Proceedings of the Operational Research Society Simulation Workshop 2012 (SW12)
B.Tjahjono, C. Heavey, S. Onggo, and D-J. van der Zee, eds.

THE POTENTIAL ROLE OF OPEN SOURCE DISCRETE EVENT SIMULATION SOFTWARE IN THE MANUFACTURING SECTOR

Mr. Néill Byrne

Dublin City University
Dublin, Ireland
neill.byrne@dcu.ie

Dr. Paul Liston

Dublin City University
Dublin, Ireland
paul.liston@dcu.ie

Dr. John Geraghty

Dublin City University
Dublin, Ireland
john.geraghty@dcu.ie

Dr. Paul Young

Dublin City University
Dublin, Ireland
paul.young@dcu.ie

ABSTRACT

Discrete event simulation (DES) is a valuable tool in the manufacturing sector due to its flexibility and ability to model even very complex systems, given sufficient time and resources. Despite its strengths, there seems to be less penetration in this industry than expected. Models are typically one-off projects that are not maintained beyond the initial analysis. Industry based research conducted through the Irish Centre for Manufacturing Research suggests that due to cost and licensing restrictions, modelling is limited to a single user and consequently the associated skillsets are not developed more widely across the organisation. A possible solution is the use of open source DES which does not carry the same cost and licensing restrictions as the equivalent proprietary software. This paper reports on some of the open source packages currently on offer and provides a case study based comparison of open source and proprietary DES software.

Keywords: Open source software, Simulation standards

1 INTRODUCTION

The manufacturing sector has provided many interesting challenges for discrete event simulation (DES) modellers over the decades and there are numerous publications detailing how it has been used to gain tangible benefits and economic return (e.g. Harrell et al. (2010)). The ability to capture stochastic and dynamic behaviour and convergent material flows allows DES to more comprehensively represent these systems than other analysis method (Huang et al. 2003). In a recent review of the literature, Jahangirian et al. (2010) report that research activity in this area has seen a substantial increase in real problem solving, and suggest that this may be due to accumulated knowledge and experience in industry, the availability of tools and greater awareness of the benefits. While this is encouraging for the DES community, given the maturity of DES methods and the wealth of potential industrial applications, the authors suggest that the take-up rate of DES across manufacturing companies is still remarkably low and chiefly confined to large well-resourced organisations.

DES has typically supported longer term strategic and capital investment decisions, and has been less often used to support regular or real time decision making. Consequently simulation projects are usually concerned with a series of once-off analyses, between which, the stakeholders go back to their daily activities or move on to other roles. Consequently, the momentum and experience gained is lost

(Hollocks 2006). This inclination towards longer term decision making can mostly be attributed to the time required to complete a simulation study. The length of time from initial problem specification through data collection, model creation, verification and validation steps, can run to many months. Jahangirian et al. (2010) noted that DES does not possess the same level of stakeholder engagement as other analysis methods and suggest that this may be due to the "difficulty and time needed for data gathering, which usually repels stakeholders in the fast pace of today's business". The majority of manufacturing companies are aware of DES, but managers are often cynical of the potential benefits in their specific systems and reluctant to commit budget allocation and resources to exploring its potential.

Perhaps simulation based analysis needs to be introduced to manufacturing environments in a more integrated fashion than that promoted by commercial products. These products generally do have integration/interfacing capabilities, but as Hollocks (2001) notes, the vendors place more emphasis on the graphical user interface and the end customer usability. While there may be a more involved installation process if simulation based analysis was tightly integrated with data management and manufacturing execution systems, there could be a significantly greater possibility of providing real-time simulation-based problem-solving capability, as called for by Fowler and Rose (2004). This would minimise the analysis time and create opportunity for far greater utilisation of DES analysis in short term decision making. In regards the simulation engines that could power this analysis, open source DES software stands out for three reasons: firstly, the structure of open source software would be more amenable to integration with other applications as it could form part of, rather than interface with, the code of other applications. Secondly, companies may be reluctant to develop decision support around software that is not transparent or is exclusive to a particular vendor. Thirdly, software cost is often cited as an obstacle to implementation and while open source software may not be completely without cost, (as discussed later), it may at least, partially circumvent this obstacle.

