Improving Software Product Management Process: Implementation of a Product Support System

Tapani Kilpi
Department of Information Processing Science, the University of Oulu
Linnanmaa 90570 Oulu, FINLAND
email: tapani.kilpi@oulu.fi

Abstract

Software industry has developed strongly towards becoming a product business in the recent years. Companies have increasing difficulties in managing all the delivered "old" product versions and the development of the "new" product versions efficiently at the same time. Software Configuration Management (SCM) provides some capable means for managing these problems, but it has also some limitations. Software Product Management (SPM) is an approach developed for modelling the software change process from the point of view of a software product.

1. Introduction

A Software Product Management (SPM) process is an activity that covers both the development of "new" products as well as the management of all the "old" products delivered to customers. The SPM-process is expected to support the development of the new products by organising the collection and the analysis of the customers' feedback data as well as the product support activity for solving customers' problems in the first place. On the one hand, these features enhance the possibilities to produce better products in future, and on the other hand, save the development unit's time when acting as a buffer between the customers and the development unit. Managing of the "old" products means on a practical level that all the customer deliveries need to be able to reproduce in their original building environments in order to be able to serve the users the former product versions. The theoretical background of SPM lies strongly in the ideas presented in the Software Configuration Management (SCM) approach. SCM is a discipline designed for managing the software change process. The idea behind SPM could be described with the same definition on a general level. SCM provides advanced models and tools for organising the management and the rebuilding of the "old" product versions. However, the recent SCM theories do not offer clear ideas and means for organising the collection and the analysis of the customers' feedback data and the customer's product support activity which are both of essential meaning for the software change process.

The need for SPM has increased rapidly in the software industry during the last years because numerous companies have modified their production towards product orientation. In many occasions designing of the SPM-process and its implementation have been problematic because of missing SPM-models and -tools. SPM has to be able to recognise and handle a software product which is defined to consist of the software, its support service and the idea behind it. This paper presents the ideas and observations discovered and made in the TPM-project (Towards Total Product Management, ESSI-project 21336). The TPM-project is run by three companies of Technopolis Oulu: Oy Quality Production & Research Ltd (QPR), Modera Point Oy and Prosoft Oy. The objective of the project is to improve the SPM-processes of the participating companies. The main-phases of the project have been defined as: 1) original status analysis, 2) definition of SPM-process model and TPM-roadmap for describing the practical level steps of improvement, 3) selection of the SCM-tools and implementation of the SCM-process, 4) selection of the SPM-level tools and the implementation of the SPM-process and finally 5), final evaluation of the progress made during the project. This paper concentrates on presenting the implementation of the SPM-process and the ideas behind its designing.

2. From SCM-discipline to SPM-practice

Software Configuration Management (SCM) is a discipline developed for managing the software change process. The first generation SCM-tools like, for example, SCCS [19], Make [6] and RCS [21] provided facilities for keeping track, building and re-building the different versions of software products. Introducing these tools was a great improvement step in managing the software change process. The amount of duplicate work, data losses, person dependency and other such things used to be seen as problems of change management were managed to decrease remarkably much. The first generation of SCM-tools managed to provide relatively capable means for technical handling of the components of different software versions. However, there were no general agreements of what SCM is and what it should cover which caused plenty of confusion and extra work in organisations.

At the close of the 80's and in the beginning of the 90's the general standards for SCM were created [8, 9, 10], and the functionality requirements of the SCM-tools widely discussed and defined [3, 4]. This made it possible to design more capable SCM-tools supporting the natural change process in the organisations. For a while it has been noticed that the technical facilities do not alone provide the required support for the software change process if the process is not defined, and if the defined process is not followed and understood in an organisation [7]. The process orientation became one of the main objectives of the SCM-development [5]. However, the changing image of the software industry required more capable systems and the slowly grown understanding of the potential of SCM made it possible to design new type of solutions. Leblanc (1994) describes the SCM-challenge of today by the need to correspond to i.e. *aggressive schedules, increasing customer demands, market pressure, customer support, requirements of different platforms, absolute accountability, productivity demands, requirements of the overall development process,* and *requirements for distribution.*

The most advanced new SCM-tools contain plenty of functionality satisfying the process requirements. These tools normally support some specific type of process model. It is also believed that the market for configuration management software is expected to grow rapidly in near future [20]. Nevertheless, each organisation has the problems of its own, and one solution is not right for everyone. Especially small companies may find it difficult to find ready solutions for their process support problems. These days more and more small companies base their business on serial products, i.e. the same product is sold to numerous customers. Keeping account of the numerous customised product versions, solving problems in connection to them and making the right development decisions are always complex tasks. In small companies it is sometimes hard to "protect" the development key-persons from, for example, customer support activities because everybody need to do everything. This may endanger the effectiveness of development and the whole business. For these reasons small companies clearly have some product management needs of their own not included in the recent SCM-solutions.

