A Common Process Ontology for Process-Centred Organisations

Stephen T. Polyak and Austin Tate

Artificial Intelligence Applications Institute (AIAI)
Institute for Representation and Reasoning (IRR)
Division of Informatics, The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, United Kingdom
Steve_Polyak@ed.ac.uk, a.tate@ed.ac.uk

Abstract

The world of business and organised work is changing. This change is driven by a shift of organisational focus away from individual fragmented tasks toward an examination of the holistic processes. New tools are being developed to assist individuals in building, evaluating, and managing these processes. The application of these tools though must be holistic as well. Organisational knowledge management should be structured in a way that encourages exchange of process knowledge. In order to effectively share information, we believe there must be an explicit account of what knowledge will be exchanged, a shared understanding. We approach this by providing an extensible ontology which presents process related concepts and terminology which are common to a range of applications and industries.

Keywords: Ontology, Process design, Knowledge sharing

1 Introduction

"The key word in the definition of reengineering is **process**: a complete end-to-end set of activities that together create value for a customer...the process-centred organisation is creating a new economy and a new world." [15]

1.1 Process-Centred Organisations

The world of business and organised work is changing indeed. The move is away from outdated modes which were acquired during the industrial revolution [16] and toward new models of work centred around holistic knowledge of the organisation's value-adding processes. With this new movement comes new challenges and new issues to face [7, 9]. Process-centred individuals (e.g. process owners, process performers, process inspectors) are struggling to adjust to their new roles and duties, cf. [15, 4]. For a process owner, this may involve familiarising themselves with concepts of process design, re-design, abstraction, and effective communication of process knowledge. Process performers are expected to understand how their activities relate to other activities which may be performed by other people, computer systems, or themselves and how this all relates to an overall specification. Both process owners and process performers must also practice the art of donning their process inspector cap in order to evaluate whether the designed processes are working efficiently in the face of actual process enactment.

1.2 The Common Process Framework

In parallel with this development, a lot of work has been done in various technology fields which may assist "process professionals" in tackling this new process-aware world. These advancements include areas such as: process/knowledge modelling; event simulation; artificial intelligence planning; requirements engineering; knowledge acquisition; and decision

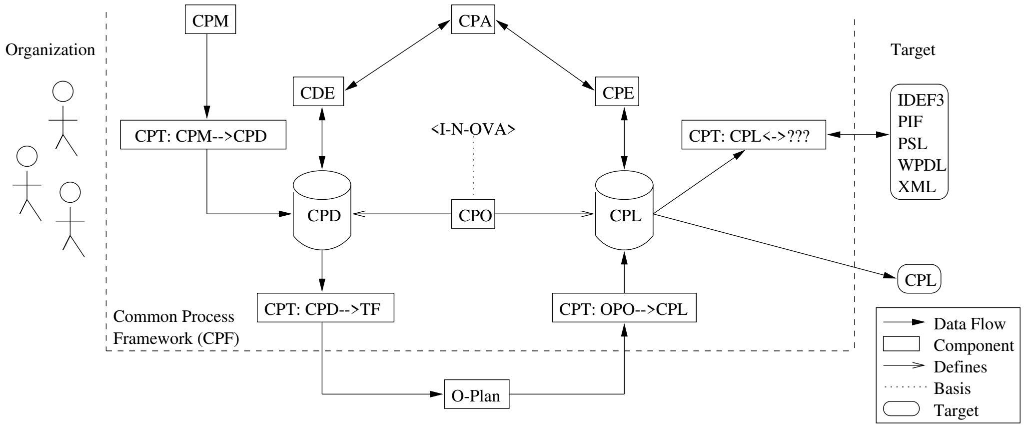

Figure 1: The Common Process Framework (CPF)

support. Unfortunately, past attempts at providing technology to support business activities have shown themselves to be encouragers of the very atomistic behaviour that a process-centred organisation hopes to overthrow. Organisations found themselves with various pockets of enterprise knowledge locked up in representations spread widely over a range of tasks. What is required of business is, in fact, what is also required in the technology to support process-centred businesses. That is, a technology-enabled process for managing organisational process knowledge should also be "a complete end-to-end set of activities that together create value for a customer", only this time the customer is the organisation which owns the processes.

In our work, we have sought to define a Common Process Framework (CPF) which encompasses a life-cycle of activities surrounding the management of process knowledge. Our approach is centred around knowledge-rich processes which are based on past and present work in Artificial Intelligence (AI) planning and plan representations. In addition to this, we have also gained insight through our involvement with standards work relating to the exchange of process knowledge.

1.3 Process Standards Work

We have have been involved in and worked with a number of initiatives to standardise shared languages within the general subject area of activities and processes. These efforts include:

- **enterprise processes** with the Process Interchange Format [19, 29]. The goal of the PIF Project is to develop an interchange format to help automatically exchange process descriptions among a wide variety of business process modelling and support systems such as flow charting tools, process simulation systems, and process repositories.

- **workflow processes** using the International Workflow Management Coalition's (WfMC) Workflow Process Definition Language (WPDL) [48, 27]. The Coalition's mission is to promote and develop the use of workflow through the establishment of standards for software terminology, interoperability and connectivity between workflow products.

- **AI planning-based processes** in the Shared Planning and Activity Representation (SPAR) project [39]. The Shared Planning and Activity Representation (SPAR)[1] is intended to contribute to a range of purposes including domain modelling, plan generation, plan analysis, plan case capture, plan communication, behaviour modelling, etc.

- **manufacturing processes** via the National Institute for Standards and Technology (NIST) Process Specification

[1] Details on the Shared Planning and Activity Representation (SPAR) are available via the WWW at: http://www.aiai.ed.ac.uk/~arpi/spar/

Language [32, 30]. NIST's Process Specification Language (PSL) project is motivated by a need for an integrated manufacturing environment in which process information can be shared amongst various applications.

In all of these projects or standards, we can envision a common ontology of processes which offer core concepts and terminology for expressing process knowledge. These definitions seek to disambiguate what is meant by the various terms and how they relate. The ontology needs to be extensible in order to be stretched to fit various domains and applications. The intent of this paper is to describe the Common Process Ontology (CPO) which is part of the CPF for managing process knowledge.

We can visualise the role of CPO in Figure 1. On one hand, we have the organisation: the actual processes it enacts; the activities and the agents who perform them; etc. On the other side we have a variety of process specification formats that describe these actual processes and which may link into application-specific environments (e.g. manufacturing environments with PSL or IDEF3, workflow tool-sets using WPDL, business environments which are PIF-aware, or web-based exchanges using the Extensible Markup Language, XML).

Figure 1 illustrates that the CPO is a central component within the framework. As we will discuss below, CPO's basis is the <I-N-OVA> constraint model of activity [37, 36]. CPO's defined terms are used to express both domain knowledge (Common Process Domain (CPD) files) and knowledge of individual processes (Common Process Language (CPL) files). Correspondingly, the framework encompasses editors for both sources of knowledge (Common Domain Editor, CDE and Common Process Editor, CPE). Both editors communicate with a web-based assistant (Common Process Assistant, CPA) which currently provides feedback for users on temporal relationships between process activities. The toolset supports both FTP and local filesystem access to an organisation's process knowledge. The Common Process Methodology (CPM) provides the initial starting point in which CPD knowledge is elicited and built in a structured way.

The interoperability of CPF is established via the use of translators (CPT) into and out of the core CPO terms. These exchanges may take place both within the framework and between the framework and external tools. For example, the translation between the CPM domain descriptions (which are largely graphical) and the CPD specifications (i.e. CPT: CPM $\rightarrow$ CPD) are internal, whereas the interoperability with the O-Plan AI planning system [6] is external. In our work, we have used O-Plan to synthesise new processes based on translated domains which are expressed in O-Plan's Task Formalism (TF) language [35, 40]. We have enriched the O-Plan plan output format (OPO) to include knowledge of causal relationships, dependencies, and resource commitments which we translate into CPL.

In the following sections we provide an overview as well as a detailed examination of the ontology. Examples are provided along with an analysis of the ontology using a range of features.

2 What is an Ontology?

The concept of "ontology" is drawn from philosophy in which it is used to indicate a systematic theory about existence. The use of this term in computational settings (e.g. in information systems, or artificial intelligence applications) tends to vary depending on particular needs and perspectives. On one extreme, people may refer to an ontology as simply a lexicon of terms for a particular application (e.g. for an automotive domain we might have: WHEEL, BODY, ENGINE, BRAKE, etc.) while on the other end of the spectrum they may mean a particularly rigorous set of logical axioms which provide detailed terms and definitions. See [43] for an overview of this range of formality and for an introduction to this field. For our purposes, we will use the following definition

"An ontology is a vocabulary of terms (names of relations, functions, individuals) defined in a form that is both human and machine readable. An ontology, together with a kernel syntax and semantics, provides the language by which knowledge-based systems can interoperate at the knowledge-level." [14]

As this definition implies, ontologies typically need to be referred to and inspected by people. People review parts or all of the ontology in order to align themselves with the "shared understanding" of the set of concepts (e.g. during translation writing, or in clarifying assumptions). Ontologies are also machine readable in order to provide automated support for tasks such as ontological model checking, cf. [18].

