Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

WORK FLOW ANALYSIS

Valerio O. Pinci
Robert M. Shapiro

Meta Software Corporation
125 CambridgePark Drive
Cambridge, Massachusetts 02140, U.S.A.

ABSTRACT

We present a new approach to the study of work flows in organizations. Using well-known modeling paradigms and existing computer-based tools, we show how to rapidly model and analyze complex work structures to improve throughput and resource utilization. The specification of a work flow is done using IDEF0 (SADT), extended with behavioral details. We have developed a program (Work Flow Analyzer) that processes the resulting model to automatically generate a Colored Petri Net (CP-net) complete with the logic for loading input files and printing reports about processing bottlenecks and idle resources. This greatly reduces the cost and time to use simulation.

1 INTRODUCTION

In the past few years we have acquired experience building and analyzing simulation models from IDEF (Marca and McGowan 1988) work flow diagrams in the following application domains: sales order processing, command and control, check processing, insurance application processing, loan origination processing, engineering design release, and project management.

The main reasons for using IDEF are:

- Use of graphics - IDEF diagrams are created by drawing rectangles that represent activities and by wiring them together with arrows representing input and output relations between them.
- Use of hierarchies - IDEF is a top-down methodology where a process is specified by systematically decomposing higher level activities into more detailed subdiagrams until the meaning (e.g. behavior) of the lowest level activities is sufficiently precise.
- Easiness of use - IDEF is easy to learn since it has very few primitives and makes extensive use of graphics. Most courses that teach how to read IDEF diagrams take half a day. Most courses that teach how to create IDEF models typically take 2-3 days.
- Widespread - IDEF has proven itself mature as a process description language. Tens of thousands of analysts throughout the US and Canada have been trained in its use. The US Government uses it widely, especially for business process analysis. The US Department of Defense has mandated its use for the Corporate Information Management Program. In Europe IDEF is used for the early specification phase of software design, particularly in the aerospace industry and in some of the larger projects (e.g., Columbus) undertaken by the European Space Agency.

On the other hand the use of IDEF alone is not enough to create performance models since attempts to provide IDEF with a well-defined semantics making it possible to execute diagrams have not been satisfactory. One of these attempts conceived of associating IDEF activities with activation rules, expressed in the form of tables, where the relationship between inputs and outputs could be precisely defined. Unfortunately, these tables were too low-level to have any real practical value and did not provide adequate semantics for concurrent processes.

CP-nets [Jensen 1992, Jensen and Rozenberg 1991) provide the capability to simulate IDEF models. IDEF and CP-nets have many common properties including the extensive use of graphics and hierarchies. Previous papers (Shapiro, Pinci and Mameli 1993, Pinci and Shapiro 1991) describe the translation of IDEF diagrams into CP-nets both conceptually and from the point of view of implementation. CP-nets have a well-defined semantics that complements IDEF with a general purpose simulation language. This allows the IDEF models to be used for performance analysis. CP-nets also add the potential of formally validating the correctness of a simulation model, though this aspect has not yet been explored.

Any executable model, such as a CP-net, is ultimately a computer program. Viewed in this way IDEF diagrams provide, via the automatic translation to CP-nets, the control structure of a simulation model/computer program. This well-defined CP-net control structure, made of places and transitions connected via arcs, but still lacking the appropriate net inscriptions, has been the starting point for our construction of simulation models.

Summarizing, the main strengths of CP-nets have been:

- Executability - this provides for the opportunity to cre-

ate performance models that can be used for optimization and trade-off analysis.

- Formal definition of concurrency - this makes it possible to understand how a CP-net model works without detailed knowledge of the internal implementation of the simulation tool used to execute it.
- Use of graphics and hierarchies - this makes it possible to map IDEF diagrams into CP-nets and retains their original structure.

From a theoretical standpoint it was not essential to use IDEF to build the initial structure of the CP-net model. However, CP-nets, unlike IDEF, do not support any specific design method. As a result, novice users of CP-nets often fail to create models because of the lack of structure in the creation process and lack of a widespread CP-net culture.

Our simulation models were constructed by importing the IDEF diagrams into a hierarchical Colored Petri Net tool (Meta Software Corporation 1992a and 1992b). Behavioral inscriptions were added to the models to capture all the intended dynamics. This included providing code for loading data from input files and generating reports characterizing the performance of the models. All behavioral inscriptions used a syntax based on an extension of the language SML (Harper 1986, Paulson 1991, Meta Software Corporation 1992b).

