KE tool documentation

Contents

1 Introduction and Scope

The KE (Knowledge Extraction) tool is a set of objects built using different Python libraries but mainly RPy (http://rpy.sourceforge.net/legacy/). The current version of the KE tool is based on RPy2 (http://rpy.sourceforge.net/), which is a redesign and rewrite of RPy.

The scope of the tool is to provide simulation modellers with a collection of objects that can be connected like "black boxes" in order to facilitate both the input and output data process in a simulation model. This collection is desired to be expandable by giving means to developers for:

- customizing existing objects by overriding certain methods
- adding brand new objects to the list

The KE tool is product of a research project funded from the European Union Seventh Framework Programme (FP7-2012-NMP-ICT-FoF) under grant agreement n° 314364. The project name is DREAM and stands for *"Simulation based application Decision support in Real-time for Efficient Agile Manufacturing"*. More information about the scope of DREAM can be found at http://dream-simulation.eu/.

DREAM is a project which kicked off in October of 2012 and finishes in September of 2015. The KE tool is an ongoing project and we do not claim that it is complete or bug-free. The platform will be expanded and validated through the industrial pilot cases of DREAM. Nevertheless, it is in a quite mature state to attract the interest of simulation modellers and software developers.

The dream repository contains the following 5 folders:

- **platform**: contains code for a GUI (Graphical User Interface) that is being built for KE tool and ManPy, which stands for "Manufacturing in Python" and it is a layer of Discrete Event Simulation (DES) objects built in SimPy (http://simpy.sourceforge.net/). This is a parallel work and it is not always synchronized to KE tool's or ManPy's latest version
- **simulation**: contains all the simulation ManPy code along with some input files and some files from a commercial simulation package that are used for verification
- **KnowledgeExtraction**: contains all the KE tool code along with some input and output files from KE tool's examples
- **plugins**: Contains Python scripts that pre- or post-process simulation data model as it is exchanged between different DREAM modules. Documentation of the JSON schema used in order to exchange simulation and instance data will be added soon.
- **test**: contains unit-tests for the project.

This document regards ONLY the KE tool part of the project. Note that the KE tool is independent from both the GUI and ManPy and can be used separately as a library of Python objects, which can be used to input data in a simulation model or to conduct output analysis on simulation results. Users can implement alternative methods to be able to customize the objects for their own needs.

The reader of this documentation needs to have basic, yet not deep knowledge of programming in Python (http://www.python.org/) and RPy2. Also the reader is expected to have a basic knowledge of statistical analysis and Discrete Event Simulation (DES).

2 How to get started

The KE tool uses the R project (http://www.r-project.org/) and the following Python libraries:

- RPy2 (https://pypi.python.org/pypi/rpy2)
- xlrd (https://pypi.python.org/pypi/xlrd/0.9.3)
- xlwt (https://pypi.python.org/pypi/xlwt/0.7.5)
- json (https://pypi.python.org/pypi/python-json/3.4)
- xml.etree (https://pypi.python.org/pypi/elementtree/)
- csv (https://pypi.python.org/pypi/csv/1.0)
- pyodbc (https://pypi.python.org/pypi/pyodbc/3.0.7)

To use the platform you need to:

- Have Python installed
- Clone our git repository (https://github.com/nexedi/dream) or download it into a folder in your computer.
- Run the setup script that is in the root folder (*python setup.py install).* This will install the KE tool along with its dependencies (e.g. RPy2)
- Then you can import the KE tool objects as it is written in the examples, e.g.:
  from dream.KnowledgeExtraction.StatisticalMeasures import BasicStatisticalMeasures
  from dream.KnowledgeExtraction.ConfidenceIntervals import Intervals

3 Architecture

The KE tool objects are written exclusively in Python and they use methods of RPy2 and other Python libraries. Figure 1 shows the current state of the generic architecture.

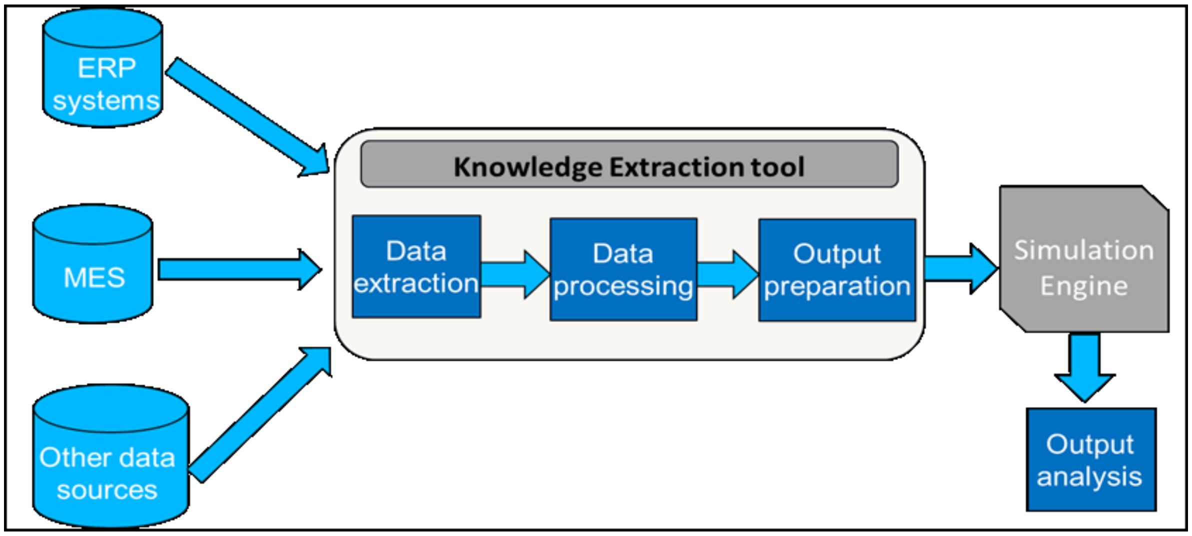


Figure 1: KE tool's architecture

In Figure 1 the four different components of the KE tool are depicted:

- Data extraction
- Data processing
- Output preparation
- Output analysis

Figure 1 illustrates at one hand the route of data to the Simulation Engine (either COTS (Commercial-off-the-self) simulation software or open-source simulation engines like ManPy) and at the other hand the analysis conducted to simulation results as an output.

The main role of the first component Data extraction relates to importing and extracting of data to the tool. After the initial extraction, some process may be needed to transform the data into useful forms. For instance, to obtain process time of a station in a production line, the stop time has to be subtracted from the start time. It is common for a system such as a manufacturing execution system (MES) to record entry and exit time-stamps. However, calculations would not be undertaken and therefore the task falls to the KE tool. Additionally, once the actual process time data points are obtained, the data should be analyzed using statistical methods in order to calculate descriptive statistical measures such as mean and spread, or fit a distribution. The above manipulations are mainly conducted by the second component of the tool named Data processing. The output of the Data processing component of the tool should is the data used by the simulation engine (which we will assume is ManPy for the remainder of this book). It needs to be provided in a readable format to ManPy, this is the role of the third component called Output preparation. At the moment, the KE tool can export data in three different data formats:

- CMSD (Core Manufacturing Simulation Data): A standard defined using Extensible Markup Language (XML) files that follow the CMSD specification,
- JSON (JavaScript Object Notation) API: A schema that is built according to the definition of the draft JSON schema (http://json-schema.org/),
- MS Excel files: Templates that are created in Excel.

The last component of the tool Output analysis comes after the simulation application is run. Output simulation analysis is the last modelling stage in a simulation study; it is concerned with the statistical analysis of the output data.

The tool is built in a way that the data input, the processing of this data and the output preparation are conducted by one 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. Therefore, this main script calls different objects so as to give as an output the actual selected data exchange file format, updated with the new available data. As it is stated in the Introduction, the main script consists of different tool’s objects connected like "black boxes" in order to facilitate both the input and output data process in a simulation model.

3.1 The KE tool objects

The set of these objects is the “heart” of the KE tool. These give the basic guidelines of how the tool is structured. Note that since this is an ongoing work, the names of the classes may change, since we currently think towards the best abstraction. Also new generic classes might be added in future versions, even though the number should be kept reasonably short. The objects include:

- **StatisticalMeasures:** calculates a variety of basic statistical measures in a given data sample.
- **DistributionFitting**: fits statistical distributions in a given data sample.
- **ReplaceMissingValues:** replaces missing values in a given data sample.
- **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.
- **CMSD_Output:** exports the outcomes of the statistical analysis in XML file that follow the CMSD standard specification.
- **JSON_Output:** exports the outcomes of the “Data processing” component into JSON (JavaScript Object Notation) file format.
- **ExcelOutput:** another export offered by the tool is in MS Excel files.
- **Plots:** represents the data points of a sample using graphs, plots and charts.
- **ConfidenceIntervals:** calculates the confidence intervals of a given data sample. It gives to user the ability to insert the probability that the confidence interval “covers” the true statistic.
- **Transformations:** calculates a variety of transformations in a given data sample.

- **DataManipulation:** a series of manipulations in the given data set is conducted applying this object.
- **DetectOutliers:** detects and deletes either just extreme outliers or both mild and extreme outliers in a given data sample.
- **Simul8XML:** the integration of the KE tool and Simul8 simulation software is conducted using the methods of this object.

We now expand on each of the objects describing details for their associated methods and the functionality provided by each.

3.1.1 ImportExceldata

The *ImportExceldata* object retrieves data from MS Excel files and imports this data into KE tool, adjusting this data in a way that can be later handled by the other objects of the tool. The extraction is achieved using the *xlrd* Python library (https://secure.simplistix.co.uk/svn/xlrd/trunk/xlrd/doc/xlrd.html?p=4966). Applying this object and particularly the *Input_data* method of this object the different data points are inserted in the tool in a form of a Python dictionary. The keys of this dictionary are the specified labels of these data points in the excel worksheet. This method as it is reasonable has as argument the name of the worksheet and the name of the input book. It is one of the "Data Extraction" objects of the KE tool.

3.1.2 ImportCSVdata

The *ImportCSVdata* object is another example of a Data Extraction object. As happens with the *ImportExceldata* object a Python library is used to develop this object, this library called csv (https://docs.python.org/2/library/csv.html#module-csv). The functionality of this object is similar to the *ImportExceldata* object.

3.1.3 ImportDatabase

One more object in the Data extraction component of the tool is the *ImportDatabase* object. Using the pyodbc (https://code.google.com/p/pyodbc/) Python library we developed a generic object that allows the users to connect with databases like MySQL, MS Access etc. A prerequisite to use this object is to download the ODBC driver for your Python and database platform, for example if want to access SQL Server connector for Python in MySQL database you should download the driver from http://dev.mysql.com/downloads/connector/python/.

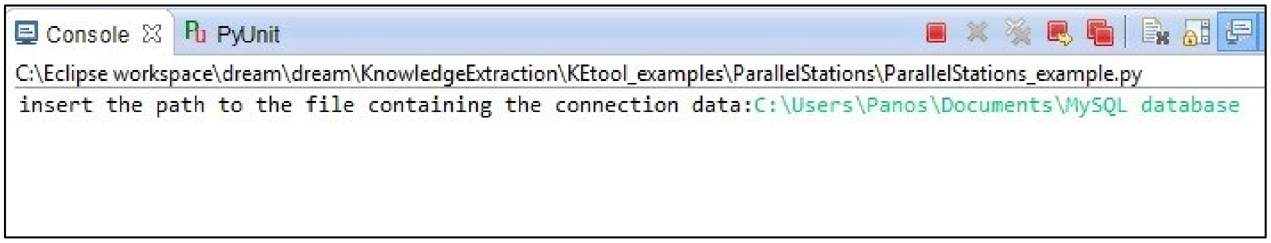
In order to apply this object the user has to create a .txt file with the connection information such as the name of the installed driver, the name of the server (which hosts the database), the port, the name of the database, the username and the password (see below).

```
driver = {MySQL ODBC 3.51 Driver}

server = localhost

port = 3306

data_base = name of the database

user = username

pass_word = xxxxx###
```

Having created the .txt file with the above info, the user has to save this file in a directory that he will be able to use it afterwards. Calling the object the user has to specify the name of the .txt file, the file extension of this file (for example .txt) and the number of cursors need to use (one cursor for each database query). See below the way that the user should call this object:

```
cnxn = ImportDatabase.ConnectionData(seekName='ServerData', implicitExt='txt',
number_of_cursors=3)
cursors = cnxn.getCursors()
```

When the user runs the model that contains the *ImportDatabase* object, the following message appears in his console "insert the path to the file containing the connection data:". The user has to write the connection data file directory. For example in the figure below as you can see we put the file directory as required.



```
C:\Eclipse workspace\dream\dream\KnowledgeExtraction\KEtool_examples\ParallelStations\ParallelStations_example.py
insert the path to the file containing the connection data:C:\Users\Panos\Documents\MySQL database
```

3.1.4 StatisticalMeasures

This is one of the most important objects of the Data processing component of the tool. Applying the method *BasicStatisticalMeasures* of this object, one is able to calculate different useful statistical measures in a data set. An example of this object, with the calculation of the length of a data set is presented below:

```
#The BasicStatisticalMeasures object
class BasicStatisticalMeasures:
# A variety of statistical measures are calculated in this object
    def length(self, data):            #Calculate the length of data sample
        data=robjects.FloatVector(data) ##The given list changes into float vector in
                                                          order to be handled by RPy2
        rlength = robjects.r['Length']  #Call length function-R function
        return rlength(data)[0]
```

Other available statistical measures like mean value, variance, standard deviation, length, summary, quantiles, frequency of each data point, range, interquartile range, min, max, mad and median can be calculated so as to extract useful information from a data sample.

The modeler can use these measures to describe a data set and provide with the results meaningful information to the simulation software. The above measures are named descriptive statistics and provide simple summaries about the sample and about the observations that have been made. The analysis that is conducted to calculate these measures is called Univariate analysis. Univariate analysis involves describing the distribution of a single variable, including its central tendency (including the mean, median, and mode) and dispersion (including the range and quantiles of the data set, and measures of spread such as the variance and standard deviation).

3.1.5 DistributionFitting

DistributionFitting is the most useful and applied object in the Data processing component of the tool. By calling this object, one is able to choose between two options, which are essentially the two classes of this object and are both ways of quantitatively selecting a statistical distribution and parameters that best represent a data set. Using the first class Distributions, the modeller is able to conduct distribution fitting using the Maximum Likelihood Estimation (MLE) statistical method (see distribution fitting example 1 below). Applying the second class DistFittest, the modeller does the same but using the Kolmogorov-Smirnov statistical goodness-of-fit test (this test calculates the maximum distance between the empirical and the fitted Cumulative Distribution Functions (CDF), which is applied for automatically selecting the best-fitting distribution (see distribution fitting example 2 below).

Example of the Distributions class (Normal Distribution)

```
#The Distributions class

class Distributions:

    def Normal_distrfit(self,data):
        data=robjects.FloatVector(data)     #The given data sample changes into float
vector in order to be handled by RPy2
        rFitDistr=robjects.r['fitdistr']    #Call FitDistr function - R function
        try:                                       #try..except syntax to test if
the data sample fits to Normal distribution
            self.Normal= rFitDistr(data,'Normal')   #It fits the normal distribution
to the given data sample
        except RRuntimeError:
            return None                            #If it doesn't fit Return None
        myDict = {'distributionType':'Normal','mean':self.Normal[0][0],'stdev':
self.Normal[0][1],'min':0, 'max':(self.Normal[0][0]+3*self.Normal[0][1])}
#Create a dictionary with keys distribution's and distribution's parameters  names
and the parameters' values
        return myDict                     #If there is no Error return the
dictionary with the Normal distribution parameters for the given data sample
```

Example of *DistFittest* class (Exponential distribution)

```
#The Distribution Fitting test class
class DistFittest:

    def Exp_kstest(self,data):
        data=robjects.FloatVector(data)        #The given data sample changes into
float vector in order to be handled by RPy2
        rkstest= robjects.r['ks.test']         #Call ks.test function - R function
        rFitDistr=robjects.r['fitdistr']       #Call FitDistr function - R function
       try:                                       #try..except syntax to test if the
                                         data sample fits to Exponential distribution
```

```
            self.Normal= rFitDistr(data,'Exponential')       #It fits the exponential
                                                          distribution to the given data sample
        except RRuntimeError:
            return None                                 #If it doesn't fit Return None
        exp=self.Normal
       self.Exptest= rkstest(data,"pexp",exp[0][0])       #It conducts the Kolmogorov-
                  Smirnov test for Exponential distribution to the given data sample
        return self.Exptest                            #If there is no error returns the
outcome of the Kolmogorov-Smirnov test (p-value,D)
```

At the time of writing this documentation the tool using the functions from RPy2 library can identify and fit data using the following distribution functions:

- Normal
- Exponential
- Poisson
- Gamma
- Logistic
- Geometric
- Cauchy
- Log-Normal
- Negative Binomial
- Weibull
- Fixed
- Beta

3.1.6 ReplaceMissingValues

Another useful functionality mostly for pre-processing is provided using the *ReplaceMissingValue* object of the tool. Applying the class *HandleMissingValues* the modeler is able to select the functionality (Python method) that suits him best. It is worth mentioning that this object inherits methods from the *StatisticalMeasures* object. Data preparation and pre-processing is a crucial step before the start of the statistical analysis. The main activity of the data pre-processing step is to handle the missing values in a data set. Using this object the modeler is able to replace the missing data with:

- zero,
- mean value
- median of the non-missing values,
- totally erase the missing data.

3.1.7 DetectOutliers

The *DetectOutliers* object is an additional feature provided by the tool for pre-processing analysis. Applying the class *HandleOutliers* the user is able to select one of the two provided approaches – Python methods. The first one deletes both the mild and extreme outliers while the second approach deletes only the extreme outliers in a given data set. Like in *ReplaceMissingValues* object, *DetectOutliers* inherits methods from the *StatisticalMeasures* object. The detection of outliers is done calculating the inner and outer fence of the data set, a point beyond an inner fence on either side (Lower Inner Fence – LIF and Upper Inner fence – UIF) is considered a mild outlier. A point beyond an outer fence (Lower Outer Fence – LOF and Upper Outer Fence – UOF) is considered an extreme outlier. These borders are calculated using the lower and upper quartiles (defined as the 25th and 75th percentiles). If the lower quartile is Q1 and the upper quartile is Q3, then the difference (Q3 - Q1) is called the interquartile range or IQ, then:

- Lower Inner Fence - LIF: Q1 - 1.5*IQ
- Upper Inner Fence - UIF: Q3 + 1.5*IQ
- Lower Outer Fence - LOF: Q1 - 3*IQ
- Upper Outer Fence - UOF: Q3 + 3*IQ

3.1.8 Transformations

The *Transformations* object is predominantly used in relation to the Data Processing component. The modeller applying the methods of this object is able to calculate a variety of useful data transformations. The provided functionality includes the calculation of the sum, the cumulative sum, the cumulative product, the cumulative min value and the cumulative max value. Also, applying the scale method the modeller can centre the data points of a data set around the mean value and scales by the standard deviation (SD). Finally, using the reverse method the modeller can reverse the order of values in the data sample.

3.1.9 DataManipulation

The *DataManipulation* object of the Data Processing component offers the modeller further calculated measures by applying the methods of the object. These include rounded values of a data sample (round method), the smallest integers bigger than the values of the data sample (ceiling method), the largest integers smaller than the values of the data sample (floor method), a list with the absolute values of the data sample (abs method) and the square root of the values of the data sample (sqrt method).

3.1.10 ExcelOutput

The KE tool can export the results from the conducted analysis in the second component of the tool Data processing in MS Excel files. The outcomes of the *StatisticalMeasures* and *DistributionFitting* objects are exported in standard templates in Excel files. Applying the Output Python class and especially either the *PrintStatisticlamesasures* method (for the results of the *SatatisticlMeasures* obect) or the *PrintDistributionFit* (for the results from the *DistributionFitting* object) the model can examine the results of these objects opening an Excel file.

3.1.11 CMSD_Output

CMSD_Output is not yet a separate object, but we aspire to develop a generic object to handle all situations in a macro level. Despite that, the tool is able to export the already processed data in XML files that follow the CMSD specification. In the example section the reader can see how we manage to output the processed data to CMSD information model using the xml.etree Pyhton library (https://docs.python.org/2/library/xml.etree.elementtree.html#).

3.1.12 JSON_Output

Similar to CMSD_Output the JSON_Output is not yet a separate object but using the json Python library (https://docs.python.org/2/library/json.html) the tool is able to export the processed data from the Data processing component of the tool to JSON files. These files follow a specification that has been defined by DREAM project for the sake of communication among the different modules KE tool, GUI, ManPy. Using this specification we have achieved the integration of the different modules of the DREAM platform.

3.1.13 Simul8XML

By using this object the KE tool integrates with the Simul8 simulation package. The integration is achieved accessing and modifying the Simul8XML file. The user is able to access the Simul8XML file of a model and modify several simulation information based on the available data. This modification is conducted applying the different features (methods) of this object. Meaningful simulation information like the processing times, the inter-arrival times, the results collection period, the warm-up time, the capacity of a queue, the MTBF (Mean Time Between Failure) and the MTTR (Mean Time To Repair) can be adjusted using this object. The user is able to initialize and run a simulation model in Simul8 simulation software using this KE tool object.

3.1.14 Plots

This object is mainly referring to Output analysis, which is the last component of the KE tool. But the modeler can also apply it during the Data processing component when he wants to present the data points in a data set in a graphical representation. So using this object the modeler can represent data using graphs, plots and charts. So far, we are able to display data in plots, scatterplots, histograms, barplots and pie charts. Each one of the above is separate method of the *Graphs*, which is the class of this object. It is also provided the functionality of the parallel representation of two data sets in a plot.

3.1.15 ConfidenceIntervals

ConfidenceIntervals is another object related to the Output Analysis component. Confidence interval estimation quantifies the confidence (probability) Confidence interval estimation quantifies the confidence (probability) that an interval “covers” the true (but unknown) statistic. The boundaries of the confidence interval are estimated using appropriate point estimates therefore, those boundaries are random variables, and the confidence interval is a random interval which varies across experiments (replications). The modeler predetermines the desired probability that the confidence interval “covers” the true statistic, e.g. a 99% CI, (the larger the probability, the wider the interval). The data sets which the modeler wants to estimate the CI along with the desired probability for are taken as the arguments by the *ConfidIntervals* method.

4 Data interfaces

4.1 CMSD

Researchers at the National Institute of Standards and Technology (NIST) in collaboration with industrial partners have developed the Core Manufacturing Simulation Data (CMSD) Product Development Group (PDG) under the guidelines and procedures of the Simulation Interoperability Standards Organization (SISO) (SISO 2010). The idea of the CMSD effort is to facilitate data exchange and sharing by using neutral, reusable data formats for managing actual operations and for simulating the performance of the manufacturing systems.

According to “Standard for: Core Manufacturing Simulation Data – UML Model” (SISO 2010) 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 and
- enabling greater manufacturing software application interoperability.

The primary objective of the CMSD is to facilitate interoperability between simulation systems and other manufacturing applications. Towards this objective it provides a data specification that enables the efficient exchange of manufacturing life-cycle data in a simulation environmen.The objective is to:

- foster the development and use of simulations in manufacturing operations,
- facilitate data exchange between simulation and other manufacturing software applications,
- enable and facilitate better testing and evaluation of manufacturing software and
- increase manufacturing application interoperability.

4.1.1 CMSD information model

The CMSD information model is a standard representation for core manufacturing simulation data. It describes the essential entities in the manufacturing domain and the relationship between those entities that are necessary to create manufacturing simulations. Although the information defined in this model may be associated with one or more different manufacturing domains such as process planning, scheduling, inventory management, production management, or supply chain management, the model is not intended to be an all-inclusive definition of the entire manufacturing or simulation domain.

One thing that must be clear to us is that there is no implied or explicit provision in the model for the direct specification of the execution behavior of a manufacturing entity in a simulation or other manufacturing application. This means that while a process can be defined that specifies a particular part resource in a particular state may be processed on a particular machine resource for a certain amount of time, the method that a simulation should use to implement this process is not specified.

No support is provided for the creation of programming language or simulation language executable constructs, or the association of such constructs with the manufacturing-related entities that can be defined in the CMSD Information Model.

The CMSD information model is presented in two different documents: (1) the information model defined using the Unified Modeling Language (UML) and (2) the information model defined using the eXtensible Modeling Language (XML) (SISO 2012). The exchange of data between simulations will be enabled through the exchange of XML instance documents that adhere to the CMSD XML Schemas. The UML diagram and XML schemas are intended to be identical representations of the same CMSD conceptual model.

