Int j simul model 14 (2015) 4, 596-608
ISSN 1726-4529 Original scientific paper

AN OPEN SOURCE TOOL FOR AUTOMATED INPUT DATA IN SIMULATION

Barlas, P.; Heavey, C. & Dagkakis, G.
Enterprise Research Centre (ERC), University of Limerick, Limerick, Ireland
E-Mail: panagiotis.barlas@ul.ie, cathal.heavey@ul.ie, georgios.dagkakis@ul.ie

Abstract
Discrete Event Simulation (DES) is one of the most effective tools for planning, designing and improving material flows in production. One of the main weaknesses of operating DES is the exertion needed and costs spent on collecting and handling the input data from different organisation's data resources. To tackle the problem of the time consuming input data process for DES projects an Open Source (OS) tool, called Knowledge Extraction (KE) tool was developed. The tool reads data from several organisations' resources; analyses it and outputs it in a format that is applicable to be used by a simulation tool, all conducted in one automated process. The primary, readable to simulation software, output format follows the Core Manufacturing Simulation Data (CMSD). This paper presents the KE tool and a test implementation, as a first step towards the validation of the tool in a real case study in the medical industry.
(Received in October 2014, accepted in April 2015. This paper was with the authors 1 month for 1 revision.)

Key Words: Input Data Management, Discrete Event Simulation, Open Source, Core Manufacturing Simulation Data, Automation

1. INTRODUCTION

Manufacturing systems, processes, and data are expanding and becoming more complicated. Product design, manufacturing engineering, and production management decisions include the consideration of various complex, co-dependent issues and variables that are too complicated for the human mind to deal with at one time. DES has proved itself to be an effective tool for complex processes analysis and it is argued that the input data procedure is the most critical and time-consuming phase in DES projects [1]. The time spent on the input data procedure is typically as much as 10-40 % of the total time of a DES project [2]. Major issues that lead to this extreme time consumption, relate to low quality data, difficulty in identifying available data sources and massive manual workload to transform raw data into simulation input [3, 4]. Skoogh and Johansson [5] stated that the extensive time consumption hindrance sometimes lures organisations to select less complicated processes to analyse the data with lighter requirements on input data quality, resulting in poor and lower quality simulation results.

The aforementioned issue of the heavy time-consumption on the input data phase was the main motivation to develop a tool that tries to address this issue and generally enhances the input data phase in DES. The tool is demonstrated as a software solution, called Knowledge Extraction (KE) tool, which extracts raw data from different organization's data sources, alters this data after processing to useful simulation information and outputs this information in data formats readable and accessible to a range of DES software. Although other data formats are also implemented (JSON (JavaScript Object Notation), spreadsheet) the Core Manufacturing Simulation Data (CMSD) is suggested as the primary output format of this tool. CMSD was designed after a collaboration of NIST (National Institute of Standards and Technology) researchers with industrial partners under the guidelines of SISO (Simulation Interoperability Standards Organization) [6, 7].

The purpose of this paper is to demonstrate the KE tool presenting the use of this tool in a real case study. Note that the KE tool is not aimed to achieve the functionality requirements

of COTS (Commercial off-the-shelf) software. It is developed as an OS tool to facilitate the automation of the input data management process in DES projects, similar to the intermediary database approach (third method) presented in Skoogh, Perera and Johansson [8]. The paper is also targets to attract simulation practitioners to use the functionalities of the KE tool and of course users with programming expertise to contribute in this project adding more functionalities covering in essence the OS nature of the tool. The remainder of this paper is structured as follows: the input data management process and the already developed approaches and methodologies are presented in the next section. Section 3 describes the CMSD standard. The KE tool is introduced in section 4 which provides a description of the functionalities of the tool. A section follows that presents the test implementation of the tool in a real case study, describing how we built the CMSD information model. We end this paper with a discussion section and presenting conclusions and future research to be carried out.

2. INPUT DATA MANAGEMENT IN DES

According to Skoogh and Johansson [9] input data management is described as the whole process of arranging quality guaranteed, and simulation modified, depiction of all pertinent input data factors for simulation models. This contains recognizing appropriate input factors, gathering all information needed to signify the factors as proper input for the simulation model, transforming unprocessed data to a quality confident representation, and recording data for upcoming recommendation and reprocess. Skoogh and Johansson [9] tried to redefine adding more aspects to the input data stage, which is one of the main steps in every DES project, typically called “Data collection”; see for example the well-known and adopted methodologies proposed in [1, 10]. These methodologies usually show the input data management step as a black box without describing or explaining in depth this stage. Input data management is a time-consuming phase, which contributes on average 31 % of the project time in simulation studies [2]. Onggo et al. [4] stated analogous results, affirming that the input data phase consumes 10-40 % of a project time. The reasons are that many different aspects and parameters of production resources are included in detailed models, and that stochastic representation of simulation parameters requires lots of raw data samples. In addition, substantial care is required due to the importance of data for model quality [11]. Thus, the expression Garbage in – Garbage out (GIGO) is often used among simulation specialists.