Writing behavioral inscriptions has proven to be an obstacle to many potential users. Even for a CP-net expert, this effort is often tedious and error-prone. A modest IDEF diagram (approximately ten pages) might take five days to fully inscribe and test. Based on this experience we have developed a technique that automatically generates the inscriptions. This greatly reduces the time required and cost of using simulation in Work Flow Analysis.

We use a set of conventions for modeling work flows in IDEF0. A computer program (Meta Software Corporation 1993) deduces the behavioral details automatically for models following these conventions. This eliminates the need to write (and debug!!) complex inscriptions. It is our hypothesis that many work flow problems can be described using this approach. In any case, the CP-net generated by the program provides the starting point for further elaboration or modification.

In our approach, a Work Flow Model describes how 'Documents' flow through an organization. The organization consists of interrelated activities which require resources for their execution. The resources are typically various categories of staff and equipment. Input files characterize the staff and equipment available for a simulation, as well as the time-ordered set of 'Documents' to be processed. The results of simulation are a set of reports which evaluate the performance of the organization and point to bottlenecks (delays) in processing as well as inefficient use of resources (idle resources).

The remainder of the paper is organized in the following way. In section 2 we present the basic constructs in IDEF for specifying behavior. In section 3 we discuss the reports generated by an analysis of a simulation, In section 4 we discuss the input files required for a simulation run. In section 5 we present advanced behavioral features. Section 6 describes aspects of the hierarchical Colored Petri Net created by the model generator (Work Flow Analyzer). This section need be read only both those readers familiar with CPN who wish to understand some details about the generation of the simulation model. Finally, in the last section we discuss future work and draw some conclusions.

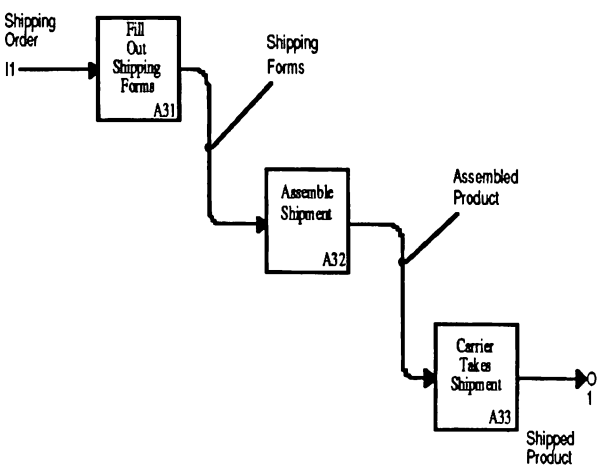
2 BASIC CONSTRUCTS

The basic paradigm for understanding the behavior of an IDEF activity is as follows. When all the inputs required by an activity are present and all the resources required are available, the activity consumes the inputs, producing outputs based on the inputs and delaying the availability of the outputs and resources according to the time required for the activity to be performed.

The Simulation Clock advances only when there is nothing more that can happen at that time. This means that an activity takes place repeatedly so long as there are inputs present and resources available. This interpretation of behavior allows true concurrency in work flows. Sequentialization is a consequence of constraints.

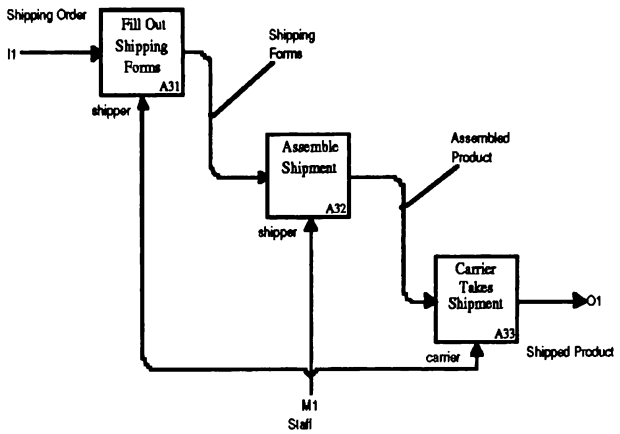
In what follows we present a series of IDEF diagrams that illustrate the rudiments of behavior for work flow models. The diagrams are from a description of a simple Sales Order Processing model. Our objective here is to convey an intuitive understanding of the relationship between IDEF diagrams and their behavioral semantics. A precise definition is provided by the CP-net constructed automatically from the IDEF diagram by the Work Flow Analyzer.

First we examine a simple case: a sequence of activities with **one input and one output**.



Each order flows through a sequence of three steps. The time to ship an order is the sum of the times of the three steps. Since there are no resources required, orders flow through unconstrained by resource limitations. A million orders can be processed in the same time it takes to process one order.

