A Common Process Methodology for Engineering Process Domains

Stephen T. Polyak

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

Abstract

Process engineering involves a search for new models of organising work. This synthesis task can become quite difficult and time-consuming as the amount of detail required and interactions between activities increases. Domain independent AI planning offers some promising techniques and representations to assist in this effort. One of the major impediments to transferring this technology to applied, real-world settings is the difficulty encountered in building the domain model which is used in the automated generation of these plans. Competence, as well as good tools, is necessary to carry out this task. A plan domain methodology should be available which provides structured organisational development activities. Users need to know what tasks they have to perform: for each step, information must be available about what input will be needed, and what output will be required, what is to be done and how it can be done well. This paper presents the Common Process Methodology (CPM) which aims at providing this support for engineering process domains.

Introduction

Process engineering involves a search for new models of organising work. This activity transcends industries. For example, manufacturing companies formulate steps for building a new product and suppliers define activities which are executed in the enactment of material and product supply chains. In either case, the amount of detail required to adequately describe these steps or activities can be difficult and time-consuming to manage. Interactions between planned steps exacerbate this process and complicate the analysis of proposed manufacturing or business plans. AI planning representations and planning systems can offer support for this synthesis activity. In fact, Tate describes how AI planning and plan representations may be used to assist in a range of activities which include (Tate 1993)

- Reliable capture and maintenance of process knowledge and models
- Decision-making using knowledge based simulation and analysis
- Synthesis of plans and schedules
- Re-engineering parts of a process or plan
- Reliably executing processes and plans
- Simulating, animating, explaining, and justifying processes and plans

Practical planners require a "knowledge rich" domain model that allows them to integrate efficiently given the demands of the surrounding environment. In order to transfer this research in AI planning to practical organisational process synthesis, we need to build this domain model. Langley and Drummond state that

> "For engineering development and technology transfer purposes, tasks that include practical difficulties will be more useful [than artificial domains]." (Langley & Drummond 1990)

An example of an AI planning system which has been designed to tackled these "practical difficulties" is the Optimum-AIV planner (Aarup *et al.* 1994; Arentoft *et al.* 1991). Optimum-AIV is a planning system implemented for the European Space Agency that is used in the assembly, integration, and verification (AIV) of spacecraft. This planner is accessible to managers that require a detailed level of interaction and control over plans. Optimum-AIV is based on the open planning architecture defined in the O-Plan research (Currie & Tate 1991). O-Plan planner concepts have also been used in a number of challenging environments including: back axle assembly process planning at Jaguar Cars, software procurement planning at Price Waterhouse, and factory production planning and job shop scheduling at Hitachi. A number of other applied planning systems that rely on rich domain representations could be added to this set as well (Wilkins 1988; Fuchs *et al.* 1990; Drabble 1990; Drummond & Tate 1992; Johnston & Adorf 1992).

For each of these uses, a domain was constructed which provides information about the activities that may be performed (at various levels of abstraction) and the tasks which may be proposed to the planning system. Unfortunately, acquiring and maintaining this domain knowledge is currently considered to be a highly significant bottleneck in utilising planning systems (Wang 1996). In fact, the activities involved in discovering, engineering, documenting, and maintaining a set of domain constructs for most domain independent AI planning-based projects can be considered ad hoc and disorganised, at best. The current sources for advice on the process of writing AI planning domain descriptions have been summarised as

> "... it is the most neglected aspect of planning, and there is not an established software-engineering methodology to guide this job". (Erol 1995)

The purpose of this paper is to present an approach, the Common Process Methodology (CPM), which is aimed at addressing this undeveloped aspect. This is an important endeavour if we wish to encourage the application of AI planning techniques to real world situations. A more disciplined approach to producing domain specifications is required in applied settings which encompasses requirements capture, analysis, and specification.

The following section describes the past work which was used as a basis for building the Common Process Methodology. Following that, we describe how this approach fits within an overall framework for managing and using rich plan representations. The methodology is then presented using a practical supply chain scenario. Finally, we examine some of the related work and discuss conclusions drawn from our experience with this approach.

Building on Past Research

Domain capture and modelling has been an issue in Edinburgh-based planning research as early as the work on the Nonlin (Tate 1977) planner. The original O-Plan overall architecture and system design, which dates from 1983, outlined a need for a defined methodology which would guide users performing various roles in the acquisition and analysis of domain requirements for planning (Currie & Tate 1991). This planning lifecycle methodology was envisioned as encompassing a set of standardised activities and methods which had well-defined design criteria, techniques, and tools. This was proposed to assist in transforming planning domain development from a craft towards more of an engineering activity.

The domain description language used by both the Nonlin and O-Plan planners is the Task Formalism (TF) (Tate 1977; Tate, Drabble, & Dalton 1994). Early prototyping efforts on a PERQ-based TF Workstation (Tate & Currie 1984; 1985) demonstrated tool-support for the domain modellers (an expert providing the structure of the domain and specialists providing the details) and planners (acting in any one of a range of roles). This tool was designed to include an "intelligent assistant" which would interact with the user via a structured dialogue which was tied to a specific domain development methodology. Research was conducted into a requirements engineering methodology which could be adapted for use in this way. The Controlled Requirements Expression (CORE) (Mullery 1979; Curwen 1991) was proposed for structuring these domain management activities. Unfortunately this work had been set aside following the initial exploration of these ideas (Wilson 1984). Recently, we provided a review of this past work which pointed toward the development of a new methodology which could revitalize this area of research (Tate, Polyak, & Jarvis 1998).

COntrolled Requirements Expression

COntrolled Requirements Expression (CORE) was a method developed by British Aerospace (Warton) and systems designers in the late 70's (Mullery 1979). As was mentioned above, initial work at Edinburgh sought to relate this method to engineering AI planning domains. Over time, the method has evolved and CORE now provides techniques for requirements capture, analysis and specification (Curwen 1991). The method can be used to partition problems into manageable modules which can be assessed using CORE analytical techniques. This helps to ensure that the requirements for a specification are complete and consistent. Some of the strengths of this methodology include decomposability of requirements and traceability mechanisms between different levels of requirements.

The CORE specifications are expressed in terms of graphics, structured text and specialised notations. These resultant requirements models start from operational requirements which influence functional requirements and, in turn, impact implementation requirements (with non-functional requirements acting as functional and implementation constraints). Viewpoints are used as logical partitionings of the system under consideration. These are divided into **bounding viewpoints**, which may be viewed from a planning context as providers of unsupervised conditions and

defining viewpoints which are analogous to activities which can achieve conditions. Viewpoint decompositions correspond to node expansions. The CORE notion of "scope" addresses choices between elements which may be included in the domain, and breaks them down into "local scopes" which designate responsibilities for domain specialists.

An adaptation of the CORE methods can be used to structure the activities of users acting in particular roles throughout the engineering of a domain. For example, a domain expert divides a domain into a series of tasks to be completed by specialists. Domain specialists can list the assumptions they will be making within their scope (e.g. for a supply chain domain these assumptions may include: order validated, delivery notice sent, etc.). Specialists can retrieve previous parts of a domain specification to modify. For each specification, a viewpoint decomposition process is applied to it. This includes some model checking based on CORE analysis techniques.

CORE provides specialised techniques for inspecting the evolving specification. One example is the "viewpoint to viewpoint role-playing" technique. Using this approach, structured documents are produced which define a particular perspective within the domain (e.g. for a supply chain domain this may be between a retailer and a distributor, or between a manufacturer and a transportation company, etc.) Techniques such as this one aid in combining the viewpoints by showing where conflicting requirements are present.

TF Method

In addition to this input from CORE, we have also drawn on initial work aimed at pulling together the experience gained in coding specific domains in the Task Formalism domain description language. This input comes from a section on "guidelines for writing TF" which is part of the TF manual (Tate, Drabble, & Dalton 1994). This section provides advice on the use of various TF forms and elements, which can be seen as a contribution toward a methodology which would structure the domain design activities.

This advice is rooted in a project management perspective which describes the need for preparatory steps and uses role identification prior to engaging in the development-oriented activities. The central controlling role was identified as the **Domain Expert** who is in charge of managing the scope of the domain and introducing a "top level" description (e.g. in a supply chain domain this person might be a consultant with overall responsibilities for coordinating the business processes). For large domain engineering efforts, a partitioning of the domain development responsibilities was recommended. These modular sections of the domain were viewed as "detailed" aspects of the top level descriptions which were provided by the domain expert. **Domain Specialists** would then be assigned to particular domain partitions and would have responsibility for providing specifications of activities, objects (resources), events and effects which were relevant to their particular needs. The specialists may be subject matter experts (e.g. in a supply chain domain this might represent a distributor, or manufacturer, etc.). More likely, the domain expert and specialists may be knowledge engineers who have performed the required knowledge elicitation and acquisition activities from those with knowledge of the domain.

The guidelines also point toward necessary project management decisions such as the choice between one of two "main approaches" toward modelling a domain: hierarchical action expansion or goal achievement (conditions on world states). While these approaches can be mixed in the specification of the domain, experience had shown that it is useful, if not important, to specify what the main approach will be for a particular domain development process and to treat the other approach as secondary to it.