The answer to product management problem is a well tailored process model for software product management. In practice this means that the recent understanding of the SCM-process model need to be extended to cover also such areas like delivery and marketing that are vital parts of the business and sometimes easily forgotten in small companies, i.e. these phases are not paid enough attention to which causes problems. General product management theories [14, 15, 17] defined to be applied across the industries offer means for modelling the whole product management process. When the process model notices the analysis of the market and the customers' needs as well as the delivery phase, there will finally be remarkably less need for customer support which saves plenty of resources.

3. Software Product Management Process

The underlying theory of this paper relies on the product oriented process model for SCM (Software Configuration Management) defined by Kilpi (1997b). The model is called here the SPM-process model (Software Product Management process model). In software industry the requirement for changing products is caused by the needs of the market and the customers. The task of a software company is to satisfy the needs of their customers, and to be able to correspond to the changing demands on the market. In the SPM-process model presented here the function of a company is modelled with four activity areas of product management [17] which are: the *Development,* the *Production,* the *Marketing* and *the Delivery* activities.

The four activities have been designed to work together and to complete each other in order to reach the company objectives. The *Delivery* activity manages the customer deliveries. The *Marketing* activity collects and analyses the market and the customer information, as well as informs the market and the customers of the products and the new releases of them. The *Production* activity provides the support facilities for the customers' problems, taking care of handling the orders and the deliveries. The *Development* activity manages the product change process by analysing all the feedback and the development ideas collected, and by planning the release projects. The SPM-process model is illustrated in figure 1. The model consists of six main processes: the *Customer Delivery*, the *Marketing & Sales*, the *Product Support*, the *Software Production*

