Conceptual Modeling of Manufacturing Automation

by
Sushil Birla
Email: birlas@eecs.umich.edu
a study directed by
Prof. K. G. Shin
The University of Michigan
Electrical Engineering and Computer Science Dept.
Ann Arbor, MI 48105

September 22, 1994

Abstract

We report results of a process for conceptual modeling of programmable servo-controlled manufacturing-equipment, where the models would be useful in developing software for real-time monitoring and control. The objective is software reusability and extensibility, i.e. reducing the net effort and time to develop related future applications, through the approach of defining initial requirements in a way that facilitates evolution. Although researchers agree, in general, that such a focus offers very high payoff, they also agree that it is a very complex and difficult subject that will require long-range research. One part of this study was to devise a suitable conceptual process, gleaning and adapting from a number of different approaches taken by researchers and developers in the fields of software engineering, database design, artificial intelligence, and manufacturing engineering. The resulting process is a combination of object-oriented analysis, domain analysis, and knowledge engineeering. Beginning from a definition of the purpose of the conceptual model, the process steps are to bound the domain, select a reference model architecture for intelligent machine control, and outline a model of the key concepts in the form of candidate object-classes that characterize programmable servo-controlled manufacturing equipment. The study demonstrates how the model is reusable across applications that have been historically treated by industry as *different*. Furthermore, the study shows how the model can be extended to applications beyond the initial boundary. The study also reports insights gained through a rapid pre-prototyping approach to explore certain ideas. Not surprisingly, one conclusion from the study is that the conceptual modeling will be an iterative and incremental process—the next iteration of rapid prototyping is planned on an object-oriented CASE Tool.

Contents

List of Figures

List of Tables

List of class structures

1 Purpose

This report addresses issues in conceptual modeling of programmable manufacturing equipment for the purpose of computer-automated control, monitoring, diagnostics and maintenance. The motivation is to make it easier to reuse, extend and integrate software for these tasks.
The purpose of the equipment-models is to maximize the life-cycle economics of computer automation. This is to be accomplished by allowing the integration of knowledge available to perform these functions (the cellular manufacturing concept [52]), using available computer technology. The initial body of knowledge should be reusable and extensible, as economics permit.

1.1 Definition of reusability

According to Prieto-Diaz [41],

> Reusability is a collection of principles and heuristics for the creation and evolution of software systems ... they provide guidance on how other technologies can best be used. In particular, they should provide guidance on how to capture and organize reusable information ...

According to Biggerstaff [7],

> Software reuse is the reapplication of a variety of kinds of knowledge about one system to another similar system in order to reduce the effort of development and maintenance of that system. This reused knowledge includes artifacts such as domain knowledge, development experience, design decisions, architectural structures,, requirements, designs, code, documentation ...

Reusability ... depends on a particular problem and problem-solving context, i.e., ... method [41, pp.10].
This study focuses on the reuse of domain knowledge, including constraints on its reusability. The results are at the level of *r*equirements analysis in the traditional software engineering life-cycle. The knowledge is organized around architectural structures, which, in turn are also a reusable resource.

1.2 Ultimate economic objective

The economic payoff of research and development in conceptual modeling of manufacturing equipment will come through improvement in the productivity (including quality and speed) of engineering innovations in systems that control and monitor manufacturing automation. Many studies and projects sponsored by the U.S. Department of Defense [1, 36] document information from many sources leading to the following conclusions that motivate this study:

- Opportunities for further creation of wealth abound in improving manufacturing productivity through intelligent automation.
- The time-window of such opportunities has typically been short. It is getting shorter.
- Human skill is a constraining resource in the timely exploitation of such opportunities.
- Flexibility or versatility or reusability of manufacturing resources is a key enabler.
- Control software modification and integration [52] have become a major bottleneck in the creation of innovative computer-automated manufacturing systems.

An appropriately-organized, reusable base of knowledge, available on-line, in an open architecture, is a key enabler in reducing the time and cost to engineer and integrate intelligent manufacturing automation [52]. This resource is identified as an "Information Base (IB)" in [48]. The acquisition and organization of the required knowledge is a well-recognized problem in the fields of Artificial Intelligence [35] and Software Engineering [41]. This study addresses the issue of organizing an initial amount of knowledge in a way that it can be extended economically.
Todorov and Levi [52] observe that the current state in practice (and, we add, in manufacturing research communities) is to interface different programs and packages, with great difficulty. They recommend a single integrated database. We focus on the element of conceptual integrity [11] implied in their recommendation.
Next, we decompose this purpose into a number of techno-economic objectives to be accomplished in stages. In the process, we bring out certain issues in accomplishing these objectives. These issues, in turn, provide the rationale for the scope and the direction of investigation chosen in this study. In this process, as recommended by Prieto-Diaz [41, p.10], we will define scope by the extent over which reuse is valued and the extent over which the problem domain is cohesive and relatively stable.

1.3 Productivity objectives

The initial conceptual model should support the basic objectives given below. We chose the conceptual model as the level of reusable resource, since the consensus of experts [6, 27, 45] is to focus on reuse at a high level of abstraction, rather than at the code-level.

1.3.1 Basic objectives

1. The conceptual model should support the creation, modification and evolution of software for the functions to be

performed. To quote Krueger [27], "...it must be easier to reuse [1] the artifacts than it is to develop the software from scratch..." or from code fragments, as used in current practice.

2. Users should be able to derive conceptual models for related manufacturing equipment more easily, in less time, at less cost and with fewer errors than is possible in current practice.

1.3.2 Post-extension objectives

1. Users should be able to integrate different manufacturing devices more easily, in less time, at less cost and with fewer errors, when applying this conceptual model in a standard open systems architecture. In current practice, these devices are often procured from different vendors. Since these devices are typically engineered at different sources with inadequate coordination in the software, their integration is a very costly, time-consuming process today. Cost and schedule overruns are common. The resulting software is not very maintainable.

2. The paradigm should be applicable to a computer-automated machining system (workstation) composed of one or more such manufacturing devices, tooling, fixturing and the workpiece.

3. The paradigm should be applicable to larger manufacturing systems that consist of several workstations or machines, several workpieces in a family and their corresponding sets of workholding fixtures and tools.

4. It should be applicable to related manufacturing automation of types other than the initially modeled class of devices.

5. It should be possible to derive capability/constraint models of manufacturing devices, for the purpose of planning the execution and monitoring of tasks to be performed by these devices. Such information in today's computer-integrated manufacturing systems is available only at a very primitive level and not in a very organized form.

6. Monitoring software within and above the cell should get meaningful, compact feedback through the abstraction and reduction possible by using the conceptual model. The feedback from today's manufacturing device control systems is very primitive and does not facilitate timely reaction, troubleshooting and diagnosis. When monitoring sensors are added, due to their poor integration with the control system, higher-level monitoring systems receive sensory data—often in bulk—without the proper context to support meaningful interpretation, leading to unnecessary overhead and incorrect interpretation.

[1] The cost of reuse includes the cost of finding, modifying and indexing for future reference.

1.4 Some Issues

There are several economic and technical issues in the creation and maintenance of such a conceptual model. No successful commercial examples exist.

1.4.1 Amortizing over expected applications

The initial cost [45] to create such a reusable resource is much higher than the cost of a single-use solution. There are many uncertainties in amortizing this cost over (future) applications that are not fully defined and specified at the beginning.
In cases where the commonality across expected applications is high (narrow domain [7]) and very visible and there is a large volume and frequency of such cases, it is relatively easy to identify the abstractions and establish their economic worthiness. For example, the benefit of parametric feature-definition in a family of parts is well-established. However, with more abstraction many issues and difficulties arise:

1. The volume and frequency of their future uses is not as visible.

2. Until these abstractions are applied in sufficient number of sufficiently different conditions, it is difficult to establish their technical and economic soundness [45].

3. As the application of this approach changes design-practice and as a result of other changes in practice over the development-period, the usage pattern may change and invalidate the original basis of the economic value of certain abstractions.

Thus, the process of defining the *right* abstractions for the *right domains* is going to be iterative by its very nature [41], adding to the complexity of the economics. This study explores two approaches to deal with these difficulties. The first approach is to search for ways to bound the domain to obtain stability. This is an open research question [41]. The second approach is to find a modeling technique that makes evolution easy. This is also an open research question [41]. However, a combination of the two approaches may be a fruitful direction.

1.4.2 Gaining user acceptance

There are two known obstacles to reuse of software:

1. Users perceive reuse to be a constraint on their creativity. One approach to avoid this problem is by concentrating

on applications of well-established engineering practice. Thus the conceptual model should reduce the labor of applying *routine engineering* and allow the engineer more time for the creative part of the task [41]. Ideally, the conceptual model should be perceived as a support for creative work.

2. The practice of function-level abstractions is not common in mechanical engineering, especially in manufacturing applications. Traditional MCAD tools do not support practitioners at this level of abstraction. Therefore, we should concentrate on problem domains where function-level abstractions are more easily accepted and pay off more quickly.

The domain of programmable servo-driven electro-mechanical machinery lends itself to function-level abstractions well. We can also show that the use of these abstractions will reduce "lower-level" programming involved in customizing control systems to different configurations of manufacturing equipment. However, there are many possible abstractions, some of which may appear better than others depending on the specific needs of a particular application. Any chosen set of abstractions would have to be evaluated over a number of application-development cycles, to gather convincing evidence of their economic value.

1.4.3 Understanding what to represent

It has been recognized, in the fields of Artificial Intelligence, Software Engineering and Object-Oriented Programming, that one of the most difficult problems is the representation of the real world in ways that are economically reusable in a variety of applications, situations and contexts. To quote Newell [35]:

> Much of the difficulty turns out to be that we don't understand what we want to represent, not that we can't find first-order ways of doing it. Another part of the difficulty is that the representation is often very awkward and indirect. Thus much research goes into finding alternative forms ... that are much easier to use or more perspicuous.

This study focuses on this issue of knowledge acquisition and representation, following the process described in Section 2.

1.4.4 Capturing undocumented information

It is not sufficient to base the conceptual model on information formally-documented in current practice, because it may not be sufficient for reuse. Davis [14] observes:

> Typically, much of the design and organizational information is not well managed. It is widely scattered, e.g., throughout comments in system code, in documents and manuals separately, and in the mind of the system architect.

1.4.5 Data-structure management

With respect to the implementation of these representations, Davis [14] explains how some of the engineering cost and time is wasted in data-structure management:

> Even adding a new instance of an existing data type is a major task (difficult to find all of the necessary information)...Various approaches to knowledge representation (predicate calculus, semantic nets, production rules, frames, etc.) have all presented a non-trivial problem in data structure management.

Prieto [41] describes this problem as the lack of a *r*euse infrastructure. As one conceptual modeling approach may be better connected to some elements of a reuse infrastructure than other approaches, this factor is also considered briefly in this study.

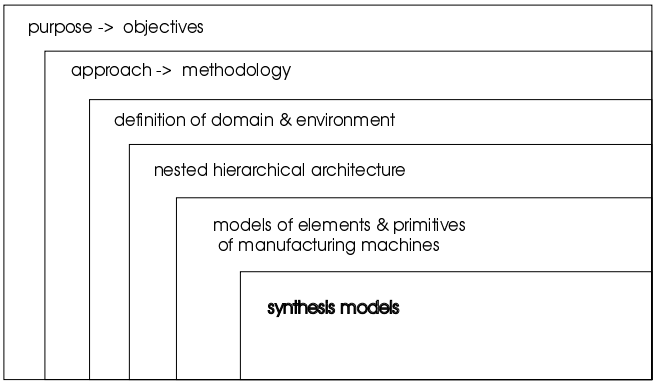


Figure 1: Overview of framework for modeling manufacturing automation.

1.4.6 Multiple disciplines

Another view of the scope of the project is in terms of identifying the established disciplines and subdisciplines of computer science and application domains that are involved in the problem, as depicted in Figure 2.

Disciplines involved in the problem:

- Software engineering
  - Requirements/specifications.
  - Miscellaneous reusable software.
- Information Storage and Retrieval
  - Content Analysis and Indexing
  - Abstracting methods
- Artificial Intelligence
  - Knowledge engineering
  - Machine learning
- Mechanical Engineering
  - Machining processes
  - Machine dynamics and kinematics
  - Machine design
- Systems Science
  - Control systems
  - Signal processing and Statistics
- Business
  - Economics
  - Market analysis and strategic planning

1.5 Organization of this report

Next, we examine the role of technology with respect to the economic objectives and the obstacles discussed above. We develop the method followed in this study as an assimilation and adaptation of a number of published methodologies. Following the chosen methodology, we further prepare the domain information. The result is a collection of natural-language-requirements for domain-analysis. As forecast by Arango and Prieto [41, p.14], this turns out to be a network of domains. As we proceed with the investigation of the domain of manufacturing equipment, along lines recommended by McCain [31], we identify subdomains of equipment. At this point, it becomes necessary to define the purpose of each subdomain. In turn, it becomes appropriate to define a reference architecture. We have done this by gleaning, adapting and synthesizing from published architectural models. Subsequently, we define subdomains for each class of components in manufacturing equipment. The various (sub-)domains are defined in terms of a taxonomy of entities and skeletal frames for each entity. Next, we address models to synthesize or integrate components into subsystems and the equipment-system.

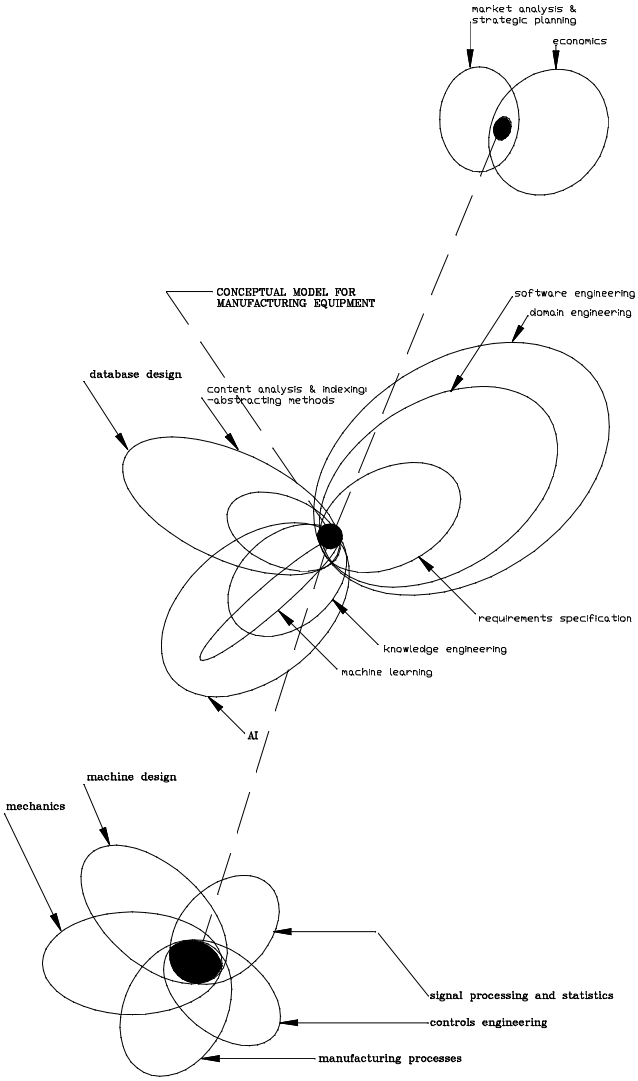


Figure 2: Disciplines involved in this study.

Figure 1 outlines our overall conceptual framework for modeling manufacturing automation.
In conclusion, we summarize the contributions made in this study and the numerous issues yet to be resolved. As this study crosses many research areas, the review of related research is presented as each topic comes up. Since the subject matter includes terminology from many different fields, the report includes appendices for acronyms, abbreviations, and a glossary.

2 Technical Approach

From a review of pertinent literature, research projects, consortia projects and industrial practice, we find many approaches to the issue of making it easier to reuse, modify, extend and integrate software for manufacturing equipment:

- building libraries of reusable code
- defining standard interfaces between application-code fragments[48]
- defining interoperability standards for implementation platforms[48]
- defining architectures in various views and at various levels[2, 48]
- applying CASE tools
- building tools and aids such as expert systems
- building special-purpose application-generators
- using the "right" programming language, e.g., ADA[12]
- defining schema for an Information Base
- creating models[12]

Many of these approaches overlap and many are complementary. Chaar [12] is the closest and most recent example of a methodology developed and applied to real-time control software for a manufacturing cell. It encapsulates complete devices into formal models and transforms these models into implementation-language-level components. The models are used for scheduling jobs to the cell, sequencing operations on the jobs and monitoring a device as a whole. In contrast, the models in our study are more concerned with the primitives from which devices are composed, with the purpose of monitoring and control of motion and the process at a finer granularity in time.

In most literature on the general problem of software reuse [6, 18, 27, 41, 45], there is general agreement among the various sectors of the research community that the critical need is to organize knowledge of the application domain. Prieto-Diaz [41] identifies this task as *d*omain analysis, as a part of the overall field of domain engineering, analogous to software engineering. We next devise a methodology for domain analysis by assimilating ideas from different sources and present the chosen method in terms of modifications to the published methodologies. This is consistent with the recommendation given by Freeman [18] where he also states that no such methodologies exist and establishing such methodologies is a long-term research task.

2.1 Methodologies surveyed

We start with an overview of tasks in domain engineering, as outlined in [41, pp.20–21].

Tasks requiring domain expertise

1. Identify relevant areas of knowledge.
2. Identify reasonable boundaries for these areas.
3. Organize and disambiguate the vocabulary in each problem domain so the semantics of application-specific concepts are refined.
4. Select and sketch representative applications to be used as case studies whose conceptual and design structure will be starting points for detailed analysis.
5. Provide rationales or justifications for specification and implementation concepts.
6. Review and critique resulting domain models with domain experts.

Other tasks of domain analysis

1. Acquire knowledge.
2. Organize and encode the acquired information into a domain model.[2]
3. Verify the correctness of the domain model with respect to representation standards.
4. Validate the information in the domain model vis-a-vis existing systems.
5. Analyze the effects of changes on the model and evolve the model.

Tasks of infrastructure analysis

1. Define the number, distribution, and structure of repositories of reusable information, access procedures, and tools.
2. Define the components to be included in each repository and how they should be represented.
3. Conduct market analysis: [3]
   - Forecast the software production needs of the organization for the given domain.
   - Assess the coverage of the infrastructure,
   - Estimate throughput of the reuse system.
   - Estimate average cost of implementations.
   - Monitor the routine operation of the reuse system.
   - Identify the need to evolve its infrastructure.
   - Assess the impact of changes in the infrastructure (coverage, costs, etc.)

[2] We cut off this study at a partial model that includes a taxonomy of entities and skeletal frames. The subsequent tasks are not addressed in this study.

[3] We modify the method by performing a brief qualitative version for defining the boundary of the domain.

In comparison, we also considered the methodology for developing reusable components, given by McCain [31]. Following is an outline of the part that corresponds to the domain engineering process given above.

1. Perform "market analysis".[4]
2. Perform domain analysis
   - for the application, and
   - for the components.
3. Perform usability analysis.
4. Implement/use prototype (includes full user interface). [5].

   ⋮

Within the revised scope of domain analysis noted earlier, we next review the steps in this task. Prieto-Diaz [40] proposes a methodology in terms of data flow diagrams, from which we extracted the following outline:

1. **Prepare domain information**
   (a) Define approach
   (b) Bound and define domain
   (c) Select knowledge sources
   (d) Define requirements for domain analysis
2. **Analyze domain**
   (a) Identify common features
   (b) Select specific functions/objects
   (c) Abstract functions/objects
   (d) Classify
   (e) Define domain language
      - Taxonomy[6]
      - Frames
      - Language
      - Model
3. **Produce reusable workproducts** [7]

McCain [31] recommends that abstract interface specifications should be defined for each reusable component. However, before we can define component interface specifications or even component functions, an architecture relating the components to the system is needed. This is a major task that we treat in a separate section, before developing the component models.

[4] As mentioned earlier, we chose this recommedation, making these cost/benefit considerations at the domain-scoping stage.

[5] We modified the methodology by creating a rapid pre-prototype, including the user interface, with only a small part of the conceptual model

[6] The output of domain analysis in this study ends at this point.

[7] Excluded from scope of this study.

2.2 Iterative rapid prototyping

We have developed a rapid pre-prototype of a user interface that integrates the functions of engineering, maintenance, setup, and operation. Thus, setting a comprehensive context for discussions with domain experts, we launched a second iteration of the pre-prototype to partially model manufacturing equipment, for further interaction with domain experts. The results of the experience are reported in Section 7.

2.3 Combining Top-down and Bottom-up Approaches

In designing a software architecture for reuse, flexibility and extension, the traditional top-down or outside-in approach is inadequate, as explained by Parnas [38], who proposes

> these (...outside-in approach and inside-out approach ...) as complementary approaches which must be used in some judicious combination according to the needs of the situation.

We have taken an approach consistent with this recommendation. Parnas [38] recommends

> ... begin with a specification of the *family* of objects one wishes to construct ... members must be highly similar items. To describe a broad family of objects we must describe a set of lower level mechanisms which will be common to all members ...

2.4 Method used in this study

After considering the methods and recommendations reviewed above, we have devised the following procedure for modeling elements of flexible manufacturing equipment:

S1: Consider some common configurations of flexible machines, as shown in Appendix D.

S2: Identify functions that these machines perform.

S3: Identify classes of physical components from which these functions and function-groups can be composed.

S4: Identify elemental functions from which the functionality of these physical components can be composed.

S5: Identify the conceptual primitives from which these elemental functions can be composed.

S6: Model the conceptual primitives.

S7: Build models of component functions and classes successively from the conceptual primitives.

The work-product of steps S3 and S7 will be classes of objects, from which it will be possible to compose many more subsystems and machines than contained in the starting collection. In other words, the process yields a generic set of models. However, to assure their adequacy, reusability, and extensibility, the process is incremental and iterative, as observed by Booch [9].

2.5 Method envisioned for validation

Next, we give a procedure to validate the adequacy of these models, although this study did not reach that stage.

S1: Identify test cases, e.g., physical components and their assemblages, deemed useful by users, but different from the ones originally identified.

S2: Build models of these physical entities on the conceptual framework to be tested.

S3: If some weakness is discovered in the underlying conceptual tree, examine the concepts to understand weaknesses.

S4: Revise the concepts as necessary.

S5: When a family of physical entities and their underlying abstractions are prototyped, review the models with some specialist with practical experience in applying that family of physical entities.

S6: Record the concerns, doubts and suggestions expressed.

S7: Validate these comments through similar reviews with others, as necessary.

S8: Revise concepts as necessary.

With this procedure, one can evaluate whether useful physical entities, assemblages, subsystems and machines can be composed, and whether the scheme can be followed easily with "natural" knowledge of the domain.

2.6 Approach taken for extensibility

Another departure from the published methodologies is the notion of a soft boundary for the chosen domain. The purpose is to improve overall lifecycle economics. The idea is to limit development-investment at the front-end by making the domain of applications extensible in small steps. We have followed the following procedure in this study, to arrive at a conceptual framework for extensibility:

S1: Establish overall scope and expectations of extensibility early on. Other Sections describe the small steps in which knowledge can be added.

