Common Process Editor (CPE):

A Tool for Process Management and Translation

Users Guide
Version 1.0

Steve Polyak
Steve_Polyak@ed.ac.uk
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge, Edinburgh, EH1 1HN, UK

1. Contents

2. Summary

The purpose of this document is to provide a guide to the functionality and features of the Common Process Editor (CPE). This tool is designed to support process management and process translation in a variety of application areas. In this context, process management includes support for: process creation and editing; library retrieval/storage; and intelligent assistance. Process translation involves a set of modules that can import/export process models encoded in various representations. In this way, the Common Process Editor is seen as "situated" within an environment where interoperability between tools is required. This means that the CPE can accept input from other tools (e.g. AI planning tools, other process editors, etc.) which express "standard" process knowledge representations and can pass knowledge either back to the source tools or along to other special purpose applications (e.g. process simulator, etc.).

This tool is part of thesis research into a Common Process Framework (CPF). The CPF entails: a methodology for developing process knowledge (CPM); a first order language used to express this knowledge (CPL); an ontology (CPO)[1] which defines the concepts and terminology in the CPL; and tool support for this framework via the Common Process Editor (CPE), an intelligent assistant sub-component (CPA), and a collection of translation modules (CPT).

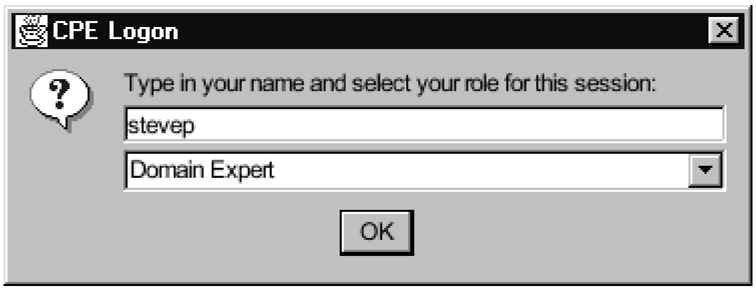
Currently, the tool is in an early release, version 1.0. While much of the basic functionality is in place for editing and inspecting processes, there is still much work which must be done to incorporate this tool into the full CPF. Plans for future versions include: support for process rationale capture using a design rationale notation, and CPT module development for import/export to the Process Interchange Format (PIF), the Process Specification Language (PSL) and O-Plan's Task Formalism (TF).

3. Getting Started

This section provides some of the background knowledge needed to understand the choices of initial logon to the CPE tool. This involves the selection of a role which will be performed during the session. The editor allows only one role at a time, but permits a user to switch between roles (i.e. start a new session) without shutting down and restarting the tool.

3.1 User Roles

When CPE is started, it asks for two pieces of information (see Figure 1). The first is simply the user name which will be associated with changes made throughout the CPE session[2]. The user name may also be used in the future to provide user-specific configuration. This configuration could be specialized to present information which has been selected as "relevant" to the particular individual.


CPE Logon

Type in your name and select your role for this session:

stevep

Domain Expert

Figure 1: CPE Logon

The second item of information involves a selection of user role. In accordance with the Common Process Methodology (CPM), session types are connected to particular roles. The selection of role identifies the possible set of activities for a CPE session and CPE configures itself accordingly to accommodate these activities. These role types are listed below along with a brief overview of the activities performed by each type.

[1] This ontology is based on the <I-N-OVA> constraint model of activity.
[2] CPE currently fetches the Java System Property "user.name" to initially populate this field.

3.2 Process Domains

The first two user roles are involved in the creation and maintenance of process domains. The notion of a domain is taken from the "planning domain description" which is part of the input to generative Artificial Intelligence (AI) planning systems. A domain describes the activities which are performed within a specific application area. So, for example, if the domain is manufacturing then we would have descriptions of drilling, milling, part assembly, etc. Domain descriptions also entail knowledge of many other aspects such as the particular resources utilized, the effects of the activities, etc. This is explored in more depth in section 6.

- **Domain Expert**
  - ⇒ The domain expert is responsible for the overall structure and design of a domain. He/she may create some of the high-level activity descriptions for the particular application. In manufacturing this might be something like: "Manufacture Part" or in a business context it might be "Replenish Inventory" if the application is supply chain process management. This individual determines the scope of the domain (i.e. drawing the boundary between the elements which will be part of the modeled domain and those that will not).
- **Domain Specialist**
  - ⇒ An individual may be assigned the role of specialist within a particular domain. It is his/her responsibility to manage some aspect of the domain created by the Domain Expert. For small domains, the same individual may play both roles of Domain Expert and Domain Specialist. In larger, more complex domains, a specialist may take over the task of detailing particular partitions of the domain that is he/she is knowledgeable of. In manufacturing, this may result in a departmental partitioning. (e.g. a sheet metal process specialist, a rubber works specialist, an electronics specialist, etc.)

3.3 Building Processes

The construction of new processes and the modification of existing processes are carried out within the overall scope of a particular domain. This work is achieved via CPE sessions enacted in any of three user roles. Note the dependence of these sessions on prior domain construction sessions. The particular user role selected is closely related to the session intention. These roles, intentions, and user activities are considered below.

- **Planner / Plan Editor**
  - ⇒ Currently, CPE treats these two roles in the same way. The first role involves a session in which a new process is created interactively by a user "from scratch". The second reflects the users intention to retrieve an existing, stored process for further manipulation. In the case of building "from scratch", this means that after the user has selected a domain, he/she takes one of the high-level domain activities (e.g. Manufacture Part, Replenish Inventory, Build House, etc.) and begins to build a specific process by expanding it down into more detailed levels and by adding constraints to suit a particular set of needs. The resultant artifact may be saved in the process library to be retrieved for future editing.
- **Task Assignor**
  - ⇒ The Task Assignor role is similar in intention to the previous two, but indicates a session in which the act of "process building" will be achieved via interoperability with another tool. The Task Assignor may retrieve a high-level domain activity or a partially edited process. He/she may add particular specifications of preferences and requirements to the process. This artifact may then be translated into a format suitable for input to another tool (e.g. AI planning system). The result of the external tool activity may then be translated back into the process editor for viewing or to be stored for future editing sessions.

4. User Interface Overview

The User Interface provided by CPE is currently separated into two main types: a Domain Editor and a Process Editor. This reflects the two divisions of activity described in sections 3.2 and 3.3. There is a high degree of overlap though in the types of activities performed in both types (i.e. process creating, visualization, and editing). This section presents the overall structure of the interface for both uses.

4.1 Process Editor Interface

The process editor is divided into four areas. The areas have been numbered below in Figure 2. Area 1 denotes the "main menu" section. The functionality of the various drop down menus is reviewed in section 4.1.1. Area 2 consists of a set of tabs which allow the user to switch between multiple open processes. When a new process is opened, a new tab is added to the list. Correspondingly, closed processes will result in a tab being deleted. Selecting a tab causes areas 3 and 4 to change to present the selected process.