Recently, it was pointed out that a detailed description of a "space of ontology applications" can be used to characterise a particular ontology [44, 3]. This characterisation aids in, among other things, promoting cross-evaluation of ontologies and identification for reuse. After we have presented the ontology, we will characterise CPO based on the attributes identified in [42].

3 CPO: An Overview

In the ontological engineering methodology, Methontology, one of the very first steps prescribed is "specification" [10, 13]. This step is meant to encourage ontology authors to address questions of purpose and scope straight away. Example issues include: "why is this ontology being built?" and "who are the intended users?". We have begun to outline the purpose and scope of CPO in the introduction, but we shall now provide a set of categories for considering these points. As in [37], we view CPO as contributing to four main areas or categories of purpose:

- **Knowledge acquisition** - assisting individuals in collecting knowledge about their organisational processes.
- **User communication** - structuring the content of process knowledge to be shared amongst individuals.
- **Formal analysis** - providing definitions for process knowledge which may be mapped to representations for automated reasoning.
- **System manipulation** - underpinning a lingua franca for sharing information between various tools or systems.

Given these intended categories of purpose along with a variety of input sources, we outlined a requirements specification for CPO in order to further delimit the scope [23]. These requirements were separated into representational (i.e. what do we want to express?) and functional (i.e. what are the intended uses of the knowledge?). Within each we provided an additional clustering of requirements around concepts (e.g. activities, agents, evaluations, etc.) or uses (e.g. editing, execution, task assignment, etc.).

One of the sources of input for this specification included the set of requirements developed for NIST's manufacturing-based Process Specification Language (PSL) [32]. These requirements were drawn from a range of process management tools and applications. During our initial involvement with the PSL project, we provided an analysis of how well several existing plan/process or activity-based ontologies could address these requirements [30]. Tate's <I-N-OVA> constraint model of activity [37, 36] provided the most flexible approach as compared to the others. The <I-N-OVA> model can be seen as a specialisation of an <I-N-CA> shared model[2].

[2] <I-N-CA> stands for Issues, Nodes, Critical and Auxiliary constraints.

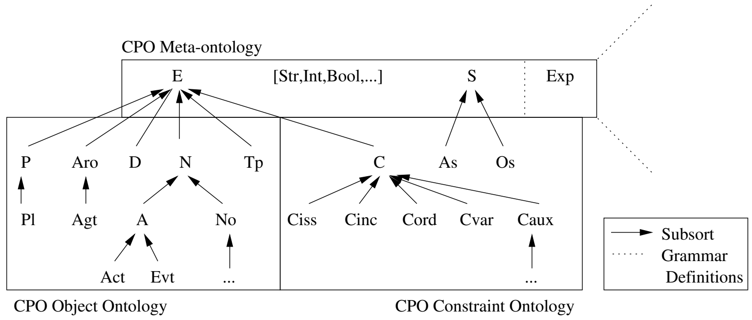

Figure 2: 3-CPO: Meta, Object, and Constraint Ontology

3.1 <I-N-OVA> Constraint Model of Activity

The <I-N-OVA> model (Issues, Nodes, Orderings/Variables/Auxiliary) is a means to represent and manipulate plans/processes as a set of constraints. The node constraints in this model set the space of behaviour within which a process may be further constrained. The issues (which could be considered to be implied, to do, or future constraints on behaviour) and remaining constraints (OVA) restrict the processes within that space which are valid. Ordering (O) and variable (V) constraints are distinguished from all other auxiliary (A) constraints since these act as cross-constraints, usually being involved in describing or further restricting the others. By having a clear description of the different components within a process, the model allows for processes to be manipulated and used separately from the environment in which they are generated. For example, we may wish to utilise an artificial intelligence planning system to synthesise a base process given some particular set of objectives and then take that information to a process editor for visualisation or further editing.

3.2 Plan Ontology

[38] developed a plan ontology based on the <I-N-OVA> model. Informally, this plan ontology envisions a plan as a specialised type of design. While a design for some artifact is considered to be a set of constraints on the relationships between the entities involved in the artifact, a plan or process further constraints these relationships to be between agents, their purposes and their behaviour. CPO is very strongly aligned with this ontology and while the above cited work presents the plan ontology primarily in terms of natural language and structured sentences, this paper will explore a sorted logic in which ontologically-based processes can be expressed and will outline a specific set of classes/sorts, functions, and relations[3].

4 CPO: Core Concepts

The CPO can be separated into three distinct parts, 3-CPO: meta-ontology, object ontology, and the constraint ontology. These elements are considered to be "core" or central to any process description. This "identifying core" approach can be found in the PIF, PSL, and SPAR projects as well. Extensions to any part of 3-CPO can be made in order to customise this set of core elements for specific domains, concepts or applications. These extensions may be packaged into manageable modules to promote shared communication between groups. This is similar to the "partial shared view mechanism (PSV)" developed in [21] which is also used in PIF. We provide examples of extensions to the core below.

[3] A similar approach to communicatating cconstraint information between planners and schedulers has also been investigated by David Joslin at CIRL.

In presenting CPO, we will describe a structured universe of discourse. We assume that this universe can be partitioned into certain sub-universes. Particular relationships exist between pairs of these sub-universes: they may be disjoint, they may have non-empty intersections or one may be completed contained in another. Further, we will state that certain mappings and relations are meaningful only for certain sub-universes. These sub-universes of discourse are given names, called sort symbols.

The set of all sort symbols is partially ordered by a sub-sort order relation, thus expressing the inclusion relationships which hold between sub-universes under consideration. We will refer to a set of sort symbols with the sub-sort order applied to it as a sort hierarchy. These sorts, combined with classical first-order logic (FOL) will give us a many-sorted, or simply sorted FOL [8, 5, 47] for expressing process knowledge[4]. The implementation of this language in CPF is called the Common Process Language (CPL). The lexicon and grammar for CPL is presented in [25] whereas the major terms and relationships for CPL are presented here.

The following sections overview various CPO function, predicate, and variables symbols which exist within these sub-universes. Sorts will be defined in scripted uppercase (e.g. $\mathcal{P}$ for processes), lower case letters indicate variables of a specific type (e.g. p is a variable of type $\mathcal{P}$) whereas uppercase letters are used for constants (e.g. P is a constant of type $\mathcal{P}$). The complete listing of sort types used in this paper can be found in the appendix. While this paper mainly overviews aspects of CPO, the detailed presentation of the ontology is available in ontolingua [14][5].

4.1 CPO Meta-ontology

As suggested in Tate's plan ontology approach [37], the very top of the CPO sort hierarchy will be reserved for meta-concepts which help to structure the universe of discourse. Broadly speaking, we can consider the ontology to be composed of a set of entities, a set of data types and a set of relationships between entities. We will define sorts $\mathcal{E}$ and $\mathcal{S}$ for entities and sets along with various elemental and composite data types such as $\mathcal{S}tr$, $\mathcal{I}nt$, and $\mathcal{E}xp$ for strings, integers and expressions, respectively. We will not reify the notion of relation, but we will be referring to various constraint types that have defined entity-relating expressions. In defining various expression types, we will specialise the base expression sort, $\mathcal{E}xp$. Figure 2 depicts the relationship between the meta-ontology and the constraint and object ontologies.

4.1.1 Entities

An entity, $\mathcal{E}$, provides the top-level root for much of the CPO sort hierarchy. In particular, $\mathcal{E}$ may be sub-classed into the sorts $\mathcal{C}, \mathcal{P}, \mathcal{N}, \mathcal{A}ro, \mathcal{T}p, \mathcal{D}$ for constraints, processes, nodes, activity-relatable objects, timepoints and domain levels respectively. This is a slightly different top-level than those found in the PIF, PSL, and SPAR ontologies.

Earlier, we referred to the sub-sort order relation which provides structure for the sort hierarchy. This relation can be expressed using a simple "isa" predicate. In this paper, we will use the following notation which expresses, in this case, that a CPO constraint is a CPO entity

$$\mathcal{C} \subset \mathcal{E} = isa(\mathcal{C}, \mathcal{E})$$

[4] Sorted logics are very similar to typed programming languages.