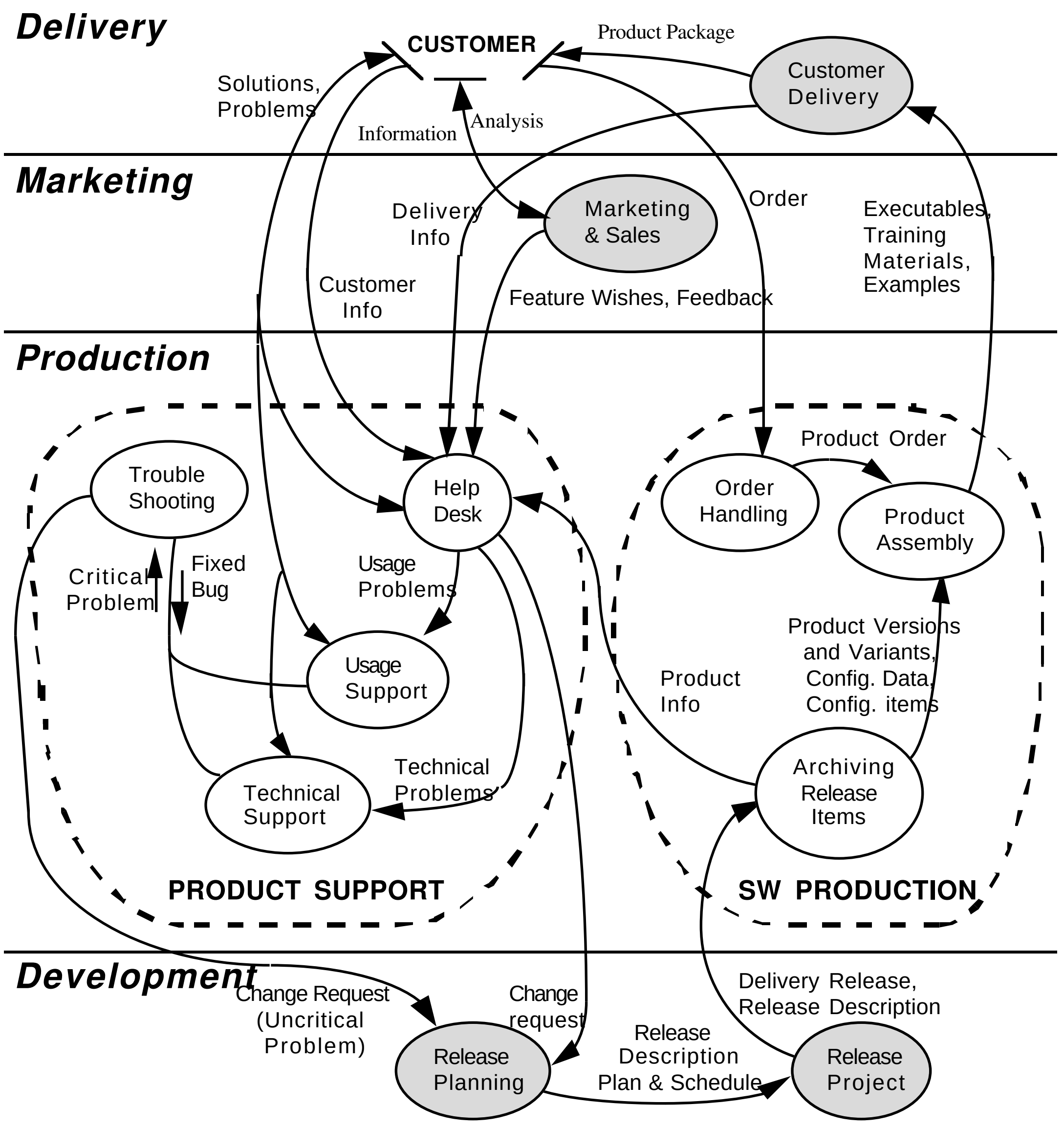


Figure 1. Software Product Management Process

(SW Production), the *Release Planning,* and the *Release Project.*

In figure 1. the *Product Support* and *SW Production* processes have been presented on a more detailed level than the other four main processes of Software Product Management (SPM). This is because of the objective of this paper which concentrate on presenting the theoretical background as well as the practical level means of implementing especially this part of the SPM-process model. The *Marketing & Sales* process collects the Market Information and the Customer Feedback, and produces Feature Wishes basing on the results of the information and the feedback analysis. The Feature Wishes are then passed to the *Product Support* process where they are classified and saved together with the Customer Problem data in the Change Request format in the Change Request database. Solutions to the customers' problems are normally provided by the *Product Support* process, and sent to the customers whereas the unsolved ones are saved as Change Requests.

The general level Product Strategy is defined as a part of the *Marketing & Sales* process, and the Release Date Plan basing on the Strategy is delivered to the *Release Planning* process. The *Release Planning* process starts to plan a new release basing on the Release Date Schedule by analysing all the collected Change Requests. As a result of the

Release Planning process detailed Release Description, Plan and Schedule are delivered to the *Release Project* process for starting the development work. The *Release Planning* process also sends Release Info of the next release to the *Marketing & Sales* process. The *Release Project* process produces a new release of a product corresponding to the Release Requirements. The accepted new Releases are delivered then as delivery Release Versions to the *SW Production* process for archiving and customer delivery management. The deliveries base on the Customer Orders which are passed to the *Customer Delivery* process normally through the *SW Production* process. In a delivery a customer gets a Delivery Package containing all the product components, i.e. the manuals and the examples. The *SW Production* process also sends Product Information to *Marketing & Sales* process for a marketing use.

The Production activity of the process model consists of two main processes: the *Production Support* and the *SW Production* processes. These two process descriptions also define the functionality of the Product Support system presented in this paper. The two processes take care of the handling of Customer Orders, Feedback, Change Requests [4] and Deliveries. In TPM *the Change Request is defined as a documented request for a change in a specified format.* The general purpose of the *Production Support* and the *SW Production* processes is to be a buffer between the Development team and the customers. If the customers contact straight the Development team in problem situations, they will soon "steal" all the valuable working time of the developers. A danger of doing product development without charging customers will be obvious if the *Product Support* process is not well designed. In order to protect the development work, and to manage the vital software change process successfully, it is very important to learn to relate the Change Request concept to the SCM-process correctly. Up to Bersoff and Davis (1991) the meaning of the Change Request concept increases along with the volume of the production and the development needs.

The *Product Support* and the *SW Production* (Software Production) processes of the SPM-process model have many functions on a more detailed level. The purpose of the *Product Support* process is to develop answers and solutions to the questions and the problems of the customers, and also to model and save all the bug and development data in the Change Request format in the Change Request database. All the questions and the problems of the customers are first pre-analysed in the *Help Desk* process. The *Help Desk* process analyses and classifies the Customers' problems as well as the Feature Wishes and the Feedback from the *Marketing & Sales* process, in order to modify the contacts of both these information types to the form of Customer Change Requests. Help Desk process also classifies the Customer Problems and passes them depending on their nature to the *Usage Support* or the *Technical Support* processes with the Customer and Request history related to the problem. Most of the problems are solved in the *Usage Support* and *Technical Support* processes, and the solutions are informed to the customers immediately. The Critical Problems unsolved by the *Usage Support* and the *Technical support* are passed to the *Trouble Shooting* process.

