A new methodology for automated Petri Net generation: method application

Christina Latsou, Sarah, J. Dunnett and Lisa M. Jackson

Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK

I. Introduction

The reliability prediction of an engineering system/process is one of the most important design considerations. Reliability modelling should be applied at the earliest stages of the design effort in order to be effective and by incorporating reliability concepts from these initial stages, failures can be avoided and mitigations put in place before they create greater financial and logistical problems later in the lifecycle. A number of mathematical reliability modelling techniques such as Reliability Block Diagrams (RBDs), Fault Trees (FTs), Binary Decision Diagrams (BDDs), Markov approaches, Petri Nets (PNs), etc. exist to conduct a reliability assessment of a system/process. However, certain limitations have been identified in several of these techniques when they are applied to large, complex systems with dependent events and loops. For example, RBDs, BDDs and static FTs are unable to handle systems/processes that include dynamic characteristics, i.e. dependencies and spares. Additionally, the analysis of large systems/processes using Markov approaches results in cumbersome and error-prone models. However, PN models have been found to be able to model complex systems/processes with dynamic characteristics without suffering the state-space explosion as happens in Markov models [1].

The analysis of reliability models once constructed has been the main focus of analysts over the years and this can now be conducted systematically, using bespoke computer software, providing advanced analysis results for a given system/process [2]. However, the model generation still requires considerable time and effort and needs the user to have experience and understanding of the technique. Automating the generation of the reliability model reduces the model construction time and cost, and minimises human error. Due to these benefits past work has been performed on automating the construction of various reliability models. The automated generation of Fault Trees and Failure Modes and Effects Analysis (FMEA) has received the most attention based on literature findings. However, many approaches proposed for these two methods present some restrictions on their applications, i.e. difficulties in handling complex systems/processes, inapplicability to a wide spectrum of systems/processes, requirement of analysts' intervention resulting in semi-automated methods, etc. Additionally, although there are several attempts targeting the generation of automated Petri Net modelling, they are lacking with regard to the level of automation since most of them use semi-automated methods and also the description diagram of the system/process needs to be generated manually by the user.

The aim of the work presented in this paper is the development of a methodology for the automated generation of Petri Nets for complex systems/processes, which takes as its input a topology diagram with the system/process description, as used in industrial sectors ranging from aerospace and automotive engineering to finance, defence, government, entertainment and telecommunications. The PN formalism has been selected to be automatically generated due to the flexibility of this model in handling complex real life scenarios such as systems/processes with a large number of components/activities, control loops, dependent events, redundant and repairable components/activities. The proposed methodology enables the detection of the most critical components and design errors at an early design stage and hence supports alternative designs.

The research focus of this paper is twofold:

- The development of an algorithm that can accept as an input the description diagram of a given system/process and generates automatically the corresponding Petri Net, demonstrated by its application to an IT asset recycling process.
- Simulation of the PN generated for the IT asset process to demonstrate the capability of the technique.

Hence, the contribution of this work is in the enhancement from semi-automated to fully automated methods, using directly the system representation from industry.

The remainder of this paper is organised as follows. Section 2 reviews past related work. A description of the Petri Net modelling approach is described in Section 3. An overview of the methodology for the automated PN generation is given in

Section 4, introducing the techniques and tools used for its implementation. In section 5, the automated PN model is generated for an IT recycling process and a simulation is carried out identifying possible limitations of the existing process. Some general conclusions are drawn in the final section.

II. Related Work

Automated reliability model generation requires accurate and complete representation of system/process models, realistic descriptions of local and global behaviours of system components or process activities and attention to the representation of components with dynamic characteristics due to the high complexity they create [3].

A complete system/process representation derived from the initial system/process description consists of the structural and functional perspectives. The structural perspective refers to the topology of the system including information about the input and output ports to and from each component and the way in which the components are connected with each other, whereas the functional perspective corresponds to the role and behaviour that each component plays in the entire system.

Over the last 40 years, there have been several attempts to automate reliability techniques. In the case of Fault Trees two main techniques have been introduced, that were considered as pioneering concepts, namely, the digraph method [4] and the decision table method [5]. An alternative to these methods is the modified decision tables [6] which is an extension of the decision table method. In addition, as the complexity of the engineering systems/processes increased due to complex structures such as circuits and loops, advanced automated methods were developed with the help of programming languages such as Java, C/C++ and others. Additional proposed techniques that target automated FT construction are Expert system methods [8], others such as HiP-HOPS [9] and AltaRica 3.0 and more recent proposed methods that use the basic concepts of the aforementioned methods incorporating new aspects such as the state transition table to describe the operational states of a component in [10], the use of System Modelling Language **(**SysML) to specify the system models in [11] and the generation of matrix-based models in [12]. Some of the methods reviewed present difficulties in handling complex systems/processes such as the decision table methods that do not provide any facilities for the detection and classification of control loops or circuits. In many cases such as in the decision table methods, digraph methods and the modified decision table method, there is not a software package available but an algorithm, which is developed manually. Additionally, some methods are applied to a few types of systems without providing a generic applicability such as the digraph methods and the modified decision table method, which focus on circuit systems. The most frequent shortcoming found during the literature review of the current methods for automated FT modelling is that the input of the system/process description is generated by the user such as in AltaRica 3.0. Although this approach combines both the UML object-oriented programming characteristics and reliability modelling capabilities of Stochastic PN (SPN), it still requires the user intervention to import system information, resulting in the semi-automated generation of models.

Several approaches have been presented for the automated construction of other reliability models such as FMEA [13], Hazard and Operability [14] and Petri Nets [15]-[19]. The work in [15] focuses on the High Level Petri Net (HLPN) generation using the UML Sequence Diagram (SD) and Class diagram to represent the structural and behavioural aspects of a given system/process. The Object Constraint Language (OCL) is also used to provide structural specifications. A HLPN model is generated by the user based on the topology information stored in the Nodes Relationship Table (NRT) that is obtained from the UML SD. The proposed methodology is not fully automated since the system information cannot be obtained automatically from the UML diagrams. Similarly, the method described in [16] is semi-automated. In this work, decision and operational mode tables are used for the component description, whereas the system description corresponds to a topology diagram that shows how the components link together. Additional input information is the failure modes and repair data. A library is also used providing reusability in the future. An algorithm developed in C++ has been proposed generating PNs. The shortcoming of this work is that the user is required to generate as input to the software a system structure file, i.e. a file that includes the structural and behavioural information of the system/process, which is an error-prone and time-consuming process. The method described in [17] focuses on the development of a platform that generates automatically models for multi-agent systems using Coloured Petri Net (CPN) models. The proposed methodology combines the Multi-Agent System Description Language (MASDL) with the Petri Net Description Language (PNDL), which is an XML based declarative language used to specify PNs. This work consists of two steps: the system/process is described by the user using MASDL and then following an algorithm a set of transformation rules are applied to the system description to obtain the CPNs. The main drawback of this method is that the user needs to develop the MASDL code so as to import the system/process information into the algorithm. In [18] a UML State Machine Diagram (SMD) is used as the starting point to generate CPN models. This method is based on the model-to-text (M2T) transformation techniques being carried out using the Acceleo tool, easily integrated into Eclipse environment. Hence, once the SMD for a given system/process is developed following the modelling frame, rules and constraints defined as they defined by the Object Management Group (OMG) SMD metamodel, the Acceleo template in which the user defines the transformation rules between the SMD metamodel, the SMD model and the final CPN model, is developed. Each rule developed in the template maps an element from the metamodel and model to the text that is generated and corresponds to the desired CPN model. Once the transformation rules are applied to the model and metamodel the desired CPN is generated in XML format, which can be imported into a CPN tool and provide the graphical model

visualization. Although the UML SMD aids the user to import automatically the system/process data/information, the Acceleo tool used for the model transformation and the CPN development presents some limitations regarding its weakness to provide advanced features i.e. functions, global variables and data structures, leading to the development of complicated codes. An additional shortcoming of this method is that the rules that define the SMD transformation into CPNs should change in case the CPN tool syntax changes in order to ensure compatibility among tools. The methodology in [19] proposes the translation of Architecture Analysis and Design Language (AADL) models into Petri Net models. The main shortcomings of this work are the manual effort and knowledge required by the user to generate the AADL text model for a given system/process and the weakness of the method to handle complex systems/processes.