Another important management decision considers the selection of a method for structuring the domain specifications. A level-oriented approach to domain modelling is proposed in this work whereby actions, events, effects, and resources are all separated into a series of defined and increasingly detailed levels. This helps to avoid the commonly experienced problem of "hierarchical promiscuity" (Wilkins 1988) or "level promiscuity" which is characterised by the inconsistent usage of various domain elements at varying areas in the overall domain description.

This level-oriented approach is further detailed via a checklist of activities which may suit either the action expansion approach or the goal achievement approach, depending on the ordering of the defined activities. This checklist includes the following activities:

- Identify the main actions (and events) that will appear at the top of a task or plan.
- Develop the detailed actions (and events) for lower action levels.
- Think about what world statements will be needed (effects) at which levels.
- Consider the conditions for actions. Ensure they are introduced at a level which is at or below the level at which the related effects are introduced.
- Add type information to restrict usage of conditions. Types are primarily used to differentiate what a con-

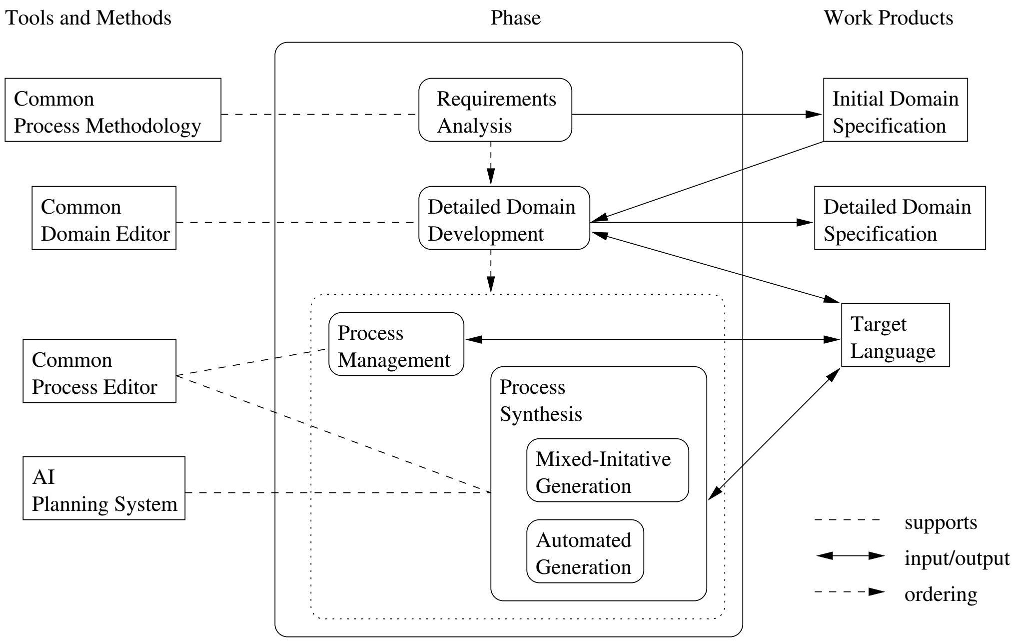


Figure 1: CPF Phases and Tools

dition means. This will lead to differences in which condition satisfaction methods apply.

- Add resources at each level.
- Consider time restrictions and related information.

This initial collection of heuristics and guidelines, combined with the CORE methods and structure provided the initial input toward the development of the Common Process Methodology. We envisage this methodology as playing a role within a defined framework for managing and using rich plan representations. The following section describes the function of CPM within a general Common Process Framework.

Fitting into a Framework

The Common Process Framework (CPF) encompasses a life-cycle of activities from the initial requirements specification of a domain model through the refinement of that model for input in generative domain independent AI planning and finally to the subsequent uses of the newly synthesised processes or plans. Competence, as well as good tools, is necessary to carry this process out. A framework such as this should offer a method which guides the required development activities. Users need to know what tasks they have to perform and with whom they are to cooperate: for each step, information must be available about what input will be needed, and what output will be required, what is to be done and how it can be done well.

As Figure 1 shows, the Common Process Methodology addresses the initial phase involving requirements analysis. The activities and work products generated in this phase are all geared toward the development of an initial domain specification. Then, within the CPF, this specification is passed along to a more detailed domain development phase. At the end of the detailed phase, the final domain model is translated into an operational target language (e.g. Task Formalism) for input to an AI planner or for other applied uses within process management tools, process evaluation tools, etc.

Common Process Methodology

The development of the Common Process Methodology was based on a number of current assumptions and perceived problems with utilising AI planning technology in large organisational settings. Some of the more important issues include

- Plan domain development projects can involve a sizable number of people (expert, specialists, knowledge engineers) who will have to cooperate in an efficient manner.
- Similar to a software system, a plan domain cannot be viewed or touched and is much more abstract than corresponding components in other types of engineering (e.g. in the building industry it is usually possible to "see" how the component parts fit together.). In building a plan domain it can be very

difficult to understand how the different parts of a domain are structured.

- The application of AI planning is a very young branch of industry. In order to succeed, it will be necessary to view it as an engineering discipline with industrial techniques.
- Plan domain development can be a very complex task involving very many significant details.

These problems and assumptions are similar to those facing software engineers in developing software systems. This was realized at least as far back as the previously cited work on exploring the links to the CORE methodology. The methods and techniques utilised in providing a more disciplined approach to producing system specifications appear to offer support for domain development as well. This research represents a step forward in validating this opinion and in extracting these methods for use in engineering process and plan domains.

Aim of CPM

CPM's aim is to partition the domain into manageable modules that can be analysed using rule-based and human-supported interaction techniques. This is strongly based on the way CORE partitions a software system design. As in CORE, the intermediate specifications will be expressed in terms of graphics, structured text, and specialised notations. The focus is on decomposing requirements into further detail and providing traceability mechanisms between different levels of requirements.

Some of the essential attributes of CPM requirements include:

Modularity The specification of a domain can become increasingly complex and it is important to break it down into modules which will enable it to be analysed. This will also assist domain specialists and experts in assessing the impact of various modifications to the specification.

Hierarchical Many AI planners support a HTN-style (Sacerdoti 1975; Tate 1977; Erol 1995) of domain abstraction. This should be supported in the specification. Structured levels of increasing detail in the specification also make it easier to understand.

High Quality High quality is meant to indicate the aim toward a balanced set of competing properties which include requirements which are: unambiguous; consistent; complete and visible. Visibility in this context means easily accessible and readable for any specification user.

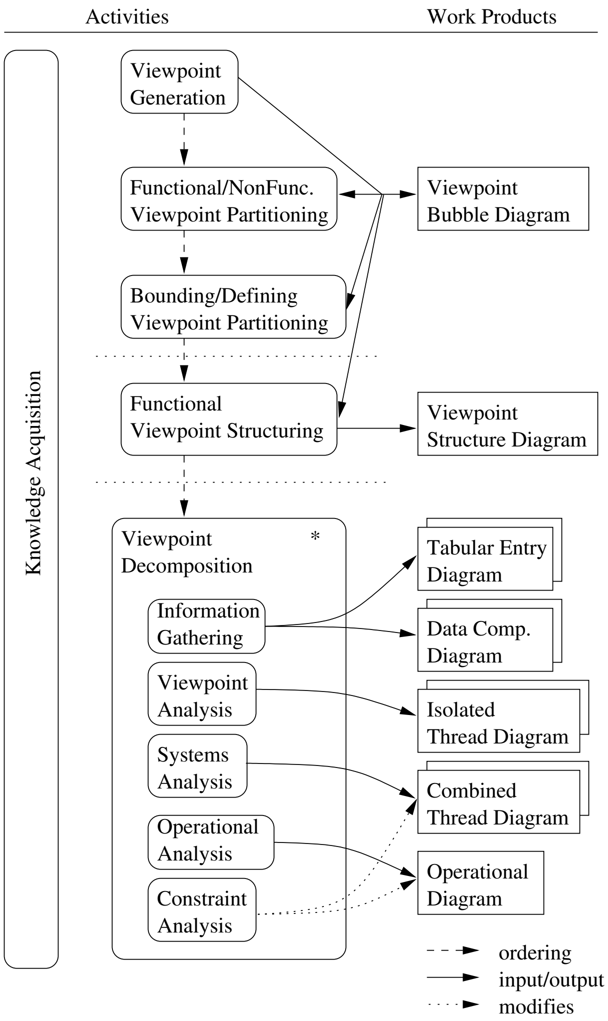


Figure 2: Common Process Methodology Steps

Verifiable Requirements need to be in a form such that they can be verified for consistency and completeness.

Main Activities

CPM is composed of various structured activities which are illustrated in Figure 2. To reiterate from above, the goal of these activities is to elicit and partition the domain knowledge and to incrementally move toward an initial domain specification. In this section, we review each of these steps and discuss the work products produced at each stage.

