Modelling and Simulation (COMP-522A), Project Report.

Multi Paradigm Modelling and Simulation for Engineering Change Management (ECM), using IDEF0, IDEF3, State Transition(ST) Petri Nets and Colored Petri Nets (CPN)

December , 2005

Imran Haider Malik, Imran Majid

imran.h.malik@gmail.com, majidimran@gmail.com

McGill University,

Montreal, Quebec, H3A 2K6, Canada

Dedicated to

our parents ..!

Acknowledgements

The motivation, help and insight to certain critical issues provided by Prof. Hans Vangheluwe (Instructor of the course COMP-522A), is remarkable. We are also thankful to the other students in our class for making the environment of learning in class more healthy with their presence. Last but not the least we acknowledge the help of Muhammad Ali Elhani (Student at Ecole Technologie Superiore (ETS), Montreal, Canada), in providing us the IDEF0 model of New Part Planning Process at an aerospace manufacturing company.

Abstract

Process planning in manufacturing environment is challenging since its iterative, due to engineering changes (ECs). In order to track the impact of engineering changes the modelling and simulation of information flow process with regard to New Part Planning (N.P.P), is required to be done at different level of abstraction, using different formalisms. This is necessary since the logic of impact trajectory may be seen in one of the lower most levels of abstraction, while the effect of them can be put into interpretable result in another view, with higher level of abstraction. This is a typical situation when the part dimensions are changed by the Engineering Design Department for instance, which effect the process plan with regard to machines chosen to manufacture the part and the time they are booked for those operations. However, the management of the organization might not be interested in the change in dimensions of the part or sub assembly or assembly; they might be interested in the impact the change has in terms of capital to be invested and the departments that are effected.

For this purpose we have transformed the truncated IDEF0 model of New Part Planning (N.P.P) process into first IDEF3, then directly to State Transition Petri Nets and Colored Petri Nets.

1 Introduction

1.1 Problem Statement

There exists an IDEF0[1] (Elhani et al., 2005) model of the information flow process, of New Part Planning (N.P.P), in the software `AI0WIN7`(for IDEF0)[2]. For the initial study purposes and to obtain the feel of different modelling techniques, we have selected one activity A11 (forms First Level of Hierarchy, FLH) and its two decompositions (The IDEF0 of these activity and their sub activities is included in the Appendix A). The first decomposition of A11 (Second Level of Hierarchy, SLH), contain six activities, out of which five of them have further decomposition, which leaves A116 without any further decomposition. The second decomposition of five activities namely A111, A112, A113, A114 and A115 in SLH, contain 3, 3, 3, 5 and 3 activities in them respectively (All five decompositions are displayed in Appendix A). The activities are selected in a way that they involve all sorts of relationships among the flow e.g. join and split.
It is desired to translate the reduced model into a formalism containing dynamics behavior since IDEF0 is a formalism with static behavior which signifies only the logical sequence of operations. The need is to elaborate on the dynamic behavior of the system and explain some characteristics which occur in real world, with regard to engineering changes.

1.2 Proposed Solutions

In order to bring the process level view of the reduced IDEF0 model, we will use IDEF3 formalism in which time information is additionally required other than the information contained in IDEF0 (IDEF3 process modelling formalism is detailed in Mayer et al., (1995)). However, the approach of Kim, C-H et al (2001) will be used to transform the IDEF0 to

[1]Draft Federal Information Processing Standards Publication 183, 1993 December 21, Announcing the Standard for INTEGRATION DEFINITION FOR FUNCTION MODELLING (IDEF0)., Knowledge Based Systems Inc. (KBSI), Texas. http://www.kbsi.com/

[2]Product of Knowledge Based Systems Inc. (KBSI), Texas. http://www.kbsi.com/

IDEF3, which transforms the IDEF0 to IDEF3 based on the assumption that the resources and mechanisms in IDEF0 are time independent and thus can be neglected while transformation to IDEF3. IDEF3 simulation will be done using `ProSim`[3] if possible on its evaluation version. In case the simulation is not possible, we will translate the IDEF3 process model to State Transition Petri nets and analyze the dead locks in the system. Another proposed approach is that the IDEF3 be transformed to Colored Petri Nets (CPN), the syntax and semantics of which are described by Jensen, K. (1996), from where we will have a compact and more comprehensive view of the actual reduced IDEF0 model.

2 IDEF0 model of N.P.P

2.1 Basic Syntax and Semantics of IDEF0 and IDEF3

The syntax of IDEF0 involves a box which represents a function Fig. 1. It is characterized by a square box with a name representing the function. There are ICOM (Input, Control, Output and Mechanism) links used to describe the relation among the functions. Inputs always are represented by the right headed arrow entering from the left side of the function block. Outputs are leaving with right headed arrow from right side of the function block. Controls are the vertical arrows entering from top of the box. Finally the mechanisms are the vertical arrows facing upwards from lower side of the box.

The semantics of the syntax described above is that when ever the inputs are available and control conditions are satisfied, with the resources (represented by the mechanism) available to the function, the outputs will be generated.

Similar to IDEF0, since IDEF3 is a process modelling formalism, the processes are represented by the Unit Of Behaviors (UOB) Fig. 1. These UOBs have a box with two small boxes at their lower end. In the upper big box the name of the process is mentioned. In the lower two boxes the number of the UOB with regard to its hierarchy is represented. These UOBs are linked to each other via links and junctions. UOBs cannot be directly linked to

[3]Product of Knowledge Based Systems Inc. (KBSI), Texas. http://www.kbsi.com/

each other via a single link, there has to be a junction in between describing the logic of start of the process downstream. There are two types of links one is single headed and the other is double headed. The single headed arrow represent that once the process(or junction has satisfied the condition) at its tail is finished, the process or junction at its head will initiate itself without physical transfer of an object. The double headed arrow represent that once the process(or junction has satisfied the condition) at its tail is finished, the process or junction at its head will initiate itself with physical transfer of an object. The logical junctions are of three types OR, AND and XOR, with the option of either being synchronous or asynchronous. Synchronous and Asynchronous property of junction is represented by the vertical line on the right side of junction box being there or not. The one in the Fig. 1 is synchronous AND which means that when all the inputs are satisfied at one time, only then all the outputs will be generated at the same time and vice versa.

The semantics of IDEF3 syntax described above is that when ever the conditions on the input of junction is satisfied the output of a junction is generated and the subsequent processes start there activity. Once the time of the process (which is also part of syntax of IDEF3 process modelling) is finished it will either trigger the junction down streams or trigger that junction by sending and object to it.

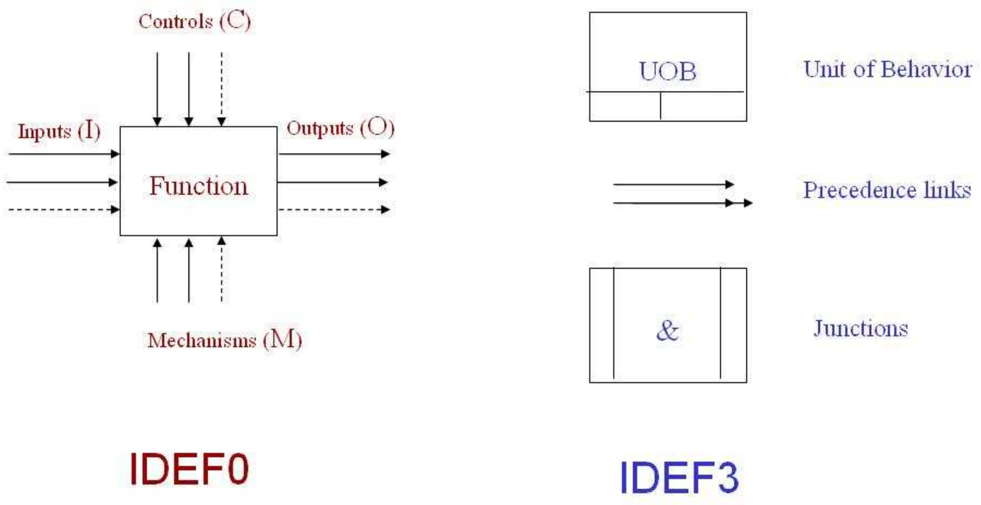


Figure 1: Components of Syntax in IDEF0 and IDEF3.

3 Transformation of Truncated IDEF0 to IDEF3

This transformation is proposed since we are required to introduce some functional dynamics in the system modelled by IDEF0, in order to realize the change trajectory (equivalent to state change trajectory). The comparison of the IDEF0 and IDEF3 is summarized in the Critique section of the report.

3.1 Manual Transformation of IDEF0 to IDEF3

The IDEF0 is first Manually Transformed to IDEF3 using the transformation strategy and assumptions described in Kim et al. (2001). These assumptions namely are

- all the Mechanisms (resources) are available to functions or now in IDEF3 case UOBs, at all times, and thus can be removed from the model,
- all the Control(conditions) on the functions are satisfied and thus can be removed from the model.

Obviously these assumptions are a little too naive. However for the purpose of study in this project of multi paradigm modelling for ECM we go along with it.
The transformation rules are reproduced here for a quick reference

- remove all the Mechanisms,
- remove all the Controls,
- replace functions by the UOBs and assign the same name to UOBs as the function name along with the activity number,
- The join and split of the links are transformed by the table used in Kim et al. (2001), and reproduced in Fig. 3.

It is assumed further here that no Object flow is there between the function, thereby resulting in the links used to join UOBs to be single headed arrows. All the split and join junctions

