Architectural Issues in the Design and Implementation of an Integrated Toolkit for Manufacturing Engineering

Alan W. Brown*
Software Eng. Institute
Carnegie Mellon University
Pittsburgh, PA 15213
(awb@sei.cmu.edu)

Robert P. Judd
Dept. of Electrical Eng. and Computer Science.
Ohio University
Athens, OH 45701
(juddrp@bobcat.ent.ohiou.edu)

Frank Riddick
Manufacturing Eng. Laboratory
Nat. Inst. of Standards and Technology
Gaithersburg, MD 20899
(riddick@cme.nist.gov)

Abstract

This paper considers the integration of commercial off-the-shelf manufacturing engineering tools to produce a manufacturing engineering toolkit. These issues are considered by describing the work taking place in an on-going project to develop such a toolkit. In particular, the approach being taken toward integration of the tools is described, concentrating on the architecture of the solution that is being developed. To this end the paper highlights four main areas: the strengths and weaknesses of an initial prototype toolkit being used as input to the project, the approach being used to describe the architecture of the integrated toolkit, the details of the main elements of a preliminary architecture for the toolkit, and some issues being faced in moving from an architecture to a working implementation that can be used in practice. As a result, the main contributions of this paper include a survey of the key issues facing designers of integrated toolkits for the manufacturing domain, an analysis of architectural alternatives for building integrated toolkits, and an illustration of choices that have been made in this regard for satisfying the requirements for a manufacturing engineering toolkit.

* Address for all correspondence.

1 Introduction

Manufacturing engineering is the set of activities that, given the design for a particular physical part, specifies the manufacturing procedures and resources required to transform the design into a finished product. As with other engineering activities, manufacturing engineering involves a large number of tasks carried out by a variety of people. Computer support for many of these tasks is commonplace, and often essential to engineers in everything from product modeling and design through to production planning and scheduling. However, the use of a collection of computer-aided support tools throughout the manufacturing engineering life-cycle raises a new set of problems — how to use the tools collectively in as efficient and productive a way as possible across the whole product life-cycle.

Currently, engineering support tools are optimized for use in restricted, relatively narrow areas of the product life-cycle. For example, Computer-Aided Design (CAD) tools for part design provide essential services for the design and analysis of a particular part. Product design data is recorded persistently in some internal format based on the needs of each tool. Clearly, the data developed in each tool is an essential resource to an organization. In particular, it is a primary input to other areas of the engineering life-cycle such as production planning, where tasks such as materials ordering and machine tooling definition are carried out. However, to the CAD tool vendor such a need is secondary to the need for producing a tool with high performance, extensive design capabilities, and the ability to operate on a variety of commonly-available computer platforms.

Hence, to allow the various tools to interact a variety of common agreements, shared interfaces, and interchange standards must be in place. Particularly important are agreements concerning the data structures used to describe the part (e.g., its geometry, tolerance, electrical properties, and so on), the events that cause the part data to be changed (e.g., a fixturing error detected during manufacturing engineering), and the mechanisms used to communicate among the tools and provide feedback from those events to end-users. This set of inter-component agreements, together with descriptions of the major functional elements of the system and their interaction, is often called the *software architecture*[1] of the system [7].

As can be expected, the architecture of an integrated set of tools for manufacturing engineering will be realized by a set of tools interacting through additional software acquired or developed specifically for that purpose. This software provides an *infrastructure* that facilitates integration and evolution of the toolkit, improving adaptability of the toolkit by reducing the burden of replacing existing tools and of adding further tools to extend the functionality of the system.

The result of this effort is a *toolkit* that ties together the various vendor-supplied tools through the infrastructure software to produce a system that is able to operate cooperatively to carry out a range of manufacturing engineering tasks. This toolkit can then be applied in practice, where it will no doubt undergo enhancements and improvements to meet evolving end-user needs.

---

1. In the remainder of this paper the term "architecture" is used as a synonym for "software architecture".

Consequently, an organization developing an integrated set of capabilities for manufacturing engineering is concerned with producing the three major artifacts highlighted above: an *architecture* consisting of a set of inter-component agreements together with a description of the major functional components of the toolkit, an integration *infrastructure* that implements the integration between the different tools, and the *toolkit* itself. In this paper we concentrate on the first of these products — the architecture of a manufacturing engineering toolkit. The main contributions of this paper include a survey of the key issues affecting the architecture of integrated toolkits for the manufacturing domain, an analysis of architectural alternatives for building integrated toolkits, and an illustration of choices that have been made in this regard for satisfying the requirements for a manufacturing engineering toolkit.

The remainder of this paper is organized as follows. Section 2 provides background on the Manufacturing Engineering ToolKit (METK) project and its integration goals. Section 3 describes the initial baseline system in detail, and highlights the limitations of that system as they will be addressed in the METK. Section 4 describes the design approach being used to understand integration as it applies to this toolkit, and to produce the architecture of the toolkit. Section 5 describes the preliminary architecture of the METK that was developed based on the approach described earlier. Section 6 discusses a number of issues being faced in producing the toolkit, and to it being adopted in practice. Section 7 summarizes the main points of the paper, and outlines some of the remaining challenges to be addressed.

2 Background

Funded by the U.S. Navy, the Computer-Aided Manufacturing Engineering (CAME) program is aimed at lowering manufacturing costs, reducing delivery times, and improving product quality through the coordinated development and use of advanced software tools. In particular, the CAME program's vision of the future of manufacturing engineering is the availability of integrated computing environments for supporting manufacturing engineering.

Addressing integration of the various computer-based components is seen as the key to reducing costs and improving quality in the CAME program. That is, through the use of common databases of manufacturing engineering data, tools will be able to share information that is used throughout the manufacturing engineering life-cycle. This will reduce redundant data entry, ensure that different engineering functions are synchronized, and aid validation and verification of the engineering data.

Use of commercial tools in creating a manufacturing engineering toolkit is seen as an important aspect of the practical viability of the approach. Commercial tools offer significant functionality to end-users, and represent a large existing investment in many manufacturing organizations. While the CAME program recognizes that there is substantial investment required to develop and maintain interfaces among these commercial tools, the alternative of developing and maintaining a complete integrated toolkit from scratch is neither cost-effective nor desirable given current organizational investments and goals.

The CAME project is focusing on providing support for a wide variety of emerging tools and techniques for designing manufacturing processes, equipment, and enterprises, as well as tools for evaluating the producibility of product designs. To allow such services to work in a coordinated fashion, the emphasis in the CAME program is on developing an infrastructure that allows toolkits supporting these functions to be more easily developed. The major components of this infrastructure include:

- an integrated framework of support services such as data management and version control;
- an operating environment of hardware and software;
- common data definitions for shared data;
- interface standards to facilitate data interchange.

To reach this goal the CAME program has established a consortium of users, researchers, developers, and vendors to share their experiences and to cooperate in the solutions that are developed [1]. Together they are working on a collection of ideas that are embodied in infrastructure components such as system architectures, database schema designs, and specific techniques for integration of manufacturing engineering tools. Based on these artifacts a prototype implementation is being developed that can act as a testbed for many of the ideas, and can be used as a vehicle for demonstrating the advantages of these techniques. It is intended that this prototype be installed at various industry locations to engender further interest and acceptance of the ideas, and to encourage the take-up of the ideas by tool vendors and third-party integrators.

An important aspect of the CAME program is the creation of a toolkit that can be used to illustrate many of the issues being addressed. This toolkit, called the Manufacturing Engineering ToolKit (METK), is currently being developed at NIST and is the focal point for examining many of the architectural and infrastructure aspects of creating integrated toolkits. In the near-term it will provide concrete realization of many of the ideas of the CAME program.

An important by-product of the work is the identification and recommendation of standards for use in the manufacturing community. It is hoped that the CAME program can act as a focal point for establishing common interfaces, common data schemas, and other necessary integration components. Such standards will expedite the integration process, and will also improve flexibility and adaptability of integrated toolkits by enabling interchangeability of components supporting commonly defined interfaces. Working through the National Institute of Standards and Technology (NIST), with the direct support and participation of tool vendors, will help to facilitate this goal.

