Decentralised Process Modelling*

Bashar Nuseibeh Jeff Kramer Anthony Finkelstein Ulf Leonhardt

Department of Computing
Imperial College
180 Queen's Gate
London, SW7 2BZ, UK
Email: {ban, jk, acwf, ul}@doc.ic.ac.uk

Abstract. In this paper, we advocate decentralised process modelling and suggest that understanding and modelling the development processes of individual development participants is the key to supporting collaborative development. Our approach relies on recognising individual developers' states ("situations") by analysing local development histories. Different situations can be used to trigger a variety of further development actions, such as consistency checks between process models of different development participants. We report on experience using regular expressions to specify particular situations and rules to associate actions with these situations.

1. Motivation and Background

A significant proportion of large software development projects involve the participation and collaboration of many development participants, who in turn may be physically distributed. Software process modelling and technology address a wide range of issues surrounding the specification and development of complex systems, including the description of the activities by which software is developed, and supporting this with automated tools [6].

While an understanding of the processes by which software systems are developed is valuable, we believe that understanding and describing "fine-grain" software processes is equally worthwhile, and an effective approach to tackling many of the problems associated with decentralised development is therefore particularly useful. By "fine-grain" we are referring to *(a) developer level* processes which describe the activities of individual development participants rather than organisations, and *(b) representation level* processes which can manipulate elements of representation schemes (e.g., specification languages) rather than treating them as "vanilla" objects - whose structure and content is irrelevant or unknown [11].

To perform fine-grain process modelling and reap its benefits, many issues of decentralised software development need to be addressed. These include the specification of coordination behaviour between both individual development participants and teams of developers. Ben-Shaul and Kaiser [1] for example, adopt an "international alliance" metaphor in which participating "countries" adhere to "treaties" (c.*f*. process models) and engage in "summits" (at which process models are enacted). This approach addresses the broad problem of decentralised development between teams of developers rather than individual participants. Engels and Groenewegen [3] propose an approach to specifying the coordinated behaviour of different "objects" using a formalism called Paradigm - which, in

* (to appear in) Proceedings of EWSPT '95, Noordwijkerhout, Holland, 3-5th April 1995, Springer-Verlag.

turn, uses a special kind of state transition diagrams. In our approach, described briefly below (and in more detail in [9]), we also use state transition diagrams to describe individual developer processes.

2. An Approach to Decentralised Process Modelling

Approach. Our approach is to represent individual process models locally by associating them with individual development participants. For each process, we maintain an ordered *work record* of development actions which also defines a particular sequence of process *states*. These states can be used to identify appropriate courses of actions. Typically, a specific course of actions will be appropriate not only for one state but for a set of similar states (which we call a *situation*)[1]. This "decision knowledge" can be expressed by *rules* which map situations to actions:

```
<situation><course of action>
```

where a `<situation>` is the pre-condition of the rule, and a `<course of action>` specifies what should be done once a decision has been made (we have omitted post-conditions for simplicity - these could be used in more elaborate applications, such as planning). In other work [4], we adopted a similar approach to inconsistency handling in multi-perspective specifications, in which we used rules of the form "`Inconsistency implies Action`" to specify how to behave in the presence of inconsistency.

Finally, enacting process models elicits one of three kinds of responses (courses of action): *informal guidance*, which may include help text, video clips, etc.; *specific recommendations*, which include a limited set of actions that a developer is advised to select/perform; and *automatic execution of specific actions*, which are only performed if a developer is aware of the consequences of these actions and is prepared to relinquish control over their execution. These three kinds of responses reflect the amount of knowledge developers have in any particular situation, and the degree of automation they wish to adopt/impose on a development process.

Implementation. We have implemented our approach by treating process models as finite state machines, represented in terms of regular expressions (thus allowing us to make use of a variety of powerful and efficient tools, which in turn facilitate the prototyping of tool support). Enacting our process models causes the work record associated with each process to be updated. These process models also analyse their respective work records in order to recognise (match) particular situations (states). Decision rules can then be used to trigger the appropriate response (action(s)).

Associated with our general approach is a communication protocol that supports asynchronous message passing. This is a necessary interaction infrastructure to support the decentralised process modelling we propose, and to facilitate distributed consistency management in this setting. The reader is referred to [9] for details.

Example. To demonstrate our approach, we introduce the notion of *tests*, which are acceptor automata defined by regular expressions. A sample test, T_A, can be expressed in the form:

```
T_A:    .*D[^R]*$   not-successfully-checked-since-D
```

The regular expression in this test uses the basic constructs of regular expressions as used in Lex [13]. Thus, this test is matched if a `D`-event, but no subsequent `R`-event, can

1 The term *situation* is a variation of the term used in the NATURE project's process meta-model [8].

be found in the local (developer) work record. D and R denote actions or communication events that are part of a local process model.

A sample rule, R_1, maps a situation to a response:

```
Rule:       R1

Situation:  TA ∧ ¬TB ∧ ¬TC

Response:   recommend: name-clash-check
```

where the situation in this case is a logical proposition combining a number of tests (T_A, T_B and T_C), which, in turn, is a pre-condition for the response - a recommendation to perform the consistency check `name-clash-check` (other responses include `display: <message>` or `do: <action>`).

