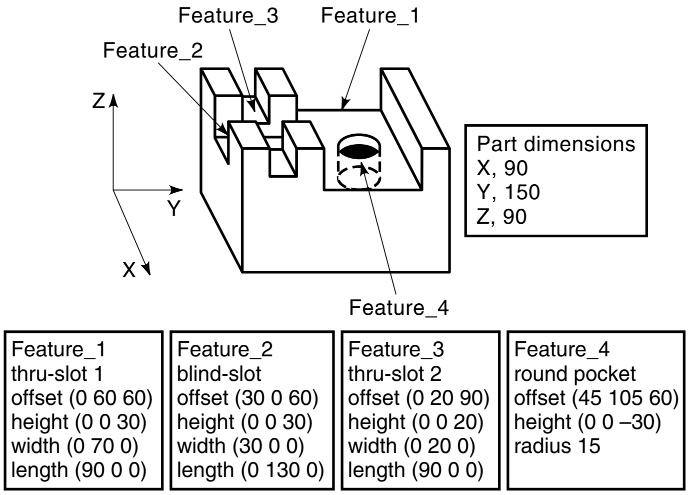
COMPUTER-AIDED PRODUCTION PLANNING

The goal of any simulation methodology in a given domain is to achieve, as close as possible, the same output as the real system for every input in the domain. It can easily be seen that the more complex the domain, the more complex the models in the methodology. One area where the models representing the real system must contain a high level of complexity is the simulation of manufacturing process plans.

One of the key characteristics of manufacturing process plans is that during plan generation, the planner typically uses high-level inferencing and generalities to produce the plan (1,2). For this reason, the actual outcome of the process plan is only known to a high degree of abstraction. An automatic planner does not take into account all possible details. Therefore there is a need to simulate process plans; to verify the process outcomes in detail and how one process/resource affects later processes/resources.



Feature_1	Feature_2	Feature_3	Feature_4
thru-slot 1	blind-slot	thru-slot 2	round pocket
offset (0 60 60)	offset (30 0 60)	offset (0 20 90)	offset (45 105 60)
height (0 0 30)	height (0 0 30)	height (0 0 20)	height (0 0 –30)
width (0 70 0)	width (30 0 0)	width (0 20 0)	radius 15
length (90 0 0)	length (0 130 0)	length (90 0 0)	

Figure 1. Example part used for discrete part manufacturing examples throughout this article.

The goal of this research is to present a methodology for the simulation of manufacturing process plans and for the automatic creation of computational models used in the simulation of a process plan. It is assumed that we are given a process plan in an intermediate representation to be used by the combined discrete/continuous methodology presented here. In the methodology, a process plan consists of a directed graph with the nodes as either intermediate process descriptions, or logical branch place-holders. The information in this article is the information produced by typical process planners (1,2); therefore, it can be assumed to be available. A sample part used to test this simulation methodology is illustrated in Fig. 1. The corresponding process plan representation for this part is shown in Fig. 2.

Given a process plan representation such as that in Fig. 2, one would like to produce the computational models for the simulation, including the qualitative component similar to that given in Fig. 4 and the dynamic models similar to those in Appendix 1. With these models, the dynamic attributes of each process can be simulated, as well as the qualitative aspects such as process goals.

Related Work and Motivation

There are two areas of research related to the ideas presented in this article: (1) geometric verification of machining operations, and (2) continuous simulation of individual processes.

The first trend in process simulation has been to simulate geometrically machining processes. In previous research in this area, Sungurtekin and Voelcker used volume removal by sweeping motions (3), Saito and Takahashi used G-buffers to perform more complex verification schemes (4), Hsu and Yang used isometric projections to greatly simplify the calculations involved in displaying a three-dimensional representation (5). Suh and Lee developed a four-axis geometric modeling system for use with rotational-free-surfaces (6). Geometric simulation does not address the dynamic aspects of the processes. Therefore, there exists the need for the simulation methodology to incorporate both the dynamic and qualitative aspects of processes.

The simulation of individual continuous processes has been the motivation for many works in model generation (the second group). These works utilize continuous computer models to represent specific aspects of a machining process (such as tool force and tool/material interface temperature), simulating each process in a manufacturing process plan, independently of the other processes in the process plan. Some of the problems previously addressed in this area include the following: the cutting forces in milling by Tarng and Chang (7), end milling by Kolarits and DeVries (8), and the cutting forces in drilling by Chandrasekharan et al. (9). Oxley extended orthogonal machining theory (10), Stephenson and Wu created models for turning and drilling (11), and Polisetty developed an integrated system for continuous simulation of NC code (12). However, simulation of individual processes leads to the problem of not accounting for the effects one process has on others within the same manufacturing process plan.

If a tool is instantiated for process #1, and is left in a condition that is less than “Good,” it may obviously have an undesirable effect on process #2. Therefore, it would be beneficial to instantiate the resources (tools) right before they are needed, in order to utilize the most recent information about them. Such limitations indicate the need for automatic model generation and resource instantiation.

This article can be summarized in the following:

- *Method for Combined Discrete / Continuous Simulation of Process Plans.* The approach developed here encompasses both the dynamic and qualitative aspects (such as postconditions that hold after a process has executed) of processes. It dynamically generates process simulation models as the simulation progresses and updates the knowledge base as the simulation results become available. The process simulation models, in this work, capture both the dynamic and qualitative aspects of the processes and the resources such as tools.
- *Method for Automatic Generation of Computational Simulation Models for Process Plans.* Based on models that capture processes, and resources such as tools in a modular fashion, this mechanism constructs computational simulation models automatically by:
  - Automatically creating active, stopping, and post conditions for the qualitative components of the process simulation models
  - Automatically instantiating and coupling the constants, variables, and port definitions for the dynamic components of the process simulation models.

Solution Overview

To accomplish combined discrete/continuous simulation of process plans, the steps of the simulation methodology shown in Fig. 5 are performed. The input to the process plan simulation algorithm is the process plan graph and the task goals list. The process plan simulation algorithm traverses the process plan graph node by node. As stated earlier, the process plan is a directed graph where the nodes are either intermediate process descriptions, or are logical branch place-holders. Every time an intermediate process description node is encountered, the individual process simulation algorithm is executed and the given process specified in that node is simulated. During the simulation of the individual process, the

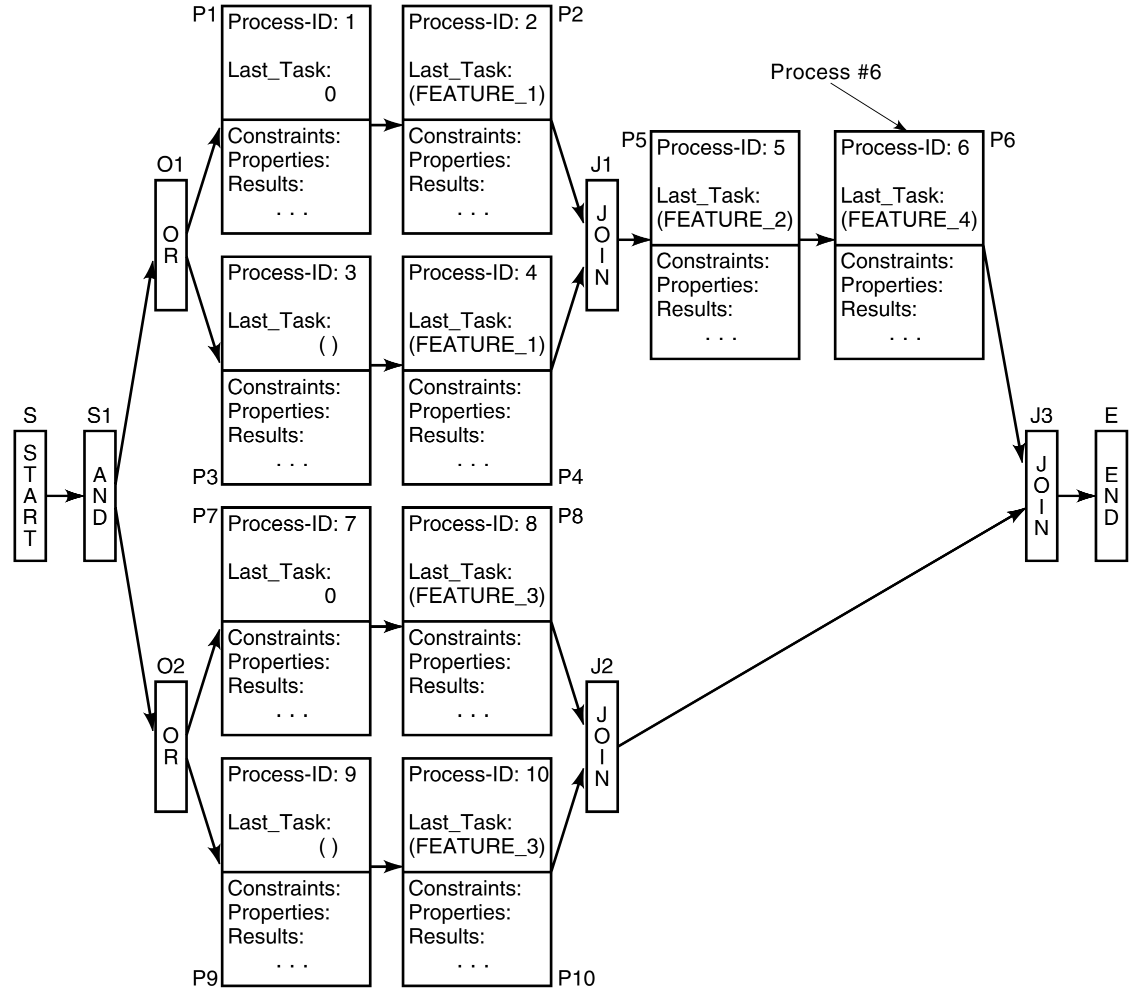


Figure 2. Process plan utilized throughout the examples in this article. This process plan represents the part given in Fig. 1. The process nodes represent intermediate process descriptions, such as that shown in Fig. 3.

process simulation model (PSM) is created using the model base of continuous descriptions and elements of the dynamically updated knowledge base.

As Fig. 6 shows, the PSM is composed of both a qualitative and a dynamic component. These represent the qualitative and the dynamic properties of a process, respectively. The dynamic component is further decomposed into a process model which delineates the dynamic properties of the process itself (e.g., force and temperature of a cutting operation) and a set of resource models that represent dynamic properties of resources active during the process (e.g., as the spindle vibration of the tool used in a cutting process). The process models as well as the resource models each utilize continuous descriptions with differential equations representating continuous properties of these models.

The dynamic and the qualitative properties of each process are then simulated using the PSM in a segment-by-segment fashion. The simulation of the individual processes, segment by segment, results in the generation and addition to the knowledge base of the knowledge (in the form of knowledge elements) describing how the execution of the process changes the environment.

