AN ALGORITHMIC STRATEGY FOR AUTOMATED GENERATION OF MULTI-COMPONENT SOFTWARE TOOLS FOR VIRTUAL MANUFACTURING

Patrick N. Bless
Graduate Research Assistant
Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, Il, 61801
Email: bless@uiuc.edu

Shiv G. Kapoor
Professor, Fellow ASME
Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, Il, 61801
Email: sgkapoor@uiuc.edu

Richard E. DeVor
Professor, Fellow ASME
Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, Il, 61801
Email: redevor@uiuc.edu

Diego Klabjan
Assistant Professor
Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, Il, 61801
Email: klabjan@uiuc.edu

ABSTRACT

This paper describes an algorithmic strategy to facilitate the generation of multi-component software tools for Computer Aided Manufacturing (CAM) and Virtual Manufacturing (VM). Components that are often used to build CAM and VM applications include a wide range of domain-specific knowledge sources and also more general utility components with often very heterogeneous characteristics. The identification of a suitable and compatible set of these components is the first and arguably most important step during the development process of any CAM or VM application. This paper presents an algorithmic strategy that automates this development step by solving a time-expanded network problem, referred to as the Component Set Identification (CSI) Problem. A definition of the CSI Problem, a mathematical formulation, a solution procedure, and some computational results are presented. Finally, an application to predict hole quality in drilling is used to illustrate the functionality of the proposed algorithmic strategy.

Keywords: Computer Aided Manufacturing, Virtual Manufacturing, Information Integration, Integer Programming, Component Set Identification Problem.

Introduction

There is a growing demand for intelligent manufacturing software to assist both researchers and practitioners in solving manufacturing engineering problems such as machine tool design, process selection, process planning, and system and process optimization. Traditionally, the need for these software tools has been satisfied by a variety of stand-alone applications [1–6]. These applications generally integrate two classes of subcomponents; components providing the specific domain knowledge (knowledge sources) and components providing process independent functionality (utility components), such as geometry analysis tools, optimization routines, or visualization software. Sometimes the domain knowledge is available in form of analytical models [7, 8] that focus not only on a particular process or system, but also specialize on a very specific aspect of the process or the system under study, such as thermal effects, static and dynamic force analysis, tool wear or economical aspects. Knowledge about manufacturing systems and processes is also available in form of simple data repositories [9, 10] and in form of expert systems, based on rules discovered by experienced professionals and by using statistical data analysis techniques [11]. The subcomponents required to build a certain CAM application generally use different semantic mappings and are therefore often incompatible. These characteristics turn the subcomponent inte-

gration process into a very complex task that has to be performed manually at development-time. This results in rather static applications that generally do not support the integration of additional components or new research results at run-time.

The above issues have motivated researchers to pursuit more flexible approaches to CAM software design. These development efforts can be divided into two areas; the development of suitable systematization schemes for the relevant domain knowledge, and the development of integration and communication paradigms for software components. Some examples for systematization efforts include the Standard for Exchange of Product Model Data (STEP) [12], or the Process Interchange Format (PIF) [13], efforts that can be described as the development of a common translation language serving as a bridge among heterogeneous data representations. Other examples include the Process Specification Language (PSL), a neutral representation for manufacturing processes developed by the National Institute of Standards and Technology (NIST) [14]; the Language for Process Specification (ALPS) [15], the Toronto Virtual Enterprise Project [16], and the Enterprise Ontology Project [17]. The main goal of these systematization efforts is to overcome the difficulties associated with the integration of semantic mappings. Often the same term may be interpreted differently by different knowledge sources, or the same concept may have multiple names, leading to overlap that can further complicate the integration of multiple semantic mappings.

Integration of and communication among software components and computer programs started with SIMULA I [18], the first computer language supporting concepts of object orientated programming developed in the early sixties. In the 1980's, the idea of reusable software modules [19] emerged and this idea eventually evolved to today's integration concepts such as agent technology and middleware standards [20]. Some of the research specifically targeted at the development of integration and communication protocols for manufacturing related applications is conducted by international research consortiums such as the Intelligent Manufacturing System Group [21] (e.g. GNOSIS Project, HMS) and the Object Management Group [22] (e.g. CORBA, UML, or XMI). Other research efforts can be credited to individual research groups [23, 24].

Considering the advances in both domain knowledge systematization, and integration and communication protocols, it seems to be possible to automate the process of CAM application development. Missing, however, are algorithmic strategies to identify suitable combinations of knowledge sources. Integrated into a suitable framework, such strategies can bridge the gap between the systematization paradigms and the integration tools. Having presented a potential architecture of an overall framework in an earlier paper [25], this paper focuses on the development of the algorithmic strategies that are required to explore the systematized domain knowledge and to identify suitable subsets of knowledge source and utility components able to perform the requested manufacturing planning, optimization, or simulation tasks.

The problem of identifying combinations of knowledge sources and utility components will be referred to as the *Component Set Identification (CSI) Problem*, a problem sharing some similarities with Directed Steiner Network Problems (DSN) [26] and scheduling problems with precedence constraints [27]. Due to the particular network topology of the CSI problem with its characteristic set of precedence constraints and the iterative solution procedure that is inherent to many manufacturing-related knowledge sources, the problem is formulated using a time-expanded network representation. The CSI problem has a unique network topography, including a generalization of time-induced precedence constraints that differentiate the CSI Problem from other optimization problems.

The remainder of this paper is organized as follows. Section 2 presents a characterization of the Component Set Identification Problem and establishes the terminology used throughout this paper. In the third Section we develop a time-indexed integer programming (IP) formulation using a dynamic (time-expanded) network representation of the problem. Section 4 presents computational results obtained by solving randomly generated test cases, addresses the limitations of the IP-formulation (exact method), and briefly discusses a heuristic approach. Finally, Section 5 illustrates the functionality of the proposed algorithmic strategies using drill-hole quality prediction as a practical example.

The CSI Problem

In order to apply the proposed algorithmic strategy to a user-defined, manufacturing-related simulation or optimization task and to automatically solve the CSI problem, the task has to be transformed into a suitable representation allowing computerized processing. Potential user-requests could be: (a) simulate a particular manufacturing process and predict the resulting surface finish and achievable tolerances; (b) compare several processes with respect to their ability to perform a specific operation; (c) optimize a particular process with respect to certain process attributes. In order to characterize these user-defined tasks analytically, a clear definition of terms is required. Table 1 provides these definitions.

