IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITP4120

Activation Styles for Changeable Order Management Systems

Joachim Berlak and Bernhard Deifel
Technische Universitaet Muenchen, Institute for Machine Tools and Industrial Management (iwb)
Joachim.Berlak@iwb.tum.de, deifel@in.tum.de

ABSTRACT

Turbulent conditions in the business environment require a need for change of enterprises. However, far too often organizational changes can not be implemented due to the rigidity of today's order management systems (OMS) during operation. Hence, the prevailing work gives advice to design complex commercial off the shelf (CCOTS) OMS and their cooperation with the order management process changeable.

INTRODUCTION

The inevitability of rapid change in the competitive environment of business is a commonly agreed fact [1, 2]. The changeful business environment, coupled with maturing sales markets, results in enormous pressures on enterprises in their efforts to be competitive [3]. Hence, enterprises have to operate their business under increasingly complex, turbulent, uncertain and unpredictable conditions. Significantly, the dynamic features of a turbulent environment are not just the outcome of interactions between organizations [4, 5, 6], they are generated by the environment itself [7]. To cope with these conditions, a permanent need for change will be the defining feature in the future business landscape [8, 9].

Enterprises produce products and/or services through the interaction of humans, organization and technology according to their order management process. Their business environment is commonly affected by the forces shown in fig. 1. In order to stay competitive, performance measurements are used to capture inefficiencies in order management, which in turn determine necessary changes.

However, adjustments to the business environment are often not possible because organizational changes cannot be implemented as intended (see fig. 1). The lacking ability of today's order management systems (OMS) to support necessary organizational changes is a substantial cause for this problem.

Reasons for the rigidity of OMS are various (see [10]). Mainly, OMS are CCOTS, which are developed for a application field, offer predefined functions and have to be customized to the specific organization before use [11]. Empirical studies ascertained that 30% of existing OMS are older than 10 years [12] and many still originate from the 1970's [13]. In summary, today's OMS are equally flexible (before use) and rigid (during operation) [14].

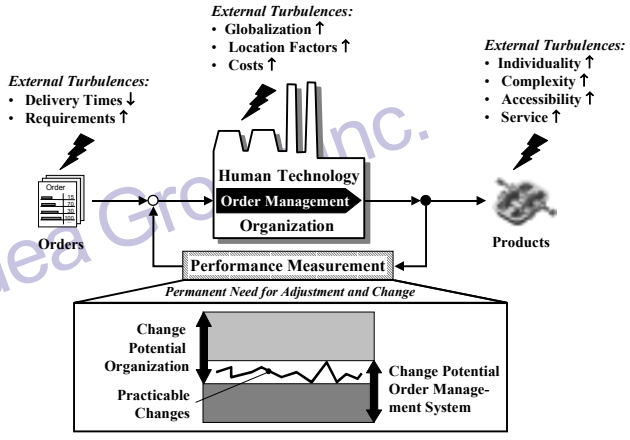
Consequently, changeable OMS have to be developed, which adequately support changes. Further, the entire cooperation between organization and OMS must be tailored towards changeability. The research project CHANGESYS of the Bavarian research consortium for software engineering (FORSOFT) focuses on this challenge. In this paper, results of this research are presented.

First, a cybernetic model of order management is developed. This approach communicates a mutual understanding of the interdisciplinary synergy of organization and OMS. Furthermore, a decision oriented approach which allows a systematic derivation of an architectural design for such kind of software is sketched. Additionally, decision orientation is applied for the first steps of architectural CCOTS-OMS design. Finally, a summary is presented.

CHANGEABLE ORDER MANAGEMENT

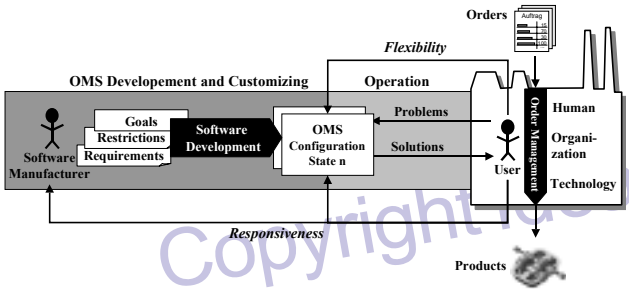
In this chapter, the theoretical concept for a changeable order management is presented. Before the so called cybernetic model of order management is described, basic terms and definitions are introduced.