Hence, the research motivations drawn from the literature review for the automated Petri Net model generation are as follows:

- A simple and direct representation of the system/process should be chosen. The system/process topology should provide a well-defined structure and hierarchical representation in order to facilitate the representation of the flow and sequence from one activity to another capturing the dynamic behaviour of the various cases. A standard notation readily understandable by all business stakeholders should be provided.
- The input graphical diagram of the system/process description and data should be performed automatically without the user intervention, whereas the aforementioned diagram should be executed in a textual format in order to be manipulated further to obtain the Petri Net incidence matrix i.e. the mathematical form of a Petri Net model.
- The methodology should be fully automated, able to deal with various systems/processes and provide a generic applicability.

III. Petri Net Model

Petri Net (PN) models, which have their origins in the thesis of C.A. Petri in 1962 [20] and for which an international standard IEC 62551 has been published (Analysis techniques for dependability – Petri net techniques) [21], are a versatile and useful tool applied to a wide spectrum of modelling systems/processes. PN can be represented graphically and mathematically. The graphical representation includes both structural and behavioural aspects, which are responsible for the static and dynamic representation of the process or system, respectively. Once, the bipartite graph, i.e. the PN structure, and the marking, i.e. the PN behaviour, are defined the user can obtain information about the behaviour of the given system/process. Additionally, a PN model can be expressed by means of mathematical equations and other mathematical models that can describe the system/process behaviour, i.e. how the system/process changes over time tracking the removal/addition of tokens through the PN places. PNs can be used as a visual communication aid to model the system/process behaviour.

The formal definition of a PN is taken from Schneeweiss [22]:

A PN, G_{PN}, is a graph with markings of nodes and edges as shown in Equation 1:

$$G_{PN} = \left(V_p, V_t, E; M(0), D, W\right) E \subseteq \left(V_p \times V_t\right) \cup \left(V_t \times V_p\right) \quad (1)$$

Where: G_{PN} corresponds to the Petri Net Graph; V_p represents the set of places; V_t is the set of transitions; E corresponds to the set of edges (ordered pairs of nodes), M(0) is the initial marking vector of the set V_p of places; D is the vector of switching delays; W is the vector of weights of edges.

The components of M and W are integers and those of D are non-negative real numbers. So, a PN is an ordered 6-tuple of two sets of nodes (V_p and V_t), edges, vector integers (M and W) and a random non-negative real vector (D).

A Petri Net includes two types of nodes: places (drawn as circles) and transitions (drawn as rectangles). There are two types of transitions, the immediate (drawn as solid rectangles) which when enabled fire immediately, and timed (drawn as hollow rectangles) which have a time delay associated with them. Arcs/edges are used to show the link between places and transitions. The tokens removal from/addition to places through the net describes the dynamic behaviour of the model. Tokens represented by solid dots can be removed from upstream places and created in downstream places only if the corresponding transition has been enabled, i.e. fires. The number of removed/added tokens is performed according to the weights (multiplicity) of the arcs. A transition is enabled and able to fire only if the number of tokens in each of its input places is at least equal to the multiplicity of the corresponding edge from that place. Another element that increases the decision power of the Petri Nets is the inhibitor arc, denoted as a dotted arc. The inhibitor arc does not allow the firing of a transition when the place it comes from includes a token. According to Figure 1, the token from place p1 can fire t1 and pass to place p3 only if p2 does not contain a token.

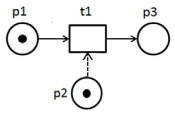

Figure 1 Inhibitor Arc

The removal/addition of tokens through a PN can be transformed into matrix form. Then the marking of the Petri Net after the r^{th} transition, M_r, can be found by Equation 2.

$$M_r = M_0 + A^T . T_1 \quad (2)$$

Where: M_0 is a column matrix (n, 1), where n is the number of places included in the net, showing the initial marking of the

net; T_1 is a column matrix (m, 1) where m is the number of transitions included in the net, showing the number of times each transition has fired in the r transitions; A is the incidence matrix (m, n) where each element a_{ij} shows the effect that transition $_i$ has on place $_j$.

The dynamic and concurrent activities of systems/processes can be simulated using the removal/addition of tokens in the net. For example, the Petri Net structure can capture and describe different components combinations such as components connected in series or parallel, repairable systems with warm spares, load sharing, multiphase missions, pooled repair, system on demand, damage tolerance [23].

IV. Methodology and Modelling Methods

The methodology proposed in this paper introduces the automated generation of a Petri Net model taking as input a UML diagram widely used in industry. The UML diagram includes all the required information, i.e. the topology of a given system/process, for the user to develop the PN model in the form of the incidence matrix A, which is the mathematical representation of a PN model. Figure 2 shows the main elements of the process.

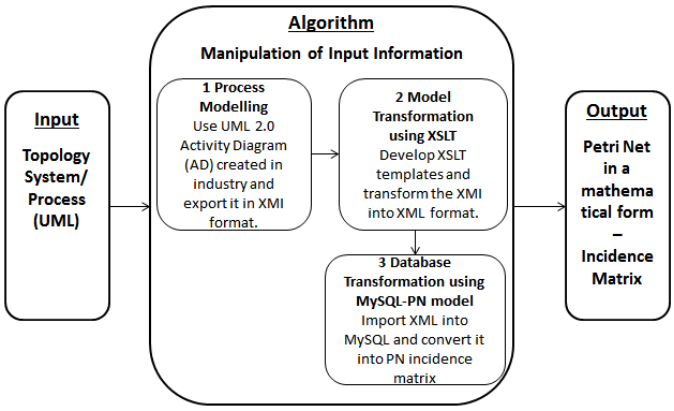

Figure 2 Methodology steps for the automated PN generation

Hence, the first step, *process modelling*, includes the UML 2.0 Activity Diagram (AD) for a given system/process which is provided by industry (though can be developed by the user). Then this diagram is exported into Extensible Markup Language (XML) Metadata Interchange (XMI) format. The structural and behavioural information of the system/process are now transformed into XMI format. The second step, *model transformation using Extensible Stylesheet Language Transformation (XSLT),* takes as input the XMI file developed in the first step, develops the appropriate XSLT templates and outputs a suitably structured XML file to be imported into a database tool for further manipulation. The third step, *database modelling using MySQL-PN model,* focuses on the Structured Query Language (SQL) code development using the system/process information as it is stored in the XML file created in the second step. The code is able to generate the PN incidence matrix.

A thorough description of the methodology steps and the modelling methods and techniques used in this work are explained in the following sections.

A. Step 1 – Process Modelling (PM)

The system/process topology can be described explicitly by Process Modelling (PM) methods. There are several PM methods used in industry such as Unified Modelling Language (UML)/System Modelling Language **(**SysML) diagrams, Business Process Modelling Notation (BPMN), Piping and Instrumentation Diagrams (P&IDs), Computer Aided Design (CAD), Graphical User Interface (GUI), etc. that focus on the mapping of the structural and behavioural aspects of components or activities in systems/processes. The PM can be either developed by a software engineer or provided directly by industry.

Unified Modelling Language (UML), a graphical modelling language applied to engineering systems, was developed by the OMG [24], International Council on Systems Engineering (INCOSE) and Application Protocol 233 (AP233 consortium). It is characterised as a critical enabler for Model Driven Systems Engineering and can cope with model and data interchange via XML Metadata Interchange (XMI®) and the evolving AP233. The UML 2.0 Activity Diagram (AD) has been chosen for this work due to its wide applicability in industry and the well-defined structure, hierarchical representation, directness and well-defined semantics that it provides. The Activity Diagram (AD) is often applied in a PM and it graphically shows the flow of actions and activities in systems/processes. This diagram consists of nodes and control flow edges enabling the dynamic/ behavioural representation of a process model. The nodes include the activity initial node, i.e. a start point illustrated as a small solid circle; the activity final node, i.e. a final point shown as a solid circle with a hollow circle inside; opaque action nodes, i.e. blocks represented as hollow rectangles; decision nodes notated as a diamond-shaped symbol with one incoming edge and two or more outgoing edges; and merge nodes shown as a diamond-symbol with two or more incoming edges and one outgoing edge. The control flow edges included in the diagram correspond to the arcs.

