, informatics historians like to think in periods and categories. Computer hardware history has more or less been put in its place as the definition of first-, second- and third-generation machines is accepted by virtually all. When fourth- or even fifth-generation equipment were born, however, is not as evident. The sixth generation is not yet mentioned by computer science history.

To determine software periods is even harder: I will not try to uncover this jungle.

When I started to write this paper, I was faced with a dilemma right at the start: is office mechanization a part of office automation history, and if so, which machines should be considered here. I remember my childhood when my father was an accounting teacher during the forties in Baja, a small Hungarian town located in Alföld (the Great Plains). To supplement his wages, he did bookkeeping for a number of local stores. I visited him many times at the office of the Kattarinka shoe store where I could watch in wonder the Odhner calculator, the automatic pencil sharpener, the amazing stamping gadgets, the stapler, the hole puncher, the typewriter, the telephone (Fig. 1); I believe there was no other machine at this office said to be modern at the time.

I saw the first "inking copier", the famous stencil, in use at many places even today, at the school my father was teaching. My friends and I used this stencil for making copies of the first self-edited student publication of Baja (Fig. 2 shows a later machine, the GESTEFAX, used to make plastic stencil sheets).

I think my first encounter with an electromechanical calculator took place sometime during the early fifties when working as an apprentice for a year prior to university (Fig. 3: SOEMTRON desktop calculator without printing device, German Democratic Republic). A secretary of the Röntgen plant asked me to take a look at her faulty calculator. Although I quickly found the problem, because the ingenious mechanism of the calculator amused me so much, I gave the machine back only late afternoon. The next period of office mechanization is perhaps signified by the electromechanical calculators.

Classing of the punched-card or Hollerith-machines is ambiguous. As it is known from informatics history, the first punched-card data processing system was created by Hermann Hollerith (1860-1929) who worked from 1879 to 1883 for the U.S. Census Office of the Department of Interior established specifically for the 1880 census. There he developed a punched-card equipment utilized during the 1890 census. I have found one of the best-preserved original

Hollerith-machines at the Popov Museum of Moscow: the turn-of the-century Russian census was done with the help of this equipment, too. Hollerith has also established a trade organization first called the Tabulating Machine Company then, in 1911, the Computer-Tabulating Recording Company. Thomas J. Watson Sr. further developed the organization to International Business Machines Corporation, today's Big Blue, or IBM.

The idea of applying punched cards is supposedly originated form John Shaw Billings who, while taking part in a census, brought to Hollerith's attention the punched cards used on the Jaquard-loom. (Perhaps it is worthwhile to mention here that Jaquard's cards was first utilized by Charles Babbage [1791-1871] for storing data. I have not found any information implying that Hollerith would have known this.) The Hollerith machines were first semi-automatic organizers without printing capability: the result was indicated by the many circular displays (Fig. 4). Printing calculators were only developed in 1906-1907. The equipment was continuously being perfected up until the appearance of computers in the 1950's. During this process special equipment were developed for the processing line, such as operational (adding and multiplying) machines, modern printers and punched-card copiers. The purpose of the copiers was partly to increase effectiveness and partly to widen the field of applications (Fig. 5 shows an IBM 514 card copier).