S2: Establish a top-level architectural model of the information in the scope, including the extensibility.

S3: Selectively allow early start on "bottom up" approach in concepts where good generalizations already exist in technical literature, even though these generalizations might not have been exploited in practice.

S4: Involve domain experts in identification of additional useful generalized concepts.

S5: Take chances in some generalizations not finding great use later on; the cost of carrying unused concept-fragments is low relative to discovering later on that the framework is too narrow.

We believe that the conceptual framework, thus established, will reduce the disadvantages of iterative and incremental (bottom up) growth (Section 2.4), namely, difficulty of maintaining the conceptual integrity of the whole system, and the data integrity. Unfortunately, schema evolution may require additional manual effort, as the tools and technology are not yet adequate.

3 Bounding the domain

This is the second step in "preparing domain information" as outlined in Subsection 2.1. Also recall that one result of this step is the identification of a network of domains.

A key issue we address is an approach to incrementally evolve a model of a domain to achieve a specified level of performance with a given target reuser. As recommended by Arango [5], we strive for a systematic and incremental approximation to a definition of an ontology and semantics for this problem domain. Per Freeman [18], this is a long-range research issue. Our approach to the domain of manufacturing equipment may be viewed as a case-study in the path of long-range research efforts in software-reuse.

We want to focus on equipment for machining [8] automotive cylinder blocks, cylinder heads, transmission cases, and such other parts that fit within the requirements and bounds imposed by these parts. There is an emerging growth in the use of programmable servo-driven equipment to machine these products. The automobile and agricultural machinery industries produce these parts in the order of 7 million sets a year.[9] However, as these products evolve and their manufacturing processes change, we want our equipment models to be reusable and extensible at minimum life-cycle-cost. One major issue is that if extensive engineering is done in the beginning:

- It delays the start of benefits
- It increases the initial investment.
- It delays the start of validation (the history of reusability in software has not been very promising [41, 12]).
- Over the course of the elaborate engineering process, conditions may keep changing.

These factors make the life-cycle-economics of extensive front-end engineering less attractive (lower net present value; higher uncertainty). On the other hand, if too little engineering is done in the beginning, the reusability may be poor. If the original engineering team is in place to perform the next round, it may be able to provide a more enduring solution, reusing its prior experience. However, the more common situation is that the original team is not in place and the project budget and schedule do not allow for the additional investment to make the engineering reusable and extensible in the future. Therefore, we need an approach that yields an optimal balance between these two extremes.

We approach this study by organizing for extensibility whatever manufacturing knowledge is readily available in the beginning, as opposed to gathering or creating more knowledge. If extension turns out to be easy, we will consider the organization of knowledge adequate. If extensions are difficult, we will conclude that either the organization or the amount of starting knowledge or the tools might have been inadequate.

The initial amount of knowledge and its organization may turn out to be more than the minimum necessary for our objectives. We believe that the added cost is worth the reduction in the risk of discovering later that the conceptual foundation was inadequate.

[8] limited to milling, boring, drilling, thread-cutting

[9] from published annual production of automobiles.

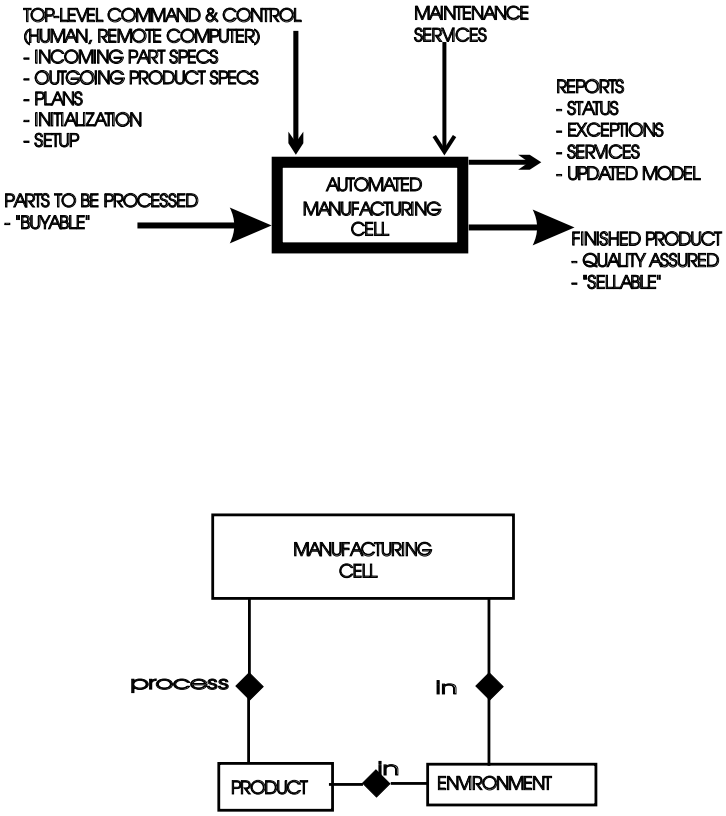
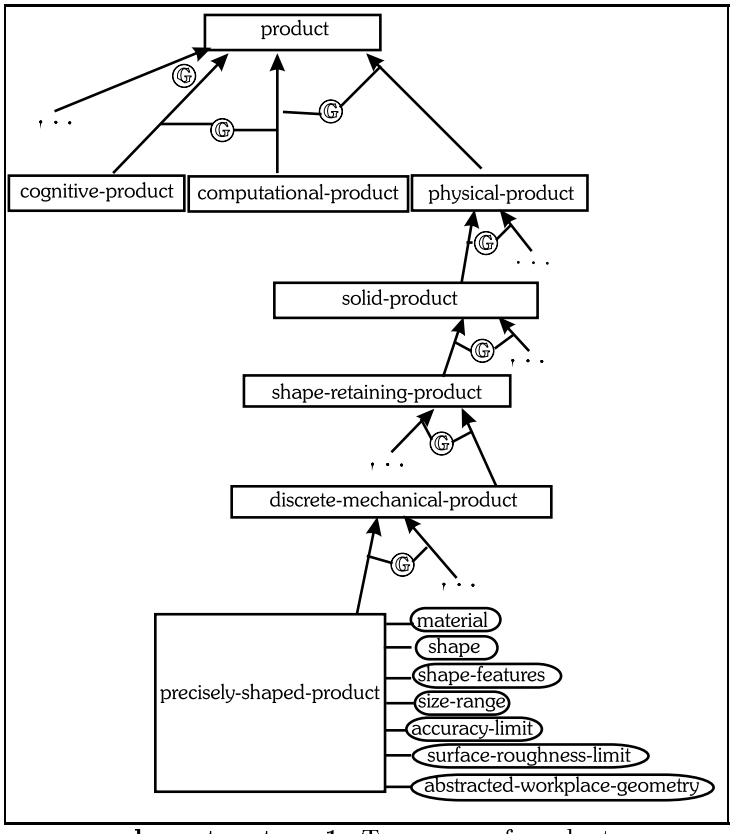


Figure 3: Scope of cell



class-structure 1: Taxonomy of product

The boundary of extensibility is represented in Class-structure 3, in terms of a system called a cell that processes some product(s) in some given environment. To bound the scope of this system, we bound the scope of the processes, the products, and the environment. Similarly, we plan for extensibility in terms of the processes that the cells will perform, the products they will accommodate, and the environmental conditions in which they will function.

3.1 Scope of manufacturing

We first establish a broad context in which we can set a relatively narrow starting scope, so that the underlying concepts can support potential future extensions of the scope.

The term *m*anufacturing is often used synonymous to the terms *producing*, *making*, and *processing*. For example, even the creation of software has been viewed as "manufacturing" software [30]. *M*anufacturing has been more commonly used in the context of making a product, which, historically, has been a physical product. However, software is also being viewed as a product. The views and usages of the term *m*anufacturing are changing with the socio-technological environment. In consideration of this trend, it would be advantageous to leave room for cognitive and computational processes and products, in the broadest definition of manufacturing, as depicted in Class-structures 1 and 3.

We limit the scope of this study to economically useful physical products and to processes that add value primarily through mechanical material-transforming action. Generally, many secondary processes support the primary process. The conceptual model should include the necessary supporting processes.

Referring to Class-structure 3, these secondary processes may be physical, computational or cognitive, or even the process of existence. *Cyc* [19] has associated the concept of existence with things, i.e., physical objects. It serves the purpose of associating a starting time and an ending time between which the entity exists in a certain context. It unifies the concepts of product and process.

Next, we further bound the scope of manufacturing by bounding the scope of the physical products manufactured and the physical processes performed on these products.

3.2 Scope of products

The product is a separate domain. We will attempt to define its intersection with the domain of manufacturing equipment. We hypothesize that focus on the abstracted properties of products (see Class-structure 1), will be adequate for our purpose. Later, it will be necessary to test the adequacy particularly for assessing the economic usefulness of manufacturing-equipment models.

Although, we could have defined, at the outset, the domain [10] chosen for this study, laying a wider foundation serves several pur-

[10] blocks, heads, cases

poses. First, the manufacturing equipment concepts that we want to model could have wider applicability than our initially chosen scope. For example, equipment for material-additive processes may be similar, but it may have to deal with the deposition of the material in a form outside the initial scope. Secondly and more importantly, the wider concept of physical-product allows us to capture the characteristics of certain byproducts of manufacturing shape-retaining products, e.g., dust, smoke, and chips, which have a twofold relationship with the manufacturing equipment. First, these byproducts affect the characteristics of the equipment. Secondly, the equipment has to handle these byproducts. This might involve reclamation and recycling in the case of material-additive processes.

Engine blocks, heads and transmission cases fall in the class of discrete mechanical products. The purpose of laying the broader taxonomy is to prepare for later extension of the manufacturing-equipment concepts to making other discrete, shape-retaining products, such as electrical and electronic parts.

It should be noted that the classes *s*olid and *s*hape-retaining-product require certain material properties. However, bounding the material properties alone is not sufficient to assure these properties.

By limiting the scope to a discrete, shape-retaining, solid, we can trace an instance of the product and its transformation through the manufacturing equipment. This allows us to simplify the domain of manufacturing equipment. Henceforth, we will refer to the product as a *workpiece* or, more commonly, *part*.

By limiting the scope to a precisely-shaped product, we require that the process be executed or successively repeated within close specified limits. By restricting the allowable variability or uncertainty, we would be able to formulate models of manufacturing equipment that match the real world more closely. The term *precise*(see glossary) is relative. Instead of providing an absolute definition here, we leave provision for bounding it through associated concepts (See section 3.2.6).

To review our scope-bounding strategy, we started with a target family of applications in mind. We began identifying parameters that allow generalization. We also began to lay successive stages of specialization-generalization of concepts, thus setting up potential stages of extension. Eventually, we have to test if the usefulness (scope) of our concepts is expandable economically.

ISO/STEP [24] has developed a taxonomy under shape-retaining products, with a subclassification for discrete mechanical parts. We adopt the ISO/STEP taxonomy.

3.2.1 Scope of shape

Shaping the workpiece to specifications is the main function of the manufacturing equipment on which we want to focus. Therefore, one aspect of bounding the domain of manufacturing equipment is by bounding the scope of overall shape of the workpiece and the shape of features on the workpiece that will relate to the manufacturing equipment.

Opitz [37] established a parts-classification scheme to organize the knowledge for processing these parts. This scheme has two main categories of workpiece shapes: *p*rismatic and *c*ylindrical. The scheme also has other classes of part-shapes.

We adopt Opitz' scheme for classifying parts. Prismatic parts account for two-thirds of the machining equipment used [8] and account for the bulk of the variety in equipment configurations. The large amount of potential usage provides the economic attraction. The large variety provides the attraction that the initial concepts will be more easily extensible. Transmission cases, engine blocks, and cylinder heads are examples of prismatic parts.

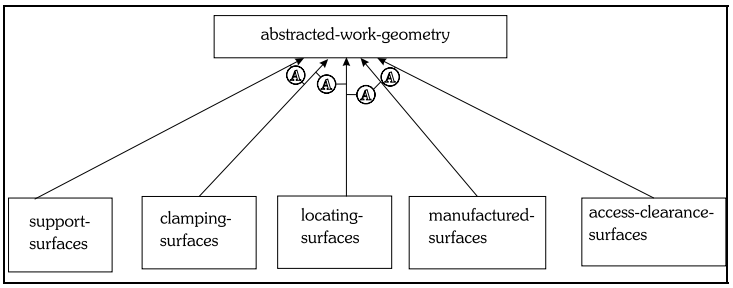
Prismatic parts also have cylindrical machined shape-features, e.g., bores and counterbores. There are commonalities in the processing of these features with the processing of cylindrical parts. We include a broader classification in the scope, in order to extend use of abstractions that are common to different types of part-shapes.

3.2.2 Scope of shape-features

A stronger commonality across different part-shapes can be found in the shapes of the manufactured features. Kramer [26] has compiled a catalog, which we adopt. Within these features, we limit the initial scope of manufactured features to shapes that are planar, cylindrical, conical, a section of a torus generated by a circular arc, or a composition of these surfaces. Practically all machined surfaces in engine blocks, heads and transmission cases fall within these bounds. This allows us to start with simple motions required from the manufacturing equipment, which, in turn, simplifies the models. However, it should be noted that the surfaces actually produced seldom conform perfectly to these nominal shapes. Therefore, monitoring and assuring quality, for even these simple surfaces will still be quite complex.

3.2.3 Abstraction of workpiece geometry

For the purpose of modeling manufacturing equipment, we abstract the workpiece as a rigid body consisting of manufactured surfaces, locating surfaces, clamping surfaces, support surfaces, and access-clearance surfaces, as shown in Class-structure 2.



class-structure 2: Abstraction of workpiece geometry

The combination of locating, supporting and clamping functions will be called *w*orkholding or, more commonly, *f*ixturing. The workpiece may have multiple sets of workholding surfaces for use in different setups. The workpiece may also have secondary fixturing surfaces to be used for handling. We limit the scope of these surfaces as defined below.

Manufactured surfaces are surfaces that will be processed by the manufacturing equipment to be modeled. We have al-

ready established a bound on the shape of manufactured features above.

Locating surfaces are surfaces that will be used in the manufacturing equipment to establish the position and orientation of the workpiece. Locating surfaces shall provide repeatability consistent with the accuracy required in the manufactured features. This is typically accomplished through a kinematically sufficient and non-redundant scheme, commonly known as the "3,2,1 locating scheme" in which three locating points establish the first locating plane, additional two points establish an orthogonal second locating plane and a third locating point establishes the third orthogonal plane, thus establishing a unique position and orientation of the workpiece.

Clamping surfaces are surfaces that will be used in the manufacturing equipment to secure the workpiece in the established position and orientation.

Support surfaces are surfaces that will be used in the manufacturing equipment to support the workpiece, in order to minimize its deformation under its own weight or under processing loads.

Access-clearance surfaces are surfaces that the manufacturing equipment, fixture and tooling will encounter in order to access the manufactured surfaces, locating surfaces, clamping surfaces or support surfaces. We limit the scope to workpieces in which the access-clearance surface can be modeled as a plane or a cylinder bounded by two planes, or a composition of such planes and cylinders. This restriction allows simplifications in geometric aspects of the manufacturing-equipment model. For example, the enveloping surfaces of a prismatic part could be modeled as planes enveloping all points in the workpiece, for the purpose of computing interferences with the manufacturing equipment.

We deliberately omit, i.e., "abstract away" geometric details about the presence and absence of material elsewhere within the workpiece, because that very detailed information is not necessary for the purpose of bounding the domain of manufacturing equipment. Although abstractions of the manufactured surfaces (under such names as form features or part features) are being used in process planning systems, abstractions of certain other surfaces, e.g., access-clearance surfaces, have not been found in literature on manufacturing research.
We limit the scope to workpieces in which other useful properties can be simply related to these surfaces. By *useful* we mean properties that affect the behavior and performance of the manufacturing equipment during operation. Examples of such properties are mass, inertia, stiffness, damping, thermal deformation and heat transfer. By *simply related* we mean that the purpose would be adequately served by such mathematical simplifications as:

- lumping mass or inertia at a single point
- lumping stiffness and damping at a few points
- lumping load at a point or treating it as some simple distribution
- uniform distribution of certain properties
- treating locating, clamping and support surfaces as points for the purpose of kinematics and dynamics
- treating contacts at these locating, clamping and supporting points as simple springs and dampers
- treating the workpiece as a beam or a plate between points of load-application and support.

3.2.4 Scope of workpiece material

We limit the scope of the material of the workpiece such that the following assumptions hold for the purpose of modeling the performance of the manufacturing equipment.

- The workpiece is a perfectly elastic body.
- Material is homogeneously and continuously distributed over its volume.
- The material is isotropic, i.e., the elastic properties are the same in all directions.

Structural materials do not satisfy the above assumptions completely [51]. Yet they are a part of well-established engineering practice for limited specified purposes. Steels, high-quality cast irons, aluminum and aluminum alloys are examples of commonly used materials in machined products for which these assumptions can be made.
Cyc [19] has a taxonomy of substances, including materials, which we adopt at the top level. STEP [24] provides a taxonomy of materials and their properties, which we adopt under the substance model. Materials for the workpiece, cutting tools and components of the manufacturing equipment will branch off under this taxonomy. Additions for machinability properties may be necessary, if the STEP model does not include them in time.

3.2.5 Scope of size

Most automotive transmission cases, engine blocks and cylinder heads fall within a cuboid smaller than 500 mm side and larger than 100 mm side. So, we want the equipment models to cover at least that range. Machines for this size range have existed for decades. There is significant experience about the behavior of these machines. Although this experience is not formally documented, it forms the implicit basis for the models used in current engineering practice. We do not know how valid these models are in different size regimes, e.g., microscopic parts or very large space structures. So we limit our initial scope to the size range that fits between the 100 mm and 500 mm cuboids arbitrarily. If an application outside the bounds is considered, then these models would be re-examined for their sensitivity to the extension.
The smallest machined feature commonly found in engine blocks, heads and transmission cases is a hole of approximately 1.5 millimeter in diameter. So we limit the scope to holes of diameter at least 1 mm (going a little beyond limits of current practice). Once again, the purpose is to limit ourselves to available experiential knowledge. We avoid unknowns associated with miniaturization. Examples of grounds for concern, as the tool diameter reduces, are:

- Structural-support properties of the tool reduce exponentially with the diameter.
- Cutting geometry and clearances cannot be maintained in the same proportion as with larger diameter tools. Therefore, conventional drilling process models may not hold.
- The equipment may not be able to sense the weaker cutting process signals well enough for monitoring.

3.2.6 Scope of accuracy and finish

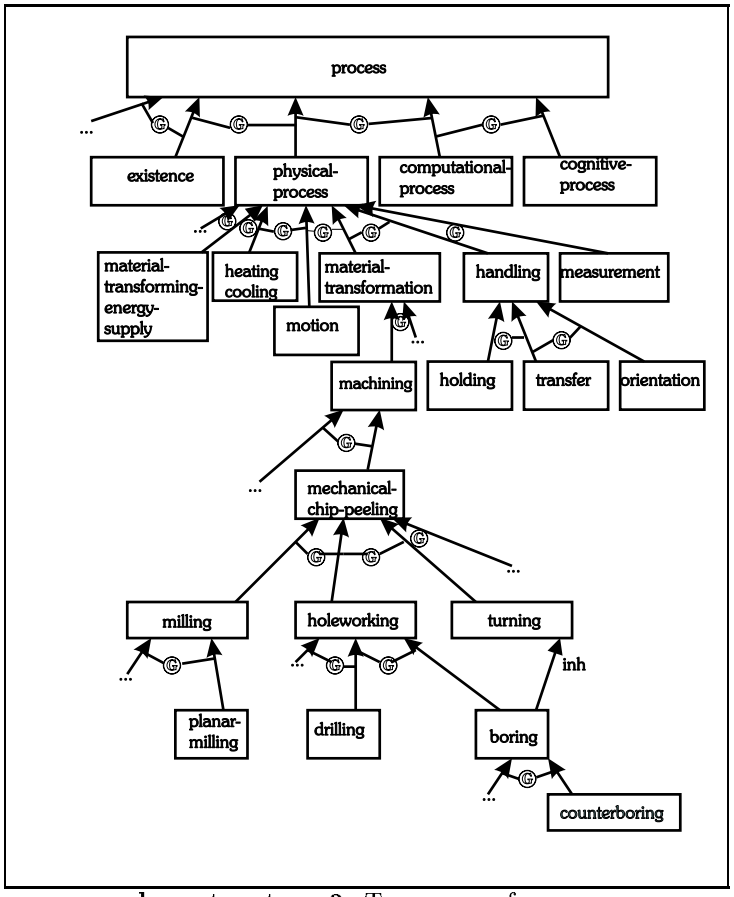
Following the same rationale, we limit the accuracy levels to those we have experience with, in engine blocks, heads and transmission cases, as given in Table 1

Table 1: Scope of tolerance limits

Parameter of machined feature	limit (microns)
Dimension of a bore	2
Cylindricity of a bore	1
Distances between bores	3
Alignment of bores	1
Dimension of a face	3
Flatness of a face	2

These tolerances, when associated with the bounds on size, exclude "super-precison" manufacturing, which, in turn, relaxes constraints on the accuracy of the equipment models.
Roughness of a surface shall not be required any finer than 0.7 micrometers (arithmetic average). This limit gives a crude indication of the dynamics-related information-content that could be useful in the equipment-model. Models of tool-to-workpiece dynamics will require re-evaluation for validity if finer surface finishes are to be monitored and assured automatically, during operation.
These limits on accuracy and finish represent the bulk of current machining practice. A crude approach was taken above, to avoid initial engineering expense. The numeric values given above are indications of *order of magnitude* rather than absolute limits. Other metrics for size and accuracy might have been more appropriate. It will take considerably more engineering effort to improve upon the benefit from the information. Such an investigation would be appropriate for a future iteration. For example, some factors for consideration could be the relationship of: "workpiece size to tolerances on machined surfaces", "manufactured-feature-size to accuracy", and "slenderness ratio of a bore to its accuracy".
A taxonomy of these concepts is still evolving in European research using AI concepts in machining [53], where the attributes identified for a hole are shown below.
hole:

nominal diameter
tolerance grade
tolerance position
positional tolerance
surface roughness
type of material



class-structure 3: Taxonomy of process

3.3 Scope of processes

Machining operations used in blocks, heads, and cases fall in the mechanical-chip-peeling class, in Class-structure 3. It accounts for the largest proportion of all equipment for transforming shape of mechanical parts [8]. In Class-structure 3, we also show processes that support machining. These attendant processes are also within the scope.
Our taxonomy separates certain processes that are not specific to machining processes. Although they are secondary to machining, they allow us to form a view of equipment characteristics in a way that can be applied to some other manufacturing processes.
Although motion processes supply the energy for material-transformation in machining, motion processes are also involved in non-machining functions. Therefore, in our taxonomy, Class-structure 3, we did not classify motion under *m*aterial-transforming-energy-supply, which then includes other forms of energy that the equipment supplies for transforming material.

