Automated generation of a Petri Net model: application to an end of life manufacturing process

C. Latsou, S.J. Dunnett & L.M. Jackson
Loughborough University, Loughborough, UK

ABSTRACT: As the complexity of engineering systems and processes increases, determining their optimal performance also becomes increasingly complex. There are various reliability techniques available to model performance, for example fault trees, simulation etc., but generating the models can become a significant task that is cumbersome, error-prone and tedious. This can result in significant resources being devoted to the generation of the models and there is much room for error. Hence over the years work has been undertaken into automatically generating reliability models. Such an approach enables the detection of the most critical components and design errors at an early design stage, supporting alternative designs and systems. The aim of the research described in this paper is the automatic generation of a Petri Net model for a given system or process. The Petri Net approach enables complex systems and processes to be modelled using a modular approach. The methodology of the automated Petri Net generation outlined in this work is to extract the information required for the model from the system description in a form used by industry, such as a UML Activity Diagram, into a database using XML transformations. An algorithm is then applied to generate the Petri Net incidence matrices of the necessary nets, which is the mathematical representation of the model. The algorithm builds the nets up in a modular fashion enabling changes to be made to the overall net in a cost effective way hence allowing various designs to be easily assessed. In this work the procedure will be demonstrated by its application to an end of life manufacturing process.

1 INTRODUCTION

One of the most important considerations when designing an engineering system or process is reliability. The techniques available to perform a reliability assessment of a system/process can be divided into two main categories, analytical and simulation techniques. The analytical reliability modelling techniques include various approaches from which an analyst is able to choose the most suitable technique for their given problem. The techniques for failure analysis consist of combinatorial models, including Reliability Block Diagrams (RBDs), Fault Trees (FTs) and Binary Decision Diagrams (BDDs), state-space models, including the subcategory of Markov approaches, and hierarchical models generated by the combination of combinatorial and state-space models, which are able to simplify the model and ease further analysis (Lanus et al. 2003). With the current development in engineering technologies, the complexity of these engineering systems/processes is ever increasing and proportionally so potentially do the risks and hazards. The aforementioned analytical models contain certain limitations when they are applied to complex structures. More specifically, the combinatorial models are limited in their ability to model dynamic characteristics, such as dependent events and spares, whereas the Markov models, which can cope with dynamic features, suffer the state-space explosion as the number of states increases. Therefore, large complex scenarios are difficult to be modelled and controlled using Markov approaches, as the final diagram can be cumbersome, error-prone and computationally costly.

However, alternative simulation approaches have been developed, such as the encoding of the state-space model in a Petri Net (PN) that can cope with all the aforementioned limitations (Zille et al. 2010). PN models are powerful, flexible structures that can be applied to complex cases without suffering the state-space explosion limitation. Additionally, the computer simulation, a ubiquitous and flexible modelling technique widely used in industrial cases for the behavioural analysis of complex models can represent the system/process in an efficient way and hence enables informed decisions to be made due to its reusability.

Therefore, the need to overcome the limitations of existing reliability models and address the challenging requirement to detect the most critical components and design errors at an early stage of the design has led to the research outlined in this paper where an algorithm to automatically generate a PN model from a system/process description is described and demonstrated.

The research focus of this paper is the development of an algorithm, applicable to complex cases, that accepts as an input the description diagram of a system/process and generates automatically the corresponding PN model, demonstrated by application to an IT asset recycling process.

This paper is organised as follows. In section 2 the main automated reliability modelling methods are reviewed, as identified in the literature, and their limitations are discussed. Section 3 introduces the Petri Net model. Section 4 provides an overview of the methodology steps for the automated PN generation introducing the techniques and tools used for its implementation. In section 4, the automated PN is generated for an IT recycling process. Some general conclusions are drawn in Section 5.

2 AUTOMATED RELIABILITY MODELLING METHODS REVIEW

Early attempts at developing automated reliability models were documented in the 1970's when two major techniques, decision table methods (Salem et al. 1977) and digraph methods (Lapp & Powers 1977), were introduced for Fault Tree automation. Since then, several techniques have been proposed for the formalisation of Fault Trees and other reliability models, such as Failure Mode and Effect Analysis (FMEA) (Papadopoulos & Grante 2003), Hazard and Operability Study (HAZOP) (Zhao et al. 2005), PNs (Bernardi et al. 2002, Stockwell & Dunnett 2013) etc. Some of these techniques, such as the component model based methods, are alternatives to the decision table and digraph methods. Additionally, as the complexity of the engineering systems/ processes increased due to complex structures such as loops and electric circuits, higher level formalisation methods were developed with the help of programming languages such as Java, C/C++, AltaRica (Rauzy 2002) and others. The automated reliability modelling methods identified are as follows: Decision Table Methods; Digraph methods; Component model based methods (Taylor 1982); Expert system methods (Xie et al. 1993); and AltaRica Data-Flow Language.