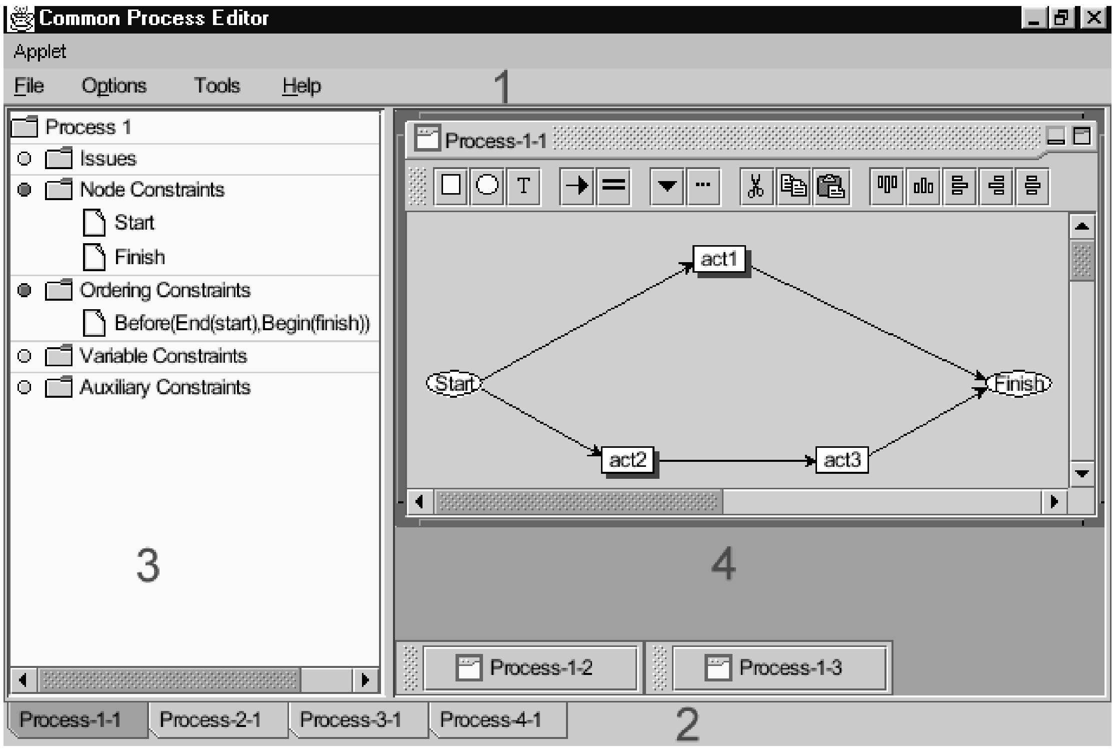


Figure 2: Process Editor Interface

The remaining areas, panel 3 and 4, comprise an integrated, dual presentation of the process knowledge. Area 3 presents a top-down, hierarchical tree view of the process constraints (Note that the tree functionality has not been completed yet, and does not fully list all of the elements for this process. This is discussed in more detail below.). The grouping of tree element constraints are based on the concepts defined in the Common Process Ontology (CPO). The "issues" section contains any of the outstanding or "future constraints" which have not yet been applied or resolved for this process. For example, the process may have been expanded down to a particular level of detail , but certain sections, or nodes may be undefined at an abstract level. These abstract nodes will be listed as future "issues" remaining unresolved for the process. They are things left "todo". The "node constraints" list those activities which have been selected to be included (or not included at this level). Expansion will result in a new sub-tree of elements under the expanded node constraint.

Area 4 presents a network-style view of the same process knowledge embodied in area 3. Eventually, the addition or deletion of process constraints in either the tree area or network area will be automatically reflected in the other panel. Unfortunately, this integration has not been completed yet as the tree presentation has not been fully completed. Currently the amount of area occupied by either panel can be changed by dragging the center-splitting bar between the two panel to either the right or left. More details on the network panel can be found in section 4.1.2.

4.1.1 Menu Options

The main menu for the process editor supports most of the “editor level” command functionality. The various commands are separated into four major types: File Menu, Options Menu, Tools Menu, and Help Menu. This section provides a brief overview of the activities supported by each.

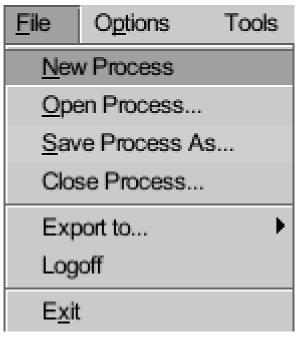


Figure 3: File Menu

The first section of the file menu provides access to the creation, retrieval, saving, and closing of processes. Selecting “New Process” results in a tab which can be used to switch to the new process. For more information on the open and save functionality, see section 8. Selecting “Close Process…” results in a dialog in which the user may select the particular process which is to be removed form the editor list of open processes.

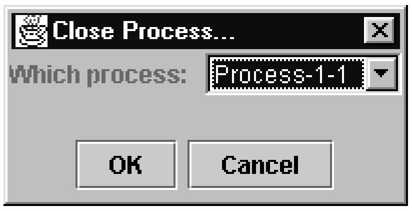


Figure 4: Close Process Dialog

The “Export to…” sub-menu provides the hook for interfacing to the Common Process Translator (CPT) modules. These modules have not been created yet, but will eventually be able to be accessed from this area. The logoff menu option allows a user to switch roles. This command will reinitialize the editor and will present the original logon dialog. Finally, the Exit command will shut down the editor.

The functionality of the options menu is presented in sections 4.3.1and 4.3.3. The only exception to this is the menu state toggle for “debugging”. This flag has been added to provide detailed system traces for various functions. The output goes to the default Java System.out stream. This is mainly for testing purposes and will eventually be removed in a later version.

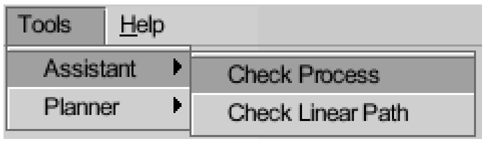
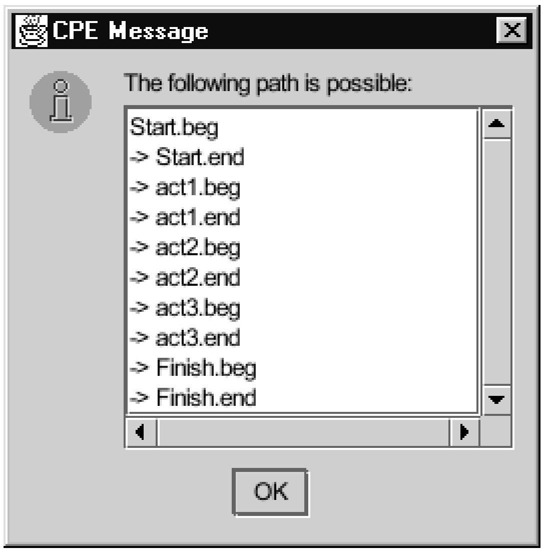


Figure 5: Tools Menu

The Tools Menu provides a hook for communicating with external tools as well as access to native Java tool modules. The Assistant sub-menu provides access to both types. The “Check Process” command is conveyed to the prolog-based Common Process Assistant (CPA). The process knowledge is sent via TCP/IP to this socket-based program. Currently the CPA has been developed to formally reason about the temporal relationships in the process. It maps the timepoint relations onto the 13 Allen Interval Relationships and uses Allen’s transitivity table to reason about the validity of the specification. The next version of the CPE will provide an option for configuration the IP address of the CPA and for providing a detailed presentation of the errors encountered. The “Check Linear Path” command provides access to a native Java module which performs another temporal check of the assigned relationships. It collapses the timepoints that are assigned to be equal (see section 5.4 “Creating Links”) and performs a topological sort to determine whether a linearization of the network is possible. If it is possible, a simple dialog like the one shown in Figure 6 will be displayed, otherwise the dialog will state “No Solution!”. The planner sub-menu is currently blank, but it is hoped that future functionality will be added to allow process/domain specifications to be sent to a generative AI planner. The result of the planning will be accessible in the CPE as well.