Hence, the PN automation can begin with the UML 2.0 AD. Once the UML 2.0 AD is created or provided directly from industry, then it is exported in XMI format. The XMI file consists of the two elements: nodes and edges as follows:

The XMI nodes are derived either from the AD activity initial/ final nodes or from the UML 2.0 AD opaque action nodes. The following attributes are included in an XMI node element:

- The “type” attribute that defines that this element is a node in the UML 2.0 AD.
  - The “id” that is a unique element identifier.
  - The “name” as this node is presented in the AD.
  - The “incoming” which corresponds to the edge id attribute that enters the node.
  - The “outgoing” attribute that corresponds to

the edge id attribute that leaves the node.

- The XMI edges are derived from the UML 2.0 AD control flow edges, i.e. the arcs. The following attributes are included in an XMI edge element:
  - The "type" which defines that this element is an edge in the UML 2.0 AD.
  - The "id" that is a unique element identifier.
  - The "name" as this edge is presented in the AD.
  - The "target" attribute that corresponds to the node id attribute in which the edge ends up
  - The "source" that corresponds to the node id attribute from which the edge starts.

B. Step 2 – Model Transformation using Extensible Stylesheet Language Transformation (XSLT)

This second methodology step refers to the model transformation process where the XMI file (source model) is transformed into an XML (target model) file. This model transformation is necessary to facilitate the use of the XML document later in the database modelling step, since this XML file is then imported in the database modelling software.

The transformation of the XMI file into an XML format is carried out using XSLT templates. This transformation is accomplished with the help of an XSLT transformer tool that takes as input the XML source model file, i.e. XMI file, and the XSLT document created by the user and creates the XML target model file, i.e. XML file, automatically. The XSLT belongs to the XML family and is used to perform XML transformations allowing the user to specify the desired structure and content of the output file. Hence, XSLT can reorder XML elements, add new elements and decide which elements should be displayed or omitted. XML elements are used to classify data in an XML document. The start and end of an XML element are represented with opening and closing tags, respectively. The transformation process is based on specific template rules defined by the user.

The nodes and edges from the XMI document created in step 1 are used to form an XML document following the XSLT rules. Two XSLT files have been developed providing the rules which are applied to the XMI file as follows:

1 The first XSLT file consists of two templates as follows:
- The first template is for the XMI nodes and retrieves the "incoming", "name" and "outgoing" attributes and their corresponding values.
- The second template is for the XMI edges and retrieves the "id", "name", "target" and "source" attributes and their corresponding values.

The output XML file includes the attributes retrieved from the XMI as mentioned above.

2 The next step in the model transformation targets the transformation of the XML attributes into XML elements. This second XSLT file that consists again of two templates is generated and applied to the XML document created from the first XSLT model transformation. The two XSLT templates developed in the second transformation transforms the XML attributes of nodes and edges into XML elements of nodes and edges, respectively.

Therefore, the two XSLT transformations have formed the target XML document that can be loaded into MySQL Workbench for further manipulation so that the PN incidence matrix can be generated.

C. Step 3 – Database Modelling MySQL–PN Model

A versatile development in the field of software engineering is the database concept that since the late 1980s has been used widely in industry [26]. Database modelling tools are able to capture analyse and organise data in an easy way to be accessed, managed and updated.

MySQL (Michael Widenius Structured Query Language) [27] is a general purpose relational database has been chosen in this work. MySQL, one of the most popular open source visual databases, is a powerful program with high-performance and scalability that uses the SQL data language. The user can create tables by storing, updating and manipulating the data. MySQL can work very quickly with large data sets and with many languages, including C, C++, JAVA, PHP, etc.

In this step the XML file created in the model transformation step is imported into the MySQL software (MySQL Workbench) and an SQL code is developed to manipulate and organise the data in a matrix form similar to that of the PN incidence matrix.

The XML file, developed following the XML transformations in step 2, is automatically loaded into the MySQL Workbench. The final step includes the manipulation and storage of the XML document information into the transpose of the PN incidence matrix using an SQL code.

The automated construction of the transpose of the PN incidence matrix was generated applying the following steps:

1. The 'node' table is created in MySQL. This table consists of four columns, i.e. the "primary_id", "incoming", "name" and "outgoing" columns. The "primary_id" column acts as a primary key giving to each row a unique identification number. The text values of the "incoming", "name" and "outgoing" elements for each node sub-element as included in the XML file, created in the second step, are stored in this table. The order that the records are stored in the 'node' table, i.e. order of rows in the table, is determined by the order that the node elements are identified in the XML file.
2. The 'edge' table is created in MySQL. This table consists of three columns, i.e. the "primary_id", "id" and "target" columns. The "primary_id" column acts as a primary key giving to each row a unique identification number. The text values for the "id" and "target" elements are also stored in this table. The order that the records stored in this table. The order is determined by the order that the edge elements identified in the XML file.

3. The SQL code traces the XML edge sub-elements that have identical targets and source values and replaces the id values of the targets with the id values of the sources, generating the 'decision' table. This table includes the "id" and "name" columns. The data values for the 'decision' table are derived from the 'edge' table and the edge elements from the XML file.
4. The 'new-edge' table is generated. This table consists of the id values derived from the combination of the 'edge' and 'decision' tables, created in steps 2 and 3. It stores for each id the corresponding "name" value as derived from the XML file.
5. The 'node' and 'new-edge' tables, generated in steps 1 and 4, are joined. The "incoming" and "outgoing" columns listed in step 1 in the 'node' table are replaced by the "names" as listed in step 4 in the 'new-edge' table. The 'final' table is developed and three columns are created, the "name_source", "name_activity" and "name_target".
6. A matrix is generated using data from the 'final' table as created in step 5. The matrix columns are defined by the entries of the "name_activity" column, whereas the rows are defined by the entries of the "name_source" column of the 'final' table (step 5). Therefore, once a "name_activity" and "name_source" are in the same row in the 'final' table, then the value -1 should be put in the corresponding matrix cell.
7. Similarly, a second matrix is generated. The matrix columns are defined by the entries of the "name_activity" column, whereas the rows are defined by the entries of the "name_target" column of the 'final' table (step 5). Therefore, once a "name_activity" and "name_target" are in the same row in the 'final' table, then the value +1 should be put in the corresponding matrix cell.
8. The transpose of the incidence matrix is generated by combining the matrices developed in steps 6 and 7.

V. Case Study – Automated Petri Net Generation and Simulation

The automated PN generation is demonstrated by its application to an end of life manufacturing process. A recycling IT asset process has been used to show the application of the methodology developed for automated PN model construction. Once constructed the PN is used to investigate the efficiency of the process.

A. Step 1 – Process Modelling

1) Recycling IT Asset Process Description & UML 2.0 AD Development

The recycling IT asset process targets the repair of electronic devices. Once a mobile device enters the process line, it can end up at in one of two states, either refurbished or scrapped. Decisions and actions along the potential paths in the process include seven different possible activities as follows:

- Asset Track (AT): Asset information is introduced into the traceability system. The characteristics of each product such as model device, battery and memory capacity, screen size, etc., are recorded.
- Visual Inspection (VI): The physical condition of each asset is assessed. If the repair or refurbishment of the device is economically viable, it is forwarded to the Functional Test activity. Otherwise, the device is forwarded to Strip and Scrap.
- Functional Test (FT): The functionality of each product is investigated conducting the following tests/activities such as charger check, battery test, LCD screen check, and ringing test, vibration, microphone and speaker test.
- Data Erasure (DE): Data is erased securely by using specific licensed software.
- Cleaning and De-Labelling (CD): Refurbished products are cleaned properly. Labels are removed and replaced only if considered necessary.
- Repair (R): A product is repaired in case its repair is economically viable.
- Strip and Scrap (SS): Failed assets are checked for any useful parts that can be salvaged and recycled to be used in other cases and are then sent for secure destruction.

All activities can handle only one device at a time except for Data Erasure that can accept 100 devices simultaneously. Each activity has a time to completion associated with it, which can vary for different devices and product types too.

Additionally, each activity has a probability of pass or fail. In practise, most of the activities are carried out at the same physical location, i.e. on the computer. The repair activity (R) however, takes place away from the main refurbishment process but in the same factory, and is not performed until there is a batch requiring repair. For that reason there is a delay between the functional test (FT) and the repair (R) activities. A schematic of the process that includes all the possible paths of the recycling IT process is presented, in Figure 3.

