LANGUAGES AND MECHANISMS FOR SOFTWARE PROCESSES AND MANUFACTURING ENTERPRISE PROCESSES: SIMILARITIES AND DIFFERENCES

Selma Arbaoui, Alain Haurat, Flavio Oquendo, Franck Theroude, Herve Verjus
LISTIC-ESIA – University of Savoie – BP 806 – 74016 Annecy Cedex (France)
Email: {arbaoui, haurat, oquendo, theroude, verjus}@esia.univ-savoie.fr
Phone: + 33 4 50 09 65 80 Fax: +33 4 50 09 65 90

Keywords: process, process support, software process, enterprise manufacturing process

Abstract: This paper tends to confront two wide and deep process fields: software process and enterprise manufacturing process (called for short, manufacturing processes). It will analyse them, present a state of the art according to a set of process requirements and conclude with some similarities and differences.

1 INTRODUCTION

The *process* of developing and maintaining products or services plays a crucial role in determining the quality level of the product or service but also the cost of developing, supporting and maintaining it.

Process has been recognized as important for decades in manufacturing, software production, etc. In fact, software producers and telecommunications service providers, from small vendors to « giants » like Microsoft and AT&T, have started to model, analyse, and re-engineer or improve the processes used to produce, support and maintain their products and services.

The term process encapsulates two things: the process of producing and maintaining the product or service (the production process), and the process of contracting, marketing, and managing the organization that produces the product or service (the business process).

Process researchers represent a very large community that covers four main domains: knowledge engineering, manufacturing engineering, software engineering and administrative process engineering.

Process modelling has been differently tackled according to concerns, culture, knowledge and objectives of each process community. These research efforts result in several environments and systems for modelling, analyse, simulating or enacting processes. Figure 1 presents the different environments along with the process domain's objectives. The latest domain addresses process aspects that were neglected by production management systems (MRPII, schedule systems) such as employment process.

Research Domain	Knowledge Engineering	Manufacturing Engineering	Software Engineering	Administrative Process Engineering
Objectives	Knowledge modelling and management	Manufacturing process modelling and improvement	Software process modelling and management	Administrative process modelling and management
Technology designation	Knowledge Base Systems (KBSs) and	Business Process Reengineering (BPR) methods, Dynamic scheduling systems, Benchmarking methods, Continuous Improvement management systems	Process Centered Environments (PCEs)	Workflow Management Systems (WfMs)

Figure 1: Process research domains.

This paper will focus on the two latest process research domains, i.e. Manufacturing and Software engineering. There is a wealth literature workflow management systems and software process technologies. Several papers compare these domains and present their common characteristics (Warboys, 1994; Ellis and Nutt, 1996). Some of the developed

process environments combined both workflow and software process technologies, they will thus show how the two technologies can learn from each other (Alloui et al. 2000; Arbaoui and Barghouti, 1996). Some other papers have focused on manufacturing and administrative process engineering (Richert and Dadam, 1998; Myers and Berry, 1998).

Regarding the lack of research works common to software and manufacturing processes, it seems us appropriate to join our efforts and knowledge of these both research domains[1], in order to examine relationships that may exist between them, how could they learn from each other and how could they complement each other.

Section 2 introduces manufacturing process and software process domains using Dowson's framework (Dowson, 1992), that was originally proposed for the latest domain. However, how we will see in section 2, this framework is enough general to be applied to other process domains. This will lead us in section 3 to a set of requirements common to both process domains. Section 4 presents how these requirements are tackled by some existing software process centred environments and manufacturing systems. In section 5 we analyse the results of the state of the art presented in the preceding section, in order to answer questions like: to what extent each requirement is fulfilled by both the process domains, why a given requirement is much more in the research concerns of one domain than another, could we use a technology developed in one domain in the other, etc. We will conclude in this section with some perspectives about how theses two technologies can proceed their development while learning from each other.

2. MANUFACTURING PROCESS AND SOFTWARE PROCESS: A TERMINOLOGY FRAMEWORK

Even if the final product is quietly different, software process and manufacturing process domains aim at developing methodologies and technologies for modelling and enacting processes that are performed by individual or group of tools and/or human agents. Even if the global objective is the same, it seems interesting to present separately these two domains using a single framework, i.e. Dowson's framework, in order to highlight their similarities and differences.

