Manufacturing Knowledge Verification Design Support Systems

Sean David Cochrane, R I M Young, K Case, J A Harding, Shilpa Dani, James X Gao, David Ian Baxter

Manufacturing Knowledge Verification Design Support Systems

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2006-IJPR-0758.R2
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	13-Jul-2007
Complete List of Authors:	Cochrane, Sean; Loughborough University, Wolfson School Young, R I M; Loughborough University, Wolfson School of Mech and Man Engineering Case, K; Loughborough University, Wolfson School of Mech and Manufacturing Engineering Harding, J A; Loughborough University, Wolfson School of Mech and Man Engineering Dani, Shilpa; Loughborough University, Wolfson School of Mech and Man Engineering Gao, James X; Cranfield University, School of Industrial and Manufacturing Science Baxter, David; Cranfield University, Department of Manufacturing
Keywords:	PROCESS PLANNING, SIMULATION, RE-USE, PRODUCT PLANNING, PROCESS MODELLING, MANUFACTURING PROCESSES, FACILITY PLANNING, ENTERPRISE MODELLING, DISTRIBUTED RESOURCE PLANNING
Keywords (user):	

Manufacturing Knowledge Verification in Design Support Systems

Sean Cochrane[a], Robert Young[a], Keith Case[a], Jennifer Harding[a], James Gao[b], Shilpa Dani[a], David Baxter[c]

a Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Leicestershire, LE11 3TU, UK

b School of Engineering, University of Greenwich Chatham Maritime, Kent, ME4 4TB, UK

c Centre for Decision Engineering, Manufacturing Department, Cranfield University, Bedfordshire, MK43 0AL, UK

Abstract

This paper identifies the need for a *verification methodology* for manufacturing knowledge in design support systems; and proposes a suitable methodology based on the concept of *ontological commitment* and the PSL ontology (ISO/CD18629). The use of the verification procedures within an overall system development methodology is examined, and an understanding of how various categories of manufacturing knowledge (typical to design support systems) map onto the PSL ontology is developed. This work is also supported by case study material from industrial situations, including: the casting and machining of metallic components. The PSL ontology was found to support the

verification of most categories of manufacturing knowledge, and was shown to be particularly suited to process planning representations. Additional concepts and verification procedures were however needed to verify relationships between products and manufacturing processes. Suitable representational concepts and verification procedures were therefore developed, and integrated into the proposed knowledge verification methodology.

Key Words: Knowledge Representation, Validation and Verification, Enterprise Modelling, Ontology, Design for Manufacture.

1. Introduction

Many organisations use information systems to support decision makers in design and manufacture (Young 2003). These systems (i.e. *design support systems)* often use models of products and manufacturing processes to predict (and therefore avoid) manufacturing issues in design. This paper describes a knowledge verification methodology for such systems that has the ultimate aim of reducing the time and energy needed to represent manufacturing knowledge. The scope of the methodology is (at this stage) limited to existing types of design support system, and the test cases described by the paper are based on typical support functions (principally the simulation of process plans derived). The following sections outline current research into system development methodologies, and describe why a manufacturing knowledge verification methodology is required.

System development methodologies generally follow the stages of elicitation and representation. *Elicitation* refers to the learning, uncovering, extracting, surfacing, and/or discovering the needs of customers, users, and other potential stakeholders (Hickey and

Davis, 2002). During elicitation, a systems engineer must understand the end user's requirements and elicit product and process knowledge from the appropriate experts. Elicitation is often complicated by the fact that an expert's knowledge may be *implicit* (i.e. can not be easily described). The elicitation process leads to the further task of *representation.* This usually starts with an *informal* description of elicited knowledge and user requirements, followed by further stages of *structuring* and *formalisation*. The end result of the formalisation process is a computer executable representation of expert knowledge that can be used to evaluate designs (according to the requirements elicited from the designer).

Knowledge validation and verification are closely related to the tasks of elicitation and representation. *Validation* involves making the right system, and *verification* involves making the system right (O'Keefe and O'Leary, 1993). Validation must therefore ensure that a system meets the requirements of end-users, and can therefore be seen as part of the elicitation process. Verification ensures that a system meets its specified requirements (Preece, 2001), and is therefore part of representation (or more specifically, formalisation). Within a design support system, verification must therefore ensure that any rules and constraints used to support decisions are at least consistent. *Indeed, from any contradictory knowledge, an agent would be able to deduce any conclusion, and it's contrary* (Gregoire and Mazure 2002).

Models for representing manufacturing process knowledge have been extensively discussed in the research literature since the 1990's. Sormaz and Khoshnevis (1997) for example, provide an object-oriented knowledge representation of process planning knowledge, and Sormaz et. al. (2004) describes the interaction between process planning

models and product design activities. Other work has focussed on developing modelling hierarchies for product and manufacturing information (e.g. Oldham et. al. 1998, Molina and Bell 1999, and Zhao et. al. 2000). Molina and Bell (1999) for example, developed an object-oriented model for manufacturing knowledge based on the concept of a *manufacturing strategy*. Strategies were themselves categorised according to four types, i.e. planning, capacity, technology, and facility. *Planning* strategies describe rules for creating and manipulating process plans, *capacity* strategies describe how many units can be produced by a facility, *technology* strategies interpreted the information associated with facilities (e.g. machining tolerances), and *facility* strategies described how and when facilities should be used to achieve manufacturing objectives.

Several reference models (or *frameworks*) have also been developed to assist system development, including: CommonKADS (Schreiber et. al. 1999), CIMOSA (Kosanke et. al. 1999), and the Reference Model for Open Distributed Processing (ISO/IEC 10746-1). A framework specifically for design support systems (referred to as the CAE-RM) was also developed by Molina and Bell (2002). All of these frameworks apply some form of information view, where knowledge is classified according to a predefined hierarchy.

More recent research has focused on the representation of globally distributed supply chains (Liu and Young, 2004), and improved knowledge sharing between design teams using ontologies (Lin et. al. 2004). Various computational techniques have also been used to improve the process planning capabilities of decision support systems, including artificial intelligence techniques (Fernandez et. al., 2005), multi-agent techniques (Pechoucek et. al 2005), and simulated annealing algorithms (Bramall et. al. 2003).

Recent work focussed on the representation of process plans has also been performed by

Bock and Gruninger (2004)b. This shows how a formal ontology, i.e. the Process Specification Language (PSL - ISO/CD18629) can be used to describe process plans. Knowledge verification has also been tackled from a general "computer science" perspective. Plant and Gamble (2003) show how verification can be supported within a system development framework (sometimes referred to as a meta-knowledge framework); and several techniques for identifying inconsistencies in formal knowledge representations (e.g. contradictory rules, and unreachable conditions) have been identified (Preece et. al. 1992, Wu and Lee 2002, and Gregoire and Mazure 2002). An approach described as *ontological commitment* (Waterson and Preece, 1999) is also described.

Ontologies provide a set of rules and constraints associated with a class schema for describing an environment (Smith et. al. 2003). Ontological commitment means that a knowledge base complies with the rules and constraints associated with the ontology. The interaction between several knowledge bases can also be verified against a shared ontology. Bock and Gruninger (2004)b show how process plans can be represented by a formal ontology (i.e. PSL). They do not however show how the semantics of the PSL ontology could be applied within a complete system development methodology (that includes knowledge verification). Section 2 of this paper describes such a methodology.

2. The Knowledge Verification Methodology

Methodology Overview

This section outlines a *Knowledge Verification Methodology* (KVM) that incorporates the concepts of ontological commitment and the PSL ontology. The KVM is part of a meta-knowledge framework for developing design support systems. The framework provides a series of views (i.e. enterprise, information, and computational) which are based on the CAE-RM (Molina and Bell, 2002). The framework's novelty lies in its improved application of standards (i.e. the ISO/IEC 10746-1), and its direct support for manufacturing knowledge verification. Figure 1 shows the main stages of the framework, and its role within a wider system development methodology.

Figure 1: System Development Methodology

The enterprise view describes how to informally represent user requirements and expert knowledge, and the enterprise model is structured according to the hierarchy provided by the information view. The information model is in turn formalised using the guidelines and definitions provided by the computational view. The resulting computational model can then be used by a decision support system to simulate manufacturing strategies.

Any model inevitably makes assumptions (and simplifications). The question therefore arises as to how detailed a description is required, and whether a valid set of assumptions has been established. This is a particularly difficult part of knowledge elicitation, as product and manufacturing experts can often not directly state these assumptions (as they are understood implicitly, rather than laid down by any explicit set of guidelines). This research therefore follows the iterative model of software development, as discussed by

Hickey and Davis (2002). This results in a series of successively more sophisticated descriptions.

Figure 2: The Meta Knowledge Framework

The iterations effectively stop when agreement is reached between end users and experts on the validity of the model (and supporting test cases). The iterations are therefore controlled by the validation stage (shown in figure 1), which reviews the enterprise model and verified simulation results with designers and experts. It should be noted that for the purposes of this research, the iterations stop short of defining a full system requirement. The experimental case study described in section 5 provides a proof of concept for the KVM, and the results of the first pass validation stage are discussed in section 6.

The KVM is based on the the well established use of t*est cases*, i.e. derived solutions from domain experts (Giarratano and Riley, 1998). Test cases are elicited from domain experts, and reviewed to ensure that they accurately and adequately describe the required system behaviour (i.e. they validate the system). Each test case describes the information that an expert expects the system to generate for given sets of product requirements. Verification is achieved by comparing these expected results with the actual output of the executable model. The Computational View provides specific support for this validation in process. A more detailed description of the enterprise, information and computational views is shown in figure 2, and is discussed in the following subsections.

The scope of the methodology does not at this stage include advanced techniques for generating and optimising process plans (such as those developed by Bramall et. al. 2003), but relies instead on the manual mapping of structured informal representations of manufacturing knowledge onto a computational model.

The Enterprise View

The enterprise view applied by this research is well documented by previous researchers, notably Molina and Bell (2002). The use of diagramming formats for enterprise views is also described by Dorador and Young (2000), and Costa et. al. (2001). These papers describe how IDEF0, IDEF3, UML use case, and UML sequence diagrams can be used to provide an informal representation of user requirements, and manufacturing strategies. These are used as part of the enterprise view, in the following ways:

1 Use case and sequence diagrams are first used to describe designer requirements, and their interactions with the decision support system. These representations follow well-documented procedures for the UML diagramming conventions (Quatrani, 1998).