Given the definitions presented in Tab. 1 any simulation or optimization problem can be characterized by defining the following six data sets. As an example the problem of simulating a metal cutting process to predict the resulting surface finish and achievable tolerances is being considered:

1. A set $\mathbf{P} = \{P_1, P_2, P_3, ..., P_m\}$ representing all properties (attributes and parameters) that are related to the manufacturing task under study and that are part of the systematized domain knowledge of the overall system; Example:

Table 1. TERMINOLOGY AND NOTATION.

(Process) Attribute	–	Inherent characteristic or quality of an entity (process)
(Process) Parameter	–	Variable quantity or quality that determines a particular aspect of an entity (process)
Property	P_j	Synonym to refer to both attributes and parameters of entities and processes
Known Property	P_j^k	Parameter or attribute j that is known to the user
Target Property	P_j^*	Parameter or attribute j that is sought by the user
Knowledge Source	K_i	Mapping between two sets of properties, $\mathbf{S}_i^{in}$ (inputs) and $\mathbf{S}_i^{out}$ (outputs)
Input Properties	$\mathbf{S}_i^{in}$	Properties that have to be supplied in order to use (consult) knowledge source K_i
Output Properties	$\mathbf{S}_i^{out}$	Properties returned by (calculated by) knowledge source K_i

P={cutting speed, cutting feed, depth of cut, tool geometry, workpiece material, tool material, workpiece dimensions, tolerances, surface finish, dimensional accuracy, radial force, tangential force, cutting temperature, etc.};

2. A subset $\mathbf{S_I} \subseteq \mathbf{P}$ representing all properties that are initially known; Example: $\mathbf{S_I}$={cutting speed, cutting feed, tool geometry, workpiece material, tool material};
3. A subset $\mathbf{S}^* \subseteq \mathbf{P}$ representing all explicit target properties of the task; Example: $\mathbf{S}^*$ ={dimensional accuracy, surface finish}; It is important to note that the set of properties that is eventually computed during a particular solution to a CSI problem instance does not necessarily correspond to $\mathbf{S}^*$. Additional properties that are not part of the set of explicit target properties might become *'known'* in one of the intermediate steps.
4. A set of task-relevant knowledge sources, $\mathbf{K} = \{K_1, K_2, K_3, ..., K_n\}$; Example: $\mathbf{K}$ ={cutting force models, chip load models, calibration models, surface generation models, cutting temperature models, ploughing models, chip formation models, empirical cutting force data, etc.};
5. Two sets of arcs, $\mathbf{S}_k^{out}$ and $\mathbf{S}_k^{in}$ for $k = 1, ..., n$ representing the input and output mappings between knowledge sources and properties; Example: Cutting force models require process parameters as inputs $\mathbf{S}^{in}$ ={cutting speed, cutting feed, depth of cut, etc.} and produce the cutting forces, $\mathbf{S}^{out}$ ={radial force, tangential force, etc.}.

The Time-Indexed IP-Formulation of the CSI Problem

The CSI Problem as a Directed Graph

Given the sets introduced in the previous section, the CSI problem is modelled as a directed graph, consisting of a set of property and knowledge source nodes and a set of arcs whose elements are ordered pairs of distinct nodes. This is illustrated in Fig. 1 for a small example problem with 25 properties and a knowledge source library consisting of twelve knowledge sources.

In Fig. 1, $\mathbf{S_I}$, the set of initially known properties includes properties P_{17} and P_{18} ($\mathbf{S_I} = \{P_{17}, P_{18}\}$), and $\mathbf{S}^*$, the set of target properties consists of properties P_9 and P_{22} ($\mathbf{S}^* = \{P_9, P_{22}\}$). All knowledge source nodes have a set of incoming and outgoing arcs connecting them with their respective input ($\mathbf{P}_j^{in}$) and output property nodes ($\mathbf{P}_j^{out}$). These input and output mappings translate into a number of precedence relationships among the twelve knowledge sources. Starting out from the set of initially known properties, $\mathbf{S_I}$, it is possible to search for feasible directed *sub-networks* connecting some or all of the elements in $\mathbf{S_I}$ with all elements of the target property set, $\mathbf{S}^*$. Each such directed network corresponds to a valid solution of the underlying problem and translates back into a combination of knowledge sources able to convert the initially given information into the sought target properties. Figure 1 illustrates a feasible solution involving six of the twelve knowledge sources. The six knowledge sources have to be arranged in a particular sequence to obtain the desired target properties, P_9 and P_{22}. Knowledge sources K_{12} and K_2 both take the two initially known properties, P_{17} and P_{18} as inputs. Using these inputs, knowledge source K_{12} generates properties P_{15} and P_{16}, and knowledge source K_2 outputs property P_{25}. Taking property P_{25} as input, knowledge source K_3 returns property P_{24}. Together, these output properties constitute the required input for knowledge sources K_4 and K_9, which in turn generate input properties P_{10}, P_{11}, and P_{12} for knowledge source K_5. In addition to generating property P_{12}, knowledge source K_4 also produces the first of the two target properties, P_{22}. Finally, knowledge source K_5 generates P_9, the seconds target property.

Time-Expanded Network Representation

In order to solve the CSI problem using an integer programming approach, this directed graph is augmented. The problem of identifying suitable combinations of knowledge sources has an underlying temporal dimension. This results from the fact that there are many time-induced precedence constraints that have to be satisfied. Without a means of capturing the sequence and timing in which knowledge sources are being used, it is impossible to enforce these implicit precedence constraints. Neglecting these constraints leads to infeasible solutions containing deadlocks. An example for such a deadlock is shown in Figure 2. At first the depicted solution seems to be feasible since there exists a path from the initially known properties, P_1 and P_2, to the target property P_9. However, knowledge source K_2 generates property P_5, a required input for knowledge source K_3 and knowledge

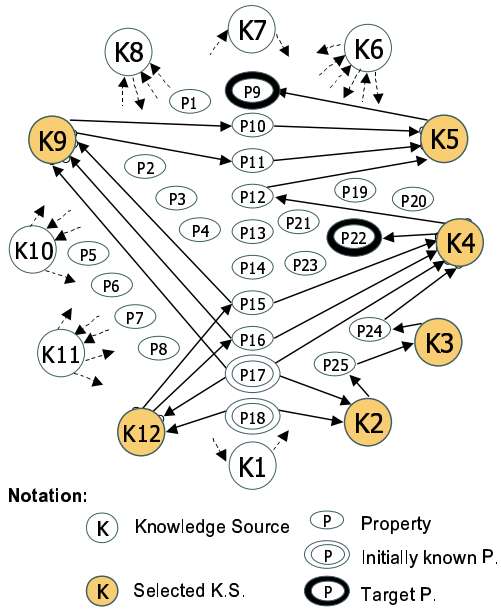


Figure 1. EXAMPLE PROBLEM - DIRECTED GRAPH OF KNOWLEDGE SOURCE AND PROPERTY NODES.

source K_3 generates property P_4, a required input property for knowledge source K_2. This clearly results in a deadlock, causing the solution to be infeasible. Another reason that calls for a temporal dimension is the fact that analytical knowledge sources are often used iteratively. CAM applications often consist of a small number of simulation models (knowledge sources) that are used in an iterative and/or alternating fashion until some convergence criterion is met.

