Computer based support in software production environments

COMPUTER BASED SUPPORT IN SOFTWARE PRODUCTION ENVIRONMENTS

ABSTRACT

Computer based support in software production environments (SPEs) is an area of increasing interest. Its main focus is on improving productivity and quality of software production by means of computer based support technologies. However, no universal set of these tools exist. It is a major problem for software producing organizations to select computer based support technologies effectively. In this paper, a customer oriented classification of production environments, taken from industrial production research is adapted to software industry. Based on the characteristics of different types of software production environments, guidelines are derived for the selection of computer based support technologies.

1. INTRODUCTION

Software industry evolved in the late 80s with the emergence of Software Development Environments (SDE), Software Engineering Environments (SEE) and Software Factories (SF), [DaEF 87], [IEE 88], [Tay 88], [Per 88], [FeO 89]. These integrated software production systems, which are to a high degree computer supported, are also denoted by the term Software Production Environment (SPE) [TaTF 90]. SPEs are aimed at a more industrial production of software [FeO 89], [Gen 91]. They cover the total support of the software production process of an organization, by integrating computer based support technologies for both technical and managerial production activities. As such, the production of software is no longer limited to a craftsmanship engineering of single products. With a properly designed SPE, organizations have the opportunity to produce professional software products, satisfying high quality standards in due time, at acceptable costs and in accordance with the varying demands of their customers.

Main components of SPEs are computer based support technologies (CBST). These technologies cover different aspects of the software production process, such as product design, management and control of the software process itself as well as reuse of previous work.

Considering the market situation, there is currently a substantial reservoir of these technologies. CBST are classified as CASE-tools (Computer Aided System Engineering tools), I-CASE-tools (Integrated CASE tools), IPSE-tools (Integrated Project Support Environment tools), Process modelling tools, Repositories and Object Management Systems, etcetera. Apparently, CBST are strongly fragmented and they differ significantly with respect to functionality and degree of integration [NoN 89]. Each of these technologies pretends to reduce repetitive and time consuming tasks and, implicitly, to increase productivity and quality. However, in previous research it was shown that by an arbitrary application of CBST these advantages cannot be gained (e.g.: [ChR 88], [TrvR 90]). Due to the absence of a single monolithic general purpose set of CBST for all software production situations, careful selection and integration is of utmost importance to ensure that CBST provide the required support. However, existing approaches for selection and evaluation have several weaknesses. They are performed in too much detail [BaS 89], they compare only some selected CBST from a particular class (e.g.: [WeHB 87] considers only CASE-tools), or they are based on common and too generally formulated requirements of the average software producing organisation [Zuc 89]. In current literature

no selection approaches can be found that are explicitly based on well defined software production characteristics of the software producing organization itself.

In industrial production research much effort has already been spent on the development of selection guidelines for computer based technologies [BeWW 90]. From this research we know that selection of suitable technologies has to be based on a proper design of the production environment. Further we know that there is no 'one best way' to design production environments. In industrial production it is commonly accepted that design guidelines have to be derived from the customer orientation characteristics of the organization [Sar 81]. Since customer orientation is a major topic in the software industry we will in this paper adapt the customer orientation characteristics from industrial organizations to software producing organizations. Consequently we will use them to derive guidelines for the design of effective SPEs and the selection of appropriate computer based support technologies.

2. LESSONS LEARNED FROM INDUSTRY

In industrial production, the customer orientation is characterized by the extent to which the production processes are governed by the customer orders [Sar 81]. We call this the customer dependency of the production process. Traditionally, two types of dependency could be distinguished. Customers demanding fast delivery were supplied with standardized products and customers with specific demands were presented with products designed to order. If a customer demanded specific products to be delivered with a short leadtime, a problem was encountered.

In order to satisfy this demand, a restructuring of the production process was necessary. Instead of the production of unique products by assembling a large number of specific components, more and more organizations assemble a range of customer specific products on the basis of a limited number of standardized components. This shift is represented in Figure 1, where the isolated pyramids represent the traditional product structures, with components specific to the product reflected by their bottoms and the (unique) client-specific end products by their tops. The hourglass figure shows the notion of assembling a variety of end products using a limited number of components. The limited number of components are reflected by the neck of the hourglass and the larger number of client-specific end products by its top.

Key to the success of such an approach is the explicit reuse of standardized components enabling a reduction of leadtime (for the customer) to be coupled with a client specific design of the final product.

Any industrial organization is now able to choose its own niche within the possibilities set out above. There are opportunities in the marketplace for organizations specialized in the fast delivery of relatively cheap standard products, but also for organizations who design expensive complex customer specific products as well as for organizations choosing the intermediate 'hourglass' approach. It is only important that the choice for such a niche be made clearly and that consequences arising from this choice are recognized.

Similar pressures from the market place, as prompted by the changes in industrial production, can be noted in the software market. In reaction to this, software production can learn from the changes that were made in industrial production.

