Modelling with Dependency in an Operating Systems Context

Nicolas Pope

An operating system is a large, complex piece of software or suite of programs that has a vast array of users and applications. Providing a rigid structure with predefined configurations and behaviours does not allow the kind of dynamic, flexible control that is desirable to enable adaptation in diverse situations. Using a computer and interacting with the operating system could be more like a modelling process where the modeller can observe everything and change anything. This is unlike existing operating systems where significant restrictions exist that are imposed by the original designers and developers. The same applies to software in general where some applications are too restricted by the original design and implementation.

By considering the operating system as more of a modelling environment I hope to remove these restrictions. Empirical Modelling has been a major influence and provides a conceptual framework for this kind of modelling. A spreadsheet is another example. One of the most appealing ideas is the use of definitions of dependencies between observables. The question that I ask then is: Can modelling with dependency further increase the adaptability of operating systems to diverse situations? There are many challenges to be overcome before such a question can be answered.

These challenges are scalability, capability and usability. Will dependency maintenance scale efficiently to millions of definitions running on multi-core and distributed systems? Secondly, is dependency capable of dealing with all operating system tasks and applications? Finally, when taken to this extreme do definitions of dependencies remain as easy to understand?

Dependency definitions can only be used if there is something to depend upon and define. In a spreadsheet you have cells and in EM you have observables, both are a form of variable but neither will scale well. Current operating systems are based around files or have a complex API, neither of which is suited to modelling with dependency. So the question is what kind of internal structure is needed? I will explore a prototype-based object-oriented approach that borrows some of the ideas behind the EM observable. Instead of files you have objects and what was the content of a file now becomes accessible attributes within an object. Attributes can then be given definitions much like in a spreadsheet.

To test these ideas I have this year been developing a software environment for modelling with dependency. This environment uses prototype-based object-orientation, allows for self-referent definitions, has a multi-core implementation and transparent network distribution. It also has a framework for interaction by external agents. Currently this has been designed to run under an existing operating system, however it has also been integrated into my own operating system. To demonstrate the tool a few different applications have been explored, including: a model of the Kinesin motor protein, a 3D game library for the Warwick Game Design society, distributed rendering on the video wall and Empirical Modelling. These applications together go some way towards answering the scalability, capability and usability questions.