There is a lack of publications on methodical guidelines and methodologies in the area of input data management in DES [12]. Existing publications concentrates mostly on how to exemplify large sets of raw data in simulation applications [13]. Therefore, there is a great deal of information accessible on how to choose the best statistical or empirical distribution. There have been lots of publications with guidelines and details about several statistical distributions, Maximum Likelihood Estimations (MLE) and goodness-of-fit tests. However, studies to address problems in the input data management process, using a methodical strategy, are less prominent in the literature. In this section we present some well-known and adopted approaches, methodologies for input data management for DES models.

Skoogh, Perera and Johansson [8] tackled the issue of time-consumption in input data management. This study describes four different methods to input the needed data to DES projects (see Fig. 1):

1. Direct data entry;
2. Manually populated external data source;
3. Automatically populated external data source;
4. Direct link to external sources.

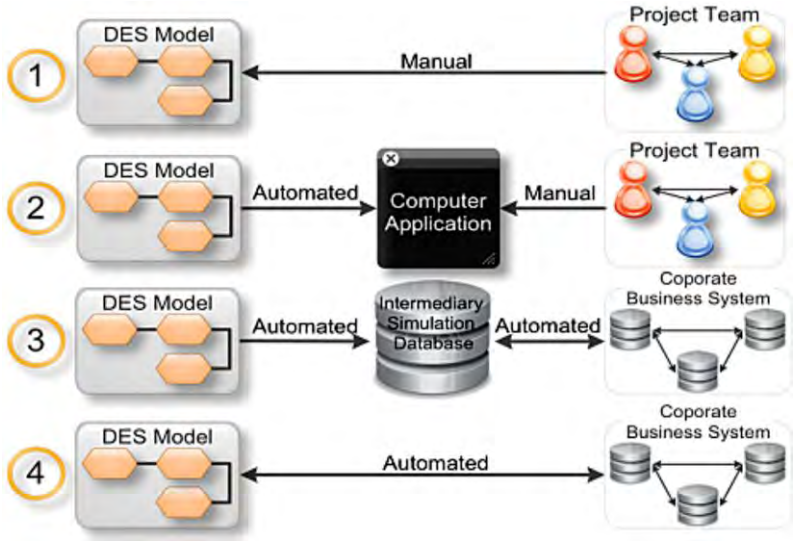

Figure 1: Possible data collection methods for model building [8].

Another well-known methodology developed based on Integration DEFintition (IDEF) [3]. The IDEF based methodology concentrates mostly on decreasing the needed time for recognition of parameters to involve in simulation models. This methodology is developed around a library of functional models and a generic data model for manufacturing environments [3]. Controllability analysis (CA) has been utilized to enlarge effectiveness mostly in problem description and data management stages of simulation projects [14]. The theory of CA is based on the Ph.D. thesis of Eloranta, with the methodology published in [15]. It is an iterative concept mainly focused on pertinent features of the problem. At each stage, the aspect of the main significance is focused upon and additionally processed so as to recognize the key factors related to the project purposes. This well-developed methodology identifies the main factors, and simplifies the data management procedure by reducing the gathering of data which is not related to the solution of the problem. More recently Skoogh and Johansson [9] presented a structured methodology comprising 13 activities and their inter-connections. The suggested methodology is based on 15 interviews where simulation practitioners contributed with their experiences from DES projects performed between 1999 and 2006. Their work mostly concentrated on connecting all activities inside the input data management process in an effective manner. They suggested the use of CMSD, designed by the Simulation Interoperability Standards Organization (SISO) [16] in order to support the identification and decision of the parameters required for the simulation application. A more generic input data methodology is proposed by Bengtsson et al. [17], which uses the GDM-Tool [18] that implements CMSD.

3. CORE MANUFACTURING SIMULATION DATA

To address interoperability issues between simulation and manufacturing applications the Core Manufacturing Simulation Data (CMSD) standard was created. Researchers at the National Institute of Standards and Technology (NIST) in collaboration with industrial partners developed the Core Manufacturing Simulation Data (CMSD) Product Development Group (PDG) under the guidelines and procedures of the Simulation Interoperability Standards Organization (SISO).

According to the "Standard for: Core Manufacturing Simulation Data – UML Model" [7] published in September 2010, the purposes of this standard include:

- enabling data exchange between simulation applications and other software applications;
- supporting the construction of manufacturing simulators;

- supporting the testing and evaluation of manufacturing software;
- enabling greater manufacturing software application interoperability.

The CMSD information model is given in two modelling languages: (1) the information model defined using the Unified Modeling Language (UML) and (2) the information model defined using the eXtensible Markup Language (XML) [6]. The exchange of data between simulations is allowed through the exchange of XML instance documents that follow the CMSD XML Schemas. The UML diagram and XML schemas are intended to be equivalent representations of the same CMSD model.

The adoption of a reusable, neutral standard interfaces are crucial for the automation of the process of inputting data from existing data sources. This paper here presents a tool and a case study that uses the CMSD standard. Nevertheless, it should be stated that the CMSD standard concentrates on data exchange and does not contain information on data processing to allow this information to be used directly by a DES tool. Additionally, CMSD cannot describe complicated concepts in manufacturing systems, e.g. line clearance and batch decomposition [19].