There are several methods for the automatic model generation in the literature that either use the methods discussed above or their combinations or introduce alternative concepts with novel characteristics (Majdara & Wakabayashi 2009, Li & Li 2014, Roth et al. 2015).

However, the main deficiencies for the automated generation of reliability models identified in the literature are briefly discussed as follows:

- The range and domain that the approach targets. There are approaches that use libraries and focus on specific domains such as mechatronics, without providing a general methodology applicable to complex engineering systems/ processes.
- The degree of automation. Some efforts result in semi-automated reliability model generation, since the analyst should compose the code for a specific simulation environment or the data is not derived automatically from the description diagram, but it is imported manually by the user.
- The level of the system's/ process's complexity. Although most methods argue that they are applicable in complex cases, only a limited number of them prove the automated model generation of complex engineering systems/ processes including dynamic characteristics.

The proposed methodology in this paper contributes to the automated area by addressing the deficiencies in current automated modelling methods and develops a generic automated model generation methodology for complex cases.

3 PETRI NET MODEL

Petri Nets, first introduced in the thesis of C.A. Petri (Petri 1962), are a powerful, visual tool that provide a rigorous and precise analysis, modelling the system/process behaviour. A Petri Net is a bipartite directed graph, consisting of two types of nodes, places and transitions. The nodes are connected together with directed arcs. The basic elements for the construction of PN graphs can be presented as follows:

- Places (circles, denoted p_i, marked with tokens).
- Transitions (squares or bars, denoted t_j, describe time delay (D), or probabilities, of the output).

The movement of the tokens between places describes the dynamic behaviour of the model. The marking of the PN after the r^{th} transition, M_r, can be found by Equation 1.

$$M_r = M_0 + A^T . T_1 \quad (1)$$

where M_0 is a column vector of size (n, 1), where n is the number of places, showing the initial marking of the net; T_1 is a column vector of size (m, 1) where m is the number of transitions, 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} corresponds to the effect that transition $_i$ has on place $_j$. Using Equation 1, the marking of a net can be tracked at any time.

PNs have been widely used in industry in several fields such as data communication processes, computer networks, workflows and manufacturing plants (Wang 2007). They are a useful modelling tool, able to simulate and analyse a system/process by predicting its performance. Hence, the PNs provide both numerical and graphical modelling analysis. Hence, the PNs provide both numerical and graphical modelling analysis, using software tools, such as Java/ C++, Matlab/ Simulink, PN Toolbox

4 METHODOLOGY STEPS & MODELLING METHODS

The novelty of this work is the automated PN generation taking as an input a UML diagram, a representation commonly used in industry, manipulating the diagram's information into a database modelling software and creating the PN mathematical representation, the transpose of the incidence matrix.

This section describes the steps followed for the automated PN generation and also introduces the modelling methods and techniques used in the methodology. Figure 1 presents a diagram outlining the methodology steps followed for the automated PN generation. Step 1 (Process Modelling) takes the description of the system/process from a UML Activity Diagram (AD) and exports it into an Extensible Markup Language (XML) Metadata Interchange (XMI) format to retrieve the structural and behavioral aspects of the system/process. Step 2 (Model Transformation using XSLT) develops Extensible Stylesheet Language Transformation (XSLT) files in order to transform the XMI file, developed in step 1, into an XML file, suitably formed to be imported into MySQL Workbench. Step 3 (Database Modelling using MySQL- PN Model) loads the XML file into MySQL Workbench and generates and Structured Query Language (SQL) code that transforms the XML data into a PN representation in the form of the transpose of the incidence matrix.

The methodology steps are explained in detail in the following sections.

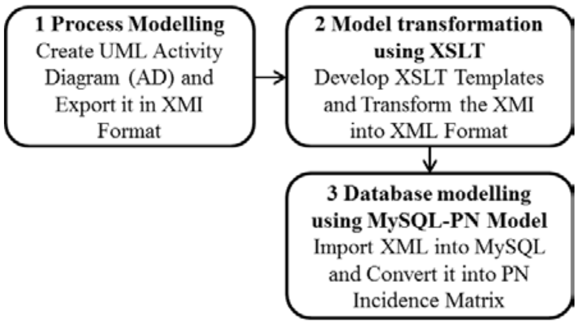

1 Process Modelling
Create UML Activity Diagram (AD) and Export it in XMI Format
2 Model transformation using XSLT
Develop XSLT Templates and Transform the XMI into XML Format
3 Database modelling using MySQL-PN Model
Import XML into MySQL and Convert it into PN Incidence Matrix

Figure 1. Methodology steps for the automated PN generation.

