Process Interchange Format (PIF) for Sharing Ontologies

Submitted to the ECAI '96 Workshop on Ontological Engineering

Jintae Lee[1], Michael Gruninger[2], Yan Jin[3], Thomas Malone[4], Austin Tate[5], Gregg Yost[6] &
other members of the PIF Working Group[7]

Table of Contents

---

1. Abstract

This document provides the specification of the Process Interchange Format (PIF) version 1.1, whose goal is to develop an interchange format to help automatically exchange process descriptions among a wide variety of business process modeling and support systems such as workflow software, flow charting tools, process simulation systems, and process repositories.

Instead of having to write ad hoc translators for each pair of such systems, each system will only need to have a single translator for converting process descriptions in that system into and out of the common PIF format. Then any system will be able to automatically exchange basic process descriptions with any other system.

This document describes the PIF-CORE 1.1, i.e. the core set of object types (such as activities, agents, and prerequisite relations) that can be used to describe the basic elements of any process. The document also describes a framework for extending the core set of object types to include additional information needed in specific applications. These extended descriptions are exchanged in such a way that the common

---

[1] University of Hawaii, Department of Decision Sciences.
[2] University of Toronto, Department of Industrical Engineering
[3] Stanford University, Department of Civil Engineering
[4] MIT, Center for Coordination Science
[5] University of Edinburgh, Computer Sciences Department
[6] Digital Equipment Corporation

[7] The PIF Working Group consists of people from industry and academia who are actively participating toward the development of PIF. The PIF Working Group's activities are supported by ARPA, NSF, Corporate Sponsors of the MIT Center for Coordination Science, and Sponsors and Organizations of the individual PIF Working Group members. The PIF Home Page can be found at http://soa.cba.hawaii.edu/pif/. We are open to any comment and suggestions. Please address them to pif-comments@mit.edu or jl@hawaii.edu (Jintae Lee Dept. of Decision Sciences. Univ. of Hawaii, 2401 Maile Way. Honolulu, HI 96822).

elements are interpretable by any PIF translator and the additional elements are interpretable by any translator that knows about the extensions.

The PIF format was developed by a working group including representatives from several universities and companies and has been used for experimental automatic translations among systems developed independently at three of these sites. This document is being distributed in the hopes that other groups will comment upon the interchange format proposed here and that this format (or future versions of it) may be useful to other groups as well. The PIF Document 1.0 was released in December 1994, and the current document reports the revised PIF that incorporate the feedback received since then.

2. Introduction

More and more companies today are attempting to improve their business by engaging in some form of business process redesign (BPR). BPR focuses on a "process view" of a business and attempts to identify and describe an organization's business processes; evaluate the processes to identify problem areas; select or design new processes, possibly radically different from those currently in place; predict the effects of proposed process changes; define additional processes that will allow the organization to more readily measure its own effectiveness; and enact, manage and monitor the new processes. The goal is a leaner, more effective organization that has better insight into how it does business and how its business processes affect the organization's health. Successful BPR projects involve the cooperation of many people over extended time periods, including workplace analysts, systems engineers, and workers at all levels of the organization.

Computer applications that support one or more aspects of BPR are becoming increasingly common. Such applications include

- Modeling tools that help a workplace analyst identify and describe an organization's processes.
- Process library browsers that help organizations find new processes that might better meet their needs.
- Process animators and simulators that help organizations visualize the effects of existing processes or potential new processes.
- Workflow management tools that help workers follow business processes.
- Outcomes analysis tools that help organizations monitor the effectiveness of their processes.

No single application supports all aspects of a BPR engagement, nor is it likely that such an application will ever exist. Furthermore, applications that do support more than one aspect rarely do them all well. For example, a workflow tool may also provide some process simulation capabilities, but those additional capabilities are unlikely to be on par with the best dedicated simulation applications. This is to be expected -- building an application that supports even one of these aspects well requires a great deal of specialized knowledge and experience.

Ideally, then, a BPR team would be able to pick a set of BPR-support applications that best suits their needs: a process modeling tool from one vendor, a simulator from another, a workflow manager from another, and so forth. Unfortunately, these applications currently have no way to interoperate. Each application typically has its own process representation (often undocumented), and many applications do not provide interfaces that would allow them to be easily integrated with other tools.