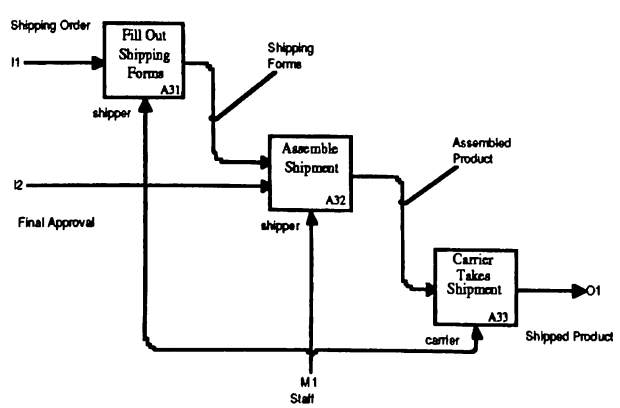
Now we add a **resource**.



Each activity is constrained by the availability of staff. If there is only one shipper, the shipping forms can be filled out for only one order at a time. The same shipper cannot fill out a form and assemble a shipment at the same time. Orders may now be delayed because staff is not available. On the other hand, staff may be idle because there are no orders to ship.

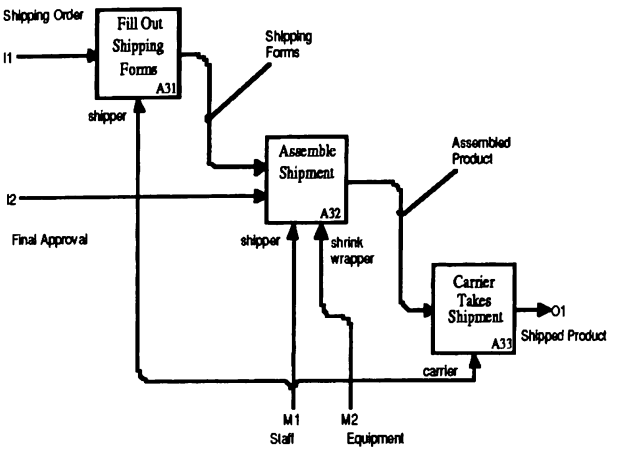
The time to service an order is determined by the difference between its arrival time and its shipping time. This may become large if a lot of orders arrive at the same time. The processing time for an order is the sum of the times of each step it goes through. This remains the same in this model irrespective of loading or staffing. The delay time for an order is the total delay it experiences. In this simple case time to service = processing time + delay time. Where multiple parts of the same order can be processed concurrently, this relationship no longer holds.

There can be **multiple** inputs.



Assemble Shipment may now be delayed for another reason. Both the Shipping Form and Final Approval are required inputs. Thus, processing delays can occur because multiple inputs arrive at different times. The critical path is the longest, the critical input is the latest.

There can be **multiple** resources.



Now Assemble Shipment requires two resources: a staff person who is a shipper and a piece of equipment, the shrink wrapper. This activity cannot take place until both are available. Various causes of processing delays are possible for this activity. The lack of one resource may force the other resource to be idle.

Activities can generate **multiple outputs**. This allows concurrent processing of different parts or copies of an order (i.e.. document). In the sequel we will discuss conditional outputs.

3 REPORTS

The content of the IDEF model is used to determine the format of three standard reports. A run of the model fills in these reports with the data collected during the run.

3.1 Activity Bottleneck Analysis

Each leaf activity in the model is analyzed. The delay each input experiences and the idle time for each resource is accumulated for the entire run. The average values of these delays and idles (per processed 'document') are displayed in a bar chart.

3.2 Summary Bottleneck Analysis

The following pieces of information are displayed: 1. Average time to process an input 'document'. 2. For each category of resource, broken down into each type within that category (e.g. Category: Staff and Type:Shipper) the idle time and the processing time (per processed 'document'). 3. For each type of 'Document',

the delay time and the processing time (per processed 'document').

3.3 Expense Analysis

The following pieces of information are displayed: 1) Average cost to process an input 'document', broken down into Overhead (the sum of all idle time costs) and Direct (the sum of all processing costs). 2) For each category and type of resource, average cost as in 1). 3) The basic cost information is provided in the Resource input files, to be described in the sequel.

4 INPUT FILES

There are three types of input files that must be provided to run a simulation: 1) Delays, 2) Resources, and 3) 'Documents' to be processed.

4.1 Delay Files

Associated with each activity is a time delay (duration time). Once an activity starts, its outputs are not available until the specified time delay has elapsed. All resources needed by the activity are also unavailable for this period of time. The time delay is assigned a default value obtained from a parameter file. The Work Flow Analyzer creates an initial set of values for the file. The user may change these at will. The user may specify the calculation of a time delay. We discuss this in the sequel.