3.3.1 Scope of machining processes

We start with those machining processes that are currently used in such prismatic parts as engine blocks, heads and transmission cases, as shown in Class-structure 3. Cylindrical parts employ

some of the same processes, but their main process is turning. Common abstractions of process models can be found for boring and turning processes, provided the bores are large enough. To the extent of capturing that commonality, we also include turning in the scope. This example is typical of the approach taken for domain-extensibility.

3.3.2 Scope of heating/cooling processes

Although heat is a secondary process, it plays many significant roles, which we define below as the scope of our interest in heating/cooling:

Limit deformation It is a major source of dimensional deformation that affects the operational accuracy of the manufacturing equipment. Our interest is in monitoring the temperature rise and in limiting the temperature-fluctuation for the purpose of estimating this deformation. Fluctuations may be minimized by addition, removal or redistribution of heat.

Limit degradation Excessive temperature rise (and secondarily the accompanying structural deformation) accelerate the degradation of equipment [13] (e.g., excessive wear in distorted bearings, detereoration of fluids, life-reduction of electronic parts). The scope of our interest is in monitoring and limiting the temperature rise.

Detect abnormality Excessive temperature rise or abnormal temperature distribution could also be monitored to yield prognostic and diagnostic information about the cutting tool and machine elements [47, 54]. The scope of our interest is in monitoring the temperature rise, distribution and trend.

Class-structure 4 is a taxonomy of the different thermal processes of concern in the domain of manufacturing equipment for machining processes. Heat-generation processes are classified by their sources. The location and model of heat-generation can be associated with the respective sources. Heat-transfer is classified by more basic modes. A different model can be associated with each type of process. That process-model can be associated with an appropriate location, e.g. the whole machine space, or the mounting interface between a source and the rest of the machine, etc. The corresponding models have not been elaborated in this study. However, we have craeted a base for extensibility here.

heat-generation (from energy-transformation)
- cutting-process
- rolling-element-bearings
  - spindle-bearings
  - drivescrew-support-bearings
  - ballscrew-nut-pair
  - ...
- sliding-bearings
- electrical-losses
  - resistance-heating
  - servo-motors
    - brushless-DC-type
    - ...
  - spindle-drive-motors
    - DC-motor
    - AC-motor
    - ...
- ...

heat-transfer
- radiation
  - hot chips
  - solar gain
  - ...
- convection
  - air circulation around machine
  - ...
- conduction (e.g., surface contacts)
- phase-transformation
  - evaporative cooling
  - ...
- mass-flow
  - cutting coolants
  - lubricants
  - ...
- ...

class-structure 4: Thermal processes in machine

3.4 Scope of monitoring and control

Monitoring and control are subclasses of computational and cognitive processes. They are complex and vary in scope considerably, across applications. Therefore, we state the broad purpose of monitoring and control here. Since they are also compositions of other computational and cognitive processes, we define the corresponding scope of those processes below.

The purpose of computer-automated monitoring and control of equipment is to allow manufacturing of parts safely, correctly, and productively and to stop operation of the equipment if it is not able to provide this service. The computer-automated system should support the maximal utilization of the total investment, to the extent that given causal laws allow. Maximizing utilization includes minimizing the execution time, and the parasitic losses, e.g., due to failures. Models in Section 5 provide the structures to set the appropriate limits and states.

3.5 Scope of computational processes

The conceptual model of manufacturing equipment should support preprogrammed computational tasks for monitoring and control of the manufacturing equipment. By "preprogrammed" we mean that the program is available before its execution is needed. The computational tasks are typically cyclic or periodic.

Examples of computational processes to support monitoring are:

1. Acquire value of some variable sensed in the controlled system.
2. Collect a prescribed time-history of such values.
3. Reduce or condense this time history to some meaningful parameter, in accordance with prescribed procedure. These procedures may use equipment models.
4. Compute the expected value of the sensed or derived parameter. These computations may use equipment models.

5. Compare the sensed value or the derived parameter with a corresponding expected value.
6. Compare the difference or deviation with allowable or prescribed limit.
7. Trigger prescribed action upon reaching or crossing such limit.
8. Store the intermediate computational results as prescribed. The prescription may include further reduction procedures and the maintenance of a time history. These reduction procedures may use equipment models.

Examples of additional tasks for control are:

1. Acquire value of some controlled variable or parameter from prescribed plan of execution (typically decomposed from a program for processing workpieces).
2. Decompose or transform this acquired value to values of variables/parameters to be controlled by execution agents. These transformations would use equipment models.
3. Distribute/transfer the values to these execution agents. The ultimate resulting values are set as outputs to some controlled actuators in the manufacturing equipment.

3.6 Scope of cognitive processes

We bring out the scope of the less structured knowledge-acquiring computational tasks under the heading of *c*ognitive tasks.
One class of such tasks is known as machine learning. Our scope of machine learning is limited to the fitting of parameter values in previously prescribed models, using prescribed model-fitting procedures (external to the equipment model). The purpose is to support the tasks of monitoring, control, prognostics, preventive maintenance, diagnostics, corrective maintenance, and enhancement or engineering improvements. The data for such machine learning may come from operational data or from controlled calibration tests, performance-evaluation tests or other engineering experiments.
The conceptual model of manufacturing equipment should provide the structure of the models needed for such machine learning. Causal laws are given in terms of parameters, but generally the exact values of parameters are not known. However, these values can be estimated with a combination of controlled calibration experiments. The given causal laws also allow estimation of the reaction time needed for each physical function to be serviced and the response time needed by the physical serving mechanism.
Equipment models should also support learning about perception. Most of the time the perception is not at the point of interest, but at some remote location (generally the closest available real-estate that provides maintenance access). Thus, feedback is correspondingly distorted and contains systemic errors, uncertainty and noise of measurement, in addition to similar deviations from the monitored process. Established causal laws and quantitative information do not allow clear isolation of these factors. Our objective is to have the system perform optimally (at the most economic level possible) and to protect itself, as much as possible, from adverse effects of the unpredictable elements. Recall that our objective is to minimize the time, cost and errors in realizing a specific implementation of such a control and monitoring system. In other words, the scope of the cognitive processes is to learn from operational data by applying available knowledge, but to generate and signal an alarm when it crosses some prescribed threshold of "ability to learn". Over the course of time, human review of the history of automated operation and learning is expected to yield improvements in these processes, thus changing the limits of "ability to learn".
Cognitive processes should support simplification and selection of models of other processes by association with the contexts and by limiting the accuracy to what is needed for the purpose. For example, we consider the case of heating or cooling. The following different reasons for cooling impose progressively more difficult modeling and computation:

1. maintain safe temperatures
2. avoid degradation of machine elements and fluids, and
3. maintain operational accuracy (most difficult).

Appropriate estimations of heat-generation and heat-transfer rates are needed to design cooling capacity in the system. For operational accuracy, on-line control of temperature would also be beneficial. Feedback control is not sufficient, because of the long time-constants in temperature-changes. Therefore, some degree of predictive control would be necessary. Some of the same engineering information that was needed for the design of the cooling system capacity could be used in predictive control, along with on-line information about the different operational speeds and loads that affect the heat-generation rate in various parts of the machine. This collective information makes it possible to regulate cooling and even active heating to minimize the fluctuation of temperatures. Simply balancing heat-gain and heat-removal could reduce much of the fluctuation. The initial models would provide the knowledge to predict and account for such first-order effects. The modeling framework should also provide the facilities to improve upon these models as experience is gained.

3.7 Scope of physical environment

We identify factors in the physical environment that will affect the models and the performance of the manufacturing equipment. Isolation and abstraction of these factors would allow us to extend models correspondingly when the need arises to extend beyond these limits. However, there are economic and technological limitations to such extensibility. There are similar limits on how well the environment can be controlled or compensated for. These factors are illustrated below with examples, providing the rationale for limiting the scope of product size and accuracy.
We limit the scope of equipment modeling to constraints on environmental factors given below. These limits are consistent with the nominal specifications in most industrial-scale manufacturing plants. Many of these specifications are not met in practice. We explain below the nature of these deviations. Equipmentmodels should be extensible to cope with these deviations.

Mounting surface for the equipment is stable, i.e., it provides a stable coordinate frame of support and reference for position and orientation and it isolates the equipment from shock and vibration, within limits that the process will tolerate. (This constraint is not explicit). However, perfectly stable floors

will seldom be available. Best achievable stability will be too costly for the ordinary applications. So the constraint has to be relaxed for less demanding processes. Only crude rule-of-thumb guidelines are available to practicing manufacturing engineers. So, the initial course we chose was to impose specific numerical limits on the accuracy goals on products (see Section 3.2.6).

Ambient fluid surrounding the equipment is air. However, it should be possible to extend equipment models to other surrounding fluids, if it has been designed for such operation and if the needed design information is available. In many applications, the work-zone is submerged under a flood of cutting fluid, either to remove process heat or to maintain uniform temperature distribution or to remove chips and other process debris or to provide lubrication in the process or to serve as an electrolytic fluid. Even though such applications have been running in production for a long time, there is little documentation available on the performance models of equipment submerged in these fluids.

Ambient temperature may vary over the range of 55–85 degrees Fahrenheit, but bulk air is supplied to maintain an average within 65–75 degrees. In practice, actual temperatures fluctuate over a wider range. Although it would be desirable to have equipment models extensible to cope with these deviations, it is very difficult. Therefore, we take the conservative, though less widely useful, approach of limiting the scope to the 55–85 degree range. We explain the various sources of larger temperature fluctuation and the resulting compromises below.

1. Vertical stratification occurs. Temperature at the floor is lower than temperature at the roof of the plant. The variation can exceed 20 degrees F. This stratification affects tall machines significantly. The phenomenon is not predictable enough to compensate for its effects. This factor imposes an upper limit on the height and size of equipment for which the equipment models would be valid.
2. Solar heat gain from windows and skylights raises temperature of equipment non-uniformly. The resulting distortion is significant on large machines. The temperature rise varies with the position of the sun, cloudiness, etc. So it has not been practical to compensate for its effect. This factor also imposes an upper limit on the size of the machine for which equipment models would be valid.
3. Winter shutdowns allow the temperature to drop considerably below the 55 degree limit. Some plants shut down the heat totally. The resulting shrinkages are severe. There is differential shrinkage between the floor and the machine. Upon restoration of normal temperatures, the machine does not return to the original relationships and locations. This affects operational accuracy. The phenomenon affects larger machines more significantly. A second effect is condensation within the machine elements and its fluids, making performance unreliable and accelerating degradation.
4. Summer shutdowns have an analogous effect on distortion. Additional effects may be:
   - Degradation of fluids with microbeal activity accelerated at the higher temperatures.
   - Drying and caking of dirt on machine elements; it increases resistance to motion and accelerates degradation.

Gravitational force is constant.

Relative humidity is maintained at an average of 50%, but does not exceed 80%. As explained above, this constraint is not held in practice during shutdowns.

Work-stoppage conditions will be kept close to steady-state operating conditions. In current practice, equipment performance models are most valid when the equipment has been running under steady-state operating conditions for some time. Work stoppages make the performance less predictable. The longer the stoppage the greater is the uncertainty until steady-state-stoppage conditions are reached. Examples of sources of uncertainty are:

1. Changing temperature-distribution across parts of the machine, changing the thermally-induced deformations. It would be desirable to extend the equipment models to compensate for predictable and measurable changes in temperature distribution.
2. Changes in preloads, and, as a result, changes in stiffness and damping properties, resulting from changes in temperature distribution. These effects are more difficult to predict. Therefore, we rule them out of the intial scope.
3. Changes in lubrication of moving parts. The effects are very difficult to predict. Therefore, we impose a constraint that joint-lubrication will be maintained during work-stoppages. Industrial-scale plants follow this practice for short work stoppages. However, the practce is not widely followed for overnight stoppages. Thus, our models will not be valid for the startup phase after such stoppages. The compromise is being made in view of the technical difficulty.

Longterm shutdown effects are outside the scope, i.e., if the equipment is not operated for a length of time, the models may not hold valid. After such a shutdown, properly engineered startup and revalidation procedures must be performed on the equipment. The interpretation of *longterm* is dependent upon the design specifications of the equipment. The longterm-shutdown effect of most concern is the effect on moveable joints in the equipment, i.e., *seizure* or "freezing" of the bearing surfaces.

Electrical power supply, EMI, RFI shall be within design specifications of the manufacturing equipment. Although most plants have such nominal specifications, often significant deviations occur. To that extent the equipment models may not be valid.

Thus, we have reviewed a number of environmental factors which are likely to cause the behavior of the equipment to be different from the predictive models. In many cases, these conditions will have to be carefully validated before applying the models. In some cases, the models can be structured for extension to account for additional variables.

3.8 Scope of manufacturing equipment

We confine the scope to equipment that has been designed with the intent to satisy the following conditions:

1. It will operate properly in the environment of the plant.
2. It is protected from emissions of the manufacturing process, e.g., chips and dust.
3. It is observable and controllable [39].
4. Its behavior within its operating range may be approximated as a linear system.
5. The noise in signals from continuous processes, e.g., position feedback in servo-controlled motion, may be treated as white Gaussian noise.

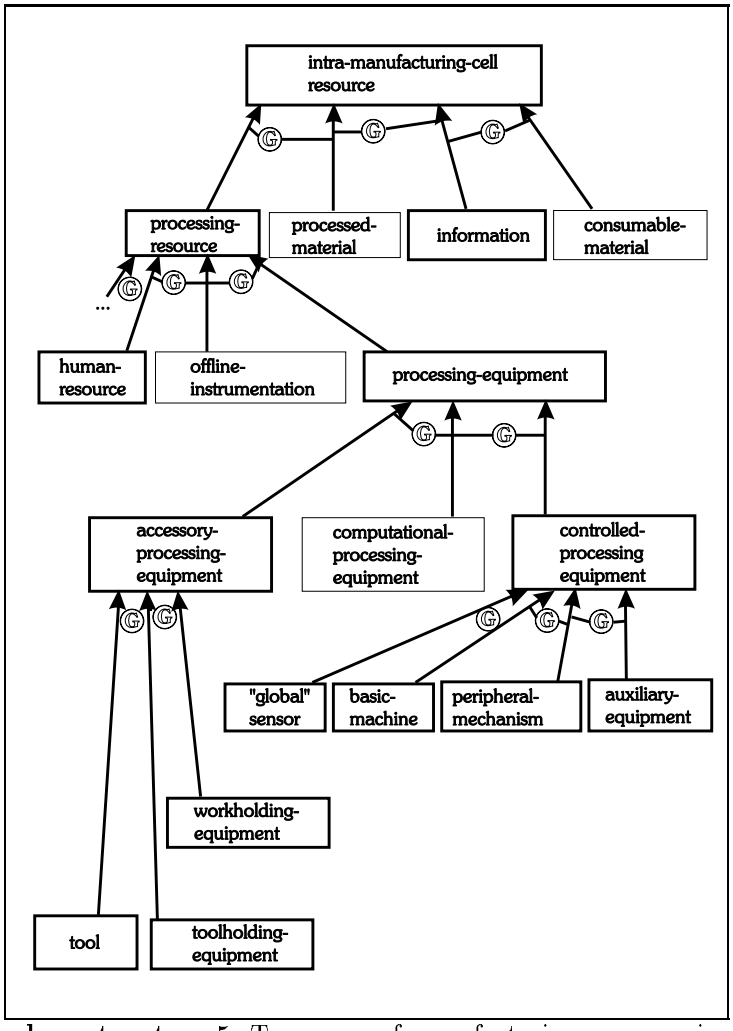
These conditions often do not hold in operation. However, they are an appropriate starting point, because they the basis of current design practice. Much of the time these assumptions are found to be workable, because the product and the manufacturing process are not sensitive to the violation of these assumptions. In Section 4 we develop a part of the conceptual framework that will allow extension beyond the assumptions of linearity and white, Gaussian noise.

Recognizing that we have already constrained the scope to equipment for the products and processes in the scope defined above, we now consider equipment-specific characteristics.

In Class-structure 5 we show a taxonomy of manufacturing resources within a typical machining cell, Class-structure 3 decoupled from the processes such equipment might implement. Our initial scope is limited to the *b*asic machine, the resource that processes the workpiece. Examples of *p*eripheral-mechanism are tool-changer and work-changer. Examples of *a*uxiliary-equipment are: hydraulic-power-unit and oil-cooler. We provide for extensibility through the generalization hierarchy of *c*ontrolled-processing-equipment, *p*rocessing-equipment, *p*rocessing-resource ... For example, there could be commonality between models of elements of a servo-driven basic-machine and a servo-driven, robotic tool-changer.

3.8.1 Correspondence of function and constituent

Most computer-automated machining cells in use today can be functionally partitioned as shown in Class-structure 5. The basic machine provides relative motion between the tool and the workpiece. We confine our scope to basic machines which will provide an unambiguous correspondence between a monitored or controlled function and a constituent unit (see Class-structure 7). We believe this constraint is reasonable in the context of machining systems for industrial-scale production as in the automobile industry. Since such systems need high reliability and maintainability,



class-structure 5: Taxonomy of manufacturing resources in a machining cell.

the construction is modular. It is desirable to make active components and subassemblies individually replaceable and testable. It is natural to have their corresponding controls modular, to facilitate diagnosis, testing, and calibration. This justifies the constraint of unambiguous correspondence between a monitored or controlled function and a constituent unit. Nevertheless, we recognize that there are some well-known phenomena in which a function does not purely correspond to one constituent. For example, a prismatic joint with the nominal function of providing translation in the direction of the x-axis may also have undesirable (but very small) motion in other directions. We will present an approach to accommodate such deviations in our conceptual model, as long as the deviation can be usefully approximated as a function of the motion of an axis-group.

In the cases of less accurate machines and certain special-purpose machines, a one-to-one correspondence between function and constituent may not be the design intent. Therefore, our conceptual model will limit its initial scope to exclude such machines.

```
workstation
    basic-machine
    peripheral-mechanism
    auxiliary-equipment
    workholding-fixture
    tooling
        cutting tool
        toolholder
        ...
    "global" sensor, i.e., not included in entities above
        tool-to-workpiece-position-sensor
        tool-to-workpiece-load-sensor
        ambient-temperature-sensor
        ...
    ...
```

class-structure 6: Constituents of an automated machining workstation

```
substructure or axis-group
    translational-joint
        joint-pair
        drivetrain
            leadscrew
            pulleys, gears ...
            coupling
        drive
            amplifier
            motor
        feedback-sensor
            position-sensor
            velocity-sensor
            current-sensor
            temperature-sensor
    rotational-joint
    spindle
    ...
```

class-structure 7: Constituents of a programmable servo-controlled basic machine

3.8.2 Scope of kinematic configurations

We limit the scope to kinematic non-redundant mechanisms that can be modeled in terms of the ISO standard for representing kinematics [22]. Furthermore, we limit configurations to one of those categorized in Appendix E of [36].

We limit our initial modeling-scope to machines, Class-structure 7, equipped with programmable servo-controllable actuation, as well as perception of pertinent state. Most common state information is the servo-position and velocity (often derived from position). Information of average actuation power is also usually available. Temperatures of external machine surfaces and fluids are also available at relatively low expense. Information on forces is often only indirect.

3.9 Review of extensibility of domain

We have identified some potential extensions that would cover the manufacturing of a wide range of discrete mechanical, optical and electronic parts. By identifying such potential extensions, we will be able to explore appropriate conceptual primitives at early stages of the project. We have also identified a possible series of progressive increments of extension that we describe below.

1. Machines of other configurations, but composed of commonly-used building blocks. See Appendix D.
2. Precision machines performing different types of functions on different types of discrete shape-retaining parts:
   - positioning, orientation and other manipulation of discrete mechanical, electrical, optical or electro-optical parts;
   - measurement of dimensions[21], locations, orientations (relative and absolute);
   - assembly of such parts by placement or insertion at predetermined absolute or relative locations;
   - combinations of such functions in the same work-cell.
3. Machines with lower precision where the repeatable performance, relative to non-repeating deviations, is not as high as in precision machines. The signal-to-noise discrimination problem becomes more severe, and representations become more complex.
4. Monitoring, tracking, estimation and use of changing dynamic and kinematic characteristics such as friction, stiffness, backlash, geometric errors of motion, and thermal deformation[21].
5. Inclusion of fixture, workpiece, tooling, and such attachments, to extend the scope of the system from the basic machine to the workstation level and then to the cell level.
6. Processing control objectives other than those specifiable in traditional numerically-controlled machining:
   - dimensional metrology of parts in process;
   - machining or assembly of parts to relative dimensions determined in process;
   - machining or assembly of parts controlled by constraints of cutting forces or by some user-defined objective function;
   - machining or assembly of such parts using combinations of control strategies mentioned above switching control strategies under prescribed rules.
7. Other sensors that may be used to enhance dimensional accuracy, performance, reliability or availability of the manufacturing cell.

4 Architectural model

By architecture, we mean the fixed structure(s) from which specific applications can be composed with minimum additional engineering time and cost. The term *fixed* is used relative to a time scale, i.e., the architecture is reusable or amortizable over a number of application-development cycles.

Biggerstaff [7] concludes that for any significant reuse of software components it is essential to have an architectural standard for the domain over which the components are to be reused.

Krueger [27] ranks architecture second among eight different approaches to reuse, when the objective is to reduce the intellectual effort required to go from the initial conceptualization of a system to a specification of the system in abstractions of the reuse technique. Application generators, the highest ranking reuse-approach, also owe their success to a standard architecture of domain semantics. Whereas application generators have been successful in very narrow domains, we can widen the applicability with a systematic evolution of an architecture for the application domain.

Recall that a conceptual model must be defined in relation to its purpose. The domain defined in Section 3 establishes the overall boundaries on the purpose and contexts. Even then, the information available within the system is very broad. We resort to a reference model architecture, for the application domain, for further organization and partitioning of information in the system to provide more accurate, timely, effective and efficient usage. There are also many fundamental issues in correct and timely inferences.

4.1 Some requirements addressed in the architecture

We present our approach to the architecture in relation to some fundamental research questions about organizing knowledge for correct and timely inferences:

Q1: How should contexts be set up to help narrow the inference process? (The hierarchical architecture, shown in Section 4.2, sets up the contexts).

Q2: How should knowledge be represented and organized for timely prediction and recovery? [43] (The concept of temporal span, shown in Section 4.6, provides for specification of the duration for which information in an architectural branch is valid).

Q3: How should knowledge be organized to extract the con- tent of the sensed information? Schank [43] puts this question under the *Inference* problem. (Appropriate contexts are set up in the hierarchy of the architecture).

Q4: How should knowledge be organized to draw conclusions from disparate data, e.g., sensed information? Schank [43] treats this question as a *generalization* problem. The concept of contexts is mentioned in [19, 29], but they did not report any general solutions or any methodology to determine what contexts are adequate and sufficient for the purpose.

