

Towards ‘Phenomenaria’ in the Teaching of Distributed Systems Concepts.

Fionnuala O’Donnell
Centre for Research in I.T. and Education
Trinity College Dublin
Ireland

Brendan Tangney
Centre for Research in I.T. and Education
Trinity College Dublin
Ireland

1. Introduction

For Perkins [1], any learning environment can be parsed into five facets, not all of which are always present. These five facets are information banks, symbol pads, construction kits, ‘phenomenaria’ and task managers [1].

Information banks are sources of explicit information about a topic like textbooks. Symbol pads are surfaces for the construction and manipulation of symbols like word processors. Construction kits refer to a collection of building blocks that when moulded together form an entity. Lego Mindstorms is an example. ‘Phenomenaria’ are areas for the specific purpose of presenting phenomena and making them accessible to scrutiny and manipulation. SimCity, a program that models a city’s environment, is an example. Task managers are elements of the environment that sets tasks to be undertaken in the course of learning guide and provide feedback regarding process and/or product. The teacher is an example [1].

Taken together these five facets offer a perspective on the general structure and style of the learning environment and its underlying assumptions about the nature of learning and teaching. Typically today’s lecture theatre contains three of the five facets (information banks, symbol pads and task managers). Implicit in this profile are the premises that learning occurs through telling students about things, that students cannot manage their own learning and that solving problems rather than constructing entities is primary [1].

On the contrary, understanding is not something that comes free with full databanks; it is something won by the struggles of the student to learn, to conjecture, to probe and to puzzle out [1]. Thus, a shift in emphasis is needed away from the information banks and more towards construction kits or phenomenaria. The reason for this shift is because the latter facets place the learner directly and emphatically in the position of having something to make sense of or with, respectively. Moreover the role of task manager falls to that of the student albeit with scaffolding from the teacher. This shift in emphasis is what the author is proposing for the teaching of distributed systems concepts. But first what is a distributed system and why the need of phenomenaria in this subject domain?

A collection of processes, which are distinct, spatially separated, and which communicate by exchanging messages constitutes a distributed system [2]. A distributed algorithm defines the steps to be taken by each process within that system, including the transfer of

messages. It must be able to deal with the failure of one or more of the processes involved in its computation and also the failure of one or more message transmissions. This makes the task of describing all states of the algorithm difficult [3].

A major problem in teaching the latter material is the ability to capture the dynamic movement of data. Typically, when demonstrating data movement on a white board, part of the existing data configuration must be erased in order to show the new configuration. Moreover, the lecturer must decide the appropriate pace at which to reveal the configurations. However, no matter what pace the lecturer chooses it will be the wrong pace for some students because different students learn at different speeds [4].

A phenomenarium offers a way forward for both lecturers and students in that it provides mechanisms not only to animate but to simulate dynamic movement in visual form. Moreover, it can be housed within an environment that permits ‘anytime anywhere’ learning, like the world wide web, and as such offers the possibility of direct manipulation of visual display in real time, something simply not possible with static media like the whiteboard. Simply stated, a phenomenarium permits a student to interact with elements of a given algorithm in their own time, and to perform operations appropriate to that algorithm thereby gaining knowledge as to the workings of that algorithm. This paper is an argument in favour of the use of the latter in the teaching of distributed algorithms.

2. A Phenomenarium

A phenomenarium is a model of some phenomenon or activity that users learn about through interaction with the phenomenarium [5]. It is a space designed by some external agent to help the user understand the domain. Its aim is to teach about something (conceptual) as opposed to teaching how to do something (procedural) [6]. As such it is analogous to a physical simulation. A physical simulation is one in which the underlying computer model of a system is transparent to the user. In a physical simulation, one learns by manipulating the various objects or variables and observing how the overall system changes as a result [5]. Similarly, in a phenomenarium, a user learns by making changes to inputs while the latter executes and by watching the resulting changes dynamically reflected in the model’s representation. Thus, phenomenaria build conceptual understanding through experimentation. Where phenomenaria differ from other instructional models is in