2.1 Requirements for an Integrated Manufacturing Environment

A goal of the CAME program is to develop an integrated environment to aid manufacturing engineers plan the production of parts and to verify the plans so that the parts are made correctly the first time. Currently, after a CAD model of a part has been created, much of the model information is manually input to a CAM package by manufacturing process experts. There are numerous problems with this approach that an integrated toolkit would address. These problems include the following:

- Standard manufacturing practices are hard to enforce. Different process planning experts use different rules for the same part and come up with different solutions. Each solution has a different impact on the quality and cost of the part.
- Good solutions are not formally documented.
- Proper selection and documentation of the tooling, inserts, tool holders and fixtures is a tedious process and prone to error.
- It is difficult to examine effects of alternative sequencing and setups.
- Its difficult to optimize the tool paths and tool changes.
- It is difficult to create accurate cost estimates.
- Most CAM systems do not have the ability to fully simulate machine tool motion to properly detect program errors. These problems are usually found only on the factory floor resulting in significant loss of time and increases in cost.

These problems are somewhat generic in nature. A more specific set of requirements was needed that could act as the driving force for the CAME program. To obtain these requirements engineers from several manufacturing organizations in government and industry were interviewed to discuss their needs, and later attended a workshop that discussed the scope of an integrated manufacturing engineering toolkit in general [1]. One of the first results of these interactions was to narrow down the scope of the work that would be considered in the METK to machining processes. This restriction provides a smaller, representative cross-section of problems to deal with since process planning of complex machined parts requires considerable expertise in understanding the features to be machined, planning the setups, determining appropriate fixtures and tooling, planning and sequencing the operations, creating the machine programs and costing the process.

Summarizing the needs expressed in those interactions, it was the consensus of these engineers that an integrated manufacturing engineering toolkit should perform the following functions:

- Examine a solid model of the part and determine the machinable features from it. This would eliminate the errors caused by omission of machinable features during process planning.
- Determine the minimum number of setups required for the part given the machine tool configuration.
- Select the appropriate machine that can achieve the desired results. This will eliminate the differences caused by experience and knowledge of the process planners.
- Determine the machining operations, the correct tools, cutting conditions, and the sequence in which these operations are performed. This will allow enforcement of standard machining practices and eliminate the errors caused by lack of experience, knowledge and omissions.
- Generate the cutter path and the correct G codes in order to eliminate the differences in numerical control (NC) programs due to differences in NC programmers' coding techniques and errors in data entry.
- Maintain a knowledge base of acceptable machining practices, cutting technologies, available machines, cutting tools, available fixtures, and their costs. This will result in easy enforcement of standard machining practices and easy updates and changes of machining practices.
- Store process planning solutions for parts. This will provide a history of how past parts were manufactured.

- Compare the predicted and actual tolerances, machine times, planning times, and costs. These comparisons can be used to refine the method, rules, and models contained in the toolkit.
- Allow manual editing at all stages of process planning. Allow deviation from standard practices whenever required.
- Provide cost estimates of the manufacturing process in order to allow the comparison of different plans and product designs.
- Provide a complete simulation of the manufacturing process. This will give complete NC program verification.
- Manage revisions and engineering change orders.
- Maintain the association of individual parts to a given product.

Furthermore, the outputs of an integrated toolkit should include:

- A list of all machining features, together with the estimated time and cost of each feature.
- The setups required for a part.
- The machine, fixtures, cutting tools, and tool holders required to manufacture a part.
- The sequence of operations to be performed.
- The NC programs and tool list for each setup.
- Simulation of the generated G codes to check for errors in the G codes, tool crashes etc.
- Cost estimates.
- Routings.

These lists of expected functions and outputs form a description of the basic requirements for the METK.

2.2 Summary

There is a wide range of requirements that can be considered applicable to an integrated manufacturing toolkit. In this section a number of them have been highlighted based on feedback from professionals in the manufacturing engineering domain. As a result, implementing a toolkit that addresses these requirements is a major undertaking requiring significant time and resources. Confronted with this problem, it is important to look for a starting point that will provide the maximum leverage.

Fortunately, such a starting point exists: an initial integrated manufacturing workbench that combines a number of commercial products to fulfill some of the requirements expressed in this section. In the following section this workbench is described, concentrating on providing an analysis of its major strengths and weaknesses.

3 The Initial Foundation for the METK

Over the past few years an Intelligent Machining Workstation (IMW) system has been developed at Ohio University [11], funded under the Ohio Aerospace Institute's Core Research Program. The IMW integrates a number of commercial manufacturing tools to aid the process planning of a part. In the development of the METK it seems appropriate that the IMW be examined closely, with the aim of using it as an important component of the baseline

METK. In this section the IMW is described in terms of its architecture, and its major strengths and weaknesses as they relate to the requirements for the METK.

3.1 Background to IMW

The past decade has seen a significant effort to automate process planning. A good summary of the issues concerning process planning is presented by [8]. Various early generative planners include CPPP[6], GARI [5], XPLANE [17], TURBOCAPP [18]. These planners captured knowledge of expert machinists, however, much of the geometry input had to be manually entered and the output manually transferred to CAM systems. METCAPP(tm) is an excellent commercial tool based on this technology. Current CAD technology is moving towards solid models. The primary representation schemes for solid models include boundary representation (B-rep) and constructive solid geometry (CSG). Researchers [10] have shown that significant reasoning about how to machine a part can be done with B-rep CAD models. Newer systems such as the Quick Turn-around Cell [8], the Automated Part Programming System [13] and Machinist [9] use solid models with feature reasoning to electronically extract data from the CAD models. They also integrate CAM software to automatically generate tool paths. However, none of these planners include support for many requirements of an integrated environment, for example in areas such as process simulation and verification, cost estimation, and version control.

3.2 Description of the IMW

Addressing the limitations with existing process planning tools, the approach in developing the IMW was to integrate the best commercially-available tools onto a single platform to form a powerful concurrent engineering environment for design and manufacturing engineers. The integration approach involved developing filters to provide point-to-point integration of tools. This approach solves the problems of integrating specific tools, but with the limitation that is not extensible to other tools and is sensitive to updates of the underlying software modules.

The IMW uses ICEM Technology's PART(tm) as the process planning package, Cognition's Cost Advantage(tm) as the costing package, and Deneb's Virtual NC(tm) as the CNC simulation package. The Mechanical Advantage CAD system is used to generate the solid models for the IMW. However, other CAD tools can be used. The machine program produced by the IMW can be downloaded to any CNC machine. The Oracle(tm) database is used as a central data repository which all tools can access.

In Figure 1 the IMW's tool components and their primary data flows are illustrated. The primary input to the system is the solid model of the part and the blank. The primary outputs are the machine programs, cost estimates, process simulations, and setup descriptions. At installation, the IMW must be configured with appropriate cost models, machine models, machining practices, available tooling, and available fixtures. This information must be maintained as conditions warrant. The remaining portion of this section will briefly describe each of the tools in the IMW and the interfaces among them.