Our goal with the PIF project is to support the exchange of business process descriptions among different process representations. The PIF project supports sharing process descriptions through a description

format called PIF (Process Interchange Format) that provides a bridge across different process representations. Tools interoperate by translating between their native format and PIF.

Any process description format, including PIF, is unlikely to ever completely suit the needs of all applications that make use of business process descriptions. Therefore, in addition to the PIF format, we have defined a framework around PIF that accommodates extensions to the standard PIF description classes. The framework includes a translation scheme called Partially Shared Views that attempts to maximize information sharing among groups that have extended PIF in different ways.

The PIF framework aims to support process translation such that:

- Process descriptions can be automatically translated back and forth between PIF and other process representations with as little loss of meaning as possible. If translation cannot be done fully automatically, the human efforts needed to assist the translation should be minimized.
- If a translator cannot translate part of a PIF process description to its target format, it should:
  - Translate as much of the description as possible (and not, for example, simply issue an error message and give up)
  - Represent any untranslatable parts in a way that lets a person understand the problem and complete the translation manually if desired
  - Preserve any uninterpretable parts so that the translator can add them back to the process description when it is translated back into PIF.

These requirements on the translators are very important. We believe that a completely standardized process description format is premature and unrealistic at this point. Therefore, as mentioned earlier, we have provided ways for groups to extend PIF to better meet their individual needs. As a result, we expect that PIF translators will often encounter process descriptions written in PIF variants that they can only partially interpret. Translators must adopt conventions that ensure that items they cannot interpret are available for human inspection and are preserved for later use by other tools that are able to interpret them. Section 5 describes PIF's Partially Shared Views translation scheme, which we believe will greatly increase the degree to which PIF process descriptions can be shared.

3. History and Current Status

The PIF project began in October 1993 as an outgrowth of the Process Handbook project at MIT and the desire to share process descriptions among a few groups at MIT, Stanford, the University of Toronto, and Digital Equipment Corporation. The Process Handbook project at the MIT Center for Coordination Science aims to create an electronic handbook of process models, their relations, and their tradeoffs. This handbook is designed to help process designers analyze a given process and discover innovative alternatives. The Spark project at Digital Equipment Corporation aims to create a tool for creating, browsing, and searching libraries of business process models. The Virtual Design Team (VDT) project at Stanford University aims to model, simulate, and evaluate process and organization alternatives. The Enterprise Modeling project at the University of Toronto aims to articulate well-defined representations for processes, time, resources, products, quality, and organization. These representations support software tools for modeling various aspects of enterprises in business process reengineering and enterprise integration.

In one way or another, these groups were all concerned with process modeling and design. Furthermore, they stood to benefit from sharing process descriptions across the different representations they used. For example, the Enterprise Modeling group might model an existing enterprise, use the Process Handbook to

analyze its tradeoffs and explore its alternatives, evaluate the different alternatives via VDT simulation, and then finally translate the chosen alternative back into its own representation for implementation.

Over the past years, through a number of face-to-face, email, and telephone meetings, members from each of the groups have:

- Articulated the requirements for PIF
- Specified the core PIF process description classes
- Specified the PIF syntax
- Elaborated the Partially Shared View mechanism for supporting multiple, partially overlapping class hierarchies
- Created and maintained a database of the issues that arose concerning PIF's design and the rationales for their resolutions
- Implemented several translators, each of which translated example process descriptions (such as a portion of the ISPW-6 Software Change Process) between PIF and a group's own process representation.

Based on this work, the PIF Document 1.0 was released on December, 1994. Since then, we have received a number of questions and comments on topics that range from individual PIF constructs to how certain process descriptions can be represented in PIF. We have been also assessing the adequacy of the PIF 1.0 by testing it against more complex process descriptions than before. AIAI at the University of Edinburgh also joined the PIF Working Group at this time bringing along their interests in planning, workflow and enterprise process modeling. The Edinburgh group is also providing a valuable service as a liaison between the PIF group and the Workflow Management Coalition as well as the AI planning community which has been concerned with the activity representation issues for a while.