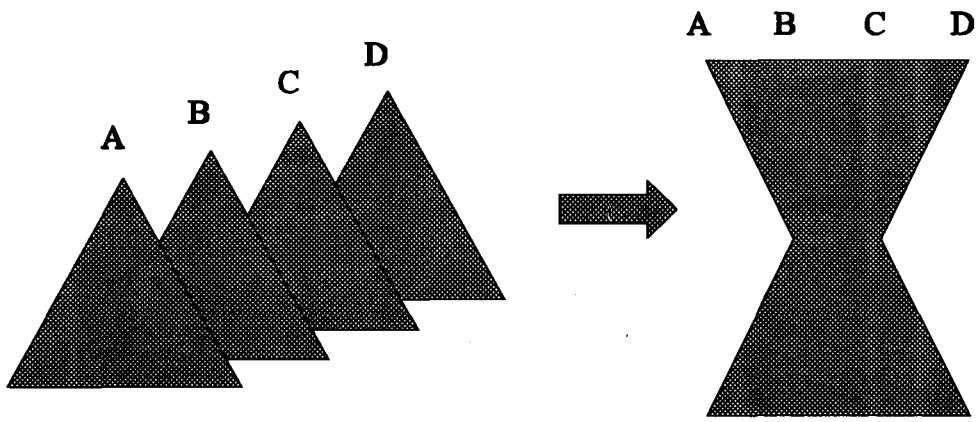

Figure 1: Different product structures

As in industrial production, the key to these changes will be the way in which reuse of previous work can be effected. Reuse of components of previous software systems as a potential for improvement has already been recognized [Pri 90], [Sel 89]. This reuse of previous work in software production is not restricted to the reuse of software product components (e.g.: design components, code modules), it also concerns the reuse of product references in the earlier phases of the life cycle (e.g.: domain analysis descriptions, conceptual data models), software process knowledge (e.g.: life cycle models) [BiP 89].

The customer dependency of production of software can be determined by the way the life cycle is covered by the software producing organization. We will call the customer dependency of the software production low when the software producing organization is oriented at an entire software consumer market. As such the software producing organization itself has already defined the main requirements and has already specified the design of the software products in a generic way. Especially in the lower part of the life cycle the production of software is based on assembling specific end products by reusing standard components. See the 'low' customer dependency part of Figure 2.

An increasing customer dependency of the software production means that the collaboration between software developers and customers increases. This will take place in the upper phases (e.g.: requirements and design) of the life cycle. In these phases process and product models are reused as an aid of reference. In the lower phases (e.g.: construction and implementation) of the life cycle, components can be reused. However, due to the increasing extent of customer dependency, adaption will be necessary. The 'moderate' customer dependency part of Figure 2 shows the interrelationship between increasing customer dependency and life cycle coverage by reusing previous work.

The ultimate customer dependency in software production can be found in software production situations where products are developed completely 'from scratch' and where

no previous work is reused. Again and again the software producing organization will cover the life cycle phases (both upper and lower) in accordance with the specific requirements and (technical) constraints of the customer organization. See the 'high' customer dependency part in Figure 2.

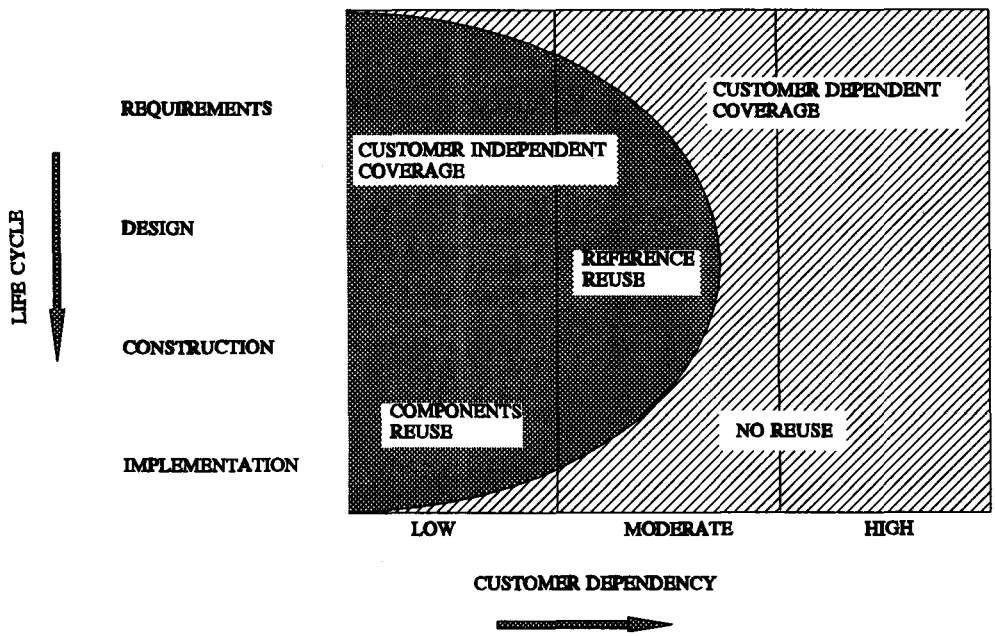

Figure 2: Customer dependency and reuse in software production

We now have distinguished two related characteristics of software production, namely the degree of customer dependency of the production process and the degree of reuse. These characteristics will be used in section 4 to describe three major types of software production environments. However, to establish a clear view on software production itself, we first have to discuss the main categories of software production activities.