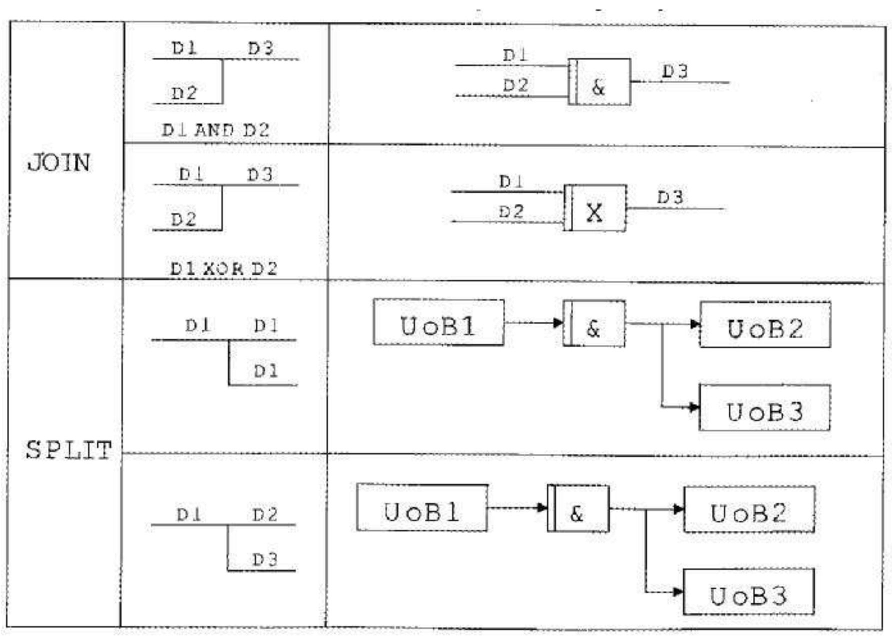
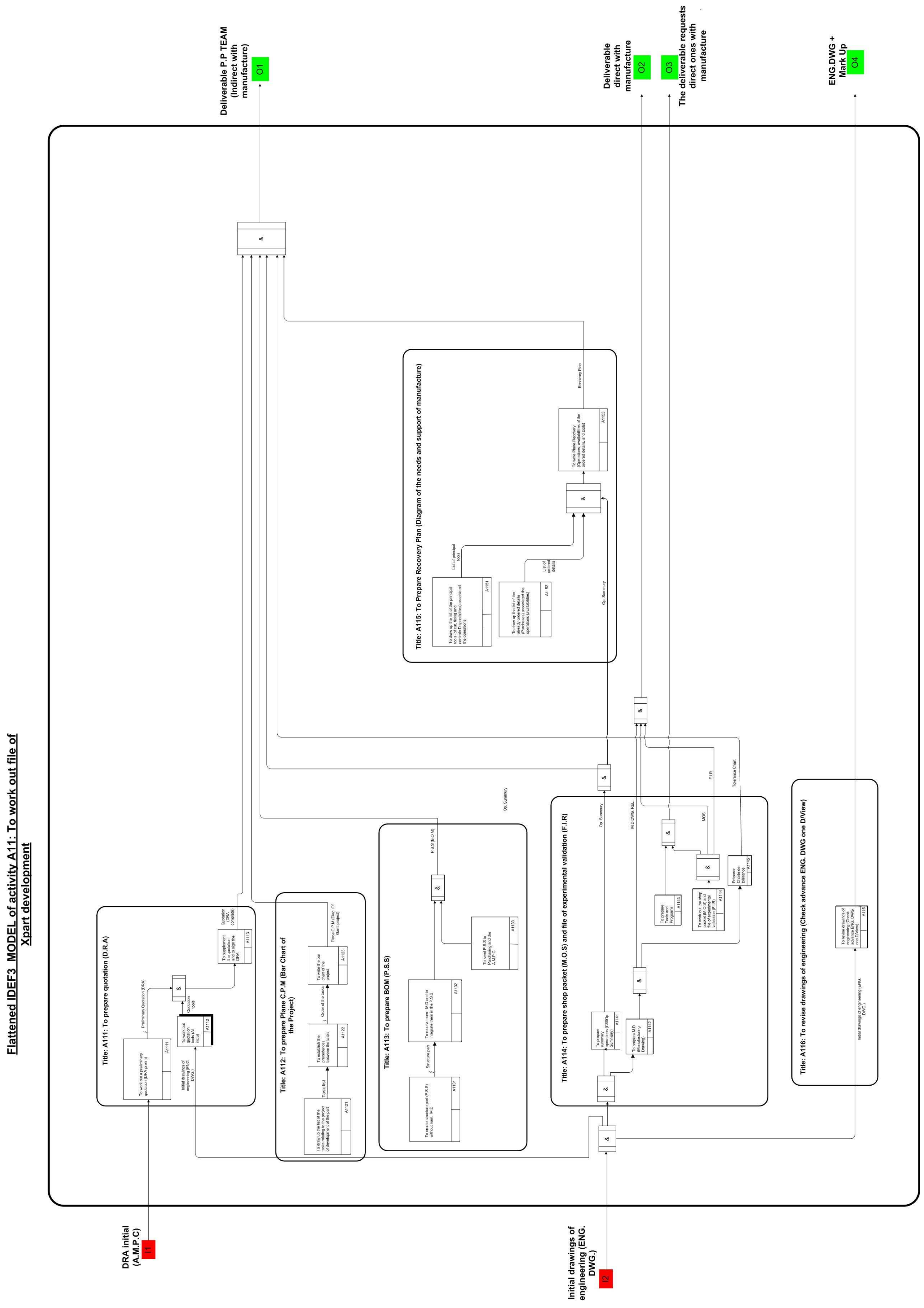


Figure 2: Transformation of join and split operations.

are assumed to have synchronous AND logic. The transformed IDEF3 model of truncated IDEF0 model (A11 activity, with 3 level of hierarchy), is drawn using `MSVisio` for better presentation. Since `AIOWIN7` allow transformation of IDEF0 model in it to be exported in `MSVisio`, couple of activity diagrams are directly used from the exported files. The resultant diagram is displayed in Fig. 3.

3.2 Semi Automated Transformation of IDEF0 to IDEF3

As `AIOWIN7` and `ProSim`(for IDEF3) are the both the product of KBSI, the IDEF0 functions along with their hierarchy can be exported from `AIOWIN7` to a `*.txt` file which then can be imported in `ProSim`. When this imported model is accessed in `ProSim`, the second level of Hierarchy (SLH) appears to six UOBS without any link between them. Since only the evaluation version was at our disposal and we were not provided with the full functionality of adding the links and junctions only some of UOBs of SLH are connected with each other via links and junctions, using the same assumptions enumerated in the last section. However,



Flattened IDEF3 MODEL of activity A11: To work out file of Xpart development
Deliverable P.P TEAM (Indirect with manufacture)
O1
Deliverable direct with manufacture
O2
The deliverable requests direct ones with manufacture
O3
ENG.DWG + Mark Up
O4
DRA initial (A.M.P.C)
I1
Initial drawings of engineering (ENG. DWG.)
I2
Title: A111: To prepare quotation (D.R.A)
Title: A112: To prepare Plane C.P.M (Bar Chart of the Project)
Title: A113: To prepare BOM (P.S.S)
Title: A114: To prepare shop packet (M.O.S) and file of experimental validation (F.I.R)
Title: A115: To Prepare Recovery Plan (Diagram of the needs and support of manufacture)
Title: A116: To revise drawings of engineering (Check advance ENG. DWG one DView)
&
A1111
A1112
A1113
A1121
A1122
A1123
A1131
A1132
A1133
A1141
A1142
A1143
A1144
A1145
A1151
A1152
A1153
A116
Preliminary Quotation (DRA)
Quotation tools
Task list
Order of the tasks
Structure part
P.S.S (B.O.M)
Op. Summary
M.D DWG. REL
MOS
F.I.R
Tolerance Chart
List of principal tools
List of ordered details
Recovery Plan

Figure 3: Flattened IDEF3 of Truncated IDEF0's A11 activity.

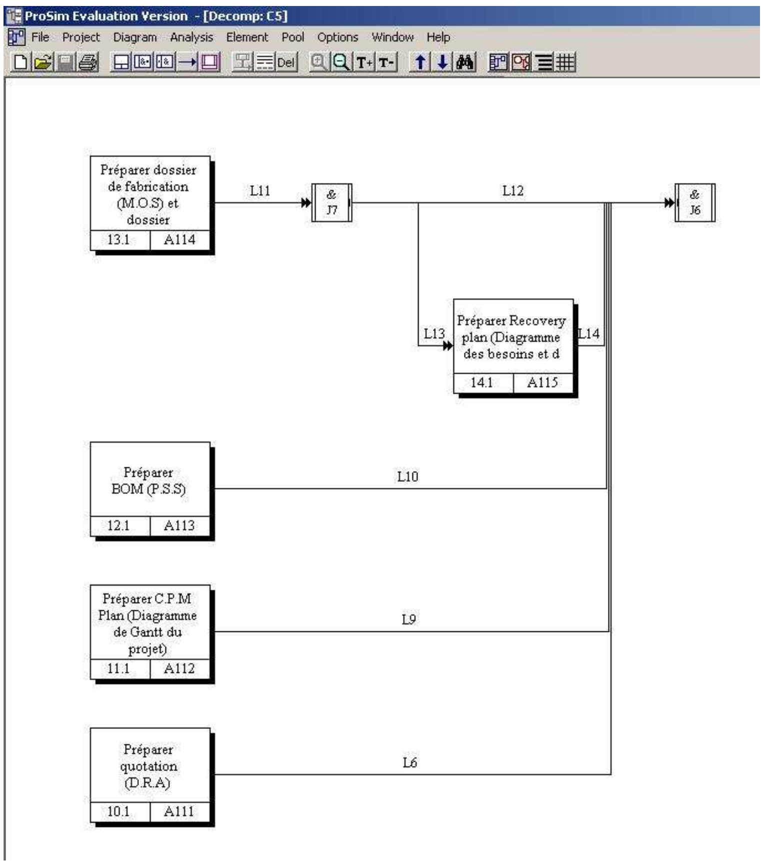

ProSim Evaluation Version - [Decomp: C5]
File Project Diagram Analysis Element Pool Options Window Help
Préparer dossier de fabrication (M.O.S) et dossier
13.1 A114
L11
& J7
L12
& J6
L13
Préparer Recovery plan (Diagramme des besoins et d
14.1 A115
L14
Préparer BOM (P.S.S)
12.1 A113
L10
Préparer C.P.M Plan (Diagramme de Gantt du projet)
11.1 A112
L9
Préparer quotation (D.R.A)
10.1 A111
L6

Figure 4: SLH in ProSim

to fulfill our curiosity of using the precedence links with object transfer, we used the double headed arrow and in its property added the object available in the UOB at its tail (either which was from an input, output, mechanism and/or control of the function represented by the UOB). Since the links and junctions are not automatically configured by `ProSim`, the transformation is named as Semi-Automated. Fig. 4 display the view of SLH in `ProSim`, evaluation version.

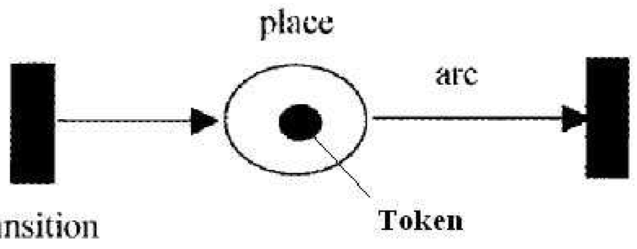


Figure 5: Basic Components of State Transition Petri Net.

4 IDEF0 to State Transition (ST) Petri Nets

4.1 Basic Syntax and Semantics of State Transition Petri Nets.

The ST Petri nets (Cassandras,1989) Fig. 5 consists of four different objects for their representation, which are namely *places*, *transitions*, *tokens* and *arcs*. Places model the storage place, transition model an action or event that happens when the transition fire, tokens model similar objects flowing in the system and arcs model links between the places to/from transitions.

The arcs are characterized by their weight which is an integer number. The arcs are used to join the places to transitions and transitions to places. Places are characterized with respect to the transitions they are connected, as either input or output places. No two places are connected to each other directly.

When the input places have sufficient number of tokens as the arcs joining them to the transition, the transition can be *fired* and tokens equal to the weight of arcs corresponding to the output places will be added to them. Initially a Petri net may have certain number of tokens placed in some places. There is a characteristic of Petri net called marking, which is a n dimensional vector, where n is number of places in the Petri net. Whenever a transition fires the marking of a Petri net is updated.

4.2 IDEF0 XML Structure

AI0WIN7 stores IDEF0 structure in XML format. This XML structure not only maintains different connections between Activities but also stores the hierarchy of the system. Our first aim was to be able to interpret and flatten this model. The involvement of the hierarchy makes the model quite intricate and difficult to interpret. Section. 4.2.1 and 4.2.2 represent the structure of IDEF0 in this XML format. State Transition Petri Nets (Cassandras, 1989).

4.2.1 Activity and Concept

Following code in verbatim represents the activity and concept pool structure in the XML model of IDEF0.

```
<?xml version="1.0"?> <idef0>
  <activity_pool>
    <activity id="25">
      <name> Developper methode de fabrication piece moteur </name>
    </activity>
    <activity id="28">
      <name> elaborer methode de fabrication XPart </name>
    </activity>
               . . .
    <activity id="2200">
      <name> Recevoir M.O.S/F.I.R (Electr. + papier) </name>
    </activity>
  </activity_pool>
  <concept_pool>
    <concept id="1867">
      <name> Livrables P.P Team (Indirects a la fabrication) </name>
    </concept>
    <concept id="1868">
      <name> P.S.S (B.O.M) </name>
    </concept>
               . . .
    <concept id="3258">
      <name> Nouvelle demande de M.O.S/F.I.R (Dossier papier) </name>
    </concept>
  </concept_pool>
```

In IDEF0, an activity represents the processing block or ICOM. Every activity has a unique id and a name. `AIOWIN7` records all the activities in an activity pool. A concept in IDEF0 represents an input (I), output (O), control (C) or mechanism (M). These are the connections between the activities. In `AIOWIN7` every concept is also assigned a unique id and is represented on the model with a name.