Q5: How should knowledge be organized to apply relevant case experience in prognostics and diagnostics about the controlled system? This question is linked to the previous question. Forbus [17] defines the issue as *merging measurement interpretation with explanation.*

4.2 Hierarchical structured systems

Many notable scientists strongly recommend an hierarchical representation [10, 32, 35]. It is a well-established approach to manage complexity. Still there is controversy in industry and the concept has not been put into practice explicitly. We offer the following additional arguments in support of an hierarchical representation:

1. The underlying body of general engineering knowledge is hierarchical, though the knowledge used in a particular system is often not explicit. Lytinen [29] brings out the notion of organizing structures for causal knowledge that comes from more general causal laws, so that the same knowledge can be applied in different situations.
2. Such physical systems are engineered in an hierarchy of constituents. Therefore, an information hierarchy corresponding to the constituent hierarchy becomes natural for some of the information, as depicted in Figure 4.
3. We have limited our scope to machines with a correspondence between constituents and their functions and behavior in Section 3.8.1.

Our conceptual model is based on deploying the following types of hierarchy in organizing the information:

- Generalization/specialization hierarchy
- Constituent hierarchy (or aggregation)
- Task (decomposition) hierarchy
- Resolution-relevance (spatial, temporal) [23]

NIST [2] has demonstrated the workability of such an architecture. Their reference model architecture for intelligent control provides a good starting point for a hierarchy. From this reference model, we constructed Figures 4–2, and Table 3.

4.3 Nested hierarchical control

Meystel [23, 33] describes the concept of nested hierarchical control. He has applied it to unmanned mobile robots in an environment considerably less structured and less predictable than the environment for our manufacturing applications. Thus, we postulate that the concept of nested hierarchical control, as shown in Figures 4–5, would not be overly restrictive.

From Figure 4, taking Level 1 as an example nesting, we show its internal structure [2] in Figure 4. It consists of:

TD Task decomposition, which includes the functions:

- Assign task or job
- Plan
- Execute the plan

Albus [2] places the knowledge required for each task in task frames.

SP Sensory Processing, i.e., monitoring or detection or perception about the controlled entity.

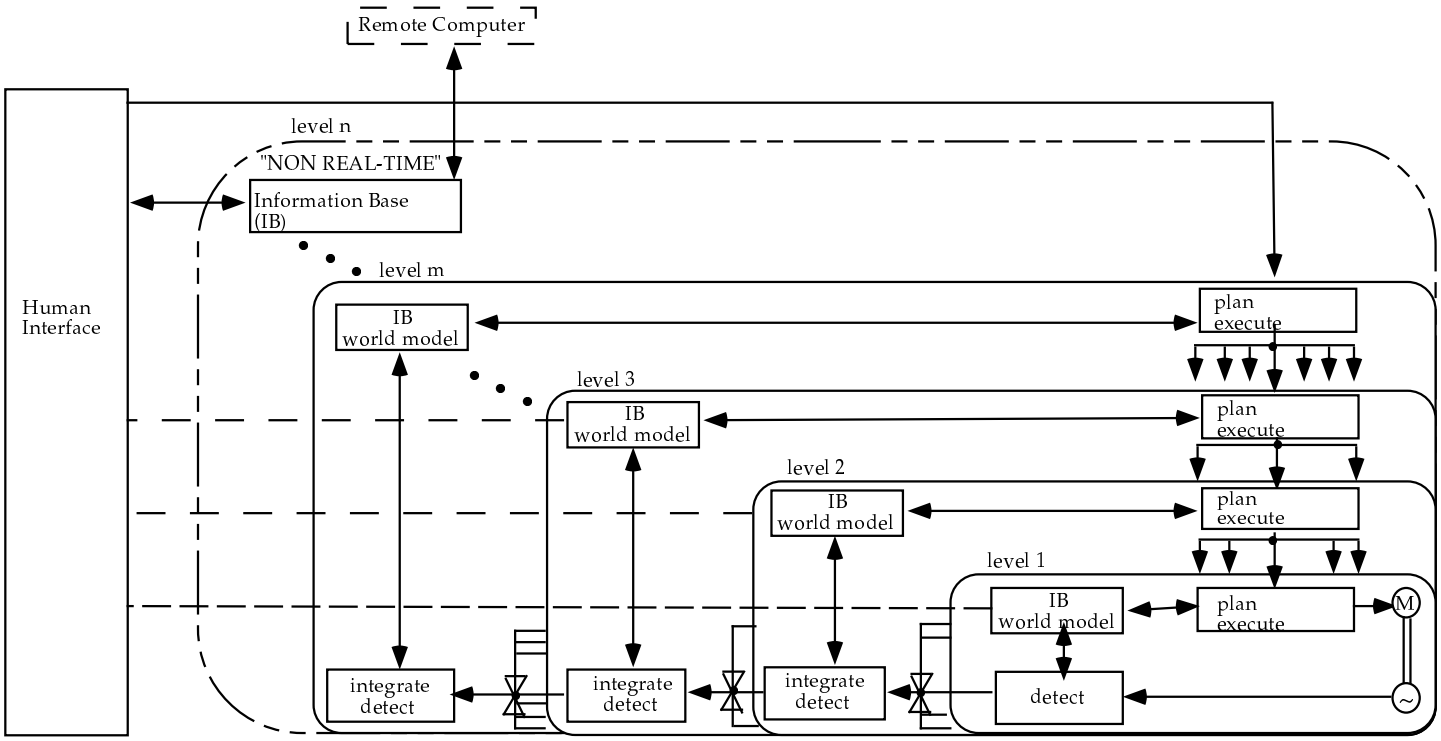


Figure 5: Control hierarchy for an integrated manufacturing system at a cell level.

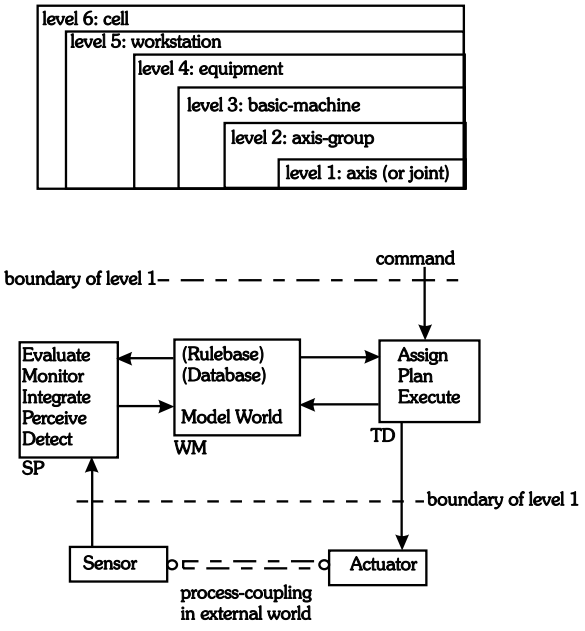


Figure 4: Levels of control and tasks within a level

WM World modeling, i.e., the state of the controlled entity. It may use and update prior information in the system. Associated with the role of world modeling is a store of information—data and knowledge—that is a part fo an IB distributed across the various levels. The knowledge may be in the form of rules [23]. Examples of accessible information are: state variables, evaluation functions, models, and programs. Albus [2] includes specifications of state information shared with other WM modules in the task frames mentioned above.

Thus, we can associate conceptual models of equipment-elements with their purpose in each module at each level in the architectural model.

For levels outside Level 1 in the nesting order, execution consists of passing commands to the subsystems, i.e., entities, at the next inner nesting, i.e., next lower level, and monitoring their results or states and integrating such information received from these entities. There is regularity in the conceptual structure of each level in the nesting. The outermost level is controlled by an authorized human, who may set up the system to take further commands from a remote computer. A subsystem at any level may take commands only from the level immediately outside, i.e., above it. Commands may be plans or programs stored at the time of setting up a particular subsystem. Additional plans or parameter values or other information may be put in place at any later time prior to the start of an execution cycle at that level. Figure 5 shows the overall hierarchical control scheme.

4.4 Task decomposition

Associated with the *jurisdictional* entities at each level are tasks or functions or responsibilities of those entities. Generally the functions and behavior, by design intent, fit into a corresponding constituent hierarchy (see Figure 4). There are deviations from this correspondence, which we will discuss further in Section 5.
For each level, let the task-specification be a sequence of (object, task, latest-tolerable-delivery-time) triples. Under that premise, in Figure 2, we show the typical tasks at each level and the objects on which the tasks are performed. The motion-related tasks are examples of operations that would be composed from primitives modeled in Section 5 and synthesized in Section 6.
Physical action in the external world is performed only at Level 1. Actions at higher levels are successive compositions of the physical actions at Level 1. Thus, the performers are only conceptual entities. Most of these semantics are established in the application-domain and are documented in literature [2, 48]. We have found some additional abstractions to be useful—we will discuss these abstractions in Section 5.

4.5 Spatial span and resolution

The spatial span and resolution at the innermost nested level have the smallest values. Spatial span increases and resolution gets coarser at each successive outer level, as shown in Table 3.
In our case, spatial span corresponds to the constituents at each level shown in Figure 4

Table 3: Resolution-relevance in the system

Level:Role	Spatial span	Temporal span
6:Cell	Several workstations	3,600,000T
5:Workstation	Intra-workstation	180,000T
4:Equipment	Multiple	20,000T
-	basic-machines	-
3:Elemental-move	Multiple axis-groups	2,000T
2:Primitive	Individual	200T
-	axis-group	-
1:Servo	Individual	20T
-	joint or axis	-

Note: T is the temporal resolution.

4.6 Temporal span and resolution

Table 3 focuses on functions related to servo-motion within a manufacturing cell. T represents the smallest temporal resolution in the whole system, i.e., the shortest state-change interval. The temporal resolution may be the shortest sampling interval in the case of a sensor or the fastest update interval in the case of a servo-actuator. Similarly, the temporal span may be the planning horizon in the case of commands to be generated and issued or it may be the length of historical traces or data-streams. The temporal span entries in Table 3, given as multiples of T, indicate orders of magnitude, assuming a just-in-time manufacturing system. For example, if T were 1 milli-second, the temporal span of a cell would be in the order of an hour.
We postulate for the case of the chosen domain, Section 3, that in a system with n levels, for $1 < k < n + 1$

1. An external-world task mapped in Level k will have a time constant coarser or longer than the longest of the time constants of its subtasks assigned to Level $k - 1$.
2. There is no need for temporal resolution at Level k to be finer than temporal resolution at Level $k - 1$.
3. There is contention for resources requiring deadline-based ordering of tasks.
4. Increase in computational capacity increases cost.
5. Computational cost can be reduced if real-time deadlines can be relaxed.
6. There is economic advantage in allowing computational tasks to be serviced in the longest time tolerable.

Figure 5 shows the overall information architecture. Each level up to Level m performs an execution cycle within a guaranteed time limit, i.e., these levels are *real-time* subsystems.
Sensory information is processed, assimilated or integrated or fused and used in the context set up for each subsystem. Furthermore, other information pertinent to that context—*temporal* and *jurisdictional*—is made available to that subsystem.
We observe that the temporal and spatial resolution of the equipment models used in each level need to be commensurate with the resolution of the tasks at that level. Also, world models at one of these levels may be valid only for the temporal and spatial span and resolution of that level. In this manner the architecture sets up *jurisdictional contexts* and *temporal contexts* for each subsystem. These contexts help address the questions raised in Section 4.1. These contexts also have significant implications in reducing computational costs for a given level of complexity in the application.

4.7 Applicability to a simple cell or machine.

In simpler compositions of a cell, the functions at certain levels may be thought of as "pass throughs". At the time of original creation, the cost of inserting a "pass-through" level in a simple system is relatively insignificant. However, at a later stage in the life cycle of the system, when functions at that level are added, the cost of retrofit is much lower than in a system that lacked such provisions in the original design.
We will show in Section 5 examples of applying the architectural model to simpler configurations, mechanisms and elements.

4.8 Hierarchy of sensory feedback

We shall illustrate the value of an hierarchical architecture through the following example. Let us consider the case of an information-request from some client-process on sensor signals. The request may concern any of the following:

- Instantaneous value

Table 2: Task hierarchy in a machining cell

LEVEL	PERFORMER	PERFORMED ON	TASK
6	Cell	Workpiece	Make
5	Workstation	Workpiece-setup	Macro-operations (Machining sequence in given setup)
4	Equipment	Workpiece-setup Fixtures Palettes Trays Tool-assembly	Elemental-move-sequences
3	Machine	Peripheral-mechanism Auxiliary-equipment Workpiece-setup Tool-assembly Fixtures Palettes Trays	Elemental-moves: move, rapid-move ...; approach, slow-approach ... Machining-tasks: mill, bore, turn, drill ... Measure Insert Holding-tasks: place, put ...; touch, clamp, grasp ...; unclamp, release ... ...
2	Axis-group	Workpiece-feature Tool-tip Other-object-features	 Go-along-path Move-to-point
1	Axis or joint	Actuator	Move: translate, rotate Note: Control position, velocity, acceleration, deceleration, force, etc. in specified profile over time. Turn-On, Turn-Off ...

- Values over current period
- Recent history
- Longer history

The example illustrates the consideration of several factors in the representation of the capabilities and limitations of manufacturing equipment. Let us take the case of axis-position feedback signals. Typically in higher-performance servo-controlled manufacturing automation, axis-position is sampled at sub-millisecond intervals (the PWM power-amplifiers can accept signals at 0.00025-second intervals). In contrast, 0.125-second update-interval is sufficient for the display screen, i.e., 1 in every 500 readings of the position-feedback. However, for human reading the interval does not have to be precise. Therefore, it could vary from 0.125 to 0.016 second in practice. If the reading sent to the display screen were to be sent to the client-process (as is the case in current practice), the sequence of readings would not be very useful for mathematical/statistical processing by the client, because of the widely varying unknown time-interval between readings.

4.8.1 Extracting servo-model parameters

Computation of parameters of the servo-subsystem model (see Figure 11 is a useful client process. Intuitively, if there is value in a command-interval of 0.00025 second (per example under consideration), the parameter-calculating process should get readings at intervals repeatable within 0.00025 second at worst. Thus, 4,000-8,000 samples per second would have to be recorded. The cost of processing, communication and storage resources has prohibited usage of position-time history data. Since this signal history contains valuable information about the real behavior of the servo-subsystem, it would be desirable to find an economic way of capturing and using this history.

4.8.2 Data reduction approaches

Several approaches are available for processing the signal close to the point of acquisition:

1. Lossless data compression and then forwarding the compressed data to some client process operating under less stringent response-time requirements in some other resource. This may be viewed as an application-independent compression technique within the domain of 8-KHz signals.
2. Extraction of useful information from the signal history close to the point of acquisition and then throwing away the raw data, e.g., a curve-fitting process based on a reference model. This may be viewed as an application-dependent compression

technique. A priori models of the servo-subsystem in current usage only capture the dominant characteristics of the system. These models may be sufficient to control a "normal" system, but not sufficient to detect and diagnose abnormalities. Such commonly occurring phenomena as backlash and stiction are not captured. Effects of common malfunctions, e.g., errors in the position-feedback device, worn gear-teeth in the drivetrain, are also not captured. It would be useful to model some of the well-known abnormalities or malfunctions to the extent necessary to detect or identify their presence. Thus, modeling the signal is closely related to modeling the servo-subsystem.

3. In-between techniques may also be worth exploring, where an a priori model of the servo-subsystem is not used as a reference, but the signal is classified by some generic characteristics of the servo-signal subsystem. An appropriate signal-compression scheme is accordingly chosen, e.g., processing in the time domain, frequency domain, time frequency domain, or frequency-time domain.

Any of these approaches is feasible with current and emerging DSP technology. DSPs are becoming common for processing the servo-control algorithm. The same processor could implement the chosen compression or information-extraction algorithm and pass on the compressed information to the client process(or). If the servo-control algorithm is being implemented at a 0.00025-second interval (4 KHz frequency), its "parent" (next higher level, say Level 2) process might be implemented at a 0.002-second interval (500 Hz frequency). Even this frequency is so high that it becomes attractive to apply some model at this process-level to further extract useful information, apply the useful information at its level and send to the next higher level (say Level 3) information reduced further, say at 0.016-second intervals (60 Hz frequency). The model at the Level-2-process should be good enough to detect a clear local failure and *broadcast* it to processes within the same axis-group (kinematic subsystem). Resulting actions could be to correct velocity of each axis in the group so as to maintain the coordinated space-time trajectory. It may suffice to store only the last 48 0.002-second-interval readings (three 0.016-second periods) at the Level-2-process. Older readings could be discarded under the premise that all information, useful beyond the time-span of Level 3, is retained in the 0.016-second-interval signal-model and the last forty-eight 0.002- second interval readings. The model at the Level-3-process should be good enough to compute a trend (change in behavior or impending failure); the trend should be stable enough for the next few Level-3 cycles (next few 0.016- second intervals). The change or new information would be disseminated and used in machine-wide peer processes. Resulting actions might be to interrupt the machine cycle or to alter (adapt) the material-processing rate.

The Level-3 process would reduce the feedback information and send it to the Level-4 process at 0.125-second interval (which brings us to the frequency of updating the human display). The timing of the computation could vary up to 0.016 seconds. This scheme would make it possible to send "good enough" feedback-data to a client process(or), without the excessive burden of sending all raw data or sampled raw data with unacceptably-uneven spacing.

Even a rate of eight sets of data per second becomes excessive for long-term storage (86,400 seconds per day!). Thus it is desirable to continue the process of successive reduction outlined above. This scheme would require a model of the system good enough for the reduction process at each successive level.

4.8.3 Equipment-degradation models

Equipment characteristics changing in a "non-convergent" manner (e.g., some parameter drifting away from the mean value or increasing variation in some parameter) over time periods in the order of months could be caused by degradation (e.g., wear causing reduction in stiffness of drive-leadscrew bearings or backlash in the drivetrain). Thus, equipment models capturing such changes in behavior should be related to equipment-degradation models (see Figures 10 and 11.

These models would be useful:

1. in planning preventive maintenance, e.g., when to schedule the next calibration or other detailed inspection;
2. in planning repair, e.g., when to order replacement parts, when to take equipment out of service for repair;
3. as feedback for design-improvements and retrofits; and
4. as a basis for future system specifications and design.

4.8.4 Data for quality audit

In certain situations, e.g., parts of high value or high consequential costs, manufacturers find it valuable to maintain an audit trail specific to each part manufactured [44]. Audit trail data become very voluminous. Therefore, even for an archival level in the information architecture, there is value in using information models that allow data reduction or compression. In current practice, the major items of data are:

1. Record of inspection, e.g., measurements on a coordinate measurement machine.
2. Part program.
3. Values modified during the execution of the part program, e.g., offsets, feedrates, spindle speeds.

Besides being voluminous, these pieces of data do not easily yield insight into the causes of a quality problem that might be discovered later on.

Also, off-line inspection is very costly. Thus, the trend is to assure quality on line, maintain knowledge of the *process-capability* of the system and use this knowledge to determine how well quality can be obtained [44]. Thus, it is expected that in the future the audit trail will also include process-capability models of the manufacturing system, especially the equipment. Nau [34] has shown off-line usefulness—with on-line-modeling, the effect of equipment on process-capability could be traced and compensated much more productively.

Storage of process-capability history, while valuable, could also be excessive in storage and retrieval costs, since each part cycle could be as short as a few seconds—commonly around 30 seconds in major automotive parts (over a million records a year!). Thus it

would be valuable to devise a modeling scheme that would reduce this burden.

Table 4: Levels of Data Reduction

Process level level	Incoming data interval	Variation in interval	Sets of data
1	0.0002 s	-	8
2	0.002 s	0.0002	8
3	0.016 s	0.002	8
4	0.128 s	0.016	8
5	1.0 s	0.128	8
6	8 s	1	8
7	64 s	8	8
8	512 s	64	8
9	4096 s	512	8
10	2exp15 s	(hour)	8
11	(days)	(hours)	8
12	(month)	(days)	8
13	year	(month)	8

Note: s denotes second(s)

This example shows 13 levels of data reduction. [11] It has eight sets of history data per level. Only 104 sets of data have to be stored over a year of operation. The example was given to indicate that a progressive data reduction scheme would allow manageable amount of data to model the real capability of the machine.

4.8.5 Issues in data reduction

Some issues, not considered in the example above, are mentioned below. These considerations will require more intermediate storage and more involved intermediate states, not modeled in this study.

1. How to preserve the low-frequency content in the data captured at the lower levels?
2. How to represent a sudden event, e.g., effect of a microfracture instead of smooth continuous wear on the change in behavior?
3. How to distinguish between "noise" and "real effect"? (Would information get inadvertently filtered out as noise and therefore delay detection of a change in behavior or lead to erroneous model of behavior?)
4. How to devise a model-validation scheme such that when significant abnormality or uncertainty is detected from running data or at some time when the machine is not engaged in productive activity, a controlled, properly designed test sequence is run to determine the real behavior? How to merge or fit the results of such tests into information derived from long-term running data? Which data-set is more real? Controlled tests seldom represent behavior under real production conditions—could the running data be considered more real and true? A representation scheme should capture these considerations.

[11] Process levels 5 and higher may be performed at architecture level 6 shown in Table 3.

5 Modeling elements of manufacturing equipment

Our primary purpose of modeling is to represent real-world basic machines in a way that captures their characteristics essential to the applications bounded in Section 3. Secondly, we want to represent this knowledge in a way that maximizes reusability, e.g., by abstracting commonalities across different basic machines.

We have focused on characteristics related to monitoring and continuous control of servo-controlled motion, with the intent to extend the model for monitoring and diagnostics.

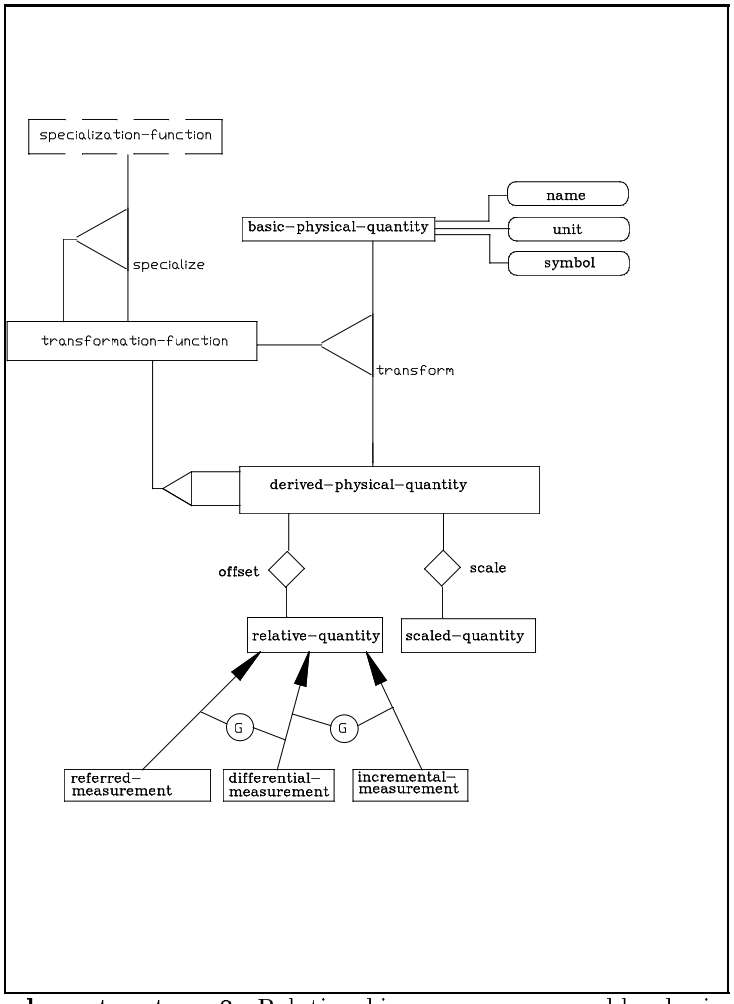
Hayes-Roth [20, p. 143–144] recommends that the conceptualization stage should focus on identifying the key concepts and their relations and not commit to formal representations prematurely—different representation frameworks may be more suitable for different concepts. Therefore, our primary focus is on modeling elements of a basic machine in the language customary to the domain and the related engineering subdomains.