The following path is possible:

```
Start.beg
-> Start.end
-> act1.beg
-> act1.end
-> act2.beg
-> act2.end
-> act3.beg
-> act3.end
-> Finish.beg
-> Finish.end
```

Figure 6: Result of "Check Linear Path"

The final main menu entry is the “Help Menu”. The help menu provides a “Contents” and an “About” command. Selecting the “Contents” option provides a dialog with a hypertext help system. This help system has not been completed in this version. The “About” command provides a basic dialog containing the program and version information.

4.1.2 Network Panel

The network panel was presented in section 4.1. This panel is used for editing a graphical network-style presentation of the selected process. This panel actually consists of two parts: a set of windows and a main desktop area. The desktop area “contains” the set of windows. When a new process is created, there is initially one window, containing a “start” and “finish” node which represent the temporal scope of the entire process. The “start” node is constrained to be before the “finish” node via a precedence link. More windows may be added to the desktop when detailed expansions are selected for particular nodes. These windows may be “opened” on the desktop along with the original top-level process window. Any window may also be “iconized” or “maximized” on the desktop in order to provide flexibility in process/sub-process editing. Note that each process/sub-process window has its own toolbar for accepting user commands. This options available in this toolbar are reviewed in section 5.1 (Note that the toolbar buttons provide “tooltip” popup identification signs when the mouse is positioned over them.)

4.2 Domain Editor Interface

The domain editor interface shares many of the same elements with the process editor. Domain Editor additions to the main menu are discussed below. The main difference to observe is that the left panel contains a presentation of the entire currently selected domain. The right panel desktop is now used to edit domain process knowledge (note that only the graphical, network-style presentation is used for domain processes). Unfortunately, much of the domain editor implementation has not been completed yet, but the following section provides an overview of the functionality and features which will soon be available.

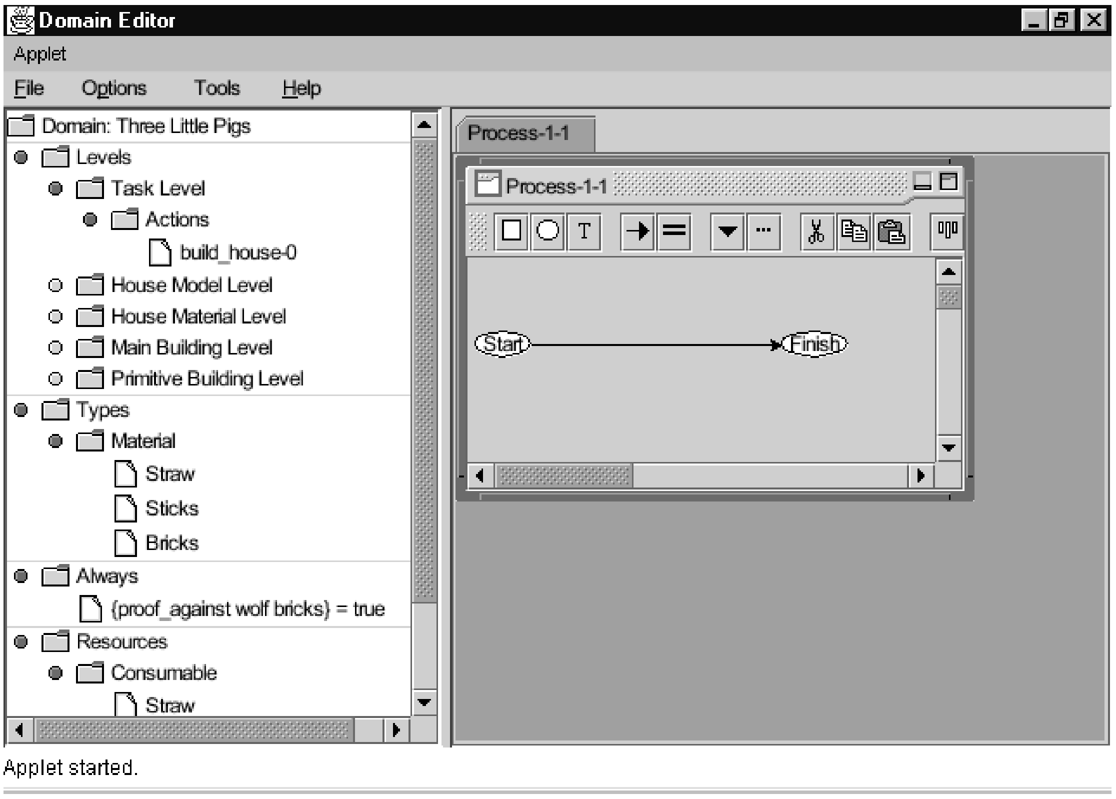


Figure 7: Domain Editor Interface[3]

When the user logs in as a Domain Expert or Domain Specialist the File Menu will also contain a section for managing domains (see Figure 8). This includes options for creating new domains, opening and saving existing ones and closing the current domain. Only one domain is allowed to be edited at a time.

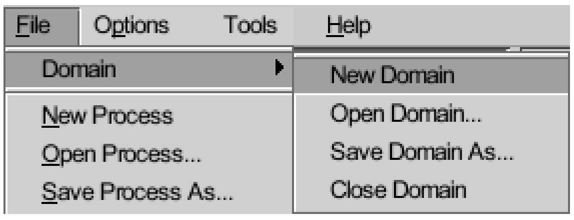


Figure 8: Domain Sub-Menu

4.2.1 Domain Panel

In accordance with the Common Process Methodology (CPM), domains are structured into levels. These levels can be given user-specific names to reflect their particular role in partitioning the domain space. For example, in Figure 7, the "build_house" activity[4] has been created at the "Task Level". The additional levels for this house-building domain include: a house model level, house material level, main building level, and a primitive building level. Levels are also intended to support the grouping of "effects, events, objects, etc.".

The next section of domain information contains the types used for this domain. The simple Three Little Pigs world described above consists of house building "materials" which may be assigned the values of {Straw, Sticks, Bricks}. The "Always" section lists the invariant domain assertions. Currently the domain editor permits syntactically unconstrained sentences for such expressions. The particular syntax used here is from the Task Formalism (TF) language. Finally, the last section describes the resource

[3] The "Three Little Pigs" domain is based on an O-Plan Task Formalism example.
[4] The "-0" ending on the domain process name is temporarily being used to indicate which slot it is in. (See section 8 for information on slots)

types which are present in the domain. These type groupings are defined in the Common Process Ontology (CPO).