4.2 Resource Files

The IDEF model defines several categories of resources. These are the mechanisms on the top level (A-0) page. Typical examples are 'Staff' and 'Equipment'. A resource file will be created for each category. Types within these categories are mentioned in association with activities whose behavior is affected by this distinction. (E.g. shipper) The Work Flow Analyzer creates an initial set of values for each type in each category. The user may change these at will by introducing new types, changing the number of each type to be present at the beginning of a simulation run. Speed and cost data associated with each individual resource may also be modified by the user. The speed affects the calculation of the duration time at the instant that an activity takes place. The maximum speed factor across all resources used by the activity, is multiplied by the Delay File parameter (for this activity) to calculate duration time. Cost Data affects the Expense Analysis.

4.3 'Document' Input Files

Each document to be processed is listed in the input file. A document is characterized by at least four pieces of information: a unique identifier for coordinating the processing within the work flow (e.g. several copies or parts of the same document); a type which may affect the way in which the document is handled; an arrival time which designates when the document arrives to be processed; and a size which may be used to affect the calculation of activity duration and the order in which Documents' are handled (e.g. a priority scheme). Discussed in the sequel.

5 ADDITIONAL BEHAVIORAL FEATURES

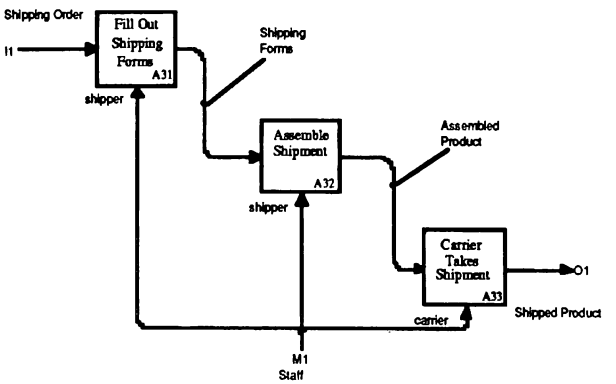
In this section we illustrate more sophisticated behaviors based on labels added to the IDEF diagram:

- •Complex Constraints
- •Calculation of Duration Time
- •Calculation of Size
- •Conditional Outputs
- •Output Transmission Time
- •Priorities
- •Branching and Joining

We start with a simple diagram presented previously, where labels associated with arrow tips designate required attributes (subtypes) of resources.

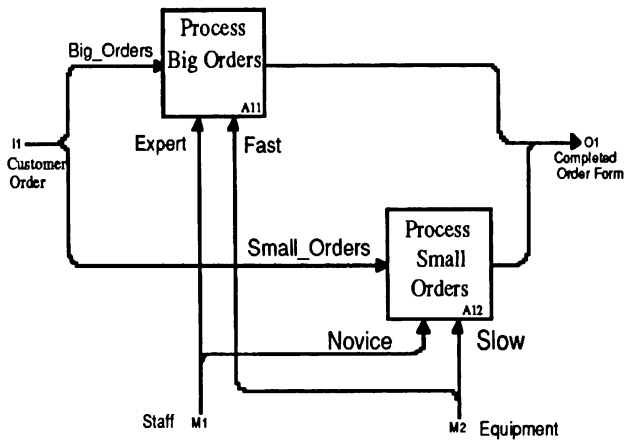
5.1 Constraints

The behavior of any activity can be constrained by specifying type requirements for specific inputs and resources.



Activities A31 and A32 require a specific type of Staff person: a shipper. Activity A33 requires a carrier. If these labels had been omitted, any staff person available would be able to perform the activity. Type constraints may be placed on any combination of inputs and resources.

The type of any output is usually determined by the type of the dominant input (i.e.. topmost). The type of an input or resource may be constrained by providing a label. In the following example, constraints force Big_Orders to be handled by Expert Staff with Fast Equipment while Small_Orders are handled by Novice Staff with Slow Equipment.



5.2 Complex Constraints

In special cases it may be necessary to specify some complex requirement on the inputs and resources of an activity. This may be expressed as a list of Boolean expressions involving input types and sizes, resource types and speeds.

5.3 Size and Duration Time

The calculation of a time delay for the completion of an activity can be influenced by the size of the dominant input. If so specified, the maximum speed factor across all resources used by the activity, multiplied by the Delay File parameter (for this activity) is further multiplied by the size to calculate duration time.