4.1 *Step 1 – Process Modelling (PM)*

The PM methods focus on the comprehension of the graphical representation of a system/process and the retrieval of information for further analysis. The development of the process model is either created by the software engineer or provided by industry.

There are various PM methods used in industry such as Unified Modelling Language (UML)/System Modelling Language **(**SysML) Diagrams, Business Process Modelling Notation (BPMN), Computer Aided Design (CAD), Graphic User Interface (GUI), Piping and Instrumentations Diagrams (P&IDs), all able to map the structural and behavioural aspects of components/ activities in systems/ processes.

UML is a visual/graphical modelling language for engineering systems that was developed by the Object Management Group (OMG), International Council on Systems Engineering (INCOSE) and the Application Protocol 233 (AP233 consortium). UML diagrams can model the structure, behaviour and architecture of a system/process supporting the business process and data structure modelling. Additionally, UML can cope with model and data interchange via XMI and the evolving AP233.
In this paper, the UML (AD) has been chosen due to its high expressiveness, simplicity and directness. It is widely used in industry and facilitates the representation of the flow and the sequence from one activity to another capturing the dynamic behaviour of the engineering scenarios. The AD is created using Eclipse software, version 4.5 Mars, which is an open source Integrated Development Environment (IDE). Once the AD is validated successfully, using the 'Validate' option available in Eclipse, it is exported in XMI format, using the 'Export' option available in the Eclipse software, for further manipulation in step 2.

The XMI file includes the two elements which are necessary for the generation of the incidence matrix: the nodes and edges. The nodes correspond to the PN places, whereas the edges to the PN transitions. The XMI nodes are derived either from the AD initial/ final nodes or from the AD opaque action nodes (blocks in the Eclipse software). Similarly, the XMI edges are derived from the AD control flow edges (arcs in the Eclipse software).

Each XMI node element consists of the following attributes: a "type" that corresponds to the node used in the AD, an "id" that acts as a unique identifier, a "name" as presented in the AD, an "incoming" that corresponds to the edge id attribute that enters the node, and an "outgoing" that corresponds to the edge id attribute that leaves the node. Similarly, each XMI edge element consists of the following attributes: a "type" that corresponds to the edge used in the AD, an "id" that acts as a unique identifier, a "name" as presented in the AD, a "target" that corresponds to the node id attribute in which the edge ends up and a "source" that corresponds to the node id attribute from which the edge starts.

4.2 *Step 2 – Model transformation using Extensible Stylesheet Language Transformation (XSLT)*

The model transformation, conducted in this step, transforms the XMI (source model) into an XML (target model) file. The transformation of the XMI file into an XML format, suitable to be manipulated by MySQL Workbench, is carried out using XSLT that provides the ability to transform XML data from one format to another automatically. The XSLT, used as a template, define the rules that should be applied to the XMI file to generate the XML.

The nodes and edges elements from the XMI file are selected to create a well formed XML file following the XSLT rules. Two XSLT files have been developed. The first consists of two templates, applied to the XMI file, one for the XMI nodes asking for the "incoming", "name" and "outgoing" attributes and values, and one for the XMI edges asking for the "id", "name", "target" and "source" attributes and values. The XML file created consists of the attributes retrieved from the XMI as mentioned above. The XML attributes should be transformed into XML elements to be imported in the MySQL Workbench. Therefore, once the target XML file from the first transformation is generated, the second XSLT file is developed. The second transformation consists again of two templates which are applied to the XML file generated from the first transformation. Hence, a template is developed and applied to the XML nodes where the XML attributes of the nodes are transformed into XML elements. Similarly, a second template is developed and applied to the XML edges where the XML attributes of the edges are transformed into XML elements. Once, the XML transformations are completed the final XML file is developed consisting of the XML elements retrieved from the XML file generated form the first transformation.

The XML file is then ready to be loaded into the MySQL Workbench to be manipulated and organised in such a way that the transpose of the PN incidence matrix can be generated.

4.3 *Step 3 – Database modelling using MySQL-PN model*

A relatively recent development in the field of software engineering is the database concept that over the last 30 years has been used widely in industry (Connolly & Begg 2005). The main idea of the database is to capture and analyse data by organising it in a straightforward way enabling it to be accessed, managed and updated. The structure of a database is called 'schema' and acts as an abstract view of the data, allowing the user to have only the general description of the process/ system requirements. Once the database (schema) has been developed the user can store into the tables the process's/ system's data (values) associated with the system/process under consideration.