Figure 1: Need for change of an enterprise



This paper appears in *Issues and Trends of Information Technology Management in Contemporary Organizations,* the proceedings of the Information Resources Management Association International Conference.

Figure 2: Context of changeability and OMS



Terms and Definitions

In this paper the term OMS is generally used for specific production planning and control systems (PPC systems), enterprise resource planning systems (ERP systems) and supply chain management systems (SCM systems). These software tools support occurring tasks in order management by planning and control functionality.

The term changeability is defined and discussed differently in literature, a unique definition did not yet intersperse itself so far [15]. The emphasis of past investigations was situated mainly in the disciplines of production management and economics, were changeability describes the ability of an enterprise to adapt to changing business conditions [16, 17, 18, 19]. In the software engineering field this term is relatively uncommon. Concerning CCOTS-OMS, the prevailing work postulates the following model for changeability (see fig. 2).

CCOTS-OMS are developed according to certain goals (e.g. return on investment, market share etc.) and restrictions (e.g. development tools, costs etc.) of the software manufacturer. Core requirements for the OMS result from the application field (e.g. user requirements etc.). Due to their characteristic CCOTS-OMS are configured to the specific organization before use (Configuration State n). However, OMS act as a service provider for problem solutions of the order management process. The changeability of OMS can be described as follows.

During operation of the OMS, software modifications can be achieved on the one hand by the implemented software functionality. The so called **flexibility** of the OMS describes a predefined change potential identified and implemented by the software manufacturer during OMS development. If changes can be realized by the flexibility of the OMS (see fig. 1), the user is able to solve the occurring problems by his own (e.g. resetting parameters, using alternate functions etc.).

Otherwise, the OMS **responsiveness** potential has to be utilized by additional activities (e.g. by the user). Hence, problem solutions can be achieved by using pre-defined OMS functionality (e.g. interfaces) and developing outside software solutions (e.g. self developed AddOn solutions). As another opportunity, organizational changes could be initiated to solve the order management problems without using additional OMS functionality. With more timely and/or costly expenditure, software solutions from other software manufacturer could be bought, installed and connected to the existing OMS in order to solve the problem. In some cases the best way to solve a problem is to acquire the software manufacturer himself. In this case new requirements will be realized within the OMS and delivered with a software release. In summary, changeability of OMS can be characterized by:

Changeability = Flexibility ⊗ Responsiveness

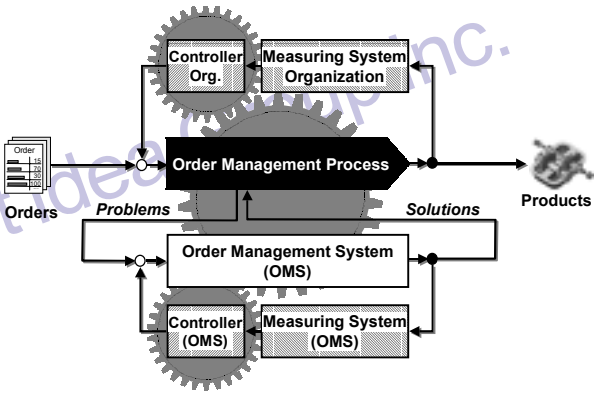
The flexibility is pre-defined in the software where as the responsiveness comprises additional adaptation abilities in order to find solutions for not assumed problems.

With this basic understanding of terms and definitions, the cybernetic model of order management is introduced.

Cybernetic Model of Order Management

In order to communicate a mutual understanding of the interdisciplinary synergy of organization and OMS, this cybernetic approach was developed. The term cybernetic belongs to control engineering which defines a closed feedback system, that remains stable despite disturbances [20]. A control loop consists of a regulated system and a controller. The latter affects the controlled system, whereby the suc-

Figure 3: Cybernetic model of order management



cess of the adjusting measure is steered.

In the cybernetic model of order management (see fig. 3) enterprises are treated as open systems exposed to a dynamic business environment, in which they fulfill orders by delivering products according to their order management process. To be competitive, a measuring system is needed, which initiates changes for occurring deviations.