These considerations have prompted the authors to pose the question; "can open source DES software help to propagate the use of DES for multi-level decision making in manufacturing companies?" Open source software may not offer a panacea for all the challenges faced when using model-based analysis in a manufacturing company. However, it may be a potential agent in the instigation and continued use of simulation in areas of decision-making where proprietary software has gained a limited foothold (e.g. real-time decision-making and the SME market). The next section of this paper briefly explains what is encapsulated by the term "open source software" and what are the implications and limitations of its use. The subsequent sections describe some examples of available open source DES products and detail two case study comparisons of an open source (SimPy) and a proprietary (ExtendSim) product.

2 OPEN SOURCE SOFTWARE

Open source development involves the creation and maintenance of software that is open, transparent and peer-reviewed by its users. The open source community advertises its software under the Free Software Definition that outlines the four requirements for a software program to be defined as 'free' (http://www.gnu.org/philosophy/free-sw.html, accessed 30 October 2011) :

1. The freedom to run the program, for any purpose.
2. The freedom to study how the program works, and change it so it does your computing as you wish.
3. The freedom to redistribute copies so you can help your neighbour.
4. The freedom to distribute copies of your modified versions to others.

Key to the continuation and promotion of open-source software is the 2nd "freedom" which insists that if you change the source code or software in any way, you must then release it under open source licensing. This prevents open source code from becoming locked into proprietary software and prevents the code from disappearing from the open source realm. This can be a concern for commercial software developers who wish to use open source code within their proprietary software. However, it does not mean that the code cannot be used in commercial software, on the contrary, it means that if you edit or change the open source code you must release it under an open source licence. Examples in the industry include open source Linux kernels, being used to create commercial computer operating systems such as Red Hat Enterprise Linux.

There is very little (if any) literature or evidence regarding open source DES software adoption in the manufacturing sector. However, it is possible to draw some conclusions from the wealth of literature that discusses the views of open source software in companies and infer similar reasoning as to why open source DES software appears to be non-existent in the manufacturing sector. Ven et al. (2008) state that there are many claims and counter-claims about the benefits of open source software and Goode (2005) conducted a survey of a number of Australian businesses, regarding their rejection or dismissal of open source software in their company (see Fig. 1). He suggests that there may be a potential role for a "technology champion" to raise awareness of the open source alternatives within the company and to assuage some of the common misconceptions.

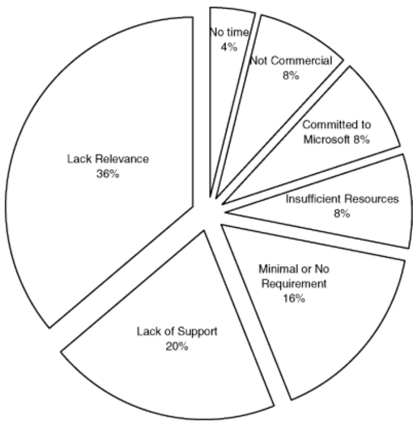

Figure 1 *Reasons for rejecting open source software, Goode (2005).*

Within the ICMR group it was found that the partner companies rarely reviewed all of the proprietary software before purchasing a simulation engine, and that none were even aware that open source alternatives existed. It was also difficult for them to assess any DES software during initial procurement as they were unaware of the project requirements, and in general, their journeys in simulation projects were very organic.

3 REVIEW OF OPEN SOURCE DES SOFTWARE

This section describes some of the available open source DES software. The list is not intended to be exhaustive but highlights the "easily-discoverable" open source DES packages with sufficient online activity and documentation to warrant inclusion here.