After the simulation methodology is introduced, we will show the method for automatic generation of computational simulation models for process plans will be shown. To achieve this, we will show how the process simulation model for each input intermediate process description within the process plan can be automatically generated. (See the automatic model generation box in Fig. 5). The first step is to generate the qualitative component (qc) by matching resource requirements with what resources are available within the environment. The knowledge base is utilized during resource assignment by inspecting the conditions of already used resources and assigning new resource instances as necessary. In generating the qualitative components for the processes, the sets of constraints are also created, to determine when the simulation of the process should end, and what is known about the process both during its simulation and after the simulation finishes. Together these elements comprise the qualitative knowledge about the process. The second step in generating the computational simulation models involves automatically instantiating and coupling the models that represent the dynamic component (dc) of each process in the manufacturing process plan. The model base of continuous descriptions is uti-

```
Process-ID: 6
Last-Task:  (FEATURE_4)
Constraints:
           (FEATURE_4-Env-Cond DIM_TOL VAR-RANGE (<= 0.01))
           (FEATURE_4-Env-Cond REL_TOL VAR-RANGE (<= 0.01))
           (FEATURE_4-Env-Cond IND_TOL VAR-RANGE (<= 0.01))
           (FEATURE_4-Env-Cond FINISH VAR-RANGE (<= 210))
           (TYPE-TWIST_DRILLING-Pro-Alloc PROCESS_6 (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR
                      ACTIVE)-START-COND (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR FINISHED)-
                      END-COND)
           (TYPE-CONICAL_POINT_TWIST_DRILL-Res-Alloc PROCESS_6 (PROCESS_6-Qual-Cond STATUS
                      DISCRETE-VAR ACTIVE)-START-COND (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR
                      FINISHED)-END-COND)
           (TYPE-CONICAL_POINT_TWIST_DRILL-Res-Cond CONDITION DISCRETE-VAR GOOD)
           (TYPE-CONICAL_POINT_TWIST_DRILL-Res-Cond HARDNESS VAR-RANGE (>= 430))
Properties:
           (TYPE-CONICAL_POINT_TWIST_DRILL-Res-Cond VELOCITY VAR-RANGE (>= 0.2, <= 0.4))
           (TYPE-CONICAL_POINT_TWIST_DRILL-Res-Cond ROT-VELOCITY VAR-RANGE (>= 500, <= 600))
           (TYPE-CONICAL_POINT_TWIST_DRILL-Res-Cond DRILL_DIAMETER VAR-RANGE (>= 15, <= 25))
           (FEATURE_4-Env-Cond CLAMP_AXIS VAR (0 1 0))
           (FEATURE_4-Env-Cond APPROACH VAR (0 0 1))
Results:
           (FEATURE_4-Env-Cond DIM_TOL VAR 0.004)
           (FEATURE_4-Env-Cond REL_TOL VAR 0.007)
           (FEATURE_4-Env-Cond IND_TOL VAR 0.006)
           (FEATURE_4-Env-Cond FINISH VAR 210)
```

Figure 3. Input intermediate process description for process #6. Each intermediate process description consists of a set of constraints, properties, and results, along with information stating what tasks the process completes.

lized to determine what parameters require instantiation within the dynamic component.

KNOWLEDGE REPRESENTATION

The first step in presenting the simulation methodology is defining a simulation model capable of representing both the dynamic and qualitative properties of a process. The following sections describe the knowledge representation utilized throughout the simulation methodology. First the representation of a process plan is defined, followed by an explanation of knowledge elements and their role within the approach. Afterwards, the process simulation model is defined.

Process Plan

A process plan (PP) can be defined as follows:

A process plan is a directed graph, PP = $\langle V, \Sigma \rangle$, with the following properties:

- V is the set of nodes, $V = \{v_1, v_2, \ldots, v_n\}$, where each vertex is one of six types; $v_i = \{START \mid END \mid \text{ipd} \mid OR \mid AND \mid JOIN\}$ with each defined as follows:
  - *START, END, OR, AND, JOIN* are nodes used for branching and place holding.
  - Ipd nodes, are intermediate process description nodes. They provide parameters for each of the processes in the process plan. Ipd nodes are defined as a 5-tuple, ipd = $\langle id, LAST\_TASK, CONS, PROP, RES \rangle$.
- Σ is the set of directed edges, $\Sigma = \{\partial_1, \partial_2, \ldots, \partial_m\}$, where each edge is represented by an ordered pair, $\partial_i = \langle v_s, v_e \rangle$, where v_s and v_e are nodes.

The process plan of Fig. 2 represents the machining operations, and their ordering, to create the part given in Fig. 1. Note the use of directed edges to represent the flow from the "Start" node to the "End" node. Each node within the directed graph of Fig. 2 is either an intermediate process description that gives the input parameters of a process, or a logical branch place-holders. In Fig. 2, two primary paths exist, emanating from the "AND" node, with subpaths on both sides emanating from "OR" nodes. The "AND" node represents the necessity to traverse all paths emanating from it in any order, whereas the "OR" node only necessitates traversal of one branch. From Fig. 2, if the top branch exiting the "AND" node is traversed first, then the choice of either using processes 1 and 2, or processes 3 and 4, is acceptable for the simulation of the generation of the first Thru-slot (Feature_1 in Fig. 1). The rest of Fig. 2 can be interpreted in a similar manner.

As already stated, one of the two categories of nodes within the process plan representation is the intermediate process model used to encapsulate information about each process node. Figure 3 shows an example intermediate process model for process #6 that contains five elements. The first two elements are the identification number, and the list of tasks the process completes (i.e., the completion of Feature_4 in Fig. 1). The constraints of the ipd contain all resource and process model allocations (i.e., allocation of a drill resource model, and twist-drilling process model) along with conditions that must remain satisfied by the environment and the allocated

```
Process-ID: 6
Active-Conditions:
          (CPTD_TOOL_1-Res-Cond STATUS RULE
                    ((if (PROCESS_6-Pro-Cond STATUS DISCRETE-VAR ACTIVE) then
                      (CPTD_TOOL_1-Res-Cond STATUS DISCRETE-VAR ACTIVE))
                    (if (PROCESS_6-Pro-Cond STATUS DISCRETE-VAR FINISHED) then
                      (CPTD_TOOL_1-Rec-Cond STATUS DISCRETE-VAR FINISHED))
                    ))
          (TD_1-MODEL-Res-Cond STATUS RULE
                    ((if (PROCESS_6-Pro-Cond STATUS DISCRETE-VAR ACTIVE) then
                      (TD_1-MODEL-Res-Cond STATUS DISCRETE-VAR ACTIVE))
                    (if (PROCESS_6-Pro-Cond STATUS DISCRETE-VAR FINISHED) then
                      (TD_1-MODEL-Rec-Cond STATUS DISCRETE-VAR FINISHED))
                    ))
          (CPTD_TOOL_1-Res-Cond CONDITION DISCRETE-VAR GOOD)
          (CPTD_TOOL_1-Res-Cond HARDNESS VAR-RANGE (>= 430))
          (CPTD_TOOL_1-Res-Cond VELOCITY VAR-RANGE (>= 0.2, <= 0.4))
          (CPTD_TOOL_1-Res-Cond ROT-VELOCITY VAR-RANGE (>= 500, <= 600))
          (CPTD_TOOL_1-Res-Cond DRILL_DIAMETER VAR-RANGE (>= 15, <= 25))
          (FEATURE_4-ENV-COND CLAMP_AXIS VAR (0 1 0))
          (FEATURE_4-END-COND APPROACH VAR (0 0 1))
Stopping-Conditions:
          (* (FEATURE_4-Env-Cond DIM_TOL VAR 0.004)
          (FEATURE_4-Env-Cond REL_TOL VAR 0.007)
          (FEATURE_4-Env-Cond IND_TOL VAR 0.006)
          (FEATURE_4-Env-Cond FINISH VAR 210) *)
          (* (FEATURE_4-Env-Cond DIM_TOL VAR-RANGE (<= 0.008))
          (FEATURE_4-Env-Cond REL_TOL VAR-RANGE (<= 0.008))
          (FEATURE_4-Env-Cond IND_TOL VAR-RANGE (<= 0.008))
          (FEATURE_4-Env-Cond FINISH VAR-RANGE (<= 213))
          (FEATURE_4-Env-Cond DIMENSION_OFFSET VAR (45 105 60))
          (FEATURE_4-Env-Cond DIMENSION_HEIGHT VAR (0 0 −30))
          (FEATURE_4-Env-Cond DIMENSION_RADIUS VAR 15) *)
          (* (TD_1-MODEL-Res-Cond STATUS DISCRETE-VAR FINISHED) *)
Post-Conditions:
          (FEATURE_4-Env-Cond DIMENSION OFFSET VAR (45, 105, 60))
          (FEATURE_4-Env-Cond DIMENSION_HEIGHT VAR (0 0 −30))
          (FEATURE_4-Env-Cond DIMENSION_RADIUS VAR 15)
          (FEATURE_5-Env-Cond DIM_TOL VAR 0.004)
          (FEATURE_4-Env-Cond REL_TOL VAR 0.007)
          (FEATURE_4-Env-Cond IND_TOL VAR 0.006)
          (FEATURE_4-Env-Cond FINISH VAR 210)
```

Figure 4. Example qualitative component. Note the use of "(*" and "*)" to group subsets of stopping-conditions.

entities. The properties give a set of process parameters that are valid while the process is active, and the results give a set of process parameters that are valid once the execution of the process completes successfully.

Knowledge Elements

A knowledge element is the smallest piece in the representation of information within the knowledge base and the process simulation models. Knowledge elements are used to represent process and environmental parameters. Four types of knowledge elements are used:

1. *Environmental Conditions.* The syntax of each is:

   (*Name*-Env-Cond *Attribute*
   (VAR[-RANGE] | DISCRETE-VAR[-RANGE]) *Value*)

2. *Qualitative Process Conditions.* The syntax of each is:

   (*Process-ID*-Qual-Cond *Attribute*
   DISCRETE-VAR[-RANGE] *Value*)

3. *Dynamic Process / Resource Model Conditions.* The syntax of each is:

   ([TYPE-]*Name*-(Pro | Res)-Cond *Attribute*
   (VAR[-RANGE] | DISCRETE-VAR[-RANGE] |
   RULE) *Value*)

4. *Dynamic Process / Resource Model Allocations.* The syntax of each is:

   ([TYPE-]*Name*-(Pro | Res)-Alloc *Process-ID*
   (*cond*)-START-COND (*cond*)-END-COND)

For example, an environmental condition that specifies the clamping axis for the feature, Feature_2, to be (0 1 0) (that is, in the *y*-axis direction) is the following: (FEATURE_2-Env-Cond CLAMP_AXIS VAR (0 1 0)).

Process Simulation Model

The process simulation model (PSM) is the representation of a process used by the simulation procedures within the simu-

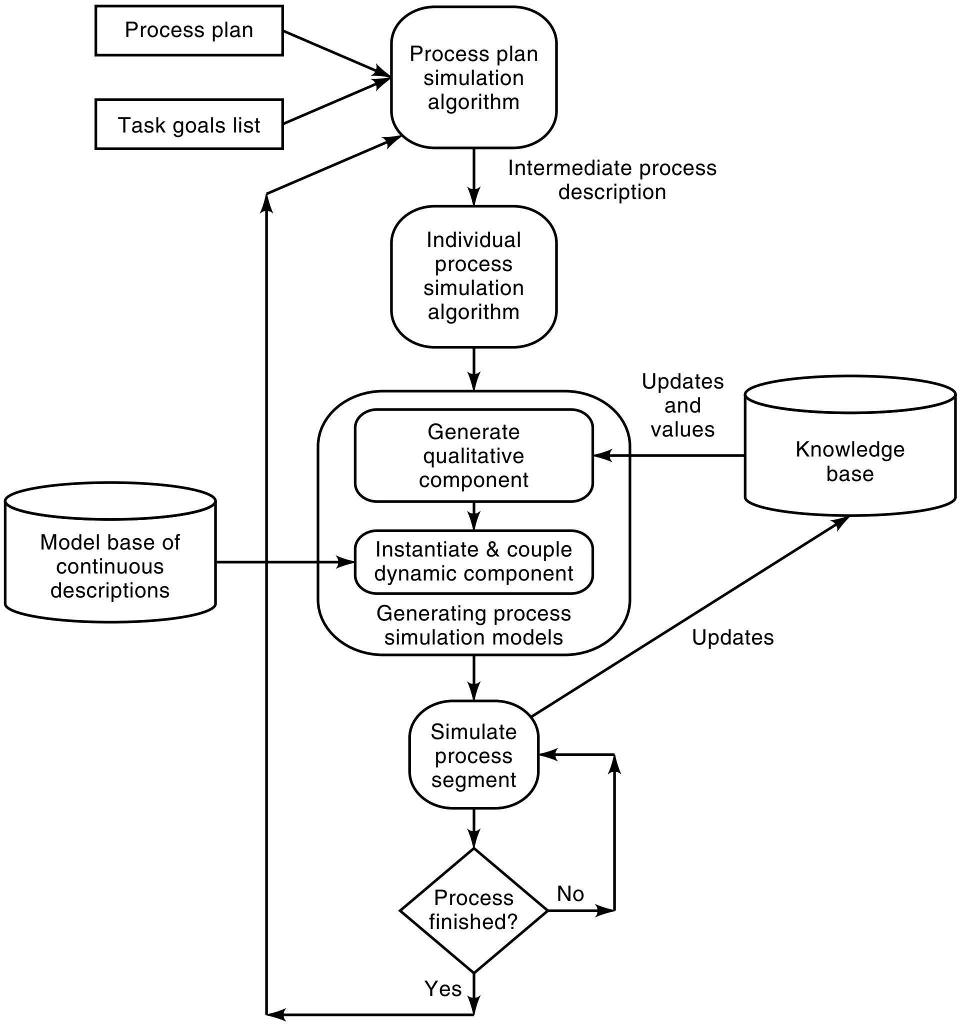

Process plan
Task goals list
Process plan simulation algorithm
Intermediate process description
Individual process simulation algorithm
Updates and values
Knowledge base
Generate qualitative component
Model base of continuous descriptions
Instantiate & couple dynamic component
Generating process simulation models
Updates
Simulate process segment
Process finished?
No
Yes

Figure 5. Diagram of major phases in the simulation of process plans.

lation methodology. The structure of the PSM is shown in Fig. 6. The process simulation model of a process is a double, PSM = ⟨qc, dc⟩, where qc is the qualitative component, and dc the dynamic component. The dynamic component is composed of the dynamic process model (dpm) and a set of dynamic resource models (DRM), dc = ⟨dpm, DRM⟩, with DRM = {drm_1, drm_2, . . ., drm_n}, where each drm_i is an individual dynamic resource model.