3. MAIN CATEGORIES OF ACTIVITIES IN SOFTWARE PRODUCTION

In software production, different related activities have to be distinguished (e.g.: [Boe 81]). In this paper we discuss three main categories of activities which we will denote as the product modelling category, the process modelling category and the category of managing the reuse of previous work. These categories are summarized in Figure 3 and their content will be discussed in the sections below. The product modelling category is the traditional category of engineering activities, which is concerned with modelling the concepts relevant to producing a software system or the WHAT that has to be done (e.g.: [You 89], [Pag 88]). In current software production research, more and more attention is given to the process modelling category, which is concerned with the modelling of the software production processes itself [Tul 88], [Per 89]. These modelling activities are concerned with the HOW, the WHEN and the BY WHOM of the software production. As was also mentioned above, there is a growing importance of the activities that are concerned with the reuse of

previous work in software production [BiR 87], [PrF 87]. We therefore consider managing of the reuse of previous work as the third main category of activities.

3.1 Software Product Modelling

Software products are always developed by modelling the product in a number of life cycle phases. A common grouping of product modelling activities into phases is: requirements, design, construction and implementation. In each phase, different types of product models are specified and in each phase the product models are interrelated with the models in the previous and in the following phases. Therefore, the modelling of the product during the life cycle has to be considered as passing different levels of abstraction by transforming product models from one level to another. In product modelling, both formal and informal methods may be used. In the upper phases of the life cycle, e.g.: requirements and design, the product models are less formal and are less detailed than in the lower phases (construction and implementation and maintenance). On each abstraction, level the particular product developer (requirements analyst, designer, constructer, builder) has to deal with different specific development constraints (e.g.: from type organizational, conceptual or technical). The ultimate product of the product modelling activities is an implemented and operational software system.

3.2 Software Process Modelling

Producing software is more than producing a collection of product models. The flow of work that has to be done by the various players in the production of software products also has to be structured and controlled. Process modelling is a means to govern this flow of work by modelling the software production processes itself. Software production processes are captured in process models describing when, how and by whom specific modelling activities (e.g.: in terms of roles and tasks) have to be performed. Since the synchronization of the resourcing of the product modelling activities is an important process modelling issue, the modelling of software processes is principally interrelated with the product modelling. Software process models are common to multiple life cycle phases and serve as an assistant and guide to developers of different types through the life cycle.

In this paper we will distinguish two levels of abstraction in process modelling: the modelling of the life cycle and the modelling of the software development activities.

On the life cycle level, process modelling is mainly concerned with the determination of a life cycle strategy, in advance of the software production. Life cycle models have been around for many years [Boe 88], [You 87]. They are highly abstract, static and informal and they serve as global guidelines for project managers and developers to deliver (sub)products within an agreed time. Main aspects of process modelling on this abstraction level are: the phases of the life cycle which have to be passed, and the sequence in which they have to be passed (e.g.: linear, incremental, evolutionary).

On the activity level, the modelling of processes is concerned with a more formal specification of the detail and the variety of the software production activities [Tul 88], [Per 89]. Activity models are complex because of the multitude of software process aspects that have to be modelled (e.g.: roles/tasks, schedules, resource allocations, documents, control structures, etcetera). Due to the often extensive changing of the work in software production, activity models are highly dynamic and often evolve during the production processes [DeG 90].

3.3 Managing the reuse of previous work

During the life cycle many different products of the system being developed will be delivered in several versions and configurations. In advanced software production environments the reuse of this variety of previous work has to be managed. Production and identification of reusable elements should be explicitly acknowledged and organized as a separate task. Identified reusable elements have to be stored properly in integrated data dictionaries which are currently known as repositories. Given the importance of reuse, and taking into account recent research on reusable software libraries, e.g.: [Bur 87], [Pri 87], we consider in this paper the most important function of a repository to be the management of reusable elements. In such repositories, facilities are necessary to navigate between the reusable elements in order to locate and retrieve them easily, [BiP 89]. Full consistency, completeness and high quality accurate specification of reusable elements are vital for reuse but cannot always be gained in software production [Som 90]. Therefore we distinguish, in the variety of reuse of previous work, two types of reusable elements: product and process. On the one hand, we consider informal product references (e.g: domain analysis models, requirements specifications etc.) and formal product components (e.g.: design specifications, interface descriptions, modules of code etc.). On the other hand, we consider informal process references (e.g.: life cycle reference models) and formal process structures (activity structure components, design knowledge). This is reflected by a dotted line in Figure 3, in which also the three main categories of activities in software production and their interrelationships are shown.