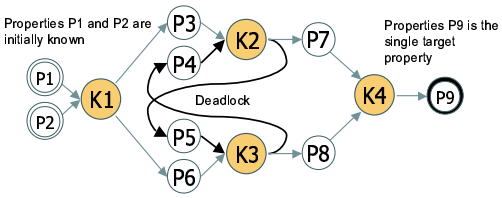


Figure 2. A DEADLOCK EXAMPLE.

One way of capturing the sequence in which knowledge sources are being used is to incorporate a temporal dimension. This can be achieved by developing an expanded network representation of the knowledge source and property nodes that is capable of accounting for the time-related precedence constraints described above. In order to construct such a time expanded network representation multiple copies of all knowledge source and property nodes are considered as shown in Fig. 3. Arcs that link the different copies describe temporal linkages in the system. Each node in the resulting network has a *time step* associated with it and hence, it is referred to as a time-expanded network representation. In the following discussion the CSI problem is defined mathematically and an IP-formulation that is based on this type of time-expanded network is presented.

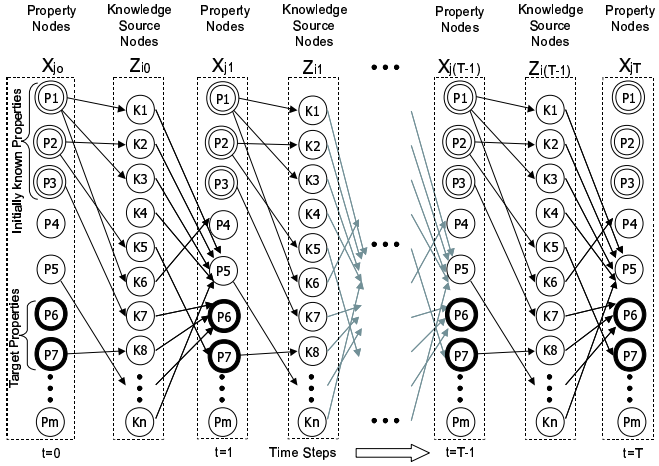


Figure 3. TIME-EXPANDED NETWORK.

IP-Formulation of the CSI Problem

The problem is modelled as a discretized time-expanded network consisting of two types of nodes, nodes representing the knowledge sources/utility components and nodes representing the properties. Due to the temporal dimension discussed above and depending on whether a particular knowledge source node can be used iteratively or not, the knowledge source library can include multiple copies of a particular knowledge source. In the integer programming formulation presented below, each of the copies of a knowledge source is then treated as a separate knowledge source. The maximum number of time steps that have to be considered is upper bounded by $T = n \cdot n_c$, where n_c is the maximum number of copies per knowledge source. This follows from the following consideration. It is easy to see that any solution to a CSI problem that has a time step in which no knowledge source is used has to be sub-optimal. If there is a feasible solution with such an *idle* time step, the solution can be improved (decrease the value of the objective function) by simply skipping the respective time step. Since the state of the system (known properties) does not change while performing a time step with no knowledge source activity, the feasibility of the solution will not be affected by skipping the *idle* time step. Hence, any optimal solution will use at least one knowledge source during any given time step. This allows to establish a bound on the maximum number of

discrete time steps of $T = n \cdot n_c$, the number of available knowledge sources (including copies). As a result, a problem with n knowledge sources and m properties will have a maximum of $O(n^2 n_c + n \cdot n_c \cdot m)$ nodes, consisting of $O(n(n \cdot n_c))$ knowledge source nodes and $O(n \cdot n_c \cdot m)$ property nodes. It should be emphasized that this is a conservative upper bound, that offers room for subsequent improvement.

To state the problem two sets of variables are introduced, Z_{it} and X_{jt}:

$$Z_{it} = \begin{cases} 1 & \text{if a request is sent to knowledge source } i \text{ at time } t, \\ 0 & \text{otherwise,} \end{cases}$$

$$X_{jt} = \begin{cases} 1 & \text{if property } j \text{ becomes known at time step } t, \\ 0 & \text{otherwise.} \end{cases}$$

It is assumed that there are *costs* associated with the consultation of each knowledge source. These *costs* can be associated with the actual costs of using a given knowledge source (e.g. proprietary knowledge sources) or with performance characteristics of a knowledge source, such as accuracy, computational complexity, or response time. The objective is to find the combination of knowledge sources and utility components that can generate the sought target properties, S^*, in the most desirable fashion. Depending on the particular cost criterion and depending on whether a single cost criteria or a weighted combinations of multiple cost criteria is used, most desirable can correspond to fast, very accurate, most 'affordable', or a combination of these criteria. In order to avoid any loss of generality and to keep the IP-formulation as flexible as possible, *costs* associated with the property nodes are considered as well.

The objective function reads

$$\min \sum_{t=0}^{T} \sum_{i=1}^{n} C_{it}^{K} Z_{it} + \sum_{t=0}^{T} \sum_{j=1}^{m} C_{jt}^{P} X_{jt}, \tag{1}$$

where C_{it}^{K} are the non-negative *costs* associated with knowledge source i and time step t, C_{jt}^{P} are the non-negative 'costs' associated with property j and time step t.

To ensure that the required target properties are computed, we need

$$\sum_{t=0}^{T} X_{jt} \geq 1 \ \ \forall j \in \mathbf{S}^*. \tag{2}$$

To model that the set of all properties in $\mathbf{S}_I$ are initially known, we require

$$X_{j0} = 1 \ \ \forall j \in \mathbf{S}_I, \tag{3}$$

$$X_{j0} = 0 \ \ \forall j \notin \mathbf{S}_I. \tag{4}$$

To ensure that all input properties for knowledge source i are available at time step t^* if knowledge source i is active at time t^*, we need

$$\sum_{t=0}^{t^*} X_{jt} \geq Z_{it^*} \ \ \forall t^*, \forall i \in \mathbf{K}, \forall j \in \mathbf{S}_i^{in}. \tag{5}$$

To guarantee consistency of the solution we need to enforce that property j is *known* at $t+1$, the time-step following the determination (calculation) of j by knowledge source i at time step t. Hence we need,

$$X_{j(t+1)} \geq Z_{it} \ \ \forall i \in \mathbf{K}, \forall t, \forall j \in \mathbf{S}_i^{out}. \tag{6}$$

The following constraints enforce that an X_{jt} can only be nonzero if at least one knowledge source with j as an output is nonzero at the same time step. In absence of these constraints, X_{jt}'s could be nonzero even if all Z_{it}'s would be zero and this would obviously result in an infeasible solution. Hence, we require

$$\sum_{i:i \to j} Z_{it} \geq X_{jt} \ \ \forall t, \forall j \in \mathbf{P}, \tag{7}$$

where $i : i \to j$ means that knowledge source i generates property j as one of its outputs.

