Simulation model reuse: definitions, benefits and obstacles

Stewart Robinson a,*, Richard E. Nance b, Ray J. Paul c, Michael Pidd d, Simon J.E. Taylor c

a Warwick Business School, University of Warwick, Coventry CV4 7AL, United Kingdom
b Systems Research Center, and Department of Computer Science, Virginia Tech, Blacksburg, VA 24061, United States
c Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex UB8 3PH, United Kingdom
d Management Science, Lancaster University, Lancaster LA1 4YX, United Kingdom

Received 13 November 2002; revised 11 November 2003; accepted for publication 24 November 2003
Available online 25 August 2004

Abstract

The term ‘simulation model reuse’ can be taken to mean various things from the reuse of small portions of code, through component reuse, to the reuse of complete models. On a more abstract level, component design, model design and modelling knowledge are prime candidates for reuse. The reuse of simulation models is especially appealing, based on the intuitive argument that it should reduce the time and cost for model development. In a discussion with four simulation modelling experts, however, a number of issues were raised that mean these benefits may not be obtainable. These issues include the motivation to develop reusable models, the validity and credibility of models to be reused, and the cost and time for familiarisation. An alternative simulation methodology was proposed, that may lend itself better to model reuse.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Discrete-event simulation; Simulation model reuse; Software reuse; Validation

* Corresponding author. Tel.: +44 0 2476 522132; fax: +44 0 2476 524539.
E-mail address: stewart.robinson@wbs.ac.uk (S. Robinson).
1. Introduction

A simulation consultancy was contacted by a water utility organisation to develop a simulation model of their maintenance engineering operation. The simulation was to represent the creation of maintenance requests, the prioritising and scheduling of maintenance jobs and the work performed by the teams of maintenance engineers, including their travelling time. A similar model had previously been developed by one of the authors (Robinson) for another utility in the telecommunications sector. Since the facets of the two models appeared to be so similar, it was decided that the telecoms model should be reused for the water utility application. This approach seemed sensible to all, albeit that it was recognised that some recoding would be required to account for differences in the two organisations. Having completed the recoding exercise, however, a very different conclusion was reached. So little of the original code was still intact that it was thought it would have been quicker and easier to have rewritten the water utility model from scratch. This conclusion resulted from two main issues. First, the time to transfer knowledge about the model constructs from one modeller to another (in this case in the same organisation). Second, the differences in model requirements between the two contexts were much greater than first envisioned, requiring a significant amount of recoding.

There is currently a great deal of interest in model reuse among the simulation community. This is certainly not a new idea, but the interest has been heightened by the development of the high level architecture (HLA) and the prevalence of the world wide web. The idea of modellers saving time and money by reusing their own, or other people’s, models and model components is appealing, and technology is apparently making it more possible. The example above, however, suggests that model reuse may not be all that straightforward, even when the modelling contexts and modellers are close to one another.

At the UK Operational Research Society Simulation Workshop held in March 2002, a group of experts were asked to give their opinions on the issue of model reuse at a panel discussion session. This paper summarises that discussion covering issues including the definition of model reuse, the benefits and pitfalls of model reuse and the obstacles to reuse. The first contributor, Professor Michael Pidd (Lancaster University), describes a spectrum of reuse, the differences between software reuse and simulation reuse, and presents a simple cost-benefit model for reuse. Richard Nance (Virginia Tech) identifies various benefits of model reuse, but sees a number of obstacles and pitfalls. He also argues that the issue of model reuse is not simply solved by adopting component-based software development. Simon Taylor (Brunel University) asks the question ‘what use is model reuse?’, and raises a number of objections particularly in the context of using commercial simulation software. Meanwhile, Ray Paul (Brunel University) questions whether model reuse fits with our current view of the simulation modelling process. He proposes an alternative, model-reuse-friendly, view of simulation model development and use.
2. Simulation software and model reuse (the view of Michael Pidd)

Software reuse is the isolation, selection, maintenance and utilisation of existing software artefacts in the development of new systems [15]. Any survey of current literature will reveal that there are many different approaches to achieving this end. Although much attention has focussed on the reuse of source code level artefacts, reuse can be productively applied to all stages of development. This may involve any element of a system, including entities from requirements specification, design, implementation and testing [15]. This discussion concentrates on code and model reuse, rather than the other elements.