Loucopoulos [28] classifies conceptual modeling orientation along one of three dimensions: data modeling, process modeling, and event modeling. The domain of manufacturing equipment requires a mix of all three dimensions. However, since this study is at the beginning stage of modeling this domain [12], it is oriented toward data modeling—the dimension least dependent on applications. Entity-Relationship (ER) Diagrams [50] are used to represent entity classes and their inter-relationships. These diagrams have been extended with other modeling notation to suit the domain. Relationships shown are mostly aggregation or inheritance. In some cases it was convenient to identify the entities in a tabular form. In most cases, attributes and constraints are not shown in the ER Diagrams for clarity. A separate textual format ("template") is used to show attributes, key operations, and basic state models of the entities. In certain cases (where the primary relationship was inheritance) it was convenient to represent functions as objects in an ER diagram, although the usage was unconventional. In some cases, features of real-world objects are further defined as mathematical formulae. In the event modeling dimension, Section 6.4 and Class-structure 23 show our approach to modeling generic states of manufacturing equipment. Thus, structure, behavior and dynamics of the manufacturing equipment are decoupled from each other, as recommended by Rumbaugh [42], although the three types of information are shown in the same textual "template" or skeletal frame. For ease of human readability, object class features are not named with concatenated or cryptic words (as done for computer-readability); instead, descriptive text is used.

We also reviewed literature on conceptual primitives for machine tools. The NGC Schema [48] is the closest work, in which a number of "primitive data types" are proposed as part of a standard for the next generation of controllers. Some examples are given in Appendix A. On the one hand many of these "primitive data types" [13] are not the conceptual primitives from which manufacturing machines can be composed. On the other hand, many conceptual primitives are missing. It seems that the domain primitives became obscure in the programmer's view of the problem.

5.1 Physical quantities and phenomena

Attributes of machine elements are described in terms of basic physical phenomena and quantities, as shown in Class-structure 8 and Appendix G. The term *quantity* is used here to mean: "a measurable attribute of phenomena or matter" [4].



class-structure 8: Relationships among measurable physical phenomena

The first column in Table 6 is the conceptual entity. The second and third columns are attributes. Using a standard basis of physical quantities will ease reuse, extension and integration of software. The vocabulary is based on the "International System of Units [4]" and STEP.

Definitions of quantities and units The physical quantities are defined in the SI standard [4]. For derived units in

[12] No adequate starting point was found in technical literature.

[13] For example, database-ms-type, which stands for database management system type

Table 5: Derived quantities used in modeling material properties
*

Physical quantity	Measurement-unit	Symbol
elastic-modulus	$newton/meter^2$	N/m^2
bulk-modulus	$newton/meter^2$	N/m^2
tensile-strength	$newton/meter^2$	N/m^2
damping-coefficient	$kilogram/meter/second$	$kg/m \cdot s^{-1}$
damping-ratio	none	none
stiffness	newtons/meter	N/m
...	...	...

Table 8 that are already expressed in terms of basic units, the entries in their symbol column also serve as the formulae defining these units in terms of the basic units. The dot represents the multiplication symbol. For derived units with different names, Table 7 provides formulae or mathematical definitions of these derived units in terms of the more basic units, as defined in [4]. The same (table) structure serves to define physical quantities that describe material properties, as shown in Table 5, where material property is a derived physical quantity. The same table structure suits the expression of other relationships between physical phenomena, e.g., impedance, by extending the interpretation of the column for formulae, represented as the transformation function in Class-structure 8. In more complex cases, different simplifications of the transformation function may be selected based on the conditions of the case. This is a form of specialization.

Other physical quantities Class-structure 8 provides the framework for defining other physical quantities (that may be needed in modeling machines) in terms of previously defined quantities. These definitions may require more complicated mathematical expressions and conditions for selecting appropriate expressions, e.g. surface roughness. Certain quantities, e.g., hardness may have to be defined in tables of numbers or curves relating them to previously defined quantities.

Unit conversions and scaling factors. Some other derived units, defined in the SI standard [4] or in usage in the trade, can be related to the units defined above by conversion factors, i.e. scaling constants (see Class-structure 8). Such scaling factors may also be defined and used to allow for different orders of magnitude. The structure of Table 7 accommodates these definitions.

Scale shifts and intervals. Industrial measurements are mostly not on an absolute scale, traceable to a standard. Industry distinguishes measurements with such terms as *relative*, *incremental*, and *differential*. Class-structure 8 shows our unifying approach of treating all industrial measurements as *relative* to some reference. An *incremental* measurement is relative to the previous measurement. A *differential* measurement is relative to some other changing or fluctuating commensurate quantity. Some examples are given below:

Duration. An interval of time during which something exists or persists.

Extent. An interval of length over which something exists or extends.

Linear-position. A set of intervals of length relative to a reference coordinate frame.

Orientation. A set of plane-angles relative to a reference coordinate frame.

5.2 Constituents of a basic machine

Class-structure 7 shows constituent elements of the the basic machine. A substructure may be given a name in the application, e.g., *t*ool-tip-motion unit as named in Class-structure 25.

5.3 Abstraction of a joint and its components

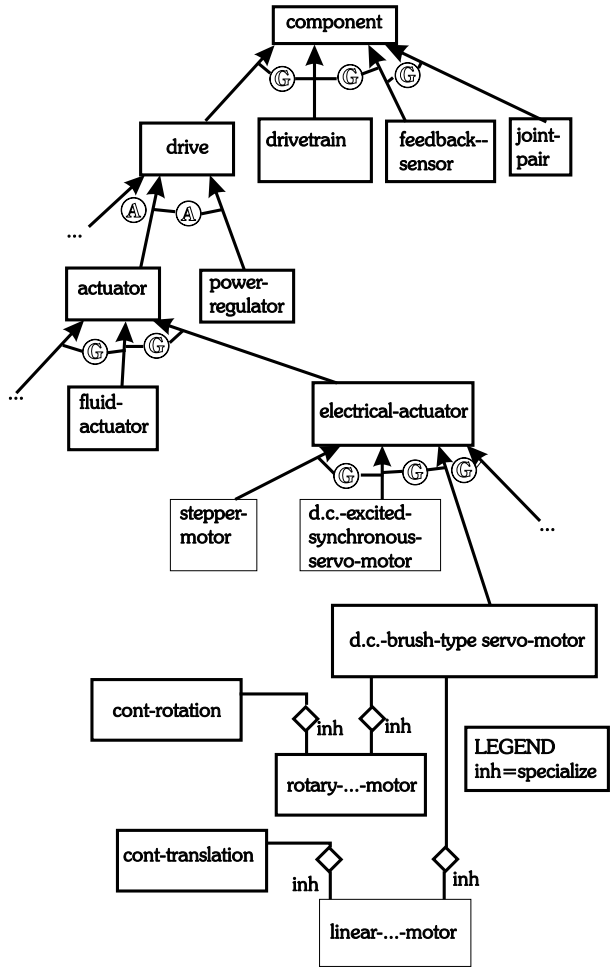
We abstract a single *j*oint or axis of motion as a composition of *j*oint-pair, *d*rivetrain, *d*rive and *f*eedback-sensor. Our approach to modeling these components is shown in this section. The synthesis of these models into joints is discussed in Section 6.
Let *j*oint-component be a generic or parameterized class of these four classes of joint-components, as modeled in Class-structures 9, 10 and 11. Many of the attributes are parameterized, and their appropriate values must be filled in, to obtain the classes *j*oint-pair, *d*rivetrain, *d*rive and *f*eedback-sensor. The generality of the approach begins with the abstractions of input and output, building on the concepts described in Subsection 5.1. This abstraction is a significant contribution toward unification of monitoring, diagnostics and control in various time-span horizons. These models capture the key features of such components used in most applications in industry today.
In Class-structure 11 the dynamics model of the component, treated as a linear multivariable continuous-time system is described [46] in terms of the following equations,

$$\vec{x}(k+1) = A\vec{x}(k) + B\vec{u}(k) \quad (1)$$

$$\vec{y}(k) = C\vec{x}(k) + \vec{v}(k) \quad (2)$$

where:
$\vec{u}$ is the input vector of dimension m,
$\vec{y}$ is the output vector of dimension p,
$\vec{v}$ is the noise vector,
$\vec{x}$ is the state vector.
n is the order of the system,
${}_nA^n$, ${}_nB^m$, ${}_pC^n$ are constant matrices (dynamics-model-parameters) that identify the dynamics of the system.

As an example, descendants of the generic **joint-component** are modeled in Class-structures 12, 13, and 14. The symbols, $u, y, v, p, m, n, A, B, C$, in these class-structures, refer to Equations 1–2.
Many of these attributes describe involved characteristics. At the generic-component level, a characteristic may be a complicated equation or a pair-sequence (or equivalent curve). For example, consider the dynamics-model of the generic component, in the next section.



class-structure 9: Joint-component hierarchy: Example for servomotors

Configuration-related attributes:
- rated-life :*time*
- mass :*mass*
- output, $\vec{y}$ (dimension p) :*physical-quantity*
- input, $\vec{u}$ (dimension m) :*physical-quantity*

Calibration-related attributes:
- **dynamics-model**
  - $\vec{x}(k+1) = A\vec{x}(k) + B\vec{u}(k)$
  - $\vec{y}(k) = C\vec{x}(k) + \vec{v}(k)$
  - calibration-procedure :*string*
  - calibrated-deadband :*real*
- internal energy losses:
  - f(load, speed, temperature)
- heat dissipation model: ...

Monitoring-related attributes:
- life-consumption or degradation :*time*
  - f(load, speed, temperature, duration)
- operating range:
  - output range:
    - upper limit, $\vec{y}_{max}$
    - lower limit, $\vec{y}_{min}$
  - input range:
    - upper limit, $\vec{u}_{max}$
    - lower limit, $\vec{u}_{min}$
  - vibration limit :*acceleration*
  - shock limit :*acceleration*
  - jerk limit :*jerk*
  - relative-humidity limit :*real*
  - contamination :*complex attribute*
  - operating temperature :*temperature*
    - upper limit
    - lower limit
    - ambient
  - dynamics-(model-)parameters:
    - upper alarm-limit
    - lower alarm-limit
    - mean
  - life(-consumption-limit) :*time*

Where the typical attribute is a complex object, e.g., contamination is the set of tuples:
<contaminant, measurement-unit, limit>

class-structure 10: Generic joint-component - Part 1

5.4 Modeling dynamic characteristics

Physical dynamic behavior is one of the essential characteristics of machine elements involved in the control of a manufacturing machine. It is shown as a complex attribute in Class-structure 10. It provides for multiple inputs m, multiple outputs p and a specifiable order n of the dynamics model through matrices A, B, C. However, this complex attribute may be specialized to some simpler form for the various subclasses of components. For example, as indicated in Class-structure 14, a precision d.c. servomotor operating within its specified range at a steady state may have a constant ratio K_e [14], of output angular-velocity to input electromotive force (emf) across armature.

[14] commonly known as velocity constant or gain

5.5 Modeling a component for different architectural levels

Attributes are grouped by their central purpose, for ease of comprehension. The model provides for abstraction of monitored information into the levels in the control architecture shown in Class-structure 5. Attributes related to configuration are primarily for use in levels 3–6. Attributes related to calibration may be used in levels 2–6. Certain information available in these structures might

Operations:
transform-energy
:[*transmit-energy, transduce-energy*]
accept discrete events
:[*enable, disable, ...*]
generate discrete events
:[fault, ok,...]
accept continuous-time signals,
$\vec{x}_{l1}(k)|x_{l1} \in \{physical\text{-}quantity\}$
generate continuous-time signals,
$\vec{x}_{l2}(k)|x_{l2} \in \{physical\text{-}quantity\}$
compute state estimate for level i —$1 \leq i \leq n$
compute dynamics-parameter-estimates

State- and parameter-estimates:
current state vector at level 1 $\vec{x}(k)$
measurement noise vector at level 1 $\vec{v}(k)$
dynamics-parameter-estimates
$(\hat{A}(k), \hat{B}(k), \hat{C}(k)$ *in Equations 1–2)*
deadband :real
state-estimate at level 1, $\hat{x}(k)$
state-estimate at level $n \in \{2, \ldots, 6\}$*:*
functions of state-estimate at level $n-1$*, e.g.,*
abnormality indices,
life-consumption,
violations of operating range.
Where:
T *is the sampling interval at level 1*
k *is the kth sample at level 1 (when time=t)*

class-structure 11: Generic joint-component - Part 2

be used at a lower level of Class-structure 5 in a different form, for meeting timing constraints.

The proposed model provides for specializations of the attributes within the same system, depending on the context of usage. Even though the dynamics-model may be complicated over the whole operating range, it can often be simplified over the narrow range in which a servo-loop is operating in any particular context. The nested hierarchical architecture discussed in Section 4 sets up these contexts. A higher level in the hierarchy may give a different simplified model at different times to different lower levels nested within it.

The architecture proposed in Section 4 provides for categorizing goals [29] which help determine which models and which approximations and simplifications are appropriate when in pursuit of a goal of a particular category. This modeling approach, including the nested hierarchical architecture discussed in Section 4, provides scaleability to cover different cost-performance tradeoff points, versatility to accommodate different kinds of devices, and extensibility beyond the original bounds of the domain.

5.6 Drive and drivetrain

Our initial focus is on programmable servo-drives for an industry that overwhelmingly uses direct-current (electrical) servomotors with pulse-width-modulated power-amplifiers coupled with ballscrew-nut pairs.

Drive. However, as shown in Class-structure 9, the drives are modelled generically to support extension to other types of drives. Class-structure 16 shows the model of a power regulator or amplifier. Although in Section 3.8, the scope of equipment was limited to linear systems, Class-structure 16 models two major sources of nonlinearity at the operating limits of the power amplifier. Saturation is one type of nonlinearity, when the power amplifier current limits or voltage limits are reached. Deadband is another type of nonlinearity—when the input is close to zero, no output is observed until some threshold value of input is crossed. The model also provides for monitoring when deadband [15] exceeds some preestablished limit.

Drivetrain. The most commonly used drivetrain component in servo-controlled machining equipment is a preloaded ballscrew-nut pair. However, as shown in Class-structure 18, we have modeled a generic drivetrain component to facilitate generation of other types of drivetrain components.

Configuration-related attributes:
output:
dimension, p = 2
y_1 :[*linear-velocity, angular-velocity*]
y_2 :*force, torque, current, flow*]
input:
dimension, m = 1
u_1 :[*emf, pressure*]
Operations:
transduce-energy as defined in input and output
= move :[*rotate, translate*]
Where:
*c*ogging is a "capacitance effect" over certain values of position or orientation;
*r*ipple means harmonics of the output.

class-structure 12: Actuator

Configuration-related attributes:
input, $u_1 = E_{in}$:*emf*
armature-resistance= R_a :*electric-resistance*
armature-inductance, L_a :*inductance*
Operations:
move :[*rotate, translate*]

class-structure 13: Electrical-actuator

[15] along with other monitored states and parameters

Configuration attributes:
output, $y_1 = \omega$:*angular-velocity*
output, $y_2 = I$:*current*
moment of inertia, J_m :*rotational moment of inertia*
viscous damping coefficient, B_m :*damping-coefficient*
torque constant, K_t
voltage constant, K_e
Monitoring-related attributes:
order of system, n = 1
Derived attributes:
Electrical time constant, $T_a = L_a/R_a$:*time*
Mechanical time constant, $T_o = R_a J_m K_e/K_t$:*time*
Damping ratio, $\zeta = 1/2\sqrt{T_o/T_a}$:*real*
Natural frequency, $\omega_o = \frac{1}{\sqrt{T_o T_a}}$:*frequency*
State space model:
$\omega = \frac{IK_t}{J_m s + B_m}$:*angular-velocity*
$I = \frac{E - K_e\omega}{L_a s + R_a}$:*current*
Derived discrete state space model:
$\omega(k+1) = (1 - \frac{B_m T}{J_m})\omega(k) + \frac{K_t T}{J_m} I(k)$
$I(k+1) = (1 - \frac{R_a T}{L_a})I(k) + \frac{TE}{L_a} - \frac{K_e T}{L_a}\omega(k)$

Operations:
rotate :*angular-velocity*
generate discrete events :[*overheated*]

Where:
$K_t I$ = output torque at motor shaft.

class-structure 14: Rotary d.c. servo-motor

dc-brush-type-motor
Monitoring-related-attributes
brush-life :*time*
brush-life-consumption=f(brush-life-factors, duration) :*time*
brush-life-factors:
commutator-surface-speed=f(output velocity) :*linear-velocity*
commutator-bar-to-bar-voltage=f(input emf) :*emf*
brush-current-density=f(current) :*current-density*
brush-commutator-interface power=f(current, emf) :*power*
commutation-limit :*current*
terminal-resistance :*electric-resistance*

class-structure 15: DC brush-type motor

5.7 Joint pair

Class-structures 19—20 show a generic model for a rotational or a prismatic joint pair. Class-structure 21 shows an example of specialization of the generic joint pair model to a rotational joint pair model. The model captures information about dynamics, static load capacity, kinematics, errors of motion, and operating limits that are specific to joint pairs. Provision is also made for modeling errors of motion as a function of temperature profile across the extent of the joint pair. Many of these characteristics are not tracked in current practice. However, research results have established the value of such information in precision manufacturing. Thus the

Configuration-related attributes
output:
dimension, p = 2
$y_1 = E$:*emf*
$y_2 = I$:*current*
input:
dimension, m = 1
v_i :*emf*
maximum input power :*power*
input power at maximum output :*power*
form factor = $f(u_1)$:*real* < 1
input to output propagation delay :*time*
slew rate
internal losses = $f(E, I)$
limits on voltage :*emf*
peak upper voltage limit, V_{vlm}
peak lower voltage limit, $-V_{vlm}$
peak upper current limit, V_{ilm}
peak lower current limit, $-V_{ilm}$
RMS upper limit, V_{rms}
RMS lower limit, $-V_{rms}$

class-structure 16: Power amplifier - Part 1

model provides for future extension to utilize these research results and emerging technologies. A noteworthy contribution of this study is the association of this information with a joint pair. Typically, such information has been a system level only—often because it is acquired by calibrating an installed operational system. Associating the information with its causal sources provides more genericity and flexibility in the deployment of the information in other contexts. Since such decomposition and allocation may not always be possible, the scope has been limited (see Section 3.8.1) to systems where such correspondence exists.
In Class-structure 20, the nominal location of the moving link relative to the fixed link (reference frame B) of the joint pair is given by ${}^B T_M$. For a prismatic joint pair (translational slide) with no errors of motion,

$$ {}^B T_M = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where:
X and Y are constant offsets of the origin of the slide coordinate system with respect to the fixed-link frame, and d is the joint variable identified in Class-structure 20.
If errors of motion were included [16],

$$ {}^B T_M = \begin{bmatrix} 1 & -\varepsilon_z & \varepsilon_y & X + \delta_x \\ \varepsilon_z & 1 & -\varepsilon_x & Y + \delta_y \\ -\varepsilon_y & \varepsilon_x & 1 & d + \delta_d \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where:
ε_x is rotational error about x-axis, *yaw*,
ε_y is rotational error about y-axis, *pitch*,

Calibration-related attributes:
calibration-procedure *:string*
calibrated-deadband *:real*
calibration model (form as in state space model)

State space model:

$E = E_v | -V_{vlm} \leq E_v \leq V_{vlm}$
$E = -V_{vlm} | E_v < -V_{vlm}$
$E = V_{vlm} | E_v > V_{vlm}$
$E_v = \frac{K_{vol}(s+\omega_{kv})}{s} E_i$
$E_i = \tilde{E}_i | -V_{ilm} \leq \tilde{E}_i \leq V_{ilm}$
$E_i = -V_{ilm} | \tilde{E}_i < -V_{ilm}$
$E_i = V_{ilm} | \tilde{E}_i > V_{ilm}$
$\tilde{E}_i = \frac{K_i}{T_{ki}s+1} v_i$

Where:
K_{vol} is amplification of voltage-amplifier stage;
K_i is amplification of current-amplifier stage;
ω_{kv} is electrical supply frequency;
T_{ki} is time constant of current-amplifier stage;

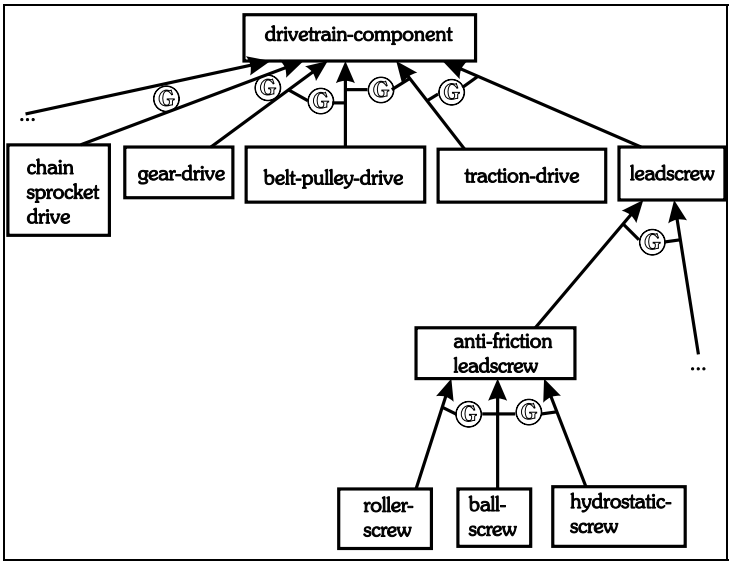
Derived discrete state space model:

$E_v(k+1) =$
$E_v(k) + K_{vol} v_i(k+1) - K_{vol} v_i(k)(1 - T\omega_{kv})$
$\tilde{E}_i(k+1) = (1 - \frac{T}{T_{Ki}}) E_i(k) + \frac{K_i T}{T_{Ki}} v_i(k)$
$E(k+1) = V_{vlm} | E_v(k+1) > V_{vlm}$
$E(k+1) = -V_{vlm} | E_v(k+1) < -V_{vlm}$
$E(k+1) = E_v(k+1) | -V_{vlm} \leq E_v(k+1) \leq V_{vlm}$
$E_i(k+1) = V_{ilm} | \tilde{E}_i(k+1) > V_{ilm}$
$E_i(k+1) = -V_{ilm} | \tilde{E}_i(k+1) < -V_{ilm}$
$E_i(k+1) = \tilde{E}_i(k+1) | -V_{ilm} \leq \tilde{E}_i(k+1) \leq V_{ilm}$