5.4 Computing a New Size

Normally the outputs of an activity have the same size value as the dominant input. The user may specify the calculation of a new value.

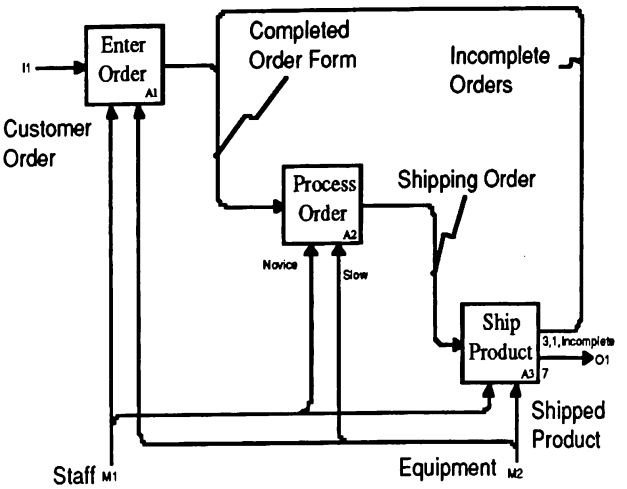
5.5 Conditional Outputs

Normally, all outputs are produced for an activity. The user may specify that only one output in a set be produced. The user provides weights for each output in the set and the simulation uses the weights to select the output.

5.6 Output Transmission Time

Normally, an output is available at a time determined by the calculated duration time for the activity. In some circumstances it is convenient to specify for individual outputs a transmission time. This delays these outputs that much longer.

In the following example we illustrate a number of these concepts:

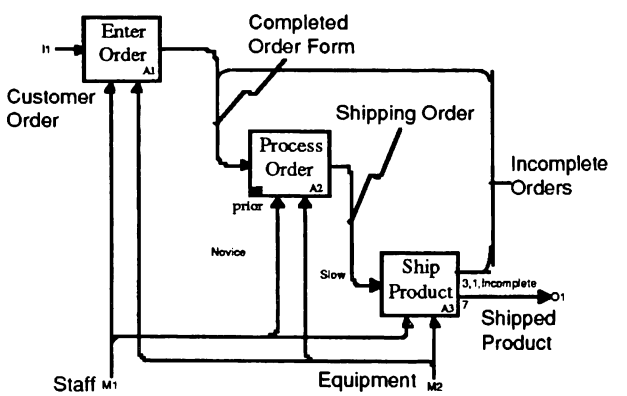


Activity A3 has two outputs. The inscriptions on its output arrows have a simple syntax. The first expression, if present, is an integer **weight**, discussed in the sequel. The second expression is a **transmission delay** for that particular pathway. The third expression specifies the **type** of the output document on that pathway.

The two outputs have relative weights 3 and 7. This means that only one of the outputs will be produced for each occurrence of the activity. The simulator chooses which one randomly, on the average selecting one 30% of the time and the other 70%. The output from A3 that goes to A2 is given an additional transmission delay of 1 time unit. The **type** field in this output is set to the value: Incomplete.

5.7 Priority

Normally, all inputs waiting to be processed by an activity have equal priority. The user may specify that a priority scheme be used to order the inputs. This might typically choose the 'document' with the largest size to be processed first. Consider the following example:



Activity A2 has been assigned a priority function. The priority function specified ('prior': a built-in function) will choose from the waiting order forms the largest, according to the **Size** field. More generally, the user may supply priority functions which order waiting documents based on the contents of each document.

5.8 Branching

Normally, a single copy of a 'document' is generated for each output. But arrow decomposition by branching may connect an output arrow to more than one leaf activity input. In this situation the user must specify whether all destinations should receive the document ("fan-out") or only one.

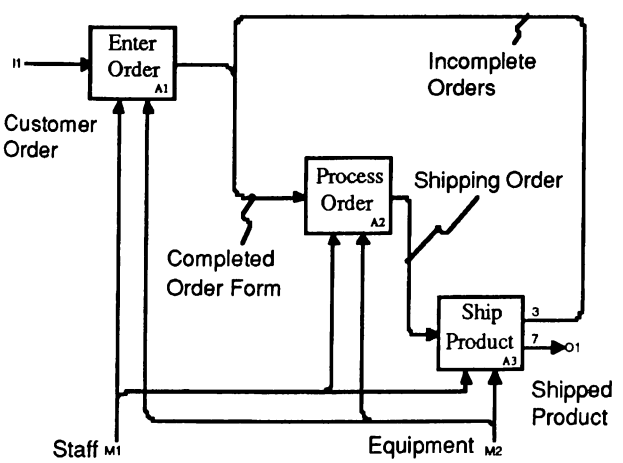
5.9 Joining

Analogously, a single 'document' is normally required as an input. But arrow recomposition by joining may connect several leaf activity outputs to the same input of an activity. In this situation the user must specify whether all sources are required to send the document ("fan-in") or only one.