2.1. A reuse spectrum

Fig. 1 shows a spectrum of different types of software reuse, cast in terms that are recognisable to the simulation community. It shows four positions on a very non-linear scale with two different horizontal axes. The first, frequency, indicates that reuse is much more frequent at the right-hand end of the spectrum, where all of us engage in code scavenging. The second axis, complexity, runs in the opposite direction, making the point that code scavenging is relatively easy, whereas successful reuse of entire simulation models can be very difficult indeed.

Code scavenging. This is the polite description of what all of us do with our computer programs. We take something that appears to work and we use this to do something new. If we know there is some code that we can use again, probably with some slight modification, then we will do so as long as we trust the person who wrote the code. Since most such scavenged code is reused by the person who wrote it in the first place, then the grounds for trust are either very high, or extremely low, depending on whether you are an optimist or pessimist. Such reuse is fine grained, with relatively small segments of code being employed and modified. Code scavenging is surely uncontroversial and is probably the way that most of us learned how to program and how we go on learning.

Function reuse. This is the next step along our spectrum of reuse and, again, this is relatively uncontroversial since we do not think hard about using built-in functions from particular languages or systems (though L’Ecuyer [6] has long warned us about the dangers of doing this with random number generators!). The functions that we reuse in this way are usually very specific in their functionality and are fine grained, which enables us to check that the function is performing as required.
Component reuse. There are many definitions of component, but for present purposes, it is an encapsulated module with a defined interface, providing limited functionality and able to be used within a defined architecture. Components are usually larger than functions and may be internally very complex. Zeigler [21] is a thorough discussion of component-based approaches in the DEVS formalism. As an example, the functionality of a machine might be represented in a component. This machine component could be linked, at run-time, with other components without the developer of the machine knowing what the others would be. That is, there is contextual independence. In a way, a component can be regarded as a function++, since its reuse should offer all the apparent benefits of function reuse, but more so. This is the point on the reuse spectrum where things get a little more tricky, especially as components get bigger and offer broader functionality.

Full model reuse. This has long been the holy grail in some parts of the simulation world, especially when models have been expensively and time-consumingly written and developed. Full model reuse might imply, at one extreme that the executable model is used in an environment other than that for which it is developed. This, clearly raises many issues about validity. On the other hand, the model might be reused many times for the same purpose, which is relatively straightforward. The first type of model reuse is one of the justifications for the HLA, with its desire to support universal and uniform inter-operability. The problem to be faced, though, is fundamental and is to do with the nature of simulation modelling.

It ought not to need saying, but whichever form of reuse we have in mind, it should be properly planned and implemented. Reusing software requires a different approach to writing from scratch.

2.2. What is a simulation model and why do we build them?

A simulation model is a device on which dynamic experiments can be conducted—this definition excludes the use of simulation models for entertainment and for game playing. Thus, the user of the model conducts experiments on the model with a view to understanding what will happen in the ‘real’ system; this being the one that the model is intended to represent, whether or not it actually exists.

2.3. The problem of validity and credibility

With this in mind, the question of model validity looms large and simply cannot be ignored. It seems widely accepted in the simulation community that though models cannot possibly be fully validated, it makes sense to have some form of quality assurance so as to ensure that the model is fit for its intended purpose. This is the most important issue in model validation, not model fidelity. A low fidelity model can be just as useful, often more so, than a high fidelity, very detailed model built at enormous expense.