The CMSD Information Model consists of the following major UML packages group related data:

- Layout
- Support package
- Resource Information package
- Production Planning package
- Production Operations package
- Part Information package

The major data categories of manufacturing information that are defined in this information model are organizational structure, calendar, resource, skill definition, setup definition, operation definition, maintenance definition, part, bill-of-materials, inventory, process plan, work, schedule, revision, distribution definition, reference, and unit defaults.

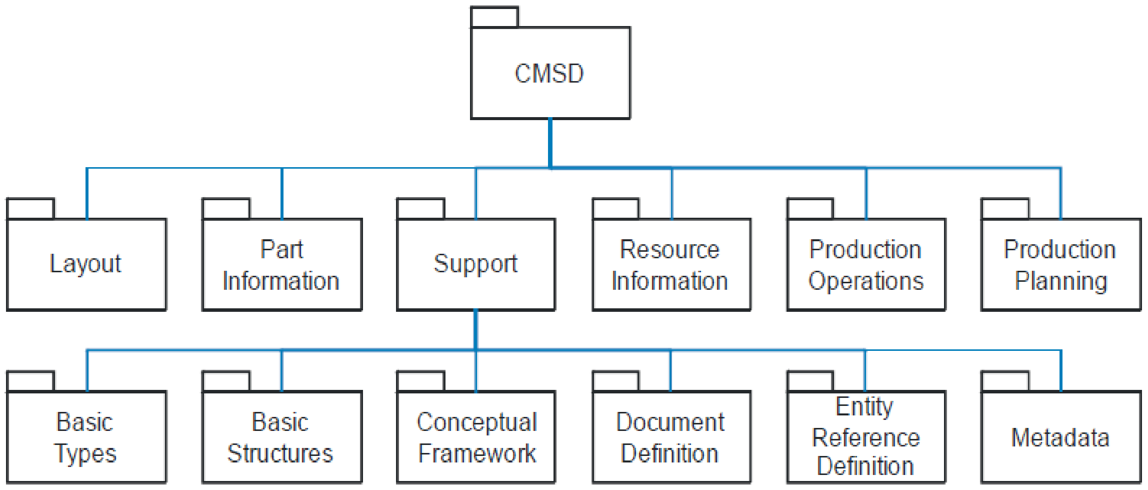


Figure 2: The Packages of the CMSD model (SISO 2010)

The CMSD model is designed as a suite of interrelated collections of information modeled as UML classes contained within UML packages, presented visually as a series of UML class and package diagrams. The primary function of the UML packages in the CMSD model is to partition and group, by major areas of manufacturing, the class definitions that realize related manufacturing concepts. The manufacturing concepts within each package are modeled as

UML classes and the characteristics associated with each entity are modeled as UML class attributes. Operation specifications are not used in defining classes and the visibility of all class attributes is public (SISO 2010).

4.2 JSON API

The schema is built according to the definition of the draft JSON schema (http://json-schema.org/). The scope of the schema defined is to provide a meaningful data structure description to the SE. The schema has the general form depicted in Figure 3.

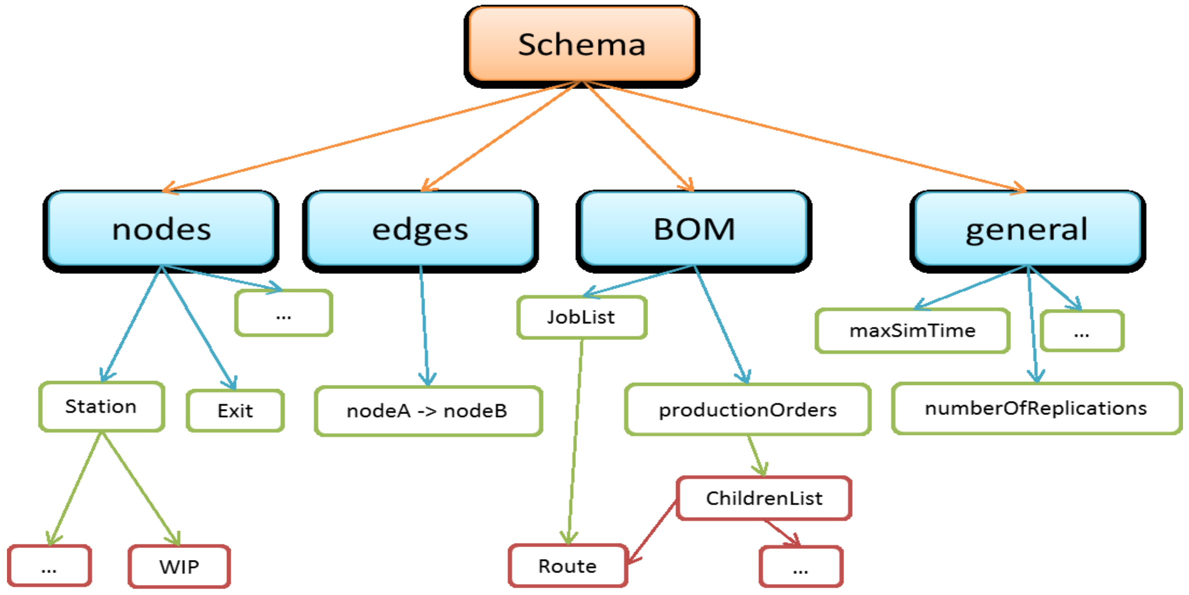


Figure 3: Structure of the JSON API

The schema developed is used to define the frame within which the transaction between an external to the SE module and the core of the SE. The JSON file sent from the external module containing the simulation model description consists of 4 echelons. “General” refers to attributes that a simulation experiment needs for each run. This may refer to the number of replications, the length of each replication etc.

The JSON file should be in position to describe the topology of the DES model in terms of directed graph nomenclature. Thus, in each data exchange file there is a collection of nodes and edges. Each node represents an object that is permanent for a DES model (e.g. a buffer, a server an assembly point etc.). These are described as JSON objects and keep all their attributes as encapsulated JSON objects. Every edge describes the connection between two nodes one of which is the starting and the other is the ending point of the connection.

Moving Entities (MEs) that move within the described system may have specific interconnections between them, for example comprise a specific order (components that have the same parent order). A “Bill Of Materials” is used to describe these data structures that cannot be uniquely identified within the system as physical presences.

4.2.1 Data definition specification

The simulation input file should contain all the needed information to initiate a simulation. Following the logic of the simulation core development, the general form of the JSON schema is given below:

```
"general":{
        "type": "object",
        "properties":{
                "_class":{"type":"string"},
                "confidenceLevel": {"type": ["number", "string"]},
                "maxSimTime": {"type": ["number", "string"]},
                "numberOfReplications": {"type": ["number", "string"]},…
        }
},
"nodes":{
        "type":"object",
        "patternProperties":{
                "[A-Za-z0-9]":{"$ref": "#/definitions/_node"}
        }
},
"edges":{
        "type":"object"
        "patternProperties":{
                "[0-9A-Za-z]":{"$ref": "#/definitions/_edge"}
        }
},
"BOM":{
        "type":"object",
        "properties":{
                "productionOrders":{
                        "type":"array",
                        "uniqueItems": true,
                        "items":{
                                "$ref":"#/definitions/_order"
                        }
                },
                "jobList":{"type":"array","items":{"$ref":"#/definitions/_entity"}},
        }
}
```
One can notice that as in Figure 1, there are four different echelons, namely general, nodes, edges, and BOM. General echelon is straightforward to understand, while the other general attributes contain references which must be explained.

The edges and the nodes are collections of separate nodes and edges. Each one of them has a unique key and the corresponding value is described as a JSON object. Focusing on the schema for such a node, we have:

```
"_node": {
        "type":"object",
        "properties":{
                "_class":{ "type":"string","required": true },
                "name": {"type":"string"},
                "id": {"type":"string"},
                "element_id":{"type":"string"},
                "capacity":{"type":"number"},
                "wip":{"$ref":"#/definitions/_wip"},
                "schedulingRule":{"type":"string"},
                "failures":{
                        "type":"object",
                        "properties":{
                                "failureDistribution":{"type":"string"},
                                        "availability":{"type":"string"},
                                        "MTTF":{"type":"number"},
                                        "MTTR":{"type":"number"},
                                        "repairman":{"type":"string"}
                                }
                        },
                        "processingTime":{"$ref":"#/definitions/_procTime"},
                        "entity":{"type":"string"},
                        "interarrivalTime":{
                                "type":"object",
                                "properties":{
                                        "distributionType":{"type":"string"},
                                        "mean":{"type":"number"},
                                        "max":{"type":"number"},
                                        "min":{"type":"number"},
                                        "stdev":{"type":"number"}
                                }
                        }
                }
}.
```

This (simplified for the presentation purposes) schema contains all the definitions needed to describe whichever station is currently developed within the SE. Nodes are described as JSON objects and keep all their attributes as encapsulated JSON objects. For example for the initiation of a source, one would need the “entity” attribute describing the type of the ME that the source creates, and an interarrivalTime describing how often the MEs are produced. A machine may have a *processingTime* attribute that is also a JSON object that describes the way the processing time of the machine is described. One example would be *"processingTime": {"distributionType": "Normal", "mean": 1.5, "stdev":0.3, "min":0.1, "max":10}*. Different distributions may require different attributes (e.g. standard deviation) and more complex distributions, such as empirical ones, may be defined. The nodes may hold a wip attribute that defines the Work In Progress (WIP) that the object keeps at the start of the simulation.

Every node should necessarily have two attributes: first, an id that would be unique for this specific node and it is the key of the whole JSON object that describe the node. Second, a class attribute, that corresponds to the class of the simulation object that the node describes. For example the definition of a Queue simulation object from the DREAM repository with capacity of one can be as simple as: *"Q1": {"_class": "Dream.Queue", "capacity": 1}*.

On the other hand, for the edges we have a shorter schema:

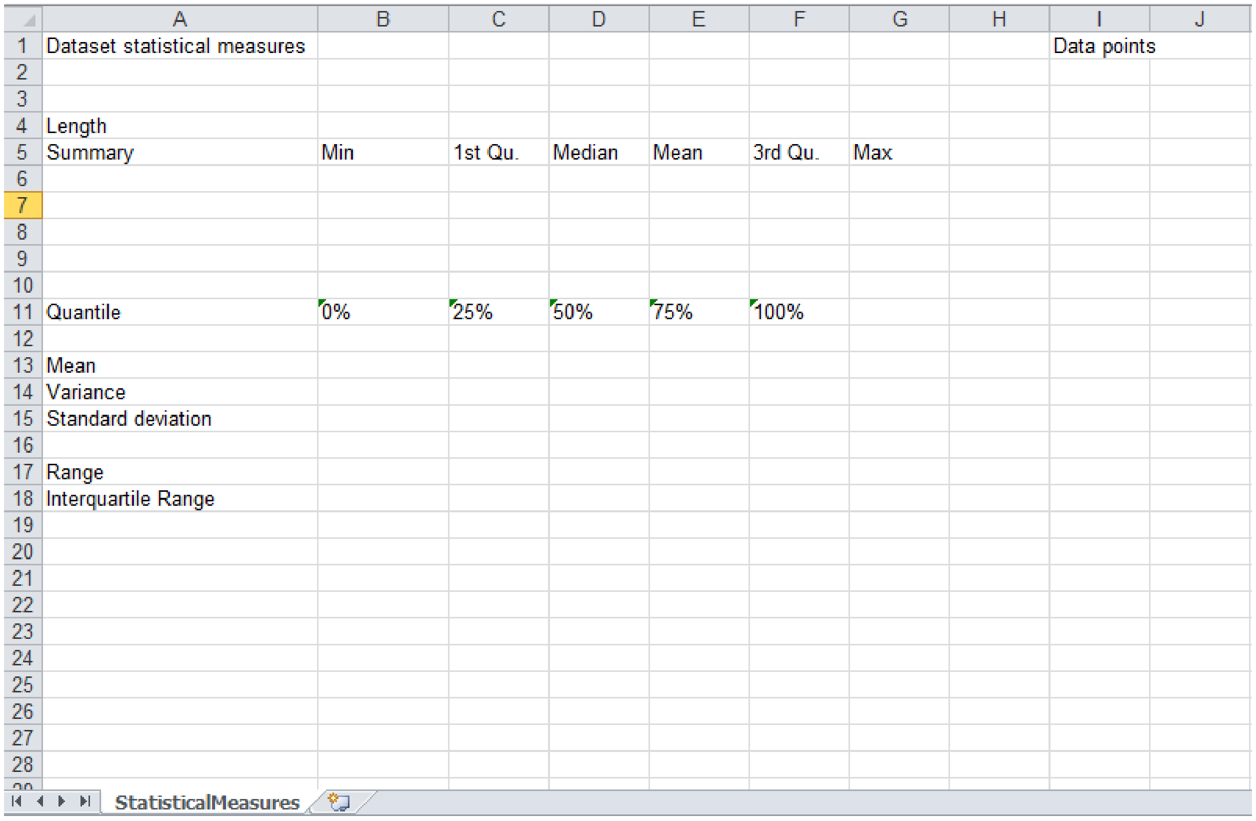
```
"_edge": {
        "type":"array",
        "items":[
                {"type": "string"},
                {"type": "string"},
                {"type": "object"}
        ],
        "minItems":2,
        "maxItems":3,
        "uniqueItems":true
}.
```

Two properties are of importance for an edge, the start and the end point of the connection between two servers or between a server and resource. This information is provided as an array and the direction of the information flow along the edge is given by the indices of the station in the array. Every edge describes the connection between two nodes, having a generic shape of *"edge_id": {"source_node_id", "destination_node_id", {edge properties}}*. So if we have a Queue with id “Q1” that is the place where parts are buffered before getting processed by Machine with id “M1”, then the node will be defined as *"0": ["Q1","M1", {}]*, where “0” is the id of the edge. Of course, “Q1” and “M1” should correspond to nodes of the same model configuration. The empty JSON object in the end of the list is reserved for properties of the edge. More complex routing complexities (e.g. a buffer may forward a job to different stations depending on its type) can be described within the nodes and the simulation class each corresponds to.

As mentioned previously, every node may have a wip attribute to define the starting work in progress for the start of simulation. Nevertheless, jobs may have specific interconnections between them, for example constitute a specific order. For this reason, we added a “Bill Of Materials” (BOM) echelon. The use of the keyword `"$ref":"#/definitions/_entity"}` pointing to a definition of ME object schema _entity. The ME should also have id and name keyword as well as a class attribute describing the corresponding logic of the simulator core. It may also contain enclosed schemas for routing description if the ME has to follow a specific route through the topology of the model.

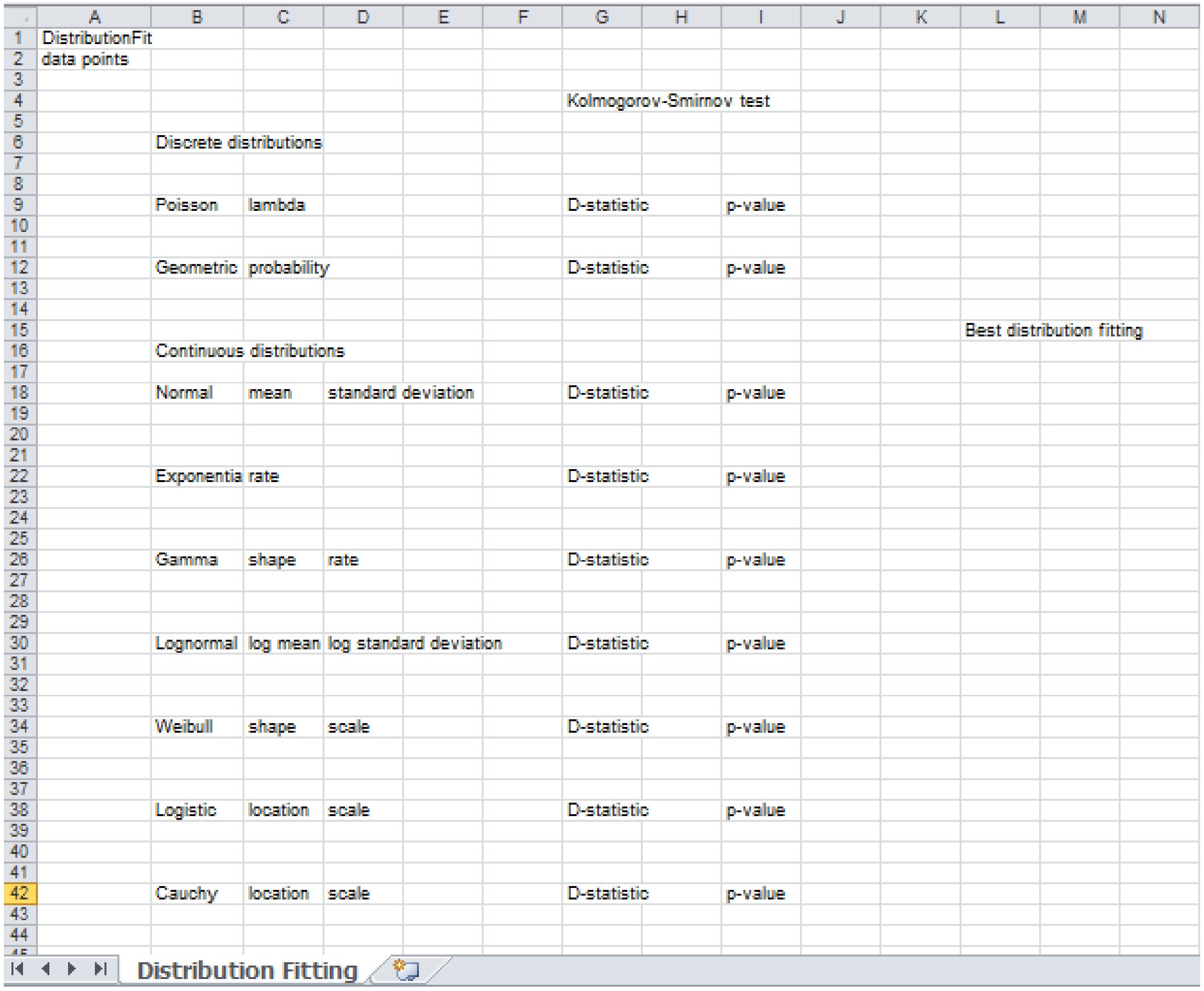
4.3 MS Excel

Another format that KE tool can export data is MS Excel. Using the xlwt Python library (https://pypi.python.org/pypi/xlwt), we are able to export the outcome of the analysis in Data processing component (Statistical measures, Distribution fitting) on standard templates in Excel worksheets. The standard templates showing the results from the analysis on dataset’s statistical measures and distribution fitting are illustrated below.



	A	B	C	D	E	F	G	H	I	J
1	Dataset statistical measures								Data points	
2										
3										
4	Length									
5	Summary	Min	1st Qu.	Median	Mean	3rd Qu.	Max			
6										
7										
8										
9										
10										
11	Quantile	0%	25%	50%	75%	100%				
12										
13	Mean									
14	Variance									
15	Standard deviation									
16										
17	Range									
18	Interquartile Range									
19										
20										
21										
22										
23										
24										
25										
26										
27										
28										

Figure 4: Template showing the exported results on different statistical measures



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	DistributionFit													
2	data points													
3														
4							Kolmogorov-Smirnov test							
5														
6		Discrete distributions												
7														
8														
9		Poisson	lambda				D-statistic		p-value					
10														
11														
12		Geometric	probability				D-statistic		p-value					
13														
14														
15												Best distribution fitting		
16		Continuous distributions												
17														
18		Normal	mean	standard deviation			D-statistic		p-value					
19														
20														
21														
22		Exponentia	rate				D-statistic		p-value					
23														
24														
25														
26		Gamma	shape	rate			D-statistic		p-value					
27														
28														
29														
30		Lognormal	log mean	log standard deviation			D-statistic		p-value					
31														
32														
33														
34		Weibull	shape	scale			D-statistic		p-value					
35														
36														
37														
38		Logistic	location	scale			D-statistic		p-value					
39														
40														
41														
42		Cauchy	location	scale			D-statistic		p-value					
43														
44														

Figure 5: Template showing the exported results on distribution fitting analysis

5 Examples

We now introduce examples to support hands-on familiarity with the KE tool. The scripts are provided along with descriptions and graphical representations of the models within which the KE tool is applied. The following examples can be found in GitHub at the following link (https://github.com/nexedi/dream/tree/master/dream/KnowledgeExtraction/KEtool_examples).

5.1 Two servers model with failures and repairman

A simple model topology is illustrated in Figure 6. There are two machines and a queue between them. The machines are vulnerable to failures and when a failure occurs they need a repairman in order to be fixed. In this model there is only one repairman available. In this model there is only one repairman available. The repairman floats between them as needed. The model itself was created in the DREAM GUI.

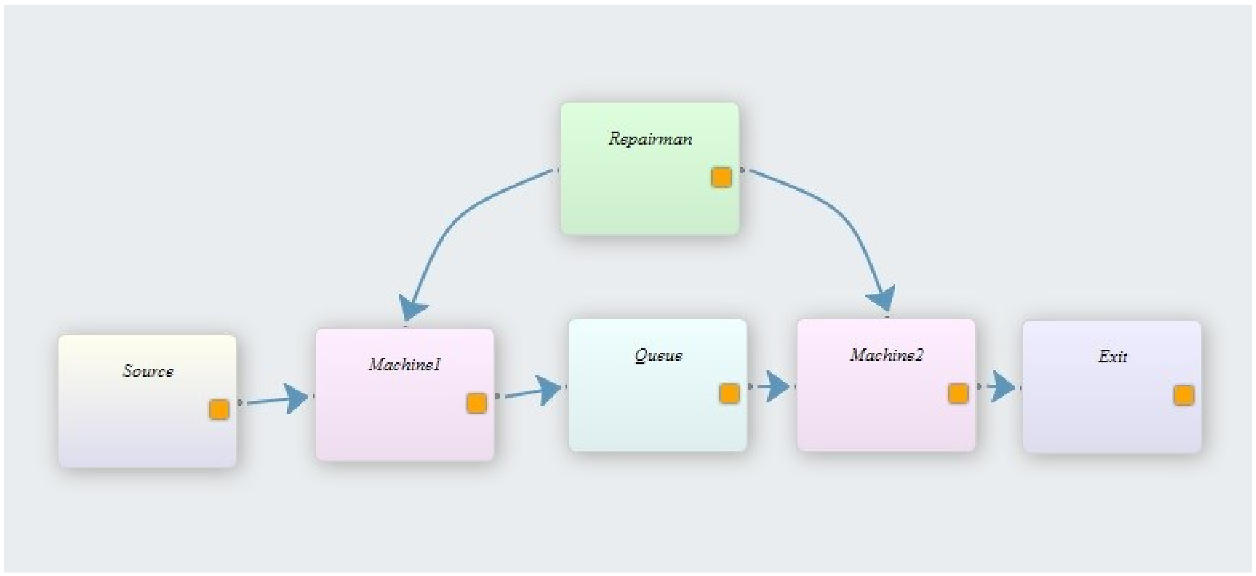


Figure 6: Two servers model with failures and repairman

For the needs of the example we assume that the processing times of the two machines are recorded in a simple Excel file. Having a sample of this data, we use the *ImportExcelData* object to retrieve these data samples with historical processing times of the two machines and import them to the tool. Then, we make use of the *ReplaceMissingValues* object and we delete the missing values in the data samples in order to continue our process without missing data in the samples. The *DistFittest* object of the DistributionFitting script is used to perform a Kolmogorov-Smirnov test to identify the best fitting statistical distributions for the two data samples. The above two objects (*ReplaceMissingValues, DistFittest*) complete the operations of the second component, Data processing for this example. Finally, four documents are exported with the updated values coming from the data process, to demonstrate the three available ways to output data from the tool. The four documents are the CMSD information model of the topology, the JSON file based again on the example's topology and two Excel files with the outcomes of the statistical analysis (calculation of basic statistical measures and distribution fitting results for one data sample). The outputs of these four ManPy readable documents fulfil the operations of tool's third component, Output preparation.

GitHub contains the main script related to this example. In addition the following are also available:

- The Excel files with the inputs to the tool (inputsTwoServers.xls).
- The CMSD information model of the topology (CMSD_TwoServers.xml).
- The JSON file of the topology (JSON_TwoServers.json).