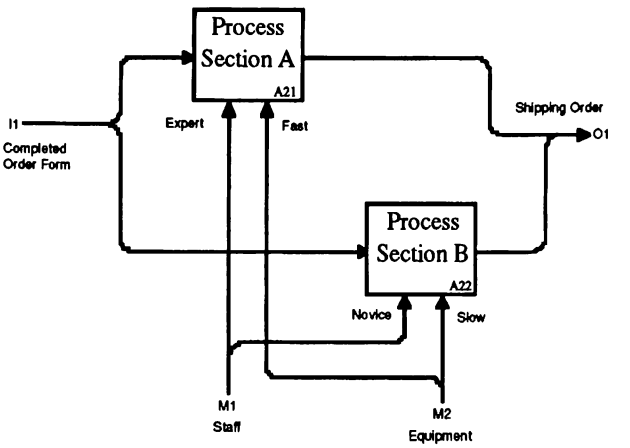
The Work Flow Analyzer detects any ambiguous branch/join structure and asks the user to decide. The answers are recorded and re-used later should the user rebuild the IDEF model. The following two level model illustrates branching and joining.

The intention in this model is to allow the order processing (activity A2) to proceed in two concurrent substeps: Section A (activity A21) and Section B (activity A22).

In this example three ambiguous structures will be detected by the Work Flow Analyzer.



Branching and Joining: Top Level



Branching and Joining: Decomposition

Activity A1 produces outputs for **both** activity A21 and activity A22. This is a fan-out situation. The same is true for activity A3.

Activity A21 requires an output from **either** activity A1 or activity A3. This is not a fan-in. The same is true for activity A22.

Activity A3 requires an output from **both** activity A21 and activity A22. This is a fan-in.

6 INSCRIPTIONS IN THE COLORED PETRI NET

The structure of the hierarchical Colored Petri Net is generated directly from the IDEF model. For a detailed description refer to (Shapiro, Pinci and Mameli 1993 and Pinci and Shapiro 1991). Every activity maps to a transition; decomposed activities become substitution transitions and leaf activities become simple transitions. The arrow structures determine the places and arcs in the CPN model. The ICOM (Input, Control, Output, Mechanism) structure determines the relationship between a substitution transition and its subpage. The color sets for places are deduced from their manner of use. Inputs, Outputs and Controls all represent 'documents'. Mechanisms represent 'resources'. The Work Flow Analyzer checks to make sure the arrow structure is consistent from this perspective.

Inscriptions are generated for leaf activities and their associated arcs. The following types of inscriptions are generated: arc expressions, guards, and code segments.

6.1 Arc Expressions

Arc expressions are based on the following information: they are classified by which edge of the transition they attach to, into Inputs, Controls, Outputs and Mechanisms, as prescribed by the IDEF methodology.

The arc expressions for Inputs, Controls and Outputs have token structures that represent 'documents'. The token structure for Mechanisms represent 'resources'.

Each arc expression is a tuple; the number and type of the components in the tuple being determined by the appropriate token structure. The components (for inputs, controls and mechanisms) are variables with the exception of the **Type** component which may be explicitly restricted to a specific value. The components of output tokens are determined by the inputs, in combination with explicit specification as well as the effect of the guard.

Every mechanism arc automatically generates a reverse arc. The corresponding arc expression returns the resource token to its place of origin, with the available time component and the time stamp updated as specified in the guard.

