Loosely integrated sets of CASE tools: our experience

Kari Känsälä

Technical Research Centre of Finland
Laboratory for Information Processing
Helsinki, Finland

Abstract

Software engineering was in serious trouble in the beginning of this decade because of lack of proper practices and procedures, tools and co-operative knowledge. Software production workstations, which combine modern hardware and software technology with understanding of both end-user's and software engineer's needs and wishes, seem to resolve many problems in the near future.

A four-level model of software production and three sets of CASE tools based on that model are presented. The tool sets are

- software project manager's workstation OHTO (developed in 1985 - 88)
- software quality/configuration manager's workstation SAMPO (developed in 1983 - 86) and
- software developer's workstation ELINA (developed in 1986 - 87).

The basic issues behind the tool sets are both flexibility and uniformity. They are achieved by using

- open tool set architecture, which allows use of popular commercial tools for general purposes (e. g. for text processing) and for special purposes (e. g. for drawing data flow diagrams) and
- both official and de-facto standards at all possible levels (e. g. hardware, system software, documentation).

The main strategies chosen for development work were:

- commercial technology transfer
- incremental development and
- intensive role of companies.

Experience gained during the six years of corresponding development work, covering both technical and co-operational issues, is described. Also a recent project related to distributed environment for software production is shortly described.

Keywords: Software production management, software project management, software configuration management, software documentation,software management tools, workstations

Loosely integrated sets of CASE tools: our experiences

Kari Känsälä

Introduction

The glorious decade of software engineering, the seventies, with its structured programming techniques, top-down analysis and design methods and waterfall models of development processes ended with confusion:

- end-users did not like cryptic methods and rigid development models
- even we ourselves did not like our own cumbersome techniques and out-of-date alphanumeric software tools.

But worst of all, we did not understand the nature and processes of our own work and therefore, did not manage it properly.

In the eighties everything seems to get better. We have accepted the existence of end-users and their requirements; sometimes some of us are even trying to see things their way. Enormous progress of hardware and software technologies has given birth to a generation of very good tools for both software engineers and software managers. And the best thing is that we are beginning to be able to merge the good things of the seventies (basically sound methods and techniques) with the good things of the eighties (tools and new mutual understanding).

In the beginning of the eighties we at VTT (Technical Research Centre of Finland) started to develop a series of software engineering workstations: first within the Software Engineering Research Program of VTT and later within the Finnish Program for Research and Development in Information Technologies (FINPRIT). We developed three workstations - OHTO, SAMPO and ELINA. The focus of these workstations is on software management activities based on a model of software production and projects. We have come a long way and learned many hard lessons, but we think we have made many right decisions on our way.

Software production and project management model

Software production is an activity of a company if it produces software applications (for its other activities or for other companies), software packages or embedded software.

Software projects usually cover most of software production activity. They may concern

- software development
- software maintenance or
- software customization

or a mixture thereof.

Software projects include many different kinds of activities on many different levels. We analyzed software production processes in several major Finnish software-producing companies /1/ using most of the well-known authorities in the field of software engineering as our theoritical background /2/. Based on that analysis we developed a *four-level model of software production and projects* (figure 1) /3/.

This model may be described as a software factory having four floors (above the ground floor including the support activities of software production).

The first floor is the manufacturing floor: there software is specified, designed, coded, unit-tested and integration-tested, maintained, enhanced and customized (*software work* or *software engineering in the narrow sense;* software engineering in the broad sense covers of course whole software production).