2 An informal description of the product(s) being designed, including a hierarchical breakdown of features (e.g. slots, holes, and grooves), and how they combine to form products.

3 A description of how products are made, including a breakdown of manufacturing strategies. IDEF3 process and object schematics assist this description, and this is structured according to the manufacturing model described by the information view.

The Adapted Product/Manufacturing Model

The information view used by this research is shown in figures 3a and 3b. These are based on the product and manufacturing models proposed by Molina and Bell (1999) and Zhao et. al. (2000). Figure 3a shows how facilities describe an aggregation of strategies that use resources to perform processes. In this way, the manufacturing model represents process hierarchies, e.g. machine tools performing drilling processes, but also describes

strategies, for deploying processes. The adaptations made to the product and manufacturing models used by this research are described below.

Firstly, figure 3b shows how the Structure class aggregates geometric characteristics and enumerated properties. Product features such as holes, cylinders, blocks, and threads, are represented by specialisations of the Structure class (note that these are not shown in figures 3a or 3b). A cylinder for example, can be represented by a particular mix of geometries, e.g. diameter, depth, and surface tolerance; and enumerated properties, e.g. "material: iron or aluminium". Methods for deriving geometric properties (e.g. area and volume) are also supported by the Structure class, and these are polymorphed by each Feature specialisation. Figure 2 shows the role of the feature library in the computational model. This stores a range of features such as holes, cylinders, blocks, and threads. The description of each feature is derived where possible from the STEP standard ISO/DIS 10303-224.3).

Component can be described as an aggregation of features (e.g. a bolt consisting of a metallic cylinder, and a thread). The concepts of atomic and complex components are also introduced. Atomic components describe the lowest (atomic) level of component found in a product description (e.g. work pieces, plus nuts, bolts, and screws). Complex components describe aggregations of components (e.g. assemblies of work pieces).

Figure 3a: Adapted Product/Manufacturing Model

Figure 3b: Adapted Product/Manufacturing Model

Secondly, previous manufacturing models have used several types of facility class to represent different levels of facility, e.g. enterprise, factory, shop and machine classes. These are replaced in figure 3 with a simpler hierarchy based on just complex and atomic

facilities. As with structures, atomic facilities represent the lowest level of granularity in the representation of facilities, and complex facilities represent aggregations of facilities. As complex facilities are also a specialised form of atomic facility, they can also deploy strategies of their own. As a general guideline each facility representation should make no assumptions about the availability of other facilities, unless they are an aggregated sub-facility (i.e. part of the complex model). A machine shop can for example be considered as an aggregation of machine tools (and is therefore a complex facility). The machine shop can also be considered as a facility in its own right, with strategies that cannot be described at the level of individual machine tools. For example strategies for selecting the "best" machine tool for a particular task cannot be described at the level of an individual machine tool (which has no knowledge of other machine tools to make an effective comparison). A machine tool model can be considered as atomic (if this is the lowest level of granularity selected by the modeler), but may also be broken down into further atomic elements (e.g. individual cutting tools, and fixtures). Again this level of granularity needs to be selected on a case by case basis, but is not limited here by the manufacturing model.

Thirdly, (as with previous manufacturing models) strategies are performed by facilities. Each strategy must however hold an objective describing what it does within the manufacturing environment (e.g. improve the tolerance of a metallic surface). This adaptation is derived from intelligent agent theory (Arazy and Woo, 2002), and allows strategies to be translated into executable modules (or agents). Strategies are further broken down into capacity, planning, technology and facility rules. These four sub-classes of rules are based on the categories of manufacturing strategy described by

Molina and Bell (1999). Note that strategies (in the hierarchy shown above) are effectively aggregations of rules.

Capacity rules describe how resources are consumed and demanded. Planning rules describe how strategies relate to production schedules. Technology rules describe how strategies change product characteristics and are constrained by the manufacturing environment. Facility rules describe when a particular strategy should be applied (note that these should only be applied to compound facilities, as atomic facilities can not make assumptions about other facilities within the manufacturing environment).

Finally, figure 3b shows interfaces to CAD and MIS platforms. These provide the information relevant to product representations (e.g. dimensions and required tolerances), and manufacturing strategies (e.g. machining times and achieved tolerances).

The Computational View and the Role of the SM-API

The computational view provides a set of methods for representing manufacturing strategies that directly assist the verification of computational models. Previous frameworks for decision support systems (including the CAE-RM) invariably rely on a system engineer's implicit understanding of the terms used to describe manufacturing strategies to generate computational models. The systems engineer is also left to develop his own implicit understanding of what constitutes an inconsistency between statements in a knowledge base. This implicit "system engineering knowledge" drives the interpretation of the enterprise and information models during formalisation and verification; and can be the source of miss alignments between shared knowledge bases. The KVM tackles this issue by providing an enhanced computational view as part of the meta-knowledge framework shown in figure 2.

The enhanced computational view includes a set of methods for formally representing manufacturing strategies, referred to as the Shared Methods – Application Programming Interface (SM-API). The SM-API is implemented as a class, and once instantiated this class provides an object that can be used to represent the manufacturing environment being modelled. Statements about the environment are made via a set of method (implemented by the SM-API class). These provide a convenient interface for systems engineers who are familiar with object-oriented programming languages and the concept of an API. The SM-API also allows systems engineers to model manufacturing environments using widely available coding and simulation tools for languages such as Java and C++.

The SM-API is described as "shared" because it provides a common interpretation (between systems engineers) of the terms used to formalise manufacturing strategies. The shared methods also include explicitly defined procedures for detecting inconsistencies between statements in knowledge bases. This allows shared knowledge to be verified against a common set of procedures, rather than relying on each system engineer's implicit understanding of exception handling within computational models.

Figure 4 shows the role of the SM-API within the verification process. Manufacturing strategies are expressed using the shared methods, and are organised according to the information hierarchy shown in figure 3, i.e. they are split between atomic and complex strategies. The overall objective of each strategy is to make the "simulated product model" match the "required product model" as closely as possible. The required product model represents the requirements specified by the designer, and the simulated product model represents the results of manufacturing strategy (and is changed by each process

simulation). This allows the effects of different manufacturing strategies to be evaluated for a range of product specifications. For the SM-API to support manufacturing strategy formalisation and verification, its methods must be able to describe the entities associated with products and manufacturing processes. It must also provide a means of extracting the information needed by designers (e.g. machining durations) and be able to generate relevant exceptions (i.e. error conditions). The SM-API is also responsible for generating the simulation results required by designers (see figure 1). These requirements will have been defined by the enterprise model (use case and sequence diagrams), and have been elaborated on by the test cases developed during knowledge elicitation and validation.

Figure 4: The Use of Test Cases to Support System Verification

It should also be noted that errors can be generated for a variety of reasons. Error conditions may for example highlight inconsistencies in manufacturing strategies. A strategy for improving the tolerance of a hole in a work piece may for example attempt to bore the hole before the hole has been created. The systems engineer would need to examine these error conditions, and correct the enterprise, information, and/or computational models accordingly. Other error conditions may be a valid system response, as defined by a test case (which should be examining boundary conditions where manufacturing strategies are not capable of meeting product specifications). The error conditions generated by the SM-API therefore provide input to the verification process (in terms of unexpected errors), and the validation process (in terms of verified simulation results and valid errors).

3. The Verification of Manufacturing Strategies

This section provides a more detailed examination of the four categories of manufacturing strategy identified by Molina and Bell (1999), i.e. planning, capacity, technology, and facility; and shows how these relate to the constructs provided by the PSL ontology. This analysis is then used to outline the requirements of the SM-API.

Manufacturing Strategy Categorisation

Planning rules describe the creation and manipulation of process plans for the manufacture of a product to a given specification, and enterprise/factory configuration. This knowledge can be used to estimate how long it takes to manufacture a product, and describes:

- Hierarchies of processes and sub-processes, e.g. drilling and milling are all sub-processes of machining.
- How processes should be sequenced, e.g. casting precedes machining, and setting must occur before a work piece can be milled.
- How to calculate the duration of a process. This is often a function of a processing rate and a geometric feature of a product.

Note that the process hierarchy is different from the hierarchy of atomic and complex facilities (shown in figure 3b). Facilities support an aggregation of strategies that use resources to perform processes. A machine tool may therefore support several strategies for drilling and milling (which form part of a wider process of machining). The machining process is also likely to be controlled by a complex facility model of a machine shop. This will aggregate rules for selecting between individual machine tools, describing how the overall machining process can be optimised.

Certain levels of planning knowledge will also be relevant to different levels of facility representation. For example, a model of an individual machine tool can describe constraints on the processes under its control (e.g. setting is required before milling), but can not assume knowledge of other facilities. A constraint on "casting preceding machining" must for example be described by a factory or enterprise level model, which makes assumptions about the availability of foundries and machine tools. This allows the machine tool model (on its own) to be reused in environments using forges and other fabrication technologies.

Capacity rules describes how many units can be produced by a facility given the availability of resources. This includes an understanding of how long each process takes, the resources it requires, and the number of products it creates/alters. For example, a milling process requires the use of a machine tool, and operates on a single work piece at a time. A casting process on the other hand, requires a foundry, but may produce a batch of casts, in a single operation. Capacity knowledge also describes which processes are best for certain volumes of production. It may for example be better to produce a more accurate cast if high volumes are required, as this reduces the amount of machining needed to achieve the required tolerances, but can require a higher initial outlay.

Technology rules interpret the information associated with the resource and process classes described above. This may include:

- Rules for maximum and minimum part dimensions, e.g. the foundry cannot cast a part larger than 2m x 2m.
- Relationships between processes and tolerances (e.g. the surface tolerance after grinding equals 25μm after grinding, and the time it takes to grind a given area).

- Limits on which processes can be performed on different materials, and different tolerance capabilities for different materials.