Viewpoint Generation

The first step in CPM is *viewpoint generation*. CPM draws on the CORE notion of *viewpoints*[1]. In the

[1]Research on viewpoints is ubiquitous in the requirements engineering field, cf. (Finkelstein *et al.* 1994;

subsequent steps, the methodology will provide guidance on how to modularise a domain into a hierarchical structure through the use of these abstract viewpoints. Viewpoints have been defined in CORE as

> "A viewpoint is a logical partitioning of the system under consideration." (Curwen 1991)

Viewpoints can be used to examine the domain in a variety of ways which enables the domain analyst to focus and capture information relevant to a specific perspective. In this way, viewpoints play the role of breaking the domain down into a number of modules. In viewpoint generation, the domain expert along with the various specialists conduct a session in which potential viewpoints are listed. This is similar to the "brainstorming session" suggested in the scoping phase of the ontology capture process described in (Uschold & Gruninger 1996).

As suggested in Figure 2, this process is centred around the development of a Viewpoint Bubble Diagram (VBD). Each candidate viewpoint is simply drawn as another bubble on a diagram space. This type of initial knowledge acquisition task is typically found in most KBS-based methodologies (cf. the data conceptualization stage in KEATS (Motta *et al.* 1991)). Viewpoints may be proposed and rejected as an exploration of possible entities yields more knowledge about the scope of the domain. For example, the diagram in Figure 3 is a simplified version of a diagram generated for a supply chain scenario (Polyak 1998) using the CPM toolset which is described in the section labelled "Tool Support" on page . This scenario will be used throughout this document to illustrate the flow of the CPM activities.

In addition to the input from experts and specialists, there may also be additional sources of information for generating domain viewpoints. The collection of this knowledge is reflected in Figure 2 as part of the ongoing knowledge acquisition activity which is enacted throughout the CPM process. These sources may include

- Existing documentation about the domain (structured or unstructured).
- Existing organisational policies and procedures which influence the domain.
- Documentation or knowledge of other domains which are similar or which reflect "best practice" processes (cf. (Malone *et al.* 1993)).

Kotonya & Somerville 1996; Easterbrook & Nuseibeh 1996)

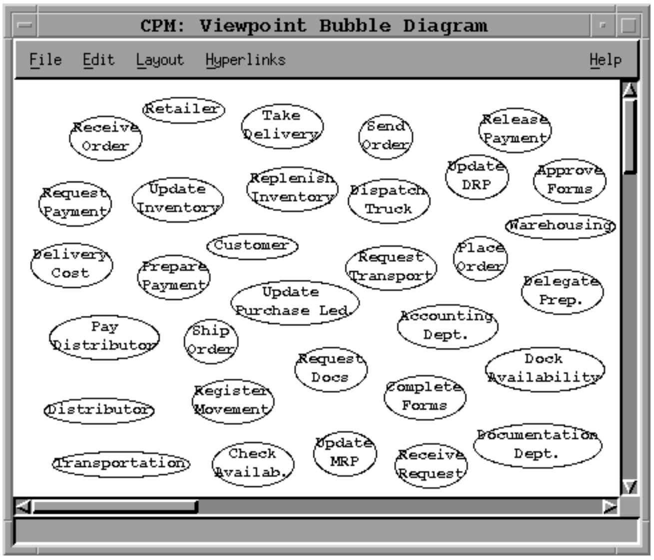


Figure 3: Initial Viewpoint Bubble Diagram produced in Viewpoint Generation

Functional/Non-functional Viewpoint Partitioning The next step in applying CPM to the engineering of a process domain is to perform an initial partitioning of the proposed viewpoints into those which are functional and those which are non-functional. Depending on the complexity of the domain and number of viewpoints generated, this may be performed immediately after the initial generation phase or over a series of subsequent sessions. In order to determine which type a particular viewpoint may be, the following definitions have been adapted from CORE:

Functional Viewpoint A logical partitioning of the domain under consideration into modules that are transformers of information.

Non-Functional Viewpoint A group of requirements that modify or constrain the functional requirements of the domain.

The "transformers of information" part of the functional viewpoint definition is very important. Functional viewpoints typically input data, transform it, and output the results. If a viewpoint cannot be characterised in this way, then it will be considered to be a non-functional viewpoint. Non-functional viewpoints constrain requirements. These constraints may be domain-wide or may only affect part of the specification.

For example, working with the VBD in Figure 3 we can identify two probable non-functional viewpoints: Delivery Cost and Dock Availability. Both of these items reflect constraints on the possible functional aspects of the domain. The remaining viewpoints appear to be functional, i.e. transformers of information in the domain. As in CORE, CPM considers "function" to be the driving, master set of requirements. The identified non-functional requirements will be subsequently expressed either in structured text or in a specialised notation, whereas the functional requirements will play a central role over the next steps in the development process.

Bounding/Defining Viewpoint Partitioning

In this phase, the functional viewpoints are examined in more detail to derive another partitioning of this subset of proposed domain elements. We will consider these viewpoints to be one of two types: bounding or defining. The following definitions can be used to determine the type.

Bounding Viewpoint A bounding viewpoint is a functional viewpoint which represents an outermost point or bounding edge in a domain. A bounding viewpoint is therefore used to generate a view of how the domain looks from a particular, fixed outer vantage point. Bounding viewpoints are non-decomposable.

Defining Viewpoint A defining viewpoint represents part or all of a segment of the domain being explored and is therefore used to describe how the domain functions internally. Defining viewpoints may be decomposed.

As an example, in our supply chain domain example we can identify the following items as being "bounding viewpoints"

- Retailer
- Customer
- Warehousing
- Documentation Dept.
- Accounting Dept.
- Distributor
- Transportation

Looking at these items, we can see that we have omitted another important, high-level bounding viewpoint which should be within the scope of this domain: the **Manufacturer**. While we may uncover more bounding viewpoints as we continue to refine our knowledge of the domain, we are interested in trying to provide a relatively complete set of top level perspectives.

Turning our attention now to the defining viewpoints, we can identify several general viewpoints which can be used to further subgroup these elements. Sometimes these general viewpoints are already present in the generated set. Alternatively, we may need to add new generalisations to group related viewpoints. Figure 4 shows our results from applying this method to the functional set in Figure 3.

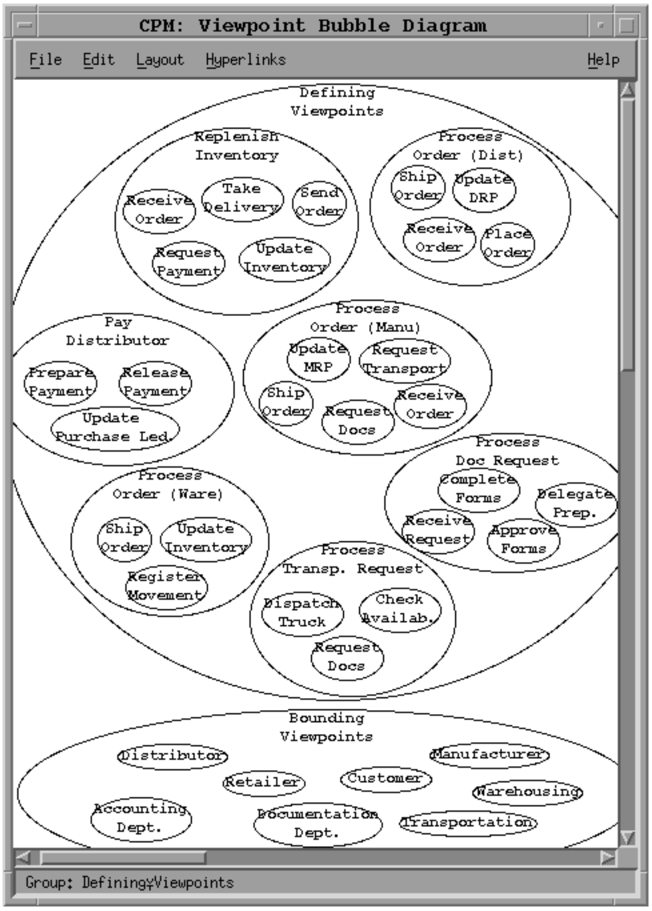


Figure 4: VBD After Partitioning Phases

Most of these general grouping viewpoints have been added to provide structure and to associate related defined viewpoints. Note that some of the originally identified viewpoints were appropriate in more than one generalised viewpoint. For example, receiving orders and shipping are common to different supplier perspectives. "Replenish inventory" is an example of an existing viewpoint which turned out to be a grouping of other defined viewpoints.

Functional Viewpoint Structuring

In this stage of engineering the process domain, we have a Viewpoint Bubble Diagram as input which was produced and refined during the prior steps. The next task is to convert this VBD into a Viewpoint Structure Diagram (VSD). The VSD will provide the framework to capture and analyse the detailed requirements for the domain. This structure consists of nodes which represent viewpoints linked in a hierarchy of detail with increasing detail towards the bottom of the diagram and decreasing detail toward the top. The top level node created simply represents the entire scope of the domain (e.g. "Supply Chain Domain"). Bounding viewpoints are then arranged with respect to their relevant levels alongside defining viewpoints. In CPM as in CORE, a general guideline is that no more than five functional viewpoints should be chosen for each decomposition of the structure.