Qualitative Component. The role of the qc within the PSM is to determine when the process should "stop" and the information that is known about the process during its execution and after its execution is complete. The qualitative component (qc) represents the qualitative aspects of the process (13). The qualitative component is a 4-tuple, qc = ⟨*id, ACT, POST, STOP*⟩, with the following properties:

- *ACT* is the set of active-conditions, *ACT* = {act_1, act_2, . . ., act_m}, where each act_i is a knowledge element.
- *POST* is the set of postconditions, *POST* = {$post_1$, $post_2$, . . ., $post_n$}, each $post_j$ is a knowledge element.
- *STOP* is the set of stop-groups, *STOP* = {$sgrp_1$, $sgrp_2$, . . ., $sgrp_o$}, where each $sgrp_k$ describes one or more stopping-conditions using a set of knowledge elements. Therefore, $sgrp_k$ = {$stop_1$, $stop_2$, . . ., $stop_q$}, where each $stop_l$ a knowledge element.

Active-conditions are elements that are only true when the given process is active. They are added to the knowledge base when the process is started. The second set of conditions,

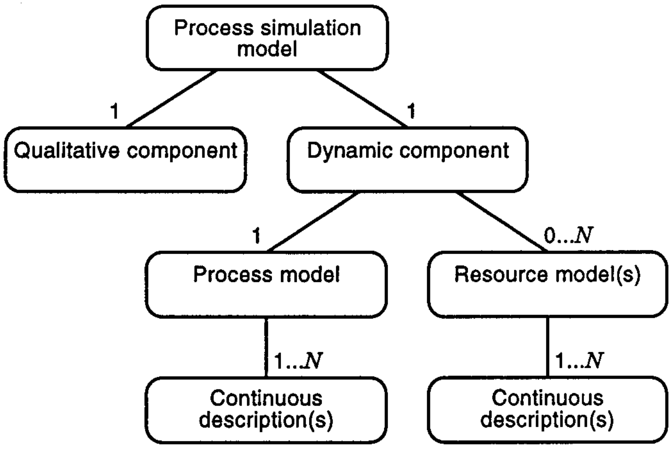


Figure 6. Diagram illustrating the composition of the process simulation model.

postconditions, are added to the knowledge base at the completion of the process; that is, when one of the stopping-conditions becomes true. The last set, the stopping-conditions, are grouped into subsets which correspond to either the task or process goals being met, or correspond to the end of process condition being detected by the dynamic process model. Figure 4 shows an example qualitative component for the process #6 of the process plan shown in Fig. 2. The qualitative component is generated from the corresponding ipd in the process plan (Fig. 3 shows the corresponding ipd for this example). In Fig. 4, the active-conditions contain rules for the allocation of both a twist drilling resource model and a twist drilling process model, along with parameters such as the rotational velocity of the drill bit. The postconditions give the state of the environment (e.g., the dimensions of the completed feature, as shown in Fig. 1) when the process successfully completes execution.

Dynamic Component. The dynamic component of a process simulation model is used to represent the dynamic aspects of both the process itself, and the resources utilized by the process, such as the force at the tip of a twist drill, and the thickness of the chip of removed material. The dynamic process model (dpm) is an augmented DEV & DESS (14) model for representing the process. The dpm is an augmented DEV & DESS combined discrete/continuous simulation model defined by the 13-tuple, dpm = $\langle id, C, V, Pt, S, DISCONV, CMOD, SEGEND, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta, \Delta T\rangle$ with the following properties [for a more detailed description, see (15)]:

- C is the set of constants, $C = \{const_1, const_2, \ldots, const_f\}$
- V is the set of variables, $V = \{\text{var}_1, \text{var}_2, \ldots, \text{var}_g\}$, each var_b is a discrete or continuous valued variable
- Pt is the set of ports for dpm, $Pt = \{pt_1, pt_2, \ldots, pt_h\}$, where each port is a triple $pt_c = \langle Port\text{-var\_name attached\_to IN} \mid \text{OUT} \mid \text{IN/OUT}\rangle$, var_name $\in V$, and attached_to is the model to which the port is attached.
- *SEGEND* is an end of segment indicator. It is a function that returns true if the current value of variables indicate the end of the current process segment has been reached. Process segments are a domain specific division of a process into portions that can be simulated individually.
- ΔT is the delta-T polling function. It sends the attached continuous descriptions in *CMOD* their required input, stores their output into variables and checks for the end of segment. Formally, ΔT is defined by,

$$\Delta T = \left\{ \begin{array}{l} (V \cup C \cup Pt) \times CMOD \rightarrow V \\ if\,(SEGEND)\,then \\ current\_state \leftarrow INTERRUPT, INTERRUPT \in S \end{array} \right\}$$

An example dynamic process model for twist drilling is shown in Appendix 1. This model represents the dynamic properties of the twist drilling process (9). It is used to interface with continuous descriptions of the force at the tip of the drill and with the end of segment indicator. The sets of constants, variables, and port definitions are encapsulated in the top of the model. They are the only elements that are dynamically assigned during the model generation. All dynamic process models contain the same internal transition, external transition, and output functions. The main goal of these three functions is to interface with the continuous descriptions and other dynamic models (utilizing the generated set of ports), and perform the iterations for simulating processes segment by segment. The delta-T polling function for this example polls the continuous force data and end of segment indicator, stores their output in variables, and determines if the end of the current segment has been reached.

A dynamic resource model (drm) is very similar to the dynamic process model in that it is also an augmented DEV & DESS (14) discrete/continuous simulation model, except it does not contain a *SEGEND* indicator: each dpm is associated with one or more drms, just as each process uses one or more resources or tools to achieve its task. The dpms and drms are both allocated when their associated rule within the knowledge base is fired. The drm represents the dynamic behavior of a resource accomplishing the task in the process. An example in discrete part manufacturing is the following scenario: given a machining process (end-milling) the qc determines major discrete changes, and the dpm represents the force the process is creating, and determines when process segments end. There may be several drms, one to represent the tool which includes a continuous description of the condition of the tool, and another to represent the vibration of the spindle with discrete conversion of the vibration into a quantity representing the quality of the cutting operation [for an example process utilizing more than one drm and for examples defining the drm, see (15)].

SIMULATION METHODOLOGY

There are four major components within the simulation methodology during the simulation of a process plan: (1) the PSM in the form of the qc and dc, (2) the model base of continuous descriptions (MB), (3) the knowledge base (KB), and (4) the simulation administrator (SA). Figure 7 illustrates the four major components, and how each interacts with the others to perform the simulation. The qualitative component contains conditions used in the simulation of an individual process, such as active-conditions that represent constraints on the parameters for the currently active process. The model base of continuous descriptions is used to instantiate dynamic models to represent continuous aspects of the process currently being simulated. The knowledge base contains tables used for variable instantiations, a repository of dynamic properties of the currently active process, and a stack of results from the executions of previous processes. The simulation administrator controls simulation of individual processes utilizing the qualitative and dynamic model components. The simulation administrator maintains control of the simulation of the process plan using the information within the knowledge base.

Knowledge Base

The knowledge base consists of the knowledge stack (KS), dynamic knowledge (DK), and table of types, values, and attributes (TVA). The knowledge stack maintains a record of changes to the environment caused by the execution of each process. The dynamic knowledge captures the current state of the active process, and the TVA table provides a static listing of available resources and their attributes.

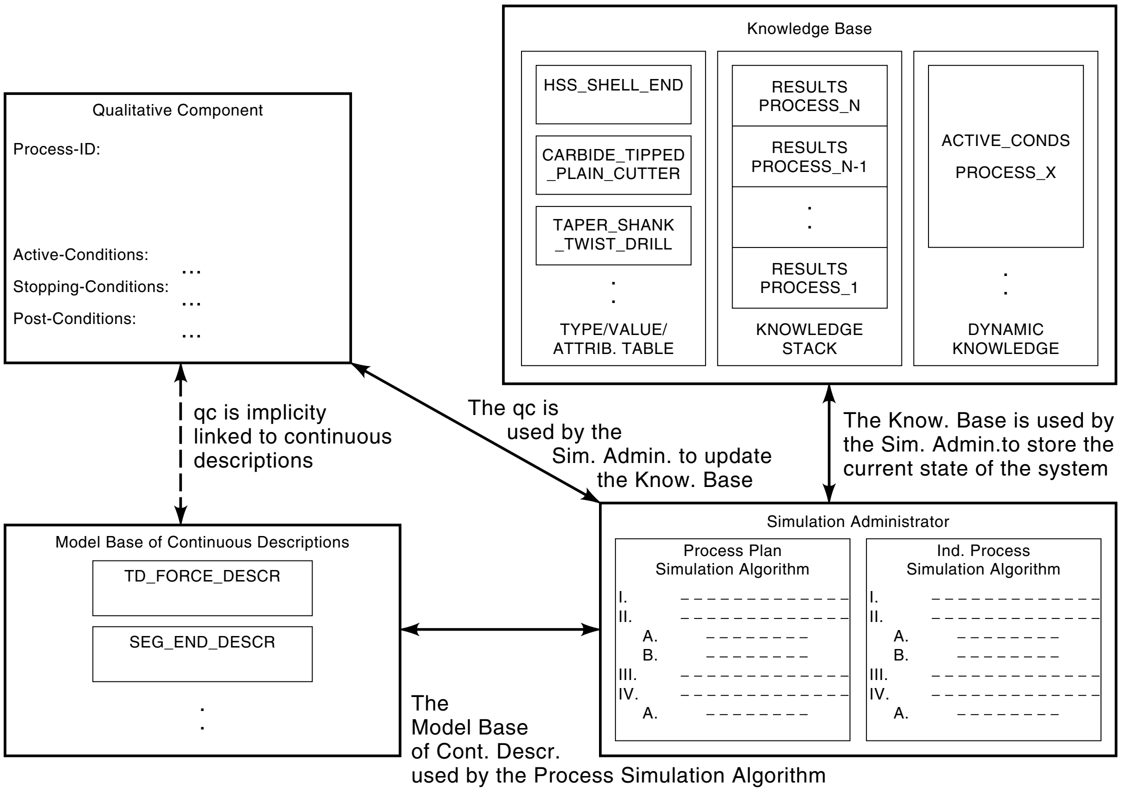

Knowledge Base
HSS_SHELL_END
CARBIDE_TIPPED _PLAIN_CUTTER
TAPER_SHANK _TWIST_DRILL
TYPE/VALUE/ ATTRIB. TABLE
RESULTS PROCESS_N
RESULTS PROCESS_N-1
RESULTS PROCESS_1
KNOWLEDGE STACK
ACTIVE_CONDS
PROCESS_X
DYNAMIC KNOWLEDGE
Qualitative Component
Process-ID:
Active-Conditions:
Stopping-Conditions:
Post-Conditions:
qc is implicity linked to continuous descriptions
The qc is used by the Sim. Admin. to update the Know. Base
The Know. Base is used by the Sim. Admin.to store the current state of the system
Model Base of Continuous Descriptions
TD_FORCE_DESCR
SEG_END_DESCR
Simulation Administrator
Process Plan Simulation Algorithm
Ind. Process Simulation Algorithm
I.
II.
A.
B.
III.
IV.
A.
The Model Base of Cont. Descr. used by the Process Simulation Algorithm

Figure 7. Major components and how they interact within the simulation methodology. Note that the simulation administrator is the only entity that has direct contact with all of the other entities.

