Analysis of Candidate PSL Process/Plan Representations

Stephen T. Polyak[*]

Department of Artificial Intelligence
The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN
United Kingdom
E-mail: Steve_Polyak@ed.ac.uk

Austin Tate

Artificial Intelligence Applications Institute
The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN
United Kingdom
E-mail: a.tate@ed.ac.uk

Abstract

This paper is a summary of analysis work completed during phase 2 of the National Institute of Standards and Technology's (NIST) Process Specification Language (PSL) project. A set of requirements for this language was produced in phase 1. These requirements were divided into separate categories. The main categories of interest for this analysis are the core and outer core requirements. Each set was then further partitioned into either representational or functional requirements. In phase 2, existing candidate process representations were proposed which were believed to satisfy much of the representational and functional needs. Most of the particular set of candidates reviewed are those representations generated by participants in the DARPA/Rome Laboratory Planning Initiative (ARPI), or representations in which participants in the initiative played a part. The results of the analyses of a number of the candidate representations with respect to the requirements are reviewed here.

[*]With contributions from PSL working group members which are specifically credited in the paper where provided.

Contents

List of Figures

1 Introduction

The National Institute of Standards and Technology's (NIST) Process Specification Language (PSL) project [Schlenoff *et al.* 96] grew out of a need for a shared process representation in a manufacturing environment. The project members have summarised their goal as

> "... to create a process representation that is common to all manufacturing applications, generic enough to be decoupled from any given application, and robust enough to be able to represent the necessary process information for any given application. This representation would facilitate communication between the various applications because they would all speak the same language" [Schlenoff *et al.* 96]

While the motivation was primarily from a manufacturing perspective, the project also recognised that this "shared representation" should apply to a broad range of systems and uses including: project management, business process reengineering, workflow management, process planning, and production scheduling. In addition to the NIST core project members, a number of individuals from academia, governmental and industrial organisations were assembled to assist in the effort. Steve Polyak and Austin Tate from the O-Plan project at the University of Edinburgh are participants in the NIST PSL project.

The project set out to achieve its goal through a series of phases that structured the work. During phase 1, a set of requirements were developed that were expected to be satisfied by the end-product language [Schlenoff *et al.* 96, Gruninger *et al.* 97]. In phase 2, a variety of existing representations were identified that could address the requirements to some degree. A number of participants performed analyses of various representations relative to the specified requirements. This paper is a summary of the analysis work completed by the authors[1] during 12/96 - 3/97 for phase 2 of the project.

Section 2 takes a look at the PSL project in more detail. A summary of the representations we examined and some of our reasons for including them are outlined in section 3. The result of the analysis is presented in section 4 which is then discussed in more depth in section 5. Finally, we review some conclusions of this work in section 6.

2 Process Specification Language

2.1 Mission

NIST's mission is to promote U.S. economic growth by working with industry to develop and apply technology and standards. One of the working areas involves the development of standards for the manufacturing industry. There exists an identified need within the manufacturing industry for a standard representation of processes and process information. The NIST Process Specification Language (PSL) is being developed as an interlingua representation that can be used for sharing process knowledge within a manufacturing environment.

2.2 Phase 1

In the initial phase of this work, a set of requirements were gathered by inspecting a number of applications which utilise process knowledge [Schlenoff *et al.* 96]. These requirements were categorised into: core, outer core, plug-in, and application-specific groupings. The analysis in this paper is only concerned with the first two categories. The core reflects those requirements that the PSL group concluded were either essential, critical, or typically common for all of their identified uses of process knowledge. The outer core contains requirements which are considered to be "pervasive" but not necessarily essential. The core and outer core are further sub-divided into either representational or functional requirements.

[1] With the exception of the OMWG CPR analysis, which was completed by Adam Pease at Teknowledge, Inc. and the PIF analysis which was jointly produced by the authors and other members of the PIF working group.

2.3 Phase 2

The approach for phase 2 was to identify existing representations that were believed to address these requirements. Each representation was then assessed by assigning an evaluation of its coverage for each requirement in the core and outer core. The possible values for each requirement were: completely satisfies, partially satisfies, cannot satisfy, or uncertain. Analysts were also asked to provide comments and/or describe constructs that supported their rating for each entry.

3 Candidate Representations

As mentioned in the previous section, a number of candidate representations were identified by various PSL members based on the phase 1 requirements. The idea was to provide an overall picture of how existing representations addressed various process/plan requirements in a number of ways. Candidates that were "strong" in certain areas might suggest good representational approaches to consider. With one exception, the representations offered and their corresponding analyses shown here were influenced strongly by the authors' knowledge and experience with DARPA/Rome Laboratory Planning Initiative (ARPI) or other DARPA-funded plan, process, and schedule representations. The exception representational analysis was ISO's STEP Part 49 for which the anlysis was performed as a validation check of another analysis and the overall approach. These representations are summarised below.

3.1 ACT

Traditionally, plan generation and reactive execution have been considered as separate activities, with few attempts to integrate them within a single system. The ACT formalism [Wilkins & Myers 95] is a language for representing the knowledge required to support both the generation of complex plans and reactive execution of those plans in dynamic environments. ACT has been used as the interlingua in an implemented system that links a previously implemented planner (SIPE-2) with a previously implemented executor (PRS).

ACT is intended to serve as a general-purpose representation language that could be used to share knowledge between many different execution and planning systems. The representational and computational adequacy of ACT has been validated by implementing the Cypress system, which uses ACT as an interlingua to enable runtime interactions between planning and execution subsystems. ACT focuses on a practical, yet sufficiently expressive representation that can address a variety of needs. Sample domains that ACT has been used in include: controlling an indoor mobile robot, and military operations planning.

The ACT formalism is a domain-independent language for representing the kinds of knowledge about activity used by both plan generation and reactive execution systems. The basic unit of representation is an Act, which can be used to encode both plan fragments and standard operating procedures (SOPs). An Act describes a set of actions that can be taken to fulfill some designated purpose under certain conditions. The purpose could be either to satisfy a goal or to respond to some event in the world. The purpose and applicability criteria for an Act are formulated using a fixed set of environment conditions. Action specifications are called the plot, and consists of a partially ordered set of actions and subgoals. The environment conditions and plots are specified using goal expressions, each of which consists of one of a predefined set of meta-predicates applied to a logical formula. The meta-predicates permit the specification of many different modes of activity, including goals of achievement, maintenance, and testing. ACT can be used to build a very strong model of the relationships between actions, temporal requirements, and resources. It has been shown to have expressive and computational adequacy in several applications. Specific manufacturing elements would need to be added as extensions to support these domain-specific requirements.

ACT has been used as the common representation for SIPE-2 and PRS in SRI's Cypress system [Wilkins *et al.* 95] [2]. SIPE-2 was also used as the core reasoning engine in SRI's SOCAP (System for Oper-

[2] Cypress = SIPE + PRS

ations Crisis Action Planning) [Wilkins & Desimone 94], which was part of the second Integrated Feasibility Demonstration of ARPI [Fowler *et al.* 96].

3.2 CPR

The DARPA-sponsored Object Model Working Group is currently developing a "core plan representation" (CPR) [Pease & Carrico 97] which is aimed at supporting the representational needs of many types of planning systems. The OMWG's stated goal is

"...to leverage common functionality and facilitate the reuse and sharing of information between a variety of planning and control systems" [Pease & Carrico 97]

CPR has utilised ARPI work on KRSL[3] [Lehrer(ed.) 93], the POCG[4], the O-Plan project [Currie & Tate 91], and the <I-N-OVA> representation (see below). CPR is composed of a set of basic plan concepts that have been assembled into a refined design framework. The initial minimal set of concepts included Action, Resource, Actor, and Objective. This set was then: expanded with more entities (e.g. Plan, TimePoint, etc.), defined with individual properties (e.g. an Actor has an Objectives slot, etc.) and structured with stated relations (e.g. a Plan contains Actions, Actions contain TimePoints, etc.)

CPR's intended application might involve the Joint Task Force Advanced Technology Demonstration (JTF-ATD) and Joint Forces Air Component Commander (JFACC) programs which are two DARPA joint force military planning applications. The OMWG has also identified the possibility of applying CPR to non-military applications as well.

3.3 <I-N-OVA>

<I-N-OVA> [Tate 96c, Tate 96b, Tate 97] is a constraint model of tasks, plans, processes, and activities which adopts the perspective that all of these sources are "constraints on behaviour". This model can be used as an ontology for shared representations amongst various operations in the planning and execution process including: knowledge acquisition, formal analysis, user communication, and system manipulation.

The acronym, <I-N-OVA> stands for: Issues; Nodes; and Ordering, Variable, and Auxiliary constraints. Issues and nodes are also expressed as constraints and can be thought of as implied constraints and activity constraints, respectively. The inclusion of "issues" in the specification of a plan or process is unique and allows the "state" of the planning process to be captured and communicated throughout the life-cycle of a plan. Tate relates these various constraint types together by stating

"Planning is the taking of planning decisions (I) which selects the activities to perform (N) which creates, modifies or uses the plan objects or products (V) in the correct time (O) within the authority, resources and other constraints specified (A)."[5]

<I-N-OVA> is not a representation language like some of the other candidates discussed in this paper (e.g. ACT, O-Plan TF). Rather, it is a conceptual model which can underly languages which describe activities, plans and processes. O-Plan's widely used domain description language (TF) can be seen as an implementation that rests upon the more general <I-N-OVA> model. The different types of constraints in the <I-N-OVA> model reflect the different types of components in an O-Plan agent (issue controller, issue handlers, and plug-in constraint managers) [Tate *et al.* 96].

[3]KRSL - Knowledge Representation Source Language

[4]POCG - Planning Ontology Construction Group, See Appendix in [Tate 96c]

[5]See <I-N-OVA> rationale at http://www.aiai.ed.ac.uk/~oplan/inova.html

3.4 OZONE

OZONE [Smith & Becker 97] is a toolkit for configuring constraint-based scheduling systems [6]. A central component of OZONE is its scheduling ontology, which defines a reusable and extensible base of concepts for describing and representing scheduling problems, domains and constraints.

It had been noticed that there is commonality in scheduling system requirements and design at several levels across application domains. Many of the concepts in the problems, domain and constraints could be considered to be reusable and extensible. The OZONE ontology provides a framework for analysing the information requirements of a given target domain, and a structural foundation for constructing an appropriate domain model. Through direct association of software component capabilities with concepts in the ontology, the ontology promotes rapid configuration of executable systems and allows concentration of modelling effort on those idiosyncratic aspects of the target domain. The OZONE ontology and toolkit represent a synthesis of extensive prior work in developing constraint-based scheduling models for a range of applications in manufacturing, space and transportation logistics.

OZONE adopts an activity-centred modelling viewpoint. There are five basic concepts of the ontology - Demand, Activity, Resource, Product, and Constraint. The ontology also defines specific inter-relationships and properties for these entities.Scheduling is defined as a process of feasibly synchronising the use of resources by activities to satisfy demands over time, and application problems are described in terms of this abstract domain model.

OZONE has a powerful architecture that permits a domain modeller to focus on those items that are special for a specific instance. The use of constraint managers assists in rapid identification of aspects to consider. While the work on OZONE reflects years of experience in the scheduling field, the ontology is still relatively new.

3.5 PIF

Critical in Business Process Reengineering or Enterprise Integration is the ability to share and interlink heterogeneous process models. The goal of the PIF (Process Interchange Format) project [Lee *et al.* 96] is to support the exchange of business process models across different formats and schemas. The project pursues this goal by developing PIF (a common translation language that serves as a bridge among heterogeneous process representations), translators between PIF and local process representations, and a mechanism for extending PIF to accommodate different expressive needs in a modular way (Partially Shared Views).

At the heart of PIF is a core set of classes. Some of these classes are described in the following excerpt.

> In PIF, everything is an ENTITY; that is, every PIF construct is a specialization of ENTITY. There are four types of ENTITY: ACTIVITY, OBJECT, TIMEPOINT, 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 TIMEPOINTS [Lee *et al.* 96].

The PIF project aims to support translations 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 minimised. If a translator cannot translate part of a PIF process description to its target format, it should: 1) Translate as much of the description as possible (and not, for example, simply issue an error message and give up) 2) Represent any untranslatable parts so that the translator can add them back to the process description when it is translated back into PIF[7].

[6]This builds on earlier work with OPIS [Smith 94]

[7]Most of this PIF summary was written for the PSL group by Jintae Lee, University of Hawaii, College of Business Administration

3.6 STEP-49

This representation analysis is the only one here that wasn't part of the authors' proposed set of representations. This analysis was performed by the authors at the PSL members' request as a verification check of another analysis performed for STEP-49. We include the results here as an additional contibution to the phase 2 effort.

Part 49 (Process structure and properties) [ISO 95] is an Integrated generic resource of STEP (Standard for the Exchange of Product model data) written in EXPRESS. It specifies the information necessary to specify the actions or potential actions to realize a process. This includes the relationships between the actions or potential actions in the process and the relationships between the processes that are used to realize a product. A process plan is the specification of instructions to realize a product. This part does not specify any particular process, but defines the elements to exchange process information. This part is applicable to all types of process definitions that can be represented in a discrete manner.

The constructs define the structure for specifying: relationships between processes, the effectivity of a process, the properties of a process, the resources required for the process, the properties of the resource, the representation of process, the representation of the resource, and the relationship of the process to the product. Together, these constructs can be combined to create a process plan.

Part 49 is broken up into three schemas: method_definition_schema; process_property_schema; and process_property_representation_schema. The method_definition_schema is the specification of the instructions required to perform a process that augments the product definition, defines the product, or contributes to the production of a product. The process_property_schema defines the properties of the actions of the process, the properties of the action_methods of the process, the properties of the resources to be used in the execution of the process, and the properties of the product that will result from the execution of the process. The process properties are the properties of the actions, resources, and products that are part of the process. The process_property_representation_schema represents the properties required by either a resource, an action, or a potential action to effect a process. This could include either a resource parameter value or an action parameter value.

Part 49 is able to cover many aspect of process representation because of the generality of the constructs. For example, there is a construct called action_method_relationship which can either be a concurrent_action_method or a serial_action_method and may be associated with a relationship_with_condition. With these constructs, one is able to model alternative tasks, concurrent tasks, parallel tasks, serial tasks, conditional tasks, iterative loops, and abstraction (decomposition). Although this representation is adequate, the level in which it is written might make it difficult for a systems developer to understand the meaning and usefulness of the constructs[8].