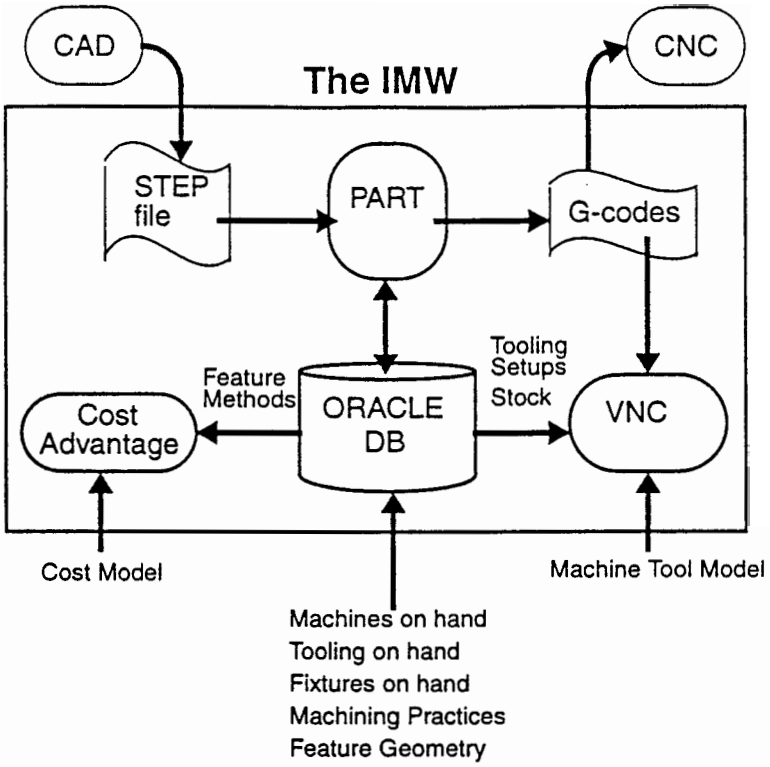

Figure 1: Tool Components and their Data Flows in the IMW

3.2.1 PART

PART uses the Oracle relational database to store both the expert knowledge required for process planning and the information generated during a process planning session. A solid model in the form of a STEP file[2] (other formats are also supported) of the part and blank are input to the tool. PART identifies the machining features, creates proper setups, determines the machining operations, selects the appropriate tooling, and generates the tool paths. Any information regarding the product, its features, the machining operations and the tools can be extracted from the database using Structured Query Language (SQL) queries. PART's knowledge base contains information about the machine tools, cutters, adapters, machine holders, tools assemblies, tool sets, cutting technology, machining practices, and product materials. The information can be easily edited and updated.

3.2.2 Cost Advantage

Cost Advantage is used as the cost estimator. One module of Cost Advantage, called the Cost Modeler(tm), is used to create a cost model for a given manufacturing environment. This cost model contains equations and design rules

2. A file containing a solid model representation conforming to the ISO 10303 standard called STEP (STandard for the Exchange of Product model data), which is an international standard for the computer-interpretable representation and exchange of product data.

needed to calculate a cost estimate. Cost Advantage then uses this model along with data specific to a particular part to calculate an estimate of the cost of that part. The estimate is stored in a flat file called the cost note.

3.2.3 Virtual NC

The Virtual NC (VNC) provides a simulation environment capable of emulating an entire machine tool and its controller. VNC provides the tools needed to create a workcell model capable of emulating any machine tool. The VNC workcell model reads the actual machine programs (G-codes) and drives an accurate kinematic model of the machine tool. Solid models of the cutting tools are geometrically subtracted from a solid model of the blank, simulating the cutting process. VC program prove-out can be done in this simulated environment, thereby saving valuable machine time and potentially disastrous tool crashes. In addition to emulation of machine tools, information pertaining to tool usage, collisions, and the volume of material removed can be determined.

3.2.4 Cost Advantage Interface

This interface is designed to be invoked at any time during a typical planning session. When invoked, it examines the Oracle database and extracts all available data to develop a cost estimate. As more information is developed by the process planner, more accurate cost estimates are developed. Typically the interface is called twice. The first invocation occurs when the machining features are known but no process planning has been performed. The interface uses the feature information to develop a rough estimate of the product cost. The second invocation occurs when information is available about all the operations and tools for each operation for each machining feature. This generates a very accurate estimate of the product cost.

The interface implementation uses SQL calls to extract feature parameters and tolerances, machine tool selection, cutting tools, required operations, and times from the database. This information is formatted into a cost note. When the cost note is read by Cost Advantage, all the appropriate calculations are performed to generate an estimate. The user can adjust any of the information in the cost note. Also there are provisions to add costs of the processes not considered by PART, such as coatings, heat treating, stress relief, and inspections.

3.2.5 Virtual NC Interface

This interface is invoked after a machine program(s) has been created by PART. The first action performed when the interface is invoked is to create all the necessary directories and subdirectories needed by a VNC workcell model. An SQL query of the database is performed to determine the proper machine tool for each setup. The kinematic model of the machine tools is copied to appropriate directories. Next, another query is performed to determine the cutting tools to be used. The interface uses the dimensions of the tools stored in the database to create the required solid models of these tools. The interface then stores the tool models in appropriate directories. Next the interface asks the user where to place the tools on the machine. With this information, the interface attaches the solid models of the tools to the correct locations of the machine model. Solid models of the blank and fixtures, the offset files, and the machine program are copied to the appropriate directories. The user is asked to confirm the location of the part, fixtures, and offset data. Finally, the interface loads the fixtures and blank onto the

machine tool model and the fully loaded workcell is saved to disk. The user can then execute the machine model for each setup and verify the process plans.

3.3 Using the IMW

In a typical design cycle engineers create a solid model of the proposed part and the blank it is to be machined from on a CAD system. A bounding representation description of the part and blank in a STEP file is used as the basic input to the IMW.

Once the models have been imported into PART, the user enters the appropriate tolerances and surface finish for the part. These can be general tolerances for the whole model or specific tolerances on a face of any feature in the model. PART supports standard methods, such as geometric tolerancing, to enter this information. After the tolerances have been defined, a product is defined consisting of the desired part with tolerances defined, the CAD model of the blank, and the desired material.

The process planning performed by the IMW relies upon manufacturing features as the primary description of the product. A manufacturing feature is defined to be a volume that should be removed by a series of machining operations. PART defines two types of features:

- Atomic features are elementary shapes that can be described using a predefined fixed set of parameters;
- Compound features group horizontally touching, intersecting, or partially overlapping atomic features.

PART reads the geometry files of the part and blank and automatically identifies all the atomic and compound features. The parametric information about these features is stored in the database. Finally, PART determines and stores all possible machining directions for the features. However, PART allows the user to edit the results.

At this point, the user can obtain a rough cut estimate of the cost of the part. By invoking the Cost Advantage translator, all the pertinent information about the features in the database is collected into a cost note for Cost Advantage. In the cost note, all geometrically identical features are grouped together as a single feature class plus a replication factor. The cost model in Cost Advantage combines this parametric data with a simplistic machining knowledge base to estimate the cost of each feature class. The user can then enter other cost items such as finishing, heat treating, inspection, and overhead to the cost note. An analysis of the cost note can be performed and recommendations made that will reduce the costs. The rough cost estimate can also be used to generate quotes to produce the part, as required. When the design and manufacturing engineers are satisfied with the rough cost estimate, detail process planning can begin.

The process planner now starts the development of the detailed operations plan for the part. The first step in this stage is to determine the required setups. PART calculates the setups based on the machine axis configuration (e.g. the degrees of freedom, whether the degree of freedom is in table or tool head, and the accuracy of each axes). PART identifies the minimum number of setups required to machine all the features and associate each feature with a proposed setup. PART selects the best direction to machine each feature to minimize the number of setups.

Additional setups may be generated if the accuracy of the machine cannot produce the tolerance required for all the features.

Next, the appropriate machine tool and cutting tools for the part is selected. PART selects the first machine tool in the database that has the proper axes configuration, can handle the size and weight of the blank, and has the required accuracy. Once the machine tool has been selected, a tool set with all the defined tools for that machine tool is selected. The machine and its tool set can also be manually overridden by the user based on availability of the machines. The user then places the fixtures to be used. The fixtures are selected from the set of all available fixtures stored in the database.

Now the machining operations required to create each feature can be determined. PART has a knowledge base of methods that describe how to create each feature. More than one method can be defined for a given feature. These methods will be selected based upon a priority system. For example, there are methods to ream, bore and drill a hole. The drill method is assigned the highest priority since it is the least expensive operation. For an elementary method to be selected a set of well defined conditions have to be satisfied. So if the position tolerance on a hole is too tight to be attained by drilling, then the next highest priority method is selected, say boring. Once a method is found that satisfies all the conditions, the method will determine the needed tooling for the operation and any pre-existing geometry that is required. For example, to drill a large hole, a pilot hole is required. The pre-existing geometry is then submitted to PART to be solved. Methods are defined for both atomic and compound features. Methods for compound features can be used to optimize the machining operations needed to create the complex features. Next, PART gathers together all the required operations and determines their optimal sequence to minimize the tool changes, table rotations and the total tool path length. The user can edit the sequence of operations or the tools to be used.