The revised structure of PIF reflects the lessons extracted from these external and internal input. In particular, two points emerged clearly. One is that the PIF-CORE has to be reduced to the bare minimum to enable translation among those who cannot agree on anything else. The other point is the importance of structuring PIF as a set of modules that build on one another. This way, groups with different expressive needs can share a subset of the modules, rather than the whole monolithic set of constructs. As a result, the PIF-CORE has been reduced to the minimum that is necessary to translate the simplest process descriptions and yet has built-in constructs for "hanging off" modules that extend the core in various ways.

4. PIF

The PIF ontology has grown out of the efforts of the PIF Working Group to share process descriptions among the group members' various tools. We have used the following guidelines in developing this hierarchy:

- Generality is preferred over computational efficiency when there is a tradeoff, for the reason that PIF is an interchange language, not a processing language. Therefore, the organization of the entity classes is not necessarily well-suited to performing any particular task such as workflow management or process simulation. Instead, our goal has been to define classes that can express a wide variety of processes, and that can be readily translated into other formats that may be more suitable for a particular application.

- The PIF constructs should be able to express the constructs of some existing common process representations such as IDEF (SADT) or Petri Nets.

- PIF should start with the minimal set of classes and then expand only as it needs to. The minimal set was decided at the first PIF Workshop (October 1993) by examining those constructs common to some major existing process representations and to the process representations used by members of the PIF Working Group.

- Additions to the standard PIF classes could be proposed by anybody, but the proposal must be accompanied by concrete examples illustrating the need for the additions. The Working Group decided, through discussions and votes if necessary, whether to accept the proposal. PIF allows groups to define local extensions at will (see Section 5), so new classes or attributes should be added to the standard PIF classes only if they seem to be of sufficiently general usefulness.

A PIF process description consists of a file of entities, such as ACTIVITY, OBJECT, and TIME-POINT entities. Each entity in the file has a unique id that other entities can use to refer to it. Each entity type (or class) has a particular set of attributes defined for it; each attribute describes some aspect of the entity. For example, an ACTIVITY entity has a Status attribute reporting the status of the activity. Entity classes fall into a class hierarchy (see Figure 1). Each entity class has all of the attributes of all of its superclasses, in addition to its own attributes. For example, all PIF entities have a Name attribute, since the class at the root of the PIF hierarchy (ENTITY) has a Name attribute.

When an attribute of one entity has another entity as its value, the attribute represents a relationship between the two entity classes. For example, an ACTIVITY's Begin attribute takes a TIME-POINT as its value, so the Begin attribute represents a relationship between the ACTIVITY and TIME-POINT classes. Figure 2 depicts the relationships among the PIF classes. All of the current PIF classes are defined in the Section 1 of the Supporting Document.

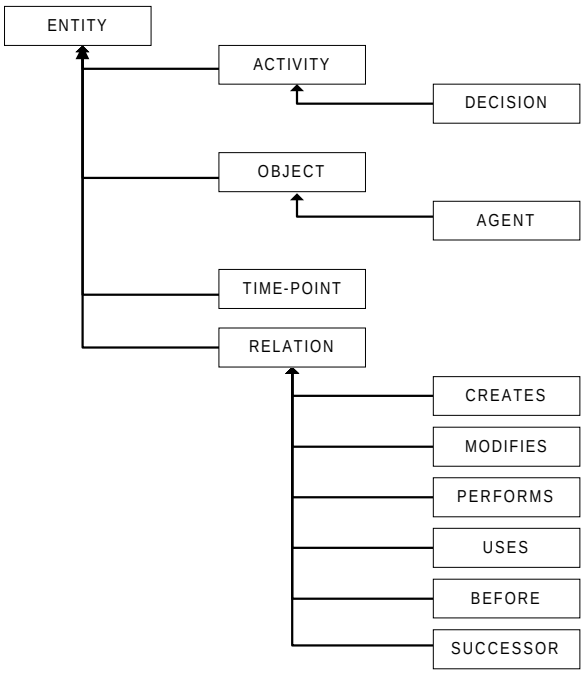


Figure 1: The PIF class hierarchy.