The MySQL (Michael Widenius Structured Query Language) database, one of the most popular open source relational databases, has been chosen for this work. The MySQL Workbench is a visual database design tool suitable for SQL development, data modelling, server administration and data migration. The relational databases avoid data duplication, provide consistent records and simple data manipulation, maintain security and enable the user to carry out complex queries.

The XML file, created from the XML transformations in step 2, is loaded into the MySQL Workbench and an SQL code is generated to manipulate and store the XML information into a transpose incidence matrix to present the PN model in a mathematical form.

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

1. Create a table in MySQL named 'node', inserting for each node sub-element the text values of "incoming", "name" and "outgoing" elements from the final XML file, created from the UML AD.
2. Create a second table in MySQL named 'edge', inserting for each edge sub-element the text values of "id" and "target" elements from the final XML file, created from the UML AD.
3. The SQL finds the edge sub-elements in the XML that have identical targets and sources text values and replaces the ids text values of the targets with the ids text values of the sources, creating a new table named 'decision'. This is performed for all the decision nodes included in the UML AD, The 'decision' table consists of the "id" and "name" columns, as derived from the 'edge' table (step 2) and the edge elements from the XML file.
4. Create a table, named 'new-edge', listing the ids text values from the combination of the 'edge' and 'decision' tables, created in steps 2 and 3, storing for each id the corresponding text value "name" as derived from the XML file.
5. Join the 'node' and 'new-edge' tables, developed in steps 1 and 4. The "incoming" and "outgoing" columns from the 'node' table in step 1 are replaced by the names as presented in the 'new-edge' table in step 4. The new table is named 'final'. The table created consists of three columns, the "name_source",

"name_activity" and "name_target" from which the matrix can be created.

6. Create a matrix with the columns defined by the activity names in the 2nd column of table 'final' created in step 5 and the rows defined by the entries in the 1st column of the 'final'. If a "name_activity" and "name_source" are in the same row in table 'final', then the value -1 should be put in the corresponding matrix cell.
7. Create a second matrix with the columns defined by the activity names in the 2nd column of table 'final' created in step 5 and the rows defined by the entries in the 3rd column of the 'final'. If a "name_activity" and "name_target" are in the same row in table 'final', then the value +1 should be put in the corresponding matrix cell.
8. Create the transpose of the incidence matrix combining tables created in steps 6 and 7.

5 CASE STUDY – PETRI NET GENERATION

A recycling IT asset process has been considered in order to demonstrate the methodology of the automated PN generation model described above.

5.1 *Step 1 – Process Modelling*

5.1.1 *Recycling IT asset process description & UML AD*

The recycling IT asset process focuses on the repair of electronic devices, mainly mobiles phones. Once the device enters in the process line, it can pass along one of the two paths, either refurbished or scrap. Considering the two paths, there are six different possible stages:

- Asset Track (AT): Device's information is introduced into the traceability system.
- Visual Inspection (VI): The physical condition of an asset is assessed.
- Functional Test (FT): A product is inspected by testing/ checking its functionality, including activities such as charger check, battery test, LCD screen check and resetting, ringing, vibration, microphone and speaker tests.
- Data Erasure (DE): Data is erased securely using specific licensed software.
- Cleaning and De-Labelling (CD): Refurbished assets are cleaned, any non-essential labels are removed from the device and a new label is placed.
- Repair (R): the repair is conducted only if it is considered economically viable.
- Strip and Scrap (SS): Failed devices are checked for any parts that can be salvaged and are then sent for secure destruction.

All stages can deal with only one device at a time except for the Data Erasure activity. Additionally, all the activities are carried out at the same physical location, i.e. on the computer, apart from the repair activity.