3.1 SimPy

SimPy (http://simpy.sourceforge.net/) short for ***Sim***ulation in ***Py***thon is a library for process-driven discrete event simulation written in the high level general purpose programming language Python. SimPy utilises Python's object oriented structure to offer a number of objects that represent common structures in the real world system, where entities compete for resources. Creation of this simulator was made possible by the inclusion of *generators* in Python version 2.3. A generator is similar to a function but differs in that it retains a memory of its variables and returns values at a number of different points in the generator function. This means that the process or lifecycle of an entity, more commonly known as a Process Execution Method (PEM) can be captured in a generator, which yields control to other entity PEMs or back to the main loop of the program. For example, the following code simulates the arrival of parts to a queueing buffer every 2 hours by defining an entity (or class) known as `Source` which is a subclass of the main SimPy PEM class `Process`.

```
class Source(Process):
    def run(self):
        while 1: # infinite loop
            yield put, self, buff, 1 # add 1 to the buffer
            yield hold, self, 2 # wait for 2 hours
```

The PEM operates in an infinite loop of adding 1 to a buffer using the `yield put` command and then waiting for 2 hours using the `yield hold` command, before repeating the process. These types of PEMs create python co-routines that yield control at various points during the execution of the simulation model. SimPy offers a number of classes that can be used to model real world process and entities, these include:

- Resources – which are requested and released,
- Stores – which hold items including other SimPy objects, and are generally used to model buffers,
- Levels – a container for a value which can be added to or subtracted from,
- Monitors – a container for values, used to monitor or trace variables in the model.

Besides these basic building blocks, there are a number of commands that allow priority requests of resources including pre-emption and interruption of PEMs by other PEMs. One of the strengths of SimPy is its ability to use the extensive and plentiful libraries of software written for Python, that include animation (e.g., *PyGame* and *Pyglet*), database interaction (all major database architectures are catered for) and advanced mathematical libraries (e.g., *SciPy*, *NumPy* and *SymPy*). Access to these libraries means that SimPy models can be integrated into larger experiments and projects quite easily. All of the aforementioned packages are also open source, which increases the usability of SimPy. There are also a number of open-source Integrated Development Environments (IDE) for Python such as *IDLE* and *PyDev for Eclipse*, which allow debugging of models in SimPy. For deployment of models or applications, Python programs can be compiled or 'frozen' as executables on most operating system platforms without the need for the end-user to install Python or any other software.

Currently, SimPy is being actively developed, version 2.3 is the latest version, with version 3.0 in the making. Users communicate via a SimPy forum and some universities courses have opted to teach DES via SimPy (e.g., http://heather.cs.ucdavis.edu/~matloff/156/PLN/).

3.2 JavaSim and C++Sim

C++Sim was originally devised based on Simula in 1990 and then later ported to Java as JavaSim (http://javasim.codehaus.org/) in 1996 (Little and McCue, 1993). Originally it was made freely available under its own licensing by Newcastle University and was one of the first open source offerings. At the request of the authors of the library, it has been released publicly under LGPL licensing. It is a process-based DES language, and the authors of the simulator claim that it is easy to learn and use, follows the programming paradigm promoted by Java, is flexible and extensive, as well as being efficient. The JavaSim package also includes some libraries for generating random streams from distributions including the uniform, exponential, erlang, hyperexponential and normal distributions.

Similar to the previous example, the following example of JavaSim code uses a defined class called `Source` from the JavaSim class `Process` to create an item and place it in a buffer or queue every 2 hours.

```
class Source : public Process
{
public:
    Source () {};
    ~Source () {};
    void Body ();
};
void Source::Body ()
{
for (;;)
```

```
            {
                Item* c = new Item();
                queue.insert(c);
                Hold(2.0);
            }
        }
```

Whilst being a very mature simulation library, there appears to be very little activity around the package and little comment on its future development.

3.3 Sim.JS

SIM.JS (http://simjs.com/index.html) licensed under the LGPL licensing is a general-purpose discrete event simulation library written entirely in JavaScript that uses the event-based design paradigm, in which changes to the system are notified via the JavaScript *callback()* function. The reason for using event-based as opposed to process-based (as in SimPy and JavaSim) is that it is less resource intensive, (an important attribute if using client side browsers), given that each entity does not need to spawn its own thread. Also, at the time of the initial writing of the library, not all browsers supported process-based design.