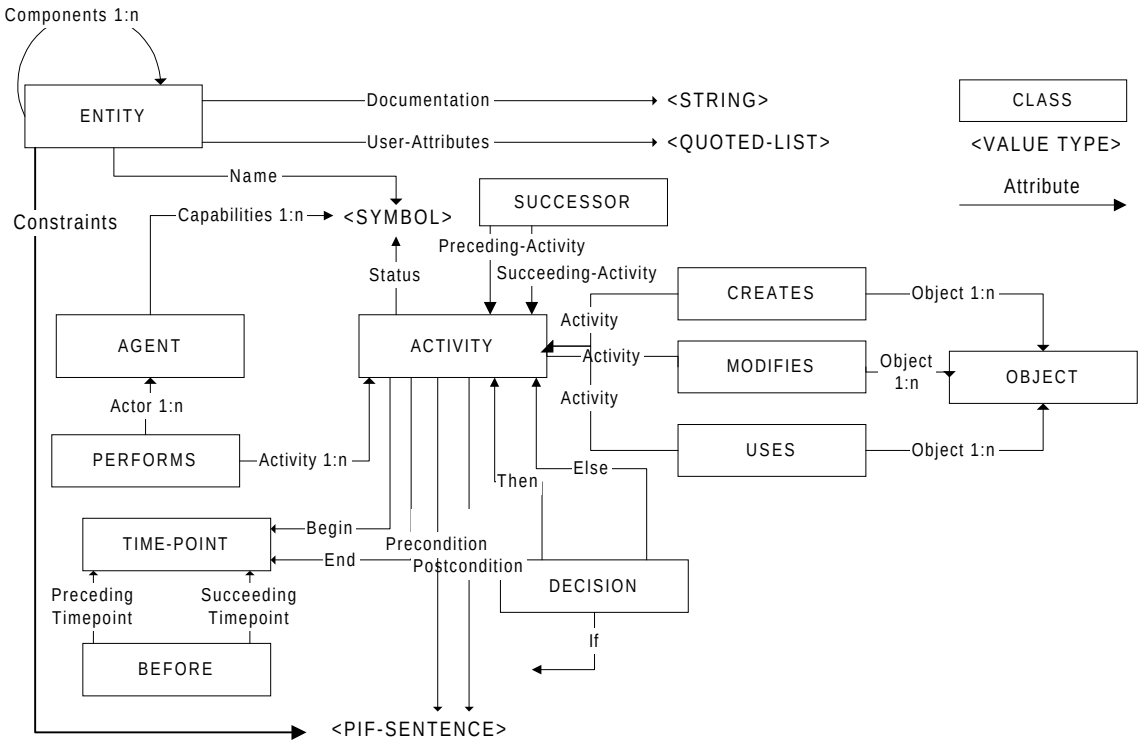


Figure 2: Relationships among PIF classes.

PIF Value Types

The value of an attribute in a PIF entity is one of the following: the unique identifier of another PIF entity, a literal value of PIF primitive value types, an expression of composite value types, or an object variable.

.

The PIF primitive value types consist of: NUMBER, STRING, and SYMBOL

- NUMBER: A numeric value. The NUMBER type is subdivided into INTEGER and FLOAT types.
- STRING: A sequence of characters.
- SYMBOL: Symbols are denoted by character sequences, but have somewhat different properties than strings. PIF symbols are a much-simplified version of symbols in the Lisp programming language (Steele, 1990). In PIF, the main difference between strings and symbols is that symbols are not case-sensitive unless specially quoted, but strings are always case-sensitive.

The PIF composite value types consist of: QUOTED-LIST and PIF-SENTENCE.

- QUOTED-LIST: A nested list structure that is quoted.
- PIF-SENTENCE: A logical expression that evaluates to TRUE or FALSE.

An object variable is of the form, object-name[.slot-name]*, which refers to either the object named or the object which is the value of the named slot (or, if there are more than one slot-names specified, the object which is the value of the named slot of the object which is the value of the next named slot, and so on.) Appendix I describes PIF's syntax, including the syntax of the primitive literals as well as the composite value types.