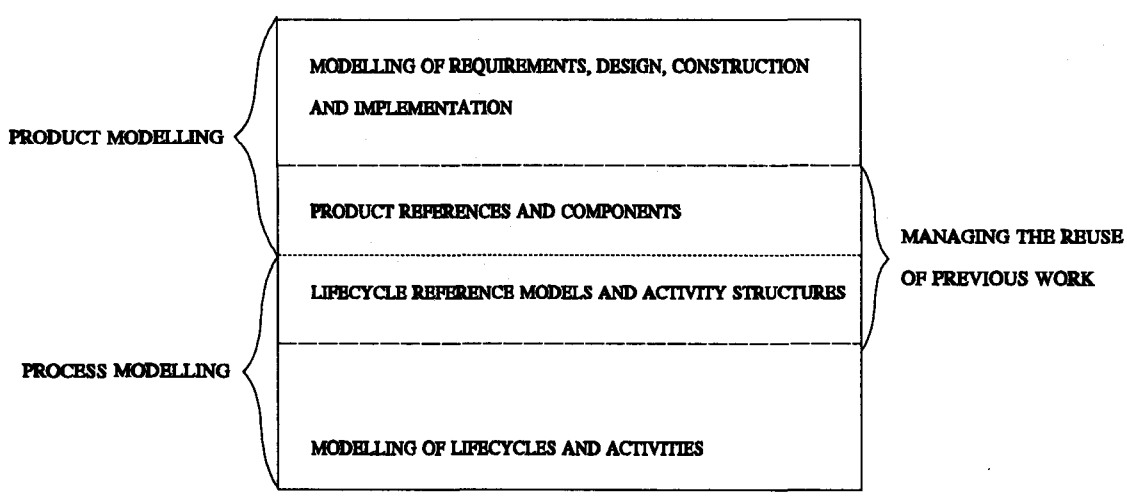

Figure 3: Product modelling, process modelling and managing the reuse of previous work

4. CUSTOMER ORIENTATION IN SOFTWARE PRODUCTION ENVIRONMENTS

In section two of this paper we adapted the characteristics of customer orientation from industrial production research, e.g.: the customer dependency and the extent of reuse of previous work, to the software industry. As such we established a notion of customer orientation for software producing organizations. We will now discuss three types of software production environments by focusing on the three main categories of software production activities. The three types of production environments were derived from a commonly accepted classification of customer oriented production environments in industrial research [Sar 81] where they are defined as:

- Selling Capacity;
- Engineer-to-order production;
- Assemble-to-order production.

4.1 Selling capacity

In this type of software production environment, purely software production capacity is sold to the customer. Capacity sellers are sometimes called jobbers. Many software houses largely behave like jobbers. These software production environments do not have typical products in which they are specialized and they are not focused on reuse of previous work. Their way of producing software varies with the strong differences in the software products that they produce and is highly customer dependent, [TrvR 90]. The need for highly consistent and accurate specifications of product and process models is strongly diminished by the absence of reuse of previous work in this type of production environment.

With regard to product modelling, capacity sellers are used to operate with different product modelling approaches in all life cycle phases. For instance, product modelling approaches are customized with respect to the preferences of the participating customer or with respect to the particular characteristics of the software application area at hand. As a consequence the product models will differ with respect to formality, completeness and consistency. This will hamper the automatic transformability of the product models from one phase to another.

With regard to process modelling, the structure of the software production processes of capacity sellers is, due to the extent of customer dependency, to a high degree unpredictable. Accurate specification of activity structures will hardly be possible. As a consequence, capacity sellers will mainly use global process models on the life cycle level. In Figure 4 the main characteristics of capacity selling production environments are shown.

4.2 Engineer-to-order software production

In engineer-to-order software production, a software production environment is specialized in the development of a range of software products aimed at specific application areas. Requirements of the client of the software product are restricted to this limited product range, thus profiting from experience gathered in the area. This can be interpreted as a partly (or moderate) 'customer dependent' coverage of the life cycle by the

software producing organization. In the requirements and the design phase of the life cycle, previous product and process models will be reused as an aid of reference. In the life cycle phases of construction as well as implementation, reusable product components and activity structure components can be of use. However, due to the extent of customer dependency of the production, adaption of reusable elements will often be necessary. This will hinder the

automatic transformation of product specifications and transformation from one phase to another will be executed semi-automatically or manually.

Due to the customer dependency in this type of production environment, movement down the life cycle will not necessarily be linear. Instead, various life cycle models will be developed dynamically, reusing life cycle reference models from previous production situations. The detailed process models on the activity level cannot be predefined completely in advance; they will also evolve during software production. As a consequence, activity structures from previous production situations have to be adapted when reused. In Figure 4, the characteristics of engineer-to-order environments are summarized.