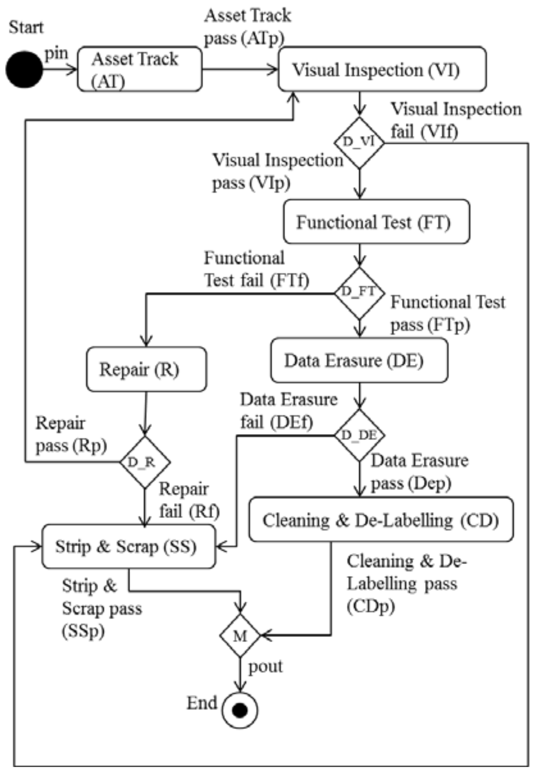


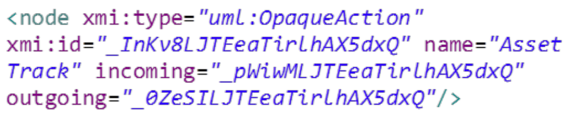
Figure 2. UML AD of the IT asset recycling process.

The repair takes place in the same factory, but it is performed away from the main refurbishment process and only takes place when there is a certain quantity of devices needing repair. The transportation of the repaired devices and the constraint of performing repair in batches create a large delay between the functional test and the repair activities. Similarly, the data erasure stage is performed separately and then the information is only logged in the process at the end. Each activity has a time to completion associated with it. There are also interval times between the activities.

Additionally, each activity has a probability of pass or fail, according to the process. The UML AD has been developed and validated successfully for the representation of all the paths included in the IT asset process, as shown in Figure 2.

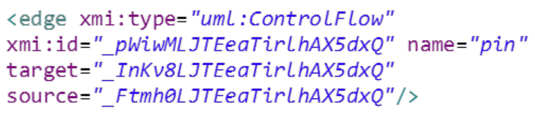
5.1.2 *AD to XMI format*

The UML AD created in section 5.1.1 for the IT asset process is exported in XMI format. The XMI file consists of the nodes such as the Start, Asset Track, Visual Inspection, End etc. as presented in Figure 2 and the edges such as the pin, ATp, VIp, VIf, pout etc. as presented in Figure 2. The "type", "id", "name", "incoming" and "outgoing" attributes of the 'Asset Track' node element are presented in Figure 3. Similarly the "type", "id", "name", "source" and "target" attributes of the 'pin' edge element are presented in Figure 4.



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

Figure 3. Node element in XMI format.



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

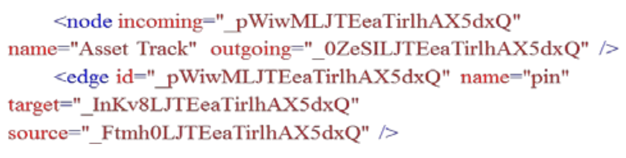
Figure 4.Edge element in XMI format.

5.2 *Step 2 – Model transformation using XSLT*

The two XSLT files, developed in this step, transform the XMI file into an XML format that can be loaded into the MySQL Workbench database environment in order to facilitate the PN incidence matrix generation for the IT asset recycling process.

Following the rules for the XSLT templates for the first XML transformation, described in section 4.2, the first XML file is created and part of it presented in Figure 5 for the 'Asset Track' node and the 'pin' edge. The node XSLT template retrieves the "incoming", "name" and "outgoing" attributes and values from the XMI, whereas the edge XSLT template retrieves the "id", "name", "target" and "source" attributes and values from the XMI file.

Additionally, following the rules for the development of the XSLT templates for the final XML, described in section 4.2, the final XML file is created and part of it presented in Figure 6 for the 'Asset Track' node and the 'pin' edge. The XSLT templates for the second transformation are applied to the first XML file. In this XSLT file the node/edge child element from the first XML file (node/edge) is transformed into node/edge root element. The attributes of the node/edge child element ("incoming", "name", "id" etc.) are then transformed into sub-elements (incoming, name, id, etc.) of the root node/edge XML element. The final XML file, part of which is presented in Figure 6 is loaded into the MySQL Workbench to generate the transpose of the PN incidence matrix for the IT asset process.