It should also be noted that planning strategies often refer to (or make use of) technology rules. For example, the time it takes to grind a surface is often calculated as a function of the area being ground (as defined in the product requirement), and the capability of the machine tool (as described by the technology rules associated with the machine tool).

Facility rules describe how and when processes should be used to achieve manufacturing objectives. This may include the selection of processes, based on required tolerances, e.g. "grind a surface if the required tolerance can not be met by milling". As with planning constraints, facility knowledge needs to be appropriate to the level of facility being described. For example, a multi purpose machine tool may not be the best choice of facility if a product only requires a simple milling operation. The choice of whether to use such a facility can however, only be made at a level that understands what machine tools are available. Facility rules are therefore best described at the complex (rather than atomic) level of facility representation.

Commitment to the PSL Ontology

The SM-API requires a basis for describing entities within a product/manufacturing environment, and a set of guidelines for detecting inconsistent strategy representations (i.e. error conditions). This can effectively be performed by a suitable ontology. The introduction to this paper highlighted the PSL ontology (ISO/CD18629) as being potentially suitable for this type of environment. An open standard such as PSL may also provide better support for knowledge sharing than a proprietary ontology (that is not subject to the public scrutiny and control of the ISO). The SM-API therefore to a large

extent bases its definition of methods, and exception handling, on the PSL ontology. Each knowledge base formalised by the SM-API is effectively committed to the interpretation of the PSL ontology built into the SM-API. This follows the principle of ontological commitment described by Waterson and Preece (1999).

Figure 5 shows how the principle of ontological commitment is applied. Knowledge bases A and B respectively represent two manufacturing facilities. These however, state contradictory relationships between timepoints t1 and t2 (which may refer to the beginning or ending of processes). The sequence of checks specified by the PSL ontology (and implemented by the SM-API) highlights these conflicting statements as a violation of the second PSL core axiom, *i.e. the before relationship is a total ordering*. A systems engineer would then need to examine the circumstances that led to these conflicting statements, and adjust the computational model accordingly. This may for example refer to a process that has been scheduled too soon. Further details of how the PSL ontology can be used to represent and verify manufacturing strategies are described below.

Figure 5: Verification Using the PSL Ontology and the SM-API

Manufacturing Strategies and the PSL Ontology

Whilst other researchers (notably Bock and Gruninger, 2004)b have shown how manufacturing knowledge can be expressed by the PSL ontology, its use as a knowledge representation and verification tool within a meta-knowledge framework (for decision support systems) is new to this research.

The PSL ontology is centred on a core set of definitions that can be supplemented by the PSL outer-core and extensions (see figure 6). The core is relatively straightforward, and includes four entity types, i.e. activities, activity_occurrences, timepoints, and objects.

These can be used to represent processes as occurrences of activities over time.

The PSL ontology describes how timepoints can be ordered using the before relationship (see above), and places additional constraints on the before relationship, including transitive expansions, i.e. if "t1 is before t2" and "t2 is before t3", then "t1 is also before t3". Processes can therefore be represented as "occurrences" of activities that are bound by two timepoints (i.e. their beginning and ending). Constraints on process sequences can therefore be described by stating that the end of one occurrence must be "before" the beginning of another (this is referred to as the "precedes" relationship). The verification process supported by the SM-API must therefore check process descriptions for consistent before relationships (and their associated sequences of processes).

Figure 6: Structure of the PSL Ontology

Planning strategies typically describe hierarchies of processes (as well as sequences). A machining process may for example involve several sub occurrences of milling and drilling. The work of Bock and Gruninger (2004b) shows that the theories of Sub-Activities, Atomic Activities, Complex Activities, and Activity Occurrences are relevant to this type of hierarchical description. The SM-API must again check for consistencies in the description of sub-occurrences (e.g. if A is a sub-occurrence of B, and B is a sub-occurrence of A, then A must be equal to B).

Further concepts (described in the PSL occurrence tree definitions) can be used to represent such constraints. These include the term "*poss"* which describes an activity that becomes possible following the occurrence of another activity. Again the SM-API must highlight any impossible occurrences by examining the "poss" statements made by a manufacturing strategy, e.g. if milling is only possible after an occurrence of setting, then

any attempt to start a milling process without setting should be identified.

The PSL "*durations"* extension may also be used to represent timing aspects of a planning strategy. The duration of an activity-occurrence is the difference between its end and beginning timepoints. In a computational implementation, "duration" can be represented by a long integer denoting the elapsed time between two timepoints.

Recent work has also gone into modelling process inputs and outputs, using the PSL concept of states (Bock and Gruninger, 2004a). This allows processes to be described using statements such as holds and priors (similar to the example shown above). These describe states that are held following the occurrence of an activity, or set as a prior condition for an occurrence; and these concepts may be particularly useful in describing technology and facility strategies.

The PSL-concept of states is most effective however, when facility and product descriptions are constant (as in many process planning applications). Here, it is often enough to describe an object's state using binary attributes, e.g. a surface has either been milled or not milled. Technology strategies however, tend to describe the inputs and outputs of processes in terms of geometries. For example, the tolerance held following a milling process may be expressed in micrometers. Maximum and minimum constraints on a milling machine may also be expressed in metres, or even Kilograms. The selection of different processes and facilities (i.e. facility strategies) also depends in many cases on numerical comparisons of requirements and outcomes. Here for example it is not enough to simply state that a surface is *milled*, as the strategy must also describe the conditions under which milling is (and is not) required, and exactly how smooth the component's surface is after the milling occurrence.

Facility rules describe how to apply resources and processes. Many of the rules associated with the representation of these types of rules need an understanding of the current state of manufacturing facilities and products, and the required (specified) state of the final product. A grinding process may for example be needed if required tolerances exceed the capability of the milling process. It is therefore useful to describe "geometric-states", that are held following activity occurrences, rather than limiting representations to the existing PSL concept of binary states. A surface tolerance may for example be held to 3.0mm after a casting occurrence, and 0.1mm following a rough milling occurrence. This allows process models to describe the behaviour of processes more precisely by allowing geometric attributes to hold specific values following occurrences and for prior conditions on geometric attributes to be set for occurrences.

Finally, the PSL theory of resources defines a resource which can be used to describe capacity strategies. Any object that is required by some activity – where ''activity'' and ''requires'' are defined elsewhere in PSL (Cutting Deceelle et. al. 2003). The theory of resources also defines the concepts of *available-quantity* and *aggregate-demand*. Broadly speaking activities can demand a quantity of resource, and the aggregate-demand is the sum of all demands running concurrently with an occurrence. The available quantity of resource can also be held to a given value by an activity occurrence, and the aggregate demand for a resource must never exceed the available quantity.

The Implementation of the SM-API

The SM-API has therefore been built in the form of an object-oriented implementation of the PSL ontology, with certain additions and interpretations (see figure 7). This allows manufacturing strategies to be described in terms of processes using resources to meet

manufacturing objectives (see figures 3a and 3b). In summary (and using the lexicon of the PSL ontology), the SM-API must be capable of representing and verifying:

- The occurrence of activities over time, the beginning, ending and duration of occurrences; and the constraining of occurrence over time.
- Process hierarchies and descriptions of how sub-occurrences occur within an overall process plan, and constraints on the possibility of occurrences.
- Product characteristics and facility conditions expressed in terms of geometries (and their associated units), and enumerated properties.
- Assessments of required and manufactured product characteristics, and the technology rules associated with atomic strategies.
- Details of how processes change and are constrained by product characteristics and facility conditions.
- The demand for, and creation and consumption of resources, during the execution of a manufacturing strategy.

Figure 7: Shared Model Class Diagram

The Timepoints class supports the representation and verification of the PSL concept of a timepoint. This allows occurrence activities and the existence of objects to be bound by begin and end timepoints, and for constraints on when occurrences/objects begin and end to be stated. The intervals between the beginning and ending of occurrence/objects can also be expressed (and manipulated) in terms of durations.

The Occurrences class allows process hierarchies to be described using sub-occurrence relationships; and the OccurrenceTrees class supports the constraining of process

sequences in using the PSL "poss" relationships. This allows complex strategy representations to describe how sub-occurrences form overall process plans.

The Structures class allows product characteristics, technology strategies and facility strategies to be expressed in terms of geometries (and their associated units), and enumerated properties. This extends the PSL concepts of states to support geometric states, and geometric "holds" and "prior" relationships. These can be used to describe how processes change product characteristics, place constraint on processes based on product characteristics and facility conditions.

The demand, creation and consumption of resources, during the execution of a manufacturing strategy, are supported by the resources class. This includes methods for describing aggregated the demand and the availability of resources.

5. Experimental Environment

Scope of the Prototype System

This research uses a case study of a jet engine combustion chamber and its manufacturing environment (Cochrane 2007) to provide a proof of concept for the proposed KVM. The study was limited to a representative, but simplified prototype system. The scope of this system was agreed with the designers and experts associated with the study as being appropriate for a proof of concept, and shows how the SM-API (with further work) could be applied in industrial situations.

The role of the different actors involved in building and using the prototype system should also be considered. The designers and experts (mentioned above) help specify the system during the elicitation stage of the development methodology shown in figure 1;

and the designers ultimately use the system to evaluate the manufacturability of new products. A "systems engineer" is however responsible for the structuring and formalisation stages of the methodology. This involves writing Java based descriptions of products, processes and process relationships. This is supported by the SM-API, which provides a set of terms (and verification procedures) for describing these relationships. This separation of roles allows designers and experts to focus on designing and manufacturing products (rather than building information systems).

A knowledge elicitation review with the designers and experts associated with the case study was performed as part of the experiment (see figure 1). During the review, the physical form of the combustion chamber was described, and the processes and process plans associated with its manufacture where informally documented. This review also provided data relating to manufacturing rates, and generated details of suitable test cases that could be used to verify a prototype computational model. These test cases highlighted the limits of certain processes, and established the dimensions and tolerances of chambers that would expose those limits. The product, process and test case descriptions generated by this review are detailed in Cochrane (2007), and are summarised below.

Simplified Description of a Combustion Chamber and its Manufacture

The chamber is manufactured by an initial fabrication process (either casting or forging), which is followed by a series of machining operations. The machining operations create additional features such as holes, and improve the tolerances of the chamber's surfaces.