Rationales

The goal of PIF is to support maximal sharing of process descriptions across heterogeneous process representations. To better serve this goal, PIF consists of not a monolithic set of constructs, but a lattice of modules. A module can build on other modules in that the constructs in a module are specializations of the constructs in the other modules. One can adopt some modules but not others depending on one's expressive needs. Hence, a module typically contains a set of constructs that are useful for a particular domain or a type of task. More details of this module structure is discussed in Section 5.

The PIF-CORE, on the other hand, consists of the minimal set of constructs necessary to translate "simple" process descriptions "commonly" encountered in the real world. There is a tradeoff between simplicity and commonality. The PIF-CORE could have been chosen to contain only the constructs necessary for describing the simplest process descriptions such as a precedence network. Such a PIF-CORE then would not be able to translate many process descriptions. On the other hand, the PIF-CORE could have contained constructs sufficient for describing process descriptions of richer complexity. Such a PIF-CORE then would contain many constructs that may not be needed for many simpler descriptions. The PIF-CORE strikes a balance in this tradeoff by first collecting process descriptions, starting from the simplest and continuing with more complex until we have reasonably many of them, and then by looking for a set of constructs that can translate the process descriptions in this collection. The following describes the rationales for each of the constructs in the PIF-CORE. The attributes of each of these constructs are described in the Supporting Document: Section 1. Section 3 of the Supporting Document provides the complete specification of the PIF-CORE 1.1.

In PIF, everything is an ENTITY; that is, every PIF construct is a specialization of ENTITY. There are four types of ENTITY: ACTIVITY, OBJECT, TIME-POINT, and RELATION. These four types are derived from the definition of process in PIF: a process is a set of ACTIVITIES that stand in certain RELATIONS to one another and to OBJECTS over TIME-POINTS.

The following provides intuitive rationales for each of these four constructs. Their precise semantics, however, are defined by the relations they have with other constructs (cf. Supporting Document: Section 1).

ACTIVITY represents anything that happens over time. DECISION, which represent conditional activities, is the only special type of ACTIVITY that the PIF-CORE recognizes. In particular, the PIF-CORE does not make any distinction among process, procedure, or event. A TIME-POINT represents a particular point in time, for example "Oct. 2, 2.32 p.m. 1995" or "the time at which the notice is received." An OBJECT is intended to represent all the types of entities involved in a process description beyond the other three primitive ones of ACTIVITY, TIME-POINT, and RELATION. AGENT is a special type of OBJECT.

RELATION represents relations among the other constructs. The PIF CORE offers the following relations: BEFORE, CREATES, USES, MODIFIES, and PERFORMS. BEFORE represents a temporal relation between TIME-POINTS. CREATES, USES, and MODIFIES represent relations between ACTIVITY and OBJECT. PERFORMS represents a relation between AGENT and ACTIVITY. SUCCESSOR (Activity-1, Activity-2) is defined to be the relation between ACTIVITIES where BEFORE (Activity-1.End, Activity-2.Begin) holds.

SUCCESSOR in PIF may not correspond exactly to the notions of successor as used in some workflow or enactment systems because it is common in these systems to bundle into a single relationship a mixture of temporal, causal, and decomposition relationships among activities. PIF provides precise, separate relationships for all three of these activities-to-activity specifications. For example, the temporal relationship

is specified with the BEFORE relation, the causal relation with the Precondition and Postcondition attributes of ACTIVTIY, and the decomposition relation with the Components attribute. Its intention is to allow the exact meaning to be communicated. Hence, one might have to combine some of these constructs to capture exactly the meaning of SUCCESSOR as used in one's own system.

The attribute value of a PIF object can be one of the PIF value types specified above. The PIF primitive value types consist of NUMBER, STRING, and SYMBOL. The PIF composite value types are QUOTED-LIST and PIF-SENTENCE. QUOTED-LIST is used for conveying structured information that is to be evaluated by an PIF interpreter, but simply passed along (e.g. as in the User-Attributes attribute of ENTITY). PIF-SENTENCE is used to specify a condition that is either true or false, as required, for example, for the Precondition and the Postcondition attributes of ACTIVITY.