4.2.2 Diagram

The diagram structure in IDEF0 is as follows.

```
<diagram id="78" parent_activity_use="49">
    <name> A112: Preparer C.P.M Plan (Diagramme de Gantt du projet) </name>
    <revision_no> 0 </revision_no>
    <review_stat> Working </review_stat>
    <activity_use id="80" pool_id="79">
    <placement_info x_coord="1" y_coord="1"/>
    <attachment file_link="          "/>
    </activity_use>
    <activity_use id="82" pool_id="81">
    <placement_info x_coord="2" y_coord="2"/>
    <attachment file_link="          "/>
    </activity_use>
```

```
<activity_use id="84" pool_id="83">
<placement_info x_coord="3" y_coord="3"/>
<attachment file_link="           "/>
</activity_use>
<concept_use id="2239" pool_id="1923" breakdown_id="0">
<attached_activity icom_type="Control" attached_activity_use_id="80"/>
<attached_activity icom_type="Control" attached_activity_use_id="82"/>
</concept_use>
<concept_use id="2256" pool_id="1937" breakdown_id="0">
<attached_activity icom_type="Control" attached_activity_use_id="80"/>
<attached_activity icom_type="Control" attached_activity_use_id="82"/>
</concept_use>
<concept_use id="2262" pool_id="2056" breakdown_id="0">
<attached_activity icom_type="Control" attached_activity_use_id="80"/>
<attached_activity icom_type="Control" attached_activity_use_id="82"/>
</concept_use>
<concept_use id="2468" pool_id="1872" breakdown_id="2574">
<attached_activity icom_type="Output" attached_activity_use_id="84"/>
</concept_use>
<concept_use id="2321" pool_id="1868" breakdown_id="0">
<attached_activity icom_type="Control" attached_activity_use_id="80"/>
</concept_use>
<concept_use id="2497" pool_id="1936" breakdown_id="2572">
<attached_activity icom_type="Control" attached_activity_use_id="80"/>
<attached_activity icom_type="Control" attached_activity_use_id="82"/>
</concept_use>
<concept_use id="2348" pool_id="2033" breakdown_id="0">
<attached_activity icom_type="Mechanism" attached_activity_use_id="84"/>
<attached_activity icom_type="Mechanism" attached_activity_use_id="82"/>
</concept_use>
```

```
    <concept_use id="2439" pool_id="2438" breakdown_id="2574">
    <attached_activity icom_type="Output" attached_activity_use_id="80"/>
    <attached_activity icom_type="Input" attached_activity_use_id="82"/>
    </concept_use>
    <concept_use id="2441" pool_id="2440" breakdown_id="0">
    <attached_activity icom_type="Output" attached_activity_use_id="82"/>
    <attached_activity icom_type="Input" attached_activity_use_id="84"/>
    </concept_use>
    <breakdown id="2572" parent_concept_use_id="2497" branch_type="Split"
    bundle_type="Type-Of"/>
    <breakdown id="2574" parent_concept_use_id="2468" branch_type="Join"
    bundle_type="Type-Of"/>
</diagram>
```

In IDEF0 the interconnections between activities and their hierarchy is maintained in terms of diagram modules. A diagram is linked with a particular activity and represents how that activity is broken down into further lower-level activities. Every diagram has a unique id and has a property `parent_activity_use`. The hierarchy is maintained through
`parent_activity_use` attribute. If it is empty then it represents that this diagram is the top most diagram in the hierarchy; otherwise it points to the activity it elaborates. In the XML diagram structure above, we have a diagram with id=78 and `parent_activity_use`=49. This means that whenever an activity is used in some figure with id=49 then in fact that activity can be elaborated further with this diagram. Every diagram has a name which is shown in `AI0WIN7`. Based on the name field it can be inferred that the generated code in XML above represents activity A112.
Every diagram elicits the lower-level activities that are taking part in this diagram. These activities constitute the breakdown of the higher-level parental activity.

```
<activity_use id="80" pool_id="79">
    <placement_info x_coord="1" y_coord="1"/>
    <attachment file_link="           "/>
</activity_use>
```

The above XML snippet represents that an activity that is stored in activity pool with id=79 is being used in the above diagram. Within the scope of this diagram this particular activity is referenced by its `activity_use` id=80.
Diagram represents the connections (ICOM) between the lower-level activities by the field `concept_use`. This represents the concept from the concept pool that is being used in this diagram.

```
<concept_use id="2439" pool_id="2438" breakdown_id="2574">
    <attached_activity icom_type="Output" attached_activity_use_id="80"/>
    <attached_activity icom_type="Input" attached_activity_use_id="82"/>
</concept_use>
```

The above XML snippet represents that a concept (which is stored in concept pool with id of 2438) is being used in this diagram. This concept is attached to two activities used in this diagram. It connects to the first activity (an activity which is referred with id=80 in this diagram) as an "Output" and connects to the second activity (an activity which is referred with id=82 in this diagram) as an "Input".

4.2.3 Join and Split

A concept can be split and connected to several activities with different names. Similarly several concepts can merge or join to form some other concept. This is accommodated in IDEF0 through the breakdown. Field in `concept_use` A concept that is taking part in some split or join has some value in the `breakdown_id`; otherwise it is empty.

```
<breakdownid="2572"parent_concept_use_id="2497"branch_type="Split"
                  bundle_type="Type-Of"/>
```

```
<breakdown id="2574"parent_concept_use_id="2468"branch_type="Join"
                      bundle_type="Type-Of"/>
```

The above XML snippet represents that all the concepts that use a `breakdown_id` of 2572 are all 'Split" from the concept 2497 (given in `parent_concept_use_id`). Similarly all the concepts using the `breakdown_id` of 2574 are all joining into the concept of 2468.

4.3 Object Oriented (OO) Representation of IDEF0 Structure

For transforming IDEF0 in State Transition Petri Nets and Colored Petri Nets, we first have to read the XML format of IDEF0 explained in section. 4.2 and parse it into an Object-Oriented model. Fig. 6 represents an overview of the Class diagram used to build the object-oriented model. In the class diagram, the class Arc represents the Concept in IDEF0 while Activity and Diagram of IDEF0 are represented by their corresponding classes. The class diagram represents that every Activity is expanded in 0 or 1 diagram. If the activity is further broken down into sub activities then its expanded diagram is non-empty otherwise it is empty. Every activity has 4 associations with the class `ArcList` corresponding to Inputs, Controls, Outputs and Mechanisms of that activity. Every diagram has a list of activities and arcs that take part in that diagram. As explained in Section 1.3, a concept can be split from some other concept or it can be joining into some other concept. To represent this relationship we have `mergesInto` and `splitsFrom` associations with class Arc. We use `org.xml.sax` API of Java to parse the XML model of IDEF0 and construct the corresponding object-oriented model in memory.

4.4 Mapping IDEF0 to ST Petri Nets

We have mapped the flattened IDEF0 model into ST Petri Nets by defining some basic transformations and devising the connection mechanism to connect these basic modules. As we are assuming that the resources are continuously available and the control conditions are always satisfied so we concentrate on the Input, Output and the Activities of IDEF0. These transformations are given in Fig.7. An input is modelled as a combination of Place and Transition such that we have continuous availability of Input. An Activity without any inputs is modelled similarly. Any Activity that has

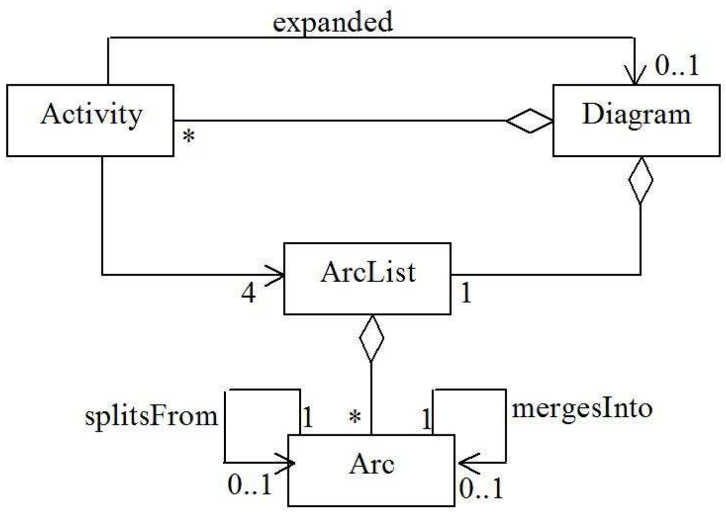


Figure 6: Class Diagram of IDEF0 model

some inputs is modelled as a combination of input-transition, a place and an output-transition. An output is represented as a place that receives input through its input-transition. In ST Petri Nets we cannot have edges from transitions to transitions therefore we need connecting places to connect these basic modules. These connecting places are represented in Fig. 8. In the current approach we have to produce a connecting place for every connection between activities, input and outputs. There are several heuristics that can be used to reduce the number of connecting places. For instance if we connect two Activities serially then the connecting place can be removed by merging the output-transition of first activity with the input-transition of the second activity. Currently, due to time constraints, we have not been able to implement such heuristics but these can be considered a part of the future work.

4.5 Generic XML representation of ST Petri Nets

We are required to convert IDEF0 into State Transition Petri Nets (ST Petri Nets) and analyze its behavior. As we are using ATOM3[4] to analyze ST Petri Nets we were initially converting the

[4]Multi-Formalism modelling and Simulation Package, Product of Modelling, Simulation and Design Lab. (MSDL), Department of Computer Science, McGill University, Montreal, Canada.

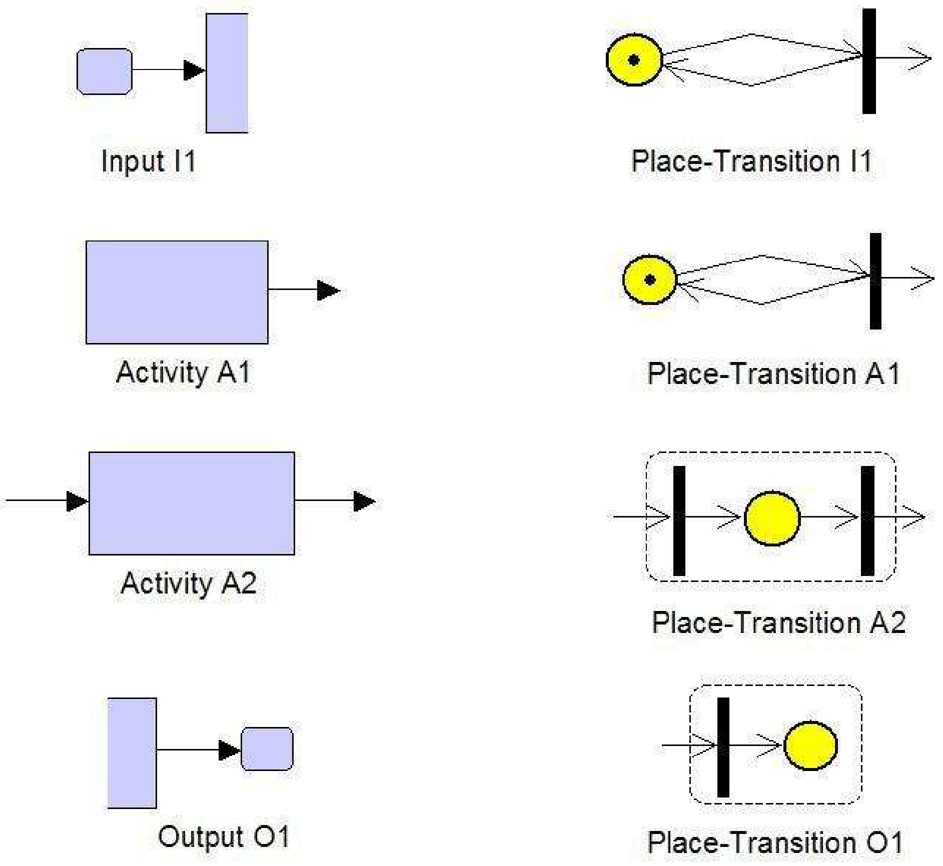


Figure 7: Basic transformation from IDEF0 to ST Petri Nets

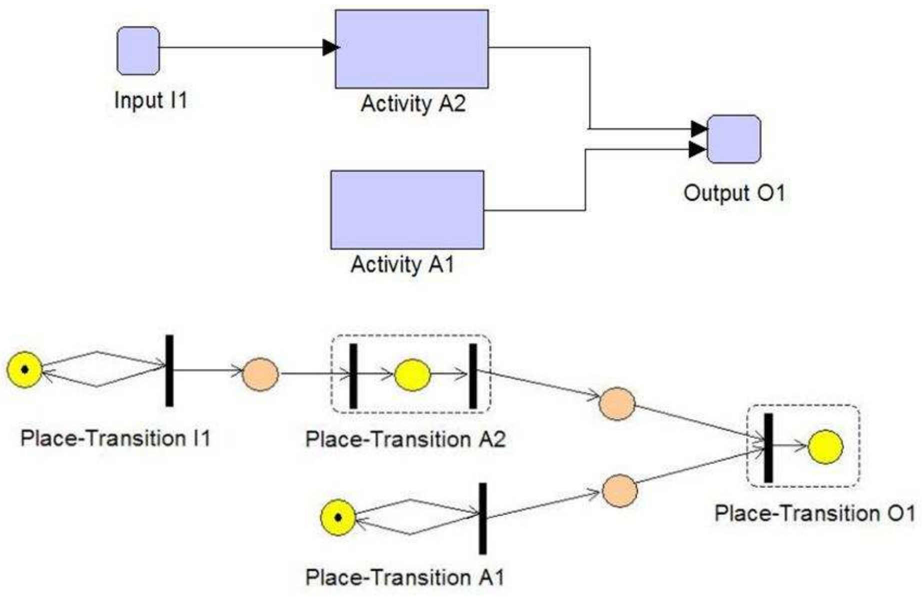


Figure 8: Representation of connecting places.