Double-Clicking on a process entry in the domain panel pops up the file Input/Output menu for retrieving the process. Future modifications will track open processes and will provide smoother integration between the domain and process panels. As with the process editor, the domain editor permits multiple open processes.

4.3 Customizing the Interface

4.3.1 Pluggable Look-and-Feel

CPE has intentionally been created as an "open platform" tool. As a Java applet, the tool can be run using any browser which provides a Java Virtual Machine (JVM) which provides adequate support for the Java Foundation Classes (JFC). CPE supports the multiple "look-and-feel" option provided by the Java "Pluggable Look-and-Feel" extensions. This facilitates the presentation of a set of visual elements which are consistent with the particular running platform. Currently the Macintosh look-an-feel has not yet been released, but the following are supported: a Windows look-and-feel which emulates Windows 95/98 or Windows NT; a motif look-and-feel which emulates a UNIX/LINUX X Window presentation; and a new platform independent Java standard look-and-feel called "Metal". This configuration can be changed using the “Options” menu. A sample of each are shown below.

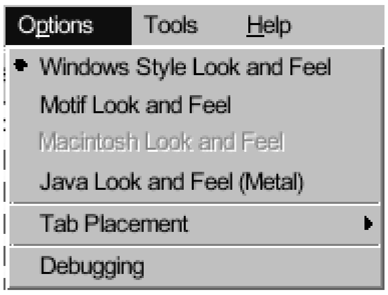


Figure 9: Look-and-Feel Menu Options

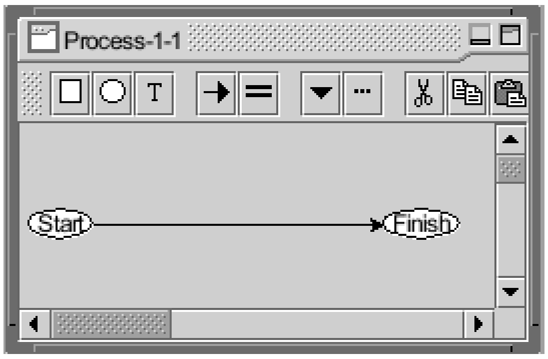


Figure 10: "Metal" Look-and-Feel

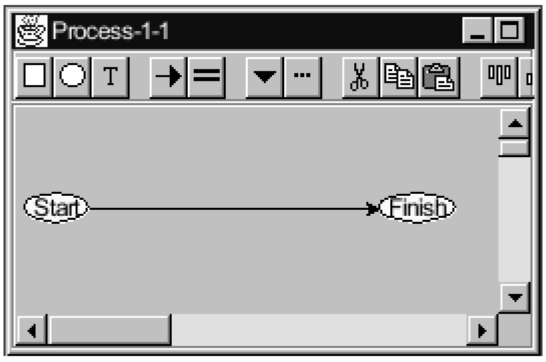


Figure 11: "Windows" Look-and-Feel

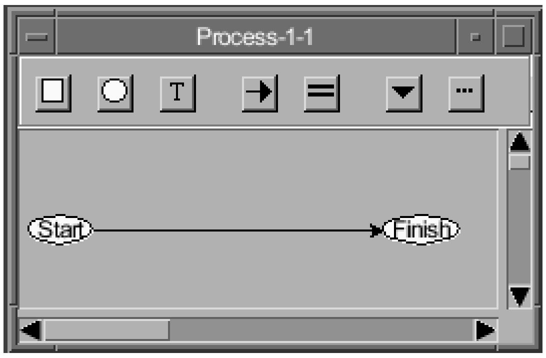


Figure 12: "Motif" Look-and-Feel

4.3.2 Docking Toolbars

Along with the choice of Java look-and-feel, there is also additional support for configuring the user interface. Part of this support is based on the dockable toolbars provided by the Java Foundation Classes (JFC). The toolbar can be “torn” from its default position at the top of the window by left-clicking on any non-button portion of the toolbar and dragging it away. The toolbar can then be dragged over any of the four sides of the window to “dock” it onto a particular side or it can be dragged outside the window and dropped in the client area to provide a “floating” tool bar. See Figure 14 and Figure 13 for examples of a floating and left-docked toolbar.

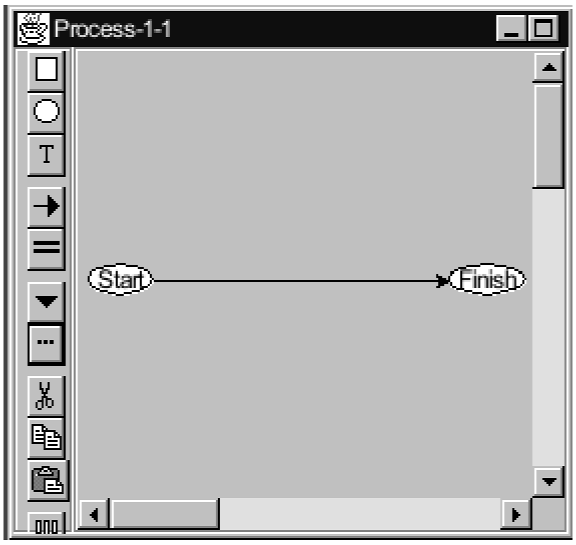


Figure 13: Toolbar Docked to the Left

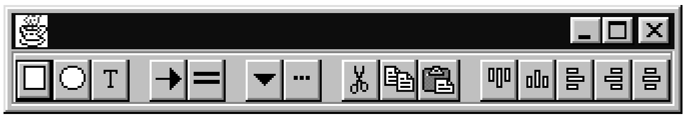


Figure 14: "Floating" Toolbar Support

4.3.3 Tab Positioning

In addition to the “dockable” toolbars described in the previous section, there is also a choice of tab placement. These tabs allow the user to switch between various processes when multiple processes have been opened. The default is to place these tabs along the top of the interface, but any of the four positions are possible. The placement along the bottom is similar to a Microsoft Excel “worksheet” presentation.

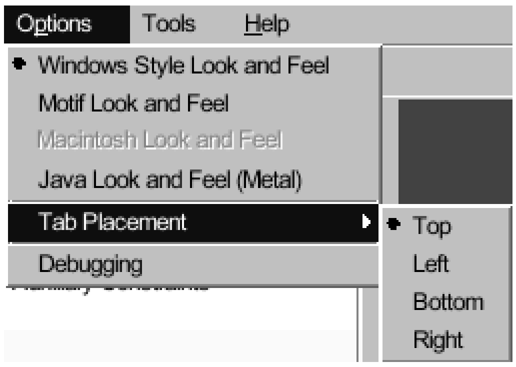


Figure 15: Tab Placement Menu

4.3.4 Multiple Interface Support

Many of the commands available in the editor can be invoked in more than one way. The two main interface options in the CPE are the context-sensitive popup menu and the toolbar. For example, a new activity can be added to a process in one of two ways. The user may click the leftmost button on the toolbar which looks like a small, white box. If it is the first activity to be added, it will automatically be placed between the start/finish nodes and the proper temporal constraints added. Any additional activities added in this way will be created in the top left corner of the process window. Alternatively, a user may right-click anywhere in the process window space and a context-sensitive menu will appear (see Figure 16). Selecting the “Add Activity” command will create a new activity node at the point where the user right-clicked on the process window. Note “context-sensitive” means that a different menu will appear if right-clicking over a node (see section 5.9).