[5] The CPO ontolingua code for the core and the extensions described in this paper is available at the CPF homepage: http://www.dai.ed.ac.uk/students/stevep/cpf

4.1.2 Sets

One of the basic assumptions of the underlying <I-N-OVA> model is that a plan or process can be represented as a set of constraints on behaviour. At the very least, we require a sort for sets, $\mathcal{S}$, which provides some of the basic set theory relations and functions. For convenience, we will use the following notations

$$
\begin{aligned}
\emptyset &= emptyset \\
\{C\} &= adjoin(C, emptyset) \\
\{C_1, C_2\} &= adjoin(C_1, adjoin(C_2, emptyset)) \\
\{C_1, C_2 | S\} &= adjoin(C_1, adjoin(C_2, S)) \\
C \in S &= member(C, S) \\
S_1 \cup S_2 &= union(S_1, S_2) \\
S_1 \cap S_2 &= intersection(S_1, S_2) \\
S_1 \subseteq S_2 &= subset(S_1, S_2)
\end{aligned}
$$

4.1.3 Strings and Expressions

One of the advantages of the <I-N-OVA> perspective is the identification of various constraint types, i.e. specialisations of $\mathcal{C}$. Each of these constraints express various relationships between CPO objects. As in SPAR, CPO requires extensions which provide "plug-in grammars" that structure constraint expressions. These expressions will be specialisations of the base $\mathcal{E}xp$ type. We will use the following notation for strings and expressions

$$
\begin{aligned}
[] &= Nil \\
[Str] &= cons(Str, Nil) \\
[Str_1, Str_2] &= cons(Str_1, cons(Str_2, Nil)) \\
[Str_1, Str_2 | Exp] &= cons(Str_1, cons(Str_2, Exp))
\end{aligned}
$$

Expressions may be composed by concatenations of various strings. These strings may be variable, function, relation, constant or logical symbols. This is analogous to the "PIF-SENTENCE" described in the PIF work [19]. One important predicate that applies to $\mathcal{E}xp$ is a unification evaluation, $unifies(Exp_1, Exp_2)$, which implies that the expressions can be made identical by appropriate substitutions for their variables [12].

4.2 CPO Object Ontology

By "object" ontology, we are mainly indicating those entities which will be involved in and referred to by various constraint expressions which are connected to particular CPO constraint types. For example, the expression of an "ordering" constraint may relate two objects of type $\mathcal{T}p$. These constraint types are described in the constraints ontology section which follows below.

4.2.1 Processes

Generically speaking, a process, $\mathcal{P}$, provides a **specification** of **behaviour** for some time interval bounded by a pair of begin/end timepoints. By **behaviour**, we mean something that one or more agents perform. The notion of **specification** is simply that of a set, $\mathcal{S}$, of constraints, $\mathcal{C}$. In the constraint ontology, we define this to be an activity specification, $\mathcal{A}s$. We associate these via the relation $activity-spec \subseteq \mathcal{P} \times \mathcal{A}s$. Note that this does not commit to a single As for a given P, although some extensions of this ontology may do so. In addition to this, we observe that it might be the case that $as = \emptyset$ which is interpreted as "do anything". Clearly the opposite extreme may be to specify "do nothing". This may also be accomplished with an $activity-spec(P, as)$ for P which contains a not-include constraint which is discussed below.

CPO requires certain functions to be defined for all objects of type $\mathcal{P}$. The following two are suggested by the informal description above:

$$\begin{aligned} start-timepoint: &\quad \mathcal{P} \rightarrow \mathcal{T}p \\ finish-timepoint: &\quad \mathcal{P} \rightarrow \mathcal{T}p \end{aligned}$$

Additionally, there are functions defined which support expansion and decompositional relationships between processes and process actions.

$$\begin{aligned} pattern: &\quad \mathcal{P} \rightarrow \mathcal{E}xp \\ expands: &\quad \mathcal{P} \rightarrow \mathcal{A} \end{aligned}$$

The expression returned by the pattern function may be matched with the patterns of various activities and represents its potential to act in a decomposition relationship. An actual decompositional commitment to a particular activity is expressed using the *expands* predicate.

4.2.2 Plans

CPO distinguishes a plan, $\mathcal{P}l$, from a process by stating that $\mathcal{P}l \subset \mathcal{P}$ with the additional constraint that a $\mathcal{P}l$ exists for some specified objectives. That is, a plan extends the definition of process to say that it: provides a specification of behaviour **for some objectives** over some time interval bounded by a pair of begin/end timepoints. Objectives, $\mathcal{O}bj$, and objective specifications, $\mathcal{O}s$, are discussed in the constraint ontology section. This is related via a required $objective-spec \subseteq \mathcal{P}l \times \mathcal{O}s$ where $\mathcal{O}s \neq \emptyset$.

4.2.3 Nodes

In the overview, we referred to the fact that node constraints in the <I-N-OVA> model set the space within which a process may be further constrained. These constraints may either specify that a node is necessarily included or cannot be included at all. The CPO node type, $\mathcal{N}$, referred to is actually an abstract structuring of more specialised CPO concepts: activity, $\mathcal{A}$, or other nodes, $\mathcal{N}o$.

The $\mathcal{N}o$ type is, in turn, another sub-structuring of the domain of discourse. Currently, the only subtypes for $\mathcal{N}o$ are the following "dummy" node types $\mathcal{N}s, \mathcal{N}f, \mathcal{N}b, \mathcal{N}e$ which denote the elements of interval endpoint pairings

{Start/$\mathcal{N}s$,Finish/$\mathcal{N}f$} and {Begin/$\mathcal{N}b$,End/$\mathcal{N}e$}. A {$\mathcal{N}s$,$\mathcal{N}f$} pair is, by convention, used for indicating the entire interval for some, possibly decomposed, process or plan whereas {$\mathcal{N}b$,$\mathcal{N}e$} is used to demarcate subprocess intervals. All of these "dummy nodes" listed above represent an instantaneous point and therefore may only be related to a single timepoint. Future extensions may include additional specialisations of $\mathcal{N}o$ such as: or-split, or-join, and-split, and-join, conditional, iteration, for-each, etc.

4.2.4 Activities

For the most part, nodes in a process are used to denote activity. To be more precise, we indicate that $\mathcal{A} \subset \mathcal{N}$ where an $\mathcal{A}$ is meant to represent activity. As with processes, activities have a temporal extent which is bounded by a begin and end timepoint.

$$\begin{aligned} begin-timepoint: &\quad \mathcal{A} \rightarrow \mathcal{T}p \\ end-timepoint: &\quad \mathcal{A} \rightarrow \mathcal{T}p \end{aligned}$$

Additionally, there are functions defined which support expansion and decompositional relationships for activity.

$$\begin{aligned} pattern: &\quad \mathcal{A} \rightarrow \mathcal{E}xp \\ expansion: &\quad \mathcal{A} \rightarrow \mathcal{P} \end{aligned}$$

Notice that this provides a doubly-linked set of decompositional relationships. Given some process, P, we can directly determine which $\mathcal{A}$ it expands (i.e. its abstraction) or conversely, given some activity, A, we can refer to its (possibly null) expansion (i.e. its decomposition), $\mathcal{P}$. Given this information we know

$$\begin{aligned} (\forall a)(\exists p).expansion(a) = p &\quad \supset \quad expands(p) = a \wedge \\ &\quad\quad\;\; unifies(pattern(p), \\ &\quad\quad\;\; pattern(a)) \\ (\forall p)(\exists a).expands(p) = a &\quad \supset \quad expansion(a) = p \wedge \\ &\quad\quad\;\; unifies(pattern(a), \\ &\quad\quad\;\; pattern(p)) \end{aligned}$$

In accordance with Tate's plan ontology, we can further specialise $\mathcal{A}$ into $\mathcal{A}ct \subset \mathcal{A}$ and $\mathcal{E}vt \subset \mathcal{A}$. The ontological distinction being made here is between actions, $\mathcal{A}ct$, which are performed by modelled agents and events, $\mathcal{E}vt$, which are performed by an unmodelled agent (this is often referred to as the "environment").

4.2.5 Timepoints

A timepoint in CPO, $\mathcal{T}p$, characterises a specific, instantaneous point that lies along a line which is an infinite sequence of time points. Pairs of timepoints for nodes and processes delimit a time interval. In particular, we can use an axiomatisation based on Hayes' catalogue of temporal theories [17] in order to map timepoints and ordering constraints into Allen's 13 relations between intervals [1, 2]. During our application of these definitions, we spotted and corrected a couple of errors in this mapping axiomatisation[6].

This axiomatisation is used in the Common Process Assistant (CPA) to map the timepoints and ordering constraints which are passed from the process and domain editing tools, upon a users request, into an interval theory for consistency checking. Allen's table of legal relationships between intervals is then used to detect errors and to provide rationale for why a process specification is incorrect (i.e. CPA explains which legal interval relationships could exist).

