Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

DEVELOPMENT OF AN OPEN-SOURCE DISCRETE EVENT SIMULATION CLOUD ENABLED PLATFORM

Cathal Heavey
Georgios Dagkakis
Panagiotis Barlas
Ioannis Papagiannopoulos

Enterprise Research Center
University of Limerick
Limerick, IRELAND

Sebastian Robin
Marco Mariani
Jerome Perrin

Nexedi
270 Bd Clémenceau
Lille, 59700, FRANCE

ABSTRACT

Discrete Event Simulation (DES) is traditionally one of the most popular operation research techniques. Nevertheless, organizations, and especially Small and Medium Enterprises (SMEs), are often reluctant to invest in DES projects. The lack of flexibility, high cost of developing and maintaining DES models, the high volume of data that these need in order to provide valid results and the difficulty in embedding them in real time problems, are some of the problems that deter organizations from adopting DES based solutions. DREAM is an FP7 project with the aim to develop an Open Source (OS) platform, which will confront the above issues. The DREAM architecture consists of three cloud enabled modules, a semantic free Simulation Engine (SE), a Knowledge Extraction (KE) tool and a customizable web-based Graphical User Interface (GUI). We present how these components cooperate and how an advanced user can manipulate them to develop tailored solutions for companies.

1 INTRODUCTION

Simulation is a powerful tool that provides the ability to allow practitioners to design and develop new systems, run experiments to observe performance and evaluate the outcome of alternative scenarios (Shannon 1998). Simulation can be used to study and compare alternative designs or to troubleshoot existing systems (Fowler 2004). Also, it has been demonstrated as technology for reducing costs and improving quality (Brown and Sturrock 2009). Given the fact that manufacturing systems, processes, and data are growing and becoming more complex (Ramirez and Nembhard 2004), product design, manufacturing engineering, and production management decisions involve the consideration of many interdependent factors and variables. These often complex, interdependent factors and variables are too many for the human mind to deal with at one time (Heilala et al. 2010). Thus, simulation is becoming a preferred modelling approach for a big variety of manufacturing systems, because due to its flexibility it does not require strict simplifying assumptions and the models' detail level can be adjusted according to the analysis purposes.

By reducing the programming effort and the time needed to develop a simulation model, Commercial-off-the-self (COTS) DES software packages have contributed significantly to the spread of DES in the academic and industrial community (Taylor et al. 2009). COTS software packages asset is that they offer the user tools for modelling, debugging and experimentation (Pidd and Cavalho 2006). The programming effort and the required time are considerably reduced as already developed DES objects can be manipulated through a user friendly GUI. The lack of flexibility to embed simulation into Decision Support Systems (DSS) due to the lack of modularity of COTS DES tools and to a lesser degree the high

license cost of these tools are important reasons, among many others, that hinder the widespread use of this technology in the manufacturing industry. A more sustainable choice to COTS DES packages, at least from a cost viewpoint, is represented by Open Source (OS) (Fitzgerald 2006) DES software, which has the potential to overcome the above-mentioned issues. Being available at zero license cost, the adoption of OS software should prove appealing to organizations with limited financial resources.

The DREAM ("simulation based application Decision support in Real-time for Efficient Agile Manufacturing", http://dream-simulation.eu/) project has the goal to overcome the aforementioned issues. The focus of DREAM is to engineer a semantic free open simulation application development platform to support the wide adoption of simulation based applications, a platform that will allow manufacturing companies themselves, consultancy firms, the OS community and researchers deliver beyond current state-of-the-art simulation decision support applications to the Manufacturing Sector (see Figure 1). Using OS tools a modular simulation platform is being developed that consists of a simulation engine (ManPy), a Knowledge Extraction module (KE tool) and a User Interface (DREAM.GUI). From the four industrial partners in the DREAM project three users where identified:

- Super User (SU) – she/he can get deep into DREAM code. She/he can customize simulation objects or create completely new ones and configure the KE tool and the DREAM.GUI to provide custom DREAM instances. Good coding and modeling skills are needed.
- Industrial Engineer (IE) – she/he is familiar with the systems of his organization and can use a custom DREAM instance to create/edit models. No coding, but modeling skills are required.
- End User (EU) – This would be a user who uses a DREAM simulation model for his every-day decision support, i.e. a shop-floor operator. No expertise are expected.

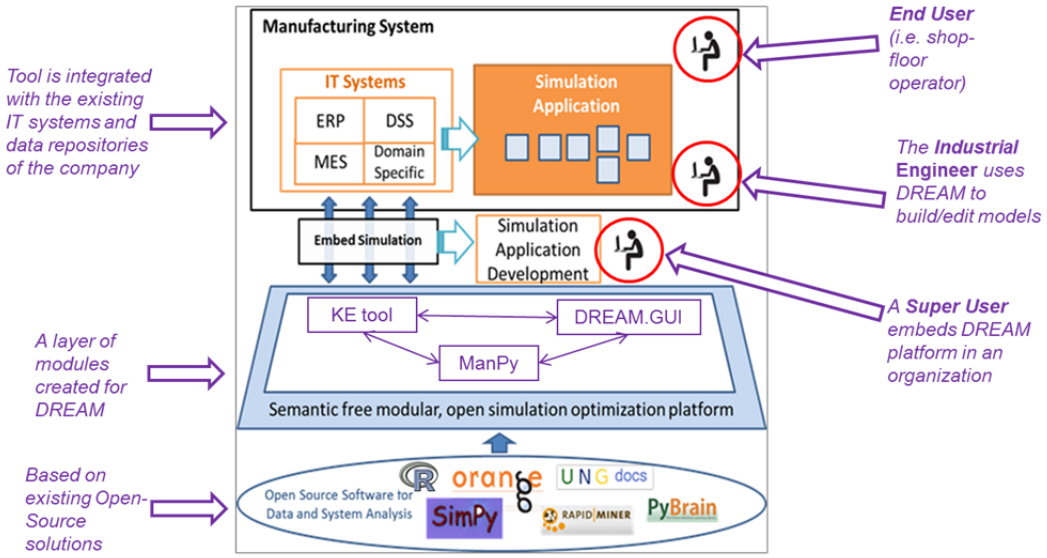

Figure 1: Identified users in the DREAM platform.