4. INTRODUCTION OF THE KE TOOL

A widely quoted rule in the simulation literature is the "40-20-40" rule, which states that 40 % of the time in a simulation project is spent at the "pre-coding stage", 20 % coding and 40 % on the analysis phase. To address the issue of efficient implementation of simulation technology in manufacturing companies, research and development is required at the "pre-coding" phase. In our effort we address part (input data preparation) of this issue by advancing and developing further methods for modelling and capturing systems knowledge. In this direction, we built 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. As it is highlighted in the Introduction in Section 1 the tool is developed as an OS tool that supports the input data management for DES. The KE tool is designed and developed as analogous to the intermediary simulation database [8] (approach 3 in Fig. 1).

The tool connects raw data recorded in different IT-systems in manufacturing with simulation software. We built the tool using different Python libraries; focusing the development of the tool on RPy2. RPy2 (http://rpy.sourceforge.net/rpy2.html) is an interface between Python, which is a popular all-purpose scripting language used by scientists, analysts and engineers doing scientific and technical computing, and R, which is a scripting language popular for data analysis, statistics and graphics. This interface gives us the ability to have full access to R functions from within a Python script. It is well developed and quite active as a project retaining a mailing list and providing thorough documentation. It can be used under the GNU Lesser General Public License (LGPL) which makes it feasible to be also used in a proprietary project [20]. RPy2 built on Python offers many convenient features for building code such as efficient list processing and flexible type casting. The drawback is that Python as a scripting language is slower than static languages such as C++ or Java [21], however in the application presenting here it is not a major issue and this drawback can be outweighed by the fact that Python is easier both to learn and use than C++ or Java.

4.1 Architecture

Data is split into three categories based on availability and collectability [22]. The first data category is already available in a company's IT resources, for instance Enterprise Resource Planning (ERP) systems, or Manufacturing Execution Systems (MES), or other database. The second category data requires work because it has to be collected throughout the simulation project and the third category, which is data neither available nor collectable. In the first

category, data can be located as raw data, for example, automatically gathered scrap quantity or process times in a station. These types of data samples are extracted from a company's ERP, or MES, or database.

Using the capabilities of the Python programming language we are able to extract and import data to the tool from different data sources. The import and extraction of data to the tool is the main role of the first component "Data extraction" (see Fig. 2). After the initial extraction, some process may be needed to transform the samples into a useful form. For example, to acquire a process time of a station in a production line, the stop time has to be subtracted from the start time. Additionally, after having the actual process time data points, this data should be analysed 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 Fig. 2). The outcome of the "Data processing" component of the tool should be provided in a readable format to simulation software, this is exactly the role of the third component called "Output preparation" (see Fig. 2). The CMSD standard is suggested as the primary output of this tool, so Extensible Markup Language (XML) files that follow the CMSD standard can be used as input for the SE. Barlas et al. [23] achieved the integration of a new OS DES library called ManPy ('Manufacturing in Python') [24] with CMSD.

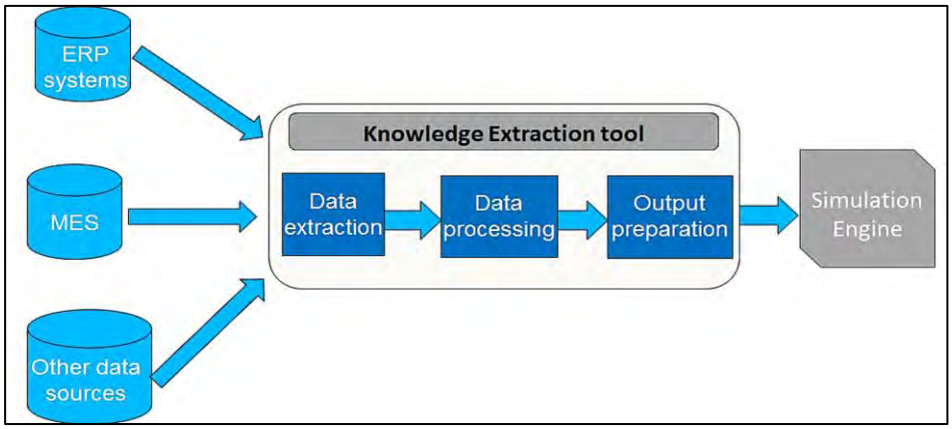

Figure 2: Knowledge Extraction tool's architecture (adopted by [18]).

4.2 Design description / Core objects

The main requirements that we followed in the development process is that the KE tool should be modular, extensible, OS 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. The development of the tool is an on-going activity and we are planning to expand the current status with more objects giving extra functionalities to the tool.