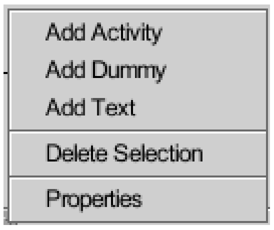


Figure 16: Process Popup Menu

5. Process Editing

5.1 General Overview

This section presents a task-oriented description of the functionality available for process editing. The steps required for each task are listed under each subheading. In most cases, there is more than one way to achieve the process editing task as described in section 4.3.4.

New top-level process windows contain a start and finish node which define the temporal scope of the process. Sub-process windows are scoped by a begin and end node pairing.   These nodes cannot be deleted, but they can be selected and positioned on the window.

5.2 Adding Nodes

Currently in the CPE there are two different “types” of nodes: activity nodes and dummy nodes. Activity nodes (shown as rectangular boxes) represent the activities performed within the process whereas the dummy nodes (shown as ellipse shapes) can be used to provide “special” support for process structuring (e.g. split/join nodes, start/finish, and/or, etc.) Future CPE versions will add more specialized node types.

The user may click either of the two leftmost buttons on the toolbar, corresponding to a new activity node or new dummy node. New dummy nodes are automatically placed in the top left corner of the process window. If it is the first activity to be added, it will automatically be placed between the start/finish nodes and the proper temporal constraints added. Any additional activities added in this way will be created in the top left corner of the process window. Alternatively, a user may right-click anywhere in the process window space and a context-sensitive menu will appear (see Figure 16). Selecting the “Add Activity” or “Add Dummy” command will create a new node at the point where the user right-clicked on the process window. The default label for new nodes is either “act#” or “dummy#” where # is simply the next value of an 32-bit integer counter. These labels can be changed in the node properties dialog (see section 5.9).

5.3 Moving Nodes

Node positions can be changed by clicking and holding the left button down inside a node. Dragging the mouse while holding the button down will change the position of the node. Releasing the left button will fix the node in the new position. Any links attached to the moved node are updated while dragging. Note that the nodes can only be dragged to the edge of the process window. Dragging out of the window will suspend the repositioning of the node until the mouse has returned to the process window. Future versions may provide: selected group moving and layout grids and “snap to grid” functionality for easier node positioning. See also section 5.4 for automatically moving nodes.

5.4 Creating Links

There are currently 2 different types of links in CPE: a before link and an equals link. The before link appears as a directed arc between two nodes and the equal link is represented by a double-line arc. All CPE nodes are actually made of two “halves”. From the left most edge to the center of the node is considered the “begin” half of the node whereas the center of the node to the right most edge denotes the “end” half of the node. For example, two links have been drawn in Figure 17. One is a before link between the begin of “act1” and the begin of “act2”. The other is an equals link between the ends of “act1” and “act2”.

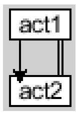
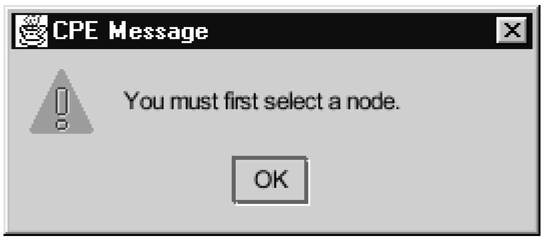


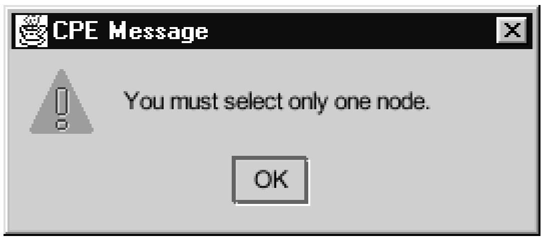
Figure 17: CPE Links Example

The node halves actually represent the begin/end timepoints which define the temporal scope of the activity. The two drawn links in Figure 17 specify the following constraints in natural language: “Activity act1 must begin before activity act2 begins. Activities act1 and act2 must finish at the same time”.
Such specifications can be created in the process network panel in one of two ways: using the toolbar or the popup menu. Let’s say we’d like to specify that act1 must be performed before act2 using the toolbar. The first step is to select the act1 which can simply be done by left-clicking the node (and releasing). Next, click the toolbar button which represents the before link (arrow pointing right). If no node has been selected or multiple nodes are selected, the user will receive an error dialog (see Figure 18 and Figure 19).


CPE Message

You must first select a node.

Figure 18: No Node Selected Error


CPE Message

You must select only one node.

Figure 19: Multiple Node Selected Error

After clicking the toolbar button, a red tracking line will appear which originates from the end half of the selected node (note that the toolbar option only supports linking from the end, if linking from the begin, use the popup menu method instead). The tracking line will follow the mouse movement until the user left clicks in the window. Left clicking on any node half will result in a link being created between the two halves or timepoints. Left clicking anywhere else in the process window will terminate the link drawing process. The exact same steps as above can be performed for drawing an equals link. Simply select the equal link button from the toolbar instead.

A more flexible option is to right-click on a node half. A context-sensitive menu will appear with a set of commands (see Figure 20). Selecting the "Add Before Link" or "Add Equals Link" will place the user in the same line tracking state as described above.

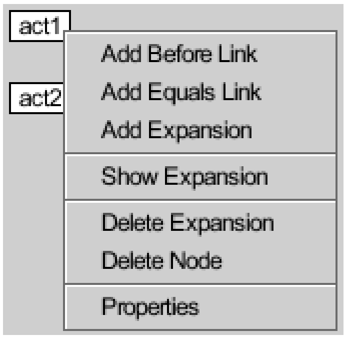


Figure 20: Node Popup Menu

Note that the start/finish or begin/end nodes are exceptions to this in that only one half of each node is accessible for participating in a link relationship. Specifically, only the end half of start/begin nodes and the begin half of finish/end nodes can be accessed. Future link drawing may support multiple "control" points for maintaining clearer arc connections.

5.5 Selecting Nodes, Links, Text

Many of the process editing commands rely on either single or multiple selections of nodes, text and links. There are a number of ways to perform these selections in CPE. The first method, which has been described earlier, is simply to left-click once on any node (note node selection and text selection is identical). The node background color will become reddish to indicate that it has been selected. Left clicking on another node will change the selection to the new node. All selections can be cleared by simple left clicking anywhere on the "blank" process window space. Multiple node selections can be incrementally made by holding down the "Control" key while left clicking on various nodes. A

"Control-Left Click" will "toggle" the selected state of the node without changing the selected status of other nodes and links.
Another way of selecting nodes (and links) is to draw out a "bounding box" around the desired area. This can be accomplished by clicking and holding down the left mouse button in any process window space. Dragging the mouse will result in a red line selection box being drawn from the original point of origin. Releasing the mouse button will result in a selection of anything that the bounding box touches. Links will be "reddish" along with nodes to indicate their selected status.