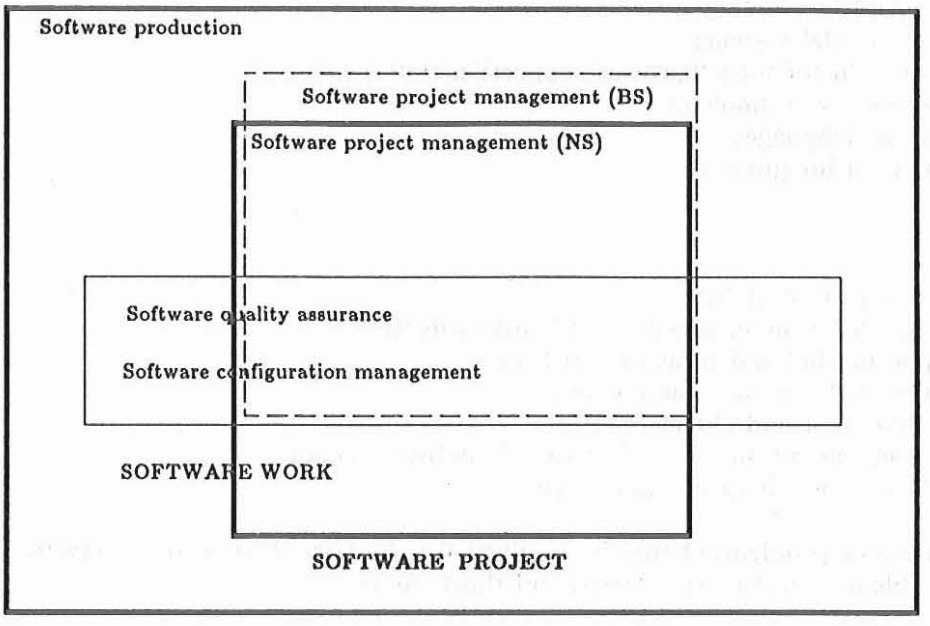

Figure 1. Four-level model of software production and projects.

The second floor is the storage and quality assurance floor: it contains all *software quality assurance and configuration management* activities within and between software projects. This floor is actually the heart of the software factory: it assures the quality and continuity of software production.

On the third floor work software project managers take care of operational project management in order to complete software projects on time, within budgets and with a specified quality. This means *software project management in the narrow sense (NS).*

On the fourth floor work software production managers who are responsible for organizing projects, acquiring and allocating resources for projects. Besides all this they have to organize an appropriate generic software engineering environment for their software production unit.

Software project management in the broad sense (BS) covers activites on the second, third and fourth floor:

- software project management (SPM)
- software quality assurance (SQA)
- software configuration management (SCM)
- that part of project management activities that has not been transferred to the project manager by his superiors; e.g. decision making in the project steering group.

Any software project consists of software project management in the broad sense and activities on the first floor.

In this paper we use the term 'software project management' in the narrow sense unless otherwise specified.

Workstation approach

There are many severe problems of managing software projects:

- a lack of agreed terminology
- incomplete and ambiguous requirements
- imprecise and incomplete specifications
- inability to model systems
- uncertainties in software/hardware apportionment
- rapidly changing technology
- suitability of languages
- problems with integration

and

- a general lack of visibility
- difficulties in resource, schedule and cost estimation
- inability to predict and measure reliability
- difficulties with progress monitoring
- complicated error and change control
- a lack of agreement on test and quality assurance metrics
- difficulty of controlling of maintenance.

The first group of problems should be resolved for the first floor of our software factory ; the second group of problems concerns the second and third floors.

We state that there is a cure for many of those problems: *software production workstations*. By our definition

***software production workstation** is an integrated set of*

- *special software tools*
- *general software tools and*
- *other (non-software) tools*
- *combined with suitable hardware and system software*

which serves one person doing his/her software work or management activies and which has been customized for his/her needs and wishes taking into account standards and procedures of the company.

Special software tools cover customized software engineering tools like software specification, design, coding and testing tools on the first floor, SCM and SQA tools on the second floor and special SPM tools like software cost estimation models on the third floor.

General software tools cover tools that are used as well in other activities than software production, e.g. text processing, spreadsheeting and data transmission tools.

Other tools include all 'manual' tools like standards, guides, method descriptions, etc.

The above definition may sound trivial and self-evident but it contains two key (and almost contrary) issues of success in software production

- *uniformity* (workstations bring concistency and coherence into software production by giving all users the same framework of methods, techniques and company procedures; this helps to gain optimal efficiency of project groups)
- *flexibility* (workstations gives all users the possibility to customize the user's working environment, which helps to gain optimal effectiveness of individuals).