The remainder of this paper is organized as follows: in the next section we present the DREAM platform, describing the architecture and the three cloud enabled modules (ManPy, KE tool and GUI) of the platform. Then we illustrate how the platform can be customizable for different use cases and user levels providing two examples, describing how the three modules are tailored to cover these examples. A section follows next with the cloud deployment of the DREAM platform. We end the paper with conclusions and future work to be carried out within the project.

2 DREAM PLATFORM

2.1 Architecture

DREAM project aims to build a simulation application platform enabling the ease-of-use application of scenario-based DES in day-to-day business at small manufacturing companies and large industrial enterprises at reasonable costs. To fit objectives, a platform with a simulation main server as its core component is used. The high level architecture of this server is depicted in Figure 2. It is running Python and uses three sub-components, all developed for the needs of the platform. These are:

- **ManPy:** stands for "Manufacturing in Python" and is an OS repository of well-defined DES objects.
- **Knowledge Extraction (KE) tool:** a module capable of interfacing with and organization's repositories, extracting and analyzing data.
- **Graphical User Interface (GUI):** a customizable web-based tool that allows the user to build and run model and retrieve the simulation results.

DREAM modules communicate using JavaScript Object Notation (JSON) format (http://www.json.org/). A communication API has been developed in order to accommodate the needs of the platform.

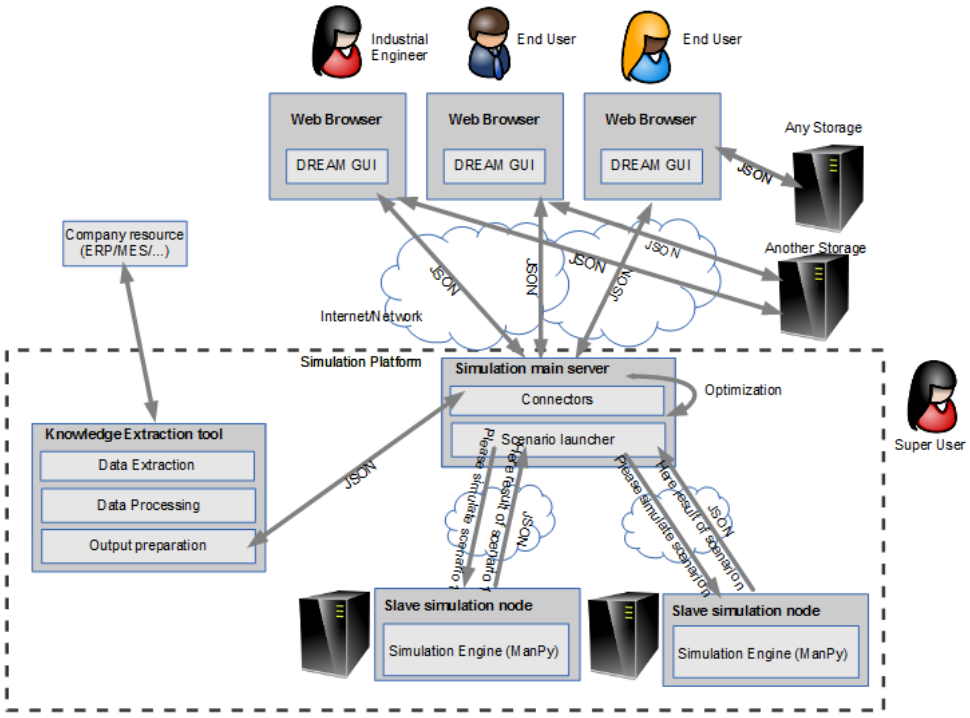

Figure 2: DREAM platform design.

The SU can manipulate these modules so that he customizes them providing tailored DREAM instances for the other levels. An IE takes a custom DREAM.GUI instance through which he can create/edit models for the types of systems of his organisation. The EU will take these tailored models that an IE created and have another DREAM.GUI instance so that he can only parameterise and run them

to help his decision support. Another possibility is that a DREAM model gets embedded to an existing decision support software, so its work is done in the background and the EU is not even aware of it, he just sees the outcome.

The simulation platform must be able to support many different users in the same time as we could see in Figure 2. User's models and preferences will be stored through the web browser to any storage on the cloud. Any user can choose a particular storage and the simulation main server will collect models and the list of simulation scenarios to launch. Then, many slave simulation nodes will do the simulation work. Being able to process any scenario for any user. For example the slave simulation node 2 could launch scenario 3 of user 1, once finished it can launch scenario 5 of user 3, then scenario 7 of user 3, and so on.

The three DREAM components will be further described in the following sub-sections of this paper.