3. Evaluation and Summary

Process modelling is the construction of abstract descriptions of the activities by which software is developed [5]. While global process models may be useful for describing global system development activities, we have suggested that a decentralised approach is more representative, and ultimately more useful, for modelling distributed and concurrent activities in collaborative software development projects. Individual developer process models are easier to construct, and better vehicles for the provision of development guidance.

Having reaped the benefits of decomposing complex global models into simpler local ones, we further proposed an approach for coordinating the activities of these different models, by managing consistency checks between these models. Such coordination is achieved by deriving development states from individual developer process models, and then using these to trigger the consistency checks, which proceed according to a well defined communication protocol.

We have implemented a simple prototype tool in Objectworks/Smalltalk™ which demonstrates our approach. The tool illustrates how states are derived from individual developer histories, and then demonstrates our protocol of communicating state machines to drive the global development process. We intend to integrate this prototype with *The Viewer* [10], an environment supporting the ViewPoints framework [7], which we have developed to support multi-perspective development. In particular, we would like to build on our work of expressing the relationships between multiple ViewPoints [12], in order to facilitate the process of coordination and collaboration in this setting [2].

Acknowledgements

We would like to thank Michael Goedicke for his feedback on our work. This work was partly funded by the Department of Trade and Industry as part of the ESF project, the UK EPSRC as part of the VOILA project, and the European Union as part of the ESPRIT Basic Research Action PROMOTER and the ISI project. A variety of papers describing our work are available by anonymous ftp from dse.doc.ic.ac.uk in directory dse-papers.

References

[1] Ben-Shaul, I. Z. and G. E. Kaiser (1994); “A Paradigm for Decentralised Process Modelling and its Realization in the Oz Environment”; *Proceedings of 16th International Conference on Software Engineering (ICSE-16)*, Sorrento, Italy, 16-21 May 1994, 179-188; IEEE Computer Society Press.

[2] Easterbrook, S., A. Finkelstein, J. Kramer and B. Nuseibeh (1994); "Coordinating Distributed ViewPoints: The Anatomy of a Consistency Check"; *Concurrent Engineering: Research and Applications*, 2(3): CERA Institute, USA.

[3] Engels, G. and L. P. J. Groenewegen (1992); "Specification of coordinated behaviour in the Software Development Process"; *Proceedings of Second European Workshop on Software Process Technology (EWSPT '92)*, Trondheim, Norway, September 1992, 58-60; LNCS, 635, Springer-Verlag.

[4] Finkelstein, A., D. Gabbay, A. Hunter, J. Kramer and B. Nuseibeh (1994); "Inconsistency Handling in Multi-Perspective Specifications"; *Transactions on Software Engineering*, 20(8): 569-578, August 1994; IEEE Computer Society Press.

[5] Finkelstein, A., J. Kramer and M. Hales (1992); "Process Modelling: A Critical Analysis"; *(In) Integrated Software Engineering with Reuse: Management and Techniques;* P. Walton and N. Maiden (Eds.); 137-148; Chapman & Hall and UNICOM, UK.

[6] Finkelstein, A., J. Kramer and B. Nuseibeh (Eds.) (1994); *Software Process Modelling and Technology*, Advanced Software Development Series, Research Studies Press Ltd. (Wiley), Somerset, UK.

[7] Finkelstein, A., J. Kramer, B. Nuseibeh, L. Finkelstein and M. Goedicke (1992); "Viewpoints: A Framework for Integrating Multiple Perspectives in System Development"; *International Journal of Software Engineering and Knowledge Engineering*, 2(1): 31-58, March 1992; World Scientific Publishing Co.

[8] Jarke, M., K. Pohl, C. Rolland and J. Schmitt (1994); "Experience-Based Method Evaluation and Improvement: A Process Modeling Approach"; *Project report*, 94-15; ESPRIT Nature 6353, RWTH, Aachen, Germany, September 1994.

[9] Leonhardt, U., A. Finkelstein, J. Kramer and B. Nuseibeh (1995); "Decentralised Process Enactment in a Multi-Perspective Development Environment"; *(to appear in) Proceedings of 17th International Conference of Software Engineering*, Seattle, Washington, USA, 14-18th April 1995, IEEE Computer Society Press.

[10] Nuseibeh, B. and A. Finkelstein (1992); "ViewPoints: A Vehicle for Method and Tool Integration"; *Proceedings of 5th International Workshop on Computer-Aided Software Engineering (CASE '92)*, Montreal, Canada, 6-10th July 1992, 50-60; IEEE Computer Society Press.

[11] Nuseibeh, B., A. Finkelstein and J. Kramer (1993); "Fine-Grain Process Modelling"; *Proceedings of 7th International Workshop on Software Specification and Design (IWSSD-7)*, Redondo Beach, California, USA, 6-7 December 1993, 42-46; IEEE Computer Society Press.

[12] Nuseibeh, B., J. Kramer and A. Finkelstein (1994); "A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification"; *Transactions on Software Engineering*, 20(10): 760-773, October 1994; IEEE Computer Society Press.

[13] _____ (1983); *UNIX Programmer's Manual*; 7th edition, Bell Laboratories, Inc., Murray Hill, New Jersey, USA.