PIF-SENTENCE is logical expression that may include variables, quantifiers, and the Boolean operators for expressing condition or constraint. A PIF Sentence is used in the Constraints slot of ENTITY, the Precondition and the Postcondition slots of the ACTIVITY, and the If slot of the DECISION. A variable in a PIF-SENTENCE takes the following positions in the three dimensions that define the possible usage.
(1) The scope of the variable is the object. That is, variables of the same name within an object are bound to the same object, whereas they are not necessarily so if they occur in different objects.
(2) A variable is assumed to be bound by an implicit existential quantifier.
(3) The constraints on variables in an object are expressed in the Constraints slot of that object. These constraints are local to the object.

These positions are expected to be extended by some PSV Modules. Some PSV modules will extend the scope of a variable beyond a single object. Some will introduce explicit existential and universal quantifiers. Yet others will allow global constraints to be stated, possibly by providing an object where such global constraints that hold across all the objects in a PIF file (e.g. All purchase order must be approved by the finance supervisor before sent out.).

Notable Absence:

We have decided not to include ROLE because a role may be defined wherever an attribute is defined. For example, the concept of RESOURCE is a role defined by the Resource attribute of the USE relation. Any object, we view, is a resource if it can be USED by an ACTIVITY. As a consequence, we have decided not to include ROLE or any construct that represents a role, such as RESOURCE. ACTOR is not included in PIF because it is another role-concept, one defined by the Actor attribute of ACTIVITY. Any object, as long as it can fill the Actor attribute, can be viewed as an ACTOR. Hence we resolved that explicit introduction of the constructs such as ACTOR or RESOURCE is redundant and may lead to potential confusions. We should note, however, that the PIF core provides the construct AGENT, which is not defined by a role an entity plays but by its inherent characteristic, namely its Capabilities (for example, of making intelligent decisions in various domains).

5. Extending PIF

PIF provides a common language through which different process representations can be translated. Because there will always be representational needs local to individual groups, however, there must also be a way to allow local extensions to the description classes while supporting as much sharing as possible

among local extensions. The Partially Shared Views (PSV) scheme has been developed for the purpose (Lee & Malone, 1990). PSV integrates different ways of translating between groups using different class hierarchies (e.g. pairwise mapping, translation via external common language, translation via internal common language) so as to exploit the benefits of each when most appropriate.

A PSV Module is a declaration of PIF entities which specialize other entities in the PIF-CORE or other PSV modules on which it builds. The class definitions in a PSV Module cannot delete or alter the existing definitions but can only add to them. Examples of PSV Modules are given at the end of this section. A group of users may adopt one or more PSV Modules as necessary for its task.

A group using a PSV module translates a PIF object X into their native format as follows:

1. If X's class (call it C) is known to the group and the group has developed a method that translates objects of class C into their native format, then apply that translation method. C is known to the group if either C is defined in one of the PSV Modules that the group has adopted or the group has set up beforehand a translation rule between C and a type defined in one of the PSV Modules adopted.
2. Otherwise, translate X as if it were an object of the nearest parent class of C for which (1) applies (i.e. its parent class in the most specific PSV Module that the group and the sender group both share, i.e. have adopted).

This translation scheme allows groups to share information to some degree even if they do not support identical class hierarchies. For examples, suppose that Group A supports only the standard PIF AGENT class, and that Group B in addition supports an EMPLOYEE subclass. When Group A receives a process description in Group B's variation on PIF, they can still translate any EMPLOYEE objects in the description as if they were AGENT objects. What happens to any information that is in an EMPLOYEE object that is not in a generic AGENT object? That will vary according to the sophistication of the translator and the expressive power of the target process representation. Ideally, the translator will preserve the additional information so that it can be viewed by users and reproduced if it is later translated back into PIF.

For example, suppose EMPLOYEE has a "Medical-plan" attribute, which is not part of the AGENT object in the PIF-CORE. Then, ideally, Group A's translator would

- Translate any Medical-plan attributes into a form that the user could view in the target system (even if it only as a textual comment)[8] AND
- When the information is re-translated into PIF in the future (from Group A's native format), it is emitted as an EMPLOYEE object with the same value for the Medical-plan attribute (and not simply as an AGENT object with no Medical-plan attribute). MIT researchers are currently investigating this general problem of preserving as much information as possible through "round trips" from one representation to another and back (Chan '95).

