A METHODOLOGY FOR INTEGRATING SENSOR FEEDBACK IN MACHINE TOOL CONTROLLERS

Fred Proctor John Michaloski Tom Kramer

Robot Systems Division
National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce
Gaithersburg, Maryland 20899[1]

ABSTRACT

A reference model architecture for real-time hierarchical control systems has been proposed by researchers at the National Institute of Standards and Technology, and has been implemented on a variety of computing platforms for manufacturing and vehicle control applications. A fundamental aspect of this architecture is the notion of nested control loops, which incorporate sensory feedback in a hierarchy whose cycle times decrease in frequency as planning becomes more abstract. The nested control loops provide a hierarchy in which to model command and control. This architecture was formalized during work with the National Aeronautics and Space Administration on the Flight Telerobot Servicer for the space station, and is known as the NASA/NBS Standard Reference Model, or NASREM. Although NASREM was intended to serve as a guideline for space application robot control, it has applicability to a wide range of real-time control applications. This paper adapts the NASREM reference model architecture to a machine tool control model. A computational architecture will be presented that describes expected behavior at each layer. A functional analysis will outline a baseline task tree vocabulary. The task tree vocabulary is given by a set of command verbs for each layer and is a critical component of task description within a hierarchical control system.

INTRODUCTION

The accuracy of a machine tool can be greatly increased through the use of high-resolution position sensors such as glass scales, or calibration of components such as lead screws or gears. Unfortunately, much of the inaccuracy in finished parts is due to dynamically varying quantities, such as tool wear or chatter, which cannot be predicted in advance. Methods have been developed which rely on sensors to measure these quantities in real time as the part is being machined, and modify the position of the machine tool to compensate for these disturbances. Incorporating these methods into machine tool control brings benefits in failure prediction, surface finish improvement, and accurate machining, all of which improve the quality and reduce the cost of manufacturing. The limit to the effectiveness of these methods lies not with the engineering principles, but in the practical effort required to interface to proprietary machine tool controllers with closed architectures.

The answer is an open architecture that integrates sensor feedback into the control structure. At NIST, an architecture for integration of sensor and control has been applied to numerous projects. This paper will review the architecture as outlined in the NASA/NBS Standard Reference Model for Telerobotic Control System Architecture (NASREM) [1] as

[1] Presented at the Second International Conference on Flexible Automation and Information Management, Washington, DC, June 30, 1992.

it applies to machining. NASREM describes a system architecture to handle the specific responsibilities of the Space Station Flight Telerobot (FTS), a multiarmed manipulator intended to assist service and construction of the Space Station Freedom. NASREM proposes a hierarchical control methodology to decompose FTS functionality from high-level Space Station directives into low-level physical actions. NASREM is not restricted to space robotics. It is a general real-time control model and architecture, adaptable to a broad range of applications.

The goal of this paper is to present canonical or base-class vocabularies of machine tool commands stratified along levels of operation. The stratification of operation subscribes to the NASREM notion of hierarchical, feedback control systems. The canonical vocabularies drew from machining experience from the Automated Manufacturing Research Facility (AMRF) at the National Institute of Standards and Technology (NIST). The AMRF serves as a testbed for developing techniques and standards for automated manufacturing. The paper is organized as follows. The first section will describe the basic NASREM architecture and present the enabling architectural design concepts. The second section will study the concept of NASREM-style commands and content of command vocabularies. The third section will present canonical commands for a machining hierarchy. Finally, an example of the sensor-integrated machining architecture will be presented.

BACKGROUND

NASREM defines an application control system as a hierarchical collection of sensory-interactive, controller nodes. These controller nodes reflect the fundamental aspect of the replicated architectural structure. A *controller node* is composed of sensory processing (SP), world modeling (WM), behavior generation (BG) and appropriate human interface (HI) components. Behavior generation determines control activity and sends either actuator signals or commands to subordinate controller nodes to effect that activity. Sensory processing obtains and processes feedback from system sensors and subordinate controller nodes. World modeling interprets sensory processing data to maintain an internal model of the world. World modeling is the part of the system which mediates sensory processing and behavior generation activities. The key enabling architectural design concepts are:

- *hierarchical organization*—levels of control are derived from a hierarchical decomposition of control functionality, and hierarchical composition of sensory-feedback knowledge.
- *well-structured*—all controller nodes have the same structure and data formats
- *cyclic execution*—node execution provides a predictable model using feedback of command and status.
- *inherently concurrent*—nodes are concurrent. NASREM controller nodes support concurrent threads of execution.
- *sensory-interactive*—closed loop control is possible. SP external world sensing in conjunction with WM internal predictions provide robust feedback.
- *one master rule*—all subordinate control processes obey a single superior control process at each instant in time. A superior may control multiple subordinates. This gives a hierarchical control tree decomposition.