PART has a knowledge base of the cutting technology based on the cutting operation, cutting tool material and the part material. Based on this knowledge the cutting conditions (speeds, feeds, and depth of cut) are determined for each operation. The effect of tool deflection on the tolerance of the feature is also used in these calculations. At this point the cutter path is determined and is described using the Automatically Programmed Tool (APT) format. PART shows these tool paths graphically. Finally, PART uses the post processor of the selected machine to generate the machine programs from the APT files. It also gives as output a list of tools to be used, their position on the tool holder, setup sketches and list of operations.

At this point, a complete operations plan has been generated for the part. The machine tool, cutting tools and machine times are precisely known and stored in the database. The user can now augment the cost note in Cost Advantage with this information. This produces the second, more accurate cost estimate of the part, with a breakdown in terms of feature classes.

The final step is to verify the machine programs. The Virtual NC interface extracts the machine tool, cutting tools, raw stock, and fixture information from the database. From this information the translator automatically constructs

a simulated workcell that can produce the part. The workcell consists of an accurate kinematic model of the machine tool, the blank fixtured to the machine table, geometric models of all the tooling mounted on the machine, and the appropriate CNC program for the part. Virtual NC simulates the machining process using these models. VNC will identify any errors in the machine programs, tool crashes and part gouges. If errors exist, appropriate editing can be performed in PART.

3.4 Limitations of the IMW

The IMW has been a success. It has created plans for several actual industry-designed parts. However, there are several limitations:

- The filters used have been designed for the specific software tools. The filters would need to be completely reconstructed if other tools are to be used in the IMW.
- There is no central repository for the information; each tool maintains its own data. This makes version control and archiving all the data for a given product difficult.
- There is no provision, except for using a naming/directory convention, for linking the data of all the individual parts that comprise an assembly.
- PART only generates plans for prismatic parts. There is no support for turned parts in the IMW.
- Because the cost filter is uni-directional any data added to a cost model will be lost the next time the cost translator is called.
- PART and VNC do not have a common CAD format to store part and fixture models. Nor are there translators available to convert between the supported CAD formats. Therefore, multiple models of the parts and fixtures need to be created, stored, and maintained for the two tools.

3.5 Summary

The IMW provides a substantial baseline for the development of the METK. In particular, the IMW has addressed a number of the problems of integrating manufacturing tools and developed a point-to-point approach to creating interfaces and translation mechanisms between its components.

However, the IMW has a number of shortcomings. Most relevant to the METK is that the IMW solution is very specific to the tools being integrated, and does not provide any integration infrastructure that is independent of those particular tools. As a result, adding new tools, or replacement of existing tools with new ones, involves a significant amount of work on behalf of the system integrators.

The METK will attempt to build upon the lessons learned from IMW and address a number of issues with respect to developing an integration infrastructure for manufacturing engineering tools. Such an infrastructure can then be used as the basis for identifying and defining standards that can make integration of manufacturing engineering tools more efficient and effective.

Similarly, the architecture of the IMW is closed in the sense that it was designed to ensure that the specific tools selected for the IMW can be made to operate together in the most pragmatic way. As a result, the IMW architecture

does not facilitate addition of new capabilities, or the use of the existing capabilities in new ways. This openness to supporting multiple tools and manufacturing processes is a key requirement for the METK.

In the next section the design approach that will be used for the development of the METK architecture will be described to provide a basis for understanding the details of the architecture.

4 High Level Design Approach

In developing integrated solutions for the manufacturing engineering community, it is important to establish an integration approach that is understandable, repeatable, and measurable. In this way the specific integrations that are attempted by the CAME program can be repeated and improved upon in subsequent work elsewhere.

Hence, in this section previous and current approaches to tool integration are examined, and based on these examples the approach to tool integration that will be applied in the CAME program is developed. This integration approach will be especially important in describing the METK architecture as it will guide the selection of integration services and mechanisms.

4.1 Previous Approaches to Tool Integration

Following is a brief examination of some common architectural approaches toward tool integration. This will give a flavor of the current state-of-the-practice in the area. Further details of tool integration approaches can be found elsewhere [4].

The obvious baseline, seen in most manufacturing enterprises, is that no integration is in place. The tools are acquired individually, and no explicit attempts have been made to integrate them. This is typical when different departments in the same organization have purchased tools, when managers and engineers have separate environments, or when different projects assemble their environments with little or no concern for others. Inter-operation of the tools takes place primarily through manual efforts -- frequent re-entry of data, use of hard copies of designs and process descriptions, and so on.

In some cases large organizations have spent many millions of dollars developing their own in-house toolkit environments. As well as the development cost they have the on-going burden of maintenance and upgrade of the toolkit. While this approach offers greater control of the environment, many companies are regretting this choice as it diverts large amounts of resources from the main activities of the organization (manufacturing, telecommunications, or whatever).

Recognizing the need expressed by customers, some vendors are beginning to cooperate with others to develop collections of tools that are more easily used together. The coalition approach is an attempt by tool vendors to react to user needs by forming specific alliances with other tool vendors to address a perceived market. Typically the vendors amend their tools to make them work together in a more consistent way, or develop specific filters and

translators to convert data from one format to another. The interfaces and protocols they develop may or may not be made public (usually depending on the size and economic importance of the customer requesting them!).

An approach that is beginning to reach operational practice is the use of a framework product to ease integration. Two approaches are evident — a database approach as exemplified by use of relational and object-oriented databases, and a service-oriented message passing approach as seen in the SoftBench(tm) and ToolTalk(tm) products, and in systems based on the Common Object Request Broker Architecture (CORBA).

The use of these framework products, together with additional standards and interchange formats to agree on the semantics of the data and events being shared between tools, is an attempt to factor out common services and data used by the tools and treat them as a separately maintained resource. The potential rewards of this approach are huge. However, it often requires a significant level of agreement between a collection of vendors, integrators, and end users - agreements that are difficult to make and reduce flexibility.

In practice, attempts at developing a tool kit environment will be a hybrid of these approaches. Pragmatic reasons dictate that this is so. For example, in buying tools and framework products from different vendors there are inevitably different ways that will be more appropriate for connecting them. Legacy tools and data, and the immaturity of much of the technology, are two more reasons that contribute to this hybrid approach. It seems as though this will be true for at least the next few years.

Unfortunately, while useful progress has been made, there is still a significant way to go. In particular, the state-of-the-practice as it is commonly found is not very advanced. Point-to-point integrations of tools and the use of scripts and filters still predominate. These ad hoc approaches are allowing people to develop and maintain quite large and complex systems, although often at great cost. The next generation of technology that will help to relieve the burden of development has been much talked about, but has yet to see serious use. Initiatives such as the CAME program provide a significant opportunity to investigate these technologies in more detail.

4.2 Integration as a Design Approach

Integrating a set of tools can best be considered as a design or engineering activity. Viewed in this way, tool integration takes place by:

- designing a toolkit environment with integration in mind;
- matching integration aspects to the needs of an organization;
- making trade-off among costs and benefits of different integration strategies that could be adopted.

This leads to considering a method for tool integration. A major element of this method is matching the technology to the needs of an organization as expressed in the understanding of their end-user processes. In particular, an organization will have a goal for their integrated solution that must be met in order for the integration to be deemed successful. Then, decisions that are made in terms of the approach to integration can be considered in the light of achieving this goal.

How should the process of examining the different aspects of the tool integration problem begin? A useful approach is to separate issues on three different levels: process, end-user service, and mechanism.

The process level consists of the procedures and policies of an organization that must be supported by the integrated environment. It is these process elements that provide the context (i.e., the requirements and constraints) that determine what functionality is required of the integrated tools, and what coordination will be required between the tools.