6.2 Guards

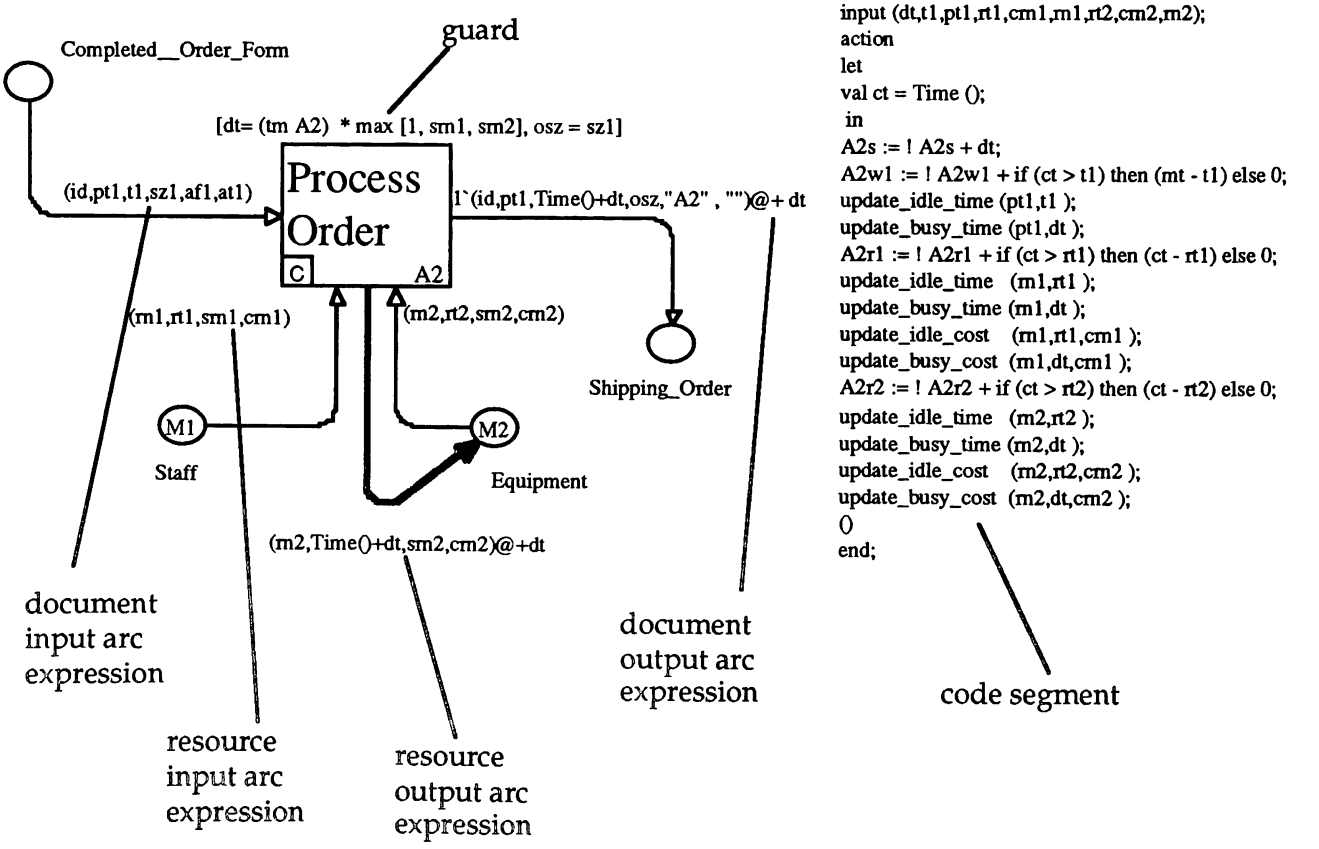
Guards are used in the following ways: 1) To calculate a time delay to be used on output arcs. The default value for a time delay is obtained from the Delay File described in section 3 above. This is multiplied by the maximum speed factor for all the resources used in an occurrence of the transition. It is further modified by the value of the **Size** component of the input 'document', if specified. 2) To provide complex constraints on the documents and resources involved. 3) To calculate new values for the Size component of output tokens. 4) To obtain the 'document' at the head of the queue when a priority scheme is being used.

6.3 Code Segments

Code Segments are used for two purposes. Firstly, to compute all the delay and idle information that is required by the reports produced at the end of a run. Secondly, to use the arc weights described above to determine which arc in a mutually exclusive set should have an output token.

All these inscriptions are generated automatically from the IDEF structure.

In the following CP-net fragment we show some of the inscriptions.



In addition to the submodels that correspond to each page in the IDEF diagram, the Work Flow Analyzer creates three additional pages in the CPN model. These are:

- The Global Declaration Page
- The Input Submodel
- The Report Submodel

Global Declaration contains all the declarations needed for the model. This means all the color sets, variables and functions, including those used in the input and report submodels.

Input Submodel is a net that contains the logic for loading the 'document', 'resource' and delay files. The document and delay files provide information that is converted into tokens that represent all the documents to be processed and all the resources available at the beginning of a simulation run. The delay file information is loaded into parameters accessed by the individual transition guards. The contents of all three files have been discussed previously in section 4.

Report Submodel is a net that contains the logic for recognizing the end of a run and triggering the analysis of the simulation data collected during the run by the code segments of the transitions in the model. Invokes the functions for creating and filling in the reports.