5.6 Aligning Nodes

CPE provides support for aligning sets of selected nodes. This functionality is available on the right most set of buttons on the process toolbar (see Figure 14). The first step is to select the nodes which will be aligned (see section 5.5 on selecting nodes). Clicking the "aligntop" button (see the tooltips) will result in all of the selected nodes being repositioned at the same y-axis position of the node nearest the top of the window. Clicking the "alignbottom" button will result in all of selected nodes being repositioned at the same y-axis position of the node nearest the bottom of the window. The "alignleft" and "alignright" buttons execute the corresponding positioning along the x-axis. The "aligncenter" button will position selected nodes along a fixed vertical line.

5.7 Adding Text

Textual annotations can be added anywhere on the process window. These are used to simply provide more information about the process (e.g. labeling sections, adding comments, etc.). Just like nodes, they can be dragged and positioned once added. There are two options for adding text. Clicking the text button (the "T") in the toolbar or right-clicking on the process window space and selecting the "Add Text" command. The advantage of using the popup menu is that the annotation will be placed at the point where the right-click occurred whereas the toolbar option will place the text in the top left corner of the process window. In either case, an "add text" dialog will appear. Multi-line text annotations can be created using this dialog (see Figure 21).

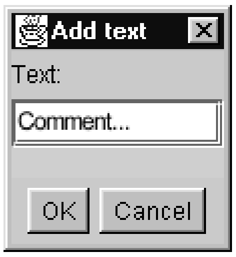


Figure 21: Add Text Dialog

5.8 Cut/Copy/Paste

CPE provides Cut/Copy/Paste operations for transferring process information from one process to another. This works on a simple "clipboard" style approach. The required nodes, links and text must first be selected in the source process window. Clicking the "Cut" or "Paste" toolbar buttons will transfer the selected items to the "clipboard". In the case of cut, it will also remove the selected elements from the process window. The user may now create a new process or switch to an existing one using the process tabs. Clicking the "Paste" toolbar button in the target process window will add the contents of the clipboard to the process window.

5.9 Working with Node Expansions

Activity nodes may be assigned a node expansion. An expansion provides detailed steps for performing an activity. Currently, activity expansions can be created by right-clicking on an activity node and selecting "Add Expansion" from the node popup menu (see Figure 20). This creates a new process window on the process panel desktop area. This sub-process window is defined within the temporal scope of the expanded node begin/end timepoints. The expanded node displays a "shadow" to indicate that an expansion has been attached to it. The new sub-process window can be minimized, opened, or maximized. Unfortunately, the delete expansion command has not been implemented yet.

5.9.1 Navigating Expansions

In order to provide intuitive navigation of the process expansions, an “expand” (simple arrow pointing down) and “up” (simple arrow pointing up) button has been added to the process toolbar. The “up” button is only present on sub-process windows (see Figure 22).

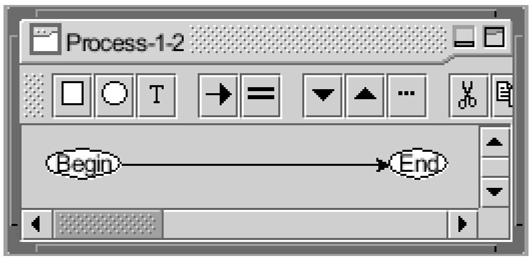
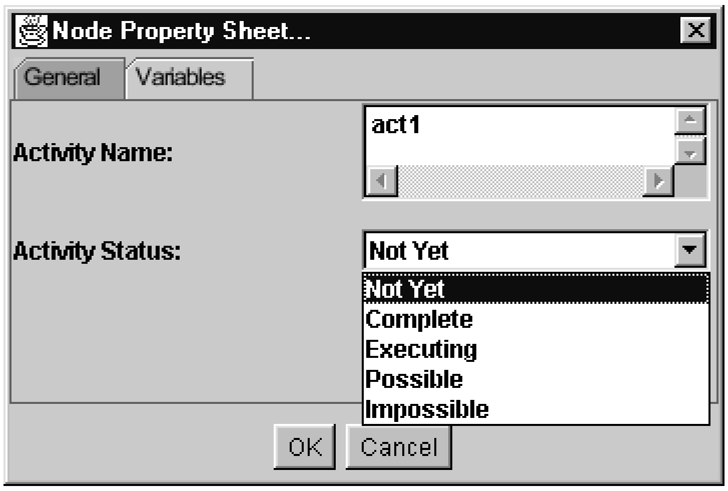


Figure 22: Sub-Process Window

In order to use the “expand” button, a node which has an underlying expansion must be selected. Clicking the “expand” button will bring the sub-process window to the “top” of the process panel desktop (note this will open a minimized sub-process windows as well). In the reverse direction, a user may wish to navigate from a sub-process window to the parent process. This can be accomplished by simply clicking the “up” toolbar button. The name of the parent process is also accessible in a sub-process by viewing the process property sheet (see section 5.11).

5.10 Editing Node Properties

Detailed information about an activity node can be accessed in the node property sheet. This sheet can be accessed in one of two ways. The user can first select a node and then click the “properties” toolbar button (the button with the “…”) or by right-clicking on the node and choosing the “Properties” menu option. Currently there is support for the activity name, status, and variables (see Figure 23). The activity name is the label that is displayed for the node. This can be a multi-line value. The activity status was used for a demonstration of process state viewing and may be removed in future versions. Selecting different activity status values will change the background color of the node. The variables section provides a simple string list interface where syntactically unconstrained expressions can be attached. This might be used to state something like “?material=straw” for assigned values or “?material=” for unassigned variables. Future work will seek to structure this input.


Node Property Sheet...

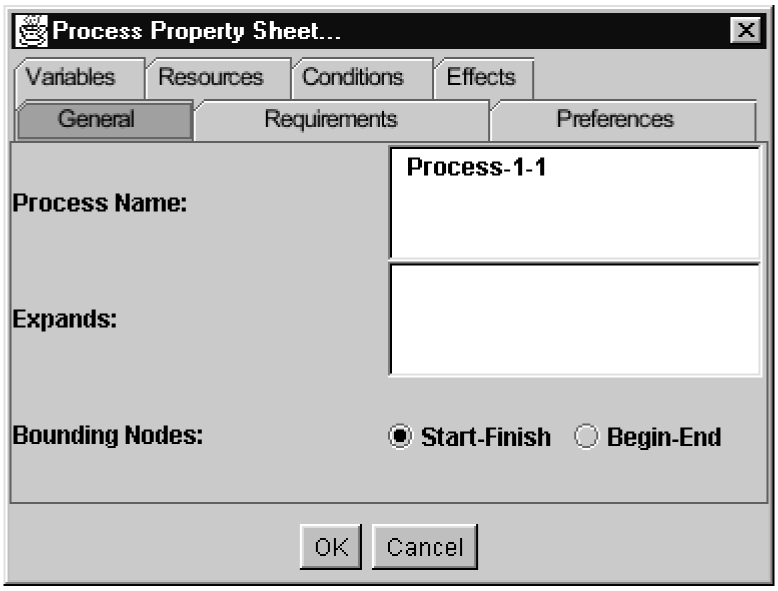
Activity Name: act1

Activity Status: Not Yet

Figure 23: Node Property Sheet

5.11 Editing Process Properties

The bulk of process knowledge accessible in the process network panel is captured in the process property sheet. This sheet can be accessed in one of two ways. The user may click the toolbar “properties” button without having any nodes selected or the user can right-click on the process window area and select “Properties” from the popup menu.