3.7 TF

The O-Plan (Open Planning Architecture) Project [Currie & Tate 91, Tate *et al.* 96] is exploring issues of coordinated command, planning and control. The objective of the O-Plan Project at the Artificial Intelligence Applications Institute (AIAI) is to develop an architecture within which different agents have command (task assignment), planning and execution monitoring roles. "Task Formalism" (TF) is a language that is used to convey a detailed description of permissible actions or operations within an application area, including information about how constraints imposed on the use of these actions should be satisfied, and their effects on the domain if the actions are used.

O-Plan is a domain independent planning system. The agents in this system require the input of a domain representation in order to complete their respective tasks. Task Formalism is used to provide this detailed knowledge. Task Formalism was originally developed for the NONLIN planner in 1975 and has been extended and refined for use in O-Plan.

[8]This STEP-49 summary was written for the PSL group by Craig Schlenoff, National Institute of Standards and Technology (NIST)

Task Formalism is used to give an overall hierarchical description of an application area by specifying the possible activities within the application domain and describing how those activities can be "expanded" into sets of sub-activities with a wide range of contraints imposed. Plans are generated by choosing suitable expansions for activities in the plan (i.e. refining those activities) and including the sets of more detailed sub-activities described by the chosen expansions. Ordering constraints are then satisfied to ensure that asserted effects of some actions satisfy, and continue to satisfy, conditions on the use of other actions. Other temporal and resource constraints are also included in the descriptions. These descriptions of actions form the main structure within TF, the schema. Schemas are also used in a completely uniform manner to describe tasks set to the planning system, in the same formalism. Other TF structures hold global information of various sorts and heuristic information about preferences for choices to be made during planning.

TF can be used to represent complex knowledge about a domain. This "rich" knowledge includes action effects and conditions, hierarchical relationships, temporal requirements, authority, resource needs, etc. Its constraint-based approach provides a strong, extensible approach to domain representation. When examining O-Plan's Task Formalism with respect to the requirements of NIST's Process Specification Language (PSL) it was noted that certain aspects that are more closely related to manufacturing were lacking in TF (cost data, milestones, etc.) O-Plan TF is a specific language for planning and lacks some of the generality provided by a conceptual model such as <I-N-OVA> on which it is based.

4 Results

The data of the analyses is presented in appendices A and B. Appendix A provides a cross-comparison matrix to easily see what the rankings were for each representation. Appendix B provides the detailed tables for each representation which lists the constructs identified and/or explanations which help to support a specific rating. In some cases where there were multiple contributors, such as the PIF detailed analysis [9], there are cited comments to show the various inputs.

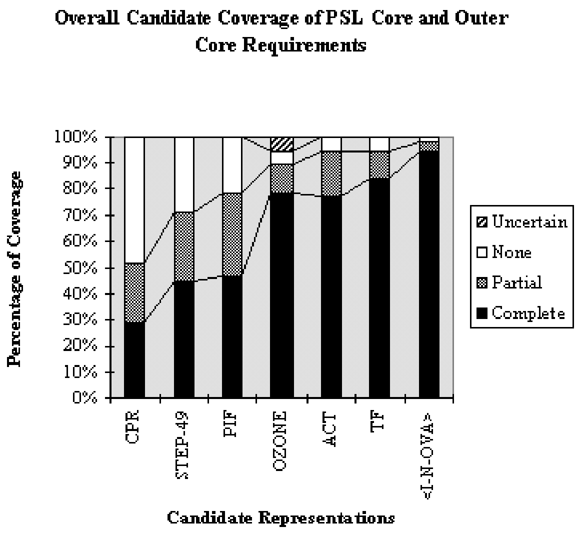

Figure 1: Overall Analysis Results

There were proportionately more outer core (41 out of 56 total, or 73%) requirements than there were

[9]The PIF analysis was compiled by a number of the PIF working group members including the authors.

core requirements (15 out of 56 total, or 23%). No effort was made to weight the importance of either the membership of the requirement to a specific set nor for its individual importance. The ratings from these tables in the appendices have been summarised in three separate figures. Figure 1 provides a chart illustrating the overall coverage for each representation. Figures 2 and 3 separate the overall results into core and outer core coverages, respectively.

In figure 1, the candidate representations are shown on the x-axis. These representations are roughly sorted by an ascending coverage rating. Each rating shows the distribution of assesments for the combined core and outer core requirements. The "complete" data set (in solid black) shows the percentage of requirements that were considered to be "completely satisfied". So, for example, OZONE and ACT were considered to "completely" satisfy around 75% of all the requirements whereas PIF and STEP-49 were around 45%. Now, if we add the set of "partially satisfied" requirements to the "completely satisfied" set we can see the summation of these percentage below the "partial" series (in grey). Given this combined set we can see, for example, that ACT and TF are around 90-95% whereas PIF is around 75%. The "none" series (in white) shows the percentage of remaining requirements that the representation was considered unable to satisfy at all. The figure also shows that a small number of uncertain entries which still remained in the OZONE analysis.

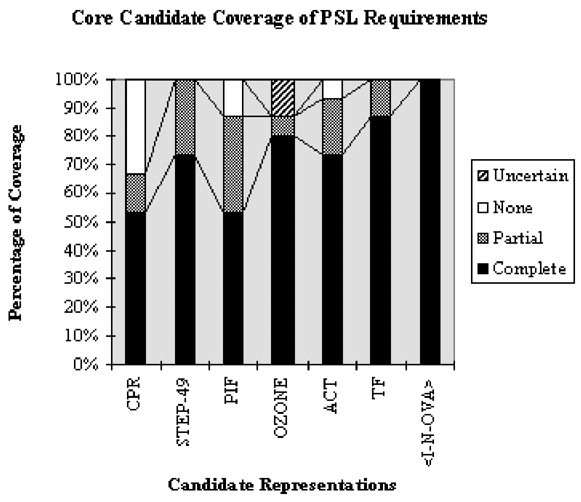

Figure 2: Results of Core Analysis

Figure 2 is expressed in the same format as just described for figure 1, but figure 2 only contains the data from the core category of the requirements. Rather than resorting this group by its coverage rating as we did in figure 1, we have left the order alone. This helps to show the difference between the overall ratings vs. the core alone. We can see, for example, that while PIF had a slightly higher overall rating (in figure 1) than STEP-49, it had a lower core rating. This somewhat surprising result is discussed more in the following section.

The outer core results are reflected in figure 3. As cited above, 73% of the requirements came from this set so it is not too surprising to see that the relative distribution of the ratings is similiar to the overall rating distribution in figure 1. This data shows a definite cut-off between the ratings assigned to PIF (and the representations to the left of it) vs. OZONE (and the representations to the right of it). A result which will also be discussed in the next section is the apparently relatively low rating of CPR.

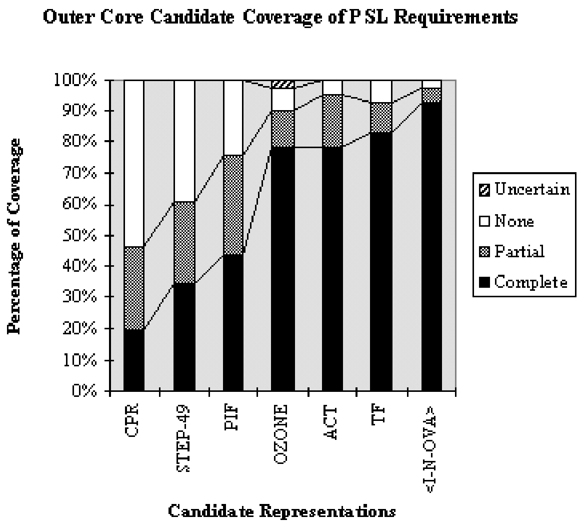

Figure 3: Results of Outer Core Analysis

5 Discussion

5.1 Analysis Issues

As we discuss these results, a few points should be made clear relating to certain issues in the analysis process. The first issue has to do with the "interpretation" of the scale provided for the analysis. This scale again was either: completely satisfy, partially satisfy, cannot satisfy, or uncertain. It is obvious that this is a very large-grained measure and is subject to a number of perspectives on what it "means" to satisfy a requirement. A more fine-grained scale could have been used but it would be even harder to explain why a representation received an 8 vs. a 7, for example.

Some members performing the analysis made the judgement that a requirement could only be met if there was a specific construct that was specifically designed to meet a given requirement. So, for example, if a representation used a frame-based syntax and did not have a slot specifically designed for "deadline management" (see requirement 2.2.4), then it did not "satisfy" the requirement. Another perspective was to look at the available constructs of a representation and to determine if there was a way in which the requirement could be expressed (e.g. deadlines can be achieved through an association of an activity status and a specific time point). The latter approach was the one taken by the authors in these analyses (with the exception of CPR, which was the result of another analyst[10]. This difference lead to a skewing of comparsions between analyses performed by separate analysts. Specifically, while the CPR analysis was included in this paper to associate its results with the other plan-based representations (e.g. ACT, TF), it must be understood that conclusions cannot be directly made on the comparisons between CPR and the rest presented here. The other representation analyses are considered to be homogenous.

Another issue that confuses the interpretation of these results is the repeated use of the term "core". In representations like PIF and CPR the term, core, is used to mean the set of elements that are central to the representation of a process or plan. This generally implies that the concepts which were considered to be superflous were moved outside of the core. On the other hand, when we are talking about the PSL "core" requirements, we are referring to those items that were judged by the PSL group to be "important" or

[10]Specifically, Adam Pease, Teknowledge, Inc.

"necessary" requirements. These requirements might map to concepts which may not be central to processes knowledge but are more reflective of PSL's perceived need. This helps to explain why the PIF-Core didn't match very well to the core requirements, but something like STEP's Part 49 did. What was core to PIF was not necessarily reflected by the PSL core requirements.

5.2 Specific Representation Results

Before we discuss some of the results of the representations reviewed here, it should be pointed out that there were 19 other representations that were reviewed as part of the phase 2 effort as well. This included representations like PERT networks, Gantt charts, KIF, Entity-Relationship diagrams, IDEF3, etc. The set in this paper was the authors' contribution[11] and contains most of the representations that were consistently rated very high amongst the complete phase 2 set. This rating again might be partially skewed due to possibly different perspectives on how to rank the entries.

It is not too surprising that <I-N-OVA> rated the highest in this set due to its intended use. As described in section 3.3, <I-N-OVA> is a conceptual model which can be thought of as a candidate to underly the future PSL language, just as it does with O-Plan TF. In fact it was interesting to review the few parts that <I-N-OVA> didn't address in the PSL requirements (e.g. probablistic uncertainty). O-Plan TF, SRI's ACT, and CMU's OZONE were all relatively equivalent and reflect their long pedigree in representing plan and process knowledge. It is easy to understand that we then take a step down from these more complete and polished representations to review the relatively newer, and spartan PIF-Core and CPR model (again it is possible that the CPR rating might have been higher, possibly more equivalent to PIF-Core, if viewed from another perspective). Possible future extensions to these representations could help them address some of the un-met requirements. It was interesting to note that while part 49 of STEP addressed a number of core items, its overall rating was closer to that of PIF-Core than ACT, OZONE, and O-Plan TF.

6 Conclusions

In a way, these results confirm many of the things one would expect to see when comparing these candidates against a set of rigorous process requirements. <I-N-OVA>, being the most general representation, was expected to be able to address almost all of the concepts that were part of the requirements. The notion of constraints provided an adequate representation for identfying ways to express the various PSL requirements. ACT, O-Plan TF, and OZONE representations, anchored in prototype AI planning and scheduling systems, were hypothesized to be roughly equivalent in what they could and could not handle. Since the PIF-Core and CPR model represent a much smaller and more compact set it was consistent to see a slight drop in their coverage.

The representations developed for AI planning, scheduling and in earlier process/plan interchange languages have been shown to be good "candidates" of ideas and concepts relative to the requirements defined in phase 1. It is hoped that these ideas and concepts can be used to feed into the future work on NIST's PSL project as it moves forward to phase 3 work on a proposed language.

[11] Unless otherwise noted.

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 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.

Thanks to Adam Pease for the CPR analysis results and to the members of the PIF and PSL working groups for their cited contributions.

References

[Currie & Tate 91] K. Currie and A. Tate. O-Plan: the open planning architecture. *Artificial Intelligence*, 52:49–86, 1991.

[Fowler *et al.* 96] N. Fowler, S.E. Cross, T.D. Garvey, and M. Hoffman. Overview: ARPA-Rome laboratory knowledge-based planning and scheduling initiative (ARPI). In Tate [Tate 96a], pages 3–9.

[Gruninger *et al.* 97] M. Gruninger, C. Schlenoff, A. Knutilla, and S. Ray. Using process requirements as the basis for the creation and evaluation of process ontologies for enterprise modeling. *ACM SIGGROUP Bulletin Special Issue on Enterprise Modelling*, 18(3), 1997.

[ISO 95] ISO. Product data representation and exchange: Part 49: Integrated generic resources: Process structure and properties. Technical Report ISO Standard 10303-49, International Standards Organization, 1995.

[Lee *et al.* 96] J. Lee, M. Gruninger, Y. Jin, T. Malone, A. Tate, and G. Yost. The PIF process interchange format and framework version 1.1. Technical Report Working Paper No 194, MIT Center for Coordination Science, 1996.

[Lehrer(ed.) 93] N. Lehrer(ed.). ARPI KRSL reference manual 2.0.2. Technical Report February, ISX Corporation, 1993.

[Pease & Carrico 97] R.A. Pease and T. Carrico. The JTF ATD core plan representation: A progress report. In *Proceedings of the AAAI Spring Symposium on Ontological Engineering*, to appear, 1997.

[Schlenoff *et al.* 96] 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.

[Smith & Becker 97] S.F. Smith and M. Becker. An ontology for constructing scheduling systems. In *Working Notes of 1997 AAAI Symposium on Ontological Engineering*, Stanford, CA, 1997. AAAI Press.

[Smith 94] S.F. Smith. OPIS: A methodology and architecture for reactive scheduling. In M. Zweben and M. Fox, editors, *Intelligent Scheduling.* Morgan Kaufmann Publishers, 1994.

[Tate 96a] A. Tate, editor. *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*, Menlo Park, CA, 1996. AAAI Press.

[Tate 96b] 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.

[Tate 96c] A. Tate. Towards a plan ontology. *AI*IA Notiziqe (Quarterly Publication of the Associazione Italiana per l'Intelligenza Artificiale), Special Issue on Aspects of Planning Research*, 9(1):19–26, March 1996.

[Tate 97] A. Tate. Mixed initiative interaction in O-Plan. In *Proceedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction*, Stanford, California, March 1997.

[Tate *et al.* 96] A. Tate, B. Drabble, and J. Dalton. O-Plan: a knowledge-based planner and its application to logistics. In Tate [Tate 96a], pages 259–266.