Below is the KE tool main script.

```
from dream.KnowledgeExtraction.ImportExceldata import Import_Excel
from dream.KnowledgeExtraction.DistributionFitting import DistFittest
from dream.KnowledgeExtraction.ExcelOutput import Output
from dream.KnowledgeExtraction.ReplaceMissingValues import HandleMissingValues
from xml.etree import ElementTree as et
import xlrd
import json

#======= This script is a simple example of the Knowledge extraction tool ==========#

#The following is the Main script, that calls two Python objects in order to conduct
the three main components of the Knowledge extraction tool
#In the following example the operation times of the topology's two machines are
given in an Excel document.
#Import_Excel object imports data from the Excel document to the tool and DistFittest
object fits the data to a statistical distribution using Kolmogorov-Smirnov test

workbook = xlrd.open_workbook('inputsKEtool.xls')      #Using xlrd library opens the
Excel document with the input data
worksheets = workbook.sheet_names()
worksheet_OperationTime = worksheets[0]            #It creates a variable that holds
the first Excel worksheet

X=Import_Excel()                                   #Call the import_Excel object
OperationTimes= X.Input_data(worksheet_OperationTime,workbook)      #It defines a
Python dictionary, giving as name OpearationTimes and as value the returned
dictionary from the import_Excel object
Machine1_OpearationTimes = OperationTimes.get('Machine1',[])        #Two lists are
defined (Machine1_OpearationTimes, Machine2_OpearationTimes) with the operation times
data of each machine
Machine2_OpearationTimes = OperationTimes.get('Machine2',[])

A=HandleMissingValues()                                            #Call the
HandleMissingValues object
Machine1_OpearationTimes= A.DeleteMissingValue(Machine1_OpearationTimes)        #It
deletes the missing values in the lists with the operation times data
Machine2_OpearationTimes= A.DeleteMissingValue(Machine2_OpearationTimes)

Dict={}
B=DistFittest()                                    #It calls the DistFittest object
Dict['M1']=B.ks_test(Machine1_OpearationTimes)               #It conducts the
Kolmogorov-Smirnov test in the list with the operation times data
Dict['M2']=B.ks_test(Machine2_OpearationTimes)
M1=Dict.get('M1')
M2=Dict.get('M2')


#====== Output preparation: output the updated values in the CMSD information model
of Topology10 ========#

datafile=('CMSD_Topology10.xml')       #It defines the name or the directory of the
XML file that is manually written the CMSD information model
tree = et.parse(datafile)                                          #This file
will be parsed using the XML.ETREE Python library

M1Parameters=[]
M1ParameterValue=[]
for index in list(Dict['M1'].keys()):
    if index is not 'distributionType':
        M1Parameters.append(index)
        M1ParameterValue.append(Dict['M1'][index])
```

```
if Dict['M1']['distributionType']=='Normal':
    del M1['min']
    del M1['max']
elif Dict['M2']['distributionType']=='Normal':
    del M2['min']
    del M2['max']

M2Parameters=[]
M2ParameterValue=[]
for index in list(Dict['M2'].keys()):
    if index is not 'distributionType':
        M2Parameters.append(index)
        M2ParameterValue.append(Dict['M2'][index])

root=tree.getroot()
process=tree.findall('./DataSection/ProcessPlan/Process')                #It creates a
new variable and using the 'findall' order in XML.ETREE library, this new variable
holds all the processes defined in the XML file
for process in process:
    process_identifier=process.find('Identifier').text                  #It creates a
new variable that holds the text of the Identifier element in the XML file
    if process_identifier=='A020':                                      #It checks
using if...elif syntax if the process identifier is 'A020', so the process that uses
the first machine
        OperationTime=process.get('OpeationTime')                       #It gets the
element attribute OpearationTime inside the Process node
        Distribution=process.get('./OperationTime/Distribution')        #It gets the
element attribute Distribution inside the OpearationTime node
        Name=process.find('./OperationTime/Distribution/Name')          #It finds the
subelement Name inside the Distribution attribute
        Name.text=Dict['M1']['distributionType']
#It changes the text between the Name element tags, putting the name of the
distribution (e.g. in Normal distribution that will be Normal)

DistributionParameterA=process.get('./OperationTime/Distribution/DistributionParamete
rA')
        Name=process.find('./OperationTime/Distribution/DistributionParameterA/Name')
        Name.text=str(M1Parameters[0])                                 #It changes the
text between the Name element tags, putting the name of the distribution's first
parameter (e.g. in Normal that will be the mean)

Value=process.find('./OperationTime/Distribution/DistributionParameterA/Value')
        Value.text=str(M1ParameterValue[0])                            #It changes the
text between the Value element tags, putting the value of the distribution's first
parameter (e.g. in Normal so for mean value that will be 5.0)

DistributionParameterB=process.get('./OperationTime/Distribution/DistributionParamete
rB')
        Name=process.find('./OperationTime/Distribution/DistributionParameterB/Name')
        Name.text=str(M1Parameters[1])                                  #It changes the
text between the Name element tags, putting the name of the distribution's second
parameter (e.g. in Normal that will be the standarddeviation)

Value=process.find('./OperationTime/Distribution/DistributionParameterB/Value')
```

```
        Value.text=str(M1ParameterValue[1])                        #It changes the
text between the Value element tags, putting the value of the distribution's second
parameter (e.g. in Normal so for standarddeviation value that will be 1.3)
    elif process_identifier=='A040':                               #It checks using
if...elif syntax if the process identifier is 'A040', so the process that uses the
second machine
        OperationTime=process.get('OpeationTime')
        Distribution=process.get('./OperationTime/Distribution')
        Name=process.find('./OperationTime/Distribution/Name')
        Name.text=Dict['M2']['distributionType']

DistributionParameterA=process.get('./OperationTime/Distribution/DistributionParamete
rA')
        Name=process.find('./OperationTime/Distribution/DistributionParameterA/Name')
        Name.text=str(M2Parameters[0])

Value=process.find('./OperationTime/Distribution/DistributionParameterA/Value')
        Value.text=str(M2ParameterValue[0])

DistributionParameterB=process.get('./OperationTime/Distribution/DistributionParamete
rB')
        Name=process.find('./OperationTime/Distribution/DistributionParameterB/Name')
        Name.text=str(M2Parameters[1])

Value=process.find('./OperationTime/Distribution/DistributionParameterB/Value')
        Value.text=str(M2ParameterValue[1])
    else:
        continue
    tree.write('CMSD_Topology10_Output.xml',encoding="utf8")
#It writes the element tree to a specified file, using the 'utf8' output encoding

    #========== Output preparation: output the updated values in the JSON file of
                              Topology10 ============#
jsonFile= open('JSON_Topology10.json','r')      #It opens the Topology10 JSON file
data = json.load(jsonFile)
#It loads the file
jsonFile.close()
nodes=data.get('coreObject',[])
#It creates a variable that holds the 'coreObject' list

for element in nodes:
    name=element.get('name')
#It creates a variable that gets the element attribute 'name'
    processingTime=element.get('processingTime',{})
#It creates a variable that gets the element attribute 'processingTime'

    if name =='Machine1':
        element['processingTime']=Dict['M1']                           #It checks
using if...elif syntax if the name is 'Machine1', so the first machine in the
Topology10
    elif name=='Machine2':
        element['processingTime']=Dict['M2']
    else:
        continue
    jsonFile = open('JSON_Topology10_Output.json',"w")     #It opens the JSON file
    jsonFile.write(json.dumps(data, indent=True))
#It writes the updated data to the JSON file
    jsonFile.close()
#It closes the file

 #========== Calling the ExcelOutput object, outputs the outcomes of the statistical
                          analysis in Excel files =======#
C=Output()
C.PrintDistributionFit(Machine1_OpearationTimes,'Machine1_DistFitResults.xls')
C.PrintStatisticalMeasures(Machine1_OpearationTimes,'Machine1_StatResults.xls')
C.PrintDistributionFit(Machine2_OpearationTimes,'Machine2_DistFitResults.xls')
C.PrintStatisticalMeasures(Machine2_OpearationTimes,'Machine2_StatResults.xls')
```

The above main script consists of four KE tool objects and it has also parts for CMSD_Output and JSON_Output. The CMSD, JSON and Excel output files can easily obtained by downloading and running the example. In Appendices the reader can see the above output files.

5.2 Production line

Another example of the KE tool is built developing a main script using the objects in a real production line. Figure 7 illustrates the graphical representation of the production line modeled in the DREAM platform GUI. In this model we have several Machines (P1 - P11) operating in parallel, Queues between them and other simulation objects between them. The Machines are vulnerable to failures so scrap parts produced. We've got information about the processing times and the scrap quantity in each of the machines in the production line.

In this example separate methods for this specific example developed for the output of CMSD and JSON files. The *CMSD_Output* object is developed using the xml.etree Python library and writing in a script the whole CMSD information model for this example, so resource, process plan and process definitions of CMSD specification has been developed based on production line's logic. The same happens with *JSON_Output* object, which again is a separate method tailored to the specific example and the only difference with the *CMSD_Output* is that the JSON file is manually developed.

The above described objects along with the main script of this example and the .xls file with the processing times and scrap quantity data are available at GitHub repository in KEtool_examples folder.

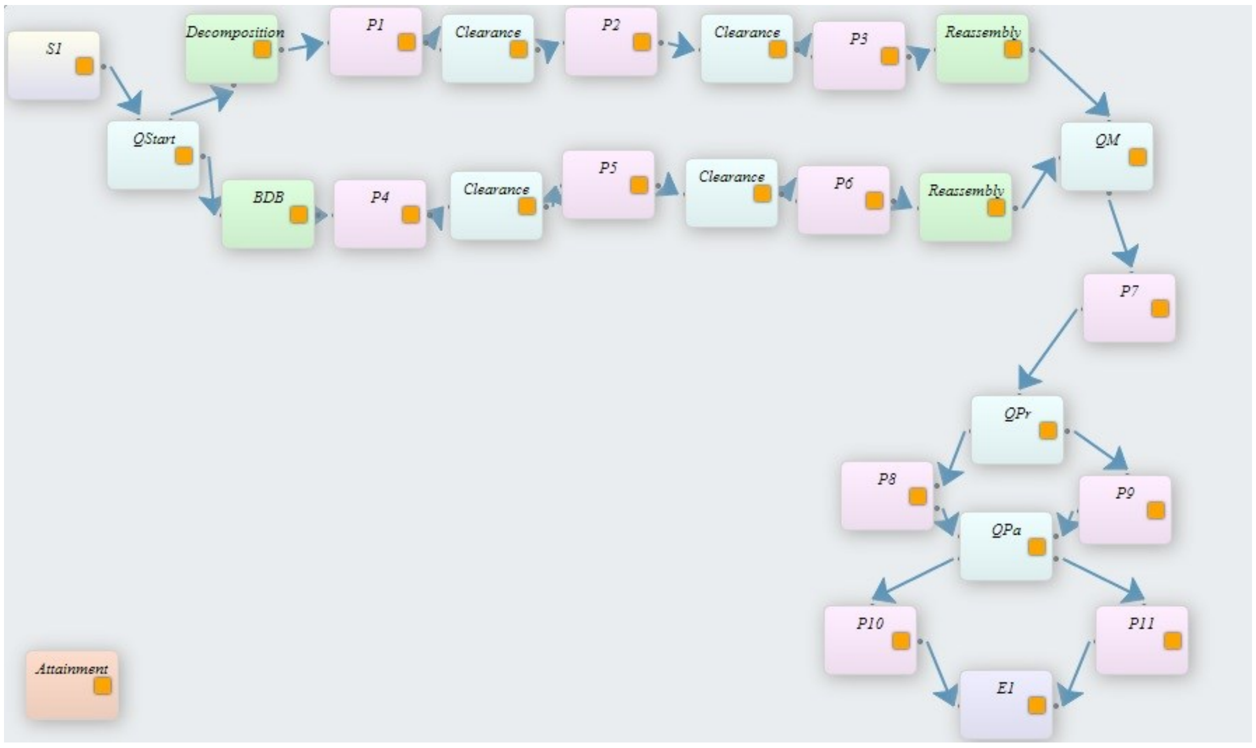


Figure 7: Production line

Below is the KE tool main script for the production line illustrated in Figure 7.

```
from dream.KnowledgeExtraction.StatisticalMeasures import BasicStatisticalMeasures
from dream.KnowledgeExtraction.DataManipulation import DataManagement
from dream.KnowledgeExtraction.DistributionFitting import DistFittest
from dream.KnowledgeExtraction.CMSD_Output import CMSD_example
from dream.KnowledgeExtraction.JSON_Output import JSON_example
from dream.KnowledgeExtraction.ExcelOutput import Output
from dream.KnowledgeExtraction.ReplaceMissingValues import HandleMissingValues
from dream.KnowledgeExtraction.ImportExceldata import Import_Excel
import xlrd

#import ManPy main JSON script
import dream.simulation.LineGenerationJSON as ManPyMain
#========================= Main script of KE tool ===============================#

#Read from the given directory the Excel document with the input data
workbook = xlrd.open_workbook('inputData.xls')
worksheets = workbook.sheet_names()
worksheet_ProcessingTimes = worksheets[1]      #Define the worksheet with the 
Processing times data
worksheet_ScrapQuantity = worksheets[0]        #Define the worksheet with the Scrap 
Quantity data

A=Import_Excel()                               #Call the Python object Import_Excel
ProcessingTimes= A.Input_data(worksheet_ProcessingTimes, workbook)    #Create the 
Processing Times dictionary with keys the different stations in the line and values 
the processing times of different batches in these stations
```

```
ScrapQuantity=A.Input_data(worksheet_ScrapQuantity, workbook)        #Create the 
Scrap Quantity dictionary with keys the different stations in the line and values the 
scrap quantity data of different batches in these stations

##Get from the Scrap Quantity dictionary the different keys and define the following 
lists with the scrap quantity data of the different stations in the topology
P1_Scrap= ScrapQuantity.get('P1',[])
P2_Scrap= ScrapQuantity.get('P2',[])
P3_Scrap= ScrapQuantity.get('P3',[])
P4_Scrap= ScrapQuantity.get('P4',[])
P5_Scrap= ScrapQuantity.get('P5',[])
P6_Scrap= ScrapQuantity.get('P6',[])
P7_Scrap= ScrapQuantity.get('P7',[])
P8_Scrap= ScrapQuantity.get('P8',[])
P9_Scrap= ScrapQuantity.get('P9',[])
P10_Scrap= ScrapQuantity.get('P10',[])
P11_Scrap= ScrapQuantity.get('P11',[])
##Get from the Processing times dictionary the different keys and define the 
following lists with the processing times data of the different stations in the 
topology
P1_Proc= ProcessingTimes.get('P1',[])
P2_Proc= ProcessingTimes.get('P2',[])
P3_Proc= ProcessingTimes.get('P3',[])
P4_Proc= ProcessingTimes.get('P4',[])
P5_Proc= ProcessingTimes.get('P5',[])
P6_Proc= ProcessingTimes.get('P6',[])
P7_Proc= ProcessingTimes.get('P7',[])
P8_Proc= ProcessingTimes.get('P8',[])
P9_Proc= ProcessingTimes.get('P9',[])
P10_Proc= ProcessingTimes.get('P10',[])
P11_Proc= ProcessingTimes.get('P11',[])
#Call the HandleMissingValues object and replace with zero the missing values in the 
lists with the scrap quantity data
B= HandleMissingValues()
P1_Scrap= B.ReplaceWithZero(P1_Scrap)
P2_Scrap= B.ReplaceWithZero(P2_Scrap)
P3_Scrap= B.ReplaceWithZero(P3_Scrap)
P4_Scrap= B.ReplaceWithZero(P4_Scrap)
P5_Scrap= B.ReplaceWithZero(P5_Scrap)
P6_Scrap= B.ReplaceWithZero(P6_Scrap)
P7_Scrap= B.ReplaceWithZero(P7_Scrap)
P8_Scrap= B.ReplaceWithZero(P8_Scrap)
P9_Scrap= B.ReplaceWithZero(P9_Scrap)
P10_Scrap= B.ReplaceWithZero(P10_Scrap)
P11_Scrap= B.ReplaceWithZero(P11_Scrap)
# #Call the BasicSatatisticalMeasures object
C= BasicStatisticalMeasures()
#Create a list with values the calculated mean value of scrap quantity on the 
different stations in the line
listScrap=[C.mean(P1_Scrap),C.mean(P2_Scrap),C.mean(P3_Scrap),C.mean(P4_Scrap),C.mean
(P5_Scrap),C.mean(P6_Scrap),C.mean(P7_Scrap),C.mean(P8_Scrap),C.mean(P9_Scrap),C.mean
(P10_Scrap), C.mean(P11_Scrap)]

D= DataManagement()
```

```
listScrap=D.round(listScrap)       #Round the mean values of the list so as to get
integers

dictScrap={}
dictScrap['P1']= listScrap[0]
dictScrap['P2']= listScrap[1]
dictScrap['P3']= listScrap[2]
dictScrap['P4']= listScrap[3]
dictScrap['P5']= listScrap[4]
dictScrap['P6']= listScrap[5]
dictScrap['P7']= listScrap[6]
dictScrap['P8']= listScrap[7]
dictScrap['P9']= listScrap[8]
dictScrap['P10']= listScrap[9]
dictScrap['P11']= listScrap[10]

E= DistFittest()      #Call the DistFittest object

dictProc={}
dictProc['P1']= E.ks_test(P1_Proc)
dictProc['P2']= E.ks_test(P2_Proc)
dictProc['P3']= E.ks_test(P3_Proc)
dictProc['P4']= E.ks_test(P4_Proc)
dictProc['P5']= E.ks_test(P5_Proc)
dictProc['P6']= E.ks_test(P6_Proc)
dictProc['P7']= E.ks_test(P7_Proc)
dictProc['P8']= E.ks_test(P8_Proc)
dictProc['P9']= E.ks_test(P9_Proc)
dictProc['P10']= E.ks_test(P10_Proc)
dictProc['P11']= E.ks_test(P11_Proc)

F= Output()
F.PrintDistributionFit(P2_Proc,"DistributionFittingResults_P2Proc.xls")
F.PrintStatisticalMeasures(P2_Proc, "StatisticalMeasuresResults_P2Proc.xls")

CMSD_example(dictProc,dictScrap)    #Print the CMSD document, calling the
CMSD_example method with arguments the dictProc and dictScrap dictionaries
JSON_example(dictProc,dictScrap)    #Print the JSON file, calling the JSON_example
method


#calls ManPy main script with the input
simulationOutput=ManPyMain.main(input_data=str((JSON_example(dictProc,dictScrap))))
# save the simulation output
jsonFile = open('ManPyOutput.json',"w")      #It opens the JSON file
jsonFile.write(simulationOutput)       #It writes the updated data to the JSON file
jsonFile.close()                     #It closes the file
```

The above main script consists of eight KE tool objects (see the comments in the script). The CMSD, JSON and Excel output files can easily obtained by downloading and running the example. In Appendices one can see the exported CMSD, JSON file and ManPy output JSON file with the simulation results.

In the tables below the simulation results after the 10 times run of the simulation model are presented. The first table shows the results of measures on the Exit station such us throughput etc., while the second table presents the results in one of model's stations, measures such as working ratio, blockage ratio and waiting ratio are illustrated with confidence intervals calculated by ManPy using the *ConfidenceIntervals* object of the KE tool.

Simulation results			
"unitsThroughput"	**"throughput"**	**"takt_time"**	**"lifespan"**
3040	32	43.69950037	761.7374959
3135	33	43.26117961	756.5203249
3230	34	42.16693359	769.0966312
2945	31	45.14615867	768.6464697
3040	32	44.95555252	801.2514402
3040	32	44.78584679	764.5391421
3135	33	43.56360076	792.4484177
3135	33	42.47776569	771.6289614
3040	32	44.60040914	769.6977449
3135	33	43.13267567	771.9918783

Simulation results		
"working_ratio"	**"blockage_ratio"**	**"waiting_ratio"**
"avg": 50.102895148547	"avg": 48.23664435861	"avg": 1.660460492836
"lb": 49.247498682836	"lb": 47.201622043610	"lb": 1.1595327223622
"ub": 50.958291614258	"ub": 49.27166667362	"ub": 2.1613882633105

5.3 Parallel stations and Queue model

Here, an example to demonstrate the use of the *ImportDatabase* object is presented. Figure 8 illustrates the model in the DREAM platform GUI. In this model there are two machines (Milling1 and Milling2) operating in parallel, one queue in advance of them, one source and one exit. The machines are subject to failures and a repairman is thus used. In this model there is only one repairman available.

Historical data on machine processing times along with the MTTF and MTTR is assumed to be available. For this particular example a simple database is developed to store this data. The database is developed in MySQL; the SQL script of this database is available in the example folder. In order to run the example the user needs to import this SQL script in their local SQL editor and create the database, using their own connection information (see *ImportDatabase* object).