Every task of the order management process needs adequate support by the OMS. Therefore, the secondary control loop is responsible to deliver suitably solutions for the occurring tasks and problems from the organizational control loop. Discrepancies between required and offered solutions are identified by an measuring system. In case of inefficiencies, the OMS has to be changed.

The OMS works in close cooperation with the organizational control loop to achieve a global optimum. The reason for this is that the required solution can be realized by the organization and/or the OMS. Either, the OMS has to be enabled to deliver additional solutions or the organizational measures make sure that the problem disappears.

DECISION-ORIENTATED SW-DEVELOPMENT

The OMS control loop described within the last chapter has two main influences on the OMS. Firstly it influences how the OMS should be customizable, i.e. how the shipped and installed software should be adaptable to customer's problems. Secondly, the evolution of the installed software is influenced. Hence, the OMS control loop results in a decision whether a customization of the installed software suffices, whether it triggers the evolution of the OMS or even both. Further it constrains what has to be done in the follow up operations.

In the rest of this paper, the focus is on the evolution of CCOTS-OMS. In [21, 22] the basics of the decision oriented development process were presented. In [23] this concept has been related to the OMS control loop described above. Here, the main ideas are sketched. For details, it is referred to the cited publications. After the process modeling consisting of roles, activities and work products is presented, a decision oriented development process and the model of work products is introduced.

Roles, Activities and Work Products

The basic concept for modeling development processes builds on the classical principle of the distinction of main element types roles, activities and work products [cp. 24]. Roles within this principle define tasks and responsibilities of persons which participate on the development – so called actors. Work products are central work pieces and results of the development including guidelines and intermediate results. Eventually activities are executed by actors and are responsible to develop work products.

Decision Oriented Development Process

Besides the trisection of development processes we introduced the principle to model development processes as a package of decision situations. A decision situation consists the elements problem space, decision object, alternative, measurement criterion, objective function solution and guidelines. The elements of a decision situation are represented by work products mentioned above and are filled stepwise within activities. The modeling of decision situations allows a systemic software development.

Work Products for Software Development

Work products are (see [21, 22]) central to the decision oriented software development process. A model of work products may be described with product types and relationships. Product types describe basic characteristics by attributes and aggregations of other product types. Product relationships express associations in the sense of E/R modeling [cp. 25]. Work products are adequately mapped to elements of corresponding decision situations.

Decision Oriented Architectural Design

In [21, 22] a model for decision oriented architectural design was developed. The term software architecture shortly denotes a hierarchical decomposition of a software system into subsystems plus a mapping from applied architectural styles to the resulting subsystems.

Aspects and Variations

In order to reach an adequate architectural design requirements of a CCOTS-OMS are structured into *aspects* and *variations* which base on the principle of the separation of concerns [cf. 26, 27, 28, 29, 30].

Shortly, products of product type *aspect* describe specific aspects of requirements at a CCOTS. These could be for financial software e.g. billing, balancing, graphical representations or statistics. Further, varying requirements of different customers lead to the need to describe their differences and commonalties adequately. We model a specialization of aspects, so called *variation-types*. They aggregate mutually excluding *variations*. A variation-type may be e.g. the country language for textual output.

Different variations may be realized within the same software but cannot be part of its function at the same time. We use the term *activation* of requirements for describing this fact. Requirement are *activated*, if they have an effective impact on the software at a given time. Else, they are *deactivated*. The activation of requirements may change during development, installation or during runtime, as we will see in the subsequent section. The activations of all realized requirements together build the configuration state of a OMS, which was mentioned in the preceding section (see fig. 2).

Architectural Elements

Now, the product types for architectural design are described. During the derivation of an architectural design, architectural styles are applied to build architectural alternatives. They fulfill the same content requirements in different ways. The software architecture is then selected according the so called decision requirements.

Within this paper we are mainly interested to present decision requirements and architectural styles supporting the activation of different variations which is subject of the subsequent section. For a detailed description of decision elements for architectural design we refer to [21, 22].