Of great importance to sensor-based machining is the concept of sensor and control integration. NASREM integration of control and sensor feedback comes out of the tradition of servo mechanisms and state-space control. Each of the controller nodes is a sensory-interactive servoed feedback loop. Each controller node accepts task commands that define goals (i.e., set-points, or attractor sets). Each controller node regularly samples sensors, computes the state of the world, and generates actions designed to reduce the difference between the current state of the world and the goal state. The overall architecture is a multilevel, hierarchical, nested control system. An implementation would typically employ

periodic “servoing” or data sampling as opposed to “event-driven” interrupt processing in order to cut down on processing overhead and at the same time ensure deterministic and verifiable behavior (in terms of execution time and response time) particularly at the lowest levels of the architecture. At higher levels of an implementation—and lower performance bandwidths—it is often convenient to transition to an asynchronous technique.

The fundamental NASREM design principle is the use of hierarchical task decomposition with stepwise refinement to reduce a larger problem into more elementary steps, as well as bottom-up aggregation of controller nodes based on equipment composition. Hierarchical aggregation of control nodes leads to the definition of a *control level* as the collection of controller nodes operating at the same spatial and temporal level of execution. NASREM decomposes control actions into levels both temporally and spatially. Temporally, each successively higher level depends on completion of a lower level task, much like increasing digits of magnitude on an odometer. Spatially, levels are similar to different magnifications of a microscope—lower levels have a higher degree of magnification, but observe less of the total picture.

As an example of this architectural structure, consider the hierarchy shown in Figure 1. This diagram depicts an abridged view of a machining cell consisting of a workstation containing a machine tool and a robot consisting of one arm with a simple gripper, and a camera with pan, tilt, zoom, focus and iris control. Applying NASREM stratification leads to a hierarchy of six levels. The path from the cell level to the robot arm in this hierarchy, for example, implements the following functionality:

- Level 6: CELL decomposes the factory orders into a sequence of workstation action commands.
- Level 5: WORKSTATION decomposes actions to be performed on batches of parts into tasks performed on individual objects.
- Level 4: TASK decomposes actions applied to object task commands into sequences of elemental moves defined in terms of motions and goal-points.
- Level 3: ELEMENTARY MOVE (EMOVE) decomposes elemental move goal-points into paths.
- Level 2: PRIMITIVE (PRIM) decomposes paths into smooth, dynamic trajectories.
- Level 1: SERVO transforms trajectories into coordinated joint motion space, using either position, velocity or force parametrization.

NASREM COMMAND VOCABULARY

Controller processes within the NASREM hierarchy communicate through message passing. These messages are defined by a language that will be referred to as the neutral messaging language, or simply NML*. NML is not a programming language; it is a type of non-procedural language. Further, NML is not a general purpose grammar. Instead, a project specific language is developed that defines the set of messages into and out of each controller node. Of course, projects that share many characteristics could in fact use an identical NML. Thus, dialects of NML can be developed for all or portions of a specific control applications. As an application language example, NASA has developed vocabularies for space robotics [2]. As a portion of a control application example, a PRIM controller node may communicate to a SERVO controller node with the NML defined by the RS-274 machine tool programming language.

The method to develop the grammars is to analyze the functionality of each controller node and develop verbs which describe appropriate commands. The set of NML verbs defines controller node input commands or task names. An NML message is then a verb

* Compare with the Neutral Manufacturing Language specified by the Next Generation Controller program. The Neutral Messaging Language specifier was chosen to evoke the connotation of the Neutral Manufacturing Language, generalized as a messaging language to more accurately reflect its suitability for control in non-manufacturing applications, such as vehicle control.

together with a list of controller node attribute values defining the task goal, object, and parameters. Command names should use the active connotation of a verb since a verb has a rather descriptive and characterizing quality of the intended action. Hence, command names will be termed as the *command* or *task verb*. The collection of all task verbs in a controller node will define a *task vocabulary*. Other vocabularies for status and world model communication are also necessary, but will not be covered in this paper.