In addition to the CPA analysis, the process and domain editors can automatically and efficiently assist users by preventing illegal or unnecessary timepoint constraints between two activities based on the knowledge provided in an $\mathcal{A}s$. As we have pointed out, each $\mathcal{A}$, has 2 timepoints which we will abbreviate as: $\mathcal{T}p^{\mathcal{A}}_{begin}$ and $\mathcal{T}p^{\mathcal{A}}_{end}$. There is one relation that always exists between an activity, A_1, timepoint pair: $before(Tp^{A_1}_{begin}, Tp^{A_1}_{end})$. No other relation can be made between these two points.

4.2.6 Analysis: Timepoints and Intervals

In our work on CPO, we explored the relationship between timepoints and time intervals in a bit more detail. In particular, we were interested in the implications of our approach whereby temporal constraints on timepoints may be incrementally added to a specification and how that affects a mapping to time interval relationships.

We indicated that two different activities in an $\mathcal{A}s$ have a set of unique pairs of timepoints, which we refered to as a $\mathcal{T}set^{\mathcal{A},\mathcal{A}}$, defined as

$$\begin{aligned} Tset^{A_1,A_2} &= \{(tp_1, tp_2) | tp_1 \in \{Tp^{A_1}_{begin}, Tp^{A_1}_{end}\} \wedge \\ &\quad tp_2 \in \{Tp^{A_2}_{begin}, Tp^{A_2}_{end}\}\} \end{aligned}$$

Each pair in a $\mathcal{T}set$ may be related in one of two possible ways or not at all. If a relationship is assigned to a pair then either one timepoint is temporally before the other or they are equal. This can be expressed in an infix notation as

$$\begin{aligned} tp_1 \prec tp_2 &\equiv before(tp_1, tp_2) \\ tp_1 = tp_2 &\equiv equal(tp_1, tp_2) \end{aligned}$$

We looked at the various combinations of constrants between all $(tp_1, tp_2) \in Tset^{A_1,A_2}$ and found that in fact, only 57 of 256 unique combinations are legal ($\approx 22\%$) and only 35 of those 57 completely specified an Allen interval relationship. This analysis provided the knowledge we used to construct efficient process editors which prevent users from specifying illegal configurations between two activities. In addition to this, it is obvious that some of the configurations contain superfluous constraints which can be eliminated without affecting their interval relationships.

4.2.7 Activity-Relatable Objects

In PIF, one of the top-level classes of entity is OBJECT. It is informally defined as "an entity that can be used, created, modified, or used in other relationships to an activity". An identical type is utilised in PSL. During our work on SPAR it was decided that the term "object" was a bit too overloaded and would perhaps confuse the understanding of this class.

[6] The axioms are listed in the appendix and this is discussed in more detail in [26].

The more specific Activity-Relatable Object (ARO) term was chosen for SPAR instead. CPO represents this in the sort, $\mathcal{A}ro$.

Some entities of type $\mathcal{A}ro$ are often referred to as "resources". As the PIF definition above suggests, these are the things which are used (e.g. drill, hammer, etc.), modified (e.g. board, metal sheet), etc. This sort also represents those things produced, which might be labelled "products". It is important to note though that these common references or labels tend to be role-defined, which we discussed in [29]. For example, an $\mathcal{A}ro$ for one activity, A_1, may be a "product" for A_1, but it might be a "resource" for A_2.

Subtypes of $\mathcal{A}ro$ are defined for domain-specific applications of CPO (e.g. manufacturing objects might include various drill, saw, lathe types, etc.) As mentioned earlier, these may be packaged into PSV-like extensions to support reuse or to encourage modularity. Domain independent extensions may also be created to provide rich structure between $\mathcal{A}ro$ types (e.g. part-of, requires relations, etc.)

A special agent sub-sort of $\mathcal{A}ro$ has been included in CPO: $\mathcal{A}gt \subset \mathcal{A}ro$. An informal reference for the $\mathcal{A}gt$ sub-sort can be found in the SPAR sentences [39] which refer to it as an "ACTIVITY-RELATABLE-OBJECT which can PERFORM ACTIVITIES and/or HOLD OBJECTIVES". The inclusion of this concept in CPO points to the influence of workflow languages like the WPDL. Specifically, in a process specification, we are interested in knowing who or what will be performing activities and in also linking the purpose of these sets of activity with the agents who held the objectives. As we shall see in the CPO constraint ontology, some aspects of an objective specification are characterised by agent's requirements while others can be considered to be preferences. Thus we have agent relationships such as

$$
\begin{aligned}
performs - activity: &\subseteq \mathcal{A}gt \times \mathcal{A} \\
performs - process: &\subseteq \mathcal{A}gt \times \mathcal{P} \\
has - requirement: &\subseteq \mathcal{A}gt \times \mathcal{O}s \times \mathcal{P} \\
has - preference: &\subseteq \mathcal{A}gt \times \mathcal{O}s \times \mathcal{P} \\
has - requirement: &\subseteq \mathcal{A}gt \times \mathcal{O}s \\
has - preference: &\subseteq \mathcal{A}gt \times \mathcal{O}s \\
has - capability: &\subseteq \mathcal{A}gt \times \mathcal{E}xp
\end{aligned}
$$

Note that sets of requirements or preferences may be universal for an agent (e.g. "prefer transportation by boat for any set of activity") or process-specific (e.g. "prefer transportation by airplane for process P_1"). As in subtypes of $\mathcal{A}ro$, the subtypes of $\mathcal{A}gt$ are specialised for a domain. Domain independent extensions may also be added to provide concepts such as organisational structure (e.g. reports-to, coaches, etc.)

4.2.8 Domain Levels

In the Common Process Methodology (CPM) [28], a level-oriented approach to domain modelling is adopted whereby actions, events, effects, and resources are all separated into a series of defined and increasingly detailed levels, $\mathcal{D}$. This helps to avoid the commonly experienced problem of "hierarchical promiscuity" [49] or "level promiscuity" which is characterised by the inconsistent usage of various domain elements at varying areas in the overall domain description. This approach is taken directly from our characterisation of the TF Method [41].

Domain levels should be assigned meaningful labels which indicate their overall perspective (e.g. "house building task level"). These levels may be structured into a domain lattice and processes assigned to a particular domain level. The following functions and relations partially support these requirements

$$\begin{aligned} label: & \quad \mathcal{D} \rightarrow \mathcal{S}tr \\ number: & \quad \mathcal{D} \rightarrow \mathcal{I}nt \\ contains: & \subseteq \mathcal{D} \times \mathcal{P} \end{aligned}$$

4.3 CPO Constraint Ontology

In this section, we describe various categories of constraints which may be placed between CPO objects. These constraint types are based on the <I-N-OVA> model and Tate's plan ontology which were introduced above. Primarily we are interested in two types of things: a single constraint, $\mathcal{C}$, and an aggregation of constraints, or a set, $\mathcal{S}$. Also, the expression of a constraint, $\mathcal{E}xp$, for each of the various types is of interest to us, but it will be defined using a highly flexible approach. In order to make this framework generically applicable, we envision a "plug-in" syntax for expressions as described in the SPAR approach. We provide examples of this below.

Tate describes a constraint as "a relationship which expresses an assertion that can be evaluated with respect to a given plan as something that may hold and can be elaborated in some language" [36]. In addition to this, it is pointed out that there is typically a need to recognise which agent added a specific constraint during a design process. At a high-level, we can relate these entities using

$$\begin{aligned} expression: & \quad \mathcal{C} \rightarrow \mathcal{E}xp \\ added-by: & \quad \mathcal{C} \rightarrow \mathcal{A}gt \end{aligned}$$

The design of a process, $\mathcal{P}$, has a relationship with a set of these constraints which denote the process activity. We will refer to this set as an activity specification, $\mathcal{A}s \subset \mathcal{S}$. In addition to this, we further distinguish that a plan, $\mathcal{P}l$ relates an $\mathcal{A}s$ to some set of objectives, $\mathcal{O}s \subset \mathcal{S}$. An objective, $\mathcal{O}bj \subset \mathcal{C}$, may be a requirement (hard constraint) or a preference (soft constraint).

$$\begin{aligned} member: & \subseteq \mathcal{C} \times \mathcal{A}s \\ member: & \subseteq \mathcal{O}bj \times \mathcal{O}s \\ soft-hard-info: & \quad \mathcal{C} \rightarrow soft, hard \end{aligned}$$

The expression of an objective, as with the other constraints, is defined by providing a structuring plug-in grammar. This approach is partially based on the way flexible tasks and goals are expressed in EXPECT [34, 33] and INSPECT [46].

4.3.1 Issues

The focus on issues in <I-N-OVA> is a unique approach which is linked to ideas found in workflow perspectives and issue-based collaborative design. Essentially an issue is, "an outstanding aim, objective, preference, task, or flaw which remains to be addressed by the process". Issues refer to "implied constraints" on the actual organisational processes. For example, an issue may refer to an abstract activity which has not been expanded yet or to some condition on an activity which still remains to be achieved.