The converted VSD for the supply chain scenario is shown in appendix A. The top level node, "Supply Chain Domain" is decomposed into three main defined viewpoints: replenish inventory, processing a distribution order, and processing a manufacturing order. The domain is considered to be bounded at a top level by a customer and an environment viewpoint. An environment viewpoint is particularly useful in relating the domain processes to external (i.e. unmodelled) influences. The additional grouping viewpoints identified in the VBD were considered to be logically part of one of these top three perspectives. For example, the warehousing and transportation aspects are in fact, a subset of an overall manufacturing viewpoint for this particular domain. It should be noted that there is no single "right way" of decomposing a domain. The important aspect is that the viewpoint structure addresses the whole of the domain and agrees with the perspectives gained during knowledge acquisition.

In the next series of steps, CPM provides methods for decomposing viewpoints structured in the VSD. The aim of this phase is to incrementally build a specification at each branch of each level in the hierarchy. These specifications are analysed and then finally combined to produce a uniform domain specification. This process involves the creation of Tabular Entry Diagrams (TED) which define the interfaces for the various viewpoints. From a given TED, we will derive a series of isolated threads of activity. These isolated threads will then be combined to unite flows between viewpoints. An operational analysis phase will bring all of these combined threads together to outline the overall structure of the processes. A final phase will add in the remaining constraints required for the domain.

Information Gathering

In this first phase of decomposing viewpoints, we will create a Tabular Entry Diagram (TED) for each functional viewpoint in our VSD. A functional viewpoint is used as a mechanism for focusing the specialist or expert's attention on a small area of the domain. A newly created TED must be assigned: a unique reference, a title, a reference to its parent. Next, a series of 5 columns are provided which can be used to record information relevant to the viewpoint. These items are

Actions Something that transforms information. Question to ask: What action does the viewpoint provide?

Inputs Information about the world that is required by an action. Question to ask: For each action, what are it's inputs?

Outputs Information about the world which is produced by an action. Question to ask: For each action, what are it's outputs?

Source The corresponding Tabular Entry Diagram which provides an input. Question to ask: For each input, do I know a specific source for the information?

Destination The corresponding Tabular Entry Diagram which provides an output. Question to ask: For each output, do I know the destination of the information?

As mentioned above, the focus of a TED is to elicit the "interface activities" for a particular viewpoint. These interface activities expose functionality which may interoperate with other activities in another viewpoint. A given viewpoint may offer alternative activities which can require similar inputs or similar outputs. For example, "transport products via truck" and "transport products via plane" both provide the similar output of transforming the location of the products. Node notes may be attached to any of the above items to provide details which help to clarify the interpretation of the item. The numbering of these diagrams is based on the IDEF3 (Mayer *et al.* 1992) style of process numbering: Parent.Decomposition Number.Unique ID.

It is important to note here that the source and destination columns are not required. This is a major step away from CORE and is an important observation used in CPM. One of CORE's assertions is that the specification should eventually be entirely statically connected (i.e. all inputs and outputs map to a specific set of sources and destinations). Lessons learned from representations used in AI planning suggest that these links should be "dynamic". This permits users to build "generic" viewpoints which may be utilized in a various parts of the specification. The source and destination columns can still be used to model "known" links between specific viewpoints, but do not reflect a limiting set of such possibilities.

This particular representation enables a domain specialist, or domain expert to perform one of two possible techniques in capturing these interface activities. These techniques are characterised by role-playing. They include

Viewpoint-to-Viewpoint Role-Playing (VVRP) In VVRP, the analyst assumes the viewpoint of the particular TED and then selects another viewpoint at the same level. The analyst then considers the possible exchanges which may take place between the two viewpoints for the domain. For example, if the TED is "Customer" and the target viewpoint is the "Replenish Inventory" process which is enacted at a retail store, the analyst may reason that the Customer will provide an "order-goods" activity which will output "selected-goods" that is required for "Replenish Inventory". Likewise, the analyst may conclude that from a customer's viewpoint, an activity should be provided called "receive-goods" which will take "retail-goods" as output.

Isolated Viewpoint Role-Playing (IVRP) IVRP is similar to VVRP, but this is a less constraining activity. Again, the analyst assumes a viewpoint for a particular TED, but then simply hypothesises what activities will be provided and what inputs and outputs will be present.

Individual domain specialists may be tasked in parallel with developing an IVRP for their scope in the domain. These diagrams then serve as focal points for discussions aimed at understanding the assumptions being made and for ameliorating the differences. In practice, VVRP and IVRP can be used in combination where necessary.

Another aspect of this phase involves support for analysing the data (i.e. inputs and outputs) which are specified in the diagrams. Separate Data Composition Diagrams can be attached to individual data entries. The graphical notation for this can be traced back to (Jackson 1975). Figure 5 illustrates an example of a simplified retail sales order from the supply chain domain. The data item box labeled "Item List" has an asterisk to indicate that there may be more than one item line per order. The "Total Cost" item is further specified to either be expressed in UK pounds or US dollars. The mutually exclusive symbol (0) is used to

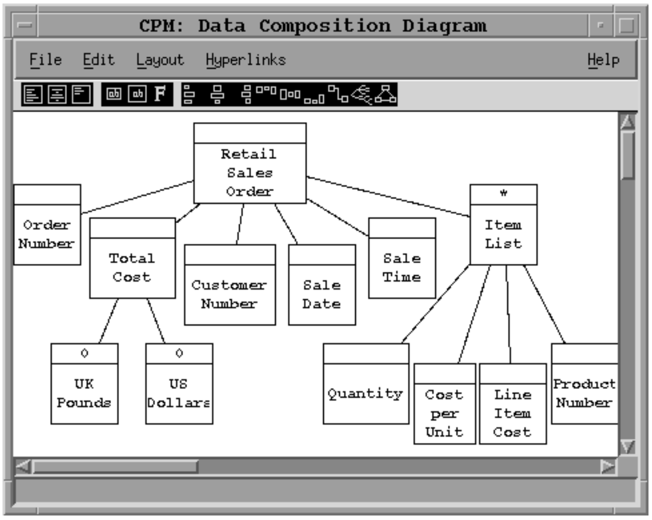


Figure 5: Data Composition Diagram

indicate this situation. As an alternative to the the graphical notation, a specialised notation (which was originally developed in (DeMarco 1978)) may be used to express the same relationships more compactly[2]:

```
Retail-Sales-Order =
 {Order Number!+
  [Total-Cost-Pounds/Total-Cost-Dollars]+
  Customer-Number+ Sale-Date+ Sale-Time+
  1{Quantity+ Cost-Per-Unit+
    Product-Number+ Line-Item-Cost}250}
```

Viewpoint Analysis

The next stage in the viewpoint decomposition phase involves viewpoint analysis. Viewpoint analysis is concerned with analysing each action of each viewpoint in isolation from one another. The main goal of this analysis is to determine: what starts a viewpoint action? and what stops a viewpoint action?. This is done by creating separate, isolated threads for each action and classifying each data input as being either: event data, control data, or data containing information. The following definitions are used to make these decisions.

Event Data When the data is generated the receiving action must occur, i.e. event data is a trigger which starts actions. Event data has no value, the data is either there or it is not there.

Control Data When the data generated is used to select the operation of different actions. These actions will be mutually exclusive. The selection may be based on the state of the control data or on limits of the value of the data.

[2]See Appendix C for a summary of this notation

Data Containing Information When the data generated contains information which is required to be used by an action.

We also need to consider whether the data is critical or non-critical. Critical data is characterised by data which once generated must eventually be used and used only once, i.e. it is consumed when read and is not overwritten. Non-critical data on the other hand is volatile and can be overwritten. This gives rise to the following possible reasons why each isolated thread action starts:

- time (which may also involve iteration)
- event data
- critical control data
- critical data containing information
- a missing critical data input

The analysis will also determine the reason why each isolated thread action stops

- if it is started by time then it could only occur a specific number of times or over a range
- if it is started by time then it could be stopped by non-critical control data (e.g. enable/disable)
- if it is started by time then it could be stopped after an internal condition changes
- if it is started by critical input then it is a "one shot action"

The result of this stage is a set of isolated threads for each viewpoint. These threads provide a basic set of building blocks which can be used to create "combined threads". A combined thread can be thought of as an operator (in traditional STRIPS-style planning), schema (in Task Formalism) or a plot (in SIPE-2 (Wilkins 1988; Myers & Wilkins 1997)).

Systems Analysis

The analysis completed so far has been concerned with the modularisation of the requirements by using viewpoints. For each viewpoint, actions have been analysed in isolation from each other further modularising the requirement. Systems analysis is the process in which the sequence and concurrency between actions at a given level is next determined. This requires a weaving of separate threads developed across viewpoints into a collection of combined thread diagrams. Each diagram represents a composite process, built from the basic isolated threads, which will be available for synthesing new "models of work".