The *Trouble Shooting* process solves the critical problems in one way or another delivering the solution back to the Support processes to be passed on to the customers. The Problems classified to be uncritical in the *Trouble Shooting* process are analysed and handled later in the *Development activity.* The purpose of the *SW Production* process is to handle the Customer Orders, and to pass the ordered products to the *Customer Delivery* process. The *SW Product* process gets the delivery items and data from the Release Project process, saving them in the databases. The archiving of the items and the data are done by the Archiving Release Items process. The database contains all the necessary source file components and document files of the released software products. The database contains the executables of all the variants and versions of all products as well as the configuration data for rebuilding a product of the items. The components of a product package, e.g. the software, the manuals and the examples are collected together and checked in the *Product Assembly* process and then passed on to the *Customer Delivery* process.

4. Example Case of a Software Product Management Process Improvement Experiment: the TPM Project

The process model of Software Product Management presented in the previous section has been applied in the TPM-project (Towards Total Product Management in Technopolis Oulu). The objective of the TPM-project is to increase the maturity level of SPM (Software Product Management, ESSI-project 21336) processes in the three companies participating the TPM-project by defining and implementing an SPM-process model for the companies. The original status of the SPM-processes in the TPM-companies was evaluated and analysed [11] by using Pr^{2}imer method [18] developed by VTT Electronics of Finland. The question series of Trillium method [22] were used in combination with Pr^{2}imer for the evaluation part. Basing on the results of the original status analysis and the methods [18, 22] used in it, a Roadmap was created in

Table 1. The Roadmap for Improving the Maturity of SPM in the TPM-companies

The Original Level	
Configuration Items	• manual archiving using ZIP or XCOPY • source codes of main delivery versions are stored • products have their own archives
Product Data	• lists of product component descriptions • change information documented manually • batch files used in builds • releases are produced automatically in version controlled code
Delivery Data	• computerised delivery register • order handling and invoicing systems
Customer Data	• computerised customer registers
Change Request Data	• manual bug lists and bug databases
The Minimum Objective Level	
Configuration Items	• all source revisions can be retained and are accessible • archive organised into components • components designed to be reused within one baseline project • advanced version control concepts in use (version labels, promotion groups)
Product Data	• products build using make • make file under version control • all releases and test versions are produced automatically of version controlled code
Delivery Data	• delivery database • delivery data contains product specific information
Customer Data	• customer database contains link to delivery database • all customer contacts are recorded • change (service) request concept is formalised
Change Request Data	• bug identification system • change requests linked to source code files • change requests are reported and linked directly to a specific product and version
The Ultimate Objective Level	
Configuration Items	• all documentation in a version controlled database • all versions of build tools retained • components reusable across several products • the whole product is configured automatically
Product Data	• make file dependencies reach all the way to version control • automated dependency generation • build environment configuration in make file • build record database (which source file revisions were used in each build)
Delivery Data	• all customer deliveries can be traced down to source code files • delivered product can be completely rebuilt later • information of customer's environment is stored
Customer Data	• change request contains links to all source, design, and product management files involved
Change Request Data	• change request classification system in use with feedback mechanism for inspection, testing, and coding • change request reports are used to optimise the software process

order to show the way for the practical level actions to be included inthe TPM project plan.

The Roadmap, illustrated in table 1, presents descriptions of three maturity levels defined to present the status of the SPM in the TPM-companies at different phases of the Process Improvement Experiment (PIE): *the Original level, the Minimum Objective level and the Ultimate Objective level.* . The *Original Level* is a description of the SPM process status in the TPM-companies in the beginning of the project, the Minimum Objective Level describes the level that is to be reached during the project, and the Ultimate Objective Level is the level that is to be reached in the near future. Each of the maturity levels contains a description of management mechanisms of five product management related data flows which are named here up to Auer (1996): *Configuration Items, Product Data, Delivery Data, Customer Data and Change Request Data.*

The three TPM-companies are all quite small-sized, and locate in Technopolis Oulu. **QPR** (Oy Quality Production & Research Ltd) specialises in developing, manufacturing and marketing of software tools on a business management area. At the moment there are about 30 employees in QPR. The technical environment of QPR is based on a Novell network. The workstations are PC-based, using Windows95 and WindowsNT. The main tools in development are Borland Delphi and Visual Basic. **Modera** (Modera Point Oy) specialises in systems developed for waterworks, peat suppliers and telecommunications. The current number of employees is 10. The technical environment of Modera is based on a Novell network. The workstations are PC-based, using Windows. Internet-addresses are provided for almost all employees. **Prosoft** (Prosoft Oy) has specialised in developing, manufacturing and marketing the testing tools for embedded software. The number of employees is 5. The technical environment of Prosoft is based on TCP/IP network with Sun server. The workstations are PC-based using Windows95, WindowsNT and OS/2. Internet-addresses are provided for almost all employees.