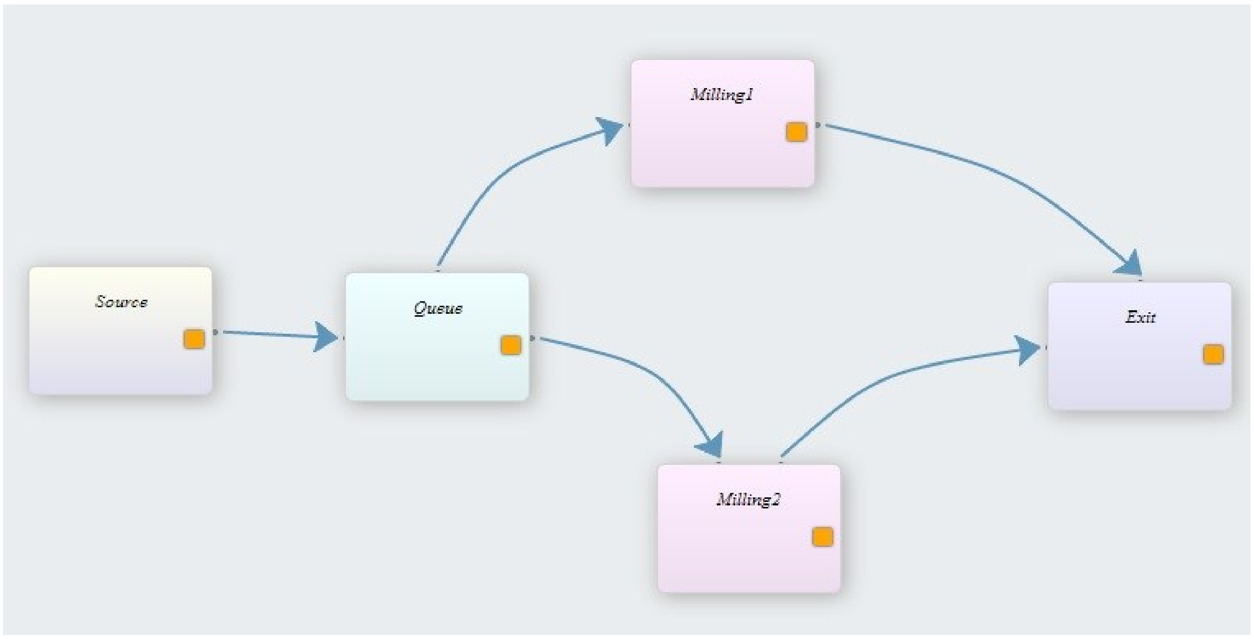


Figure 8: Parallel stations and queue model

Below is the fully commented KE tool main script for the production line illustrated in Figure 8.

```
from dream.KnowledgeExtraction.Transformations import BasicTransformations
from dream.KnowledgeExtraction.DistributionFitting import DistFittest
from dream.KnowledgeExtraction.DistributionFitting import Distributions
from dream.KnowledgeExtraction.ExcelOutput import Output
import dream.KnowledgeExtraction.ImportDatabase
import json

#====================== Extract data from the database ===================#
cnxn=ImportDatabase.ConnectionData(seekName='ServerData', implicitExt='txt',
number_of_cursors=3)
cursors=cnxn.getCursors()
```

```
a = cursors[0].execute("""
        select prod_code, stat_code,emp_no, TIMEIN, TIMEOUT
        from production_status
                """)
MILL1=[]
MILL2=[]
for j in range(a.rowcount):
    #get the next line
    ind1=a.fetchone()
    if ind1.stat_code == 'MILL1':
        procTime=[]
        procTime.insert(0,ind1.TIMEIN)
        procTime.insert(1,ind1.TIMEOUT)
        MILL1.append(procTime)
    elif ind1.stat_code == 'MILL2':
        procTime=[]
        procTime.insert(0,ind1.TIMEIN)
        procTime.insert(1,ind1.TIMEOUT)
        MILL2.append(procTime)
    else:
        continue
```

Database query to extract the needed data for processing

```
transform = BasicTransformations()
procTime_MILL1=[]
for elem in MILL1:
    t1=[]
    t2=[]
    t1.append(((elem[0].hour)*60)*60 + (elem[0].minute)*60 + elem[0].second)
    t2.append(((elem[1].hour)*60)*60 + (elem[1].minute)*60 + elem[1].second)
    dt=transform.subtraction(t2, t1)
    procTime_MILL1.append(dt[0])

procTime_MILL2=[]
for elem in MILL2:
    t1=[]
    t2=[]
    t1.append(((elem[0].hour)*60)*60 + (elem[0].minute)*60 + elem[0].second)
    t2.append(((elem[1].hour)*60)*60 + (elem[1].minute)*60 + elem[1].second)
    dt=transform.subtraction(t2, t1)
    procTime_MILL2.append(dt[0])


b = cursors[1].execute("""
        select stat_code, MTTF_hour
        from failures
                """)

c = cursors[2].execute("""
        select stat_code, MTTR_hour
        from repairs
                """)
MTTF_MILL1=[]
MTTF_MILL2=[]
for j in range(b.rowcount):
    #get the next line
    ind2=b.fetchone()
    if ind2.stat_code == 'MILL1':
        MTTF_MILL1.append(ind2.MTTF_hour)
    elif ind2.stat_code == 'MILL2':
        MTTF_MILL2.append(ind2.MTTF_hour)
    else:
        continue

MTTR_MILL1=[]
MTTR_MILL2=[]
for j in range(c.rowcount):
    #get the next line
    ind3=c.fetchone()
    if ind3.stat_code == 'MILL1':
        MTTR_MILL1.append(ind3.MTTR_hour)
    elif ind3.stat_code == 'MILL2':
        MTTR_MILL2.append(ind3.MTTR_hour)
    else:
        continue

#====================== Fit data to statistical distributions ====================#
dist_proctime = DistFittest()
distProcTime_MILL1 = dist_proctime.ks_test(procTime_MILL1)
distProcTime_MILL2 = dist_proctime.ks_test(procTime_MILL2)

dist_MTTF = Distributions()
dist_MTTR = Distributions()
distMTTF_MILL1 = dist_MTTF.Weibull_distrfit(MTTF_MILL1)
distMTTF_MILL2 = dist_MTTF.Weibull_distrfit(MTTF_MILL2)

distMTTR_MILL1 = dist_MTTR.Poisson_distrfit(MTTR_MILL1)
distMTTR_MILL2 = dist_MTTR.Poisson_distrfit(MTTR_MILL2)

#==Output preparation: output the updated values in the JSON file of this example ==#
jsonFile = open('JSON_example.json','r')      #It opens the JSON file
data = json.load(jsonFile)
#It loads the file
jsonFile.close()
nodes = data.get('nodes',[])
#It creates a variable that holds the 'nodes' dictionary

for element in nodes:
    processingTime = nodes[element].get('processingTime',{})        #It creates a
variable that gets the element attribute 'processingTime'
    MTTF_Nodes = nodes[element].get('MTTF',{})                            #It creates
a variable that gets the element attribute 'MTTF'
    MTTR_Nodes = nodes[element].get('MTTR',{})                            #It creates
a variable that gets the element attribute 'MTTR'

    if element == 'M1':
        nodes['M1']['processingTime'] = distProcTime_MILL1         #It checks using
if syntax if the element is 'M1'
        nodes['M1']['failures']['MTTF'] = distMTTF_MILL1
        nodes['M1']['failures']['MTTR'] = distMTTR_MILL1
    elif element == 'M2':
        nodes['M2']['processingTime'] = distProcTime_MILL2         #It checks using 
if syntax if the element is 'M2'
        nodes['M2']['failures']['MTTF'] = distMTTF_MILL2
        nodes['M2']['failures']['MTTR'] = distMTTR_MILL2

    jsonFile = open('JSON_ParallelStations_Output.json',"w")     #It opens the JSON 
file
    jsonFile.write(json.dumps(data, indent=True))
#It writes the updated data to the JSON file
    jsonFile.close()
#It closes the file

 #= Calling the ExcelOutput object, outputs the outcomes of the statistical analysis 
                                 in xls files ==#
export=Output()

export.PrintStatisticalMeasures(procTime_MILL1,'procTimeMILL1_StatResults.xls')
export.PrintStatisticalMeasures(procTime_MILL2,'procTimeMILL2_StatResults.xls')
export.PrintStatisticalMeasures(MTTF_MILL1,'MTTFMILL1_StatResults.xls')
export.PrintStatisticalMeasures(MTTF_MILL2,'MTTFMILL2_StatResults.xls')
export.PrintStatisticalMeasures(MTTR_MILL1,'MTTRMILL1_StatResults.xls')
export.PrintStatisticalMeasures(MTTR_MILL2,'MTTRMILL2_StatResults.xls')

export.PrintDistributionFit(procTime_MILL1,'procTimeMILL1_DistFitResults.xls')
export.PrintDistributionFit(procTime_MILL2,'procTimeMILL2_DistFitResults.xls')
export.PrintDistributionFit(MTTF_MILL1,'MTTFMILL1_DistFitResults.xls')
export.PrintDistributionFit(MTTF_MILL2,'MTTFMILL2_DistFitResults.xls')
export.PrintDistributionFit(MTTR_MILL1,'MTTRMILL1_DistFitResults.xls')
export.PrintDistributionFit(MTTR_MILL2,'MTTRMILL2_DistFitResults.xls')
```

The above main script consists of five KE tool objects (see the comments in the script). The JSON and Excel output files can easily obtained by downloading and running the example. In Appendices one can see the exported JSON file.

5.4 Assembly and Dismantle model

This model contains two sources, one assembly station, one machine, one dismantling station and two exits. Figure 9 illustrates the graphical representation of the simulation model modeled in the DREAM platform GUI. The machine in this example is vulnerable to failures and we've got information about the time to failures (TTF) and time to repairs (TTR). Also, we have info about the processing times of the machine. The three categories data are recorded in an .xls file.

Five objects are used in this model. The steps conducted in the main script described are listed below:

1. Import the needed objects in order to run the main script.
2. Read from the given directory the document with the necessary data.
3. Call the *Import_Excel* object in order to input the data to the tool.
4. From the imported data (python dictionaries) select the required data and put them in separate lists.

5. Call the *ReplaceMissingValues* object and apply its method *ReplaceWithMean*, which replaces the missing values with the mean of the non-missing values in the list.
6. Call the *Distributions* (Maximum Likelihood Estimation) and *DistFittest* (Kolmogorov-Smirnov fitting test) methods and apply them to the data, it is selected to conduct Kolmogorov-Smirnov test in the processing times data and to fit in Exponential distribution the MTTF and MTTR data.
7. Export the processed data (statistical distributions of processing times, MTTF and MTTR) to the developed JSON file of the model.
8. Call the *ExcelOutput* object and using its methods export the statistical analysis and distribution fitting results of the three data categories.

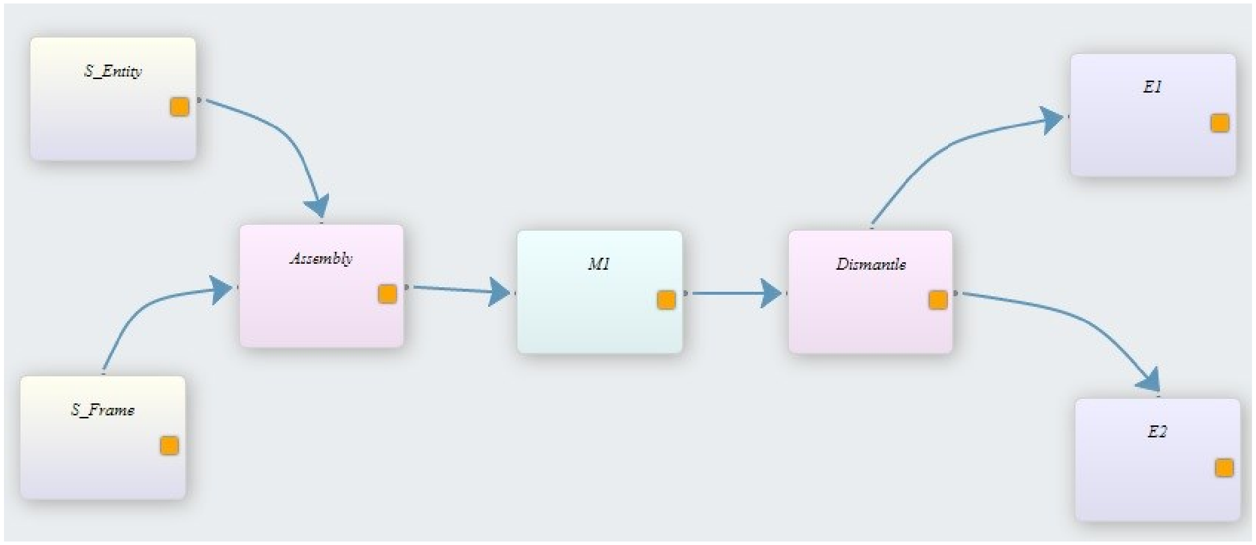


Figure 9: Assembly and dismantle model

Below is the KE tool main script.

```
from dream.KnowledgeExtraction.ImportExceldata import Import_Excel
from dream.KnowledgeExtraction.ReplaceMissingValues import HandleMissingValues
from dream.KnowledgeExtraction.DistributionFitting import Distributions
from dream.KnowledgeExtraction.DistributionFitting import DistFittest
from dream.KnowledgeExtraction.ExcelOutput import Output
import xlrd
import json
```
1

```
#Read from the given directory the Excel document with the input data
workbook = xlrd.open_workbook('inputData.xls')
worksheets = workbook.sheet_names()
worksheet_ProcessingTimes = worksheets[0]     #Define the worksheet with the 
Processing times data
worksheet_MTTF = worksheets[1]       #Define the worksheet with Time-to-Failure data
worksheet_MTTR = worksheets[2]       #Define the worksheet with Time-to-Repair data
```
2

```
A = Import_Excel()                     #Call the Python object Import_Excel
ProcessingTimes = A.Input_data(worksheet_ProcessingTimes, workbook)   #Create the
Processing Times dictionary with key the Machine 1 and values the processing time
data
MTTF=A.Input_data(worksheet_MTTF, workbook)        #Create the MTTF dictionary with
key the Machine 1 and time-to-failure data
MTTR=A.Input_data(worksheet_MTTR, workbook)        #Create the MTTR Quantity
dictionary with key the Machine 1 and time-to-repair data
```
3

```
##Get from the above dictionaries the M1 key and define the following lists with
data
ProcTime = ProcessingTimes.get('M1',[])
MTTF = MTTF.get('M1',[])
MTTR = MTTR.get('M1',[])
```
4

```
#Call the HandleMissingValues object and replace the missing values in the
Lists with the mean of the non-missing values
B =HandleMissingValues()
ProcTime = B.ReplaceWithMean(ProcTime)
MTTF = B.ReplaceWithMean(MTTF)
MTTR = B.ReplaceWithMean(MTTR)
```
5

```
C = Distributions()      #Call the Distributions object
D = DistFittest()      #Call the DistFittest object

ProcTime_dist = D.ks_test(ProcTime)
MTTF_dist = C.Exponential_distrfit(MTTF)
MTTR_dist = C.Exponential_distrfit(MTTR)
```
6

```
#== Output preparation: output the updated values in the JSON file of this
example==#
jsonFile = open('JSON_AssembleDismantle.json','r')      #It opens the JSON file
data = json.load(jsonFile)
jsonFile.close()#It loads the file
nodes = data.get('nodes',[]) #It creates a variable that holds the 'nodes' dictionary

for element in nodes:
    processingTime = nodes[element].get('processingTime',{})  #It creates a variable
that gets the element attribute 'processingTime'
    MTTF_Nodes = nodes[element].get('MTTF',{}) #It creates a variable that gets the
element attribute 'MTTF'
    MTTR_Nodes = nodes[element].get('MTTR',{}) #It creates a variable that gets the
element attribute 'MTTR'

    if element == 'M1':
        nodes['M1']['processingTime'] = ProcTime_dist  #It checks using if syntax if
the element is 'M1'
        nodes['M1']['failures']['MTTF'] = MTTF_dist
        nodes['M1']['failures']['MTTR'] = MTTR_dist
        continue

    jsonFile = open('JSON_AssembleDismantle_Output.json',"w") #It opens the JSON file
    jsonFile.write(json.dumps(data, indent=True))
#It writes the updated data to the JSON file
    jsonFile.close()  #It closes the file
```
7

```
    #== Calling the ExcelOutput object, outputs the outcomes of the statistical
                        analysis in xls files ===============#
C=Output()
C.PrintStatisticalMeasures(ProcTime,'ProcTime_StatResults.xls')
C.PrintStatisticalMeasures(MTTR,'MTTR_StatResults.xls')
C.PrintStatisticalMeasures(MTTF,'MTTF_StatResults.xls')
C.PrintDistributionFit(ProcTime,'ProcTime_DistFitResults.xls')
C.PrintDistributionFit(MTTR,'MTTR_DistFitResults.xls')
```

As happens with all the examples of the documentation the above main script is available in GitHub. Again the JSON and Excel output files can easily obtained by downloading and running the example.

5.5 Parallel stations model

Here, an example to demonstrate the ImportExceldata object is presented. The parallel stations model previously introduced (see Figure 12) is used. Historical processing times are recorded in a spreadsheet named inputData.xls.

ManPy is called from the KE tool main script and the simulation model is run. Below is the KE tool main script.

```
from DistributionFitting import DistFittest

from DistributionFitting import Distributions
from ImportExceldata import Import_Excel
from ExcelOutput import Output
from ReplaceMissingValues import HandleMissingValues
import xlrd
import json
#import ManPy main JSON script
import dream.simulation.LineGenerationJSON as ManPyMain

#Read from the given directory the Excel document with the input data
workbook = xlrd.open_workbook('inputData.xls')
worksheets = workbook.sheet_names()
worksheet_ProcessingTimes = worksheets[0]     #Define the worksheet with the
Processing times data

inputData = Import_Excel()                              #Call the Python object
Import_Excel
ProcessingTimes = inputData.Input_data(worksheet_ProcessingTimes, workbook)   #Create
the Processing Times dictionary with key Machines 1,2 and values the processing time
data
##Get from the above dictionaries the M1 key and define the following lists with data
M1_ProcTime = ProcessingTimes.get('M1',[])
M2_ProcTime = ProcessingTimes.get('M2',[])
#Call the HandleMissingValues object and replace the missing values in the lists with
the mean of the non-missing values
misValues =HandleMissingValues()
M1_ProcTime = misValues.ReplaceWithMean(M1_ProcTime)
M2_ProcTime = misValues.ReplaceWithMean(M2_ProcTime)
```

```
MLE = Distributions()      #Call the Distributions object (Maximum Likelihood
Estimation - MLE)
KS = DistFittest()      #Call the DistFittest object  (Kolmoghorov-Smirnov test)

M1ProcTime_dist = KS.ks_test(M1_ProcTime)
M2ProcTime_dist = MLE.Normal_distrfit(M2_ProcTime)
#======================== Output preparation: output the updated values in the JSON
file of this example ================================#
jsonFile = open('JSON_TwoParallelStations.json','r')      #It opens the JSON file
data = json.load(jsonFile)
#It loads the file
jsonFile.close()
nodes = data.get('nodes',[])
#It creates a variable that holds the 'nodes' dictionary

for element in nodes:
    processingTime = nodes[element].get('processingTime',{})        #It creates a
variable that gets the element attribute 'processingTime'
    if element == 'St1':
        nodes['St1']['processingTime'] = M1ProcTime_dist         #It checks using if
syntax if the element is 'M1'
    elif element == 'St2':
        nodes['St2']['processingTime'] = M2ProcTime_dist         #It checks using if
syntax if the element is 'M2'
    jsonFile = open('JSON_ParallelStations_Output.json',"w")     #It opens the JSON
file
    jsonFile.write(json.dumps(data, indent=True))
#It writes the updated data to the JSON file
    jsonFile.close()
#It closes the file
#==================== Calling the ExcelOutput object, outputs the outcomes of the
statistical analysis in xls files ==========================#
export=Output()
export.PrintStatisticalMeasures(M1_ProcTime,'M1_ProcTime_StatResults.xls')
export.PrintStatisticalMeasures(M2_ProcTime,'M2_ProcTime_StatResults.xls')
export.PrintDistributionFit(M1_ProcTime,'M1_ProcTime_DistFitResults.xls')
export.PrintDistributionFit(M2_ProcTime,'M2_ProcTime_DistFitResults.xls')
#calls ManPy main script with the input
simulationOutput=ManPyMain.main(input_data=json.dumps(data))
# save the simulation output
jsonFile = open('ManPyOutput.json',"w")     #It opens the JSON file
jsonFile.write(simulationOutput)        #It writes the updated data to the JSON file
jsonFile.close()                        #It closes the file
```

The folder TwoParallelStations contains the files needed to run the simulation model and obtain the results applying the KE tool main script (TwoParallelStations_example.py). We now present the output trace from the simulation model. This illustrates how ManPy picked up the input data produced by the KE tool and used that in running the model. Therefore, it can now be seen that all modules of the platform have combined to build a model, import and manipulate data to a useful format and then pass this to the simulation engine for model execution.

Figure 10 depicts the different JSON files for the stations, St1 and St2. On the left are the default instances; when a station is created by dragging and dropping with the GUI. On the right are the JSON files for the same objects after the spreadsheet data (containing data on the processing times for St1 and St2) has been imported and analysed and suitable distributions and parameters have been chosen and applied for each station. It can be noted that a Gamma distribution was determined as the best fit for St1 and therefore the shape and rate (also known as scale) parameters were calculated. A normal distribution was determined for St2. Therefore, the associated min, max, mean and standard deviation parameters were subsequently calculated.

```
0 Batch0 got into Queue
0 Batch0 got into Milling2
1.0 Batch1 got into Queue
1.0 Batch1 got into Milling1
2.0 Batch2 got into Queue
39.0494038135 Batch0 got into Exit
39.0494038135 Batch2 got into Milling2
39.0494038135 Batch3 got into Queue
98.8013056079 Batch2 got into Exit
98.8013056079 Batch3 got into Milling2
98.8013056079 Batch4 got into Queue
0 Batch0 got into Queue
0 Batch0 got into Milling2
1.0 Batch1 got into Queue
1.0 Batch1 got into Milling1
2.0 Batch2 got into Queue
55.0034334831 Batch0 got into Exit
```

Figure 10: Stations simulation input data before run the KE tool (left), updated simulation input data (right)

5.6 Assembly line

The KE tool main script was created and tested in a real assembly line. Figure 11 illustrates the graphical representation of the assembly line modeled in the DREAM platform GUI. In this model we have several Machines operating in parallel, Queues between them and other simulation objects between them. The Machines are vulnerable to failures so scrap parts produced.

This case study uses different data sources to record operational information. Raw data from the assembly line are recorded in batch level on the local MES 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. Figure 12 demonstrates the format of data in the MES database, the columns contain different information about the events, for example start time, stop time, station name, container name, quantity of the container etc.

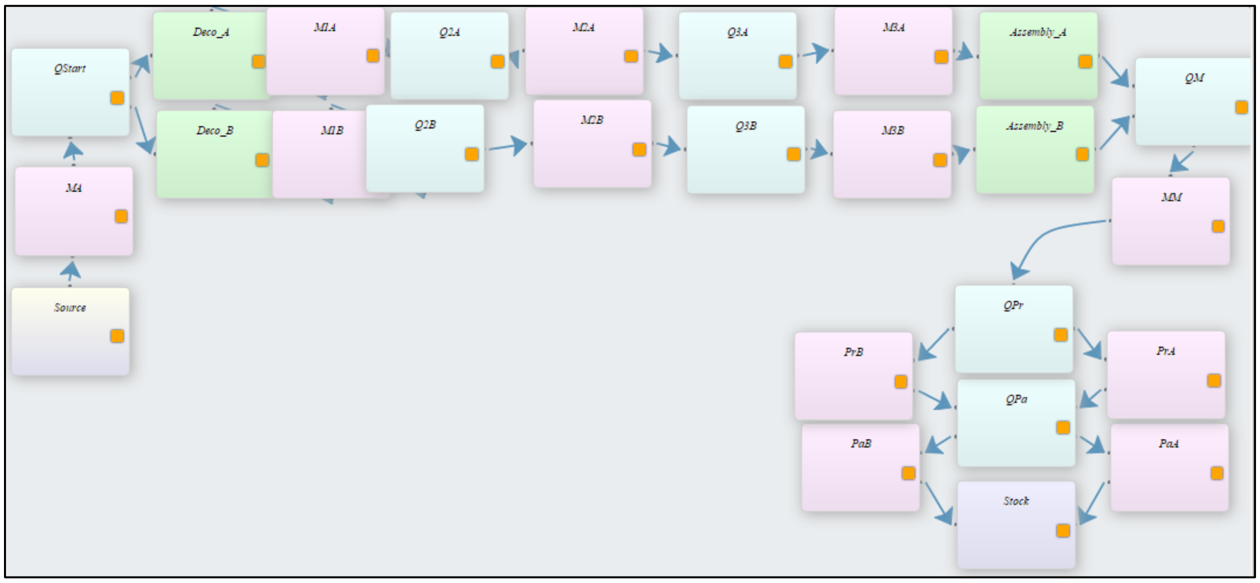
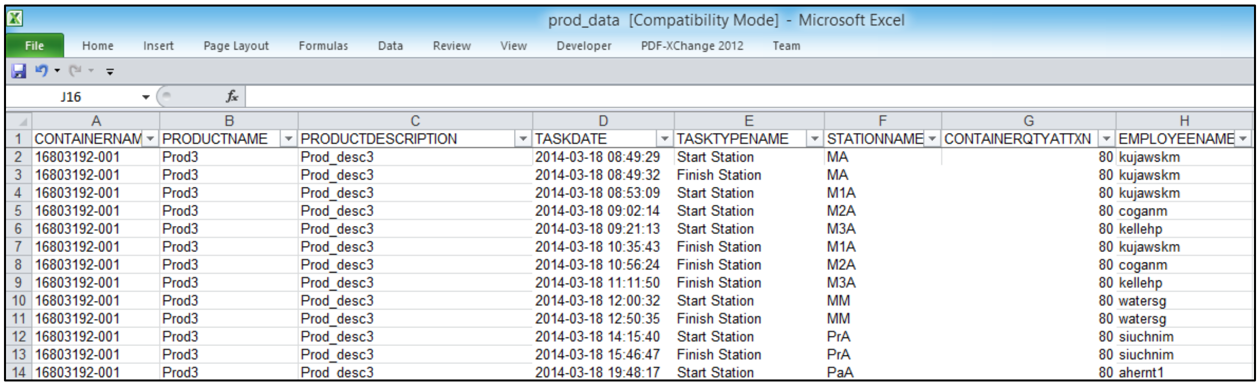


Figure 11: Assembly line modeled in DREAM.GUI



	A	B	C	D	E	F	G	H
1	CONTAINERNAM	PRODUCTNAME	PRODUCTDESCRIPTION	TASKDATE	TASKTYPENAME	STATIONNAME	CONTAINERQTYATTXN	EMPLOYEENAME
2	16803192-001	Prod3	Prod_desc3	2014-03-18 08:49:29	Start Station	MA	80	kujawskm
3	16803192-001	Prod3	Prod_desc3	2014-03-18 08:49:32	Finish Station	MA	80	kujawskm
4	16803192-001	Prod3	Prod_desc3	2014-03-18 08:53:09	Start Station	M1A	80	kujawskm
5	16803192-001	Prod3	Prod_desc3	2014-03-18 09:02:14	Start Station	M2A	80	coganm
6	16803192-001	Prod3	Prod_desc3	2014-03-18 09:21:13	Start Station	M3A	80	kellehp
7	16803192-001	Prod3	Prod_desc3	2014-03-18 10:35:43	Finish Station	M1A	80	kujawskm
8	16803192-001	Prod3	Prod_desc3	2014-03-18 10:56:24	Finish Station	M2A	80	coganm
9	16803192-001	Prod3	Prod_desc3	2014-03-18 11:11:50	Finish Station	M3A	80	kellehp
10	16803192-001	Prod3	Prod_desc3	2014-03-18 12:00:32	Start Station	MM	80	watersg
11	16803192-001	Prod3	Prod_desc3	2014-03-18 12:50:35	Finish Station	MM	80	watersg
12	16803192-001	Prod3	Prod_desc3	2014-03-18 14:15:40	Start Station	PrA	80	siuchnim
13	16803192-001	Prod3	Prod_desc3	2014-03-18 15:46:47	Finish Station	PrA	80	siuchnim
14	16803192-001	Prod3	Prod_desc3	2014-03-18 19:48:17	Start Station	PaA	80	ahernt1

Figure 12: Example of the MES database structure in the assembly line

In this example the KE tool interacts with the data file executing the three main components "Data extraction", "Data processing" and "Output preparation". The Python objects needed for the three components incorporated that form the KE tool are included in the main script for this example.