4.3 Assemble-to-order software production

Assemble-to-order refers to a production environment where product variety is restricted to a number of preselected families of products. The customer dependency of the software production in this type of production environment is low, because the components are market oriented developed (i.e.: generic) instead of customer dependent developed (i.e.: specific). It will be clear that although emphasis in assemble-to-order software production is on assembling components, the initial development of the software components is an important engineering activity.

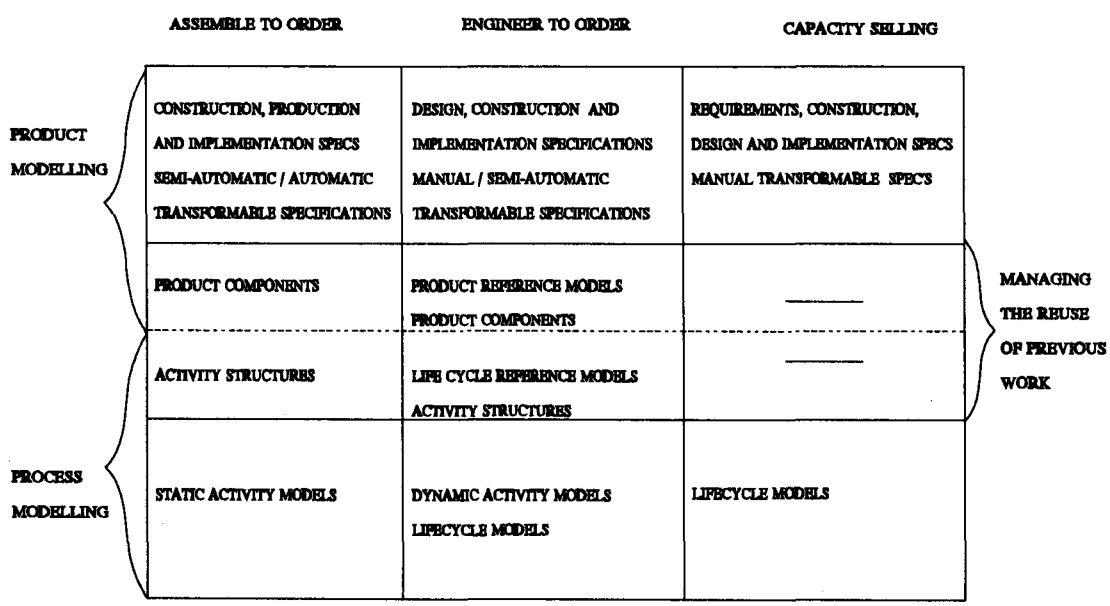

Figure 4. Characteristics of software production environments

In assemble-to-order production environments, the product and the process elements to be reused can be considered as reusable building blocks that are ideally unchanged in their use. With regard to the product modelling activities, assemble-to-order software production environments are usually restricted to the lower phases of the life cycle, e.g.:

construction and implementation. Since the reusable product components are highly consistent and accurately specified, in these lower phases of the life cycle, the transformability of specifications from one phase to the other can to a large degree be semi-automatic or even automatic, [Big 90].

Assemble-to-order software production environments are restricted to the lower life cycle phases which are passed in a linear sequence. Therefore, no life cycle modelling will take place. The process models on the activity level are linear and deterministic. These static activity structures do not evolve during assembly and can be considered as reusable components. In Figure 4 the characteristics of assemble-to-order production environments are shown.

5. COMPUTER BASED SUPPORT TECHNOLOGIES FOR CUSTOMER ORIENTED SOFTWARE PRODUCTION ENVIRONMENTS.

As stated before, there is no single 'best' production environment for all software producing organizations. Software production environments have to be designed properly according to their customer orientation characteristics. In this section we will present a basis for the derivation of guidelines for the design of SPEs and the selection of computer based support technologies for the three different types of software production environment. We will address three major classes of computer based support technologies by describing in which way they support the main activities in software production. Consequently, for each type of software production environment the appropriate technologies will be discussed.

5.1 Computer Aided System Engineering (CASE) - tools

The term CASE tools focuses on individual tools providing functional support for specific software product modelling activities within the life cycle of a software product, [Mar 89]. In this class of tools, upper CASE (e.g.: analysis or design tools) and lower CASE (e.g.: 4GL code generation, prototyping tools) can be distinguished. Individual CASE tools are focused on product modelling during only one single phase of the life cycle [Ter 90]. Tools of this type are unrelated and as a consequence only isolated product specifications are produced which are not automatic transformable to another phase of the life cycle. Because of the absence of shared data by means of a central and integrated repository, reuse of previous work will hardly be possible.