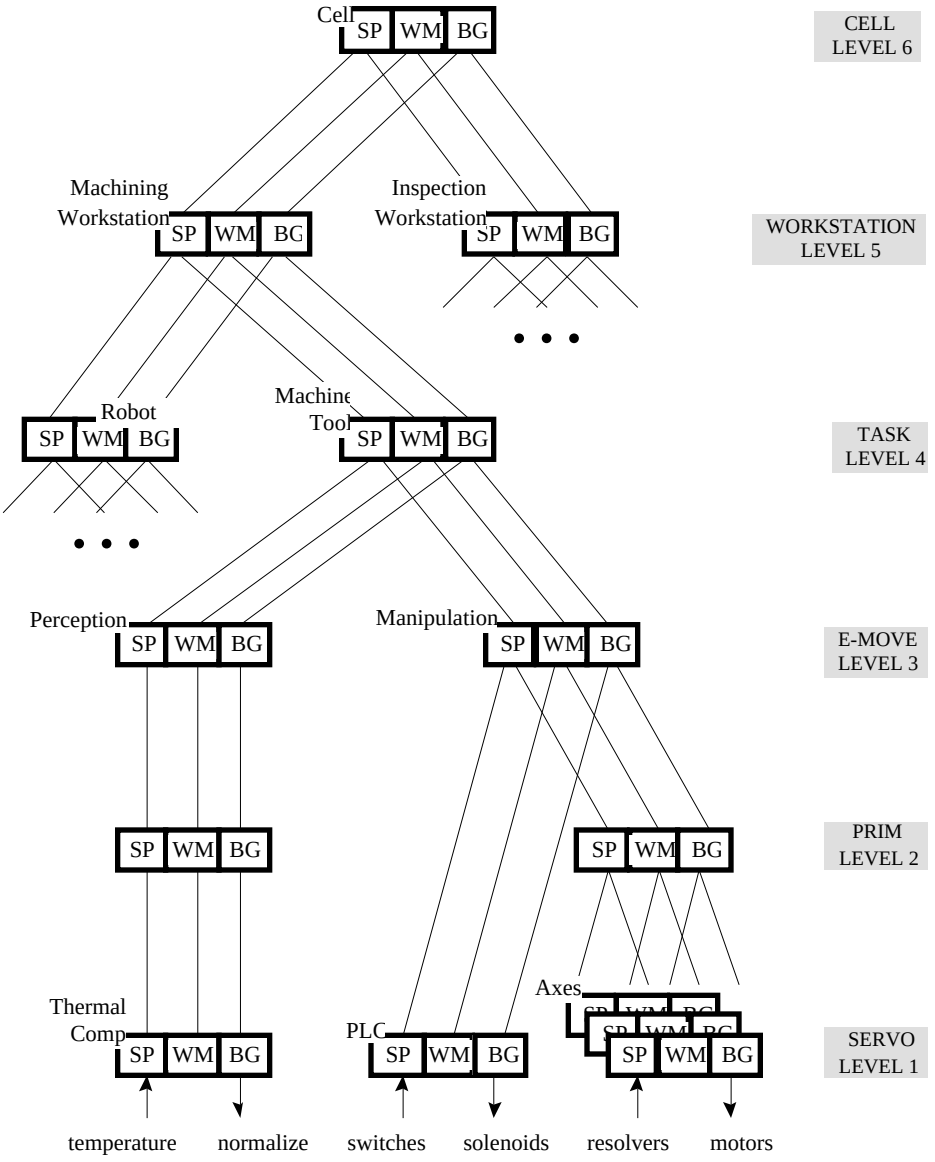

Figure 1. Machining Cell Control Hierarchy

Each controller node should have a baseline set of verbs describing the atomic actions of that control node. Verbs should be unambiguous and descriptive. Thus, verbs such as "do," which don't provide the necessary level of description, should be avoided. Defining a controller node task vocabulary with a set of verbs is a fuzzy linguistic process [3]. One must match the needs of the task to the capabilities of the controller node. What makes the linguistic process fuzzy is that there exist numerous equally valid grammars that can define the task vocabulary. For example, one can specialize the task vocabulary to match specific project goals or one can use a smaller, general vocabulary that assumes extra steps. For example, one could have the task verb CHANGE-OUT, if this were a specific need of the

project, or one could use more general verbs, such as REMOVE and INSTALL. In the second case, a CHANGE-OUT can be accomplished with a composition of REMOVE and INSTALL tasks.

To define controller node vocabularies, one must stratify functionality into the NASREM hierarchy. Some functionality transcends levels within the hierarchy. This leads to the concept of an internode function or task. For example, all levels of the hierarchy which exhibit motion could have the verb MOVE. Redundancy of the verb MOVE across levels is necessary to capture the different time and spatial capabilities. To illustrate the stratification of motion control within the functional model, a partial verb vocabulary for a single-chain of controller nodes is presented in Figure 2.

In this diagram, TASK communicates task commands to EMOVE with the verbs APPROACH, DEPART, MOVE, and GRASP. APPROACH and DEPART are obviously moves, but imply some special parametrization that is required as one nears contact. Further, the PRIM to SERVO interface vocabulary contains only a single command verb. Different motion requirements in this interface do not translate nicely into different verbs, so that a single verb with different sets of parameters is required. Each level issues new commands based on feedback from the lower level. Such new commands and parametrization can be issued upon a simple acknowledgment of task completion or a complicated adaptive control modification based on sensed feedback.