Constraints of Equation 8 enforce that each knowledge source is used at most once throughout a particular solution and hence prevent the generation of sub-optimal solutions. As indicated above, knowledge sources might have multiple copies within $\mathbf{K}$ and each of these copies can be used exactly once. Hence, we add

$$\sum_{t=0}^{T} Z_{it} \leq 1 \ \forall i \in \mathbf{K}. \tag{8}$$

Note that even without these constraints the formulation is correct, however, they are included since they were found to lead to an improved performance of the branch and bound algorithm and to reduced solution times.

By definition we have

$$Z_{it} \in \{0,1\}, \tag{9}$$

$$X_{jt} \in \{0,1\}. \tag{10}$$

The integer programming formulation consists of objective function of Equation 1, constraints of Equations 2-8, and integrality requirements of Equations 9 and 10. Following the standard procedure of NP-completeness theory, it can be shown that the CSI Problem is NP-hard [28].

Computational Results for Randomly-Generated Test Networks

This section presents computational results obtained from solving randomly generated CSI problem instances using the time-indexed formulation presented above. All computations were carried out on a Windows 2000 Server PC (Pentium III) with a clock speed of 1Ghz. The IP-formulation is solved with the commercial solver CPLEX™, version 8.0. The PC was equipped with 512MB RAM.

To speed up the creation of random problem instances, a random network generation procedure was implemented. In addition to the desired size of the problem (number of property- and knowledge source nodes) the procedure takes several other parameters into account, including maximum and minimum number of input properties for a given knowledge source, maximum and minimum number of output properties, and a fixed range for the costs associated with the knowledge source nodes and the property nodes. In order to simulate the type of network expected to occur in manufacturing knowledge source selection problems more closely, the network generation procedure is able to generate and include different clusters of random knowledge sources. This mimics the fact that there are often sets of knowledge sources that *perform similar tasks*. In terms of the CSI network, *performing similar tasks* translates into taking and returning similar sets of input and output properties. A good example for such a cluster would be a set of knowledge sources that provide cutting force data, including mechanistic force models, empirical databases, and more advanced simulation tools such as a finite element model.

Figure 4-a shows a typical example network consisting of 70 nodes, including 50 knowledge source nodes and 20 property nodes. In this particular problem instance, two properties are initially known, $\mathbf{S}_I = \{P_4, P_9\}$ and the target property set, $\mathbf{S}^*$, consists of three unknowns, including P_8, P_{10}, and P_{11}. The resulting IP-formulation is comprised of 1,420 variables and 7,093 constraints. After 0.7 seconds of preprocessing, using various preprocessing techniques and the default settings of CPLEX™ 8.0, the IP has 5,746 rows, 1,227 columns, and 39,735 nonzeros. The problem was solved to optimality in 1 minute and 16 seconds. During this time, the branch and bound algorithm evaluated 512 nodes. A graphical representation of the optimal solution is shown in Fig. 4-b.

Many randomly generated problem instances of varying dimensions have successfully been solved. The experiments reveal that for instances with up to 50 knowledge sources and 20 property nodes an optimal solution can be found in less than 10 minutes in 90% of the cases. With an increasing number of knowledge source and property nodes this percentage value decreases rapidly and instances with 50 knowledge sources and 30 properties, for example, can only be solved in less than 10 minutes in approximately 20% of all cases. This trend is depicted in Figure 5. As for most NP-hard problems, the optimality gap for instances that cannot be solved to optimality within a given time limit can vary considerably and is a function of the size of the problem instance.

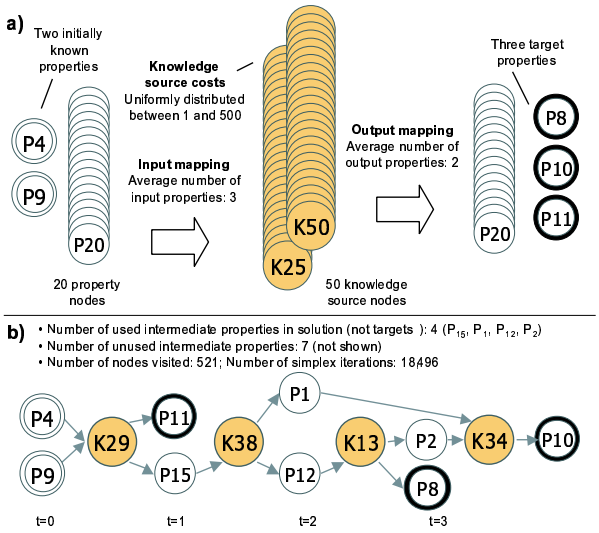


Figure 4. EXAMPLE 1: a) SCHEMATIC OF RANDOM NETWORK - b) OPTIMAL SOLUTION FOUND BY EXACT METHOD.

Since the decision version of the CSI problem is NP-complete, no polynomial time algorithm for the problem is likely to exist and the observed rapid increase in CPU time for problems with increasing dimensions is expected. Hence, future research efforts should be focused on the development of heuristic solution methods able to find good solutions. In a first attempt to tackle problem instances that cannot be solved to optimality by the exact method described above a simple search heuristic has been developed [29]. The search heuristic is based on the conversion of the original network into a tree-like network topology. This conversion can be performed in polynomial time ($O(n^2)$) and hence, can be applied to arbitrarily large problem instances. This search heuristic serves as a starting point for the ongoing research effort to develop more sophisticated heuristical approaches and also to improve the exact method introduced in this paper.