In CPO, an issue, $\mathcal{C}_{iss} \subset \mathcal{C}$, requires some plug-in syntax which defines the legal grammar for its expression, $\mathcal{E}xp_{iss} \subset \mathcal{E}xp$. For example, we may specify the following structure for an issue (using BNF):

```
<issue-expression> ::= <rtq-sent> | <rt-sent> |
                       <r-sent>
        <rtq-sent> ::= <issue-relconst> <term>
                       [<term>*] <logsent>
         <rt-sent> ::= <issue-relconst> <term>
                       [<term>*]
          <r-sent> ::= <issue-relconst>
  <issue-relconst> ::= "achieve" | "expand"...
         <logsent> ::= {not <sentence>} | etc.
```

Thus, an example of a specific issue constraint, C_{iss1}, which simply states that an activity, A_1 remains to be expanded would indicate $Exp_{iss1} \equiv$ ["expand", "A1"]. This corresponds to the rt-sent defined in the extension.

4.3.2 Node Constraints

Node constraints are the backbone of the activity specification constraint set. They provide the space of behaviour on which many of the other constraints seek to further define. Node constraints are so important that a special case has been made for them. Their expression does not require a plug-in syntax, instead there are two built-in functions for declaring that a particular node is either to be included or specifically not to be included

$$\begin{aligned} include-node: & \quad \mathcal{C} \to \mathcal{N} \\ not-include-node: & \quad \mathcal{C} \to \mathcal{N} \end{aligned}$$

In fact, there are special cases of both constraints which can be used to refer to an entire class of entities or type. For example, we may wish to specify "do nothing" or "don't do any transportation action". These two concepts use the form $not-include-node: \mathcal{C} \to \mathcal{S}tr$ where $\mathcal{S}tr$ references a type name (e.g. "cpo-action" or "transport-action").

4.3.3 Ordering Constraints

A central aspect of most process specifications is the subset of temporal relationships which define the order in which actions or events will occur. In CPO, this aspect involves those ordering constraints $\mathcal{C}_{ord} \subset \mathcal{C}$. These temporal constraints could be expressed directly between entities of type $\mathcal{N}$ which would be similar to interval relationship approaches (e.g. after, meets, finishes, etc.) but as we showed earlier, CPO uses a more expressive default ordering approach between timepoints, $\mathcal{T}p$. In particular, part of a default BNF for a $\mathcal{C}_{ord}$ is

```
<ordering-expression> ::= <ordering-relconst>
                          ( <term>, <term> )
  <ordering-relconst> ::= "before_tp" | "equal_tp"
```

4.3.4 Variable Constraints

Co-designation and non-co-designation constraints between variables relate activity relatable objects in the domain and are quite common in plan and process specifications. These variable constraints, $\mathcal{C}_{var} \subset \mathcal{C}$, limit the range of values which may be assigned to particular variables in CPO expressions. For example, some activity labelled "replace drill bit" may be defined with a pattern "replace-drill-bit ?old ?new". The specification of this activity may include a variable constraint, C_{var1}, which has an expression, Exp_{var1} that specifies that the old bit cannot be the new bit (e.g. $Exp_{var1} \equiv$ ["not_equal_var","(", "?old", ",", "?new", ")"]). Thus part of a default BNF for a $\mathcal{C}_{var}$ may be

```
<variable-expression> ::= <variable-relconst>
                          ( <indvar>,  <indvar> )
  <variable-relconst> ::= "equal_var" |
                          "not_equal_var"
```

4.3.5 Auxiliary Constraints

Up to this point, we can see that an activity specification, As_1, can be viewed as the union of a set of defined constraint subsets. Specifically we know that

$$\begin{aligned}
(As_1 \equiv S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5) \Leftrightarrow \\
((S_1 = \{ciss_1 | ciss_1 \in As1\}) \wedge \\
(S_2 = \{cinc_1 | cinc_1 \in As1) \wedge \\
(S_3 = \{cord_1 | cord_1 \in As1\}) \wedge \\
(S_4 = \{cvar_1 | cvar_1 \in As1\}) \wedge \\
(S_5 = \{caux_1 | caux_1 \in As1\}))
\end{aligned}$$

The final set that hasn't been addressed yet are the auxiliary constraints, $\mathcal{C}_{aux} \subset \mathcal{C}$, denoted by S_5. This constraint type has a defined sub-sort order structure which is detailed in the CPO ontolingua version. In this section, we will briefly consider some of the common subtypes: $\mathcal{C}_{inp}, \mathcal{C}_{out}, \mathcal{C}_{alw}, \mathcal{C}_{res}$ and $\mathcal{C}_{ann}$.

The first two constraint types, $\mathcal{C}_{inp}$ and $\mathcal{C}_{out}$, relate world state expressions, $\mathcal{E}xp_{ws}$, to particular timepoints. This can be used to express state-based conditions and effects for process activities. The partial grammar outlined below has been used in the CPF for expressions based on the Task Formalism's approach of $< pattern >=< value >$.

```
<world-state-expression> ::= [<ws-type>] <LBRACE>
                             <term>* <RBRACE>
                             [= <term>*]
                             at <term>
               <ws-type> ::= "supervised" |
                             "achieve" |
                             "unsupervised" |
                             "only_use_if" |
                             "only_use_for_query"
                <LBRACE> ::= "{"
                <RBRACE> ::= "}"
```

So, a particular C_{inp_1} which has an $Exp_{ws1} \equiv$ ["supervised", "{","have ?material","}","at","N12"] may depend on a C_{out1} which has an $Exp_{ws2} \equiv$ ["{","have bricks","}","at","N10"].

A $\mathcal{C}_{alw}$ constraint differs from those above in that it its assertion is not tied to a particular timepoint, it is defined as always holding in all states. We can modify the $\mathcal{E}xp_{ws}$ grammar above to define a new expression $\mathcal{E}xp_{wsa}$ in which the "$at < term >$" tokens are not required.

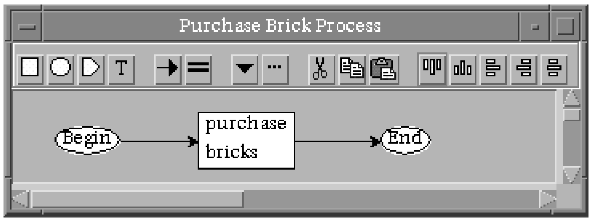

Figure 3: Simple Process Example

The resource constraints, $\mathcal{C}_{res}$, can be used to describe an activities required allocation of resource objects, producible/consumable resource effects, etc. While it is possible to lump resource constraints into the general notion of input and output constraints it is beneficial to separate them out as many tools are largely geared toward working with this knowledge (e.g. scheduling tools, etc.). The resource expression $Exp_{res1} \equiv$ ["consumes","{","resource","money","}","=","50 pounds ","at","N10"] may be derived from a grammar which roughly corresponds to

```
<resource-expression> ::= <res-type> <LBRACE>
                          resource <term>*
                          <RBRACE>
                          [= <term>*]
                          at <term>
           <res-type> ::= "consumes" |
                          "produces" etc.
             <LBRACE> ::= "{"
             <RBRACE> ::= "}"
```

Finally, the simplest of these is the annotation constraint, $\mathcal{C}_{ann}$, which can be used to attach unstructured strings to activity specifications. This might be used for attaching additional notes, comments, instructions or possibly to provide links to non-textual or external data related to the process such as CAD and multimedia filenames, web site addresses, or printed policy/standards document references.

5 CPO: Example and Extensions

In order to provide a detailed example of a CPL process specification which utilises CPO terms and concepts, we will restrict the content to a rather simplified process. The example "Purchase Brick Process" might be part of a much larger building domain and represents a particular transaction activity whereby money is consumed to acquire some supply of brick building material. As we can see in Figure 3, it is bounded by a begin/end node pairing and contains only one action, "purchase bricks". As mentioned earlier, the lexicon and grammar for CPL is described in [25]. The following specification characterises this process

```
%define-domain{my-building}
SORT cpo-action{A1}
SORT cpo-activity-specification{AS1}
SORT cpo-begin{B1}
SORT cpo-end{E1}
SORT cpo-include-constraint{IC1-IC3}
SORT cpo-ordering-constraint{OR1,OR2}
SORT cpo-output-constraint{OC1}
```

```
SORT cpo-process{P1}
SORT cpo-resource-constraint{RC1}
SORT cpo-timepoint{TP1-TP4}

label(P1)="Purchase Brick Process"
start-timepoint(P1)=TP1
finish-timepoint(P1)=TP4
pattern(P1)="{purchase bricks}"
label(B1)="begin"
timepoint(B1)=TP1
include-node(IC1)=B1
member(IC1,AS1)
label(E1)="end"
timepoint(E1)=TP4
include-node(IC2)=E1
member(IC2,AS1)
label(A1)="purchase bricks"
begin-timepoint(A1)=TP2
end-timepoint(A1)=TP3
include-node(IC3)=A1
member(IC3,AS1)
expression(OR1)="before(TP1,TP2)"
member(OR1,AS1)
expression(OR2)="before(TP3,TP4)"
member(OR2,AS1)
expression(OC1)="{have bricks} at A1"
member(OC1,AS1)
expression(RC1)="consumes
  {resource money} = 50 pounds at A1"
member(RC1,AS1)
```

5.1 Tools-Specific Extensions

CPO provides a core set of concepts which may be extended to capture specialised process-related knowledge. One class of extensions can be considered to be tool-specific. Tool-specific extensions are used to express new or specialised sorts or relations which address aspects linked to a particular tool's ontology. Two examples are provided here for extensions related to O-Plan's TF and the process/domain editors in CPF.