Figure 8: Simplified Combustion Chamber

Combustion chambers can be regarded as a series of cylindrical rings with varying inner

and outer diameters (figures 8, 9, and 10). Rings form the main chamber, and additional rings of increased thickness may be added to contain fractures. Each ring takes the form of a conical frustum, and holes may be placed on any part of the ring. The flanges at either end of the chamber are described as rings (with a short length, straight edges and increased thickness). The dimensions and surface areas associated with each ring are shown in figure 6. R1 and R2 represent the outer radii of each ring, and r1 and r2 its inner radii. The surface areas calculated for each ring need to be machined by both rough and finish turning processes to achieve the required tolerances (see tables 1 and 2).

The tolerance of each ring surface is initially set to the output tolerance of the casting process that was used to create the initial shape, i.e. 3mm. Each surface requires multiple rough turning processes, and a possible finish turning process, to achieve its required tolerance. Under these conditions, each rough turning operation improves the surface tolerance by either 1.0mm, or achieves the output tolerance of 0.4mm. Finish turning will be required when tolerances of less than 0.4mm are specified. Table 2 uses the surface area calculations, machine settings and required tolerances to calculate the total machining time for the chamber. An initial 300 seconds is included for machine setting (this is based on a simplified model of machine setting durations).

Figure 9: Ring Faces (Cross Section)

Figure 10: Ring Section (Length)

The Combustion Chamber Test Case

Tables 1, 2 and 3 below, show a series of calculations made for machining chambers to specified dimensions and tolerances. The prototype system built as part of this research needs to show (during system verification) that it can generate these results. These test

cases where developed as part of the knowledge elicitation process with designers and experts, and provide a basic set of calculations (with expected results).

Table 3 shows the machining times for two holes in the surface of the cylinders. These calculations use the volumes of each hole to estimate drilling durations, and the surface area of each hole to estimate boring and reaming durations. Both holes have a diameter greater than 15mm, and therefore require pilot drilling processes. This brings the total machining time (including all rough turning, finish turning, drilling, boring and reaming processes) to 1.4 days, or 6 days and 23hrs for five units.

Table 1: Surface Area Calculations

Table 2: Rough Machining and Turning Durations

Table 3: Hole Specification and Machining Durations

The diameter, depth and position tolerances of the required holes are also shown in table 3 (Rtol, Dtol, and Ptol respectively). Depending upon the selected manufacturing facility, positioning errors may be observed for the specified holes if the Ptol tolerance exceeds the machine tool's positioning capability. If this tolerance is set to 80μm for drilling, boring and reaming strategies errors should be expected in the simulation results.

The Manufacturability Analysis Platform

The Manufacturability Analysis Platform (figure 11) was used to evaluate the Knowledge Verification Methodology. Manufacturing strategies (describing casting, and machining facilities) were expressed using the terms described by the SM-API. These were implemented as Java methods, which also perform the verification procedures described above (based on the axioms of the PSL ontology).

The MAP uses a tiered architecture to separate information describing processes and resources (stored in relational databases and/or spreadsheet tables) from rules and constraints (coded in Java and expressed using the shared terms/methods). The platform also implements required and manufactured product models; and separates atomic strategies (e.g. drilling individual holes and machining surfaces) from complex strategies for manufacturing whole components.

Each manufacturing strategy was generated manually (rather than by any form of automated technique), but relied heavily on the reuse of atomic facility descriptions, and the processes used to generated the required knowledge for each model are described in section 3 (starting with the elicitation of user requirements).

Figure 11: The Manufacturability Analysis Platform (MAP)

Simulation Results

Figure 12 shows the results of a manufacturing strategy simulation. Each simulation examines a set of product requirements, and constructs a model of the processes and resources needed to manufacture the product(s). Work pieces are for example instantiated (using the Structure class), and their geometries and properties are manipulated by a series of activity occurrences. The final work piece(s) are then compared against the original requirement. The manipulations performed by processes are derived from the "holds" relationships that have been described by the manufacturing strategies (using the SM-API). Each simulation highlights any aspects of the model that contradict the axioms of the SM-API's underlying ontology (i.e. PSL). This may for example include violation of any prior constraints on occurrences. The model can be updated as and when changes to the product requirement are made.

The results shown in figure 12 contain unexpected errors in the representation of the machined structure, and in the allocation of resources to the stages of manufacture. These unexpected errors are consistent with the role of the SM-API shown in figure 4, and indicate some form of inconsistent statements in the representation of manufacturing strategies. The cause of each error needs to be traced by the systems engineer responsible for formalising the strategy representations, and corrected. Cochrane (2007) provides a detailed examination of several categories of errors, including violations in the description of:

- Process sequences, durations and hierarchies
- Sub-occurrences and possibility trees
- Product characteristics and technology rules (e.g. geometries)
- Process inputs (required prior states) and outputs
- The demand for, and creation and consumption of resources.

Figure 13 shows the corrected machining strategy, and the simulated process durations matching the test case expected results, including the expected simulation errors (predicted by the test case). These provide feedback to the system user (rather than the systems engineer building the system), and indicate that the capability of the selected machine tool needs to be improved, or the required positioning tolerance of the two holes needs to be relaxed. These errors were predicted by the test cases (see previous section).

Figure 12: Unexpected Errors in the Machining Strategy

Figure 13: Verified Machining Calculations

6. Discussion and Conclusions

The main achievement of this research has been the development of an explicit definition of how to represent and verify manufacturing strategies within decision support systems. This has been achieved within a broader framework for the development of decision support systems based on the CAE-RM. The explicit definition has taken the form of the Shared Model – Application Programming Interface (SM-API), and is based broadly on the PSL ontology (i.e. an open standard). The SM-API was also shown to be able to present planning, capacity, technology and facility strategies.

Certain interpretations, simplifications and developments of the PSL standard were however found to be necessary. These included the extrapolation from the PSL ontology of the concepts such as: geometric states, properties and substructures. These provided a way of constraining process inputs and describing process outputs. A number of limitations were also identified by the case study and associated test cases. These included:

- The need for an "inspection process" to drive the comparison between the required and manufactured product models during strategy simulation. This was not part of the PSL ontology or the SM-API specification.
- The SM-API often replicated error messages, as the examination of transitive timing relationships and possibility trees often lead to the multiple identifications of a single error (this was more of an annoyance when interpreting the simulation outputs than a significant issue with the SM-API).

- The verification processes only identified inconsistent strategies. Consistent strategies can still be based on incorrect data (e.g. machining rates), and so the overall accuracy of the decision support system is still highly dependant on the identification of suitable test cases. Inconsistencies in the modelling of capacity strategies were for example, only identified when processes actually ran out of resource. Additional test cases that predict the availability of resource at instances in time are therefore needed.
- Further improvements in the strategies modelled by the test cases were suggested during the first pass validation stage (see section 3). These included the use of volumes rather than surface areas to calculate rough machining times, and a more accurate representation of machine setting times. The approximations used were however sufficient as a proof of concept.
- The inclusion of cost information in the selection of processes (i.e. facility strategies) is especially important, and should be considered as one of the next stages of development for the SM-API. This will require consideration of how the PSL based ontology can handle concepts related to cost.
- The need to integrate the SM-API with improved techniques for providing feedback to designers and experts during system validation was also identified, as was the need to integrate the SM-API with more automated planning process techniques such as simulated annealing algorithms (Bramall et. al. 2003).
- The need to integrate formal frameworks (such as the SM-API) with decision support systems for the conceptual design stage. These could be based on informal ontologies for the improved search and categorisation of tacit design knowledge.

- Finally, the methodology described by this research does not eliminate the need for test cases derived from domain experts, Giaratanno and Rilley (1998). These are central to the verification approach, and are supported by a set of verification procedures derived from the PSL ontology.

Abbreviations

CIMOSA	Computer Integrated Manufacture Open System Architecture
KBS	Knowledge Based Systems
MOSES	Model Oriented Simultaneous Engineering System
NIST	National Institute of Standards and Technology
OWL	Web Ontology Language
PSL	Process Specification Language
RM-ODP	Reference Model of Open Distributed Processing
STEP	**ST**andard for the **E**xchange of **P**roduct information
UML	Unified Modelling Language

Glossary

Data: symbols (e.g. text and numbers) with no specified meaning or context.

Capability: description of what a manufacturing strategy can (and can not) achieve, expressed in terms of manufacturing processes inputs, outputs, and constraints.

Elicitation: the learning, uncovering, extracting, surfacing, and/or discovering the needs of customers, users, and other potential stakeholders.

Exception handling: the detection and resolution of error conditions highlighting inconsistencies in the computational model of a manufacturing strategy.

Explicit knowledge: includes both informal and formal knowledge representations, e.g.

text, diagrams, procedures, rules and constraints.

Formalisation: describes the process of representing informal knowledge in terms of executable rules and constraints.

Formal knowledge: rules and constraints describing how to apply information, expressed in a computer language (e.g. Java), or a formal language (e.g. KIF).

Framework: provides a set of guidelines and procedures for developing knowledge based decision support systems.

Implicit knowledge: can be inferred or implied from observable behaviour, and potentially developed into explicit knowledge.

Informal knowledge: a natural language (e.g. English) or diagrammatic description of how to apply information.

Manufacturing strategy: a description of a manufacturing process in terms of: its objectives and constraints, and the resources that it demands and consumes.

Representation: the process of representing elicited knowledge, initially in terms of a structured informal representation, followed by a formal computational model.

Structuring: describes the process of representing elicited knowledge in terms of a structured but still informal representation.

Tacit knowledge: describes informal representations such as videos and transcripts of conversations that can not always be formalised.

Validation: ensuring that a system meets the requirements of its end users, i.e. making the right system.

Verification: ensuring that a system meets its specified requirements, i.e. making the system right.

Acknowledgements

This work is part of a research project entitled "Knowledge Representation and Reuse for Predictive Design and Manufacturing Evaluation". This has been funded under EPSRC GR/R64483/01, and actively supported by Rolls Royce plc and BOC Edwards Ltd.