An open source Integrated Development Environment (IDE), Eclipse software, version 4.5 Mars [25], has been used for the UML 2.0 AD development. The AD has been developed representing all of the paths in the IT asset process, and validated successfully. The diagram consists of an initial node ('Start') which corresponds to the start of the process i.e. where a mobile device enters the system, a final node ('End') when the process is completed for a device, 7 opaque action nodes ('Asset Track (AT)', 'Visual Inspection (VI)', etc.), that correspond to the activities carried out through the process, 1 merge node that is used when the output of two activities have a common source node, 4 decision nodes which are used when one activity has two target nodes and control flows with unique names that correspond to the links between the nodes. Additionally, a control loop starting from node 'D_R' exists in the AD, as seen in Figure 3, increasing process complexity.

This is the starting point for the automated PN model generation, which is typically available from industry. From this point all information is extracted automatically involving

the various steps of conversion.

2) AD to XMI format

The Activity Diagram for the IT asset process is exported in XMI format, using the 'Export' option available in Eclipse Mars (4.5). The XMI file consists of nodes such as the Start, Asset Track, Visual Inspection, End etc. and edges such as pin, ATp, VIp, VIf, pout etc. as shown in Figure 3. The XMI nodes correspond to the PN places, whereas the XMI edges to the PN transitions. The XMI "type", "id", "name", "incoming" and "outgoing" attributes for the 'Asset Track' node element are included in Figure 4. The id value is unique for each element. The text next to Figure 4 shows how the XMI node elements are related to the UML 2.0 AD as presented in Figure 3. Similarly, the XMI "type", "id", "name", "source" and "target" attributes of the 'pin' edge element are presented in Figure 5. The text next to Figure 5 shows how the XMI edge elements are related to the UML 2.0 AD in Figure 3.

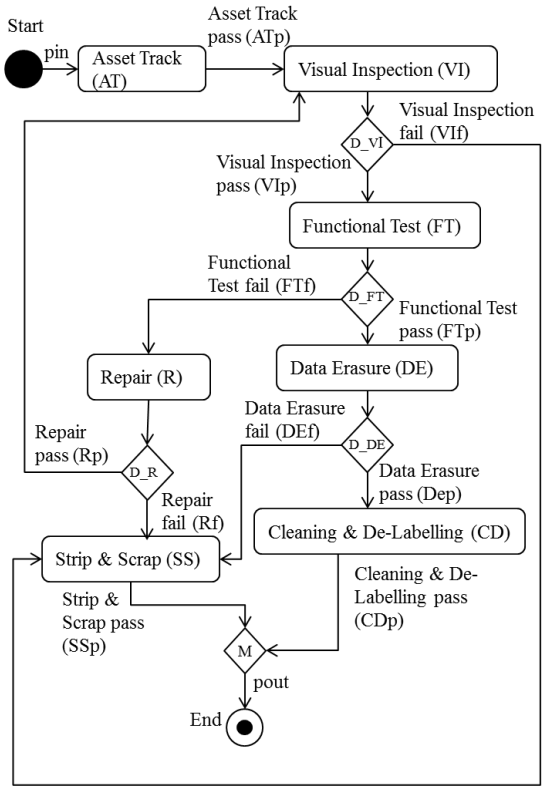

Figure 3 UML 2.0 AD of the IT Asset Recycling Process

```
<node xmi:type="uml:OpaqueAction"
xmi:id="_InKv8LJTEeaTirlhAX5dxQ" name="Asset
Track" incoming="_pWiwMLJTEeaTirlhAX5dxQ"
outgoing="_0ZeSILJTEeaTirlhAX5dxQ"/>
```

'Asset Track (AT)'
'Id' of pin
'Id' of Asset Track pass

Figure 4 Node Element in XMI Format

```
<edge xmi:type="uml:ControlFlow"
xmi:id="_pWiwMLJTEeaTirlhAX5dxQ" name="pin"
target="_InKv8LJTEeaTirlhAX5dxQ"
source="_Ftmh0LJTEeaTirlhAX5dxQ"/>
```

pin
'Id' of Asset Track (AT)
'Id' of Start

Figure 5 Edge Element in XMI Format

B. Step 2 – Model Transformation using XSLT

This step focuses on the XML file development that is loaded in the next step into the MySQL Workbench database environment for automated PN construction. The XML document is obtained from the second stage XMI file transformation using two XSLT files.

Using the methodology described in Section IV, Part B, the first XML file is created and part of it is presented in Figure 6 for the 'Asset Track' node and the 'pin' edge. The text next to Figure 6 shows how the XML node elements are related to the UML 2.0 AD in Figure 3.

```
<? xml version="1.0"?>
<groups xmlns="uri:sadf">
  <node id="_InKv8LJTEeaTirlhAX5dxQ"
        incoming="_pWiwMLJTEeaTirlhAX5dxQ"
        activity="Asset_Track"
        outgoing="_0ZeSILJTEeaTirlhAX5dxQ"/>
  <edge id="_pWiwMLJTEeaTirlhAX5dxQ" name="pin
        target="_InKv8LJTEeaTirlhAX5dxQ"
        source="_Ftmh0LJTEeaTirlhAX5dxQ" />
</groups>
```

'Id' of Asset Track
'Id' of pin
'Id' of Asset Track pass
'Id' of pin
'Id' of Asset Track
'Id' of Start

Figure 6 First XML Format developed from the XMI using XSLT

The second XSLT document applied to the XML file developed from the first XSL transformation consists again of two templates. The text next to Figure 6 describes how the XML edge elements are related to the UML 2.0 AD developed for the IT asset process. In the second XSLT transformation stage, the node or edge child element as included in the first XML node or edge file accordingly is transformed into node or edge root element respectively. Then, the attributes of the node/edge child element, i.e. "incoming", "name", etc., are transformed into sub-elements of the root node or edge XML element.

The XML files developed in this section for the IT asset process consists of 31 elements, 17 are edges, and 14 are nodes. Hence, their size is the same as the XMI document created.

C. Step 3 – MySQL Database Modelling

The final XML file is loaded into MySQL Workbench and the general SQL code developed is used to generate the PN model. The steps to generate the transpose of the PN incidence matrix for the IT asset process were applied as follows:

1 The 'node' table is created using the SQL code as described in Section IV, Part C, step 1. The table developed is illustrated in Table 1. In the activity column the "M" value corresponds to the Merge nodes from the UML 2.0 AD.

2 The 'edge table is created using the SQL code as introduced in Section IV, Part C, step 2. The 'edge' table is presented in Table 2. So, for example it can been seen that the id and target values for the edge presented in the XML document in Figure 6 are placed to the 1st row of Table 2.

Table 1 MySQL 'Node' Table

id	incoming	activity	outgoing
2	_pWiwMLJTEeaTirlhAX5dxQ	Asset_Track	_0ZeSILJTEeaTirlhAX5dxQ
3	_lLpAkpLJUEeaTirlhAX5dxQ	Visual_Inspection	_Ao6DcLJTEeaTirlhAX5dxQ
4	_0ZeSILJTEeaTirlhAX5dxQ	Visual_Inspection	_Ao6DcLJTEeaTirlhAX5dxQ
5	_FrRKELJUEeaTirlhAX5dxQ	Functional_Test	_NbtAgLJUEeaTirlhAX5dxQ
6	_N9kXcLJUEeaTirlhAX5dxQ	Data_Erasure	_XK0xMLJUEeaTirlhAX5dxQ
7	_JXp8QLJUEeaTirlhAX5dxQ	Strip_Scrap	_pJgkULJUEeaTirlhAX5dxQ
8	_d8He0LJUEeaTirlhAX5dxQ	Strip_Scrap	_pJgkULJUEeaTirlhAX5dxQ
9	_jOjo0LJUEeaTirlhAX5dxQ	Strip_Scrap	_pJgkULJUEeaTirlhAX5dxQ
10	_QEscALJUEeaTirlhAX5dxQ	Repair	_h5c3ELJUEeaTirlhAX5dxQ
11	_Ysk30LJUEeaTirlhAX5dxQ	Cleaning_De_Labelling	_vA0V8LJUEeaTirlhAX5dxQ
21	_pJgkULJUEeaTirlhAX5dxQ	M	_1tIc8LJUEeaTirlhAX5dxQ
22	_vA0v8LJUEeaTirlhAX5dxQ	M	_1tIc8LJUEeaTirlhAX5dxQ