The knowledge base is defined as KB = ⟨KS, DK, TVA⟩ a triple, with each of the components defined in the following (examples for each are shown in the sections that follow; for a more detailed definition, see Ref. 15):

- The knowledge stack (KS) is an ordered list of knowledge elements. Knowledge elements are added to the knowledge stack every time a process starts or stops. At the end of the simulation of a process plan, the knowledge stack contains a history of the results of all executed processes.
- The dynamic knowledge (DK) is a set of knowledge elements containing the current values of variables which represent the state of the current process and the rules which can be fired while the process is active. At the end of simulating a process segment, all variables within the dynamic knowledge are updated with results from continuous equations using the dynamic process and resource models.
- The type, value, and attributes table (TVA) is a table used in “type” variable instantiation. This table contains entries for all resources available for use in the current environment, the attributes of each resource, and the default value for each attribute.

Simulation of Individual Process

The individual process simulation algorithm simulates an individual process utilizing both qualitative and dynamic components of the process. Figure 8 shows the individual process simulation algorithm. This algorithm interacts with the process simulation model and the knowledge base. Simulation proceeds in a segment-by-segment fashion, as illustrated in the example that follows.

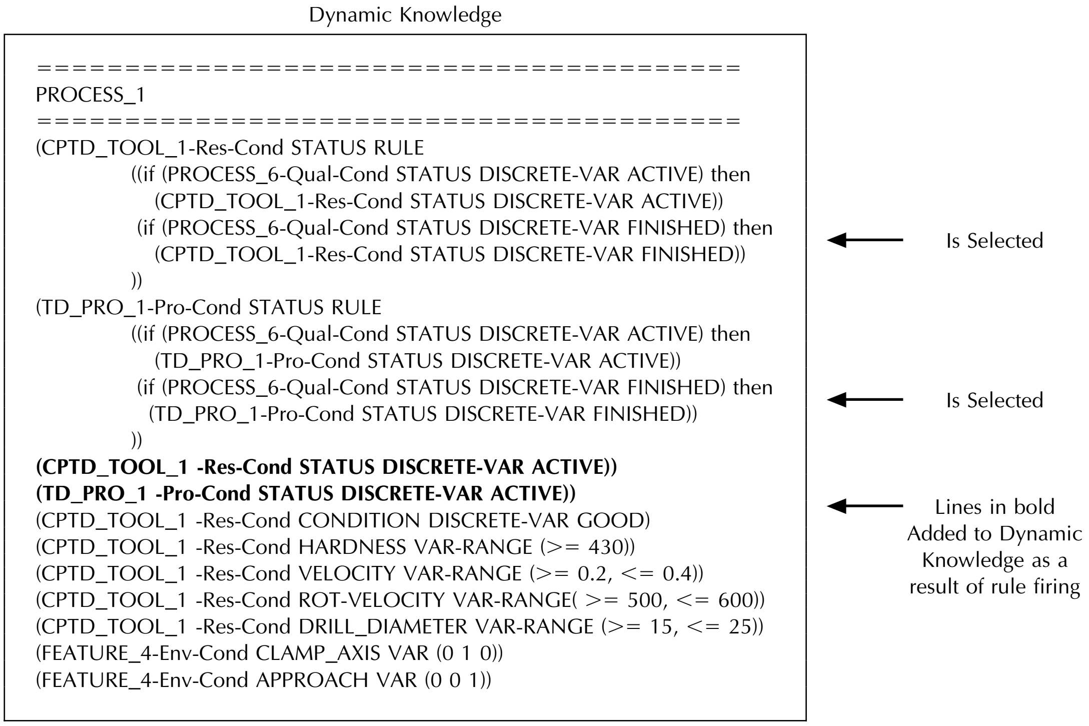
As an example, assume we have just finished process #5, and the next step is the simulation of process #6. The first two steps of the individual process simulation algorithm are the creation of the qc, which is given in Fig. 4, and the addition of the active-conditions from this qc to the dynamic knowledge in the knowledge base (KB). Afterwards, the rules within the dynamic knowledge are evaluated. The result of this evaluation indicates to use the two dynamic models as indicated by the two knowledge elements which are added to the knowledge base as shown in bold type in Fig. 9. The next two steps entail generating the constants, variables, and port definitions for the dynamic models (given in Appendix 1) which are active, and sending the “init” message (containing the active-conditions) to these dynamic models. Figure 10 illustrates the state diagram of the dpm for steps IV and V. Both of the dynamic models are now in the “waiting-to-start” state.

The next two steps update the knowledge stack with the dynamic knowledge and begin the simulation of the dynamic models. At the point when the “iterate” message is sent to the dynamic models, control is passed to the dynamic process model (TD_PRO_1 in Appendix 1). At this point, each of the dynamic models progresses to the “segment-simulation” state and enters a loop to poll continuous descriptions attached to

```
Input:  Uninstantiated Intermediate Process Description ( ipd )
Output:  Update to knowledge Stack representing the simulation of a process

I.      P ← Generate_QC(ipd)
II.     Dynamic_Knowledge ← Dynamic_Knowledge + Active_Conditions(P)
III.    Evaluate_Rules(Dynamic_Knowledge)
IV.     Instantiate_All_Ports_And_Vars(dpm  and DRM)
V.      send.''init'' message to dpm and DRM
VI.     Push(Dynamic_Knowldge, Knowledge_Stack)
VII.    send ''iterate'' message to dpm and DRM
VIII.   finished ← false
IX.     while ( not(finished) ) do {
            A.      wait for response from dpm and Update(Dynamic_Knowledge) - Control returns to S.A.
            B.      Evaluate_Rules(Dynamic_Knowledge)
            C.      if ( true(Stopping-Conditions(P)) ) then {
                        1.      send ''end-process'' message to DRM
                        2.      wait for responses and Update(Dynamic_Knowledge)
                        3.      Push(Post-Conditions(P), Knowledge_Stack)
                        4.      Push(Dynamic_Knowledge, Knowledge_Stack)
                        5.      finished ← true
            D.      } /* end if */
            E.      else {
                        1.      send ''end-segment'' message to DRM
                        2.      wait for responses and Update(Dynamic_Knowledge)
                        3.      send ''iterate'' message to dpm and DRM - Control goes to dpm
            F.      } /* end else */
X.      } /* end while */
XI.     send ''finish'' message to dpm and DRM
XII.    Return Success
```

Figure 8. Individual process simulation algorithm.



Dynamic Knowledge

```
========================================
PROCESS_1
========================================
(CPTD_TOOL_1-Res-Cond STATUS RULE
        ((if (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR ACTIVE) then
           (CPTD_TOOL_1-Res-Cond STATUS DISCRETE-VAR ACTIVE))
         (if (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR FINISHED) then
           (CPTD_TOOL_1-Res-Cond STATUS DISCRETE-VAR FINISHED))
        ))
(TD_PRO_1-Pro-Cond STATUS RULE
        ((if (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR ACTIVE) then
           (TD_PRO_1-Pro-Cond STATUS DISCRETE-VAR ACTIVE))
         (if (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR FINISHED) then
           (TD_PRO_1-Pro-Cond STATUS DISCRETE-VAR FINISHED))
        ))
(CPTD_TOOL_1 -Res-Cond STATUS DISCRETE-VAR ACTIVE))
(TD_PRO_1 -Pro-Cond STATUS DISCRETE-VAR ACTIVE))
(CPTD_TOOL_1 -Res-Cond CONDITION DISCRETE-VAR GOOD)
(CPTD_TOOL_1 -Res-Cond HARDNESS VAR-RANGE (>= 430))
(CPTD_TOOL_1 -Res-Cond VELOCITY VAR-RANGE (>= 0.2, <= 0.4))
(CPTD_TOOL_1 -Res-Cond ROT-VELOCITY VAR-RANGE( >= 500, <= 600))
(CPTD_TOOL_1 -Res-Cond DRILL_DIAMETER VAR-RANGE (>= 15, <= 25))
(FEATURE_4-Env-Cond CLAMP_AXIS VAR (0 1 0))
(FEATURE_4-Env-Cond APPROACH VAR (0 0 1))
```

Is Selected

Is Selected

Lines in bold Added to Dynamic Knowledge as a result of rule firing

Figure 9. Dynamic knowledge after the addition of the active-conditions for process #6. Here the rule elements that will be selected are shown. These rules will be fired because of knowledge elements added to the knowledge stack corresponding to process #6 becoming active.

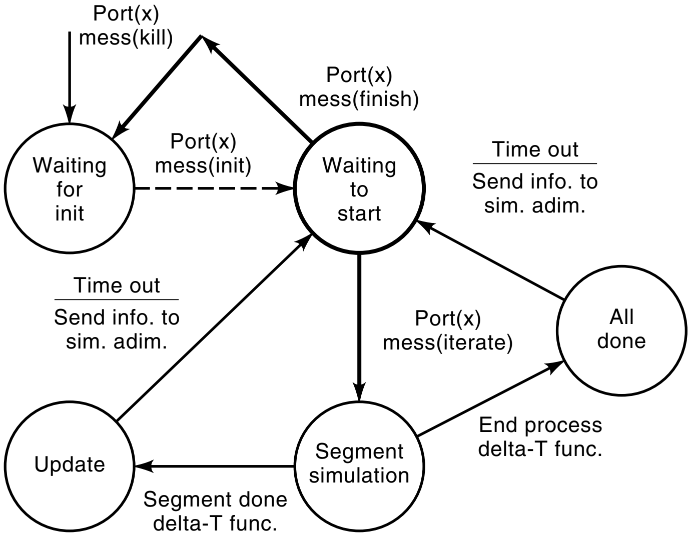


Figure 10. State diagram illustrating steps IV and V in the individual process simulation algorithm. After the “init” message is received, all dynamic models enter the “waiting to start” state. In this and subsequent state diagrams, the dark circle represents the current state, the heavy dashed arrow represents the path taken to the state, and the heavy solid arrows indicate what states can be entered next.

the models; as shown in Fig. 11. Of particular interest is the end of segment indicator attached to the dpm, which is responsible for determining when the current process completes.

The table in Fig. 12 presents the initial values of some of the key variables within the dpm; the output of the two attached continuous descriptions, and end of segment. Every time a polling takes place (according to the delta-T polling function contained within each dynamic model), all of the needed input values are collected and sent to the appropriate description according to the *IN_PARAM* set. The output is then stored in the corresponding variable by using the *OUT_PARAM* set and transferring the values via the IN/OUT port. This continues until the end of segment is reached in the dpm.