[Wilkins & Desimone 94] D.E. Wilkins and R.V. Desimone. Applying an AI planner to military operations planning. In M. Fox and M. Zweben, editors, *Intelligent Scheduling*, pages 685–709, San Mateo, CA, 1994. Morgan Kaufmann Publisers, Inc.

[Wilkins & Myers 95] D.E. Wilkins and K.L. Myers. A common knowledge representation for plan generation and reactive execution. *Journal of Logic and Computation*, 5(6):269–301, 1995.

[Wilkins *et al.* 95] D.E. Wilkins, K.L. Myers, J.D. Lowrance, and L.P. Wesley. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical AI*, 7(1):197–227, 1995.

A Representation Cross-Comparison Matrix

Key: √ - Completely satisfies req., * - Partially satisfies req., X - Cannot satisfy req., ? - Uncertain

	PSL Requirement	**ACT**	**CPR**	**<I-N-OVA>**	**OZONE**	**PIF**	**STEP-49**	**TF**
1	**Core Requirements**							
1.1	*Representational Req.*							
1.1.1	Ad hoc Notes	√	√	√	?	*	√	√
1.1.2	Cost Data	*	X	√	*	X	*	*
1.1.3	Level of Effort	X	√	√	√	X	*	√
1.1.4	Product Characteristics	*	*	√	√	*	*	*
1.1.5	Resource	√	√	√	√	√	√	√
1.1.6	Resource Requirements for a Task	√	√	√	√	*	√	√
1.1.7	Simple Groupings	√	√	√	√	√	√	√
1.1.8	Simple Resource Capability/Characteristics	*	*	√	√	*	√	√
1.1.9	Simple Sequences	√	√	√	√	√	√	√
1.1.10	Simple Task Representation and Characteristics	√	X	√	?	√	√	√
1.1.11	Task Duration	√	√	√	√	√	√	√
1.1.12	Task Executor	√	√	√	√	√	√	√
1.2	*Functional Req.*							
1.2.1	Extensibility	√	X	√	√	√	√	√
1.2.2	Resource Allocation /deallocation for one or many tasks	√	X	√	√	*	*	√
1.2.3	Simple Precedence	√	X	√	√	√	√	√
2	**Outer Core Req.**							
2.1	*Representational Req.*							
2.1.1	Composition /Decomposition	√	√	√	*	√	X	√
2.1.2	Alternative Task	√	*	√	√	√	√	√
2.1.3	Associated Illustrations and Drawings	√	√	√	?	*	√	√
2.1.4	Complex Groups of Tasks	√	X	√	√	*	√	√
2.1.5	Complex Resource Characteristics	*	*	√	√	X	√	√
2.1.6	Complex Sequences	√	*	√	√	√	√	√
2.1.7	Complex Task Representation and Parameters	√	X	√	√	*	√	√
2.1.8	Concurrent Tasks	√	√	√	√	√	√	√
2.1.9	Conditional Tasks	√	√	√	*	√	√	√
2.1.10	Confidence Levels	*	√	X	X	X	X	X
2.1.11	Constraints	√	*	√	√	√	*	√
2.1.12	Multiple Duration(s)	*	*	√	*	*	X	√
2.1.13	Implicit/Explicit Resource Association	*	*	√	√	√	X	X
2.1.14	Iterative Loops	√	X	√	X	√	*	X

Appendix A - Cross-Comparison Matrix

	PSL Requirement	**ACT**	**CPR**	**<I-N-OVA>**	**OZONE**	**PIF**	**STEP-49**	**TF**
2.1.15	Manual vs. Automated Tasks	√	X	√	√	X	*	√
2.1.16	Manufacturing Product Quantity	X	*	√	√	X	*	√
2.1.17	Material Constraints	X	*	√	√	X	*	*
2.1.18	Parallel Tasks	√	*	√	√	√	*	√
2.1.19	Parameters and Variables	√	X	√	√	√	√	√
2.1.20	Pre- and Post-processing Constraints	√	*	√	√	√	*	√
2.1.21	Queues, Stacks, Lists	*	X	*	*	*	X	*
2.1.22	Resource Categorization and Grouping	√	X	√	√	*	*	√
2.1.23	Resource Location	√	√	√	√	X	√	√
2.1.24	Resource/Task Combined Characteristics	√	X	√	√	X	X	√
2.1.25	Serial Tasks	√	*	√	√	√	√	√
2.1.26	State Existence Constraints	√	X	√	√	√	X	√
2.1.27	State Representations	√	X	√	√	*	X	√
2.1.28	Temporal Constraints	√	√	√	√	*	*	√
2.1.29	Uncertainty/Variability /Tolerance	*	√	√	*	X	X	√
2.2	*Functional Req.*							
2.2.1	Ability to Insert or Attach a Highlight(milestones)	*	X	*	X	X	X	*
2.2.2	Complex Precedence	√	X	√	√	√	√	√
2.2.3	Convey the Ancestry or Class of a Task	√	X	√	√	√	X	√
2.2.4	Deadline Management	√	X	√	√	*	X	√
2.2.5	Dispatching	√	X	√	√	X	*	√
2.2.6	Eligible Resources	√	X	√	√	*	√	√
2.2.7	Exception Handling and Recovery	√	X	√	X	√	*	*
2.2.8	Information Exchange Between Tasks	√	X	√	√	*	X	√
2.2.9	Mathematical and Logical Operations	√	X	√	√	√	√	√
2.2.10	Support for Task/Process Templates	√	X	√	√	*	X	√
2.2.11	Support for Simultaneously Maintained Associations of Mult Lev of Abstraction	√	X	√	√	√	X	√
2.2.12	Synchronization of Multiple, Parallel Task Sequences	√	X	√	√	*	X	√

Appendix A - Cross-Comparison Matrix

B Detailed PSL Analysis for Each Candidate Representation

B.1 SRI's ACT Formalism Detailed PSL Analysis

Key: $\surd$ - Completely satisfies req., * - Partially satisfies req., X - Cannot satisfy req., ? - Uncertain

	PSL Requirement	**Rank**	**Description**
1	**Core Requirements**		
1.1	*Representational Requirements*		
1.1.1	ad hoc notes and annotations optionally associated with any component of a plan - on-the-fly, off-the-cuff notes and documentation. This could be voice, video, as well as text. A person's observation of a process might go here.	$\surd$	An individual ACT's environment conditions contains a comment slot where text can be inserted. This may include a file reference to drawings, etc. Also, there is a property slot that holds property/value pairs that can be used-defined.
1.1.2	cost data - the cost associated with a resource or task. This could be a fixed cost, cost rate, or a cost derived from other attributes such as duration and level of effort. Costs associated with uncertainty, variability, tolerances, etc	*	ACT permits a reusable resource model that could be utilized to represent some aspects of cost.
1.1.3	level of effort - description of the amount of a resource needed, in any given unit, to accomplish a task. Some example levels of effort are equipment-hour, labor-hour, and crew size.	X	ACT does not support a quantification of resource needs.
1.1.4	product (work item) characteristics - information about an intermediate and final product which a process will produce.	*	This implementation of the ACT formalism does not support a "produceable" model of resources that could be interpreted as products.
1.1.5	resource - a single resource or a group of resources. Some types of resources are equipment, people, information, and in-progress goods.	$\surd$	Resources can be logically represented for use in ACTs.
1.1.6	resource requirement(s) for a task (with quantity) - the relationship between one or more resources and a task.	$\surd$	A simple association of resources and an ACT is established with the resource slot.
1.1.7	simple groups of tasks - very basic, high-level set of tasks. One example is the grouping of tasks and sequences that make up a process plan or that make up a phase.	$\surd$	An ACT's plot consists of a directed graph of nodes that represents a grouping of actions
1.1.8	simple resource capability/characteristics - a high-level description of the characteristics of a resource. More detailed descriptions can be found in the outer core.	*	A rough approximation of characteristics of a resource can be inferred by the typed system used in ACT. (i.e. an Airplane.1 has differentcharacteristics than a Boat.2).

B.1 - SRI's ACT

	PSL Requirement	**Rank**	**Description**
1.1.9	simple sequences - linear, time-sequential groups of tasks. More sophisticated relationships such as parallel and alternative tasks can be found in the outer core.	$\surd$	An ACT's plot consists of a directed graph. The nodes in this graph represent actions and the links represent a partial ordering that can provide simple sequences.
1.1.10	simple task representation and characteristics - a simple, high-level description of the task. More detailed representations can be found in the outer core.	$\surd$	An individual ACT's environment conditions contains a comment slot where text about the actions can be inserted. Also, there is a property slot for user-defined property/value pairing that can also annotate characteristics.
1.1.11	task duration - the time required to complete a task or group of tasks. Only simple durations are represented here.	$\surd$	Time-constraints impose any of the 13 Allen relations between actions. In addition to these constraints, time windows can be setup for start/end, duration, etc.
1.1.12	task executor - who is responsible for executing a task or group of tasks. Examples include a person, controller, or external company if the task is contracted out.	$\surd$	While there isn't an explicit slot for task executor, this property/value could be inserted into the properties slot.
1.2	*Functional Requirements*		
1.2.1	extensibility - there must be a mechanism in place to allow a user to add additional information to the pre-defined data constructs. One such mechanism could be the addition of stubs for user-defined information.	$\surd$	The ACT properties slot provides a mechanism to allow additional user-defined information to be added to the representation.
1.2.2	resource allocation/deallocation for one or many tasks - the assignment and release of one or more resources to a task of group of tasks.	$\surd$	An ACT USE-RESOURCE statement is used to provide a representation for resource allocation/deallocation.
1.2.3	simple precedence - a high-level description of the precedence constraints of one task on another. A more detailed description of precedence and constraints can be found in the outer core.	$\surd$	Various constraints can be placed on the precedence orderings of actions. Temporal constraints can be used to create specific time windows, preconditions can be used to express situational constraints that must be satisfied in order to apply the act.
2	**Outer Core Requirements**		
2.1	*Representational Requirements*		
2.1.1	abstraction - within the scope of this project, there are three concepts of abstraction that must be captured. (hierarchy, incompleteness, ambiguity)	$\surd$	ACTs can be arranged in a hierarchical fashion that links thru the Cue gating slot in a plot. In fact, an ACT is an abstraction of a set of actions and those actions may be abstractions of other ACTs.

B.1 - SRI's ACT

	PSL Requirement	Rank	Description
2.1.2	alternative task - (see complex sequences)	√	ACTs support alternative tasks in a variety of ways. A set of ACTs may share the same cue environment conditions offering alternative choices. Within a plot, conditional arcs can offer disjuctive paths.
2.1.3	associated illustrations and drawings	√	Property/value slots can be assigned to ACTs that contain filename pointers to graphical files, etc.
2.1.4	complex groups of tasks - groups of tasks which have a common tie.	√	An ACT is essentially a complex grouping of tasks (in a plot). ACTs are also connected together via Achieve, Achieve-by, etc.
2.1.5	complex resource characteristics - a detailed description of the characteristics of a resource or group of resources.	*	Since ACT uses typed variables, resources could be grouped as instances of certain classes. Characteristics of a class could then be inferred.
2.1.6	complex sequences - complex ordering relationships between tasks	√	ACTs (and plot nodes) can be ordered in complex relationships that support a variety of conditional, temporal possibilities.
2.1.7	complex task representation and parameters - a detailed representation of a task or group of tasks.	√	Plots provide a very detailed expression of how a task (or grouping of actions) may be completed.
2.1.8	concurrent tasks - (see complex sequences)	√	Concurrency is possible with parallel nodes in plots that can split/join the network.
2.1.9	conditional tasks - a task that only needs to be performed under some predefined circumstance.	√	ACTs (and plot nodes) can use test metapredicates to provide conditional processes.
2.1.10	confidence levels - a measure of certainty that some attribute is true.	*	While there isn't direct support for this within ACT itself, there is a subsystem in the implementation (Gister-CL) that can reason about uncertain information about the world and actions.
2.1.11	constraints - implicit or explicit constraints associated with a task or resource.	√	A variety of constraints can be placed on an ACT that can control things like applicability, temporal limits, etc.
2.1.12	date(s) and time(s) and/or multiple duration(s) - the association of one or more dates and times and/or multiple durations with a resource or task	*	The temporal elements (windows, durations, etc.) are all relative points to other temporal elements in the representation.
2.1.13	implicit/explicit resource association - an implicit or explicit dependency of a resource on another type of resource.	*	This can partially be achieved implicity. For instance, whenever we wish to say that if you use x, you must have y as well. (USE-RESOURCE (x y))

B.1 - SRI's ACT

	PSL Requirement	**Rank**	**Description**
2.1.14	iterative loops - a situation when a task or group of tasks repeats until a desired condition is met	$\surd$	Looping is possible by linking a plot node back to an ancestor node in the graph. Test metapredicates control the number of times.
2.1.15	manual vs. automated tasks - characteristics of a task can differ depending on if a human or a machine is performing that task.	$\surd$	Precondition gating slots can filter which ACT is applicable.
2.1.16	manufacturing product quantity - the amount of the product that is to manufactured.	X	This is not part of ACT.
2.1.17	material constraints	X	This is not supported in ACT.
2.1.18	parallel tasks	$\surd$	Parallel tasks can be defined using parallel plot nodes.
2.1.19	parameters and variables - place holders that can store a constantly changing value.	$\surd$	ACT has a typed variable system that can be bound and rebound as needed.
2.1.20	pre- and post-processing constraints	$\surd$	Preconditions and effects provide both of these.
2.1.21	queues, stacks, lists - the representation of an ordered or unordered group.	*	ACTs support lists of items.
2.1.22	resource categorization and grouping - a logical grouping of resources with a common tie.	$\surd$	Logical categorization and grouping can be done because resource can be considered to belong to a class of resource. (e.g. airplane.1 is an airplane, etc.)
2.1.23	resource location. - identification of the location of a resource.	$\surd$	On pg. 23 of the cited paper, there is an example ACT that tracks resource locations.
2.1.24	resource/task combined characteristics - qualities of a resource that are dependent on a particular task, or qualities of a task that are dependent on a particular resource.	$\surd$	Multiple ACTs can be defined with different gating conditions and effects that can be used to express this requirement.
2.1.25	serial tasks	$\surd$	Serial ordering of tasks is supported.
2.1.26	state existence constraints	$\surd$	The test metapredicate can be used to evaluate state existence.
2.1.27	state representations - the description of a process in terms of any combination of the states of the process and/or resource.	$\surd$	State representations are central to the ACT representation.
2.1.28	temporal constraints	$\surd$	A rich set of temporal constraints can be used to cover all 13 relations.
2.1.29	uncertainty/variability/tolerance - the representation of the deviation from the nominal.	*	There are many ways that ACTs can express tolerance or variability of values. (For example, you can define an earliest/latest starting time, etc.)

