An Evolutionary Approach to Process System Development

R. Mark Greenwood, Ian Robertson, Brian C. Warboys, and Ben S. Yeomans

Informatics Process Group (IPG) Department of Computer Science
University of Manchester, Oxford Road, Manchester, M13 9PL, UK.
{markg,ir,brian}@cs.man.ac.uk

Abstract. When a process system is adopted and used by any organization, the way in which improvements in the process system are delivered to its users has to be addressed. Through our experience we have adopted a particular approach for our Process*Web* system: the system can rebuild itself in an evolutionary manner. In this paper we identify the factors which have led to this approach. This raises some general issues which are relevant to the whole process community.

1 Background

Over the past decade the Informatics Process Group have developed a number of enactable process models, both as demonstrations of the capability of such technology, and to teach students. Our background is in software process modelling, in particular the IPSE 2.5 project [7], but the majority of more recent work has been in the area of general business process modelling [9].

Initially our enactable models were developed using ICL's ProcessWise Integrator (PWI)[1] [2], but more recently we have extended PWI with a Web interface to give our Process*Web* system [3, 10]. The original goal of Process*Web* was to exploit the Web as an interface to a process enactment system, and in this it has been successful. It has also prompted other changes; in this paper we review the evolutionary approach to developing Process*Web*.

With our older PWI-based models, the typical situation was for each model to have its own copy of the PWI system. The developers' model would specialise the PWI system to the particular process to be supported, and a process specific system would be created. There was a clear separation between the model and the PWI system; the interface being PWI's PML language used for writing the models. While it was possible for PWI to enact multiple processes, this was not done because there was little benefit. The PWI system itself was occasionally updated, in a typical versioned fashion, and models had to be recompiled for the new version.

With Process*Web* there is an integrated development environment within which new models are created. The Process*Web* system is a service which supports multiple users involved in multiple enacting processes[2]. It extends the PWI system with standard facilities which include: handling the interface with Web browsers, the logging on and

[1] ProcessWise Integrator has, more recently, transferred from ICL to TeamWARE

[2] Details of the Process*Web* demonstration service are available at http://processweb.cs.man.ac.uk/

off of users, managing libraries of models, and deleting models which are no longer required. There is no longer the clear separation between system and model. There is the basic system, libraries of standard facilities, and the models. In this manner modellers have a richer set of facilities available to them, and as more models are developed opportunities for new or improved standard facilities emerge.

The overall effect is that we want to deliver a new enhanced Process*Web* much more frequently than the once or twice per annum that PWI was updated. However, because Process*Web* may be supporting several, distinct, long-running process enactments it is organizationally (rather than technically) more difficult to regularly rebuild the system from scratch.

2 Architecture and Libraries

The demand to rebuild Process*Web* regularly comes from the improvements of the standard facilities which it provides on top of PWI. These are generic facilities which are not specific to any model, and thus are opportunities for re-use between process models. These facilities can be categorized into two types: architecture facilities and libraries. This separates facilities which are related to the specifics of the Process*Web* architecture, from other more general facilities.

Architecture facilities represent conventions which models follow to fit within the Process*Web* environment. A couple of examples will illustrate the general issues involved. The "User Role" abstracts from a model the detail of communication with the Web browser and server. Output from the model is given to a "User Role". This caches the output and sends it to the appropriate browser when required. As the output is cached, the "User Role" handles the situations where it needs to be redisplayed. This can occur in normal execution where the user has moved the browser to another page and later returns to the model, or in error conditions where some communication failure means that the output must be refreshed. Input from the browser uses the standard Web CGI (Common Gateway Interface) mechanisms. The detail of this is hidden from the model. It simply receives, from a "User Role", a set of name-value pairs whenever input is supplied by a user's browser. By using the "User Role" facility a model fits within the Process*Web* architecture, and can ignore the details of the communications involved. A second example of an architectural facility is the "Housekeeper". Models create a "Housekeeper" which is used to record the information needed to delete a model when it is no longer required. (This information could also be used to evolve the model.) The "Housekeeper" copes with the situation where a model fails to complete successfully. This is of particular importance in Process*Web* since removing a failed model by reloading the system from scratch is not a viable option.

Library facilities are generic fragments which may be useful in many models. In Process*Web* examples include: string manipulation facilities, routines for formatting output in a standard manner, and routines for interfacing with an Oracle database. These last two examples indicate that some libraries may be defined by the organization which is developing process models, rather than by the developers of the process enactment system. In terms of input and output to users, it may be that the organization wants to adopt its own standards of good practice across models. In our current Process*Web*

system we have a set of routines which are specialised for accessing the Oracle database system at the University of Manchester. This was developed from some specific research investigating the linking of a process system (Process*Web*) and a conventional database (Oracle). To date there has been no demand for more generic database access routines.

It would be possible to collect together the basic system, and the standard facilities, both architecture and libraries, as a new language for writing Process*Web* models. However this would make it more difficult for an organization, or another community of users, to share libraries. A further disadvantage of this approach is that it creates a false impression of confidence that the correct standard facilities are available. Initial estimates are that the rate of change of standard facilities is at least an order of magnitude more than for the basic system.

3 System Rebuild

Our current approach to evolving Process*Web* is to have a "System Rebuild" model which supports changing the system itself. New, or more often newly modified, code is compiled into the system. The model knows the system dependencies and so ensures that other related code is also re-compiled to check that it is compatible. Once this is done new models will pick up the latest definitions in the usual way. The rebuild model also enables the existing architecture objects, (e.g. "User Roles") to be updated if required. In essence the "System Rebuild" process is a configuration management system for the Process*Web* system.