References

Bock C, Gruninger M, 2004a. Inputs and Outputs in the Process Specification Language, NISTIR 7152, NIST, Gaithersburg, MD, 2004. Web Accessed 17th February 2005: http://www.nist.gov/msidlibrary/doc/nistir7152.pdf

Bock C, Gruninger M, 2004b. PSL: A Semantic Domain for Flow Models. Software and Systems Modeling Journal, 2004. Web Accessed 17 Feb 05: www.mel.nist.gov

Bullinger H, Lentes H, Scholtz O, 2000. Challenges and chances for innovative companies in a global information society. International Journal of Production Research, 38(7), 1469–1500.

Bramall D, McKay K, Rogers B, Chapman P, Cheung W, and Maropoulos P, 2003. Manufacturability analysis of early product designs. Int. J. Computer Integrated Manufacturing, 2003, Vol. 16, No. 7–8, 501–508.

Chira O, Chira C, Tormey D, Brennan A, and Roche T, 2004. A Multi-agent Architecture for Distributed Design. Lecture Notes in Computer Science. Volume 2744

/ 2004 Title: Holonic and Multi-Agent Systems for Manufacturing. ISBN: 3-540-40751-0. Chapter: pp. 213 – 224. Online Date: January 2004.

Cochrane 2007. Manufacturing Knowledge Verification in Design Support Systems. PhD Thesis, Loughborough University, 2007.

Cutting-Decelle, A, Young R, Anumba C, Baldwin A, Bouchlaghem N, 2003. The Application of PSL in Product Design across Construction and Manufacturing. CERA Journal, Vol 11, No 1, March 2003, pp65-75, ISSN 1063-293X

Fernandez S, Aler R, and Borrajo D, 2005. Machine Learning In Hybrid Hierarchical And Partial-Order Planners For Manufacturing Domains, Applied Artificial Intelligence, 19:783–809, 2005

Giarratano J, Riley G, 1998. Expert Systems: Principles and Programming, (3rd edn). Boston: PWS-Publishing Company.

Gregoire E, Mazure, B, 2002. About the incremental validation of first-order stratified knowledge-based decision-support systems. Information Sciences 142 (2002) 117–129.

Gruninger M, Sriram R, Cheng J, Law K, Process Specification Language for Project Information Exchange. International Journal of IT in Architecture, Engineering & Construction, 2003.

Hickey A, Davis A, 2002. Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes. Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03) 0-7695-1874-5/03 2002 IEEE.

ISO/CD18629. Industrial Automation System and Integration — Process Specification Language: Part 1: Overview and Basic Principles. ISO TC184/SC4/JWG8.

ISO/IEC 10746-1. Reference Model for Open Distributed Processing.

ISO/JTC1/SC32/WG2. Knowledge Interchange Format, Part 1: KIF-Core. WD, International Organization for Standardization (ISO).

Kosanke K, Vernadant F, and Zelm M., 1999, CIMOSA: enterprise engineering and integration. Computers in Industry, 40, 83–97.

Lin H, Harding J, Shahbaz M, 2004. Manufacturing system engineering ontology for semantic interoperability across extended project teams. Int. J. Production. Research, 15 December 2004, vol. 42, no. 24, 5099–5118.

Liu S, Young R, 2004. Utilizing information and knowledge models to support global manufacturing co-ordination decisions. Integrated Journal of Computer Integrated Manufacturing, September 2004, Vol. 17, No. 6, 479–492.

Molina A, Bell R, 1999. A Manufacturing Model representation of a flexible manufacturing facility. Journal of Engineering Manufacture: Proceedings of the Institution of Mechanical Engineers, Vol 213, Part B, pp.225-246.

Molina A, Bell R. 2002. Reference models for the computer aided support of simultaneous engineering. Int. J. Computer Integrated Manufacturing, 2002, Vol 15, No 3, 193-213.

O'Keefe R, O'Leary, D, 1993. Expert system verification and validation: a survey and tutorial. Artificial Intelligence Review, 7, 3–42.

Oldham, K, Kneebone, S, Callot, M, Murton, A and Brimble, R, in N. Mårtensson, R. Mackay and S. Björgvinsson (eds.), 1998. Changing the Ways We Work. Advances in Design and Manufacturing, Vol 8, Proceedings of the Conference on Integration in

Manufacturing, Göteborg, Sweden, IOS Press, Amsterdam, October 1998, 198-207. Accessed 01/Jun/04: http://www.kbe.cov.ac.uk/moka

Pechoucek, M. Vokrinek, J. Becvar, P. , ExPlanTech: multiagent support for manufacturing decision making, Intelligent Systems, IEEE Jan.-Feb. 2005 Volume: 20, Issue: 1, p. 67- 74

Plant R, Gamble R, 2003. Methodologies for the development of knowledge-based systems, 1982–2002* *The Knowledge Engineering Review*, Vol. 18:1, 47–81, 2003.

Preece A. Evaluating Verification and Validation Methods in Knowledge Engineering. In R Roy (ed), *Micro-Level Knowledge Management*, Morgan-Kaufman, pages 123-145, 2001

Preece A, Shinghal R, Batarekh A, 1992. Principles and practice in verifying rule based systems. Knowledge Engineering Review, 7(2):114-141, 1992.

Schreiber G, Akkermans H, Anjewierden A, de Hoog R, Shadbolt N, Van de Velde W, Wielinga B, December 1999. The CommonKADS Methodology. ISBN 0-262-19300-0. http://www.commonkads.uva.nl

Smith M, Welty C, McGuinness D, 2003. Web Ontology Language (OWL) Guide Version 1.0. W3C Working Draft 10 February 2003. Accessed 14Jul/03: http://www.w3.org/TR/2003/WD-owl-guide-20030210

Sormaz D, Arumugam J, Rajaraman S, 2004. Integrative Process Plan Model and Representation for Intelligent Distributed Manufacturing Planning, *International Journal of Production Research*, Vol. 42, No. 17, p. 3397 - 3417, 2004

Sormaz D, B. Khoshnevis B, 1997. "Process Planning Knowledge Representation using an Object-oriented Data Model", International Journal of Computer Integrated *Manufacturing,* Vol. 10, No. 1-4, p. 92-104, 1997.

Uschold M, Gruninger M, 1996. Ontologies: Principles, Methods, and Applications. Knowledge Engineering Review, 1996, Vol. 11, pp. 96-137.

Waterson A & Preece A, 1999. Verifying Ontological Commitment in Knowledge-Based Systems. *Knowledge-Based Systems*, 12, 45-54, 1999.

Wu C, Lee S, 2002. KJ3: a tool assisting formal validation of knowledge based systems. Int. J. Human-Computer Studies (2002) 56, 495–524

Young R, 2003. Informing decision makers in product design and manufacture. International Journal of Computer Integrated Manufacturing, Vol. 16, No, 6, pp. 428-438.

Zhao J, Cheung W, Young R, 2000. The influence of manufacturing information models on product development systems, EDC Conference, 2000.