Considering the modelling of the processes, individual CASE tools will offer hardly any flexible support. With regard to the life cycle level of process modelling, each of the tools is restricted to one or two phases of the "freezed" waterfall life cycle model. Within a phase, they prescribe a linear sequence of product modelling activities.

Considering the way CASE tools support de main activities in software production, we conclude that various individual CASE tools can, in spite of their limitations, offer sufficient computer based support with respect to the great variety of product modelling activities in capacity selling production environments. However, the offered support in life cycle modelling (restricted to single life cycle phases) will be hardly sufficient. Further we state that the absence of a central dictionary is of minor importance in capacity selling production environments because this type of production environment is not focused on reuse. Considering the limitations of CASE tools, it will be clear that they do not offer sufficient support in both engineer-to-order and assemble-to-order environments. In Figure 5 the support characteristics of CASE-tools are shown.

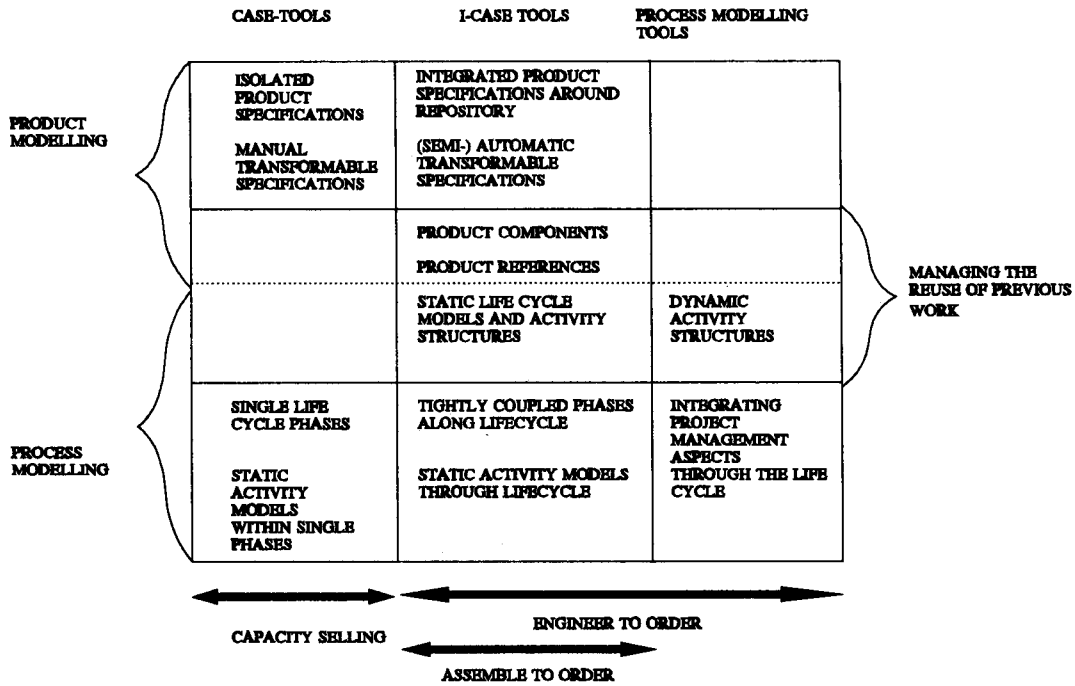

Figure 5. Characteristics of computer based support technologies

5.2 Integrated CASE (I-CASE) tools

I-CASE tools are built up from a set of product modelling (CASE) tools around a repository and as such they enable integration of product modelling activities across the life cycle [Mar 89], [MeMN 90], [Ter 90]. In I-CASE tools, not only product reference models and product components but also global life cycle reference models and linear activity structures models can be stored in one underlying repository. As stated before, we consider in this paper the management of the reuse of this vast amount of product and process elements which are generated all along the life cycle as the key task of this repository. As such, a repository should also provide the functionality to identify and to locate reusable elements across the life cycle. Since the tools in I-CASE share a repository, the various product models are specified highly accurately and consistently in the particular language of that repository. As a consequence, this will benefit the (semi-automatic and automatic) transformation of product reference models and product component specifications from one phase to another.

The main draw back of I-CASE tools is that they are based on the classical 'waterfall' life cycle model. In I-CASE tools, the phases of this life cycle model are tightly coupled. Within each phase, a linear sequence of activities is prescribed. Deviations from the predefined standard life cycle model or modifications on the activity model will interfere with the integrated support of the product modelling activities.

Considering the characteristics of I-CASE tools we conclude that these tools are useful in both assemble-to-order and engineer-to-order software production environments. The main reason is that in these two types of software production environments, as distinct from capacity selling environments, reuse support and integrated support are of great importance.

Assemble-to-order software production is focused on software production activities in the lower cycle phases which are passed in a linear sequence. Within these phases software production is focused on the reuse of both product and process components. I-CASE tools offer this kind of support and, especially the lower, construction and implementation parts of it, are a useful aid for assemble-to-order software production.