Hollerith-machines were also used for calculating astronomy tables in 1928. Although utilized for data processing as well, only large organizations could afford to buy and operate these big and expensive machines. In Hungary the first Hollerith machines were delivered by IBM. (IBM was the only company not nationalized during the 1940's. This was probably because computing capacity was much needed and perhaps because IBM was delivering parts to Hungary via Switzerland during World War II despite the U.S. being in a state of war with Hungary.) The largest users were: *Diósgyőri Gépgyár* (DIMÁVAG; Machine Factory of Diósgyőr), *Elektromos Művek* (Electrical Works), *Ganz-Mávag Központi Statisztikai Hivatal* (KSH; Central Statistics Office) and others. There was also a payroll processing center at the *Kohó- és Gépipari Minisztérium Gépi Adatfeldolgozó Vállalata* ((KGM GAV; Data Processing Company for the Ministry of Metallurgical and Engineering Industry) founded in 1950, where mostly data processing for ministries and companies was done. The countries largest data processing company was founded in 1951 within the framework of KSH. Renamed to *Statisztikai Gépiadatfeldolgozó Vállalat* (SGAV; Statistical Data Processing Company) in 1953, the company was developed into the later country-wide *Számítástechnikai Ügyvitelszervezési Vállalat* (SZÜV; Computing Company for Management Organization). The company first had partly IBM-, partly Soviet-made (SAM), also Hollerith-type machines (Fig. 6: punched-card organizer). The first electronic machines, a Polish UMC-1 and a Danish GIER (Fig. 7: A GIER at the Museum of Helsingør), arrived at the beginning of the sixties. The GIER was a wonderful machine since it operated much

more reliably than both our M-3 created from 1957-1959 and the URAL 2 bought from the Soviet Union. These machines, together with later computers, were being used for continuous payroll calculations throughout the country. Perhaps the founding of this organization was the second largest step in Hungarian office automation after the Hollerith-machines.

Let us talk a little about accounting, too, which is a very special part of office work. Automating accounting did not only speed up financial calculations but also multiplied their level of precision.

I started this paper with my father's accounting work. Back then there were no accounting machines: instead, my mother and I added up the companies' long columns of numbers every evening to check the work of bookkeepers. I can certainly appreciate the advantages of accounting automation.

Since socialist countries generally attempted to establish a self-sufficient economy, in this part of the world - according to the regulations in effect in socialist countries - mostly equipment developed in other "friendly" countries could be bought (Fig. 8: Optima office system). Contrary to capitalist systems, a very strict centralized decision-making process were in effect, in which politics dictated the terms of not only distribution but of technical development, as well. Market had no role in socialism since it itself operated according to dictated terms. In general it can be said that one big difference between the socialist and capitalist production was the amount of money allocated to development: as this amount was very low in socialist countries, the technical standards of socialist products were sinking ever lower; the development of certain technologies either never or only years later took place. The best example for this effect perhaps is the production of semiconductors and, later, integrated circuits.

In my opinion the so-called specialization also did much harm to the socialist economy. This meant that, due to generally political and not technical decisions, the development and production of certain product groups were relocated to other socialist countries for reasons of economy. This is how the mainstream of business machine production got located mostly in the German Democratic Republic and partly in Czechoslovakia. Thanks to this decision, the promising development of a punched-card machine stock started fairly late (in 1953) at the *Irodagép Kisérleti Vállalat* (Business Machines Exploratory Company) was stopped. At the same time a very strong business machine industry was created in the GDR within the framework of the SOEMTRON and ROBOTRON companies. Another but much smaller basis of business machine production was the Czech ARITMA.

An interesting chapter of office automation history I took part in started in Hungary somewhat later, in 1959. The group of engineers and mathematicians just having finished the development of M-3 together with a few engineers at *Telefongyár*

(Telephone Factory) started the development of an application-specific computer, the EDLA-1 and, later, the EDLA 2. The machine was the invention of Dr. László Edelényi and Dr. László Ladó. The development resulted in a working model when it was terminated at the beginning of the sixties.

From the above it is perhaps evident that office mechanization was but a step on the way towards office automation; only certain parts of office processes could be made more effective with the help of equipment already mentioned.

Another important part of office work is telecommunication, which meant about two things until the eighties: the telephone and the telex, called teletypewriter officially (Fig. 9: Siemens teletypewriter with a ribbon printer). Before World War II, the telephony industry was very strong in Hungary. One of the centers of production was the *Standard* factory, where modern equipment partly under licence and partly of their own inventions were being manufactured.