We decided to set several further development principles for our workstations in order to enhance their capabilities

- *open architecture;* tools can be added, changed and removed for flexibility reasons according to uniformity restrictions
- *use of common general tools;* people certainly prefer to use e.g. same text processing software in software workstations as for other purposes
- *use of best special tools;* it is very sensible to use the best special tools available in the commercial marketplace for every single special purpose within software production whenever it is cost-effective
- *use of other tools in accordance with software tools;* in principle prodedures and practices, methods and techniques should be chosen first and software tools accordingly (today, in many cases it is wise to do just the opposite)
- *use of documentation (and other) standards;* there are not so many international (or even national) software standards around, but every activity which increases their use is a step towards software industry which does not have to be ashamed of its unmaturity as a serious business sector
- *use of 'de-facto' standard system software;* it is very desirable to have workstations that are transferrable over hardware model generations and over hardware vendor restrictions
- *use of 'de-facto' standard hardware;* it is work-consuming not to have troubles with 'almost standard' system software and software tools.

We chose three main strategies for our development work:

- *commercial technology transfer;* we decided to find all possible tools and other components in the commercial marketplace according to the issues and principles mentioned above; if there were several applicaple tools, we had three choices:
  - -- choose one of them (usually by the uniformity issue)
  - -- choose many of them (and let the flexibility issue to decide)
  - -- development several parallel versions of the workstation and use one or some tools in one workstation and some other tools in others (usually according to software tool interfaces)
- *incremental development;* we decided to build at least three consecutive versions of each workstation in order to
  - -- study and learn things step by step
  - -- be able to change the most important choices (system software and central special tools) if necessary
  - -- give companies in our project support groups chance to influence our development decisions as often as desirable
- *intensive role of companies;* we decided to tie several major Finnish software-producing companies as closely as possible to our development work right from the beginning in order to
  - -- get financial support (which is a benefit for the continuing motivation in the companies except for our financies)
  - -- form an active group of contact persons (which is a benefit for getting workstations piloted in the companies except for getting comments all the time)
  - -- get piloting experiences of each workstation version.

Software project manager's workstation OHTO

We developed three consecutive versions of workstation OHTO for software project managers. In the following, the capabilities of the latest workstation called OHTO-3 are described.

OHTO-3 is a workstation that supports software project managers covering all project management activies:

- project goal-setting and estimation
- project planning and execution
- project monitoring and control
- project terminating and evaluation
- project administration.

Figure 2 presents the functions mentioned above and the relationships between them which are supported within OHTO-3.