Table 2 MySQL 'Edge' Table

id_p	id	target
1	_pWiwMLJUEeaTirlhAX5dxQ	_lnKv8LJTEeaTirlhAX5dxQ
2	_0ZeSILJTEeaTirlhAX5dxQ	_MAZVkLJTEeaTirlhAX5dxQ
3	_Ao6DcLJTEeaTirlhAX5dxQ	_iwxWELJTEeaTirlhAX5dxQ
4	_FrRKELJTEeaTirlhAX5dxQ	_M-adALJTEeaTirlhAX5dxQ
5	_JXp8QLJTEeaTirlhAX5dxQ	_OIQZ8LJTEeaTirlhAX5dxQ
6	_NbtAgLJUEeaTirlhAX5dxQ	_jxsAgLJTEeaTirlhAX5dxQ
7	_N9kXcLJUEeaTirlhAX5dxQ	_NahxQLJTEeaTirlhAX5dxQ
8	_QEscALJUEeaTirlhAX5dxQ	_PslTcLJTEeaTirlhAX5dxQ
9	_XK0xMLJUEeaTirlhAX5dxQ	_ksMg8LJTEeaTirlhAX5dxQ
10	_Ysk30LJUEeaTirlhAX5dxQ	_QDcUQLJTEeaTirlhAX5dxQ
11	_d8He0LJUEeaTirlhAX5dxQ	_OIQZ8LJTEeaTirlhAX5dxQ
12	_h5c3ELJUEeaTirlhAX5dxQ	_ltzH4LJTEeaTirlhAX5dxQ
13	_jOjo0LJUEeaTirlhAX5dxQ	_OIQZ8LJTEeaTirlhAX5dxQ
14	_lLpAkpLJUEeaTirlhAX5dxQ	_MAZVkLJTEeaTirlhAX5dxQ
15	_pJgkULJUEeaTirlhAX5dxQ	_9vls8LJTEeaTirlhAX5dxQ
16	_vA0v8LJUEeaTirlhAX5dxQ	_9vls8LJTEeaTirlhAX5dxQ
17	_1tIc8LJUEeaTirlhAX5dxQ	_ex_zYLJTEeaTirlhAX5dxQ

3 A table named 'source' is created by defining its columns, i.e. 'id_p', 'name' and 'source', and the types of its data. The rows of the table are identified by the 'edge' element following the order presented in the final XML file. The 'name' and 'source' columns hold information derived from the edge elements of the final XML file, as can be seen from Table 3. In Table 3, it is noted that some rows have the same values of sources. For example, the 3rd and 4th rows have the same source value. This happens because those two edges correspond to the outputs of the "D_VI decision node" of the UML 2.0 AD and hence they have the same source element. Therefore, the 'decision' table, shown in Table 4, is created as described in Section IV, Part C, step 3. In this table the records from Table 3 that have the same source values are stored. This is conducted for all the decision nodes.

Table 3 MySQL 'Source' Table

id_p	name	source
1	pin	_Ftmh0LJTEeaTirlhAX5dxQ
1	ATp	_lnKv8LJTEeaTirlhAX5dxQ
4	VIp	_iwxWELJTEeaTirlhAX5dxQ
5	VIf	_iwxWELJTEeaTirlhAX5dxQ
7	FTp	_jxsAgLJTEeaTirlhAX5dxQ
8	FTf	_jxsAgLJTEeaTirlhAX5dxQ
10	DEp	_ksMg8LJTEeaTirlhAX5dxQ
11	DEf	_ksMg8LJTEeaTirlhAX5dxQ
13	Rf	_ltzH4LJTEeaTirlhAX5dxQ
14	Rp	_ltzH4LJTEeaTirlhAX5dxQ
15	SSp	_OIQZ8LJTEeaTirlhAX5dxQ
16	CDp	_QDcUQLJTEeaTirlhAX5dxQ
17	pout	_9vls8LJTEeaTirlhAX5dxQ

Table 4 MySQL 'Decision' Table

id	id_p	name
_Ao6DcLJTEeaTirlhAX5dxQ	4	VIp
_Ao6DcLJTEeaTirlhAX5dxQ	5	VIf
_NbtAgLJUEeaTirlhAX5dxQ	7	FTp
_NbtAgLJUEeaTirlhAX5dxQ	8	FTf
_XK0xMLJUEeaTirlhAX5dxQ	10	DEp
_XK0xMLJUEeaTirlhAX5dxQ	11	DEf
_h5c3ELJUEeaTirlhAX5dxQ	13	Rf
_h5c3ELJUEeaTirlhAX5dxQ	14	Rp

4 The 'new_edge' table, illustrated in Table 5, is created following the SQL code as described in Section IV, Part C, step 4.

5 The code creates the 'final table', presented in Table 6, as explained in Section IV, Part C, step 5. The 'activity' column of Table 6 corresponds to the transitions as presented in the PN model, whereas the 'id_name' and 'name' columns correspond to the input and output places of each transition respectively for each row. The 'id_name' and 'name' columns are the input and the output control flows respectively of each node in the UML 2.0 AD. For example, the 'Asset Track' has input the 'pin' and output the 'ATp'.

6 The code described in Section IV, Part C, step 6 creates the negative matrix as presented in Table 7. A matrix is created including the PN transitions, as presented in the 'activity' column from the 'final' table, in the 1st row and the places, as presented in the 'id_name' column from the 'final' table, in the first column. Once a transition and a place from the 'final' table are in the

same row, then the value -1 is put in the corresponding matrix cell. For example, if the 'Asset Track' is in the same row with the 'pin' in the 'final' table, then the SQL code adds in the corresponding cell of the matrix the value -1.

7 The code described in Section IV, Part C, step 7 generates the positive matrix as presented in Table 8. Similar to step 6, a second matrix with the +1 value is created. Therefore, for example it can be seen that if 'Visual Inspection' is in the same row with 'VIp' / 'VIf' in the 'final' table, then the SQL code adds in the corresponding cell of the matrix the value 1.

Table 5 MySQL 'New-edge' Table

mono	id	name
1	_pWiwMLJUEeaTirlhAX5dxQ	pin
2	_0ZeSILJTEeaTirlhAX5dxQ	ATp
3	_pJgkULJUEeaTirlhAX5dxQ	SSp
4	_vA0V8LJUEeaTirlhAX5dxQ	CDp
5	_1tIc8LJUEeaTirlhAX5dxQ	pout
6	_Ao6DcLJTEeaTirlhAX5dxQ	VIp
7	_Ao6DcLJTEeaTirlhAX5dxQ	VIf
8	_NbtAgLJUEeaTirlhAX5dxQ	FTp
9	_NbtAgLJUEeaTirlhAX5dxQ	FTf
10	_XK0xMLJUEeaTirlhAX5dxQ	DEp
11	_XK0xMLJUEeaTirlhAX5dxQ	DEf
12	_h5c3ELJUEeaTirlhAX5dxQ	Rf
13	_h5c3ELJUEeaTirlhAX5dxQ	Rp

Table 6 MySQL 'Final' Table

primary_id	id_name	activity	name
1	pin	Asset_Track	ATp
2	VIf	Strip_Scrap	pout
3	DEf	Strip_Scrap	pout
4	Rf	Strip_Scrap	pout
5	DEp	Cleaning_De_Labelling	pout1
6	ATp	Visual_Inspection	VIp
7	Rp	Visual_Inspection	VIp
8	ATp	Visual_Inspection	VIf
9	Rp	Visual_Inspection	VIf
10	VIp	Functional_Test	FTp
11	VIp	Functional_Test	FTf
12	FTp	Data_Erasure	DEp
13	FTp	Data_Erasure	DEf
14	FTf	Repair	Rf
15	FTf	Repair	Rp

Table 7 Input Matrix for the IT Asset Process

id_name	AT	VI	FT	DE	R	CD	SS
ATp	0	-1	0	0	0	0	0
DEf	0	0	0	0	0	0	-1
DEp	0	0	0	0	0	-1	0
FTf	0	0	0	0	-1	0	0
FTp	0	0	0	-1	0	0	0
pin	-1	0	0	0	0	0	0
Rf	0	0	0	0	0	0	-1
Rp	0	-1	0	0	0	0	0
VIf	0	0	0	0	0	0	-1
VIp	0	0	-1	0	0	0	0

Table 8 Output Matrix for the IT Asset Process

name	AT	VI	FT	DE	R	CD	SS
ATp	1	0	0	0	0	0	0
DEp	0	0	0	1	0	0	0
DEf	0	0	0	1	0	0	0
FTf	0	0	1	0	0	0	0
FTp	0	0	1	0	0	0	0
pout	0	0	0	0	0	1	1
Rf	0	0	0	0	1	0	0
Rp	0	0	0	0	1	0	0
VIp	0	1	0	0	0	0	0
VIf	0	1	0	0	0	0	0

8 The code generates the transpose of the overall PN incidence matrix for the IT asset recycling process unifying the tables/ matrices developed in steps 6 and 7, as shown in Table 9.