In Figure 6, we can see how a sampling of the isolated threads within the customer, replenish inventory, and retailer viewpoints (at the top) were combined to form the combined threads (at the bottom) which represent composite supply chain domain processes. The dashed lines represent critical data containing information (e.g. Selected-Goods) or critical event data (e.g. Request). The solid lines represent non-critical data containing information (e.g. Retail-Order-Details). The means of evaluating the relationships is to examine the nature (critical/non-critical) of the data crossing viewpoint boundaries. From the definition of critical data, it can be seen that an action produces critical data which then triggers the receiving action. This means that actions linked by critical data must occur in a sequence as the receiving action can only occur after the producing action.

The way non-critical data has been defined states that the generating action periodically produces new data that overwrites the previously determined value, the receiving action using the latest value available. In this case, the relationship between the producing and receiving actions is a concurrent one because the producing action does not have to occur before the receiving action. Thus, critical and non-critical data can be used as one means of detecting sequence and concurrency between the isolated threads of different viewpoints.

Operational Analysis

In systems analysis we produced a number of combined threads which show the required sequences (schemas) in the domain. Operational analysis is concerned with bringing all these separate combined threads together onto one diagram – the operational diagram.

The operational diagram represents sketches of potential life-cycles for high level processes. These high level processes may be considered to be "task schemas" in the Task Formalism. For example, in manufacturing, we may have a high-level "build product GT-350" (Polyak & Aitken 1998) or in supply chain management we may have a "enact supply chain" process. The life-cycle is constructed by identifying the major events that occur within the domain for that high level task. In the manufacturing domain example (where the product GT-350 is a model car), this may include: making the interior, building the drive, adding the trim, making the engine, and affixing the drive.

The completed operational diagram, therefore, provides a total picture of how the domain will be logically configured for addressing a task. While building

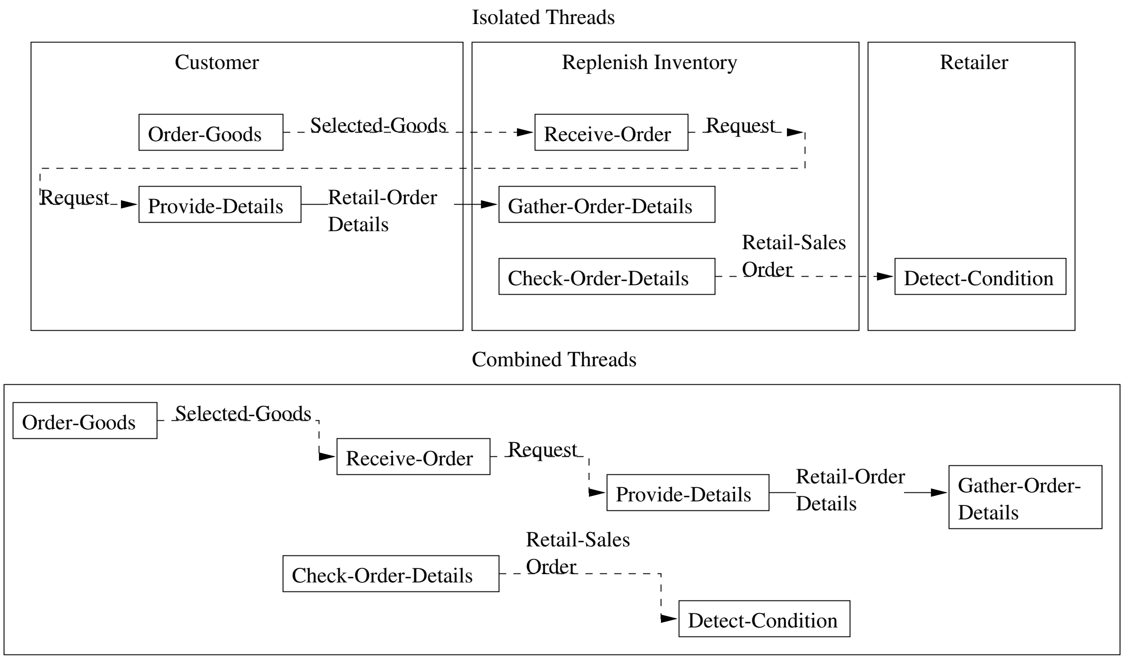


Figure 6: Creation of Combined Threads from Isolated Threads

up this complete picture, a final check should be made to ensure that no actions which are required over the life-cycle are omitted. The resulting operational diagram effectively provides an index to the detail of the domain which is shown on the combined threads.

Constraints Analysis

The final technique in CPM takes us back to the non-functional requirements which we elicited back in the initial viewpoint generation phase. As mentioned previously, these non-functional requirements will introduce a modifying or constraining influence on the functional requirements developed above. They may involve adding new functionality or new flows to the combined thread or operational diagrams. These may also be expressed via detailed node notes as well.

To some degree, this sharp division between functional and non-functional domain requirement considerations is an idealisation. In practice, several constraining influences will have crept into the earlier analysis stages. However, this separate stage serves as a reminder to go back and consider those unaddressed aspects and to formally complete the work on the initial specification.

Tool Support

The Common Process Methodology is supported by the CPM toolset. The toolset is a customisation of the HARDY[3] hypertext diagramming meta-case-tool which was developed at the Artificial Intelligence Applications Institute (AIAI). The toolset runs on UNIX X Windows (Solaris 1.x or 2.x.) as well as Microsoft Windows (3.x, 95/98, NT).

The specialised diagram types (VBD, VSD, TED, etc.) have been encoded in HARDY in order to provide window-specific pallets and presentations. The pallets display the available diagram options (i.e. various node and arc types) from which a user selects. Arcs are constrained to only allow connections between appropriate nodes (e.g. an I-A arc on a TED can only relate inputs and actions, etc.). The support for hypertext linking and retrieval in the CPM toolset enables efficient, intuitive browsing between the various diagrams.

Besides the Human-Computer Interface benefits provided for creating and managing the work products, the CPM toolset also contains an embedded expert system shell for rule-based processing and analysis of the diagram contents. This is made possible by HARDY's built in link to NASA's rule-based and object-oriented language CLIPS 6.0 (Giarratano 1994). CLIPS close link with HARDY is utilised for preformatting newly created diagrams, analysing diagrams for completeness and consistency and exporting the initial specification for use within the Common Process Framework[4].

[3]See http://www.aiai.ed.ac.uk/project/ for information on the HARDY project at AIAI.

[4]The mapping of the CPM specification onto the Com-

Currently, the only implemented completeness and consistency checks we have encoded in CLIPS have to do with those described in CORE for the Tabular Entry Diagrams. These checks aid in detecting and correcting conflicts between different parts of the domain description. In the following section, we will present the basic language and axiomatisation which underlies the implemented CLIPS rules and representation of the diagram constructs.

Basic Language and Axioms for Tabular Entry Diagrams

In this section we summarise a simple axiomatisation of the tabular entry diagrams described earlier. We have sorts $\mathcal{A}$, $\mathcal{I}$, $\mathcal{O}$, $\mathcal{S}$, $\mathcal{D}$ for actions, inputs, outputs, sources, and destinations which may be entered on a Tabular Entry Diagram, $\mathcal{T}$. Variables are denoted by lower case letters (with or without subscripts), and constants are denoted by upper case letters (with or without subscripts). Unless otherwise stated, letters a, i, o, s, d, t (A, I, O, S, D, T) and used for variables (constants) of sorts $\mathcal{A}$, $\mathcal{I}$, $\mathcal{O}$, $\mathcal{S}$, $\mathcal{D}$, $\mathcal{T}$ respectively.

Firstly, we have the simple association that all objects of type $\mathcal{A}, \mathcal{I}, \mathcal{O}, \mathcal{S}, \mathcal{D}$ are required to be associated with only one T. So, we will repackage these all with appropriate 2-place predicates:

$$(\forall a)(\exists t).action(t,a) \tag{1}$$
$$(\forall i)(\exists t).input(t,i) \tag{2}$$
$$(\forall o)(\exists t).output(t,o) \tag{3}$$
$$(\forall s)(\exists t).source(t,s) \tag{4}$$
$$(\forall d)(\exists t).destination(t,d) \tag{5}$$

Thus, a given tabular entry diagram, T, contains a tuple of sets $< \{A_0, A_1, ... A_n\}$, $\{I_0, I_1, ... I_n\}$, $\{O_0, O_1, ... O_n\}, \{S_0, S_1, ... S_n\}, \{D_0, D_1, ... D_n\} >$. Next we have the following relations which may be used to associate the objects within a particular T:

$$(\forall a,t,i).has-input(t,a,i) \supset action(t,a) \wedge input(t,i) \tag{6}$$
$$(\forall a,t,o).has-output(t,a,o) \supset action(t,a) \wedge output(t,o) \tag{7}$$
$$(\forall i,t,s).has-source(t,i,s) \supset input(t,i) \wedge source(t,s) \tag{8}$$
$$(\forall o,t,s).has-destination(t,o,d) \supset output(t,o) \wedge destination(t,d) \tag{9}$$

mon Process Language (CPL) will be presented in a future paper.