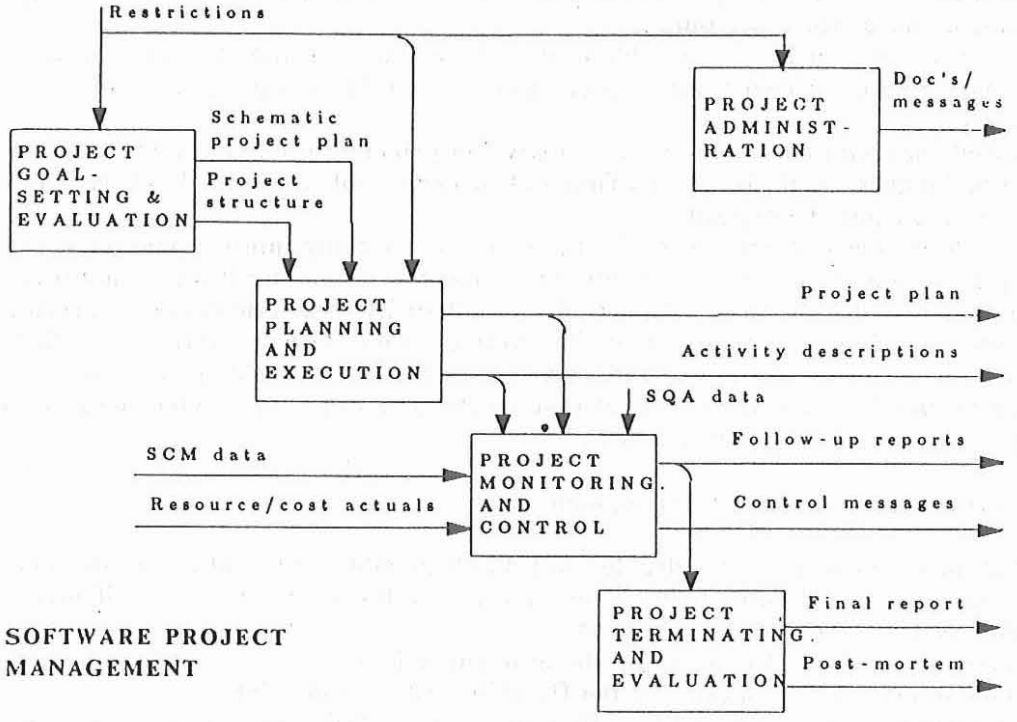

Figure 2. A function model of software project management

OHTO-3 runs on IBM PC/AT (or compatible) with 640 K memory, hard disk, EGA or Hercules graphics, mouse and laser or matrix printer. Its user interface is picture-oriented, menu-based and mouse-driven under MS Windows environment.

OHTO-3 has several groups of user interface screens. *Opening screen* gives the list of identifications of started projects; after one of them has been chosen, the full names of the project and its manager are given and it is possible to continue to the function screen.

Function screen gives the functional diagram of software project management activities (or sub-functions) presented in figure 2. For every function there is a *sub-function screen* covering several tasks. Every sub-function screen shows after every choice of tasks a *dialog box* with 'radio' buttons for further choices and there is a *help screen* for every screen and task.

There are five tasks under *'Goal-setting and estimation'*. Task 'Construct schematic project plan' means writing and composing a document which includes

- main goals of project
- overall structures of project and software
- estimates of resources, schedule and costs.

Schematic project plan may be used as a project idea, proposal or offer and it contains the results of three other tasks of goal-setting and estimation. We used Windows Write for text processing purposes in OHTO-3, because it is easy to transfer screens of other tools into Write text using the internal Clipboard facility of Windows and Windows Draw for diagramming purposes for the same reason.

Task 'Structure software and estimate size' includes constructing the overall software structure (programs, modules), estimating sizes of the software components and summing them up to get the size estimate of the whole system for cost estimation. It is also possible to use Function Point Analysis /4/ with some conversion data base for size estimates to perform a 'sanity check'. We built a MS Multiplan application for these purposes and developed a link which transfers software structure and size estimates to the software cost model WICOMO described below.

Task 'Estimate project' contains the estimation of resources, schedule and costs. We applied WI's WICOMO (based on Boehm's COCOMO /5/) in OHTO-3 by developing many enhancements and by transforming its phase/activity model for our purposes.

Task 'Structure project' includes dividing the overall project structure into phases and other activities like project management, quality assurance and configuration management. We developed a link between WICOMO and MS Project to get a Gantt chart based on WICOMO schedule estimates (using our own phase/activity model).

Task 'Assess project risks' covers assessing related risks based on McFarlan's ideas /6/. We used an expert system shell called VP-Expert for implementation.

Other sub-functions presented in figure 2 have similar tasks. In the following only the most important tools and standards used within those sub-functions are shortly described.

Within '*Project planning and execution*' we

- included software quality assurance plan and configuration management plans as appendices of the project plan using the corresponding ANSI/IEEE standards /7,8/ as base examples for plan formats
- built a resource allocation link between MS Project and MS Multiplan because of Project's insufficient resource allocation capabilities. Preliminary scheduling is done within Project, then preliminary resource allocation (persons - activities) within Multiplan and last fine-tuning of schedule and resource usage levels within Project
- used MS Multiplan for spreadsheeting and Windows Graph for presenting cumulative resource and cost curves, thus being able to use links between them and MS Project.

Within '*Project monitoring and control*' we developed in co-operation with a major Finnish company a resource monitoring system prototype using MS Multiplan and linked that system to OHTO-3 in order to produce required resource and cost follow-up reports.