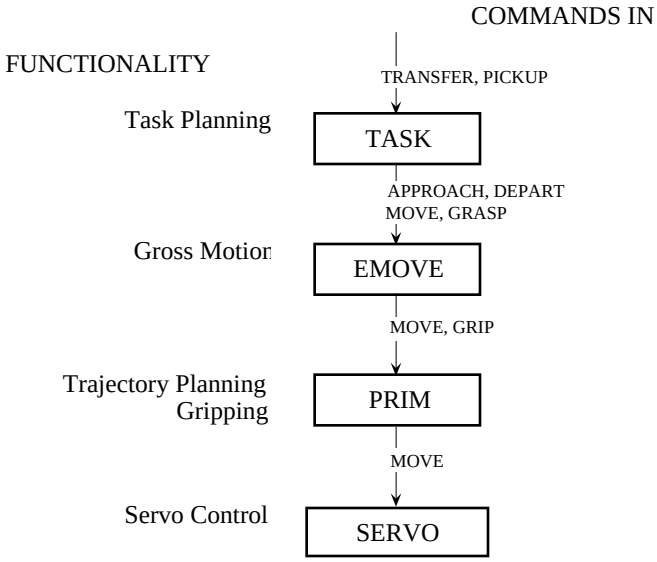

Figure 2. Sample Command Hierarchy

CANONICAL COMMANDS FOR 3-AXIS MACHINING

For the general case, a CELL would be capable of making parts itself and would contain both machining and inspections workstations. A turning, vertical machining, and inspection workstation, for example, might comprise a CELL. A single part might be routed through all three workstations. Each WORKSTATION (Level 5) would containone or more pieces of equipment, each of which would have a controller at the TASK level (Level 4). The machining workstation of Figure 1 is typical of this view. The activity of a workstation would be to issue commands to subordinate equipment to get it to transform the workpiece according to the capabilities of the workstation.

For a machining workstation (vertical or horizontal, and any number of axes) the command to the workstation would result in the partial or complete machining of a part, starting from a piece of stock or a partially machined workpiece, and ending with a partially machined workpiece or a completely machined part. The vocabulary of CELL and

WORKSTATION commands greatly depends on allocation and scheduling strategies that will not be addressed in this paper. Instead, command vocabularies will be presented for TASK and subordinate controllers that are invariant of time and scheduling considerations.

A given controller might decompose a command from a superior by simply taking a pre-built process plan off the shelf and constructing commands from the process plan, or it might plan and build commands in real time. It is expected that a command would be very similar to the step of a process plan from which it was built. The command would have a variety of information relating to timing and sequencing not found in the process plan, and some process plan information (such as expressions evaluated by the controller to get explicit values) would not be contained in the command.

A command might be a macro that decomposes into a series of commands for the same controller. In machining, this is already done with existing controllers in implementations of RS-274-D, for example, (commands g80 through g89 are macros which decompose into g0, g1, g2, g3 and other elementary commands). The discussion of canonical vocabularies will concentrate on commands for making parts and not with commands for setting up or taking down a hierarchy of controllers, dealing with error conditions, etc. Some method for doing these tasks would, of course, be required but is out of the scope of this paper. These issues might or might not be handled similarly to the handling of manufacturing tasks.

The remainder of this section will outline a list of proposed canonical commands for a 3-axis machining center. Most of these commands should be suitable for machining centers with more than three axes. Commands at Level 4(TASK), Level 3 (EMOVE), and Level 2 (PRIMITIVE) of the NASREM hierarchy are given. For each command, the name of the command and a description of the effects of the command are given. Parameters are given in italics for Level 4 and Level 3 commands. Parameters for Level 2 commands are generally required for each command and are included in a follow-up discussion on control and data parametrization.

TASK Level

Each workstation would have its own component equipment, and each would require its own set of task level commands. In a machining workstation, there might be controllers for a robot which would load parts, unload parts, move tools in and out of the machining center, and adjust passive fixtures. There may also be separately controllable equipment, such as a powered adjustable fixture. For machining, the task level is responsible for interpreting the geometry of a part and then generating a process plan to machine that part.

The REMOVE_VOLUMES (*plan_id, design_id, material_removal_volumes_id, setup_id, workpiece_id, fixture_id*) command causes a set of material removal volumes to be removed from a workpiece. The command parameters include a process plan identifier (*plan_id*), a design identifier (*design_id*), an identifier for the set of material removal volumes referenced in the program (*material_removal_volumes_id*), an identifier for setup instructions (*setup_id*), an identifier for the workpiece that will be machined (*workpiece_id*), and an identifier for the fixture to be used (*fixture_id*). This command is used to do the work done in a single fixturing of a workpiece. The workpiece may start as a piece of stock or as a partially machined workpiece. The workpiece may end as a completely machined part or as a partially machined workpiece.