B.1 - SRI's ACT

	PSL Requirement	Rank	Description
2.2	*Functional Requirements*		
2.2.1	ability to insert or attach a highlight (milestones) - the ability for a user to highlight a section of the process	*	This can be roughly approximated by adding property/value entries that are user-defined as milestones.
2.2.2	complex precedence - the ability to convey a series of tasks' ordering requirements within a given process.	$\surd$	ACT provides a rich set of gating conditions.
2.2.3	convey the ancestry or class of a task - the ability to describe a task as it relates to the specialization of another, higher-level task.	$\surd$	An ACT's plot is essentially a specialization of the overall task.
2.2.4	deadline management - the ability to consider a predetermined deadline when making decisions.	$\surd$	Time windows can express a variety of deadlines (e.g. x must happen before time1, etc.)
2.2.5	dispatching - the determination and representation of rules and guidelines to decide when items should be released for production.	$\surd$	There isn't an explicit mechanism that is designed for this purpose, but looping, rebinding of variables, and a test metapredicate should be sufficient.
2.2.6	eligible resources - the ability to determine which resources can be chosen for a task (selection rules)	$\surd$	The same mechanism used to describe location of a resource can be used to create custom eligibility needs.
2.2.7	exception handling and recovery - the ability to specify corrective action when a task fails.	$\surd$	This is a central concern for PRS (which uses ACT). Conditional actions provide means to describe recovery procedures.
2.2.8	information exchange between tasks - the ability to represent the flow of information among tasks	$\surd$	Information is exchanged via variable bindings.
2.2.9	mathematical and logical operations - the language must be able to perform mathematical and logical operations.	$\surd$	ACT supports FOL as its representation system.
2.2.10	support for task/process templates - the language must allow for templates of a task or process.	$\surd$	ACTs are essentially a process templates become further detailed by other ACTs.
2.2.11	support for simultaneously maintained associations of multiple levels of abstraction - the ability to associate information at multiple levels of abstraction with a task.	$\surd$	Information can be associated with an ACT that is appropriate for that ACT's relative level in the process representation.
2.2.12	synchronization of multiple, parallel task sequences - the ability to specify a mechanism to coordinate two or more tasks that occur at the same time.	$\surd$	Parallel nodes in ACT plots' serve to synch parallel task sequences where necessary.

B.1 - SRI's ACT

B.2 Core Plan Representation(CPR) Detailed PSL Analysis

Key: $\surd$ - Completely satisfies req., * - Partially satisfies req., X - Cannot satisfy req., ? - Uncertain

	PSL Requirement	Rank	Description
1	**Core Requirements**		
1.1	*Representational Requirements*		
1.1.1	ad hoc notes and annotations optionally associated with any component of a plan - on-the-fly, off-the-cuff notes and documentation. This could be voice, video, as well as text. A person's observation of a process might go here.	$\surd$	Annotation object is contained in PlanObject superclass
1.1.2	cost data - the cost associated with a resource or task. This could be a fixed cost, cost rate, or a cost derived from other attributes such as duration and level of effort. Costs associated with uncertainty, variability, tolerances, etc	X	No comment
1.1.3	level of effort - description of the amount of a resource needed, in any given unit, to accomplish a task. Some example levels of effort are equipment-hour, labor-hour, and crew size.	$\surd$	Contained in the CPR specialization objects of ConsumableResource
1.1.4	product (work item) characteristics- information about an intermediate and final product which a process will produce.	*	Work products can been given as the underspecified object DomainObject.
1.1.5	resource - a single resource or a group of resources. Some types of resources are equipment, people, information, and in-progress goods.	$\surd$	Resource object or its specializations
1.1.6	resource requirement(s) for a task - the relationship between one or more resources and a task.	$\surd$	Action objects (tasks) may contain Resource objects
1.1.7	simple groups of tasks - very basic, high-level set of tasks. One example is the grouping of tasks and sequences that make up a process plan or that make up a phase.	$\surd$	Actions may contain sub-Actions
1.1.8	simple resource capability/characteristics - a high-level description of the characteristics of a resource. More detailed descriptions can be found in the outer core.	*	A suggested set of specializations to Resource is provided including Consumable, Reusable, SynchronouslyReusable, ExactCapacity and NonSharable.

B.2 - Core Plan Representation (CPR)

	PSL Requirement	**Rank**	**Description**
1.1.9	simple sequences - linear, time-sequential groups of tasks. More sophisticated relationships such as parallel and alternative tasks can be found in the outer core.	$\surd$	Constraints may be assigned to Actions which enfore parallelism or serialism.
1.1.10	simple task representation and characteristics - a simple, high-level description of the task. More detailed representations can be found in the outer core.	X	No comment
1.1.11	task duration - the time required to complete a task or group of tasks. Only simple durations are represented here.	$\surd$	Actions have start and end times
1.1.12	task executor - who is responsible for executing a task or group of tasks. Examples include a person, controller, or external company if the task is contracted out.	$\surd$	Actions have associated Actors
1.2	*Functional Requirements*		
1.2.1	extensibility - there must be a mechanism in place to allow a user to add additional information to the pre-defined data constructs. One such mechanism could be the addition of stubs for user-defined information.	X	CPR is a representation only. However, any object may be extended through inheritance.
1.2.2	resource allocation/deallocation for one or many tasks - the assignment and release of one or more resources to a task of group of tasks.	X	No comment
1.2.3	simple precedence - a high-level description of the precedence constraints of one task on another. A more detailed description of precedence and constraints can be found in the outer core.	X	CPR is only a representation. However, precedence constraints on Actions can be specified.
2	**Outer Core Requirements**		
2.1	*Representational Requirements*		
2.1.1	abstraction - within the scope of this project, there are three concepts of abstraction that must be captured. (hierarchy, incompleteness, ambiguity)	$\surd$	There is no implied enforcement of completeness. Uncertainty and Imprecision (fuzzy logic) constructs are included.
2.1.2	alternative task - (see complex sequences)	*	Actions can be given aribtrary constraints but there is no specified construct to describe one as an alternative to another.
2.1.3	associated illustrations and drawings	$\surd$	Arbitrary Annotations may be linked to any plan object

B.2 - Core Plan Representation (CPR)

	PSL Requirement	**Rank**	**Description**
2.1.4	complex groups of tasks - groups of tasks which have a common tie.	X	No comment
2.1.5	complex resource characteristics - a detailed description of the characteristics of a resource or group of resources.	*	A hierarchy of resource types is provided
2.1.6	complex sequences - complex ordering relationships between tasks	*	Arbitary types of constraints may be given to specify parallelism or serialism
2.1.7	complex task representation and parameters - a detailed representation of a task or group of tasks.	X	No comment
2.1.8	concurrent tasks - (see complex sequences)	$\surd$	Actions may be constrained to run concurrently or may be unconstrained allowing concurrent execution if possible.
2.1.9	conditional tasks - a task that only needs to be performed under some predefined circumstance.	$\surd$	Actions may have constraints on execution. Assumptions may also be included which trigger new Actions if the assumptions are violated.
2.1.10	confidence levels - a measure of certainty that some attribute is true.	$\surd$	All low level data may be tagged with Uncertainty or Imprecision measures. High level objects like Entity or Action may be encapsulated in an UncertainEntity object which has an associated uncertainty or imprecision
2.1.11	constraints - implicit or explicit constraints associated with a task or resource.	*	Examples are given for temporal and pre- and post-condition constraints but the Constraint object is relatively underspecified.
2.1.12	date(s) and time(s) and/or multiple duration(s) - the association of one or more dates and times and/or multiple durations with a resource or task	*	CPR includes TemporalPoint a specialization of which is the OMG universal time object which has both time and date. Action may be specialized to contain other durations but the base class only contains start and end.
2.1.13	implicit/explicit resource association - an implicit or explicit dependency of a resource on another type of resource.	*	A Resource may contain a Constraint which specified dependency on another Resource.
2.1.14	iterative loops - a situation when a task or group of tasks repeats until a desired condition is met	X	No comment
2.1.15	manual vs. automated tasks - characteristics of a task can differ depending on if a human or a machine is performing that task.	X	No comment
2.1.16	manufacturing product quantity - the amount of the product that is to manufactured.	*	DomainObjects with associated quantity may be specified as products of Actions.

B.2 - Core Plan Representation (CPR)

	PSL Requirement	**Rank**	**Description**
2.1.17	material constraints	*	Constraints may state ranges about arbitrary attibutes of an Entity.
2.1.18	parallel tasks	*	Arbitary types of constraints may be given to specify parallelism or serialism.
2.1.19	parameters and variables - place holders that can store a constantly changing value.	X	No comment
2.1.20	pre- and post-processing constraints	*	Examples are given for temporal and pre- and post-condition constraints but the Constraint object is relatively underspecified.
2.1.21	queues, stacks, lists - the representation of an ordered or unordered group.	X	No comment
2.1.22	resource categorization and grouping - a logical grouping of resources with a common tie.	X	Resources may have subResources but only hierarchical arrangements are currently allowed
2.1.23	resource location. - identification of the location of a resource.	$\surd$	Resources may be constrained to have a particular SpatialPoint
2.1.24	resource/task combined characteristics - qualities of a resource that are dependent on a particular task, or qualities of a task that are dependent on a particular resource.	X	No comment
2.1.25	serial tasks	*	Arbitary types of constraints may be given to specify parallelism or serialism.
2.1.26	state existence constraints	X	No comment
2.1.27	state representations - the description of a process in terms of any combination of the states of the process and/or resource.	X	Cannot.
2.1.28	temporal constraints	$\surd$	Actions have associated TimePoints which constrain their execution
2.1.29	uncertainty/variability/tolerance - the representation of the deviation from the nominal.	$\surd$	Uncertainty and Imprecision (fuzzy logic) constructs are included and may be specified for any object including TimePoints.
2.2	*Functional Requirements*		
2.2.1	ability to insert or attach a highlight (milestones) - the ability for a user to highlight a section of the process	X	No comment
2.2.2	complex precedence - the ability to convey a series of tasks' ordering requirements within a given process.	X	No comment
2.2.3	convey the ancestry or class of a task - the ability to describe a task as it relates to the specialization of another, higher-level task.	X	No comment

B.2 - Core Plan Representation (CPR)

	PSL Requirement	**Rank**	**Description**
2.2.4	deadline management - the ability to consider a predetermined deadline when making decisions.	X	No comment
2.2.5	dispatching - the determination and representation of rules and guidelines to decide when items should be released for production.	X	No comment
2.2.6	eligible resources - the ability to determine which resources can be chosen for a task (selection rules)	X	No comment
2.2.7	exception handling and recovery - the ability to specify corrective action when a task fails.	X	No comment
2.2.8	information exchange between tasks - the ability to represent the flow of information among tasks	X	No comment
2.2.9	mathematical and logical operations - the language must be able to perform mathematical and logical operations.	X	No comment
2.2.10	support for task/process templates - the language must allow for templates of a task or process.	X	No comment
2.2.11	support for simultaneously maintained associations of multiple levels of abstraction - the ability to associate information at multiple levels of abstraction with a task.	X	No comment
2.2.12	synchronization of multiple, parallel task sequences - the ability to specify a mechanism to coordinate two or more tasks that occur at the same time.	X	No comment

B.2 - Core Plan Representation (CPR)

B.3 <I-N-OVA> Detailed PSL Analysis

Key: $\surd$ - Completely satisfies req., * - Partially satisfies req., X - Cannot satisfy req., ? - Uncertain

	PSL Requirement	Rank	Description
1	**Core Requirements**		
1.1	*Representational Requirements*		
1.1.1	ad hoc notes and annotations optionally associated with any component of a plan - on-the-fly, off-the-cuff notes and documentation. This could be voice, video, as well as text. A person's observation of a process might go here.	$\surd$	A - Misc-Annotation constraint.
1.1.2	cost data - the cost associated with a resource or task. This could be a fixed cost, cost rate, or a cost derived from other attributes such as duration and level of effort. Costs associated with uncertainty, variability, tolerances, etc	$\surd$	A - Misc constraint - in global <I-N-OVA> model if not specific to a given process or plan, or in plan's <I-N-OVA> representation if it is specific to that.
1.1.3	level of effort - description of the amount of a resource needed, in any given unit, to accomplish a task. Some example levels of effort are equipment-hour, labor-hour, and crew size.	$\surd$	A - Resource (or A-Resource-Agent) constraint.
1.1.4	product (work item) characteristics - information about an intermediate and final product which a process will produce.	$\surd$	V - entity/variable constraint.
1.1.5	resource - a single resource or a group of resources. Some types of resources are equipment, people, information, and in-progress goods.	$\surd$	A - object used in resource constraint.
1.1.6	resource requirement(s) for a task (with quantity) - the relationship between one or more resources and a task.	$\surd$	A - Resource constraint
1.1.7	simple groups of tasks - very basic, high-level set of tasks. One example is the grouping of tasks and sequences that make up a process plan or that make up a phase.	$\surd$	N - include activity constraint.
1.1.8	simple resource capability/characteristics - a high-level description of the characteristics of a resource. More detailed descriptions can be found in the outer core.	$\surd$	V - global <I-N-OVA> entity/variable constraint for object to be used as a resource.
	B.3 - <I-N-OVA>		

	PSL Requirement	**Rank**	**Description**
1.1.9	simple sequences - linear, time-sequential groups of tasks. More sophisticated relationships such as parallel and alternative tasks can be found in the outer core.	$\surd$	O - Ordering constraint on time point associated with begin or end of any activity.
1.1.10	simple task representation and characteristics - a simple, high-level description of the task. More detailed representations can be found in the outer core.	$\surd$	N - Name of activity.
1.1.11	task duration - the time required to complete a task or group of tasks. Only simple durations are represented here.	$\surd$	O - Metric temporal constraint between time points associated with begin and end of an activity.
1.1.12	task executor - who is responsible for executing a task or group of tasks. Examples include a person, controller, or external company if the task is contracted out.	$\surd$	A - Resource-Agent constraint. This allows for a specific "performer" of an activity.
1.2	*Functional Requirements*		
1.2.1	extensibility - there must be a mechanism in place to allow a user to add additional information to the pre-defined data constructs. One such mechanism could be the addition of stubs for user-defined information.	$\surd$	A - Open framework for adding any information in the form of a constraint or annotation.
1.2.2	resource allocation/deallocation for one or many tasks - the assignment and release of one or more resources to a task of group of tasks.	$\surd$	A - resource constraints should be expressive enough to support this.
1.2.3	simple precedence - a high-level description of the precedence constraints of one task on another. A more detailed description of precedence and constraints can be found in the outer core.	$\surd$	O - Ordering constraints. For example: O-Plan Task Formalism allows A —> B implying time point at end of activity A is before time point at begin of activity B. O-Plan TF also allows SEQUENCE A, B, C, ... to A—>B, B—>C, C —> ...
2	**Outer Core Requirements**		
2.1	*Representational Requirements*		
B.3 - <I-N-OVA>			