Basically Dowson's framework presents process concepts through three domains: process model domain, process performance domain and process enactment domain, that aim at covering all phases starting from the process needs to the process termination.

2.1. Software process domain

The terminology classifies software process concepts in the three domains of Dowson's framework (see figure 2).

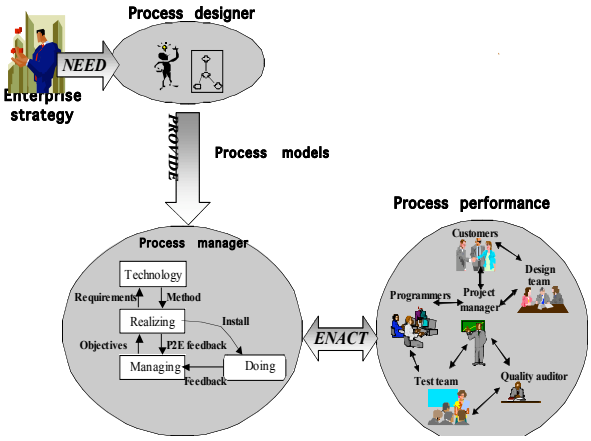

Figure 2: Software process domains.

Process model definition domain

It contains characterizations of processes or fragments of processes, expressed in some notation, e.g. software process modelling language, in terms of how they could or should be performed. Its major related term is the *software process model* that is a software process abstraction described with a formal or semi-formal language called a Process Modelling Formalism or Language (PML, for short). The *meta-process model* represents the set of (meta-) activities to model, to analyse software processes and to support their evolution[2].

Process models enactment domain

This domain encompasses what has to be provided to support process performance governed by process

[1] F. Theroude and A. Haurat are researchers in the industrial engineering field, the three other authors work on the software process technology.

[2] Evolution means the act of changing existing process models or process instance. This notion will be more detailed in sub-section devoted to PCEs requirements.

model. *Enactment* means the manual or automatic execution of an instantiated model.
Process Centred Environments (PCEs) have been developed. They are software systems that assist in the modelling and automation through enactment of software process models. A *customized process model* is said to be *instantiated* when its artefacts are linked with concrete products and project resources. A *customized process model* is the result of the refinement and adaptation of a generic process model to a specific project. Finally, a *PCE (*called also *PSE, Process Sensitive Environments* or *PSS Process Support System, PSEE Process-centered Software Engineering Environment)* is defined as the set of mechanisms that provides a variety of support (assistance, guidance, monitoring, automation, etc) to software process performers by enacting an explicit representation (i.e. model) of this process. Process models enactment results in automating the activities that can be carried out without the intervention of human agents and supporting people in performing the activities that require their participation. In order to facilitate process models communicability, PML designers have to emphasis on the enactment of these languages but not at the expense of their clarity and legibility.

Process performance domain
It encompasses the actual tasks and activities that are performed by the process agents (human or not) in the course of a software process. *A software process* is defined by the set of technical and managerial activities carried out in the production and the maintenance of software. It is a partially ordered set of activities each of them is associated with its related artefacts, human and computerizes resources, constraints, policies, etc. In the process performance domain, one may discern between the *performer* who may be a project manager, a programmer, a system analyst, a quality auditor, a tester, or the *end-user* who is the user of the product that is developed.
Dowson pointed out that for providing relevant and useful support to software development, process enactment and process performance have to be coupled in PCEs. Several research works on software process evolution support maintained that process model enactment state and process performance state consistency is a key feature and if the gap between them becomes important the process enactment will no long influence process performance and the PCE will be rejected by users (Arbaoui and Oquendo, 1993). However, the adoption of this technology in real software projects pointed out that this thesis is too restrictive, mainly PCEs failed in supporting people when unforeseen situations are encountered. An original approach to process evolution based on tolerating deviations between process enactment and performance support introduces two supplementary terms. The first one concerns the process performance domain (Cugola, 1998): the *actual process* that is the actual process as it is performed in the real world. The second one concerns the enactment domain: *the observed process*, that is the partial view that a PCE has of the actual process and that depends entirely on performer's feedback. The PCE is only aware of the actions the user is performing under its control.