The most significant period of office automation started when, at the end of the seventies and the beginning of the eighties, a small, not too expensive, complex data processing equipment could get into the offices. I think I do not even have to say that the equipment in question was the IBM PC along with the small printer (Fig. 10: Proper 16, the first Hungarian PC-clone made by SzKI). In my opinion this is the time when office automation began. More exactly, even a bit later, when mass-produced software reached the market.

First I would mention VISI-CALC which simplified office computations, spreadsheets and all kinds of financial calculations.

The other breakthrough was signified by the appearance of word processors, as a result of which the era of typewriters then of electric- and memory typewriters were terminated.

The third large group of office automation software - we are still at the beginning of the eighties - consisted of database programs which caused the extinction of the wonderful filing systems, file storage technologies, carrousels and other office wonder-products.

I have not mentioned the dry copy or xerographic method which arrived quite late in Hungary. Perhaps it is worthwhile to mention that the first copy was made by Chester Carlson in 1938: the first showing of the machine took place at the ASTORIA in New York on October 27. The theoretical results of the Hungarian physicist Pál Selényi unsuccessful in making a machine or any profit was utilized by the inventor to develop the first copy machine. Business only started in 1955 in the U.S. when XEROX Corporation was founded (Fig. 11: Rank Xerox D.E.O. copier). I have heard such opinion, as well, that the real copier revolution started when the licence was made available to others, too. The Japanese giants, for example,

integrated their own inventions, i.e. the application-specific computers, in copiers.

A very important moment of office automation history was the birth of the telecopier or fax machine, which, in simple terms, worked as two xerox machines connected to each other: one of them read the paper while the other one wrote on it, and vice versa.

We have arrived at the end of the eighties when almost everything in office work was automated; intelligent electronics appeared in machines in the form of computers and, often, application-specific computers. A very important part of office automation were programs that, for example, performed accounting so there was no need for accounting machines; that stored files so there was no need for filing cabinets; that stored pictures electronically so there was no need for storage of drawings, pictures and documents. All documents were created on computers so many in-between processes and, more importantly, duplication of inputting data could be spared.

Data networks connecting offices with each other were built, which could replace the limited-capability telephone, telex and fax systems. World-wide data networks were formed and the walls of the perhaps-never-to-be-completely-paperless office disappeared: by today the whole world has become a huge electronic office.

In today's office the main role is played by the computer and everything else can be considered its periphery. The first of these is the keyboard which will be the only input device until reliable voice input and voice recognition software will be available. Since there is no paperless office yet, the printer is another important periphery; its newest form is the copier-printer.

Storage devices play a very important role in the offices: not only those in computers but external equipment, especially CD-ROMs, too. Although I realize prophecy is a dangerous business, I still predict that the days of rotating mechanical, magnetic and optical storage devices, such as the hard disk drive, floppy disk drive, CD-ROM and CDi, are numbered. Not that they have a less than perfect mechanism but because they are rotating, their lifetime is limited. The storage devices of the new age will be semiconductors, will operate statically, and will differ from today's semiconductors in that they will not require power to keep their contents. Perhaps it could be guessed that I am talking about flash memories, the first very usable versions of which have been manufactured by INTEL for some time.

Telephones and fax machines are peripheries, as well: we can forget about faxes as they could be replaced by software. Moreover, e-mail has almost completely overtaken office correspondence.

The modern automated office is really an information processing plant where matter takes different forms of information: mostly

text, as well as pictures, graphics and voice. At this plant everything is directed by a computer. The system and the method will probably change very little during the development process: peripheries and storage devices will have to be changed at most as more modern equipment always replace the old.

Transmission devices of telecommunications equipment will change with high probability, too, as transmission speeds will have to be increased due to the transport of still and moving pictures.

We can expect the biggest change in the area of application programs: these software will be able to solve ever more complex office tasks very intelligently.

Acknowledgements: I thank the Siemens Museum of Munich for the pictures of Figs. 1, 4 and 9. Further thanks to the *Központi Statisztikai Hivatal* (Central Statistics Office) for a part of the rest of the pictures; the remaining pictures have been made by the author.