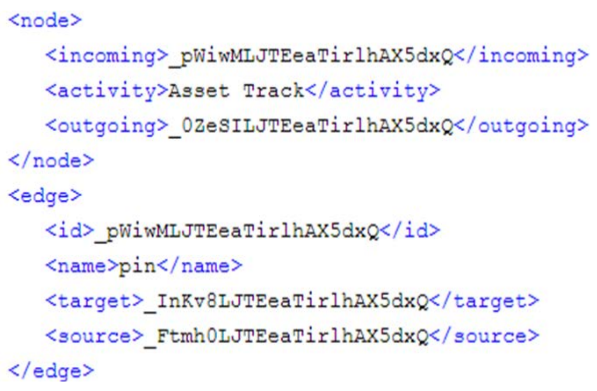


```
    <node incoming="_pWiwMLJTEeaTirlhAX5dxQ"
name="Asset Track"  outgoing="_0ZeSILJTEeaTirlhAX5dxQ" />
    <edge id="_pWiwMLJTEeaTirlhAX5dxQ"  name="pin"
target="_InKv8LJTEeaTirlhAX5dxQ"
source="_Ftmh0LJTEeaTirlhAX5dxQ" />
```

Figure 5. First XML format developed from the XMI using XSLT.

5.3 *Step 3 – MySQL database modelling*

The final XML file is loaded into the MySQL Workbench and an SQL code has been developed to



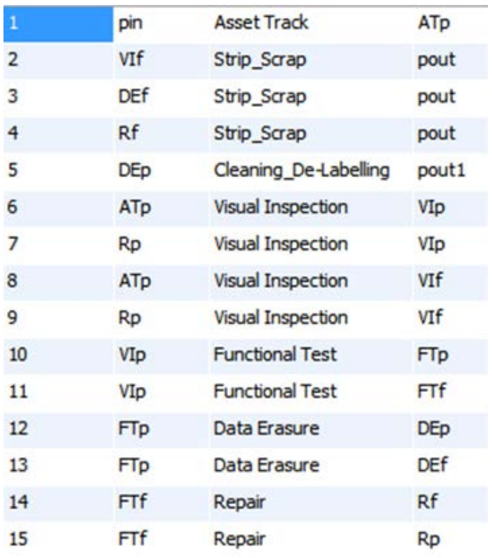
```
<node>
   <incoming>_pWiwMLJTEeaTirlhAX5dxQ</incoming>
   <activity>Asset Track</activity>
   <outgoing>_0ZeSILJTEeaTirlhAX5dxQ</outgoing>
</node>
<edge>
   <id>_pWiwMLJTEeaTirlhAX5dxQ</id>
   <name>pin</name>
   <target>_InKv8LJTEeaTirlhAX5dxQ</target>
   <source>_Ftmh0LJTEeaTirlhAX5dxQ</source>
</edge>
```

Figure 6. Final XML format developed from the first XML using XSLT.

generate the transpose of the PN incidence matrix for the IT asset process, following the steps defined in section 4.3. The steps were applied as follows:

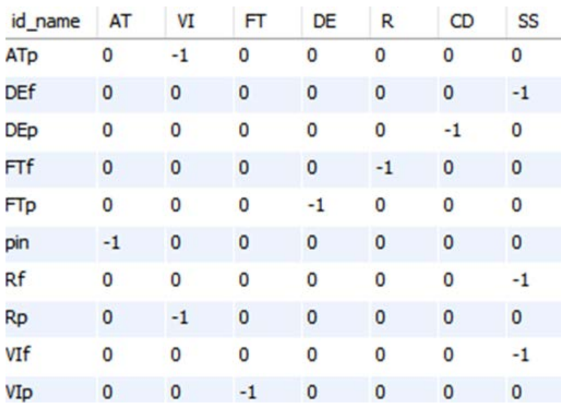
1. The 'node' table is created, using the values of the incoming, name and outgoing elements from the final XML file, as presented in Figure 6 for the 'Asset Track'.
2. The 'edge' table is created, using the values of the id and target elements from the final XML file, as presented in Figure 6 for the 'pin'.
3. The code finds in the XML file the edge sub-elements where the source and target values are the same and replace the id value that corresponds to the target with the id value that corresponds to the source. This is conducted for all the decision nodes. The 'decision' table then is created using the id derived from the 'edge' table and the name derived from the XML file.
4. The table 'new-edge' is created as described in section 4.3, step 4.
5. The 'final' table, presented in Figure 7, is created following step 5 as described in section 4.3. Each row of the 'final' table describes a transition with its activity name in the 3rd column, its input place in the 2nd column and its output place in the 4th column.
6. A matrix is generated listing the transitions (3rd column from the 'final' table) in the 1st row and the places (2nd 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 should be 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 add in the corresponding cell of the matrix

the value -1. The input matrix is presented in Figure 8.