The selections of the SCM-tool and the tool suitable for implementing the aimed product management features on the SCM-system have perhaps been the most critical tasks of the project. These days the selection of potential SCM-tools is numerous on the market, which causes some extra work to the selectors in getting familiar with many tools and comparing them with each other. The conclusion of the TPM-tool evaluation team was to recommend the selection of PVCS-tool for supporting the SCM-processes in the TPM-companies. Other serious candidates in the selection were *Continuus/CM, ClearCase, ExcoConf and Visual SourceSafe.* The selection process is described in detail by Kilpi (1997a).

The SCM-implementations made have satisfied the requirements presented in the TPM-roadmap for the Minimum Objective level of the *Configuration Items* and the *Product data.* Another major achievement of the project has been the designing and the implementation of a remarkable part of the product management functionality defined in the SPM-process model presented earlier in this paper. This part of the TPM-project is described in the next section.

5. Experiences of Implementing a Product Support System in the TPM-project

Some product management functionality and features needed to be added to the created SCM-facilities in order to reach the TPM-roadmap Minimum Objective level completed in all the participating three companies. The requirements for the product management part have been defined in the columns *Delivery data, Customer data* and *Change request data*, defined in the TPM-roadmap. In order to create capabilities for satisfying these requirements a separate Product Support system was found necessary to create in the TPM-project. The creation of the Product Support system was carried out in the sub-project of the TPM-project named ECHO. The TPM/ECHO Product Support system was built by using Lotus Notes development tool. The selection of Lotus Notes for TPM was made basing on the results of several benchmarking events to other companies that proved Lotus Notes to be a capable tool for creating a product management support system and also compatible to be used together with PVCS. The technical environment of the system consists of the LANs (Local Area Network) of the companies with Notes server, work stations used as Notes clients and a printer for printing reports. The system has also connection to WWW (World Wide Web). The databases of the Product Support system locate and are updated on the server. The functional principles of the TPM/ECHO Product support system is presented on a general level in the figure 2.

The main purpose of the Product Support system is on the one hand to support to solve the customers' problems, and on the other hand, to collect the feedback data concerning the company products. A basic element of the Product Support system functionality is one problem solving cycle, a workflow. A workflow starts when the Helpdesk-process receives a Customer's service request. The

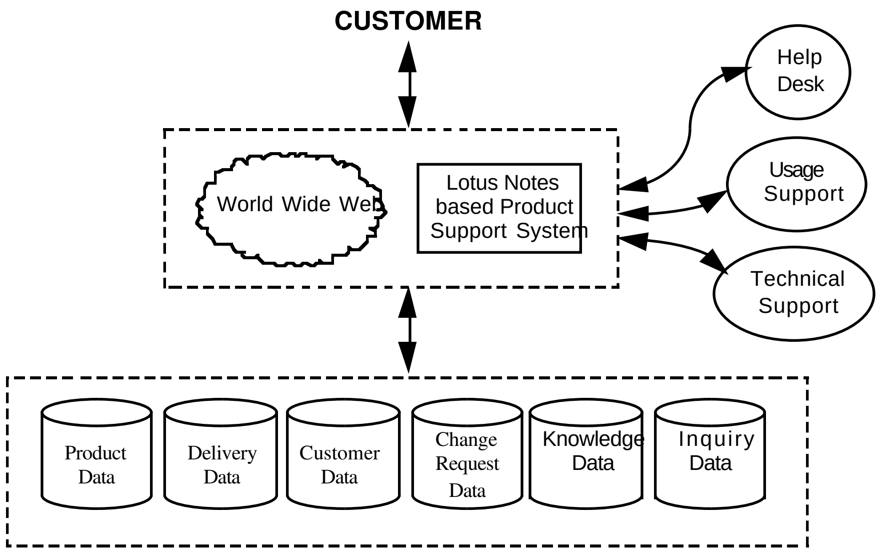


Figure 2. Functionality of the TPM/ECHO Product Support System

customer can deliver a request by using a phone, an email or a WWW-form. The request is saved in the Change request database either by a company employee or by the customer himself. If the customer sends a request straight to the database through WWW, the Helpdesk-process gets a notification of this automatically through email. A solution to the customer's problem is tried to find in the Helpdesk-process by using the saved knowledge, product, customer and delivery information as help. The customer is notified of the solution through email or by telephone. In the case that a solution is not found, or the problem situation needs further evaluation the Helpdesk-process saves a Knowledge entry of the problem. These entries are solved later in the Technical Support, Usage Support and Release Planning processes.