Within '*Project terminating and evaluation*' the most important task is to update corresponding company history data bases. The risk-related data base was implemented using MS Multiplan having a link with the VP-Expert implementation of our risk assessment model. The project history data base was implemented using Rbase System V DBMS package using more than 200 different data elements. The data base was designed also in order to help the calibration of cost drivers of software cost model WICOMO described before.

Within '*Project administration*' we used Windows Cardfile and Calendar for the corresponding tasks.

OHTO-3 is primarily software project manager's workstation, but a software production manager can also use OHTO-3 e.g. to

- combine project plans or follow-up reports of multi-projects (combinations of several projects)
- check project estimates
- set schedule dependencies between projects
- allocate resources over different projects and
- combine monitoring information of different projects.

Software configuration manager's workstation SAMPO

We developed two consecutive versions of software configuration manager's workstation SAMPO. The latest version called SAMPO-2 is described below.

SAMPO-2 runs on Microvax II computer under Ultrix-32m operating system and uses Empress/32 as its (relational) data base management system (RDBMS).

SAMPO-2 is a multi-user workstation with three virtual desks as figure 3 shows.

A desk can be either on a 'dumb' terminal or on a PC (emulating terminal) or e.g. in the window emulating terminal on a PC.

The 'author's desk' must be used by every member of the project group when author wants to get objects accepted that he/she has 'created' or 'modified'. Objects can be either 'initial' (e.g. modules) or 'derived' (e.g. software packages). Author 'proposes' the object by sending it to the 'configuration manager's desk' who initiates the planned software quality assurance procedure and, after that, either 'rejects' or 'accepts' the object. 'Accept' puts the object to the 'archive desk' (which is also in configuration manager's responsibility). There are many other functions within SAMPO-2, e.g.

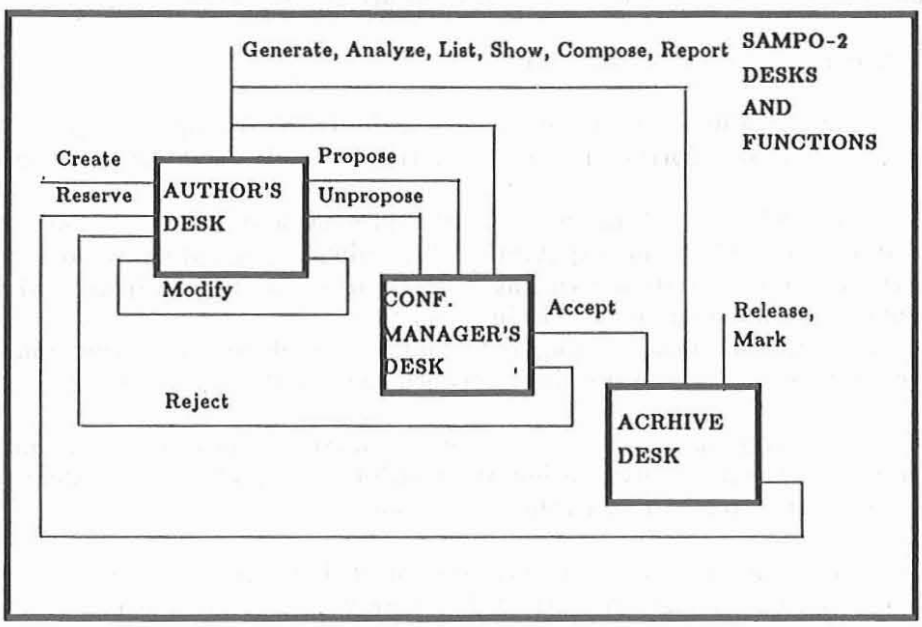

Figure 3. SAMPO-2 desks and functions

- author can 'reserve' objects so that nobody else cannot make any changes to that object
- author can 'unpropose' any object if he/she finds an error in the proposed object
- configuration manager can 'mark' an accepted object, which is erroneous, but cannot be fixed immediately
- objects can be 'analyzed' with several analyzers during specification phase (static analysis of completeness, traceability and concistency), design phase (SD-based IDES-method) and testing phase (concistency)
- it is possible to get reports about objects, their hierarchies and configurations (e.g. phase products).