Bibliography

[1] Goldscheider, Peter, and Zemanek, Heinz, *Computer, Werkzeug der Information* (Computer, the Information Tool), Springer Verlag, 1971

[2] Goldstine, Herman H., *The Computer from Pascal to von Neumann*, Princeton University Press, 1980

[3] Kovács, Győző, *A számítógép-fejlesztés korai szakasza Magyarországon* (The early period of computer development in Hungary). Published as an appendix of [2]

[4] Kovács, Győző, *Magyarok a számítástechnikában* (Hungarians in computer science), Technikatörténeti Szemle , 1988-89

[5] Kovács, Győző, *A fénymásolás rövid históriája* (The short history of photocopying), Természet Világa, 1993

[6] Szentiványi, Tibor, *A számítástechnika kezdetei Magyarországon* (The beginnings of computer science in Hungary), Természet Világa, 1994

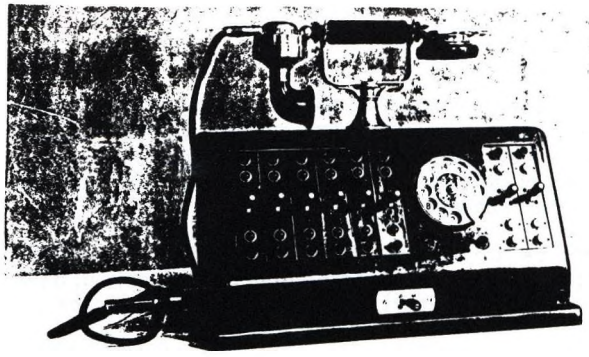


Fig. 1.

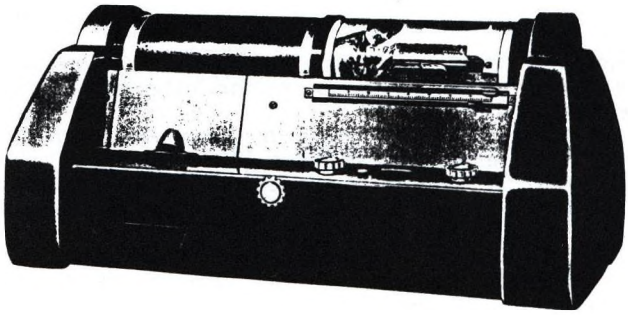


Fig. 2.

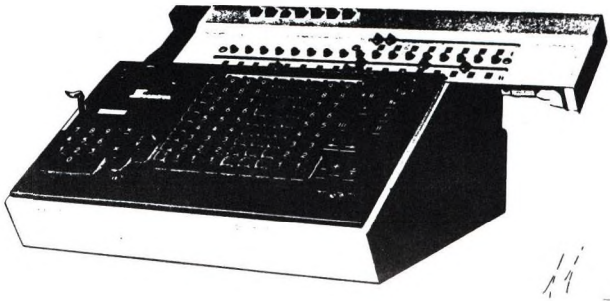


Fig. 3.

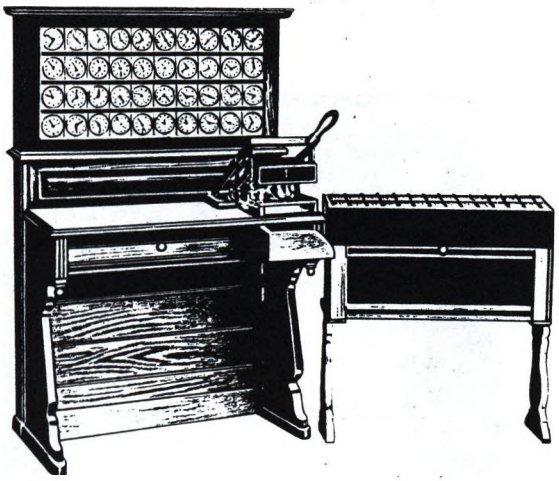


Fig. 4.