Operations:
transduce-energy as defined in input and output
accept continuous-time signals:
:[*current*
linear-velocity or angular velocity
position or orientation]
accept discrete events :[*commutation-directions*]

Where up to 3 out of 6 commutation-directions may be signalled at once

class-structure 17: Power amplifier - Part 2



class-structure 18: Drivetrain-component hierarchy

input-output
output:
dimension, p = 1
y_1 :[*linear-velocity, angular-velocity*]
input:
dimension, m = 1
u_1 *:[linear-velocity, angular-velocity]*

service
serviceable-constituents
:[limit-switches[enum],
semi-durable-components[enum]

Travel limits
location of home, d_{home} *or* θ_{home} *:[length, plane-angle]*
location of overtravel limits :[length, plane-angle]
forward limit, d_{fwdlim} *or* θ_{fwdlim}
reverse limit, d_{revlim} *or* θ_{revlim}
location of travel stops :[length, plane-angle]
forward stop, $d_{fwdstop}$ *or* $\theta_{fwdstop}$
reverse stop, $d_{revstop}$ *or* $\theta_{revstop}$

class-structure 19: Joint-pair - Part 1

ε_z is rotational error about z-axis, *roll*,
δ_x is translational error about x-axis,
δ_y is translational error about y-axis,
δ_d is translational error about z-axis,
$\delta_x = \tilde{\delta}_x + \alpha_{xz} \cdot d$,
$\delta_y = \tilde{\delta}_y + \alpha_{yz} \cdot d$,
$\tilde{\delta}_x$ is straightness error of slide in direction of x-axis,
$\tilde{\delta}_y$ is straightness error of slide in direction of y-axis,
α_{xz} is angular error between axis of motion of slide and x-axis of fixed-link frame B,
α_{yz} is angular error between axis of motion of slide and y-axis of fixed-link frame B, and
all errors, particularly angular errors, are small.
This kinematic model of motion becomes the foundation upon which the (kinematic) motion of the work point in a complete machine can be modeled, as shown in Section 6.2.

kinematic constituents :[*base-link, moving-link*]
fixed-link frame, B :coordinate frame
moving-link frame, ${}^{B}T_{M}$ *:coordinate frame*
link offset, d :length
joint angle, θ *:angle*
joint-variable = d or θ *:[length, plane-angle]*
location of center of gravity = $f(d, B, {}^{B}T_{M})$ *:length*
location of centroid = $f(d, B, {}^{B}T_{M})$ *:length*

Calibration-related attributes:

calibration-procedure :string
calibrated kinematic model
$${}^{B}T_{Mc} = f(d, t_f, l_f, {}^{B}T_{M})$$

Monitoring-related attributes

fixed-link temperature profile, $t_f = f(l_f)$

Where:

link offset, joint angle, link-frame
are terms from the Denavit-Hartenberg notation;
fixed-link
supports the measurement reference,
provides the guidance for the moving-link, and
provides the actuation for the moving-link;
${}^{B}T_{M}$, ${}^{B}T_{Mc}$
are 4x4 homogeneous transformation matrices;
${}^{B}T_{Mc}$
includes errors of motion;
l_f
= distance from B
along ${}^{B}T_{M}\ \hat{Z}$*, moving-axis;*

Dynamics

calibrated deadband :real
friction-model
stiction
inertia :[mass, rotational moment of inertia]
stiffness, $\vec{K} = f(d, B, {}^{B}T_{M})$
maximum-static-load, $\vec{Load} =$
$f({}^{B}T_{M}, jointvariable, deflection)$
:[force, moment]

where:

$\vec{K}$
= vector of stiffness in 6 directions;
$\vec{Load}$
= vector of static-load limit in 6 directions;
Deflection *= allowable deformation under load*
in given direction :[length, plane-angle]

Operations

transmit-energy = guide-motion
:[guide-translation, guide-rotation]
generate discrete events:
forward overtravel limit
reverse overtravel limit
derive nominal kinematics matrix, ${}^{B}T_{M}$
derive calibrated kinematics matrix, ${}^{B}T_{Mc}$
derive dynamics properties

class-structure 20: Joint-pair - Part 2

Configuration-related attributes

output:
dimension, p = 1
y_1 :[*angular-velocity*]
input:
dimension, m = 1
u_1 *:[angular-velocity]*
inertia :rotational moment of inertia
location of home, θ_{home} *:plane-angle*
location of overtravel limits :plane-angle
forward limit, θ_{fwdlim}
reverse limit, θ_{revlim}
location of travel stops :plane-angle
forward stop, $\theta_{fwdstop}$
reverse stop, $\theta_{revstop}$

Operations

guide-rotation

class-structure 21: Rotational joint-pair

6 Synthesis of machine models

Ideally, previously modeled knowledge about machine elements, e.g., *joint-components* should be reusable to obtain models of their compositions, e.g., *joints*. However, there is very little published literature on this subject. Chaar [12] models workstation-level assemblages as a simple union of the constituents. However, in the case of a joint, we need a model of an assemblage of its actuator(s), sensor(s), drive-train, etc. as shown in Class-structure 7.Recall that the purpose of the model is to support monitoring, control and diagnostics. A simple union will not suffice for this purpose. Biggerstaff [7] observes, in the context of software components,

> The composing process imposes the most challenging requirements on the representation used to specify components...The notions of functional composition drawn from mathematical theory are largely inadequate. Mathematical compositions produce their results through straightforward combination of the local effects of individual components...Unfortunately, such composition systems are too limited. Human designers produce compositions of components having both global and local effects ...

Considering Biggerstaff's observation analogously in the domain of manufacturing equipment, we observe that the composition may contain new properties and conditions that do not exist in the individual components. For example, take a 3-axis machine where each axis individually has a certain range of travel. However, when the three axes are considered together, certain positions of one or two axes may prevent interference-free travel of the third axis. More such "global effects" arise as a result of other attachments to the machine (e.g., fixture, tool). Another example is a change in the temperature-rise-profile, due to changes in air-flow patterns. In view of the difficulty involved in composition, we have limited the scope of the modeling effort mainly to the synthesis of kinematics and dynamics of motion, for the purpose of monitoring and control and diagnostics within the planned operating parameters. In other words, these models are to be used under the premise that the process was correctly planned and designed. Any other constraints to be applied would be explicitly specified by the user. The synthesis models are given below in terms of mathematical equations, example rules of composition, and textual explanations.

6.1 Composing features of a joint from its constituents

Using Chaar's model [12], i.e., an assembly as a union of components, as a point of departure, we show in Class-structure 22 samples of different rules for composing different features of a joint:

- the simple aggregation relationship between constituents and assembly,
- connectivity relationship for the purpose of dynamics and kinematics,
- resultant operations,
- resultant states.

Configuration-related attributes
constituents = $\bigcap_j$ (components of the joint) [enumeration]
connectivity of components modeled per [22]
input = input of first component in connectivity chain
output = output of last component in connectivity chain
serviceable constituents = $\bigcup$ (serviceable constituents of components)
Kinematic and dynamics model
lower (intra-joint) kinematic model
derived from models of constituents
higher (extra-joint) kinematic model
= model of jointpair component
Travel limits
$\bigcup lim_{eq}(j)$
Dynamics model
derived from models of constituents
Calibration-related attributes
calibration procedure = $\bigcup CP(j)[CP_k]$
calibrated deadband :*real*
calibration model = $f([CM(j)]$
Operations
$\bigcup_j \bigcup_i (op(i,j) \bigcap (restriction(i,j)) \bigcup_k op(k)$
derive lower kinematic model from models of constituents
derive dynamics model from models of constituents
Equations 4–6
States
$f[S(j)] \bigcup S_k$
Where:
$op(i,j)$: operation i of component j, and
$restriction(i,j)$: restriction imposed on it at joint-level
$op(i,j)$ is "independent" of operations in other components
$\bigcup_k op(k)$: set of additional operations at joint-level
$[S(j)]$: set of states of components j
S_k: additional state-information at joint-level
$[CP(j)]$: set of calibration of procedures of components j
$[CP_k]$: set of additional calibration procedures at joint-level
$[CM(j)]$: set of calibration models of components j
$lim_{eq}(j)$: equivalent limits of component j

class-structure 22: Joint

6.2 Kinematic synthesis model

Our basic synthesis model is the ISO standard [22] for kinematic modeling, which is close to the well-known Denavit-Hartenberg (D-H) model [15]. This draft standard had its genesis in robotics, primarily oriented toward a single robotic device. Since manufacturing equipment could consist of multiple such devices working on a single workpiece or set of workpieces, we extend the ISO kinematic model, by building on the notion of substructures mentioned in the ISO model, to provide for the inclusion of kinematic models for fixtures, workpieces, and tooling. The D-H model is also extended to include kinematic errors of motion, as described by Donmez [16]. The composed property of interest is the motion of the work-point as a result of motions of the joints (or vice versa). Mathematical transformations for non-redundant compositions are available [16] as an established body of knowl-

edge. In general, if there are n links in an open kinematic chain, sequentially numbered starting from ground, i.e., some fixed frame of reference in space, U, and
the work-point is located on link n, and
${}^{i-1}T_i$ is a homogeneous transformation that gives the location of the ith link-frame relative to the $(i-1)$th link-frame (referring to Section 5.7 and Class-structure 19),
then, the location of the work-point is given by:

$$ {}^{U}T_n = {}^{U}T_1 \quad {}^{1}T_2 \quad {}^{2}T_3 \ldots {}^{n-1}T_n \tag{3} $$

The kinematics model also supports the model of dynamics and states, developed below. At the joint-level, these models are assimilated in Class-structure 22. At the level of a multi-joint machine, the kinematics-model and the calibration-model would follow the principles documented in [16]. The other features would be a union [12] of the features of all the joints in the machine, with certain conditions, restrictions, and values added for specialized cases or specific instances. A model of a machine composed in this manner can be applied to lathes, milling machines, drilling machines, machining centers, grinders, coordinate measurement machines, and robotic mechanisms. Thus this approach to abstraction is more generic than the approach used in [48].

6.3 Synthesis of dynamics model

Since we modeled the elements of manufacturing equipment with the property of linearity, their assemblages can be modeled as a network of linear system components. The composed property of interest is the dynamics of the resultant system, generally, the dynamics model between the input at an actuator and the output motion at some reference point on the joint. Mathematical transformations are available [3, 49] to derive the dynamics model for joints having compositions of the type shown in Class-structure 7. As an example, the different types of joint components identified in Section 5.3 are shown assembled in a schematic in Figure 6.
As shown in Class-structure 22, we derive most of the properties of a joint from the properties of its constituents. The (application-developer) user has to define the constituents and their connectivity, and would have the option of defining additional operations, state information, and constraints at the joint-level. Thus, by capturing a feature in the model of its source, we minimize the effort required to update the joint model, if and when a particular component is changed. As an example of deriving properties, consider the following well-known procedure.

Procedure for deriving dynamic properties. The transfer function of a joint, between its input R and output C, is derived from the following rules, applied to the connectivity network model of the joint, e.g., Figure 6.
S1: Derive the transfer function, G_k, for each forward leg in a loop k, of components $1 \ldots n$ connected in series, from the equation:

$$ G_k = \prod_{i=1}^{i=n} Gi \tag{4} $$

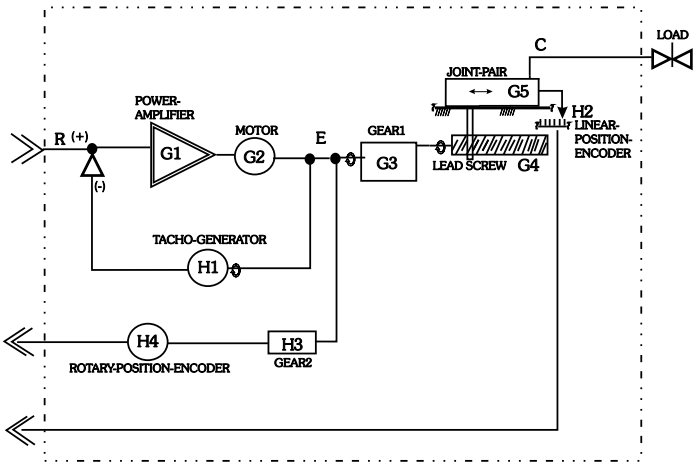


Figure 6: Dynamics model of a typical joint

where, Gi is the transfer function for component i
S2: Derive the tranfer function, H_k for each feedback leg in loop k of components $1 \ldots m$ connected in series, from the equation:

$$ H_k = \prod_{j=1}^{j=m} Hj \tag{5} $$

where Hj is the transfer function for component j.
S3: Derive the transfer function for each loop k from the equation:

$$ \frac{C_k}{R_k} = \frac{G_k}{1 + G_k * H_k} \tag{6} $$

where:
C_k is the output of loop k
R_k is the input of loop k
S4: Apply steps **S1**—**S4** to the network until it is reduced to a simple loop containing one forward leg and one feedback leg.

6.4 Modeling of states

First we review recent work in modeling states of manufacturing machines.
The NGC Schema [48] defines a number of state-related primitive data types and entity-types,
shown in Section B, as types of attributes for other models, e.g.,

- The system as a whole
- Components, e.g., robot-hand
- History-logs, plans, requests, goals, events.

We note several issues in this review [48]:

1. The distinction amongst the definitions of state, event, goal-outcome and value is unclear.
2. Different types of states address the same characteristic, i.e., there is unnecessary overlap or duplication.
3. There is inconsistency in abstraction, i.e., some types enumerate values that are too specific.
4. It is not clear how states and semantics specific to a particular kind of component or task should fit into the generic types of states being established.

These issues led to two questions:

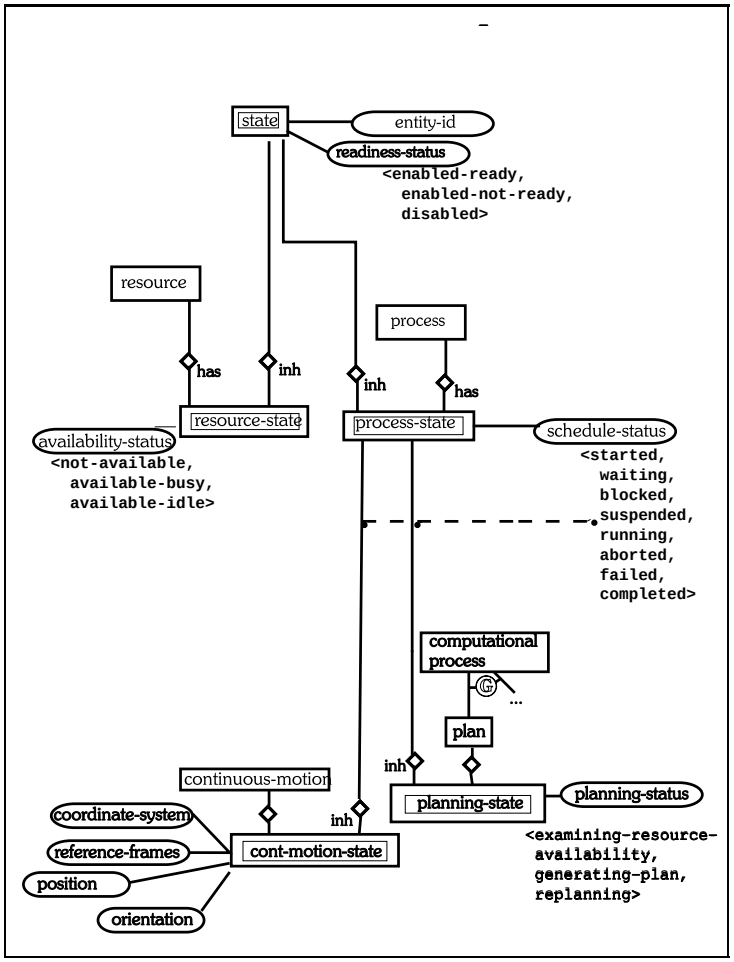
1. Should states be modeled as a weak entity (that is only an attribute for other entities) and not as a primitive data type, as defined in the NGC Schema [48]?
2. Would the architecture be simpler and more useful if a unified hierarchy of states were established?

We found that certain states of (automated) manufacturing equipment can be modeled in a generalization hierarchy, as mentioned by Rumbaugh [42, p. 94–98].
In Class-structure 23, we show an integrated approach to modeling the various types of states pertinent to a machine capable of continuous servo-controlled motion. At the most general level *state* has the attributes, *entity-id*, (whose state) and *readiness-status* (state of readiness). These attributes are almost independent of the process or the equipment.
At the next level, we have two orthogonal specializations:

1. By association with the entity *process*, we have *process-states* that could apply to almost any automatically controlled process.
2. By association with the entity *resource*, we have *availability* as a *resource-state*, applicable to any type of resource in the system.

The states of a planning task—a subclass of the entity *process*—form a simple example of nesting of states[42, pg 97].
The class *continuous-motion-state* is obtained by associating *process-state* with the *continuous-motion* class of tasks. In Class-structure 23, we show *continuous-motion-state* as an aggregation of orthogonal components [42, p. 95]. *continuous-motion-state*

coordinate-system, e.g., cartesian, cylindrical ...
reference frames
position
orientation



class-structure 23: Partial model of states of a manufacturing resource

We identyify three specializations (subclasses) of continuous-motion-state (not shown in Class-structure 23):

spherical-space-motion-state
cylindrical-space-motion-state
cartesian-space-motion-state

Some of the state-components are different for each of the subclasses. For example, cartesian-space-motion-state is described in terms of the attributes shown in Class-structure 24.
Next we apply this scheme to the specific case of the motion of a tool-tip in cartesian-space in a machining center. The motion task is to change the state of the tool-tip from some initial state, e.g., the current state, to some final state, e.g., goal of the motion task. Class-structure 25 shows the aggregation of the components of state, including the inherited state information.
Now we give some examples of how this general model can be applied to different configurations of machines, different types of moves and different ways to enhance accuracy.
In the case of a simple, stable single-axis motion, only one attribute may be changing.
In the case of a simple, stable translational multi-axis motion,

cartesian-space-motion-state
reference-coordinate-frame-id
reference-position-x
reference-position-y
reference-position-z
reference-orientation-gamma
/*rot about ref-x-axis*/
reference-orientation-beta
/*rot about ref-y-axis*/
reference-orientation-alpha
/*rot about ref-z-axis*/
position-x
position-y
position-z
orientation-gamma
orientation-beta
orientation-alpha
velocity-x
velocity-y
velocity-z
velocity-gamma
velocity-beta
velocity-alpha

class-structure 24: Cartesian-space-motion-state

instance-id: id of the object, e.g., tool-tip
readiness, e.g., enabled-ready
process-state, e.g., running
reference-coordinate-frame-id
reference-position-x
reference-position-y
reference-position-z
reference-orientation-gamma (rot about ref-x-axis)
reference-orientation-beta (rot about ref-y-axis)
reference-orientation-alpha (rot about ref-z-axis)
position-x
position-y
position-z
orientation-gamma
orientation-beta
orientation-alpha
velocity-x
velocity-y
velocity-z
velocity-gamma
velocity-beta
velocity-alpha

class-structure 25: Tool-tip motion-state in a cartesian coordinate system machine

many of the orientation attributes may have fixed values.
The reference positions and orientations may be defined as
a constant nominal value
+ a function of certain temperature measurements
+ a function of these temperature measurements and axes-positions, as mentioned in Class-structure 19.
The reference coordinate-frame itself may be defined in terms of some other reference and may be changing as a function similar to that described above.
Thus, even if no axis in a multi-axis tool-tip motion was being actuated to move, but temperatures changed, the result would constitute a change of state for the tool-tip motion.
The novelty in this approach is the organization of information for on-line use of emerging technologies to enhance machine accuracy.

6.5 Synthesis at a workstation level

Let us consider manufacturing resources organized as an automated workstation, in Class-structure 6. A specific workstation would be an aggregation of resources of one or more of the types shown in the struct. Its operations would be the union [12] of the operations of its constituents.

6.6 Modeling maintainability aspects of machine tool performance

The focus of this study in the earlier parts had been on modeling the operational capabilitites, constraints and behavior of servo-controlled manufacturing automation. One problem with such models is that operational behavior does not stay the same over time. However, modifications of parameter values in the model help in maintaining fidelity of representation, in many cases. For example, when the changes are reversible [16] or the changes represent "settling" or "wearing in" and are "convergent".
However, there are forms of equipment degradation ("irreversible" changes) due to wear and tear in "normal" operation that will affect the operational behavior of the machine over time. Incorporation of leading causal factors in the model of the machine tool may help track key changes in a beneficial manner. This potential benefit is the motivation for modeling and tracking maintenance-related characteristics of a machine.

6.6.1 Relation to prior machine-behavior modeling work

Modeling machine tool behavior in academic work has been pursued primarily to facilitate design and adaptation of control. A second objective has been to design and improve operational plans and process programs. A third objective, not well pursued, has been to assist in prognostics and diagnostics. In certain cases, the same models, tests and procedures that help in the first objective also serve the third objective. However, degradation does not show up soon enough in these tests. Some of the reasons for which degradation-modeling has not been pursued very much are:

1. Complexity of the physical phenomena
2. Unknown variability of the operating conditions
3. Very long research time-span [17]
4. Very costly implementation [18].

[16] e.g., effects of tolerable fluctuations in bulk temperature
[17] many years
[18] cost of monitoring equipment and labor

The following premises acn help reduce the complexity of the problem:

1. Some aspects of the degradation processes are predictable.
2. Some aspects are observable at relatively less cost during "normal" use.
3. Some types of on-line record-keeping at the machine has become more affordable in terms of computer-resources and record-keeping labor.
4. Setup [19] of such record-keeping could be made more affordable by identifying classes of *machine elements* and *subsystems* that exhibit similar degradation behavior.
5. Some behavior-changes could be anticipated or detected early; thus, corrective action could be identified and performed sooner.
6. With provision for capturing unanticipated degradation events, the system could support a *learning component* of the system.

The generic component model, Class-structure 10 and 11, includes attributes concerning component life, degradation, and life-consumption, designed to help in preventive maintenance. Some factors affecting life-consumption are easily available during operation, e.g. duration for which the monitored element is subject to some load, speed, or temperature. However, typical controllers of manufacturing machinery do not provide on-line facilities to capture this data. Typically, separate maintenance-management systems are installed to monitor the equipment, but without the benefit of the crucial operational data, e.g., the time history of load, speed, and temperature to which the component is subjected. We demonstrated the potential of this idea through a small prototype reported in Subsection 7.3.

7 Early prototyping

As recommended by McCain in Subsection 2.1, we experimented with rapid protoyping of a few ideas embodied in the vision described in this report. Four different teams participated at four different times to work on four different parts: user-interface specifications, user-interface screen-views, machine-tool accuracy enhancement, and preventive maintenance.