Company's requirement was the extraction of the processing times, the scrap quantity and the WIP levels. For the fulfilment of the above requirements a method was developed (that is called by the main script) apart from the KE tool main script in order to identify the WIP levels in the line at the time that the engineer in the line desires to run the simulation model. Executing the KE tool main script a JSON file that follows the JSON schema (data exchange format for DES) is exported with info needed for the simulation model, for example the continuous statistical distributions for the processing times of each station, the discrete statistical distribution (Geometric distribution) for the scrap quantity appeared in each station and the WIP level either in queues (buffers) or stations (machines). In the latter case (WIP in machines) it is also estimated the approximate quantity of units left to be processed, the estimation was conducted calculating the average processing time in a particular machine, and the quantity of time between the start time in this machine and the stop time (time that the engineer wants to run the simulation model, so the KE tool main script gets executed).

A number of objects were used to form the main script and the outcome is the exported JSON file that contains the required information for the simulation model. The KE tool main script along with the spreadsheet data (see Figure 12) can be found at GitHub repository in KEtool_examples folder. In the Appendices can be found the exported JSON file for this example.

5.7 Example using the Plots object

KE tool offer methods for output analysis of the simulation results. One of the developed objects that mostly referring to Output analysis component of the tool (see Figure 1) is the *Plots* object. Applying *Plots* object in given data sets, we can get plots and charts with the representations of the data points in charts.

A simple example to demonstrate the use of this object is developed. This example as happens with the other examples is in GitHub repository. In this example data retrieved from a CSV file using the *ImportCSVdata* object and apply the data sets in *Plots'* methods returning the different charts.

- In the beginning the Graphs and the Import_CSV modules are imported:

  from dream.KnowledgeExtraction.Plots import Graphs

  from dream.KnowledgeExtraction.ImportCSVdata import Import_CSV

- Then we call the Import_CSV module and using its method *Input_data* import the data set from the CSV file to the tool

  filename = ("DataSet.csv")

A=Import_CSV()
Data = A.Input_data(filename)

- After we get from the returned Python dictionary the two data sets:

  M1 = Data.get('M1',[])
  M2 = Data.get('M2',[])

- Then a Graph object is created and all its methods applied to the data sets (M1,M2)

  #create a graph object

  B=Graphs()

  B.Plots(M1, 'M1SimplePlot.jpg')

  B.ScatterPlot(M1, M2, 'Scatterplot.jpg')

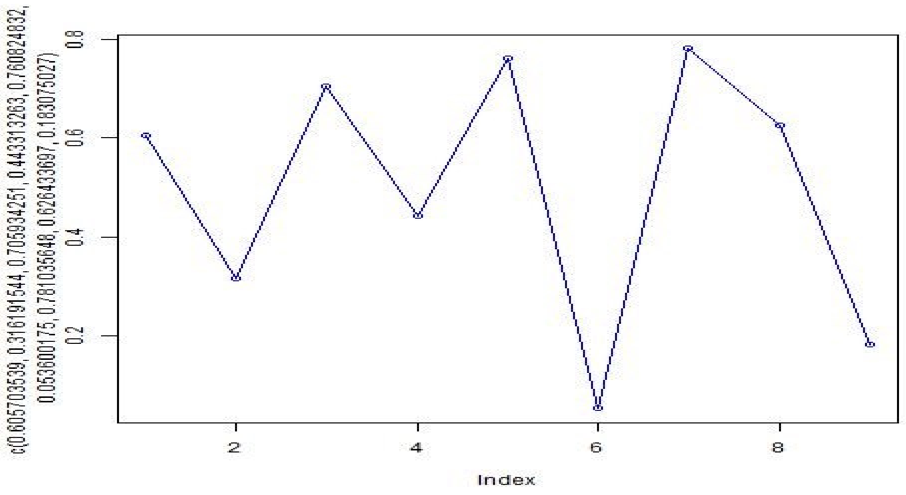
  B.Barplot(M2, 'M2Barplot.jpg')

  B.Histogram(M1, 'M1Histogram.jpg')

  B.TwoSetPlot(M1, M2, 'M1M2Plot.jpg')

  B.Pie(M2, 'M2PieChar.jpg')

Below find an example of the obtained .jpg files with the above charts. All the files can be easily obtained running the example.



As another example we present the dream\simulation\Examples\TwoServersPlots.py, this example outputs a pie chart that presents graphically the percentage of time that the repairman is busy or idle.

The new entries on the already existing dream\simulation\Examples\TwoServers.py on the code are:

- In the beginning the Graphs module is imported:

  from dream.KnowledgeExtraction.Plots import Graphs

- After the simulation run the values for the pie are calculated:

  #calculate the percentages for the pie

  *repairmanWorkingRatio=R.totalWorkingTime/G.maxSimTime*100*

  *repairmanWaitingRatio=R.totalWaitingTime/G.maxSimTime*100*

- Then a Graph object is created and the *Pie* method is called in order to create the output file

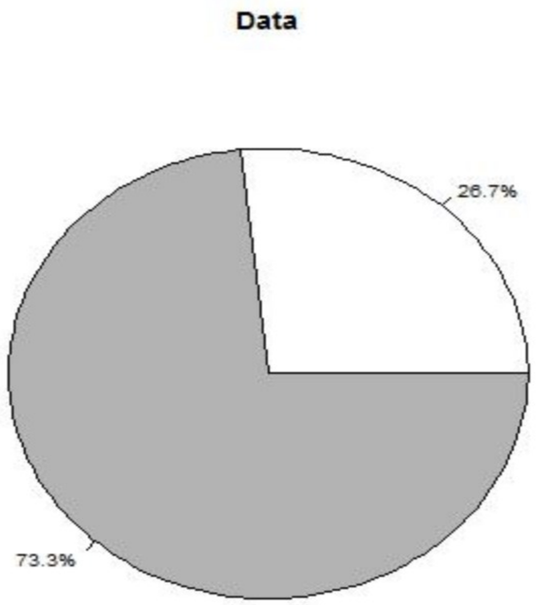
  #create a graph object

  graph=Graphs()

  #create the pie

  graph.Pie([repairmanWorkingRatio,repairmanWaitingRatio], "repairmanPie.jpg")

Running the script the user gets in addition to the console output repairmanPie.jpg that contains the following graph:



5.8 Example using the Confidence Intervals object

A very simple example is developed in order to describe the use of the *ConfidenceIntervals* object of the tool. This object offers the functionality to calculate confidence intervals in given data sets. The calculation of confidence intervals is really crucial in the output analysis of simulation.

This example as happens with the other examples is in GitHub repository in dream\KnowledgeExtraction\KEtool_examples\ConfidenceIntervals\ConfidenceIntervals_example.py. In this example data retrieved from a CSV file using the *ImportCSVdata* object and apply the data sets in *ConfidenceInterval's* method returning the actual lower and upper bound (confidence intervals) of the data sets.

- In the beginning the ConfidenceInterval and the Import_CSV modules are imported:

  from dream.KnowledgeExtraction.ConfidenceIntervals import Intervals

  from dream.KnowledgeExtraction.ImportCSVdata import Import_CSV

- Then we call the Import_CSV module and using its method *Input_data* import the data set from the CSV file to the tool

  ```
  filename = ("DataSet.csv")
  data=Import_CSV()
  Data = data.Input_data(filename)
  ```

- After we get from the returned Python dictionary the three data sets:

  ```
  #get from the returned Python dictionary the following three data sets

  ProcTime = Data.get('ProcessingTimes',[])
  MTTF = Data.get('MTTF',[])
  MTTR = Data.get('MTTR',[])
  ```

- Then a Intervals object is created and all its method applied to the three data sets

  ```
  #create a Intervals object

  CI=Intervals()

  #print the confidence intervals of the data sets applying either 90% or 95% probability

  print CI.ConfidIntervals(ProcTime, 0.95)

  print CI.ConfidIntervals(MTTF, 0.90)

  print CI.ConfidIntervals(MTTR, 0.95)
  ```

Below the returning lists with the calculated upper and lower bound (confidence intervals) of the three data sets.

```
[0.4558596307362209, 0.5351412918337792]

[0.49255403041288176, 0.5575672826771183]

[0.4719902587261917, 0.5545222108338084]
```

5.9 Single Server in Simul8

A simple example is developed to demonstrate the integration of the KE tool and the Simul8 simulation package. In Figure 13 the graphical representation of the Single Server model in the Simul8 software is illustrated. The simple model is consisted of one queue (Queue for Activity 1), one server (Activity 1), one source (Source) and one exit (End).

The available information in this example is the processing times of the Server (Activity 1) and the inter-arrival times of the work items. This data is provided in two different Excel documents. The KE tool main script starts with the extraction of this data using the *ImportExceldata* object, the distribution fitting is the next task using both the Maximum Likelihood Estimation (MLE) and the Kolmogorov-Smirnov test. For the inter-arrivals times the Exponential distribution is used for the statistical representation of this data, while the Kolmogorov-Smirnov distribution fitting test is used for the processing times. Finally, using the different features of the Simul8XML object the simulation information come from the distribution fitting update the values in the XML file of this example.

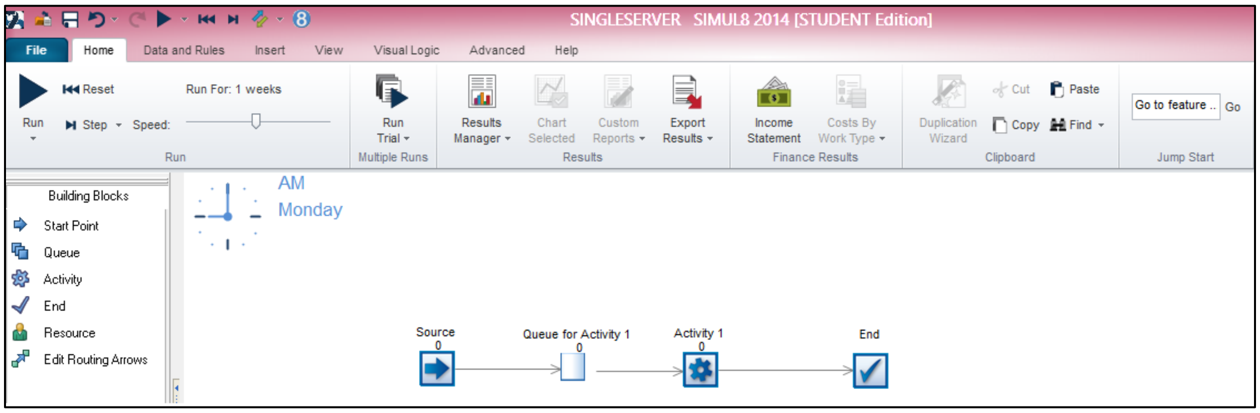


Figure 13: Single server model in Simul8

As it happens with the other examples of the KE tool, more information of this example can be found in GitHub. The KE tool main script of this example along with the Excel documents and the simul8 XML file are available in the repository. By executing the KE tool main script the updated XML file is exported, opening this file the user is able to run the simulation model in the Simul8 simulation software.

5.10 Parallel Stations with failures in Simul8

An additional similar to the example 3.3 is created to demonstrate the integration of the KE tool and the Simul8 simulation package. Figure 14 illustrates the graphical representation of the model in Simul8 simulation software. In the model there are two Milling machines MILL1 and MILL2 operating in parallel, one Queue before them, one Source and one exit (End). The Milling machines are prone to failures; a repairman is responsible to fix these failures when they happen.

A database is available for this specific example with information on the processing times, the MTTR and MTBF in the two Milling machines. In order to run the example the user needs to import this SQL script in his local SQL editor and create the database, using their own connection information (see *ImportDatabase* object). After the importing of this data, distribution fitting is conducted using the *DistributionFitting* object. Finally, using the different features of the *Simul8XML* object the simulation information come from the distribution fitting update the values in the XML file of this example.

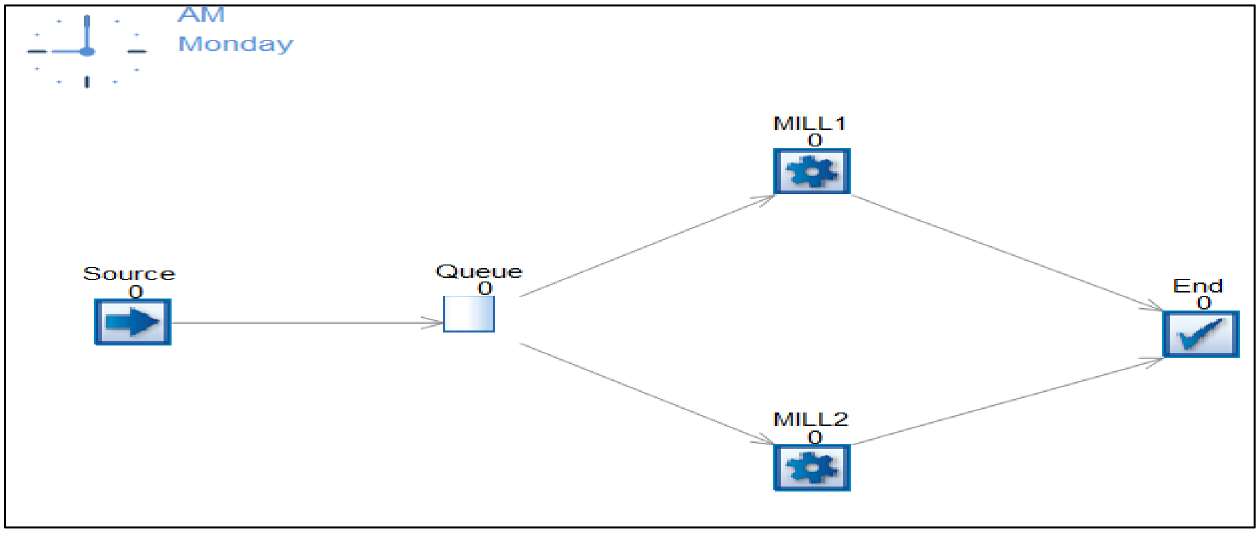


Figure 14: Parallel stations with failures in Simul8

The KE tool main script of this example along with the developed database and the simul8 XML file can be found in GitHub at the examples of the KE tool. By running the ParallelStationsFailures.py (KE tool main script) one can obtain the updated based on the available data simul8 XML file, opening this file is able to run the simulation model in the Simul8 simulation software.

5.11 Topology1 in Simul8

Another example is created between the KE tool and the Simul8 simulation package. In Figure 15 the graphical representation of the Topology 1 model is illustrated in Simul8 package. The Topology1 model is consisted of three Activities (two of them operating in parallel), two queues, one Source and one End.

The inter-arrival times for this example along with the MTTF and MTTR data for the Activity 1 are recorded in Excel document, while the processing times of Activity2 and Activity3 are available in a CSV file. The KE tool main script starts with the extraction of the above data from the Excel and CSV file using the *ImportCSVdata* and *ImportExceldata* objects. As for the Data processing component of the tool, the *DistributionFitting* object is used, both the Kolmogorov-

Smirnov distribution fitting test and the fitting using the MLE method are applied. As for the third component of the tool (Output preparation), different features of the *Simul8XML* object are used.

The KE tool main script along with the Excel and CSV files with the data and the simul8 XML of this example are available in GitHub repository. By executing the KE tool main script the updated XML file is exported, opening this file the user is able to run the simulation model in the Simul8 simulation software.

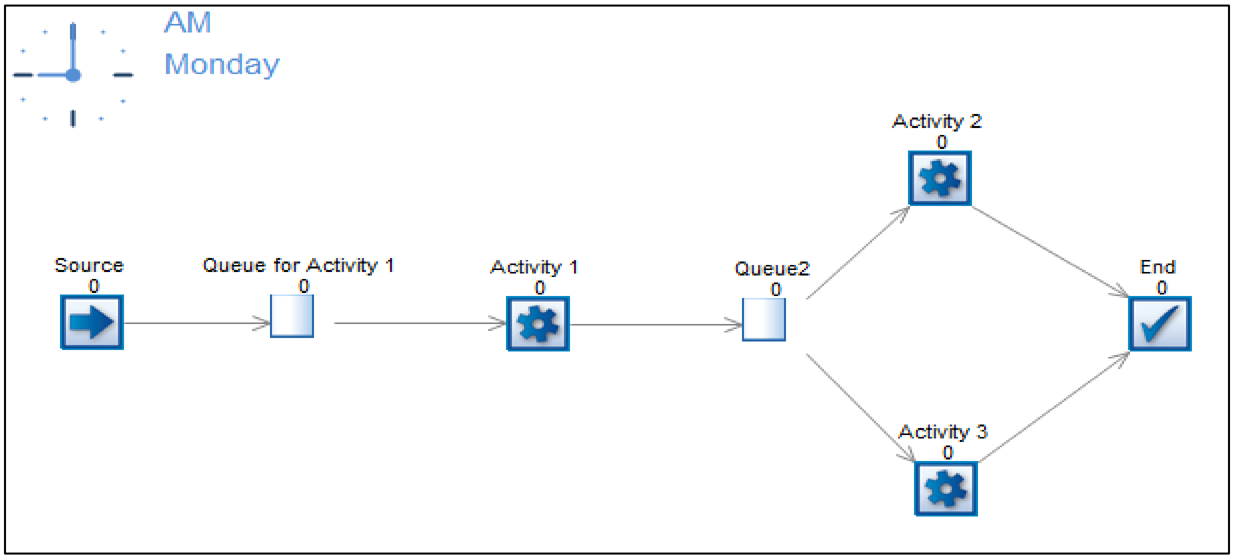


Figure 15: Topology1 in Simul8

Appendices

- **Example Two servers model with failures and repairman**

Below the three different output data formats are presented. The highlighted parts reveal the inputs of the KE tool in the processing times of Machine 1 and 2.

CMSD information model of the example

```
<?xml version='1.0' encoding='utf8'?>
<CMSDDocument>

       <DataSection>
              <PartType>
                     <Identifier>Part1</Identifier>
              </PartType>
              <PartType>
                     <Identifier>UnfinishedPart1</Identifier>
              </PartType>
            <Resource>
                     <Identifier>S1</Identifier>
                     <Description>The source of the topology</Description>
                     <ResourceType>Source</ResourceType>
                     <Name>RawMaterial</Name>
               </Resource>
              <Resource>
                     <Identifier>M1</Identifier>
                     <Description>The lathe of the topology</Description>
                     <ResourceType>Machine</ResourceType>
                     <Name>Machine1</Name>
               </Resource>
              <Resource>
                     <Identifier>M2</Identifier>
                     <Description>The moulding machine of the topology</Description>
                     <ResourceType>Machine</ResourceType>
                     <Name>Machine2</Name>
               </Resource>
              <Resource>
                     <Identifier>Queue</Identifier>
                     <Description>The queue of the topology</Description>
                     <ResourceType>Queue</ResourceType>
                     <Name>Queue</Name>
               </Resource>
              <Resource>
                     <Identifier>Exit</Identifier>
                     <Description>The exit of the topology</Description>
                     <ResourceType>Exit</ResourceType>
                     <Name>Stock</Name>
               </Resource>

              <Resource>
                     <Identifier>A</Identifier>
                     <ResourceType>employee</ResourceType>
              </Resource>
              <Resource>
                     <Identifier>B</Identifier>
                     <ResourceType>employee</ResourceType>
              </Resource>
              <Resource>
                     <Identifier>Repairman</Identifier>
                     <Description>This element describes a class of
         employees</Description>
                     <ResourceType>employee</ResourceType>
                     <Name>W1</Name>
              </Resource>
         ********************************Process Plan************************************

              <ProcessPlan>
                     <Identifier>ProcessPlan:Part1</Identifier>
                     <PartsProduced>
                         <Description>The part produced the process</Description>
                         <PartType>
                             <PartTypeIdentifier>Part1</PartTypeIdentifier>
                         </PartType>
                         <PartQuantity>1</PartQuantity>
                     </PartsProduced>
                     <PartsConsumed>
                         <Description>The part(s) consumed the
         process</Description>
                         <PartType>

       <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                         </PartType>
                         <PartQuantity>1</PartQuantity>
                     </PartsConsumed>
                     <FirstProcess>
                         <ProcessIdentifier>MainProcessSequence</ProcessIdentifier>
                     </FirstProcess>
                     <Process>
                         <Identifier>MainProcessSequence</Identifier>
                         <RepetitionCount>1</RepetitionCount>
                         <SubProcessGroup>
                             <Type>sequence</Type>
                             <Process>
                                 <ProcessIdentifier>A010</ProcessIdentifier>
                             </Process>
                             <Process>
                                 <ProcessIdentifier>A020</ProcessIdentifier>
                             </Process>
                             <Process>
                                 <ProcessIdentifier>A030</ProcessIdentifier>
                             </Process>
                             <Process>
                                 <ProcessIdentifier>A040</ProcessIdentifier>
                             </Process>
                             <Process>
                                 <ProcessIdentifier>A050</ProcessIdentifier>
                             </Process>
                         </SubProcessGroup>
                     </Process>
```