EMOVE Level

The elemental move level is the transition point between the abstract notion of geometrical part description and the physical notion of machining. EMOVE is responsible for feature interpretation and generation of tool or motion paths to achieve removal of some volume of material. Each machine tool requires its own geometric and physical model of

volume removal and tool path generation.

The BORE (*tool_type_id, material_removal_volume, spindle_speed, feed_rate*) command results in a hole being bored. The cutter must be a boring tool.

The CENTER_DRILL (*tool_type_id, material_removal_volume, spindle_speed, feed_rate*) command results in a small starter hole being made with a center drill cutter by a single-stroke plunge into the material.

The COUNTERBORE (*tool_type_id, material_removal_volume, spindle_speed, feed_rate*) command results in an existing hole being enlarged.

The END_PROGRAM (*no parameters*) command indicates the end of a program has been reached. It may cause activities such as spindle retract, return to home position, cleanup of the world model, resetting machine parameters, etc.

The FACE_MILL (*tool_type_id, material_removal_volume, spindle_speed, feed_rate, pass_depth, stepover*) command results in the material removal volume being machined away by a face mill cutter.

The FINISH_MILL (*tool_type_id, material_removal_volume, spindle_speed, feed_rate, stepover*) command results in the removal with a finish end mill (with cutter nose geometry suitable for the material removal volume) of any material in the material removal volume, so that the resulting surfaces meet some desired quality specification. Only a small thickness of material should be removed in this operation. The *stepover* parameter is required for milling with the flat portion of the nose of the cutter, where an area larger than the area of the nose of the tool is being finished.

The FLY_CUT (*tool_type_id, material_removal_volume, spindle_speed, feed_rate, pass_depth, stepover*) command results in the material removal volume being machined away by a fly cutter.

The INITIALIZE_PROGRAM (*program_name, design_id, material_removal_volumes_id, setup_id, workpiece_id, material, fixture_id, program_x_zero, program_y_zero, program_z_zero*) command initializes the controller to be ready to accept additional Level 3 commands, all of which, up to an END_PROGRAM command, are logically parts of a single program for machining a single workpiece using a single fixture. The command identifies the name of the program (*program_name*), a design identifier (*design_id*), an identifier for the set of material removal volumes referenced in the program (*material_removal_volumes_id*), an identifier for setup instructions (*setup_id*), an identifier for the workpiece that will be machined (*workpiece_id*), the name of the type of material being machined (*material*), an identifier for the fixture to be used (*fixture_id*), and the location of the program zero in machine coordinates (*program_x_zero*, *program_y_zero*, and *program_z_zero*). This command causes no motion in the machining center. This command is not used to bring the machining center task controller to a ready state from a cold start; that must be done before an INITIALIZE_PROGRAM command is given.

The MACHINE_CHAMFER (*tool_type_id, material_removal_volume, spindle_speed, feed_rate*) command results in an edge being chamfered. The cutter must be a chamfer tool (tool profile is a cone, possibly truncated).

The MACHINE_COUNTERSINK (*tool_type_id, material_removal_volume, spindle_speed, feed_rate*) command results in a hole being countersunk with a countersink cutter.

The MACHINE_ROUND (*tool_type_id, material_removal_volume, spindle_speed, feed_rate*) command results in an edge being rounded. The cutter must be a rounder (side of tool profile is an arc of a circle).

The PERIPHERAL_MILL (*tool_type_id, material_removal_volume, spindle_speed, feed_rate, pass_depth, stepover*) command is for milling an exterior or interior contour by milling at the periphery only. Unlike ROUGH_MILL, it may not contain any plunging or slotting. The cutter must be an end mill (with nose geometry suitable for the material removal volume).

The REAM (*tool_type_id, material_removal_volume, spindle_speed, feed_rate*) command causes a small amount of material to be removed from the inside of an existing hole. The cutter must be a ream. The material cut away must be a very small thickness around the surface of the material removal volume.

The ROUGH_MILL (*tool_type_id, material_removal_volume, spindle_speed, feed_rate, pass_depth, stepover*) command results in the milling with an end mill of the designated material removal

volume. It is expected that requirements on the surfaces created by this operation will be such that no consideration needs to be given to surface quality in determining machining methods. The operation may include plunging, slotting, and both conventional cutting and climb cutting. The cutter which is used may be a rough end mill or a finish end mill (with nose geometry suitable for the material removal volume).