	PSL Requirement	**Rank**	**Description**
2.1.1	abstraction - within the scope of this project, there are three concepts of abstraction that must be captured. (hierarchy, incompleteness, ambiguity)	$\surd$	A,N - Constraints of various types (in particular A-World State constraints) may be modelled at any abstraction level. Activity decompositons (Include activity constraints in process or activity description library) (N). Missing constraints just imply a wider allowed space of behaviour. The <I-N-OVA> model is specifically designed to allow for incompleteness and uncertainty in process and activity descriptions. The <I-N-OVA> model is specifically designed to allow for incompleteness and uncertainty in process and activity descriptions. Specific constraints would need to have uncertainty in their formulation and expression <I-N-OVA> makes no commitment to this.
2.1.2	alternative task - (see complex sequences)	$\surd$	disjunctive constraints may be included in the <I-N-OVA> model in any place - and this is not limited to disjunctions within any one specific constraint type or sub-type. An other node can also represent conditional activities.
2.1.3	associated illustrations and drawings	$\surd$	A - Associated information and annotations may be states as "annotation constraints" or more generally "Miscellaneous constraints".
2.1.4	complex groups of tasks - groups of tasks which have a common tie.	$\surd$	N - other nodes that contain sub-plans can be used to group a task for a common purpose (I.e. the detailed expression of an activity).
2.1.5	complex resource characteristics - a detailed description of the characteristics of a resource or group of resources.	$\surd$	A - Resource constraints or Agent constraints can describe these characteristics.
2.1.6	complex sequences - complex ordering relationships between tasks	$\surd$	O - ordering constraints can describe a variety of necessary relationships.
2.1.7	complex task representation and parameters - a detailed representation of a task or group of tasks.	$\surd$	N - Nodes that include activities can take into account concepts such as applicability, performance limits,resource usage, number of constraints on its conditions, suitable parameter bindings, etc.
2.1.8	concurrent tasks - (see complex sequences)	$\surd$	O - activities can be constrained to have "concurrent" execution.

B.3 - <I-N-OVA>

	PSL Requirement	**Rank**	**Description**
2.1.9	conditional tasks - a task that only needs to be performed under some predefined circumstance.	$\surd$	N - other nodes may also represent a conditional "if then else" within the plan.
2.1.10	confidence levels - a measure of certainty that some attribute is true.	X	<I-N-OVA> does not contain a mechanism for probablilistic certainty.
2.1.11	constraints - implicit or explicit constraints associated with a task or resource.	$\surd$	<I-N-OVA> views a plan as a set of constraints.
2.1.12	date(s) and time(s) and/or multiple duration(s) - the association of one or more dates and times and/or multiple durations with a resource or task	$\surd$	O - metric temporal constraints can relate a given time point to an actual time or calendar reference.
2.1.13	implicit/explicit resource association - an implicit or explicit dependency of a resource on another type of resource.	$\surd$	A,N - Resource constraints can explicitly be attached to an activity. A node that contains sub-plans implicity constrains resource usage though its sub-constraints.
2.1.14	iterative loops - a situation when a task or group of tasks repeats until a desired condition is met	$\surd$	N - other nodes can represent an encapsulation of iteration or for-each.
2.1.15	manual vs. automated tasks - characteristics of a task can differ depending on if a human or a machine is performing that task.	$\surd$	A - Misc contraints can be created to characterize specialized attribute requirements.
2.1.16	manufacturing product quantity - the amount of the product that is to manufactured.	$\surd$	A - Resource constraints can be used to control the maximum allowable amount of the resource.
2.1.17	material constraints	$\surd$	A - resource constraints can be used to describe specialized characteristics. "always" constraints can be used to declare unchanging global information.
2.1.18	parallel tasks	$\surd$	O - ordering constraints can describe activities that occur in parallel.
2.1.19	parameters and variables - place holders that can store a constantly changing value.	$\surd$	V - entity/variable constraints can be used to manage "place holders" that can take on a range of values.
2.1.20	pre- and post-processing constraints	$\surd$	O - input and output temporal constraints are used to describe what should hold immediately before or after a given timepoint.
2.1.21	queues, stacks, lists - the representation of an ordered or unordered group.	*	<I-N-OVA> does not have an explicit representation for data structures such as queues or stacks.
2.1.22	resource categorization and grouping - a logical grouping of resources with a common tie.	$\surd$	It is anticipated that a representation language that expresses the <I-N-OVA> model will use a sorted first order logic.

B.3 - <I-N-OVA>

	PSL Requirement	**Rank**	**Description**
2.1.23	resource location. - identification of the location of a resource.	√	A, V - A-Resource constraints can add information such as location, entity/variables can be used to update a location attribute.
2.1.24	resource/task combined characteristics - qualities of a resource that are dependent on a particular task, or qualities of a task that are dependent on a particular resource.	√	O,N - This requirement can be met by creating alternate "include activity" nodes that utilize the same resources, but may have different input temporal constraints.
2.1.25	serial tasks	√	O - ordering constraints are used to declare activites in serial.
2.1.26	state existence constraints	√	O - input temporal constraints specify those things that are required to hold before a given timepoint (which may be attached to an activity).
2.1.27	state representations - the description of a process in terms of any combination of the states of the process and/or resource.	√	A - World State constraints act on the plan state representation.
2.1.28	temporal constraints	√	O - Temporal modelling is performed by using timepoints and ordering constraints.
2.1.29	uncertainty/variability/tolerance - the representation of the deviation from the nominal.	√	The <I-N-OVA> model is specifically designed to allow for incompleteness and uncertainty in process and activity descriptions. Specific constraints would need to have uncertainty in their formulation and expression <I-N-OVA> makes no commitment to this.
2.2	*Functional Requirements*		
2.2.1	ability to insert or attach a highlight (milestones) - the ability for a user to highlight a section of the process	*	A - Misc or Annotation constraints can be attached to nodes to give them "milestone significance".
2.2.2	complex precedence - the ability to convey a series of tasks' ordering requirements within a given process.	√	O - Ordering constraints can be generally specified to establish node precedence.
2.2.3	convey the ancestry or class of a task - the ability to describe a task as it relates to the specialization of another, higher-level task.	√	N - other node constraints can be used to encapsulate specialized sub-plans.
2.2.4	deadline management - the ability to consider a predetermined deadline when making decisions.	√	O - Ordering constraints are used to arrange activities within specified temporal constraints.
2.2.5	dispatching - the determination and representation of rules and guidelines to decide when items should be released for production.	√	O - Input temporal constaints can be placed on activities that release represent releasing items for production.

B.3 - <I-N-OVA>

	PSL Requirement	Rank	Description
2.2.6	eligible resources - the ability to determine which resources can be chosen for a task (selection rules)	$\surd$	A - Resource constraints for an activity describe a sorted requirement for resource usage.
2.2.7	exception handling and recovery - the ability to specify corrective action when a task fails.	$\surd$	O - input and output temporal constraints can be used to specify what should hold before and after a timepoint (therefore an activity).
2.2.8	information exchange between tasks - the ability to represent the flow of information among tasks	$\surd$	V - Information is shared between nodes thru entity/variable constraints.
2.2.9	mathematical and logical operations - the language must be able to perform mathematical and logical operations.	$\surd$	The expressions in <I-N-OVA> are considered to be based in first order logic that will allow for logical and mathematical manipulation.
2.2.10	support for task/process templates - the language must allow for templates of a task or process.	$\surd$	N - other nodes and include activity nodes are linked in a "generic process template" that is applicable for use assuming the constraints are satisfied.
2.2.11	support for simultaneously maintained associations of multiple levels of abstraction - the ability to associate information at multiple levels of abstraction with a task.	$\surd$	A - Constraints can be attached at any level of a node hierarchy that would be appropriate for that model.
2.2.12	synchronization of multiple, parallel task sequences - the ability to specify a mechanism to coordinate two or more tasks that occur at the same time.	$\surd$	O - Temporal constraints can be attached to activities that make the hard requirement that begin/end timepoints are equal.
B.3 - <I-N-OVA>			

B.4 OZONE Scheduling Ontology Detailed PSL Analysis

Key: √ - Completely satisfies req., * - Partially satisfies req., X - Cannot satisfy req., ? - Uncertain

	PSL Requirement	**Rank**	**Description**
1	**Core Requirements**		
1.1	*Representational Requirements*		
1.1.1	ad hoc notes and annotations optionally associated with any component of a plan - on-the-fly, off-the-cuff notes and documentation. This could be voice, video, as well as text. A person's observation of a process might go here.	?	Need more information.
1.1.2	cost data - the cost associated with a resource or task. This could be a fixed cost, cost rate, or a cost derived from other attributes such as duration and level of effort. Costs associated with uncertainty, variability, tolerances, etc	*	There is no explicit cost property for resources or tasks in OZONE, but some aspects ofcost can be treated as a property that is a function of the domain (i.e. the same was as LAND or SPEED are noted in the paper).
1.1.3	level of effort - description of the amount of a resource needed, in any given unit, to accomplish a task. Some example levels of effort are equipment-hour, labor-hour, and crew size.	√	Demands can be defined that explicitly represent the quantity required. Activity RESOURCE-REQUIREMENTS impose resource usage/comsumption constraints for the activity to execute.
1.1.4	product (work item) characteristics- information about an intermediate and final product which a process will produce.	√	OZONE uses a distinct concept definition for a product. Intermediate product information and work item characteristics can be attached directly to a product.
1.1.5	resource - a single resource or a group of resources. Some types of resources are equipment, people, information, and in-progress goods.	√	A resource is a distinct concept definition in OZONE. A variety of resource types are supported.
1.1.6	resource requirement(s) for a task - the relationship between one or more resources and a task.	√	An activity can be defined with relationships to resources that it requires.
1.1.7	simple groups of tasks - very basic, high-level set of tasks. One example is the grouping of tasks and sequences that make up a process plan or that make up a phase.	√	OZONE supports the grouping of tasks in a variety of ways. Tasks (activities) can be grouped into those that fulfill a demand, produce a product, or are involved in a hierarchical ordering.
1.1.8	simple resource capability/characteristics - a high-level description of the characteristics of a resource. More detailed descriptions can be found in the outer core.	√	A variety of capabilites/characteristics can be assigned to a resource via properties. (e.g. capacity, amount of set-up time needed, etc.)

B.4 - OZONE Scheduling Ontology

	PSL Requirement	**Rank**	**Description**
1.1.9	simple sequences - linear, time-sequential groups of tasks. More sophisticated relationships such as parallel and alternative tasks can be found in the outer core.	$\surd$	OZONE contains INTERVAL-RELATIONS that can easily handle simple linear sequencing.
1.1.10	simple task representation and characteristics - a simple, high-level description of the task. More detailed representations can be found in the outer core.	?	Need more information.
1.1.11	task duration - the time required to complete a task or group of tasks. Only simple durations are represented here.	$\surd$	OZONE activities contain a "duration" property for this purpose.
1.1.12	task executor - who is responsible for executing a task or group of tasks. Examples include a person, controller, or external company if the task is contracted out.	$\surd$	A task executor can be modelled as a required resource for the activity.
1.2	*Functional Requirements*		
1.2.1	extensibility - there must be a mechanism in place to allow a user to add additional information to the pre-defined data constructs. One such mechanism could be the addition of stubs for user-defined information.	$\surd$	OZONE puts forward a concept of model specialization. Elements can be added that specialize the representation for a target domain.
1.2.2	resource allocation/deallocation for one or many tasks - the assignment and release of one or more resources to a task of group of tasks.	$\surd$	Resources provide Allocate-Capacity and Deallocate-Capacity capabilities and Activities provide reserve-resources and free-resources capabilities.
1.2.3	simple precedence - a high-level description of the precedence constraints of one task on another. A more detailed description of precedence and constraints can be found in the outer core.	$\surd$	Various constraints can be defined to regulate precedence relationships of activities.
2	**Outer Core Requirements**		
2.1	*Representational Requirements*		
2.1.1	abstraction - within the scope of this project, there are three concepts of abstraction that must be captured. (hierarchy, incompleteness, ambiguity)	*	To a degree, it can be stated that a constraint-based approach permits a model to be incomplete and vague on everything, except those items that are necessary to meet requirements. (e.g. Schedule these tasks in any order you like, but just make sure C is after B, etc.) What is described in the requirement though is more of a runtime test condition.

B.4 - OZONE Scheduling Ontology

	PSL Requirement	**Rank**	**Description**
2.1.2	alternative task - (see complex sequences)	$\surd$	Two activities can be defined that have the same effects. The scheduler can then select an alternative that satisfies the requirement.
2.1.3	associated illustrations and drawings	?	Need more information.
2.1.4	complex groups of tasks - groups of tasks which have a common tie.	$\surd$	OZONE supports complex grouping of tasks. For example, a set of tasks can be grouped that meet the requirements for a specific demand, a set of tasks that produce a work item can be attached to the specific product as well.
2.1.5	complex resource characteristics - a detailed description of the characteristics of a resource or group of resources.	$\surd$	OZONE provides a variety of ways to assign characteristics to resources. For example: associating state information with a resource, physical properties (range, speed), capacity models, etc.
2.1.6	complex sequences - complex ordering relationships between tasks	$\surd$	Complex ordering relationships can be defined via INTERVAL-RELATIONS. (e.g. BEFORE, SAME-END, CONTAINS, etc.)
2.1.7	complex task representation and parameters - a detailed representation of a task or group of tasks.	$\surd$	An activity can be defined with a complex set of properties. OZONE activities support an explicit set of pararmeters that can influence the representation of the task.
2.1.8	concurrent tasks - (see complex sequences)	$\surd$	An activity can contain temporal relationships to other activities. If a relationship of same-start and same-end is defined then the two activities are constrained to be concurrent.
2.1.9	conditional tasks - a task that only needs to be performed under some predefined circumstance.	*	At a high level, we can say that an activity is conditional because its execution is dependent on outstanding demands. However, there does not seem to be an explicit conditional structure.
2.1.10	confidence levels - a measure of certainty that some attribute is true.	X	No support for a measure of certainty.
2.1.11	constraints - implicit or explicit constraints associated with a task or resource.	$\surd$	OZONE presumes an underlying constraint-based solution framework.