Figure 6 provides the results for a problem that was solved with both the IP-formulation and the search heuristic. This problem instance has the same characteristics as the previous example and consists of 50 knowledge source nodes and 20

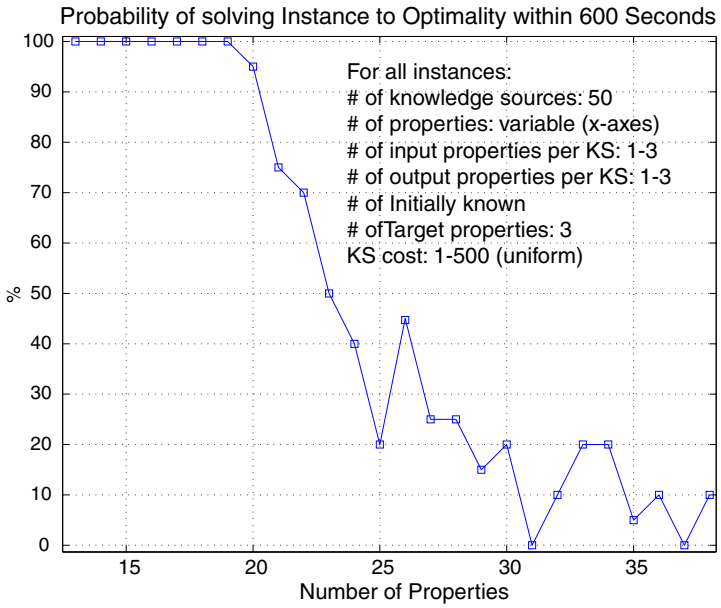


Figure 5. COMPUTATIONAL LIMIT OF EXACT SOLUTION PROCEDURE.

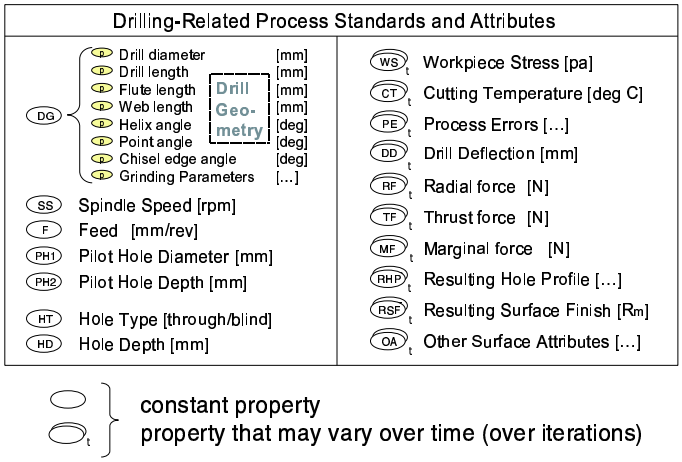
property nodes. As before, two properties are initially known, $\mathbf{S}_I = \{P_1, P_2\}$ and the target property set, $\mathbf{S}^*$, consists of three unknowns (P_6, P_8, and P_{20}). The best solution found by the exact method within a time limit of 600 seconds has a total cost of 130. For this particular problem the heuristic outperforms the exact method in terms of CPU time and is able to find an optimal solution with a total cost of 59 in 25 seconds. It is important to note that the exact solution method by definition will eventually arrive at the optimal solution as well. However, these results show that there is great potential for the development of heuristic methods, especially for problem instances with increased dimensions.

Manufacturing Example: Hole Quality Prediction and Optimization

This section presents an example of a typical CAM application built using the algorithmic strategies presented in this paper and the overall development environment discussed in [25]. The example illustrates the development of a tool for hole quality prediction in drilling operations. Hole quality can be defined in terms of attributes (properties), such as cylindricity and roundness and depends on a wide range of properties, including drill geometry, cutting conditions, workpiece material, etc. The hole quality attributes are functions of the hole profile, which in turn depends on the drill design, the process variables, and the overall characteristics of the drilling process.

The objective here is to demonstrate the structure of the problem formulation. Since we are now dealing with domain-specific knowledge sources, the details of which are beyond the scope of this paper, we make certain assumptions with respect to the costs and the input and output properties of the knowledge sources considered in this example.

Table 2. COMPLETE SET OF PROPERTY NODES FOR DRILL HOLE QUALITY PREDICTION.



Systematization of the Drilling Process Domain

In order to use the proposed approach for the development of CAM-applications for hole quality prediction, we have to provide a suitable systematization of the relevant domain knowledge that is able to integrate the different semantic mappings used by individual knowledge sources. Table 2 provides a top-level ontology for drilling operations and gives an overview of all relevant properties (attributes and parameters), including drill geometry, machining condition, and workpiece attributes. The table represents the complete list of properties considered in our example problem. Each property corresponds to a property node according to the definition presented in the previous sections. Properties in the left column of Tab. 2 are constant for a given drilling operation, whereas the properties in the right column may vary over the duration of the drilling operation.

Domain Knowledge and Component Libraries

Numerous attempts have been made to model the fundamental physics and the performance characteristics of drilling operations and researchers have developed a wide range of theoretical models to simulate the various issues of the drilling process, including cutting force models, cutting temperature models, drill wandering models, surface generation models, tool wear models, etc. [30, 31]. This has resulted in a large number of heterogenous knowledge sources and data representations, making hole quality prediction and optimization an interesting application for the automated development approach presented here. In addition to analytical models there are many other useful knowledge sources for drilling operations, such as drill geometry databases, cutting

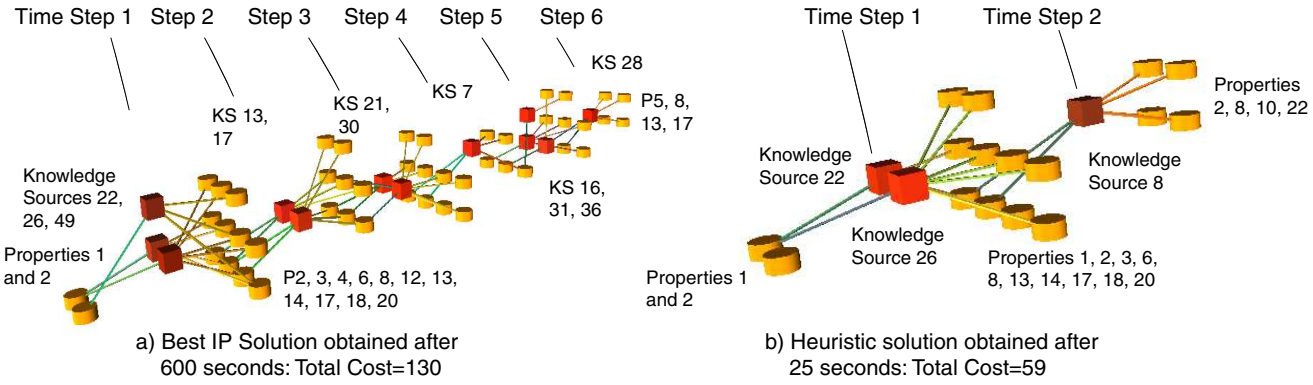


Figure 6. EXAMPLE 2: a) IP-SOLUTION - b) SOLUTION OBTAINED BY SEARCH HEURISTIC.

force databases, repositories of material calibration values, rule-based expert systems, etc. [9–11].