object-oriented model of IDEF0 directly into ATOM3 as represented in Fig. 9. Mapping one format into another format is generally a difficult process and it can be efficiently managed by introducing an intermediary format. To achieve this purpose we have devised an XML format for representing State Transition Petri Nets in XML format. We convert our object-oriented model into this XML format and from this XML format we generate the target ATOM3 code. This process is displayed in Fig. 10. Even though we are performing two transformations instead of one in the previous case, this is a much more generalized solution. It also facilitates any other third party to write ST Petri Nets in our devised XML format and out transformer can build the corresponding ATOM3 related Python code for it too.

4.5.1 XML representation of ST Petri Nets

The XML DTD used to represent the ST Petri Nets in XML format is given as follows.

```
<!ELEMENT PetriNet (place*)>
<!ELEMENT place (input, output)>
```

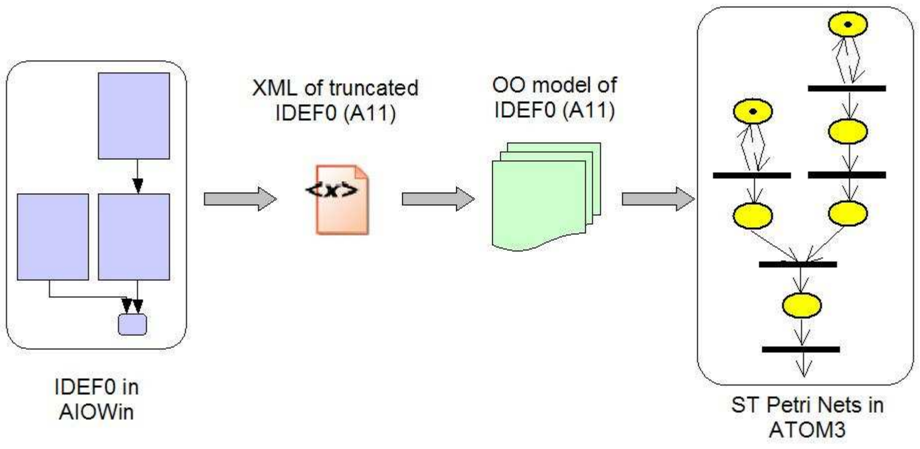


Figure 9: Initial transformation Process

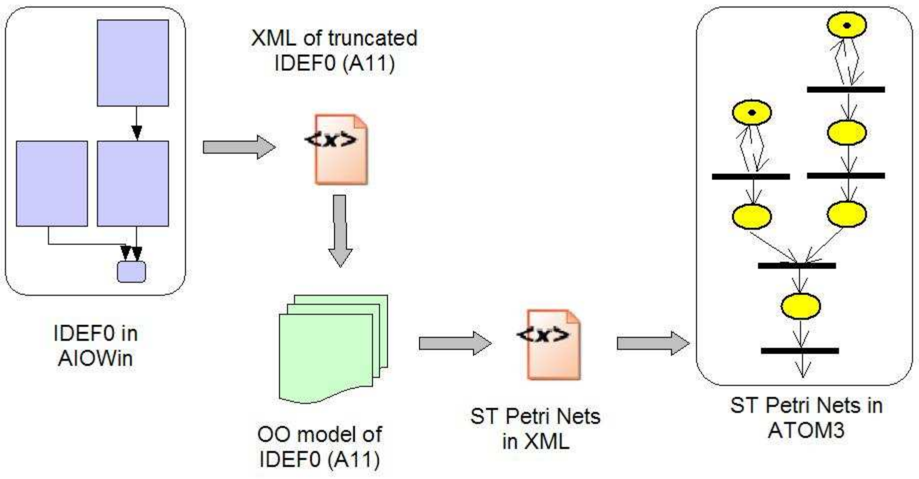
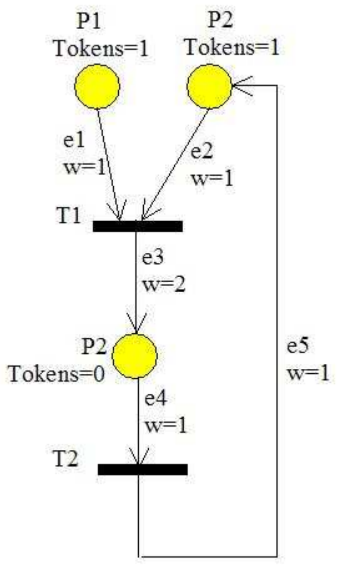


Figure 10: Final transformation Process.

```
<!ATTLIST place
    name CDATA #REQUIRED
    tokens CDATA #REQUIRED>
<!ELEMENT input (edge*)>
<!ELEMENT output (edge*)>

<!ELEMENT edge()>
<!ATTLIST edge
    name CDATA #REQUIRED
    transition CDATA #REQUIRED
    weight CDATA #REQUIRED>
```

We have defined a top-level element PetriNet that consists of several Place elements. Every Place element has attributes name and tokens. A Place also contains an input element and an output element. Input and output both consist of zero or many edge elements. An edge has a name attribute, a transition attribute and a weight attribute.
Using the above DTD the ST Petri Net given in Fig. 11 can be represented in XML format.



```
<?xml version="1.0"?>
<PetriNet>
   <place name="P1" tokens="1">
     <input>
     </input>
     <output>
       <edge name="e1" transition="T1" weight="1"></edge>
     </output>
   </place>
   <place name="P2" tokens="1">
     <input>
       <edge name="e5" transition="T2" weight="1"></edge>
     </input>
     <output>
       <edge name="e2" transition="T2" weight="1"></edge>
     </output>
   </place>
   <place name="P3" tokens="0">
     <input>
       <edge name="e3" transition="T1" weight="2"></edge>
     </input>
     <output>
       <edge name="e4" transition="T2" weight="1"></edge>
     </output>
   </place>
 </PetriNet>
```

Figure 11: Simple Petri Net with the XML generated code.

4.6 Analysis of ST Petri Nets

We have written a complete transformer that maps IDEF0 into corresponding ST Petri Nets. This transformer is available as an executable jar file. It takes an IDEF0 model in its XML format and produces a ST Petri Net both in a generic XML format and also in a Python model that can be viewed in ATOM3. It can be executed by the following command

```
    java -jar idef0transformer.jar filename.xml
```

The above command will produce `filename_MDL.py` and `Genericfilename.xml` at the same location. We have also written a generic transformer that takes ST Petri Nets in Generic XML form (described in section. 4.5) and produces the corresponding `name_MDL.py` file to represent the ST Petri Nets in ATOM3. This transformer is written in Java and is available as a jar file. It can be invoked by the following instruction

```
java -jar transformer.jar -st filename.xml
```

The above command generates `filename_MDL.py` file at the same location. ATOM3 has an extensive support to analyze ST Petri nets. We have analyzed the generated ST Petri Nets for A111 activity (Fig. 12)of IDEF0 by creating their coverability tree and doing conservation analysis of A111 in Fig. 13 and Fig. 14 respectively.

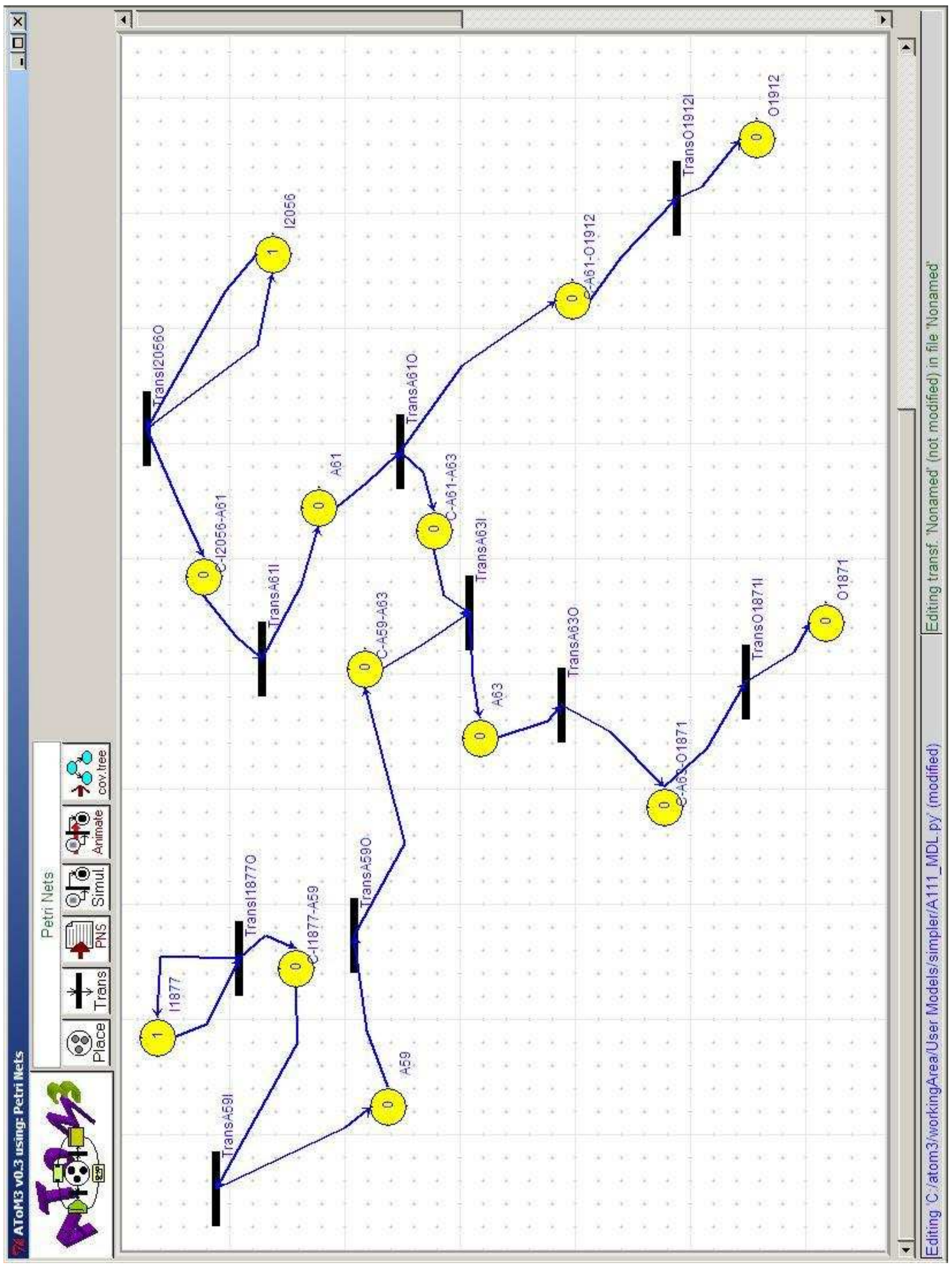


Figure 12: A ST Petri Net for IDEF0 Activity A111.