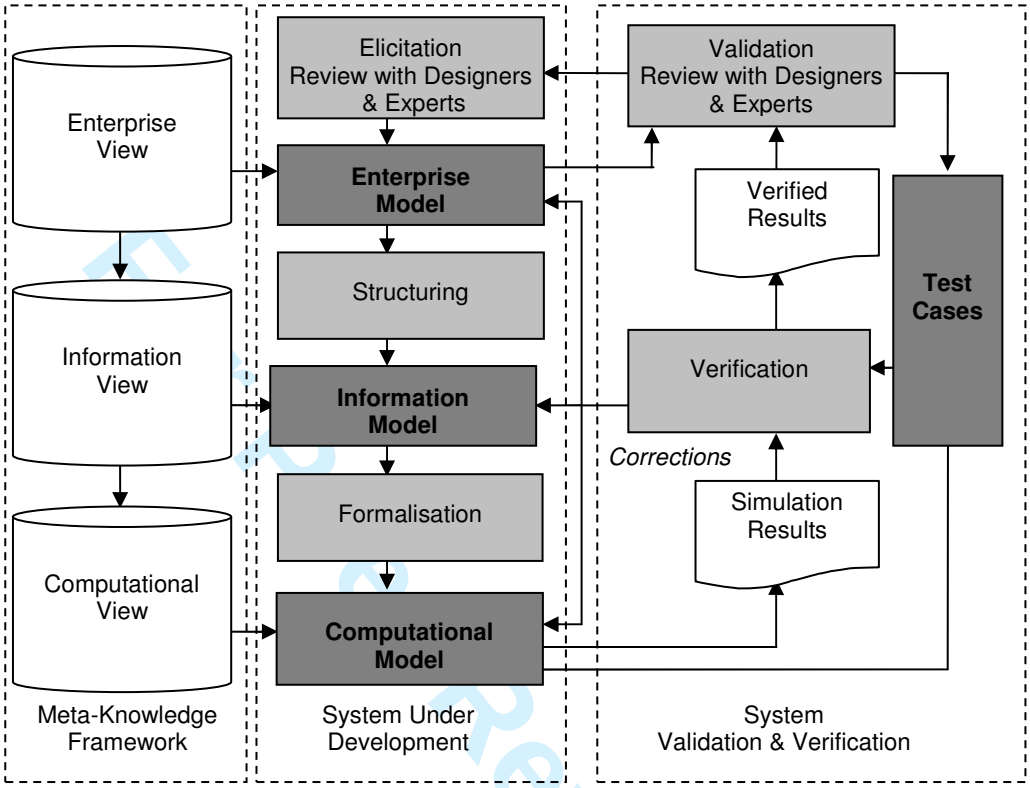

Figure 1: System Development Methodology

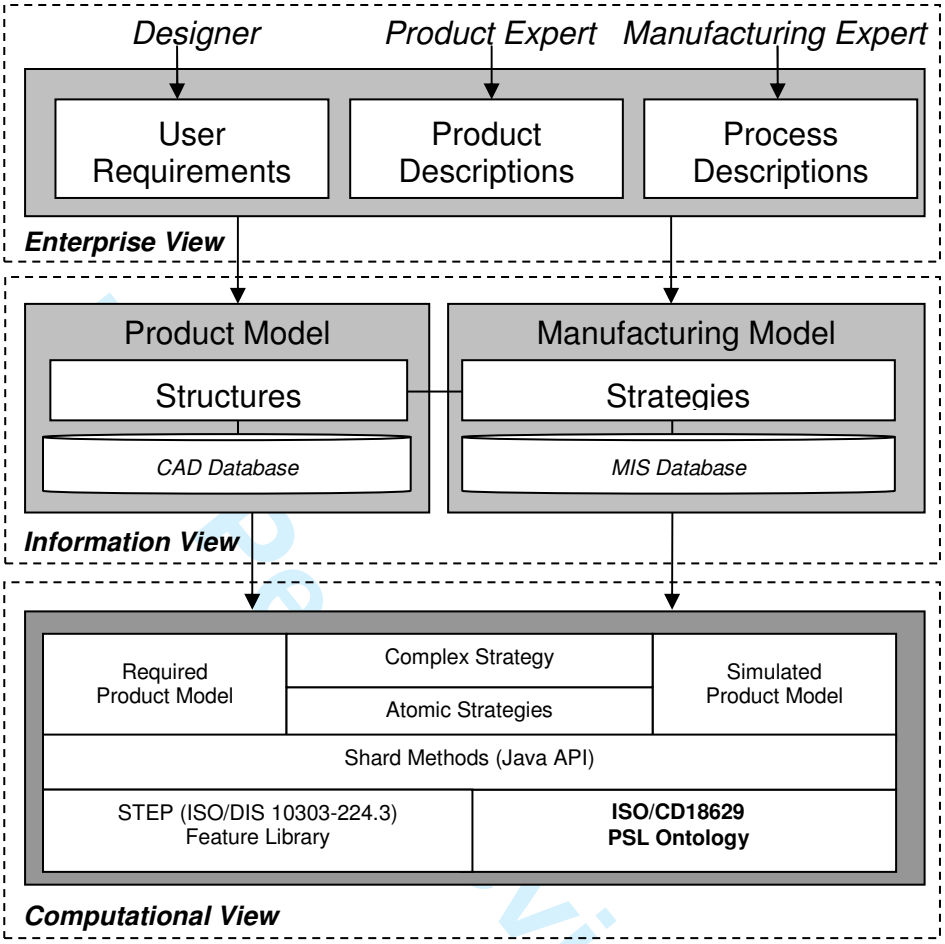

Figure 2: The Meta Knowledge Framework

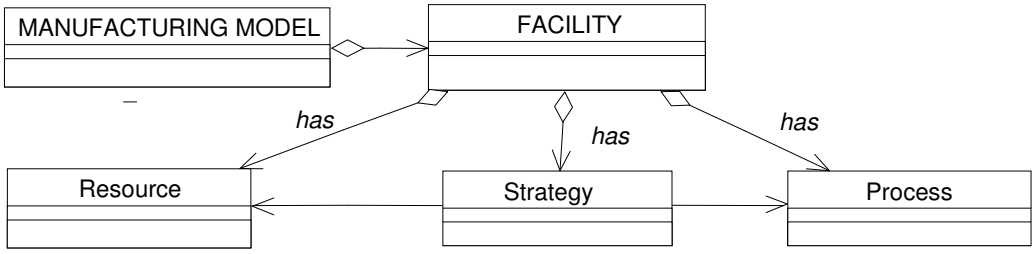

Figure 3a: Adapted Product/Manufacturing Model

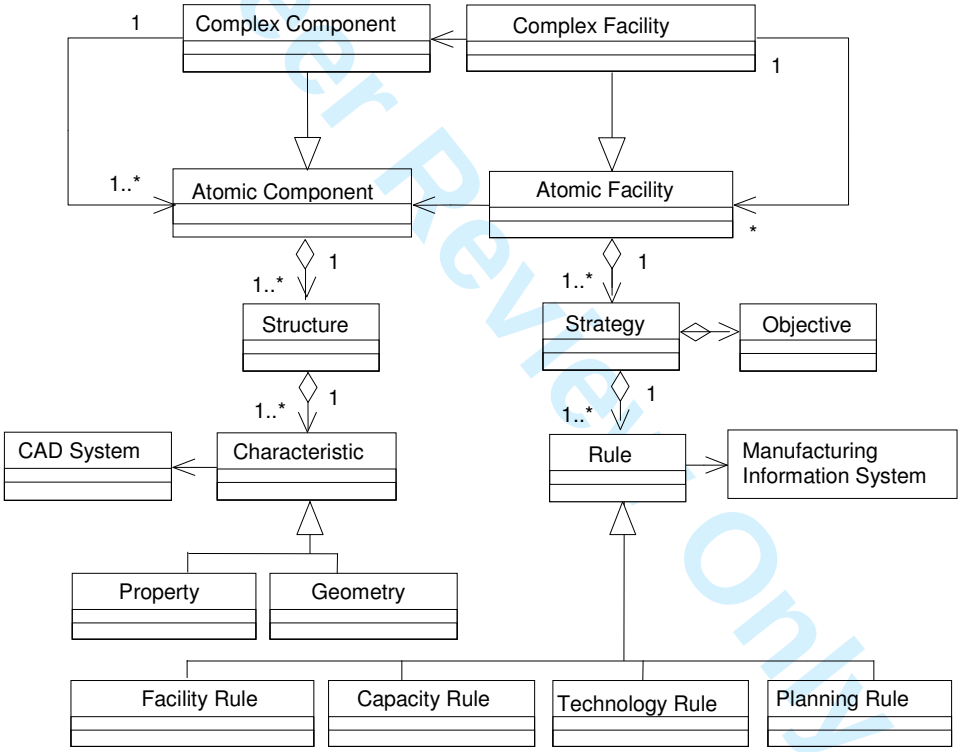

Figure 3b: Adapted Product/Manufacturing Model

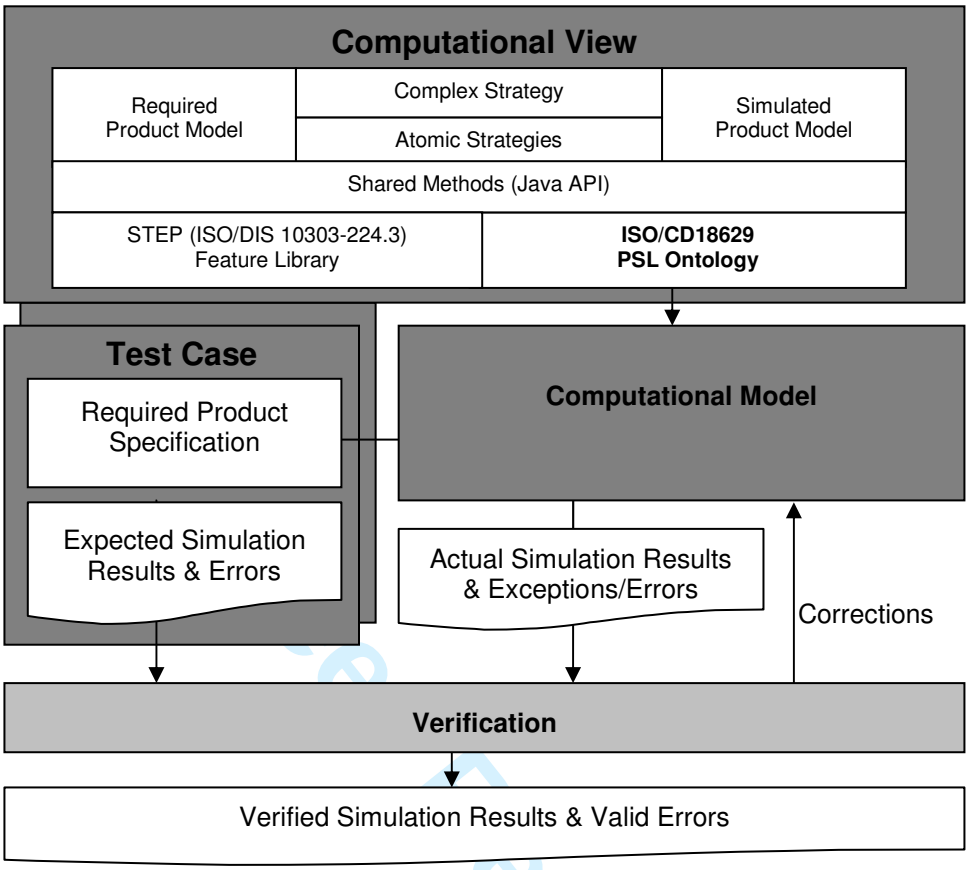

Figure 4: The Use of Test Cases to Support System Verification

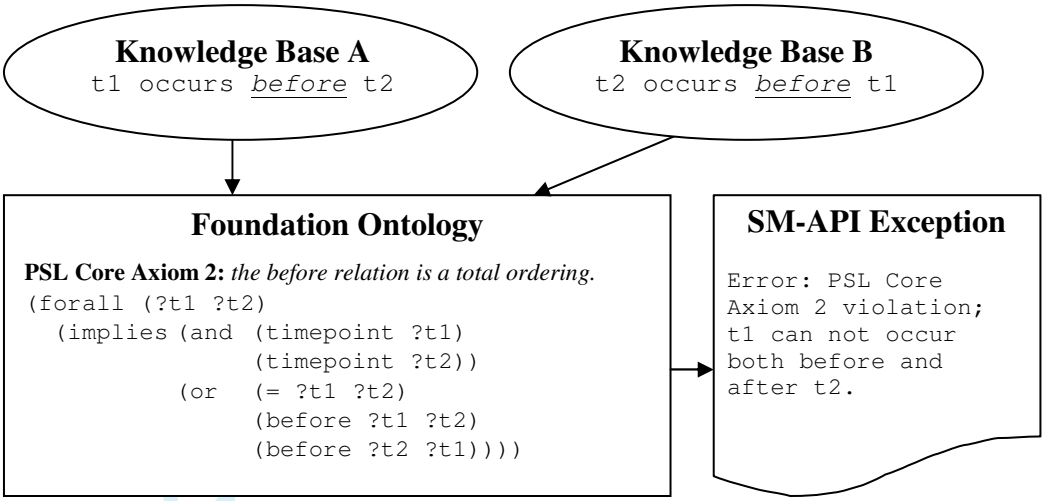

Figure 5: Verification Using the PSL Ontology and the SM-API

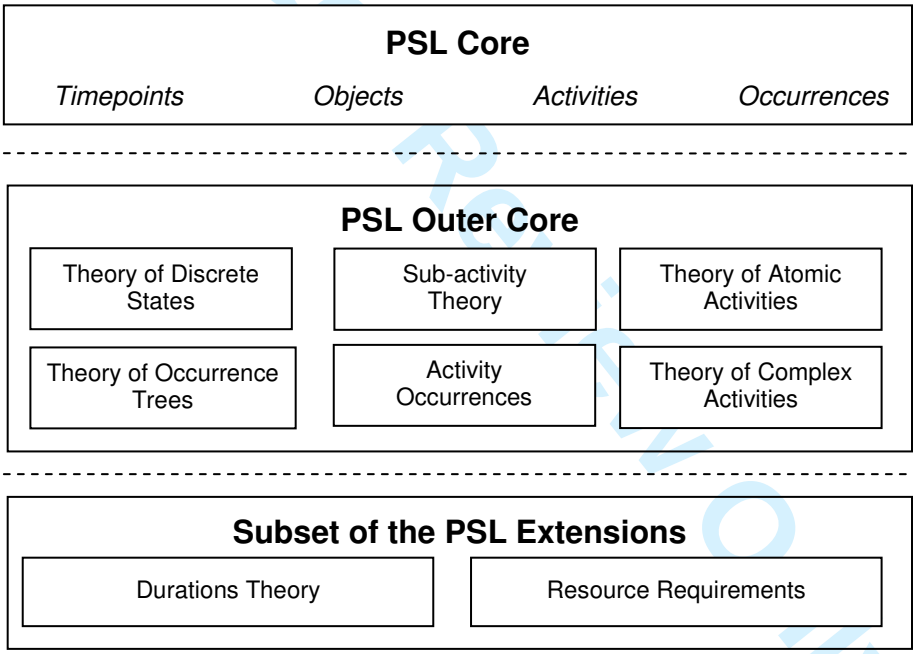

Figure 6: Structure of the PSL Ontology

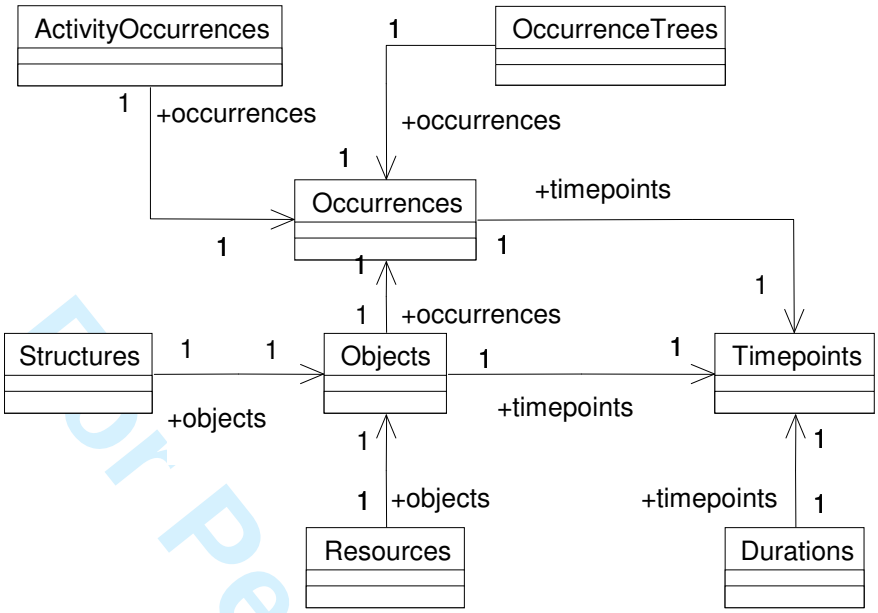

Figure 7: Shared Model Class Diagram

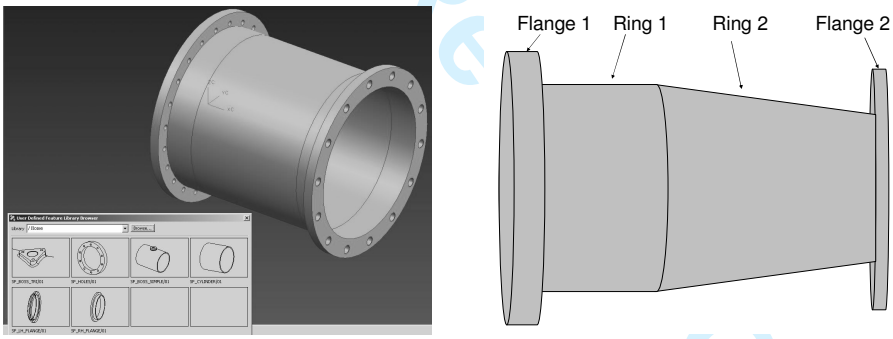

Figure 8: Simplified Combustion Chamber

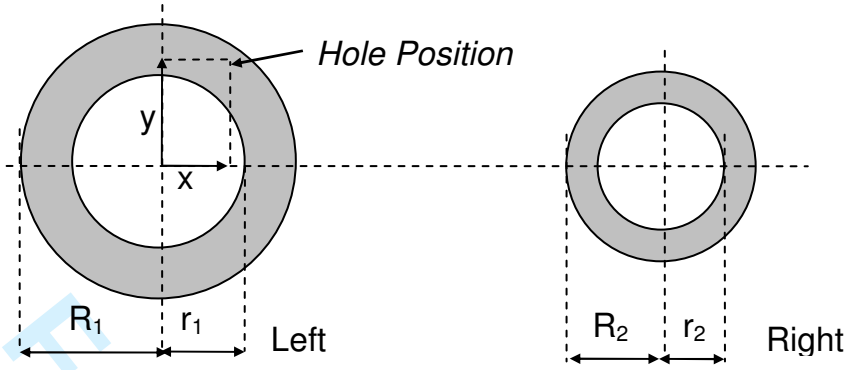

Figure 9: Ring Faces (Cross Section)

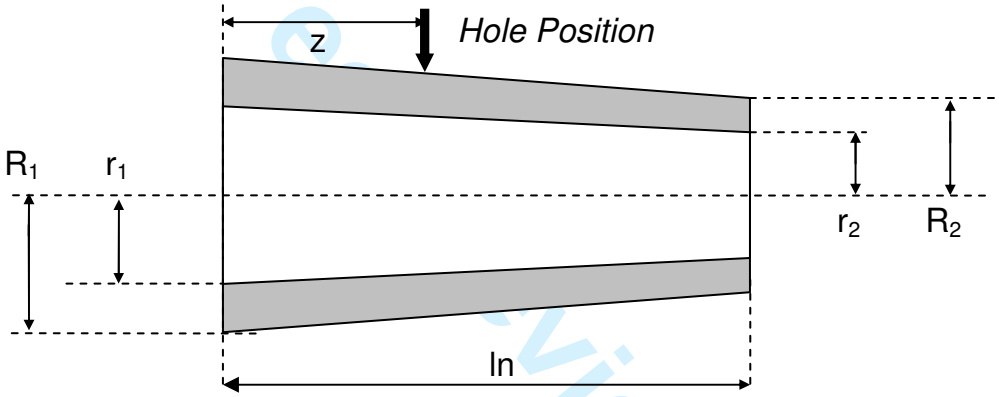

Figure 10: Ring Section (Length)

Surface Area = $\pi * (r_1+r_2) * ((r_1-r_2)^2 + \ln^2)^{½}$

	R1 m	R2 m	ir1 m	ir2 m	ln m	Face 1 Area	Face 2 Area	Outer Area	Total Outer	Total Inner
Flange 1	0.3	0.3	0.22	0.22	0.04	0.131	0.086	0.075	0.292	0.055
Ring 1	0.25	0.35	0.22	0.32	0.1	0.000	0.000	0.267	0.267	0.240
Ring 2	0.35	0.35	0.32	0.32	0.2	0.000	0.000	0.440	0.440	0.402