7.1 Prototyping User-interface

The first step was to develop specifications for a user interface that integrates the functions of engineering, maintenance, setup, and operation. The motivation was to share (and thus reuse) information across these functions, traditionally not integrated at the point of use. One output of this effort was to identify the roles of different types of users, as documented in Appendix C and reviewed with a wide cross-section of industry representatives in the *Next Generation Controller* (NGC) program. A second output was a specification of a set of user-interface screen-views, assimilated from studies of user interfaces on commercial numerical controllers, and robots and other specifications for such user interfaces, and our own experiences with needs on other projects. Our initial focus was on content of information that should be integrated and reused, rather than on the aesthetics of screen layout.
In the second step, most of this specification was prototyped on an IBM-PC-compatible desktop computer, using Borland's *ObjectVision*, a software tool that integrates screen views [20] with a relational database under Borland's *Paradox* database engine. This "live-screen" prototype gave us a base for discussions with domain experts in industry and faculty in the Mechanical Engineering Dept at the University of Michigan. These reviewers were satisfied that the information-content represented and integrated in the scope was comprehensive.

7.2 Prototyping model to enhance accuracy of machine tools

In the third step, we launched a rapid prototype to model the kinematics of simple machine tools, for further interaction with machine tool researchers, at the University of Michigan, experienced in enhancement of kinematic accuracy of machine tools. These researchers were not applying software-reuse technology in their research projects to develop and prototype accuracy-enhancement software for machine tools. These researchers are now evaluating the prototyped software-reuse approach, by building error-correction software for different machine configurations. Figure 7 shows the scheme. Generic prismatic joints and rotational joints are combined in different arrangements to form generic or reusable substructures. To define the configuration of a particular machine, appropriate substructures are reused from the library and connected as required, and specific values for the joint parameters are provided. Toolholder assemblies and fixture assemblies are also treated as substructures. Thus, the scheme allows the kinematic modeling of the complete loop from the work-point on the tool, through the toolholder, tool-supporting substructure of the machine, ground, workpiece-supporting substructure of the machine, fixture, and, finally, the work-point on the workpiece. Connectivity is defined by extending the ISO kinematic model [22].
The mechanical engineering researchers [21] agreed that the scheme could save software-creation labor by reusing previous knowledge captured in the database. However, they also felt that the commercial software tools used in the prototype were not sufficiently robust to commit a project for industrial use.

7.3 A prototype for preventive maintenance of machine elements

In order to demonstrate the potential of software-reuse in preventive maintenance, as mentioned in Section 6.6, we prototyped a database to record the usage history, maintenance performed and to schedule preventive maintenace, based on given degradation model and life-consumption, calculated from the usage history. A

[19] possibly machine-specific customization

[20] as forms

[21] Dr. Milton MU and Zhang Bai

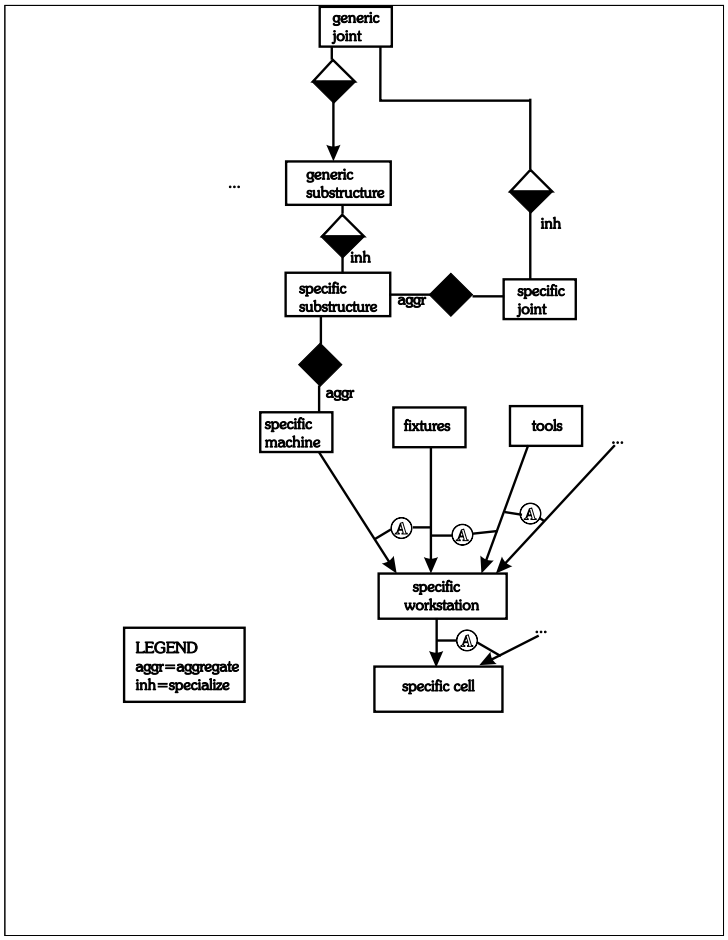


Figure 7: Scheme for synthesizing kinematic models of machine tools

key part of the database was a multi-level aggregation-hierarchy (from a degradeable component to a machining cell). This model is shown in Figure 8. The model was built using the Oracle relational database management system, very soon after installation of their X-Windows version of *SQL Forms* at the University of Michigan. The conclusion of the student team was that the *triggers* in the *SQL Forms* were so cumbersome to build that such software-tools might not be acceptable for factory-use.

8 Conclusion

We have defined and experimentally prototyped a process to develop a conceptual model of the domain of manufacturing automation. We defined the domain in a manner that supports economic extension, building on prior knowledge, including conceptual primitives, and an architecture for monitoring and control applications. Through this experience and study, we have also identified certain technological limitations that require high-skill effort in the realization of novel applications in the manufacturing automation

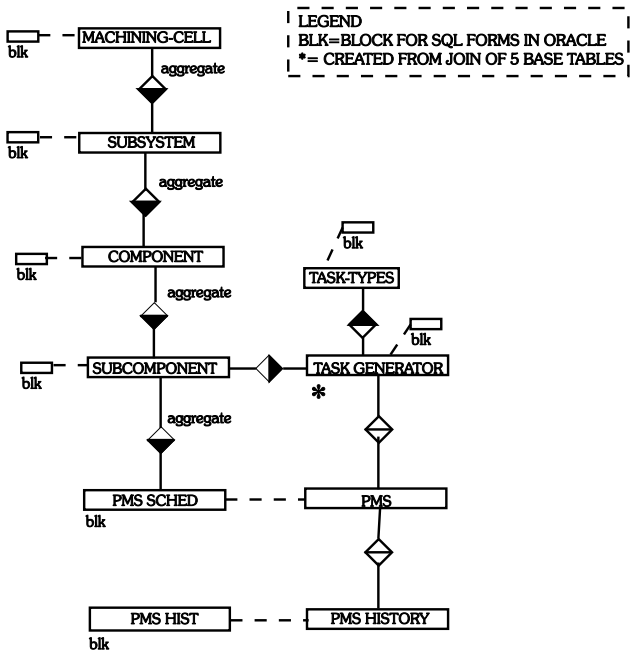


Figure 8: Aggregation hierarchy used to monitor degradable machine components

domain. We propose future work to investigate and resolve some of these issues.

8.1 Contributions in methodology

Our method to develop the conceptual model for manufacturing equipment is a synthesis of recommendations from many diverse sources, with some novelties that we describe next.

Early user-interface prototyping Our first step was a departure from published methodologies. We began rapid prototyping a user interface at a very early stage, for partial validation of the scope of applications and functions that users [22] would value. We are building on the lessons learned, in an iterative and incremental manner.

Early economic tradeoff assessment Another departure from published methodologies was the performance of an early version of a *market analysis* [31] in terms of potential *gain* in applicability versus the *pain* of assuring applicability, i.e., assessing degree of flexibility against degree of development uncertainty, cost, and time.

Building on readily available scientific knowledge Rather than proceeding to acquire more knowledge about the domain, we chose to begin the modeling process with *readily available*

[22] researchers and developers of applications

knowledge, in keeping with our incremental, iterative approach. By *readily available knowledge* we mean the established body of knowledge in the field of manufacturing engineering and the underlying foundations of mechanical engineering and physics. We believe that the use of this systemized body of knowledge leads to an architecture with better conceptual integrity and completeness than the traditional approach in expert systems of extracting knowledge from a human expert, or the approach of evolving generalizations from a "representative" set of applications or scenarios. Our approach also has an advantage over general knowledge-building approaches such as Cyc, in terms of cost-effectiveness.

Anticipating extensions Another new feature of our process was to establish expectations of extensibility early on. Again, we have addressed the subject from the perspective of underlying issues documented in literature. Our approach has an advantage over the traditional industrial approaches of *customer surveys* and *forecasting trends*, in terms of enabling the evolution of a conceptual architecture with integrity and completeness.

8.2 Contributions in domain analysis

We have employed the notion of a soft boundary [23] for the domain, as an approach to treating reusability and extensibility as joint goals.
We have identified a network of inter-related (sub)domains that help characterize the domain of programmable servo-controlled manufacturing equipment. Some of these domains help bound the scope of the tasks that the equipment will perform and the environment in which it will operate. These boundaries, in turn, assist in bounding the scope of the manufacturing equipment. We have characterized these tasks and the environment in a way that allows extension of the conceptual model from the initially chosen domain of a basic machine to the more useful, but wider, domain of a small-cell-level manufacturing system. We have characterized the products to be manufactured and the processes to be executed on the manufacturing equipment in such a way that initial equipment-models can be developed with relatively low cost and risk of error. Yet, our characterizations allow extension of the domains of products and processes in relatively small steps.

8.3 Contributions in architectural model

Our conceptual model integrates multiple approaches to software architectures for such automation. This whole effort—from the statement of purpose to the model of an actuator or sensor—can be viewed as architectural modeling at different *levels* from different perspectives.

Hierarchical model-specialization We have proposed a nested hierarchy of model-specializations that provides a systematic way of simplifying equipment-models to suit the purpose and timing-constraints of lower control levels. To our knowledge, such systematic specialization in the same system has not been reported.

[23] However, it also results in an involved definition of the limitations of the modeling framework.

State-hierarchy We have proposed a generalization-hierarchy of states of a (manufacturing) machine that is new, to our knowledge, in terms of the use of multiple-inheritance, the conceptual primitives devised, and the width of their applicability.

8.4 Contributions in conceptual building blocks

We have proposed a model of elements of a basic machine in terms of a combination of a product-ontology and a process-ontology. Although the idea is borrowed from Cyc, its application to elements of manufacturing machines is new.

Prototype for kinematic modeling A prototype based on a relational model was constructed to define and apply the kinematic configuration of multi-axis numerically-controlled machine tools.

8.5 Limitations and further work proposed

The major technical limitation is that this type of modeling is an iterative and incremental process [9]. Significant effort will be required to evaluate the reusability and extensibility of software based on the proposed conceptual model. This study is a small early step in a long spiral road of incremental iterations.

Inadequacy of tools poses another obstacle. Early pre-prototyping experiences using relatively "mature" relational database systems showed that these tools are still not easy to use. In certain cases, it appeared that multiple inheritance could reduce reuse effort, but support for implementation of multiple inheritance was not available in commercial tools.

Future work We propose to construct a rapid prototype of a subset of the conceptual model, using an object-oriented CASE Tool, for evaluation, as mentioned above. Some of the steps in the prototyping process are given below.

1. Develop a more precise boundary for the domain, in consultation with researchers and developers in the application domain, particularly for the subdomains described in Sections 3.5 and 3.8.
2. Select a subset of the proposed model and prototype it.
3. Develop a test plan and metrics for evaluating the technical aspects of the reusability and extensibility of the model, including review by application developers. Follow Procedure for validating the adequacy of these models, outlined in Section 2.4.
4. Select the prototyping infrastructure and tools.
5. Modify the model to fit within the limitations of the available tools.
6. Implement the prototype.
7. Perform the designed tests.
8. Analyze the experiences.

9. Document the results.
10. Plan further work.

A Primitive entities in the NGC Schema

The NGC Schema [48] starts with a number of primitive data types—some examples are shown in Figure 9. Some other examples are listed below:

1. controller-action-type: Actions that cause an entry in the controller-monitor-log.
2. exception-effect-type: Occur when exception raised, but without successful recovery.
3. exception-type: that occur in the system, e.g., some limit exceeded.
4. effector-type:
5. function-name-type: those that can be remotely executed.
6. measurement-type:
7. measurement-units-type:
8. sensor-type:

TYPE component-type
    ENUMERATION OF
        (switch ...)
coordinate-type
    SELECT
        (cartesian-, spherical-, cylindrical-coordinate)
coordinate-system-type
    ENUMERATION OF
        (cartesian- ...cylindrical-space)
coordinate-space-type
    ENUMERATION OF
        (work-space, actuator-space)
database-ms-type
    ENUMERATION OF
        (paradox, oracle, ingress, sybase, dbase)
euler-angle = REAL
entried-type
    ENUMERATION OF
        (table, log)
execution-direction-type
    ENUMERATION OF
        (forward, backward)

Figure 9: Examples of primitive data types identified in the NGC Schema.

B States, goal-outcomes, values and events in NGC Schema

Following are extracts from the NGC Schema [48] that pertain to definitions of state, event, goal-outcome and value, along with our comments. First is an extract of state- and status-related types:

component-state-type: ENUMERATION OF (open, closed, flood, mist, empty, ...) [24]

robot-hand-state: ENUMERATION OF (open, closed) [25]

discrete-states: ENUMERATION OF (on-state, off-state, ...)

log-status: ENUMERATION OF (requesting-log, generating-log, finished-log) [26]

generic-state: (pertains to motion) [27]
    resource
    phenomenology [28]
    sensed-value
    units
    coordinate-system
    abs-rel
    position
    orientation

system state: [29] ENUMERATION OF (quiescent, started, idle, running, suspended, halted)

Next is an excerpt [48] showing types related to events and goal-outcomes.

goal-status: ENUMERATION OF (ready, executing, halted, suspended, aborted, completed) (Comment: How is a goal-status different from the state of the task?)

goal-outcome: ENUMERATION OF (success, failure) (Comment: Why is outcome not simply one of the states of a task?)

event-type: ENUMERATION OF (exception1, ..., finished-plan-step, successful-func-call, aborted-func-call, ...) [30]

C Types of users of machine controllers

Following is a description of different types of users of a system to monitor and control flexible manufacturing equipment. These user types were identified in the NGC Requirements Definition Document [36] and the NGC Schema [48]. The descriptions also refer to the *Needs analysis document* [44]. The descriptions are ordered by increasing scope of tasks and responsibilities. This order is based on the premise that a user is trained in all tasks up to the user's own type in the ordered list.

[24] Comment: The last 3 examples do not apply to many components.

[25] Comment: The fingers of a robot could be continuously controlled too. If they have only two discrete states, then how are they different from other discrete-state-components? How are they different from the tool-retention-mechanism in a spindle-nose?

[26] Comment: Why should logging be distinguished from any other task?

[27] Comment: It is not clear how a model of states should differ from a model of the measurement

[28] other attributes focus on position.

[29] Comment: These states seem to be a mixture of states for components or resources and states for tasks. Perhaps, there should be a generic set of states for equipment or resource that will execute a task.

[30] Comment: How are these events different from the states of any process, task or request?

Attendant Performs routine, normal operation and associated repetitive tasks. End-user of functions described in *Needs analysis document Table 3.2-3 items 2d, 2f, 4, 5, 6; Table 3.2-7 items 5, 6e, 6f; Table 3.2-9 item 20.*

Operator Additionally, performs associated planning, preparatory and evaluative tasks, e.g., *manual data input.* End-user of functions described in *Needs analysis document Table 3.2-3 items 2c, 2e, 2g-i, 2l-n, 3a, 7, 8, 9a, 9c, 13, 16, 17; Table 3.2-6 items 1b2, 1b3, 1b6, 1j; Table 3.2-7 items 6h1; Table 3.2-9 items 12d, 20.*

Setter and manager of workstation Additionally, user of functions described in *Needs analysis document Table 3.2-1 items 7a, 7i-j, 7p, 9a, 9c; Table 3.2-3 items 10a, 12, 15, 18; Table 3.2-1 item 22c; Table 3.2-2 items 1b-h, 1l-m, 6; Table 3.2-6 items 1a, 1b1, 1c, 1e, 1f, 1g, 1h, 2, 3d, 10, 12; Table 3.2-9 items 12, 20.; Table 3.2-9 items 5c-e, 12d, 20.*

Process planner, programmer for specific parts For complex tasks, user of functions described in *Needs analysis document* Table 3.2-1 items 2c3, 5h; Table 3.2-2 items 6; Table 3.2-6 items 1k, 1m, 3c; Table 3.2-7 item 15; Table 3.2-9 item 11, 12, 20.

Process planner, programmer for *master process programs* e.g., programming families of parts *Needs analysis document Table 3.2-6 items 1a.* User of capabilities described in *Needs analysis document Table 3.2-9 item 20.*

Maintenance–troubleshooter User of capabilities described in *Needs analysis document Table 3.2-1 items 6a, 7a.4, 7c.4-6, 7d, 7e-g, 7k-l; Table 3.2-9 item 20.*

Maintenance–repair-person User of capabilities described in *Needs analysis document Table 3.2-1 items 6d; Table 3.2-9 item 20.*

Maintenance–test and calibration-technician User of capabilities described in *Needs analysis document Table 3.2-1 items 7c.8-9, 7m; Table 3.2-9 item 20.*

Process development engineer Updates knowledge about manufacturing processes; includes process-related properties of workpiece materials; includes knowledge to determine processing parameters; includes part-feature-definitions; includes sequence of operations for defined part features; includes tooling configurations, how to select and apply tooling, performance characteristics, tool-degradation- models, procedures to monitor, inspect and calibrate tools and corresponding corrective, protective and replacement procedures; includes knowledge for statistical process control. *Needs analysis document Table 3.2-3 items 5a, 6b4, 6c, 6f5, 9b, 12b, 15, 18; Table 3.2-1 items 2.e, 5l, 7a, 18, 22c, 22e; Table 3.2-2 items 1b, 1c, 1d, 1g, 1j, 1m, 4a3, 4a4, 8; Table 3.2-4 items 2f, 3m; Table 3.2-5 items 2e, 5, 6, 7; Table 3.2-6 items 1c, 1e, 1f, 1g, 1i, 1k, 1m, 2, 7a, 10, 12; Table 3.2-7 items 5, 6, 8.* User of capabilities described in *Needs analysis document Table 3.2-1 items 7h; Table 3.2-9 item 11, 20.*

Equipment development engineer Updates knowledge about equipment configuration, capabilities, constraints; includes kinematic and dynamic models of driven mechanisms; includes procedures to monitor, test and calibrate the equipment and its elements and corresponding corrective or protective procedures and help- or repair-guide. *Needs analysis document Table 3.2-3 items 5a; Table 3.2-1 items 2e, 5l, 7a, 7c, 7d, 7j, 7k, 9c, 22d, 22f, 22g, 22h; Table 3.2-4 items 3r, 3s; Table 3.2-5 items 3, 6; Table 3.2-6 items, 3c2.*

User of capabilities described in *Needs analysis document Table 3.2-1 items 7n-o, 7q, 20, 22b; Table 3.2- 4 items 2a-d; Table 3.2-9 item 20.*

Control logic designer Creates, updates machine control logic.

User of capabilities described in *Needs analysis document Table 3.2-5 item 4; Table 3.2-9 item 20.*

Adaptive and Servo Control engineer Creates, updates knowledge or models or algorithms or parameters for controlling motion and other processes. (NA Table 3.2-1, 2e; Table 3.2-4 items 3, 4).

User of capabilities described in *Needs analysis document Table 3.2-1 items 2e.5, 2e.7; Table 3.2-4 items 7; Table 3.2-5 item 2d; Table 3.2-6 item 1b5; Table 3.2-9 item 20.*

System integrator–application development User of capabilities described in *Needs analysis document Table 3.2-1 items 2c, 2d.4, 2d.5, 2e, 3, 4, 5.a-m, 5.o-t, 8; Table 3.2-2 item 12; Table 3.2-3 items 1, 11, 13, 14, 16-18; Table 3.2-4 items 2, 3a, 5; Table 3.2-5 items 2.a-c; 8; Table 3.2-7 items 4, 7, 16, 17; Table 3.2-8 item 2; Table 3.2-9 items 3, 14f-g, 17, 19, 20.* Creator of capabilities described in *Needs analysis document Table 3.2-3 item 4-7; Table 3.2-5 items 3, 5-7; Table 3.2-6 items 1c; Table 3.2-7 items 5, 6, 9, 11a, 12, 13, 14.*

System integrator- multi-station machines, multi-machine systems User of capabilities described in *Needs analysis document Table 3.2-1 items 2a, 2d, 2e; Table 3.2-9 items 2, 4, 5, 14f-g, 16, 20.*

System integrator- controller to equipment within a workstation User of capabilities described in *Needs analysis document Table 3.2-1 items 2c, 2d, 2e.2, 2e.3, 2e.7-12; Table 3.2-4 items 6; Table 3.2-7 item 11b-c; Table 3.2-8 items 1, 3; Table 3.2-9 items 6, 20.*

System integrator- real-time-level Replaces, modifies or tests function and performance of some module (hardware or software or combination) in the real-time-control-platform or in the application running on it. User of capabilities described in *Needs analysis document Table 3.2-1 items 1, 2e.5, 2e.9; Table 3.2-4 items 1, 2; Table 3.2-5 items 1; Table 3.2-7 items 3, 11d; Table 3.2-9 items 10, 20.*

D Common kinematic configurations in manufacturing equipment

Following is an excerpt from the *NGC Requirements Definition Document* [36, Appendix E] that describes different types of machine configurations in common use in manufacturing industries.

The list is arranged in order of increasing complexity. The reviewers of the *NGC Requirements Definition Document [36]* considered these categories very comprehensive.

1. 1-axis motion(s) + a spindle to rotate tool or workpiece.
   (a) Few translational independent axes of motion.
   (b) Many independent, but concurrent, translational processing motions with closely coupled workpiece changing motions.
2. 2-axis motion(s) + a spindle to rotate tool or workpiece.
   (a) One such translational pair coordinated for continuous-path motion.
   (b) Several independent, but concurrent, translational-axis-groups, with closely coupled workpiece-changing motions.
3. "Two-and-a-half-axis" motion(s) [31].
   (a) One 3-translational-axis group, with only two axes coordinated for continuous-path motion.
   (b) One group as above, with part probing capability.
   (c) One such group with two rotational axes coordinated for continuous-path motion and the third axis providing concurrent translational motion, possibly including servo-controlled, instrumented end-effector.
   (d) Several axis-groups of the two categories above, with overlapping workzones or work volumes.
4. 3-axis translational motion(s), coordinated for continuous-path motion, + spindle-rotation motion(s) to rotate tool or workpiece; includes part-probing capability.
   (a) One such group, with up to three axes.
   (b) One such group, with spindle capable of being coordinated with other axes for continuous-path motion.
   (c) Two or more such groups, processing one or more workpiece(s), with overlapping workzones.
5. 4-axis motion(s), with three translational axes and one rotational axis; including spindle with automated clamping; including part-probing capability.
   (a) One such group, with three translational axes coordinated for continuous-path motion.
   (b) One such group, with all four axes coordinated for continuous- path motion.
   (c) One group, as above, with part-probe providing continuous signal proportional to displacement, driving axes to minimize signal deviation from neutral value.
   (d) One such group, possibly with peripheral automation for changing measuring tools and workpieces.
6. 5-axis motion, with three translational axes and two rotational axes; including spindle with automated clamping; rotational axes and spindle axis are orthogonal with common intersection; including part-probing capability.
   (a) One such group, with the rotational axes used for positioning only and the other three axes used for continuous-path motion.
   (b) One such group, as above; part-probe providing continuous signal proportional to displacement, driving axes to minimize signal-deviation from neutral value.
   (c) One such group with 5-axis-continuous-path motion.
   (d) One such group as above; part-probe providing continuous signal proportional to displacement, driving axes to nullify signal.
   (e) One such group, as above; utilizing nondestructive testing probes to measure other properties of workpiece.
   (f) Combinations of various types of axis groups described above, possibly with peripheral automation for changing and measuring tools, workpieces.
7. 5-axis motions as in item above but rotational axes and spindle axis are not orthogonal and do not have common intersection.
8. More than five axes of coordinated motions including tool rotation axis and workpiece-rotation axis.
9. Other kinematic combinations of axes generating planetary, helical, cycloidal and other geometrically definable paths.
10. Other combinations of axes including serially chained articulated joints.
11. Other combinations of axes including parallel-driven joints.