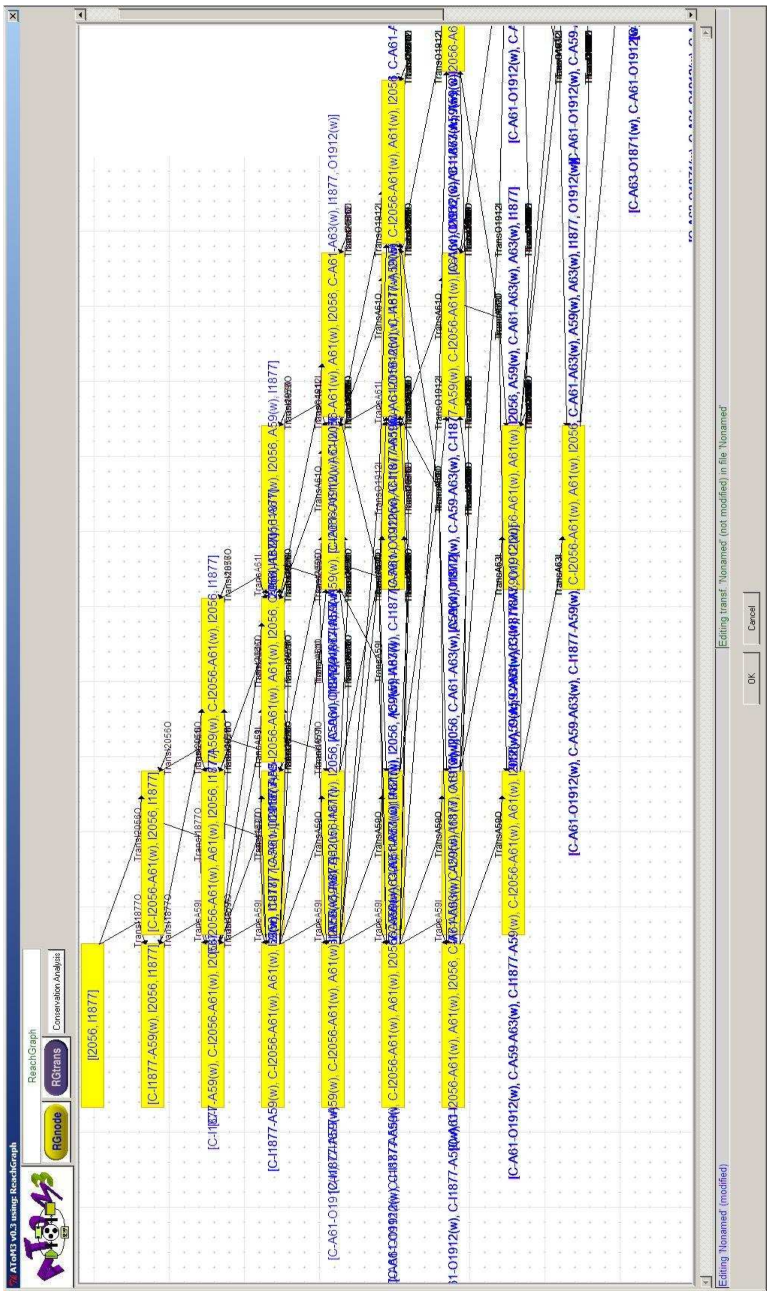
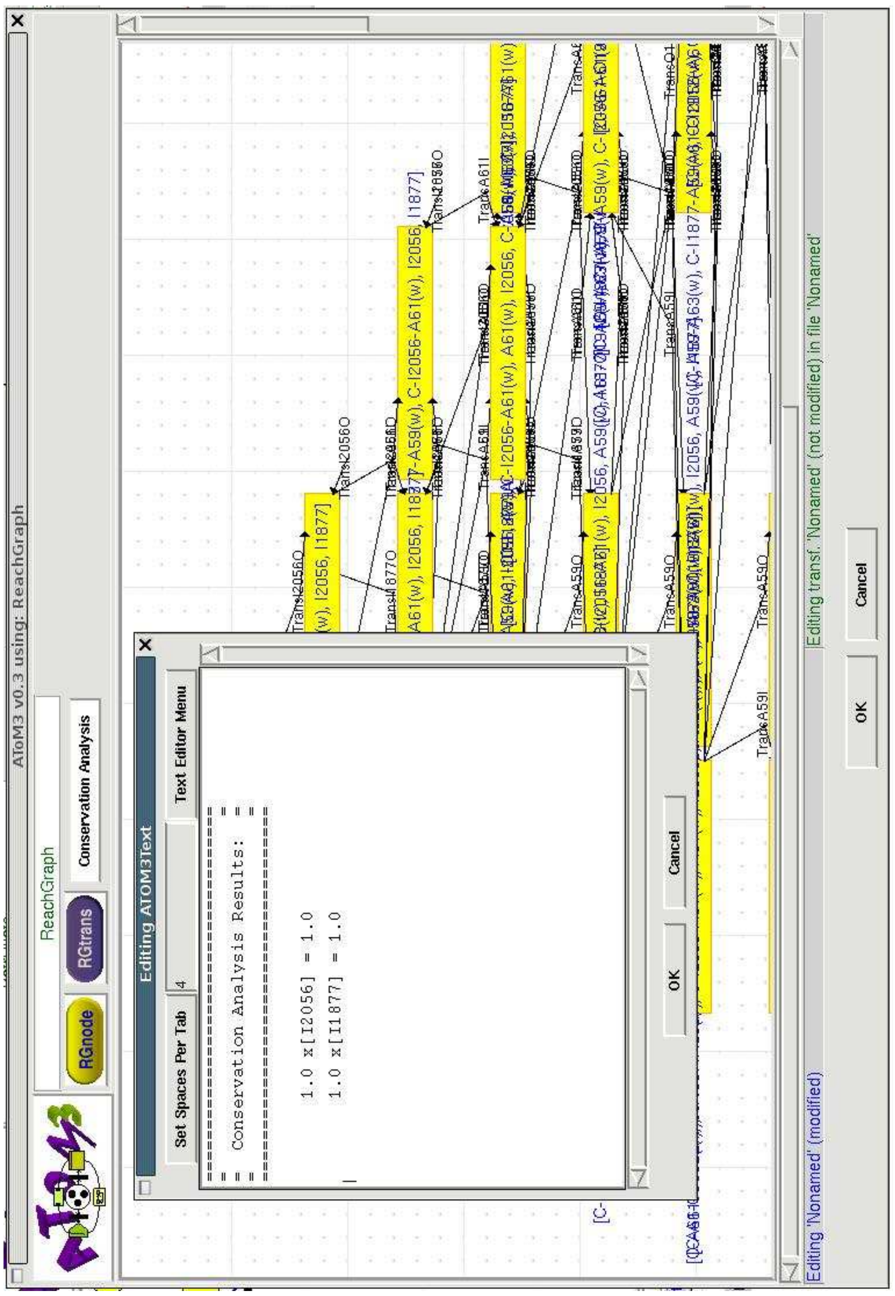

AToM3 v0.3 using: ReachGraph
RGnode
RGtrans
ReachGraph
Conservation Analysis
[I2056, I1877]
[C-I1877-A59(w), I2056, I1877]
[C-I2056-A61(w), I2056, I1877]
Editing 'Nonamed' (modified)
Editing transf. 'Nonamed' (not modified) in file 'Nonamed'
OK
Cancel

Figure 13: Coverability tree for A111.


AToM3 v0.3 using: ReachGraph
ReachGraph
RGnode
RGtrans
Conservation Analysis
Editing ATOM3Text
Set Spaces Per Tab
4
Text Editor Menu
OK
Cancel
Editing 'Nonamed' (modified)
Editing transf. 'Nonamed' (not modified) in file 'Nonamed'

```
==================================
= Conservation Analysis Results: =
==================================

1.0 x[I2056] = 1.0
1.0 x[I1877] = 1.0
```

Figure 14: Conservation Analysis of ST Petri Net for A111.

5 Transformation of IDEF0 to CPN

5.1 Basic Syntax and Semantics of State Transition Petri Nets.

CPN (Jensen, 1996) combines the strength of ST Petri Nets with the strength of programming languages. Like Petri Nets it provides the primitives for describing synchronization of concurrent processes and similar to programming languages it provides the primitive for definition of data types and manipulation of these data types. Similar to ST Petri Net, CPN consists of places and transitions.

5.1.1 Place

Each place has an associated **type** that determines the kind of data that can be placed there. There are four standard types of data int, boolean, string and event e. More complex data types can be defined by making tuples of these basic types. Each data type is also referred as the **color-set**, and the value of the tokens as the colors.

Each place can contain more than one token. All these tokens should be of the same type as that of the place. The state of a CPN is called its **marking**. It consists of the number of tokens at each place in the CPN. The tokens present at a place are called its marking and a place can be defined to have some initial tokens, called *initial marking.*

In Fig. 15 the place I2058 has type INT and initial marking of 1‘1. It means it initially had one token of type INT and value 1. The number of tokens present at a place is shown in a small green circle next to the place. If a place contains no token then the green circle disappears.

5.1.2 Transition

Actions in CPN are modelled as Transitions. Unlike ST Petri Nets, the transitions in CPN are not just enabled by the availability of tokens in the input places but it can also be configured on certain criteria. A Boolean expression can be added to a transition, called the *guard.* It specifies that this transition is enabled only when this guard evaluates to true.

5.1.3 Arc

Arcs are the means of connection between different components of the CPN. An arc joins a place and a transition. An arc has its **arc expression** that contains free variables. These free variables are bound with the values of the tokens at the place when a transition occurs. Tokens are passed from places to transitions via these free variables.
In Fig. 15 the place I2058 is connected to transition I2058O by an arc having expression 'i'. In this expression 'i' represents a free variable and it will be binded with the token from place I2058O in case the transition I2058O is enabled.

5.2 IDEF0 in CPN

In ST Petri Nets, we cannot distinguish between different tokens at a Place and cannot track the movement of a particular artifact through the workflow. As Petri Net models just the flow of events there is no mechanism to handle the timing information. Above all ST Petri Net is a flat structure and we can not model hierarchy in it. Colored Petri Nets (CPN) address all the above-mentioned issues. We have analyzed CPN to determine its suitability to model the workflow of IDEF0. CPN is a whole new modelling structure and it can be visually analyzed through the CPN tool[5]. Due to time constraints we were not able to explore and exploit all the features of CPN but an initial study identifies that we can

- Produce different classes of tokens
- Identify an individual token so we can analyze different tokens of the same class at various stages of the workflow
- Track the flow of a particular token by assigning a timing attribute to it
- Assign timing constraints to the Transitions which are enabled not only by the availability of the tokens at the input places but also takes into account the timing constraint when that transition can be fired
- Model hierarchy

[5]Department of Computer Science, University of Aarhus, http://www.daimi.aau.dk/ kjensen

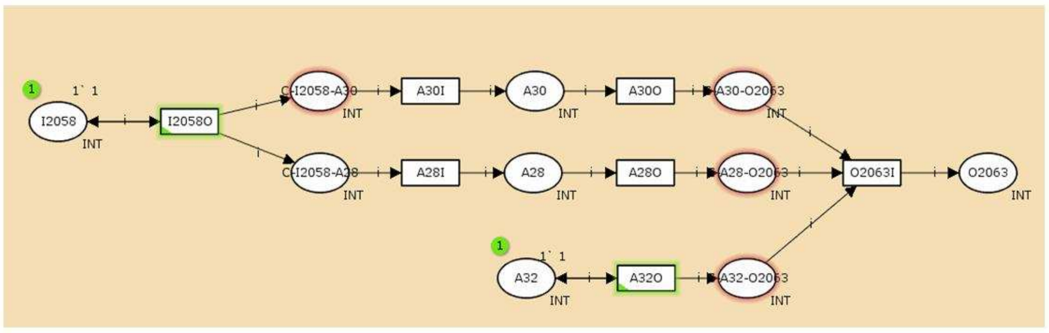


Figure 15: A sample generated CPN.