The SET0_CENTER (*tool_type_id, near_x, near_y, x_offset, y_offset, near_diameter*) command results in a probe cycle being run in which a hole with its axis parallel to the z-axis is probed. Program x_zero and y_zero are set at the center of the hole or by offsetting from the center. The tool must be a probe.

The SET0_CORNER (*tool_type_id, near_x, near_y, x_offset, y_offset, corner_type*) command results in a probe cycle being run in which a corner is probed. The corner must be formed by two planes parallel to the z-axis. Program x_zero and y_zero are set at the corner or by offsetting from the corner. The tool must be a probe.

The SET0_Z (*tool_type_id, x_location, y_location, z_offset*) command results in a probe cycle being run in which a surface parallel to the xy-plane is probed. Program z_zero is set at the surface or by offsetting from the surface. The tool must be a probe.

The SLOT_MILL (*tool_type_id, material_removal_volume, spindle_speed, feed_rate, pass_depth*) command results in a slot being milled. The shape of the material removal volume will be such that it may be produced by having the tool follow a path (simple or complex) in which the tool will generally be cutting across its full width to form a slot.

The TAP (*tool_type_id, material_removal_volume, spindle_speed, feed_rate*) command results in the inside of an existing hole being threaded. The cutter must be a tap.

The TWIST_DRILL (*tool_type_id, material_removal_volume, spindle_speed, feed_rate, pass_depth*) command results in a hole being drilled. The pass_depth parameter is used if the user's TWIST_DRILL strategy is to perform peck drilling. It may be desirable to split this command into three commands: PLUNGE_DRILL, PECK_DRILL, and SMALL_HOLE_DRILL.

PRIMITIVE Level

The PRIMITIVE level is responsible for generating a time sequence of closely-spaced goal states from a static description of a desired motion. As such, it generates primitive trajectories or motion profiles. The output commands of the EMOVE level are time-independent descriptions of motions, for example static position or position and force paths, or directional fields. In the first case, the position and force commands are in the form of parametrized paths to be followed. In the case of a directional field description, the command takes the form of position-dependent fields which indicate the desired direction of motion or force application. In addition to these types of commands, EMOVE can also simply specify a set of termination conditions, or goal states, along with an algorithm specification which determines the strategy to be used to achieve them. This type of command is useful for sensory-interactive algorithms.

The ARC_FEED (*first_axis_coordinate, second_axis_coordinate, rotation, axis_end_point*) command describes a move in a helical arc from the current location at the existing feed rate. The axis of the helix is parallel to the x, y, or z axis, according to which one is perpendicular to the selected plane. The helical arc may degenerate to a circular arc if there is no motion parallel to the axis of the helix. If the selected plane is the xy-plane, *first_axis_coordinate* is the axis x-coordinate, *second_axis_coordinate* is the axis y coordinate, and *axis_end_point* is the z-coordinate of the end of the arc. If the selected plane is the yz-plane, *first_axis_coordinate* is the axis y-coordinate, *second_axis_coordinate* is the axis z-coordinate, and *axis_end_point* is the x-coordinate of the end of the arc. If the selected plane is the xz-plane, *first_axis_coordinate* is the axis x-coordinate, *second_axis_coordinate* is the axis z coordinate, and *axis_end_point* is the y-coordinate of the end of the arc. The *rotation* parameter represents the number of degrees or radians in the arc. *Rotation* is positive if the arc is traversed counterclockwise as viewed from the positive end of the coordinate axis perpendicular to the currently selected plane. The radius of the helix is determined by the distance from the current location to the axis of the helix.

The DWELL (*duration*) command indicates a pause in motion for the amount of time specified by *duration*. The ability to dwell is useful in finish cutting.

The PARAMETRIC_2D_CURVE_FEED (*first_function, second_function, start_parameter_value, end_parameter_value*) command describes a move along a parametric curve in the selected plane. We will call the parameter *u*. If the selected plane is the xy-plane, *first_function* gives x in terms of *u* and second_function gives y in terms of *u*. Analogous assignments are made if the selected plane is the xz-plane or the yz-plane. Allowable functions should include at least cubic polynomials and elementary trigonometric functions. The current position of the spindle must be at coordinates corresponding to the *start_parameter_value*. The final position of the spindle is at coordinates corresponding to the *end_parameter_value*.

PARAMETRIC_3D_CURVE_FEED (*x_function, y_function, z_function, start_parameter_value, end_parameter_value*) describes a move along a parametric curve in three dimensions, where x, y, and z are each functions of the parameter *u*. Allowable functions should include at least cubic polynomials and elementary trigonometric functions. The current position of the spindle must be at the coordinates corresponding to the *start_parameter_value*. The final position of the spindle is at the coordinates corresponding to the *end_parameter_value*.