As a simulation model passes an increasing number of well-designed tests, then confidence in that model should increase, though full validation will not be achieved.
Law and Kelton [7], Sargent [19] and Balci [2] suggest a whole range of such tests. However, no test can guarantee that a model is valid, instead they are part of a process in which the credibility of a model is demonstrated. The harder the tests passed by the model, the more credible it becomes. However, there will always be tests that a model might fail, not least because it may be used in ways unanticipated by its developers and these may be inappropriate. It is important to recognise, though, that a simulation model may still be regarded as useful even though its complete validation is out of reach.

This surely serves to emphasise that a simulation model should only ever be (re-)used for the same purpose for which it was originally constructed. This is possible when a model is used on a routine basis to support tactical decision making within known and defined limits. It is not possible to be sure that reuse is valid when a model is used for a purpose different from that for which it is built or is used in combination with other models that might be based on different sets of assumptions. If a model is to be reused for a purpose other than that for which it is constructed, then it is vital to devise a new credibility assessment process against which the model’s validity may be assessed in terms of its new use. Assuming that its credibility will transfer from one application to another is simply not justified.

Proper credibility assessment does not come free and its cost must be built into any estimates of the value of model reuse.

2.4. Costs and benefits of reuse

A simple financial model of the costs and benefits of software reuse can easily be constructed as follows. Suppose that the following can be known:

\[ C = \text{cost to develop the software for its first use}, \]
\[ A = \text{cost to adapt for reuse each time it is reused}, \]
\[ N = \text{number of times that the software is reused}, \]
\[ K_N = \text{average cost/use}. \]

Thus, \[ K_N = \frac{C + A(N - 1)}{N}. \]

In these terms, reuse \( N \) is worthwhile if \( K_N < C \).

Obviously this model is too simple, in that the cost of adaptation is unlikely to be constant for each instance of reuse and the cost of reuse includes many elements, including credibility re-assessment and software adaptation. It would also be sensible to apply a discount rate to future costs so as to allow for the time value of money. Nevertheless, the basic model illustrates the usual economic argument made in favour of software reuse. It must be noted, though, that this implies the existence of a software architecture that supports reuse. Developing and agreeing this costs both time and money.

However, as discussed earlier, the main problem with cost models of this type is that they assume that all the costs are borne by the same group, whereas this is often not the case. The costs of adhering to the architecture are part of the cost of initial model and software development and these are borne by the initial developer. The
benefits of this, and the correspondingly decreased costs of each instance of reuse, accrue to the reusing group.

On the other hand, the re-users must bear another set of costs, for few large grained components and full models are reused without modification. Some of this modification may be needed if the artefact is to be reused within an architecture for which it was not originally defined. To do this may require the development of wrappers as containers for a piece of code, to make it compliant with a particular architecture, of adapters to bridge between non-compliant components and the rest of the application and of mediators to provide semantic agreement between components that are otherwise compliant.

2.5. So, what do we conclude?

The first conclusion is obvious; software reuse has been with us for a long time and shows no sign of going away. The second is equally obvious; to achieve the benefits claimed for reuse requires a properly developed strategy. This must ensure that costs and benefits are shared and that there is an agreed software architecture for reuse. When we come to model reuse, the picture gets much cloudier, for the question of validity and fitness for purpose loom very large. There is no magic wand that can be waived over a reused model to ensure that it is fit for purpose and proper validation costs time and money.

3. Model reusability: definition, benefits, obstacles and pitfalls (the view of Richard Nance)

3.1. Defining model reuse

Model reusability appears to be a topic of notable interest in the simulation research community and in selected application areas, the most apparent being the military. No doubt, the economic argument provides motivation since the investment in models and modelling by the defence sector, including both government agencies and defence contractors, is quite significant. This interest is not new, for papers focused on reusability issues appear in major conferences as early as 1986, see [18].

