"The World and the Machine" A Critical Perspective on Process Technology*

Wolfgang Emmerich, Anthony Finkelstein
Interoperable Systems Research Centre
City University
London EC1V 0HB, UK
{emmerich | acwf}@cs.city.ac.uk

Carlo Montangero
Dip. di Informatica
University of Pisa
56125 Pisa, Italy
monta@di.unipi.it

Abstract

This short paper sets out a critical perspective process technology. It uses an analytical framework drawn from the work of Jackson as a means of identifying some important concerns and looks at the way research in the broad area of process technology might respond to these concerns. The paper is deliberately open and discursive.

1 Introduction

In the quest for a new research agenda for process technology, we apply the vision of software development, Michael Jackson set out in his ICSE-17 keynote talk [Jackson, 1995] on "The World and the Machine", to software process technology. Jackson identified the need for software engineers to balance the concern between the world, in which the machine they build serves a useful purpose, and the machine itself.

In process technology, machines are process support systems (PSSs). Examples of these are process-centred software engineering environments, workflow management systems or business process re-engineering systems. These machines are supposed to serve a useful purpose in processes, such as software production processes, workflow processes or business processes. These processes are the world in which PSSs operate. PSS designers are the equivalent of Jackson's engineers, who build machines.

Though we have applied our analysis to software process technology we would argue that similar exercises could usefully be undertaken for the other research domains to gather under the process technology umbrella and further that this might lead to a better understanding of the commonalities/diversities among the converging disciplines.

In the next section, we discuss the relationship between the world and the machine, in the case of software process technology. In doing so we take a fresh view of the core intuition underlying Lee Osterweil's seminal paper "Software Processes are Software Too" [Osterweil, 1987]. In Section 3, we review Jackson's four kinds of "denial" and find evidence for these denials in the development of software process technology. We then discuss the application of Jackson's "principles of description", which leads us to conclude that the current concerns in software process technology research are biased towards the "machine". We identify items in the "world" to be put on a research agenda in order to restore the balance.

*This work was performed while Carlo Montangero was an EPSRC Visiting Fellow at City University funded through Grant No. GR/L54561

2 Relationship between the World and the Machine

In his paper Jackson identifies four different facets of the relationship between the world and the machine, all of which apply directly to process technology. The *modelling facet* is concerned with the embodiment within the machine of a model of some aspect of the world. The *interface facet* is concerned with the shared phenomena through which the machine and the world interact. The *engineering facet* is concerned with the control the machine exerts on the world. Finally, both world and machine can be very complex, the *problem facet* is concerned with the relationships between the structure of the world and the machine. We now discuss the implications of adopting this division as an analytical framework for software process technology.

Modelling: Applied to software process technology, this suggests that the execution of processes in the real world should be separated from the representation of those processes within the PSS. Only recently have some process modelling approaches made this separation, most notably in LATIN [Cugola et al., 1995] and the Process Discovery approach proposed by [Cook and Wolf, 1995].

Interface: Sharing of phenomena in process technology occurs at the interface between the process performed in the world and the PSS. Shared phenomena can either be events that happen in the world and that should be known by the PSS, or they are the result of PSS reasoning that should be known by the world. Research on software process technology, in particular on PSEE architectures, has had a tendency to consider software engineering tools to be in the world, rather than in the machine. Hence research has identified shared phenomena passed between tools and process engines [Valetto and Kaiser, 1995, Emmerich et al., 1996] rather than shared phenomena between *the performers and the environment*. A notable exception is discussed in [Cugola et al., 1996].

Engineering: Jackson's framework allows us to distinguish between requirements, which identify the characteristics of processes in the world, process-centred environments, which implement the required support, and specifications, which identify the shared phenomena between the requirements and the programs. This critical distinction is rarely made in research on software process technology. The majority of techniques and languages have been developed for the purpose of writing enactable software process programs. There has been a lot of attention paid to generic requirements for process technology for process technology, such as process evolution, and a few software process specification schemes, such as [Montangero and Semini, 1996], but we are not aware of any attacks on the issue of project-specific software process requirements.