The modelling structure consist of entities that compete for resources such as *Facilities*, *Buffers* or *Stores*. Entities communicate via *Events* or *Messages*. Recording and monitoring structures are also provided to trace statistics generated during the model execution. The authors also created a library for random number generation for a few of the more common distributions, as they felt that the standard JavaScript equivalent library was not suited towards discrete event simulation. The following brief example code shows a source that creates entities, places them in a buffer and then waits for 2 hours before repeating the process again.

```
var Source = {
    start: function () {
    this.putBuffer(buff,1); // adds 1 item to the buffer
    this.setTimer(2.0).done(this.start); //recall after 2 hours
    },
};
```

It is the aim of the authors of the simulator to bring DES to the web and to encourage simulation models to utilise the client-side scripting capability of modern browsers. Their online simulator example (http://simjs.com/queuing/index.html, accessed on 8 November 2011) offers a number of basic objects which include queues, traffic sources, traffic sinks and traffic splitters that can be dragged and dropped into a modelling canvass and connected to create a network. This allows the user to construct simple DES models within the browser and experiment with different configurations in a user-friendly GUI.

Of the three open source DES simulators investigated, there are a number of advantages and disadvantages, when comparing them against their commercial equivalents. Most notable is that they are all written in one of a number of popular high level programming languages (e.g. Python, Java or JavaScript). This means that integrating models with current manufacturing information systems is more viable and less restricted. Commercial packages tend to operate in a single compiled executable that has limited integrating abilities. Many allow interfacing with the code via Visual Basic COM links with specific API's, whereas the open-source programs available, can be directly included in the code of external information systems and architectures.

Basing the software language on a popular high level language allows users proficient in that language to migrate easier to the open source simulator, whereas many commercial packages use a custom language which is unique to their software and users must familiarise themselves with the syntax. For example, *ExtendSim* uses *ModL*, *PlantSim* uses *Visual Logic* which is based on Visual Basic and *Simul8* uses *SimTalk* which is based on C++.

This point is emphasised if users are attempting to build more than just a basic model. Very few commercial packages are completely flexible and typically some coding is required. Banks and Gibson (1997) argue on the claim of commercial software vendors that models can be built without any programming. They state that this is only possible with very application-specific simulators which are not applicable to problems outside of that domain. Therefore, if programming is a inescapable requirement of the simulation practitioner, it may be more beneficial for them to adopt an open source code based simulator in the first place.

4 CASE STUDY I: PREDICTIVE CAPACITY PLANNING TOOL

In order to investigate the appropriateness and ability of open source DES software to model a complex real system, some research was carried out on behalf of the Irish Centre for Manufacturing Research (www.icmr.ie). The aim of the research group is to investigate the maintainability of simulation models and their deployment in the manufacturing sector. Through consultations with partner companies, which consist of a number of multinational manufacturing facilities in Ireland, the research group found that the companies experience with simulation models was quite varied and that often models had a very short lifecycle beyond their initial scope and goal. The research group found that these limitations were due to a multitude of reasons, the most common being that the owners of the models either moved on to newer projects or couldn't find the time to maintain the model. Furthermore, limitations of licensing, whereby the companies only purchased one or two licences due to the investment cost, was thought to be a significant detractor from the successful penetration of simulation in the companies.

A pilot study at one of the companies involved creating a predictive capacity planning tool for a product line, which would allow engineers to determine future resource requirements and allocate personnel and resources. The model was driven by a production schedule consisting of future orders and contracts undertaken by the company. The company had previously engaged in a simulation modelling project which was undertaken in conjunction with a local university. Once the owner of the model, a student from the university graduated, the model and its use, quickly became redundant. Furthermore, the software license for the commercial package, which was purchased for the project, was locked down to a single laptop, and had not been used after the initial project. Whilst this serves as anecdotal evidence of some of the difficulties associated with deployment of simulation models, it is not a isolated incident, Hollocks (2006) discussed that he had found identical phenomena.

SimPy was used to model the line and was embedded in a Python application as a standalone executable that took, as its inputs, a spreadsheet containing the planned production schedule for the forthcoming 10 months, as well as, a snapshot of the current WIP profile in the line. The output displayed various performance indicators of each station on the line, including cycle time, queue sizes and utilisation, an example of which is given in Figure 1. The user configurable variables included the number of resources and personnel allocated to each station on the line, as well as various scheduling strategies and setup minimization techniques at the machines. This allowed management to experiment with various configurations on the line and smooth planned production, whilst determining the likely sources of future bottlenecks.