2.2 ManPy

ManPy stands for "Manufacturing in Python" and is the simulation engine of DREAM. The first step for the design of ManPy has been the conduct of a review of the state of the art in OS DES (Dagkakis, Heavey, and Papadopoulos 2013). Among the 23 OS DES projects we reviewed was also SimPy (http://simpy.readthedocs.org). SimPy stands for "Simulation in Python" and according to its authors it is "a process-based DES framework based on standard Python". We believe Python gives the merit of being more attractive for researchers that are not necessarily software experts (Fangohr 2004), providing also means for code development that follows intuitive and clean syntax (Grandell et al. 2006). The side effect is that Python, as a scripting language, is by default slower than static languages such as C++ or Java (Dawson 2002). Another asset of SimPy, is that it is distributed in a permissive OS license (LGPL for SimPy2 and MIT for SimPy3), which makes it also possible to be expanded to closed solutions for companies (McGowan 2005). More information on why we selected SimPy as the basis of our work can be found in Dagkakis, Heavey, and Papadopoulos (2013).

From our use of SimPy we established that a user needs to be a proficient programmer even to build simple models. We wanted to provide something new; developing manufacturing objects that would be well defined and a user can connect like "black boxes", in the same fashion that COTS DES packages work. Advanced users can have deeper interaction with the platform, editing the code of the objects. This must be done at different levels, so that the needs of all the three defined user levels are addressed. ManPy as a simulation engine is written in Python and takes advantage of SimPy's efficient use of Python generators that is achieved via the SimPy.Process class. ManPy was initially developed using SimPy2 but it is being currently upgraded to the recently released version of SimPy3.

In order to have a really extensible platform, structured object orientation was a prerequisite in the design of ManPy. The high level architecture of ManPy is depicted in Figure 3. On the top of Figure 3 (and lowest level of architecture to be precise) lies SimPy, which ManPy adopted as its basis. At the next level we have five categories of abstract classes which are important in ManPy, since they offer generic methods that all ManPy objects inherit and potentially override. This is an attempt to abstract the simulation model into a set of CoreObjects that exchange Entities. Both these objects may need ObjectResources for certain operations and ObjectInterruptions may affect the availability of an object at a given simulation time. Auxiliary classes and methods help in interfacing with other software and in creating and executing simulation models. Below the abstract classes, ManPy objects that a user can take and add in a model exist. ManPy offers an API of methods that allows the user to customize the objects or create new ones and incorporate them into the platform. A detailed description of these methods is available in ManPy documentation in the DREAM repository in GitHub (https://github.com/nexedi/dream).

2.3 Knowledge Extraction Tool

Large scale organizations continuously record raw data, and are therefore able to collect large amounts of resource event information. However, usually it is difficult to extract and reuse data for future DES

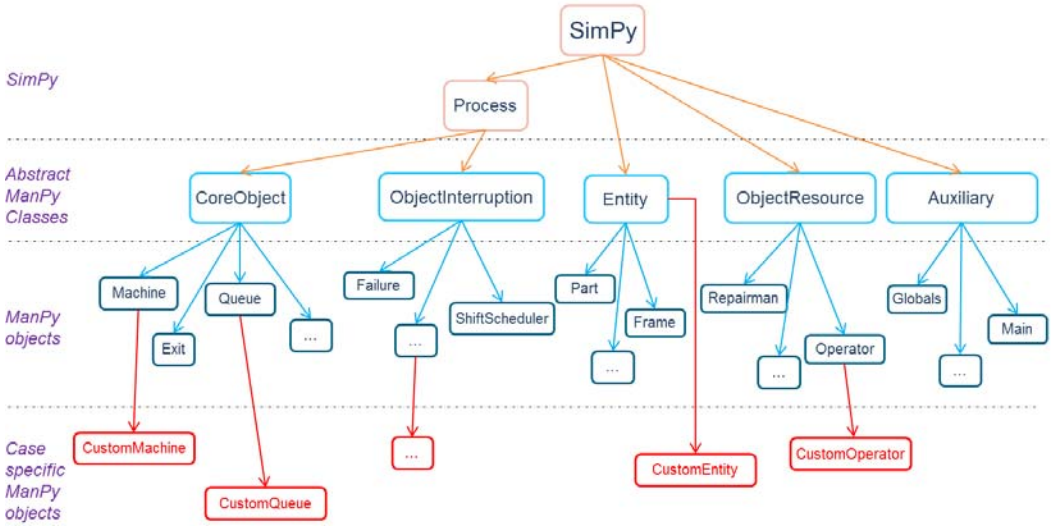

Figure 3: ManPy architecture.

projects (Johansson et al. 2007). The difficulty is based on the sharing between data files and simulation models. To address the above issue, research and development is required at the "pre-coding" phase. In this direction, we are building a tool that facilitates the extraction of required simulation data, the analysis of this data and finally the output in a format that is readable by simulation software.

The tool is aimed to link production data stored in different organization's IT-systems with the simulation software. Taking into consideration that the Python programming language is the core of our project, we decided to base the development of the tool in Python, which is a popular all-purpose scripting language. Using Python, we can have access in a wide range of functionalities provided by the different Python packages. For example, we ended up using the functionalities of R (http://www.r-project.org/) that is a state-of-the-art OS tool for statistical computing through RPy2 (http://rpy.sourceforge.net/rpy2/doc-2.1/html/), which is an interface between Python and R scripting language. This interface gives us the ability to have full access to R functions from a Python script.