Software developer's workstation ELINA

We developed two parallel versions of software developer's workstation ELINA

- ELINA-E for documentation of embedded software and
- ELINA -T for documentation of data processing applications.

ELINA runs - like OHTO-3 - on IBM PC/AT (or compatible) with 640 K memory, hard disk, EGA or Hercules graphics, mouse and matrix printer or laser. Its user interface is picture-oriented, menu-based and mouse-driven under MS Windows environment.

Its user interface is similar with OHTO-3. In figure 4 is the software development process implemented as an ELINA-E screen.

In the case of figure 4 'Software specification' is the chosen document. In the low left corner of the screen appears a dialog box with two sub-boxes: the other one is for 'text skeletons' and the other one is for 'diagram skeletons'. Text skeletons can be either

- 'tables of contents' (in this case there are two choices: a FIPS-standard and VTT's own specification standard) or
- older versions of the corresponding document.

We used most major software documentation standards, e.g. IEEE's, as text skeletons.

Diagram skeletons can be either

- 'symbol libraries' with the total set of symbols of a diagramming technique (in the case of figure 4 there is only one choice: IDEF- (SADT-) symbol library) or
- a diagramming sheet (in the case of figure 4 there are IDEF sheets both in Finnish and in English) or
- older versions of diagrams.

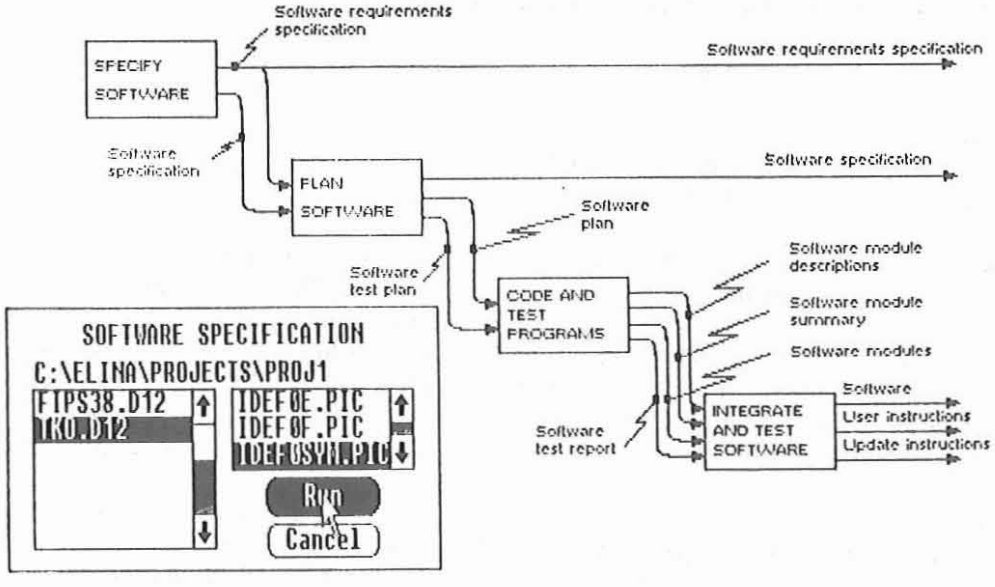

Figure 4. Software development process screen of ELINA-E workstation.

We have implemented symbol libraries for many well-known diagramming techniques, e.g. SA, SD and SADT.

Texts and diagrams can then be combined arbitrarily to produce the chosen document.

The other version ELINA-T for data processing applications was developed using a commercial Finnish information system development model (including e.g. Swedish-based system analysis and design method ISAC, conceptual analysis method ER/EAR and structured programming method JSP).

The present situation of OHTO, SAMPO and ELINA

All our workstation projects had one main goal: to produce a useful workstation prototype which could be further customized to useful workstations in Finnish companies.