Table 9 Overall Transpose of the PN Incidence Matrix for the IT Asset Process

name	AT	VI	FT	DE	R	CD	SS
ATp	1	-1	0	0	0	0	0
DEf	0	0	0	1	0	0	-1
DEp	0	0	0	1	0	-1	0
FTf	0	0	1	0	-1	0	0
FTp	0	0	1	-1	0	0	0
pin	-1	0	0	0	0	0	0
pout	0	0	0	0	0	1	1
Rf	0	0	0	0	1	0	-1
Rp	0	-1	0	0	1	0	0
VIf	0	1	0	0	0	0	-1
VIp	0	1	-1	0	0	0	0

D. Automated Petri Net Model Generation

From the matrix generated from the automated process and shown for this example in Table 9, a PN model can be developed and is presented in Figure 7. The PN consists of 7 transitions, which correspond to activities, and 11 places. Each transition included in the overall PN in Figure 7 consists of a sub-PN. Figure 8 shows a generalised sub-Petri Net for any activity with a start and end place (Activity Starts and Activity Ends), a transition time (Activity Time), two probability transitions (pass and fail probability transitions) and their corresponding places (pass and fail probability places), the time between two activities (interval activity pass and fail) and the next activity places for the pass and fail paths, correspondingly. All the PN activities are represented by such a net.

Four cases have been identified for the sub-PNs as follows:

- Initial activity where a 'device arrives' place and an 'immediate' transition should be added in the net.
- One probability path, in this case only the pass probability path is required.
- Two probability paths, in this case both the pass and fail probability paths are required.
- Final activity where a 'device leaves' place is used. This is the last place of all the sub-nets.

The incidence matrix for the generalised sub-Petri Net shown in Figure 8 has been created and presented in Table 10.

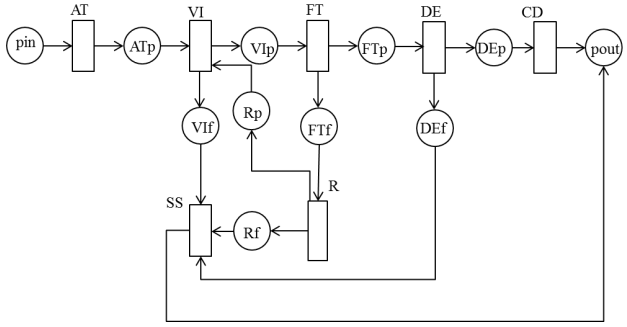

Figure 7 Overall PN model for the IT asset process

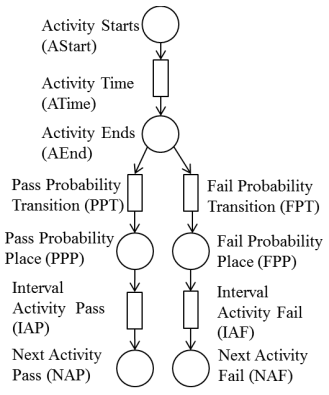

Figure 8 Generalised sub-PN model

Table 10 Incidence matrix for the generalized sub-PN model

	AStart	AEnd	PPP	NAP	FPP	NAF
ATime	-1	1	0	0	0	0
PPT	0	-1	1	0	0	0
IAP	0	0	-1	1	0	0
FPT	0	-1	0	0	1	0
IAF	0	0	0	0	-1	1

Therefore, all the sub-PNs and the corresponding matrices for the IT asset process are developed following the rules defined in this section. For example, for the Asset Track (AT) the sub-PN is presented in Figure 9 and the corresponding transpose of the incidence matrix is shown in Table 11. The AT sub-net relates to the first case identified in this section for the development of the sub-PNs, which includes the initial activity part and hence the AT sub-net consists of the DA place, which declares the existence of a device in the process. The immediate transition allows the device to start with the AT activity immediately after the device is in the process. After the AT activity has completed a token will exists in the ATEnd place and then the probability part is presented. In this case, there is only one path for the device to follow after Asset Track, see Figure 7, and hence the pass probability of the AT is equal to 1 and there is no fail probability path. The ATPPP place enables the removal/addition of a token (device) from the ATPPT to the IATP that corresponds to the interval between the end of the AT activity and the next activity, i.e. the VI.

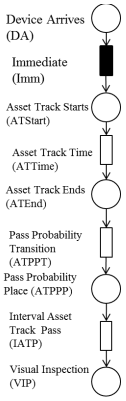

Figure 9 Sub-PN model for the Asset Track Activity

Table 11 Transpose of the sub-PN incidence Matrix for the Asset Track

	DA	ATStart	ATEnd	ATPPP	VIStart
Immediate	-1	1	0	0	0
ATTime	0	-1	1	0	0
ATPPT	0	0	-1	1	0
IATP	0	0	0	-1	1

E. Petri Net Model Simulation

1) Process Data and Petri Net Simulation

This section presents the simulation steps followed following the automated generation of the PN model described above for the IT asset process. The input data for the simulation consists of the values for the time taken for all the activities and probabilities for the pass and fail. This corresponds to the times for all the transitions presented in the sub-PNs developed according to the overall PN in Figure 7. Table 12 shows the pass and fail probabilities for each activity, PPT and FPT in Figure 8, and the minimum and maximum times needed to complete each activity. Table 13 shows the minimum and maximum times required for a device to move one activity to another, used to determine IAP and IAF in Figure 8. The data used for the simulation comes from 2113 mobile phones processed over 323 hours.

Table 12 Pass and fail probabilities & minimum and maximum activity times for the IT asset process.

Activity Time	Pass Probability	Fail Probability	min_time (secs)	max_time (secs)
AT	1	0	107	148
VI	0.688	0.312	5	10
FT	0.733	0.267	60	180
R	0.294	0.706	240	900
DE	0.971	0.029	30	40
CD	1	0	30	60
SS	1	0	30	60

Table 13 Minimum and maximum interval times for the IT asset process

Interval Pass/Fail Activity Time	min_time (secs)	max_time (secs)
AT pass	30	120
VI pass	300	1800
VI fail	300	3600
FT pass	1800	7200
FT fail	7200	8640
R pass/fail	1800	28800
DE pass/fail	1800	10800
CD pass	0	0
SS pass	0	0

The algorithm developed creates a connection between the MySQL and Eclipse retrieving the generated PN incidence matrix and reads the data from Tables 12 and 13 stored in Excel. During the simulation the activity times and interval activity times needed in the sub-PNs are generated using Equation 3. The sojourn time in the current state, found by applying Equation 4, can be computationally modelled using any cumulated distribution function, such as exponential, related to the time of occurrence of the corresponding event. This indicates that the developed model follows the SPN concept where the delays are randomly chosen by sampling distributions associated with transitions. Hence, the SPN model that adds flexibility and a wider range of applicability is considered.

$$t = \min\_time + (\max\_time - \min\_time) * x \quad (3)$$

$$tsojourn = (t\_activity + t\_interval\_activity) * x \quad (4)$$

Where: t is the time in the current activity/interval state; tsojourn is the sum of activity, i.e. service, time and interval, i.e. waiting, time; *min_time* and *max_time* are those given in Tables 12 and 13; and x is a uniformly distributed random number in the interval (0, 1).

Random numbers for all the PN transitions, i.e. activity transitions, probabilistic transitions and interval activity transitions, are generated, and hence the PN model is transformed into a stochastic PN (SPN) since time t obtained applying Equation 3 is a random variable. For the probabilistic transitions, if the random number generated is lower than or equal to the pass probability of an activity then the device is assumed to pass, otherwise the device fails. Equation 2 presented in Section III is also used in the simulation using the matrices generated for the overall PN and sub-PNs for the IT asset process. This equation can show each time the removal/addition of the token into the PN model.