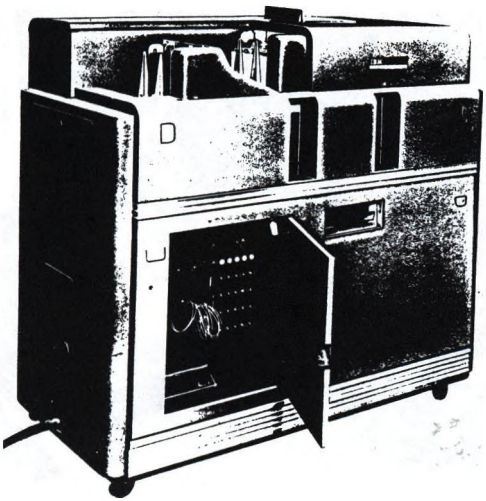


Fig. 5.

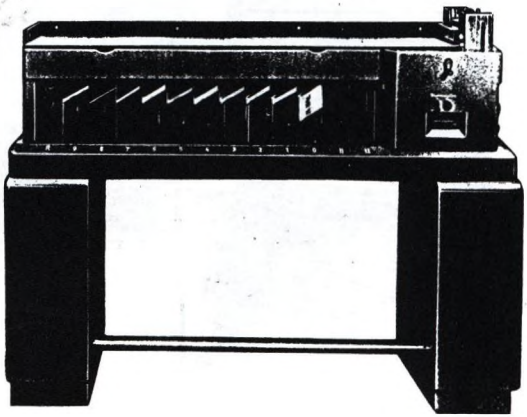
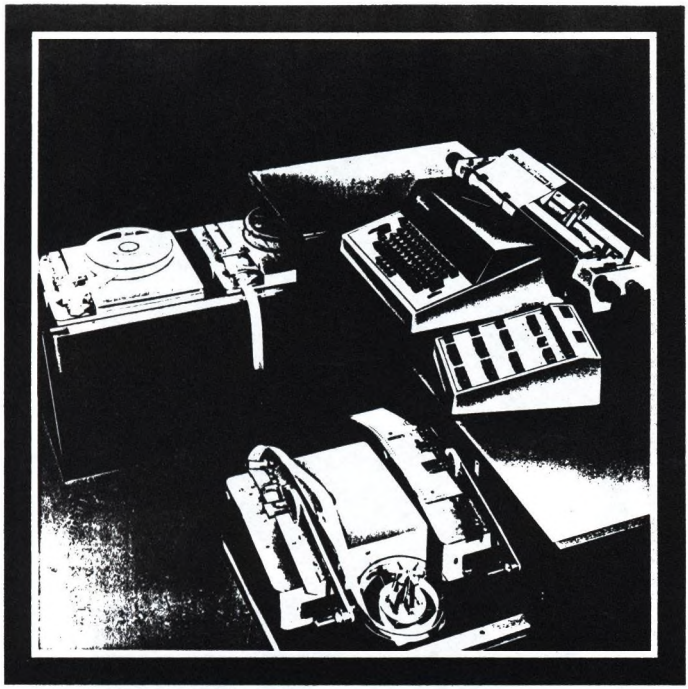


Fig. 6.

Fig. 7.



Optima

Fig. 8.

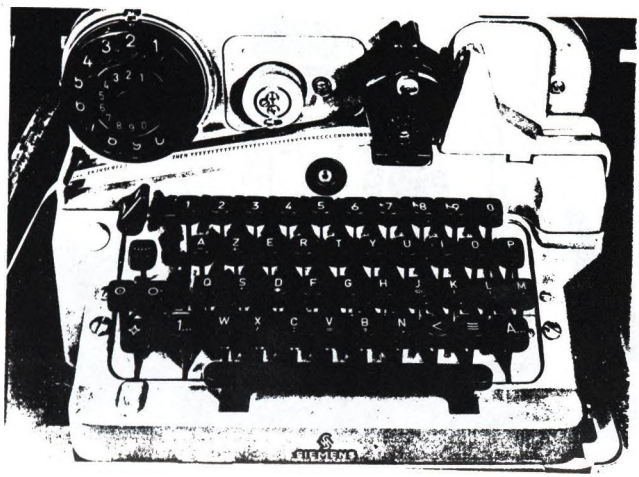


Fig. 9.