In conjunction with this SimPy version, a similar model was built using ExtendSim to validate both models against each other, and also investigate how the users in the company responded to the two models. In terms of interaction with the spreadsheet information, both packages performed quite well. ExtendSim imported the information in a timely fashion and the SimPy model used a third party Python library called *xlrd,* which accessed the spreadsheet information. One of the biggest difficulties in the project involved modelling the complex shift patterns of staffing. In ExtendSim, this was accomplished with relative ease - *Shift* blocks in the ExtendSim models took care of preventing operation during offline times, whereas, in the SimPy version it was more complicated to interrupt a worker from performing a task and have them return to the same task where they left off.

Initially, the model users were more accepting of the ExtendSim version and impressed with the animation, as they could see the items moving through the line. However, once the novelty of this wore off, the model results became the focal point, and users would often run the simulation with the animation screen minimised. It was also found that the lead time required to build the SimPy model was a lot longer than the ExtendSim model, primarily as it acted as a black box and users were not familiar enough with the code to make changes, and hence a lot more error handling was required before the final deployed version could be rolled out.

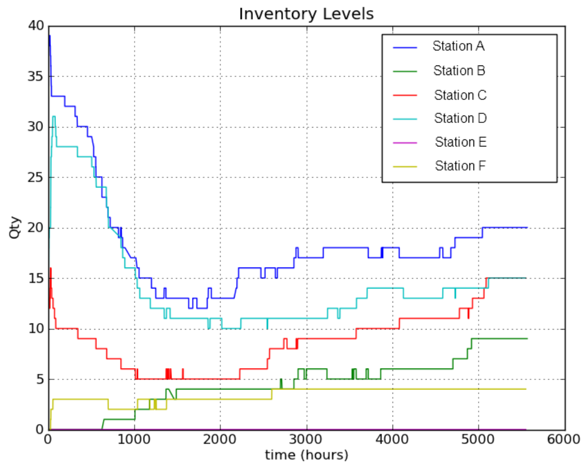

Figure 2 *Predicted inventory levels at each station on the manufacturing line using open source SimPy modelling package.*

The biggest advantage of the SimPy model was that the standalone executable could be emailed and deployed to any or all of the end users without the need to install any software or deal with operating system issues. In this way, it was possible to locate the program on a central server and invite interested parties to download and experiment with their own configurations. It was found that this method of deployment generated a lot of user discussion and debate about the best possible setup for the line, and subsequently piqued the interest of many employees in DES modelling. At the request of the company, the ICMR agreed to offer a training module which had previously been successfully delivered to other companies.

The initial training course was offered using ExtendSim as it was felt that participants with no background in programming or simulation might find it more conducive to their learning. At the initial stages of the course when ExtendSim was first introduced, the participants were impressed at being able to build simple models quickly and with very little effort. Towards the later stages of the course SimPy was briefly introduced, and the participants appeared impressed with its simplicity and conciseness. It was also felt by the module teachers, that SimPy forced the participants to think more about their system in a ‘discrete event’ way rather than building a ‘map’ of the real system which is often the learning paradigm of commercial packages.

5 CASE STUDY II: SEMICONDUCTOR MANUFACTURING MODEL

A flexible and reusable model of a semiconductor manufacturing system was built using both ExtendSim and SimPy to compare and examine the benefits of each software package. Both models used the publicly available *Semiconductor Wafer Data Format Specification* as test data. A full design of experiments for the simulations was programmed in Visual Basic, which controlled ExtendSim as a background process. Similarly, the same design of experiments algorithm was implemented in Python which controlled the SimPy model. Both models outputted an operational characteristic for the semiconductor fab and allowed engineers to compare a profile of the utilisation/cycle time relationship. As can be seen from Figure 2, the results for both models were identical for the same dataset, with a difference only in the high utilisation zones where both models were deemed to be overloaded and unable to attain steady state.