Flange2	0.4	0.4	0.32	0.32	0.04	0.118	0.181	0.101	0.399	0.080
								Total Surface Areas m2	**1.398**	**0.778**

Table 1: Surface Area Calculations

Process		Setting	Rough 1	Rough 2	Rough 3	Turning		Requ.
Tolerance mm		3.00	2.00	1.00	0.40	0.25		Tol
Minutes per m2			167	167	167	667		mm
Feature	Area	Time in Minutes						
Inner	m2							
Flange 1	0.292	5	49	49	0	0		1.00
Ring 1	0.267		44	44	44	178		0.30
Ring 2	0.440		73	73	73	293		0.30
Flange2	0.399		67	67	0	0		1.00
Outer								
Flange 1	0.055		9	9	9	37		0.30
Ring 1	0.240		40	40	40	160		0.30
Ring 2	0.402		67	67	67	268		0.30
Flange2	0.080		13	13	13	54		0.30
Roughing and Turning time per unit						**1967**	**minutes**	
						1.37	**days**	

Table 2: Rough Machining and Turning Durations

Parent	Part	Att	Diam mm	Depth mm	Rtol um	Dtol um	Ptol um
Chamber	Hole01	Flange1	25.00	30.00	100.00	90.00	51.00
Chamber	Hole02	Flange1	20.00	15.00	80.00	60.00	30.00

Process			Pilot	Drill	Bore	Ream
Minutes per mm2 or mm3			0.6	0.6	9.0	36.0
Hole 01	Surace area mm2	2356			21206	84823
	Volume mm3	14726	8836	8836		
Hole 02	Surface area mm2	942				33929
	Volume mm3	4712	2827	2827		

Drilling, boring and reaming time per unit		**45 minutes**
		0.03 days
Total machining time	**1 unit**	**1.40 days**
	5 units	**6.99 days**
Verification of simulated machining time		**6.99 days**