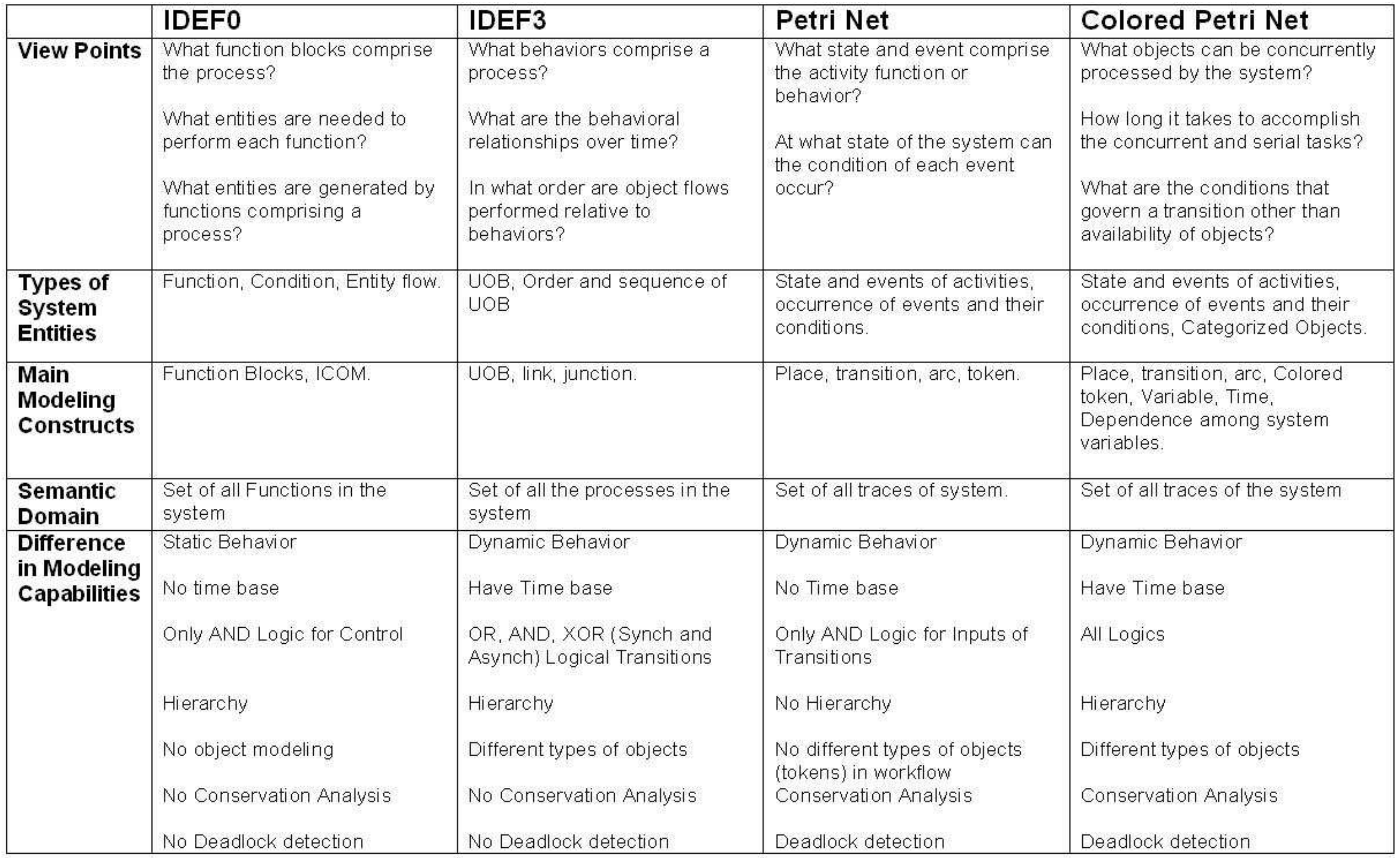
Our complete transformer also maps IDEF0 into corresponding CPN. Given an IDEF0 model in XML format it produces a CPN model that can be viewed and analyzed in CPN tool. It can be executed by the following command

```
java -jar idef0transformer.jar filename.xml
```

The above command will produce filename.cpn at the same location. Similar to the ST Petri Nets, we have also improved the functionality of the transformer to generate a simple Colored Petri Net from the generic XML file. Following instruction produces a *filename*.cpn file.
A simple generated CPN is shown in Fig. 15.

6 Critique

In the table in Fig. 16, all the four formalisms are compared for what kind of views they provide, what system entities they contain, their main modelling constructs, semantic domain and difference in their modelling capabilities. The view points, Types of system entities and main modelling construct, comparisons for the IDEF0, IDEF3 and Petri net are obtained from the Kim et al. (2001). Last two rows and last one column in the table are the result of the analysis in the project. The analysis on IDEF3 model in `ProSim` is not performed since the evaluation version was provided. However, the exercise of transformation from the IDEF0 was beneficial in terms of gaining a more detailed view of object or decision flow. This is so since all the UOBs are linked via a logical junction. State Transition Petri Net model of the truncated IDEF0 can be analyzed by making coverability



	IDEF0	IDEF3	Petri Net	Colored Petri Net
View Points	What function blocks comprise the process? What entities are needed to perform each function? What entities are generated by functions comprising a process?	What behaviors comprise a process? What are the behavioral relationships over time? In what order are object flows performed relative to behaviors?	What state and event comprise the activity function or behavior? At what state of the system can the condition of each event occur?	What objects can be concurrently processed by the system? How long it takes to accomplish the concurrent and serial tasks? What are the conditions that govern a transition other than availability of objects?
Types of System Entities	Function, Condition, Entity flow.	UOB, Order and sequence of UOB	State and events of activities, occurrence of events and their conditions.	State and events of activities, occurrence of events and their conditions, Categorized Objects.
Main Modeling Constructs	Function Blocks, ICOM.	UOB, link, junction.	Place, transition, arc, token.	Place, transition, arc, Colored token, Variable, Time, Dependence among system variables.
Semantic Domain	Set of all Functions in the system	Set of all the processes in the system	Set of all traces of system.	Set of all traces of the system
Difference in Modeling Capabilities	Static Behavior No time base Only AND Logic for Control Hierarchy No object modeling No Conservation Analysis No Deadlock detection	Dynamic Behavior Have Time base OR, AND, XOR (Synch and Asynch) Logical Transitions Hierarchy Different types of objects No Conservation Analysis No Deadlock detection	Dynamic Behavior No Time base Only AND Logic for Inputs of Transitions No Hierarchy No different types of objects (tokens) in workflow Conservation Analysis Deadlock detection	Dynamic Behavior Have Time base All Logics Hierarchy Different types of objects Conservation Analysis Deadlock detection

Figure 16: Comparison of IDEF0, IDEF3, State Transition Petri Net and Colored Petri Net.

and reachability trees. Certain markings (combination of tokens in all the places of ST Petri net) which may be pointed out by the system analyst of the company, to be critical. These critical markings may be be cause of forming deadlocks in the system. When ever an EC (Engineering Change) is evoked, the first job of system analyst and the Program Manager (within the Change Management and Configuration Management functions of a manufacturing organization), is to decide if the change can be executed within the current available resources (both Human and Machine), time left in deadline to deliver the product to client and capital allowed to manufacture without taking the project into debt. The analysis of current available resources and simulation of what if analysis can be performed by the ST Petri nets, by defining the capacity constraints at the bounded resource places (equivalent to functions or UOBs in IDEF0 or IDEF3 respectively).
Dependance of decision of executing EC on time resource available can be answered in the CPN model, by either defining time delay on the transitions (which are fired once the input condition of the transition are satisfied and subsequently the time delay is passed after it) or defining the time to stay on each of the colored token. The later can signify the different service times required by different objects, in case of EC requiring a new documentation to be distributed among its concerned departments (modelled as places (ST Petri Nets or CPN), functions (IDEF0) or UOBs (IDEF3) along with their hierarchy in CPN, IDEF0 and IDEF3).
Cost analysis can both be performed either by CPN or IDEF3. In CPN a cost variable can be assigned to each place or transition which can be incremented conditionally for tokens of different colors. However, IDEF3 explicitly provide the feature of associating cost data to each each UOB process.

7 Future Work

We recommend following four directions in future work.

7.1 Completion of IDEF3 in ProSim on the imported model of IDEF0

IDEF3 model can be completed in `ProSim` by collection of real process data from the manufacturing organization. This will help us performing the EC impact analysis (both time and cost) for the N.P.P (New Part Planning) process.

7.2 Complete model transformation in ST Petri Nets from IDEF0

The complete IDEF0 model transformation to ST Petri Net will help in simulation of Model for a given EC (which can be input to model as initial marking of Petri net) and analyzing the N.P.P process for identification of deadlock states, critical states (may be a cost defective state) and conservation analysis which can in turn provide the places with the capacity constraints on them. Another use of this can be to identify the nature of task execution (whether its executed once, twice, finite times or infinitely many time) by liveness analysis of the transitions in the ST Petri net.

7.3 Hierarchy and Time analysis in Colored Petri Nets

Since CPN formalism support both Hierarchy and time analysis (using state space analysis), the complete IDEF0 transformation to CPN may allow us to retain same structure of the model yet allowing us to model the dynamic behavior of the N.P.P process for certain EC.

The manufacturing organizations are interested in how they can reconfigure their resources for concurrent task execution system with different types of objects (in case of aerospace engine manufacturer different parts of one engine, or same parts of different engine (even the engine model and geometry is same they are assigned different number called *Effectivity* in aerospace manufacturing terminology))?

Well this question can be answered by first modelling the current manufacturing setup in CPN (Calling the model Base CPN), then analyze it for given critical set of critical ECs and the performance of system, while making recommendations for different set of tasks to be executed in concurrency with other tasks on which they might not depend fully. The Base CPN can then be reconfigured on the recommendations resulted from its tests, into a new version (Version 1) of CPN. All the version can be kept in record for role back if desired. This reconfiguration of model and generation of new versions of CPN models can go on until a specified performance index of the system reach its optimum. For validating the configuration of each version of model for set of ECs, is may be required to transform them back to IDEF0 for conforming with the given structure at certain level of hierarchy in it (above which the reconfiguration of the manufacturing setup might be impossible or unfeasible for the company).

7.4 Transformation of completed IDEF3 to Discrete EVent Systems (DEVs)

An interesting formalism called Discrete EVent Systems (DEVs) introduced by Zeigler et al. (2000), is yet to be explored for its features useful for ECM analysis. This formalism provides almost all the distinct features which IDEF0, IDEF3, ST Petri Nets and CPN provide individually. However, since the formalism is low level of abstraction, still a formalism at higher level of abstraction, like the IDEF0 may be required for the ECM analysis.

References

[Cassandras,1989] Cassandras, C.G., 1989, *Discrete Event Systems*, Petri Nets and Timed Models.

[Elhani,2005] Elhani, M.A., Rivest, L., 2005. IDEF0 Model of New Part Planning Process, Ecole Technologie Superiore.

[Jensen,1996] Jensen, K., 1996, An Introduction to the Practical Use of Colored Petri Nets, Department of Computer Science, University of Aarhus,E-mail: kjensen@daimi.aau.dk, WWW: http://www.daimi.aau.dk/ kjensen

[Kim,2001] Kim, C-H., Yim, D-S., Weston, R. H., 2001, An Integrated use of IDEF0, IDEF3 and Petri net methods in support of business process modelling. Proceedings of the Institution of Mechanical Engineers, Part E : Journal of Process Mechanical Engineering, Vol 215, pp 317-330.

[Mayer,1995] Mayer, R.H., Menzel, C.P., Painter, M.K., DeWitte, P.S., Perakath, T.B.B., 1995,"Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report", Knowledge Based Systems Inc. (KBSI), Human Resources Directorate Logistics Research Division, Incorporated One KBSI Place 1500, University Drive East College Station, Texas, 77840-2335.