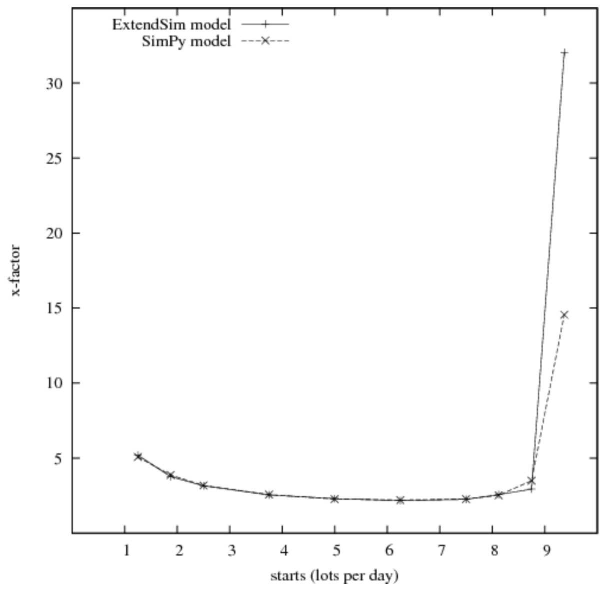

Figure 3 SimPy and ExtendSim operational characteristic curves for semiconductor fab using Sematech minifab dataset.

Overall though, the Python/SimPy application was easier to implement and maintain as it did not require any cross communication between different software platforms, whereas communication between the Visual Basic GUI and ExtendSim was a little more cumbersome. ExtendSim provides COM links which were used to control the simulations from ExtendSim. Scripts were created to feed the data into ExtendSim which were found to be quite slow and error prone. In subsequent iterations of the application, it was found to be far more efficient to send a message to ExtendSim to import the data directly into ExtendSim internal database.

Another issue regarding the ExtendSim application, was implementation of the modelling strategy required to auto-generate the simulation models. The modelling strategy involved conceptualising the many aspects of the system such as tools, operators, downtimes and batches as items, which circulated the system and paired with each other to define an activity. For example, if a tool item, repair item and operator item were paired and delayed for a period of time, it signified that a repair of the tool was being performed by the operator. This type of strategy required using ExtendSim in a less than usual manner. ExtendSim is representative of most modern ‘drag and drop’ simulation software packages in that it is job-driven. As described by Fowler and Rose (2004), job-driven refers to “manufacturing jobs as active system entities while system resources, such as machines, are passive. The simulation model is created by describing how jobs move through their processing steps, seizing available resources whenever they are needed”. However, for a flexible modelling system that can generate a flexible model based on the data specification, it is not possible to use this modelling strategy. Therefore, implementing the model in ExtendSim required the programming of some custom modular blocks to facilitate this flexibility. This is not an uncommon experience when using a simulation software package, and Banks (1999) explains this as the ‘90% rule’ whereby you find that the software has sufficient rudimentary capability to achieve 90% of the original objective, and an extra 2-5% being achieved by using the software’s functions in unusual or unorthodox ways.

Overall though, the SimPy model was much faster (of an order of 5:1) as it did not have any animation cycles, and did not require execution of the superfluous code in ExtendSim which makes it feature-full general purpose simulator. However, any statistics of interest needed to be manually generated in SimPy, whereas in ExtendSim, this information is collected by default. More care and attention in the SimPy version was required to model the events that could happen to an entity during its lifecycle. For example if a lot entity was occupying a tool entity, it was difficult to interrupt this, say, if the tool went down or a shift came to an end. ExtendSim includes canned blocks of code that can account for this, however, one might argue that there is value in having to understand the complex interactions and activities of the real system entities when creating a simulation model.

6 CONCLUSIONS

It is widely considered that DES has not achieved the level of uptake amongst manufacturing companies that the reported benefits of its use would warrant. Licensing cost is one purported barrier to simulation software use. This paper has identified three examples of DES software that are freely available under open source licences and therefore overcome the accessibility issue of purchasing cost.

Two case study based comparisons between one of the open source options (SimPy) and a proprietary DES package (ExtendSim) have been described. It was found that there were strengths and weaknesses to both options. From a technical perspective both software packages were able to adequately represent the systems under analysis, access the required external data and provide the necessary experimental capability. In terms of usability there are two perspectives: the first is that of the modeller and the second is that of the model user. For the modeller in the case examples, the lack of animated GUI in SimPy (as is the case in all other open source DES currently available) was not a significant drawback in the modelling building phase but the animation provided by ExtendSim was beneficial when initially introducing end users to the models. The trade-off for this visualisation was a negative impact on model execution times, which gave the open source alternative a substantial advantage after the model was accepted and the user became only interested in model results. A further advantage of the open source option was that model use could spread quickly through the organisation to new users without licensing restrictions. However, although the proprietary software did not spread as quickly, it was found that the GUI allowed new users to more quickly understand the model and discover the parameters they wished to edit.