O-Plan's Task Formalism language [40, 35] encompasses a copious set of terms and concepts for expressing plan/process domain knowledge. For this particular TF extension example though, we are simply interested in providing additional support for capturing TF resource-related information. One facet of this information is "resource units". Resource unit statements in TF are used to define unit types for resources such as person/people, gallons, kilograms, etc. These units have their own properties in TF (e.g. type, which could have the values: count; size; weight; or set).

In the TF extension, we define a new class in ontolingua, called resource unit, which will correspond to a new sort, $\mathcal{U} \subset \mathcal{E}$. Two new functions are designated for $\mathcal{U}$ to express both its label (e.g. pounds) and its type (e.g. count).

$$
\begin{aligned}
label: &\quad \mathcal{U} \rightarrow \mathcal{S}tr \\
type: &\quad \mathcal{U} \rightarrow \mathcal{E}xp
\end{aligned}
$$

The type expression above can be syntactically constrained to be ["count"] $\vee$ ["size"] $\vee$ ["weight"] $\vee$ ["set"]. In addition to this, we need to add functions and a relation to the activity-relatable object sort, $\mathcal{A}ro$. In particular, we need to be able to express whether a $\mathcal{A}ro$ is going to play the role of a TF resource and if so, what its TF resource type is (i.e. ["consumable_strictly"] $\vee$ ["consumable_producible_by_agent"] $\vee$ ...). Finally, we also need to be able to relate a $\mathcal{A}ro$ to our new resource unit.

$$
\begin{aligned}
is-resource: &\subseteq \mathcal{A}ro\\
resource-type: &\quad \mathcal{A}ro \rightarrow \mathcal{E}xp\\
unit: &\subseteq \mathcal{A}ro \times \mathcal{U}
\end{aligned}
$$

Some tool-specific extensions are related to presentation information or internal state information (e.g. nodes selected, etc.) associated with processes. In both the Common Domain Editor (CDE) and the Common Process Editor (CPE) in CPF, process presentation information is attached to various parts of the domain specification. The CPF tools extension defines additional support for this such as

$$
\begin{aligned}
xpos: &\quad \mathcal{P} \rightarrow \mathcal{I}nt\\
ypos: &\quad \mathcal{P} \rightarrow \mathcal{I}nt\\
width: &\quad \mathcal{P} \rightarrow \mathcal{I}nt\\
height: &\quad \mathcal{P} \rightarrow \mathcal{I}nt\\
label: &\quad \mathcal{P} \rightarrow \mathcal{S}tr\\
xpos: &\quad \mathcal{N} \rightarrow \mathcal{I}nt\\
ypos: &\quad \mathcal{N} \rightarrow \mathcal{I}nt\\
type: &\quad \mathcal{N} \rightarrow \mathcal{E}xp\\
status: &\quad \mathcal{N} \rightarrow \mathcal{I}nt\\
label: &\quad \mathcal{N} \rightarrow \mathcal{S}tr\\
xpos: &\quad \mathcal{C}_{ann} \rightarrow \mathcal{I}nt\\
ypos: &\quad \mathcal{C}_{ann} \rightarrow \mathcal{I}nt\\
top-level: &\subseteq \mathcal{P}\\
selected: &\subseteq \mathcal{N}
\end{aligned}
$$

5.2 Rationale Extension

While the extensions discussed in the previous section were labelled tool-specific, we can also have extensions which are tool-independent, or more appropriately, concept-specific. Concept-specific extensions provide terms and definitions which are centred around a general set of closely associated entities and relations. One example of such an extension is the rationale extension we have developed for CPO.

In our review of plan rationale [31], we explored the epistemological nature of this category of knowledge and described it from the perspectives of dependencies, causal relationships and decisions. While there has been much work done on both plan/process causality and dependencies, there has been correspondingly less research into plan decision rationale.

We proposed a "design space analysis (DSA)" approach to plan decision rationale [24] which was based on research from the design rationale (DR) field [22, 20]. If we envision the <I-N-OVA> approach, which CPO has adopted, as describing a "space of behaviour" we can also consider a "space of decisions" which is navigated in creating this behavioural specification. It is possible then to augment a process description with the rationale that went into designing this artifact.

Both CPE and CDE support this DSA approach (i.e. provide graphical editing of a DSA) and rely on the CPF rationale extension to define the DSA terms and concepts which are expressed in CPL. In this extension, we refer to an entity called a decision rationale, $\mathcal{D}r \subset \mathcal{E}$, which represents the overall "decision space" for a process design. In the CPO core, an $\mathcal{A}s$ groups the constraints which form the "space of behaviour". Analogously, a rationale specification, $\mathcal{R}s$, groups the constraints which form the "space of decisions". So, the CPF rationale extension includes

$$\begin{aligned} decision-rationale: & \quad \mathcal{P} \rightarrow \mathcal{D}r \\ rationale-spec: & \subseteq \mathcal{D}r \times \mathcal{R}s \end{aligned}$$

While a plan is described in Tate's plan ontology as "a set of constraints on the relationships between agents, their purposes and their behaviour" a decision rationale can be viewed as "a set of constraints on the relationships between questions (or design issues), options (or answers to these questions), and evaluative criteria. The CPF rationale extension includes the sorts $\mathcal{Q}, \mathcal{O}pt, \mathcal{C}rt$ for questions, options and criteria respectively.

Questions pose key issues for structuring the space of alternatives (options). The role of questions is to define local contexts in a design space which help to ensure that certain options are compared with each other. Criteria represent the desirable properties of the process and requirements that it must satisfy. They form the basis against which to evaluate the options. These elements can be included into a $\mathcal{R}s$ and interrelated via a set of defined constraints which represent relationships such as

$$\begin{aligned} has-option: & \subseteq \mathcal{Q} \times \mathcal{O}pt \\ selected: & \subseteq \mathcal{O}pt \\ supports: & \subseteq \mathcal{C}rt \times \mathcal{O}pt \\ detracts: & \subseteq \mathcal{C}rt \times \mathcal{O}pt \\ sub-question: & \subseteq \mathcal{O}pt \times \mathcal{Q} \end{aligned}$$

6 CPO Application Space Attributes

In [42], a multi-dimensional framework is proposed in order to aid cross-evaluation of ontologies and identification for reuse. The attributes of this framework form a "space of ontology applications" in which ontological authors can roughly indicate where their ontology lies within that space. Ontology consumers can use the space to determine if there is an overlap between their particular needs and a prospective ontology. The following listing provides CPO values for these attributes.

Purpose:

The broad goal of the CPO is to support the management of organisational process-specific knowledge. The role of the ontology is to facilitate communication amongst humans and systems involved in this process. The ontology provides definitions for the concepts and terms which are exchanged.

Representation Languages and Paradigms:

Core aspects of the ontology were based on an informal representation expressed in natural language and diagrams.

The CPO exists as a formal encoding in Ontolingua. An operational language used to exchange CPO is based on a sorted FOL. Some parts (e.g. temporal relationships) have also been implemented in Prolog.

Meaning and Formality:

Mostly primitive terms with some axioms, semantics of terms mainly through existence of relationships between types of entities or via connection to a more detailed foundational theory.

Foundational theories: situation calculus, extended situation calculus, complex theory of actions, parametric constraints, KIF-meta, frame ontology.

Subject Matter:

Meta ontology: Simple high-level modelling concepts (entity, sets, strings, etc.)

Constraint ontology: General terms used to describe and model processes as sets of design constraints.

Core Ontology: General objects and concepts referred to in the constraints (e.g. processes, timepoints, activities, agents, etc.)

Scale:

Roughly equivalent to Uschold's enterprise ontology [45, 11], O(150) definitions.

Development:

Research prototype.

Conceptual Architecture:

A framework (methods, tools, representation) for the management and exchange of process knowledge intended to support a range of purposes: knowledge acquisition, user communication, formal analysis, system manipulation.