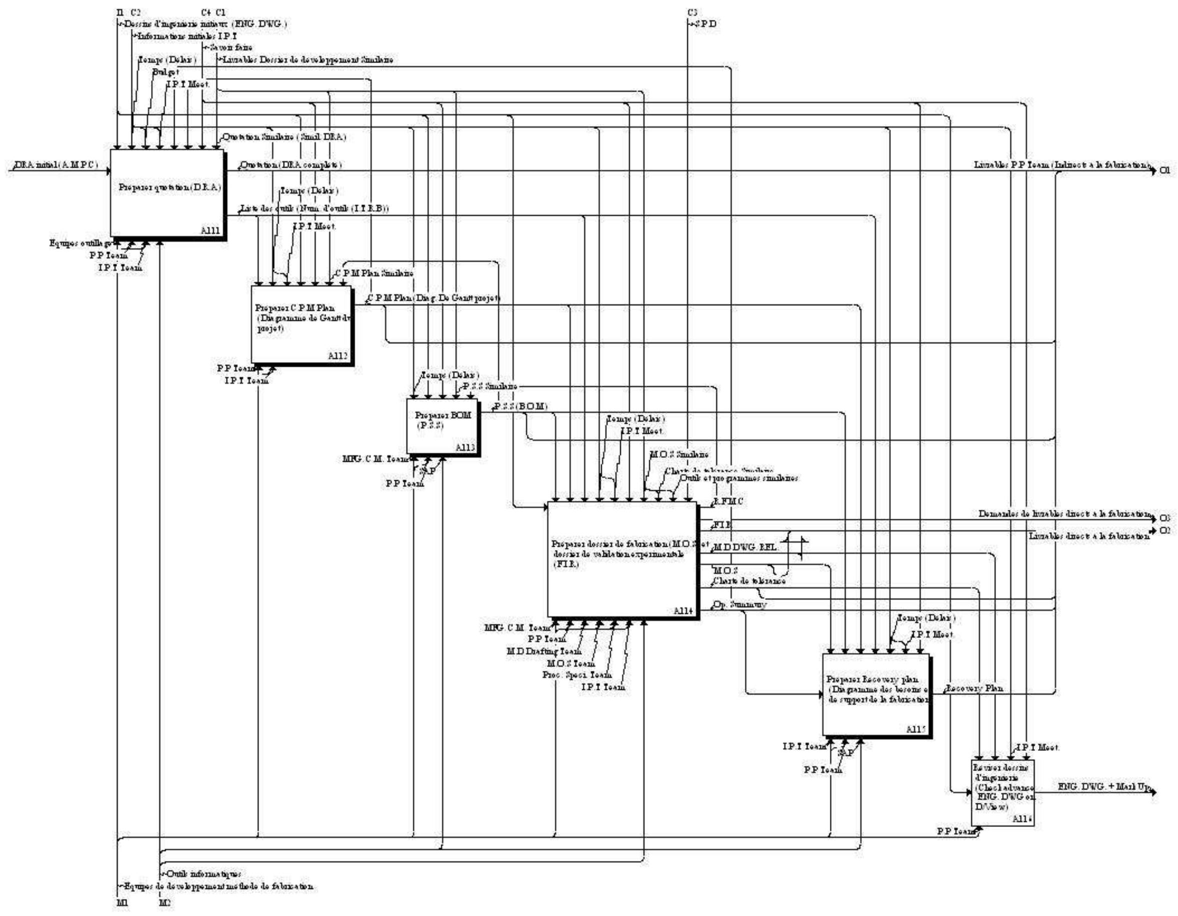
[Vangheluwe,2005] Vangheluwe, H., 2005,"System Specifications", Course Notes, Course:Modeling and Simulation (COMP522A), McGill University, Montreal, Canada.

[Vangheluwe,2005] Vangheluwe, H., 2005,"The Discrete Events System Specification (DEVS) formalism", Course Notes, Course: Modeling and Simulation (COMP522A), McGill University, Montreal, Canada.

[Zeigler,2000] Zeigler, B.P., Praehofer, P., Kim, T.G., 2000. *The foundations of modelling and simulation: Theory of Modelling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, second edition, 2000. Chapters 1 - 9, 17, 18.

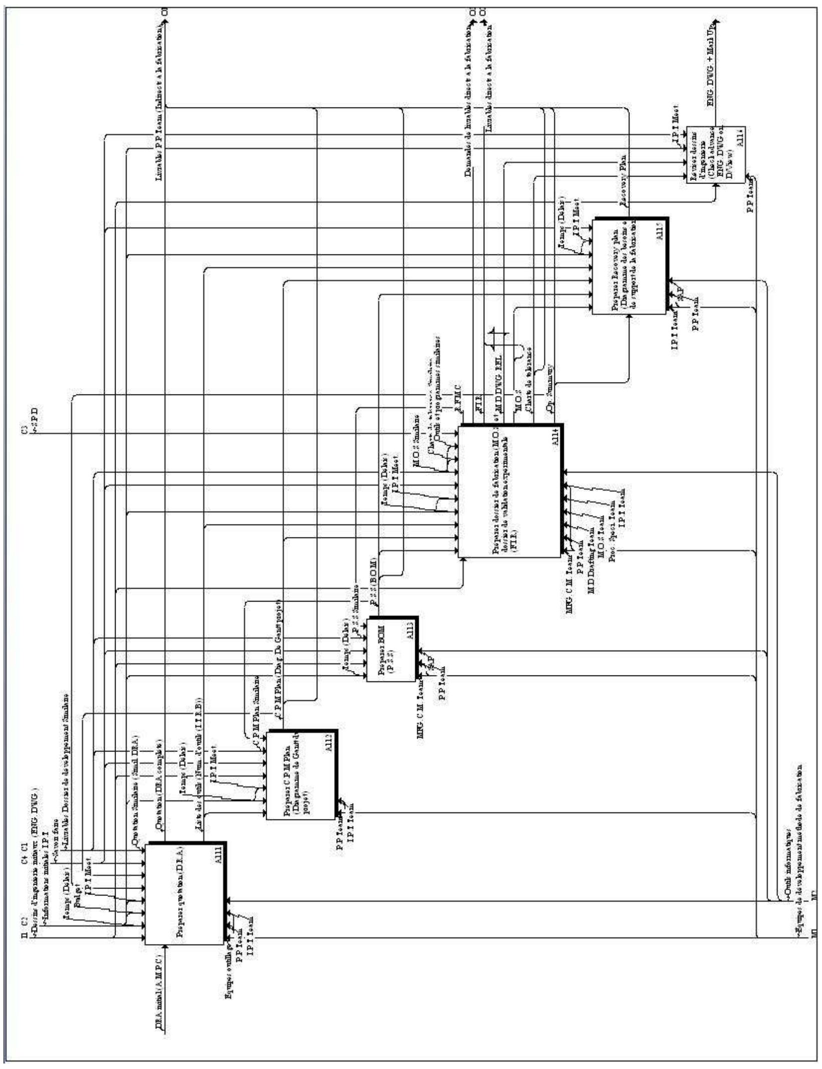
Appendix

A11, First level of Hierarchy

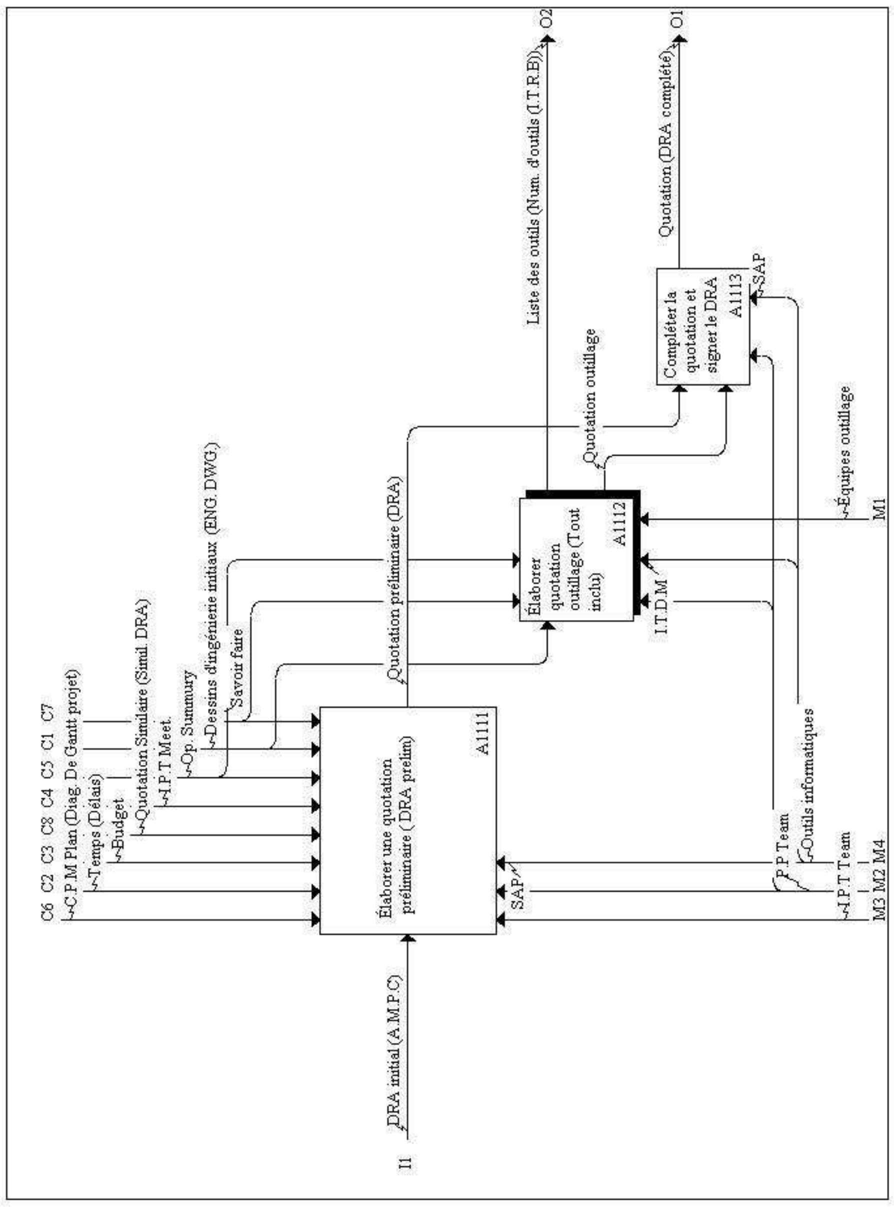


I1 C2
C4 C1
C3
Dessins d'ingénierie initiaux (ENG. DWG.)
Informations initiales I.P.I
Savoir faire
Livrables Dossier de développement Similaire
Temps (Delay)
Budget
I.P.I Meet.
S.P.D
DEA initial (A.M.P.C)
Préparer quotation (D.E.A.)
A111
Equipe outillage
P.P Team
I.P.I Team
Quotation Similaire (Simil. DEA)
Quotation (DEA complète)
Liste des outils (Num. d'outils (I.I.R.B))
Préparer C.P.M Plan (Diagramme de Gantt du projet)
A112
C.P.M Plan Similaire
C.P.M Plan (Diag. De Gantt projet)
Préparer BOM (P.S.S)
A113
P.S.S Similaire
P.S.S (BOM)
MFG. C.M. Team
SAP
M.O.S Similaire
Charte de tolérance Similaire
Outils et programmes similaires
R.F.M.C
Préparer dossier de fabrication (M.O.S) et dossier de validation expérimentale (F.I.R.)
A114
F.I.R
M.D DWG. REL
M.O.S
Charte de tolérance
Op. Summary
M.D Drafting Team
M.O.S Team
Proc. Spec. Team
Préparer Recovery plan (Diagramme des besoins de support de la fabrication)
A115
Recovery Plan
Réviser dessins d'ingénierie (Check advance ENG. DWG on D/View)
A116
ENG. DWG. + Mark Up
Livrables P.P Team (Indirect à la fabrication) O1
Demandes de livrables direct à la fabrication O3
Livrables direct à la fabrication O2
Outils informatiques
Equipe de développement méthode de fabrication
M1
M2