***********************************Process***************************************

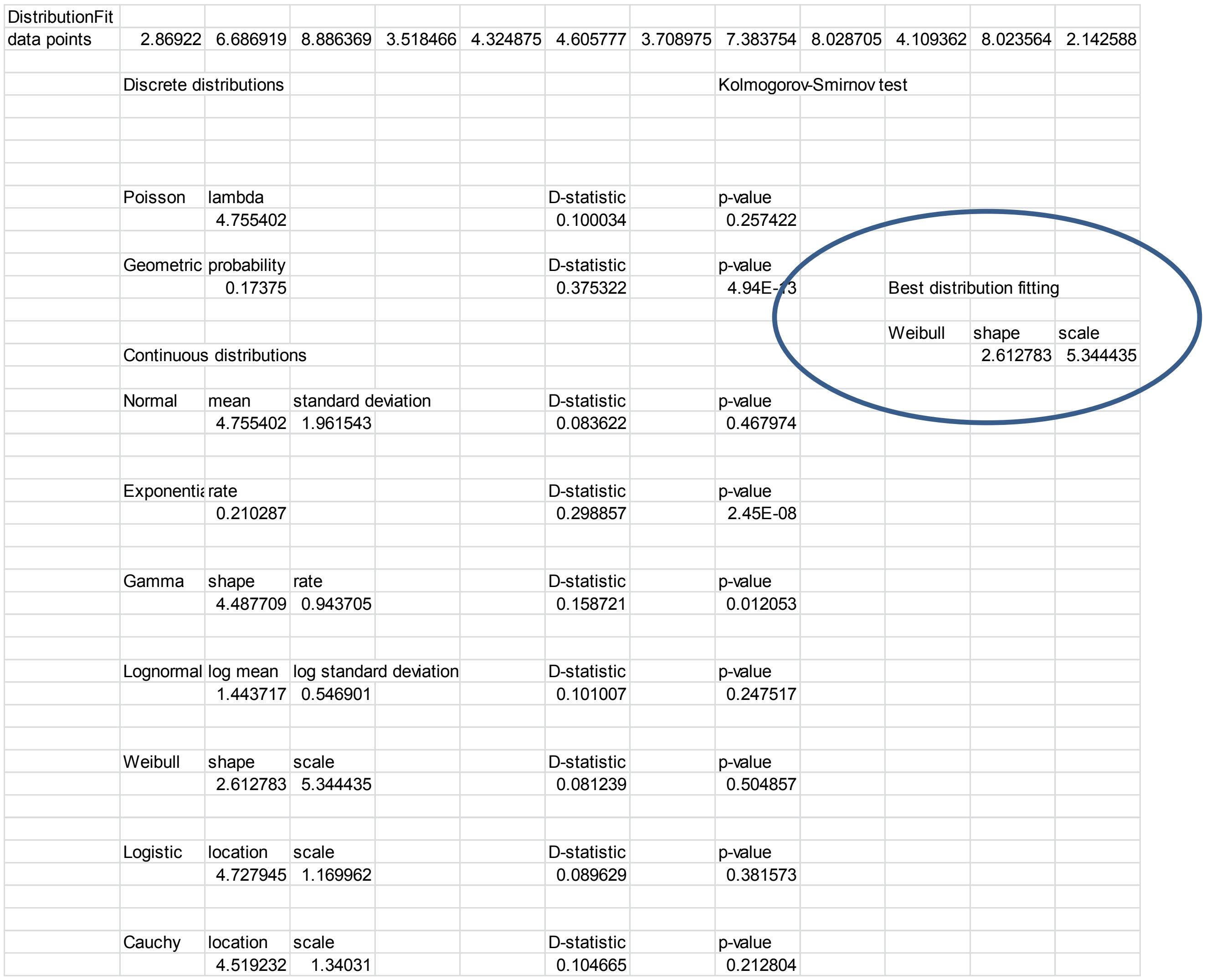
```
                    <Process>
                          <Identifier>A010</Identifier>
                          <Description>Process 1</Description>
                          <ResourcesRequired>
                                <Description>Source</Description>
                                <Resource>
                                      <ResourceIdentifier>S1</ResourceIdentifier>
                                </Resource>
                          </ResourcesRequired>
                          <Property>
                                <Name>interarrivalTime</Name>
                                <Unit>minutes</Unit>
                                <Distribution>
                                      <Name>Fixed</Name>
                                      <DistributionParameter>
                                          <Name>mean</Name>
                                            <Value>0.5</Value>
                                      </DistributionParameter>
                                </Distribution>
                          </Property>
                          <Property>
                                <Name>partType</Name>
                                <Value>Part</Value>
                          </Property>
                    </Process>

                    <Process>
                          <Identifier>A020</Identifier>
                          <Description>Process 2</Description>
                          <PartsProduced>
                                <Description>...</Description>
                                <PartType>

      <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                </PartType>
                                <PartQuantity>1</PartQuantity>
                          </PartsProduced>
                          <PartsConsumed>
                                <Description>...</Description>
                                <PartType>

      <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                </PartType>
                                <PartQuantity>1</PartQuantity>
                          </PartsConsumed>
                          <ResourcesRequired>
                                <Description>The employee performing the
operation.</Description>
                                <Resource>
                                      <ResourceIdentifier>A</ResourceIdentifier>
                                </Resource>
                          </ResourcesRequired>
                          <ResourcesRequired>
                    <Description>Machine1.</Description>
                    <Resource>
                        <ResourceIdentifier>M1</ResourceIdentifier>
                    </Resource>
                </ResourcesRequired>
                <OperationTime>
                    <Unit>minutes</Unit>
                    <Distribution>
                    <Name>Weibull</Name>
                        <DistributionParameterA>
                            <Name>shape</Name>
                            <Value>2.6127833842</Value>
                        </DistributionParameterA>
                        <DistributionParameterB>
                            <Name>scale</Name>
                            <Value>5.3444350739</Value>
                        </DistributionParameterB>
                    </Distribution>
                </OperationTime>
                <Property>
                    <Name>MeanTimeToFailure</Name>
                    <Unit>minutes</Unit>
                    <Distribution>
                        <Name>Fixed</Name>
                        <DistributionParameter>
                            <Name>mean</Name>
                            <Value>60</Value>
                        </DistributionParameter>
                    </Distribution>
                </Property>
                <Property>
                    <Name>MeanTimeToRepair</Name>
                    <Unit>minutes</Unit>
                    <Distribution>
                        <Name>Fixed</Name>
                        <DistributionParameter>
                            <Name>mean</Name>
                            <Value>5</Value>
                        </DistributionParameter>
                    </Distribution>
                    <Property>
                        <Name>RepairmanRequired</Name>
                        <ResourcesRequired>
                            <Description>The employee performing
the operation.</Description>

    <ResourceIdentifier>W1</ResourceIdentifier>
                        </ResourcesRequired>
                    </Property>
                </Property>
            </Process>

            <Process>
                <Identifier>A030</Identifier>
                <Description>Process 3</Description>
                <ResourcesRequired>
                    <Description>Queue1.</Description>
                    <Resource>
                        <ResourceIdentifier>Q1</ResourceIdentifier>
                    </Resource>
                </ResourcesRequired>
                <Property>
                    <Name>capacity</Name>
                    <Value>1</Value>
                </Property>
            </Process>

            <Process>
                <Identifier>A040</Identifier>
                <Description>Process 4</Description>
                <PartsProduced>
                    <Description>...</Description>
                    <PartType>

    <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                    </PartType>
                    <PartQuantity>1</PartQuantity>
                </PartsProduced>
                <PartsConsumed>
                    <Description>...</Description>
                    <PartType>

    <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                    </PartType>
                    <PartQuantity>1</PartQuantity>
                </PartsConsumed>
                <ResourcesRequired>
                    <Description>The employee performing the
operation.</Description>
                    <Resource>
                        <ResourceIdentifier>B</ResourceIdentifier>
                    </Resource>
                </ResourcesRequired>
                <ResourcesRequired>
                    <Description>Machine2.</Description>
                    <Resource>
                        <ResourceIdentifier>M2</ResourceIdentifier>
                    </Resource>
                </ResourcesRequired>
                <OperationTime>
                    <Unit>minutes</Unit>
                    <Distribution>
                    <Name>Normal</Name>
                        <DistributionParameterA>
                            <Name>stdev</Name>
                            <Value>0.110403961533</Value>
                        </DistributionParameterA>
                        <DistributionParameterB>
                            <Name>mean</Name>
                            <Value>0.678259396839</Value>
```

```xml
                        </DistributionParameterB>
                    </Distribution>
                </OperationTime>
                <Property>
                    <Name>MeanTimeToFailure</Name>
                    <Unit>minutes</Unit>
                    <Distribution>
                        <Name>Fixed</Name>
                        <DistributionParameter>
                            <Name>mean</Name>
                            <Value>40</Value>
                        </DistributionParameter>
                    </Distribution>
                </Property>
                <Property>
                    <Name>MeanTimeToRepair</Name>
                    <Unit>minutes</Unit>
                    <Distribution>
                        <Name>Fixed</Name>
                        <DistributionParameter>
                            <Name>mean</Name>
                            <Value>10</Value>
                        </DistributionParameter>
                    </Distribution>
                    <Property>
                        <Name>RepairmanRequired</Name>
                        <ResourcesRequired>
                            <Description>The employee performing
the operation.</Description>

    <ResourceIdentifier>W1</ResourceIdentifier>
                        </ResourcesRequired>
                    </Property>
                </Property>
            </Process>

            <Process>
                <Identifier>A050</Identifier>
                <Description>Process 5</Description>
                <ResourcesRequired>
                    <Description>Exit.</Description>
                    <Resource>
                        <ResourceIdentifier>E1</ResourceIdentifier>
                    </Resource>
                </ResourcesRequired>
            </Process>
        </ProcessPlan>
    </DataSection>
</CMSDDocument>
```

JSON file of the example

```
{
 "modelResource": [
  {
   "_class": "Dream.Repairman",
   "capacity": "1",
   "id": "W1",
   "name": "W1"
  }
 ],
 "_class": "Dream.Simulation",
 "coreObject": [
  {
   "name": "Raw Material",
   "entity": "Part",
   "interarrivalTime": {
    "distributionType": "Fixed",
    "mean": "0.5"
   },
   "successorList": [
    "DummyQ"
   ],
   "_class": "Dream.Source",
   "id": "S1"
  },
  {
   "processingTime": {
    "shape": 2.6127833842030075,
    "scale": 5.344435073902506,
    "distributionType": "Weibull"
   },
   "name": "Machine1",
   "predecessorList": [
    "DummyQ"
   ],
   "successorList": [
    "Q1"
   ],
   "failures": {
    "MTTR": "5",
    "failureDistribution": "Fixed",
    "repairman": "W1",
    "MTTF": "60"
   },
   "_class": "Dream.Machine",
   "id": "M1"
  },
  {
   "processingTime": {
    "stdev": 0.11040396153317669,
    "distributionType": "Normal",
    "mean": 0.6782593968389655
   },
   "name": "Machine2",
   "predecessorList": [
    "Q1"
   ],
   "successorList": [
    "E1"
   ],
   "failures": {
    "MTTR": "10",
    "failureDistribution": "Fixed",
    "repairman": "W1",
    "MTTF": "40"
   },
   "_class": "Dream.Machine",
   "id": "M2"
  },
  {
   "capacity": "1",
   "name": "DummyQ",
   "isDummy": "True",
   "predecessorList": [
    "S1"
   ],
   "successorList": [
    "M1"
   ],
   "_class": "Dream.Queue",
   "id": "DummyQ"
  },
  {
   "capacity": "1",
   "name": "Q1",
   "isDummy": "False",
   "predecessorList": [
    "M1"
   ],
   "successorList": [
    "M2"
   ],
   "_class": "Dream.Queue",
   "id": "Q1"
  },
  {
   "predecessorList": [
    "M2"
   ],
   "_class": "Dream.Exit",
   "id": "E1",
   "name": "Stock"
  }
 ],
 "general": {
  "maxSimTime": "1440",
  "_class": "Dream.Configuration",
  "numberOfReplications": "1",
  "trace": "Yes",
  "confidenceLevel": "0.95"
 }
}
```

Excel file showing the distribution fitting test in the processing times of Machine 1



DistributionFit												
data points	2.86922	6.686919	8.886369	3.518466	4.324875	4.605777	3.708975	7.383754	8.028705	4.109362	8.023564	2.142588
	Discrete distributions							Kolmogorov-Smirnov test				
	Poisson	lambda				D-statistic		p-value				
		4.755402				0.100034		0.257422				
	Geometric	probability				D-statistic		p-value				
		0.17375				0.375322		4.94E-13		Best distribution fitting		
										Weibull	shape	scale
	Continuous distributions										2.612783	5.344435
	Normal	mean	standard deviation			D-statistic		p-value				
		4.755402	1.961543			0.083622		0.467974				
	Exponentia	rate				D-statistic		p-value				
		0.210287				0.298857		2.45E-08				
	Gamma	shape	rate			D-statistic		p-value				
		4.487709	0.943705			0.158721		0.012053				
	Lognormal	log mean	log standard deviation			D-statistic		p-value				
		1.443717	0.546901			0.101007		0.247517				
	Weibull	shape	scale			D-statistic		p-value				
		2.612783	5.344435			0.081239		0.504857				
	Logistic	location	scale			D-statistic		p-value				
		4.727945	1.169962			0.089629		0.381573				
	Cauchy	location	scale			D-statistic		p-value				
		4.519232	1.34031			0.104665		0.212804				

Excel file showing the distribution fitting test in the processing times of Machine 2

DistributionFit												
data points	0.623414	0.618009	0.669085	0.714608	0.728286	0.572689	0.756631	0.633322	0.865086	0.322813	0.516356	0.594359
	Discrete distributions							Kolmogorov-Smirnov test				
	Poisson	lambda				D-statistic		p-value				
		0.678259				0.5075		4.44E-16				
	Geometric	probability				D-statistic		p-value				
		0.595855				0.595855		4.44E-16		Best distribution fitting		
										Normal	stdev	mean
	Continuous distributions										0.110404	0.678259
	Normal	mean	standard deviation			D-statistic		p-value				
		0.678259	0.110404			0.057582		0.882533				
	Exponentia	rate				D-statistic		p-value				
		1.474362				0.492123		4.44E-16				
	Gamma	shape	rate			D-statistic		p-value				
		34.3284	50.61251			1		4.44E-16				
	Lognormal	log mean	log standard deviation			D-statistic		p-value				
		-0.40286	0.176532			0.076091		0.594621				
	Weibull	shape	scale			D-statistic		p-value				
		7.112427	0.724066			0.07006		0.695245				
	Logistic	location	scale			D-statistic		p-value				
		0.680653	0.063103			0.076566		0.586776				
	Cauchy	location	scale			D-statistic		p-value				
		0.680082	0.072959			0.097241		0.292919				

- **Example Production line**

Below the CMSD and JSON output data formats are presented. The highlighted parts reveal as example the inputs of the KE tool in the processing times and scrap quantity of stations P6 and P10.

CMSD information model of the example

```
<CMSDDocument xmlns="urn:cmsd:main"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="urn:cmsd:main main.xsd">
<!--<CMSDDocument>-->
       <DataSection>
              <PartType>
                     <Identifier>Part1</Identifier>
              </PartType>
              <PartType>
                     <Identifier>UnfinishedPart1</Identifier>
              </PartType>
              <Resource>
                     <Identifier>resource1</Identifier>
                     <Description>This resource describes the first of two parallel
stations in Section PA</Description>
                     <ResourceType>station</ResourceType>
                     <Name>P1</Name>
              </Resource>
              <Resource>
                     <Identifier>resource2</Identifier>
                     <Description>This resource describes the second of two parallel
stations in Section PA</Description>
                     <ResourceType>station</ResourceType>
                     <Name>P4</Name>
              </Resource>
              <Resource>
                     <Identifier>resource3</Identifier>
                     <Description>This resource describes the first of two parallel
stations in Section PA</Description>
                     <ResourceType>station</ResourceType>
                     <Name>P2</Name>
              </Resource>
              <Resource>
                     <Identifier>resource4</Identifier>
                     <Description>This resource describes the second of two parallel
stations in Section PA</Description>
                     <ResourceType>station</ResourceType>
                     <Name>P5</Name>
              </Resource>
              <Resource>
                     <Identifier>resource5</Identifier>
                     <Description>This resource describes the first of two parallel
stations in Section PA</Description>
                     <ResourceType>station</ResourceType>
                     <Name>P3</Name>
              </Resource>
```

```
		<Resource>
			<Identifier>resource6</Identifier>
			<Description>This resource describes the second of two parallel 
stations in Section PA</Description>
			<ResourceType>station</ResourceType>
			<Name>P6</Name>
		</Resource>
		<Resource>
			<Identifier>resource7</Identifier>
			<Description>This resource describes the machine in Section 
PB</Description>
			<ResourceType>machine</ResourceType>
			<Name>P7</Name>
		</Resource>
		<Resource>
			<Identifier>resource8</Identifier>
			<Description>This resource describes the first of two parallel 
stations in Section PC</Description>
			<ResourceType>station</ResourceType>
			<Name>P8</Name>
		</Resource>
		<Resource>
			<Identifier>resource9</Identifier>
			<Description>This resource describes the second of two parallel 
stations in Section PC</Description>
			<ResourceType>station</ResourceType>
			<Name>P9</Name>
		</Resource>
		<Resource>
			<Identifier>resource10</Identifier>
			<Description>This resource describes the first of two parallel 
stations in Section PD</Description>
			<ResourceType>station</ResourceType>
			<Name>P10</Name>
		</Resource>
		<Resource>
			<Identifier>resource11</Identifier>
			<Description>This resource describes the second of two parallel 
stations in Section PD</Description>
			<ResourceType>station</ResourceType>
			<Name>P11</Name>
		</Resource>

		<Resource>
			<Identifier>A</Identifier>
			<ResourceType>employee</ResourceType>
		</Resource>
		<Resource>
			<Identifier>B</Identifier>
			<ResourceType>employee</ResourceType>
		</Resource>
		<Resource>
			<Identifier>C</Identifier>
			<ResourceType>employee</ResourceType>
		</Resource>
```

```
        <Resource>
            <Identifier>D</Identifier>
            <ResourceType>employee</ResourceType>
        </Resource>
        <Resource>
            <Identifier>E</Identifier>
            <ResourceType>employee</ResourceType>
        </Resource>
        <Resource>
            <Identifier>F</Identifier>
            <ResourceType>employee</ResourceType>
        </Resource>

        <ProcessPlan>
            <Identifier>ProcessPlan:PPPlan1</Identifier>
            <PartsProduced>
                <Description>The part produced the process</Description>
                <PartType>
                    <PartTypeIdentifier>Part1</PartTypeIdentifier>
                </PartType>
                <PartQuantity>1</PartQuantity>
            </PartsProduced>
            <PartsConsumed>
                <Description>The part(s) consumed the
process</Description>
                <PartType>

    <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                </PartType>
                <PartQuantity>1</PartQuantity>
            </PartsConsumed>
            <FirstProcess>
                <ProcessIdentifier>PFirst</ProcessIdentifier>
            </FirstProcess>

            <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>
```

```
<Process>
       <Identifier>PA</Identifier>
       <SubProcessGroup>
              <Type>decision</Type>
              <Process>
                     <ProcessIdentifier>PA1</ProcessIdentifier>
              </Process>
              <Process>
                     <ProcessIdentifier>PA2</ProcessIdentifier>
              </Process>
       </SubProcessGroup>
</Process>

<Process>
       <Identifier>PA1</Identifier>
       <SubProcessGroup>
              <Type>sequence</Type>
              <Process>
                     <ProcessIdentifier>P1</ProcessIdentifier>
              </Process>
              <Process>
                     <ProcessIdentifier>P2</ProcessIdentifier>
              </Process>
              <Process>
                     <ProcessIdentifier>P3</ProcessIdentifier>
              </Process>
       </SubProcessGroup>
</Process>

<Process>
       <Identifier>PA2</Identifier>
       <SubProcessGroup>
              <Type>sequence</Type>
              <Process>
                     <ProcessIdentifier>P4</ProcessIdentifier>
              </Process>
              <Process>
                     <ProcessIdentifier>P5</ProcessIdentifier>
              </Process>
              <Process>
                     <ProcessIdentifier>P6</ProcessIdentifier>
              </Process>
       </SubProcessGroup>
</Process>

<Process>
       <Identifier>PB</Identifier>
       <SubProcessGroup>
              <Type>sequence</Type>
              <Process>
                     <ProcessIdentifier>P7</ProcessIdentifier>
              </Process>
       </SubProcessGroup>
</Process>
```

```
                    <Process>
                          <Identifier>PC</Identifier>
                          <SubProcessGroup>
                                <Type>decision</Type>
                                <Process>
                                      <ProcessIdentifier>P8</ProcessIdentifier>
                                </Process>
                                <Process>
                                      <ProcessIdentifier>P9</ProcessIdentifier>
                                </Process>
                          </SubProcessGroup>
                    </Process>

                    <Process>
                          <Identifier>PD</Identifier>
                          <SubProcessGroup>
                                <Type>decision</Type>
                                <Process>
                                      <ProcessIdentifier>P10</ProcessIdentifier>
                                </Process>
                                <Process>
                                      <ProcessIdentifier>P11</ProcessIdentifier>
                                </Process>
                          </SubProcessGroup>
                    </Process>

                    <Process>
                          <Identifier>P1</Identifier>
                          <Description>P1</Description>
                          <PartsProduced>
                                <Description>...</Description>
                                <PartType>

      <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                </PartType>
                                <PartQuantity>1</PartQuantity>
                          </PartsProduced>
                          <PartsConsumed>
                                <Description>The part that is an input to this
process</Description>
                                <PartType>

      <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                </PartType>
                                <PartQuantity>1</PartQuantity>
                          </PartsConsumed>
                          <ResourcesRequired>
                                <Description>The employee performing the
operation.</Description>
                                <Resource>
                                      <ResourceIdentifier>A</ResourceIdentifier>
                                </Resource>
                          </ResourcesRequired>
                          <ResourcesRequired>
                    <Description>The resource where the operation is
being performed.</Description>
                    <Resource>
    
    <ResourceIdentifier>resource1</ResourceIdentifier>
                    </Resource>
                </ResourcesRequired>
                <OperationTime>
                    <Unit>minute</Unit>
                    <Distribution>
                        <Name>Gamma</Name>
                        <DistributionParameter>
                            <Name>shape</Name>
                            <Value>29.4228251359</Value>
                        </DistributionParameter>
                        <DistributionParameter>
                            <Name>rate</Name>
                            <Value>94.5230276629</Value>
                        </DistributionParameter>
                    </Distribution>
                </OperationTime>
                <Property>
                    <Name>ScrapQuantity</Name>
                    <Name>mean</Name>
                    <Value>0.0</Value>
                </Property>
            </Process>

            <Process>
                <Identifier>P2</Identifier>
                <Description>P2</Description>
                <PartsProduced>
                    <Description>...</Description>
                    <PartType>
    
    <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                    </PartType>
                    <PartQuantity>1</PartQuantity>
                </PartsProduced>
                <PartsConsumed>
                    <Description>The part that is an input to this
process</Description>
                    <PartType>
    
    <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                    </PartType>
                    <PartQuantity>1</PartQuantity>
                </PartsConsumed>
                <ResourcesRequired>
                    <Description>The employee performing the
operation.</Description>
                    <Resource>
                        <ResourceIdentifier>B</ResourceIdentifier>
                    </Resource>
                </ResourcesRequired>
```

```
                        <ResourcesRequired>
                                <Description>The resource where the operation is
being performed.</Description>
                                <Resource>

        <ResourceIdentifier>resource3</ResourceIdentifier>
                                </Resource>
                        </ResourcesRequired>
                        <OperationTime>
                                <Unit>minute</Unit>
                                <Distribution>
                                        <Name>Gamma</Name>
                                        <DistributionParameter>
                                                <Name>shape</Name>
                                                <Value>1225.91219908</Value>
                                        </DistributionParameter>
                                        <DistributionParameter>
                                                <Name>rate</Name>
                                                <Value>6153.93113526</Value>
                                        </DistributionParameter>
                                </Distribution>
                        </OperationTime>
                        <Property>
                                <Name>ScrapQuantity</Name>
                                <Name>mean</Name>
                                <Value>2.0</Value>
                        </Property>
                </Process>

                <Process>
                        <Identifier>P3</Identifier>
                        <Description>P3</Description>
                        <PartsProduced>
                                <Description>...</Description>
                                <PartType>

        <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                </PartType>
                                <PartQuantity>1</PartQuantity>
                        </PartsProduced>
                        <PartsConsumed>
                                <Description>The part that is an input to this
process</Description>
                                <PartType>

        <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                </PartType>
                                <PartQuantity>1</PartQuantity>
                        </PartsConsumed>
                        <ResourcesRequired>
                                <Description>The employee performing the
operation.</Description>
                                <Resource>
                                        <ResourceIdentifier>C</ResourceIdentifier>
                                </Resource>
                        </ResourcesRequired>
                        <ResourcesRequired>
                              <Description>The resource where the operation is
being performed.</Description>
                              <Resource>

      <ResourceIdentifier>resource5</ResourceIdentifier>
                              </Resource>
                        </ResourcesRequired>
                        <OperationTime>
                              <Unit>minute</Unit>
                              <Distribution>
                                    <Name>Normal</Name>
                                    <DistributionParameter>
                                          <Name>mean</Name>
                                          <Value>0.402962037037</Value>
                                    </DistributionParameter>
                                    <DistributionParameter>
                                          <Name>stdev</Name>
                                          <Value>0.0960106899098</Value>
                                    </DistributionParameter>
                              </Distribution>
                        </OperationTime>
                        <Property>
                              <Name>ScrapQuantity</Name>
                              <Name>mean</Name>
                              <Value>0.0</Value>
                        </Property>
                  </Process>

                  <Process>
                        <Identifier>P4</Identifier>
                        <Description>P4</Description>
                        <PartsProduced>
                              <Description>...</Description>
                              <PartType>

      <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                              </PartType>
                              <PartQuantity>1</PartQuantity>
                        </PartsProduced>
                        <PartsConsumed>
                              <Description>The part that is an input to this
process</Description>
                              <PartType>

      <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                              </PartType>
                              <PartQuantity>1</PartQuantity>
                        </PartsConsumed>
                        <ResourcesRequired>
                              <Description>The employee performing the
operation.</Description>
                              <Resource>
                                    <ResourceIdentifier>A</ResourceIdentifier>
```

```
                    </Resource>
                </ResourcesRequired>
                <ResourcesRequired>
                    <Description>The resource where the operation is
being performed.</Description>
                    <Resource>

    <ResourceIdentifier>resource2</ResourceIdentifier>
                    </Resource>
                </ResourcesRequired>
                <OperationTime>
                    <Unit>minute</Unit>
                    <Distribution>
                        <Name>Normal</Name>
                        <DistributionParameter>
                            <Name>mean</Name>
                            <Value>0.290917148148</Value>
                        </DistributionParameter>
                        <DistributionParameter>
                            <Name>stdev</Name>
                            <Value>0.0476508792381</Value>
                        </DistributionParameter>
                    </Distribution>
                </OperationTime>
                <Property>
                    <Name>ScrapQuantity</Name>
                    <Name>mean</Name>
                    <Value>1.0</Value>
                </Property>
            </Process>


            <Process>
                <Identifier>P5</Identifier>
                <Description>P5</Description>
                <PartsProduced>
                    <Description>...</Description>
                    <PartType>

    <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                    </PartType>
                    <PartQuantity>1</PartQuantity>
                </PartsProduced>
                <PartsConsumed>
                    <Description>The part that is an input to this
process</Description>
                    <PartType>

    <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                    </PartType>
                    <PartQuantity>1</PartQuantity>
                </PartsConsumed>
                <ResourcesRequired>
                    <Description>The employee performing the
operation.</Description>
```

```
                    <Resource>
                        <ResourceIdentifier>B</ResourceIdentifier>
                    </Resource>
                </ResourcesRequired>
                <ResourcesRequired>
                    <Description>The resource where the operation is
being performed.</Description>
                    <Resource>

    <ResourceIdentifier>resource4</ResourceIdentifier>
                    </Resource>
                </ResourcesRequired>
                <OperationTime>
                    <Unit>minute</Unit>
                    <Distribution>
                        <Name>Gamma</Name>
                        <DistributionParameter>
                            <Name>shape</Name>
                            <Value>47.655</Value>
                        </DistributionParameter>
                        <DistributionParameter>
                            <Name>rate</Name>
                            <Value>95.94111769</Value>
                        </DistributionParameter>
                    </Distribution>
                </OperationTime>
                <Property>
                    <Name>ScrapQuantity</Name>
                    <Name>mean</Name>
                    <Value>1.0</Value>
                </Property>
            </Process>

            <Process>
                <Identifier>P6</Identifier>
                <Description>P6</Description>
                <PartsProduced>
                    <Description>...</Description>
                    <PartType>

    <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                    </PartType>
                    <PartQuantity>1</PartQuantity>
                </PartsProduced>
                <PartsConsumed>
                    <Description>The part that is an input to this
process</Description>
                    <PartType>

    <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                    </PartType>
                    <PartQuantity>1</PartQuantity>
                </PartsConsumed>
                <ResourcesRequired>
```

```
                                <Description>The employee performing the
operation.</Description>
                                <Resource>
                                        <ResourceIdentifier>C</ResourceIdentifier>
                                </Resource>
                        </ResourcesRequired>
                        <ResourcesRequired>
                                <Description>The resource where the operation is
being performed.</Description>
                                <Resource>

        <ResourceIdentifier>resource6</ResourceIdentifier>
                                </Resource>
                        </ResourcesRequired>
                        <OperationTime>
                                <Unit>minute</Unit>
                                <Distribution>
                                        <Name>Normal</Name>
                                        <DistributionParameter>
                                                <Name>mean</Name>
                                                <Value>0.588020851852</Value>
                                        </DistributionParameter>
                                        <DistributionParameter>
                                                <Name>stdev</Name>
                                                <Value>0.107534393532</Value>
                                        </DistributionParameter>
                                </Distribution>
                        </OperationTime>
                        <Property>
                                <Name>ScrapQuantity</Name>
                                <Name>mean</Name>
                                <Value>1.0</Value>
                        </Property>
                </Process>

                <Process>
                        <Identifier>P7</Identifier>
                        <Description>P7</Description>
                        <PartsProduced>
                                <Description>...</Description>
                                <PartType>

        <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                </PartType>
                                <PartQuantity>1</PartQuantity>
                        </PartsProduced>
                        <PartsConsumed>
                                <Description>The part that is an input to this
process</Description>
                                <PartType>

        <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                </PartType>
                                <PartQuantity>1</PartQuantity>
                        </PartsConsumed>
                        <ResourcesRequired>
                              <Description>The employee performing the
operation.</Description>
                              <Resource>
                                    <ResourceIdentifier>D</ResourceIdentifier>
                              </Resource>
                        </ResourcesRequired>
                        <ResourcesRequired>
                              <Description>The resource where the operation is
being performed.</Description>
                              <Resource>

      <ResourceIdentifier>resource7</ResourceIdentifier>
                              </Resource>
                        </ResourcesRequired>
                        <OperationTime>
                              <Unit>minute</Unit>
                              <Distribution>
                                    <Name>Normal</Name>
                                    <DistributionParameter>
                                          <Name>mean</Name>
                                          <Value>1.10071188889</Value>
                                    </DistributionParameter>
                                    <DistributionParameter>
                                          <Name>stdev</Name>
                                          <Value>0.154492557514</Value>
                                    </DistributionParameter>
                              </Distribution>
                        </OperationTime>
                        <Property>
                              <Name>ScrapQuantity</Name>
                              <Name>mean</Name>
                              <Value>2.0</Value>
                        </Property>
                  </Process>

                  <Process>
                        <Identifier>P8</Identifier>
                        <Description>P8</Description>
                        <PartsProduced>
                              <Description>...</Description>
                              <PartType>

      <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                              </PartType>
                              <PartQuantity>1</PartQuantity>
                        </PartsProduced>
                        <PartsConsumed>
                              <Description>The part that is an input to this
process</Description>
                              <PartType>

      <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                              </PartType>
                              <PartQuantity>1</PartQuantity>
```

```
                            </PartsConsumed>
                            <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>resource8</ResourceIdentifier>
                                    </Resource>
                            </ResourcesRequired>
                            <OperationTime>
                                    <Unit>minute</Unit>
                                    <Distribution>
                                            <Name>Gamma</Name>
                                            <DistributionParameter>
                                                    <Name>shape</Name>
                                                    <Value>55.0859511268</Value>
                                            </DistributionParameter>
                                            <DistributionParameter>
                                                    <Name>rate</Name>
                                                    <Value>56.1808214867</Value>
                                            </DistributionParameter>
                                    </Distribution>
                            </OperationTime>
                            <Property>
                                    <Name>ScrapQuantity</Name>
                                    <Name>mean</Name>
                                    <Value>1.0</Value>
                            </Property>
                    </Process>


                    <Process>
                            <Identifier>P9</Identifier>
                            <Description>P9</Description>
                            <PartsProduced>
                                    <Description>...</Description>
                                    <PartType>

        <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                    </PartType>
                                    <PartQuantity>1</PartQuantity>
                            </PartsProduced>
                            <PartsConsumed>
                                    <Description>The part that is an input to this
process</Description>
                                    <PartType>

        <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
```

```
                        </PartType>
                        <PartQuantity>1</PartQuantity>
                  </PartsConsumed>
                  <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>stdev</Name>
                                    <Value>123.27</Value>
                              </DistributionParameter>
                        </Distribution>
                  </OperationTime>
                  <Property>
                        <Name>ScrapQuantity</Name>
                        <Name>mean</Name>
                        <Value>2.0</Value>
                  </Property>
            </Process>


            <Process>
                  <Identifier>P10</Identifier>
                  <Description>P10</Description>
                  <PartsProduced>
                        <Description>...</Description>
                        <PartType>

      <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                        </PartType>
                        <PartQuantity>1</PartQuantity>
                  </PartsProduced>
                  <PartsConsumed>
                        <Description>The part that is an input to this
process</Description>
                        <PartType>
```

```
       <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                 </PartType>
                                 <PartQuantity>1</PartQuantity>
                           </PartsConsumed>
                           <ResourcesRequired>
                                 <Description>The employee performing the
operation.</Description>
                                 <Resource>
                                       <ResourceIdentifier>F</ResourceIdentifier>
                                 </Resource>
                           </ResourcesRequired>
                           <ResourcesRequired>
                                 <Description>The resource where the operation is
being performed.</Description>
                                 <Resource>

       <ResourceIdentifier>resource10</ResourceIdentifier>
                                 </Resource>
                           </ResourcesRequired>
                           <OperationTime>
                                 <Unit>minute</Unit>
                                 <Distribution>
                                       <Name>Normal</Name>
                                       <DistributionParameter>
                                             <Name>mean</Name>
                                             <Value>0.834371018519</Value>
                                       </DistributionParameter>
                                       <DistributionParameter>
                                             <Name>stdev</Name>
                                             <Value>0.216771082413</Value>
                                       </DistributionParameter>
                                 </Distribution>
                           </OperationTime>
                           <Property>
                                 <Name>ScrapQuantity</Name>
                                 <Name>mean</Name>
                                 <Value>1.0</Value>
                           </Property>
                     </Process>

                     <Process>
                           <Identifier>P11</Identifier>
                           <Description>P11</Description>
                           <PartsProduced>
                                 <Description>...</Description>
                                 <PartType>

       <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                 </PartType>
                                 <PartQuantity>1</PartQuantity>
                           </PartsProduced>
                           <PartsConsumed>
                                 <Description>The part that is an input to this
process</Description>
```

```
                                <PartType>

        <PartTypeIdentifier>UnfinishedPart1</PartTypeIdentifier>
                                </PartType>
                                <PartQuantity>1</PartQuantity>
                        </PartsConsumed>
                        <ResourcesRequired>
                                <Description>The employee performing the
operation.</Description>
                                <Resource>
                                        <ResourceIdentifier>F</ResourceIdentifier>
                                </Resource>
                        </ResourcesRequired>
                        <ResourcesRequired>
                                <Description>The resource where the operation is
being performed.</Description>
                                <Resource>

        <ResourceIdentifier>resource11</ResourceIdentifier>
                                </Resource>
                        </ResourcesRequired>
                        <OperationTime>
                                <Unit>minute</Unit>
                                <Distribution>
                                        <Name>Normal</Name>
                                        <DistributionParameter>
                                                <Name>mean</Name>
                                                <Value>0.887490333333</Value>
                                        </DistributionParameter>
                                        <DistributionParameter>
                                                <Name>stdev</Name>
                                                <Value>0.12872179624</Value>
                                        </DistributionParameter>
                                </Distribution>
                        </OperationTime>
                        <Property>
                                <Name>ScrapQuantity</Name>
                                <Name>mean</Name>
                                <Value>1.0</Value>
                        </Property>
                </Process>
                </ProcessPlan>
        </DataSection>
</ CMSDDocument >
```