The services level provides a means of describing the main functional components of the integrated tools and the coordination policies that must be supported. This allows for discussion of the conceptual integration of the functionality in support of an organization's process needs. This intermediate level provides a buffer that bridges the gap between the detailed mechanistic aspects of the components, and the high level needs and constraints of users.

The mechanism level is concerned with the physical characteristics of the available technology— the internal structure of the tools, the peculiarities of specific framework and infrastructure technology, the ad hoc "glue" that must be developed, and so on. This level is concerned with making components work together to provide the required functionality in an efficient and effective way.

One way in which this distinction between process, service, and mechanism is useful is in considering who "owns" the integration problem. Different aspects of the problem can be considered and evaluated based on where the primary responsibility for that aspect lies.

At the process level technology consumers have the major role. End-user organizations have in depth knowledge of their processes and practices, and provide the criteria for establishing the success or failure of an integrated toolkit environment.

At the services level the technology integrators usually hold the key. They will be familiar with the range of technology components available in the marketplace, and have a general understanding of an organization's process needs. Matching the technology to those needs is their main goal.

At the mechanism level technology producers such as tool vendors have a deep understanding of the inner workings of their tools, their relationship with framework products, and so on. Wrapping and adapting tools to make them more usable, or easier to integrate, can typically best be handled by specialist system integrators with direct support from the tool vendors themselves.

Identifying and describing these three levels of integration is important because it leads to a method for tool integration based on describing the required system at each of these three levels. These descriptions provide documentation which is vital for the validation and evolution of the system. In particular, the decision points concerning what needs to be done, and how it will be achieved, will inevitably need to be revisited. This leads to an interactive process where many steps in this method will be revisited and decisions will be changed. By documenting this process there is the raw material to allow such controlled change to take place, and for decisions concerning integration of tools to be made within the context of the overall needs, constraints, and goals of an organization.

4.3 Summary

In this section the current mechanisms available for integrating tools and the approach of considering integration as a design activity have been described. Combining these ideas produces a method for designing a toolkit environment to achieve an integrated result. This approach follows the classic design life-cycle of modeling the current physical situation, abstracting the current logical situation, devising the future logical situation, and then implementing that in a future physical situation. At each of these steps there are products to produce, and methods and techniques that facilitate the production of those products.

The CAME program will build on these design ideas in the development of the METK. In particular, the architecture of the METK will be particularly concerned with ensuring:

- process aspects of the METK are assigned to a separate component of the toolkit to allow different manufacturing engineering processes to be supported by the METK;
- functional capabilities of the METK are designed as a set of services with well-defined abstract interfaces;
- mechanisms used for implementing the integration infrastructure of the toolkit are sufficiently flexible and open to allow the addition of new functional components as the METK evolves.

5 The Architecture of the METK

The METK architecture describes how information models, validation methods, and Commercial Off-the-Shelf (COTS) applications are to be integrated to provide a system for the development of machined parts using valid manufacturing engineering practices. In this section a description of the architecture is provided, followed by more detailed discussion of some of the major elements of that architecture.

5.1 An Overview of the Architecture of the METK

An architecture has been defined for the METK. This has been produced as a result of examining the design and implementation of the IMW, and developing an enhanced architecture based on the requirements for a manufacturing engineering toolkit expressed at the CAME workshops. The basic components of the METK architecture are illustrated in Figure 2 and briefly described below. Further details of the major components are provided in the sub-sections that follow.

Figure 2 illustrates that there are the three primary sources of integration in the METK:

- The *Workflow and Product Data Manager* — responsible for coordination of all actions with the METK, and for the management of multiple shared versions of product and process data.
- The *Application Programming Interfaces* (APIs) — a library of operations provided by the COTS tools to allow access to the major services that will be provided by the METK.
- The *Common Data Repository* — long-term persistent data that can be archived and used for re-creation of manufacturing data concerning a part at some future date.

The METK architecture is discussed by considering a typical operation using the METK. End-users of the METK initially interact with a workflow and product data manager. This is responsible for maintaining business workflow

models of the manufacturing engineering process being enacted by an organization, and for controlling multiple versions of the major product data items. Activities being carried out by end-users will be tasks that are represented in a workflow model within the workflow and product data manager. Additionally, this component records information on the product design and quality plans. Hence, it acts as the major coordinating agent for all actions that take place in the METK.

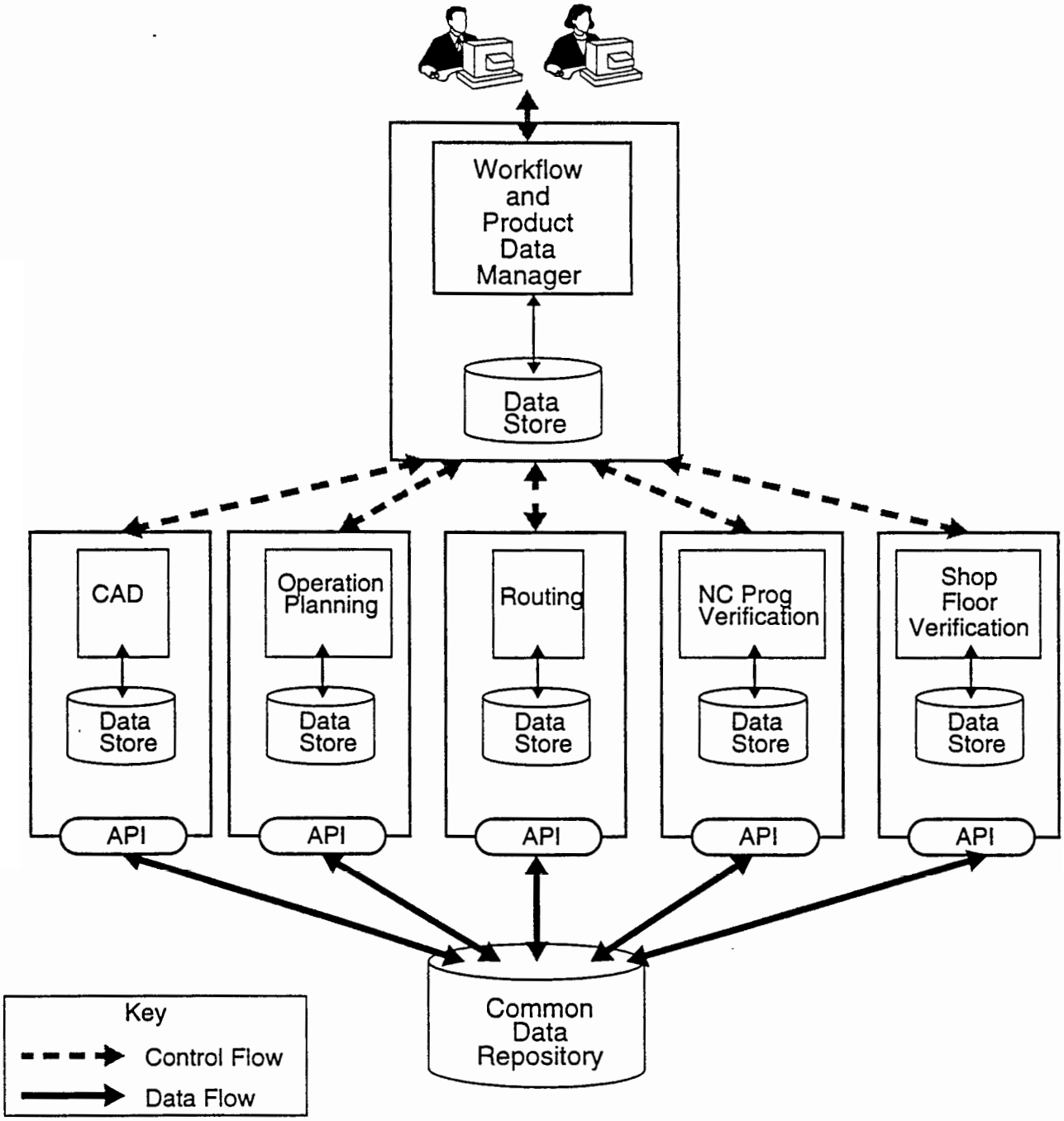

Figure 2: The Main Components of the METK Architecture

Typically, an end-user working within the workflow and product data manager will request that some action takes place (e.g., generation of a tool list, update to a routing sheet, etc.). The effect of this will be that the appropriate

service is requested by the workflow and product data manager through the application programming interface (API) of the tool that provides that service. An API is essentially a library of operations provided by the tool vendor to allow external access to internal tool functionality. To integrate tools it is often necessary to write additional code that converts between each tool's internal data format, make calls to operations with appropriate parameter information, and so on. Essentially much of the complexity of building a toolkit is embedded in this code. Hence, understanding each tool's API and writing appropriate integration code will be a primary focus of the METK developers.