ACTIVATION STYLES

The section before we sketched our general view at architectural design as a process of different development decisions. Now we detail the impact of variations on architectural design. During requirements engineering requirements and variations are prioritized. Now, for a software design, requirements and variations have to be selected and especially their activation styles and their technical realization mechanisms have to be chosen:

Selecting Activation Style. With the activation style of a variation we mean the way users of a CCOTS can activate their specific variations. Roughly, activation styles of predefined variations can be classified into parameterization and modularization (cf. e.g. [31, 32]). By parameterization the software realizes all possible combinations of different variations within one product. In this case setting configuration parameters of this product activates the variations. Compared to this modularization denotes software which is separated into different modules. Here a user can set his configuration state by selecting and combining appropriate modules. A special case for modularization of a CCOTS is the separation of a software system into different products which are shipped independently. As we will see later, criteria which influence the selection for activation kinds come from customer requirements, marketing, development etc.

Selecting Realization Mechanism. Depending on the selected activation style a matching realization mechanism has to be chosen. This step may be supported by configuration management, by special techniques within programming languages (e.g. inheritance and

polymorphism) or directly by manually programming branches.

The decision for the activation style and appropriate realization mechanisms for variations is one of the first steps of architectural design. The rest of this section concentrates on activation styles. These are a special class of architectural styles. Firstly, important decision requirements and four different activation styles are presented.

Decision Requirements for Selecting Activation Styles

For each variation type an adequate activation style has to be selected. Generally, a larger modularization comes together with higher logistical efforts and a stronger cross linking of modules. Further, the selected activation style constrains applicable realization mechanisms and consequently influences development costs and time. In the following, the absolute and relative criteria which are both adapted from the taxonomy for requirement changes introduced in [33], are presented.

Absolute Criterion: Time of Change: The time of change indicates when a change has to be done.

Runtime: the change has to be done during runtime of the software. E.g. production systems may change their configuration without stopping the production process.

No Runtime: the change can be done when the software is not running.

Absolute Criterion: Product Separation: In order to increase the total income a marketing strategy may be to reach a clever separation of the CCOTS into different products which are sold independently.

Product Separation: The CCOTS can be separated into different products.

No Product Separation: Every variation has to be sold within one integrated product.

Absolute Criterion: Delivery Time: Closely related to the product separation criterion is the development and logistics criterion on delivery time. Suppose different variations of a CCOTS can be developed concurrently. Then the most flexible way is that the variations can be delivered asynchronously as well.

Synchronous Delivery: all variations are delivered synchronously after a CCOTS has been developed. This leads to easier test procedures and an easier logistics.

Asynchronous Delivery: the completion time of different variations is differently. This leads to a more flexible development.

Both criteria product separation and delivery time require for the possibility to modularize the CCOTS. Note that this is incorporated with an increasing number of modules the system complexity and logistics efforts increase as well (cf. [31]).

Relative Criterion: Change frequency: Another customer criterion is to support requirement changes adequately within the software in order to reach a high usability. Depending on the change frequency the need for a comfortable configuration of the software system is different. Comfortable configuration may conflict with development costs etc. We distinguish:

Long term periods: the intermediate time between two requirement changes is relatively long.

Short term periods: the intermediate time between two requirement changes is relatively short.

As a rule of thumb short term periods shall be supported by more comfortable configuration facilities.

Activation Styles

We now present four activation styles which can be found in practice. We do not take into account the possibility to define variations with script programming and open interfaces. We presume that a variation already is developed and show possibilities for activating that. The main difference between the activation styles concerns how and when a user has to choose the activation of a variation. We distinguish between purchase time, installation time and runtime. We firstly introduce activation styles and then relate them the decision requirements.

Activation Style: Setting of Runtime Options: Here all variations of a variation type are purchased and installed at once on a computer system. A user can activate variations during runtime by changing settings of the program e.g. switching radio-buttons or control-buttons within predefined windows.

Activation Style: Automatic configuration/runtime installation: This activation style does not presume that all variations have to be purchased and consequently installed on a computer system. The software can be extended with new variations during runtime (e.g. the addition of plug-ins in commercial web-browsers). This presumes two steps: an automatic installation of new variations and afterwards an automatic reconfiguration of already installed software. The new functionality is included within modules that are connected to the already installed software. The user here only is passive and is responsible for making the new module available.

