A New Paradigm for Parallelism in Engineering Applications

(position paper submitted to EASE Workshop on Developing Parallel Engineering Applications, February 1993)

Meurig Beynon
Dept of Computer Science, University of Warwick, Coventry CV4 7AL

It is well-recognised that traditional programming paradigms have limitations where parallelisation is concerned [1]. These limitations strongly suggest that a radically new perspective on programming is needed if we are to develop general-purpose methods of parallel programming.

Our research at Warwick has focussed on the development of a new programming paradigm that is relevant to the theme of the Workshop in two respects:

1) it offers excellent prospects for parallelisation [3]
2) it has been successfully applied to engineering design and simulation [2].

From an engineering perspective, our approach can be seen as building upon the traditional principles applied in constructing a large system. An engineer interprets the behaviour of a complex system through considering the experimental observations that are required to describe the interaction between each component of the system and its environment. The justification of an engineering design is based upon knowledge of how each component behaves in isolation. This knowledge can be either theoretical or empirical in nature, but is ultimately based upon experimental evidence. This suggests that the most appropriate way to describe an engineering design is to identify components and their associated experimental observations, and to represent the correlation between these observations that can be confirmed through experiment.

The programming principles we adopt are well-suited to modelling of this nature. Each component of the system is represented by an **agent** whose interface to other agents is explicitly specified using the special-purpose notation LSD. In effect, the LSD specification identifies the experimental observations that would have to be made in order to explain the behaviour of the agent in isolation. It distinguishes in particular between observations to which an agent responds (its **oracles**) and those, which it can conditionally change (its **handles**). These correspond respectively to the parameters which the engineer measures and changes when conducting an experiment to justify the design of a component. In general, there are inviolable relations between parameters that are changed in an experiment and other parameters that are observed. Dependencies between observations that are identified in this way are represented in our approach by sets of definitions[1] ("definitive scripts"). The result is a programming paradigm based on "agent-oriented modelling over definitive representations of state" [5,7].

The power and versatility of our approach has been demonstrated in the simulation of a vehicle cruise controller [5,8,9]. Of most relevance to the theme of the workshop is the fact that – though our implementation is sequential – the mode of specification reflects the concurrent activity of components of the vehicle. The state of the system, as specified by a comprehensive set of observations defining the roles of the engineering components, is represented by a definitive script that includes several hundred definitions. Changes of state to the system are represented by redefinitions of appropriate variables and concurrent behaviour of agents is expressed by redefinitions that can occur in parallel. In particular, since the designer is free to experiment with the relations between observations that specify component interaction at any stage, design and simulation can be performed in parallel.

Agent-oriented modelling is a general-purpose technique for describing the behaviour of concurrent systems [4]. As such, it can be developed as a method of parallel programming. Definitive scripts have many qualities that suggest that they are a suitable basis for parallel programming:

- the state defined by a definitive script is independent of the order of the definitions,
- parallel redefinition of variables in a definitive script is a conceptually simple model of concurrent action,

[1] Here "definition" is used in a technical sense to refer to definitions of variables somewhat similar to those used to specify the values in a spreadsheet, and "definitive" to mean "definition-based".

- making the same redefinition twice has the same effect as making it once,
- a definitive script records the dependencies between variable values, and thus eliminates traditional problems concerning data dependencies and side-effects,
- definitive scripts can accommodate undefined values gracefully, and readily represent the effects of incomplete computational processes.

We believe that the practical exploitation of these characteristics will lead to the development of environments for engineering design and simulation that combine rich expressive power with efficient implementation on parallel hardware.

References

1. D. Baldwin *Why we can't program multiprocessors the way we're trying to do it now* University of Rochester TR#224, 1987
2. W. M. Beynon, M. D. Slade, Y. W. Yung *Parallel computation in definitive model* CONPAR'88, BCS Workshop Series CUP 1989, 359-367
3. W. M. Beynon *Parallelism in a definitive programming framework* Parallel Computing 89, Advances in Parallel Computing Vol 2, North-Holland 1990, 425-430
4. W. M. Beynon, M. T. Norris, R. A. Orr, M. D.Slade *Definitive specification of concurrent systems* Proc UKIT'90, IEE Conference Publications 316, 1990, 52-57
5. W. M. Beynon, I. Bridge, Y. P. Yung *Agent-oriented Modelling for a Vehicle Cruise Controller* Proc. Eng. Sys. Design & Analysis Conf., ASME PD-Vol. 47-4, 1992, 159-65
6. W. M. Beynon, A. J. Cartwright *Enhancing Interaction in Computer-Aided Design* Proc "Design and Automation" Conference, HK, August 1992, 643-8
7. W. M. Beynon, Y. P. Yung *Agent-oriented Modelling for Discrete-Event Systems* Proc IEE Coll. "Discrete-Event Dynamic Systems", Digest #1992/138 June 1992
8. G. Booch *Object-Oriented Development* IEEE Trans Software Engineering, SE-12(2), 1986, pp. 211-221
9. M. S. Deutsch *Focusing Real-time Systems Analysis on User Operations* IEEE Software, Sept 1988, pp. 39-50