The Lotus Notes based TPM/ECHO Product Support system is introduced in more detail in the following sections. The system bases on the use of menus and views. The main menu provides access to the different views of all the databases: Requests, Knowledge, Deliveries, Products and Customers. The following sections present the parts of the TPM/ECHO product support system which are of central meaning in order to satisfy the Minimum Objective Level requirements of *Delivery data, Customer data* and *Change request data* presented in the TPM-roadmap.

5.1. Delivery data

The basic unit of Delivery data is the information accompanied with one delivered product. As a concept the *Delivery* is wider than the *Product.* In addition to the *Product data Delivery data* contains information of customers and delivery dates. This additional knowledge makes each delivery a unique action. Archived *Delivery data* makes it possible to track the specific product versions that each of the customers have afterwards and also the customisations included in each of the delivery packages. The TPM-roadmap, presented in section 3, defines the following objectives for the Minimum Objective Level of *Delivery data*:

- Delivery database
- Delivery data contains product specific information

In practice the definition of the Minimum Objective Level of the *Delivery data* means that archives for delivery specific and product specific information, *the Delivery database and the Product database*, need to be included in the Product Support system. A possibility to create links between these two databases is also needed in order to be able to combine the *Product data* with the *Delivery data* entries.

The TPM/ECHO-system satisfies both the demands which are presented for the Minimum Objective Level of *Delivery data* in this section. The system recognises the concept of Delivery, and provides a possibility to save information of an individual delivery in a *Delivery database.* There is also a possibility to create links between the Delivery database and the Product database. A link can be created by selecting the delivered product straight from a key word list which contains all the product names. This selection automatically adds the attributes *of Product name, Product* *version number* and *Product serial number* to the *Delivery data* entry of a delivery. The information of the Customer consists of one attribute, the *Customer name*. The customer name is selected also from a key word list containing the names of all the customers. In addition to the four attributes presented here a Delivery data entry contains also *Delivery date* attribute which is automatically generated to the Delivery data entry when adding a new entry to a database. The recent implementation of TPM/ECHO-system covers also some features of the *Ultimate Objective Level* of the TPM-roadmap. The feature of *"Delivered product can be completely rebuilt later"* is provided by the renewed SCM-process and -tools implemented to the TPM-companies during an earlier phase of the TPM-project. The future objectives are to add the information of the customer technical environment to the *Delivery data* and to add the feature of using the links that combine a delivery with the source code files corresponding to the delivered product.

5.2. Customer data

The basic unit of *Customer data* is the information accompanied with one customer. In addition to the customer identification, name, address and contact information this means also the information of the interactions, contacts and communication with the customer. Along with identifying and contacting the customers the archived *Customer data* makes it possible to track and remember all the necessary customer contacts and service requests. This enhances the collection of customer feedback information, the customer problem solving process and the treatment of the customer relationships. The TPM-roadmap, presented in section 3, defines the following objectives for the Minimum Objective Level of *Customer data*:

- customers' database contains link to delivery database
- all customer contacts are recorded
- change (service) request concept is formalised