The tool is built in such a way that the data input, the processing of this data and the output preparation, are conducted by a separate script. We refer to this script as the "main script". This main script is the only one to be changed in order to read data from different files and transform the raw data in order to be handled by the "Data processing" objects of the tool. The objects of the above three components can be connected as black boxes in order to form the main script. Therefore, this main script calls different objects in order to give as an output the actual selected data exchange file format, updated with the processed data. In this way there is the possibility for a developer to configure the main script so that it reads data from a Graphical User Interface (GUI) or a company's data resources such as MES or ERP systems. Furthermore, a user with programming expertise can manipulate the Application

Programming Interface (API) that the KE tool offers, to customize the objects or create completely new ones and incorporate them into the library.

Table I: Brief description of KE tool's different objects.

Data extraction
ImportExceldata: retrieves data from a MS Excel file and imports this in the tool.
ImportCSVdata: retrieves data from a CSV file and imports this in the tool.
ImportDatabase: allows the user to connect with a database given that the user has provided the connection data in a .txt file.
Data processing
StatisticalMeasures: calculates a variety of basic statistical measures in a given data sample.
DistributionFitting: fits statistical distributions in a given data sample using both Maximum Likelihood Estimation (MLE) and Kolmogorov-Smirnov test. The object can identify and fit data using the following distributions: Normal; Exponential; Poisson; Uniform; Triangular; Beta; Gamma; Logistic; Geometric; Cauchy; Log-Normal and Negative Binomial.
ReplaceMissingValues: replaces missing values in a given data sample. Using this object the modeller is able to replace the missing data with zero, mean value or median of the non-missing values, and totally erase the missing data.
Transformations: calculates a variety of transformations in a given data sample.
DataManipulation: a series of manipulations in a given data set is conducted applying this object.
Output preparation
CMSDOutput: exports the outcomes of the statistical analysis into an XML file that follow the CMSD standard specification.
JSONOutput: exports the outcomes of the "Data processing" component into JSON (JavaScript Object Notation) file format.
ExcelOutput: another export offered by the tool is to MS Excel files.

Fig. 3 depicts a graphical representation of workflows of stepwise activities of the tool. It is a UML activity diagram that intends to illustrate the sequence of the activities needed to be linked in order to form the KE tool main script. The rounded rectangles represent the different states (Import data, Data pre-processing etc.), whereas the rounded rectangles with the bold italics letters represent the activities, so the different objects of the tool.

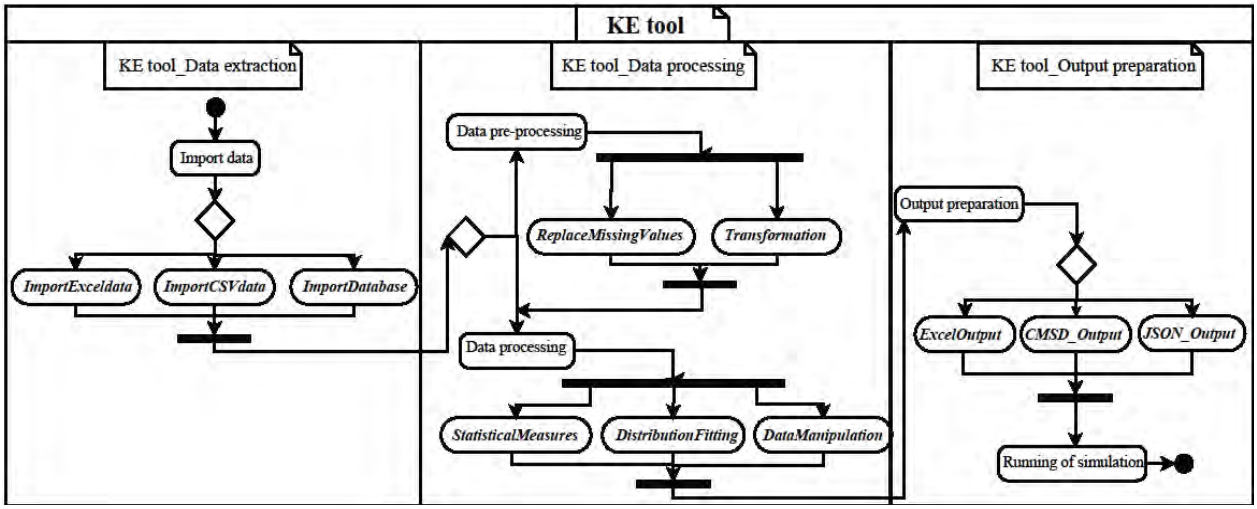

Figure 3: Activity diagram of the KE tool.

More information about the different developed objects is available in GitHub at the following URL (https://github.com/nexedi/dream/tree/master/dream/KnowledgeExtraction). In the repository one can find and download the developed objects themselves, examples with the development of the KE tool main script in different simulation topologies and a thorough documentation of the tool. The code kept under version control with Git (http://git-scm.com/), the user can clone and manipulate the different versions of the code through the project repository in GitHub.