Process Property Sheet...

Variables | Resources | Conditions | Effects | General | Requirements | Preferences

Process Name: Process-1-1

Expands:

Bounding Nodes: [✓ Start-Finish] Begin-End

OK Cancel

Figure 24: Process Property Sheet

The process property sheet is divided into 7 sections (see Figure 24): general, requirements, preferences, variables, resources, conditions, and effects. The general section currently contains the process label which is displayed on the process window title bar. The "expands" entry is for viewing only and is used to display the parent process label for sub-processes. The bounding nodes option is used to select the appropriate scope nodes. Start-Finish are reserved for the relative bounding points for the entire process whereas begin-end should be used to bound sub-processes. The other 6 sections are all currently unstructured lists of strings. The "Requirements" section contains expressions which must be true for this process (i.e. hard constraints) the "Preferences" section contains expressions which are desired to be true (i.e. soft constraints). The "Conditions" and "Effects" section can be used for standard STRIPS-style expressions found in most AI planning systems. The "variables" section provides the same support as described in the node property sheet (see 5.10). Finally, the resources section defines the resource usage of the activities in the process.

6. Domain Editing

6.1 Status of Domain Editing Support

While much of the process editing support is in place, the domain editor implementation still needs much work. The tree presentation of the domain will eventually be able to support adding/deleting of domain items and dragging/drop of tree nodes onto the process desktop. A simple file format for saving and retrieving the domain will be added as well. Once this has been done, the process editor can be properly constrained to work within the bounds of a selected domain.

7. Intelligent Assistance

7.1 Common Process Assistant (CPA)

The common process assistant is currently implemented as a socket-based sicstus prolog program which can reason over the temporal relationships in a process specification designed using the CPE. The CPA maps the timepoint based relations from CPE into a set of Allen interval relationships. Transitivity relations are then used to infer the relationships between any three intervals. Using Allen's transitivity table for intervals, each relationship is examined to determine whether it is legal with respect to the other defined relationships. Any "abnormal" or "inappropriate" assignments are returned back to the editor for display to the user. This return of information from CPA to CPE has not been implemented yet. Future information passed back from the CPA may also contain the set of valid alternatives for the inappropriate relationship.

7.2 Checking for a Consistent Path

Native temporal checking of the process is also possible by scanning for a valid path given the partially ordered network. This module was presented in section 4.1.1. This tool works by collapsing all of the designated “equal” time points into single points. The total set of unique timepoints are then toplogically sorted based on the precedence relationships. If a valid linearization is possible the module will display it, otherwise it returns “No Solution!”. (See Figure 6).

7.3 State Viewing

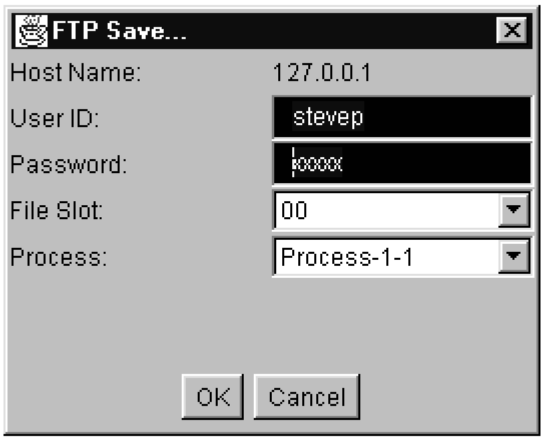
One of the future intelligent tool modules will allow an interactive query of the state at any point in the network. This will be implemented using a Modal Truth Criterion (MTC) or Question Answering (QA) approach. Additional state support may also be provided via the visualization of the process goal structure (GOST) and table of multiple process effects (TOME).

8. File Input/Output

Currently File Input/Output for the CPE applet is performed using the FTP protocol. At the time of writing this applet, the Java sandbox security model had prevented access to the local filestore. In order to provide a secure, reliable file input/output mechanism, FTP was selected for file transfer. There is a further constraint imposed by the Java security model which requires the FTP server to reside at the same location as the World Wide Web server. The next version will provide a “digitially signed” applet that has Java permission for writing to/from the local filestore.

8.1 Saving A Process

Process representations can be saved to a file area which can be accessed via FTP. The file format is discussed below. When the user selects “Save Process As…” from the main menu, the following dialog is displayed (see Figure 25). The hostname cannot be changed due to the constraint discussed above. The user ID and password required to access the FTP server must be entered. Currently, CPE uses a simple “fileslot” system for saving and retrieving files. This translates into a file called “cpe##.txt” where ## is the selected slot. The source process must be selected from a drop-down list of currently open processes.

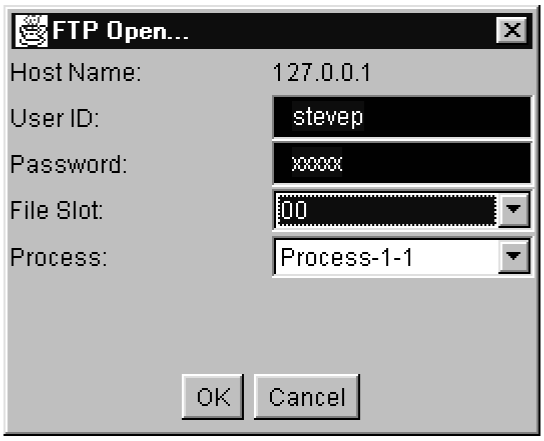


Host Name: 127.0.0.1
User ID: stevep
Password: xxxxx
File Slot: 00
Process: Process-1-1

Figure 25: CPE Save Dialog

8.2 Loading A Process

Process representations can be loaded from a file area which can be accessed via FTP. The file format is discussed below. When the user selects “Open Process…” from the main menu, the following dialog is displayed (see Figure 26). The hostname cannot be changed due to the constraint discussed above. The user ID and password required to access the FTP server must be entered. Currently, CPE uses a simple “fileslot” system for saving and retrieving files. This translates into a file called “cpe##.txt” where ## is the selected slot. The target process must be selected from a drop-down list of currently open processes.


FTP Open...

Host Name: 127.0.0.1

User ID: stevep

Password: xxxxx

File Slot: 00

Process: Process-1-1

Figure 26: CPE Open Dialog

8.3 Process File Format

The current CPE process file layout has a very simple structure. A single ASCII-based text file represents an entire process (along with sub-process information). A sample file which corresponds to the Figure 2 process network is shown in section 10. The future versions of CPE will use the Common Process Language (CPL) for native input/output.

The files are divided into multiple "line types". There are currently 5 types: comment_line, frame_line, node_line, edge_line, and text_line. The first character of each line determines which of these types a particular line represents: {%, F, N, E,T}. Comment lines (%) can contain any characters after the comment symbol and before the newline character.