the degree to which the underlying model is visible to the user and in the instructional approach employed. A phenomenarium favours the use of an expository approach [6]. The latter is one where students are given complete representations of a system with which to experience, explore, experiment and practice [5]. Automated visualisation systems like phenomenaria differ from traditional methods of teaching distributed algorithms in that they allow a student to learn in their own time and permit direct manipulation of visualised content in real time. Traditional methods can be defined as those that make use of text and a visual display medium, like a blackboard, to teach the algorithms. Alternatively in a discovery approach, students learn by modelling that is by building their own representations of a domain. Their aim is to try to figure out the underlying model thus making the latter visible would be counterproductive [6]. Such an approach to learning is exemplified in Papert's classic treatment of the microworld concept using the Logo programming language. A microworld is a place where learners can build, create their own conceptual understandings of a domain through the language of that domain [7]. This language can be either a symbol notation or a programming language. This paper favours the use of phenomenaria as they allow for the easy integration of support facilities and do not require any language proficiency. Often the need to be proficient in the language of the system overrides the need to understand the complex interoperations of the system's model.

A phenomenarium does not simply replicate a phenomenon. It simplifies it by omitting, changing or adding details or features. The purpose is to help learners build their own mental models of the phenomena and provide them with opportunities to explore, practice, test and improve their models safely and efficiently. This can be done more effectively when the model is simplified [5]. The universe of possible actions within a phenomenarium is constrained. This simplifies the choices to be made by the student and annihilates extraneous information. Students make decisions with a predetermined set of tools. Also the concept of time within a phenomenarium is user defined. A phenomenarium should embody expository learning and it should engender elements of procedural thinking that is it should allow a user to repeatedly perform operations and to reset the state of the representation if need be.

4. Token Ring Phenomenarium.

The underlying model encapsulated in the above is the token ring algorithm and its design is informed by principles outlined in section 3. It is initially presented to the student in its simplest form, bereft of failure, monitors and holding a minimum of two nodes. The aim is to allow a user to build algorithm representations of increasing complexity. He or she can vary the size of the network, create different network configurations, such as include a monitor or exclude a monitor and determine token speed constraints. Through a predefined set of

controls the student can analyse, observe the effect of such layouts on the ability of nodes to send messages, to cope with token failure, packet failure, monitor failure and/or node failure. The controls enable expository learning. With respect to node failure, a student can cause a node to fail at any time during algorithm execution. The same is true of packet, monitor and token failure. The scope of the phenomenarium is limited and restricted to the failure scenarios exhibited by the token ring algorithm. The latter are considered to be the most problematic areas for students to learn and understand. Additional features provided by the above are control mechanisms for manipulating the presentation of an algorithm and a display mechanism for textual feedback during algorithm execution. With respect to the former a student has the option of playing, resetting, pausing, stepping forward or stepping backwards through the algorithm. The algorithm can be reset or temporarily suspended at any time during execution. All operations are repeatable. Where the above differs from other visualisation systems in the same domain is in the level of detail afforded to behaviour scenarios and in particular monitor failure. Once a node detects monitor failure, a new window is open to depict the election process that ensues. The latter follows the same design principles as those outlined in section 3 and as such is itself a phenomenarium.

5. Future Direction.

Of immediate concern is the creation of more phenomenaria. Next, hypertext material should be developed to wrap around each phenomenarium to give it a context. Finally, an investigation of the contribution of the latter to the learning process is warranted.

6. References.

1. Perkins, D.N., *Technology meets Constructivism: Do they make a marriage?* Educational Technology, 1991. **13**: p. 18-23.
2. Lamport, L., *Time, clocks and the ordering of events in a distributed system.* Communications of the association for computing machinery, 1978. **21**: p. 558-565.
3. Coulouris, G., *Distributed Systems: Concepts and Designs.* 3rd ed. 2000: Addison-Wesley.
4. Stern, L., H. Sondergaard, and L. Naish. *A Strategy for Managing Content Complexity in Algorithm Animation.* In *Proceedings of the 4th annual SIGCSE/SIGCUE ITICSE on Innovation and Technology in Computer Science Education.* 1999. Krakow, Poland: ACM Press.
5. Alessi, S.M. and S.R. Trollip, *Simulations, in Multimedia for Learning (Methods and Development),* Allyn and Bacon, Editors. 2001: Boston.
6. Alessi, S., *Designing educational support in system dynamics based interactive learning environments.* Simulation and Gaming, 2000. **31**(2): p. 178 - 196.
7. Papert, S., *MindStorms: Children, Computers and Powerful Ideas.* Second Edition ed. 1993, New York: Harvester Wheatsheaf.