2.2. Manufacturing enterprise process domain

Industrial process domain deals with the whole process of enterprise. In order to satisfy client requirements, enterprises need to react rapidly facing external events (clients needs) as well as internal events (dysfunction, urgencies, etc.). Enterprise is a complex entity that deals with a lot of constraints (organisational, functional, informational, etc.) that have many interactions each other.

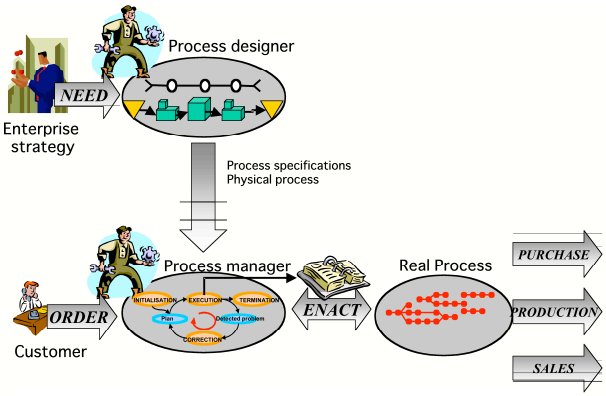

Figure 3: Manufacturing process domains.

This notion of complexity induces the need of modelling to make the enterprise more intelligible and to facilitate the actor's activities.
Processes capture enterprise expertise, knowledge and allow to be more efficient. Process modelling is a crucial step in enterprise comprehension and continuous improvement (ISO 9001 V2000).
Different approaches have been proposed according to objectives that have to be reached. Each approach addresses different aspects such as simulation, continuous improvement, business process re-engineering, etc., in different application domains.
Modelling is a important need to be able to catch enterprise complexity because organisation

processes are complex, long time, and also involve lots of resources (humans, means, etc.) they have to be modelled.
We could identify three major domains.

Process design domain
This is the first step of process life cycle. The process design start due to an explicit need expressed at enterprise strategic decision level. This domain could be considered in the same way as the software process model definition (same terminology and concepts).

Process management domain
It deals with the *management* of the process in terms of definition of process plan, process execution[3] studying, process correction, and process evaluation.
This domain encompasses what has to be provided to support execution. *Enactment* is used in the same way as in the software process domain.
The *instantiation* is relying on resources available in accordance on customer order. A process evolves in a restricted space defined by an initial and final state (goal). The construction of a *process instance* implies the definition of a research space. This space defines all reachable states in respect of technical constraints (availability of resources, precedence constraint, etc) and objective constraint (Theroude et al., 2002). A state is reach through activity realisation (which is a combination of resources, operation, etc). The construction of a process instance could use many different techniques such as Backtracking, Means and analysis, least commitment (Jain and Meeran, 1999), Hierarchical task network planning, case based planning (Myers and Berry, 1998), user definition, etc.
This enactment consists in information exchanges between the process manager (commands sending) and the *real process* (sending current state and exceptions). Those informations allow the process manager to take decisions to succeed in the stated objective associated to the customer order. This manager could use *scheduling tools*, *computer aided decision making tool*, *simulation* in order to build an instance or to change a given process instance in order to reach the stated objective according to the environment constraints.

3 Execution of the process means the realisation of the activities by physical systems in response to an order that is mostly performed by an MRP-based system.

Real process domain
It represents all enterprise process in execution such as purchase process, production process, sales process, etc.
This domain encompasses the actual tasks and activities that are performed by the process agents (human or not) in the course of a process. A *real process* is defined by the set of technical and managerial activities carried out in the production and the maintenance of product. It is a partially ordered set of activities each of them is associated with its related artefacts, human and means, constraints, policies, etc.
This execution is constantly evaluated in order to detect problems and/or dysfunctions. The evaluation relies on performance formalisation, which specifies the *performance indicator* that is defined by: "a variable indicating the effectiveness and/or efficiency of a part or whole of the process or system against a given norm/target or plan". In this sense, a model of performance indicator involves the three following topics (Berrah et al., 1998):

- expression of the objective to be reached;
- acquisition and comparison of effected measure with the objective;
- appreciation of acquired measure in accordance with the context and the know-how of the observer.