Using the capabilities of the Python programming language we are able to extract and import data to the tool from different organization's data sources. The import and extraction of data to the tool is the main role of the first component "Data extraction" (see Figure 4). After the initial extraction, processing may be needed to transform the samples into a useful form. For instance, the analysis of process times data points using statistical methods in order to calculate statistical measures or fit a distribution. The above work is mainly conducted by the second component of the tool called "Data processing" (see Figure 4). The outcome of the "Data processing" component of the tool should be provided in a readable format to the simulation tool, this is exactly the role of the third component called "Output preparation"(see Figure 4). The Core Manufacturing Simulation Data (CMSD) is the primary output of this tool (SISO 2010), so Extensible Markup Language (XML) files that follow the CMSD standard can be used as input for ManPy (Barlas, Dagkakis, and Heavey 2013). Additionally, JSON files are also used facilitating the data transfer between KE tool, ManPy and DREAM.GUI (see Figure 2). The last component of the tool comes after the run of the simulation application called "Output analysis" (see Figure 4), which is concerned with the statistical analysis of the output data.

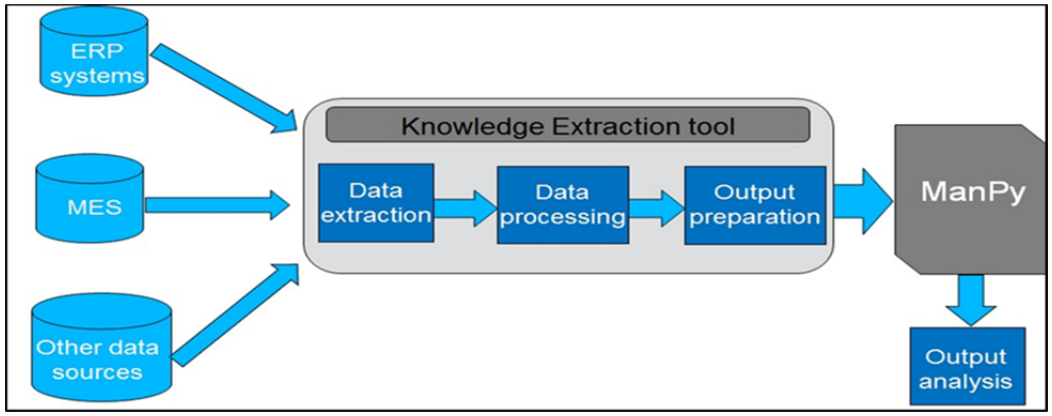

Figure 4: Knowledge Extraction tool's architecture.

The main requirements that we followed in our development process is that the KE tool should be modular, extensible, OS, fast and flexible. Taking into consideration the above we developed objects that cover the core components of the tool. Every object has its own attributes encapsulated in its code and outputs its results when is called to be executed. These different objects, as happens with the two other modules of DREAM platform, are publicly available in GitHub.

2.4 Graphical User Interface – GUI

The DREAM.GUI has been developed with two ideas in mind: firstly it should help the user to develop a simulation model visually and it should also provide the user with straightforward and understandable results of simulation experiments that should help him understanding at a glance the output of the model by showing the results in formats such as bar charts, plots, Gantt diagrams or spreadsheets. DREAM.GUI is primarily aimed for use by an IE, i.e. someone who is familiar with simulation but not an SU (see Figure 1).

DREAM.GUI is designed based on the Javascript language so as to be supported by web browsers like Firefox, Chrome, Opera, Apple Safari, etc. Searching for a Javascript flowcharting library that covers satisfactorily criteria, like project activity, documentation, interaction, look, code quality, unit testing, flowcharting, we ended up with the jsPlumb (http://jsplumbtoolkit.com/demo/home/dom.html) library Therefore, jsPlumb library is the base for the development of the Javascript User Interface.

3 EXAMPLES OF EXTENSIONS TO TAILORED SOLUTION

In the previous section we described the DREAM architecture at high level and then each of the three modules individually. In order to elaborate the concept of platform extensibility and also the accommodation of three user levels, in this section we provide two examples of how the platform can be customized for various types of systems. A high level depiction of how DREAM can be used by different users in different cases is given in Figure 5. Based on this an SU having access and knowledge of DREAM components, customizes them implementing tailored objects for an organization. For example, for Use Case 3 (Figure 5) the SU will develop manipulating the DREAM.GUI code a user interface to be used by an IE (see Figures 8 to 10 which show an IE GUI for the two Pilot Cases below). Since DREAM is an OS project, all the code is available to the SU to build the software into a decision support process. For example, in Figure 5, Use Case 2 could depict the simulation software being used within a scheduling tool, and Use Case 1 as a tool to support a supply chain decision process. These are examples of how the software could be used to support an EU within their respective decision support process.

To elaborate more on the support for an IE we provide two examples inspired by DREAM pilot cases. A brief description, reduced in complexity, is given in the bullets below:

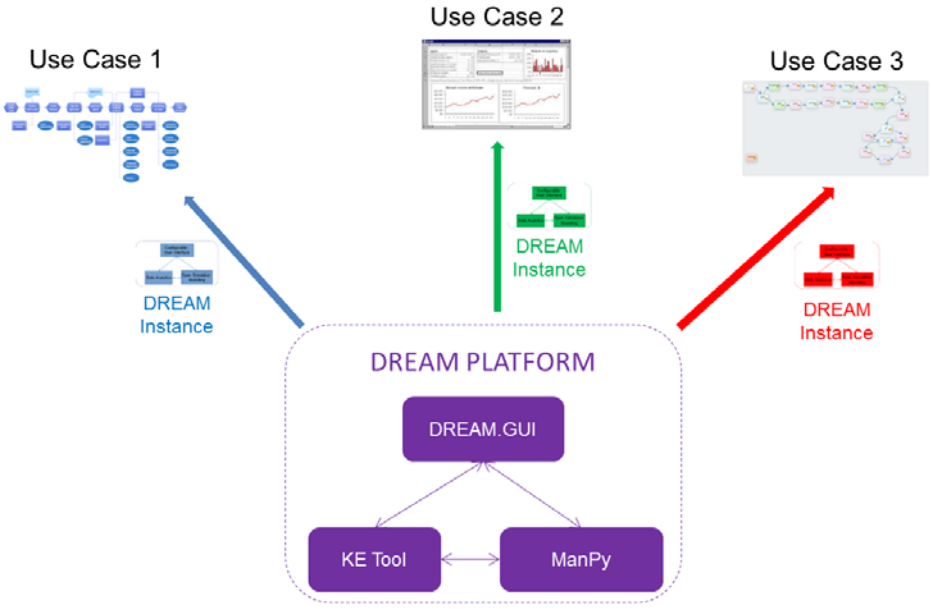

Figure 5: Customization of DREAM for specific instances.

- **Pilot Case 1:** An organization that has several production lines where the products flow in batches and can be decomposed into sub-batches and reassembled to form the original batch.
- **Pilot Case 2:** A job-shop where specific jobs flow through different stations. The sequence of stations that each job visits and the processing time depend solely on the specific attributes of the job.

Most COTS DES software allow extensions to their basic modelling elements to be developed, some even through the use of GUI functions, but most will require code to be written. To model the two pilot cases above, simulation code in ManPy is required, to extend the basic modelling elements of the ManPy simulation objects. These customized objects are listed under *Case Specific ManPy Objects* in Figure 6. For example:

- **Batch:** an Entity that contains a number of units.
- **BatchDecomposition:** a CoreObject that takes an Entity of the type Batch and decomposes it into a number of SubBatches.
- **LineClearance:** a customized buffer that does not allow SubBatches that belong to different Batches to reside in the same physical place.

To model Pilot Case 2 objects shown under *Jobshop specific ManPy objects* and *Case specific ManPy objects* in Figure 7 required to be developed. Objects developed include:

- **Job:** an Entity that holds information about the processes it has to go through and the processing time that is required in each.
- **MachineJobShop:** a Machine that reads information on the processing time from the Entity it processes.
- **OperatorManagedJob:** An operator that is not assigned to stations, but to specific Jobs.

The SU also makes tailored GUI instances for the cases. For example, in Pilot Case 1 the user has to be able to define the per-unit processing time for *Station1* (see background of Figure 8). In the left part of

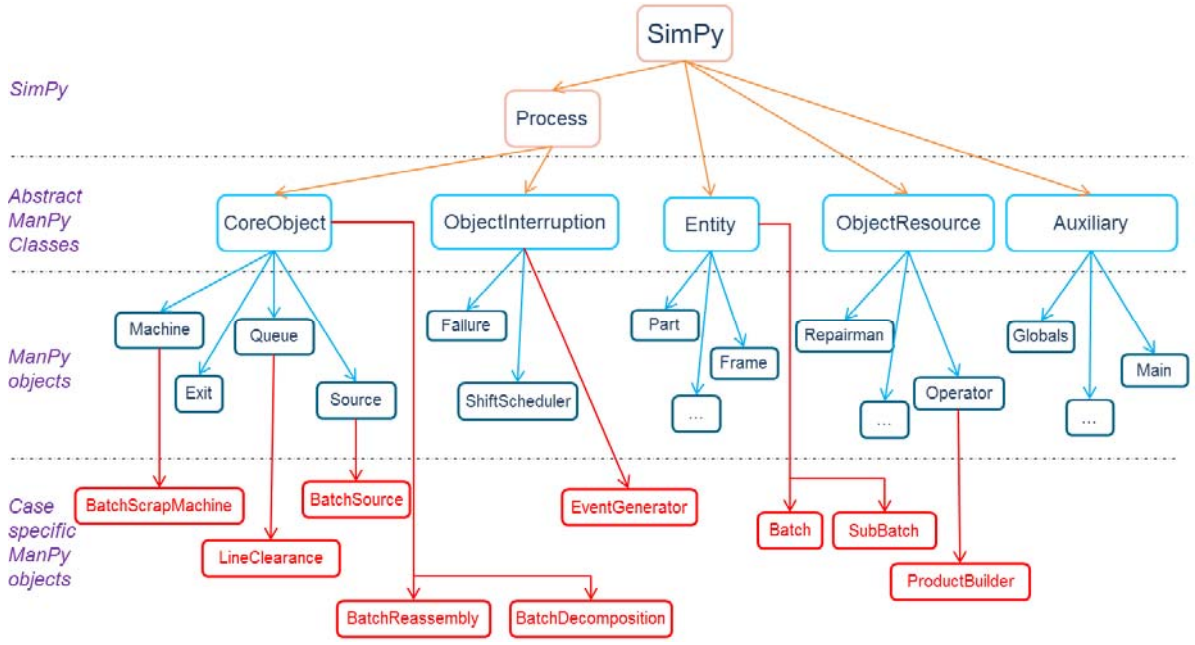

Figure 6: Extension of ManPy for Pilot Case 1.