Activation Style: Installation Selection: Here a user activates a variation during the installation time of the software. In this case he buys all functionality of a variation type together with the purchased package. During installation he selects, which variations should be activated. Usually installation selection is supported by installation programs or by editable configuration files. The main reason for using installation selection is the possibility to support static changes like static modularization, i.e. dividing the software into parts which can be exchanged when the software is not running, or the static setting of program options. Changes of activations of variations are done by reinstallations of the software.

Activation Style: Purchase of different products: A software producer may develop software as a package of different separately offered products. In this case a customer has to select variations by purchasing adequate products. The activation of a variation is done by separate installations corresponding products.

Decision Situation for Activation Styles

At first view the presented the activation styles seem to be very simple. But the main reason why we presented the different styles becomes clearer, if the selection criteria are mapped to activation styles within a decision situation.

Note again that all required absolute criteria must be fulfilled in a selected activation style. Compared to this, relative criteria should be fulfilled as far as possible. Each activation style allows a different combination of the two absolute criteria. In order to cover all “must”-cases each activation style is necessary.

SUMMARY AND OUTLOOK

The prevailing work presents the cybernetic model of order management and first steps towards decision oriented architectural OMS design. They cover organizational and information-technical views on order management in order to couple business processes and supporting software adequately. Identified inefficiencies of the organization can be adapted by structural and/or process changes of the organization. Closely cooperated is the OMS control loop, where OMS are service providers for organizational problems. Inefficiencies of provided services lead to adaptations of OMS structure and/or processes. The cooperation between organization and OMS control is coordinated by cooperate goals in order to reach a global optimum for an enterprise.

The OMS control loop is connected to the software development by means of the presented decision oriented approach. Software development hereby is seen as a collection of activities, mainly structured by decisions. The decision orientation leads to the presented work products for describing aspects, variations and architectural elements. Aspects and variations serve for structuring requirements of the whole market of an OMS. Basing an those architectural elements help to systematically derive an architectural design which is easily changeable due to the traceability of the underlying model of work products.

Generally an architecture derivation has to pass to main steps for realizing variations. The first step is the selection of an appropriate mechanism for the activation of variations. Hence, criteria and activation styles in order to support this selection were presented.

Present and future research focus the profound specification of the presented cybernetical approach. Further, the second step of architecture derivation will be focused in more detail, i.e. finding mechanisms which support a systematic realization of presented activation styles.

REFERENCES

[1] T. K. Das and B. Elango, „Managing Strategic Flexibility: Key to Effective Performance“, Journal of General Management 20 (1995) 3, pp. 60-74.

[2] T. J. Tetenbaum, „Shifting paradigms: From Newton to Chaos“, Organizational Dynamics 26 (1998) 4, pp. 21-32.

[3] T. E. Vollmann, W. L. Berry and D. C. Whybark, “Manufacturing Planning and Control Systems”, McGraw-Hill, New York, 1997.

[4] B. Chakravarthy, “A new Strategy Framework for Coping with Turbulence”, Sloan Management Review, Winter 1997, PP. 69-82.

[5] G. Reinhart, S. Duerrschmidt, A. Hirschberg and C. Selke, “Reaktionsfaehigkeit fuer Unternehmen: Eine Antwort auf turbulente Maerkte”, ZWF 94 (1999) 1/2, PP. 21-24.

[6] G. Schreyoegg, “Organisation: Grundlagen moderner Organisationsgestaltung”, Gabler, Wiesbaden, 1999.

[7] F. E. Emery and E. L. Trist, “The causal Texture of Organizational Environments”, Human Relations (1965) 18, S. 21-32.

[8] J. L. Bower, „Jack Welch: General Electric’s Revolutionary, Harvard Business School (1994) 4, pp. 1-22.

[9] J. Milberg, „Produktion – Eine treibende Kraft fuer unsere Volkswirtschaft“, In: J. Milberg and G. Reinhart (Edt.), „Mit Schwung zum Aufschwung“, Muenchner Kolloquium 1997, Utz, Muenchen, 1997, pp. 29-38.

[10] Berlak, J.: Changeable Order Management. In: Proceedings of the International Workshop on Open Enterprise Solutions: Systems, Experiences, and Organizations (OESSEO), 14.-15.09.01, Rom.