Table 3: Hole Specification and Machining Durations

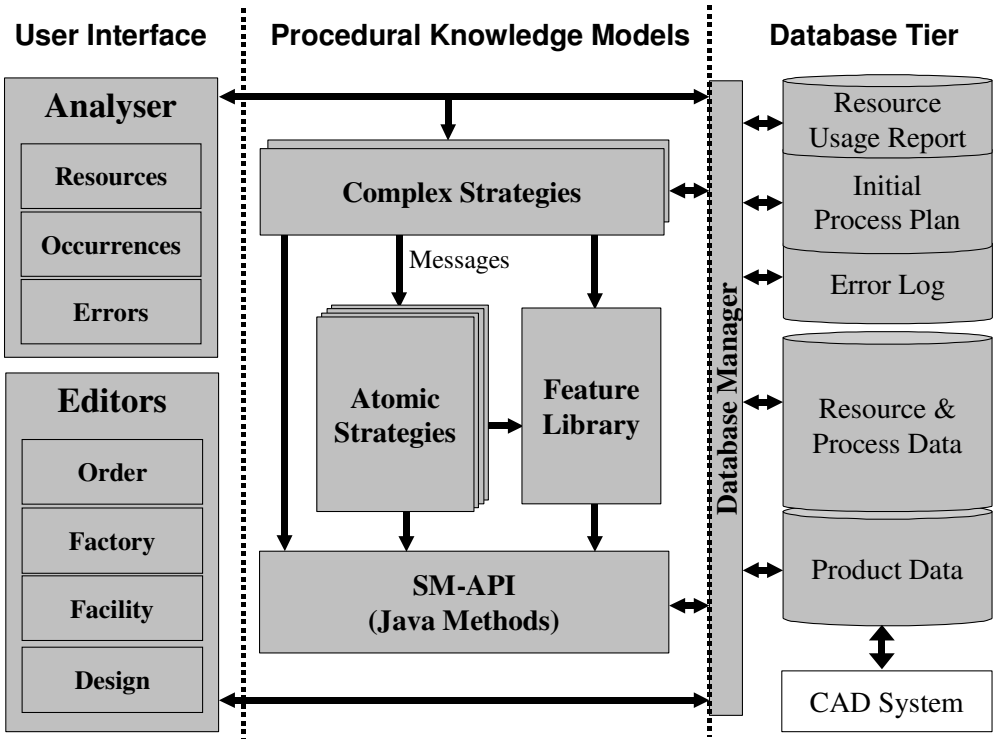

Figure 11: The Manufacturability Analysis Platform (MAP)

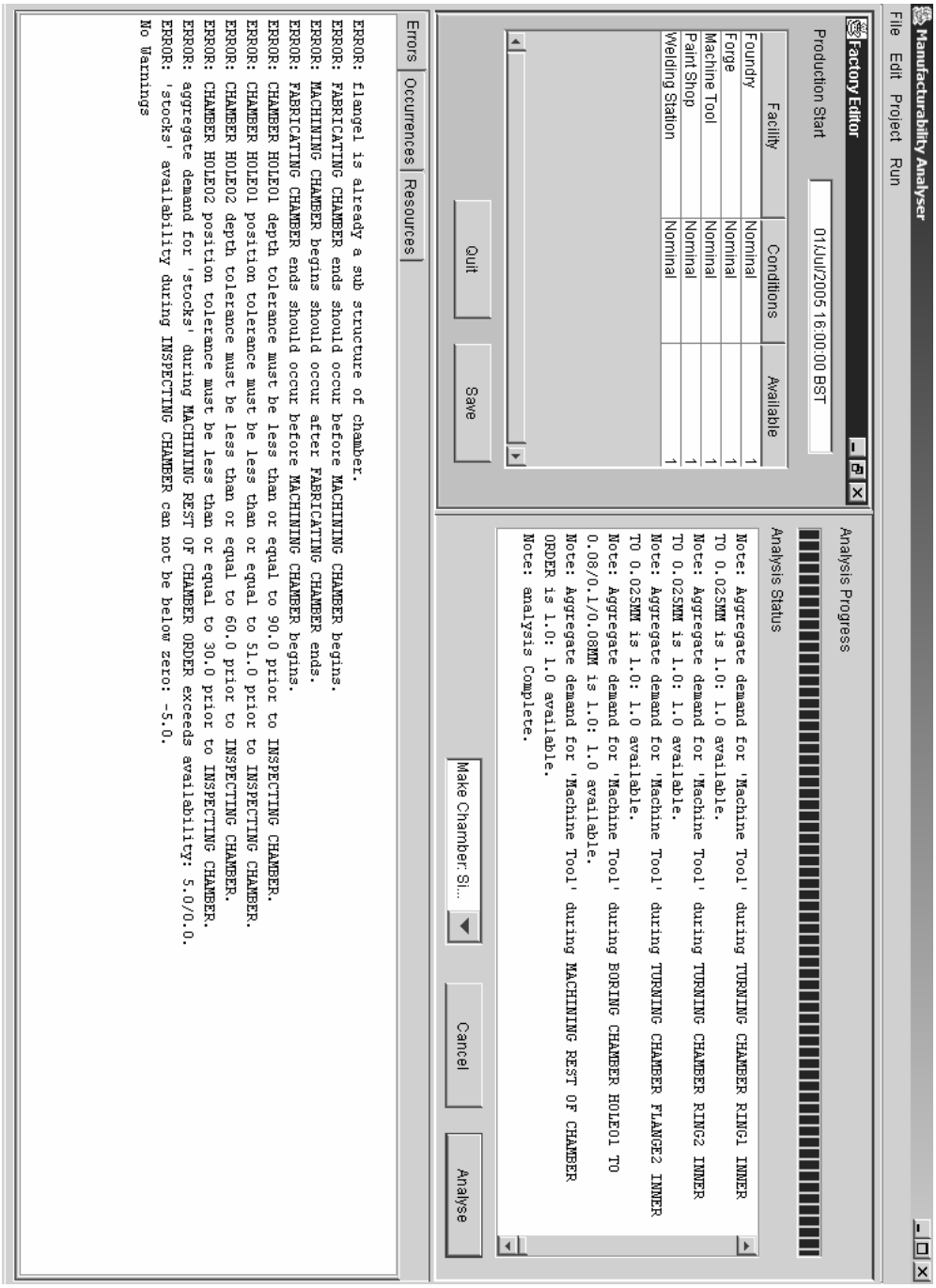

Figure 12: Unexpected Errors in the Machining Strategy

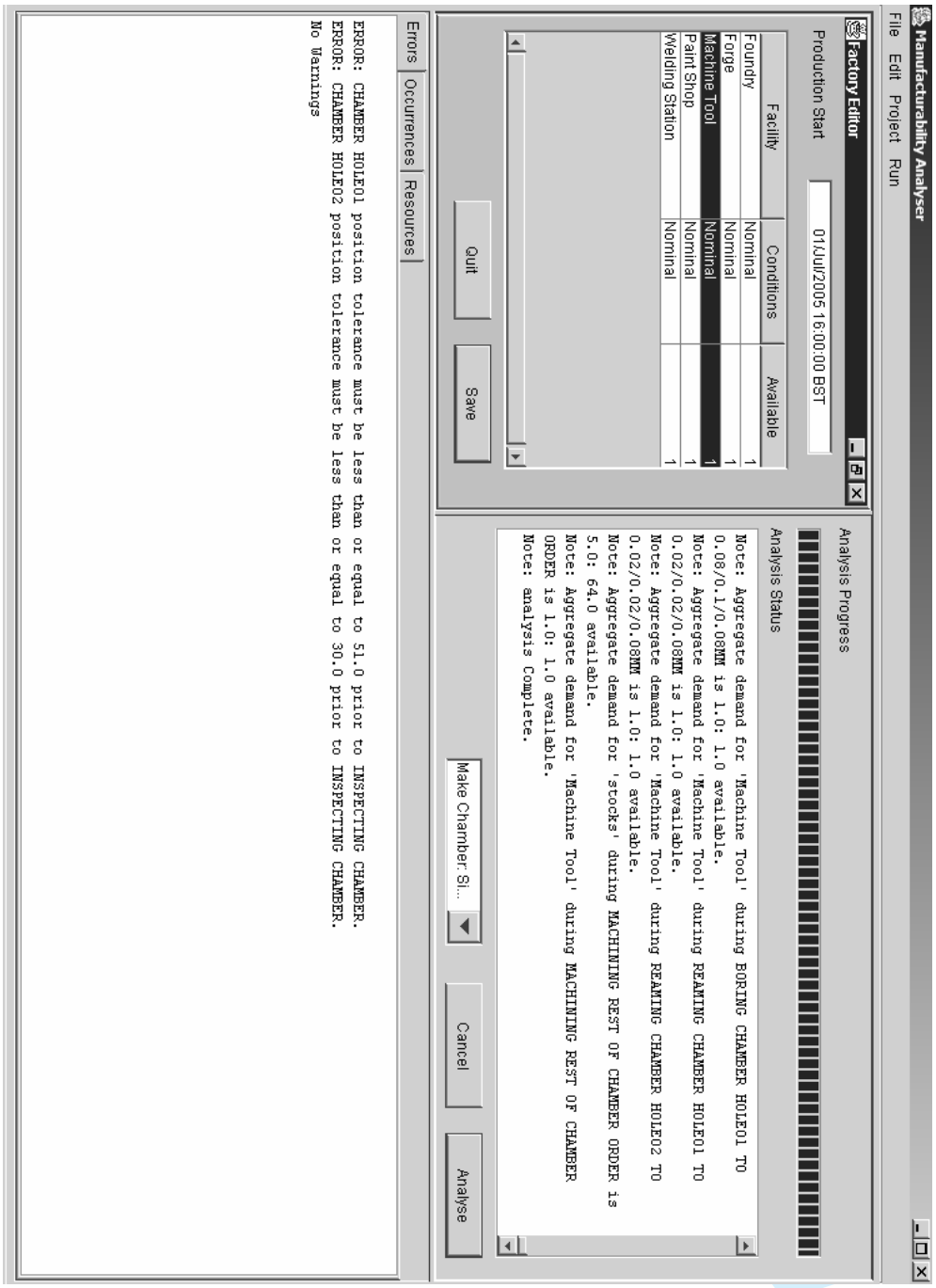

Figure 13: Verified Machining Calculations