B.4 - OZONE Scheduling Ontology

	PSL Requirement	**Rank**	**Description**
2.1.12	date(s) and time(s) and/or multiple duration(s) - the association of one or more dates and times and/or multiple durations with a resource or task	*	OZONE uses various date/time relationships and assumes the existence of TIME-POINTS, and TIME-INTERVALS. While there is support for multiple durations (e.g. duration of an activity, duration of setup-time for a resource, etc.), a specific requirement of multiple durations for the overall activity does not seem possible.
2.1.13	implicit/explicit resource association - an implicit or explicit dependency of a resource on another type of resource.	$\surd$	Various levels of implicit/explicit resource associations can be made (i.e. sub-resources for aggregation, dynamic compatibility between 2 resource assignments, etc.)
2.1.14	iterative loops - a situation when a task or group of tasks repeats until a desired condition is met	X	OZONE does not appear to support iteration or looping constructs.
2.1.15	manual vs. automated tasks - characteristics of a task can differ depending on if a human or a machine is performing that task.	$\surd$	OZONE does not make an explicit distininction of this type, but it would seem possible to create two activities, one that represented the manual task and one that represented the automatic task and any "differing" would be defined by each respective activity.
2.1.16	manufacturing product quantity - the amount of the product that is to manufactured.	$\surd$	An explicit slot for specifying product quantity is part of a demand in OZONE.
2.1.17	material constraints	$\surd$	OZONE has an explicit slot for material constraints as part of a demand (i.e. the type of material to be used).
2.1.18	parallel tasks	$\surd$	Nodes in OZONE's networks of activities can be ordered in parallel.
2.1.19	parameters and variables - place holders that can store a constantly changing value.	$\surd$	The OZONE ontology has parameters (e.g. an activity accepts a quantity from demand) and variables (e.g. recording changes in state).
2.1.20	pre- and post-processing constraints	$\surd$	A variety of pre and post processing constraints apply to activities. (e.g. (pre) state existence (post) duration before next activity, etc.)
2.1.21	queues, stacks, lists - the representation of an ordered or unordered group.	*	Lists of elements only.
2.1.22	resource categorization and grouping - a logical grouping of resources with a common tie.	$\surd$	OZONE supports a rich set of categories and groupings of resources based on their usages, atomicity, capacity, etc.

B.4 - OZONE Scheduling Ontology

	PSL Requirement	**Rank**	**Description**
2.1.23	resource location. - identification of the location of a resource.	$\surd$	OZONE has an explicit slot in a demand for the ORIGIN and DESTINATION for a material.
2.1.24	resource/task combined characteristics - qualities of a resource that are dependent on a particular task, or qualities of a task that are dependent on a particular resource.	$\surd$	Combined activity/resource characteristics are utilized in evaluating static and dynamic compatibility constraints.
2.1.25	serial tasks	$\surd$	Simple serial assignment falls under a “before” interval.
2.1.26	state existence constraints	$\surd$	Activities and resources can be in a given state and requirements about state existence can be applied.
2.1.27	state representations - the description of a process in terms of any combination of the states of the process and/or resource.	$\surd$	Activities and resources can be represented as being in certain states.
2.1.28	temporal constraints	$\surd$	A variety of constraints: absolute-time-constraint, relative-time-constraint (interval-relations, duration-constraints)
2.1.29	uncertainty/variability/tolerance - the representation of the deviation from the nominal.	*	Various upper/lower bounded values support variability and tolerance of assignment values, but probabilistic uncertainty is not supported.
2.2	*Functional Requirements*		
2.2.1	ability to insert or attach a highlight (milestones) - the ability for a user to highlight a section of the process	X	No support for this.
2.2.2	complex precedence - the ability to convey a series of tasks’ ordering requirements within a given process.	$\surd$	Duration-Constraints, interval-relations, state requirements, and aspects of demand management all combine to provide complex precedence mechanisms.
2.2.3	convey the ancestry or class of a task - the ability to describe a task as it relates to the specialization of another, higher-level task.	$\surd$	Class ancestry in OZONE is expressed thru its extension mechanism of model specialization.
2.2.4	deadline management - the ability to consider a predetermined deadline when making decisions.	$\surd$	Deadline management is possible via RELEASE-DATE, DUE-DATE properties of a demand.
2.2.5	dispatching - the determination and representation of rules and guidelines to decide when items should be released for production.	$\surd$	Dispatching is encompassed in the demand-product combined capabilities. Work item generation is linked to explicit elements of demand.

B.4 - OZONE Scheduling Ontology

	PSL Requirement	**Rank**	**Description**
2.2.6	eligible resources - the ability to determine which resources can be chosen for a task (selection rules)	$\surd$	OZONE maintains the “eligibility” of resources and also provides other USAGE-RESTRICTIONS that can allow a richer model of restrictions (e.g. UNAVAILABILITY-INTERVALS reflect time periods where a resource is not eligible, etc.)
2.2.7	exception handling and recovery - the ability to specify corrective action when a task fails.	X	No support for this.
2.2.8	information exchange between tasks - the ability to represent the flow of information among tasks	$\surd$	OZONE supports paramaters passing to exchange information between various elements (e.g. demand information is passed to an activity, etc.)
2.2.9	mathematical and logical operations - the language must be able to perform mathematical and logical operations.	$\surd$	Constraint expressions use mathematical and logical constructs in OZONE.
2.2.10	support for task/process templates - the language must allow for templates of a task or process.	$\surd$	The ontological element “activity” is a template for what a task should be. The various properties are expected to be filled in and new slots can be added to extend this base concept.
2.2.11	support for simultaneously maintained associations of multiple levels of abstraction - the ability to associate information at multiple levels of abstraction with a task.	$\surd$	Constraints can be added at any level of abstraction to further define the requirements on the target space. In the example listed, you would require 5 people (resources). Next you may add a constraint on those resources (special ability). Next you may add a very specific contraint (who they are), etc.
2.2.12	synchronization of multiple, parallel task sequences - the ability to specify a mechanism to coordinate two or more tasks that occur at the same time.	$\surd$	Multiple activities can be synchronized when parallel via INTERVAL-RELATIONS.

B.4 - OZONE Scheduling Ontology

B.5 PIF (v.1.1) Detailed PSL Analysis

Key: √ - Completely satisfies req., * - Partially satisfies req., X - Cannot satisfy req., ? - Uncertain

	PSL Requirement	**Rank**	**Description**
1	**Core Requirements**		
1.1	*Representational Requirements*		
1.1.1	ad hoc notes and annotations optionally associated with any component of a plan - on-the-fly, off-the-cuff notes and documentation. This could be voice, video, as well as text. A person's observation of a process might go here.	*	This requirement is partially filled through the use of the "documentation" slot and the "user-attributes" that can be attached to PIF entities.
1.1.2	cost data - the cost associated with a resource or task. This could be a fixed cost, cost rate, or a cost derived from other attributes such as duration and level of effort. Costs associated with uncertainty, variability, tolerances, etc.	X	PIF does not address resource or task cost.
1.1.3	level of effort - description of the amount of a resource needed, in any given unit, to accomplish a task. Some example levels of effort are equipment-hour, labor-hour, and crew size.	X	PIF can say that an activity uses some object, but does not have a mechanism to quantify the usage.
1.1.4	product (work item) characteristics- information about an intermediate and final product which a process will produce.	*	While PIF can represent objects that are created, modified, or used during an activity, there is no provision for attaching characteristics to that object. 2-Dec-96, As per Lee: should be * (partial) because the PIF-CORE provides a means of specifying such characteristics via User-Attributes and PSV's.
1.1.5	resource - a single resource or a group of resources. Some types of resources are equipment, people, information, and in-progress goods.	√	Activities can specify which objects (resources) were created, modified or used
1.1.6	resource requirement(s) for a task (with quantity) - the relationship between one or more resources and a task.	*	PIF cannot represent quantity of an object (resource).
1.1.7	simple groups of tasks - very basic, high-level set of tasks. One example is the grouping of tasks and sequences that make up a process plan or that make up a phase.	√	PIF can express grouping of activities through decompositional relationships. 2-Dec-96, Gruninger: satisfied if the grouping is a deterministic activity, but is not satisfied in general for non-deterministic activities

B.5 - PIF (v1.1)

	PSL Requirement	**Rank**	**Description**
1.1.8	simple resource capability or characteristics - a high-level description of the characteristics of a resource. More detailed descriptions can be found in the outer core.	*	16-Nov-96, X to * as per Tate's input: PIF Core does not have an explicit mechanism to describe the "capability" of an object (when used in the role of resource) but it does allow for attributes of such objects to be stated. Capability limited to agents.
1.1.9	simple sequences - linear, time-sequential groups of tasks. More sophisticated relationships such as parallel and alternative tasks can be found in the outer core.	$\surd$	The use of timepoints and temporal relationships will provide simple sequences. 2-Dec-96, Gruninger: This requirement is satisfied insofar as we can write the definition of an activity for simple and complex sequences. However, we cannot express the definition of simple and complex sequences using PIF-Core.
1.1.10	simple task representation and characteristics - a simple, high-level description of the task. More detailed representations can be found in the outer core.	$\surd$	PIF can represent a task with its effects, conditions, etc. Also, textual high-level descriptions can be attached via the documentation attribute.
1.1.11	task duration - the time required to complete a task or group of tasks. Only simple durations are represented here.	$\surd$	16-Nov-96, Change from $\surd$ to * as per Tate's input: An activity can contain begin and end points, but PIF Core itself does not support quantities for duration. 4-Dec-96, Changed back to $\surd$ per Gruninger and Lee: This can be captured, since the axiomatization of time points in the situation calculus means that time points are isomorphic to the real numbers.
1.1.12	task executor - who is responsible for executing a task or group of tasks. Examples include a person, controller, or external company if the task is contracted out.	$\surd$	PIF can describe a "performs" relationship between activities and agents.
1.2	*Functional Requirements*		
1.2.1	extensibility - there must be a mechanism in place to allow a user to add additional information to the pre-defined data constructs. One such mechanism could be the addition of stubs for user-defined information.	$\surd$	PIF has a "user-attributes" slot defined at the highest level of the hierarchy that can store user-defined information.
	B.5 - PIF (v1.1)		

	PSL Requirement	**Rank**	**Description**
1.2.2	resource allocation/deallocation for one or many tasks - the assignment and release of one or more resources to a task of group of tasks.	*	16-Nov-96, Change from * to √ as per Tate's input that pointed out that specifying individual resource units are not part of the requirement. PIF can represent objects that are created, modified, or used during an activity. 4-Dec-96, Changed back to * as per Gruninger and Lee. The feeling is that resource exclusivity etc. is not explicit.
1.2.3	simple precedence - a high-level description of the precedence constraints of one task on another. A more detailed description of precedence and constraints can be found in the outer core.	√	PIF can provide a detailed description of activities' relationships to other activities. Temporal, causal, and decompositional relationships can be used to impose constraints on the precedence.
2	**Outer Core Requirements**		
2.1	*Representational Requirements*		
2.1.1	abstraction - within the scope of this project, there are three concepts of abstraction that must be captured.	√	PIF supports decompositional relationships between activities. Therefore activities can be arranged in an abstract, incomplete, or ambiguous fashion.
2.1.2	alternative task - (see complex sequences)	√	PIF's use of decisions allows for a selection of alternative tasks. 4-Dec-96 Gruninger: Although decisions can be used to select an activity based on state, this cannot be used to define arbitrary nondeterministic choices e.g. do A or do B.
2.1.3	associated illustrations and drawings	*	16-Nov-96, Change from X to * as per Tate's input: "[a PIF user] can use the documentation or user attribute slots for this."
2.1.4	complex groups of tasks - groups of tasks which have a common tie.	*	This requirement asks for information that goes beyond specifying which subactivities occur in a group and asks whether there is explicit representation about the overall group (total cost, total resources used in decomposition, etc.) 2-Dec-96, Gruninger: satisfied if the grouping is a deterministic activity, but is not satisfied in general for nondeterministic activities
2.1.5	complex resource characteristics - a detailed description of the characteristics of a resource or group of resources.	X	See 1.1.8.

B.5 - PIF (v1.1)

	PSL Requirement	**Rank**	**Description**
2.1.6	complex sequences - complex ordering relationships between tasks	$\surd$	PIF can handle concepts such as: alternative, parallel, serial, concurrent activities. Timepoints and temporal relationships provide these requirements. 2-Dec-96, Gruninger: This requirement is satisfied insofar as we can write the definition of an activity for simple and complex sequences. However, we cannot express the definition of simple and complex sequences using PIF-Core.
2.1.7	complex task representation and parameters - a detailed representation of a task or group of tasks.	*	PIF's highly expressive pif-sentences can be used to give a detailed representation of what an activity needs and does (hierarchical activities are considered "grouped"). More specialized charac. (e.g. uniterruptability) cannot be expressed.
2.1.8	concurrent tasks - (see complex sequences)	$\surd$	See 2.1.6.
2.1.9	conditional tasks - a task that only needs to be performed under some predefined circumstance.	$\surd$	PIF uses the entity, decision, to represent a conditional activity.
2.1.10	confidence levels - a measure of certainty that some attribute is true.	X	PIF sentences are boolean.
2.1.11	constraints - implicit or explicit constraints associated with a task or resource.	$\surd$	Activities and objects (resources) inherit constraint slots for such purposes.
2.1.12	date(s) and time(s) and/or multiple duration(s) - the association of one or more dates and times and/or multiple durations with a resource or task	*	Activities can express durations through relative begin and end timepoints.
2.1.13	implicit/explicit resource association - an implicit or explicit dependency of a resource on another type of resource.	$\surd$	PIF representation of object (resource) component can be interpreted to some extent as a dependency. (e.g. Object A has components Object B and C. Therefore using Object A implies using Object B and C as well.) 2-Dec-96, * to $\surd$, Lee: Represent [a] dependency via Precondition and Postcondition of an activity. A resource X requires another resource Y if the Use activity that uses X has as a precondition the availability of the resource Y
2.1.14	iterative loops - a situation when a task or group of tasks repeats until a desired condition is met	$\surd$	This can be satisfied using decisions or preconditions and postconditions of activities.