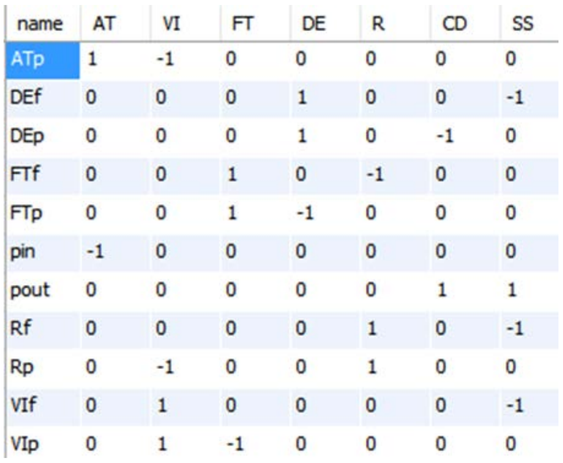
1	pin	Asset Track	ATp
2	VIf	Strip_Scrap	pout
3	DEf	Strip_Scrap	pout
4	Rf	Strip_Scrap	pout
5	DEp	Cleaning_De-Labelling	pout1
6	ATp	Visual Inspection	VIp
7	Rp	Visual Inspection	VIp
8	ATp	Visual Inspection	VIf
9	Rp	Visual Inspection	VIf
10	VIp	Functional Test	FTp
11	VIp	Functional Test	FTf
12	FTp	Data Erasure	DEp
13	FTp	Data Erasure	DEf
14	FTf	Repair	Rf
15	FTf	Repair	Rp

Figure 7. Final table as created from the SQL code in step 5.



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

Figure 8. Input 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

Figure 9. Overall PN incidence matrix for the IT asset process.

7. Similarly to step 6, a second matrix with the +1 values is generated using the 3rd and 4th columns from the 'final' table created in step 5.

8. The transpose of the overall PN incidence matrix for the IT asset recycling process is created by combining the matrices developed in steps 6 and 7, as shown in Figure 9.

5.4 *PN model generation*

The overall PN model for the IT asset process has been developed manually from the transpose of the incidence matrix values given in Figure 9 and is presented in Figure 10. The PN consists of 7 transitions and 11 places and 6 paths through the net have been identified for the IT asset process.

Each transition presented in the overall PN, Figure 10, consists of a sub-PN as can be seen in Figure 11. Figure 11 represents a generalised sub-PN for an 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) as well as the next activity places for the pass and fail paths respectively. Any activity can be represented by such a net.

For the 1st transition of the overall PN the device arrives place and immediate transition are added in the sub-PN. In general there are four cases for the sub-PNs as follows:

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

Hence, the incidence matrices for the sub-PNs can be developed given the above cases. Figure 12 shows the incidence matrix developed for the generalised sub-PN, presented in Figure 11. Due to lack of space the sub-PNs and their corresponding matrices for the overall PN in Figure 10 are omitted.

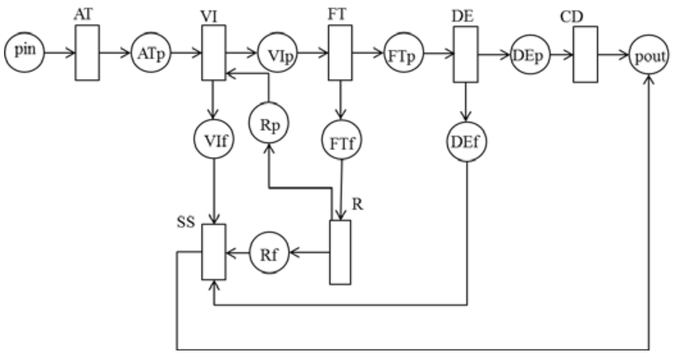


Figure 10. Overall PN model for the IT asset process.

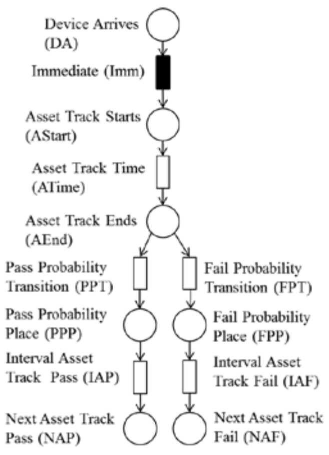
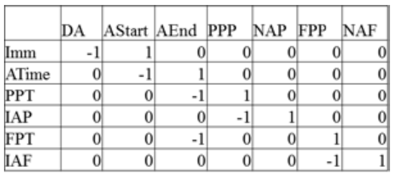

Device Arrives (DA)
Immediate (Imm)
Asset Track Starts (AStart)
Asset Track Time (ATime)
Asset Track Ends (AEnd)
Pass Probability Transition (PPT)
Fail Probability Transition (FPT)
Pass Probability Place (PPP)
Fail Probability Place (FPP)
Interval Asset Track Pass (IAP)
Interval Asset Track Fail (IAF)
Next Asset Track Pass (NAP)
Next Asset Track Fail (NAF)

Figure 11. Generalised sub-PN model.