Figure 7 illustrate the knowledge source component library in form of a *matrix*. The purpose of this figure is to further illustrate that there are multiple candidates for a specific type of knowledge source. Each row of the matrix corresponds to one specific aspect of the drilling process. The first row, for example, contains knowledge sources able to predict drilling forces. Each knowledge source uses a different methodology and a unique set of input properties to predict the force data. In addition, each one offers different performance characteristics (*costs*) and sometimes slightly different output properties. We are considering two *cost* criteria in this example, *'computational complexity'* (C_{it}^{K1}) and *'potential for loss of accuracy'* (C_{it}^{K2}). We assume that the *costs* associated with the knowledge sources are not a function of time and hence, we have $C_{it}^{K} = C_{i}^{K}$ for every t. Both *cost* criteria have arbitrarily been given a range of 100. A value of 0 corresponds to an extremely favorable and a value of 100 to an extremely unfavorable *cost*.

The first cutting force knowledge source, for example, is based on a mechanistic model, with relatively low computational complexity ($C_{it}^{K1} = 25$) and a fairly good accuracy ($C_{it}^{K2} = 25$). The second cutting force knowledge source uses a FEM-based force model with a considerably higher computational complexity ($C_{it}^{K1} = 75$) and a decreased potential for loss of accuracy ($C_{it}^{K2} = 5$). Finally, the cutting force knowledge source shown as the last entry in the first row of the *matrix* represents a simple database of empirical cutting force data for drilling operations. The computational complexity of retrieving information from a database is low ($C_{it}^{K1} = 3$) but the risk of inaccurate predictions is much higher ($C_{it}^{K2} = 50$). The dots between the FEM-based force model and the force database knowledge source indicate that there are additional force-related knowledge sources in the knowledge source library that are not explicitly shown in the figure. The remaining rows of the matrix are structured similarly and contain the knowledge sources related to all other aspects of the drilling process, such as cutting temperature, drill wandering, hole surface generation, drill deflection, etc.

Having defined the property nodes in form of the top-level ontology (Table 2) and the knowledge source nodes in form of a component library (Fig. 7), we can now combine the two types of nodes and their relationships in a time-expanded network similar to the generic example presented earlier in Fig. 3.

IP-Formulation and Identification of a suitable Set of Knowledge Sources

In order to clearly state the problem we need to define $\mathbf{S}_I$, the set of initially know properties, and $\mathbf{S}^*$, the set of target properties. Initially known in our example are the process conditions (speed, feed, etc.), the drill geometry (diameter, web length, etc.), and the process errors. The set of target properties consists of the hole profiles at time steps $t = 1, 2, 3, ..., T$. Considering our top-level ontology (Table 2), and the knowledge source library (Fig. 7), the example includes 50 knowledge sources and 30 properties. We are consider a weighted average of two cost criteria, including computational complexity and loss of accuracy. The overall cost of a combination of knowledge sources is fully determined by the sum of the costs of the used knowledge sources and hence, we can associate zero cost with all property nodes (C_j^P's $= 0$). Therefore, we can neglect the second summation included in the general definition of the objective function (Equation 3). Taking these considerations into account the objective function reads

$$\min \sum_{t=0}^{T} \sum_{i=1}^{50} C_i^K Z_{it} + \underbrace{\sum_{t=0}^{T} \sum_{j=1}^{30} C_j^P X_{jt}}_{=0} = \min \sum_{t=0}^{T} \sum_{i=1}^{50} C_i^K Z_{it},$$

where C_i^K's represent an equally weighted combination of the

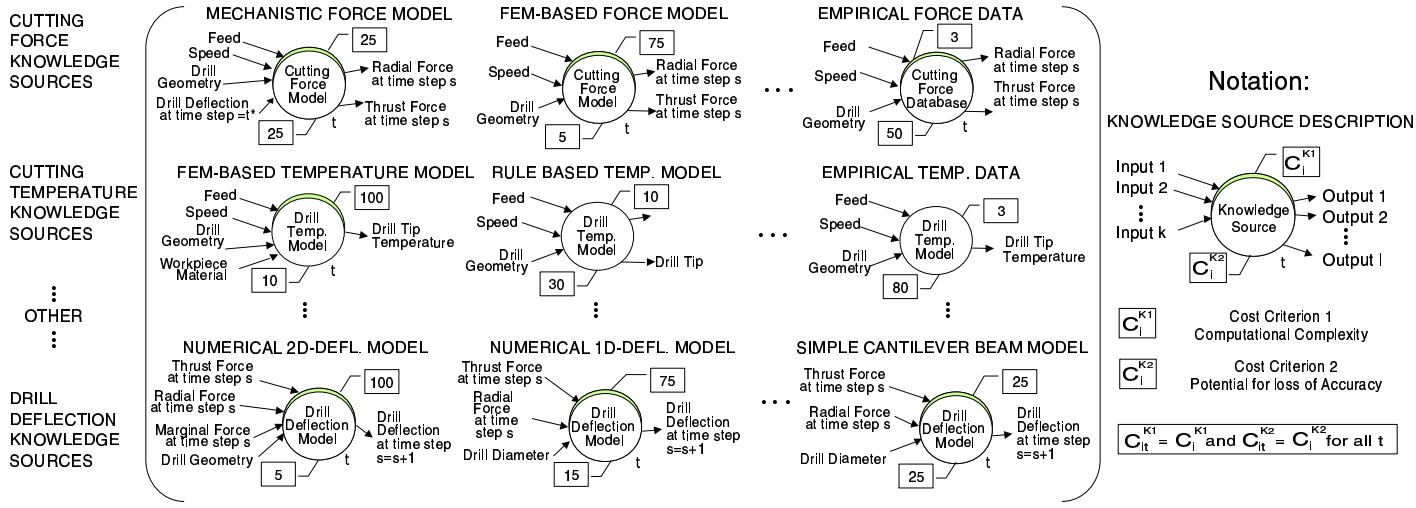


Figure 7. KNOWLEDGE SOURCE AND COMPONENT LIBRARY FOR DRILL HOLE QUALITY PREDICTION.

two cost criteria, *computational complexity* and *potential for loss of accuracy*. Since both cost criteria have the same value range (0-100), we can simply add the respective numbers for each knowledge source to obtain an equally weighted combination of the two criteria.