Translators that can follow these conventions will minimize information loss when processes are translated back and forth between different tools. The details of PSV can be found in (Lee & Malone, 1990). In the current version of PIF, each PIF file begins with a declaration of the class hierarchy for the objects described in the file. PSV uses this class hierarchy to translate objects of types that are unknown to a translator. To eliminate the need for PIF translators to do any other inheritance operations, however, all PIF objects should contain all of their attributes and values. For instance, even if the value of a given attribute is inherited without change from a parent, the attribute and value are repeated in the child.

---

[8] If the target representation happens to be PIF (albeit Group A's variant of it), the uninterpretable attributes would be stored as text in the User-Attribute attribute, which all PIF entities have

As the number of PSV modules grows large, we need a mechanism for registering and coordinating them so as to prevent any potential conflict such as naming conflict. Although the exact mechanism is yet to be worked out, we are envisioning a scenario like the following. The user who needs to use PIF would first consult the indexed library of PSV modules, which documents briefly the contents of each of the modules and the information about the other modules it presupposes. If an existing set of modules does not serve the user's purpose in hand and a new PSV module has to be created, then the information about the new module and its relation to other modules is sent to a PSV registration server, which then assigns to it a globally unique identifier and updates the indexed library. We foresee many other issues to arise such as whether any proposed PSV module should be accepted, if not who decides, whether to distinguish an ad-hoc module designed for temporary quick translation between two local parties from a well-designed module intended for global use, and so on. However, rather than addressing these issues in this document, we will address them in a separate document as we gain more experience with PSV modules.

Example of PSV Modules

To date, two PSV Modules have been specified: Temporal-Relation-1 and IDEF-0 Modules. The Temporal-Relation-1 Module is specified in Appendix III. It extends the core PIF by adding all possible temporal relations that can hold between two activities (cf. Fig. 3). The IDEF-0 Module adds the constructs necessary for translating between IDEF-0 descriptions and PIF. IDEF-0 is a functional decomposition model, which however has been historically used widely as a process model description language. IDEF-0 has been used in various ways with no single well-defined semantics. Hence, the IDEF-0 PSV Module supports translation between PIF and one particular version of IDEF-0. It introduces two additional relations, USES-AS-RESOURCE and USES-AS-CONTROL, as specializations of the USES relation. They are meant to capture the Control and Mechanism input of IDEF-0. The Input and Output relations of IDEF-0 may be translated into PIF by using the Precondition and Postcondition attribute of ACTIVITY. The IDEF-0 Module is specified in Appendix IV. The mapping between IDEF and PIF is shown in Fig. 4. These modules have not been officially registered. They are presented here only to provide examples of PSV modules. We are soliciting further inputs before we register them.