The details of how the system rebuild is performed depends on the particular facilities available within the PWI system, and are not significant here. In general terms the key facilities are: the ability to "suspend and change" the implementation of an object without any disturbance its interface on which other objects depend, and the ability to locate the objects which need to be changed.

It is worth noting that "System Rebuild" was not an essential part of the Process*Web* design. It was developed because of the need to make updating the system easier. It has some privileged facilities, being able to identify and update architecture objects which are hidden from normal models, but in most other ways it is a general enactable process model which supports its user. The current simple model is:

1. Obtain input from the user identifying the new Process*Web* sources.
2. Compile the new sources, and re-compile those which depend upon them. If there are errors, report them and return to 1.
3. Query the current Process*Web* state and identify the objects affected by the newly-compiled changes.
4. Accept input from the user confirming that the changes should be applied, or requesting a return 1. If the attempt to apply the changes fails, report the errors and return to 1.
5. Confirm update completed. Return to 1 to await next change.

This model could be elaborated to implement a partial change which was then tested and fully applied if approved. It could also record information of the nature and time of changes for future reference.

The notion of a process support system which has the ability to adapt itself to new demands is not unique. Endeavours keeps an internal model of itself which can be used by enactable models supporting the development, installation, customizing and reconfiguring of the system [1]. Amber has experimented with a metalinguistic approach, where callbacks to instance specific code is used to modify the systems default enactment behaviour [5]. However, most emphasis has been on the adaption of a process system to the demands of a specific model, rather than to the demands of several models over a sustained period of time.

4 Conclusions

The system rebuild facility for Process*Web* has emerged through a series of pragamatic choices as the system has been both used and enhanced over the past couple of years. The general issue of system development which it illustrates is relevant across the whole spectrum of process technologies.

- How are system improvements, including bug fixes, delivered to users?
- If a process enactment fails, say half way through a 12 month process, what should be done?
- What rate of system improvements is appropriate for users?
- When standard facilities are used to promote re-use between models, what are the implications for the distributed heterogeneous process systems?

These issues are closely related to those which emerge when delivering a process system with new versions of standard processes, to customers who have made their own modifications to an earlier version [4].

Our background in software process modelling and interest in evolutionary approaches to the design of large, long-lived systems makes us biased towards an evolutionary approach. The CADES environment has been used for many years to support the process of developing the VME operating system [6]. Its more general impact was limited because of the problems involved in adapting it to cope with new methods and tools [7]. This has influenced out preference for an "inside-out" approach [8], where a process system being developed is the means of its own development. In contrast the traditional "outside-in" approach considers the development process as separate from the system which is being developed. Do process systems in general need to have the means to develop themselves, or does it depend on the processes which they are going to support?

The introduction of process technology is rarely separated from wider business process changes. In many circumstances the exact effect of the new technology is difficult to predict in full, and exaggerates rather than dampens the other pressures for changes. In this environment the way in which process technology itself changes to cope with new demands can have a significant impact on how well it supports business changes.

Acknowledgements

Our work on Process*Web* (`http://processweb.cs.man.ac.uk`) has been supported by the UK EPSRC, projects GR/L34433 and GR/L46229, and the European Comission, Esprit-IV PIE project and PROMOTER working group.

References

1. G.A. Bolcer and R.N. Taylor. Endeavors: A Process System Integration Infrastructure. In *Proceedings of the Fourth International Conference on the Software Process*, pages 76–85, Brighton, UK, December 1996. IEEE Computer Society Press.
2. R.F. Bruynooghe, R.M. Greenwood, I. Robertson, J. Sa, R.A. Snowdon, and B.C. Warboys. PADM: Towards a total process modelling system. In A. Finklestein, J. Kramer, and B. Nuseibeh, editors, *Software Process Modelling and Technology*, pages 293–334. Research Studies Press, 1994.
3. R.M. Greenwood and B.C. Warboys. ProcessWeb - Process Support for the World Wide Web. In Carlo Montangero, editor, *Software Process Technology - Proceedings of the* 5^{th} *European Workshop*, pages 82–85, Nancy, France, October 1996. Springer-Verlag. Lecture Notes in Computer Science 1149.
4. V. Gruhn. Configuration Management Support for Software Process Models. In Carlo Montangero, editor, *Software Process Technology - Proceedings of the* 5^{th} *European Workshop*, pages 132–136, Nancy, France, October 1996. Springer-Verlag. Lecture Notes in Computer Science 1149.
5. G.E. Kaiser, I.Z. Ben-Shaul, and S.S. Popovich. A Metalinguistic Approach to Process Enactment Extensibility. In *Proceedings of the Fourth International Conference on the Software Process*, pages 90–101, Brighton, UK, December 1996. IEEE Computer Society Press.
6. B.C. Warboys. VME/B a model for the realisation of a total system concept. *ICL Technical Journal*, 2(2):132–146, November 1980.
7. B.C. Warboys. The IPSE 2.5 project: Process modelling as the basis for a support environment. In *Proceedings of the First International Conference on Software Development, Environments and Factories*, Berlin, 1989. Pitman Publishing.
8. B.C. Warboys. The Practical Application of Process Modelling - some early reflections. In A. Fugetta, R. Conradi, and V. Ambriola, editors, *First European Workshop on Software Process Modelling*, pages 255–264, Milan, Italy, May 1991. published by A.I.C.A.
9. B.C. Warboys, P. Kawalek, I. Robertson, and R.M. Greenwood. *Business Information Systems: a Process Approach*. McGraw-Hill, 1999.
10. B.S. Yeomans. Enhancing the World Wide Web. student project report, 1996.