In making a request via a API, the appropriate tool for that API will need to be executing. At this point the workflow and product data manager must make a copy of the appropriate version of the product data from its own data store into the specific data store for that tool. In this way the METK can resolve the problem of ensuring the correct version of product data is being used. If user interaction is required to carry out the task, the end-user is then positioned within the executing tool and can interact with the tool.

Once the task is completed the end-user will exit from the tool. At this point the workflow and product data manager will again take control and make copies of any amended data in its own data store. Appropriate version control principles will be applied to ensure that the new data becomes the latest version. The end-user will then be positioned within the workflow and product data manager and can execute further tasks.

The Common Data Repository provides persistent data storage and common access capabilities for reference information relating to manufacturing. Reference information defines the characteristics of the tools, machines, and materials which will be used in the manufacturing process. The common interface for accessing reference information will allow all of the tools which are a part of the METK to base their operation on a common set of data, which will increase the consistency of, and aid in verifying, the correctness of the information produced by each tool. Manufacturing information other than reference data is stored by the Workflow and Product Data Manager. It is responsible for maintaining and providing access to product and process data relating to the manufacturing of specific products.

The remainder of this section describes the major components of the METK in greater detail. However, while both the information models for manufacturing data and validation methods may be referenced in the discussion of the elements of the architecture, specific details about these components are beyond the scope of this paper. The components of METK which will be discussed are: Workflow and Product Data Management, Data Management, Application Interfaces, COTS Computer-Aided Manufacturing (CAM) tools, the Reporting and Maintenance module, and the hardware platform and system software.

5.2 Workflow and Product Data Management

The Workflow and Product Data Manager will act as a central point of control for the development of manufacturing information for the production of a machined part. It will provide mechanisms for:

- the definition, configuration management, and version control of the manufacturing engineering data for a part that is to be produced;

- the definition of a workflow plan for the development of the manufacturing engineering data for a part. The workflow plan defines all of the processes that are needed to move a engineering data from the conceptual stage until the data has been verified as ready to release to production. It is a specific instance of the METK workflow model, modified for the specific characteristics of the part to be produced;
- maintaining status information on the current state of each component of the manufacturing engineering data such as NC programs, tooling lists, CAD files, etc.;
- controlling access to and recording use of all applications that are integrated into METK;
- specifying and recording information necessary to support the verification models being developed for METK.

As stated previously, efforts are currently underway for the definition of workflow and verification models. This application is the vehicle through which the processes and procedures defined for the models are enforced in the METK system.

5.3 Data Management in METK

A key component of the METK is the common data repository. This will conceptually consist of a set of databases, each recording information on a different aspect of the METK. Such an arrangement is often called a federated database (FDB). The METK federated database will be used to hold information that changes infrequently and is common to two or more applications in the toolset. A federated database is a database system constructed by building a common interface to a collection of existing databases [15]. In the following discussion two views of the federated database will be distinguished: issues relating to the federated database as a whole will be referred to as "global", as in "global schema", while issues relating to one of the component databases of the federated database will be referred to as "local", as in "local schema".

Following is an examination of the data the FDB will contain, the organization of data in the FDB, the mechanisms for accessing database information, and the reason for choosing an FDB architecture.

5.3.1 Local Databases of the Federated Database

The local databases that METK FDB comprise include:

- a *tools database* which contains the physical and operational characteristics of all tools and tool assemblies that can be used by METK applications for specifying manufacturing processes;
- a *machine database* which contains the physical and operational characteristics of all machines that can be used by METK applications for specifying manufacturing processes;
- a *fixture database* which contains physical and operational characteristics of all fixtures that can be used by METK applications for specifying manufacturing processes;
- a *material database* which contains physical characteristics of all materials that can be used by METK applications for specifying manufacturing processes;
- a *process reference database* which contains information on how different manufacturing processes should be carried out such as feature recognition and classification, operation selection, etc.;

- an *archive database* which contains snapshots of information pertaining to a manufacturing engineering data package, stored together and in a compressed format, such that the state of the manufacturing engineering data package at the time of the snapshot could be restored from the information in the snapshot.

5.3.2 Database Organization

With the exception of the archive database, the information in these databases is highly interrelated. An effort concurrent with the development of the METK architecture is underway to define the information models for each of the classes of data that make up the local databases. The local database schemas for each of these databases will be based on those information models. A global database schema will also be developed that provides a logical view of all of the data in the FDB.

The structure of the information to be contained in the local databases and the nature of the relationships between information units in different local databases is highly complex. While flat file or a relational database could be used for storing these complex relationships, the METK will explore the hypothesis that the problems associated with inter- and intra-database complexity can best be mitigated through using an object-oriented database (OODB). There has been a significant amount of research showing the advantages of using an OODB for persistent storage of data related to engineering (e.g., [13, 3]). That work describes how an OODB can be used not only for the local database schemas, but also for the global database schema representing all data in the system. In this way the relationships of the global schema are not logical, but are real relationships which can be encoded into and enforced by the database. This approach will be explored in detail in the METK.

5.3.3 Database Access

Access to the data in the FDB will be through a common data interface (CDI). The CDI provides encapsulation of many of the details related to the federated databases. Encapsulation allows for modifying the implementation of the local databases and database relationships in ways that are advantageous to the METK system. Clients will access the services provided by the interface for common data definition and data manipulation functions. Also if it is determined that several applications will frequently need to perform the same complex access, the CDI's interface can be extended to provide a single point of access to that function.

5.3.4 The Need for a Federation

Since the global database schema can be directly supported by the OODB that will be used to contain the data, the services of the CDI will be implemented in terms of the underlying data definition language (DDL) and data manipulation language (DML) provided by the OODB. It is anticipated that many of the services can be implemented simply as pass through functions. The structure of the FDB can be characterized as an OODB with an interface program to provide a somewhat thin layer of encapsulation. The need for the CDI may seem minimal when the underlying capabilities of the OODB can provide for all of METK's data storage needs. This might be true given the initial METK architectural requirements, but it is highly probable that as the METK project progresses data storage and access requirements will be expanded to support additional classes of data. Separating the interface for

accessing data from its implementation makes the architecture more robust in that it will be resilient to the inevitable changes that occur as systems evolve.

5.4 Application Interfaces

Application Programming Interfaces (APIs), also called Application interfaces, are software components that provide access to the services needed to implement the process capabilities. These services are implemented through the integration of system mechanisms. In section 4.2 an approach was discussed for designing an integrated system by logically partitioning system capabilities into 3 levels called process, service, and mechanism. The mechanism level consists of the COTS applications, operating system tools, and utilities that provide the low level functionality of the system. The process level provides the means for defining and enforcing the domain specific practices, procedures, and policies of the organization that is to use the system. The service level is the means through which the processes are implemented using the mechanisms. Application interfaces provide the service level for METK.

Following is an examination of the architecture of an application interface, the services that will be supported, some reasons for including application interfaces in the METK architecture, and some of the advantages that application interfaces provide to the METK architecture.

5.4.1 Application Interface Specifications and Implementations

There are two logical parts to an application interface, the interface specification and the interface implementation. The interface specification provides functions and procedures that are invoked by architectural elements of METK. This includes COTS CAM applications, the workflow and product data manager, and other application interfaces. Procedure invocation is the means through which service level capabilities are accessed by the process level. The interface implementation is the collection of implementations of all of the procedures and functions in the interface definition. The relationship of the interface specification to the interface implementation is analogous to the relationship of an Ada package specification to an Ada package body, or a C++ class definition to the class member and class method implementations. The procedure and function implementations will make use a combination of UNIX scripts, UNIX utilities, custom written C/C++ programs, embedded SQL, and any functionality provided by APIs of the COTS CAM applications in METK.