The SPINDLE_RETRACT (*no parameters*) command describes a retract at traverse rate to fully retracted position.

The STRAIGHT_FEED (*x, y, z, probe*) command describes a move in a straight line at existing feed rate (or using the existing z-force) from the current point to the point given by the *x*, *y* and *z* parameters. If z-force is enabled, the values of *x* and *y* must be the same as those of the current point. The *probe* parameter is optional. If the *probe* parameter is present, the feed motion will stop when the probe is tripped or when the endpoint is reached, whichever happens first.

The STRAIGHT_TRAVERSE (*x, y, z*) command describes a move in a straight line at traverse rate from the current point to the point given by the *x*, *y* and *z* parameters.

Other PRIM Physical Activities The FLOOD_OFF, FLOOD_ON, MIST_ON, and MIST_OFF commands enable or disable flood or mist coolants. The START_SPINDLE_CLOCKWISE and START_SPINDLE_COUNTERCLOCKWISE commands turn the spindle in the appropriate direction at the currently set speed rate. The STOP_SPINDLE_TURNING command stops spindle rotation. The LOCK_SPINDLE_Z command locks the spindle against vertical motion. The UNLOCK_SPINDLE_Z command unlocks the spindle to permit vertical motion. The SET_SPINDLE_FORCE (*force*) sets the force with which the spindle is pushed in the z-direction, potentially useful in tapping operations. This command also specifies whether or not to use the spindle force. The CHANGE_TOOL (*tool id*) command indicates a tool change. The TURN_PROBE_ON and TURN_PROBE_OFF commands turns the machine probe on or off. The ORIENT_SPINDLE (*orientation, direction*) command turns the spindle to the given orientation and direction at the current spindle speed, then stops the spindle.

PRIM Data and Control Parametrization In a normal machining operation, the execution is open-loop: a series of commands are issued by one level and interpreted at a lower level. This execution mode does not accommodate servo update parametrization. For example, it may be desirable to specify the feed rate with each commanded motion path every servo update. Traditional parametrization prefers to minimize communication, and only send this parameter once with a set command. However, to effect the servo-update mode of command feedback, all relevant parameters should be included within every command, with default parametrization established with the SET command. This leaves the following parameters as either command parameters or isolated set data commands:

FEED_RATE (*feed*) sets the feed rate that will be used when the spindle is told to move at the currently set feed rate. TRAVERSE_RATE (*rate*) sets the traverse rate that will be used when the spindle traverses. SPINDLE_SPEED (*speed*) sets the spindle speed that will be used when the spindle is turning. CUTTER_RADIUS_COMPENSATION (*radius*) sets the radius value to be used in cutter radius compensation, and enables or disables cutter radius

compensation when executing spindle translation commands. The PROGRAM_ORIGIN (*origin*) parameter specifies an absolute or relative origin. TOOL_LENGTH_OFFSETS (*offset transformation*) parameter specifies normal, modified or no tool length offsets. SPEED_FEED_SYNCHRONY imposes or cancels the requirement that feed and speed rates be synchronized exactly.

EXAMPLE

This section describes a prototype architecture for sensory-interactive machining. Figure 3 illustrates the architecture consisting of 4 levels of controller nodes. A job in this architecture performs the following steps. First, a command to manufacture a single part or batch of parts is sent to the TASK level, which interprets the part CAD boundary representation to produce a Constructive Solid Geometry (CSG) model of the part. With the CSG model, part features are extracted and a process plan is generated to cut each feature. Then, part features from the process plan are passed to the EMOVE level to determine the proper machining paths. Feature path geometries are sent to the PRIMITIVE level, which generates path segments. The SERVO level interprets these path segments to produce actuator signals. As commands cascade down the BG leg of the hierarchy, feedback percolates up the SP hierarchy. This section will review the use of sensor feedback at applicable levels of operation [4]. As status feedback filters up the hierarchy, a superior may use lower-level feedback to alter its control strategy. For example, EMOVE may modify cutting parameters if the SERVO level senses tool vibration.