Problem: The structuring of processes is very complex because the processes incorporate quite a number of different, orthogonal aspects. Examples of these aspects are configuration management, process management and process change. It has been recognised for a long time in software process research that any attempt to describe these in a hierarchical and homogeneous way is bound to fail and that multi-paradigm process descriptions are needed [Deiters et al., 1989]. Though this is known and understood it has received little attention. Some interesting directions are suggested by work on standards, such as ISO-12207 [ISO/IEC, 1995], that are organised around a number of aspects or viewpoints. These aspects refer to different overlapping subsets of the phenomena of the world, which are perceived as meaningful and manageable views of the problem.

3 Four Kinds of Denial

Denial has to do with how we evade the responsibility that, according to Jackson, software engineers have undertaken, to deal with the part of the world that furnishes the context for the machines we design, and how we justify this evasion. The only legitimate denial of the need to analyse the world is denial by *prior knowledge*, which is applicable when the requirements are well-understood and standardised, as in cases of well established engineering practice, automobile engineering, and so on. On the illegitimate side, we have denial by *hacking*, where the world is disregarded, because of the fascination of the machine. There is a long tradition here, that leads us to offer our customer representations of the machine in place of the description and analysis of their problems and needs. Denial by *abstraction*,

occurs when the world is disregarded, because of the fascination of the abstract mathematical nature of computations. According to Jackson, this occurs mainly in education, where it is implicitly taught that software development problems can be captured in few words, and that all the difficulty is in devising a solution. Finally, denial by *vagueness* occurs when descriptions of the machine are offered, but with the vague implication that they are descriptions of the world. We now discuss the extent to which denial in these forms manifests itself in software process technology.

Prior Knowledge: Software process technology provides support for software engineering. One might argue, that we have a fast track to understand software process as we are software engineers ourselves. In addition, there are standard process definitions, such as ISO-12207 [ISO/IEC, 1995], IEEE 1074 [IEEE, 1995] and PSS-05 [Mazza et al., 1994], that are widely applied. It might be concluded that the requirements associated with software process technology do not have to be made explicit. It must, however, be questioned whether these perceptions are really valid. If, for instance, the software process community had achieved prior knowledge, i.e. a common understanding of concepts, definitions and requirements, the definition of the glossary in [Derniame et al., 1998] would not have been as difficult as it proved to be. Moreover, software development processes in academia, although they might deliver impressive prototypes, are considerably different from software production in industry and these differences are poorly understood. How many academic PSEEs, for instance, have been constructed by applying standard processes?

Hacking: Rather than trying to understand the world of software processes, software process technology research have focussed on the features of languages and their interpreters. We ourselves have come to the temptation of offering customers our modelling schemes in place of listening to their needs [Emmerich et al., 1996]. The development of Harel's Statecharts are a good counter-example. They have been developed starting from an appreciation of the users' needs and practices and have then be extended to a fully formal specification language. For software process technology this may mean taking a fresh look at the process representations that are used in the actual project management. Some suggestions are given in [Emmerich et al., 1997].

Abstraction: Software process technology may seem almost immune from this kind of denial, although the ISPWx series of exercises shows some signs of it: the answers to the original ISPW6 problems have mostly been used to demonstrate how the machine was specified, not how the problem was attacked, the collected answers have been almost forgotten, and many 'simplified' versions of the problem series are only used to demonstrate new features of the machine descriptions. Other cases in which denial by abstraction shows up are [Montangero and Semini, 1996, Cugola et al., 1996]. Both take into account the real world, but at such a high level of abstraction that their utility must be questioned; the former by falling into the trap of using too simple a case study, the latter by introducing a model of the real world which is to admit any specific link with process technology.

Vagueness: This form of denial occurs in process technology as the few applications tend to make great play of those parts of the software process that were automated in a process program, while they neglected the (probably more significant) subprocesses that were not amenable to automation. In the example reported in [Emmerich et al., 1996] it was claimed that a C++ class library management process was captured, modelled and improved. In the light of Jackson's observations it must be admitted that this is a form of vagueness as, in the end, only the automated part of the process was modelled.