7 CONCLUSIONS

The Work Flow Analyzer greatly reduces the time involved in creating a simulation model suitable for identifying bottlenecks and idle resources in a complex work flow. It accomplishes this by assigning to the activities of an IDEF model a simple behavioral interpretation, based on the arrow structure in IDEF and the behavioral semantics of transitions in hierarchical Colored Petri Nets.

This makes it possible for a computer program to generate, from the IDEF model, the entire structure of the CP-net, including all inscriptions and the additional logic required to load input files and generate reports.

As a result. a much broader group of people will now be able to build models to study the performance characteristics of work flow systems.

It remains to be seen whether the specific interpretation of behavior is general enough to handle a high percentage of work flow problems. Our objective in the near future is to extend the functionality based on the experiences of early users of the approach.

Immediate plans include the following:

1) Reports: No set of fixed reports can satisfy every need. By exporting the raw data collected during simulation, users can format and generate graphs of their own choosing, using standard statistics and presentation packages (e.g. EXCEL).

2) Input Generators: Making up the input sets for a simulation run can be tedious, especially when detailed historical knowledge is lacking. Users will be able to characterize inputs in terms of groups of inputs satisfying specified statistical distributions.

3) More Complex 'Documents' and 'Resources': The user will be able to extend the number of components in a document or resource token and use the new component values to affect the details of occurrence of activities.

4) Resource Schedules: The user will be able to specify time periods for the availability of resources. This will facilitate staff schedules etc.

5) Resource Roles: The user will be able to specify different roles that resources may play. This will simplify the accurate representation of staffing problems in work flow.

6) Formal Analysis: The model may have errors in it which result in 'documents' being stuck at some point in the work flow. If this occurs only in connection with low-probability pathways, testing by simulation alone is a poor technique for finding such errors. Occurrence Graph Analysis (Meta Software Corporation 1992c) for a small input set can discover many of these errors.

REFERENCES

Harper, R. 1986. Introduction to Standard ML. Technical Report ECS-LFCS-86-14, Department of Computer Science, University of Edinburgh, Edinburgh, Scotland.

Jensen, K. 1991. Coloured Petri Nets: A High-level Language for System Design and Analysis. In: *Advances in Petri Nets 1990*, ed. G. Rozenberg, 342-416. Lecture Notes in Computer Science Vol. 483, New York: Springer-Verlag.

Jensen, K. 1992. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1: Basic Concepts.* EATCS Monographs on Theoretical Computer Science. New York: Springer-Verlag

Jensen, K. and Rozenberg, G. (eds.). 1991. *High-level Petri Nets. Theory and Application.* New York: Springer-Verlag.

Marca, D. A., and McGowan, C. L. 1988. *SADT.* New York: McGraw-Hill.

Meta Software Corporation. 1992a. *Design/IDEF User's Manual.* Cambridge, Massachusetts: Meta Software Corporation.

Meta Software Corporation. 1992b. *Design/CPN User's Manual.* Cambridge, Massachusetts: Meta Software Corporation.

Meta Software Corporation. 1992c. *The Design/CPN Occurrence Graph Analyzer.* Cambridge, Massachusetts: Meta Software Corporation.

Meta Software Corporation. 1993. *Work Flow Analyzer Tutorial.* Cambridge, Massachusetts: Meta Software Corporation.

Paulson, L. C. 1991. *ML for the Working Programmer.* Cambridge University Press.

Pinci, V. O., and Shapiro, R. M. 1991. An Integrated Software Development Methodology Based on Hierarchical Colored Petri Nets. In *Advances in Petri Nets 1991*, ed. G. Rozenberg, 227-252. Lecture Notes in Computer Science Vol. 524. New York: Springer-Verlag.

Shapiro, R. M., Pinci, V. O., and Mameli, R. 1993. Modeling a NORAD Command Post Using SADT and Colored Petri Nets. In *Functional Programming, Simulation and Automated Reasoning*, Lecture Notes in Computer Science. New York: Springer-Verlag.

AUTHOR BIOGRAPHIES

VALERIO O. PINCI is Product Manager for Simulation Tools at Meta Software Corporation. His current responsibilities include the design and management of software tools to support business process simulation. His expertise lies in modeling, simulation and analysis of colored Petri nets. He often lectures and presents tutorials at international conferences and has had numerous articles published in books and periodicals.

ROBERT M. SHAPIRO is Chairman of the Board, CEO, and Founder of Meta Software Corporation, a privately held company that develops, sells and supports modeling and simulation tools for business process reengineering. A 36 year veteran of the information systems industry, he has pioneered the development of better and faster computer-based simulation using Petri net technology. He has had a number of his writings published in books and journals.