B.5 - PIF (v1.1)

	PSL Requirement	Rank	Description
2.1.15	manual vs. automated tasks - characteristics of a task can differ depending on if a human or a machine is performing that task.	X	No information is explicitly captured for this in the PIF-CORE.
2.1.16	manufacturing product quantity - the amount of the product that is to manufactured.	X	Not in PIF-CORE, sounds like a PSV.
2.1.17	material constraints	X	PIF contains no support for material constraints.
2.1.18	parallel tasks	$\surd$	See 2.1.6. 4-Dec-96 Gruninger: This requirement is satisfied insofar as we can write the definition of an activity for parallel or serial tasks. However, we cannot express the definition of parallel or serial tasks using PIF-Core.
2.1.19	parameters and variables - place holders that can store a constantly changing value.	$\surd$	PIF activities utilize variables in pif-sentences.
2.1.20	pre- and post-processing constraints	$\surd$	PIF declares what must be true before an activity is performed and also asserts what must be true after the activity completes.
2.1.21	queues, stacks, lists - the representation of an ordered or unordered group.	*	PIF provides a list structure.
2.1.22	resource categorization and grouping - a logical grouping of resources with a common tie.	*	To some extent, PIF can address this by explicitly listing which objects (resources) each activity uses/creates/modifies. PIF can also describe which objects are components of other objects, but logical grouping (outside of activity) seems to be absent.
2.1.23	resource location. - identification of the location of a resource.	X	There is no facility in the PIF-CORE to address this relationship.
2.1.24	resource/task combined characteristics - qualities of a resource that are dependent on a particular task, or qualities of a task that are dependent on a particular resource.	X	There is no facility in the PIF-CORE to address this relationship.
2.1.25	serial tasks	$\surd$	PIF can order activities in serial. 4-Dec-96 Gruninger: This requirement is satisfied insofar as we can write the definition of an activity for parallel or serial tasks. However, we cannot express the definition of parallel or serial tasks using PIF-Core.

B.5 - PIF (v1.1)

	PSL Requirement	Rank	Description
2.1.26	state existence constraints	$\surd$	PIF can constrain when activities can be executed using activity preconditions.
2.1.27	state representations - the description of a process in terms of any combination of the states of the process and/or resource.	*	PIF can describe state changes for activities but not for objects (resources).
2.1.28	temporal constraints	*	PIF can relate activities through shared begin/end points, but a PSV is required to appropriately address temporal relationships (other than "before"). 2-Dec-96, X to *, Lee: BEFORE is available and ... some temporal constraints can be expressed using the begin and end timepoints.
2.1.29	uncertainty/variability/tolerance - the representation of the deviation from the nominal.	X	PIF does not have a facility for managing uncertainty.
2.2	*Functional Requirements*		
2.2.1	ability to insert or attach a highlight (milestones) - the ability for a user to highlight a section of the process	X	Not in PIF-CORE
2.2.2	complex precedence - the ability to convey a series of tasks' ordering requirements within a given process.	$\surd$	The use of preconditions and decisions allows for complex, conditional activity orderings. 2-Dec-96, Gruninger: This requirement is satisfied insofar as we can write the definition of an activity for simple and complex sequences. However, we cannot express the definition of simple and complex sequences using PIF-Core.
2.2.3	convey the ancestry or class of a task - the ability to describe a task as it relates to the specialization of another, higher-level task.	$\surd$	PIF's decompositional relationships define a hierarchy of specialization.
2.2.4	deadline management - the ability to consider a predetermined deadline when making decisions.	*	PIF's activities utilize an activity-status relation which is linked to timepoints. Therefore PIF can set timepoints for when an activity must be at a certain status.
2.2.5	dispatching - the determination and representation of rules and guidelines to decide when items should be released for production.	X	This level of object detail is not explicitly represented in PIF.
B.5 - PIF (v1.1)			

	PSL Requirement	Rank	Description
2.2.6	eligible resources - the ability to determine which resources can be chosen for a task (selection rules)	*	In terms of agents (as resources) PIF can explicitly describe their capability, thus making them eligible. PIF does not provide the "eligibility" of other non-agent objects.
2.2.7	exception handling and recovery - the ability to specify corrective action when a task fails.	$\surd$	6-Nov-96 as per Yan Jin's input: PIF's conditional activities can respond to exception handling and recovery. 4-Dec-96 Gruninger: Depends on interpretation. PIF cannot (a) while the (b) is simply a decision activity. (see below) There are two interpretations of this construct: a) global occurrence constraint which must be satisfied regardless of what activities are occurring at any point in time.b) a conditional activity which occurs at specific points during a complex activity.
2.2.8	information exchange between tasks - the ability to represent the flow of information among tasks	*	The flow can be partially mapped out by illustrating which activities create/modify/use objects.
2.2.9	mathematical and logical operations - the language must be able to perform mathematical and logical operations.	$\surd$	PIF has a mechanism to derive boolean results for conditionals, etc.
2.2.10	support for task/process templates - the language must allow for templates of a task or process.	*	PIF does not provide this explicit form of meta element, but PIF design elements can be reused.
2.2.11	support for simultaneously maintained associations of multiple levels of abstraction - the ability to associate information at multiple levels of abstraction with a task.	$\surd$	PIF decomposition allows a designer to attach/modify activity relationships at any level of abstraction.
2.2.12	synchronization of multiple, parallel task sequences - the ability to specify a mechanism to coordinate two or more tasks that occur at the same time.	*	PIF does not contain any real-time event signalling and notification that could manage multiple, parallel, activities. 4-Dec-96 Gruninger X to *: The definition of synchronized activities does not depend on real-time event signalling and notification; this only becomes an issue when coordinating agents within an organization.

B.5 - PIF (v1.1)

B.6 STEP ISO 10303-49 (Part 49) Detailed PSL Analysis

Key: √ - Completely satisfies req., * - Partially satisfies req., X - Cannot satisfy req., ? - Uncertain

	PSL Requirement	Rank	Description
1	**Core Requirements**		
1.1	*Representational Requirements*		
1.1.1	ad hoc notes and annotations optionally associated with any component of a plan - on-the-fly, off-the-cuff notes and documentation. This could be voice, video, as well as text. A person's observation of a process might go here.	√	The various description properties could be used to capture this information. Elements such as voice, and video could be related via textual filenames stored in these properties.
1.1.2	cost data - the cost associated with a resource or task. This could be a fixed cost, cost rate, or a cost derived from other attributes such as duration and level of effort. Costs associated with uncertainty, variability, tolerances, etc	*	We can partially address this by: adding a resource_property_representation that can describe the cost of the action_resource_requirement (e.g. dollars/hr for a person, etc.).
1.1.3	level of effort - description of the amount of a resource needed, in any given unit, to accomplish a task. Some example levels of effort are equipment-hour, labor-hour, and crew size.	*	An example: We could setup "crews" as action_resources. I could add an action_resource_requirement that states the requirement, "crew with 10 members" or "crew with 5 members". Then a requirement_for_action_resource reflects a satisfying crew.
1.1.4	product (work item) characteristics- information about an intermediate and final product which a process will produce.	*	Some aspects of the product can be defined as product_definitions in the product_definition_process.
1.1.5	resource - a single resource or a group of resources. Some types of resources are equipment, people, information, and in-progress goods.	√	A resource is defined as an action_resouce in part 49.
1.1.6	resource requirement(s) for a task - the relationship between one or more resources and a task.	√	Requirements can be expressed via resource_property_representations as the value of a resource_property that is part of an action_resource_requirement.
1.1.7	simple groups of tasks - very basic, high-level set of tasks. One example is the grouping of tasks and sequences that make up a process plan or that make up a phase.	√	action_method_relationships can be established to group action_methods in a very basic, high-level set.
1.1.8	simple resource capability/characteristics - a high-level description of the characteristics of a resource. More detailed descriptions can be found in the outer core.	√	Descriptions of a resource can be added as resource_properties that are related to a resource.
B.6 - STEP, Part 49			

	PSL Requirement	**Rank**	**Description**
1.1.9	simple sequences - linear, time-sequential groups of tasks. More sophisticated relationships such as parallel and alternative tasks can be found in the outer core.	$\surd$	A sequential_method can be defined that assigns a sequence position to each action_method in the grouping.
1.1.10	simple task representation and characteristics - a simple, high-level description of the task. More detailed representations can be found in the outer core.	$\surd$	actions and action_methods are used to define basic tasks. These basic tasks can be further defined by relating them to a specific product or service that they provide in the manufacturing process.
1.1.11	task duration - the time required to complete a task or group of tasks. Only simple durations are represented here.	$\surd$	Page 19 of ISO DIS 10303-49 describes the use of the action_property to define the time it takes to perform the task (or any other characteristic). Also see page 28 for action_property_representation.
1.1.12	task executor - who is responsible for executing a task or group of tasks. Examples include a person, controller, or external company if the task is contracted out.	$\surd$	There are a couple of ways to address this requirement in part 49. You can treat the executor in the same way as duration (listed above) or you can define a resource that represents the executor and associate it to the task.
1.2	*Functional Requirements*		
1.2.1	extensibility - there must be a mechanism in place to allow a user to add additional information to the pre-defined data constructs. One such mechanism could be the addition of stubs for user-defined information.	$\surd$	action_properties can be assigned to action_methods and actions to capture user-defined information intended to extend the information about a task.
1.2.2	resource allocation/deallocation for one or many tasks - the assignment and release of one or more resources to a task of group of tasks.	*	The support for resource allocation/deallocation is possible, but only at a very high grain. For example: a milling machine can be thought of as being allocated to a task and it will be deallocated once the task has completed, etc. Needs a lower level of support.

B.6 - STEP, Part 49

	PSL Requirement	**Rank**	**Description**
1.2.3	simple precedence - a high-level description of the precedence constraints of one task on another. A more detailed description of precedence and constraints can be found in the outer core.	√	There is the support for ordering constraints (see simple sequences) as well as informational constraints on the performance of tasks. For example, given the example on page 41, we can define exactly when "maintain speed" should follow "drive down street" based on the condition of the light being green. Context_dependent_relationship_conditions can be strung together to define the precedence of tasks based on the information in condition_descriptions.
2	**Outer Core Requirements**		
2.1	*Representational Requirements*		
2.1.1	abstraction - within the scope of this project, there are three concepts of abstraction that must be captured. (hierarchy, incompleteness, ambiguity)	X	The constructs are very definite and highly structured. They do not permit such things as ranges, abstract levels, etc.
2.1.2	alternative task - (see complex sequences)	√	In part 49, it is possible to relate actions in a "replacement_relationship" that means that the related_action could be used as an alternate implementation of the task. The context_dependent_relationship_conditions can also be used to determine which tasks can be alternatively performed, based on some criteria.
2.1.3	associated illustrations and drawings	√	Part 49 has a unique implementation that explicitly incorporates documentation via entities like, action_method_with_specification_reference and action_method_with_ specification_reference_constrained.
2.1.4	complex groups of tasks - groups of tasks which have a common tie.	√	Complex groupings can be made via the context_dependent_relationship_conditions and action_method_to_select_from entities.
2.1.5	complex resource characteristics - a detailed description of the characteristics of a resource or group of resources.	√	resource_properties and resource_property_relationships can be defined to describe complex characteristics.

B.6 - STEP, Part 49

	PSL Requirement	**Rank**	**Description**
2.1.6	complex sequences - complex ordering relationships between tasks	$\surd$	A variety of entities support the definition of complex sequences: sequential, serial, concurrent, context-dependent. The key entity here is the context_dependent_relationship_condition which permits the expression of preconditions of a task.
2.1.7	complex task representation and parameters - a detailed representation of a task or group of tasks.	$\surd$	Action_properties and action_property_relationships can be used to express detailed aspects of a task.
2.1.8	concurrent tasks - (see complex sequences)	$\surd$	concurrent_action_methods can be used for this requirement.
2.1.9	conditional tasks - a task that only needs to be performed under some predefined circumstance.	$\surd$	context_depednent_relationship_conditions can be defined to model predefined circumstances.
2.1.10	confidence levels - a measure of certainty that some attribute is true.	X	No support for ranges, or measures of certainty.
2.1.11	constraints - implicit or explicit constraints associated with a task or resource.	*	While it could probably be said the context_dependent_relationship_conditions represent a type of constraint, as well as the ordering, etc. the support for constraint expression is certainly not enough to say that it completely covers this requirement.
2.1.12	date(s) and time(s) and/or multiple duration(s) - the association of one or more dates and times and/or multiple durations with a resource or task	X	No support for timepoints, or time intervals, etc. This could be listed as an action_property but this temporal support should have stronger core support.
2.1.13	implicit/explicit resource association - an implicit or explicit dependency of a resource on another type of resource.	X	Cannot find a construct that relates a dependency between two resources, only a resource dependency to an action or task.
2.1.14	iterative loops - a situation when a task or group of tasks repeats until a desired condition is met	*	This may even be a completely, but there is at least a partially based on the ability to characterize what an action or action_method does, providing preconditions via context_dependent_action_method_relationships, and ordering these tasks to cycle based on some condition (or context).
2.1.15	manual vs. automated tasks - characteristics of a task can differ depending on if a human or a machine is performing that task.	*	This could be considered a property of the action (action_property).