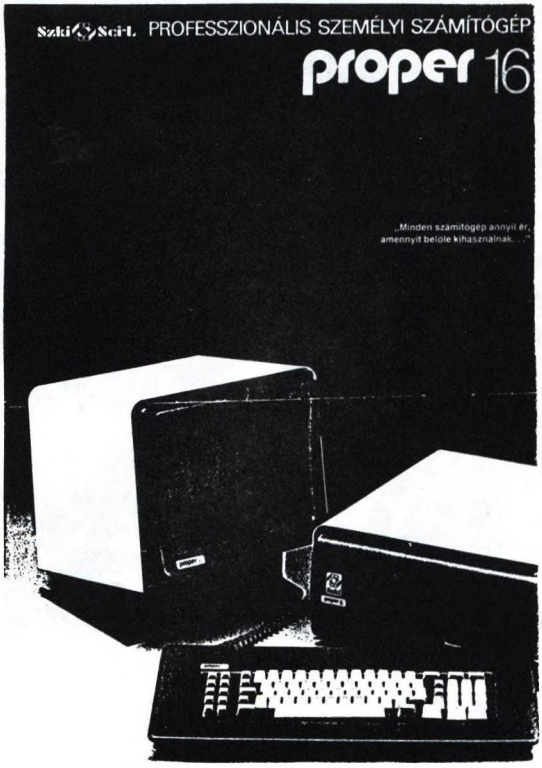

SzKI SzIP PROFESSZIONÁLIS SZEMÉLYI SZÁMÍTÓGÉP
proper 16
„Minden számítógép annyit ér,
amennyit belole kihasznalnak. . .“

Fig.10.

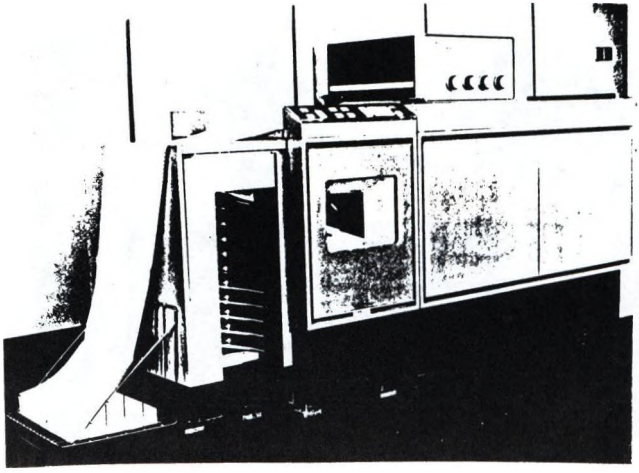


Fig.11.

Workflow System

User Survey 1994

G. Chroust[1]
J. Bergsmann[2]

1.0 Is Workflow Management Desirable?

Workflow Management, *the computer assisted management of business processes through the execution of software whose order of execution is controlled by a computerized representation of the business process* [WFC_94] is today considered to be one of the key technologies for providing efficiency and effectiveness in the office [Bergsmann_94].

Its basic idea is to formally define the flow of work in the office, then to use a computer environment to effectively support this flow. This concept has received considerable attention:

- First of all this idea is based on a paradigm which is in use in many other industries (software engineering, manufacturing etc. [Chroust_92c]).
- Secondly the rigid definition of procedures was always one of the cornerstones of bureaucracy.
- Thirdly we observe currently a general fascination with CSCW (Computer Supported Cooperative Work [Friedrich_91] [Turner_92]).