[31] 2 axes perform continuously coordinated motion and the third axis is a positioning axis

E Abbreviations and Acronyms

ASTM. American Society for Testing and Materials

IB. Information Base

ISO. International Standards Organization

IEC. International Electrotechnical Commission

MAP/TOP. Manufacturing Automation Protocol/Technical Office Protocol

mfg. manufacturing

MMS. Manufacturing Message Service

NGC. Next Generation Machine/Workstation Controller

NGC SOSAS. "Next Generation Machine/Workstation Controller" Specifications for Open Systems Architecture Standards

OSI. Open Systems Interface

PDES. Product Data Exchange System

SNMP. Simple Network Management Protocol

STEP. Standard for Exchange of Product Data

F Glossary

ACTUATOR. A device, e.g., a motor, that converts electrical, hydraulic or pneumatic energy to motion.

ARCHITECTURE. A view of a system that satisfies the overall system requirements. It provides for the definition, understanding and integration of all components of the system and their interrelationships and impact on others.

ARM. An interconnected set of links and joints between an end-effector and its support structure.

ATTRIBUTES. Data elements that represent characteristics to describe an object.

AXIS. Equivalent to a joint in a robot.

AXIS GROUP (COORDINATED). A group of axes whose motion is coordinated to produce a resultant trajectory within a specified tolerance (limit on deviation from specified trajectory and rate).

CELL. A manufacturing unit that has the capacity to manufacture (or completely transform) a (family of) product(s) from a "buyable" incoming form, e.g., casting or forging, to a "sellable" finished form with assured conformance to specs, with all the necessary resources within its own control.

CLASS. The abstract data typing mechanism of the object-oriented paradigm that groups objects with commonality of attributes and services.

COMPUTING PLATFORM. The computing infrastructure consisting of processor(s), memory and other storage media, other related hardware, system software, utility software, communication mechanisms and I/O interfaces.

CONCEPTUAL SCHEMA. Schema that is not configured for a specific implementation.

CONFIGURATION. The arrangement of a manufacturing workstation as defined by the nature, number, interaction and main characteristics of its functions. The features, customer options, software modules and engineering specifications that collectively define the functionality of a manufacturing workstation.

CONTAINER CLASS. A class whose instances are collections of other objects. Container classes may denote homogeneous collections (all of the objects in the collection are of the same class) or heterogeneous collections (each of the objects in the collection may be of a different class, although all must share a common superclass). Container classes are most often defined as generic or parameterized classes, with some parameter designating the class of the contained objects.

CONTEXT. The conditions that bound and relate discrete events temporally, spatially and semantically.

DATA ABSTRACTION AT CONCEPTUAL LEVEL. (in the context of databases, per Korth [25] A description of what data are actually stored in the database, and the relationships that exist among the data ... in terms of a small number of relatively simple structures ... (the abstraction) hides certain details of how the data is stored and maintained.

DATA MODEL. A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

DATA STRUCTURE. A formal representation of information for communication and processing purposes without regard to actual storage configuration.

DYNAMIC BINDING. An association of software elements not ncessarily made until the moment just before execution of that software.

EFFECTOR. A device or tool that physically performs an action or transformation function.

ELEMENT. Some component of a manufacturing system. It may be a subsystem decomposable into other elements or a logically "indivisible" element.

END EFFECTOR. An effector, located at the end of the last joint of a mechanism closest to the workpiece, by which an action on an object can be performed.

ENTITY-RELATIONSHIP MODEL. An object-based logical data model of a real world. It consists of basic objects called entities, and relationships among these objects ... and constraints to which the contents of a database must conform. An entity is an object that is distinguihable from other objects by a specific set of attributes. A relationship is an association among several entities.

ESPRIT. European Strategic Program for Research and Development in Information Technology—a major manufacturing technology program that includes all countries of the European economic community.

EXCEPTION HANDLING. Managing unacceptable deviations from the planned or nominal production or process.

HIERARCHICAL CONTROL. Control in which tasks are decomposed into sequences of subtasks that are passed on to the next lower level in the hierarchy until the lowest level is reached.

INFORMATION BASE. An abstract storage mechanism that may consist of multiple coordinated information stores (e.g., data bases, knowledge bases, memory maps). The conceptual structure to which all requests to store, access or manipulate shared data are made.

INSTANCE. A distinguished occurrence of a particular object.

INSTANTIATION. The creation of an instance.

INTEGRATION. The formation of one complete and harmonious entity or coordinated entities of different applications or processes that will accomplish a complete process solution.

INTERCHANGEABILITY. A characteristic of system components that makes it possible to replace one component with another of equivalent functionality, where the replacing component may be made by another vendor.

INTERFACE. The definition of a common boundary or a common application point of interaction between two or more distinct entities, defined by functional characteristics and input/output specifications.

INTEROPERABILITY. A characteristic of distinct entities in a system that makes it possible for these entities to intercommunicate and work together properly, to provide the required system functionality.

LATENCY. Time between the request for an action and the initiation of response to that request.

MANUFACTURE. *verb* **1.a.** To make or process (a raw material) into a finished product, especially by means of a large-scale industrial operation. **1.b.** To make or process (a product), especially with the use of industrial machines. **2.** To create, produce, or turn out in a mechanical manner. *noun* **1.** The act, craft, or process of manufacturing.

MANUFACTURING WORKSTATION. A material transforming device along with a workholding subsystem or device or fixture, a toolholding mechanism or device, a controller, including the computing platform and the human interface, and possibly such peripherals as work-exchanging and work-handling subsystem, tool-exchanging and tool-handling subsystem, part-gaging subsystem, and tool-gaging subsystem.

MODEL. A structured data representation that provides a description of a system or its elements, e.g., machine or process, and simulates its properties and characteristics of interest. Also see *data abstraction at conceptual level.*

OBJECT. An abstraction of an entity representing an encapsulation of its attributes and services on those attributes.

OBJECT-ORIENTED. A viewpoint that models data and behavior as objects.

OBJECT-ORIENTED PROGRAMMING. Programming in a language that embodies the concepts of objects, classes, inheritance, polymorphism and dynamic binding.

OBJECT-ORIENTED STRUCTURES. Two methods used to manage complexity, (i) classification structure, which captures class/member organization and (ii) assembly structure, which portrays whole/part organization.

OPEN SYSTEM. A system that provides capabilities that enable properly implemented applications to run on a wide variety of platforms from multiple vendors, interoperate with other system applications and present a consistent style of interaction with the user.

OPEN SYSTEM ARCHITECTURE. A specification of the capabilities or services that provide the interconnection structure and defines the interface between interoperating components, thus allowing applications to be integrated with a consistent style of interaction.

PARAMETERIZED CLASS. A class that serves as a template for other classes, in which the template may be parameterized by other classes, objects, and/or operations. A parameterized class must be instantiated (its parameters filled in) before instances can be created. Parameterized classes are typically used as container classes; the terms *generic class* and *parameterized class* are interchangeable [9].

PHYSICAL. *adjective* **1.** Of or pertaining to the body, as distinguished from the mind or spirit; bodily; corporeal. **2.** Of or pertaining to material things. **3.** Of or pertaining to matter or energy or the sciences dealing with them, especially physics.

POLYMORPHISM. In an object-orineted language, the characteristic of an operator that allows it to work on objects of different classes.

PORTABILITY. The ability to operate the same component on different computing platforms.

PRECISE. *adjective* Capable of, resulting from, or designating an action, performance, or process executed or successively repeated within close specified limits.

PROCESS. *noun* **1.** A system of operations in the production o something. **2.** A series of actions, changes, or functions that brings about an end or result.

verb **1.** To put through the steps of a prescribed procedure. **2.** To prepare, treat or convert by subjecting to some special process.

PRODUCTION. The creation of value or wealth by producing goods and services.

PRODUCT. Anything produced by human or mechanical effort or by a natural process.

PROTOCOL. A set of semantic and syntactic rules for achieving communication.

REAL TIME. Within a strict, predictable, bounded time interval.

REUSABLE SOFTWARE. Software that can be used for more than one implementation.

SCALABILITY. The capacity to increase or decrease the functionality of a system or its elements.

SCHEMA. A data model that defines all abstract data types including relationships, attributes and constraints.

SOFTWARE LIBRARY. A repository for software components.

UPGRADEABILITY. The characteristic of enhancing speed, capacity or functionality of a system or its elements.

VERSION. A formal record used to help track an object's evolution over time; a version may be associated with a context and loaded according to that context.

VIRTUAL MACHINE SERVICES. A set of domain-independent mechanisms that provide fundamental capabilities and a common software interface to (possibly independently developed) domain-specific functional entities, enabling their integration, independent of the implementation underlying these services.

VIEW. A presentation with limited perspective based on context.

WORKING ENVELOPE. A defined boundary representing the maximum extent or reach of a machine in all directions.

WORKING RANGE. All reachable positions within the working envelope. The range of any variable within which the system normally operates.

WORLD COORDINATE. A position relative to a frame of reference fixed on the manufacturing workstation.

WORLD MODEL. The system's estimate and evaluation of the history, current state and possible future states of the world, including the states of the system being controlled.

(Some definitions have been adopted from the NGC Schema [48]).

G Physical quantities and measurement units

The information shown in Tables 6—8 on page 42 corresponds to measurement-type and measurement-units-type in Appendix A.

Table 6: Basic physical quantities

Physical-quantity (basic)	Measurement-unit	sym-bol
length	meter	m
mass	kilogram	kg
time	second	s
electric-current	ampere	A
thermodynamic-temperature	kelvin	K
luminous-intensity	candela	cd
amount-of-substance	mole	mol

Table 7: Formulae for derived units with different names

Physical-quantity	Measurement-unit	formula
energy	joule	$N \cdot m$
charge	coulomb	$A \cdot s$
electromotive-force	volt	W/A
capacitance	farad	C/V
electric-resistance	ohm	V/A
magnetic-flux	weber	$V \cdot s$
inductance	henry	Wb/A
magnetic-flux-density	tesla	Wb/m^2
luminous-flux	lumen	$cd \cdot s$
illuminance	lux	lm/m^2
temperature	degrees Celsius	$T - T_o$
⋮	⋮	⋮

Note: In the formula for temperature, T=thermodynamic temperature and T_o=273.15 K.

Table 8: Derived physical quantities

Physical-quantity (derived)	Measurement-unit (derived unit)	symbol
plane-angle	radian	rad
solid-angle	steradian	sr
area	$meter^2$	m^2
volume	$meter^3$	m^3
frequency	per second	s^{-1}
density	$kilogram/m^3$	$kg \cdot m^3$
linear-speed	meters/second	$m \cdot s^{-1}$
linear-velocity	meters/second	$m \cdot s^{-1}$
linear-acceleration	$meters/second^2$	$m \cdot s^{-2}$
linear-jerk	$meters/second^3$	$m \cdot s^{-3}$
angular-velocity	radians/second	$rad \cdot s^{-1}$
angular-acceleration	$radians/second^2$	$rad \cdot s^{-2}$
force	newton	N
pressure	$newton/meter^2$	N/m^2
torque	$newton \cdot meter$	$N \cdot m$
moment	$newton \cdot meter$	$N \cdot m$
couple	$newton \cdot meter$	$N \cdot m$
kinematic-viscosity	$meter^2/second$	m^2/s
dynamic-viscosity	$newton \cdot second/m^2$	$N \cdot s/m^2$
energy	joule	J
work	joule	J
heat	joule	J
power	watt	W
charge	coulomb	C
potential-difference	volt	V
electromotive-force	volt	V
electric-field-strength	volt/meter	V/m
electric-resistance	ohm	Ω
capacitance	farad	F
magnetic-flux	weber	Wb
inductance	henry	H
magnetic-flux-density	tesla	T
magnetic-field-strength	ampere/meter	A/m
magnetomotive-force	ampere	A
luminous-flux	lumen	lm
luminance	$candela/meter^2$	cd/m^2
illuminance	lux	lx
wave-number	per meter	m^{-1}
temperature	degrees Celsius	$°C$
entropy	joule/kelvin	J/K
specific-heat-capacity	joule/kilogram kelvin	$J/kg \cdot K$
thermal-conductivity	watt/meter kelvin	$W/m \cdot K$
radiant-intensity	watt/steradian	W/sr

Bibliography

[1] Air Force Systems Command Aeronautical Systems DIV/PMRRC, *Department of Defense—1987 Machine Tool/Manufacturing Technology Development Conference—Executive Summary, Volume 1*, WPAFB, OH 45433-6503, June 1987. Documents industry needs formulated through discussions of invited presentations by over 300 conferees, subgroup workshops, and panel ranking.

[2] J. S. Albus, "RCS: A reference model architecture for intelligent control," *Computer*, vol. 25, no. 5, pp. 56–59, May 1992.

[3] Y. Altintas and C. L. Dong, "Design and analysis of a modular cnc system for machining control and monitoring," in *Modeling of Machine Tools: Accuracy, Dynamics, and Control*, ASME, 345 East 47th Street, New York, N.Y. 10017, 1990.

[4] *Standard practice for the use of the International System of Units (SI)*, American Society for Testing and Materials, 1989. ASTM Designation: E 380–89a.

[5] G. Arango, "From art form to engineering discipline," in *Proceedings of the 5th International Workshop on Software Specifications and Design*, pp. 152–159, 1989. Provides a unifying conceptual framework for domain analysis methods.

[6] T. Biggerstaff, "Reusability framework, assessment, and directions," *IEEE Software*, pp. 41–49, July 1987. advocates semantic binding; case for partial specs.

[7] T. Biggerstaff and A. Perlis, *Software Reusability Volume I*, ACM Press, New York, New York, 1989. introduction.

[8] S. Birla, J. Korein, et al. *Next Generation Workstation/Machine Controller (NGC)*, November 1987. draft requirements and program plan.

[9] G. Booch, *Object Oriented Design with Applications*, The Benjamin/Cummings Publishing Company, 390 Bridge Parkway, Redwood City, California 94065, 1991.

[10] R. J. Brachman, "I lied about the trees or Defaults and definitions in knowledge representation," *The AI Magazine*, pp. 80–93, 1985.

[11] F. P. Brooks, "No silver bullet," in *Information Processing 1986*, Elsevier Science Publishers B.V. North Holland, 1986.

[12] J. K. Chaar, *A methodology for developing real-time control software for efficient and dependable manufacturing systems*, PhD thesis, University of Michigan, Ann Arbor, Michigan, 1990.

[13] R. A. Collacott, *Mechanical Fault Diagnosis and Condition Monitoring*, Chapman and Hall, London, 1977. wide range of techniques, particularly for rotating machinery and other equipment with periodicity and patterns in emitted signals.

[14] R. Davis and B. Buchanan, "Meta-level knowledge: overview and applications," in *Proc. IJCAI*, pp. 920–927, August 1977.

[15] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, pp. 215–221, june 1955.

[16] M. A. Donmez, C. R. Liu, and M. M. Barash, "A generalized mathematical model for machine tool errors," in *Modeling, Sensing, and Control of Manufacturing Processes*, K. Srinivasan et al., editors, ASME Press, 1988.

[17] K. Forbus, "Qualitative physics: past, present, and future.," in *Exploring Artificial Intelligence: Survey talks from the national conferences on artificial intelligence*, H. Shrobe, editor, pp. 239–296. Morgan Kaufmann, 1988.

[18] P. Freeman, editor, *Tutorial: Software Reusability*, IEEE Computer Society Press, Washington, D.C., 1987.

[19] R. Guha and D. Lenat, "Cyc,: A midterm report," *AI Magazine*, pp. 33–59, Fall 1990.

[20] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, *Building Expert Systems*, volume 1 of *Teknowledge series in Knowledge Engineering*, Addison-Wesley Publishing Company, Inc., 1983.

[21] R. Hocken, "How to cope with on-machine measurement," *Manufacturing Engineering*, pp. 8–10, August 1993.

[22] *ISO CD 10303-105 Product Model Data Representation and Exchange: Part 105—Kinematics*, International Standards Organization, july 1991. Version: Sapporo 91.

[23] C. Isik and A. Meystel, "Pilot level of a hierarchical controller for an unmanned mobile robot," *IEEE Journal of Robotics and Automation*, vol. 4, no. 3, pp. 241–255, 1988.

[24] *Product data exchange specification, First working draft*, October 1988. Standard for the Exchange of Product Model Data.

[25] H. F. Korth and A. Silberchatz, *Database System Concepts*, McGraw-Hill, Inc., 1991.

[26] T. R. Kramer, "A library of material removal shape volumes (mrsevs)," Technical report, National Institute of Standards and Technology, Gaithersburg, Maryland 20899, 1992. NISTIR 4809: catalogs machined features coded in EXPRESS.

[27] C. Krueger, "Software reuse," in *acm computing surveys*, S. March, editor, pp. 131–183, Association for Computing Machinery, Inc., 1515 Broadway, New York, NY 10036, June 1992. Introduces notion of cognitive distance in reuse.

[28] P. Loucopoulos and R. Zicari, *Conceptual Modeling, Databases, and Case*, John Wiley & Sons, Inc, 1992.

[29] S. Lytinen, "Conceptual dependency and its descendants," *Computer Math. Applic.*, vol. 23, no. 2–5, pp. 51–73, 1992.

[30] Y. Matsumoto, "A software factory: An overall apoproach to software production," in *Tutorial: Software Reusability*, P. Freeman, editor, pp. 155–178, The Computer Society of the IEEE, 1987. Examples from the Fuchu software factory, Toshiba, Japan.

[31] R. McCain, "Reusable software component construction: A product-oriented paradigm," in *Proceedings of the 5th AIAA/ACM/NASA/IEEE Computers in Aerospace Conference*, pp. 125–135. AIAA, oct 1985. methodology—introduces domain analysis for components of an application.

[32] D. McDermott and E. Davis, "Planning routes through uncertain territory," *Artificial Intelligence*, vol. 22, pp. 107–156, 1984.

[33] A. Meystel, "Theoretical foundations of planning and navigation for autonomous robots," *International Journal of Intelligent Systems*, vol. 2, pp. 73–128, 1987.

[34] D. S. Nau, G. Zhang, and S. K. Gupta, "Generation and evaluation of alternative operation sequences," in *Quality Assurance through Integration of Manufacturing Processes and Systems*, volume PED-Vol.56, ASME, 1992.

[35] A. Newell, *Unified Theories of Cognition*, Harvard University Press, Cambridge, MA, U.S.A., 1990.

[36] *Next Generation Workstation/Machine Controller (NGC) Requirements Definition Document (RDD)*, 1989. Martin Marietta, Denver, Colorado.

[37] H. Opitz, *Werkstuckbeschreibendes Klassifizierungssystem*, Girardet, Essen, Germany, 1968. Genesis of group technology and part feature-based processing.

[38] D. L. Parnas and D. P. Siewiorek, "Use of the concept of transparency in the design of hierarchically structured systems," *Communications of the ACM*, vol. 18, pp. 401–408, July 1975.

[39] C. L. Phillips and H. T. Nagle, *Digital Control System Analysis and Design*, Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632, 1984.

[40] R. Prieto-Diaz, "Domain analysis for reusability," in *Proceedings of COMPSAC '87*, pp. 23–29. IEEE, 1987. Gives methodology in terms of data flow diagrams.

[41] R. Prieto-Diaz and G. Arango, *Domain Analysis and Software Systems Modeling*, The IEEE Computer Society Press, Los Alamitos, California, 1991. IEEE Computer Society Press tutorial.

[42] J. Rumbaugh et al., *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1991.

[43] R. C. Schank, "What is AI, anyway?," *AI Magazine*, Winter 1987.

[44] R. Schappell et al. *Next Generation Workstation/Machine Controller (NGC) Needs Analysis*, February 1990. Also contains ranking by criteria.

[45] M. A. Simos, "The domain-oriented software life cycle: Towards an extended process model for reusability," in *Proceedings of the Workshop on Software Reusability and Maintainability*. The National Institute of Software Quality and Productivity, October 1987.

[46] N. K. Sinha and B. Kuszta, *Modeling and Identification of Dynamic Systems*, Van Nostrand Reinhold Company, 135 West 50th Street, New York, N.Y. 10020, 1983. systemizes model construction, learning and validation.

[47] V. Solaja and D. Vukelja, "Identification of tool wear rate by temperature variation of a carbide tip," in *Annals of the International Institution for Production Engineering Research, Volume 22/1*, 1973.

[48] *Next Generation Workstation/Machine Controller (NGC) Specification for Open System Architecture Standard (SOSAS), Volumes I-VI*, March 1992. Martin Marietta.

[49] G. Spur, "Model for computing quasi-static and dynamic displacements of turning centers," in *Modeling of Machine Tools: Accuracy, Dynamics, and Control*, ASME, 345 East 47th Street, New York, N.Y. 10017, 1990.

[50] T. J. Teorey, *Database Modeling and Design: The Entity-Relationship Approach*, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[51] S. P. Timoshenko and J. N. Goodier, *Theory of Elasticity*, McGraw-Hill Book Company, 1970.

[52] A. Todorov, "An integrated software system supporting a machining cell in mechanical engineering," in *Complex Machining and AI-Methods*, pp. 90–98, Elsevier Science Publishers (North-Holland), November 1991. Proceedings of the IFIP TC5/WG5.3 Working Conference, Gausssig, Germany, 27–29 November, 1991.

[53] H. G. Vogt and P. Zaring, "Planning of operation-sequences with an ai-based knowledge-acquisition tool," in *Complex Machining and AI-Methods*, pp. 99–113, Elsevier Science Publishers (North-Holland), November 1991. Proceedings of the IFIP TC5/WG5.3 Working Conference, Gausssig, Germany, 27–29 NOvember, 1991.

[54] I. Yellowley and F. Barrow, "The stress-temperature method of assessing tool life," in *Proceedings of the 14th International MTDR Conference*, 1973.