5.4.2 Services Provided by Application Interfaces

An application interface component will be associated with each COTS CAM application in METK. It will provide some services that are common to all application interfaces and some services specific to interfacing with its associated COTS CAM application. The complexity of the capabilities provided by different services will vary greatly. Some services will be simple, such as "Copy File xxx" or "Delete File xxx." Some will be of a medium complexity, such as "Convert file xxx to aaa format." But other services may be very complex, such as "Create Process Plan for part 'abc123'." Because the information models, workflow model, and list of COTS CAM tools for the METK are being developed concurrently with the definition of the METK architecture, it is not possible at this time to state a

definitive list of services that are to be provided by the application interfaces. The best that can be said is that the services provided will be sufficient to support the required functionality defined by the processes of the process level. Furthermore the list of services provided by the application interfaces is required to be extensible to support the definition of new policies, procedures, and practices in the process level.

5.4.3 Adaptability and Extensibility using Application Interfaces

Traditional system analysis and design often proceeds based on assumptions that the system requirements are well understood and as a result that changes to the system will be minimal [14]. The ongoing collaborative nature of METK's requirement specification effort and the ongoing concurrent development of METK's information and workflow models makes such assumptions false. Hence, a key architectural motivation was to make the METK architecture as resilient to change and as open to extension as possible. The flexibility afforded to METK through the use of application interfaces should allow for integrating existing COTS CAM tools while minimizing the impact of each tool's unique operational and architectural idiosyncrasies.

More specifically, application interfaces provide many advantages to the METK architecture:

- Support separation of concerns and information hiding. Clients of the services in an application interface need not be concerned with how those services are provided. The workflow model can be developed based on the services that are provided and not need detailed knowledge of application internals.
- Support enhanced access to each application's functionality. The services of an application interface provide a common interface to each application's processing capabilities which can be accessed by many application interfaces.
- Support enhanced adaptability/extensibility of COTS applications. Through its services, application interfaces allow a COTS CAM application's native interface to be extended and adapted for use in the METK system.
- Provide for straightforward extension of METK toolset. Adding a new kind of tool to METK's toolset can be accomplished through defining the services that are to be provided by the new tool's application interface and then implementing the services using the new tool.
- Provide for tool substitutability. A new tool which provides processing capabilities similar to those provided by an existing tool in METK's toolset can replace the existing tool by implementing the existing application interface's services using the new tool.
- Support network deployment. Although METK is now slated for deployment on a single workstation, network deployment could be accomplished through implementing the application interface's services using remote procedure call (RPC) or more sophisticated mechanisms for achieving distribution such as the Common Object Request Broker Architecture (CORBA).
- Support incorporating future computer integrated manufacturing research innovations. Much research into computer aided manufacturing is being done which might, at some future time, be advantageous to include in METK. Areas such as distributed knowledge representation (data is not stored commonly but is distributed throughout the system), distributed control logic (workflow and PDM capabilities are distributed throughout the system), and intelligent agents (data and control logic distributed throughout the system) could provide interesting enhancements to METK's capabilities. Application interfaces would provide a starting point for incorporating these research ideas.

5.5 COTS CAM Tools

The integration of COTS CAM tools is the major focus of the METK project. Manufacturing engineers are currently using these tools to help run their operations, but there is little integration between the tools and it is difficult to replace a tool with another that provides similar capabilities. A candidate list of tools to be integrated as part of the METK has been chosen, but the architecture should not be directly based on the capabilities of the chosen tools. The architecture should treat tools to be included in METK in a generic way. Focusing on the processing capabilities that a tool provides will lessen the chance that shortcomings of a chosen tool will unduly influence the architecture. Following are descriptions of processing capabilities required of the COTS tools that will be integrated as part of METK. In some cases more than one tool will be needed to provide the desired functionality.

The METK system will include COTS tools that provide the capability of performing the following manufacturing engineering functions:

- Computer Aided Design (CAD);
- Process planning;
- NC Program Verification;
- Shop floor Process Verification.

A brief description of each of these functions and key inputs and outputs follows.

5.5.1 Computer Aided Design(CAD)

This function provides for the development of geometric models and the specification of the physical characteristics of a part that is going to be manufactured, and of the machines, tools, tool assemblies, and fixtures that will be used in the manufacturing process. The data generated may be used as an input to other functions and therefore the tool should produce its output in a standard CAD format.

5.5.2 Process planning

This function will be used to produce reports which define how a part is to be produced. These reports include tooling lists, operation plans, NC programs, and routings. Tooling lists define the tools needed to produce the part. Operation plans define what each machine will do to produce a part. NC programs define operations for numerical controlled machines. Routings define the movement from machine to machine of a part being produced. A key input to achieve this is a CAD model describing the product to be produced.

5.5.3 NC Program Verification

This function will take a NC program and verify that it can be used to enable a NC machine to perform the specified operations that transforms a part from one state to another state. CAD models for the machine and for each tool and fixture used in the process may be required as input. Also CAD models for the part at the beginning state, ending state and each intermediate state may be required. Both reports and a visual presentation of the NC machine operating to perform the operation may be produced to satisfy verification.

5.5.4 Shop Floor Processing Verification

This function will verify that the routing of the part through machines in the shop floor will produce the desired part. CAD models for the machines, tools, fixtures, and the part at each stage of its manufacture may be required. Again, both reports and a visual presentation of the parts flow through a representation of the shop floor may be produced to satisfy verification.

5.6 Reporting and Maintenance Module

Each of the components being integrated as part of METK has the capability of generating various reports relating the processing it does and the data that it generates. Since METK is to provide an integrated system, there is a need for reports to be generated which are made of information from several of these tools and from the federated database. The Reporting and Maintenance Module will provide the capability to define and generate such reports. Also some basic querying and maintenance capabilities for end-users to interact with the federated database will be provided.

5.7 Hardware Platform and System Software

The METK system will execute on a Silicon Graphics ONYX Extreme(tm) workstation. A workstation of this class is needed because many of the COTS tools to be integrated require a high performance workstation. Fortunately, all tools in the current toolset support this platform.

The operating system is the IRIX(tm) 5.3 operating system, a UNIX variant for Silicon Graphics workstations. Future plans for the METK system may include porting it to other platforms, and using a UNIX variant should facilitate porting. It is expected any new software modules developed for METK, or customizations to COTS tools, will not be carried out at the system programming level nor require operating system kernel extensions.

Any new software modules developed for METK that require a graphical user interface (GUI) must be developed in a platform independent way. This can be accomplished through the use of commercially available cross-platform libraries, which will also facilitate porting to other hardware platforms.

5.8 Summary

In this section an overview of the architectural elements of METK has been presented. Some of the project goals, stated requirements, and derived requirements were examined for their impact on the architecture. The key factor influencing the architecture is that it must be adaptable to facilitate change, since the project is being run as a collaborative effort with inter-dependent parts of the project being developed concurrently. The application interfaces and federated database should provide a firm enough architectural basis for the project to proceed and provide the flexibility necessary for the METK to evolve.

6 Commentary

The design, implementation, and adoption of the METK raises a number of important issues with respect to integrated manufacturing engineering support. In this section we examine a number of these issues, in each case describing the problem being faced and the approach we are taking.

6.1 Working with COTS Product Vendors

The primary inhibitor to the integration of COTS products is the fact that the COTS products themselves are not constructed with their future integration needs in mind. As a result, COTS products must be carefully examined in terms of various characteristics that increase the likelihood that they can be used in combination [15]. Hence, an important outcome of the CAME program as a whole, and the METK in particular, is to work with software vendors to encourage them to make amendments to their tools to facilitate COTS integration.