A11, Second level of Hierarchy

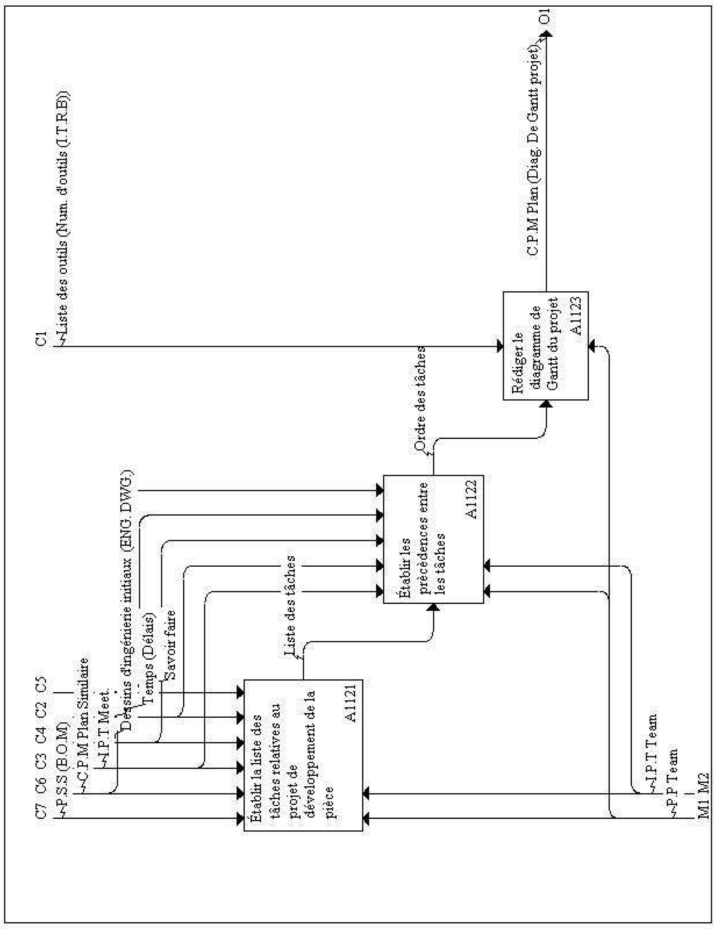


A111, Third level of Hierarchy

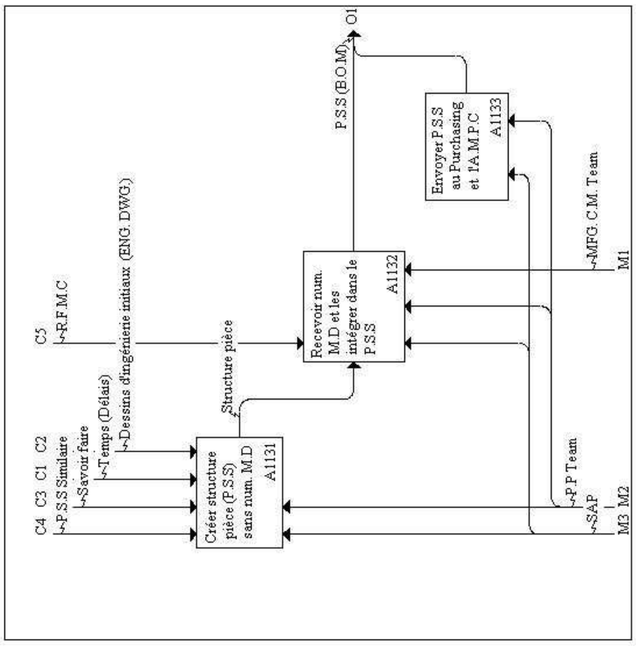


C6 C2 C3 C8 C4 C5 C1 C7
C.P.M Plan (Diag. De Gantt projet)
Temps (Délais)
Budget
Quotation Similaire (Simil. DRA)
I.P.T Meet.
Op. Summary
Dessins d'ingénierie initiaux (ENG. DWG)
Savoir faire
Quotation préliminaire (DRA)
Élaborer une quotation préliminaire (DRA prelim)
A1111
DRA initial (A.M.P.C)
I1
SAP
P.P Team
Outils informatiques
I.P.T Team
M3 M2 M4
Élaborer quotation outillage (Tout inclu)
A1112
I.T.D.M
Équipes outillage
M1
Quotation outillage
Liste des outils (Num. d'outils (I.T.R.B))
O2
Compléter la quotation et signer le DRA
A1113
SAP
Quotation (DRA complété)
O1

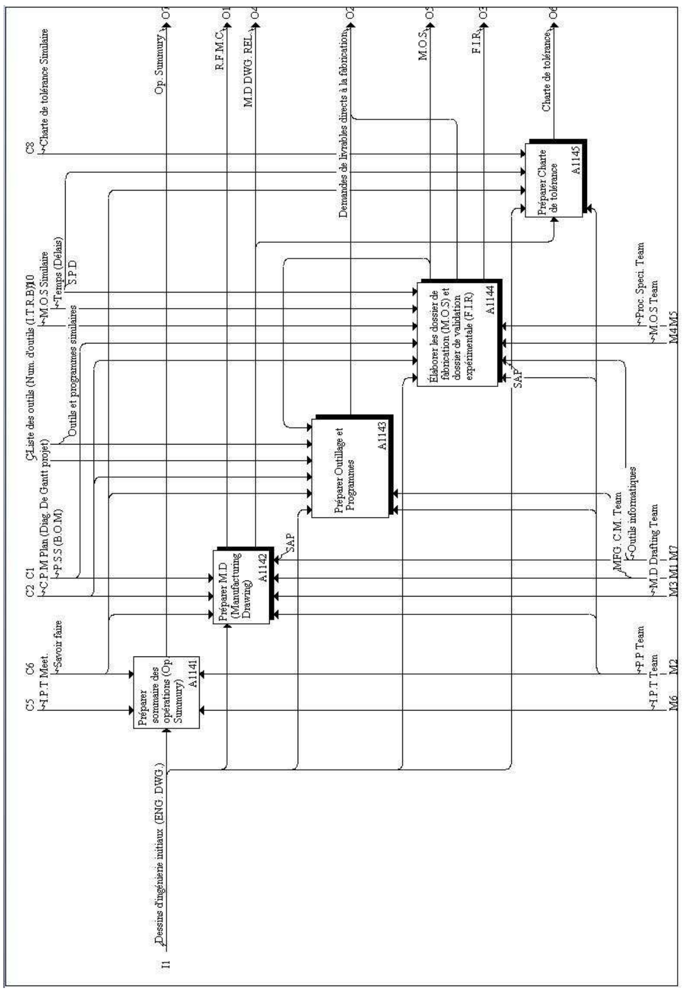
A112, Third level of Hierarchy


C1
Liste des outils (Num. d'outils (I.T.R.B))
C.P.M Plan (Diag. De Gantt projet)
O1
Rédiger le diagramme de Gantt du projet
A1123
Ordre des tâches
Dessins d'ingénierie initiaux (ENG. DWG.)
Établir les précèdences entre les tâches
A1122
Liste des tâches
Liste des tâches
Temps (Délais)
Savoir faire
C7 C6 C3 C4 C2 C5
P.S.S (B.O.M)
C.P.M Plan Similaire
I.P.T Meet.
Établir la liste des tâches relatives au projet de développement de la pièce
A1121
I.P.T Team
P.P Team
M1 M2

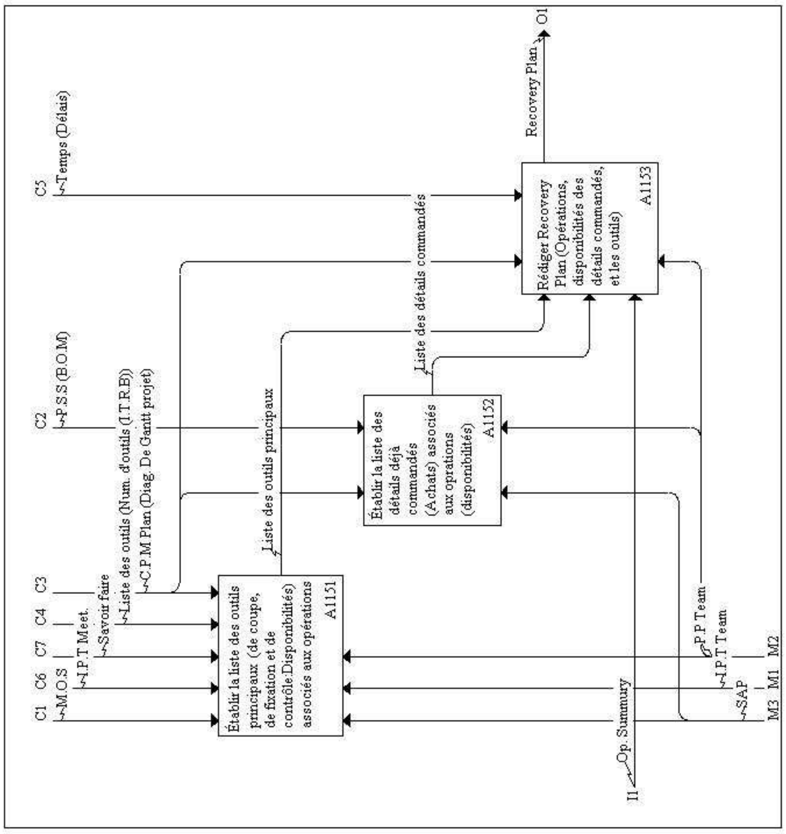
A113, Third level of Hierarchy


C4
P.S.S Similaire
C3
Savoir faire
C1
Temps (Délais)
C2
Dessins d'ingénierie initiaux (ENG. DWG.)
C5
R.F M.C
Créer structure pièce (P.S.S) sans num. M.D
A1131
Structure pièce
Structure pièce
Recevoir num. M.D et les intégrer dans le P.S.S
A1132
P.S.S (B.O.M)
O1
Envoyer P.S.S au Purchasing et l'A.M.P.C
A1133
MFG. C.M. Team
M1
P.P Team
SAP
M3
M2

A114, Third level of Hierarchy


Préparer sommaire des opérations (Op Summury) A1141
Préparer M.D (Manufacturing Drawing) A1142
Préparer Outillage et Programmes A1143
Élaborer les dossier de fabrication (M.O.S) et dossier de validation expérimentale (F.I.R) A1144
Préparer Charte de tolérance A1145
Dessins d'ingénierie initiaux (ENG. DWG.)
I1
C5 I.P.T Meet.
C6 Savoir faire
C2 C1 C.P.M Plan (Diag. De Gantt projet)
P.S.S (B.O.M)
C4 Liste des outils (Num. d'outils (I.T.R.B))
Outils et programmes similaires
C10 M.O.S Similaire
Temps (Délais)
S.P.D
C8 Charte de tolérance Similaire
Op. Summury O7
R.F.M.C. O1
M.D DWG. REL. O4
Demandes de livrables directs à la fabrication O2
M.O.S. O5
F.I.R. O3
Charte de tolérance O6
SAP
SAP
M6 I.P.T Team
M2 P.P Team
M3 M1 M7 M.D Drafting Team
Outils informatiques
MFG. C.M. Team
M4M5 M.O.S Team
Proc. Speci. Team

A115, Third level of Hierarchy


C5
Temps (Délais)
C2
P.S.S (B.O.M)
C3
Liste des outils (Num. d'outils (I.T.R.B))
C.P.M Plan (Diag. De Gantt projet)
C4
Savoir faire
C7
I.P.T Meet.
C6
M.O.S
C1
Établir la liste des outils principaux (de coupe, de fixation et de contrôle.Disponibilités) associés aux opérations
A1151
Liste des outils principaux
Établir la liste des détails déjà commandés (Achats) associés aux oprations (disponibilités)
A1152
Liste des détails commandés
Rédiger Recovery Plan (Opérations, disponibilités des détails commandés, et les outils)
A1153
Recovery Plan
O1
Op. Summury
I1
P.P Team
I.P.T Team
SAP
M2
M1
M3