[11] J. Frank, “Standard-Software: Kriterien und Methoden zur Beurteilung und Auswahl von Softwareprodukten”, Mueller, Koeln-Braunsfeld, 1980.

[12] IPA (Edt.), “PPS-Angebot und Verbreitung”, IPA, Hannover, <http://ftp.ipa.fhg.de/100/projekte/pps/>

[13] B. Wessler, “Wettlauf um Milliarden – Der deutsche Softwaremarkt befindet sich im Umbruch”, Business Computing Special (1995) 3, PP. 8-12.

[14] S. Biskamp, “ERP-Loesungen: Starre Softwarekolosse haben ausgedient”, Information Week (1999) 8, PP. 20ff.

[15] S. Duerrschmidt, „Planung und Betrieb wandlungsfaehiger Logistiksysteme“, Utz, Muenchen, 2001.

[16] G. Reinhart, S. Duerrschmidt, A. Hirschberg and C. Selke, „Wandel – Bedrohung oder Chance ?“, io Management 68 (1999) 5, pp. 20-24.

[17] G. Reinhart, „Im Denken und Handeln wandeln“, In: G. Reinhart and H. Hoffmann, H. (Edt.), „Nur der Wandel bleibt, Wege jenseits der Flexibilitaet“, Muenchner Kolloquium 2000. Utz, Muenchen, 2000, pp. 17-40.

[18] M. Hartman and M. Spiewack, „Wandlungsfaehigkeit“, In: H. Kuehnle (Edt.), „STRATEMA – Wachstumsstrategien durch marktorientierte Wandlungsfaehigkeit und produktnahe Dienstleistungen“, IMS, Magdeburg, 1999, pp.13.

[19] E. Westkaemper, „Die Wandlungsfaehigkeit von Unternehmen“, Werkstattstechnik 89 (1999) 4, pp. 131 f.

[20] W. F. Daenzer, “Systems Engineering”, Verlag Industrielle Organisation, Zuerich, 1989.

[21] B. Deifel, “Requirements Engineering komplexer Standardsoftware”, TU Muenchen, Muenchen, 2001.

[22] B. Deifel, W. Schwerin and S. Vogel, “Work Products for Integrated Software Development”, Technische Universitaet Muenchen, Technical Report, 1999.

[23] J. Berlak, B. Deifel, “Designing Changeable Order Management Systems”, OOPSLA-Workshop on Engineering Complex Object Oriented Software for Evolution, Tampa Bay, Florida, 2001

[24] J.-C. Derniame, B.A. Kaba and D. Wastell, „Software Process: Principles, Methodology, and Technology“, Springer, Berlin, 1999.

[25] Chen P.P., “The entity-relationship model – towards a unified view of data”, ACM Transactions on Database Systems, 1(1976) 1, pp. 9-36.

[26] E. Dijkstra, “A Discipline of Programming”, Prentice Hall, New York, 1976.

[27] P. Tarr, H. Ossher, W. Harrison and S. M. Sutton, “N Degrees of Separation: Multi-Dimensional Separation of Concerns”, ICSE´99, Los Angeles, 1999.

[28] S. Clarke, P. Tarr and H. Ossher, “Designing for Evolution with Subjects”, ICSE´99, Los Angeles, 1999.

[29] G. Kiczales, J. Lamping, A. Mendhekar and et. al., “Aspect-Oriented Programming”, Springer, 1997

[30] R. J. Walker, E. L. A. Baniassad and G. C. Murphy, “An Initial Assessment of Aspect-oriented Programming”, ICSE´99, Los Angeles, 1999.

[31] R. Conradi, B. Westfechtel, “Towards a Uniform Version Model for Software Configuration Management”, SCM7, Boston, MA, LNCS 1235, 1997.

[32] Karhinen A., Ran A., Tallgren T., “Configuring Designs for Reuse”, SSR’97, Boston, MA, USA, 1997.

[33] B. Deifel and C. Salzmann, “Requirements and Conditions for Dynamics in Evolutionary Software Systems”, Proceedings of the International Workshop on the Principles of Software Evolution, IWPSE99, Fukuoka, 1999.