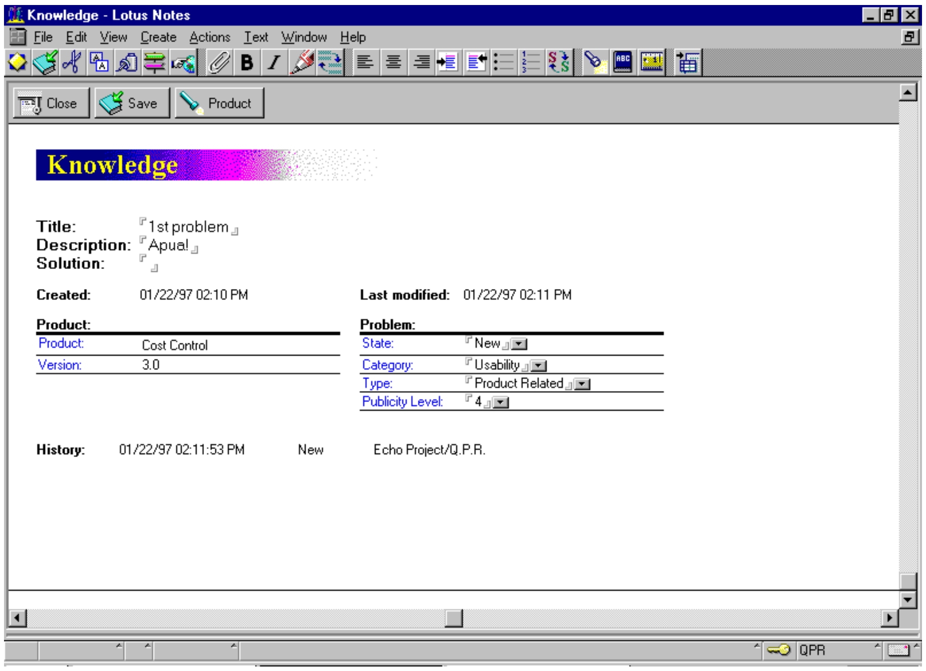
In practice the definition of the Minimum Objective Level of the *Customer data* means that the Product Support system needs to be included in archives for customer specific and delivery specific information, *the Customer database and the Delivery database*. A possibility to create links between these two databases is also needed in order to be able to combine the *Customer data* with the *Delivery data* entries and vice versa. Archiving of the customer contacts requires the formalisation of the concept of customer contact, defined in the TPM/ECHO-specification the *Customer's Service Request,* and an archive for saving the requests, *the Request database*. The TPM/ECHO-system satisfies all the three demands presented above with the exception that at the moment the customer contacts can be saved and handled only in the Request format. The system recognises the formalised concept of Change (Service) request, and provides a possibility to save information of an individual service request situation in a *Request database.* There is also a possibility to create links between the Customer database and the *Delivery database.* A link can be created by selecting the customer straight from a key word list containing all the customer names and a connection to the *Customer database*. A *Customer's Service Request* entry contains the following data as the attributes: Request number, Title, Description, Dates of creation and modification, Priority, Time limit, Status, Status date (when defined), Status description, Delivery (from the *Delivery database*), Contact (person), Solutions (from the *Knowledge database*) and Updated (date of the last update). Normally the request forms are filled by an employee of the company basing on a telephone call, but a customer also has a possibility to fill and send the request forms through the WWW. In order to reach the Ultimate Objective Level defined in the TPM-roadmap the future objective is to add the links that combine a request with the source code, design and product management files.

5.3. Change request data

Change request data concerns the handling and the analysation of the *Customer's service requests*. The purpose of collecting and saving this type of information is to enhance the possibilities to fix the identified bugs, and also to avoid duplicate work in solving the problem situations. In the long term the analysation of the Change request data enhances the improvement of the whole product support and development processes. The TPM-roadmap (section 3) defines the following objectives for the Minimum Objective Level of *Change request data*:

- bug identification system
- change requests linked to source code files
- change requests are reported and linked directly to a specific product and version

In practice the definition of the Minimum Objective Level of the *Change request data* means that an archive for saving the descriptions of the problems with their solutions as well as the bugs and the bug fixes made, the *Knowledge database* needs to be included to the Product Support. A mechanism for using links that connect the Customer's service request (presented in the previous section) entries to the corresponding source code files is also needed as well as the links combining together the *Service request*



Title:	1st problem
Description:	Apua!
Solution:	

Created:	01/22/97 02:10 PM	Last modified:	01/22/97 02:11 PM

Product:		Problem:	
Product:	Cost Control	State:	New
Version:	3.0	Category:	Usability
		Type:	Product Related
		Publicity Level:	4

History:	01/22/97 02:11:53 PM	New	Echo Project/Q.P.R.

Figure 3. TPM/ECHO Knowledge Form for Identifying Bugs

database and the *Product database*The TPM/ECHO Knowledge form is illustrated in figure 3.

The TPM/ECHO-system satisfies two of the three demands presented above. The system recognises and uses the concept of Knowledge entry for saving problem and bug data with a solution and fixing descriptions. The system also contains the possibilities to create links between *Service request database* and *Product database*. At the moment there is no possibility to combine Customer's Service request straight with the source code files of the corresponding product. This feature will be implemented to the TPM/ECHO Product Support system when the integration of the SCM-system of TPM and the Product Support system has been developed onto a more advantaged level. The Knowledge entry contains the attributes: Title, Description (problem), Solution, Dates of creation and last modification, Product (from the Product database), Problem status (keyword-list), Problem category (keyword-list), Problem type (keyword-list, technical/other), Publicity level (is the document allowed to publish in the WWW, for example) and History (of a document).