4 Principles for Descriptions

Further hints for a research agenda for process technology can be obtained by looking at the descriptive principles gathered by Jackson. From an analysis of *Von Neumann's* theory of games, Jackson identifies the need to clarify the concepts and issues relevant to a system by establishing the vocabulary and identifying the phenomena of interest using informal but rigorous rules to recognise them in the world. The principle of *reductionism* suggests starting with those phenomena for which we can give the most exact and reliable recognition rule. *Shanley's* principle is the direct negation of separation of concerns. It is best exemplified by an application to rocketry engineering where the fuel pressure inside the tank

is used to improve the rigidity of the external rocket skin. In essence, Shanley's principle demands the parallel structuring of views of and problems in the world. Finally, *Montaigne's* principle requires that we clearly distinguish between the *indicative* and the *optative* mood of descriptions, i.e between what we assert to be true, and what we desire to be true. Below we briefly examine what these principles mean for descriptions of processes and their implications for the process research agenda.

von Neumann's Principle and the Reductionism Principle: These basic principles suggest a rather natural modelling strategy. However, they have not received the attention of the process research community that they deserve. Attempts have been made to define the concepts and the terminology used for software process [Lonchamp, 1993] and to clarify the distinction between the world and the machine [Dowson and Fernström, 1994]. These do not yet approach the the analytic recognition rules demanded by Jackson.

Shanley's Principle: An example of this parallel structuring is the structure of software development standards discussed above. These try to capture the world of software development in different views, which share people and documents. The problem how these views and their interactions are captured in a more precise way that fully exploits Shanley's principle has yet to be tackled. In Jackson's terms, we are looking for a global model that relates to the parts it is composed of, just like a colour picture relates to the Cyan, Magenta, Yellow and Black colour separations. The extent to which approaches like ViewPoints [Nuseibeh et al., 1994] are appropriate for this purpose has yet to be assessed. Such approaches also seem to be a prerequisite for the effective application of the framework in [Cugola et al., 1996] to full blown standard processes.

Montaigne's Principle: It is easy to find examples in the area of software process technology where the distinction between indicative and optative have been slurred. The same concern, for instance configuration management, can be treated both optatively when analysing repository structures and indicatively when treating project management support. This is not in itself a problem but can lead to difficulties if not made explicit.

5 Recommendations

Our recommendation is, that we as a community perform a serious analysis of the balance of world versus machine concerns in software process technologies, and that the same exercise is performed for related technologies such as workflow and computer supported cooperative work. If, as we expect, the results show that the imbalance is always towards the machine, a major item in the agenda should be to restore the correct balance everywhere, before attempting to merge the technologies. In any event, the exercise should provide a good characterisation of the new discipline. For instance, knowledge might be collected about the differences between different kinds of human-centred systems, such as workflow management systems, which are controlling the flow of many instances of rather simple documents, and those, like process-centred environments, which are mainly concerned with the design of a few issues of a very complex body of documents. Such a clarification may then deliver an assessment of the feasibility of merging workflow and software process technology.

References

[Cook and Wolf, 1995] Cook, J. E. and Wolf, A. L. (1995). Automating Process Discovery through Event-Data Analysis. In *Proc. of the* 17^{th} *Int. Conf. on Software Engineering, Seattle, Washington*, pages 73–92. ACM Press.

[Cugola et al., 1995] Cugola, G., Di Nitto, E., Ghezzi, C., and Mantione, M. (1995). How To Deal With Deviations During Process Model Enactment. In *Proc. of the* 17^{th} *Int. Conf. on Software Engineering, Seattle, Washington*, pages 265–273. ACM Press.

[Cugola et al., 1996] Cugola, G. P., Di Nitto, E., Fuggetta, A., and Ghezzi, C. (1996). A Framework for Formalizing Inconsistencies in Human-Centred Systems. *ACM Transactions on Software Engineering and Methodology*, 5(3).