5. TEST IMPLEMENTATION OF THE KE TOOL IN A CASE STUDY

5.1 Description of case study

Several production lines operate in the medical device fabrication facility. The pilot line chosen as a use case for supporting the expansion and validation of KE tool is located in a clean room where other lines also operate. Even though there are dimensional variations in the products fabricated on this line, the pilot line can be considered product dedicated, which means the production flow is the same for the all products. The product in question is a medical device for use in operating theatres. It consists of a two part stent; a 1.8 m flexible shaft, and a balloon capable of inflation within a human vein. The length of the shaft impacts on product storage capacity along the line as the shaft is kept straight for most of its processing. Items are produced in batches of 100 units.

The industrial company has conducted simulation based projects in the past using COTS DES tools, but wanted to develop real-time based DES support. One of the identified reasons for this failure is the lack of integration between stored data in different IT systems and the simulation model used. It has been proved time consuming and cumbersome to parameterise the model in real time situations, where the decision support is required within a short time period.

Input data in the company is stored in various manufacturing information systems, such as an ERP, MES and an auxiliary database system. The company in general is data rich and wants to improve the interaction with the stored data. To prepare the input data for the DES model, extraction and processing of the relevant manufacturing information from these databases will be required. This input data process will be both time-consuming and work-intensive. Therefore, in our case one of the company's requirements is to achieve the integration of the required input simulation data with the SE. Towards this goal, the use of the KE tool with the CMSD standard is used.

Fig. 4 illustrates the process flow diagram of the production line. The process flow consists of four sections; these sections are denoted in Fig. 4 as PA, PB, PC and PD.

Figure 4: Process flow diagram of the production line.

The first section has two lines (PA1, PA2) operating in parallel in series (see Fig. 4); each one of these lines has three stations in series (P1, P2, P3 and P4, P5, P6). Initial processing of the product occurs here before it proceeds in section PB. Section PA stations must operate in sequence on the product. Section PB has just one station and it is followed by section PC, which consists of two stations (P8, P9) that operate in parallel. The final section of the line contains two stations (P10, P11) that also operate in parallel. Each one of these eleven processes contains information related to processing times, scrap quantity, buffer capacity and operation details.

As happens for most of the lines in the plant, the process is labour intensive; each station requires one or more operators. The presence of more than one operator per station is mandatory in the last section (PD) where three operators are required to run the two parallel stations. The production is carried out based on two daily shifts running on weekdays; in order to facilitate Work-In-Process (WIP) balancing, production capacity is suitably adjusted in different shifts. For example, with PB the bottleneck in the line, one of the sub-lines in section PA is shut down during the evening shift in order to reduce the WIP built up in section PB during the day shift.

This case study uses different data sources to record operational information. Raw data from the production line are recorded in batch level on the local CAMSTAR MES (http://www.camstar.com/) platform servers and transferred every 3-6 minutes to a local SAP server. Raw data are extracted in spreadsheet files coming from the local MES server. As it is stated the KE tool is not constrained to these files, the tool is able to extract data from databases and other data formats.

The KE tool interacts with the data file executing the three main components “Data extraction”, “Data processing” and “Output preparation” (see Fig. 2). The Python objects needed for the three components incorporated that form the KE tool are included in the main script for this case study. The needed objects for this case study conduct the actions depicted in Fig. 5. Starting with the data extraction from spreadsheets, data tables are developed to be handled by RPy2, missing values are replaced since some cells were blank (we choose to replace these values with the mean value from the other data points (see ***ReplaceMissingValues*** object in Table I)), the Kolmogorov-Smirnov distribution fitting test are ran, then the form of the statistical representations in XML that follows the CMSD specification are generated and finally this information is exported to the SE. The simulation application can automatically access this XML file during simulation runs.

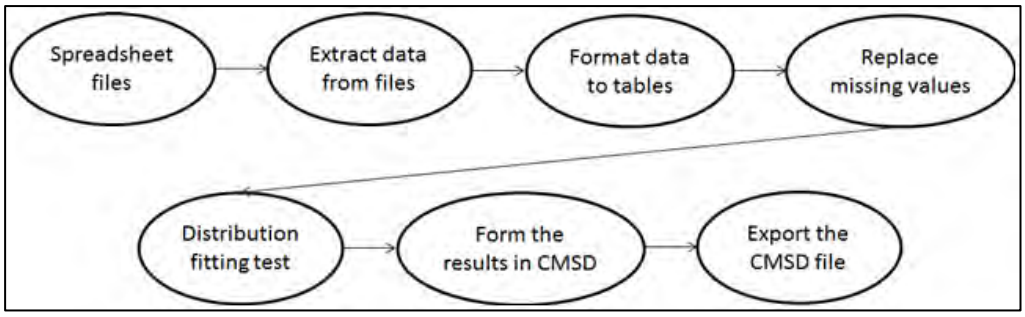

Figure 5: Example of actions conducted by Python objects in the main script.

In this case study we use the CMSD standard to enable the efficient data exchange between the simulation application and the KE tool. In addition, the already processed data can be exported to JSON and spreadsheet files. Moreover, in this case study ManPy was used to demonstrate data exchange with the SE. Therefore we developed a translator between the CMSD and ManPy [23]. The translator is an independent Python script that gets as input the CMSD XML file and translates the data from this file to meaningful information for ManPy.