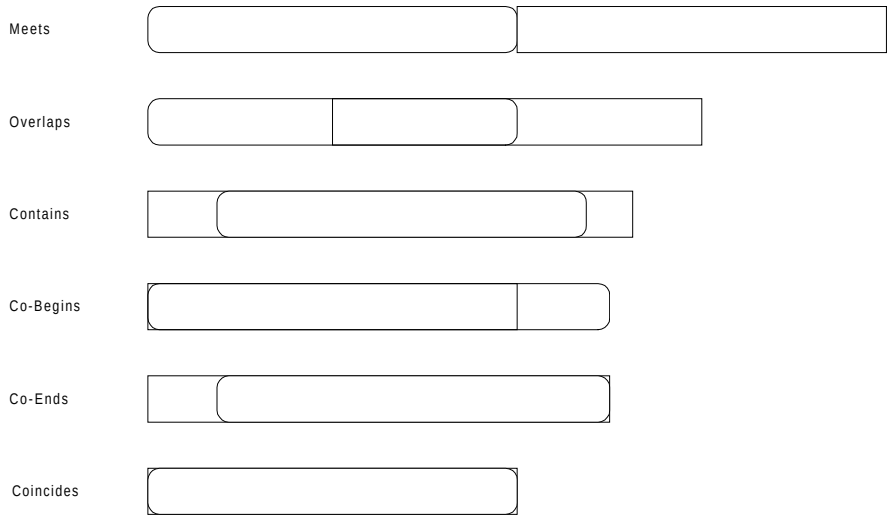
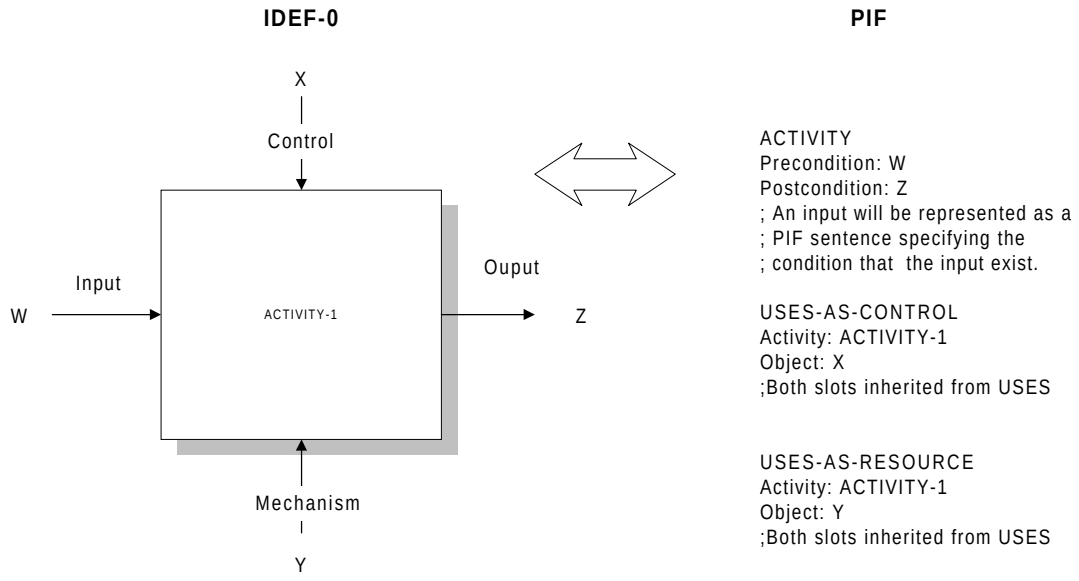


Figure 3. Possible Temporal Relations between Two Activities



---

6. References

Chan, F. Y. 1995 The Round Trip Problem: A Solution for the Process Handbook. Unpublished Master's Thesis. MIT Dept. of Electrical Engineering and Computer Science, May 1995.

Genesereth, M. & Fikes, R. (1992). Knowledge Interchange Format v.3 Reference Manual. Available as a postscript file via anonymous ftp from www-ksl.stanford.edu:/pub/knowledge-sharing/papers/kif.ps.

Gruber, T. (1993). Ontolingua: A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 199-200. Available via anonymous ftp from www-ksl.stanford.edu:/pub/knowledge-sharing/papers/ongolingua-intro.ps

Lee, J. & Malone, T. (1990). Partially Shared Views: A scheme for communicating between groups using different type hierarchies. ACM Transactions on Information Systems, 8(1), 1-26.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R. Senator, T., & Swartout, W. R.(1991). Enabling technology for knowledge sharing. AI Magazine, 12(3), 16-36.

Steele, G. (1990). Common Lisp: The language. Second edition. Digital Press.

Tate A, (1995) Characterizing Plans as a Set of Constraints - the <I-N-OVA> Model - a Framework for Comparative Analysis. ACM SIGART Bulletin, Special Issue on "Evaluation of Plans, Planners, and Planning Agents", Vol. 6 No. 1, January 1995. Available as a postscript file via ftp://ftp.aiai.ed.ac.uk/pub/documents/1995/95-sigart-inova.ps

Uschold, M., M. King, S. Moralee, and Y. Zorgios (1995) The Enterprise Ontology. Available via WWW URL http://www.aiai.ed.ac.uk/~entprise/enterprise/ontology.html