The evaluation returned by performance indicators can be characterised by: (1) performance evaluation flexibility that can be categorical (is or is not fully satisfied) or gradual (partly satisfied) and, (2) performance evaluation uncertainty. This problem is principally raised by uncertainty of the measure (Berrah et al., 1998).

3. MANUFACTURING PROCESS AND SOFTWARE PROCESS FIELDS: THE REQUIREMENTS

The analysis of both software and manufacturing processes using a single framework leads us to a set of common process requirements that will be presented hereafter. In the following, we will use the term "process systems" to designate both software systems and manufacturing systems.

Processes modelling and enactment
In order to satisfy client requirements, enterprises need to react rapidly facing external events (clients needs) as well as internal events (dysfunctions, urgencies, etc.). Enterprise is a complex entity that deals with a lot of constraints (organizational,

functional, informational, etc.) that have many interactions each other.
This notion of complexity induces the need of modelling to make the enterprise more intelligible and to facilitate the actor's activities.
Processes capture enterprise expertise and knowledge and they allow to be more efficient. Process modelling is therefore a crucial step in enterprise comprehension and continuous improvement.
Different approaches have been proposed according to objectives that have to be reached, such as simulation, enactment, continuous improvement, business process re-engineering, etc.
Process enactment allows enterprise to reach a stated objective. This execution induces a bi-directional communication between the physical system (depending of the domains: PCE or manufacturing systems) and the information system.
Process systems must provide a process modelling language with a specific syntax and semantics as well as an enactment support. Process models enactment result in automating the activities that can be carried out without the intervention of human agents and supporting people in performing the activities that require their participation.

Control and quality
Enterprises have to take into account quality in order to stay competitive.
Investigations on processes (software processes as well as manufacturing processes), consider process control and improvement as a key topic allowing to increase the quality of processes results (the software products or the manufacturing products). Other researches are dedicated on the products (software or manufacturing) quality itself that is not directly covered by the process domains.
A process system must provide control, measurement and analysis mechanisms in order to reach stated objectives relatives to quality.

Evolution
Evolution is an intrinsic characteristic of processes. Indeed, they are human-intensive processes that involve a large number of people for a long time and which often cannot be completely defined in advance. Furthermore, people are often faced to unforeseen situations (arising from human or means) that force them to deviate from the process model.
A process system is aware of process evolution thanks to process performance (real process) feedback. This feedback allows the enactment domain to gain knowledge about process performance (real process) reality in order to make its « image of the reality » as consistent as possible with the state of the process in the performance (real process) domain. However the quality of this feedback will be subject to the vagaries of human nature or means dysfunctions. Process performers may forget to provide the needed feedback, or provide erroneous information, they may be unsure about the validity of intermediate results or they may bypass the points at which feedback is expected.
When correctly provided, feedback may results in different kind of process evolution. (Arbaoui et al., 2002; Reichert et al., 1998) identified two kinds:

- *shared evolution* (also called *permanent evolution*). This process model evolution will affect the enacted process model and therefore will concern current and future observed processes (see section 2) that will be modified in order to be adapted to process performance (real process;
- *private evolution* (also called *temporally evolution*). In this case the process model is not changed, change will be local to the process model instance that is currently enacted, i.e. to the observed process. Private evolution is used to face unforeseen and exceptional situations that don't necessitate the process model modification but only the current observed process.

A process system should support shared and private evolution and should provide mechanisms to handle the consistency problem between process performance (real process) and enactment.

Distribution
Process is a highly distributed activity. This issue involves different processes in different locations that cooperate to achieve a given objective. Two kinds of process distribution are identified: internal distribution in an enterprise and external distribution out of an enterprise.
Theses imply a distribution of responsibilities, technologies (means or software), resources management to provide a given product.
This distribution involves several actors with different objectives. Each actors of a given process (customer, producer, etc.) has its own strategy, information system and production schedule.
Process systems requirements for distribution support concerns two facets: social and technical. The former consists of requirements for mediating social interaction when possible, by providing automated environment support for it, and therefore reducing the gap between enactment and

performance (real) states. The technical requirement is motivated by the recent evolution in software production and its intensive distribution. Indeed, it is not necessarily monolithic but can be composed of numerous process fragments, each one running on a remote site and has to interoperate with the other processes.
A process system must be able to support distribution, heterogeneity and interoperability.