The test implementation of the KE tool is performed achieving the execution of the KE tool main script and providing the CMSD document as input to ManPy.

The main script of the KE tool for this particular concept can be found as one of the examples of the KE tool in GitHub at the following URL (https://github.com/nexedi/dream/tree/master/dream/KnowledgeExtraction/KEtool_examples/ProductionLine).

More information for this example along with the exported CMSD information model can be found in the KE tool's documentation. The following subsection describes the development of the CMSD information model.

5.2 CMSD information model

The CMSD standard offers a variety of classes which can be used for representing data. The data section in the CMSD file begins with a list of part types and resources, followed by a process plan and process definitions. Resource information describes the people and equipment that perform manufacturing activities. In CMSD the resource types include machines, station, employees of the production line. The process plan information specifies the set of production activities needed to transform materials and subcomponents into finished products. Therefore, process plans are intended to specify part routing information. The processes in the process plans indicate each task that will be performed to create a part and the resources machines/employees used to execute these tasks.

The production planning information contains classes and relationships to create plans that describe the sequence of steps to manufacture products using the available resources. It defines information such as orders, resources, and operation time description. Properties like MTTF (Mean Time To Failure), MTTR (Mean Time To Repair), scrap quantity, etc. used in CMSD information model to define characteristics and capabilities of equipment and employees.

Fig. 6 shows the process definition for station P9, which is one of the two parallel stations in the PC section (see Fig. 4). The operation time is specified by the Gamma statistical distribution (with shape 36.13 and rate 123.27) based on operation time information provided in the spreadsheet file of this example. It is worth noting that the statistical distributions in the CMSD file will be updated as the provided data sets are updated.

A process plan object contains one or more process objects (see Fig. 6). Each process object may represent either an individual process or a process group. The process plan indicates which process executes first. A process group indicates that a group of processes either executes in a sequence (sequence group), only one process in the group executes (decision group), or all processes in the group execute (concurrent group). With the above rules the CMSD approach provides flexibility to define the common manufacturing relationships but as it is stated in Section 3 it cannot describe complicated concepts (e.g. line clearance, batch decomposition).

Fig. 7 depicts a part of the process plan definition (ProcessPlan:PPPlan1). In this part the PFirst process is defined. The PFirst process is defined as a sequence of four subgroups (PA, PB, PC, PD) (see Fig. 4). Based on the process flow diagram in Fig. 4, the process plan that describes the routings through the production line is: PPPlan1= (P1, P2, P3, P4, P5, P6, P7, PFirst, PA, PB, PC, PD, PA1, PA2), where PFirst, PB, PA1 and PA2 describe a sequence process group containing processes, while PA, PC and PD describe a decision process group containing processes:

- PFirst={PA,PB,PC,PD},
- PA={PA1|PA2},
- PB={P7},
- PC={P8|P9},

- PD={P10|P11},
- PA1={P1,P2,P3},
- PA2={P4,P5,P6}.

```
<Process>
    <Identifier>P9</Identifier>
    <Description>StationName</Description>
    <ResourcesRequired>
        <Description>The employee performing the operation.</Description>
        <Resource>
            <ResourceIdentifier>E</ResourceIdentifier>
        </Resource>
    </ResourcesRequired>
    <ResourcesRequired>
        <Description>The resource where the operation is being performed.</Description>
        <Resource>
            <ResourceIdentifier>resource9</ResourceIdentifier>
        </Resource>
    </ResourcesRequired>
    <OperationTime>
        <Unit>minute</Unit>
        <Distribution>
            <Name>Gamma</Name>
            <DistributionParameter>
                <Name>shape</Name>
                <Value>36.13</Value>
            </DistributionParameter>
            <DistributionParameter>
                <Name>rate</Name>
                <Value>123.27</Value>
            </DistributionParameter>
        </Distribution>
    </OperationTime>
</Process>
```

Figure 6: Process definition in XML-based CMSD.

```
<ProcessPlan>
    <Identifier>ProcessPlan:PPPlan1</Identifier>
    ...
    <Process>
        <Identifier>PFirst</Identifier>
        <RepetitionCount>1</RepetitionCount>
        <SubProcessGroup>
            <Type>sequence</Type>
            <Process>
                <ProcessIdentifier>PA</ProcessIdentifier>
            </Process>
            <Process>
                <ProcessIdentifier>PB</ProcessIdentifier>
            </Process>
            <Process>
                <ProcessIdentifier>PC</ProcessIdentifier>
            </Process>
            <Process>
                <ProcessIdentifier>PD</ProcessIdentifier>
            </Process>
        </SubProcessGroup>
    </Process>
```

Figure 7: Part of process plan definition in XML-based CMSD.