[Deiters et al., 1989] Deiters, W., Gruhn, V., and Schäfer, W. (1989). Process Programming: A structured Multi-Paradigm Approach Could be Achieved. In *Proc. of the* 5^{th} *Int. Software Process Workshop*, pages 54–57. IEEE Computer Society Press.

[Derniame et al., 1998] Derniame, J. C., Warboys, B., and Kaba, A., editors (1998). *Software Process: Principles, Methodology, Technology*. To appear.

[Dowson and Fernström, 1994] Dowson, M. and Fernström, C. (1994). Towards Requirements for Enactment Mechanisms. In Warboys, B., editor, *Proc. of the* 3^{rd} *European Workshop on Software Process Technology, Villard-des-Lans*, volume 772 of *Lecture Notes in Computer Science*. Springer.

[Emmerich et al., 1996] Emmerich, W., Bandinelli, S., Lavazza, L., and Arlow, J. (1996). Fine grained Process Modelling: An Experiment at British Airways. In *Proc. of the* 4^{th} *Int. Conf. on the Software Process, Brighton, United Kingdom*, pages 2–12. IEEE Computer Society Press.

[Emmerich et al., 1997] Emmerich, W., Finkelstein, A., Montangero, C., and Stevens, R. (1997). Standards Compliant Software Development. In *Proc. of the ICSE Workshop on Living with Inconsistency, Boston*. To appear.

[IEEE, 1995] IEEE (1995). *IEEE Standard for Developing Software Life Cycle Processes (1074-1995)*. IEEE Computer Society Press.

[ISO/IEC, 1995] ISO/IEC (1995). *International Standard, Information Technology Software Life Cycle Process. 12207*.

[Jackson, 1995] Jackson, M. (1995). The World and the Machine. In *Proc. of the* 17^{th} *Int. Conf. on Software Engineering, Seattle, Washington*, pages 283–292. ACM Press.

[Lehman, 1996] Lehman, M. M. (1996). Laws of Software Evolution Revisited. In Montangero, C., editor, *Proc. of the* 5^{th} *European Workshop on Software Process Technology, Nancy, France*, volume 1149 of *Lecture Notes in Computer Science*, pages 108–124. Springer.

[Lonchamp, 1993] Lonchamp, J. (1993). A structured conceptual and terminological framework for software process engineering. In *Proc. of the* 2^{nd} *Int. Conf. on the Software Process, Berlin, Germany*, pages 41–53. IEEE Computer Society Press.

[Mazza et al., 1994] Mazza, C., Fairclough, J., Melton, B., De Pablo, D., Scheffer, A., and Stevens, R. (1994). *Software Engineering Standards*. Prentice Hall.

[Montangero and Semini, 1996] Montangero, C. and Semini, L. (1996). Applying Refinement Calculi to Software Process Modelling. In *Proc. of the* 4^{th} *Int. Conf. on the Software Process, Brighton, UK*, pages 63–74. IEEE Computer Society Press.

[Nuseibeh et al., 1994] Nuseibeh, B., Kramer, J., and Finkelstein, A. (1994). A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification. *IEEE Transactions on Software Engineering*, 20(10):760–773.

[Osterweil, 1987] Osterweil, L. J. (1987). Software Processes are Software Too. In *Proc. of the* 9^{th} *Int. Conf. on Software Engineering, Monterey, Cal.*, pages 2–13. IEEE Computer Society Press.

[Valetto and Kaiser, 1995] Valetto, G. and Kaiser, G. (1995). Enveloping "Persistent" Tools for a Process-Centred Environment. In Schäfer, W., editor, *Proc. of the 4th European Workshop on Software Process Technology, Nordwijkerhout, The Netherlands*, volume 913 of *Lecture Notes in Computer Science*, pages 200–204. Springer.

[Votta and Porter, 1995] Votta, L. G. and Porter, A. (1995). Experimental Software Engineering: A report on the state of the art. In *Proc. of the* 17^{th} *Int. Conf. on Software Engineering, Seattle, Washington*, pages 277–279. ACM Press.