From a software vendor perspective, however, making major modifications to their products to aid integration is typically a hard sell. Most of these vendors are relatively small organizations with many competing demands on their resources, with the result that their priorities are usually focused on tasks associated with their current and immediate sales: fixing urgent bugs, adding new user features, optimizing system performance, porting to popular operating systems and hardware platforms, etc.

While recognizing the difficulty of this task, the CAME program believes it is in a good position to affect the COTS product vendors. This is a result of a number of factors.

First, the CAME program has the backing of a strong consortium of manufacturing organizations in government, industry, and academe. This raises the prospect that the vendors supporting the CAME work and its recommendations will have an eager market ready to accept their products.

Second, the CAME program is taking place within the context of the National Institute of Standards and Technology (NIST). NIST has an important role within both government and industry, and has been instrumental in developing, supporting, and introducing a wide range of standards and technologies into widespread commercial use.

Third, the development of the METK will provide a realistic demonstration of capabilities to the COTS product vendors. This demonstration can act as a prototype of future capabilities to focus design decisions, and can be used as the basis of requirements elicitation from potential customers.

6.2 Adoption Issues

Regardless of the technical qualities of the METK, it is vital that the technology has a realistic adoption path for it to be acceptable to practicing manufacturing engineers. Many attempts at developing technology improvements have failed due to the lack of such considerations.

Technology transition issues have been an important concern of the CAME program. The intent is that the METK is provided as a complete, packed prototype that will be installed a a number of organizations for trial use. To assist in this effort a number of important steps will be taken:

- produce the METK as a stand-alone system on a single Silicon Graphics Onyx machine, and provide the complete system of software and hardware to an organization for a trial period lasting several weeks;
- encode a range of actual part designs in the METK that are applicable to the organizations — the task of identifying, encoding, and recording these part designs is already underway;
- develop a range of support materials (e.g., user manuals, training guides, etc.) that will accompany the METK;
- consider the practicality of providing a knowledgeable system administrator on-site with the METK at each trial installation.

It is expected that further adoption steps will be decided in the coming months.

6.3 Balancing Short-term vs. Long-term Goals

The goals of the CAME program are very ambitious. In particular, in the development of the METK we recognize a number of goals in which potential areas of conflict may arise. It is important to recognize these and define approaches that reduce the risk of that conflict.

One source for potential problems is in the relationship between the short-term goals of building a demonstration system based on current COTS product technology, and the long-term goals of creating standards and interfaces that influence future COTS products. For example, in prioritizing costs and resources, decisions must be made as to how much effort should be devoted to the immediate activity of hand-crafting, specific integration solutions for the initial set of tools as opposed to the more long-term activity of creating generalized integration solutions that may be appropriate to a wider selection of COTS products.

It is not possible to take a single approach to address such conflicts (e.g., to assert a fixed set of priorities). Rather, such conflicts must be resolved individually as they arise. For example, in the case of specific versus general integration solutions a mixed approach is being pursued. On the one hand, substantial use is being made of the existing IMW prototype, with plans to improve and enhance its capabilities. On the other hand discussions are taking place with COTS product vendors to define data schemas that can be used as the basis for a more general approach to data sharing among a range of manufacturing engineering products.

7 Summary and Conclusions

Integrating manufacturing engineering tools is a complex task requiring a great deal of resources and effort. This paper has described one large program that is directed at this challenge. It involves a consortia of government, industry, and academe, and is aimed at developing practical techniques, standards, and interfaces that will enable integrated manufacturing engineering toolkits to be developed in a predictable, cost-effective manner. Key concerns

being addressed include support for flexibility and adaptability of the toolkit through an architecture that clearly separates process, service and mechanism aspects.

The program is facing a number of technical, organizational, and political challenges in carrying out its work:

- *Technically*: There are many issues in the design and implementation of the METK that must be addressed in building a substantial demonstration that can be used to convince potential users of the value of integrated manufacturing engineering system;
- *Organizationally*: The different strengths of the various organizations must be harnessed to ensure measured progress is being achieved in a reasonable timescale;
- *Politically*: The individual needs of the government, industrial, and academic organizations must be balanced to ensure that widely applicable results are achieved that can be readily transitioned into off-the- shelf products while addressing the interesting technical challenges that advance the state-of-the-art.

Thus far the program is making steady progress toward its goals. A prototype METK has been developed and has been demonstrated to members of the consortia. Many essential requirements have still to be addressed, and further enhancements of the METK are planned.

Progress has also been made in the development of standard data schemas for integrating manufacturing engineering tools. Substantial input to these data schemas from tool vendors, system integrators, and end-user organizations provide the promise that such schemas will achieve relatively wide-scale acceptance. Drafts of these schemas are under review and will be revised based on further feedback from the CAME consortia members.

Acknowledgments

Work described in this paper was sponsored by the U.S. Navy Manufacturing Technology Program. No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied. The work described was funded by the United States Government and is not subject to copyright.

The IMW was funded by a grant from the Ohio Aerospace Institute under their Collaborative Research Program.

The Software Engineering Institute is sponsored by the U.S. Department of Defense.

References

1 "Computer-Aided Manufacturing Engineering Forum First Technical meeting proceedings," NISTIR5699, March 21-22 1995.

2 Birman, K.P. 1991. Maintaining Consistency in Distributed Systems. Technical report, Cornell University Department of Computer Science, Technical Report TR91-1240,pp 19-20.

3 Brown, A.W. 1991. "Object-Oriented Databases: Applications in Software Engineering", McGraw-Hill.

4 Brown, A.W., Carney, D.J., Morris, E.J., Smith, D.B., Zarrella, P.F. 1994. "Principles of CASE Tool Integration", Oxford University Press.

5 Descotte, Y. and Latombe, J. 1984, "GARI: A Problem Solver that Plans how to Machine Mechanical Parts," IJCAI, pp 41-54.

6 Dunn, M. S. and Mann, W. S. 1978, "Computerized Production Process Planning," Proc. of the 15th Numerical Control Society Annual Meeting and Conference, Chicago, IL.

7 Garlan, D. and Shaw, M. 1994. "An Introduction to Software Architecture", Technical report CMU/SEI-94-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

8 Ham, I. and Lu, S. C. Y., "CAPP Present and Future," CIRP Annals, V 37, N. 2, pp. 591-601.

9 Hayes, C. and Wright, P., "Automated Process Planning: Using Feature Interactions to Guide Search," Journal of Manufacturing Systems, V. 8, N. 1, pp. 1-13.

10 Joshi, S. and Chang, T. C., 1987. "CAD Interface for Automatic Process Planning," Proc. of the 19th CIRP International Seminar on Manufacturing Systems, V. 1, N. 1, pp. 39-45.

11 Judd, R., Parks, C., Renuka, S., and Kashyap, M. 1995. "Intelligent Machining Workstation," Proc. of the 1995 Summer Computer Simulation Conference," Ottawa, Ontario, Canada.

12 Kanumury, M. and Chang, T., "Process Planning in an Automated Environment," Journal of Manufacturing Systems, V. 10, N. 1, pp. 67-78.

13 Loomis, M.E.S. 1995. "Object Databases: The Essentials", Addison-Wesley, 1995, pp 197-200.

14 Lorenz, M. 1993. "Object Oriented Software Development: A practical Guide", P T R Prentice-Hall, pp 6-8.

15 Nejmeh, B. 1989. Characteristics of Integrable Software Tools. Technical Report INTEG_S/W_TOOLS-89036-N Version 1.0, Software Productivity Consortium, May 1989.

16 Preiss, K. and E. Kaplansky, E., "Automated Part Programming for CNC Milling by Artificial Intelligence Techniques," Journal of Manufacturing Systems, V. 4, N. 1, pp. 51-63.

17 van't Erve, A.H. and Kals, H. J. J. 1986, "XPLANE, a generative Computer Aided Process Planning System for Part Manufacturing," Annals of CIRP, V. 2, pp. 324-9.

18 Wang, H.P. and Wysk, R. A. 1986, "Intelligent Reasoning for Process Planning," technical paper, Department of Industrial and Systems Engineering, Penn-State University, PA.