6. DISCUSSION

This paper describes and demonstrates the use of the KE tool as a tool to achieve the automation of simulation input data into DES models. A necessary prerequisite is the availability of production raw data in structured data sources. The KE tool is a module of the simulation based application platform called DREAM (http://dream-simulation.eu/) is an OS and publicly available tool in GitHub. The authors' main targets were to demonstrate a tool

that facilitates the input data phase in simulation applications and through this demonstration to attract users either to simply use the tool or to expand it by writing code to add more functionalities. In the presented case study, CMSD is used to exchange data with the simulation model. CMSD enables automated import of processed information to a large variety of COTS simulation software, such as Plant Simulation, Arena and QUEST. Therefore, using CMSD doesn't make the tool reliant on particular simulation software, in this case study we utilise the capabilities of a new SE, called ManPy. Additionally, the KE tool is able to export the simulation input data to JSON and spreadsheet files.

The performance of the KE tool is validated against the conventional manual approach by comparing the time consumption and the quality of the input simulation data. The conventional procedure consists of three main steps; these are Data Cleaning, Data Identification and Distribution Fitting. The first two steps were performed using Microsoft Excel while the third step were conducted using EasyFit. The comparison between the automated approach, through the use of the KE tool, and the manual approach came from the measurement of the time consumption and the output simulation results between the manual and the automated approach. In order to test the quality of input data we compare the output simulation results using three metrics, these are the throughput expressed in units, the takt time and the lifespan. The results from our experiments, reporting a time reduction of 81 % compared to a manual approach, proves that automation can drastically decrease the time-consumption in input data process. The quality of input data is proven to be similarly reliable as for the manual approach. It is worth noting that as a result of the validation process, one bug in the KE tool that would cause an error in the distribution fitting has been identified and fixed. This shows the importance of comparing the KE tool performance against the manual approach and subsequently COTS packages (Microsoft Excel, EasyFit); comparison analyses based on sensible experimental plans prove fundamental in order to verify the KE tool's logic and set effective development guidelines.

The KE tool is being developed as analogous to the third method presented in [8]. The case study presented should be considered as proof of the KE tool for the automation of input data for DES. Research studies that also contribute in this area are presented by [25] and [26]. One of the main differences of the KE tool presented in this paper and these two studies is that the tool presented here provides "Data processing", in which the raw data is altered to map onto simulation input data, for example through distribution fitting. The provision of features for data processing is considered crucial since in many cases IT systems, such as ERP, MES or databases, lack the means to provide the required input for DES models.

In contrast to [18] the major advantage of this tool is that it is OS (GNU Lesser General Public License (LGPL)), so easily accessible by anyone interested in developing the concept of automating simulation input data. The KE tool is also developed using Python, a language that is widely being used by scientist due to the ease of programming [27]. The different objects of the three main components can be interconnected forming the KE tool main script, giving the user the ability to create a tailored solution based on his/her needs. Nevertheless, since it is a newly developed tool work needs to be done for further development. It is prioritised as the next step the development of a user friendly GUI as a front-end of the tool. Good inspiration for the development of the GUI is the user interface demonstrated by Skoogh et al. [27] for the GDM-Tool. Using this interface the user will be able to execute the components of the tool through a GUI that calls the different objects in the back-end.

7. CONCLUSIONS

In this paper we present a new developed OS tool that offers functionalities that cover the three phases of the input data management (Data collection, Data processing and Data

interface) defined by Skoogh and Johansson [9]. We also present the use of this tool in a real case study and tested the implementation using ManPy as the SE.

We would like to highlight that the KE tool targets the automation of the input data phase in DES, requiring just from the simulation engineer to specify the location of the data sources in his or her system. After setting the location of the data sources, the objects from the different components of the tool are executed and the CMSD information model will be updated with the most recent raw data. If data is available in a structured database this tool will reduce the time needed for the input data process and since the input data phase consumes 10-40 % of the total project time, a potential reduction on the required time for the input data process will automatically reflect a decrease in the total required time to apply a DES project.

The KE tool is being currently validated through two additional industrial pilot cases, which also dictate requirements for the technical characteristics of the tool. As future research, we highlight the activities aiming to establish a mature structure for the automation of the input data management process. To start with the expansion of the KE tool adding more Python objects and consequently new features in the four components of the tool. As indicated above the development of a GUI for the tool will enhance the functionalities offering an easy to handle solution. Moreover, additional case studies are needed in order to test and validate the tool and examine the possibility to achieve automated simulation data input. Finally, as a future step there is a need to integrate the KE tool with COTS simulation software as a proof-of-concept.

ACKNOWLEDGEMENTS

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. The authors would also like to thank Frank Riddick, who is computer scientist in the Manufacturing Simulation and Modeling Group in The National Institute of Standards and Technology (NIST), for his valuable contribution on the development of the CMSD information model.