JSON file of the example

```
{
 "nodes": {
  "Q2A": {
   "top": 0.7727272727272727,
   "_class": "Dream.LineClearance",
   "capacity": "2",
   "name": "Q2A",
   "left": 0.6968085106382979
  },
  "Q2B": {
   "top": 0.7727272727272727,
   "_class": "Dream.LineClearance",
   "capacity": "2",
   "name": "Q2B",
   "left": 0.6968085106382979
  },
  "Q3B": {
   "top": 0.7727272727272727,
   "_class": "Dream.LineClearance",
   "capacity": "2",
   "name": "Q3B",
   "left": 0.6968085106382979
  },
  "S1": {
   "name": "Source",
   "top": 0.9545454545454546,
   "entity": "Dream.Batch",
   "interarrivalTime": {
    "distributionType": "Fixed",
    "mean": 0.5
   },
   "batchNumberOfUnits": 100,
   "_class": "Dream.BatchSource",
   "left": 0.6968085106382979
  },
  "QPa": {
   "top": 0.7727272727272727,
   "_class": "Dream.Queue",
   "capacity": "3",
   "name": "QPa",
   "left": 0.6968085106382979
  },
  "QPr": {
   "top": 0.7727272727272727,
   "_class": "Dream.Queue",
   "capacity": "3",
   "name": "QPr",
   "left": 0.6968085106382979
  },
  "P6": {
   "name": "P6",
   "top": 0.5909090909090908,
   "scrapQuantity": 1.0,
 "processingTime": {
  "max": 0.9106240324475632,
  "stdev": 0.10753439353190378,
  "min": 0,
  "distributionType": "Normal",
  "mean": 0.5880208518518518
 },
 "failures": {},
 "_class": "Dream.BatchScrapMachine",
 "left": 0.4414893617021277
},
"QStart": {
 "top": 0.7727272727272727,
 "_class": "Dream.Queue",
 "capacity": "1",
 "name": "StartQueue",
 "left": 0.6968085106382979
},
"P5": {
 "name": "P5",
 "top": 0.5909090909090908,
 "scrapQuantity": 1.0,
 "processingTime": {
  "shape": 47.655,
  "rate": 95.94111768995835,
  "distributionType": "Gamma"
 },
 "failures": {},
 "_class": "Dream.BatchScrapMachine",
 "left": 0.4414893617021277
},
"P10": {
 "name": "P10",
 "top": 0.5909090909090908,
 "scrapQuantity": 1.0,
 "processingTime": {
  "max": 1.4846842657580035,
  "stdev": 0.21677108241316168,
  "min": 0,
  "distributionType": "Normal",
  "mean": 0.8343710185185185
 },
 "failures": {},
 "_class": "Dream.BatchScrapMachine",
 "left": 0.4414893617021277
},
"P11": {
 "name": "P11",
 "top": 0.5909090909090908,
 "scrapQuantity": 1.0,
 "processingTime": {
  "max": 1.2736557220538487,
  "stdev": 0.12872179624017177,
  "min": 0,
  "distributionType": "Normal",
   "mean": 0.8874903333333334
  },
  "failures": {},
  "_class": "Dream.BatchScrapMachine",
  "left": 0.4414893617021277
 },
 "E1": {
  "top": 0.045454545454545414,
  "_class": "Dream.Exit",
  "name": "Stock",
  "left": 0.2978723404255319
 },
 "P2": {
  "name": "P2",
  "top": 0.5909090909090908,
  "scrapQuantity": 2.0,
  "processingTime": {
   "shape": 1225.9121990768035,
   "rate": 6153.9311352581,
   "distributionType": "Gamma"
  },
  "failures": {},
  "_class": "Dream.BatchScrapMachine",
  "left": 0.4414893617021277
 },
 "P3": {
  "name": "P3",
  "top": 0.5909090909090908,
  "scrapQuantity": 0.0,
  "processingTime": {
   "max": 0.690994106766528,
   "stdev": 0.09601068990983029,
   "min": 0,
   "distributionType": "Normal",
   "mean": 0.40296203703703704
  },
  "failures": {},
  "_class": "Dream.BatchScrapMachine",
  "left": 0.4414893617021277
 },
 "Q3A": {
  "top": 0.7727272727272727,
  "_class": "Dream.LineClearance",
  "capacity": "2",
  "name": "Q3A",
  "left": 0.6968085106382979
 },
 "P1": {
  "name": "P1",
  "top": 0.5909090909090908,
  "scrapQuantity": 0.0,
  "processingTime": {
   "shape": 29.422825135928104,
   "rate": 94.52302766286348,
   "distributionType": "Gamma"
  },
  "failures": {},
  "_class": "Dream.BatchScrapMachine",
  "left": 0.4414893617021277
 },
 "BDA": {
  "name": "Batch_DecompositionA",
  "top": 0.5909090909090908,
  "processingTime": {
   "distributionType": "Fixed",
   "mean": "0"
  },
  "numberOfSubBatches": 4,
  "_class": "Dream.BatchDecomposition",
  "left": 0.4414893617021277
 },
 "P7": {
  "name": "P7",
  "top": 0.5909090909090908,
  "scrapQuantity": 2.0,
  "processingTime": {
   "max": 1.5641895614304642,
   "stdev": 0.15449255751385838,
   "min": 0,
   "distributionType": "Normal",
   "mean": 1.100711888888889
  },
  "failures": {},
  "_class": "Dream.BatchScrapMachine",
  "left": 0.4414893617021277
 },
 "P4": {
  "name": "P4",
  "top": 0.5909090909090908,
  "scrapQuantity": 1.0,
  "processingTime": {
   "max": 0.4338697858624245,
   "stdev": 0.0476508792380921,
   "min": 0,
   "distributionType": "Normal",
   "mean": 0.2909171481481482
  },
  "failures": {},
  "_class": "Dream.BatchScrapMachine",
  "left": 0.4414893617021277
 },
 "BDB": {
  "name": "Batch_DecompositionB",
  "top": 0.5909090909090908,
  "processingTime": {
   "distributionType": "Fixed",
   "mean": "0"
  },
  "numberOfSubBatches": 4,
  "_class": "Dream.BatchDecomposition",
 "left": 0.4414893617021277
},
"P8": {
 "name": "P8",
 "top": 0.5909090909090908,
 "scrapQuantity": 1.0,
 "processingTime": {
  "shape": 55.08595112679275,
  "rate": 56.18082148673797,
  "distributionType": "Gamma"
 },
 "failures": {},
 "_class": "Dream.BatchScrapMachine",
 "left": 0.4414893617021277
},
"P9": {
 "name": "P9",
 "top": 0.5909090909090908,
 "scrapQuantity": 2.0,
 "processingTime": {
  "shape": 36.13,
  "rate": 123.27,
  "distributionType": "Gamma"
 },
 "failures": {},
 "_class": "Dream.BatchScrapMachine",
 "left": 0.4414893617021277
},
"QM": {
 "top": 0.7727272727272727,
 "_class": "Dream.Queue",
 "capacity": "3",
 "name": "QM",
 "left": 0.6968085106382979
},
"BRB": {
 "name": "Batch_ReassemblyB",
 "top": 0.5909090909090908,
 "processingTime": {
  "distributionType": "Fixed",
  "mean": "0"
 },
 "numberOfSubBatches": 4,
 "_class": "Dream.BatchReassembly",
 "left": 0.4414893617021277
},
"BRA": {
 "name": "Batch_ReassemblyA",
 "top": 0.5909090909090908,
 "processingTime": {
  "distributionType": "Fixed",
  "mean": "0"
 },
 "numberOfSubBatches": 4,
 "_class": "Dream.BatchReassembly",
  "left": 0.4414893617021277
 }
},
"_class": "Dream.Simulation",
"edges": {
 "24": [
  "QPa",
  "P11",
  {}
 ],
 "25": [
  "P10",
  "E1",
  {}
 ],
 "26": [
  "P11",
  "E1",
  {}
 ],
 "20": [
  "QPr",
  "P9",
  {}
 ],
 "21": [
  "P8",
  "QPa",
  {}
 ],
 "22": [
  "P9",
  "QPa",
  {}
 ],
 "23": [
  "QPa",
  "P10",
  {}
 ],
 "1": [
  "QStart",
  "BDA",
  {}
 ],
 "0": [
  "S1",
  "QStart",
  {}
 ],
 "3": [
  "BDA",
  "P1",
  {}
 ],
"2": [
 "QStart",
 "BDB",
 {}
],
"5": [
 "Q2A",
 "P2",
 {}
],
"4": [
 "P1",
 "Q2A",
 {}
],
"7": [
 "Q3A",
 "P3",
 {}
],
"6": [
 "P2",
 "Q3A",
 {}
],
"9": [
 "BRA",
 "QM",
 {}
],
"8": [
 "P3",
 "BRA",
 {}
],
"11": [
 "P4",
 "Q2B",
 {}
],
"10": [
 "BDB",
 "P4",
 {}
],
"13": [
 "P5",
 "Q3B",
 {}
],
"12": [
 "Q2B",
 "P5",
 {}
],
  "15": [
   "P5",
   "BRB",
   {}
  ],
  "14": [
   "Q3B",
   "P6",
   {}
  ],
  "17": [
   "QM",
   "P7",
   {}
  ],
  "16": [
   "BRB",
   "QM",
   {}
  ],
  "19": [
   "QPr",
   "P8",
   {}
  ],
  "18": [
   "P7",
   "QPr",
   {}
  ]
 },
 "general": {
  "trace": "No",
  "_class": "Dream.Configuration",
  "confidenceLevel": "0.95",
  "maxSimTime": "1440",
  "numberOfReplications": "10"
 }
```