Workflow is considered to be an important area of CSCW. At first building a network of work stations was a small technical enhancement to the state-of-the-art technology. Very soon the organisational and communicational possibilities of such networks triggered a completely new way of work and of cooperation.

If we look more closely it seems that most CSCW products have come into life due to a technically attractive solution and not because of an explicit user requirements: Having many individuals, especially information workers, connected via a network, was a natural starting point for asking whether one could not support their work electronically and thus provide a complete new way to do work.

Despite all the heralding and excitement it seems that the actual acceptance of such products is quite low [Chroust_94h] [Grudin_94]. Exaggerating somewhat, somebody recently stated *"The only successful CSCW-product is e-mail!"*.

It is an old experience that products become successful only if technical, sociological and commercial interests really overlap.

---

[1] Systems Engineering and Automation, J. Kepler Universität Linz, 4040 Linz, Austria

[2] AGENS GmbH, 4020 Linz

It seems that this is the case with workflow:

- The technology for workflow is provided by the existence of networks and by the groundwork done for software engineering environments.
- In bureaucracies the observation of predefined, detailed procedures ('process models') has a tradition of several thousand years.
- Both the drive for business reengineering [Hammer_94] and the need to become ISO-9000 certified generate a need to explicitly write down office procedures and *ensure their observation*. It is generally recognized that the reorganisation and the efficient execution of business processes will be an increasingly important factor for the achievement of a company's strategic goals.

Nevertheless the acceptance of workflow products has been slower than anticipated by industry. Various reasons could be made responsible for that.

2.0 A User Survey on Workflow

In order to sound out these questions the Deptartment of Systems Engineering and Automation of the Kepler University Linz together with GES, a professional training company, started a user survey on the subject. This survey follows the successful 'User Survey on the Use of CASE-Tools in 1993 ' [Schulz_93]. A consulting company (AGENS Linz) and an institute for market analysis (IMAS GmbH Linz) joined the team.

The survey was restricted to the German speaking part of Europe (Austria, Germany, Switzerland). Approximately 10.000 forms have been mailed out, with a return deadline of October 25, 1994. Answers are anonymous. Preliminary Results are expected to be available by year end 1994, the final report [Chroust_95] will be ready in March 1995 [3].

3.0 Structure of the Questionaire

From past experience and from literature study [Bergsmann_94] several problem areas were identified, which where expanded in the survey. In total the questionaire contains 68 questions, offering a total of approx. 300 subquestions. The questionaire addresses both enterprises which are already using workflow management and those which are not yet using it.

The main question areas were

structure of organisation (5 questions)
organisation, qualification of personell
business processes (6 questions)
their number, involved users, documentation
computer equipment (7 questions)
type of hardware, software and network, groupware products
document administration (7 questions)
creation, storage and retrival of documents
workflow system (if already in use, **12 questions)**
system type, penetration, important features, expectations, productivity gains, quality gains
workflow system (if not in use, **3 questions)**
reasons for not using it, future plans

[3] Copies of the report can be ordered from OCG - Austrian Computer Gesellschaft, Wollzeile 1-3, 1010 Wien, Austria, tel. 512-02-35, fax 512-02-35-9, email ocg@vm.univie.ac.at

workflow introduction (13 questions)
initiation, responsibility, strategy, duration and effort, use of external consultant, cost
quality management (4 questions)
effect, responsibility, ISO 9000
expectations of various groups (4 questions, many subquestions)
computers and workflow, chances and risks, change of personal
gross enterprise data (7 questions)
(used for calibration of answers) area, size, country, turnover

4.0 Results

At the moment no results are available yet, at the time of the conference a few tentative results will be presented.