The topic also makes recurring appearances in the papers of Reese and Wyatt [15] and Wyatt [20]. Despite this early recognition and continuing attention in the research community, a cursory examination of widely used simulation texts in 2002 reveals that none includes ‘reuse,’ ‘reusability,’ or ‘model reuse’ as an index term. One conclusion is that model reuse is considered to be synonymous with software reuse and deserves no special treatment in a simulation book. An alternative conclusion is that a preoccupation with federated modelling and the High-Level Architecture (HLA) has created a myopia concerning reuse only at a very coarse level of granularity; i.e. reuse at the model level.

Featured paper sessions on model composability, see Kasputis and Ng [5] and Davis et al. [3] provided a coherent statement of the composite modelling objectives,
drawing attention to the difficulty of reusability when the levels of model resolution, treatments of time, and application domains can vary among components. The claim that model composability is a ‘frontier subject’ [5, p. 1577] is substantiated in part by the observation that popular contemporary books on simulation contain nothing about reusability.

3.2. The dimensions of reuse

Despite the temptation to view model reusability as a sub-problem solved in the realisation of component-based software development, differences between the two become apparent as one ploughs below the surface. These differences can be organised in three categories that might be visualised as dimensions, although any claim to an orthogonal relationship among the axes is purely notional. The remainder of this section addresses these dimensions, the first of which also establishes a rather catholic meaning to ‘reuse’ and ‘reusability.’

3.2.1. Representational artefacts

Models can exist in vastly different representational forms. Mental models are used by humans on a daily basis in fundamental types of learning and decision making. Simulation model representations, viewed as a representational continuum from the most concrete to the most abstract, are illustrated in the upper portion of Fig. 2. Object implementations anchor the least abstract or concrete section of the continuum while knowledge and experience characterise the most abstract. Communicative models are non-executable representations that can be communicated among persons. Such representations, unlike the purely conceptual or mental models noted above, enable much ambiguity to be removed in the understanding of model assertions and assumptions [1]. A similar comparison is given for software in the lower portion. Note that reuse (model or software) can occur in any of the forms shown.

3.3. Object granularity

The second dimension is based on the size and function of the component intended for reuse. Following a theoretical development for object-based modelling

---

**Simulation Artefacts**

<table>
<thead>
<tr>
<th>Object Implementation</th>
<th>Programmed Model</th>
<th>Communicative Model(s)</th>
<th>Model(s) Data</th>
<th>System/Objectives Definition</th>
<th>Problem Formulation Data</th>
<th>Knowledge (Experience)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concrete</td>
<td>Program</td>
<td>Design (UML)</td>
<td>Project</td>
<td>Requirements Specification</td>
<td>Needs Statement</td>
<td>Knowledge (Experience)</td>
</tr>
<tr>
<td></td>
<td>Segment</td>
<td>Models</td>
<td>Specific Data</td>
<td></td>
<td>Statement (Concept Definition)</td>
<td></td>
</tr>
</tbody>
</table>

**Software Artefacts**

<table>
<thead>
<tr>
<th>Code Segments</th>
<th>Program</th>
<th>Design (UML)</th>
<th>Project</th>
<th>Requirements Specification</th>
<th>Needs Statement</th>
<th>Knowledge (Experience)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Models</td>
<td>Specific Data</td>
<td></td>
<td>Statement (Concept Definition)</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2. The abstraction continuum for simulation and software.
that is rooted in SIMULA, a model of a system is an object that can be comprised of objects and the relationships among objects [8, p. 10]. Coarse-grained model composition with the linkage of quasi-autonomous models exemplifies federated computing. The most extreme alternative is fine-grained composition of objects with limited size and function. Model composition or composable modelling has the objective of piecing together components (objects) of arbitrary size and complexity. The proponents argue that effective and efficient reusability can be achieved only with the coupling of multiple-grained components during model development. Such a capability offers even higher rewards for long-lived models undergoing evolutionary changes.