The simulation can provide the average time each path requires to be completed, the average time for each transition, the most common visited places in each path, as well as the paths resulted most in failure and the nodes most involved with route to failure.

2) Simulation Results and Discussion

The results obtained from the simulation, conducted to investigate the process performance and identify possible deficiencies that exist in the IT process, are discussed in this section. The six paths identified in the IT asset process, using the incidence matrix in Table 9 and the overall PN in Figure 7, and then simulated are shown in Table 14 in the 'Path Activities' column.

Table 14 Paths and average times of each path for the IT asset process

Path ID	Path Activities	Average Path Time (secs)
1	pin-AT-ATp-VI-VIp-FT-FTp-DE-DEp-CD-pout	12398.14
2	pin-AT-ATp-VI-VIp-FT-FTp-DE-DEf-SS-pout	1229.05
3	pin-AT-ATp-VI-VIp-FT-FTf-R-Rp-VI-VIp-FT-FTp-DE-DEp-CD-pout	38698.8
4	pin-AT-ATp-VI-VIp-FT-FTf-R-Rp-VI-VIp-FT-FTp-DE-DEf-SS-pout	35140.8
5	pin-AT-ATp-VI-VIp-FT-FTf-R-Rf-SS-pout	24657.25
6	pin-AT-ATp-VI-VIf-SS-pout	2158.8

All the six paths listed in Path Activities column in Table 14 can be found in the overall PN in Figure 7. The average times for each path obtained by applying the Monte Carlo Simulation have been processed by applying the chi-square law, estimating the confidence interval of 0.9. This enables estimation of the precision of results obtained from the simulation. These results are shown in Table 14. The simulation results have shown that the 3rd and 4th paths are the longest. This is due to the Repair stage, which is the most time consuming stage in both paths.

To conclude the simulation findings, the average interval repair pass or fail time creates long delays in the repair path. This is a limiting factor in the process and happens because the repair of the devices is completed in a different location from the rest of the activities and hence additional time is required for the transportation of the assets. From the simulation results, some recommendations are provided to improve the process's performance.

- Increase the ability of the activities to accept multiple devices simultaneously as the Data Erasure does.
- Locate the activities at the same place in order to decrease the interval times and the manpower required.

The PN obtained by the automated procedure for the recycling IT asset process has been verified successfully, by checking its structural and behavioural properties such as boundedness, liveness and safeness. Additionally, once timing and probabilistic data was introduced into the corresponding PN transitions, the automated PN generation procedure has been validated via the PN's simulation. Initially, the simulation algorithm visually checked the movement of tokens through the PN paths, validating that the paths followed the same route as the paths existing in the UML AD provided for the recycling IT asset process. The algorithm has also been validated by comparing data from the IT process, with simulation results, which were estimated by following the various PN paths, proving that the PN is a realistic representation of the recycling IT asset process. Therefore, the algorithm used for the automated PN model generation is correct, complete and develops PN models with accuracy satisfying its intended purpose. Therefore, the PN model is necessary to: (i) check the correctness of the algorithm developed for the automation procedure; and (ii) be simulated to investigate the system/process performance identifying possible deficiencies. In this work, the PN model for the recycling IT asset process has been automatically generated and used for the simulation. However, the full benefits of the methodology are in the application to more complex systems/processes with a larger number of components/activities and paths, where the PN visualisation, animation and graphical representation of simulation results can improve the quality of decision-making.

F. Comparison of PNs Construction Methods

In this section, a comparison between the current work and the methods that focus on the semi-automated and automated construction of PNs, as they were reviewed in Section II (Related Work), has been made in terms of the number of steps involved in the process. From the methods identified in the literature and included in Section II for the PN model generation, i.e. [15] – [19], only method [18] is fully automated. All the other methods require the user's intervention in order to input the system/process representation into the algorithm. The methodology outlined in the paper retrieves the topology information from the graphical diagram, i.e. the UML/SysML AD, of the system description, without user intervention, and generates the mathematical representation of the corresponding PN model. Hence, a comparison between the current work and the previous methods has been made in terms of the number of steps involved in the process, indicating level of user input needed. This comparison, as viewed in block A in Figure 10, indicates the novelty and enhancement of the technique in this paper regarding removing the model pre-processing that is required with the techniques in the literature.

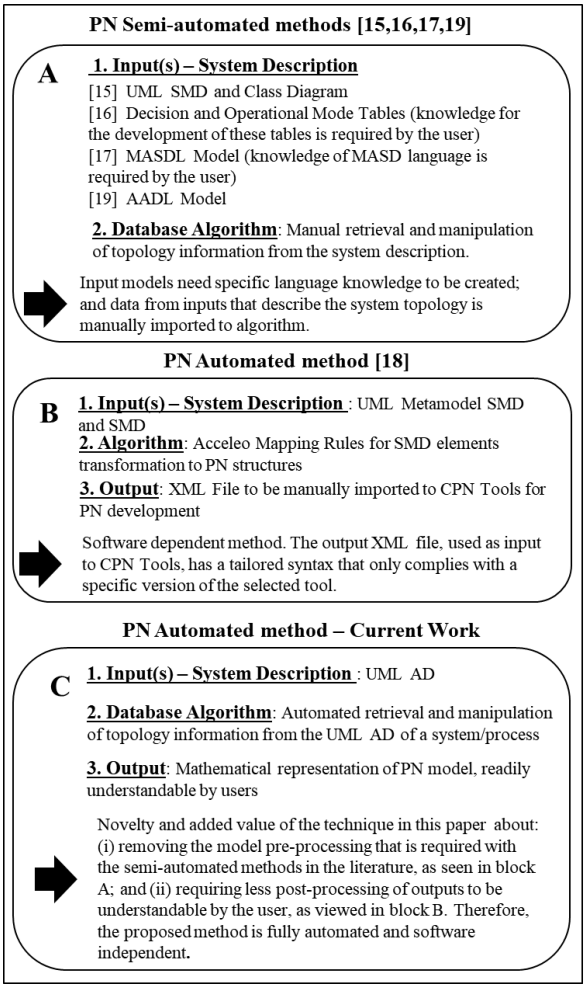

Figure 10 Comparison of PNs Construction Methods

The automated method described in [18] uses as inputs the OMG State Machine Diagram (SMD) metamodel with some modifications and a SMD and for each element of the metamodel and model (SMD) a template is developed using the model-to-text transformation tool, named Acceleo. These templates act as a translation mechanism including mapping

rules that transform the SMD elements such as states to corresponding PN structures such as places. The output of this algorithm, as seen in Figure 10, is an XML file, which is manually imported into CPN Tools to obtain the graphical PN model visualisation. Although this method can automatically retrieve the topology information of a system from a UML SMD and generate PN models, is software dependent on the destination syntax, i.e. syntax of CPN Tools. The output XML file, used as input to CPN Tools, has a tailored syntax that only complies with a specific version of the selected tool. This syntax may be incomplete or faulty after a version update and hence the PN model is considered inaccurate. Therefore, method [18] lacks efficiency and cannot provide a robust and rigorous methodology. Like for the semi-automated cases the comparison of steps in the algorithm is made with our approach, as viewed in block B in Figure 10. This comparison indicates that the proposed method requires less post-processing of outputs to be understandable by the user.

Finally, in block C in Figure 10, the methodology steps, as well as the novelty and added value of the current work compared to the reviewed techniques identified in the literature are summarised.

VI. Conclusions

In this paper, a methodology for the automated PN model generation from the system/process description has been described. The methodology has been demonstrated by applying it to an IT asset recycling process example where the PN incidence matrix has been generated from a UML 2.0 Activity Diagram and then simulated to assess the process performance. The methodology is based on three main steps: the process modelling, i.e. the UML 2.0 AD development; the model transformations using the XSLT, i.e. XSLT templates development and XMI transformation into XML format; and database modelling using SQL code, i.e. XML file manipulation developing SQL code for the matrix generation.

An algorithm has been developed to establish a connection between the incidence matrix in the database software and the Java Integrated Development Environment (IDE), read data stored in Excel files and run a simulation. Currently the simulation is being extended to consider various scenarios such as having multiple devices in the process, to identify improvements in the process.