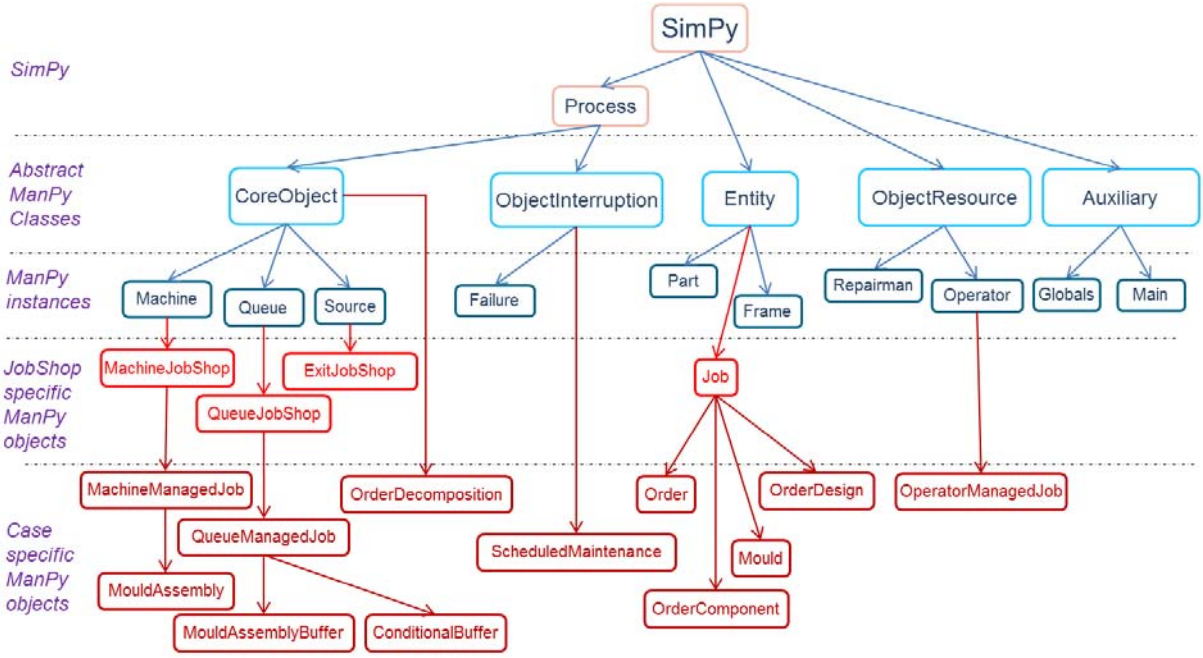

Figure 7: Extension of ManPy for Pilot Case 2.

Figure 8 we can see the menu which pops up on the double click of *Station1*. Moreover, on the right we see such values updated by the KE tool, which is also customized to be able to interface with the data repositories and perform distribution fitting.

In Pilot Case 2 there is no use in providing the same menu in the stations, since the processing time depends on the specific job. Therefore, in order to be able to define the data of the jobs, the IE has access to a spreadsheet (Figure 9), which was considered as more appropriate for such information. If this data is

digitised in the company data repositories, the KE tool can extract it and automatically update this information.

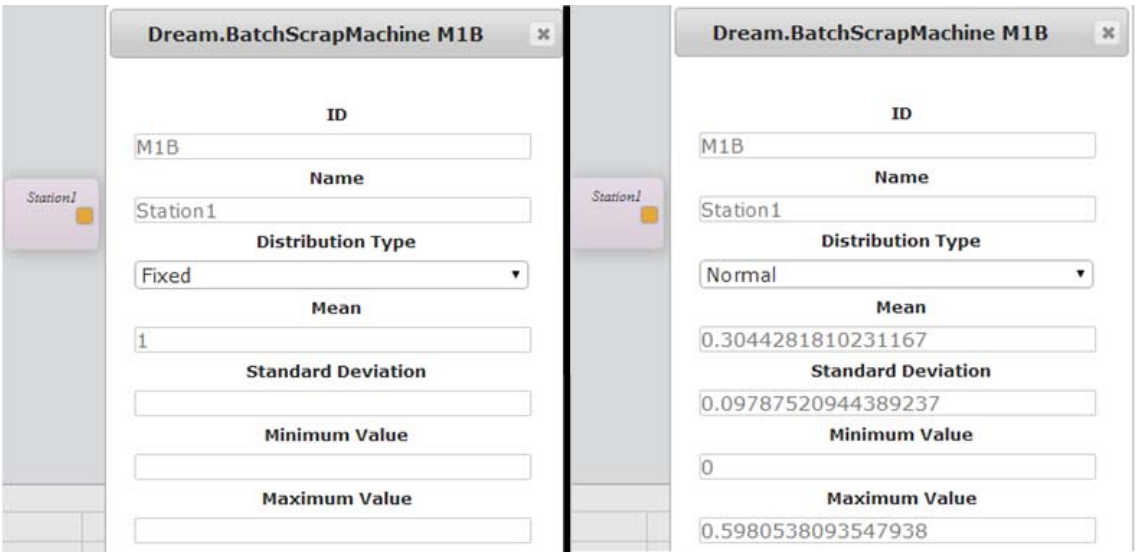

Figure 8: Processing times in Pilot Case 1 before (left) and after KE (right).

After the DES components were identified and developed in ManPy, DREAM.GUI instances were synchronized in order to provide the specific tools for every case. In Figure 10 we can see the tools for Pilot Case 1 on the left and Pilot Case 2 on the right. In this way the IE has access only to objects that are custom for the type of systems in his company. Also, the GUI instances use naming conventions that each IE is familiar with. For example, we can see that both instances provide a "Decomposition" component, which makes sense for each system. Nevertheless, these objects implement different logics, so they map to different classes in the ManPy backend, BatchDecomposition and OrderDecomposition, respectively. An IE can use these objects to easily configure models and provide them to the EU. As the DREAM project is OS an SU also may not use the IE interface and instead insert simulation within any decision support process as shown in Figure 5.