To ensure that the required target properties are computed, we need

$$\sum_{t=0}^{T} X_{jt} \geq 1 \;\; \forall j \in \mathbf{S}^* = \{\text{Hole profile at time steps } t = 1, 2, ..., T\},$$

Furthermore we require

$$X_{j0} = 1 \;\; \forall j \in \mathbf{S}_I = \{\text{feed, speed, drill geometry, process errors}\},$$
$$X_{j0} = 0 \;\; \forall j \notin \mathbf{S}_I.$$

After including the remaining six sets of constraints (Equations 5-10) we obtain an IP formulation of the problem with approximately 5,000 variables and 10,000 constraints.

Figure 8 sketches the solution that is obtained when solving the described IP-formulation to optimality, using the described algorithm and the assumptions discussed earlier. The particular *'path'* that is identified and depicted in the figure consists of five analytical models, including a wandering model, a cutting lip surface generation model, a cutting force model, a drill margin and hole wall interaction model, and a drill deflection model. The path clearly establishes all precedence relationships required to combine the five models. The first model that is being used is a wandering model that determines the deflection of the drill during the initial phase of the drilling operation. The deflection at time zero is then used as an input for a cutting lip surface generation model that can calculate the hole geometry at time zero. In a third step a mechanistic cutting force model and a drill margin and hole wall interaction model is used to calculate the cutting forces and to update the hole geometry at time zero. By combining the five different sub-models in the sequence indicated in Fig. 8, and by repeating the last three blocks of the solution iteratively (viz. time step 2, 3, and 4), it is possible to generate the desired hole quality attributes for t=1, 2, 3, ..., T.

Having used equal weights for the two cost criteria, the solution shown in Fig. 8 is a tradeoff between computational complexity (time required to predict hole quality) and accuracy of the prediction. We can obtain alternative solutions to the hole quality prediction problem by changing these weights. To illustrate this, we simply modify our objective. We now focus on finding a solution to the hole quality prediction problem quickly (80%) and we put a much smaller weight on accuracy (20%). When we resolve the problem with these modified weights, we obtain the solution shown in Fig. 9. The main differences are that the alternative solution uses a simple empirical cutting force database (CFD) instead of a mechanistic force model (MFM) and a simple cantilever beam model (CBM) instead of a two-dimensional numerical model (2DNM) to predict drill deflection. As indicated in Fig. 7, the empirical cutting force database has a much lower computational complexity than the mechanistic force model ($C^{K1}_{CFD} = 3$, $C^{K1}_{MFM} = 25$). Hence, if the main emphasis lies on computational complexity it makes sense to replaces the mechanistic force model by the empirical force database. The same holds true for the deflection models. The cantilever beam model has a much lower computational complexity than the two-dimensional numerical deflection model ($C^{K1}_{CBM} = 25$, $C^{K1}_{2DNM} = 100$, compare Fig. 7).

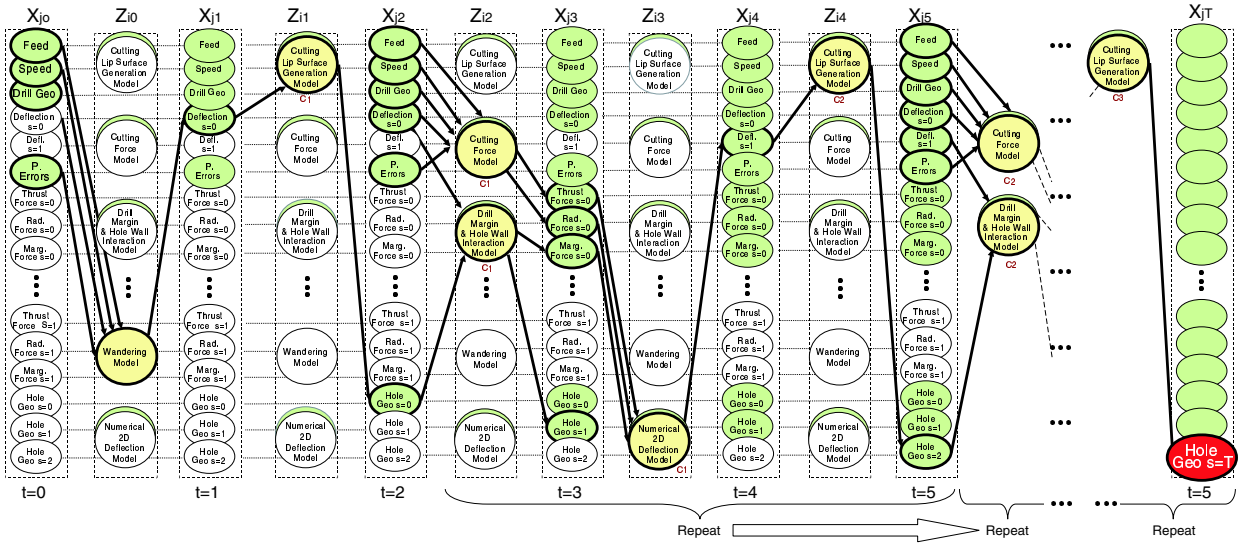


Figure 8. SOLUTION TO IP-FORMULATION - EQUAL EMPHASIS ON COMPUTATIONAL COMPLEXITY AND ACCURACY.

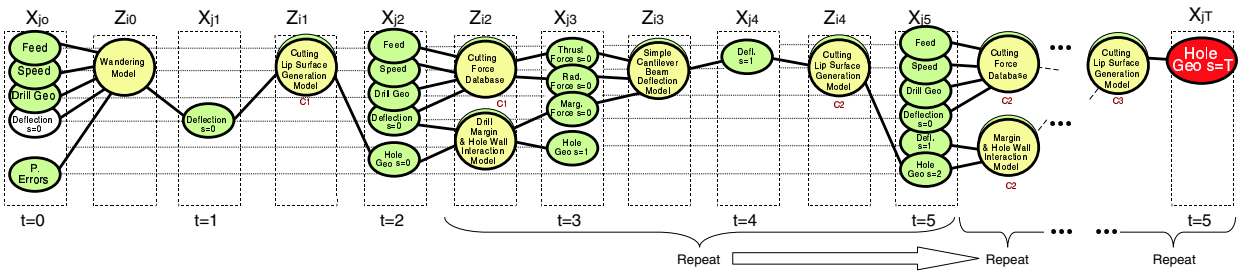


Figure 9. SOLUTION TO IP-FORMULATION - EMPHASIS ON COMPUTATIONAL COMPLEXITY.

Summary and Conclusions

This paper presented an algorithmic strategy required to implement a new paradigm for the development of a global network-based development environment in which researchers can generate advanced CAM software tools in a collaborative and partially automated fashion.