In addition, a T is required to have one and only one parent, T_{parent}. This hierarchy of parents ultimately stems from the top level node anchoring the VSD.

$$(\forall t)(\exists t_1)).has-parent(t,t_1) \wedge \neg((\exists t_2).has-parent(t,t_2) \wedge t_1 \neq t_2) \tag{10}$$

From this hierarchy of parent relations we can infer a standard, transitive $ancestor-of \subseteq \mathcal{T} \times \mathcal{T}$ relation which will be used in establishing the traceability of data interface definitions between modelled levels. This is inferred by the rules:

$$(\forall t1,t2).has-parent(t1,t2) \supset ancestor-of(t2,t1) \tag{11}$$

$$(\forall t1,t2,t3).has-parent(t1,t2) \wedge has-parent(t2,t3) \supset ancestor-of(t3,t1) \tag{12}$$

TED Specification Analysis

Given this representation, the following checks are made for the entire set of tabular entry diagrams. These checks can be considered to be of two types. Those checks which involve specification constructs within a single viewpoint and those which involve constructs which span viewpoints.

Within viewpoints Within viewpoints, the following checks can be made in order to determine whether the requirements for that viewpoint diagram are consistent and complete.

Rule 1 All inputs have a possible source (either statically assigned or dynamically determined)

$$(\forall t,i).input(t,i) \supset ((\exists s).has-source(t,i,s) \wedge source(t,s)) \vee ((\exists t_1).output(t_1,o) \wedge i = o) \tag{13}$$

Rule 2 All outputs have a possible destination (either statically assigned or dynamically determined)

$$(\forall t,o).output(t,o) \supset ((\exists d).has-destination(t,d,s) \wedge destination(t,d)) \vee ((\exists t_1).input(t_1,i) \wedge o = i) \tag{14}$$

Rule 3 All actions have at least one input or one output.

$$(\forall t, a).action(t, a) \supset$$
$$(((\exists i).has-input(t, a, i) \wedge input(t, i)) \vee$$
$$((\exists o).has-output(t, a, o) \wedge output(t, o))) \quad (15)$$

Rule 4 All items are correctly connected by lines.

See axioms 6 through 9.

Across Viewpoints Across viewpoints, the following checks can be made in order to determine whether the requirements for the entire set of viewpoint diagrams are consistent and complete.

Rule 5 All inputs must appear as outputs on the given sources. This source must also agree that this data is either being passed directly to the T or to an ancestor of the T.

$$(\forall t_1, i, s).input(t_1, i) \wedge has-source(t_1, i, s) \supset$$
$$(\exists t_2, o, d).t_2 = s \wedge output(t_2, o) \wedge i = o \wedge$$
$$has-destination(t_2, o, d) \wedge$$
$$((d = t_1) \vee (ancestor-of(d, t_1))) \quad (16)$$

Rule 6 All outputs must appear as inputs on the given destinations. This destination must also agree that this data is either being passed directly from the T or from an ancestor of the T.

$$(\forall t_1, o, d).output(t_1, o) \wedge$$
$$has-destination(t_1, o, d) \supset$$
$$(\exists t_2, i, s).t_2 = d \wedge input(t_2, i) \wedge o = i \wedge$$
$$has-source(t_2, i, s) \wedge$$
$$((s = t_1) \vee (ancestor-of(s, t_1))) \quad (17)$$

At any point during the viewpoint decomposition phase, a CPM toolset user may run the consistency checks given the representation and rules described above. Errors in the specification are presented to the user which include the type of error and the diagram(s) on which they occur. An example of this output is shown in Figure 7.

Related Work[5]

There has been a gradual move within AI planning research toward a more disciplined, rigorous approach to

[5]Some parts of this review were contributed by Peter Jarvis, Artificial Intelligence Applications Institute (AIAI) which appeared in a joint paper (Tate, Polyak, & Jarvis 1998).

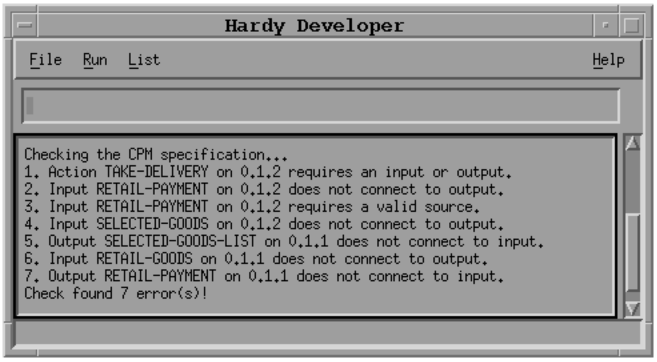


Figure 7: CPM Toolset Reported Errors

applied knowledge engineering. Work on knowledge-level analyses of planning systems (Valente 1995; Cottam *et al.* 1995; Barros, Valente, & Benjamins 1996; Kingston, Shadbolt, & Tate 1996) have provided researchers with more insight into the type of knowledge required and manipulated by AI planners. Recently, a mini-issue of the International Journal of Human-Computer Studies presented a sampling of these works (Benjamins & Shadbolt 1998). Continuing examples of this engineering trend were presented at a workshop held at the biannual Artificial Intelligence Planning Systems 1998 (AIPS '98) conference. The "Knowledge Engineering and Acquisition for Planning: Bridging Theory and Practice" workshop featured cross-discipline research taken from software engineering, knowledge acquisition, design rationale, object-oriented analysis, and machine learning communities (Nunes de Barros *et al.* 1998).

Each of these approaches seeks to address the difficulties encountered in various ways. As expected there are benefits and tradeoffs to be considered. For example, in one application of machine learning (Wang 1996) the researchers take a set of example plans described in terms of the actions in each plan and the state of the world before and after each action. The system then examines these examples and generates preconditions and effects of operator descriptions. The technique assumes that the user can provide example plans described in terms of the state of the world before and after each action. Unfortunately, it provides no assistance for the construction of these example plans. In addition, the technique is only applicable to STRIPS style planning not HTN.

Similar to CPM, another example is aimed at providing domain analysis techniques and tools (Chien 1996). This work describes two types of software engineering tools for planning knowledge base development: static KB analysis techniques to detect certain classes of syntactic errors and completion analysis techniques to iteratively debug the planning knowledge base. The tool set supports typical user questions when investigating these types of error. One problem with this approach though is that can only be effectively used after a significant portion of a domain description has been elicited. It doesn't directly address how this initial description is to be constructed.

Other interesting possibilities are related to adaptations from object-oriented analysis work. For example, in an object-centred specification approach (McCluskey & Porteous 1997), the authors seek to provide support for constructing planning domain descriptions by adapting methodological steps and notations of the object-oriented community. This approach utilises the notion of "lifting" domain representation from the level of the literal to the level of the object. Once a domain has been described in terms of a state transition graph, the author's algorithms compile the diagram into a STRIPS style action representation. There are also a few issues here as well. This work assumes that a domain can be described as a state transition graph (STG). Current work is ongoing to extend this to include techniques which use hierarchies of STGs (McCluskey & Kitchin 1998).

A number of other important contributions from AI planning, Requirements and Software Engineering, and KBS communities may offer possibly complementary ideas. The construction of models for a problem domain has been recognized by the KBS community as an important component in the overall task of knowledge acquisition for expert systems relative to some specific problem-solving framework (Davis & Bonnell 1991). These works include architectures, such as the EXPECT knowledge acquisition architecture (Swartout & Gil 1996) which dynamically forms expectations about the knowledge that needs to be acquired by the system and then uses these expectations to interactively guide the user through the knowledge acquisition process. Earlier we also mentioned the KEATS (Motta *et al.* 1991) toolset which can be seen as a precursor to the VITAL workbench which aims to provide methodological and software support for developing large, embedded KBS applications (Domingue, Motta, & Watt 1993).

There are are also specialised techniques, for example: knowledge acquisition on the fly (i.e. during planning) (desJardins 1996)[6]; adapting generic "design patterns" for specialising planners (Yang, Fong, & Kim 1998); using object abstraction (Chang, Kannan, & Wong 1991); various computer-supported strategies for addressing conflicts in domain specifications (Easterbrook 1991); incorporating risk management in specifications (Bustard, Greer, & Tate 1994); and tools for editing operators and domain knowledge (e.g. Act editor (Myers & Wilkins 1997), Operator editor (desJardins 1996), Task Formalism Workstation (Tate & Currie 1984; 1985), HARDY-based Task Formalism Editor, etc.).

[6]Both Nonlin (Tate 1977) and its predecessor, Interplan, also allowed a user to type in missing parts of the domain model when the planner detected that no TF schema was present to address its current needs

Conclusions

In this work, we sought to provide a more disciplined approach to producing domain specifications which would encompass requirements capture, analysis and domain requirements engineering. The Common Process Methodology has taken a step toward this goal. In order to evaluate the success of this approach, we turn to a set of compulsory guidelines which were produced to evaluate a requirements engineering methodology (Sommerville & Sawyer 1997). The following set of guidelines were suggested by the authors as being necessary for any organisation engaging in requirements engineering.