<u>ManPy JSON file (simulation results)</u>

```
{
 "elementList": [
  {
   "_class": "Dream.LineClearance",
   "id": "Q2A"
  },
  {
   "_class": "Dream.LineClearance",
   "id": "Q2B"
  },
  {
   "_class": "Dream.Machine",
   "id": "P3",
  "results": {
   "working_ratio": {
    "avg": 99.1232932718127,
    "lb": 99.06288622969795,
    "ub": 99.18370031392745
   },
   "blockage_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "waiting_ratio": {
    "avg": 0.8767067281873027,
    "lb": 0.8162996860725587,
    "ub": 0.9371137703020467
   },
   "off_shift_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "setup_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "failure_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "loading_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   }
  }
 },
 {
  "_class": "Dream.Queue",
  "id": "QPa"
 },
 {
  "_class": "Dream.Queue",
  "id": "QPr"
 },
 {
  "_class": "Dream.Queue",
  "id": "QStart"
 },
 {
  "_class": "Dream.Machine",
  "id": "P5",
  "results": {
   "working_ratio": {
        "avg": 3.8484863685556765,
        "lb": 3.835591640224164,
        "ub": 3.861381096887189
       },
       "blockage_ratio": {
        "avg": 94.03753457294879,
        "lb": 93.89318733982336,
        "ub": 94.18188180607422
       },
       "waiting_ratio": {
        "avg": 2.11397905849554,
        "lb": 1.9687688716941143,
        "ub": 2.259189245296966
       },
       "off_shift_ratio": {
        "avg": 0.0,
        "lb": 0.0,
        "ub": 0.0
       },
       "setup_ratio": {
        "avg": 0.0,
        "lb": 0.0,
        "ub": 0.0
       },
       "failure_ratio": {
        "avg": 0.0,
        "lb": 0.0,
        "ub": 0.0
       },
       "loading_ratio": {
        "avg": 0.0,
        "lb": 0.0,
        "ub": 0.0
       }
      }
     },
     {
      "_class": "Dream.Machine",
      "id": "P10",
      "results": {
       "working_ratio": {
        "avg": 33.8643006065708,
        "lb": 33.251018516779276,
        "ub": 34.477582696362326
       },
       "blockage_ratio": {
        "avg": 0.0,
        "lb": 0.0,
        "ub": 0.0
       },
       "waiting_ratio": {
        "avg": 66.13569939342919,
        "lb": 65.52241730363767,
        "ub": 66.74898148322072
       },
   "off_shift_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "setup_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "failure_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "loading_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   }
  }
 },
 {
  "_class": "Dream.Machine",
  "id": "P11",
  "results": {
   "working_ratio": {
    "avg": 32.80895290519304,
    "lb": 32.20957202613055,
    "ub": 33.408333784255525
   },
   "blockage_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "waiting_ratio": {
    "avg": 67.19104709480696,
    "lb": 66.59166621574448,
    "ub": 67.79042797386944
   },
   "off_shift_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "setup_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "failure_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "loading_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   }
  }
 },
 {
  "_class": "Dream.Exit",
  "id": "E1",
  "results": {
   "unitsThroughput": [
    3040.0,
    3135.0,
    3230.0,
    2945.0,
    3040.0,
    3040.0,
    3135.0,
    3135.0,
    3040.0,
    3135.0
   ],
   "throughput": [
    32,
    33,
    34,
    31,
    32,
    32,
    33,
    33,
    32,
    33
   ],
   "takt_time": [
    43.69950037158746,
    43.26117960529305,
    42.166933591369705,
    45.14615867326605,
    44.95555252303127,
    44.785846788046754,
    43.56360076413797,
    42.477765687359685,
    44.60040914035664,
    43.13267566550554
   ],
   "lifespan": [
    761.7374958812431,
    756.5203248846857,
    769.0966311650603,
    768.6464696934038,
    801.2514402023118,
    764.5391421235206,
    792.4484177273526,
    771.6289613968129,
    769.6977448551118,
    771.9918783156244
   ]
  }
 },
 {
  "_class": "Dream.Machine",
  "id": "P2",
  "results": {
   "working_ratio": {
    "avg": 50.102895148547525,
    "lb": 49.24749868283626,
    "ub": 50.95829161425879
   },
   "blockage_ratio": {
    "avg": 48.236644358616054,
    "lb": 47.201622043610996,
    "ub": 49.27166667362111
   },
   "waiting_ratio": {
    "avg": 1.6604604928364235,
    "lb": 1.159532722362285,
    "ub": 2.161388263310562
   },
   "off_shift_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "setup_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "failure_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "loading_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   }
  }
 },
 {
  "_class": "Dream.LineClearance",
  "id": "Q3B"
 },
 {
  "_class": "Dream.LineClearance",
  "id": "Q3A"
```

```
},
{
 "_class": "Dream.Machine",
 "id": "P1",
 "results": {
  "working_ratio": {
   "avg": 73.78981870841058,
   "lb": 72.44696726539648,
   "ub": 75.13267015142468
  },
  "blockage_ratio": {
   "avg": 26.175459069367207,
   "lb": 24.83260762635311,
   "ub": 27.518310512381305
  },
  "waiting_ratio": {
   "avg": 0.034722222222218276,
   "lb": 0.034722222222192324,
   "ub": 0.03472222222224423
  },
  "off_shift_ratio": {
   "avg": 0.0,
   "lb": 0.0,
   "ub": 0.0
  },
  "setup_ratio": {
   "avg": 0.0,
   "lb": 0.0,
   "ub": 0.0
  },
  "failure_ratio": {
   "avg": 0.0,
   "lb": 0.0,
   "ub": 0.0
  },
  "loading_ratio": {
   "avg": 0.0,
   "lb": 0.0,
   "ub": 0.0
  }
 }
},
{
 "_class": "Dream.Machine",
 "id": "P6",
 "results": {
  "working_ratio": {
   "avg": 0.7151499038589795,
   "lb": 0.5631193514563365,
   "ub": 0.8671804562616224
  },
  "blockage_ratio": {
   "avg": 96.37469764512025,
   "lb": 96.17598423856082,
   "ub": 96.57341105167968
```

```
    },
    "waiting_ratio": {
     "avg": 2.9101524510207653,
     "lb": 2.783623796389208,
     "ub": 3.0366811056523226
    },
    "off_shift_ratio": {
     "avg": 0.0,
     "lb": 0.0,
     "ub": 0.0
    },
    "setup_ratio": {
     "avg": 0.0,
     "lb": 0.0,
     "ub": 0.0
    },
    "failure_ratio": {
     "avg": 0.0,
     "lb": 0.0,
     "ub": 0.0
    },
    "loading_ratio": {
     "avg": 0.0,
     "lb": 0.0,
     "ub": 0.0
    }
   }
  },
  {
   "_class": "Dream.Machine",
   "id": "P7",
   "results": {
    "working_ratio": {
     "avg": 93.16310446225327,
     "lb": 90.46881987070287,
     "ub": 95.85738905380367
    },
    "blockage_ratio": {
     "avg": 0.0,
     "lb": 0.0,
     "ub": 0.0
    },
    "waiting_ratio": {
     "avg": 6.836895537746727,
     "lb": 4.142610946196325,
     "ub": 9.53118012929713
    },
    "off_shift_ratio": {
     "avg": 0.0,
     "lb": 0.0,
     "ub": 0.0
    },
    "setup_ratio": {
     "avg": 0.0,
     "lb": 0.0,
    "ub": 0.0
   },
   "failure_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "loading_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   }
  }
 },
 {
  "_class": "Dream.Machine",
  "id": "P4",
  "results": {
   "working_ratio": {
    "avg": 6.666597647254015,
    "lb": 6.468351865246149,
    "ub": 6.864843429261882
   },
   "blockage_ratio": {
    "avg": 93.33340235274599,
    "lb": 93.13515657073812,
    "ub": 93.53164813475387
   },
   "waiting_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "off_shift_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "setup_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "failure_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   },
   "loading_ratio": {
    "avg": 0.0,
    "lb": 0.0,
    "ub": 0.0
   }
  }
 },
{
 "_class": "Dream.Machine",
 "id": "P8",
 "results": {
  "working_ratio": {
   "avg": 35.81052909393301,
   "lb": 34.58078555801364,
   "ub": 37.04027262985239
  },
  "blockage_ratio": {
   "avg": 0.0,
   "lb": 0.0,
   "ub": 0.0
  },
  "waiting_ratio": {
   "avg": 64.18947090606699,
   "lb": 62.95972737014761,
   "ub": 65.41921444198636
  },
  "off_shift_ratio": {
   "avg": 0.0,
   "lb": 0.0,
   "ub": 0.0
  },
  "setup_ratio": {
   "avg": 0.0,
   "lb": 0.0,
   "ub": 0.0
  },
  "failure_ratio": {
   "avg": 0.0,
   "lb": 0.0,
   "ub": 0.0
  },
  "loading_ratio": {
   "avg": 0.0,
   "lb": 0.0,
   "ub": 0.0
  }
 }
},
{
 "_class": "Dream.Machine",
 "id": "P9",
 "results": {
  "working_ratio": {
   "avg": 35.31986524561169,
   "lb": 34.26751219317828,
   "ub": 36.3722182980451
  },
  "blockage_ratio": {
   "avg": 0.0,
   "lb": 0.0,
   "ub": 0.0
  },
    "waiting_ratio": {
     "avg": 64.68013475438832,
     "lb": 63.62778170195491,
     "ub": 65.73248780682172
    },
    "off_shift_ratio": {
     "avg": 0.0,
     "lb": 0.0,
     "ub": 0.0
    },
    "setup_ratio": {
     "avg": 0.0,
     "lb": 0.0,
     "ub": 0.0
    },
    "failure_ratio": {
     "avg": 0.0,
     "lb": 0.0,
     "ub": 0.0
    },
    "loading_ratio": {
     "avg": 0.0,
     "lb": 0.0,
     "ub": 0.0
    }
   }
  },
  {
   "_class": "Dream.Queue",
   "id": "QM"
  }
 ],
 "_class": "Dream.Simulation",
 "general": {
  "totalExecutionTime": 12.667999982833862,
  "_class": "Dream.Configuration"
 }
}
```

Example Parallel stations and Queue

Exported JSON file

```
{
 "nodes": {
  "Q1": {
   "capacity": 1,
   "name": "Q1",
   "top": 0.5906862745098039,
   "isDummy": "0",
   "_class": "Dream.Queue",
   "left": 0.639751552795031
  },
"S1": {
 "name": "Raw Material",
 "top": 0.9534313725490196,
 "entity": "Dream.Part",
 "interarrivalTime": {
  "distributionType": "Fixed",
  "mean": 0.5
 },
 "_class": "Dream.Source",
 "left": 0.639751552795031
},
"M1": {
 "name": "MILL1",
 "top": 0.40931372549019607,
 "processingTime": {
  "scale": 5.943555041732533,
  "location": 51.57623425532299,
  "distributionType": "Logistic"
 },
 "failures": {
  "MTTR": {
   "distributionType": "Poisson",
   "lambda": 0.1053658536585366
  },
  "repairman": "W1",
  "MTTF": {
   "shape": 3.1671825421393747,
   "scale": 0.7571939493062068,
   "distributionType": "Weibull"
  }
 },
 "_class": "Dream.Machine",
 "left": 0.6335403726708074
},
"W1": {
 "top": 0.5906862745098039,
 "_class": "Dream.Repairman",
 "capacity": 1,
 "name": "W1",
 "left": 0.18012422360248448
},
"M2": {
 "name": "MILL2",
 "top": 0.40931372549019607,
 "processingTime": {
  "scale": 1.7219415441266923,
  "location": 49.732494067271205,
  "distributionType": "Cauchy"
 },
 "failures": {
  "MTTR": {
   "distributionType": "Poisson",
   "lambda": 0.1423076923076923
  },
  "repairman": "W1",
    "MTTF": {
     "shape": 3.1975046230623905,
     "scale": 0.6805471087485552,
     "distributionType": "Weibull"
    }
   },
   "_class": "Dream.Machine",
   "left": 0.1863354037267081
  },
  "DummyQ": {
   "capacity": 1,
   "name": "DummyQ",
   "top": 0.7720588235294118,
   "isDummy": "1",
   "_class": "Dream.Queue",
   "left": 0.639751552795031
  },
  "E1": {
   "top": 0.04656862745098034,
   "_class": "Dream.Exit",
   "name": "Stock",
   "left": 0.40993788819875776
  }
 },
 "_class": "Dream.Simulation",
 "edges": {
  "1": [
   "W1",
   "M2",
   {}
  ],
  "0": [
   "W1",
   "M1",
   {}
  ],
  "3": [
   "M1",
   "E1",
   {}
  ],
  "2": [
   "S1",
   "DummyQ",
   {}
  ],
  "5": [
   "DummyQ",
   "Q1",
   {}
  ],
  "4": [
   "M2",
   "E1",
   {}
  ],
  "7": [
   "Q1",
   "M2",
   {}
  ],
  "6": [
   "Q1",
   "M1",
   {}
  ]
 },
 "general": {
  "trace": "No",
  "_class": "Dream.Configuration",
  "confidenceLevel": "0.95",
  "maxSimTime": "1440",
  "numberOfReplications": "1"
 }
}
```

- **Example Parallel stations model**

<u>Exported JSON file</u>

```
{
 "general": {
  "numberOfReplications": 5,
  "trace": "No",
  "processTimeout": 10,
  "seed": "",
  "confidenceLevel": 0.95,
  "maxSimTime": 1000,
  "currentDate": "2014/06/16"
```

```
},
"edges": {
 "con_5": [
  "S1",
  "Q1",
  {}
 ],
 "con_25": [
  "St2",
  "E1",
  {}
 ],
 "con_15": [
  "Q1",
  "St2",
  {}
 ],
 "con_10": [
  "Q1",
  "St1",
  {}
 ],
 "con_20": [
  "St1",
  "E1",
  {}
 ]
},
"capacity_by_station_spreadsheet": [
 [
  "Machine",
  "Day 0",
  "Day 1",
  "Day 2",
  "Day 3",
  null
 ],
 [
  null,
  null,
  null,
  null,
  null,
  null
 ]
],
"shift_spreadsheet": [
 [
  "Day",
  "Machines",
  "Start",
  "End"
 ],
 [
  null,
  null,
  null,
  null
 ]
],
"capacity_by_project_spreadsheet": [
 [
  "Project Name",
  "Sequence",
  "Capacity Requirements"
 ],
 [
  null,
  null,
  null
 ]
],
"wip_part_spreadsheet": [
 [
  "Order ID",
  "Due Date",
  "Priority",
  "Project Manager",
  "Part",
  "Part Type",
  "Sequence",
  "Processing Times",
  "Prerequisites Parts"
 ],
 [
  null,
  null,
  null,
  null,
  null,
  null,
  null,
  null,
  null
 ]
],
"nodes": {
 "Q1": {
  "element_id": "DreamNode_2",
  "_class": "Dream.Queue",
  "capacity": 1,
  "name": "Queue",
  "schedulingRule": "FIFO"
 },
 "S1": {
  "name": "Source",
  "entity": "Dream.Batch",
  "interarrivalTime": {
   "distributionType": "Fixed",
   "mean": 1
   },
   "batchNumberOfUnits": 80,
   "element_id": "DreamNode_1",
   "_class": "Dream.BatchSource"
  },
  "E1": {
   "element_id": "DreamNode_5",
   "_class": "Dream.Exit",
   "name": "Exit"
  },
  "St1": {
   "processingTime": {
    "shape": 30.69,
    "rate": 100.14,
    "distributionType": "Gamma"
   },
   "failures": {},
   "element_id": "DreamNode_3",
   "_class": "Dream.BatchScrapMachine",
   "name": "Milling1"
  },
  "St2": {
   "processingTime": {
    "max": 1.0094712814384956,
    "stdev": 0.11040396153317669,
    "min": 0,
    "distributionType": "Normal",
    "mean": 0.6782593968389655
   },
   "failures": {},
   "element_id": "DreamNode_4",
   "_class": "Dream.BatchScrapMachine",
   "name": "Milling2"
  }
 },
 "preference": {
  "coordinates": {
   "Q1": {
    "top": 0.4898081240173095,
    "left": 0.28592454969899506
   },
   "S1": {
    "top": 0.48224351978538194,
    "left": 0.053083038762682624
   },
   "E1": {
    "top": 0.5011550303652008,
    "left": 0.838470725910555
   },
   "St1": {
    "top": 0.22882927801580868,
    "left": 0.5199724933344594
   },
   "St2": {
    "top": 0.7243108552070638,
     "left": 0.5157499788874278
    }
   }
  }
}
```

- **Example Assembly line**

Below the JSON output data format is presented. The highlighted parts reveal as example the inputs of the KE tool in the processing times and scrap quantity for M1A and MM and the WIP levels in PrB and QPr.

```
{
 "nodes": {
  "M1A": {
   "name": "M1A",
   "shift": {
    "endUnfinished": 0,
    "shiftPattern": [
     [
      360,
      780
     ],
     [
      1800,
      2220
     ]
    ]
   },
   "scrapQuantity": {
    "probability": 0.5,
    "distributionType": "Geometric"
   },
   "processingTime": {
    "shape": 11.29570384628515,
    "rate": 10.206416837150321,
    "distributionType": "Gamma"
   },
   "wip": [],
   "failures": {},
   "element_id": "DreamNode_6",
   "_class": "Dream.BatchScrapMachine"
  },
  "Q2A": {
   "capacity": "2",
   "name": "Q2A",
   "isDummy": "0",
   "wip": [],
   "element_id": "DreamNode_17",
   "_class": "Dream.LineClearance",
   "schedulingRule": "FIFO"
  },
"Q2B": {
 "capacity": "2",
 "name": "Q2B",
 "isDummy": "0",
 "wip": [],
 "element_id": "DreamNode_18",
 "_class": "Dream.LineClearance",
 "schedulingRule": "FIFO"
},
"M1B": {
 "name": "M1B",
 "shift": {
  "endUnfinished": 0,
  "shiftPattern": [
   [
    360,
    1260
   ],
   [
    1800,
    2700
   ]
  ]
 },
 "scrapQuantity": {
  "probability": 0.5,
  "distributionType": "Geometric"
 },
 "processingTime": {
  "logmean": 0.09892785211508301,
  "logsd": 0.3125963747837659,
  "distributionType": "Lognormal"
 },
 "wip": [],
 "failures": {},
 "element_id": "DreamNode_7",
 "_class": "Dream.BatchScrapMachine"
},
"S1": {
 "name": "Source",
 "entity": "Dream.Batch",
 "interarrivalTime": {
  "distributionType": "Fixed",
  "mean": "0.5"
 },
 "batchNumberOfUnits": 100,
 "element_id": "DreamNode_25",
 "_class": "Dream.BatchSource"
},
"QPa": {
 "capacity": "3",
 "name": "QPa",
 "isDummy": "0",
 "wip": [],
 "element_id": "DreamNode_22",
 "_class": "Dream.Queue",
 "schedulingRule": "FIFO"
},
"QPr": {
 "capacity": "3",
 "name": "QPr",
 "isDummy": "0",
 "wip": [
  {
   "_class": "Dream.Batch",
   "numberOfUnits": "80",
   "name": "Batch",
   "id": "16839092-001"
  }
 ],
 "element_id": "DreamNode_23",
 "_class": "Dream.Queue",
 "schedulingRule": "FIFO"
},
"PrA": {
 "name": "PrA",
 "shift": {
  "endUnfinished": 0,
  "shiftPattern": [
   [
    360,
    780
   ],
   [
    1800,
    2220
   ]
  ]
 },
 "scrapQuantity": {
  "probability": 0.4,
  "distributionType": "Geometric"
 },
 "processingTime": {
  "scale": 0.2176769587098079,
  "location": 1.1936521519564098,
  "distributionType": "Logistic"
 },
 "wip": [],
 "failures": {},
 "element_id": "DreamNode_15",
 "_class": "Dream.BatchScrapMachine"
},
"QStart": {
 "capacity": "1",
 "name": "StartQueue",
 "isDummy": "0",
 "wip": [],
 "element_id": "DreamNode_24",
 "_class": "Dream.Queue",
  "schedulingRule": "FIFO"
 },
 "PrB": {
  "name": "PrB",
  "shift": {
   "endUnfinished": 0,
   "shiftPattern": [
    [
     360,
     1260
    ],
    [
     1800,
     2700
    ]
   ]
  },
  "scrapQuantity": {
   "probability": 0.5,
   "distributionType": "Geometric"
  },
  "processingTime": {
   "scale": 0.19146677117107616,
   "location": 1.149889151198035,
   "distributionType": "Logistic"
  },
  "wip": [
   {
    "unitsToProcess": "13.0",
    "_class": "Dream.Batch",
    "numberOfUnits": "80",
    "name": "Batch",
    "id": "16839094-001"
   }
  ],
  "failures": {},
  "element_id": "DreamNode_16",
  "_class": "Dream.BatchScrapMachine"
 },
 "PaA": {
  "name": "PaA",
  "shift": {
   "endUnfinished": 0,
   "shiftPattern": [
    [
     360,
     780
    ],
    [
     1800,
     2220
    ]
   ]
  },
  "scrapQuantity": null,
 "processingTime": {
  "shape": 4.8004017272798825,
  "scale": 1.4977931476819555,
  "distributionType": "Weibull"
 },
 "wip": [],
 "failures": {},
 "element_id": "DreamNode_13",
 "_class": "Dream.BatchScrapMachine"
},
"PaB": {
 "name": "PaB",
 "shift": {
  "endUnfinished": 0,
  "shiftPattern": [
   [
    360,
    1260
   ],
   [
    1800,
    2700
   ]
  ]
 },
 "scrapQuantity": {
  "probability": 0.5,
  "distributionType": "Geometric"
 },
 "processingTime": {
  "shape": 18.715189491869822,
  "rate": 12.84936820932644,
  "distributionType": "Gamma"
 },
 "wip": [],
 "failures": {},
 "element_id": "DreamNode_14",
 "_class": "Dream.BatchScrapMachine"
},
"E1": {
 "element_id": "DreamNode_5",
 "_class": "Dream.Exit",
 "name": "Stock"
},
"M2A": {
 "name": "M2A",
 "shift": {
  "endUnfinished": 0,
  "shiftPattern": [
   [
    360,
    780
   ],
   [
    1800,
    2220
   ]
  ]
 },
 "scrapQuantity": {
  "probability": 0.5,
  "distributionType": "Geometric"
 },
 "processingTime": {
  "shape": 3.4463150512174194,
  "scale": 1.342301040240394,
  "distributionType": "Weibull"
 },
 "wip": [],
 "failures": {},
 "element_id": "DreamNode_8",
 "_class": "Dream.BatchScrapMachine"
},
"M2B": {
 "name": "M2B",
 "shift": {
  "endUnfinished": 0,
  "shiftPattern": [
   [
    360,
    1260
   ],
   [
    1800,
    2700
   ]
  ]
 },
 "scrapQuantity": {
  "probability": 0.5,
  "distributionType": "Geometric"
 },
 "processingTime": {
  "logmean": 0.2207444731393393,
  "logsd": 0.2793072333110925,
  "distributionType": "Lognormal"
 },
 "wip": [],
 "failures": {},
 "element_id": "DreamNode_9",
 "_class": "Dream.BatchScrapMachine"
},
"Q3B": {
 "capacity": "2",
 "name": "Q3B",
 "isDummy": "0",
 "wip": [],
 "element_id": "DreamNode_20",
 "_class": "Dream.LineClearance",
 "schedulingRule": "FIFO"
```

```
},
"Q3A": {
 "capacity": "2",
 "name": "Q3A",
 "isDummy": "0",
 "wip": [],
 "element_id": "DreamNode_19",
 "_class": "Dream.LineClearance",
 "schedulingRule": "FIFO"
},
"BDA": {
 "processingTime": {
  "distributionType": "Fixed",
  "mean": "0"
 },
 "element_id": "DreamNode_1",
 "_class": "Dream.BatchDecompositionStartTime",
 "numberOfSubBatches": 4,
 "name": "Deco_A"
},
"BDB": {
 "processingTime": {
  "distributionType": "Fixed",
  "mean": "0"
 },
 "element_id": "DreamNode_2",
 "_class": "Dream.BatchDecompositionStartTime",
 "numberOfSubBatches": 4,
 "name": "Deco_B"
},
"EV": {
 "name": "attainment",
 "argumentDict": {},
 "interval": "1440",
 "start": "1440",
 "_class": "Dream.EventGenerator",
 "method": "Dream.Globals.countIntervalThroughput"
},
"MM": {
 "name": "MM",
 "shift": {
  "endUnfinished": 0,
  "shiftPattern": [
   [
    360,
    1260
   ],
   [
    1800,
    2700
   ]
  ]
 },
 "scrapQuantity": {
  "probability": 0.5,
  "distributionType": "Geometric"
 },
 "processingTime": {
  "scale": 0.08886185000322964,
  "location": 0.44185006205738864,
  "distributionType": "Cauchy"
 },
 "wip": [],
 "failures": {},
 "element_id": "DreamNode_12",
 "_class": "Dream.BatchScrapMachine"
},
"BRB": {
 "processingTime": {
  "distributionType": "Fixed",
  "mean": "0"
 },
 "element_id": "DreamNode_4",
 "_class": "Dream.BatchReassembly",
 "numberOfSubBatches": 4,
 "name": "Assembly_B"
},
"BRA": {
 "processingTime": {
  "distributionType": "Fixed",
  "mean": "0"
 },
 "element_id": "DreamNode_3",
 "_class": "Dream.BatchReassembly",
 "numberOfSubBatches": 4,
 "name": "Assembly_A"
},
"QM": {
 "capacity": "3",
 "name": "QM",
 "isDummy": "0",
 "wip": [],
 "element_id": "DreamNode_21",
 "_class": "Dream.Queue",
 "schedulingRule": "FIFO"
},
"M3B": {
 "name": "M3B",
 "shift": {
  "endUnfinished": 0,
  "shiftPattern": [
   [
    360,
    1260
   ],
   [
    1800,
    2700
   ]
  ]
```

```
    },
    "scrapQuantity": {
     "probability": 0.5,
     "distributionType": "Geometric"
    },
    "processingTime": {
     "shape": 12.679227945840147,
     "rate": 10.380578241576085,
     "distributionType": "Gamma"
    },
    "wip": [],
    "failures": {},
    "element_id": "DreamNode_11",
    "_class": "Dream.BatchScrapMachine"
   },
   "M3A": {
    "name": "M3A",
    "shift": {
     "endUnfinished": 0,
     "shiftPattern": [
      [
       360,
       780
      ],
      [
       1800,
       2220
      ]
     ]
    },
    "scrapQuantity": {
     "probability": 0.5,
     "distributionType": "Geometric"
    },
    "processingTime": {
     "scale": 0.2614287816354414,
     "location": 1.1954189574091665,
     "distributionType": "Logistic"
    },
    "wip": [],
    "failures": {},
    "element_id": "DreamNode_10",
    "_class": "Dream.BatchScrapMachine"
   }
  },
  "edges": {
   "con_165": [
    "Q2A",
    "M2A",
    {}
   ],
   "con_160": [
    "M1A",
    "Q2A",
    {}
```

```
],
"con_205": [
 "M2B",
 "Q3B",
 {}
],
"con_185": [
 "BRA",
 "QM",
 {}
],
"con_200": [
 "Q2B",
 "M2B",
 {}
],
"con_180": [
 "M3A",
 "BRA",
 {}
],
"con_140": [
 "S1",
 "QStart",
 {}
],
"con_145": [
 "QStart",
 "BDA",
 {}
],
"con_250": [
 "PrB",
 "QPa",
 {}
],
"con_255": [
 "QPa",
 "PaA",
 {}
],
"con_230": [
 "MM",
 "QPr",
 {}
],
"con_270": [
 "PaB",
 "E1",
 {}
],
"con_170": [
 "M2A",
 "Q3A",
 {}
```

```
],
"con_175": [
 "Q3A",
 "M3A",
 {}
],
"con_195": [
 "M1B",
 "Q2B",
 {}
],
"con_215": [
 "M3B",
 "BRB",
 {}
],
"con_235": [
 "QPr",
 "PrA",
 {}
],
"con_210": [
 "Q3B",
 "M3B",
 {}
],
"con_150": [
 "QStart",
 "BDB",
 {}
],
"con_155": [
 "BDA",
 "M1A",
 {}
],
"con_240": [
 "QPr",
 "PrB",
 {}
],
"con_225": [
 "QM",
 "MM",
 {}
],
"con_245": [
 "PrA",
 "QPa",
 {}
],
"con_220": [
 "BRB",
 "QM",
 {}
```

```
   ],
   "con_260": [
    "QPa",
    "PaB",
    {}
   ],
   "con_190": [
    "BDB",
    "M1B",
    {}
   ],
   "con_265": [
    "PaA",
    "E1",
    {}
   ]
  },
  "spreadsheet": [
   [
    "Jobs",
    "ID",
    "Order Date",
    "Due Date",
    "Priority",
    "Material",
    "Sequence",
    "Processing Times"
   ],
   [
    null,
    null,
    null,
    null,
    null,
    null,
    null,
    null
   ]
  ],
  "general": {
   "trace": "No",
   "processTimeout": "0.5",
   "confidenceLevel": "0.95",
   "maxSimTime": "2880",
   "numberOfReplications": "1"
  }
 }
```