6. Conclusion

The experiences of the TPM-project ("Towards Total Product Management in Technopolis Oulu", ESSI project Nr 21336), clearly underline some meaningful characteristics necessary to notice when improving the SPM-process of an organisation. The most important requirement is to be able to model the SPM-process in a realistic way. The objective is, on the one hand, to provide facilities for managing and rebuilding all the former product versions delivered to customers, and on the other hand, to be able to offer a capable product support service for solving the customers' problems and at the same time collecting some valuable feedback data to be analysed later and used for designing the future product versions. By implementing these facilities in the TPM-project the software product management maturity levels were managed to increase in the companies. The practical level benefits of the improved maturity can be outlined as follows:

- *saving of time*
- *improved efficiency*
- *faster customer service*
- *easier planning of new version releases*

The TPM experiment also taught some lessons of how an SPM development approach should be organised in an organisation. It was also figured out that a tool alone is never a complete solution. There are plenty of advanced models and tools for designing and implementing the SCM-process. However, this is not the case with SPM. The organisation willing to improve its SPM-process need to be ready to create an SPM process model of its own for the development purposes. There do not either exist ready and working tools for managing the SPM-process, and in practice separate software systems have to be created for this purpose. This can be done with some of the existing tools suitable for this purpose (i.e. Lotus Notes). If the objective is to develop the SPM-process into an advanced level the, "SCM-part" and the "SPM-part" of the complete SPM-system need to be integrated tightly. This is not easy because it is difficult to create mechanisms, for example, for tracking source code files straight from design and bug report documents. However, the SPM-process can be developed onto a quality level even without the integration. Consequently, it is possible to reach meaningful results already by a reasonable amount of resources used for the process improvement activity.

References

[1] Auer, A. 1996. Seminar Presentation Material of the Final Presentation of LEIVO-project 14.2.1996 in Helsinki. Organised by VTT Electronics.

[2] Bersoff, E. H. & Davis, A. M. 1991. Impacts of Life Cycle Models on software Configuration Management. Communications of ACM. August Vol 34 No 8.

[3] Dart, S. 1990. Spectrum of Functionality in Configuration Management Systems. Technical Report. CMU/SEI-90-TR-11. ESD-90-TR-212.

[4] Dart, S. 1991. Concepts in Configuration Management Systems. Proceedings of the 3rd International Workshop on Software Configuration Management, June 1991, pp. 1 - 18.

[5] Estublier, J. 1995. Process Session Introduction. In the proceedings of SCM-4 and SCM-5 workshops, Lecture Notes in Computer Science 1005, Springer, pp. 136-137.

[6] Feldman, S. 1979. Make- a program for maintaining computer programs. Software-Practice and Experience, 9(4):255-265, April 1979.

[7] Humphrey, W. S. 1988. Characterizing the Software Process. IEEE Software 5(2):73-79, March 1988.

[8] IEEE ANSI standard. 1987. IEEE Guide for Software Configuration Management. IEEE/ANSI Standard 1042-1987. IEEE Press.

[9] IEEE ANSI standard. 1990a. IEEE Standard for Software Configuration Management Plans. IEEE/ANSI Standard 828-1990. IEEE Press.

[10] IEEE ANSI standard. 1990b. IEEE Standard glossary of software engineering terminology. IEEE/ANSI Standard 729-1983. IEEE Press.

[11] Kilpi, T. 1996. Evaluating the Maturity of Software Product Management: A Case Study in Three Companies. Proceedings of the Ninth Australian Software Engineering Conference ASWEC '96 July 14 to July 18, 1996. IREE Society.

[12] Kilpi, T. 1997a. Choosing a VC/CM-tool: a Framework and Evaluation. Proceedings of the 8th Conference on Software Engineering Environments SEE '97 8 - 9 April, 1997.

[13] Kilpi, T. 1997b. Product Management Requirements for SCM Discipline. Proceedings of the 7th International Workshop on Software Configuration Management SCM7 18 - 19 May, 1997.

[14] Kotler, P. 1997. Marketing Management: Analysis, Planning, Implementation, and Control. Prentice Hall, Upper Saddle River, New Jersey 07458. pp. 1-107.

[15] Kuczmarski, T. D. 1992. Managing New Products The Power of Innovation. Prentice Hall, Englewood Cliffs, New Jersey 07632.

[16] Leblanc, D. 1994. The CM Challenge: Configuration Management that Works. In Tichy W. F: Configuration Management, pp. 1-38. John Wiley&Sons, England.

[17] Pine, J. B. 1993. Mass Customization The New Frontier in Business Competition. Harvard Business School Press, Boston, Massachusetts.

[18] Pr^{2}imer.1997.http: // www.ele.vtt.fi / projects/primer/ primer.htm. VTT Electronics/Technical Research Centre of Finland.

[19] Rochind, M. 1975. The source code control system. IEEE Transactions on Software Engineering, pp. 364-370, December 1975.

[20] Thompson, G. 1995. Ch-ch-ch-changes. HP Professionl, October.

[21] Tichy, W. 1985. RCS: A System for Version Control Software: Practice and Experience, 15(7):637-654, July 1985.

[22] Trillium. 1994. Trillium: Model for Telecom Product Development & Support Process Capability. Bell/Canada. Release 3.0, December.