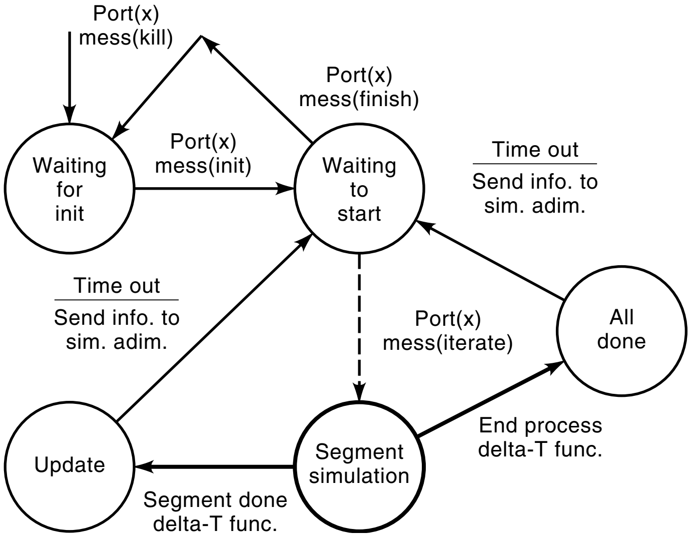


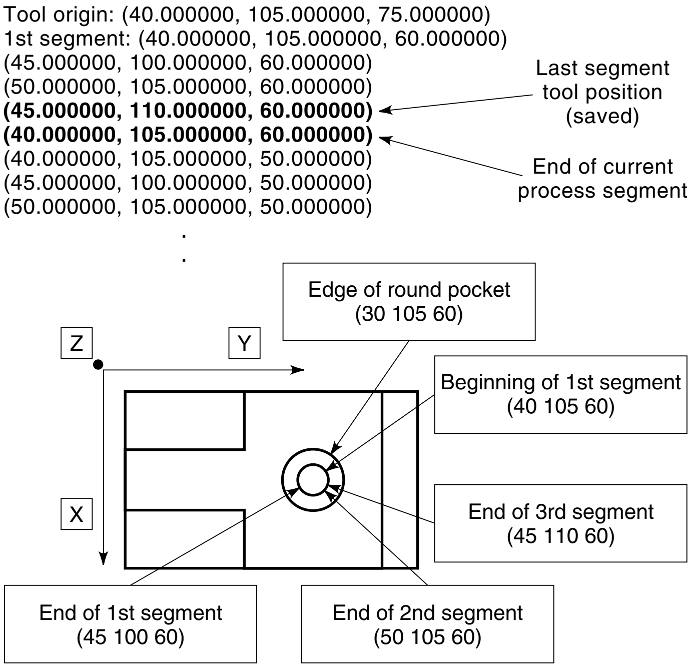
Figure 11. State diagram for the dpm after the “iterate” message is received.

Time	Force (N)	tool_pos	End of Seg?
0	0	(40.0, 105.0, 60.0)	no
1 ΔT	1900	(40.7, 104.4, 102.5)	no
2 ΔT	35000	(41.2, 103.8, 102.5)	no
. . .	. . .	. . .	. . .

Figure 12. Sample table of the first three polls for the dynamic process model. This table shows only the time and output from the two attached descriptions for force, tool_pos, and end of segment.

The first time the end of segment description is polled, it creates a set of segment ending conditions; in this example, the ending points of path segments for the tool (as shown in Fig. 13). Let us assume that we have been polling the continuous descriptions attached to the dynamic models, and checking the end of segment. Let us also assume that the current tool position is (41.5, 106.7, 60.0). The end of segment indicator can determine if the end of the current segment has been reached by calculating the distance (within some allowable tolerance) from the current tool position to the next segment ending. In this example, let the allowable difference be 0.5 × 10^{-2} in., which means that the end of the current process segment is not yet reached.

The dynamic models poll the continuous descriptions, record their values, and check for the end of segment until the end of segment is reached. When the end of the process seg-



```
Tool origin: (40.000000, 105.000000, 75.000000)
1st segment: (40.000000, 105.000000, 60.000000)
(45.000000, 100.000000, 60.000000)
(50.000000, 105.000000, 60.000000)
(45.000000, 110.000000, 60.000000)
(40.000000, 105.000000, 60.000000)
(40.000000, 105.000000, 50.000000)
(45.000000, 100.000000, 50.000000)
(50.000000, 105.000000, 50.000000)
.
.
```

Figure 13. Partial listing of segment ending points which is created the first time the end of segment description is called. This example shows the last segment ending position (stored in the model) and the tool position for the end of the current segment. The bottom portion of this figure gives the relative positions of segments with respect to their three-dimensional location on the part being machined (in hundredths of an inch).

```
Stopping-Conditions:
        (*(FEATURE_4-Env-Cond DIM_TOL VAR 0.004)                  }
        (FEATURE_4-Env-Cond REL_TOL VAR 0.007)                    } Stop Group 1
        (FEATURE_4-Env-Cond IND_TOL VAR 0.006)                    }
        (FEATURE_4-Env-Cond FINISH VAR 210)*)                     }
        (*(FEATURE_4-Env-Cond DIM_TOL VAR-RANGE (<= 0.008))               }
        (FEATURE_4-Env-Cond REL_TOL VAR-RANGE (<=0.008))                  }
        (FEATURE_4-Env-Cond IND_TOL VAR-RANGE (<= 0.008))                 }
        (FEATURE_4-Env-Cond FINISH VAR-RANGE (<= 213))                    } Stop Group 2
        (FEATURE_4-Env-Cond DIMENSION_OFFSET VAR (45 105 60))             }
        (FEATURE_4-Env-Cond DIMENSION_HEIGHT VAR (0 0 −30))               }
        (FEATURE_4-Env-Cond DIMENSION_RADIUS VAR 15)*)                    }

        (*(TD_PRO_1-Pro-Cond STATUS DISCRETE-VAR FINISHED)*)                 } Stop Group 3
```

Figure 14. Stopping-conditions for the qc. The bolded knowledge element represents the dpm stopping due to completion of the last process segment.

ment has been detected, control returns to the simulation administrator, and the dynamic knowledge is updated to include the final values for all of the variables in each of the dynamic models. Subsequently, the rules in the dynamic knowledge are evaluated, and the stopping-conditions are checked.

This process of sending the "iterate" message and waiting for a response, then evaluating rules and checking the stopping-conditions continues until the process is finally stopped (in which case the dpm enters the all-done state). The method to determine if the simulation can stop consists of attempting to match all of the stopping-conditions within a stop group of the current qc (e.g., one of the three stop groups shown in Fig. 14) with knowledge elements in the dynamic knowledge. In this example, the process simulation stops because the dynamic process model has finished simulation of the final process segment.

Following step IX-C of the individual process simulation algorithm, when the process simulation stops, the final values of the variables within the DRM are stored within the dynamic knowledge. Afterwards, the postconditions of the qualitative component, and the final values in the dynamic knowledge are pushed onto the knowledge stack.

After a process is simulated, the set of knowledge elements corresponding to the process on the knowledge stack contains a history of the simulation of the process. This history, illustrated in Fig. 15, stores the starting and ending values of all dynamic process/resource model conditions associated with the simulated process.

Simulation of Entire Manufacturing Process Plans

Figure 16 shows the process plan simulation algorithm. This algorithm traverses through the process plan, proceeding to the next appropriate node if currently at a nonprocess node, otherwise calling the individual process simulation algorithm to simulate the process at an ipd node. We discuss this algorithm through an example, using the process plan of Fig. 2.

In the beginning of the simulation of an entire process plan, both the knowledge stack and the dynamic knowledge

```
==================================
PROCESS_3
==================================
(CPTD_TOOL_1-Res-Cond STATUS DISCRETE-VAR FINISHED)
(TD_PRO_1-Pro-Cond STATUS DISCRETE-VAR FINISHED))
(TD_PRO_1-Pro-Cond FORCE VAR 114000)
(TD_PRO_1-Pro-Cond TOOL_POS VAR (40.0 105.0 75.0))                Final value of
(CPTD_TOOL_1-Res-Cond VELOCITY VAR 0.25)                ◄——————      dynamic
(CPTD_TOOL_1-Res-Cond ROT-VELOCITY VAR 550)                         knowledge
(CPTD_TOOL_1-Res-Cond CONDITION DISCRETE-VAR MEDIUM)
(CPTD_TOOL_1-Res-Cond HARDNESS VAR 475)
(CPTD_TOOL_1-Res-Cond DRILL_DIAMETER VAR 20)
(FEATURE_4-Env-Cond CLAMP_AXIS VAR (0 1 0))
(FEATURE_4-Env-Cond APPROACH VAR (0 0 1))
(FEATURE_4-Env-Cond DIMENSION_OFFSET VAR (45 105 60))
(FEATURE_4-Env-Cond DIMENSION_HEIGHT VAR (0 0 −30))
(FEATURE_4-Env-Cond DIMENSION_RADIUS VAR 15)                      Post-conditions
(FEATURE_4-Env-Cond DIM_TOL VAR 0.004)                  ◄——————     in bold
(FEATURE_4-Env-Cond REL_TOL VAR 0.007)
(FEATURE_4-Env-Cond IND_TOL VAR 0.006)
(FEATURE_4-Env-Cond FINISH VAR 210)
(PROCESS_6-Qual-Cond STATUS DISCRETE-VAR FINISHED)                  Qualitative
                                                        ◄——————   condition added
(CPTD_TOOL_1-Res-Cond STATUS DISCRETE-VAR ACTIVE)
(TD_PRO_1-Pro-Cond STATUS DISCRETE-VAR ACTIVE)
                    .
                    .
                    .
```

Figure 15. Knowledge stack after the end of the simulation of process #6. Note the added qualitative process condition indicating that the simulation for the current process has finished. Also note that the post-conditions of the qc have been added along with the final values within the dynamic knowledge.

```
Input:    Process Plan In The Form of An AND/OR Graph With Unique Node Identifiers ( PP ), Each Process Node Is Ar
          Intermediate Process Description ( ipd )
Output:   A Knowledge Stack Containing Detailed Information About the State of the System Whenever
          A Process Is Started, or Ends ( KS ).

I.        trace_stack ← empty, current_node ← Start, branch_stack ← empty
II.       Push(current_node, trace_stack)
III.      while current_node ≠ End
          A.        switch ( Type(current_node ))
                    1.        Start :
                              a)        current_node ← next_node
                              b)        Push(current_node, trace_stack)
                    2.        OR :
                              a)        current_node ← first node of unmarked branch, and mark
                              b)        Push(current_node, trace_stack), Push (OR, branch_stack)
                    3.        AND :
                              a)        current_node ← first node of unmarked branch, and mark
                              b)        Push(current_node, trace_stack), Push (AND, branch_stack)
                    4.        JOIN :
                              a)        if (top(branch_stack) = OR ) then
                                        (1)       current_ ← next_node
                                        (2)       Push(current_node, trace_stack), Pop (branch_stack)
                              b)        else {
                                        (1)       backtrack to last AND node
                                        (2)       if (AND node has no more unmarked branches) then
                                                  (a)       current_node ← next_node
                                                  (b)       Push(current_node, trace_stack), Pop(branch_stack)
                                        (3)       else
                                                  (a)       current_node ← first node of unmarked branch, and mark
                                                  (b)       Pop( trace_stack),  Push(current_node, trace_stack)
                              c)        } /* end else */
                    5.        Process :
                              a)        Process_Sim_Alg(current_node, Knowledge_Base, Model_Base)
                              b)        current_node ← next_node
                              c)        Push(current node, trace stack)
```

Figure 16. Process plan simulation algorithm.

are empty (step I of the process plan simulation algorithm). The first node encountered is the Start node, and the second node encountered is the first branching node; an AND node. The top branch is randomly chosen, and the second branching node is encountered (an OR node), followed by the first process node corresponding to process #1 (assuming we again arbitrarily choose the top branch). At this point, the individual process simulation algorithm is invoked to simulate the process (step III-A-5-a of the process plan simulation algorithm). Figure 17 shows the contents of the knowledge stack after the simulation of process #1 has been completed.