Order ID	Due Date	Priority	Project Manager	Part	Part Type	Sequence	Processing Times
Order 1	2014/03/15	1	PM1	Design	Design	CAD	6
				Part1	Basic	CAM-MILL-EDM-MILL-MASS	8-4-2-8-0
				Part2	Basic	CAM-MILL-EDM-MILL-EDM-MASS	10-7-8-4-13-8-0
				Assemble	Mould	MASS-IM	2-12
Order 2	2014/03/14	1	PM2	Design	Design	CAD	6
				Part1	Basic	CAM-MILL-EDM-MILL-MASS	8-4-2-8-0
				Part2	Basic	CAM-MILL-EDM-MASS	20-15-8-8-0
				Assemble	Mould	MASS-IM	1-12
Order 3	2014/03/15	1	PM1	Design	Design	CAD	6
				Part1	Basic	CAM-MILL-EDM-MASS	8-4-2-0
				Part2	Basic	CAM-MILL-EDM-MASS	7-15-8-8-0
				Assemble	Mould	MASS-IM	1-3

Figure 9: Spreadsheet to define jobs in Pilot Case 2.

4 CLOUD DEPLOYMENT OF THE PLATFORM

In this section, we briefly describe the methods and challenges in order to allow the cloud deployment of the platform. To provide an elastic deployment, DREAM will rely on SlapOS (https://www.slapos.org/), an OS multi-cloud platform project. By using SlapOS, a DREAM simulation model can be distributed

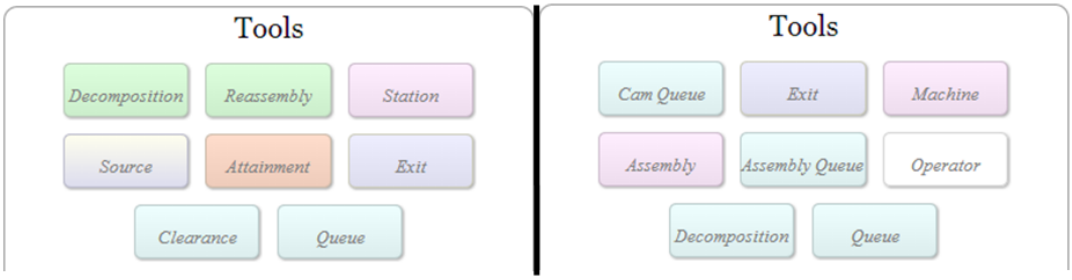

Figure 10: IE tools in Pilot Case 1 (left) and Pilot Case 2 (right).

across an assorted set of servers, virtual machines and even mobile devices. Multiple scenario can be run in parallel independently of the processor architecture or underlying virtualization technology. Billing is provided natively by SlapOS which opens the door to future industrial applications using a Software as a Service (SaaS) business model. SlapOS also includes a web-based, Platform as a Service (PaaS) environment that can be used to program and configure DREAM instances.

A SlapOS network is composed of a SlapOS master and several node servers. The nodes provide capacity in the form of partitions, segregated by the standard Unix permission system, where each partition belongs to a different Unix user. A given Service can reside in one or more partitions, and can span across nodes (as in the case of resilient applications). The nodes, commonly running a minimal Linux distribution, can be hosted in datacenters, offices or even houses.

The SlapOS Master, an ERP5 (http://www.erp5.com/) application, provides orchestration of the nodes, an application store with a web user interface, monitoring and billing. A user connecting to the SlapOS master web interface (or the equivalent command-line client) can request the provisioning of a given software release. Unless a specific node is designated, the master looks for one that has enough capacity and informs the chosen node of the request. The node builds the software release and creates the service, which is then ready to use.

Since every simulation instance is sent to ManPy using a separate JSON configuration file, parallelisation of experiments can be straightforward. In stochastic cases, where we need to run several replications of the model, the same configuration file changing only the seed of the random number generator can be sent to different nodes. The platform will gather the results and send them to KE tool for output analysis. Moreover, in optimization or scenario analysis, different configurations can be sent to different simulation nodes for evaluation.

5 CONCLUSIONS AND FUTURE WORK

In this paper we presented a new OS DES cloud enabled platform called DREAM. We started describing the DREAM architecture at high level and then detailed each of the components, namely ManPy, KE tool and DREAM.GUI. We also gave two parallel examples to elaborate on how this platform can be tailored to accommodate the needs of different user levels in specific cases. Finally, we explained the methods and challenges in order to allow the cloud deployment of the platform.

DREAM as an FP7 project is at half of its lifecycle within a 3 year project. Furthermore, DREAM as an OS project has the ambition to remain active and evolve through development of an OS community. Thus, there is a variety of future steps in our research. Our pilot cases give a variety of complex problems and suggestions for new modelling elements and methods to get incorporated into the platform. Additionally, they are used for the validation of the DREAM concept, especially the satisfaction of the needs of all the defined user categories. In the long run, more pilot cases in different domains should be tested. It is also of great importance to provide third party developers with the means to use the project or contribute to it, creating this way an OS DES community for manufacturing and logistics systems. This requires technical infrastructure, such as online training systems, a means to publish developed simulation

objects, better integration of the DREAM.GUI and ManPy, more advancement of the KE tool and also actively updated and completed documentation of all DREAM modules.

ACKNOWLEDGMENTS

The research leading to the results presented in this paper has received funding from the European Union Seventh Framework Programme (FP7-2012-NMP-ICT-FoF) under grant agreement n° 314364.

REFERENCES

Barlas, P., G. Dagkakis, and C. Heavey. 2013. "A prototype integration of ManPy with CMSD." In *Proceedings of the 27th European Simulation and Modelling Conference*, 85-90.