We had ten organizations piloting OHTO, two organizations piloting SAMPO and four organizations piloting ELINA. Nevertheless, it is not sure whether any of those companies will use our workstations in the future. There are very obvious and acceptable reasons fot that situation which can be shortly summarized as follows:

- the user interface implementation of OHTO-3 using Windows Software Development Kit is too tedious to be customized for different purposes of different companies. We have to get reasonable tools for Windows (or Presentation Manager) user interface development.

- the DBMS implementation of SAMPO-2 is far too slow for production purposes. We have to get faster RDBMS packages.

- software documentation itself is not a sufficient topic for a software developer's workstation if there are no linkages to more genuine CASE tools. We have to get CASE interface standards to be used within open CASE tool sets like ELINA.

We feel that we have had many sound ideas before having the opportunities to implement them efficiently.

Experience gained around workstation development and piloting

During five years of our R&D we got important experience concerning

- organizing workstation projects
- choosing hardware and system software
- organizing piloting in companies.

Software production workstation projects are like any other software package projects in many ways. To start with, there has to be a very small group, which believes in what they are doing, does not listen too much and pushes their way through difficulties. It is not sensible to have a big group of people expressing their own views from one meeting to another especially if they work many hundred kilometres from each other. Later on, the group must listen very much and be very sensitive to realize the real needs and wishes of the users. And last but not least, we could see that to develop a product from a prototype must take an incredible amount of time, resources and money.

Choosing hardware and system software for workstations is not a very easy job. For example: we started SAMPO project with Altos micro, Xenix operating system and Informix RDBMS, then changed Altos to Intel. Then we went over to VAX 750 and VMS operating system with Eunice Unix-emulator, then changed Informix to Mistress. After that we used Convergent with Unix, then Microvax II with Ultrix and finally updated Mistress to Empress. And all this happened in three years! Concerning OHTO and ELINA we would have been lucky to choose Windows environment (because it is now de-facto standard of windowing environments) if we had had better tools for user interface implementation.

We noticed that piloting is very sensitive: if something goes wrong, it is a tough job to gain back the confidence of the piloting user, no matter whose fault it was in the first place. Anyway, our experiences concerning piloting were positive: we avoided many painful mistakes by listening the piloting users.

Our next workstation

We have already started our next project within the Finnish Research Program for Software Technology (FINSOFT) by the name 'Management, Communication and Collaboration within DIStributed PROjects producing Software (DISSPRO). The main goal is to produce a 'DIStributed Software Engineering Environment GENerator' (DISSGEN) which could be used to produce tailorable environments for software engineers working in software projects which are either geographically or organizationally distributed. The first prototype will be implemented on our Sun 3 network using a hypertext package for user interface implementation and collaboration support and Unix mail facilities for communication support.

We are aware of the danger of producing our prototypes once again a few years before they are implementable for production purposes. Anyway, compared with the other choices - producing academic papers or competing with software tool companies - it is our only possible way to a successful R&D result.

References

/1/ Känsälä, K., et. al., Analysis of software-producing companies (in Finnish). VTT Research Notes 253. Espoo, October 1983. 31 p. + app. 19 p.

/2/ Känsälä, K., Savola, J., Laitinen, T. & Hakkarainen, K., Software production management and software cost estimation models - a survey (in Finnish). VTT, Laboratory for Information Processing, research paper nr 4. Helsinki, October 1985. 30 p. + 87 p.

/3/ Känsälä, K., Software project management. VTT Research Notes 920. Espoo, 1988. 21 p.+app.3 p.

/4/ Albrecht, A.J., Measuring application development productivity. In: Proc. Joint SHARE/GUIDE/IBM Application Development Symposium, Oct. 1979. Pp. 83 - 92.

/5/ Boehm, B.W., Software Engineering Economics. Prentice-Hall, Englewood Cliffs, 1981. 767 p.

/6/ McFarlan, F. W. Portfolio approach to information systems. Harward Business Review (1981) September-October. P. 142 - 150.

/7/ ANSI/IEEE Std 730-1984. Standard for Software Quality Assurance Plans. IEEE, 1984. 12 p.

/8/ ANSI/IEEE Std 828-1983. Standard for Software Configuration Management Plans. IEEE, 1983. 11 p.