3.4. Organisational level of commitment

The third dimension is described as the organisational level where commitment to reuse is made. Clearly reuse is a longstanding practice as well as a desirable goal for the individual modeller or programmer. Component reuse is identified as a requirement in major project reviews. However, is reuse frequently stated as a project goal? Are measures of the degree of reuse specified? Is a presentation of these measures required at project reviews? Are comparisons made—in time for a single project or across multiple projects? I suspect that answers in the negative for each of these questions is predominant for both simulation studies and general software development projects.

The reuse of model components within a single study is limited. An organisational commitment is needed—with its expression in measurable goals that are monitored at the project level for compliance. Further, process measures are needed to gauge the degree of support for reusability as a project objective by software and systems methodology and tools.

3.5. Key benefits of reuse

Typically, those citing the benefits of reuse rely on the argument that such components are selected from a library thus reducing both time and cost when compared with a new development. Repeated use increases the experience with the components, furnishing added ‘testing’ albeit rather ad hoc, with the consequence that library components are more reliable and less prone to faulty behaviour.

The benefits of reuse through model composability are identified by Kasputis and Ng [5] as higher quality, more cost effective simulation studies with results produced in less time. They proceed to elaborate on ‘higher quality’ as more comprehensive representations achieved through highly consistent models composed from components at different levels of resolution. The resulting composite is subject to ‘more concentrated verification and validation resources’ [5, p. 1580]. A conceptual model composition system is attributed to Page [9] and Page and Opper [10].

A benefit not cited for composability is the flexibility of creating reusable objects at varying levels of granularity. An organisation producing, installing and servicing computer/communication networks might need library components ranging from a
flow control buffer to an entire local area network in the modelling to support its activities. A particular study could require the examination of fine-grained components with subtle behavioural distinctions. Another study could entail the interfacing of local area networks with differing basic protocols. In contrast, a consulting organisation not specialising in communications networks might have little motivation to maintain a library of fine-grained components. Library components should be created at a granularity that responds to the modelling needs, and in so doing the reusable components become organisational assets.

3.6. Formidable obstacles to reuse

The work of Page and Opper [10] is especially noteworthy for its advancement of a formal model of composable simulation. This model enables a distinction of four variations of the general composability problem, and the characterisation of the difficulty in model composition is thought provoking. The conclusion offered is 'composability can induce complexity into the modelling task as well as alleviate it' [10, p. 558]. Other technical obstacles to reuse lie in the differences in the fundamental approaches of coarse-grained (federated) modelling and fine-grained (composite) modelling. Beyond the technical hurdles is the difficult financial question: Why should a project manager expend unallocated resources for enhancing reusability? The absence of contractual incentives for reuse is generally acknowledged to be the major inhibitor. Lacking project level incentives, few organisations have examined the benefits sufficiently to warrant establishing reusability as an organisational objective. Until that occurs the potential economic benefits are likely to remain hidden.

The connection between model reuse, software reuse and the open systems architecture movement should not go unrecognised. While interoperability is the objective more often stated for open-systems architectures and standards, recent design and development techniques such as pyramid structures offer gains in flexibility and reusability according to Joel Moses (http://esd.mit.edu/HeadLine/mosesspeaks.html). Moses admits that pyramid structures introduce additional overhead, but strongly supports reusability at the design level especially for very complex systems, where one should 'reuse its components like mad!'

3.7. Major pitfalls in reuse

Overcoming the obstacles cited above represents a significant challenge. Finding project level incentives that can accrue to effect sufficient evidence of benefits to induce organisational commitment to reusability is a management hurdle. The risk in placing reusability at a high priority in the absence of incentives is daunting.

The major technical pitfall might lie with the abstraction challenge. The view long taught and predominantly held in modelling and simulation is that the simplest model that meets the study objectives is the best model. When reuse is placed as a high priority objective, then might an inherent contradiction arise in identifying the proper
level of model abstraction? The reverse situation—force-fitting an existing component that does not exhibit ready compatibility—also poses a pitfall.