1. *Define a standard document structure*

   CPM provides a standard document structure for engineering domain specifications based on the diagrams and notations derived from the CORE methodology.

2. *Make the document easy to change*

   The largely graphical nature of the documentation along with automated support from the CPM toolset for managing and navigating the documents make changes relatively easy to execute.

3. *Uniquely identify each requirement*

   Individual requirements can be traced to the particular diagram in which some knowledge about the domain was expressed, modified or constrained in some way. Greater tool support should be added to provide search functionality for related requirements over all the of diagrams. In addition, we may wish to separate those requirements which came from experts or specialists vs. those which were synthesized due to other constraints.

4. *Define policies for requirements management*

CPM takes a hands-off approach to this issue. CPM and the corresponding toolset provide the general methods and functionality for engineering the initial domain specification, but defer many of the policy decisions to an organisation's particular implementation.

5. *Define standard templates for requirements description*

   The CPM toolset provides the standard templates based on the appropriate CPM diagrams.

6. *Use language simply, consistently and concisely*

   Again, the largely graphical nature of the documentation supports a simple and concise expression of the requirements. The toolset also enforces a consistent expression throughout the various phases of development.

7. *Organise formal requirements inspections*

   CPM recommends these activities but assumes that the structure and frequency of these detailed activities are related to an individual organisation's requirements.

8. *Define validation checklists*

   CPM encodes CLIPS-based expert system checklists for validating specification diagrams.

9. *Use checklists for requirements analysis*

   The automated requirements analysis enacted by CPM is directly based on CPM's internal representation of the requirements and the CLIPS-based checklists described above.

10. *Plan for conflicts and conflict resolution*

    Certain techniques such as VVRP and IVRP along with the automated detection of such conflicts provides support for focusing the discussion on these issues outstanding in the domain specification.

Work will continue to explore the feasibility of this methodology for building process domains. We hope to offer more automated support for the detailed aspects of building and analysing the domain. We are encouraged by our initial results and we believe the benefits of this approach will outweigh the overhead incurred in creating the intermediate work products. This approach will hopefully continue in the trend toward a more disciplined, robust transference of AI planning to applied settings.

Acknowledgements

The 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 F30602-97-1-0022. 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,AFOSR or the U.S. Government.

Thanks to Austin Tate and Peter Jarvis for reviewing the paper and for their helpful comments and suggestions.

References

Aarup, M.; Arentoft, M.; Parrod, Y.; Stader, J.; Stokes, I.; and Vadon, H. 1994. OPTIMUM-AIV: A knowledge based planning and scheduling system for spacecraft AIV. In Fox, M., and Zweben, M., eds., *Knowledge Based Scheduling*. Morgan Kaufman.

Arentoft, M.; Parrod, Y.; Stader, J.; Stokes, I.; and Vadon, H. 1991. OPTIMUM-AIV: A planning and scheduling system for spacecraft AIV. *Telematics and Informatics* 8(4):239–252.

Barros, L.; Valente, A.; and Benjamins, R. 1996. Modeling planning tasks. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, 11–18. Edinburgh, Scotland: Morgan Kaufmann.

Benjamins, V., and Shadbolt, N. 1998. Preface: Knowledge acquisition for planning. *International Journal of Human-Computer Studies* 48(4):409–416.

Bustard, D.; Greer, D.; and Tate, G. 1994. Enhancing the soft systems methodology with risk management techniques. In *Proceedings of the 2nd International Conference on Software Quality Management*, 145–157.

Chang, A.; Kannan, P.; and Wong, B. 1991. Design of an object-oriented system for manufacturing planning and control. In *Proceedings of the Rensselar's 2nd International Conference on Computer Integrated Manufacturing*.

Chien, S. 1996. Static and completion analysis for planning knowledge base development and verification. In Drabble (1996), 53–61.

Cottam, H.; Shadbolt, N.; Kingston, J.; Beck, H.; and Tate, A. 1995. Knowledge level planning in the search and rescue domain. In *Research and Development in Expert Systems XII, proceedings of BCS Expert Systems'95.*

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

Curwen, P. 1991. System development using the CORE method. Military Aircraft Ltd. BAe/WIT/ML/GEN/SWE/1227, British Aerospace, PLC, Warton Aerodrome, Preston, UK.

Davis, J., and Bonnell, R. 1991. A framework for constructing visual knowledge specifications in acquiring organizational knowledge. *Knowledge Acquisition* 3(1):79–115.

DeMarco, T. 1978. *Structured Analysis and System Specification.* New York: Yourdon Press.

desJardins, M. 1996. Knowledge acquisition tools for planning systems. In Tate (1996), 53–61.

Domingue, J.; Motta, E.; and Watt, S. 1993. The emerging VITAL workbench. In *Proceedings of the 7th European Knowledge Acquisition for Knowledge-Based Systems Workshop (EKAW'93).*

Drabble, B. 1990. Mission scheduling for spacecraft: Diaries of T-Sched. In *Expert Planning Systems*, 76–81. Institute of Electrical Engineers.

Drabble, B., ed. 1996. *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96).* Edinburgh, Scotland: Morgan Kaufmann.

Drummond, M., and Tate, A. 1992. PLANIT interactive planner's assistant – rationale and future directions. AIAI AIAI-TR-108, University of Edinburgh.

Easterbrook, S., and Nuseibeh, B. 1996. Using viewpoints for inconsistency management. *Soft. Engin. Journ.* January.

Easterbrook, S. 1991. Handling conflict between domain descriptions with computer-supported negotation. *Knowledge Acquisition* 3(3):255–289.

Erol, K. 1995. *Hierarchical Task Network Planning: Formalisation, Analysis, and Implementation.* Department of Computer Science, University of Maryland, College Park, USA.

Finkelstein, A.; Gabbay, D.; Hunter, A.; Kramer, J.; and Nuseibeh, B. 1994. Inconsistency handling in multi-perspective specifications. *Trans Software Eng* 20(8):569–578.

Fuchs, J.; Gasquet, A.; Olalainty, B.; and Currie, K. 1990. PlanERS-1: An expert planning system for generating spacecraft mission plans. In *First International Conference on Expert Planning Systems*, 70–75. Brighton, United Kingdom: Institute of Electrical Engineers.

Giarratano, J. 1994. CLIPS 6.0 user's guide. Software Technology Branch JSC-25013, Lyndon B. Johnson Space Center, Information Systems Directorate.

Jackson, M. 1975. *Principles of Program Design.* New York, USA: Academic Press.

Johnston, A., and Adorf, A. 1992. Scheduling with neural networks: The case of the hubble space telescope. *Computers and Operations Research* 19(3–4):209–240.

Kingston, J.; Shadbolt, N.; and Tate, A. 1996. CommonKADS models for knowledge based planning. Artificial Intelligence Application Institute AIAI-TR-199, University of Edinburgh, Edinburgh, Scotland.

Kotonya, G., and Somerville, I. 1996. Requirements engineering with viewpoints. *Soft. Engin. Journ.* 11(1).

Langley, P., and Drummond, M. 1990. Toward an experimental science of planning. In Sycara, K., ed., *Proceedings workshop on innovative approaches to planning and scheduling approaches*, 109–114. San Diego, CA: Morgan Kaufmann Publishers, ISBN 1-55860-164-3.

Malone, T.; Crowston, K.; Lee, J.; and Pentland, B. 1993. Tools for inventing organizations: Towards a handbook of organizational processes. In *Proceedings of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises.*

Mayer, R.; Cullinane, T.; deWitte, P.; Knappenberger, W.; Perakath, B.; and Wells, M. 1992. Information integration for concurrent engineering (IICE) IDEF3 process description capture method report. Technical Report AL-TR-1992-0057, Armstrong Laboratory, Logistics Research Division, Wright-Patterson AFB, OH 45433 USA.

McCluskey, T., and Kitchin, D. 1998. OCLh: An object-centred language for HTN planning. School of computing and mathematics, University of Huddersfield, Huddersfield, UK, Submitted to ICTAI '98.

McCluskey, T., and Porteous, J. 1997. Engineering and compiling planning domain models to promote validity and efficiency. *Artificial Intelligence* 95(1):1–65.

Motta, E.; Rajan, T.; Domingue, J.; and Eisenstadt, M. 1991. Methodological foundations of KEATS, the knowledge engineer's assistant. *Knowledge Acquisition* 3(1):21–47.

Mullery, G. 1979. CORE: A method for controlled requirements specification. In *Proceedings of the 4th International Conference on Software Engineering*.

Myers, K., and Wilkins, D. 1997. The Act-Editor user's guide: A manual for version 2.2. SRI International Artificial Intelligence Center, Stanford University, Menlo Park, CA.

Nunes de Barros, L.; Benjamins, R.; Shahar, Y.; Tate, A.; and Valente, A. 1998. Workshop on knowledge engineering and acquisition for planning: Bridging theory and practice. AIPS '98 AAAI Technical Report WS-98-03, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Polyak, S., and Aitken, S. 1998. Manufacturing process interoperability scenario. Artificial Intelligence Applications Institute AIAI-PR-68, University of Edinburgh, Edinburgh, Scotland.