The proposed methodology overcomes the limitations of human-aided reliability model construction by saving time and effort. Additionally, the novel methodology, proposed for the automated PN model generation, applying a database (MySQL) algorithm, contributes to knowledge through the combination of the following:

- **Fully automated PN model generation capability**: the novel method retrieves the topology information from the UML AD of a system/process, without the user intervention and directly transforms this information into a PN model, overcoming the weakness of the most current attempts reviewed that require the user intervention to import system information to the algorithm, resulting in the semi-automated generation of PN models.
- **Systems/Processes modelling characteristics**: the proposed method can handle and efficiently model systems/processes with control loops.
- **Generic domain applicability**: the proposed algorithm provides a wide applicability spectrum, without targeting specific domains.
- **Software independence**: The output matrix of the proposed methodology is readily understandable by the user without being based upon the syntax of any industrial software which can be easily modified after a version update, and hence to fail the desired model generation. (Software dependent can be considered a methodology that generates outputs in XML format, which are then imported to tools to produce either a matrix or a net that can be meaningful to users.)

However, further generic capability of the method needs to be explored since the developed methodology cannot provide valid matrices for any UML AD, since only certain elements such as opaque action, decision, merge, etc., have been considered. Thus, as future work, the proposed algorithm should be extended to a generic methodology that provides transformation rules for mapping all the AD elements into PNs, for any potential AD provided by industry.

Future investigation could involve the automated generation of the mathematical form of the PN for the recycling IT asset process increasing the complexity of the algorithm by: (i) accepting multiple devices with several tokens into the start PN place; and (ii) introducing inhibitor arcs processing only one device at the same time. Additional future work could include the application of the developed methodology in complex systems/processes so as to detect possible limitations, optimise the algorithm and ensure the generic applicability of the method. Another line of investigation includes the automated graphical representation/visualisation of the PN model that can facilitate the understanding of the structure and behaviour of the net.

Additional extension of this work is to check the correctness of the algorithm developed for the PN automation procedure by: (i) verifying that the PN model obtained performs the correct function by checking behavioural and structural properties of PN via reachability graph or place/transition invariants; and (ii) validating the PN model obtained accurately represents the system architecture by checking a) visually the system's behaviour playing the token game and b) the model's quality by obtaining numerical results and comparing these numerical results with observed in the real world system (numerical simulation). Finally, an animated graphical user interface (GUI), readily understandable by the user, could be built to represent the results of calculations made on the low-level model (SPN) into the high-level model (UML).

ACKNOWLEDGMENT

THE RESEARCH REPORTED IN THIS PAPER ALIGNS TO THE WORK BEING RESEARCHED AS PART OF THE EPSRC GRANT EP/K014137/1.

REFERENCES

[1] J. V. Zille, C. Bérenguer, A. Grall and A. Despujols, Simulation of maintained multicomponent systems for dependability assessment", In Faulin, Javier and Juan, Angel A. and Martorell, Sebastian and Ramirez-Marquez, J.E. (eds), *Simulation Methods for Reliability and Availability of Complex Systems*, Springer Series in Reliability Engineering, vol. 12, no. 1, pp. 253-272. London, U.K.: Springer, 2010.

[2] J. B. Dugan, K. J. Sullivan and D. Coppit, "Developing a Low-Cost High-Quality Software for Dynamic Fault-Tree Analysis", *IEEE Transactions on Reliability*, vol. 49, no. 1, pp. 49-59, 2000.

[3] A. Carpignano and A. Poucet, "Computer Assisted Fault Tree Construction: A Review of Methods and Concerns", *Reliability Engineering and System Safety*, vol. 44, pp. 265-278, 1994.

[4] S. A. Lapp and G. J. Powers, "Computer-aided Synthesis of Fault Trees", *IEEE Transactions on Reliability,* vol. 26, no.1, pp. 2-13, 1977.

[5] S. L. Salem, G. E. Apostolakis and D. Okrent, "A new methodology for the computer-aided construction of fault trees", *Ann. Nucl. Energy,* vol. 4, no.9-10, pp.417-433, 1997.

[6] J. D. Andrews and J. J. Henry, "A computerized fault tree construction methodology", Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering, vol. 211, no. 3, pp.171-183, 1977.

[7] A. Rauzy, "Mode automata and their compilation into fault trees", *Reliability Engineering and System Safety,* vol. 78, no. 1, pp. 1-12, 2002.

[8] G. Xie, D. Xue and S. Xi, "Tree-Expert: A tree based expert system for fault tree construction", *Reliability Engineering and System Safety, vol.* 40, no. 1, pp. 295-309, 1993.

[9] T. Prosvirnova, M. Batteux, P.-A. Brameret, A. Cherfi, T. Friedlhuber, J.-M. Roussel and A. Rauzy, "The AltaRica 3.0 project for Model-Based Safety Assessment", *Proceedings of 4th IFAC Workshop on Dependable Control of Discrete Systems, DCDS 2013*, York (Great Britain), September 2013. IFAC.

[10] A. Majdara and T. Wakabayashi, "Component-based modelling of systems for automated fault tree generation", *Reliability Engineering System Safety*, vol. 94, no 6, pp.1076-1086, 2009.

[11] J. Xiang, K. Yanoo, Y. Maeno and K. Tadano, "Automatic Synthesis of Static Fault Trees from System Models", *Proceedings of the 5th International Conference on Secure Software Integration and Reliability Improvement (SSIRI '11)*, PP. 127-136, 2011.

[12] M. Roth, M. Wolf and U. Lindemann, "Integrated matrix-based Fault tree generation and evaluation", *Procedia Computer Science*, vol. 44, no. 1, pp. 599-608, 2015.

[13] Y. Papadopoulos and C. Grante, "Evolving car designs using model-based automated safety analysis and optimisation techni*que*", *The Journal of Systems and Software,* vol. 76, no. 1, pp. 77-89, 2005.

[14] C. Zhao, M. Bhushan and V. Venkatasubramanian, "PHASUITE: An automated HAZOP analysis tool for chemical processes: Part I. Knowledge Engineering Framework", *Process Safety and Environmental Protection,* vol. 83, no. B6, pp. 509-532, 2005.

[15] A. Alhroob, K. Dahal and H. Alamgir, H. "Transforming UML Sequence Diagram to High Level Petri Nets", *IEEE International Conference of Software Technology and Engineering*, pp. 260-264, 2011.

[16] K. S. Stockwell and S. J. Dunnett, "Automatic construction of a reliability model for a phased mission system", *Proceedings of the 20th Advances in Risk and Reliability Technology Symposium,* pp. 192-204, 2013.

[17] M. Taibi, M. Ioualalen and R. Abdmeziem, "An Automatic Petri-net Generator for Modelling Multi-agent Systems', *ICSEA 2013: The Eighth International Conference of Software Engineering Advances,* 2013

[18] É. André, M. M. Benmoussa and C. Choppy, "Translating UML State Machines to Coloured Petri Nets Using Acceleo: A Report", J. Pang and Y. Liu (Eds.) 3rd International Workshop on Engineering Safety and Security Systems 2014 EPTCS 150, 2014.

[19] H. Reza and A. Chatterjee, "Mapping AADL to Petri Net Tool-Sets Using PNML Framework", *Journal of Software Engineering and Applications*, vol. 7, no. 11, pp. 920-933, 2014.

[20] C. A. Petri, 1962. "Kommunikation mit Automaten". Rheinisch-Westfaelisches Institut fuer Instrumentelle Mathematik and der Universitaet Bonn, Schrift Nr. 2; English Translation: "Communication with Automata, Griffiss Ari Force Base, New York, RADC-TR-65-377, vol. 1, suppl. 1, 1996.

[21] IEC62551, "Analysis techniques for dependability – Petri net techniques", International Electrotechnical Commission, Geneva, 2012.

[22] W. G. Schneeweiss, Petri nets for reliability modelling: in the fields of engineering safety and dependability, LiLoLe Verlag, Hagen, 1990.

[23] V. V. Volovoi, "Modelling of System Reliability Petri Nets with Aging Tokens", *Reliability Engineering and System Safety*, vol. 84, no. 2, pp. 149-161, 2004.

[24] OMG DMN Specification. Available at: http://www.omg.org/spec/DMN/1.0/Beta2

[25] Eclipse. 2015. http://www.eclipse.org

[26] T. M. Connolly and C. E. Begg, Database systems: A practical approach to design, implementation, and management. (4. [rev.] ed.), Harlow: Addison-Wesley, 2005.

[27] MySQL Workbench.2017. http://mysql.com