Mechanisms and Techniques:

CPO supports ontological extensions, Plug-in expressions, Translation to and from source/target languages, Process/Domain dependencies, Methodology and toolset (for initial requirements engineering), Design rationale.

Implementation platform:

The toolset built to manipulate CPO representations was designed to be as platform independent as possible and to support exchange of knowledge via the internet. These tools are implemented using Java, AIAI's HARDY meta-case tool (which runs on Unix and Windows 95/NT), Clips and Prolog.

7 Summary

In this paper, we have presented the Common Process Ontology which provides terms and concepts for expressing process knowledge from a design constraint perspective. Specialisations of the ontology were discussed which customise the ontology and extend it to meet particular expressive requirements. We believe this conceptualisation provides a strong foundation which can facilitate exchange of knowledge between people and information systems involved in organisational process management. As our research into a framework based on this ontology has shown, it is possible to envision a framework which encompasses a life-cycle of process management activities which all translate knowledge into and out of a shared understanding of the process-centred organisation.

Acknowledgements

The O-Plan project is sponsored by the Defense Advanced Research Projects Agency (DARPA) and Rome Laboratory, Air Force Materiel Command, USAF, under grant number F30602-95-1-0022. The O-Plan project is monitored by Dr. Northrup Fowler III at the USAF Rome Laboratory. One author is sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-96-1-0348 – an AASERT award monitored by Dr. Abe Waksman and associated with the O-Plan project. The U.S. Government and the University of Edinburgh is authorised to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of DARPA, Rome Laboratory, AFOSR or the U.S. Government.

References

[1] J. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1984.

[2] J.F. Allen. Towards a general theory of action and time. In J.F. Allen, J. Hendler, and A. Tate, editors, *Readings in Planning*, pages 464–479. Morgan Kaufmann, Palo Alto, CA, 1984.

[3] J. Arpírez, A. Gómez, A Lozano, and H.S. Pinto. (ONTO)2Agent: An ontology-based WWW broker to select ontologies. In *ECAI '98 Workshop on Applications of Ontologies and Problem-Solving Methods*, pages 107–111, 1998.

[4] J.R. Caron, S.L. Jarvenpa, and D.B. Stoddard. Business reengineering at CIGNA corporation: Experiences and lessons learned from the first five years. *Management Information Systems Quarterly*, 18(3), 1994.

[5] A.G. Cohn. On the solution of schubert's steamroller in many sorted logic. In *IJCAI*, pages 1169–1174, 1985.

[6] K. Currie and A. Tate. O-Plan: The open planning architecture. *Artificial Intelligence*, 52:49–86, 1991.

[7] T.H. Davenport, S.L. Jarvenpa, and M.C. Beers. Improving knowledge work process. *Sloan Management Review*, pages 53–65, 1996.

[8] E. Davis. *Representation of Commonsense Knowledge.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[9] S. Davis and J. Botkin. The coming of knowledge-based business. *Harvard Business Review*, September-October:165–170, 1994.

[10] M. Fernández, A. Gómez-Pérez, and N. Juristo. Methontology: From ontological art towards ontological engineering. In *Workshop on Ontological Engineering, Spring Symposium Series, AAAI97*, Stanford, USA, 1997.

[11] J. Fraser and A. Tate. The enterprise tool set – an open enterprise architecture. In *Proceedings of the Workshop on Intelligent Manufacturing Systems, International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, 1995.

[12] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, Palo Alto, CA, 1987.

[13] A. Gómez-Pérez, M. Fernández, and A. De Vicente. Towards a method to conceptualize domain ontologies. In *Workshop on Ontological Engineering, ECAI'96*, pages 41–51, 1996.

[14] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[15] M. Hammer. *Beyond Reengineering.* Harper Collins Business, London, UK, 1996.

[16] M. Hammer and J. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution.* Nicholas Brealey Publishing, London, 1994.

[17] P. Hayes. A catalog of temporal theories. Technical Report Technical Report UIUC-BI-AI-96-01, The Beckman Institute, University of Illinois, 1996.

[18] Y. Kalfoglou and D. Robertson. Error checking in process interchange format (PIF). Division of Informatics RP 887, University of Edinburgh, Edinburgh, Scotland, 1997.

[19] J. Lee, M. Gruninger, Y. Jin, T. Malone, A. Tate, and G. Yost. The process interchange format and framework. *Knowledge Engineering Review*, 13(1), 1998.

[20] A. MacLean, R. Young, V. Bellotti, and T. Moran. Design space analysis: Bridging from theory to practice via design rationale. In *Proceedings of Esprit '91*, pages 720–730, Brussels, November 1991.

[21] T.W. Malone and J. Lee. Partially shared views: A scheme for communicating among groups that use different type hierarchies. *ACM Transactions on Information Systems*, 8(1), January 1990.

[22] T.P. Moran and J.M. Carroll, editors. *Design Rationale: Concepts, Techniques, and Use.* Lawrence Erlbaum Associates, 1996.

[23] S. Polyak. Requirements for a rich, shared plan representation. Department of Artificial Intelligence DP 187, University of Edinburgh, Edinburgh, Scotland, 1997.

[24] S. Polyak. Applying design space analysis to planning. In *Proceedings of the AIPS-98 workshop on Knowledge Engineering and Acquisition for Planning: Bridging Theory and Practice AAAI Technical Report WS-98-03*, Carnegie-Mellon University, June 1998.

[25] S. Polyak. The common process language. Department of Artificial Intelligence RP 933, University of Edinburgh, Edinburgh, Scotland, 1998.

[26] S. Polyak. Mapping timepoint-based constraints into interval relationships. Department of Artificial Intelligence RP 932, University of Edinburgh, Edinburgh, Scotland, 1998.

[27] S. Polyak. A supply chain process interoperability demonstration using the process interchange format (PIF). Department of Artificial Intelligence RP 917, University of Edinburgh, Edinburgh, Scotland, 1998.

[28] S. Polyak. A common process methodology for engineering process domains. In D. Bustard, editor, *Submitted to: Systems Modelling for Business Process Improvement (SMBPI)*. Artech House, 1999.

[29] S. Polyak, J. Lee, M. Gruninger, and C. Menzel. Applying the process interchange format (PIF) to a supply chain process interoperability scenario. In *ECAI '98 Workshop on Applications of Ontologies and Problem-Solving Methods*, pages 88–97, 1998.

[30] S. Polyak and A. Tate. Analysis of candidate PSL process/plan representations. Artificial Intelligence Applications Institute AIAI-PR-66, University of Edinburgh, Edinburgh, Scotland, 1997.

[31] S. Polyak and A. Tate. Rationale in planning: Causality, dependencies, and decisions. *Knowledge Engineering Review*, 13(3):247–262, September 1998.

[32] C. Schlenoff, A. Knutilla, and S. Ray. Unified process specification language: Requirements for modeling process. Technical Report NISTIR 5910, National Institute of Standards and Technology, Gaithersburg, MD, 1996.

[33] B. Swartout and Y. Gil. EXPECT: Explicit representations for flexible acquisition. In *Proceedings of the Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop*, February 1995.

[34] W.R. Swartout and Y. Gil. EXPECT: A user-centered environment for the development and adaptation of knowledge-based planning aids. In A. Tate, editor, *Advanced Planning Technology: Technological Advancements of the ARPA/Rome Laboratory Planning Initiative*, pages 250–258, Menlo Park, CA, 1996. AAAI Press.

[35] A. Tate. Generating project networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-77)*, pages 888–893, Cambridge, MA, 1977.

[36] A. Tate. Characterising plans as a set of constraints – the $<$ I-N-OVA $>$ model – a framework for comparitive analysis. *ACM Sigart Bulletin*, 6(1), January 1995.

[37] A. Tate. Representing plans as a set of constraints – the $<$ I-N-OVA $>$ model. In B. Drabble, editor, *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 221–228, Edinburgh, Scotland, May 1996. Morgan Kaufmann.

[38] A. Tate. Towards a plan ontology. *AI*IA Notizige (Publication of the Associazione Italiana per l'Intelligenza Artificiale), Special Issue on Aspects of Planning Research*, 9(1):19–26, 1996.

[39] A. Tate. Roots of SPAR - Shared Planning and Activity Representation. *The Knowledge Engineering Review*, 13(1):121–128, 1998.

[40] A. Tate, B. Drabble, and J. Dalton. The Task Formalism Manual. Artificial Intelligence Applications Institute AIAI-TF-Manual, University of Edinburgh, Edinburgh, UK ftp://ftp.aiai.ed.ac.uk/pub/documents/ANY/oplan-tf-manual.ps.gz, 1994.

[41] A. Tate, S. Polyak, and P. Jarvis. TF Method: An initial framework for modelling and analysing planning domains,. AIPS '98 Workshop on Knowledge Engineering and Acquisition for Planning: Bridging Theory and Practice AAAI Technical Report WS-98-03, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1998.