4. MANUFACTURING PROCESS AND SOFTWARE PROCESS FIELDS: STATE OF THE ART

Processes modelling and enactment
Software process domain: several Process Modelling Languages (mostly enactable) have been proposed along the last decade that are based on different approaches and paradigms (Promoter, 1999; Arbaoui et al., 2002): logic-based, procedural, petri nets, production rules, object-oriented, active-database.

Manufacturing process domain: Different approaches have been proposed (Vernadat, 1996) where enactment is not the main issue. These approach address modelling SADT, IDEF 1.X, ACNOS, ARIS, modelling architecture CIMOSA, GERAM, information system modelling MERISE, OLYMPIOS, decision system modelling GRAI, GRAI/GIM (Doumeingts, 1984).

Control and quality
Software process domain: software process studies motivated the development of numerous methodological approaches to structuring, organizing, documenting and formally describing processes in order to evaluating or improving them, such as the Capability Maturity Model (CMM) (Paulk et al., 1993), Bootstrap (Kuvaja, 1994), and the Quality Improvement Paradigm (QIP) (Basili and Green, 1994), Software Process Improvement and Capability dEtermination (SPICE, 1998), etc. As for the control support, measurement and analysis methods and systems have been developed such as Tailoring a Measurement Environment (TAME) (Basili and Rombach, 1987), Amadeus (Selby et al., 1991), Balboa (Cook and Wolf, 1998), Software Management Environment (SME) (Hendrick et al., 1994), OMEGA (Cîmpan and Oquendo, 2000). Most of them perform *post-mortem* (*off-line*) analysis, while the others perform *on-line* analysis.

Manufacturing process domain: in manufacturing systems, an efficient control has to be efficiently control. Since the works of R. Anthony in the 60's (Tardieu and Theys, 1987), the enterprise control function is often structured in three layers (Orlicky, 1975; Vollmann et al., 1988): the strategic layer, the tactical layer, the operational layer.
The layer organisation previously presented allows to define a control architecture showing how control activities are distributed in an information system. Based on Dilts et al. work and Bongaerts thesis (Dilts et al., 1991; Bongaerts, 1998), five different control architectures have been identified: Centralised, Proper hierarchical, Modified hierarchical, Heterarchical, Holonic. These different architectures propose control activities with more or less decision autonomy and wideness in action possibility.
Furthermore, in order to satisfy quality different kind of certifications has been defined. (depending of the certification) (ISO 9001 V2000; AFAQ; CEN).

Evolution
Software process domain: a lot of ink has been spilled over software process evolution in order to identify its major factors (Why?), what are the process elements that are affected by the evolution, (What?), how to model software process evolution and how to enact evolutive process models? (Lehman and Belady, 1985; Madhavdji 1992; Kaba and Derniame, 1996).
A great part of PCEs provide an inter-fragment on line evolution support (the process fragment concerned by the change is not already enacted and can be modified or replaced by another fragment). Examples of such systems are: ADELE/TEMPO (Belkhatir et al., 1992), Process Wise (Bruynooghe et al., 1991), MARVEL (Kaiser and Ben-Shaul, 1993). Modifying a process fragment during its enactment is partially handled (Arbaoui et al., 2002). Furthermore, some approaches aim at modelling the evolution process itself (Greenwood et al., 2002) and provide environments for the enactment of the resulting models (Cunin, 2002).

Manufacturing process domain: the evolution problem is not directly addressed in manufacturing process domain. This issue could be tackled:

- in a dynamic schedule (Jain and Meeran, 1999) solving problem in accordance with the physical environment constraint. This approach allows the modification of the initial schedule based on information coming from the physical part;

- relying on workflow approach that defines different way for evolution. When a dysfunction is detected (low process evaluation), the process manager has to evolve the process instance. This change implies to understand the problem (specific to the enterprise) and to propose a solution (Myers and Berry, 1998). This solution is then integrated (Casati et al., 1996) in the process instance (see *Process management domain*).