REFERENCES

[1] Banks, J.; Carson, J. S.; Nelson, B. L. (1996). *Discrete-Event System Simulation*, Prentice-Hall, Upper Saddle River

[2] Skoogh, A.; Johansson, B. (2007). Time-consumption analysis of input data activities in discrete event simulation projects, *Proceedings of the 2007 Swedish Production Symposium*

[3] Perera, T.; Liyanage, K. (2000). Methodology for rapid identification and collection of input data in the simulation of manufacturing systems, *Simulation Practice and Theory*, Vol. 7, No.7, 645-656, doi:10.1016/S0928-4869(99)00020-8

[4] Onggo, B. S. S.; Hill, J.; Brooks, R. J. (2013). A survey on data identification and collection in simulation projects, *Proceedings of the 27th European Simulation and Modelling Conference*, 23-25

[5] Skoogh, A.; Johansson, B. (2009). Mapping of time-consumption during input data management activities, *Simulation News Europe (SNE)*, Vol. 19, No. 2, 39-46

[6] Leong, S.; Lee, Y. T.; Riddick, F. (2006). A Core Manufacturing Simulation Data Information Model for Manufacturing Applications, *Simulation Interoperability Workshop*, Simulation Interoperability and Standards Organization, 1–7

[7] 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

[8] Skoogh, A.; Perera, T.; Johansson, B. (2012). Input data management in simulation – Industrial practices and future trends, *Simulation Modelling Practice and Theory*, Vol. 29, 181-192, doi:10.1016/j.simpat.2012.07.009

[9] Skoogh, A.; Johansson, B. (2008). A methodology for input data management in discrete event simulation projects, *Proceedings of the 2008 Winter Simulation Conference*, 1727-1735
[10] Law, A. M.; Kelton, W. D. (2000). *Simulation Modelling and Analysis* (3rd ed.), McGraw-Hill, Boston
[11] McNally, P.; Heavey, C. (2004). Developing simulation as a desktop resource, *International Journal of Computer Integrated Manufacturing*, Vol. 17, No. 5, 435-450, doi:10.1080/0951192031000165428 3
[12] Hollocks, B. W. (2001). Discrete-event simulation: an inquiry into user practice, *Simulation Practice and Theory*, Vol. 8, No. 6-7, 451-471, doi:10.1016/S0928-4869(01)00028-3
[13] Robinson, S. (2004). *Simulation: The Practice of Model Development and Use*, John Wiley & Sons Ltd, Chichester
[14] Lehtonen, J.-M.; Seppala, U. (1997). A methodology for data gathering and analysis in a logistics simulation project, *Integrated manufacturing systems*, Vol. 8, No. 6, 351-358, doi:10.1108/09576069710188760
[15] Eloranta, E.; Räisänen, J. (1986). Reconsidering ten conventions of production management, *Proceedings of the Conference on New Technologies for Production Management Systems*, Tokyo, 23-38
[16] Lee, Y.-T. T.; Riddick, F. H.; Johansson, B. J. I. (2011). Core Manufacturing Simulation Data – a manufacturing simulation integration standard: overview and case studies, *International Journal of Computer Integrated Manufacturing*, Vol. 24, No. 8, 689-709, doi:10.1080/0951192X.2011.574154
[17] Bengtsson, N.; Shao, G.; Johansson, B.; Lee, Y. T.; Leong, S.; Skoogh, A.; Mclean, C. (2009). Input data management methodology for discrete event simulation, *Proceedings of the 2009 Winter Simulation Conference*, 1335-1344
[18] Skoogh, A.; Johansson, B.; Stahre, J. (2012). Automated input data management: evaluation of a concept for reduced time consumption in discrete event simulation, *Simulation*, Vol. 88, No. 11, 1279-1293, doi:10.1177/0037549712443404
[19] ISO 13485:2003 – Medical devices – Quality management systems – Requirements for regulatory purposes
[20] Fogel, K. (2005). *Producing open source software: How to run a successful free software project*, O'Reilly Media, Sebastopol
[21] Dawson, B. (2002). Game scripting in Python, *Proceedings of the 2002 Game Developers Conference*, San Jose
[22] Robinson, S.; Bhatia, V. (1995). Secrets of successful simulation projects, *Proceedings of the 1995 Winter Simulation Conference*, 61-67
[23] Barlas, P.; Dagkakis, G.; Heavey, C. (2013). A prototype integration of ManPy with CMSD, *Proceedings of the 27th European Simulation and Modelling Conference*, 85-90
[24] Dagkakis, G.; Heavey, C.; Robin, S.; Perrin, J. (2013). ManPy: An open-source layer of DES manufacturing objects implemented in SimPy, *8th EUROSIM Congress on Modelling and Simulation (EUROSIM 2013)*, 357-363, doi:10.1109/EUROSIM.2013.70
[25] Randell, L. G.; Bolmsjö, G. S. (2001). Database driven factory simulation: a proof-of-concept demonstrator, *Proceedings of the 2001 Winter Simulation Conference*, 977-983
[26] Ingemansson, A.; Ylipää, T.; Bolmsjö, G. S. (2005). Reducing bottle-necks in a manufacturing system with automatic data collection and discrete-event simulation, *Journal of Manufacturing Technology Management*, Vol. 16, No. 6, 615-628, doi:10.1108/17410380510609474
[27] Fangohr, H. (2004). A comparison of C, Matlab, and Python as teaching languages in engineering, Bubak, M.; van Albada, G. D.; Sloot, P. M. A.; Dongarra, J. (Eds.). *Lecture Notes on Computational Science 3039, ICCS 2004*, 1210-1217, Springer-Verlag, Berlin