[42] M. Uschold. Where are the killer apps? In *ECAI '98 Workshop on Applications of Ontologies and Problem-Solving Methods*, pages 107–111, 1998.

[43] M. Uschold and M. Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2), 1996.

[44] M. Uschold, M. Healy, K. Willamson, P. Clark, and S. Woods. Ontology reuse and application. In N. Guarino, editor, *Proceedings of the First International Conference on Formal Ontology in Information Systems*, pages 179–192. IOS Press, 1998.

[45] M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13, Special Issue on Putting Ontologies to Use(1), 1998.

[46] A. Valente, W.R. Swartout, and Y. Gil. A representation and library for objectives in air campaign plans. USC, Information Sciences Institute Technical Report, University of Southern California, 1996.

[47] C. Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation.* Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1985.

[48] WfMC. Workflow Management Coalition Glossary, Workflow Management Coalition specification. Technical Report November '94, Workflow Management Coalition, http://www.wfmc.org, 1994.

[49] D. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann, 1988.

A. CPO Sort Table

The following table presents the abbreviations used in this paper for referring to various core and extended CPO sort types.

$\mathcal{A}$		Activity
$\mathcal{A}ro$		Activity-relatable object
$\mathcal{A}s$		Activity specification
$\mathcal{A}ct$		Action
$\mathcal{C}$		Constraint
	$\mathcal{C}_{alw}$	Always constraint
	$\mathcal{C}_{ann}$	Annotation constraint
	$\mathcal{C}_{aux}$	Auxiliary constraint
	$\mathcal{C}_{inc}$	Include constraint
	$\mathcal{C}_{inp}$	Input constraint
	$\mathcal{C}_{iss}$	Issue constraint
	$\mathcal{C}_{ord}$	Ordering constraint
	$\mathcal{C}_{out}$	Output constraint
	$\mathcal{C}_{res}$	Resource constraint
	$\mathcal{C}_{var}$	Variable constraint
$\mathcal{C}rt$		Criteria
$\mathcal{D}$		Domain level
$\mathcal{D}r$		Decision rationale
$\mathcal{E}$		Entity
$\mathcal{E}vt$		Event
$\mathcal{E}xp$		Expression
$\mathcal{I}nt$		Integer
$\mathcal{N}$		Node
	$\mathcal{N}o$	Other node
	$\mathcal{N}s$	Start node
	$\mathcal{N}f$	Finish node
	$\mathcal{N}b$	Begin node
	$\mathcal{N}e$	End node
$\mathcal{O}bj$		Objective
$\mathcal{O}pt$		Option
$\mathcal{O}s$		Objective specification
$\mathcal{P}l$		Plan
$\mathcal{P}$		Process
$\mathcal{Q}$		Question
$\mathcal{R}s$		Rationale specification
$\mathcal{S}$		Set
$\mathcal{S}tr$		String
$\mathcal{T}p$		Timepoint
$\mathcal{T}set^{\mathcal{A},\mathcal{A}}$		Timepoint pair set
$\mathcal{U}$		Resource unit

B. Interval Definitions for CPA

This axiomatisation is based on Hayes' definitions [17] of the isomorphic relationship between data structures in a timepoint-based theory and an interval-based theory [1, 2].

$$\begin{aligned}(\forall a).(before(begin - timepoint(a),\\ end - timepoint(a)))\\ \supset timeinterval(a)\end{aligned}$$

$$\begin{aligned}(\forall tp_1 tp_2).(tp_1 = begin - timepoint(between(tp_1, tp_2)) \wedge\\ tp_2 = end - timepoint(between(tp_1, tp_2)))\\ \Leftrightarrow before(tp_1, tp_2)\end{aligned}$$

$$\begin{aligned}(\forall a_1 a_2).(timeinterval(a_1) \wedge timeinterval(a_2) \wedge\\ before(end - timepoint(a_1), begin - timepoint(a_2)))\\ \Leftrightarrow precedes(a_1, a_2)\end{aligned}$$

$$\begin{aligned}(\forall a_1 a_2).(timeinterval(a_1) \wedge timeinterval(a_2) \wedge\\ before(begin - timepoint(a_1), begin - timepoint(a_2)) \wedge\\ before(begin - timepoint(a_2), end - timepoint(a_1)) \wedge\\ before(end - timepoint(a_1), end - timepoint(a_w)))\\ \Leftrightarrow overlaps(a_1, a_2)\end{aligned}$$

$$\begin{aligned}(\forall a_1 a_2).(timeinterval(a_1) \wedge timeinterval(a_2) \wedge\\ begin - timepoint(a_1) = begin - timepoint(a_2) \wedge\\ before(end - timepoint(a_1), end - timepoint(a_2)))\\ \Leftrightarrow starts(a_1, a_2)\end{aligned}$$

$$\begin{aligned}(\forall a_1 a_2).(timeinterval(a_1) \wedge timeinterval(a_2) \wedge\\ before(begin - timepoint(a_2), begin - timepoint(a_1)) \wedge\\ before(begin - timepoint(a_1), end - timepoint(a_1))) \wedge\\ before(end - timepoint(a_1), end - timepoint(a_2)))\\ \Leftrightarrow during(a_1, a_2)\end{aligned}$$

$$\begin{aligned}(\forall a_1 a_2).(timeinterval(a_1) \wedge timeinterval(a_2) \wedge\\ before(begin - timepoint(a_2), begin - timepoint(a_1)) \wedge\\ end - timepoint(a_1) = end - timepoint(a_2))\\ \Leftrightarrow finishes(a_1, a_2)\end{aligned}$$

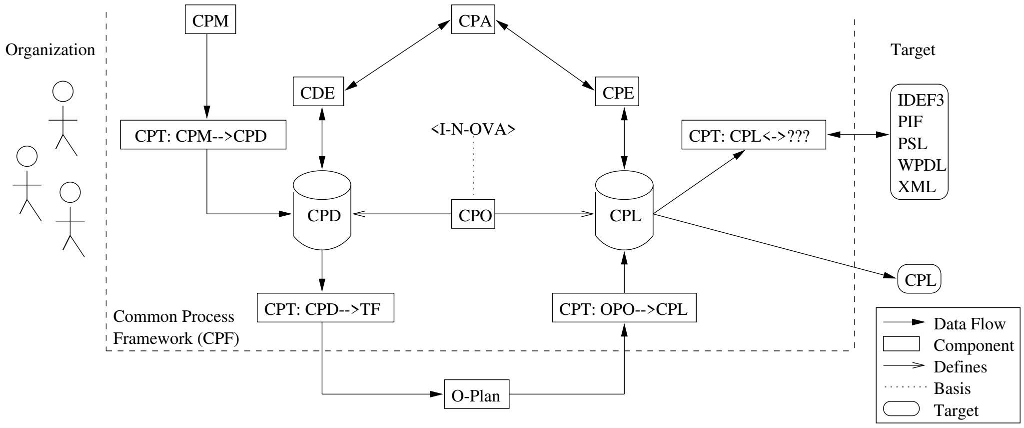

Figure 1: The Common Process Framework (CPF)
Title: A Common Process Ontology for Process-Centred Organisations
Authors: Polyak, S., & Tate, A.

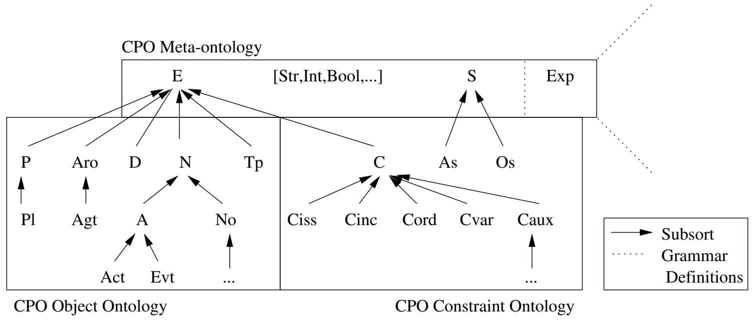

Figure 2: 3-CPO: Meta, Object, and Constraint Ontology
Title: A Common Process Ontology for Process-Centred Organisations
Authors: Polyak, S., & Tate, A.

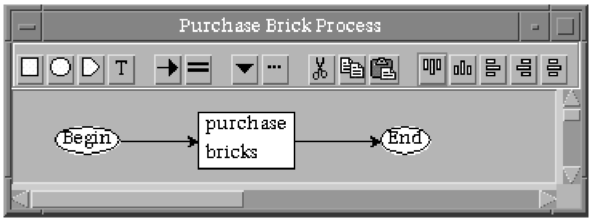

Figure 3: Simple Process Example
Title: A Common Process Ontology for Process-Centred Organisations
Authors: Polyak, S., & Tate, A.