Brown, J., and D. Sturrock. 2009. "Identifying cost reduction and performance improvement opportunities through simulation." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill and B. Johansson, A. Dunkin, R. G. Ingalls, 2145-2153. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Dagkakis, G., C. Heavey, and C. T. Papadopoulos. 2013. "A Review of Open Source Discrete Event Simulation Software." In *Proceedings of the 9th Conference on Stochastic Models of Manufacturing and Service Operations*, 27-35.

Dawson, B. 2002. "Game scripting in Python." In *Proceedings of the 2002 Game Developers Conference*.

Fangohr, H. 2004. "A comparison of C, MATLAB, and Python as teaching languages in engineering." In *Proceedings of the 2004 International Conference on Computational Science*, 1210-1217.

Fitzgerald, B. 2006. "The transformation of open source software." *MIS Quarterly* 30(3): 587-598.

Fowler, J. W. 2004. "Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems." *SIMULATION* 80 (9): 469–476.

Grandell, L., M. Peltomäki, R. J. Back, and T. Salakoski. 2006. "Why complicate things?: introducing programming in high school using Python." In *Proceedings of the 8th Australasian Conference on Computing Education,* 71-80.

Heilala, J., J. Montonen, P. Järvinen, and S. Kivikunnas. 2010. "Decision Support Using Simulation for Customer-Driven Manufacturing System Design And Operations Planning." In *Advances in Decision Support Systems*, edited by G. Devlin, 978-953.

Johansonn M., B. Johansonn, A. Skoogh, S. Leong, F. Riddick, Y. T. Lee, G. Shao, and B. Klingstam. 2007. "A Test Implementation of the Core Manufacturing Simulation Data specification." In *Proceedings of the 2007 Winter simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 1673-1681. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

McGowan, D. 2005. "Legal aspects of free and open source software." In *Perspectives on Free and Open Source Software*, edited by J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani, 361-392. MIT Press

Pidd, M., and A. Carvalho. 2006. "Simulation software: Not the same yesterday, today or forever." *Journal of Simulation* 1(1): 7-20.

Ramirez, Y. M., and D. A. Nembhard. 2004. "Measuring knowledge worker productivity." *Journal of Intellectual Capital* 5(4): 602-628.

Shannon, R. E. 1998. "Introduction to the art and science of simulation." In *Proceedings of the 1998 Winter Simulation Conference*, edited by D. J. Medeiros, E. F. Watson, J. S. Carson and M. S. Manivannan, 7-14. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

SISO 2010. "Simulation Interoperability Standards Organization (SISO) Standard for: Core Manufacturing Simulation Data ─ UML Model." Core Manufacturing Simulation Data Product Development Group, Simulation Interoperability Standards Organization.

Taylor, S. J., N. Mustafee, S. J. Turner, K. Pan, and S. Strassburger. 2009. "Commercial-off-the-shelf simulation package interoperability: issues and futures." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill and B. Johansson, A. Dunkin, R. G. Ingalls, 203-215. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

CATHAL HEAVEY is an Associate Professor in the Department of Design & Manufacturing Technology at the University of Limerick. He is an Industrial Engineering graduate of the National University of Ireland (University College Galway) and holds a M. Eng.Sc. and Ph.D. from the same University. He has published in the areas of queuing and simulation modelling. His research interests includes, simulation modelling of discrete-event systems; modelling and analysis of supply chains and manufacturing systems; process modelling; component-based simulation and decision support systems. His email address is cathal.heavey@ul.ie.

GEORGIOS DAGKAKIS received his M. Eng in Electrical Engineering and Computer Science from the Aristotle University of Thessaloniki in 2007. He completed his M. Sc. in Informatics and Management at the same University in 2009. He is currently a PhD student in the University of Limerick. His expertise is in Discrete Event Simulation modelling. His email address is georgios.dagkakis@ul.ie.

PANAGIOTIS BARLAS received his M. Eng in Mechanical Engineering from the Aristotle University of Thessaloniki in 2010. He completed his M.Sc. in Logistics Management at the same University in 2012. He is currently a Ph.D. student in the University of Limerick. His research interests are in Discrete Event Simulation modelling and Data Analytics. His email address is panagiotis.barlas@ul.ie.

IOANNIS PAPAGIANNOPOULOS is an Electrical and Computer Engineer. After graduating from Aristotle University of Thessaloniki he undertook doctoral studies in the field of Heat Transfer in Electronics with a broader interest in simulation techniques and programming. He recently commenced postgraduate studies in the field of Discrete Event Simulation. His email address is ioannis.papagiannopoulos@ul.ie.

SEBASTIEN ROBIN is a Software Project Manager in the free software company Nexedi (Lille, France). He has an Engineering diploma from ENSIAME located in Valenciennes (France). He conducted several large ERP projects like for a central bank in Africa or in the area of Aerospace for Airbus Defense & Space. He is interested in modelling business models, on doing analysis of supply chains and manufacturing systems and to design user interfaces for end users. His email address is seb@nexedi.com.

MARCO MARIANI is a Software Developer in the free software company Nexedi (Paris, France). He has a varied experience as a full-stack web developer, ERP and system administration. Other than software development and future trends, he is interest in cognitive psychology and critical thinking. His email address is marco.mariani@nexedi.com.

JEROME PERRIN is software engineer in the free software company Nexedi (Lille, France). He was graduated with a master of information technology from Nancy's university (France). He has been involved in several large ERP deployments and some research projects. He is interested in business modeling, human computer interaction and the impact of free software on scientific community and society. His email address is jerome@nexedi.com.