4. What use is model reuse? (the view of Simon Taylor)

Consider simulation modellers who develop computerised models using a commercial-off-the-shelf (COTS) simulation package. Each package typically comes with a set of predefined components that represent entry/exit points, queues, workstations, resources and entities. New models are built by combining these to form an appropriate representation of the conceptual model. Examples include the identification of a bottleneck in a production line, what call handling strategy to use in a call centre, or the impact of different triage policies in an accident and emergency ward. In some cases, models can be built from more complex components that are models that have been previously developed elsewhere, i.e. these models are reused. Experienced modellers have access to models that they have previously built and it might be possible for these models, or parts of models used to analyse analogous problems and systems, to be adapted for use in different contexts. Similar arguments can be made for modellers working in a modelling team who have access to a shared model library, or for those COTS modelling packages that have libraries of modelling components.

For example, take the case of an owner of a factory who was unsure how to increase production. The factory owner takes on the services of a modeller to help develop a strategy to accomplish this. The modeller and the factory owner work together to develop a conceptual model. To implement this as a computerised model, the modeller has several apparent opportunities to save time on building the model by model reuse. These are:

- **Reuse of basic modelling components.** The modeller reuses the basic modelling components (workstations, resources, etc.) that come with the COTS modelling package.
- **Reuse of subsystem models.** The modeller has models of various ‘generic’ factory parts that he or she has previously developed or has access to through a model library (a conveyor subsystem is often a good example of this) that can be adapted and used with a new model representing the factory. Alternatively, the factory owner might have models of factory parts that were previously developed and makes these available to the modeller.
- **Reuse of a similar model.** The modeller has previously developed a model that has similar features to the factory being studied. The model is adapted appropriately.

Do these actually save time? The first of these, ‘reuse of basic modelling components’ is performed by the modeller selecting and using the modelling component. Experienced modellers will know that this is not entirely the full story. Take for example a workstation component. The developers of this component have made some assumptions about how workstations work. A modeller using this workstation
in a model will have to test the workstation to understand how this actually works in the COTS modelling package as there is no standard cross-package behaviour. When the modeller uses the workstation component, if it cannot appropriately model a particular machine, the modeller can take advantage of programming facilities or links to other programs (such as a spreadsheet) that most COTS packages make available. The implications of this is that models built this way come with 'baggage', i.e. programmed behaviour and/or supporting components that are required for the model to be simulated. This is made worse by 'baggage' being extremely dependent on the version of the package, the platform being used, and even the way in which the operating system has been configured. The conclusion to this 'reuse of basic modelling components' is that they are reused, but only after testing has been performed and modifications have been made or added. The basic component often evolves significantly beyond its original form.

In the second of our opportunities for reuse, 'reuse of subsystem models,' the modeller identifies part of the factory that can be quickly modelled by reusing a previously developed subsystem component that comes from the modeller's own library or from the library of the modelling package he or she is using. Either way, the subsystem model has to be tested to determine if it correctly models the subsystem and then modified appropriately. If this complex component has 'baggage', then these also have to be checked and understood. The implication of this is that unless a subsystem component is quite simple, a modeller will have to spend a great deal of time understanding how the component works. Additionally, one must ask what is the likelihood that the subsystem component will conveniently model the equivalent factory subsystem? The conclusion to this 'reuse of subsystem models' is that for most cases the reuse of a subsystem model could be more costly than developing it from scratch.

Similar arguments can be made about 'reuse of a similar model' where the thorough testing of the model will only take longer than testing a subsystem component. It is possible to see a similar model (with appropriate modifications) being reused as the system it represents evolves. However, it is never likely that a model will be capable of being used to model another similar system. For example, production lines appear similar in that they tend to be a linear series of buffers and processing stations. Will two production lines really be that similar when studied in detail? Would a modeller be better off starting afresh rather than spending time attempting to establish how a similar model works and what modifications need to be made?