CPE currently writes out 10 comment lines at the beginning of a file to provide information about the saved file. For example, the following text shows that this file was saved on Nov. 20, 1997 by a Macintosh user, using Netscape. Comment lines are also used to mark off file sections and to provide "header" information for line entries. The end of the file is marked by an %End line (note these comments are not required, but provide a more "readable" presentation).

```
%Common Process Editor, version 1.0, AIAI, 1997
%Date: Thu Nov 20 11:00:44 GMT+01:00 1997
%--
%java.version: 1.1.2
%java.vendor: Netscape Communications Corporation
%java.class.version: 45.3
%os.name: Mac OS
%os.arch: PowerPC
%os.version: 7.5
```

A frame_line contains the following data (Note that the x, y, width, and height data is no longer used). The unquoted vertical bar, | denotes optional values whereas the quoted bar '|'is a token in the file. The brackets, [ ] are used to indicate optional values.

- frame_line symbol: F<blank>
- x position: <integer><blank>
- y position: <integer><blank>
- frame width: <integer><blank>
- frame height: <integer><blank>
- current node count: <integer>
- process key: {<string>}
- process label: {<string>}
- top-level process flag: {[Y|N]}
- StartFinish flag: {[Y|N]}

- Precondition list: {[<string>’|’<string>’|’...<string>]}
- Effects list: {[<string>’|’<string>’|’...<string>]}
- Requirements list: {[<string>’|’<string>’|’...<string>]}
- Preferences list: {[<string>’|’<string>’|’...<string>]}
- Variables list: {[<string>’|’<string>’|’...<string>]}
- Resources list: {[<string>’|’<string>’|’...<string>]}

A node_line contains the following data. The unquoted vertical bar, | denotes optional values whereas the quoted bar ‘|’is a token in the file. The unquoted brackets, [ ] are used to indicate optional values.

- node_line symbol: N<blank>
- node key: {<string>}<blank>
- node x position: <integer><blank>
- node y position: <integer><blank>
- node type (activity or special): N|S<blank>
- node status: 0|1|2|3|4<blank>
- selected status: Y|N<blank>
- node label: {<line1>[~ <line2>~<line3>...<lineN>]}
- expand process key: {<string>}
- Variables list: {[<string>’|’<string>’|’...<string>]}

An edge_line contains the following data. The unquoted vertical bar, | denotes optional values whereas the quoted bar ‘|’is a token in the file.

- edge_line symbol: E<blank>
- from node key: {<string>}
- from node half: B|E
- to node key: {<string>}
- to node half: B|E
- link type: {<|=}

A text_line contains the following data. The unquoted vertical bar, | denotes optional values whereas the quoted bar ‘|’is a token in the file. The unquoted brackets, [ ] are used to indicate optional values.

- text_line symbol: T<blank>
- text label: {<line1>[~ <line2>~<line3>...<lineN>]}<blank>
- text x position: <integer><blank>
- text y position: <integer><blank>
- text selected flag: Y|N

9. Question and Answer

- Why Java?
  ⇒ The main reason for choosing Java was the platform independence. The tool can now be run on most of the main platforms without recoding the editor.

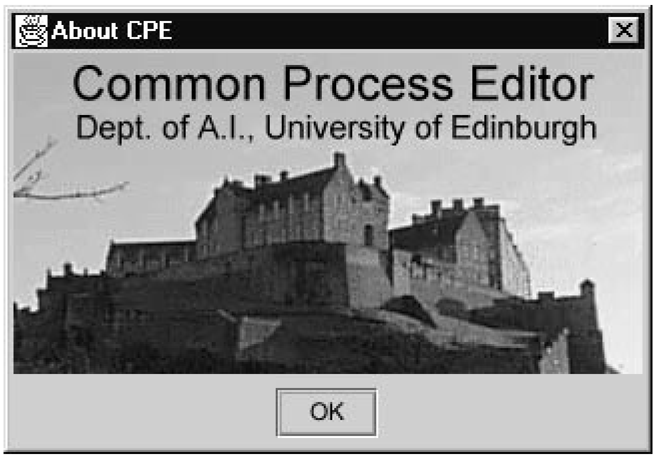
- How Do I Right-Click If I am Using a One-Button Mouse System (e.g. Macintosh)?
  ⇒ Hold down the “command” key when clicking.

- etc.

10. Appendix A: Sample Process File

This sample CPE process file corresponds to the process network diagram shown in Figure 2.

```
%Common Process Editor, version 1.0, AIAI, 1997
%Date: Mon Apr 06 19:20:34 GMT+00:00 1998
%--
%java.version: 1.2beta2
%java.vendor: Sun Microsystems Inc.
%java.class.version: 45.3
%os.name: Windows NT
%os.arch: x86
%os.version: 4.0
%--
%Frame: x,y,w,h,count,key,label,top,StartFinish,pre,eff,req,pref,var,res
F 0 0 800 800 4{Process-1-1}{Process-1-1}{Y}{ }{Y}{ }{ }{ }{ }{ }{ }
%--
%Nodes: N {key} x y type [N,S] , status [0-4], selected [Y,N], label, expand, {vars|}
N {Finish} 322 50 S 0 N{Finish}{ }{}
N {Start} 41 93 S 0 N{Start}{ }{}
N {act1} 160 39 N 0 N{act1}{Process-1-2 }{}
N {act2} 160 149 N 0 N{act2}{Process-1-3 }{}
N {act3} 270 143 N 0 N{act3}{ }{}
%--
%Edges: from, to
E {Start}E{act1}B{<}
E {act1}E{Finish}B{<}
E {Start}E{act2}B{<}
E {act2}E{act3}B{<}
E {act3}E{Finish}B{<}
%--
%Text: text, x, y, selected [Y,N]
%--
%Frame: x,y,w,h,count,key,label,top,StartFinish,pre,eff,req,pref,var,res
F 0 0 800 800 1{Process-1-2}{Process-1-2}{N}{Process-1-1 }{Y}{ }{ }{ }{ }{ }{ }
%--
%Nodes: N {key} x y type [N,S] , status [0-4], selected [Y,N], label, expand, {vars|}
N {Finish} 200 50 S 0 N{Finish}{ }{}
N {Start} 20 50 S 0 N{Start}{ }{}
%--
%Edges: from, to
E {Start}E{Finish}B{<}
%--
%Text: text, x, y, selected [Y,N]
%--
%Frame: x,y,w,h,count,key,label,top,StartFinish,pre,eff,req,pref,var,res
F 0 0 800 800 1{Process-1-3}{Process-1-3}{N}{Process-1-1 }{Y}{ }{ }{ }{ }{ }{ }
%--
%Nodes: N {key} x y type [N,S] , status [0-4], selected [Y,N], label, expand, {vars|}
N {Finish} 200 50 S 0 N{Finish}{ }{}
N {Start} 20 50 S 0 N{Start}{ }{}
%--
%Edges: from, to
E {Start}E{Finish}B{<}
%--
%Text: text, x, y, selected [Y,N]
%End
```


About CPE
Common Process Editor
Dept. of A.I., University of Edinburgh
OK