Distribution

Software process domain: technically the PCEs differ according to the fact that they are monolithic systems, open or closed, providing some capabilities in components interoperability and/or heterogeneity. These are managed by common interaction languages and meta-models (WfMC, 1996), (Lee et al., 1998) or common repository that manages a global process state (Heimbigner, 1992; Estublier et al., 2001).
Social aspects are not well covered. Few of PCEs, using PMLs based on formal languages, aim at taking into account negotiations among their performers. For instance PEACE+ uses an interaction language based on a modal logic approach for negotiation and intention formalization, LITTLE-JIL language defines activities coordination that are distributed process elements.

Manufacturing process domain: the internal distribution of the process depends on the control architecture definition (Dilts and al., 1991) where each architecture has advantages and disadvantages.
The external distribution of the process consists in an externalisation of a part of the product life cycle. In this case we could find several work concerning virtual enterprise and extended enterprise models who aim at taking care of the distributed nature of the resulting enterprise (Janowski et al., 1999), (Burlat et al., 2001). The Supply Chain Management (SCM) in most of the cases addresses this process distribution. This approach provides to enterprises, concepts and method in order to support external distribution for measuring a supply chain global performance (Swaminathan et al., 1998; SCOR V5).
Furthermore the external distribution implies information exchange and flow coordination address respectively in (Cannon, 1993).

5. ANALYSIS AND CONCLUSION

In this paper we have tried to explore two wide fields and to confront them: software process and enterprise manufacturing process. At this end, we have used the Dowson's framework as a mean for analysing them and retrieve a set of common requirements. In this section we will present our analysis results concerning each of these requirements. These fields are so wide and so complex that it was not possible for us to be exhaustive due to the constraints of the number of pages.

Process modelling and enactment

Both software process and manufacturing process fields provide modelling support (languages, meta-models), with the same global objectives: process understanding and analysis, process improvement and process communicability. However both do not address process enactment and assistance support; this is specific to software process. Manufacturing process aims at supporting simulation, scheduling and process reengineering.

Control and quality

Both software process and manufacturing process fields tend to reach a fixed objective with a given quality. It seems also difficult to obtain relevant information (feedbacks) about process execution (i.e. performance). It is due to the fact that processes are human-intensive processes.
However there are some differences in the control support: software process engineering aims at providing reflexive approaches for describing several control policies. As for manufacturing process researchers, they identified a set of control policies that are already used in the manufacturing enterprises, and they have defined and characterised each of them.

Evolution

Both software process and manufacturing process fields address the process evolution problem. For manufacturing process we speak about scheduling, re-scheduling (for changing the process configuration) and we use predictive and reactive scheduling tools. As for the software process discipline it provides meta-models (reflexive approaches) and mechanisms that allow to evolve processes (using for instance process evolution processes).

Distribution

Both processes (software and manufacturing) are distributed. However, fragments process exchange is not possible in manufacturing engineering, as resources, skills and physical flows are material. At the opposite, software processes involves immaterial flows (electronic data, knowledge, etc.) that can be electronically distributed. On the other hand, in this domain, interactions have been formalised through adapted languages. This is still an isolated case in manufacturing process discipline.

What can we learn from each other?

The major differences that exist between software and manufacturing process fields are essentially due to the nature of the developed/manufactured product, in terms of product life duration and process instantiation frequency, electronic/material products and resources (computer vs. manufacturing machines). In manufacturing domain process is more frequently instantiated than in the software process. From each process instantiation result a set of products (ex: a root).

We surely have to learn from each other. Thus, we have seen in this paper that the two fields share several topics that are tackled differently: software process uses top down approach (starting from a global need) while manufacturing process mostly uses a bottom-up one (starting from an observed need). Software process formalisms and mechanisms could be used for modelling and enacting processes that were identified and characterised by manufacturing domain. This latest domain would benefit from process formalisation in terms of: manufacturing process models reasoning, manufacturing process enactment, etc. This paper is a first attempt, additional and deeper investigations are the subject of an ongoing work.

REFERENCES

Alloui I., Cîmpan S., Oquendo F., Verjus H., 2000. ALLIANCE : An agent-based case environment for entreprise process modelling, enactment and quantitative control, *Enterprise Information Systems,* Joaquim F.,Ed., Kluwer Academic Publishers.