In summary, I ask the question 'What use is model reuse?' In the world of COTS simulation packages it is difficult to see practically how one can trust a model without detailed verification and validation that may be more costly than developing the model from the start. The answer to this question may well therefore be 'No use!' However, it is pleasing to note, anecdotally at least, that this is not always the case. Several businesses are beginning to realise what cost savings model reuse might actually give them if supported properly. What is emerging is the concept of a 'best practice' bureau that organises the practice of simulation modelling within the organisation. The contribution of this? A set of models developed using common practices and terminology that can be reused within the context of organisation
guidelines. So, again asking the question ‘What use is model reuse?’ I might now be persuaded to answer ‘Some use but with the caveat of careful planning and foresight!’

5. Model reuse: an alternative simulation methodology (the view of Ray Paul)

It might be argued that model reuse is essentially dependent on trust. If a modeller cannot trust a model then surely they cannot reuse it. It seems to follow that for a modeller to reuse a model, then the modeller must build trust, a process that might take more time than building the model from the start. Are we missing the point?

Simulation modelling is a decision aiding technique. Discrete event simulation modelling is a quantitative technique. Experimentation with operational models produces numerical results that can be used to indicate that one decision is better than another. However, numbers cannot represent all possible factors at play in the system being studied (the relationships between stakeholders, for example). One must remember that the process of simulation modelling is not designed to find the answer or answers. It is there to help decision makers make decisions, or to help decision makers gain an understanding of their problem. It may be that the numerical output of the simulation model in itself is of no intrinsic value. Learning about the processes of the interactions that go on within a complex environment, the relationships between the variables, is probably the dominant reason for using simulation modelling.

With the world wide web we are faced with the potential to change the way in which we model. There are many applications that are used on the web that loosely foster a ‘suck it and see’ approach. Browsing and adventure games encourage the participant to try out alternatives with rapid feedback, avoiding the need to analyse a problem with a view to deriving the result. In terms of simulation modelling, we might advocate development tools that allow for fast model building and quick and easy experimentation, tools that allow simulation models to be used for problem understanding [13,14]. ‘Web-enabled’ simulation analysts will use these tools to assemble rather than build models.

Fig. 3 shows a possible methodology based on assembly rather than build. In this the webber-analyst grabs and glues bits of the model that might be deemed sufficiently appropriate. Running the quickly assembled model enables its fitness for purpose to be established. If satisfactory, problem understanding is attained. If unsatisfactory, it is rejected and ‘grab-and-glue’ is tried again. The webber-analyst follows this $G^2R^3$ approach (Grab-and-Glue, Run, Reject, Retry) at a fast rate, getting insights during the $G^2R^3$ process and satisfying the stakeholders of the problem at a time acceptable to them. It is implicit in this approach that a $G^2R^3$ model would not necessarily have to mimic the real world in which the problem exists. The $G^2R^3$ model would only need to characterise the system being studied.

Page et al. [11] discuss these issues at length. Quality is raised as an issue, but of course no software can be ‘proved’ correct in these circumstances. Why should modellers take so long to get an answer(s) from a ‘traditional’ simulation model when
that model cannot be proved to be correct? However, if it becomes possible to ‘glue’
bits of a model together fast enough and experimentally, then we might see a shift of
emphasis from ‘is the model correct?’ to ‘is the analysis, albeit with unproven soft-
ware, acceptable given the large experimentation that swift modelling has enabled
us to carry out in a short space of time?’ In other words, the search space might
be dramatically reduced not by accuracy (the old way), but by massive and rapid
search conducted by an empowered webber-analyst (the new way). Models are re-
used in this way without trust but as part of an intellectual process that fosters
understanding. Surely, this is a more attractive, practical future to model reuse than
the alternatives currently on offer?