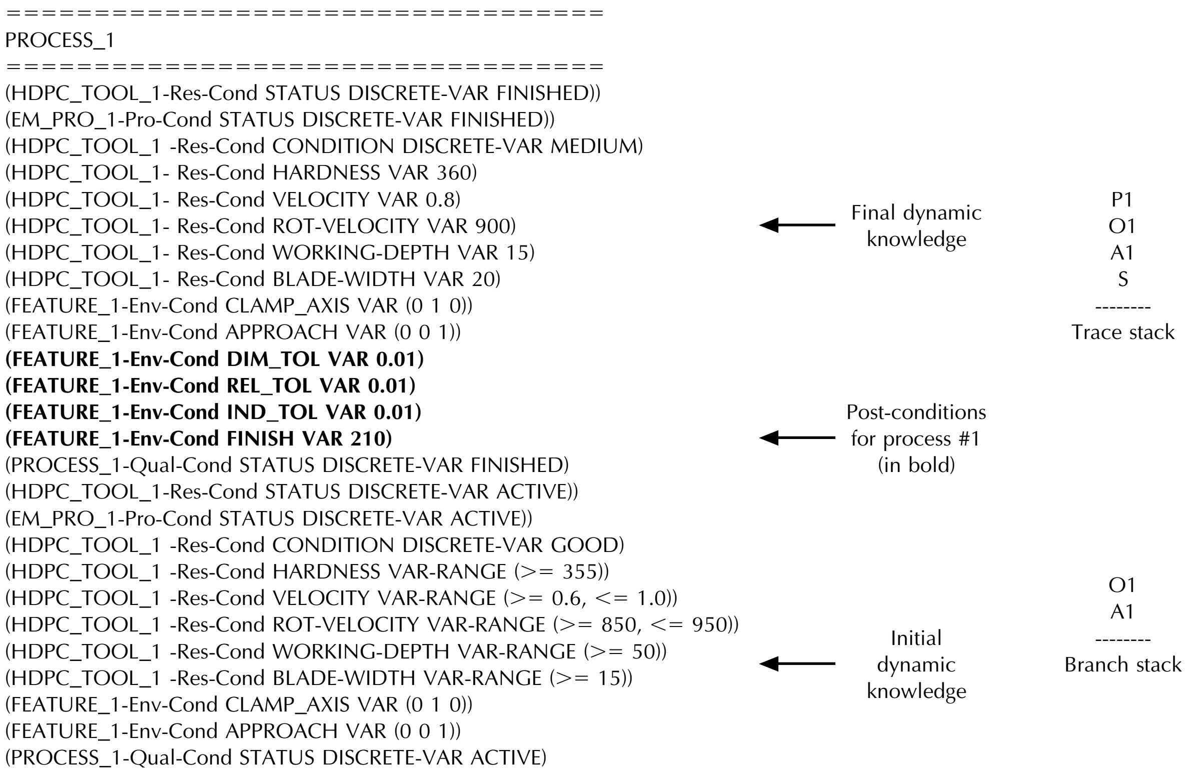
The trace_stack (also shown in Fig. 17), which is used to keep track of the simulation position in the process plan graph, now consists of a pointer to process #1 above a pointer to the OR node, AND node, and Start node. The branch_stack, which is used by the algorithm to ensure proper branch control, consists of an OR node above an AND node.

The next node encountered after process #1 is process #2. This process is simulated and the branch emanating from the OR node has been completely simulated. The next node traversed is the first JOIN node. Because the top of the branch_stack is an OR node (i.e., the first JOIN terminates an OR branch), the traversal continues through process #5 and process #6.

After process #6 has been simulated, the first branch of the AND node has been simulated, and we encounter another JOIN node. Because the top of the branch_stack is AND, we must ensure that there exist no untraversed branches emanating from the AND node before proceeding through the JOIN node. The simulation continues through the lower branch of the AND node, encountering a second OR node, and the top branch is arbitrarily chosen. The simulation continues and process #7 and process #8 are simulated before encountering the second and third JOIN nodes. Since all branches emanating from the AND node have been visited, simulation proceeds through the node after the third JOIN node; the End node. The simulation of the process plan is complete. The top of the final knowledge stack and trace_stack are shown in Fig. 18. The final trace_stack shows which processes in the plan were simulated and in which order.

AUTOMATIC SIMULATION MODEL GENERATION

To simulate a process plan as we have shown in the previous section, simulation models must be present that match the characteristics of the actual processes. The complexity of the processes involved necessitates the ability to quickly and accurately generate the needed simulation models. However, it is too costly to manually create simulation models when they are needed; this is especially true in the domain of discrete part manufacturing. Therefore, it is beneficial to automatically generate the qc for a process and automate linking together the associated dynamic models of the dc.



```
==================================
PROCESS_1
==================================
(HDPC_TOOL_1-Res-Cond STATUS DISCRETE-VAR FINISHED))
(EM_PRO_1-Pro-Cond STATUS DISCRETE-VAR FINISHED))
(HDPC_TOOL_1 -Res-Cond CONDITION DISCRETE-VAR MEDIUM)
(HDPC_TOOL_1- Res-Cond HARDNESS VAR 360)
(HDPC_TOOL_1- Res-Cond VELOCITY VAR 0.8)
(HDPC_TOOL_1- Res-Cond ROT-VELOCITY VAR 900)
(HDPC_TOOL_1- Res-Cond WORKING-DEPTH VAR 15)
(HDPC_TOOL_1- Res-Cond BLADE-WIDTH VAR 20)
(FEATURE_1-Env-Cond CLAMP_AXIS VAR (0 1 0))
(FEATURE_1-Env-Cond APPROACH VAR (0 0 1))
(FEATURE_1-Env-Cond DIM_TOL VAR 0.01)
(FEATURE_1-Env-Cond REL_TOL VAR 0.01)
(FEATURE_1-Env-Cond IND_TOL VAR 0.01)
(FEATURE_1-Env-Cond FINISH VAR 210)
(PROCESS_1-Qual-Cond STATUS DISCRETE-VAR FINISHED)
(HDPC_TOOL_1-Res-Cond STATUS DISCRETE-VAR ACTIVE))
(EM_PRO_1-Pro-Cond STATUS DISCRETE-VAR ACTIVE))
(HDPC_TOOL_1 -Res-Cond CONDITION DISCRETE-VAR GOOD)
(HDPC_TOOL_1 -Res-Cond HARDNESS VAR-RANGE (>= 355))
(HDPC_TOOL_1 -Res-Cond VELOCITY VAR-RANGE (>= 0.6, <= 1.0))
(HDPC_TOOL_1 -Res-Cond ROT-VELOCITY VAR-RANGE (>= 850, <= 950))
(HDPC_TOOL_1 -Res-Cond WORKING-DEPTH VAR-RANGE (>= 50))
(HDPC_TOOL_1 -Res-Cond BLADE-WIDTH VAR-RANGE (>= 15))
(FEATURE_1-Env-Cond CLAMP_AXIS VAR (0 1 0))
(FEATURE_1-Env-Cond APPROACH VAR (0 0 1))
(PROCESS_1-Qual-Cond STATUS DISCRETE-VAR ACTIVE)
```

Figure 17. Knowledge stack after process #1 has been completed. The bottom of the stack contains the initial values of the dynamic knowledge, the highlighted middle contains the post-conditions, and the top contains the final values for the knowledge elements in the dynamic knowledge.

The problems related to the automatic generation of the process simulation models can be decomposed as follows:

(i) Automatically generate the qc
- Generate active-conditions
- Generate post-conditions
- Generate stopping-conditions within the stop groups

(ii) Automatically link together the dynamic models by instantiating and coupling the C, V, and Pt sets for all dynamic models
- Instantiate variables, V, and couple the ports, Pt, for dpm and all drmi ∈ DRM
- Assign remaining input to the constants, C, for all dynamic models

Before we can show the automatic generation of the PSM, we first define the input which describes our starting information.

Starting Information

The two items which comprise the input to the simulation methodology are the process plan, and the task goals list. The task goals list (TGL) is defined:

The task goals list is defined as a set of *tasks,* TGL = $\{task_1, task_2, \ldots, task_p\}$, where each $task_i$ is defined as a double, $task_i = \langle id, plist\rangle$.

- *id* is the unique task identifier.
- *plist* is a list of knowledge elements which represent the task goals, $plist = [el_1, el_2, \ldots, el_q]$.

The task goals list contains goals that become true once a series of processes have been completed. An example of a task in discrete part manufacturing is a specification of a feature. The goals of the task are the environmental conditions describing the feature (e.g., dimensions and surface finish), and the task is said to be completed once the last process to produce the invoked feature has finished. We only need to record, in the ipd, the tasks in which a given process is the last process.

Completing the definition of a process plan, the intermediate process description (ipd) is a 5-tuple, ipd = $\langle id, LAST\_TASK, CONS, PROP, RES\rangle$.

- *LAST_TASK* is the set of tasks the process completes, $LAST\_TASK = \{task_1, task_2, \ldots, task_k\}$, where $task_i$ is the ID of a task.
- *CONS* is the set of constraints, $CONS = \{cons_1, cons_2, \ldots, cons_l\}$, where each $cons_i$ is a knowledge element.
- *PROP* is the set of properties, $PROP = \{prop_1, prop_2, \ldots, prop_m\}$, where each $prop_i$ is a knowledge element.
- *RES* is the set of results, $RES = \{res_1, res_2, \ldots, res_n\}$, where each res_i is a knowledge element.

Automatic Generation of the Qualitative Component

We begin our discussion of the automatic generation of the qc by identifying the subparts of the model that must be created. From the definition of the qc given earlier, there are three sets of conditions that must be generated: active-conditions, post-conditions, and stopping-conditions. Also, the TYPE variables of dynamic process/resource model allocations and con-

```
===================================
PROCESS_8
===================================
(SPINDLE_MOD_1-Res-Cond STATUS DISCRETE-VAR FINISHED)
                         .
                         .
                         .
(SPINDLE_MOD_1-Res-Cond STATUS DISCRETE-VAR ACTIVE)
(CTPC_TOOL_1-Res-Cond STATUS DISCRETE-VAR ACTIVE)
(FEM_PRO_1-Pro-Cond STATUS DISCRETE-VAR ACTIVE)                  E
(PROCESS_8-Qual-Cond STATUS DISCRETE-VAR ACTIVE)                 J3
===================================                              J2
PROCESS_7                                                        P8
===================================                              P7
(PSM_TOOL_2-Res-Cond STATUS DISCRETE-VAR FINISHED)               O2
(SFM_PRO_1-Pro-Cond STATUS DISCRETE-VAR FINISHED)                P6
(PROCESS_7-Qual-Cond STATUS DISCRETE-VAR FINISHED)               P5
(PSM_TOOL_2-Res-Cond CONDITION DISCRETE-VAR MEDIUM)              J1
(PSM_TOOL_2-Res-Cond VELOCITY VAR 2.3)                           P2
                         .                                       P1
                         .                                       O1
                         .                                       A1
===================================                              S
PROCESS_6                                                     --------
===================================                          Trace stack
(CPTD_TOOL_1-Res-Cond STATUS DISCRETE-VAR FINISHED))
(TD_PRO_1-Pro-Cond STATUS DISCRETE-VAR FINISHED))
(CPTD_TOOL_1 -Res-Cond CONDITION DISCRETE-VAR MEDIUM)
(CPTD_TOOL_1 -Res-Cond HARDNESS VAR 475)
(CPTD_TOOL_1 -Res-Cond VELOCITY VAR 0.25)
                         .
                         .
                         .
```

Figure 18. Top portion of the final knowledge stack and full trace_stack (the branch_stack is empty) after the simulation of the entire example process plan of Fig. 2.

ditions must be instantiated. The creation of the qc will be discussed through the generation of one shown in Fig. 4 for process #6. Figure 3 shows the input intermediate process description for process #6 (as discussed earlier) that will be used in the qc generation. Figure 19 shows the partial listing of task goals for the process plan (both Fig. 3 and Fig. 19 are derived from the part shown in Fig. 1).