Polyak, S. T. 1998. A supply chain process interoperability demonstration using the process interchange format (PIF). Department of Artificial Intelligence Report Number 889, University of Edinburgh, Edinburgh, Scotland.

Sacerdoti, E. 1975. The nonlinear nature of plans. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-75)*, 206–214.

Sommerville, I., and Sawyer, P. 1997. *Requirements Engineering: A Good Practice Guide*. John Wiley and Sons.

Swartout, W., and Gil, Y. 1996. EXPECT: A user-centered environment for the development and adaptation of knowledge-based planning aids. In Tate (1996), 250–258.

Tate, A., and Currie, K. 1984. The O-Plan Task Formalism Workstation. Artificial Intelligence Applications Institute (AIAI) AIAI-TR-7, University of Edinburgh.

Tate, A., and Currie, K. 1985. The O-Plan task formalism workstation. In *Proceedings of the Third Workshop of the UK Alvey Programme's Planning Special Interest Group*. London, UK: Institute of Electrical Engineers.

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

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

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

Tate, A. 1993. Putting knowledge-rich plan representations to use. In *Papers of the 14th Machine Intelligence Workshop*.

Tate, A., ed. 1996. *Advanced Planning Technology: Technological Advancements of the ARPA/Rome Laboratory Planning Initiative*. Menlo Park, CA: AAAI Press.

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

Valente, A. 1995. Knowledge-level analysis of planning systems. SIGART *Bulletin* 6(1).

Wang, X. 1996. Planning while learning operators. In Drabble (1996), 229–236.

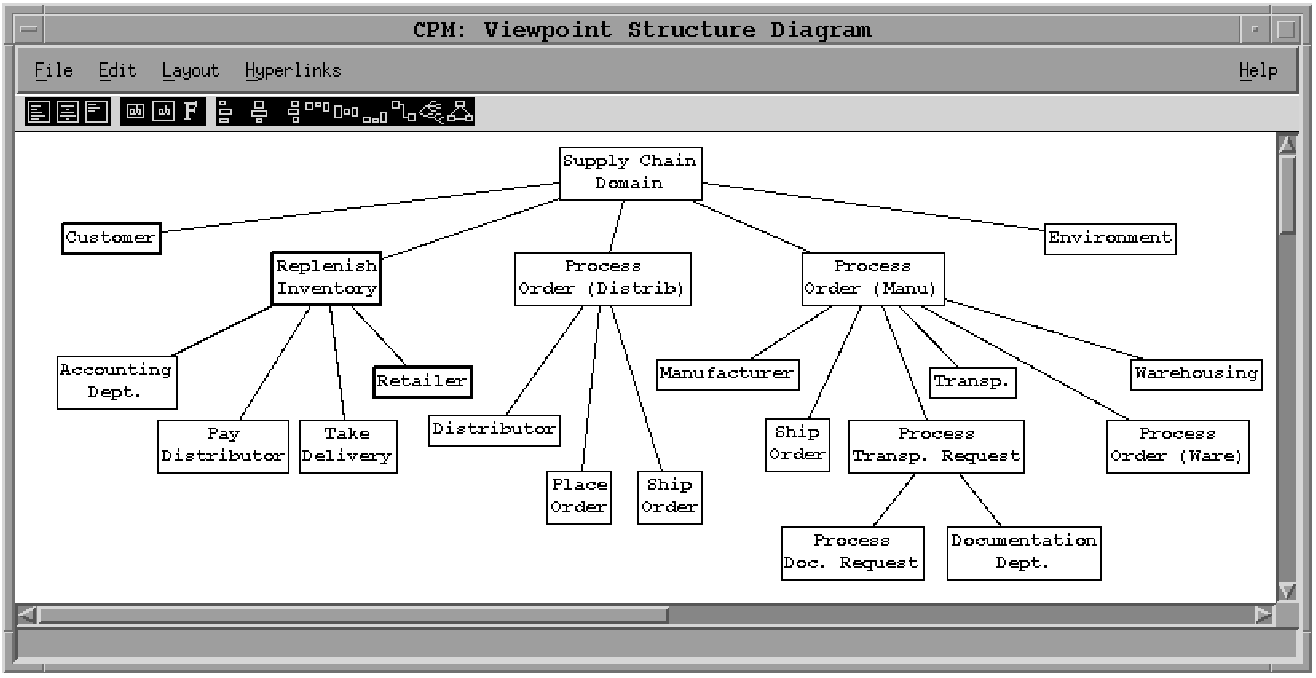
Wilkins, D. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann.

Wilson, A. 1984. *Information for Planning*. M.Sc. Thesis, Department of Artificial Intelligence, University of Edinburgh, UK.

Yang, Q.; Fong, P.; and Kim, E. 1998. Design patterns for planning systems. AIPS '98 Workshop on Knowledge Engineering and Acquisition for Planning: Bridging Theory and Practice AAAI Technical Report WS-98-03, Carnegie Mellon University, Pittsburgh, Pennsylvania.

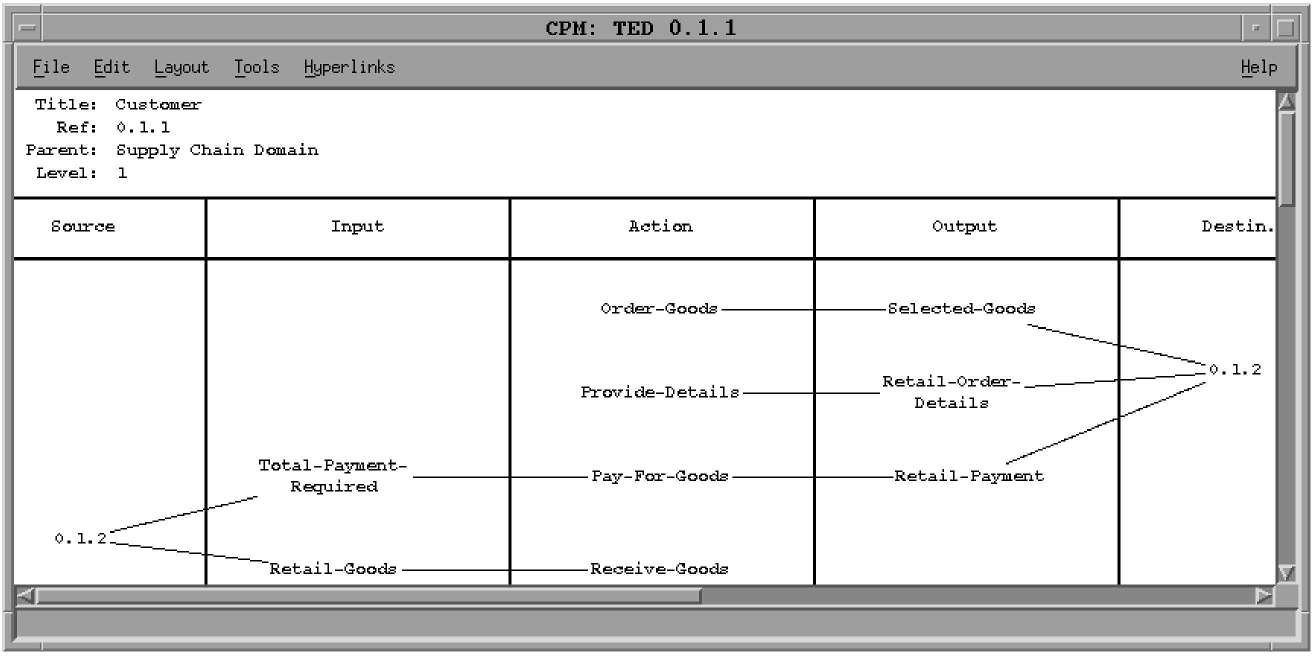
A. Viewpoint Structure Diagram

This is a screen shot taken from the CPM toolset of an initial Viewpoint Structure Diagram (VSD) which provides the framework for eliciting and analysing the detailed domain requirements. The three highlighted viewpoints are utilised in the examples found in this paper.



B. Tabular Entry Diagram

This is a screen shot taken from the CPM toolset of a Tabular Entry Diagram (TED) for a customer viewpoint which provides interface activities associated with this bounding functional viewpoint.



C. Data Composition Notation

This notation was originally developed in (DeMarco 1978). Within CPM, it provides a way for analysts to describe the composition of data elements which are manipulated by the domain processes. A summary of the symbols used and an example are shown below.

Notation Table

Symbol	Meaning
=	is equivalent to
+	and
/	separates alternatives
[ ]	encloses alternatives
{ }	encloses repeating elements or structures or both
m{	specifies the minimum number of repeating elements, structures or both
}n	specifies the maximum number of repeating elements, structures or both
!	key component identifier

Example from the Supply Chain Domain

```
Retail-Sales-Order =
 {Order Number!+
  [Total-Cost-Pounds/Total-Cost-Dollars]+
  Customer-Number+ Sale-Date+ Sale-Time+
  1{Quantity+ Cost-Per-Unit+
    Product-Number+ Line-Item-Cost}250}
```