To illustrate this consider the following scenario. A bank manager has a problem.
She has observed that at peak times customers queue for too long and often leave the
bank. She wants to reorganise the bank to try and reduce the queue. She has staff
who can be used to serve customers and staff who can perform the back office tasks
that need to be done before the bank closes. She enlists the services of a simulation
analyst who builds a model of the bank in the ‘traditional’ way with input from the
stakeholders, experiments with it and reports that two people should be moved from
the back office without an adverse effect on the day’s commitments. Using the report
she rings the changes.
Following a G²R³ approach, an alternative future is possible. The webber-analyst enlisted by the bank manager searches the web for an appropriate model or models. He finds one similar to the front office and one similar to the back office. Technology notwithstanding, the webber-analyst ‘grabs and glues’ them together and studies the behaviour of the new model’s output as he varies the input. After a short while he determines that the back office is not appropriate for this task (for example a model with graphics relevant to a sports centre) and searches the web for another (Run, Reject, Retry). He finds another, ‘glues’ it to the front office and tests it again. Satisfied he now presents the model to the bank manager. Rather than producing a fully validated model of the bank, he has quickly generated one that can teach ‘lessons’. The bank manager knows her bank but might not be a ‘systems thinker’. As the webber-analyst shows how this generalised bank behaves under different inputs and configurations, the manager reflects on how her bank might react under similar conditions. Again she rings the changes.

Critics of this approach may well shout ‘foul!’ at this point. However, consider an emerging use of simulation. Robinson [17] considers several modes of simulation use. Of his modes, mode 2 represents ‘Simulation as a Process of Organisational Change’ and mode 3 ‘Simulation as Facilitation.’ In mode 2, the simulation analyst is an ‘agent of change’ whose task is to help the user [4]. In mode 3, the analyst and the problem owners use the model in an interactive manner as a means for understanding the real world and for promoting discussion on potential improvements. He further supports mode 3 with a case study of simulation performed in this manner [16]. Relating this to our bank example, mode 2 is representative of the ‘traditional’ approach and mode 3 is similar to the approach supported by G²R³. This is not to say that one is better than the other! I cite Robinson’s modes to illustrate that simulation analysts already use ‘non-traditional’ methods of simulation.

Further, consider the excellent and varied computer games being played by children today. The ‘Sims’ is one of the best selling and is a good example of this. In this game players begin with a basic framework in which the ‘Sims’ come to live. The player changes their social behaviour and tries different living environments to make their simulated ‘lives’ better. The measure of success is the amount of money that the player has to improve their environment. The approach that they use to improve their game playing success is by taking what is effectively a G²R³ approach. The variety of possible combinations of social habits facilitate the grab and glue of many different approaches. Each are tested (run) and solutions that work are kept and built upon while solutions that perform poorly are rejected and others retired. The point of this? These players are the analysts of tomorrow!

In conclusion, it is obvious that reuse is a complex subject and there are applications of simulation where careful validation is required. The G²R³ approach I advocate may well be infeasible in terms of technology (although developments in the semantic web show promise). However, I must remind readers that those that follow us, as discussed above, are already building complex worlds to play games and to solve problems by a ‘grab and glue’ approach. This is an exciting opportunity for re-use that may well harness new realms of creativity that will make a greater impact on decision making than our current ‘traditional’ approaches.
6. Conclusion

Some key themes emerge from the views expressed above. Although there is an association between model reuse and software reuse, the motivations behind modelling and software development are quite different. These must be taken into account when transferring ideas from one field to the other.

Model reuse does not simply mean the reuse of complete simulation models. There is a spectrum of reuse, from using small portions of code, through larger components, to complete models. While reuse at the higher end of the spectrum is problematic, reuse at the lower level is simpler.

The benefits of reuse should accrue from the reduced time and cost for model development. There are obstacles, however. First, there is little motivation for model developers to adopt procedures that would enable model reuse. To do so would increase the cost of model development, while the benefits would be gained by others. Second, there is an issue with the confidence that can be placed in code obtained from another context. Third, there is the time and cost of familiarisation with someone else’s code, which may outweigh the time and cost benefits of reuse.

It may be that there is little fit between the ideas of model reuse and current views on simulation modelling methodology. An alternative methodology may make model reuse a more practical prospect in the future.

References