In engineer-to-order software production environments, product and process models will be used as an aid of reference in the upper part of the life cycle.
In the lower parts of the life cycle, software production can consist for an important part of adapting reusable product and process components. Considering the support characteristics of I-CASE tools we conclude that, especially with regard to product modelling and reuse of product components, tools of this class will be useful to engineer-to-order environments. However, due to the necessity of dynamic process modelling in engineer-to-order environments, I-CASE tools will be of little use with regard to process modelling. So engineer-to-order software production needs complementary computer based support. In Figure 5, the support characteristics of I-CASE tools are summarized.

5.3 Process modelling tools

Software process modelling is aiming at providing assistance and guidance to all people involved in the software production process [MaGD 90]. Process modelling tools have integrated functionalities to model the software production process itself as well as its interrelations. An important aspect of process modelling tools is the support they offer to the formal and detailed specification of dynamic activity structures [Tul 86]. These activity structures, in which design knowledge is captured, can be considered as important reusable elements in specific software production processes. In section 4.2 we concluded that especially in engineer-to-order software production environments, life cycle models and activity structures evolve during software production. As a consequence, the static and linear process modelling support of respectivily CASE and I-CASE tools are of little use in engineer-to-order software production. This type of software production has to consider process modelling tools as a necessary and complementary support to CASE and I-CASE tools, see Figure 5.

6. CONCLUSIONS

Software production environments within organizations have to deal with a huge fragmentation of computer based support technologies. Obviously, there is not a single set of computer based technologies available on the market which supports all the possible activities in software production. Selection of CBST should be based on a proper evaluation of the structure of the software production environment. However, there are no guidelines for designing SPEs and selecting computer based support technologies.

In this paper we considered software production within organizations from the perspective of customer orientation. We adapted characteristics of customer oriented production from industrial production research to the software industry. Furthermore we have elaborated a view on the main activities of software production and used this view as a basis to present characteristics of three particular types of customer oriented software production environments. The characteristics of those three types serve as a basis to derive guidelines for the design of SPEs. Finally we presented the characteristics of computer based support technologies so that they can be selected and applied effectively in the different types of software production environments.

We realize that in practice one will rarely find software production environments that exactly conform to the descriptions given in this paper. The three types of software production environments presented, are ideal types, i.e. extremes chosen for the sake of clarifying the concepts introduced. However, it is our contention that, as in other industries, the recognition of the niche in the market place that the organisation is aiming at and the consequences this aim has for the software production processes, will provide useful insights in the design of the software production environment and the selection of appropriate computer based support technologies.

Further steps in this research are aimed at the validation of the proposed three distinguished types of software production environments, their production activities and tools, and their interrelationship.

7. ACKNOWLEDGEMENTS

The authors would like to thank prof. dr. T.M.A. Bemelmans and prof. dr. ir. J. C. Wortmann for their helpful and inspiring comments.

8. REFERENCES.

[BaS 89] Baram G., G. Steinberg, Selection criteria for analysis and design of CASE Tools, ACM Software Engineering Notes, Vol. 14, no. 6, Oct. 1989.

[Bem 86] Bemelmans T.M.A., "Bedrijfskundig ontwerpen van bestuurlijke informatiesystemen", published in "Automatisering met een menselijk gezicht", in P.A. Cornelis, J.M. van Oorschot, Kluwer, Deventer, 1986.

[BeWW 90] Bertrand J.W.M., J.C. Wortmann, J. Wijngaard, Production Control, A Structural and Design Oriented Approach, Elsevier, Amsterdam, 1990.

[BiP 89] Biggerstaff T.J., A.J. Perlis, Introduction in Software reusability, Volume 1: Concepts and Models, ACM Press, New York, 1989.

[BiR 87] Biggerstaff T.J., C. Richter, "Reusability framework, Assessment and directions", IEEE Software, pp. 41-49, March 1987.

[Boe 81] Boehm B.W., "Software engineering economics", Englewood Cliffs, NJ, 1981.

[Boe 88] Boehm B., A spiral model of software development and enhancement, Computer, May 1988, pp 61-72.

[Bur 87] Burton B.A. et al., The Reusable Software Library, IEEE Software, April 1987, pp. 25-33.

[ChR 88] Chikofsky E.J., B.L. Rubenstein, 'CASE: Reliability Engineering for Information Systems', IEEE Software, March 1988, pp. 11-15.

[DaEF 87] Dart S.A., R.J. Ellison, P.H. Feiler, A.N. Habermann: Software Development Environments, IEEE Computer, Vol. 20, No. 11, November 1987, pp. 18-28.

[DeG 90] Deiters W., V. Gruhn, Managing Software Processes in the Environment MELMAC, ACM, Software Engineering Notes, Vol. 15, No. 6, Dec. 1990.