Specifically, Fig. 19 shows the task goals for a round_pocket (Feature_4) created by process #6. The algorithm in Fig. 20 shows how the active-conditions of the qc are generated. Note that this algorithm is also responsible for instantiating TYPE variables.

```
Task 1:  FEATURE_1
           (FEATURE_1-Env-Cond DIM_TOL VAR-RANGE (<= 0.01))
                                   .
                                   .
                                   .
TASK 3:  FEATURE_3
           (FEATURE_3-Env-Cond DIM_TOL VAR-RANGE (<= 0.001))
                                   .
                                   .
                                   .
TASK 4:  FEATURE_4
           (FEATURE_4-Env-Cond DIM_TOL VAR-RANGE (<= 0.008))
           (FEATURE_4-Env-Cond REL_TOL VAR-RANGE (<= 0.008))
           (FEATURE_4-Env-Cond IND_TOL VAR-RANGE (<= 0.008))
           (FEATURE_4-Env-Cond FINISH VAR-RANGE (<= 213))
           (FEATURE_4-Env-Cond DIMENSION_OFFSET VAR (45 105 60))
           (FEATURE_4-Env-Cond DIMENSION_HEIGHT VAR (0 0 −30))
           (FEATURE_4-Env-Cond DIMENSION_RADIUS VAR 15)
```

Figure 19. Partial listing of task goals for the input process plan.

The first step in the generation of the active-conditions involves identifying all process/resource conditions and allocations that contain TYPE variables. This is performed by searching all three components of the ipd and copying those that meet the said criteria.

The next step is to match the names with those found in the TVA table (step II-A of the algorithm). The TYPE variables within the knowledge elements are matched against those in the TVA table in order to find possible values. Once all of the TYPE variables have been instantiated for a given group, then each of the dynamic process/resource model allocations within that group are converted into rules and placed within the active-conditions (step II-B of the algorithm). All of the dynamic process/resource model conditions are placed back into the intermediate process description in an instantiated state and added to the active-conditions. Figure 21 illustrates the rules that are generated by the methodology in place of the dynamic process/resource model allocations. The final step in the creation of the active-conditions is the addition of all properties to the Active-Conditions; which are now fully instantiated. The final set of Active-Conditions is shown within the qc of Fig. 4.

After all of the active-conditions have been generated and the ipd has been instantiated, the next step is the generation of the stopping-conditions, which are used to determine if the process has been successfully simulated. This algorithm gen-

```
Input:  Uninstantiated Intermediate Process Description ( ipd ), Knowledge Stack ( KS ).
        Type/Value/Attribute Table (TVA)
Output: Instantiated Intermediate Process Description, Active-Conditions ( ACT )

I.    Let A $\leftarrow$ all dynamic process/resource allocations and conditions within ipd that have TYPE
      variable names; grouped according to identical names.
II.   foreach r $\in$ A do
      A.   match all elements of r with the first available name from Type/Value/Attribute
           table that conform to the process/resource allocations and conditions within r using
           info. contained in the Knowledge Stack
      B.   foreach u $\in$ r do
           1.   if u is a process/resource allocation then
                a)   create rule and add to Active-Conditions
           2.   else
                a)   add to Active-Conditions
III.  Add all Properties (except process/resource allocations) to Active-Conditions
```

Figure 20. The active-conditions generation algorithm. This algorithm generates all of the active-conditions for the qc, and instantiates all TYPE variables.

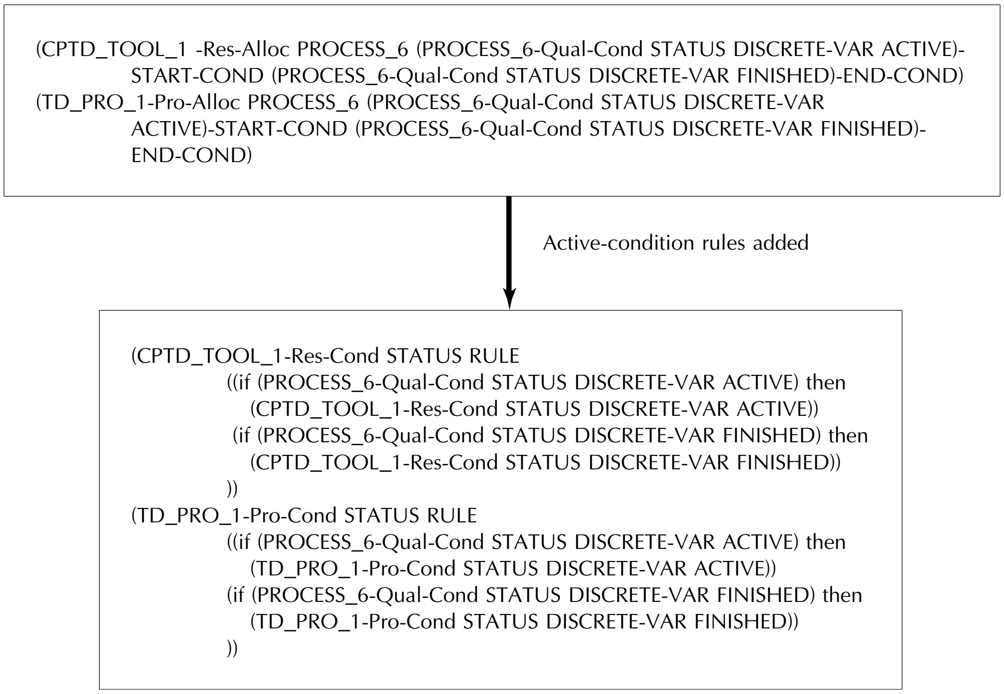
erates three subsets of stopping-conditions; one for the completion of the results, one for the completion of all tasks, and one for the completion of the dpm. If any of these subsets is completely satisfied, then the process stops. For this example, we simply show the three subsets of stopping-conditions that are generated in Fig. 22.

The last set of conditions that must be generated to complete the qc is the set of post-conditions that represent the expected outcome of the process. The generation of the post-conditions is accomplished through combining the results of the ipd and task goals for the process. The algorithms for the post-conditions generation and for stopping-conditions generation are straightforward and the reader is directed to Ref. 15 for further details.

Automatic Generation of Constants, Variables, and Port Definitions for the Dynamic Component

The second operation that must be performed in the creation of the PSM is to link together the dynamic models through the automatic generation of constants, variables, and port definitions (C, V, Pt) for all dynamic process/resource models associated with the process. The algorithm for generating them is shown in Fig. 23.

The input required to create C, V, and Pt for the dynamic models is fully contained within the model base of continuous descriptions (MB) once it is determined which dynamic models are required using the rules contained in the dynamic knowledge. The MB contains the information listing the input to and output from all of the continuous descriptions (in the



```
(CPTD_TOOL_1 -Res-Alloc PROCESS_6 (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR ACTIVE)-
         START-COND (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR FINISHED)-END-COND)
(TD_PRO_1-Pro-Alloc PROCESS_6 (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR
         ACTIVE)-START-COND (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR FINISHED)-
         END-COND)
```

```
(CPTD_TOOL_1-Res-Cond STATUS RULE
        ((if (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR ACTIVE) then
           (CPTD_TOOL_1-Res-Cond STATUS DISCRETE-VAR ACTIVE))
         (if (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR FINISHED) then
           (CPTD_TOOL_1-Res-Cond STATUS DISCRETE-VAR FINISHED))
        ))
(TD_PRO_1-Pro-Cond STATUS RULE
        ((if (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR ACTIVE) then
           (TD_PRO_1-Pro-Cond STATUS DISCRETE-VAR ACTIVE))
        (if (PROCESS_6-Qual-Cond STATUS DISCRETE-VAR FINISHED) then
           (TD_PRO_1-Pro-Cond STATUS DISCRETE-VAR FINISHED))
        ))
```

Figure 21. Rules generated by dynamic process/resource model allocations that are added to the active-conditions. The conversion to rules aids in the determination if a dynamic model is needed; if its rules evaluate to true, then the dynamic model is allocated.

$$A = \left\{ \begin{array}{l} \text{(FEATURE\_4-Env-Cond DIM\_TOL VAR 0.004)} \\ \text{(FEATURE\_4-Env-Cond REL\_TOL VAR 0.007)} \\ \text{(FEATURE\_4-Env-Cond IND\_TOL VAR 0.006)} \\ \text{(FEATURE\_4-Env-Cond FINISH VAR 210)} \end{array} \right.$$

$$B = \left\{ \begin{array}{l} \text{(FEATURE\_4-Env-Cond DIM\_TOL VAR-RANGE (<= 0.008))} \\ \text{(FEATURE\_4-Env-Cond REL\_TOL VAR-RANGE (<=0.008))} \\ \text{(FEATURE\_4-Env-Cond IND\_TOL VAR-RANGE (<= 0.008))} \\ \text{(FEATURE\_4-Env-Cond FINISH VAR-RANGE (<= 213))} \\ \text{(FEATURE\_4-Env-Cond DIMENSION\_OFFSET VAR (45 105 60))} \\ \text{(FEATURE\_4-Env-Cond DIMENSION\_HEIGHT VAR (0 0 } -30\text{))} \\ \text{(FEATURE\_4-Env-Cond DIMENSION\_RADIUS VAR 15)} \end{array} \right.$$

$$C = \left\{ \begin{array}{l} \text{(TD\_PRO\_1-Pro-Cond STATUS DISCRETE-VAR FINISHED)} \end{array} \right.$$

Figure 22. Stopping-condition subsets (stop groups) that are generated for process #6.

form of the *IN_PARAM* and *OUT_PARAM* sets contained in each continuous description) connected to each dynamic model. For this example, the required input to the continuous descriptions given by (9) and (11) were utilized (see Fig. 24 for the continuous description of chip thickness used by the dynamic resource model of the conical point twist drill). The continuous description of Fig. 24 is divided into three sections; the IN_PARAM set, the OUT_PARAM set, and the actual functions used in the calculations. Both the IN_PARAM and OUT_PARAM sets consist of symbol/description pairs, an example is the symbol that represents the chisel edge angle of the drill tip. The functional component of continuous descriptions utilize the symbols defined in the IN_PARAM set to calculate the value of output parameters in the OUT_PARAM set. With this information given, we can create the three needed sets to satisfy all of the input to all attached continuous descriptions.

The first step of the algorithm is to create ports from the dpm to the drms for time synchronization. Ports are used when multiple dynamic models need the same input value(s) for one or more of their attached continuous descriptions. Note that the representation of a port is first the variable it is associated with, followed by what it is connected to, and finally if it is IN, OUT, or both.

After the time ports have been added, the algorithm can proceed with adding variables for all of the output from the continuous descriptions attached to the dpm. Also, ports from these variables are added to the required input of continuous descriptions attached to all drms (in step III). Figure 25 shows the state of the constants, variables, and port definitions after steps I, II, and III have been performed. Here an example is the value of the tool_pos variable being transferred via a port to the CPTD_TOOL_1 dynamic resource model. The necessity of the port for tool_pos from the TD_PRO_1 dynamic process model to the CPTD_TOOL_1 dynamic resource model can be seen in the continuous description of Fig. 24.

Just as variables (and ports where possible) are created for the dpm, we must cycle through each drm and perform the same operations (step IV of the algorithm). Now that all of the output from all continuous descriptions attached to the dynamic models have been accounted for, we can assume that all other input values to these continuous descriptions must be constants. There is no other means to change the value of this needed input, because all change comes about as a result (indirectly in the case of discrete conversion) of the use of continuous descriptions. The final values of the sets of constants, variables, and port definitions are shown in Fig. 26.