The specific conclusions from the work presented in this paper are:

1. The Component Set Identification (CSI) Problem has been identified as one of the key problems that needs to be addressed in order to automate the generation of CAM software tools and to bridge the gap between domain knowledge systematization and software integration efforts. This paper developed a mathematical characterization that allows to fully specify any instance of the CSI problem.
2. It has been shown that any instance of the CSI problem can be modeled as a directed graph, and subsequently be formulated as an integer program using a time-expanded network representation and a time-indexed IP-formulation.
3. Using randomly generated test networks and a Branch-and-Bound algorithm some initial computational results have been obtained. These results show that it is possible to solve many medium size problems (50 knowledge source and 20 property nodes) to optimality using the IP-formulation and the solution procedures discussed in this paper.
4. Based on both the computational results and the fact that the CSI problem is NP-hard, the need for polynomial-time search heuristics has been identified. Using an example problem and a basic search heuristic, it has been shown that alternative solution methods can be used to find good solutions for problem instances that cannot be solved to optimality by the exact method discussed in this paper.
5. Using hole quality prediction as an example we illustrated the practical relevance and the value of the proposed al-

gorithmic strategy. We presented a systematization of the drilling process domain, established a suitable knowledge source and component library, and formulated the problem according to the dynamic network representation developed in this paper. Two alternative solutions were developed by using two sets of weights for the two cost criteria considered in the example (computational complexity and potential for loss of accuracy).

Acknowledgements

The authors gratefully acknowledge funding of this project by the National Science Foundation under Grant NSF DMI-00-04226.

REFERENCES

[1] Yu, J., Lotfi , S., Ishii, K., and Trageser, A., 1993. "Process selection for the design of aluminum components". *ASME Computers in Engineering*, **1**, pp. 181–188.

[2] Smith, C., 1999. *The Manufacturing Advisory Service: Web Based Process and Material Selection*. PhD thesis, University of California, Berkeley.

[3] Wright, P., and Dornfeld, D., 1998. "Cybercut: A networked manufacturing services". *Transactions of NAMRC/SME*, **XXVI**, pp. 218–286.

[4] Stori, J., and King, C., 1999. "Integration of process simulation in machining parameter optimization". ASME J. Mfg. Sci. Engr., **121** (1), pp. 134–143.

[5] ElBaradie, M., 1997. "A fuzzy logic model for machining data selection". Int. J. Mach. Tools Manufact., **37** (9), pp. 1353–1372.

[6] Bless, P., 1999. A model-based machining advisor for cutting parameter selection in end-milling. Master's thesis, University of Illinois, Urbana-Champaign.

[7] Kline, W., DeVor, R., and Shareef, 1982. "The prediction of surface accuracy in end-milling". *Journal of Engineering for Industry - Transactions of the ASME*, **104**, pp. 272–278.

[8] Fu, H., DeVor, R., and Kapoor, S., 1984. "A mechanistic model for prediction of the force system in face-milling operations". *Journal of Engineering for Industry - Transactions of the ASME*, **106**, pp. 81–88.

[9] Metcut Research Associates Machining Data Handbook, Machinability Data Center, The Otto Zimmermann and Sons Company, Inc., Newport, Kentucky, 1990.

[10] Tool and Manufacturing Engineers Handbook Vol.1. Society of Manufacturing Engineers. Dearborn, Mich., 1983.

[11] Vitanov, V., 1995. "An expert system shell for the selection of metal cutting parameters". *Journal of Materials Processing Technology*, **55**, pp. 111–116.

[12] NIST, 1992. Product data representation and exchange. Tech. Rep. ISO 10303-1, National Institut of Standards.

[13] Lee, J., Gruninger, M., Jin, Y., Malone, T., Tate, A., and Yost, G., 1998. "The pif process interchange format and framework". *Knowledge Engineering Review*, **13**, pp. 91–120.

[14] Lubell, J., and Schlenhoff, C., 1999. "Process representation using architectual forms: Accentuating the positive". In Proceedings of the MarkUp Technologies Conference.

[15] Catron, B., and Ray, S., 1991. "Alps: A language for process specifi cation". *International Journal of Computer Integrated Manufacturing*, **4**, pp. 105–113.

[16] Fox, M., 1992. "The tove project: A common-sense model of the enterprise". In *Industrial and Engineering Applications of Artifi cial Intelligence and Expert Systems Lecture Notes in Artifi cial Intelligence Biometrical Gentics*, F.Belli and F. Radermacher, Eds. Springer-Verlag, Berlin, pp. 25–34.

[17] Uschold, M., King, M., Morales, S., and Zorgios, Y., 1998. "The enterprise ontology". *The Knowledge Engineering Review*, **13**.

[18] Dahl, O., Myrhaug, B., and Nygaard, K., 1970. Simula common base language. Tech. Rep. 22, Norwegian Computing Center.

[19] Cheng, B. H. C., and Jeng, J.-J., 1997. "Reusing analogous components". Knowledge and Data Engineering, **9** (2), pp. 341–349.

[20] Wooldridge, M., and Jennings, N., 1995. "Intelligent agents: Theory and practice". *Knowledge Engineering Review*, **10**, pp. 330–337.

[21] Intelligent Manufacturing System Group, www.ims.org.

[22] Object Management Group, www.omg.org.

[23] Virtual Factory Laboratory, Georgia Institute of Technology, http://www.isye.gatech.edu/vfl/.

[24] Berkeley Manufacturing Institute, University of California at Berkeley, http://www.me.berkeley.edu/.

[25] Bless, P., DeVor, R., and Kapoor, S., 2002. "An architecture for the development of multi-component software tools for virtual manufacturing". In Proceedings of ASME International Mechanical Engeneering Congress and Exposition.

[26] Hwang, F. K., Richards, D. S., and Winter, P., 1992. *The Steiner Tree Problem, Annals of Discrete Mathematics*. North-Holland.

[27] Kolisch, R., and Padman, R., 2001. "An integrated survey of deterministic project scheduling". *Omega*, **29 (3)**, pp. 249–272.

[28] Garey, M., and Johnson, D., 1979. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York.

[29] The Virtual Machine Tool Project, University of Illinois at Urbana-Champaign, Department of Mechanical and Industrial Engineering, www.knowledgesources.org.

[30] Gupta, K., Ozdoganlar, O., Kapoor, S., and DeVor, R., 2003. "Modeling and prediction of hole profi le in drilling, part 1: Modeling drill dynamics in the presence of drill alignment errors". *ASME Journal of Manufacturing Science and Engineering*, **125:1**, pp. 6–13.

[31] Gupta, K., Ozdoganlar, O., Kapoor, S., and DeVor, R., 2003. "Modeling and prediction of hole profi le in drilling, part 2: Modeling hole profi le". *ASME Journal of Manufacturing Science and Engineering*, **125:1**, pp. 14–20.