Licensing cost, while a significant barrier, is not the only barrier to simulation uptake and the availability of freely available open source DES software (at least as it currently exists) will not entirely overcome the accessibility issues of simulation software. With its steeper learning curve, it might even be more difficult to convince the manufacturing industry of the benefits of open source simulation software. It is believed that a more holistic approach to removing the barriers to entry, which addresses simulation awareness/understanding, model build effort/time, and integration with existing systems/practices, is required and perhaps open source DES software can play a role in this effort.

ACKNOWLEDGMENTS

This research is undertaken as part of the Irish Centre for Manufacturing Research (www.icmr.ie). The project is funded by the Irish Government and the European Union under the Irish National Development Plan 2007-2013 via the Enterprise Ireland and IDA Ireland (Industrial Development Agency) Technology Centre Initiative.

REFERENCES

Banks J (1999). What does industry need from simulation vendors in Y2K and after? A panel discussion. *Proceedings of the 1999 Winter Simulation Conference*.

Fowler J W and Rose O (2004). Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems. *Simulation* **80(9)**: 469-476.

Goode S (2005). Something for nothing: management rejection of open source software in Australia’s top firms. *Information and Management*, **42(1)**: 669-681.

Harrell C, Winsor S and Teichert G (2010). Increasing Throughput in an Automated Packaging Line with Irreducible Complexity. In: Johansson B, Montoya-Torres S J J, Hugan J, and Yücesan E (eds). *Proceedings of 2010 Winter Simulation Conference*, Baltimore, MD IEEE, 1567-1573.

Hollocks B W (2001). Discrete-event simulation: an inquiry into user practice. *Simulation Practice and Theory*, **8(6-7)**: 451-471.

Hollocks B W (2006). Forty Years of Discrete-Event Simulation: A Personal Reflection. *The Journal of the Operational Research Society*, **57(12)**: 1383-1399.

Jahangirian M, Eldabi T, Naseer A, Stergioulas L K and Young T (2010). Simulation in manufacturing and business: A review. *European Journal of Operational Research*, **203(1)**: 1-13.

Little M C and McCue D L (1993). Construction and Use of a Simulation Package in C++. *Department of Computing Science, University of Newcastle upon Tyne*, doi: 10.1.1.53.8198

Ven K, Verelst J and Mannaert H (2008). Should you adopt open source software? *IEE Software*.

AUTHOR BIOGRAPHIES

NEILL BYRNE received a BE in Mechanical Engineering from University College Dublin in 2002 and received an MSc in Computer Aided Mechanical and Manufacturing Engineering from Dublin City University in 2006. He is currently reading for a PhD at the same University and engaged as a researcher for the Irish Centre for Manufacturing Research (www.icmr.ie). His research interests include maintainable discrete event simulation modelling, with a focus on its applications in the manufacturing sector.

PAUL LISTON is a Research Fellow at the Enterprise Process Research Centre, in Dublin City University. He holds a BEng in Industrial Engineering, an MTech in Computer Integrated Manufacturing, and a PhD from the Manufacturing and Operations Engineering department in the University of Limerick. His research interests include supply chain analysis, non-hierarchical networks, service modelling and the development of simulation-based decision-support systems. Paul is currently working with the Irish Centre for Manufacturing Research (www.icmr.ie) on an industry-led collaborative research initiative.

JOHN GERAGHTY is a Lecturer in the School of Mechanical and Manufacturing Engineering at Dublin City University, Ireland. He is a founding member of the Enterprise Process Research Centre at DCU. His teaching interests include operations research, manufacturing systems simulation, quality management and project management. His research interests include modelling and analysis of semi-conductor fabrication systems, lean manufacturing, supply-chain management and operations excellence.

PAUL YOUNG is the director of the Enterprise Process Research Centre at Dublin City University, Ireland. His teaching interests include operations research, mechanics of machines and mechanical systems simulation. His research interests range from simulation of mechanical systems to simulation and analysis of manufacturing systems.