Arbaoui, S., Barghouti, 1996. Process Support and Workflow management with Improvise. In *Proceedings of the NSF workshop on workflow and process automation in Information System*s, USA.

Arbaoui, S., Derniame, J.C., Oquendo, F., Verjus, H., 2002. A Comparative Review of Process-Centered Software Engineering Environments, *Annals of Software Engineering (ASE)*, Y. Wang, A. Bryant, Eds., Kluwer Academic publishers, VOL 14.

Arbaoui, S., Oquendo, F., 1993. Managing inconsistencies between process enactment and process performance states. *Proc. of the 8th International Software Process Workshop*, Warden (Germany).

Basili, V., R., Rombach, H.,D., 1987. TAME : Integrating Measurement into Software Environments, *Technical Report TR-1674,* TAME-TR-1-1987.

Basili, V.R., Green, S., 1994. Software Process Evolution at the SEL. *IEEE Software*, 58-66.

Belkhatir, N., Melo, W.L., Adam, J.M., 1992. TEMPO a Software Process Model Based on Object Context Behaviour. In *Proc. of the 5th International Conference on Software Engineering and its Applications*, France.

Berrah, L., Mauris, G., Foulloy, L., Haurat, A., 1998. Possibility/necessity semantics for the evaluation of manufacturing processes performance, *SMC'98*.

Bongaerts, L., 1998. Integration of scheduling and control in holonic manufacturing systems. *Doctoraat in die Katholieke Universiteit Leuven*.

Bruynooghe, R.F., Parker J.M., Rowles, J.S., 1991. PSS: a system for process enactment", In *Proc. of the lrst International Conference on the Software Process, Manufacturing Complex Systems*, IEEE Computer Society Press, Redondo Beach, CA.

Burlat, P., Vila, D., Besombes, B., Deslandres, V., 2001. Un cadre de modélisation des trajectoires d'évolution des groupements d'entreprises. *4ème congrès internationnal de Génie Industriel*, France.

Cannon, E., 1993. EDI GUIDE : a step by step approach, *International THOMSON computer press*.

Casati, F., Ceri, S., Pernici B., Pozzi G., 1996. WorkFlow Evolution. In *Proc. 15th Int. Conf. On Conceptual Modeling*, Cottbus, Germany: Springer

Cîmpan, S., Oquendo, F., 2000. OMEGA: a language and system for on-line monitoring of software processes. *ACM Software Engineering Notes*, Vol. 25, No. 4.

Cook, J.E., Wolf, A.L., 1998. Event-Based Detection of Concurrency, *Proc. of SIGSOFT '98 : Sixth International Symposium on the Foundations of Software Engineering*.

Cugola, C., 1998. Tolerating Deviations in Process Support Systems via Flexible Enactment of Process Models. *IEEE Transactions on Software Engineering*, VOL. 24, No. 11.

Cunin, P.Y., 2000. The PIE project: An Introduction. In *Proc. of the 7th European Workshop on Software Process Workshop Technology*, R. Conradi, Ed., LNCS 1780, Austria.

Dilts, D.M., Boyd, N.P., Whorms, H.H., 1991. The evolution of control architectures for automated

manufacturing systems, *Journal of Manufacturing Systems,* 10, 1.

Doumeingts, G., 1984. Méthode Grai : méthode de conception des systèmes en productique, *thesis (in french)*, University of Bordeaux I.

Dowson, M., 1992. Consistency Maintenance in Process Sensitive Environments. In *Proc. of the Process Sensitive SEE Architectures Workshop,* Boulder, VA.

Ellis, C. A., Nutt, G. J., 1996. Workflow : the process spectrum. In *Proc. of the NSF workshop on workflow and process automation in Information System*s, Georgie, USA.

Estublier, J., Verjus, H., Cunin, P.Y., 2001. Building Software Federation. In *Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications*, Las-Vegas.

Greenwood, M., Robertson, I., Warboys B., 2000. A Support Framework for Dynamic Organizations. In *Proc. of the 7th European Workshop on Software Process Technology*, R. Conradi, Ed., LNCS 1780, Kaprun, Austria.

Heimbigner, D., 1992. The ProcessWall: a Process State Server Approach to Process Programming. In *Proceedings of the 5th ACM/SIGSOFT Conference on Software Development Environments*, USA.

