SOME RESERVATIONS ON SOFTWARE PROCESS PROGRAMMING

M M Lehman

Imperial Software Technology Limited

In what follows the term 'process programs' is used as shorthand for the stated intended workshop focus 'executable and interpretable (surely 'interpretable and executable'?) models of the software (development?') process and their prescriptive application to directly controlling software project activities'.

At the recent ICSE9 conference [LEH87] I indicated my strong reservations about process programs and the role they could or should play in software development. I questioned whether their pursuit or development would yield more insight into the software development process, produce better understanding of that process or lead to its significant improvement. I also expressed some concern at popularisation of process programming. These reservations and concerns have been intensified by the workshop announcement which appears to reflect the very euphoria I feared. There is no need to repeat the details here and I use this opportunity to briefly explore some underlying and intrinsic reasons for my attitude and concern.

Process programming is, unquestionably, one approach to process modelling. Superficially it appears to have an advantage over other, less formalised, approaches in that its models can be machine interpreted and can, therefore, be used as a process control mechanism. It may for example, be suggested that a program driven mechanism can be used to select and invoke the application of a sequence of IPSE tools. The IPSE itself could then be tuned to the needs of a particular application of the process by preparing and loading an appropriate process program or by adjusting parameters in a program already loaded. Any benefit from an implementation of this approach would, however, be virtual rather than real. The implementors, the manager responsible for progress and the software engineer {LEH86] responsible for the dynamic process composition would have to intervene on completion of each constituent activity to select the next action on the basis of circumstance dependent considerations that cannot be predicted; that must be determined in real time. Indeed, where such intervention is considered to be never necessary, tools being used, and the actions they implement, would be coupled and comprises an entity. In so far as a, so called, process program determines that coupling it simply represents a part of a more complex tool.

With this view the process program is seen to comprise a series of calls for specific actions with a human having to take the decision as to which action - and tool - is next to be invoked.

This situation arises because the process is heavily context dependent. The direction and nature of further progress from any stage is a function of progress already made and how it was achieved; on the nature and properties of the continuously developing process product and on the essential and central role of human creativity in the creation of that product. The process followed to implement a particular program or software system includes intrinsic inter-dependencies between different steps or actions in the process as well as dependencies on the problem or application domain, the solution approach adopted and human input, creative and otherwise, during process execution. Detailed process structure and composition cannot be predicted. Both are dynamic, determined and controlled amongst other factors, by multi-loop feedback paths and effects. Process descriptions, whether formal or informal, are essentially imprecise and non-deterministic.

Other, equally important, fundamental problems with process programs arise because computer application for which software is developed occurs, in general, in an essentially continuous and infinite application domain. Any digital computer-based process model is discrete and finite. Abstraction, discretisation and data selection must occur during

0-89791-314-0/88/0005/0111 $1.50

program creation, though binding may, sometimes, be delayed till initiation of each process step. A 'program' based model limits, therefore, the scope and power of what can be achieved. The process it represents differs fundamentally, from human controlled processes where the context dependent abstraction and discretisation necessary during process execution occurs, in general, when required rather than when planned. That process can take into account all circumstances relevant to progress at the time when decisions for further action must be taken, when binding must occur. Undoubtedly controls are then required to discipline and direct the human contribution. Models can and must play a significant role in the development and implementation of such controls and the tools that mechanise them. But that is a different, more limited role than that envisaged by the workshop announcement.

On the basis of this outline analysis one may question many of the implied assumptions underlying the workshop announcement and develop specific answers to the questions posed. For example, interest should be focussed on GOOD processes for *execution*, not on 'GOOD processes for *modelling*'. I question the validity, at this point in time, of a search for *requirements* for prescriptive models or that formalisms are *needed*, as distinct from possibly being *useful* in a limited way. Formalisation and mechanisation do not *enhance* human intelligence. Formalism can provide *support* for human understanding; mechanisation will reduce or replace such human repetitive activity for which the need and make-up is predictable. And so on.

Equally, answers to some of the questions as posed in the call are clear. In my judgement it is unquestionable, for example, that there are limits, severe limits, to which it is practical or desirable to automate the software (development) process arising both from technological and sociological issues. It is to be hoped that the workshop will consider real issues and not follow philosophical willow-the-wisps.

Process models of whatever sort must not become an obsession. Their prime purpose is to aid human investigation, achievement of understanding and process design and refinement. They will also prove useful in a limited way to guide IPSE and tool architecture and to help control IPSE usage. In the latter role they will undoubtedly use active information retrieval mechanisms which may include inferential properties operating on data in the IPSE information repository.

[LEH86] Lehman M M ' Advanced Software Technology - Development and Introduction to Practice', Info. Proc. '86, Proc. IFIP Congr. 1986, Dublin, Sept. 1-5, Elsevier Sci. Pub. (North Holland), 1986, pp. 605-611

[LEH87] idem., 'Process Models, Process Programs, Programming Support - Invited Response To A Keynote Address By Lee Osterweil', Proc. 9th Int. Conf. on Softw. Eng., Monterey, CA, 30 March - 2 Apr. 1987, IEEE Comp. Soc. pub. no. 767, IEEE Cat. no. 87CH2432-3, pp. 14 - 16