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

Figure 12. Incidence matrix for the generalized sub-PN model.

6 CONCLUSIONS

In this paper, an automated PN generation methodology has been proposed. The methodology has been applied to an IT asset process where the transpose of the PN incidence matrix has been generated automatically from a UML AD. The methodology follows three main steps: the development of a UML AD for the process description, model transformations using the XSLT and database modelling using SQL code.

Future work involves the application of the automated PN generation methodology in complex processes to verify its applicability. Also, the automated graphical representation of the PN is currently being investigated. Additionally, an algorithm, written in Java, is being developed to simulate the process using the generated incidence matrix with the aim to identify possible limiting factors of the process and make recommendations for the improvement of the process's efficiency. Currently this algorithm has been applied to the case study and initial results obtained for the average times for each transition, presented in Figure 10, have been estimated. From the results the Repair activity (R in Figure 10) takes the longest time, 15879.15 seconds, to be completed, since it takes place in a different physical location from the other activities.

7 ACKNOWLEDGEMENT

The research reported in this paper aligns to the work being researched as part of the EPSRC grant EP/K014137/1.

8 REFERENCES

Bernardi, S., Donatelli, S. & Merseguer, J. 2002. From UML sequence diagrams and statecharts to analysable Petri net models. *Proceedings of the Third International Workshop on Software and Performance (WOSP2002),* Rome, Italy, ACM (2002) 35-45.

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

Eclipse. 2015. http://www.eclipse.org.

Lanus, M., Yin. L. & Trivedi, K.S. 2003. Hierarchical composition and aggregation of state-based availability and performability models. *IEEE Transactions on Reliability*, 52(1): 44-52.

Lapp, S.A. & Powers, G.J. 1977. Computer-aided Synthesis of Fault Trees. *IEEE Transactions on Reliability,* 26(1): 2-13.

Li. S. & Li, X. 2014. Study on generation of fault trees from Altarica models. *Procedia Engineering*, 80(1): 140-152.

Majdara, A. & Wakabayashi, T. 2009. Component-based modeling of systems for automated fault tree generation. *Reliability Engineering and System Safety, 94*(6): 1076-1086.

OMG Unified Modeling Language (OMG UML), Superstructure. 2011. OMG Systems Modeling Language (OMG SysML), Version 2.4.1, formal/2011-08-06.

Papadopoulos, Y. & Grante, C. 2005. Evolving car designs using model-based automated safety analysis and optimisation techni*ques. The Journal of Systems and Software,* 76(1): 77-89.

Petri, C.A. 1962. *Kommunikation with Automaten*. English Translation, 1966: *Communication with Automata,* Technical Report RADC-TR-65-377, Rome Air Dev. Center, New York.

Rauzy, A. 2002. Mode automata and their compilation into fault trees. *Reliability Engineering and System Safety,* 78(1): 1-12.

Roth, M., Wolf, M. & Lindemann, U. 2015. Integrated matrix-based Fault tree generation and evaluation. *Procedia Computer Science*, 44(1): 599-608.

Salem, S.L., Apostolakis, G.E. & Okrent, D. 1977. A new methodology for the computer-aided construction of fault trees. *Ann. Nucl. Energy,* 4(9-10): 417-433.

Stockwell, K.S. & Dunnett, S.J. 2013. Automatic construction of a reliability model for a phased mission system. In Jackson, L.M., and Andrews, J.D. (eds), *Proceedings of the 20th Advances in Risk and Reliability Technology Symposium*, 192-204. Loughborough University, Leicestershire.

Taylor, J.R. 1982. An algorithm for Fault Tree construction. IEEE *Transactions on Reliability,* R-3(2): 137-146.

Wang, J. 2006. Petri nets dynamic event-driven system modelling. In Paul Fishwick (eds), *Handbook of Dynamic System Modeling*: 1-17. CRC Press.

Xie, G., Xue, D. & Xi, S. 1993. Tree-Expert: A tree based expert system for fault tree construction. *Reliability Engineering and System Safety,* 40(1):295-309.

Zhao, C., Bhushan, M. & Venkatasubramanian, V. 2005. PHASUITE: An automated HAZOP analysis tool for chemical processes: Part I. Knowledge Engineering Framework. *Process Safety and Environmental Protection,* 83(B6): 509-532.

Zille, V., Bérenguer, C., Grall, A. & Despujols, A. 2010. 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, 12(1):253-272. London: Springer London.