Hendrick, R., Kistler, K., Valett, J., 1994. Software management environment (SME) components and algorithms, *Technical Report SEL-94-001,* Software Engineering Laboratory.

Jain, A.S., Meeran, S., 1999. Deterministic Job-Shop Scheduling; Past, Present and Future. *European Journal of Operational Research*, volume 113, issue 2.

Janowski, T., Lugo, G.G., Zheng, H., 1998. Composing Enterprise Models: The Extended and the Virtual Enterprise. In *Intelligent Systems for Manufacturing: Multi-Agent Systems And Virtual Organizations*, Chapman and Hall.

Kaba, A.B., Derniame, J.C., 1996. Modeling processes for change: basic mechanisms for evolving process fragments. In *Proc. of the 5th. European Workshop on Software Process Technomogy,* LNCS 1149, C. Montangero, Ed., Springer-Verlag, France.

Kaiser, G.E., Ben-shaul, I.Z., 1993. Process evolution in the marvel environment. In *Proc. of the 8th International Software Process Workshop*, IEEE Computer Society Press, Germany.

Kuvaja, P., 1994. Software process assessment and improvement: The bootstrap approach. Blackwell, Oxford, UK.

Lee et al., 1998, The PIF Process Interchange Format and Framework. Version 1.2, *The Knowledge Engineering Review*, 13, 1, Cambridge University press.

Lehman, M., Belady. L.A., 1985. Program Evolution. *APIC studies in Data Processing*, Academic Press.

Madhavdji, N., 1992. Environment Evolution: The prism model of changes. *IEEE Transaction on Software Engineering*, vol. SE-18 (5).

Myers, K.L., Berry, P.M., 1998. Workflow management systems: An AI perspective. *Technical report, Artificial Intelligence Center*, SRI International, Menlo Park, CA.

Orlicky, J., 1975. Material Requirements Planning, *Mc Graw-Hill*.

Paulk, M.,C., Curtis, B., Chrissis, M.B., Weber, C.V., 1993. Capability Maturity Model for Software, Version 1.1. *Software Engineering Institute, CMU/SEI-93-TR-24, ESC-TR-93-177*.

Promoter, 1999. Software Process : Principles, Methodology, Technology. *J.C. Derniame, A.B. Kaba, D. Wastell (Eds), Springer-Verlag 1999, LNCS 1500, ISBN 3-540-65516-6199*.

Reichert, M., Dadam, P., 1998. ADEPTflex – Supporting Dynamic Changes of Workflows Without Loosing Control, *Journal of Intelligent Information Systems*, Special Issue on Workflow Management Systems, Vol. 10, No. 2.

Selby, R.W., Porter, A.A., Smith, D.C., Berney, J., 1991. Metric-Driven Analysis and Feedback Systems for Enabling Empirically Guided Software Development, *Proc.of the 13th International Conference on Software Engineering*, IEEE CS Press, Los Alamitos, USA.

SPICE, 1998. SPICE Project Phase 2. *Trials Interim Report Version 1.0*.

Swaminathan, J.M., Smith, S.F., Sadeh N.M., 1996. A Multi Agent Framework for Modeling Supply Chain Dynamics. *Technical Report*, The Robotics Institute, Carnegie Mellon University.

Tardieu, H., Theys, M., 1987. Système d'information et pilotage de l'entreprise. *Revue Internationale de Systémique*, vol. 1, n°4, 1987.

Theroude, F., Braesch, C., Haurat, H., 2002. How simulation could help in a controlled process model. *14th European simulation symposium (simulation in industry)*, Dresden Germany.

Vernadat, F., 1996. Enterprise modelling and integration, Principles and applications, *Chapman & Hall*.

Vollmann, T.E., Berry, W.L., Whybark, C.C., 1988. Manufacturing Planning an Control Systems. *Dow Jones Irwin*.

Warboys, B., 1994. Reflections on the Relationship Between BPR and Software Process Modelling. In *P. Loucopoulos editor, Business Modelling and Re-engineering proc. of the 13th International Conference on the Entity-Relationship Approach*, Manchester.

WFMC, 1996. WPDL. *Document WFMC TC-0020*, Workflow Management Coalition.