5.0 References

[Bergsmann_94] Bergsmann J.: Workflow im gewerblichen Bereich.- Diplomarbeit, Kepler Universität Linz, Sommer 1994.
[Chroust_92c] Chroust G., Leymann F.: Interpretable Process Models for Software Development and Administration.- Trappl R. (ed.): Cybernetics and Systems Research 92, Vienna, April 1992, World Scientific Singapore 1992, pp. 271-278
[Chroust_94h] Chroust G.: Groupware - Wer will es? ADV (ed.): ADV-Fachkonferenz Groupware'94, 8.-9.Juni 1994, Wien
[Chroust_95] Chroust G., Bergsmann J.: Repräsentativumfrage WORKFLOW 1994 Schriftenreihe d. Österr. Computergesellschaft, Oldenbourg 1995
[Friedrich_91] Friedrich J., Rödiger K.H. (eds.): Computergestützte Gruppenarbeit (CSCW) - Fachtagung, Universität Bremen 1991.- Teubner Stuttgart 1991.
[Grudin_94] Grudin J., Eight Challenges for Developers.- Comm. ACM vol. 37 (1994) no. 1, pp. 93-105
[Hammer_94] Hammer M., Champy J.: Business Reeengineering - Die Radikalkur für das Unternehmen.- Campus Frankfurt/M, 3. Auflage, 1994
[Schulz_93] Schulz A.: Repräsentativ-Befragung zum Stand der Software-Entwicklung in den deutschsprachigen Ländern.- GesmbH, Im Vogelsang 14-16, D-7753 Allensbach, Spring 1993
[Turner_92] Turner J., Kraut R. (eds.): CSCW-92, Sharing Perspectives.- Proc. Conf. on Computer-supported Cooperative Work, Oct.13-Nov.2, 1992 Toronto., ACM Press 1992
[WFC_94] Workflow Coalition (ed.): Working Paper.- Vienna, July 5, 1994

Workflow Applications in Hungary

J. Szlanko

Paper not received in time.

Workflow Management — History and Goals

G.A. Pitschek

Paper not received in time.

Tutorials

Tutorial:

Active Database Systems

Klaus R. Dittrich [1]

Many applications (e.g., computer integrated manufacturing, office workflow control, process control, program trading, and system management) require timely response to critical situations. However, the points in program execution at which those situations occur often cannot be predicted or preplanned.

Such applications are not well served by conventional database systems, because these systems are passive: they execute queries and transactions only in response to explicit requests from users or application programs. An active database system, on the other hand, monitors events (happening inside and outside the database) and conditions (defined against the states of the database), and automatically (i.e., without user intervention) invokes specified actions when these events and/or conditions occur.

Active databases have recently emerged as an important (and flourishing) area of research, and several prototype systems are being built. This tutorial will provide motivation and a historical perspective on active database systems, survey the state-of-the-art in this field, identify the key technical problems in the design of an active database system, and summarize some approaches being taken in research projects and (at least to some extent) in commercial products.

[1] Database Technology Research Group, University of Zurich, Switzerland

Tutorial:

Workflow

Clarence (Skip) Ellis [2]

This one day seminar will introduce, describe, and explain the concepts, technologies, and applications of workflow systems. Workflow systems are networked computer systems employed in organizational settings to assist groups of people in executing work procedures. These systems usually contain knowledge of how work normally flows through the organization so that they can act as coordinators. A workflow system typically contains two components: a workflow modelling component, and a workflow enactment component. Based upon the instructors current work and his many years of work in this domain, this tutorial will explain and explore in detail these two components.

More specifically, the workflow tutorial will cover the following topics: definition and history of workflow systems, their underlying technology and architecture, the relationship between workflow design and business process re-engineering, as well as a survey of currently available workflow products. The tutorial closes with an analysis of existing pitfalls and an outlook to further developments.

2 University of Colorado, Boulder, CO USA

ISBN 3-486-23159-6 Oldenbourg München
ISBN 3-7029-0397-6 Oldenbourg Wien
ISBN 3-85403-076-2 Österreichische Computer Gesellschaft