CONCLUSION

A simulation methodology capable of capturing the dynamic and qualitative aspects of processes, and automatically generating simulation models was given. Detailed examples in the discrete part manufacturing domain illustrated the methodology's ability to simulate both the dynamic attributes of an individual process and the qualitative aspects of an entire process plan. The contributions of this work include:

- *Method for Combined Discrete / Continuous Simulation of Process Plans.* The approach developed here encompasses both the dynamic and qualitative aspects (such as post-conditions that hold after a process has executed) of processes. It dynamically generates process simulation models as the simulation progresses and updates the knowledge base as the simulation results become available. The process simulation models, in this work, capture both the dynamic and qualitative aspects of processes and resources such as tools.
- *A Method for Automatic Generation of Computational Simulation Models for Process Plans.* Based on capturing processes, resources such as tools, and simulation models in a modular fashion, the mechanism achieves this goal by:
  - Automatically creating active, stopping, and post conditions for the qualitative components of the process simulation models

```
Input:  Uninitialized augmented DEV & DESS models dpm and DRM, Listing of input and output of all
        continuous descriptions connected to augmented DEV & DESS dynamic models
Output: Initialized dpm and any drms with port bindings, variables, and constants. ( C, V, Pt )

I.      Create ports from dpm to DRM for time
II.     Create variables for all output (and discrete conversions of output) of dpm continuous
        descriptions.
III.    Create ports from these variables to applicable input in DRM.
IV.     For_each dpm do
            A.    Create variables for all output of continuous descriptions connected to the drm and
                  variations on the output.
            B.    Create ports from these variables to applicable input in other drm or drm continuous
                  description.
V.      Remaining input is represented by constants
```

Figure 23. Constants, variables, and port definitions generation algorithm for all attached dynamic models. This algorithm is responsible for creating these three sets for each dynamic model. Although some of these elements exist by default in the uninstantiated dynamic models, most must be automatically added.

IN_PARAM

t_0 = constant_1

t_1 = constant_2

t_q = constant_3

$2R_d$ = drill_diameter

$2t$ = web_thickness

2ρ = point_angle

ψ = chisel_edge_angle

β = helix_angle

r = radial_distance_drill_axis

r_0 = tool_pos

Ω = rot_velocity

$$R_c(r) = \frac{t_2(r)}{2} + \frac{\Omega r \cos(i(r))}{2Ct_2(r)},\ \sin[i(r)] = \frac{-t\sin(\rho)}{r},$$

$$\tan[\alpha(r)] = \frac{(r^2 - t^2\sin^2(\rho))\tan(\beta)}{R_d\sqrt{r^2 - t^2}} - \frac{t\cos(\rho)}{\sqrt{r^2 - t^2}},\ t_2(r) = t_0 + t_1(r - r_0) + t_q(r - r_0)^2,\ r_0 = -t\big/\cos(\psi)$$

. . .

OUT_PARAM

t_2 = chip_thickness

Figure 24. Continuous description for the chip-thickness generated in conical point twist drilling. Here the IN_PARAM set is the needed input to the model, and the OUT_PARAM set consists of one element, the chip thickness. Note that only a portion of the equations needed for the calculation of the chip thickness is given (9).

```
DEV & DESS Dynamic Process Model - TD_PRO_1

Constants: ;
Variables:
             Discrete:  Current-State;
             Cont.:  Time, Delta-T, seg_end, force, torque, tool_pos;

Ports:
             Discrete:  (Port-x Simulation-Admin. IN) (Port-y Simulation-Admin. OUT)
             Cont.:     ((Port-seg_end Seg-End-Model IN/OUT) (Port-force TD-Force-Model IN/OUT)
                        (Port-torque TD-Force-Model IN/OUT) (Port-tool_pos TD-Force-Model IN/OUT)
                        (Port-Time (CPTD_TOOL_1 OUT)(Port-tool_pos (CPTD_TOOL_1 OUT)

. . .
```

```
DEV & DESS Dynamic Process Model - CPTD_TOOL_1

Constants: Discrete-Conversion-Table;
Variables:
             Discrete:  Current-State;
             Cont.:  ;

Ports:
             Discrete:  (Port-x Simulation-Admin. IN) (Port-y Simulation-Admin. OUT)
             Cont.:     ((Port-chip_thick Cont-Chip-CPTD IN/OUT) (Port-Time TD_PRO_1 IN)
                        (Port-tool_pos TD-PRO_1 IN)

. . .
```

Figure 25. Constants, variables, and port definitions after the variables for the output of the continuous descriptions connected to the dpm, and the ports connecting these variables to other dynamic models have been created. Note that when a port is added to transfer the value of a variable, both an OUT port at the source, and an IN port at the destination must be created.

```
DEV & DESS Dynamic Process Model - TD_PRO_1

Constants: drill_diameter, depth_cut, feature_dimensions, chisel_edge_angle, feedrate, inclination_angle,
           yield_shear_stress_const, normal_specific_cutting_pres_const, tang_specific_cutting_pres_const,
           point_angle, drill_diameter, indentation_radius, radial_distance_cutting_lips, pilot_hole_radius,
           rot_velocity, web_thickness, helix_angle, rake_angle, chisel_edge_normal_rake_angle,
           web_length_diam_ratio;
Variables:
           Discrete:  Current-State;
           Cont.:  Time, Delta-T, seg_end, force, torque, tool_pos;
Ports:
           Discrete:  (Port-x Simulation-Admin. IN) (Port-y Simulation-Admin. OUT)
           Cont.:     (Port-seg_end Seg-End-Model IN/OUT) (Port-force TD-Force-Model IN/OUT)
                      (Port-torque TD-Force-Model IN/OUT) (Port-tool_pos TD-Force-Model IN/OUT)
                      (Port-Time CPTD_TOOL_1 OUT)(Port-tool_pos (CPTD_TOOL_1 OUT)
                      (Port-chip_thick CPTD_TOOL_1 IN)
. . .
```

```
DEV & DESS Dynamic Process Model - CPTD_TOOL_1

Constants: Discrete-Conversion-Table, drill_diameter, depth_cut, constant_1, constant_2, constant_3,
           web_thickness, point_angle, chisel_edge_angle, helix_angle, radial_distance_drill_axis,
           rot_velocity;
Variables:
           Discrete:  Current-State, tool_condition;
           Cont.:  chip_thick;
Ports:
           Discrete:  (Port-x Simulation-Admin. IN) (Port-y Simulation-Admin. OUT)
           Cont.:     (Port-chip_thick Cont-Chip-CPTD IN/OUT) (Port-Time TD_PRO_1 IN)
                      (Port-tool_pos TD-PRO_1 IN) (Port-chip_thick TD_PRO_1 OUT)
. . .
```

Figure 26. Final sets of constants, variables, and port definitions for all dynamic models associated with process #6. This figure shows the constants that are added to each dynamic model because no method of changing the value existed.

- Automatically instantiating and coupling the constants, variables, and port definitions for the dynamic components of the process simulation models.

APPENDIX 1. DYNAMIC MODELS

Dynamic process model for twist drilling operation (process #6). Note that the constants, variables, and port definitions are those derived in the text. The attached boxes represent the continuous computer models used for dynamic simulation of the process (continuous descriptions). Algorithms for all associated functions are provided as well. Note that the algorithms for all dpms will be exactly the same except for the polling calls in the delta-T polling function.

Dynamic resource model for conical point twist drill. Note that the constants, variables, and port definitions are those derived in the text. The attached boxes represent the continuous computer models used for dynamic simulation of the process (continuous descriptions). Algorithms for all associated functions are provided as well; note that the algorithms for all drms will be exactly the same except for the polling calls in the delta-T polling function. The "table of discrete conversions" refers to the table responsible for converting the continuous output "chip_thick" into the discrete "tool_condition" value.

BIBLIOGRAPHY

1. M. Marefat and J. Britanik, Plan reuse and plan merging in manufacturing process planning, *Eng. Des. Autom.,* **3** (1): 1996.
2. M. Marefat and R. L. Kashyap, Geometric reasoning for recognition of three-dimensional object features, *IEEE Trans. Pattern Anal. Mach. Intell.,* **12**: 949–965, 1990.
3. U. A. Sungurtekin and H. B. Voelcker, Graphical Simulation and Automatic Verification of NC Machining Programs, *Proc. IEEE Int. Conf. Robot. Autom.,* April 1986, pp. 156–165.
4. T. Saito and T. Takahashi, NC machining with G-buffer method, *Comput. Graph.,* **25** (4): 207–216, 1991.
5. P. L. Hsu and W. T. Yang, Realtime 3D simulation of 3-axis milling using isometric projection, *Comput. Aided Des.,* **25** (4): 215–224, 1993.
6. S.-H. Suh and K.-S. Lee, A prototype CAM system for four-axis NC machining of rotational-free-surfaces, *J. Manuf. Syst.,* **10** (4): 322–331, 1991.

7. Y. S. Tarng and W. S. Chang, Dynamic NC simulation of milling operatoins, *Comput. Aided Des.,* **25** (12): 769–775, 1993.
8. F. M. Kolarits and W. R. DeVries, A mechanistic dynamic model of end milling for process controller simulation, *ASME J. Eng. Ind.,* **113** (2): 176–183, 1991.
9. V. Chandrasekharan, S. G. Kapoor, and R. E. DeVor, A mechanistic approach to predicting the cutting forces in drilling: With applications to fiber-reinforced composite materials, *ASME J. Eng. Ind.,* **117** (4): 559–570, 1995.
10. P. L. B. Oxley, Modeling machining processes with a view to their optimization and to the adaptive control of metal cutting machine tools, *Robot. Comput.-Integr. Manuf.,* **4** (1/2): 103–119, 1988.
11. D. A. Stephenson and S. M. Wu, Computer models for the mechanics of three-dimensional cutting processes—Parts I & II, *ASME J. Eng. Ind.,* **110** (1): 32–43, 1988.
12. S. Polisetty, *Simulation and Performance Evaluation of Metal Cutting Processes from NC Programs,* Masters Thesis, Department of Systems and Industrial Engineering, University of Arizona, Tucson, 1992.
13. K. D. Forbus, Qualitative process theory, *Artif. Intell.,* **24** (1–3): 85–168, 1984.
14. H. Praehofer, System theoretic formalisms for combined discrete-continuous system simulation, *Int. J. Gen. Syst.,* **19** (3): 219–240, 1991.
15. J. T. Olson, *Combined Discrete / Continuous Simulation of Process Plans and Applications in Discrete Part Manufacturing,* Masters Thesis, Department of Electrical and Computer Engineering, University of Arizona, Tucson, 1997.

Reading List

M. Brielmann et al., Simulation of Hybrid Mechatronic Systems: A Case Study, *Proc. 1997 IEEE Eng. Comput. Based Syst. Conf.,* 1997, pp. 256–262.

B. A. Caton and S. R. Ray, ALPS: A language for process specification, *Int. J. Comput. Integr. Manuf.,* **4** (2): 105–113, 1991.

P. A. Fishwick and B. P. Zeigler, A multimodel methodology for qualitative model engineering, *ACM Trans. Modeling Comput. Simul.,* **2** (1): 52–81, 1992.

K. D. Forbus, Qualitative process theory: Twelve years after, *Artif. Intell.,* **59** (1–2): 115–123, 1993.

A. M. Foss, Towards a general procedure for dynamic model development, *Trans. Inst. Meas. Control,* **12** (4): 174–177, 1990.

G. G. Hendrix, Modeling simultaneous actions and continuous processes, *Artif. Intell.,* **4** (3–4): 145–180, 1973.

W. Jacak and J. W. Rozenblit, Automatic simulation of a robot program for a sequential manufacturing process, *Robotica,* **10** (1): 45–56, 1992.

S. Kalpakjian, *Manufacturing Engineering and Technology,* Reading, MA: Addison-Wesley, 1992.

R. P. Otero, D. Lorenzo, and P. Cabalar, Automatic induction of DEVS structures, Lect. Notes *Comput. Sci.,* **1030**: 305–315, 1996.

S. R. Ray, Using the ALPS process plan model, *Proc. Manuf. Int.,* 1992.

J. A. Stori and P. K. Wright, A Knowledge-Based System for Machining Operation Planning in Feature Based, Open Architecture Manufacturing, *Proc. 1996 ASME Design Eng. Tech. Conf. Comput. Eng. Conf.,* 1996, pp. 1–11.

B. P. Zeigler, *Object-Oriented Simulation with Hierarchical Modular Models: Intelligent Agents and Endomorphic Systems,* Boston: Academic Press, 1990.

JOHN T. OLSON
JOHN BRITANIK
MICHAEL M. MAREFAT
University of Arizona

COMPUTER ALGORITHMS. See DATA STRUCTURES AND ALGORITHMS.

COMPUTER AND COMMUNICATIONS TECHNOLOGY. See INFORMATION TECHNOLOGY.