[Dow 86] Dowson M., ISTAR - An Integrated Project Support Environment, Proceedings of ACM Sigsoft/Sigplan Software Engineering Symposium, Palo Alto, California, Dec. 9-11, 1986, pp. 27-33.

[FeO 89] Fernstrom C., L. Ohlsson, ESF - an Approach to Industrial Software Production. In: "Software Engineering Environments - Research and Practice", ed. K.H. Bennett, Ellis Horwood Books in Information Technology, 1989.

[Gen 91] Genuchten van M., Towards a Software Factory, Thesis, June 1991, University of Technology Eindhoven.

[IEE 88] IEEE Transactions on Software Engineering, Special Issue on Software Engineering Environments, IEEE Transactions on Software Engineering, Vol. 14, No. 6, June 1988.

[LeT 87] Lehman M.M., W.M. Turski, Essential properties of IPSEs, ACM Sigsoft Software Engineering notes, Vol.12, No.1, Jan. 1987, pp. 52-55.

[Mar 89] Martin J., CASE Tools Comparison & Review: The Definitive CASE Seminar, Software Technology Seminars, December 1989- June 1900, James Martin Productivity Series Library, 1989.

[MeMN 90] Mercurio V.J., B.F. Meyers, A.M. Nisbet, G. Radin, AD/Cycle strategy and architecture, IBM systems journal, Vol.29, No. 2, 1990.

[MaGD 90] Madhavji N.H.,V. Gruhn, W. Deiters and W. Schafer, Prism = Methodology and Process-oriented Environment. In Proceedings of the 12th International Conference on Software Engineering, Nice, France, March 1990.

[NoN 89] Norman R.J., J.F. Nunamaker, Jr., CASE Productivity Perceptions of the Software Engineering Professional, Comm. of the ACM, Vol. 32, Sept. 1989, pp 1102-1108.

[Pag 88] Page-Jones M., Practical Guide to Structured Systems Design, 2nd edition, Prentice Hall, 1988.

[Per 88] Perry D.E., G.E. Kaiser, Models of Software Development Environments, Proceedings of the 10th International Conference on Software Engineering, Singapore, April 1988, pp. 60-68.

[Per 89] Perry D.E., editor, Proceedings of the 5th international Software Process Workshop, Kennebunkport, Maine, USA, September 1989.

[PrF 87] Prieto Diaz R., P. Freeman, Classifying software for Reusability, IEEE Software, Jan. 1987.

[Pri 90] Prieto Diaz R., "Implementing faceted classifying for software reuse", Proceedings of the International Conference on Software Engineering, pp. 300-304, Nice, 1990.

[Sar 81] Sari J.F., The MPS and the Bill of Material go hand-in hand, Rischard C. Ling Inc., 1981.

[Sel 89] Selby R.W., "Quantitative studies of software reuse", in Software reusability, Vol. 2, application and experience, editors Biggerstaff, T.J., Perlis A.J., ACM Press, New York, 1989.

[Som 90] Sommerville I., DSCC components catalog tool for software reuse, University of Lancaster, IEEE software, May 1990, pp. 72.

[Ter 90] Terry B., D. Logee, Terminology for Software Engineering Environment (SEE) and Computer-Aided Software Engineering (CASE), ACM SIGSOFT, Sofware Engineering Notes Vol. 15, No. 2, April 1990.

[TaTF 90] Tardieu H., R. E. Thomas, C. Fernstrom, O. Hesse, Proceedings of the Berlin seminar on the Eureka Software Factory Project, Nov. 1990.

[Tul 86] Tully C.J., Software Process Models and Iteration. In M. Dowson, editor, "Iteration in the Software Process", Proceedings of the 3rd International Software Process Workshop, Beckenbridge, Colorado, USA, November 1986.

[Tul 88] Tully C., editor, Proceedings of 4th. International Software process Workshop, ACM Software Engineering Notes, Vol. 14, No. 4, June 1989.

[Tay 88] Taylor R. et al, Foundations for the ARCADIA Environment Architecture. In Proceedings of the 1988 ACM SIGSOFT SIGPLAN Software Engineering Symposium on Practical Software Development Environments. pp. 1-13, Boston M.A. 1988.

[TrvR 90] Trienekens J.J.M., A.J. van Reeken, Effectiviteit van workbenches, Serc onderzoeksrapport 90/08, Software Engineering Research Centrum (SERC), 1990.

[WeHB 87] Weiderman N.H., A.N. Habermann, M.W. Borger, M.H. Klein, A Methodology for Evaluating Environments, SigPlan Notices, January 1987, pp. 199-207.

[You 87] Yourdon E., Managing the Systems Life Cycle, Yourdon Press, 1987.

[You 89] Yourdon E., "Modern Structured Analysis", Yourdon Press, New York, 1989.

[Zuc 89] Zucconi L., Selecting a CASE-tool, ACM, Software Engineering Notes, Vol. 14, no. 2, April 1989.