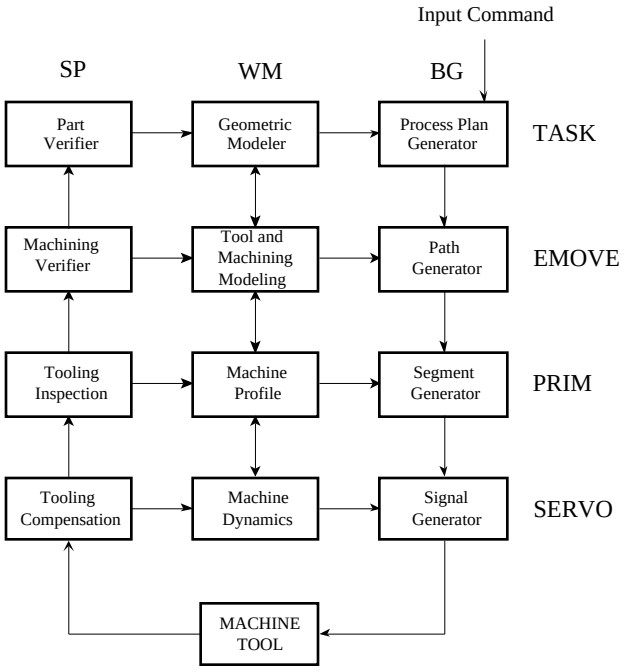

Figure 3. Machining Functional Hierarchy

The TASK level will analyze geometric models of workpieces, fixturing and tools. A typical command is to take the model of an “as-is” blank workpiece and generate a process plan describing feature volumes to be removed that will generate the “to-be” model of the workpiece. Another responsibility will be to test whether the swept volume of a tool following a path intersects the volume occupied by fixturing. For sensor feedback, the TASK level monitors the progress of the job to check deadlines, and expected worksystem

failures. For example, if a machined part has exceeded some overall tolerance limit, then the part is discarded and a new blank part and alternative process plan must be undertaken.

The EMOVE level will interface the feature-based machining to the physical machine tool. EMOVE will do Tool Path BG to generate the tool path required to carry out a step from a process plan. It will also identify canned cycles (such as peck drilling) if that is appropriate to the plan. The Tool Path Generator will undertake geometric analysis of the cutting situation in order to generate efficient paths. For example, it will find entry points for peripheral milling, and it will avoid cutting air where material has been removed by earlier operations. The EMOVE BG will use world modeling to determine tools, spindle speeds, feed rates, stepovers, pass depths.

For sensor feedback, EMOVE can do machining verification. If an inspection probe is available, feature tolerance verification can be performed on the part. Out of tolerance machining could result in replanning to bring the part within tolerance, in cases where the part is oversized. EMOVE could also perform tool wear detection and replacement by measuring tool sharpness with either a torque, temperature or acoustic sensor. A camera sensor could do visual inspection to detect excess chip accumulation and invoke a clean-up procedure.

The PRIM Segment Generator will translate EMOVE paths into straight line, arc or helix path segments. PRIM validates that the tool is proper for its machining use. Then, Machine Profile conditions the path to check for suitable machine limits, spindle speed, feed rate, cutting depth, and horizontal stepover. For sensor feedback, PRIM can inspect cutter radius to compensate for different sizes of the cutter. Further, torque feedback on the tool shaft can be used to adjust the feed rate.

The SERVO level will apply a control law to transform path segments into the necessary actuator signals. For feedback, SERVO is responsible for minimizing machining error through such techniques as chatter detection and thermal compensation [5]. SERVO feedback can detect chatter by sensing tool vibration with an accelerometer and then alter the feed rate, or change the spindle speed. Thermal compensation can fix positional errors due to machine thermal expansion [6].

SUMMARY

A reference model for hierarchical control has been applied to the control of a machine tool workcell, emphasizing the requirement for sensor feedback in areas other than traditional position control. As part of the development of this reference model, a candidate set of messages for three-axis machining has been generated. This messaging language for machine tools grew from consideration of a scenario involving the four lower levels of a machine tool hierarchy, described in the final section.

REFERENCES

1. Albus, J.S., McCain, H.G. and Lumia, R., "NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM)", Technical Note 1235, National Institute of Standards and Technology, Gaithersburg, MD, June 1987.
2. "Flight Telerobotic Servicer, Task Analysis Methodology," Ocean Systems Engineering, McLean, VA., NASAcontract NAS5-30897, April 1991.
3. Meystel, A. "Representation of Descriptive Knowledge for Nested Hierarchical Controllers," Proceedings of the IEEE 27th Conference on Decision and Control, Austin, Texas, December 1988, pp. 1805-1811.
4. Donmez, M. A., "Progress Report of the Quality in Automation Project for FY90," National Institute of Standards and Technology Internal Report 4536, March 1991.
5. Gavin, R.J., and Yee, K.W., "Implementing Error Compensation on Machine Tools," Southern Manufacturing Technology Conference Proceedings (Sponsored by NMTBA, The Association for Manufacturing Technology), Charlotte, NC, May 1989.

6. Donmez, M. A., Blomquist, D., Hocken, R., Liu, C., and Barash, M., “A General Methodology for Machine Tool Accuracy Enhancement by Error Compensation,” Precision Engineering, Publication No. 0141-6359/86/040187-10, 1986.