B.6 - STEP, Part 49

	PSL Requirement	Rank	Description
2.1.16	manufacturing product quantity - the amount of the product that is to manufactured.	*	This could again be a property, but it is not well-positioned for use in an application that needs to reason about product quantity.
2.1.17	material constraints	*	If materials are manipulated as resources to be used in a manufacturing process, it would seem possible to express material constraints via action_resource_requirements.
2.1.18	parallel tasks	*	Unordered, non-related tasks can be thought of as parallel, but greater support is needed to make it completely.
2.1.19	parameters and variables - place holders that can store a constantly changing value.	$\surd$	Supported via the Express language.
2.1.20	pre- and post-processing constraints	*	Since we have conditional tasks, we can express the pre-processing constraints by stating what must be true to execute this task.
2.1.21	queues, stacks, lists - the representation of an ordered or unordered group.	X	While there are sets, there is no support for these types of data structures.
2.1.22	resource categorization and grouping - a logical grouping of resources with a common tie.	*	Resources with identical "requirement_for_action_resource" can be thought of as "grouped" by capability.
2.1.23	resource location. - identification of the location of a resource.	$\surd$	This could be treated as a resource_property or requirement_for_action_resource depending on its intended use.
2.1.24	resource/task combined characteristics - qualities of a resource that are dependent on a particular task, or qualities of a task that are dependent on a particular resource.	X	I don't think this is supported.
2.1.25	serial tasks	$\surd$	Part 49 has a serial_action_method entity to describe this relationship.
2.1.26	state existence constraints	X	Need smaller grained conditionals to express state representations.
2.1.27	state representations - the description of a process in terms of any combination of the states of the process and/or resource.	X	
2.1.28	temporal constraints	*	High-level constraints for temporal relationships. (serial, complex, etc.)
2.1.29	uncertainty/variability/tolerance - the representation of the deviation from the nominal.	X	No support.
2.2	*Functional Requirements*		

B.6 - STEP, Part 49

	PSL Requirement	Rank	Description
2.2.1	ability to insert or attach a highlight (milestones) - the ability for a user to highlight a section of the process	X	No support for highlighting a section.
2.2.2	complex precedence - the ability to convey a series of tasks' ordering requirements within a given process.	$\surd$	The key here again is the context_dependent_relationship_conditions which permit the use of specialized informational requirements.
2.2.3	convey the ancestry or class of a task - the ability to describe a task as it relates to the specialization of another, higher-level task.	X	No hierarchical/decompositional support.
2.2.4	deadline management - the ability to consider a predetermined deadline when making decisions.	X	If we take deadline a deadline to be somehow connected to a date/time representation than there is no support.
2.2.5	dispatching - the determination and representation of rules and guidelines to decide when items should be released for production.	*	This would have to be partial again since there is a method for conditionally controlling the applicability of task for a given representation.
2.2.6	eligible resources - the ability to determine which resources can be chosen for a task (selection rules)	$\surd$	Resources contain requirements_for_action_resource entries that determine this.
2.2.7	exception handling and recovery - the ability to specify corrective action when a task fails.	*	This partly depends on interpretation, but you could imagine adding a conditional task after an operative task that would do something in the event that the condition following the operative task fails.
2.2.8	information exchange between tasks - the ability to represent the flow of information among tasks	X	No support.
2.2.9	mathematical and logical operations - the language must be able to perform mathematical and logical operations.	$\surd$	EXPRESS contains math/logic support.
2.2.10	support for task/process templates - the language must allow for templates of a task or process.	X	No support.
2.2.11	support for simultaneously maintained associations of multiple levels of abstraction - the ability to associate information at multiple levels of abstraction with a task.	X	No support.
2.2.12	synchronization of multiple, parallel task sequences - the ability to specify a mechanism to coordinate two or more tasks that occur at the same time.	X	No support.

B.6 - STEP, Part 49

B.7 O-Plan TF (v.2.3) Detailed PSL Analysis

Key: √ - Completely satisfies req., * - Partially satisfies req., X - Cannot satisfy req., ? - Uncertain

1	PSL Requirement **Core Requirements**	Rank	Description
1.1	*Representational Requirements*		
1.1.1	ad hoc notes and annotations optionally associated with any component of a plan - on-the-fly, off-the-cuff notes and documentation. This could be voice, video, as well as text. A person's observation of a process might go here.	√	Notes via comments and "tf info" items. Individual plan items can contain "annotation-constraints". Extended documentation for schemas can be achieved by linking "info" attribute/value pairs with filenames of associated drawings, etc.
1.1.2	cost data - the cost associated with a resource or task. This could be a fixed cost, cost rate, or a cost derived from other attributes such as duration and level of effort. Costs associated with uncertainty, variability, tolerances, etc	*	O-Plan TF can be used to describe an action that consumes a resource (e.g. money, in the case of cost). Uncertainty costs, variability, etc. is incorporated by the use of upper/lower bounds on numerical values.
1.1.3	level of effort - description of the amount of a resource needed, in any given unit, to accomplish a task. Some example levels of effort are equipment-hour, labor-hour, and crew size.	√	O-Plan TF has a rich set of resource elements that can describe the units, types, and number of resource items that are required by an action.
1.1.4	product (work item) characteristics - information about an intermediate and final product which a process will produce.	*	O-Plan TF can be used to model a class of resources that are "producible" when an action is applied. This "produced" item can be an intermediate product that is used to supply a condition for another action.
1.1.5	resource - a single resource or a group of resources. Some types of resources are equipment, people, information, and in-progress goods.	√	O-Plan TF can be used to describe resources and resource types.
1.1.6	resource requirement(s) for a task (with quantity) - the relationship between one or more resources and a task.	√	O-Plan TF resource statements can quantify an action's usage of a resource.
1.1.7	simple groups of tasks - very basic, high-level set of tasks. One example is the grouping of tasks and sequences that make up a process plan or that make up a phase.	√	Action schemas can define partially ordered sub-actions and action schemas can be arranged hierarchically through the use of "expands" action patterns.
1.1.8	simple resource capability/characteristics - a high-level description of the characteristics of a resource. More detailed descriptions can be found in the outer core.	√	O-Plan TF can give resource characteristics that can be used to select the appropriate resource for a task. (e.g. attributing "wolf-proof" characteristics to "bricks" in a sample domain.)
	B.7 - O-Plan TF (v.2.3)		

	PSL Requirement	**Rank**	**Description**
1.1.9	simple sequences - linear, time-sequential groups of tasks. More sophisticated relationships such as parallel and alternative tasks can be found in the outer core.	$\surd$	O-Plan TF has a number of ways to express temporal relationships. "At" links actions to a specific timepoint. "Duration" specifys a range. TF can also express "delay_between" as a means to specify a latency period between the end and begin of two actions.
1.1.10	simple task representation and characteristics - a simple, high-level description of the task. More detailed representations can be found in the outer core.	$\surd$	Simple high-level descriptions can be attached via the schema annotations that were described in 1.1.1.
1.1.11	task duration - the time required to complete a task or group of tasks. Only simple durations are represented here.	$\surd$	As per Tate (22-Nov): In O-Plan a user can express duration in metric time points against a reference basis of zero time. (e.g. day 45 12:00:00 for example for noon on day 45 of a project.)
1.1.12	task executor - who is responsible for executing a task or group of tasks. Examples include a person, controller, or external company if the task is contracted out.	$\surd$	O-Plan TF can select modeled resources to be associated with an instantiated action. (e.g. selecting vehicles in pacifica sample domain). TF can also be used to directly model the "contracting" relationship using [un]supervised conditions.
1.2	*Functional Requirements*		
1.2.1	extensibility - there must be a mechanism in place to allow a user to add additional information to the pre-defined data constructs. One such mechanism could be the addition of stubs for user-defined information.	$\surd$	O-Plan "other-constraints" can be used to record additional information.
1.2.2	resource allocation/deallocation for one or many tasks - the assignment and release of one or more resources to a task of group of tasks.	$\surd$	O-Plan TF can be used to model assignment and release of resources.
1.2.3	simple precedence - a high-level description of the precedence constraints of one task on another. A more detailed description of precedence and constraints can be found in the outer core.	$\surd$	O-Plan TF conditions, effects, and expands can be used to form interschema relationships while orderings are used to define intraschema sub-action relationships.
2	**Outer Core Requirements**		
2.1	*Representational Requirements*		

B.7 - O-Plan TF (v.2.3)

	PSL Requirement	Rank	Description
2.1.1	abstraction - within the scope of this project, there are three concepts of abstraction that must be captured. (hierarchy, incompleteness, ambiguity)	$\surd$	Schemas arranged in a hierarchical fashion can abstract the details of various plan expansions. TF can be arranged into plan levels/phases that allows O-Plan to control how far to plan (incompleteness). More than 1 schema can be appropriate (ambiguity).
2.1.2	alternative task - (see complex sequences)	$\surd$	More than one TF schema may be appropriate for a plan node expansion.
2.1.3	associated illustrations and drawings	$\surd$	Textual items (comments) can be attached to O-Plan TF items and extended documentation for the domain can be achieved by linking tf_info attribute/value pairs with filenames of associated drawings, etc.
2.1.4	complex groups of tasks - groups of tasks which have a common tie.	$\surd$	O-Plan TF can describe an explicit grouping of actions (e.g. install services). TF can also address constraints related to the overall group. (e.g. describing how much resource an action and its expansions can consume.)
2.1.5	complex resource characteristics - a detailed description of the characteristics of a resource or group of resources.	$\surd$	Resources can have a “specific” type that affects how the planning system may use the resource. (movable_objects vs. objects, etc.)
2.1.6	complex sequences - complex ordering relationships between tasks	$\surd$	O-Plan TF schemas can explicity represent complex sequences as well as express the elements necessary to create more ordering relationships during generative planning.
2.1.7	complex task representation and parameters - a detailed representation of a task or group of tasks.	$\surd$	Action schemas can take into account concepts such as applicability (only_use_if), performance limits (time windows, resource consumption), and a number of constraints on its conditions, suitable parameter bindings, etc.
2.1.8	concurrent tasks - (see complex sequences)	$\surd$	20-Nov-96 via Tate: “Two actions can be constrained to have the same begin and end times by giving a zero duration link between their begin points and the same zero duration link between their end points.”
2.1.9	conditional tasks - a task that only needs to be performed under some predefined circumstance.	$\surd$	TF schema filters (only_use_if) control the applicability of a specific schema.
2.1.10	confidence levels - a measure of certainty that some attribute is true.	X	O-Plan TF does not have a means to express certainty degrees.
B.7 - O-Plan TF (v.2.3)			

	PSL Requirement	**Rank**	**Description**
2.1.11	constraints - implicit or explicit constraints associated with a task or resource.	√	O-Plan has a rich set of constraint types to limit the plan behavior (this includes actions and resource usage).
2.1.12	date(s) and time(s) and/or multiple duration(s) - the association of one or more dates and times and/or multiple durations with a resource or task	√	O-Plan TF can be used to express relative temporal relationships that are tied to an initial zero date/time.
2.1.13	implicit/explicit resource association - an implicit or explicit dependency of a resource on another type of resource.	X	There are no dependency relationships between resource types in O-Plan TF.
2.1.14	iterative loops - a situation when a task or group of tasks repeats until a desired condition is met	X	While the use of an “iterate” or “foreach” node type is planned, TF version 2.3 does not contain this functionality. (Now in O-Plan version 1 January 1997.)
2.1.15	manual vs. automated tasks - characteristics of a task can differ depending on if a human or a machine is performing that task.	√	Separate action schemas can be designed with constraints on agent binding types. If a schema is instanitated with an agent binding of type “machine” there will be a certain seq. whereas the type “human” schema would be different.
2.1.16	manufacturing product quantity - the amount of the product that is to manufactured.	√	The amount of product to be produced can be expressed as an achieve condition in a task schema and the action schemas can be designed to “produce” the resource based on constraints.
2.1.17	material constraints	*	Materials can be qualified through the use of resource types and “always” assertions. (e.g. bricks are wolf-proof, etc.)
2.1.18	parallel tasks	√	O-Plan actions are arranged in a partially ordered fashion that can represent parallel tasks.
2.1.19	parameters and variables - place holders that can store a constantly changing value.	√	O-Plan plan state variables can be used to bind values to various aspects of the plan.
2.1.20	pre- and post-processing constraints	√	This is achieved through the use of O-Plan conditions (pre) and effects (post).
2.1.21	queues, stacks, lists - the representation of an ordered or unordered group.	*	O-Plan TF utilizes “sets” but does not have specific data structures such as queues or stacks.
2.1.22	resource categorization and grouping - a logical grouping of resources with a common tie.	√	Logical resource grouping is created by using specific resource types.

B.7 - O-Plan TF (v.2.3)

	PSL Requirement	**Rank**	**Description**
2.1.23	resource location. - identification of the location of a resource.	$\surd$	The “pacifica” TF sample shows how resource location can be represented using an “at OBJ = LOC”.
2.1.24	resource/task combined characteristics - qualities of a resource that are dependent on a particular task, or qualities of a task that are dependent on a particular resource.	$\surd$	The simplest way to address this requirement is to create alternate action schemas that utilize different resources and can also thereby have different time constraints.
2.1.25	serial tasks	$\surd$	O-Plan TF can be used to impose a total ordering between actions where necessary.
2.1.26	state existence constraints	$\surd$	This requirement can be expressed in detail by selecting an appropriate condition type in O-Plan TF.
2.1.27	state representations - the description of a process in terms of any combination of the states of the process and/or resource.	$\surd$	O-Plan uses a state-based approach for plan domain representations (I.e. conditions and effects relative to a world state)
2.1.28	temporal constraints	$\surd$	Time “windows” can be expressed for actions in O-Plan TF.
2.1.29	uncertainty/variability/tolerance - the representation of the deviation from the nominal.	$\surd$	Numerical variables can be represented via Min/Max pairs and a “computed” value that must lie within this range. This allows for tolerance and variability of a value.
2.2	*Functional Requirements*		
2.2.1	ability to insert or attach a highlight (milestones) - the ability for a user to highlight a section of the process	*	As per Tate: O-Plan can support the attachment of milestones or statements (effects) about some point in the plan. But the ability to “highlight” or annotate some area of the plan is outside of what TF is trying to do.
2.2.2	complex precedence - the ability to convey a series of tasks’ ordering requirements within a given process.	$\surd$	O-Plan action orderings can be specified within an action schema or implied through the conditions and effects.
2.2.3	convey the ancestry or class of a task - the ability to describe a task as it relates to the specialization of another, higher-level task.	$\surd$	The “expands” entry in an action schema denotes how it extends a higher level action.
2.2.4	deadline management - the ability to consider a predetermined deadline when making decisions.	$\surd$	O-Plan can handle tasks with relative time constraints, durations, etc.
2.2.5	dispatching - the determination and representation of rules and guidelines to decide when items should be released for production.	$\surd$	The preconditions of an action can be utilized as a mechanism for stating dispatching rules.

B.7 - O-Plan TF (v.2.3)

	PSL Requirement	**Rank**	**Description**
2.2.6	eligible resources - the ability to determine which resources can be chosen for a task (selection rules)	√	In O-Plan TF, conditions on using resources can be defined that meet this requirement.
2.2.7	exception handling and recovery - the ability to specify corrective action when a task fails.	*	Alternative schemas (and orderings) can be chosen to satisfy a task when a suggested course of action fails.
2.2.8	information exchange between tasks - the ability to represent the flow of information among tasks	√	Information is "passed" between actions via plan state variables.
2.2.9	mathematical and logical operations - the language must be able to perform mathematical and logical operations.	√	O-Plan TF can be used to express the necessary mathematical and logical operations for this requirement.
2.2.10	support for task/process templates - the language must allow for templates of a task or process.	√	via Tate (22-Nov): All Task Formalism schemas are "generic processes" or "task descriptions" that meet this requirement.
2.2.11	support for simultaneously maintained associations of multiple levels of abstraction - the ability to associate information at multiple levels of abstraction with a task.	√	Constraints can be attached at any level of an action hierarchy that would be appropriate for that schema.
2.2.12	synchronization of multiple, parallel task sequences - the ability to specify a mechanism to coordinate two or more tasks that occur at the same time.	√	See 2.1.8.

B.7 - O-Plan TF (v.2.3)